Bayesian Optimisation & Priors

Improving Air Pollution Monitoring and Hyperparameter Optimisation

Sigrid Passano Hellan

Doctor of Philosophy

School of Informatics
The University of Edinburgh

2023

Abstract

Bayesian optimisation is a powerful machine learning method for optimising black-box functions. It relies on a surrogate model, typically a Gaussian process, for modelling the optimisation objective. Optimisation proceeds by iteratively evaluating, also called sampling, the function at selected inputs. In the regime of small sample budgets, due for instance to slow or expensive evaluations, only a small number of samples are available to fit the surrogate model. This is particularly challenging because we need to learn both the surrogate hyperparameters and the surrogate model itself. To maximise the usefulness of our limited samples we therefore want to introduce additional information into our model to reduce its uncertainty. This additional information comes from evaluations of related optimisation tasks, an approach that is often called transfer learning.

We consider two types of transfer learning. The traditional approach is to assume the optimal input on a new task to be correlated with the optimal inputs on at least some of the previous tasks, so good inputs on previous tasks will perform well on new tasks. We examine the setting where these tasks come with an inherent ordering. For instance, the tasks could come from repeatedly performing hyperparameter optimisation on a machine learning model as more training data are collected. This could be a movie recommendation system which is retrained after new movie ratings are submitted.

In the second scenario we assume that we cannot use the optimal inputs on previous tasks directly. One example is locating pollution maxima, where each task is a different city; we do not expect the maximisers to be the same, e.g. 1 km north and 3 km east of the geographical centre, but we do expect the pollution gradients to be similar as we move away from roads. Instead of learning useful inputs, we therefore learn a useful prior for the hyperparameters of the surrogate model. The hyperparameters are important because they encode how we expect the function to vary over its domain. For instance, if we assume a slowly-varying function we will disregard larger areas around low-performing samples.

Our methods extend existing approaches to transfer learning for Bayesian optimisation. For the ordered setting we show that a simple method exploiting this ordering can outperform more sophisticated state-of-the-art transfer methods. Our method, SimpleOrdered, works by first evaluating the best inputs from the most recent tasks, before performing non-transfer Bayesian optimisation.

For the second scenario, when we cannot use the optimal inputs directly, we develop a method for transferring hyperparameter priors, which we call PLeBO. It consists of two stages. In the preprocessing stage we use Markov chain Monte Carlo and hierarchical modelling to collect samples of the hyperparameter distributions. In the second stage, during the optimisation of a new task, we use importance weighting to fit the hyperparameter samples to the new evaluations. We show on synthetic and air pollution data that PLeBO can be applied to problems where the traditional transfer approach does not work.

Our two application domains are hyperparameter optimisation and air pollution monitoring. In hyperparameter optimisation we optimise the hyperparameters of machine learning models, e.g. learning rates and model sizes. The optimisation is expensive, as each evaluation requires training a potentially large model. For air pollution monitoring, the problem is that of iteratively placing sensors in order to locate the pollution maximum in a given area. This is a new problem for Bayesian optimisation, and we prepare data sets for further work. Both applications are difficult, with samples being expensive to collect, so Bayesian optimisation can be strengthened by including transfer learning.

Lay Summary

Machine learning is a collective term for many ways of automatically making decisions from data. This thesis considers one such method, called Bayesian optimisation.

Many problems can be approached as optimisation problems, provided there is some input we can adjust, and we want to find which input results in the highest or lowest output value. For instance, the input could be the selling price of a product, and the output could be the company's profit. Or the input could be a location in a city, and the output the pollution level at this location. Bayesian optimisation works as a type of informed trial and error, where we use the results of the previous trials to guide what to try next. We do so by building up a model of the problem we are optimising, for instance the pollution in each location in a city.

One of the advantages of Bayesian optimisation is that it is efficient in terms of how many trials we need before finding the highest or lowest output. But, as so often happens, we want to make our methods even more efficient. To do so, we use data we have collected from other, related optimisation problems. For instance, this could be pollution measurements from similar cities. The idea of using related optimisation problems is not new, but we present new ways of doing so.

The first problem we consider in this thesis is using Bayesian optimisation for *air pollution monitoring*. Here we want to find the most polluted location in a city, and we consider different locations in a city to be our problem inputs, and pollution levels to be the output. To optimise, we place sensors one by one to measure the pollution. Our goal is to place a sensor in the most polluted location.

Our second problem is *hyperparameter optimisation*. Many machine learning models come with hyperparameters — settings — which are critical to the model's performance. For instance, these can specify how noisy we think the data are, or specify whether we prefer a fast or an accurate model. We use Bayesian optimisation to find the best values of these hyperparameters.

In the thesis, we show that while standard, 'from-scratch' Bayesian optimisation struggles to solve these problems, doing similarly or worse than random guessing, our improved Bayesian optimisation version does a lot better — meaning we need to try fewer inputs to our problem before we find the highest or lowest output. This is particularly useful when trying inputs takes a lot of time, as is the case when we need to measure pollution or train a machine learning model.

In addition to these improvements, we also carried out a survey which summarises climate change related applications of Bayesian optimisation. We identified four main problems to which other people have applied Bayesian optimisation: discovering new or improved materials, planning the layout of wind farms, controlling renewable generators, and environmental monitoring.

For a further introduction to the concepts explored in this thesis, see the motivational example in Section 1.3 which does not assume knowledge of machine learning.

Acknowledgements

I am very grateful for the trust and mentorship provided by my supervisors Nigel Goddard and Chris Lucas. I have learned a great deal from both of them, and consider myself lucky to have been able to work with them. I have also enjoyed taking part in both research groups, and particularly want to thank Simon Valentin, Chentian Jiang, Lynda Webb, Jonathan Kilgour and Martin Pullinger.

The School of Informatics and the wider University of Edinburgh is an impressive community, and I also benefited from encouragement, help and feedback from Iain Murray, Michael Gutmann, Amos Storkey, Elliot Crowley, David Sterratt and Douglas Finch. In the background, Sally Galloway, Sio Carroll and the rest of the administrative teams did a massive job in making our studies more enjoyable and the paperwork less confusing.

I am grateful to my examiners, Arno Onken and Joaquin Vanschoren. Thanks to them, my viva was an insightful and friendly discussion of the thesis work, as well as its limitations, and I feel fortunate to have had that experience. Arno and Joaquin also provided valuable feedback which has improved the thesis.

A number of people made my time in Berlin special. I would like to thank my colleagues Huibin Shen, Aaron Klein, François-Xavier Aubet, David Salinas, Jan Gasthaus, Michael Bohlke-Schneider, Bhaskar Dhariyal, Marcel Kollovieh, Nils Malte Kiele, Fabian Falck, Alex Langevin, Valerio Perrone, Martin Wistuba, Lorenzo Stella and Abdul Fatir Ansari. And I would like to thank the people who welcomed me to Berlin, especially Walter, Inna and Sascha Meng, Sverre Aune and Vera Houck — *vielen Dank!*

My friends have shaped my experience, both academically and personally. There are too many to name all of them, but special thanks, in no particular order, go to Andrew Wood, Linus Ericsson, Ondrej Bohdal, William Toner, Julie-Anne Meaney, Eleanor Platt, Kate McCurdy, Tiffany Vlaar, Kaan Öcal, Markus Schneider, Nicole Meng-Schneider, Aidan Marnane, Tom Sherborne, Lorna Donnelly, Antoine Lain, Elaine Farrow, James Owers-Bardsley, Francisco Grifo, Ayodeji Akintunde and David Wilmot. Edinburgh University Orienteering Club is a welcoming and wholesome group, and the training sessions and trips away let me explore Edinburgh and Scottish forests while

taking a break from research. James Garforth, though too humble to realise it, is and has been an inspiration, as well as a welcome source of distraction, perspective and pizza. Katarzyna Pruś made our flatshare a home, and kept my plants thriving. I am grateful to all of you.

My family, Ottar, Elizabeth and Jon Kåre, has provided constant support and encouragement. I cannot imagine having done this without you. Jon Kåre read the whole thesis multiple times, and provided tireless proofreading throughout my PhD. He has read more versions of this research than anyone else ever will. I also want to thank Beth Howe for relating her own PhD experiences, Mac Passano for gentle support and blueberry muffins, Sissel Rogne for her hospitality and interest, and Laura Arlov for encouragement.

This work was supported by the EPSRC Centre for Doctoral Training in Data Science, funded by the UK Engineering and Physical Sciences Research Council (grant EP/L016427/1) and the University of Edinburgh. During my PhD I was fortunate enough to intern with AWS in Berlin, where I learned a lot from talented colleagues. I am also grateful for a grant from G-Research which helped cover conference expenses at the end of the PhD.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

Sigrid Passano Hellan

Contents

Abstract iii

Lay Summary v

Acknowledgements vii

Declaration ix

Figures and Tables xvii

Abbreviations xxi

1 Introduction 1

1.1 Bayesian optimisation . . . 1
1.2 Transfer learning . . . 3
1.3 Motivational example . . . 3
1.3.1 Ordered transfer . . . 8
1.3.2 Prior transfer . . . 10
1.4 Terminology . . . 13
1.5 Applications . . . 14
1.6 Contributions . . . 15
1.6.1 Bayesian optimisation against climate change . . . 16
1.6.2 Prior learning . . . 17
1.6.3 Ordered transfer learning . . . 18

2 Background 19

2.1 Bayesian optimisation . . . 20
2.2 Gaussian processes . . . 21
2.3 Hyperparameters . . . 23
2.3.1 Priors . . . 24
2.3.2 Approximate inference . . . 26
2.4 Acquisition functions . . . 27
2.4.1 Optimistic and improvement-based policies . . . 28

2.4.2 Information-theoretic policies . . . 29
2.4.3 Self-adjusting acquisition functions . . . 29
2.5 Model misspecification . . . 30
2.6 Transfer learning . . . 32
2.6.1 Direct transfer . . . 34
2.6.2 Prior transfer . . . 37
2.6.3 Applications of transfer Bayesian optimisation . . . 39
2.7 Alternatives to Bayesian optimisation . . . 39
2.7.1 Active learning . . . 39
2.7.2 Particle swarm optimisation . . . 40
2.7.3 Reinforcement learning . . . 41
2.8 Libraries . . . 42

3 Bayesian Optimisation Against Climate Change 43
3.1 Introduction . . . 43
3.2 A note on chapter ordering . . . 46
3.3 Author contributions . . . 46
3.4 The paper . . . 46
3.5 Discussion . . . 60
3.6 Future work . . . 61
3.7 Limitations . . . 62

4 Bayesian Optimisation for Air Pollution Monitoring 65
4.1 Introduction . . . 65
4.2 Background . . . 67
4.2.1 Air pollution and monitoring . . . 67
4.2.2 Air pollution and Gaussian processes . . . 72
4.2.3 Automated planning of monitoring networks . . . 72
4.2.4 Air pollution and wind . . . 73
4.2.5 Suitability of Bayesian optimisation . . . 74
4.3 Data . . . 74
4.3.1 Satellite data . . . 76
4.3.2 Ground-level London data . . . 79
4.4 Gaussian process models . . . 82
4.5 Experimental set-up . . . 85
4.5.1 Problem statement . . . 85
4.5.2 Metrics . . . 86

4.5.3 Baselines . . . 87
4.5.4 Experiments outline . . . 88
4.6 Initial experiments . . . 88
4.6.1 Methodology . . . 88
4.6.2 Results . . . 89
4.6.3 Discussion . . . 89
4.7 Simple prior experiments . . . 91
4.7.1 Methodology . . . 91
4.7.2 Results . . . 92
4.7.3 Discussion . . . 92
4.8 Sampled prior experiments . . . 93
4.8.1 Methodology . . . 94
4.8.2 Results . . . 98
4.8.3 Discussion . . . 100
4.9 Limitations and future work . . . 105
4.9.1 Simplifying assumptions in our experiments . . . 105
4.9.2 Future work . . . 106
4.10 Conclusion . . . 107

5 Data-driven Prior Learning for Bayesian Optimisation 109
5.1 Introduction . . . 109
5.2 Method description and improvements . . . 111
5.2.1 Preprocessing — learning the prior samples with MCMC . . . 113
5.2.2 Optimising — using the prior samples . . . 114
5.3 Related work . . . 114
5.3.1 Ensembles . . . 115
5.3.2 Deep kernels . . . 116
5.3.3 Gaussian process hyperparameters . . . 116
5.4 Experimental set-up . . . 118
5.5 Synthetic benchmark experiments . . . 119
5.6 Air pollution experiments . . . 125
5.7 Discussion . . . 127
5.7.1 Upper confidence bound acquisition function . . . 127
5.7.2 Limitations of prior transfer . . . 128
5.7.3 PLeBO hyperparameter estimates . . . 129
5.7.4 Computational complexity . . . 132

5.7.5 Comparing prior transfer methods . . . 135
5.8 Conclusion . . . 136
5.8.1 Limitations . . . 137
5.8.2 Future work . . . 137

6 Ordered Transfer Hyperparameter Optimisation 139
6.1 Introduction . . . 139
6.2 Hyperparameter optimisation for machine learning . . . 140
6.2.1 XGBoost . . . 141
6.2.2 Random forest . . . 142
6.2.3 Approximate k-nearest neighbours . . . 143
6.2.4 Elastic net . . . 143
6.2.5 Support vector machines . . . 144
6.2.6 Summary . . . 144
6.3 NewsVendor benchmark . . . 144
6.4 Author contributions . . . 145
6.5 The paper . . . 145
6.6 Discussion . . . 174
6.7 Future work . . . 175

7 Conclusion 177
7.1 Findings . . . 178
7.2 Limitations . . . 179
7.3 Future work . . . 181
7.3.1 Baselines for climate benchmarks . . . 181
7.3.2 Temporal monitoring . . . 181
7.3.3 New PLeBO application . . . 181
7.3.4 Ordered air pollution monitoring . . . 182
7.4 Direct, prior and ordered transfer . . . 183

Bibliography 185

Appendices

A Publications 209
A.1 Optimising Placement of Pollution Sensors in Windy Environments . 209
A.2 Bayesian Optimisation for Active Monitoring of Air Pollution . . . 218

A.3 Bayesian Optimisation Against Climate Change: Applications and Benchmarks . . . 232
A.4 Obeying the Order: Introducing Ordered Transfer Hyperparameter Optimisation . . . 232
A.5 Data-driven Prior Learning for Bayesian Optimisation . . . 232

B Chapter 4 Supplementary Material 243
B.1 London data . . . 243
B.2 Hyperparameter optimisation . . . 243
B.3 Reproducibility . . . 244
B.4 MCMC sampling . . . 246
B.4.1 Initialisation . . . 246
B.4.2 Sampling theta . . . 246
B.4.3 Sampling eta . . . 247

Figures and Tables

Figures

1.1 Visualisation of the BO optimisation loop when making jam. 2
1.2 Apple jam tastiness 4
1.3 Apple jam tastiness for different apple bushels 5
1.4 Apple jam tastiness for four different apple bushels 6
1.5 Illustration of TryBest transfer learning method 6
1.6 Illustration of SimpleOrdered transfer learning method 7
1.7 Comparison of ordered and unordered problems using jam example . . . 9
1.8 Tastiness of apple, cloudberry, gooseberry and cherry jams 10
1.9 Illustration of direct transfer learning between different types of fruit . . 11
1.10 Tastiness meta-model 12
1.11 Blueberry jam tastiness 12
1.12 Comparison of hierarchical models 14

2.1 Visualisation of Bayesian optimisation loop 19
2.2 Fitting a Gaussian process to synthetic apple jam data. 21
2.3 Comparison of hyperparameter priors from literature 25
2.4 Fitting Gaussian processes with different hyperparameter priors to jam data 26
2.5 Comparison of different lengthscales on synthetic BO task 31
2.6 Regrets for optimisation in Fig. 2.5 32
2.7 Example tasks suitable to direct transfer 33
2.8 Comparison of prior and direct transfer, direct best 35
2.9 Comparison of prior and direct transfer, prior best 35

3.1 Example integration of BO with the optimal renewable control benchmark 61

4.1 Comparison of densities of traditional monitoring sensors 68
4.2 Example traditional and low-cost monitoring sensors 69
4.3 Example of low-cost sensor size and locations 71
4.4 Example NO_2 concentration images from the satellite data set 76
4.5 Illustration of the process of splitting the satellite data into subsets . . . 78

4.6 Examples of data snapshots from the LAQN data set 80
4.7 Comparison of LAQN sensors available for different pollutants 81
4.8 Wind kernel illustration 83
4.9 Visualisation of some of the kernels used 84
4.10 Visualisation of a Gaussian process model in an urban grid setting 85
4.11 Toy example of when the distance metric can be misleading 87
4.12 Results plot for initial experiments 90
4.13 Results plot for simple prior experiments 93
4.14 Diagram of differences between simple priors and sampled priors 95
4.15 Visualisation of hierarchical structure for sampled priors 96
4.16 Results plot for sampled prior experiments on Strong and Selection subsets 100
4.17 Results plot for sampled prior experiments on Selection test & London data 101
4.18 Exploration scores on the London data 104
4.19 Expected correlation as function of distance for London data 104

5.1 Example optimisation tasks 110
5.2 PLeBO hierarchical structure 112
5.3 Results for synthetic optimisation tasks 120
5.4 Overview of optimisation results 121
5.5 Summary of optimisation results 121
5.6 Comparison of true and learned hyperparameters 123
5.7 Comparison of the likelihood of true and learned hyperparameters 124
5.8 Comparison of true and learned hyperparameter priors 124
5.9 Results for Strong subset 126
5.10 Results for Selection subset 127
5.11 Comparison of performance of PLeBO, EI and BoTorch 128
5.12 Comparison of lengthscale estimates of PLeBO and EI 130
5.13 Comparison of signal variance estimates of PLeBO and EI 130
5.14 Hyperparameter estimate errors of PLeBO and EI 131
5.15 Distribution of PLeBO candidate weights 135
5.16 Comparison of prior transfer methods 136

6.1 Visualisation of the SimpleOrdered method 174

B.1 Sampling heights for LAQN stations used 243
B.2 Number of days with different number of station readings 244

Tables

1.1 Optimal observed sugar contents for the bushels in Fig. 1.4. 5

2.1 Example GP hyperparameter priors from literature 25

4.1 Comparison of reference sensors and low-cost sensors 71
4.2 Comparison of the two data sets used for air pollution monitoring. 75
4.3 Overview of data subsets for the satellite and London data 79
4.4 Summary of the hyperparameters of each kernel. 83
4.5 Overview of hyperparameters in the GP kernels. 84
4.6 Summary of the metrics used. 87
4.7 Hyperparameter bounds for MI version of the initial experiments 89
4.8 Confidence intervals for initial experiments 91
4.9 Confidence intervals of maximum ratio for sampled prior experiments . . 99
4.10 Confidence intervals of maximiser distance for sampled prior experiments 99
4.11 Confidence intervals for sampled prior experiments on Selection test subset 102
4.12 Confidence intervals for sampled prior experiments on London data set . 102
4.13 Run times for sampled prior experiments 103

5.1 PLeBO preprocessing times. 133
5.2 Optimisation durations of PLeBO and baselines 134
5.3 Toy higher-dimensional air pollution data. 138

7.1 Summary of direct, prior and ordered transfer 183

B.1 Parameters of proposal distribution 247

Abbreviations

BO	Bayesian optimisation
CO	Carbon monoxide
EI	Expected improvement
GP	Gaussian process
HPO	Hyperparameter optimisation
HVAC	Heating, ventilation and air conditioning
KANN	Approximate k-nearest neighbours
KNN	K-nearest neighbours
LAQN	London Air Quality Network
MAP	Maximum a posteriori (estimator)
MCMC	Markov chain Monte Carlo
MI	Multiple initialisations method from Chapter 4
ML	Machine learning
MLE	Maximum likelihood estimator
MoGP	Mixture of GPs
MSE	Mean square error
NO	Nitrogen oxide
NO_2	Nitrogen dioxide
O_3	Ozone
OTHPO	Ordered transfer hyperparameter optimisation
PES	Predictive entropy search
PI	Probability of improvement
PLeBO	Prior learning for Bayesian optimisation
$PM_{2.5}$	Particulate matter with diameter less than $2.5\,\mu m$
PM_{10}	Particulate matter with diameter less than $10\,\mu m$
PSO	Particle swarm optimisation
RBF	Radial basis function
RGPE	Ranking-weighted Gaussian process ensemble
SI	Single initialisation method from Chapter 4
SO_2	Sulphur dioxide
SVM	Support vector machine
UCB	Upper confidence bound
WHO	World Health Organization
XGBoost	Extreme gradient boosting

1

Introduction

Somebody who eats pancakes and jam can't be so awfully dangerous.

– Tove Jansson, *Finn Family Moomintroll*, 1948

This thesis is concerned with optimisation — finding minima and maxima. We use Bayesian optimisation (Section 1.1), a machine learning method designed for black-box optimisation, i.e. for optimising problems where we do not know the structure and do not have access to gradients. We mainly optimise two types of problems (Section 1.5), hyperparameter selection for machine learning models, and the placement of urban air pollution monitors.

Our focus is on optimisation tasks where we have access to evaluations of other, related optimisation tasks, so that we can apply transfer learning (Section 1.2). For example, we could know what the optimal hyperparameter setting is when the machine learning model is trained on a different data set. We consider two types of transfer: for the first we assume the optimal inputs are correlated, e.g. that the optimal hyperparameter settings are similar for different data sets. For the second type of transfer learning we only assume that the structure of the tasks is similar, e.g. in the number of local optima. Both of our applications are difficult problems with expensive evaluations, so they will benefit from improved transfer learning methods.

1.1 Bayesian optimisation

Bayesian optimisation (BO) is a machine learning (ML) method for global optimisation that uses a probabilistic model of the optimisation task. It consists of three steps: i) developing a probabilistic model of the optimisation task, ii) determining the next value to try and iii) evaluating that value. For instance, if we were optimising the

sugar content while making jam the steps would become i) update the model of tastiness ratings vs sugar added, ii) determine what sugar content to try next and iii) make a batch of jam and have your friends try it, see Fig. 1.1. We introduce Bayesian optimisation more thoroughly in Section 2.1.

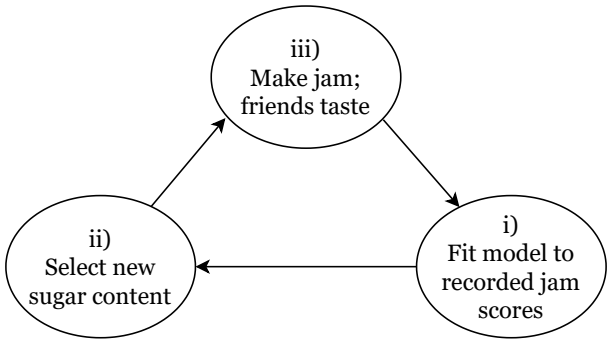


Figure 1.1: Visualisation of the BO optimisation loop when making jam.

The idea behind Bayesian optimisation is to use the model to make each evaluation as informative as possible. This is especially well-suited to optimisation problems where evaluations are expensive, for instance when they require making a batch of jam, training a machine learning model or placing an air pollution sensor. The standard choice for the probabilistic model is a Gaussian process (GP). It is defined by the choice of mean and covariance functions. These are in turn parameterised using hyperparameters. So by choosing hyperparameters, e.g. lengthscales, we also choose our model. We will use this to develop one of our methods.

Bayesian optimisation is suited to problems that have expensive evaluations, a complex, unknown structure which we cannot model directly, and involve minimisation or maximisation. Compared to particle swarm optimisation — which also is used to optimise problems with complex structures — it is suited for problems with more expensive evaluations. Within machine learning, Bayesian optimisation is related to reinforcement learning and active learning. The main differences to the former is that reinforcement learning works with reward signals, and a changing state, whereas in standard BO we assume we can freely choose the next input and evaluate it independently of previous choices. The main difference to active learning is that it seeks to approximate a function, whereas Bayesian optimisation tries to find the optimum of that function. The relationship between Bayesian optimisation and these related methods is discussed more fully in Section 2.7.

1.2 Transfer learning

Since BO is often applied to problems where evaluations are expensive, a natural question is how we can make it even more sample efficient, i.e. capable of finding optima using only a few samples. We improve the sample efficiency by using data from related optimisation tasks; this is the standard approach known as transfer learning (Bai et al., 2023). Transfer learning can take many forms, for instance selecting good initial inputs to evaluate, or learning a joint model of the new and past tasks. But transfer learning typically assumes that

1. we need to learn[1] which related tasks are most relevant; and
2. we are learning about the optimisation target directly, i.e. if a set of input values works well for earlier optimisation tasks it will work well for the new task.

In this thesis we consider separately the cases where those two assumptions do not hold. Our two approaches go in different directions compared to standard transfer learning. By altering assumption 1 we are assuming we have additional information available, i.e. we know which previous task is most relevant. By altering assumption 2 we are instead making the transfer task harder, as we are assuming the previous tasks are less relevant to our situation.

To better describe the two assumptions, we start with a motivational example based again on making jam.

1.3 Motivational example

Disclaimer: The motivational example has been intentionally simplified compared to the kinds of problems usually solved by Bayesian optimisation. For example, we assume that we only have one optimum, and can learn a parametric model. This especially makes a difference for the prior transfer, which learns the shape of the function, instead of the Gaussian process hyperparameters. The simplifications have been made to be able to explain the different methods conceptually without first detailing Gaussian processes or introducing mathematical equations.

1. This can be done either during the optimisation by comparing evaluations, or before the optimisation using meta-features of the optimisation tasks.

Consider the problem of making jam. You want to optimise the amount of sugar added per kilo of fruit, and to evaluate the jam you give it to your friends to try. Your friends are not very experienced jam-tasters, so they are not able to say whether they would like the jam more or less sweet, only to give a tastiness rating of it.

Initially, you specialise in apple jam, and buy a bushel[2] of apples. By keeping records of how well the different batches of jam are received, you are able to develop a model to predict tastiness as a function of sugar content, as visualised in Fig. 1.2. But just as you feel you have mastered the art of jam making, you run out of apples. You are able to buy a new bushel, but quickly realise that the optimal sugar content has now changed, see Fig. 1.3.

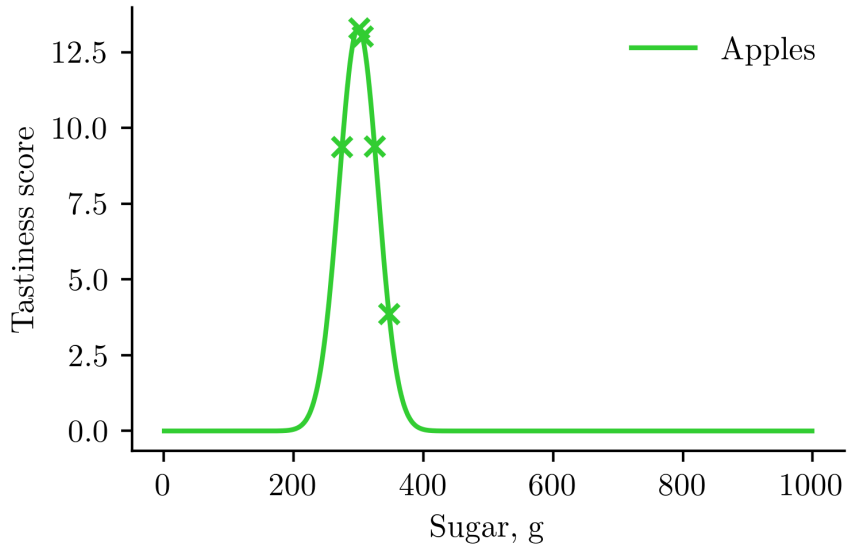


Figure 1.2: Tastiness vs added sugar for apple jam. The crosses indicate jam batches that have been rated. Synthetic data.

One approach would be to start the search again from scratch. In transfer learning we instead choose to keep the observations we have for the old apple bushel, in order to speed up optimising the sugar content for the new bushel. Different types of transfer learning do this differently, based on different assumptions of how different the optimisation tasks are. Typically, we have multiple previous tasks, which corresponds to having records for multiple previous bushels of apples, see Fig. 1.4.

2. A unit of measurement defined as 8 Imperial gallons (Britannica, 1998), or about 36 litres. You do not need to remember these numbers.

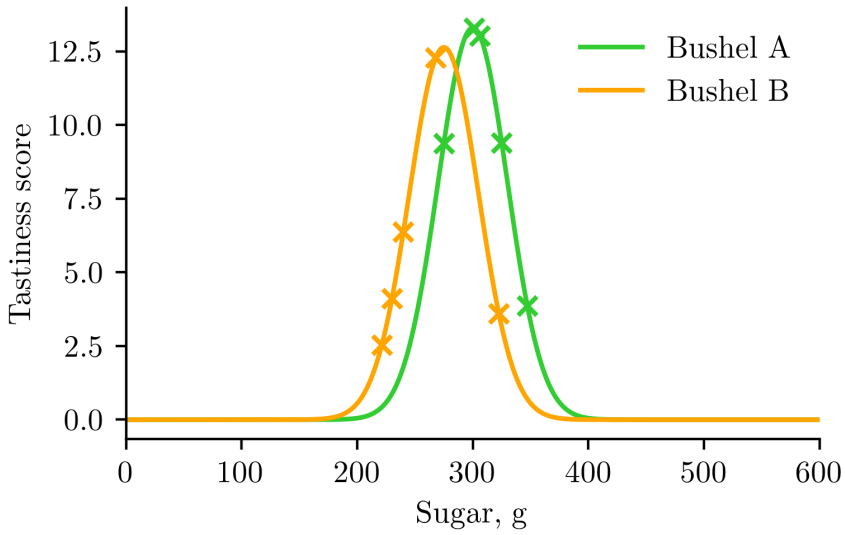


Figure 1.3: Jam tastiness vs added sugar for different bushels of apples. Synthetic data.

TryBest: A common idea, found for instance in Feurer, Springenberg, and Hutter (2015), is to start by evaluating sugar contents that worked well on previous bushels. A simple version would be to test them in order of how well they worked. Reading off Fig. 1.4, we get the values in Table 1.1. Using them, we would decide to make our first four batches of jam using 245, 301, 268 and 232 grams of sugar. Fig. 1.5 shows how this would work for a new bushel of apples.

	Sugar, g	Tastiness score
Bushel C	245	13.8
Bushel A	301	13.3
Bushel B	268	12.3
Bushel D	232	11.6

Table 1.1: Optimal observed sugar contents for the bushels in Fig. 1.4.

We will now consider two special cases which call for different approaches. In the first, the tasks come with an inherent ordering, and in the second the tasks are less closely related.

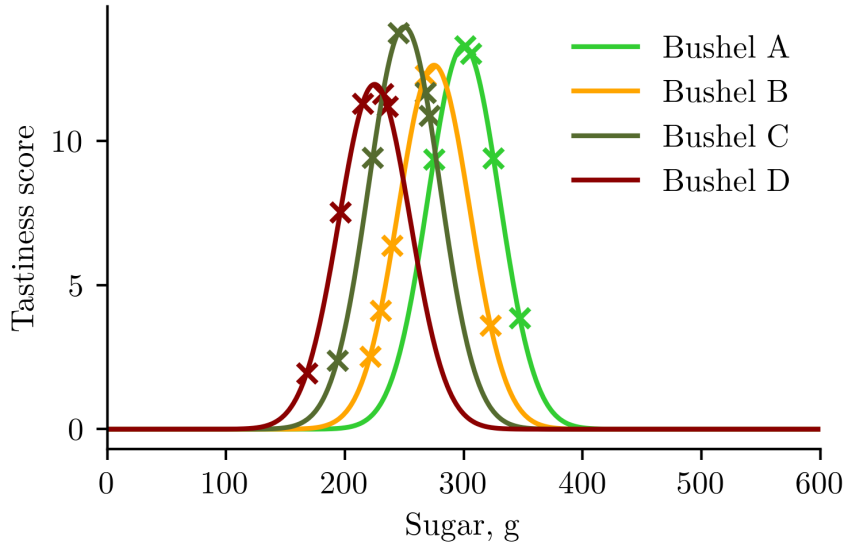


Figure 1.4: Jam tastiness vs added sugar for four bushels of apples. These bushels represent the past evaluations which we want to use to learn what sugar levels to try next. Synthetic data.

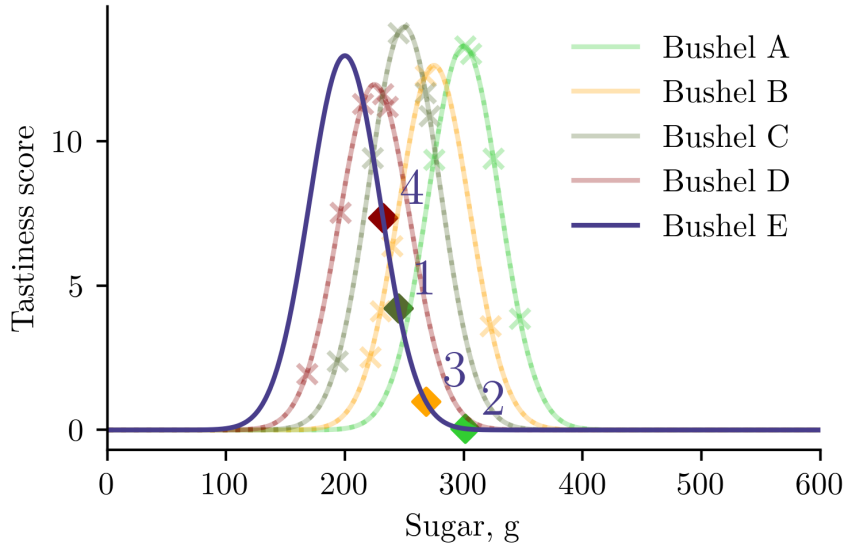


Figure 1.5: Using the TryBest strategy to optimise the sugar content of a new bushel E. The numbers indicate the order that the sugar levels are tried. We evaluate the top sugar contents from previous bushels in order of how well they worked. Synthetic data.

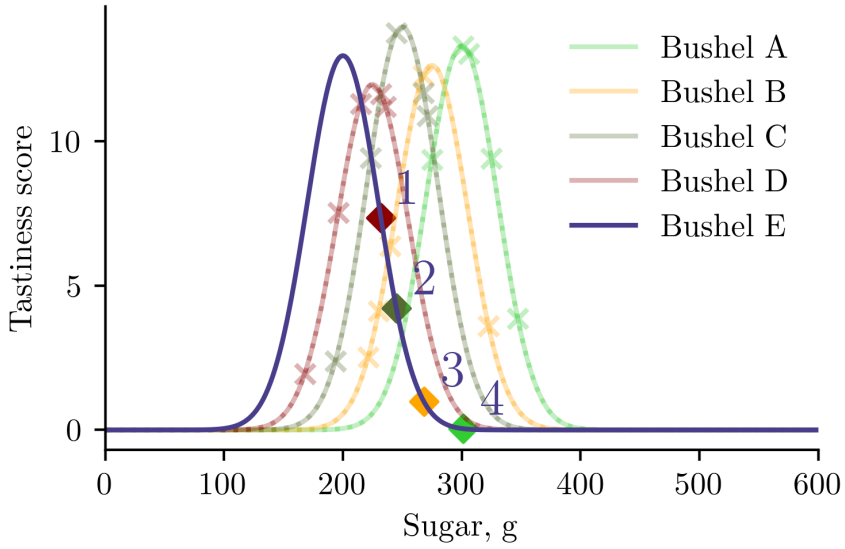


Figure 1.6: Using the SimpleOrdered strategy to optimise the sugar content of a new bushel E. The numbers indicate the order that the sugar levels are tried. We evaluate the top sugar contents from previous bushels in order of how recent they are. Synthetic data.

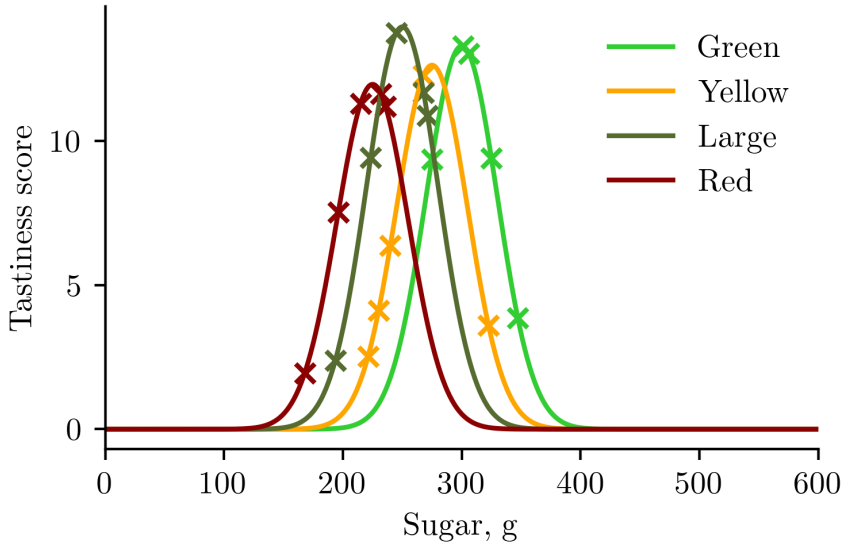
1.3.1 Ordered transfer

As time passes, you realise that the changes between apple bushels are not random as you first assumed. Instead, there is a gradual shift in the optimal sugar contents from bushel A to bushel D, see Fig. 1.4. While the apples you got early in the season were quite sour and needed more sugar, the later ones are sweeter and your friends prefer you to add less sugar to them. Based on this, you decide to take the ordering of the bushels into account when learning from previous optimisation tasks.

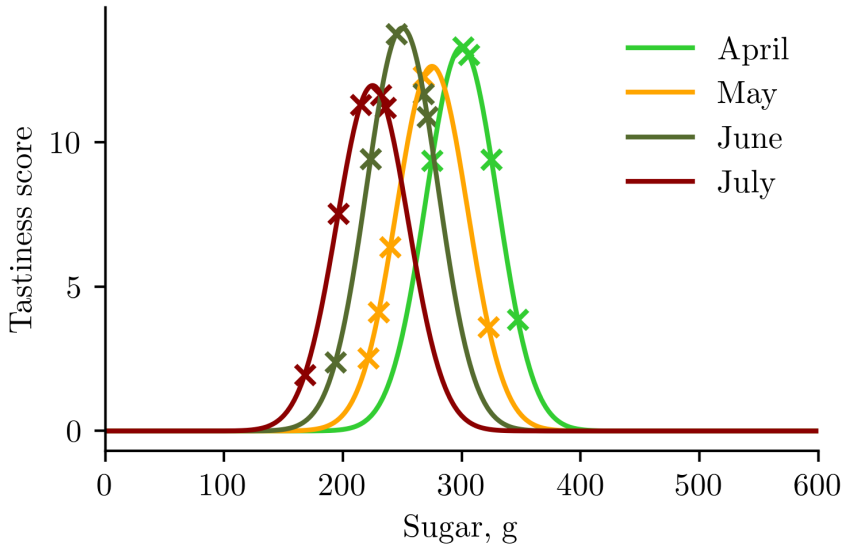
In Chapter 6 we introduce a simple method called **SimpleOrdered**. It builds on the common approach of TryBest. In this jam-making example, it uses the same approach of taking good sugar levels from previous bushels of apples, but instead of sorting them by how well the jam was rated, we sort them by how recent the bushel was. For an illustration, see Fig. 1.6. An alternative approach would be to add time as another dimension to our optimisation problem, and develop a joint model of jam tastiness given the month and added sugar. This would also be ordered transfer, but SimpleOrdered is simpler to implement and works well empirically.

The choice between TryBest and SimpleOrdered is really one of deciding whether we have a useful ordering of the tasks which we can use. Assume Fig. 1.7 represents evaluations from optimising past bushels of apples, and that you now have a new bushel of apples to make jam from. The new bushel has small apples, and was purchased in August. The two plots are identical except the labels. But in the top plot we would not know how to order the tasks from most to least relevant. In the bottom plot, on the other hand, we can see that the optimal sugar content is decreasing from month to month, and so we would consider the batches of jam from July apples to be most relevant.

By doing so, you can exploit additional information to speed up the optimisation process. If you did not know which bushels are most related you could only do the less efficient approach of assuming all bushels equally relevant when starting out on a new bushel. You could then learn which bushel was most closely related by comparing the scores obtained, but this would be slower and require more evaluations. Alternatively, you could use the bushel number or month as an extra dimension in the optimisation problem. But this is unnecessarily complicated for our current approach.



(a) Unordered problem: we do not know how to order the bushels.



(b) Ordered problem: the bushels come with an inherent order, based on the month the bushel is from.

Figure 1.7: For each bushel of apples, there is an optimal amount of added sugar. In 1.7a the bushels do not have a natural ordering: we need to learn it by comparing evaluations for the different bushels. In 1.7b the different months provide a natural ordering. Synthetic data.

1.3.2 Prior transfer

Having won new friendships with your apple jam, you become ambitious and decide to make jam from other kinds of fruit. Initially, you experiment with cherry jam, cloudberry jam and gooseberry jam, learning the optimal sugar contents from scratch. You keep meticulous records, and come up with the model in Fig. 1.8 of tastiness vs sugar content for the four types of fruit.

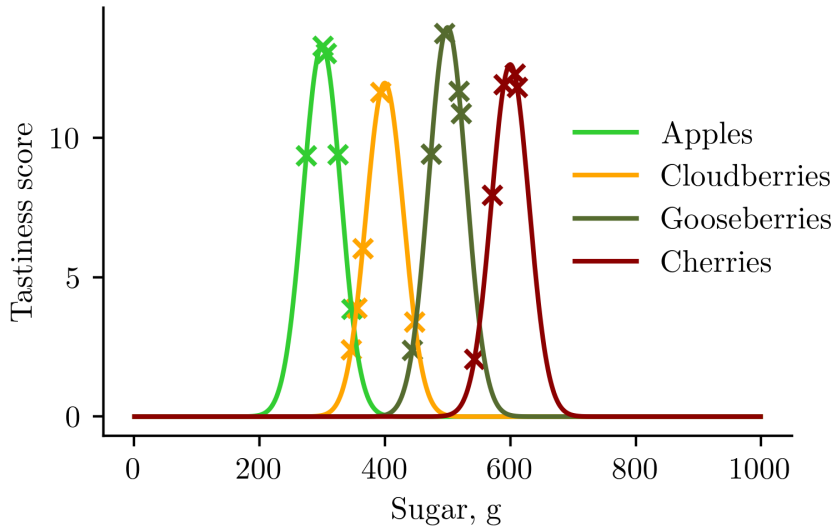


Figure 1.8: Tastiness vs added sugar for apple, cloudberry, gooseberry and cherry jams. Synthetic data.

It is clear that directly using the optimal sugar contents from previous fruits is not enough anymore. Using the ratings for apples and cherries would tell you that 300g and 600g are the best sugar contents, while 400g is very bad. But for cloudberries that is the best choice. This is also demonstrated in Fig. 1.9: none of the optimal sugar levels for previous jams are usable for blueberry jam. TryBest and SimpleOrdered are examples of **direct transfer**. When making jam from a new fruit or bushel of fruit we choose sugar levels that have worked well previously. But this multi-fruit example instead calls for a less direct approach. You realise that while the optimal sugar content is quite different for the four fruits, the shape of the curves is very similar. We call learning the shape **prior transfer**[3].

3. More formally, we are learning priors for the Gaussian process hyperparameters. For the motivational example, we are learning the shape of the function to keep the examples simple while introducing the concepts.

In **PriorLearningMotivation**[4] we learn a meta-model of the optimal sugar levels for different fruit, see Fig. 1.10. For the next kind of fruit, say blueberries, you need to try fewer sugar levels, because you can more easily fill in the gaps between the samples you already have. Because the width of the meta-model is a bit under 200 grams, we make jam batches using 100 grams, 200 grams, up to 900 grams of sugar. Then we will get at least one, probably two, batches of jam with decent ratings that we can use to fit the meta-model. See Fig. 1.11.

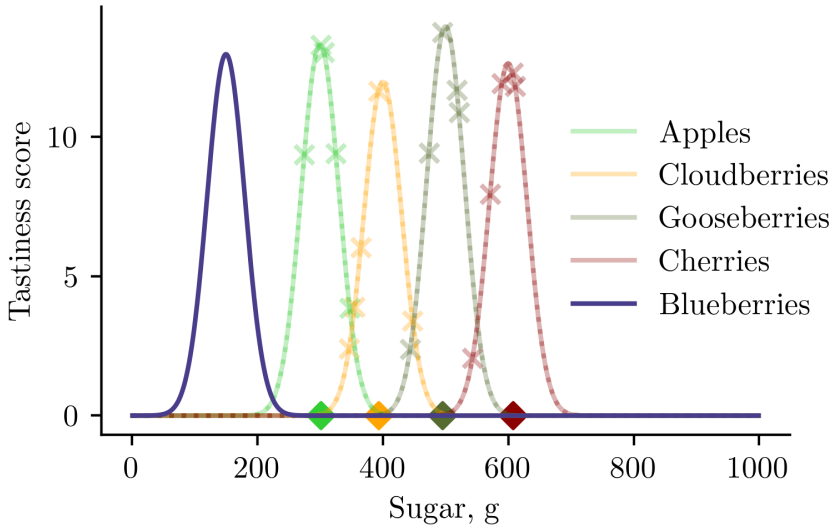


Figure 1.9: Using the direct transfer approach when switching between different types of fruit. Synthetic data.

Prior transfer is less powerful than direct transfer. If you are able to, you should use direct transfer, possibly in combination with prior transfer. But direct transfer relies on stronger assumptions on the relationship between the previous and current optimisation tasks. Prior transfer is a way of extracting some information from the related tasks, even in situations where direct transfer would be unhelpful. This is also reflected in the number of points used in the examples. In Fig. 1.11 we assume ten sugar levels are tested before we try to fit the meta-model, while in Fig. 1.6 and Fig. 1.5 we have reasonable jams after one and four evaluations, respectively.

4. This is a simplified version of the method developed in Chapters 4 and 5.

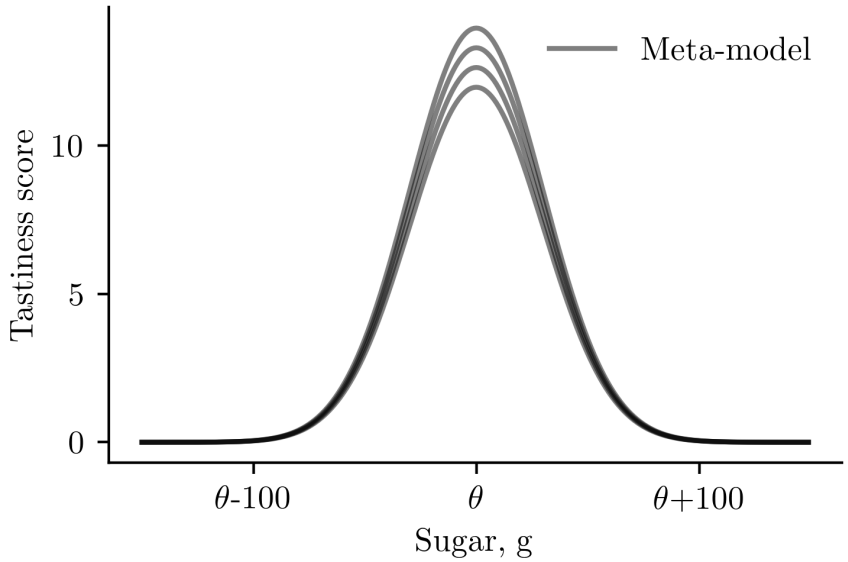


Figure 1.10: Meta-model of tastiness vs added sugar for new types of jam. You do not know what the optimal sugar content θ is for a new fruit, but you know the shape of the curve.

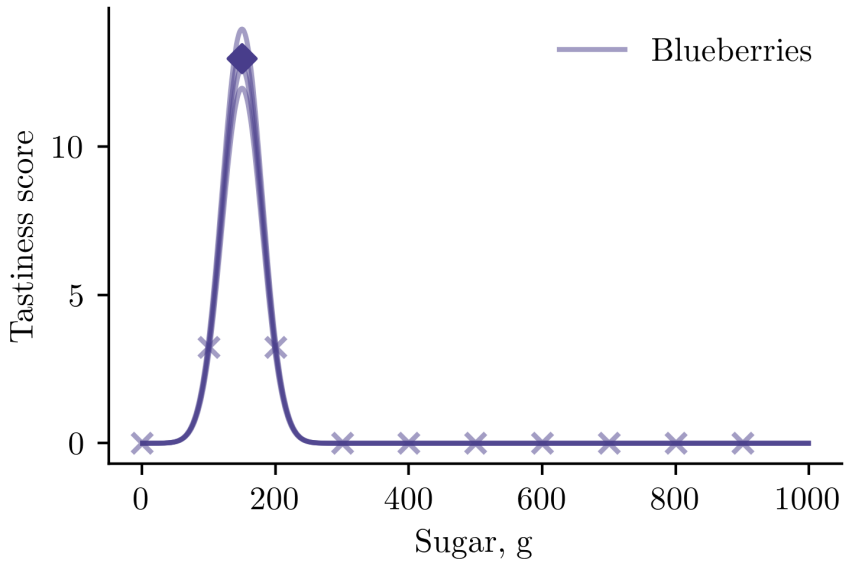


Figure 1.11: Tastiness vs added sugar for blueberry jams. Despite only having received ten jam-ratings, you can fill in the ratings for other sugar contents based on your meta-model, and use this to decide the next sugar content to use. Synthetic data.

1.4 Terminology

Using the motivational example, we now introduce some terminology. We call optimising the sugar content for a bushel of fruit an optimisation *task*. We call testing a sugar content, i.e. creating and getting feedback on a batch of jam, collecting a *sample* or an *evaluation*.

We call the first type of problem, where we were trying to learn the optimal sugar content for jam from (unordered) bushels of apples, a **direct transfer** problem. This is also the standard set-up for transfer learning (Wistuba & Grabocka, 2021). We call the second type of problem, where we can exploit an inherent ordering, an **ordered transfer** problem. We call the third type of problem, where we are learning properties of the surrogate model instead of transferring directly, a **prior transfer** problem. To summarise:

- Direct transfer: The optima are related, e.g.
  different bushels of the same fruit.
- Ordered transfer: The optima are related, and ordered, e.g.
  bushels of the same fruit as the season progresses.
- Prior transfer: Only the shapes of the function responses are related, e.g.
  different types of fruit.

More formally, for **prior transfer** we assume that the tasks have similar structure, e.g. the number and width of the optima, but that the locations of the optima themselves do not relate to one another. For **direct transfer** we assume that the locations of the optima are similar between the tasks. For **ordered transfer** we additionally assume that the optima gradually shift with the task ordering. The different assumptions are visualised in Fig. 1.12.

To determine what kind of transfer learning is best suited to a particular optimisation task, we need to consider whether the optimal inputs are similar, and whether there is an inherent ordering between the new task and past tasks. If they are similar and we have an ordering, we should apply ordered transfer. If they are similar but we do not have an ordering, we should apply direct transfer. Prior transfer is designed for applications where direct transfer is not suitable.

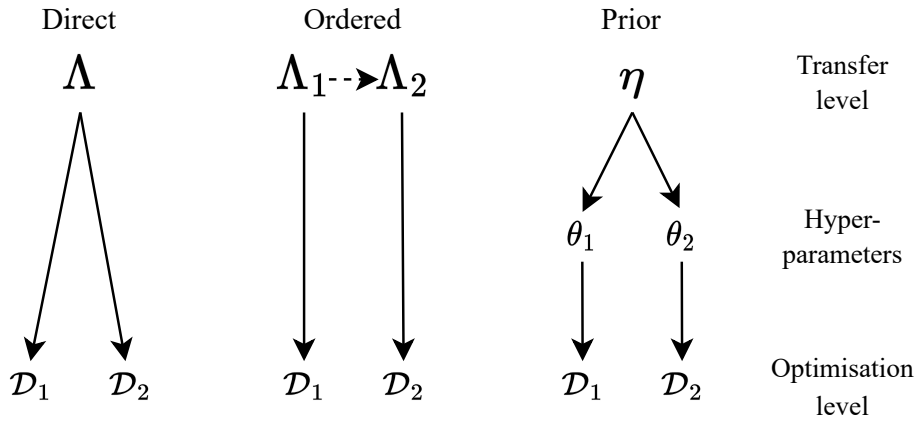


Figure 1.12: Hierarchical models of direct, ordered and prior transfer. For direct transfer we want to directly learn a model $\boldsymbol{\Lambda}$ of the optimal input values, e.g. sugar contents. For ordered transfer the model $\boldsymbol{\Lambda}$ evolves with time, making the most recent observations the most useful. This means that neighbouring $\boldsymbol{\Lambda}$s are more similar, e.g. $\boldsymbol{\Lambda}_5$ is more similar to $\boldsymbol{\Lambda}_4$ than to $\boldsymbol{\Lambda}_1$. For prior transfer we want to learn a meta-model $\boldsymbol{\eta}$ of the surrogate model hyperparameters $\boldsymbol{\theta}$. $\boldsymbol{\mathcal{D}}_1$ and $\boldsymbol{\mathcal{D}}_2$ are the observations from optimisation tasks 1 and 2.

1.5 Applications

In our experiments we do not optimise the sugar content of jam. Instead, we consider hyperparameter optimisation for machine learning, and air pollution monitoring.

Hyperparameter optimisation for machine learning (HPO) is the problem of optimising model hyperparameters, e.g. the learning rate which determines how quickly the model parameters are updated, or the size of the machine learning model. Often, the models take a long time to train, which motivates efficient ways of optimising the hyperparameters. More specifically, we consider the problem of optimising the hyperparameters as the size of the training data set increases.

Outdoor air pollution has grave health impacts around the world (Cohen et al., 2017). To evaluate adherence to legal limits, monitoring is needed. We consider a version of air pollution monitoring where the problem is that of finding the location of maximum pollution in a city using as few samples as possible. By knowing the maximum pollution concentration in a city, we know both whether an intervention is needed and where. We use historical data, and so limit the sampling to places which currently have sensors, using data from the London Air Quality Network (Imperial College London, 1993) and the Sentinel-5P satellite of the Copernicus programme (Copernicus Sentinel-5P (processed by ESA), 2018).

We treat HPO as an ordered transfer problem. We expect the hyperparameters we optimise, such as the learning rate, to gradually change from one optimisation task to the next as we increase the size of the training data. We treat air pollution monitoring as a prior transfer problem because our motivation is learning between cities. If the maximum pollution in London happens to be 5 km east and 2 km north of the city centre, we do not expect that to be the case for Berlin or Paris. But we do expect the patterns to be similar, e.g. in terms of how quickly the pollution concentration changes as we move away from a road.

The extensions we consider to Bayesian optimisation transfer learning have other applications as well. Ordered transfer has applications whenever the set of optimisation tasks comes from a series of adjustments. One example would be using Bayesian optimisation to configure the settings of a climate model we are fitting to historical data, and re-tuning every month as more data are collected. Prior transfer has applications when we have related data but either the tasks are not closely related or we have reason to believe the optima to be in different parts of the input space. For instance, this could be optimising different types of chemical reactions or the selling price of different products.

1.6 Contributions

The contributions of the thesis are summarised below. We first present a survey of applications and benchmarks for Bayesian optimisation applied to climate-related problems in Chapter 3. Then we introduce the air pollution monitoring optimisation problem, and a novel method for transfer learning motivated by this setting, in Chapter 4. Afterwards, we further analyse and develop this method in Chapter 5. In Chapter 6 we develop an improvement to direct transfer for hyperparameter optimisation using the task ordering.

Chapter 3: Survey of Bayesian optimisation applied to climate-related problems. This chapter is based on the paper *Bayesian Optimisation Against Climate Change: Applications and Benchmarks* with Christopher Lucas and Nigel Goddard, presented at the Data-centric Machine Learning Research Workshop at ICML 2023 (Hellan, Lucas, & Goddard, 2023a).

Chapter 4: Applying Bayesian optimisation to air pollution monitoring. This chapter builds on *Optimising Placement of Pollution Sensors in Windy Environments* with Christopher Lucas and Nigel Goddard, presented at the AI for Earth Sciences Workshop at NeurIPS 2020 (Hellan, Lucas, & Goddard, 2020), and *Bayesian Optimisation for Active Monitoring of Air Pollution* with Christopher Lucas and Nigel Goddard, presented at AAAI 2022 (Hellan, Lucas, & Goddard, 2022).

Chapter 5: Improvements to the method developed in Chapter 4, and comparison to other transfer learning methods. Based on *Data-driven Prior Learning for Bayesian Optimisation* with Christopher Lucas and Nigel Goddard, presented at the NeurIPS 2023 Workshop on Adaptive Experimental Design and Active Learning in the Real World (Hellan, Lucas, & Goddard, 2023b).

Chapter 6: Ordered transfer learning applied to hyperparameter optimisation. Based on *Obeying the Order: Introducing Ordered Transfer Hyperparameter Optimisation* with Huibin Shen, François-Xavier Aubet, David Salinas and Aaron Klein, colleagues during an internship at Amazon Web Services. Presented at the AutoML 2023 conference workshop (Hellan, Shen, Aubet, Salinas, & Klein, 2023).

So far we have discussed ordered transfer learning before prior transfer, as the former is conceptually simpler. We will also use this ordering in the background chapter, Chapter 2. But in structuring the rest of the thesis we swap orders and considered prior transfer first. This allows the thesis to shift from an application focus in Chapters 3 and 4 to a methodology focus in Chapters 5 and 6.

1.6.1 Bayesian optimisation against climate change

In Chapter 3 we present a survey of applications and benchmarks for Bayesian optimisation within the topic of climate change. It is based on Hellan, Lucas, and Goddard (2023a). We identify four main application domains: material discovery, wind farm planning, renewable generator control and environmental monitoring. Surveys existed of climate change applications of machine learning in general (Donti & Kolter, 2021; Rolnick et al., 2022), and of surrogate based optimisation — of which Bayesian optimisation is one example — for sustainability (Bliek, 2022). But machine learning

is a very broad spectrum, and different methods are useful for different applications. Knowing what applications benefit from supervised learning does not tell you where to use Bayesian optimisation. And the survey by Bliek focuses on methodology, not applications and benchmarks, so this survey and our survey complement each other.

Chapter 3 is different to the others in that it is not directly interested in transfer learning for Bayesian optimisation. This is because it is a survey of relevant *applications* of Bayesian optimisation, and is agnostic to the specific flavour of Bayesian optimisation chosen. But we expect transfer learning to be instrumental in improving the performance on the benchmarks, as most of them require high sample efficiency.

1.6.2 Prior learning

Our method for prior transfer was motivated by and designed for air pollution monitoring: we first applied standard Bayesian optimisation, but found that we needed to introduce hyperparameter priors. We first used two techniques to introduce prior information: we adapted the GP covariance function to model the prevailing wind direction, and we learned a basic distribution for the GP hyperparameters. We presented this work in Hellan et al. (2020). For harder optimisation tasks this was not enough to beat random sampling, so we moved on to a more sophisticated approach for learning the hyperparameter priors. This uses Markov chain Monte Carlo (MCMC) to generate candidate hyperparameter values and then leverages them using importance weighting, as presented in Hellan et al. (2022). These two contributions are presented together in Chapter 4.

In Chapter 5 we extend and analyse the method from Chapter 4. We call the improved method Prior Learning for Bayesian Optimisation, or PLeBO. We put it in the context of other transfer learning methods, both direct transfer and prior transfer, and compare its performance on air pollution monitoring as well as synthetic data. The synthetic data are useful as they let us analyse the quality of the generated hyperparameter candidates. We also provide a new implementation using existing libraries for the Gaussian processes and MCMC, which makes the method more modular, as well as easier to modify and extend. And we compare the computational complexities of the methods. Some of the content from this chapter is presented in Hellan, Lucas, and Goddard (2023b).

1.6.3 Ordered transfer learning

Our approach for ordered transfer learning is motivated by repeated hyperparameter optimisation as more training data are collected. In Chapter 6, based on Hellan, Shen, et al. (2023), we introduce Ordered Transfer Hyperparameter Optimisation (OTHPO), and our method SimpleOrdered, a simple method that leverages the task ordering to outperform standard transfer learning methods. SimpleOrdered starts by evaluating the top hyperparameters for the most recent tasks, and then continues using standard Bayesian optimisation. We evaluate it on a set of ten benchmarks, nine of which represent HPO on five different types of machine learning models. Our work follows common assumptions when retuning hyperparameters, but these had not previously been thoroughly tested.

2

Background

For sale, especially fluid mechanics.

– Unknown student, NTH 1970s

This chapter presents background material on Bayesian optimisation and transfer learning, which we build on in the later chapters. Background material on the applications — air pollution and hyperparameter optimisation — can be found in Chapters 4 and 6, respectively. The section on prior transfer, Section 2.6.2, is especially relevant for Chapters 4 and 5, while the section on direct transfer, Section 2.6.1, is especially relevant for Chapter 6.

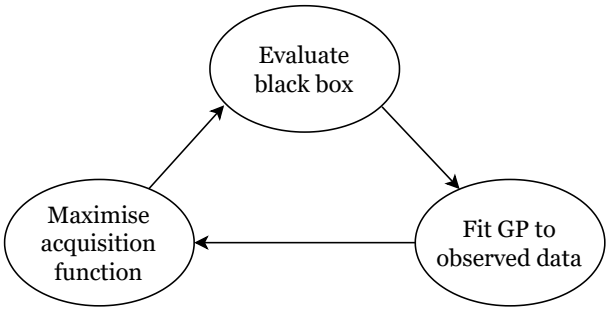


Figure 2.1: Visualisation of the BO loop with a Gaussian process model.

We start in Section 2.1 by describing Bayesian optimisation and its main components. Fig. 2.1 is a version of Fig. 1.1 focusing on the methodology. The model we use is a Gaussian process, described in Section 2.2, and the process of fitting its hyperparameters is described in Section 2.3. The process of selecting the new input value is automated through the use of an acquisition function, which uses the model to score the utility of potential inputs, see Section 2.4. Section 2.5 illustrates the problem of model misspecification, and Section 2.6 introduces transfer learning for Bayesian optimisation. Finally, Section 2.7 discusses alternatives to Bayesian optimisation and Section 2.8 acknowledges key software libraries used in the thesis.

2.1 Bayesian optimisation

In Bayesian optimisation (Garnett, 2023; Shahriari, Swersky, Wang, Adams, & de Freitas, 2015; Snoek, Larochelle, & Adams, 2012) we are interested in solving problems of the type

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}} \ f(\boldsymbol{x}) \tag{2.1}$$

where $\boldsymbol{x}$ are the inputs to a problem, e.g. the amount of sugar added to a jam recipe. $f(\boldsymbol{x})$ is the unknown function relating $\boldsymbol{x}$ to the output, e.g. the tastiness of the jam, which we want to maximise. Note that we can use the same methodologies for minimisation and maximisation, as minimising $f(\boldsymbol{x})$ is equivalent to maximising $-f(\boldsymbol{x})$. We assume that we do not have access to an analytical expression of $f(\boldsymbol{x})$, or its derivatives[1]. Instead we treat $f(\boldsymbol{x})$ as an expensive black box we can evaluate. We use a surrogate model $g(\boldsymbol{x}; \boldsymbol{\theta})$ — typically, as in our work, a Gaussian process (Section 2.2) — to approximate $f(\boldsymbol{x})$. $\boldsymbol{\theta}$ represents the model hyperparameters, which are typically tuned within each BO loop, for instance using gradient-based optimisation on the negative log marginal likelihood of $\boldsymbol{\theta}$ (Shahriari et al., 2015). To initialise $g(\boldsymbol{x}; \boldsymbol{\theta})$ we pick a set of S starting points, for instance by random sampling. These starting points, sometimes called the initial design, are the inputs $\boldsymbol{x}$ we evaluate before starting Bayesian optimisation. We use an acquisition function (Section 2.4) to determine which input $\boldsymbol{x}$ to evaluate next.

A popular application of Bayesian optimisation is hyperparameter optimisation (Feurer & Hutter, 2019). We apply BO to HPO in Chapter 6 and give an introduction to HPO in Section 6.2. In HPO we are trying to learn the hyperparameters of a machine learning method, for instance random forest. Note that by HPO we mean using BO and its surrogate to fit the hyperparameters of a different ML model, not tuning the hyperparameters of the BO surrogate model.

In general, Bayesian optimisation is well suited to problems that combine the following properties:

- A complex, unknown structure which can only be efficiently modelled through a surrogate model, e.g. the performance of a machine learning model.
- Expensive or slow evaluations requiring informative sampling choices, e.g. installing an air pollution monitor or learning a machine learning model on a large data set for a given set of hyperparameter values.

---

1. Bayesian optimisation can be extended to include derivative information, see e.g. Ament and Gomes (2022), but we commonly assume we do not have access to derivatives.

- Require identifying extrema, e.g. making the tastiest possible batch of jam.

2.2 Gaussian processes

The most commonly used surrogate models for Bayesian optimisation are Gaussian processes (Garnett, 2023, p. 15), described in detail in Rasmussen and Williams (2006). Gaussian processes are probabilistic machine learning models. They are popular for relatively small data sets, where the Bayesian treatment aims to avoid overfitting, but do not scale naturally to large data sets due to a cubic scaling term for posterior computations or model fitting[2]. Fig. 2.2 shows an example one-dimensional Gaussian process, using the example jam problem from Section 1.3. As can be seen, Gaussian processes can model our uncertainty regarding the underlying function.

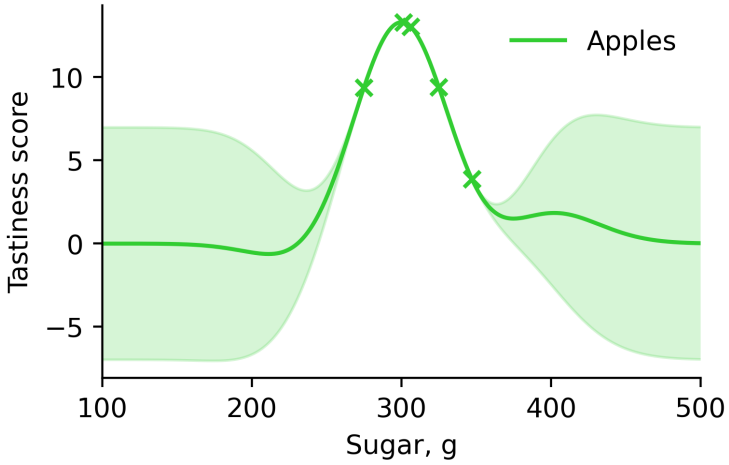


Figure 2.2: Fitting a Gaussian process to synthetic apple jam data.

Gaussian processes are defined by the choice of mean and covariance functions. We will follow common practice and assume a prior mean of zero (De Ath, Fieldsend, & Everson, 2020). The covariance function $k(\boldsymbol{x}_1, \boldsymbol{x}_2; \boldsymbol{\theta})$ takes a pair of feature vectors and outputs the expected covariance of the function $f(\boldsymbol{x})$ for those inputs. We then stack these covariances into matrices which we use to define the posterior predictive distribution in Eqs. (2.2) and (2.3) (Rasmussen & Williams, 2006, p. 16). $\boldsymbol{K}_{pp}(\boldsymbol{\theta})$ is the covariance function evaluated on the test points $\boldsymbol{X}_p$. $\boldsymbol{K}_p(\boldsymbol{\theta})$ is the covariance

2. Various approaches to scalable Gaussian processes exist, but generally require approximations. One approach is to summarise the data set with a smaller number of data points (X. Liu et al., 2020).

between the observed points and the test points. $\boldsymbol{K}(\boldsymbol{\theta})$ is the covariance between the observed points. σ_n^2 is the variance of the Gaussian noise in the likelihood. $\boldsymbol{y}$ is the observed function evaluations. $\boldsymbol{\mu}$ is the predicted mean and $\mathbb{V}$ the predicted covariance of the test points. $\boldsymbol{\mu}$ and $\mathbb{V}$ represent the posterior model and can be used to generate posterior samples.

$$\boldsymbol{\mu}(\boldsymbol{X}_p;\boldsymbol{\theta}) = \boldsymbol{K}_p(\boldsymbol{\theta})^T \left(\boldsymbol{K}(\boldsymbol{\theta}) + \sigma_n^2 \boldsymbol{\mathcal{I}}\right)^{-1} \boldsymbol{y} \tag{2.2}$$

$$\mathbb{V}(\boldsymbol{X}_p;\boldsymbol{\theta}) = \boldsymbol{K}_{pp}(\boldsymbol{\theta}) - \boldsymbol{K}_p(\boldsymbol{\theta})^T \left(\boldsymbol{K}(\boldsymbol{\theta}) + \sigma_n^2 \boldsymbol{\mathcal{I}}\right)^{-1} \boldsymbol{K}_p(\boldsymbol{\theta}) \tag{2.3}$$

The $\sigma_n^2 \boldsymbol{\mathcal{I}}$ term in Eqs. (2.2) and (2.3) comes from the observation noise, which we assume to be independent, homogeneous and Gaussian. Other modelling choices are possible, but mostly make the calculations less straightforward. For instance, we could assume a Poisson observation model and heteroskedastic noise (Rasmussen & Williams, 2006, pp. 190–191; Cowen-Rivers et al., 2022).

Our choice of covariance function $k(\boldsymbol{x}_1, \boldsymbol{x}_2; \boldsymbol{\theta})$, also called the kernel, and hyperparameters $\boldsymbol{\theta}$ encode our beliefs about the function. Two common covariance functions are the radial basis function (RBF) in Eq. (2.4) and Matérn 5/2 in Eq. (2.5) (Rasmussen & Williams, 2006, pp. 83–85). These are both stationary, so only dependent on the difference $\boldsymbol{\tau} = \boldsymbol{x}_1 - \boldsymbol{x}_2$ between input values.

$$k_{RBF}(\boldsymbol{\tau}) = \sigma_r^2 \exp\left(-\frac{\boldsymbol{\tau}^T \boldsymbol{\tau}}{2l^2}\right) \tag{2.4}$$

$$k_{M2.5}(\boldsymbol{\tau}) = \sigma_r^2 \left(1 + \frac{\sqrt{5}\,||\boldsymbol{\tau}||_2}{l} + \frac{5\,\boldsymbol{\tau}^T \boldsymbol{\tau}}{3l^2}\right) \exp\left(-\frac{\sqrt{5}\,||\boldsymbol{\tau}||_2}{l}\right) \tag{2.5}$$

2.3 Hyperparameters

The GP's covariance function is typically selected manually[3]. Given a set of covariance functions under consideration, we would fit the hyperparameters for each model, and then compare the best fits. The comparison can be done using the Bayesian information criterion, which adds a penalty term to the hyperparameter likelihood based on the number of hyperparameters (Schwarz, 1978; Garnett, 2023, p. 79; Murphy, 2022, p. 185). Given a covariance function, the hyperparameters $\boldsymbol{\theta}$ are set based on the observed data. Generally, most of the hyperparameters come from the covariance function. For a standard set-up with an RBF kernel, the hyperparameters would be the noise variance σ_n^2, the lengthscale l, and the signal variance σ_r^2. The noise variance comes from the surrogate model likelihood, i.e. what noise distribution we assume, and the lengthscale and signal variance from the covariance function. A simple approach to tuning the hyperparameters is to use gradient-based optimisation on the negative log marginal likelihood. We can calculate the log marginal likelihood as (Rasmussen & Williams, 2006, pp. 113–114)

$$\log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta}) = -\frac{1}{2}\boldsymbol{y}^T\boldsymbol{K}_y(\boldsymbol{\theta})^{-1}\boldsymbol{y} - \frac{1}{2}\log|\boldsymbol{K}_y(\boldsymbol{\theta})| - \frac{n}{2}\log 2\pi \tag{2.6}$$

where $\boldsymbol{K}_y(\boldsymbol{\theta}) = \boldsymbol{K}(\boldsymbol{\theta}) + \sigma_n^2\boldsymbol{\mathcal{I}}$ and n is the number of observed points. Note that this computation can become expensive, as it scales cubically in n. With this equation, we can calculate derivatives with respect to $\boldsymbol{\theta}$, $\frac{d}{d\boldsymbol{\theta}}\log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta})$, and use these in gradient-based optimisation. We then obtain the **MLE** (maximum likelihood estimator).

Alternatively, we could define a prior on our hyperparameters, $p(\boldsymbol{\theta})$, for instance specifying $p(l) = \Gamma(l;3,6)$ for the lengthscale l from Eqs. (2.4) and (2.5), as is done in the software package BoTorch (Balandat et al., 2020). Then we can follow the gradient of $\boldsymbol{\theta}$, $\frac{d}{d\boldsymbol{\theta}}\log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta}) + \log p(\boldsymbol{\theta})$. This is called the maximum a posteriori or **MAP** estimator.

Both the MLE and MAP estimates of $\boldsymbol{\theta}$ are point estimates. An alternative is to use Markov chain Monte Carlo to generate a set of samples that approximates $p(\boldsymbol{\theta}|\boldsymbol{y})$ (Lalchand & Rasmussen, 2020). This is more able to deal with uncertainty regarding the best hyperparameter values, which is relevant for the early iterations of BO which

3. The kernel can also be learned, but this requires either many observations from the current task, which we generally do not have in BO, or access to evaluations of related optimisation tasks. We discuss the latter as a version of transfer learning in Section 2.6.2.

require fitting a Gaussian process to very little data. The flip side is that this approach is slower, both in the hyperparameter tuning step, which then requires collecting MCMC samples, and in the posterior prediction step, as the surrogate model becomes a mixture of Gaussians.

2.3.1 Priors

We can set priors on the hyperparameters of the Gaussian process to enable MAP estimates or MCMC approximations. These priors can improve performance, but are generally set by experts and encode assumptions about the underlying function. They are specific to the choice of covariance function, and need to be appropriate to the problem at hand. For instance, if we know that our measurements are very accurate, we can specify a narrow and low band on the noise variance. But if we do not know the measurement accuracy we need to keep the prior broad to account for the possibility that the measurements are very noisy. And a broad prior is less informative. Similarly, if we know that our underlying function has many optima we can restrict the lengthscale to be small, but otherwise we need to keep that broad too.

In addition to qualitative choices such as 'small lengthscale', we need to choose what distribution to encode our choice with, and the parameter values of this distribution. For instance, we could use a uniform distribution, a log normal distribution, or a gamma distribution. These priors can be tricky to reason with. For instance, many people do not have an intuition for how to visualise the parameters of the gamma distribution. And we need to set a prior for each of the covariance function's hyperparameters. Therefore, a method for automatically setting the hyperparameter priors would be advantageous.

To illustrate the diversity of priors in current use, we consider three examples of recently used priors, summarised in Table 2.1 (Balandat et al., 2020; Cowen-Rivers et al., 2022; Müller, Feurer, Hollmann, & Hutter, 2023). For BoTorch we consider version 0.8.5[4], but set the mean to zero to align with the other priors. All three combine the priors with input and output transformations[5]. Fig. 2.3 illustrates these priors for the signal variance. As can be seen, the priors vary greatly. This is partly due to other modelling choices: the three methods use different kernels, and the applied input transformation is also different. Fig. 2.4 shows GP models fitted to the jam example

4. Available at `https://github.com/pytorch/botorch/tree/v0.8.5`
5. BoTorch does not do it automatically, but requests the user to do so.

Model	kernel	lengthscale	output	noise
Uniform	Matern 5/2	-	-	-
BoTorch	Matern 5/2	$\Gamma(3, 6)$	$\Gamma(2, 0.15)$	$\Gamma(1.1, 0.05)$
HEBO	Linear + Matern 3/2	-	$\Gamma(0.5, 1)$	Lognormal(-4.63, 0.5)
PFNs4BO	Matern 3/2	Γ(1.2107, 1.5212)	Γ(0.8452, 0.3993)	Lognormal(-4.63, 0.5)

Table 2.1: Examples of GP hyperparameter priors from BoTorch (Balandat et al., 2020), HEBO (Cowen-Rivers et al., 2022) and PFNs4BO (Müller et al., 2023), and a uniform, improper prior for comparison. The gamma parameters are the rate and concentration.

in Fig. 2.2 using priors from Table 2.1. We see that while the priors can be useful — the PFNs4BO prior comes up with a much better model after two observations than uniform — they can also be misleading. For 5 observations the uniform prior works better than BoTorch. The priors matter for small amounts of data, but become less important when there are many observations.

The above example shows that there is no universally accepted covariance function or hyperparameter prior, which motivates the research question of how we can set the prior automatically. In the method developed in Chapters 4 and 5 we automatically learn gamma priors from the available data. Müller et al. (2023) did a simple version of this, where they do a random search on the hyperparameter priors using a validation set.

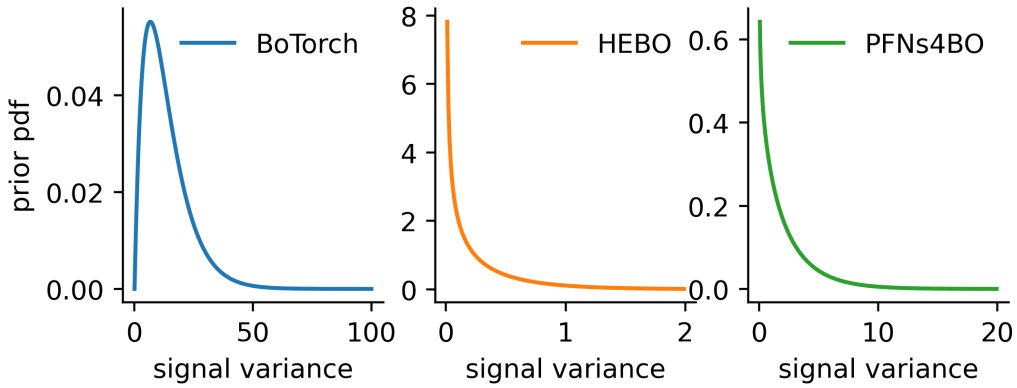


Figure 2.3: Comparison of signal variance priors used in BoTorch (Balandat et al., 2020), HEBO (Cowen-Rivers et al., 2022) and PFNs4BO (Müller et al., 2023). Note the difference in axes between plots.

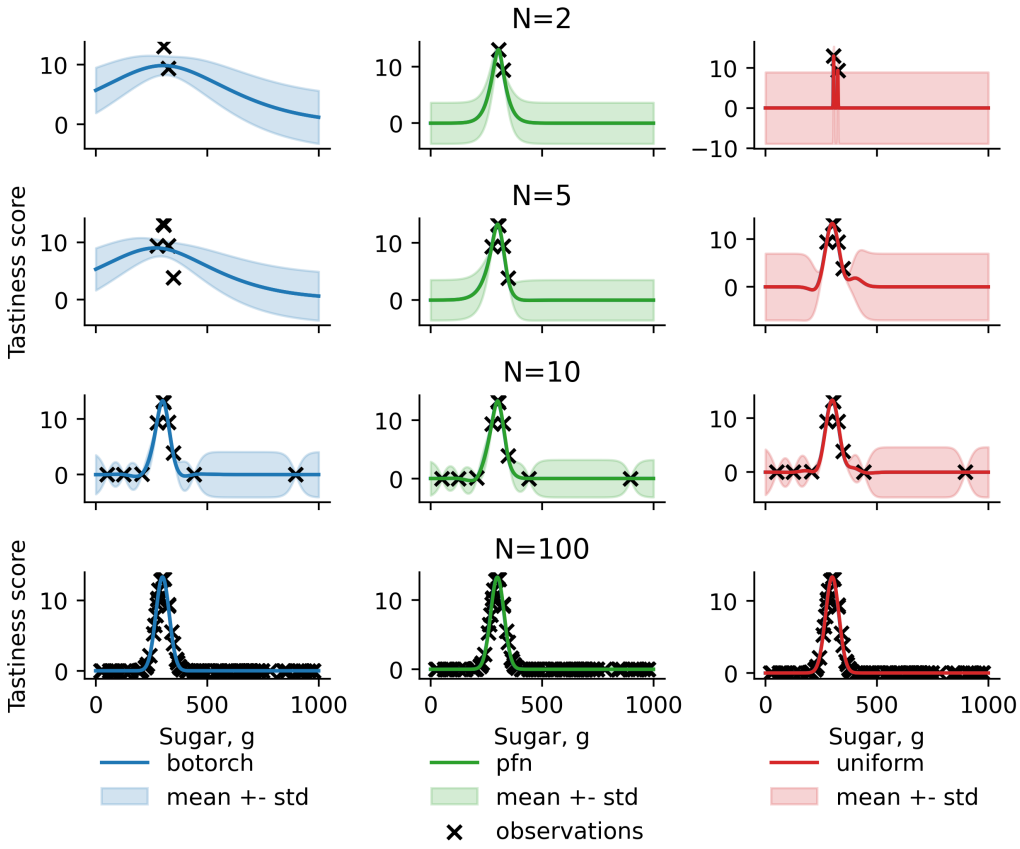


Figure 2.4: Fitting Gaussian processes with different hyperparameter priors to synthetic jam data. The left column uses the BoTorch prior (Balandat et al., 2020), the central column the PFNs4BO prior (Müller et al., 2023) and the right column a uniform prior. We did not use the input and output transformations.

2.3.2 Approximate inference

Although a simple approach for BO is to use a single point estimate for the hyperparameters $\boldsymbol{\theta}$, a more Bayesian approach has several advantages, including potentially better sample efficiency and more accurate estimates of posterior uncertainty (De Ath, Everson, & Fieldsend, 2021; Snoek et al., 2012). Instead of a single sample, a set of samples (Monte Carlo approximation) or a parameterised distribution (variational inference) can be used to represent the hyperparameters. However, it makes the inference procedure more computationally challenging. The posterior can be calculated in closed form for the simple GP model with a point estimate of the hyperparameters (Rasmussen & Williams, 2006, p. 16), but this is not possible for the more Bayesian models with sets of distributions for hyperparameter values. Following previous work, we use Monte Carlo approximation rather than variational inference in Chapters 4

and 5, as De Ath, Everson, and Fieldsend (2021) found variational inference to be less accurate. MCMC also has the advantage that we can make our approximation more and more accurate by increasing the number of samples. The Bayesian treatment of hyperparameters has also been considered for non-BO Gaussian processes, e.g. in Lalchand and Rasmussen (2020), Murray and Adams (2010) and Williams and Rasmussen (1995).

Conceptually, MCMC consists of generating a path of samples by making small adjustment to the current sample $\boldsymbol{\theta}_i$. The whole path should then represent the underlying distribution, with more samples in regions with higher probability density. A basic approach to MCMC is Metropolis-Hastings (Chib & Greenberg, 1995). Within each step in the path we then i) generate a new sample proposal $\boldsymbol{\theta}'_i$, ii) calculate an acceptance ratio $a_\theta(\boldsymbol{\theta}'_i, \boldsymbol{\theta}_i)$, and iii) accept the new sample with probability proportional to $a_\theta(\boldsymbol{\theta}'_i, \boldsymbol{\theta}_i)$. If the new sample is accepted we set $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}'_i$, otherwise $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i$.

The proposal distribution $q(\boldsymbol{\theta}'_i, \boldsymbol{\theta}_i)$ denotes the probability of proposing $\boldsymbol{\theta}'_i$ given $\boldsymbol{\theta}_i$. $\pi(\boldsymbol{\theta})$ is the unnormalised target density we want to approximate. The acceptance ratio is defined as

$$a_\theta(\boldsymbol{\theta}'_i, \boldsymbol{\theta}_i) = \min\left[\frac{\pi(\boldsymbol{\theta}'_i)q(\boldsymbol{\theta}_i, \boldsymbol{\theta}'_i)}{\pi(\boldsymbol{\theta}_i)q(\boldsymbol{\theta}'_i, \boldsymbol{\theta}_i)}, 1\right] \tag{2.7}$$

We then generate a value r uniformly between 0 and 1, and accept $\boldsymbol{\theta}'_i$ if $r < a_\theta(\boldsymbol{\theta}'_i, \boldsymbol{\theta}_i)$.

In practice, more sophisticated MCMC methods, e.g. NUTS (Hoffman & Gelman, 2014; Murphy, 2023, p. 510) are often used (Garnett, 2023, pp. 75–76). NUTS uses gradients and auxiliary variables to improve the sample efficiency. Metropolis-Hastings is simpler to understand and implement, and is what we use in Chapter 4. In our improved method in Chapter 5 we instead use NUTS.

2.4 Acquisition functions

Acquisition functions use the probabilistic surrogate model to determine what input values to evaluate next. The standard acquisition functions for Bayesian optimisation can be split in two categories: policies based on improvement or optimism, and information-theoretic policies (Shahriari et al., 2015). The former tries to choose a point to maximise the function directly, e.g. the point with the highest expected improvement over the best observation yet according to the surrogate model (Jones,

Schonlau, & Welch, 1998). The information-theoretic ones calculate the information gain using mutual information. One example is predictive entropy search (Hernández-Lobato, Hoffman, & Ghahramani, 2014), where we maximise our information gain regarding the maximiser.

The acquisition functions discussed are all myopic, in that they do not take the remaining evaluation budget into account, and only plan for one point at a time. Other work has extended Bayesian optimisation to be non-myopic (González, Osborne, & Lawrence, 2016) and to plan multiple evaluations at a time (Snoek et al., 2012).

2.4.1 Optimistic and improvement-based policies

This category comprises the most straightforward and commonly used acquisition functions. Eqs. (2.8) to (2.10) give three common acquisition functions: UCB (Upper Confidence Bound, Srinivas, Krause, Kakade, and Seeger (2012)), EI (Expected Improvement, Jones et al. (1998)) and PI (Probability of Improvement, Kushner (1964)). UCB is intuitive and simple to implement: we have two terms that we get directly from the predictive equations, Eqs. (2.2) and (2.3). The first is the posterior mean $\mu(\boldsymbol{x};\boldsymbol{\theta})$, and the second a scaled version of the posterior standard deviation $\sigma(\boldsymbol{x};\boldsymbol{\theta}) = \sqrt{\operatorname{diag}(\mathbb{V}(\boldsymbol{x};\boldsymbol{\theta}))}$. The combination therefore favours a high mean and high uncertainty. A drawback is the β parameter balancing them which needs to be set. $\boldsymbol{\mathcal{D}}_n$ is the observed data. Expected improvement and probability of improvement both use the concept of improvement $f(\boldsymbol{x}) - f'$ which is how much we can improve over the incumbent best evaluation f'. α_{PI} is simply the probability that $f(\boldsymbol{x})$ will be better than f'. α_{EI} additionally considers how big the improvement will be. EI is the most commonly used in practice. It is simple to implement and performs well compared to PI and UCB. Both EI and PI have closed-form expressions (Shahriari et al., 2015).

$$\alpha_{\mathrm{UCB}}(\boldsymbol{x};\beta,\boldsymbol{\theta}) = \mu(\boldsymbol{x};\boldsymbol{\theta}) + \beta\sigma(\boldsymbol{x};\boldsymbol{\theta}) \qquad \mathrm{UCB} \qquad (2.8)$$

$$\alpha_{\mathrm{EI}}(\boldsymbol{x};\boldsymbol{\theta}) = \mathbb{E}_{p(f(\boldsymbol{x})|\boldsymbol{x},\boldsymbol{\mathcal{D}}_n,\boldsymbol{\theta})}\left[\max\left(0, f(\boldsymbol{x}) - f'\right)\right] \qquad \mathrm{EI} \qquad (2.9)$$

$$\alpha_{\mathrm{PI}}(\boldsymbol{x};\boldsymbol{\theta}) = \mathbb{P}\left[f(\boldsymbol{x}) > f' \mid \boldsymbol{x}, \boldsymbol{\mathcal{D}}_n, \boldsymbol{\theta}\right] \qquad \mathrm{PI} \qquad (2.10)$$

The performance of the acquisition functions is sensitive to the correct hyperparameter specifications. In Cowen-Rivers et al. (2022) noise is added to the acquisition function to make it more robust, but the amount of noise needs to be picked and this is not linked directly to our uncertainty over hyperparameters.

2.4.2 Information-theoretic policies

An alternative to the above policies is information-theoretic ones. For instance, in predictive entropy search (PES, Hernández-Lobato et al. (2014)) the next sample is chosen to be the one that most reduces the uncertainty of the global maximiser $\boldsymbol{x}^*$. This is given in Eq. (2.11), where H is the differential entropy.

$$\alpha_{\mathrm{PES}}(\boldsymbol{x}) = \mathrm{H}[p(\boldsymbol{x}^*|\boldsymbol{\mathcal{D}}_n)] - \mathbb{E}_{p(y|\boldsymbol{\mathcal{D}}_n,\boldsymbol{x})}\Big[\mathrm{H}\big[p(\boldsymbol{x}^*|\boldsymbol{\mathcal{D}}_n \cup (\boldsymbol{x}, y))\big]\Big] \tag{2.11}$$

Other information-theoretic acquisition functions include entropy search (Hennig & Schuler, 2012), max-value entropy search (Wang & Jegelka, 2017) and joint entropy search (Hvarfner, Hutter, & Nardi, 2022). Although conceptually neat, they require hard to implement approximations, and have not become as popular as the policies based on improvement or optimism. We initially experimented with PES, but did not obtain better results than when using UCB. In the results presented in the thesis we use UCB and EI.

2.4.3 Self-adjusting acquisition functions

The acquisition functions in BO are usually derived assuming that the hyperparameters are good fits for the model. But when we start out this is not necessarily the case. We commonly have at least three hyperparameters to fit, where the data can be explained in different ways as either noisy or quickly varying. Two acquisition functions have been proposed recently to account for this. Starting from expected improvement, in Benjamins, Raponi, Jankovic, Doerr, and Lindauer (2023) the balance between the exploration and exploitation terms is adjusted when the current setting stops showing improvement. Hvarfner, Hellsten, Hutter, and Nardi (2023) base their acquisition function on an information-theoretic approach, but extend it to include the uncertainty from hyperparameter settings as well. We do not use these acquisition functions, but they represent complementary approaches to getting Bayesian optimisation to work on practical problems. In particular, in Hvarfner et al. (2023) the acquisition function rewards samples that are informative regarding the model hyperparameters.

2.5 Model misspecification

Gaussian processes are powerful models that produce uncertainty as well as mean estimates. Based on these estimates, principled acquisition functions can be derived that balance exploration and exploitation. However, both the uncertainty estimates and the reliability of the acquisition functions are dependent on the model being correctly specified. This is true of the choice of covariance function, but also of the hyperparameters. The simple way of using the maximum likelihood estimator risks underestimating the uncertainty and therefore disregarding useful regions, and incorrect priors can mislead the hyperparameter selection.

Within the context of hyperparameter optimisation, Cowen-Rivers et al. (2022) do a thorough analysis of a set of modelling assumptions, and their impact on optimisation performance. Though they do not consider the problem of hyperparameter misspecification, their analysis of noise homoskedacity and non-stationarity demonstrate general pitfalls of model misspecification. Hvarfner et al. (2023) demonstrate the effect of different hyperparameter priors on a synthetic problem, showing that the optimisation is faster when using the correct hyperparameters.

The effect of misspecification was shown in Fig. 2.4 and is illustrated further in Figs. 2.5 and 2.6. To produce these figures a function was drawn from a GP with an RBF kernel, a lengthscale of 1, a signal variance of 1, and negligible noise. Then expected improvement was applied, using three different models. In green the true lengthscale (1) was used. In yellow (short) a lengthscale of 0.25 was used and in red (long) a lengthscale of 4 was used. The signal variances were all set to the true value of 1. As can be seen, the model misspecification greatly hinders the optimisation. When setting the lengthscale too long the model is overconfident, and only explores near the highest observed value. When setting it too short the model is underconfident. It spends all ten observations exploring around the highest initial observation.

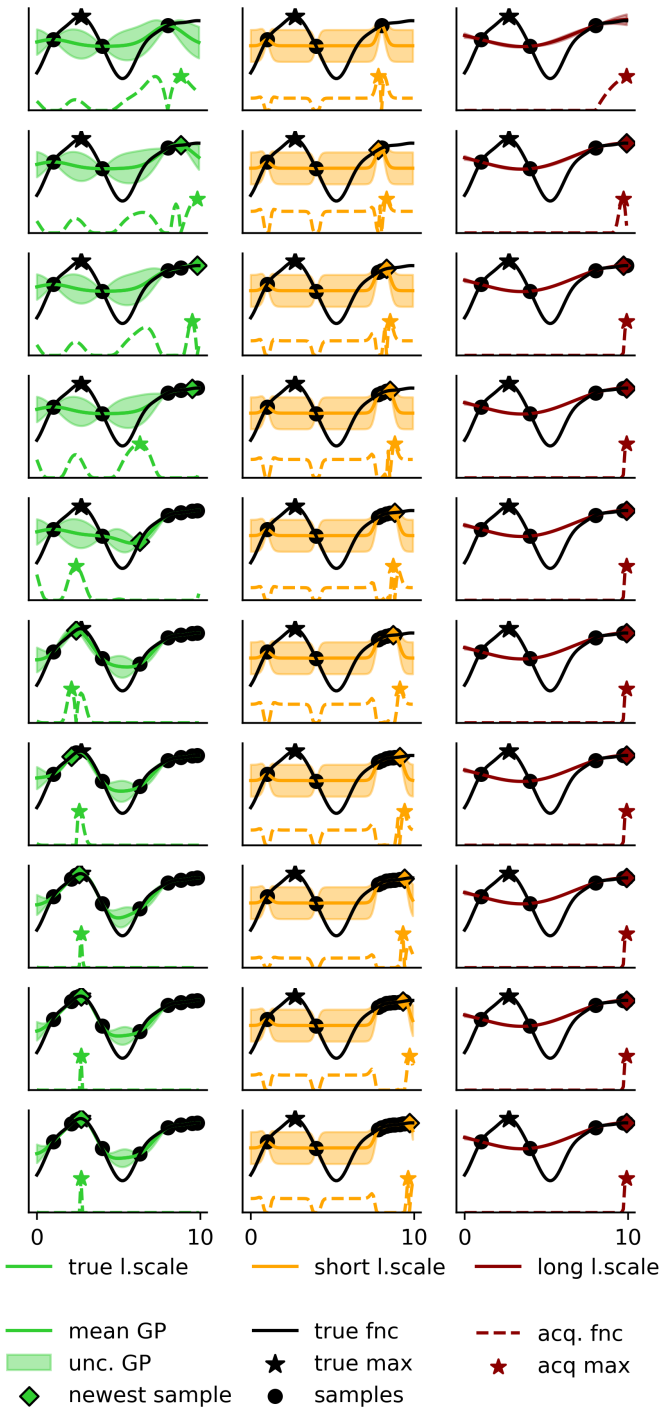


Figure 2.5: 10 steps of Bayesian optimisation on a synthetic problem using EI and models with the true (green), too short (yellow) and too long (red) lengthscales. The uncertainty bound is the model's mean $\pm$ one standard deviation.

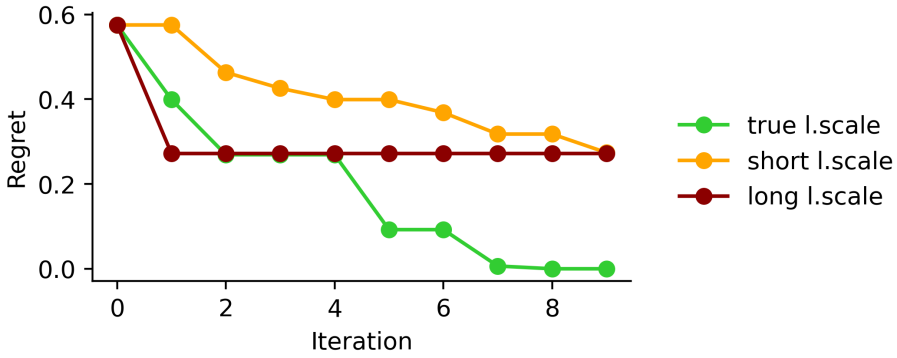


Figure 2.6: Regrets (true maximum - estimated maximum) for the optimisation in Fig. 2.5.

2.6 Transfer learning

Often, even Bayesian optimisation requires many input evaluations before it finds a good input value, due to the size of the search space. As in other machine learning paradigms (Weiss, Khoshgoftaar, & Wang, 2016), a popular solution is to use related data. This can be samples from previous optimisation tasks, but can also be unguided sets of observations. In the context of Bayesian optimisation, both the term transfer learning and the term meta learning (Vanschoren, 2019) are used to refer to using related data to reduce the number of evaluations needed of a new task.

In this context, we modify Eq. (2.1) to

$$\boldsymbol{x}_i^* = \arg\max_{\boldsymbol{x}} f_i(\boldsymbol{x}) \tag{2.12}$$

where we use a model $g(\boldsymbol{x}; \boldsymbol{\theta}_i)$ to approximate $f_i(\boldsymbol{x})$. We can query the function, collecting samples $\{\boldsymbol{x}_{i,j}, y_{i,j}\}_{j=1}^{N_i}$ where N_i is the number of samples for task i. We assume that we already have solved I similar optimisation problems, so we have data $\{\{\boldsymbol{x}_{i,j}, y_{i,j}\}_{j=1}^{N_i}\}_{i=1}^{I}$

Mostly, transfer learning for Bayesian optimisation has consisted of *direct transfer*. That is, we want to learn good values of $\boldsymbol{x}$ for task $I+1$. To do so, we rely on the strong assumption that good values of $\boldsymbol{x}$ on the new task were also good on (at least some of) the previous tasks. Fig. 2.7 shows three examples of when this assumption holds. The shifted and scaled lines are simple examples of this where

$f_{I+1}(x) = f_I(x - \delta)$ or $f_{I+1}(x) = cf_I(x)$. The final example, 'joint optima' also satisfies this assumption, even though the change from $f_I(x)$ to $f_{I+1}(x)$ is greater. However, in general this assumption does not hold, as we saw in the example on making jams from different types of fruit in Section 1.3.

Instead of learning input values $\boldsymbol{x}$, one could learn the hyperparameters $\boldsymbol{\theta}$ in the model $g(\boldsymbol{x}; \boldsymbol{\theta})$. We call this *prior transfer*. One could also combine the two approaches. Previous tasks can also be described as transfer tasks or base tasks, and the current task as the target task. A review of transfer learning for Bayesian optimisation is provided in Bai et al. (2023). A shorter summary can also be found as part of Feurer, Letham, Hutter, and Bakshy (2022). The summary below differs from these in that we use the top-level classifications of direct and prior transfer.

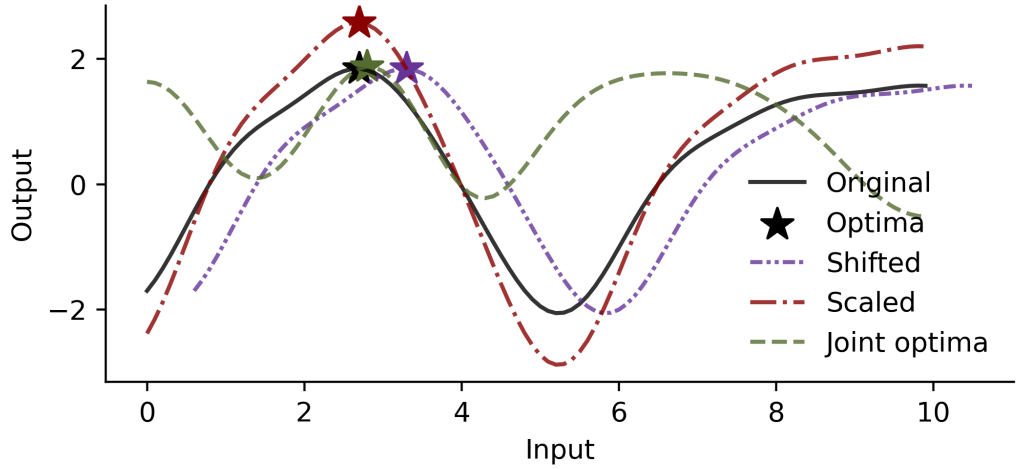


Figure 2.7: Example of changes between original and new task that are suitable to direct transfer.

Figs. 2.8 and 2.9 illustrate these two approaches, and the situations in which they do well and the ones in which they do not. For both, the top subfigure shows our starting point: we have many observations of the training function, and two observations of the test function. Based on these two observations and the many observations of the training function we now have to choose the next input value to evaluate the test function on. In the bottom left subfigures we follow a strategy from direct transfer: we evaluate the input value that worked best on the training function. In the bottom right subfigures we instead assume the test function is a shifted copy of the training

function. So the optima will not be in the same place, but the shape is the same. So for this we try to shift our training observations to line them up with the test observations, and then evaluate the test function on the shifted optimal input value from the training function.

As can be seen in the figures, we need our modelling assumptions to line up with the problem we are optimising. In Fig. 2.8 the direct approach works best, because the functions have very similar optima. In Fig. 2.9 it does not work at all. Similarly, the prior transfer approach only works well on the problem in Fig. 2.9.

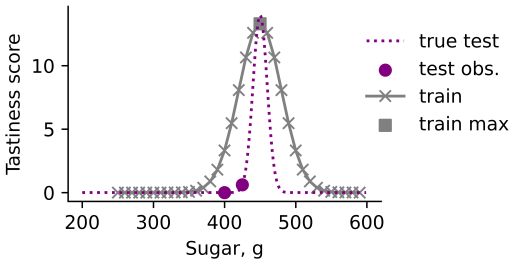
> Note that Figs. 2.8 and 2.9 are simplifications. For prior transfer we do not assume we can learn the shape itself, just the hyperparameters of the surrogate model. So we would not be lining them up and shifting the optimal value, but we would be using the hyperparameters to decide how large an area we can discard around low values.

Previous work on Bayesian optimisation transfer learning has focused on the application of hyperparameter optimisation for machine learning models. A contribution of this thesis is to extend this to other applications, in our case air pollution monitoring. We next describe direct and prior transfer in more detail. The differences between our new methods and existing methods are discussed in Chapters 5 and 6.

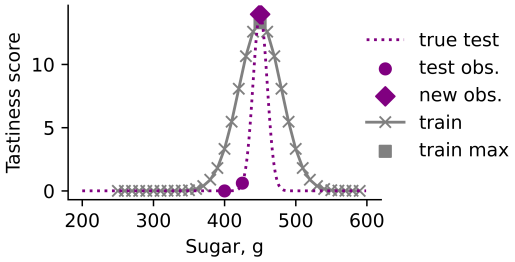
2.6.1 Direct transfer

The main approach to transfer learning for Bayesian optimisation has been through direct transfer, where we assume that good features $\boldsymbol{x}$ for $f_i(\boldsymbol{x})$, $i \leq I$ will also be good for $f_{I+1}(\boldsymbol{x})$.

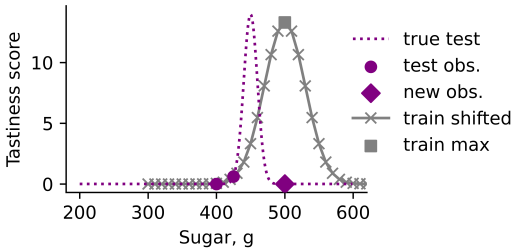
One common approach is to use good points from previous tasks to choose the S **initial points** to sample, before continuing with single-task Bayesian optimisation. The method we develop in Chapter 6, SimpleOrdered, is a version of this. In Feurer et al. (2015) the S points are chosen as the maximisers of the most related tasks, where the distance measure comes from meta-features. Their application is HPO, and they use meta-features such as the number of classes and number of features of the data set the machine learning method is applied to. Wistuba, Schilling, and Schmidt-Thieme (2015b) take the idea of initial points one step further by using the entire



(a) Problem set-up.

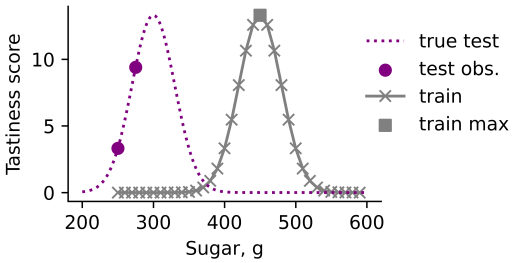


(b) Next observation using direct transfer.

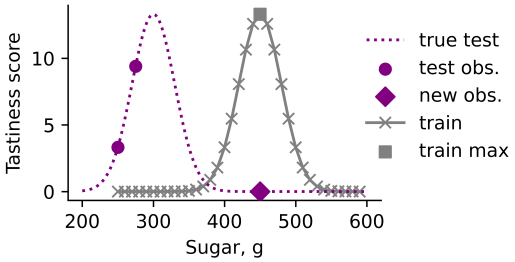


(c) Next observation using prior transfer.

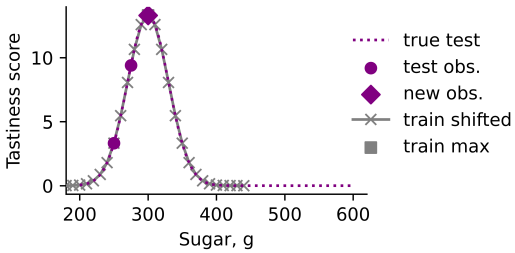
Figure 2.8: Transfer set-up in traditional scenario where the optimal input features are similar. Here direct transfer works well, while prior transfer works poorly.



(a) Problem set-up.



(b) Next observation using direct transfer.



(c) Next observation using prior transfer.

Figure 2.9: Transfer set-up in scenario where the response shapes are similar. Here prior transfer works well, while direct transfer works poorly.

optimisation budget on evaluating features that perform well on previous tasks, and so do not use a surrogate model at all. Wistuba and Grabocka (2021) avoid the need for meta-features, and instead learn a set S using an evolutionary algorithm evaluated on the previous tasks.

Alternatively, one can learn a **joint model**, for instance a GP (Swersky, Snoek, & Adams, 2013), where the kernel learns the correlations between tasks, or the observations from previous tasks are considered noisy observations of the new task (Shilton, Gupta, Rana, & Venkatesh, 2017), or relevant previous tasks are identified using data set meta-features (Yogatama & Mann, 2014). The downside of this approach is that it scales poorly, as it is cubic in the total number of observations. The approach in Swersky et al. (2013) has also been applied for Bayesian optimisation outside of HPO, namely for optimising chemical reactions (Taylor et al., 2023).

An alternative to a joint GP model is an **ensemble model**: a separate model is learned for each previous task, before being combined. Different approaches vary in the models used and in how they combine the models into an ensemble. Feurer, Letham, and Bakshy (2018) and Wistuba, Schilling, and Schmidt-Thieme (2018) present two such approaches where each task has a separate GP, which are combined using ensemble weights based on similarities between the previous tasks and the new tasks. These similarities can be based on meta-features or the relative rankings of input values.

A different approach is pursued in Perrone, Shen, Seeger, Archambeau, and Jenatton (2019), where previous tasks are used to **reduce the search space**. In each dimension the space is reduced to that between the minimum and maximum value among the optima observed on previous tasks.

Yet another approach is to use a **transfer acquisition function**. Wistuba et al. (2018) present a version of this related to ensemble models. But instead of combining the models into an ensemble first and then applying a standard acquisition function, the idea is to combine the various models within the acquisition function. Feurer et al. (2022) further explore the use of transfer acquisition functions. Volpp et al. (2020) instead learn the acquisition function through reinforcement learning. They extend the input, so that in addition to the mean and variance estimate it also takes the input

features themselves, the current iteration index and the total number of iterations. The motivation is that this approach can learn to adapt to the observations seen, and to the amount of optimisation budget remaining. For instance, it could choose to be more exploratory in the beginning. The method can scale to many source tasks.[6]

Van Rijn and Hutter (2018) present a method for learning distributions of good input values, but do not combine it with Bayesian optimisation. Van Rijn and Hutter refer to their method as learning priors, but they mean priors on the input values, not on the GP hyperparameters. This is therefore also direct transfer.

We use direct transfer in Chapter 6. We identify a subset of transfer set-ups where we have an explicit ordering, e.g. because we are regularly retuning the hyperparameters of a model as we increase the data set. We develop an ordered version of the initial points approach to show the benefit of this ordering.

2.6.2 Prior transfer

We now consider situations in which the best settings on previous tasks are not the best settings on new ones. In these situations we cannot use direct transfer. This can occur for related tasks within HPO (Swersky et al., 2013; Wistuba & Grabocka, 2021), but occurs more frequently when the tasks are more different. For instance, direct transfer is not suitable when locating the pollution maxima in different cities.

Related data can be used to **learn hyperparameter priors** for the GP. In Müller et al. (2023) a validation set is combined with random sampling to choose the best prior parameter values. Wang et al. (2022) can be seen as a special case of learning hyperparameter priors, but where we only learn one set of hyperparameters across all tasks, which corresponds to learning Dirac delta priors. Fan, Han, and Wang (2022) generalise Wang et al. (2022) to variable-size search spaces. They again assume that the hyperparameters for each search space are the same, but learn a prior to generalise between sizes of search space.

Alternatively, the previous tasks can be used for **kernel learning**. In Perrone, Jenatton, Seeger, and Archambeau (2018) a linear kernel is learned from previous tasks as well as the current observations. Wistuba and Grabocka (2021) use a neural network to learn a feature transformation which can be used with any GP kernel. As

6. However, the method was examined further in Wistuba and Grabocka (2021, Appendix B), who show that the method does not scale well with the optimisation budget. For a budget of 50 iterations it did not outperform a random search.

a preprocessing step they learn the neural network based on data from related tasks, and they then fine-tune the neural network within the BO loop. Their approach means that they only learn one set of hyperparameters and feature transformations, i.e. they come up with a point estimate, not a distribution.

Wang, Kim, and Kaelbling (2018) assume the same GP hyperparameters across the previous and new tasks, and try to **infer the mean and covariance functions using empirical estimates**. An advantage is that the method side-steps the need to choose a covariance function (e.g. the RBF kernel). But there are significant limitations in that the mean and covariance functions are estimated once and do not change as we collect more data, and that we require all the tasks to share Gaussian process hyperparameters. Additionally, the method is designed for large numbers of previous problems: 100 for a synthetic function and 1500 – 1800 for two robot-planning problems.

In prior transfer we can also use transfer learning methods developed for Gaussian processes. Corani, Benavoli, and Zaffalon (2021) developed a similar method to PLeBO for prior transfer, transferring between GPs for time series forecasting. Earlier, the assumption of shared GP kernels (Wang et al., 2022) had been used to learn GP hyperparameters in Lawrence and Platt (2004) and GP kernels in Schwaighofer, Tresp, and Yu (2004).

Volpp et al. (2020) (see Section 2.6.1) also apply their reinforcement-learning **acquisition function** approach to a transfer learning scenario where the only structure was in the lengthscales. Their method performs similarly to expected improvement without transfer.

The method we develop and explore in Chapters 4 and 5 fits into this category of transfer learning. We use data from related tasks to learn a prior for the hyperparameters, represented through a set of samples. Within the Bayesian optimisation loop we then weight the samples by their likelihood on the evaluations of the new task. Learning hyperparameter priors is related to Corani et al. (2021); Fan et al. (2022); Wang et al. (2022). Our method is also related to transfer acquisition functions (Wistuba et al., 2018).

2.6.3 Applications of transfer Bayesian optimisation

As previously noted, most work on transfer BO has focused on hyperparameter optimisation for machine learning models, as well as hyperparameter optimisation of related computing systems (Bai et al., 2023). It has also been used for robotic task and motion planning (Wang et al., 2018). An exception is Taylor et al. (2023) who use multi-task BO for optimising chemical reactions. This focus on HPO is a natural result of the predominance of HPO as a Bayesian optimisation application. But it leaves a gap in our understanding regarding how well these methods work on other applications. Some of the approaches, such as reusing well-performing features, can be easily applied to a new application domain. But others, e.g. Feurer et al. (2015), rely on meta-features of the data set used to train the ML model. This is not directly applicable to other applications such as air pollution monitoring — new meta-features would have to be developed.

2.7 Alternatives to Bayesian optimisation

This thesis only considers Bayesian optimisation, but there are other ways of approaching optimisation and data acquisition. This section discusses three related methods — active learning, particle swarm optimisation and reinforcement learning.

2.7.1 Active learning

Within machine learning, active learning (Settles, 2012) refers to methods that choose what labels to collect next, as opposed to the standard set-up of a fixed set of training data in supervised learning. The motivation is that labelling data can be expensive, and that we will need fewer labels if we choose which inputs to label in an intelligent manner. Active learning can also be referred to as query learning. Active learning encompasses both regression and classification methods (Settles, 2012, pp. 3–5).

Some active learning methods are closely related to Bayesian optimisation (Murphy, 2023, p. 1127). The difference is that in active learning we want to approximate the function throughout its domain, while in Bayesian optimisation we just want to find its maximum. There are many possible approaches to active learning, but it can be done very similarly to Bayesian optimisation: choosing a Gaussian process model, fitting it to the observations and deciding where to sample next. For instance, we can use

the UCB acquisition function, Eq. (2.8), for Bayesian optimisation, while for active learning we could use just the uncertainty term: $\alpha(\boldsymbol{x}) = \sigma(\boldsymbol{x})$, a version of uncertainty sampling (Murphy, 2023, p. 1128). Ideas can be shared between active learning and Bayesian optimisation, as in Hvarfner et al. (2023) where ideas from active learning are used to include hyperparameter selection in the acquisition function.

There are examples of transfer learning for active learning as well. In appendix G of Balandat et al. (2020), they fit a Gaussian process to historical data of parasite rates and use an acquisition function to determine the most informative locations to sample. In Lange, Cole, Van Horn, and Mac Aodha (2023) active learning is used for species range estimation. They use a set of models representing the range of species from the training data, and model the range of the new species as a weighted combination of these models. This is conceptually similar to what we do in Chapters 4 and 5 by learning candidate model hyperparameters, but Lange et al. assume access to much more data and use linear models.

Active learning is closely related to **experimental design** (Murphy, 2022, p. 648; Valentin et al., 2023), and active learning can be seen as a special case of experimental design (Rainforth, Foster, Ivanova, & Smith, 2023). In active learning we are trying to learn a mapping $y = f(\boldsymbol{x})$. In experimental design we could learn this mapping, or learn other properties of the problem, e.g. a hyperparameter θ of our model.

2.7.2 Particle swarm optimisation

Particle swarm optimisation (PSO) is an optimisation approach inspired by social animals (Marini & Walczak, 2015). In PSO a set of particles jointly move stochastically through the optimisation space, interacting with one another. The particle movements are impacted by the individual and joint best observations, but there is no modelling of the underlying function as in BO.

The movement of the particles between evaluations is determined by (Marini & Walczak, 2015)

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t+1) \tag{2.13}$$

$$\boldsymbol{v}_i(t+1) = \boldsymbol{v}_i(t) + c_1(\boldsymbol{p}_i - \boldsymbol{x}_i(t))\boldsymbol{R}_1 + c_2(\boldsymbol{g} - \boldsymbol{x}_i(t))\boldsymbol{R}_2 \tag{2.14}$$

Here i is the particle index, $\boldsymbol{x}_i$ is the feature to evaluate, $\boldsymbol{v}_i$ is the particle velocity, $\boldsymbol{p}_i$ is the best observation of particle i and $\boldsymbol{g}$ is the best observation overall. c_1 and c_2 are constants and $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ are diagonal matrices of random numbers generated uniformly on [0,1].

The main differences to Bayesian optimisation are that:

- Many evaluations are done in parallel.[7]
- There is no model of the underlying function.
- The particles follow trajectories keeping evaluations close to each other between iterations.

This means that PSO is more suited to problems with cheaper evaluations, especially where there is a movement cost between evaluations.

2.7.3 Reinforcement learning

Another class of machine learning methods that interact with the problem is reinforcement learning. There are two key differences between Bayesian optimisation and reinforcement learning in how the problem is considered (Murphy, 2022, pp. 17–19). Firstly, reinforcement learning assumes a reward signal, not labels for each input value. That means that we do not get immediate feedback on how good our choice of input was, but instead an aggregate response to multiple past actions. For instance, reinforcement learning can be used to teach a model to play chess (Silver et al., 2018). We do not get feedback on every move, but instead find out at the end whether we won or lost the game.

Secondly, instead of freely choosing the next input value, reinforcement learning deals with context. Continuing the chess example, it needs to choose the best move *given the current board set-up*. There are extensions to Bayesian optimisation that handle contextual information, for instance when using it for path planning (Folch et al., 2022; Marchant & Ramos, 2014), but it is more central to reinforcement learning. As previously mentioned, reinforcement learning can also be used to replace a part within the Bayesian optimisation framework (Volpp et al., 2020).

---

7. While there are extensions to standard Bayesian optimisation that allow batched evaluations, e.g. González, Dai, Hennig, and Lawrence (2016), these are typically for relatively small numbers, while 50 is considered a reasonable choice in PSO (Marini & Walczak, 2015).

Contextual bandits can be seen as a simple version of reinforcement learning (Bouneffouf, Rish, & Aggarwal, 2020), in which the reward is instantaneous, and the context is not dependent on previous actions (Murphy, 2023, pp. 1109–1100). While for chess our current context is determined by our — and our opponent's — previous moves, an example of a contextual bandit problem is a recommendation system where we have some user attributes which we use to come up with a recommendation (Bouneffouf et al., 2020).

2.8 Libraries

The thesis has benefited from the following key Python packages:

- **Autograd:** Automatic differentiation of NumPy code (Autograd, since 2015). Used for calculating derivatives for fitting hyperparameters in Chapter 4.
- **BoTorch:** Bayesian optimisation implementations (Balandat et al., 2020). Used for Bayesian optimisation implementations in this chapter and Chapters 3 and 6 as well as a baseline in Chapter 5.
- **GPy:** Gaussian process library (GPy, since 2012). Used for verifying GP implementations, in this chapter and in Chapter 5.
- **Matplotlib:** Library for plotting (Hunter, 2007). Used throughout the thesis for generating plots.
- **NumPy:** Library for fast computation (Harris et al., 2020). Used throughout the thesis.
- **NumPyro:** Library for probabilistic programming (Phan, Pradhan, & Jankowiak, 2019). Used for computing MCMC prior samples in Chapter 5.
- **SciPy:** Algorithms for scientific computing (Virtanen et al., 2020). Used throughout the thesis, mostly for calculating probability distributions and optimisation.
- **Syne Tune:** Library for hyperparameter optimisation (Salinas et al., 2022). Used to run experiments in Chapter 6.

3

Bayesian Optimisation Against Climate Change

What are men to rocks and mountains?

– Jane Austen, *Pride and Prejudice*, 1813

This chapter is based on the paper Bayesian Optimisation Against Climate Change: Applications and Benchmarks*, presented at the Data-centric Machine Learning Research Workshop at ICML 2023 (Hellan, Lucas, & Goddard, 2023a). It is a survey of Bayesian optimisation and related methods applied to problems connected to climate change.*

3.1 Introduction

If what we care about is the real-world impact of Bayesian optimisation — for instance how much more renewable electricity can be generated because of it — then we should not focus solely on developing new methods and evaluating them on synthetic problems. We should also survey and encourage work on applying the methods to real data, and make data sets and benchmarks available for this purpose. By making these data sets available we encourage the methods to be tested on more realistic data, and so steer development towards techniques that work in practice. And to the extent that new methods are needed, they will be motivated by the needs of the applications, for instance for handling contexts and large numbers of samples, or learning useful priors.

These were the motivations for the short survey presented in this chapter, conducted primarily in July and August 2022. It surveys applications to climate-related problems of Bayesian optimisation and related methods such as Bayesian experimental design. We understand climate-related problems as a broad group of problems linked to

understanding and mitigating climate change, so include among others climate modelling, environmental monitoring and renewable energy technology. Additionally, the problems had to be relevant to the methods, so they are overwhelmingly optimisation problems of one type or another: for instance optimising a desirable quantity such as renewable electricity directly, or improving forecasting modelling of electricity demand. We identified four main clusters of work: material discovery, wind farm layout, optimal renewable control and environmental monitoring. For the first three we identified benchmarks from the literature. For environmental monitoring we found no suitable benchmark in the literature, so we cleaned up and shared the code used in Chapter 4 to generate optimisation tasks from London air pollution data (Imperial College London, 1993). The code can be used to reproduce these tasks, or easily modified to generate tasks using data from other pollutants, site types or years. So it can prepare a large number of optimisation tasks using real data. We also checked that the benchmarks were still usable, and not, for instance, relying on broken dependencies. This is described in more detail in Section 3.5.

The survey is intended to fulfil two functions. Firstly, it is meant as a reference for existing work applying Bayesian optimisation to climate-related problems. As part of this, it clusters the work by application domains, and explains both how the application domain relates to climate change, and how Bayesian optimisation can be applied. The second function is as a starting point for people with expertise in Bayesian optimisation who want to apply it to climate-related problems. By highlighting four benchmarks to get started on, and discussing their differences, it should be much faster to start a project. We envisage the survey as a useful starting point for MSc projects or for a new PhD student.

The survey is built around benchmarks. Benchmarks drive research forward for multiple reasons. Firstly, they make research much faster, as finding, choosing and then preparing data for machine learning is time-consuming. This was experienced in this thesis: multiple air pollution data sets were considered before we settled on the Satellite (Copernicus Sentinel-5P (processed by ESA), 2018) and LAQN (Imperial College London, 1993) data sets. We wanted a data set that combined the following properties:

- Available for research and publishing
- High-quality measurements, or quantifiable measurement errors
- Enough sample locations that we could meaningfully optimise, and enough sets of samples that we could run replications

- Dense enough coverage that we could optimise within a city

Two data sets we considered had to be discarded because the researchers collecting the data sets were not ready to share them. A data set of low-cost sensors was discarded, because we could not easily quantify the measurement error. Additionally, public data from multiple cities, e.g. Edinburgh, were not suitable because there were not enough separate sensors to create a reasonable search space. The satellite data set combined the first three desirable properties, but was on a much coarser scale than we preferred, with each sample covering multiple kilometres. Therefore we also used the London data, which was higher-resolution, even though there were many fewer sample locations. Because of the amount of work needed to identify these data sets, we made our preprocessing of the LAQN data available as part of the survey.

Benchmarks also make research findings more comparable. If all methods are evaluated on the same benchmark then results can be compared directly. An alternative would be for researchers to implement competing methods as baselines, but that is time-consuming. And often these implementations have hyperparameters or other implementation choices that impact the performance. Tuning these is also time-consuming, and researchers are not incentivised to spend their time improving the baselines' performance. Finally, benchmarks also make research findings more reliable. For established benchmarks it is easier to recognise weaknesses such as cherry-picking results. And it is easier for others to rerun the experiments in order to reproduce the results.

This chapter is thematically linked to Chapter 4, as both focus on the application of Bayesian optimisation to problems related to climate change and the environment. But the focus of this chapter is slightly different from the rest of the thesis, in that it does not focus on transferring knowledge between optimisation tasks. This is partly a matter of the chapter focusing on *problems*, while transferring knowledge is one of several *solutions*. Using related problems to determine informative priors is useful in any optimisation problem, but requires related data sets or domain knowledge. In new domains, such as those discussed in the paper, data are naturally scarce. And while more data sets exist, many of them will be proprietary. A manufacturing company might not want their competitors to know how to make the most efficient solar cells, or what material combinations to avoid. While for the simulation problems it would be possible to generate priors by evaluating related optimisation tasks, it is

more efficient to use the computational resources on the problem directly: if we have enough computational budget to evaluate 50 wind farm layouts, we should use the entire budget on the area we are planning, instead of evaluating 10 layouts in an area with a different wind direction.

Over time, however, we are likely to plan wind farms in multiple areas. If we keep our evaluations, we can use these to learn priors and reduce the number of evaluations needed to plan later areas. But such methodological explorations were outside the scope of the paper, the purpose of which was to survey existing work. These extensions would be beneficial, but should be done with input from domain experts in wind farm planning and optimal control. The different simulation set-ups should correspond to the differences we would see when evaluating different wind farms and solar panels in the real world. Otherwise we risk having the prior be artificially close to the problem, contributing to unrealistic expectations of how well the methods can perform.

3.2 A note on chapter ordering

The survey presents LAQN-BO as a new benchmark. Chronologically, the survey comes after Chapter 4, which introduces the data set. But thematically the survey fits in after the background section, as it is a survey of existing work, and not an experimental paper. The code for producing the optimisation tasks is based on that used in Chapter 4, but was cleaned up and documented to enable reuse.

3.3 Author contributions

The survey was written and researched by me, with input and feedback from Lucas and Goddard. The code was written by me.

3.4 The paper

arXiv:2306.04343v1 [cs.LG] 7 Jun 2023

Bayesian Optimisation Against Climate Change: Applications and Benchmarks

Sigrid Passano Hellan

Christopher G. Lucas

Nigel H. Goddard

School of Informatics
University of Edinburgh

Abstract

Bayesian optimisation is a powerful method for optimising black-box functions, popular in settings where the true function is expensive to evaluate and no gradient information is available. Bayesian optimisation can improve responses to many optimisation problems within climate change for which simulator models are unavailable or expensive to sample from. While there have been several feasibility demonstrations of Bayesian optimisation in climate-related applications, there has been no unifying review of applications and benchmarks. We provide such a review here, to encourage the use of Bayesian optimisation in important and well-suited application domains. We identify four main application domains: material discovery, wind farm layout, optimal renewable control and environmental monitoring. For each domain we identify a public benchmark or data set that is easy to use and evaluate systems against, while being representative of real-world problems. Due to the lack of a suitable benchmark for environmental monitoring, we propose LAQN-BO, based on air pollution data. Our contributions are: a) identifying a representative range of benchmarks, providing example code where necessary; b) introducing a new benchmark, LAQN-BO; and c) promoting a wider use of climate change applications among Bayesian optimisation practitioners.

1 Introduction

The use of machine learning (ML) to tackle climate change is gaining traction, including ML-wide surveys of relevant applications (Rolnick et al., 2022; Donti & Kolter, 2021). To facilitate technical innovation and its adoption into practice, it is important to complement these reviews with concise summaries of problems that specific ML frameworks are well-matched to, along with example data sets. Here, we provide a summary, paired with data sets, for Bayesian optimisation. This is important for three reasons: a) it reduces start-up costs and overhead, as the time-consuming task of identifying suitable data sets does not have to be repeated; b) it helps researchers focus on problems that are timely and/or important, and have a structure that is suitable for the framework(s) they are using; and c) it makes it easier to compare methods, as numerical results can be compared directly. The benchmarks should be similar to real-world applications, because that provides more realistic expectations of performance. It also shows which versions of Bayesian optimisation are most likely to perform well, for instance in terms of choice of acquisition function and underlying model. More realistic benchmarks also let us identify challenges which still need to be solved, for instance in terms of generating useful priors from related data.

Bayesian optimisation (BO), is a machine-learning approach to black-box optimisation (Shahriari et al., 2015). It is particularly suited to problems that:

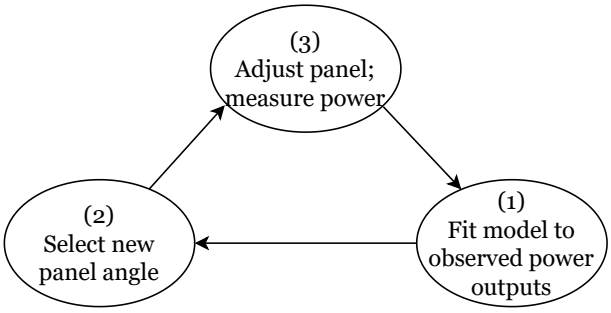


Figure 1: Application of Bayesian optimisation to optimising the power generation of a solar panel by adjusting the panel angle.

a. Have a complex unknown structure which can only be efficiently modelled through a surrogate model, e.g. the conductivity of solar panel materials.
b. Are expensive or slow to evaluate, requiring sample efficiency from trying out fewer bad or uninformative choices, e.g. when discovering new materials that must be synthesised in order to be evaluated.
c. Require identifying extrema, either for optimisation, e.g. maximising power generation, or for intervention, e.g. emission monitoring.

An additional benefit of Bayesian optimisation is that it explicitly models its uncertainty, which can be used to determine when to stop optimising (Makarova et al., 2022), or to trade off exploration and exploitation (De Ath et al., 2021).

Bayesian optimisation consists of fitting a surrogate model, typically a Gaussian process (Williams & Rasmussen, 2006), to observed data, and using the surrogate model to choose what input value to evaluate next. Then the surrogate model is adjusted, another input value evaluated, and so on. To choose the next input value an acquisition function is used, which balances exploration and exploitation. As an illustrative example, visualised in Fig. 1, Bayesian optimisation can be used to find the angle of a solar panel that maximises power output: (1) model the output as a function of angle based on all available data and/or domain knowledge; (2) select a promising angle to try next, e.g., an angle that is a good candidate for maximising power output or that is maximally informative about the best angle; and (3) try this angle before updating the model and returning to Step 1.

We present a summary of Bayesian optimisation as applied to climate change problems. We identify the main application domains, emphasise how they impact climate change responses, and identify beginner-friendly benchmarks for each. We found a benchmark to be lacking for environmental monitoring, and therefore provide one, the LAQN-BO benchmark, based on air pollution data (Imperial College London, 1993). The four use cases are: *material discovery* — accelerating the development of e.g. new solar panels; *wind farm layouts* — choosing where to place individual turbines; *optimal renewable control* — choosing operating parameters of solar panels or wind turbines; and *environmental monitoring* — choosing where to place sensors. Bliek (2022) is a related survey of Bayesian optimisation and other surrogate-based optimisation methods. Where Bliek takes a higher-level approach, focusing on classifying previous work by methodology, we instead group work by applications in order to identify data sets and applications for new projects. When selecting benchmarks, we have emphasised ease of adoption. So we have only considered ones that are publicly available, and well documented.

This paper is predominantly concerned with Bayesian optimisation. However, Bayesian optimisation is closely related to other methods, like Bayesian experimental design (Rainforth et al., 2023), and we have included some examples of applications of these methods as well, e.g. Tran & Ulissi (2018); Kleinegesse & Gutmann (2020). Bayesian optimisation can be used within Bayesian experimental design (Valentin et al., 2023), and can also be seen as a special case of Bayesian experimental design (Rainforth et al., 2023), especially when using information-theoretic acquisition functions. Both are powerful methods, with useful applications in the response to climate change and similar challenges, such as the choice of priors and scaling to large numbers of samples. Our focus on Bayesian optimisation does not reflect a preference of one method over the other, but rather that more work has been done applying Bayesian optimisation against climate

change than for Bayesian experimental design. Because the methods are so closely related, and because our focus is on the applications and benchmarks, not on the details of the methods used for solving them, we have opted to include some examples of related methods as well.

Bayesian optimisation is not a complete solution, but an important tool when combined with domain knowledge, and can help improve responses to climate change. Performance in real applications is the ultimate test of any ML method, including Bayesian optimisation. Simulated problems often fail to anticipate crucial challenges, e.g. in terms of finding suitable inductive biases (Hellan et al., 2022). Currently, Bayesian optimisation is often evaluated on synthetic functions or hyperparameter optimisation (HPO) benchmarks, either exclusively (AV et al., 2022; Papenmeier et al., 2022; Hvarfner et al., 2022; Thebelt et al., 2022) or mostly (Oliveira et al., 2022; Folch et al., 2022; Song et al., 2022). But HPO is just one kind of problem. There are other less used benchmarks, like robotics (Song et al., 2022; Nguyen et al., 2022) and chemistry simulations (Folch et al., 2022; Tu et al., 2022). Ramesh et al. (2022) test their method on optimising the control of an airborne wind energy system, but has not made their setup available as a benchmark. To increase the impact of Bayesian optimisation, and its prevalence, we should adopt a broader set of benchmarks. That would not only demonstrate which methods work well in practice, but open up new research problems to solve. Using benchmarks from climate change has the advantage of doing so in a dynamic field, increasing the chance of adoption, and with real impact. Adopting climate change motivated benchmarks, and using them to compare methods, will make Bayesian optimisation more relevant outside academia. We now proceed to the main applications domains and their respective benchmarks, starting with material discovery.

2 Material discovery

Material discovery means developing materials with superior combinations of properties (Frazier & Wang, 2016), e.g. more efficient solar cells. Bayesian optimisation and related methods have been suggested for optimising a wide range of materials, from concrete (Severson et al., 2021) and Direct Air Capture of CO_2 (Ortiz-Montalvo et al., 2021) to solar panel glass (Haghanifar et al., 2021), electrifying the transport and chemical industry sectors (Annevelink et al., 2022), and for electrocatalysts for CO_2 reduction and H_2 production (Tran & Ulissi, 2018; Zhong et al., 2020; Frey et al., 2022). The benefits are in reducing greenhouse gas emissions or increasing their capture; making renewable energy generation or storage more efficient; or making more sectors able to use renewable energy. If evaluating the material requires synthesising and testing it, there are huge potential gains in time and cost by only doing so on promising candidates identified by Bayesian optimisation.

Liang et al. (2021) provide three material discovery data sets (Sun et al., 2021; Bash et al., 2021; Mekki-Berrada et al., 2021) related to solar panels, with 3-5 features each and between 94 and 178 unique evaluated data points. Using Fig. 1 to illustrate the Bayesian optimisation process, the steps become: (1) model the material performance as a function of ingredient proportions used; (2) select a new combination of ingredient proportions to try; and (3) produce and evaluate the material. The data sets enable us to replace (3) with looking up the evaluations in a table.

- **Benchmark:** `https://github.com/PV-Lab/Benchmarking` (Liang et al., 2021)
- **Features:** Material properties: 3-5 dimensions
- **Data type:** CSV files
- **Sampling:** Manufacture material. In benchmark replaced by table look-up
- **Objective:** Material performance: conductivity, absorbance spectrum score or stability
- **Impact:** Better solar panels, which gives more renewable generation and lower climate gas emissions

3 Wind farm layout

Wind turbines are typically grouped together in wind farms, with several turbines in relatively close proximity, to reduce installation costs and environmental impacts. Before construction, the locations of individual wind turbines must be planned. Wind turbines work by extracting kinetic energy from the air, so the wind is weaker and more turbulent after flowing past a turbine. This leads to less power generation (Park &

Law, 2015) and greater dynamic loads and fatigue on the downwind turbines (Dong et al., 2022). This is illustrated in Fig. 2, where the two turbines need to be placed within the green area. Determining the optimal wind turbine layout is a difficult but important optimisation problem, as more renewable electricity can be generated without requiring more turbines to be installed. The efficiency of Bayesian optimisation is important for this problem, as the type of simulations used can take 15 seconds to run on one CPU even for five wind turbines (Bliek et al., 2021). Bayesian optimisation can be applied to the problem by replacing the steps in Fig. 1 with (1) modelling the collective power output as a function of the wind turbine placements; (2) selecting a new combination of locations to evaluate; and (3) running the power output simulation.

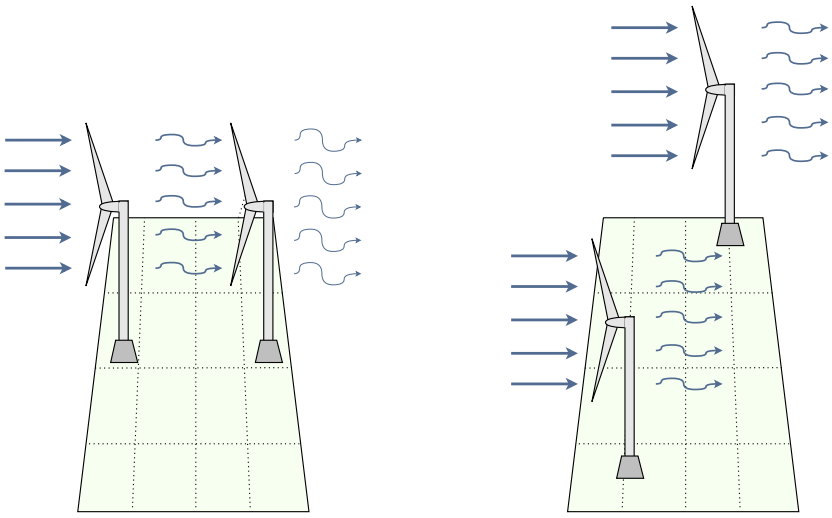


Figure 2: Planning of wind turbines within limits shown in green. The wind is shown by the blue arrows: wider arrows indicate stronger wind and more undulating arrows indicate more turbulence. In the layout on the left one turbine is directly downwind of the other, resulting in weaker and more turbulent wind. To the right, both turbines get unhindered wind.

Bliek et al.'s (2021) set of benchmarking problems and baselines include a wind farm layout problem, *windwake*, based on FLORIS (NREL, 2020) simulations. The benchmark assumes a fixed number of wind turbines; this is relaxed in Chugh & Ymeraj (2022), where the two objectives of cost and power generation are jointly optimised, using simulations based on Pedersen et al. (2019). In related pieces of work, Mern et al. (2021) build wind maps for later layout planning, and Tillmann et al. (2020) plan the layout of bifacial solar panel arrays. We use the benchmark in Bliek et al. (2021) as it has been been prepared for further use.

- **Benchmark:** `https://github.com/AlgTUDelft/ExpensiveOptimBenchmark` (Bliek et al., 2021)
- **Features:** Spatial locations: 10 dimensions. 2 for each of 5 turbines
- **Data type:** Simulation
- **Sampling:** Simulate layout
- **Objective:** Energy production
- **Impact:** More renewable generation, which reduces climate gas emissions

4 Optimal renewable control

Having planned the layout of a new renewable power generation plant, the next question is how best to operate it. For wind turbines the yaw and pitch angles need to be set (Doekemeijer et al., 2019; Park & Law, 2015; Park, 2020; Yang et al., 2022), as well as the induction factor (Park, 2020) and electric load resistance (Park & Law, 2015). For solar panels the voltage level applied (Abdelrahman et al., 2016; Lyden et al., 2018) and the angle of the panel (Abel et al., 2018) can be adjusted. By adjusting these the amount of energy generated can be maximised, which in turn reduces the amount of fossil fuel consumed for a given

amount of electricity demand. A review of different control methodologies for wind turbines is given in Dong et al. (2022). For wind farms, the optimisation should occur jointly, as the available wind energy for a downstream turbine is impacted by the operating conditions of the upstream turbine (Park & Law, 2015). Bayesian optimisation is suitable for this problem as it allows the response of the non-linear problem to be learnt through data collection. Extensive work on wind turbine control has also been done using Bayesian Ascent, a version of Bayesian optimisation which limits the changes in inputs between iterations (Park & Law, 2016a;b; Park et al., 2016; 2017; 2018). Relatedly, Moustakis et al. (2019) use Bayesian optimisation to tune the parameters of traditional control methods, and Fiducioso et al. (2019) consider the problem of controlling a HVAC system. In Mulders et al. (2020), Bayesian optimisation is used to reduce blade fatigue. Andersson & Imsland (2020) use a method related to Bayesian optimisation for wind turbine control.

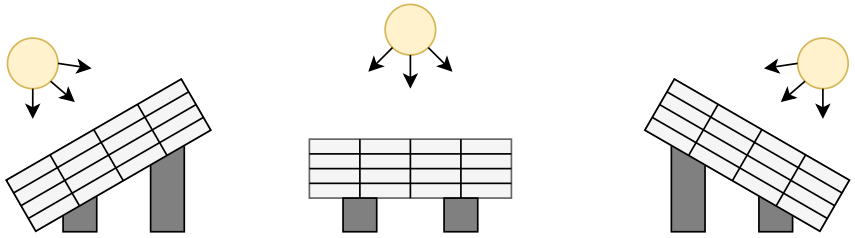


Figure 3: Adjusting solar panels to optimise power generation as the sun moves across the sky.

Abel et al. (2018) contribute a benchmark for solar panel optimisation[1]. The goal is to maximise the energy yield by adjusting the angles of the solar panel, as illustrated in Fig. 3. They evaluate contextual bandits and reinforcement learning methods, but the problem is also suitable for Bayesian optimisation. We provide a basic example of this at `https://github.com/sighellan/solar_panels_rl`[2]. Fig. 1 shows how Bayesian optimisation is applied to the problem. We model power output as a function of panel angles; decide on a new angle to try; and run the simulation to evaluate the new angle. The challenge is not to locate the sun; its location can be calculated and is given as input to the methods. Instead, the idea is to adapt to other changes like cloud cover, shading from nearby buildings and changes in the amount of reflected sunlight, for instance after snowfall. However, many solar panels are not designed to track the sun, and instead keep a fixed angle. Wind turbines have more control settings, interact in less predictable ways, and have seen more applications of Bayesian optimisation (see previous paragraph). We chose the benchmark by Abel et al. (2018) because it is well-documented and publicly available. But real-world benefits are more likely to come from applications to wind turbines than to solar panels.

- **Benchmark:** `https://github.com/david-abel/solar_panels_rl` (Abel et al., 2018)
- **Features:** Panel direction: 2 dimensions
- **Data type:** Simulation
- **Sampling:** Simulate collected sunlight
- **Objective:** Collected energy
- **Impact:** More renewable generation, which reduces climate gas emissions

5 Environmental monitoring

An important problem within environmental monitoring is that of choosing where to place the monitoring sensors. This is difficult as the planning has to be done without knowing the usefulness of each site, as that depends on future data. Bayesian optimisation and related methods are useful tools in this setting, as they build probabilistic models of the environmental characteristic of interest — e.g. air pollution — which they use to make efficient choices. If we are mainly interested in identifying the worst-hit locations, e.g. for targeting interventions or evaluating compliance with legal limits, Bayesian optimisation lets us

[1]To make it work, we used v0.784 of the simple_rl Python package (Abel, 2019).

[2]We built the example using BoTorch (Balandat et al., 2020). In addition to integrating our basic Bayesian optimisation implementation into the framework, we also provide a requirements file and port the code to Python 3.

do that efficiently. To build accurate environmental maps for an entire area we can instead use Bayesian experimental design, for instance if we want to know the water quality throughout a lake.

Monitoring is important for tracking the impacts of climate change, including oceans (Sanchez-Pi et al., 2020), for locating gas leakages (Asenov et al., 2019; Gao & Bai, 2022; Kleinegesse & Gutmann, 2020), and the related task of air pollution monitoring (Ainslie et al., 2009; Morere et al., 2017; Marchant & Ramos, 2012; Singh et al., 2010; Hellan et al., 2020; 2022; Leu & Ho, 2020). Samaniego et al. (2021) provide a benchmark for path-planning based on water quality monitoring, also used by Folch et al. (2022), but the benchmark uses synthetic data.

The LAQN-BO benchmark

We present a new benchmark for the climate change Bayesian optimisation community, using air pollution data from the London Air Quality Network (LAQN) (Imperial College London, 1993). Air pollution monitoring is related to climate change in that the sources and hence required interventions are often interlinked, e.g. car exhaust. Also, environmental changes are often sped up by climate change, making their monitoring more important. The optimisation problems in the LAQN-BO benchmark are similar to those in Hellan et al. (2022). But while that focuses on Bayesian optimisation methodology, we focus on the problems themselves. We provide simple scripts and step-by-step instructions for generating the problems and put them in the context of other climate-related Bayesian optimisation problems and benchmarks.

The optimisation objective is the NO_2 concentration, i.e. to find the location of maximum pollution from the set of available locations. This is useful as higher pollution concentrations mean greater impacts on human health. Again using Fig. 1 as a basis, we replace the steps with: (1) model the NO_2 concentration as a function of spatial location; (2) determine a new location to evaluate; and (3) measure the pollution in the new location. We use the provided benchmark to replace (3) with looking up the historical measurement. We construct a training set from the 2015 data, and a test set from the 2016 data. Each problem corresponds to data from a single day, so multiple days give multiple problems. As in Hellan et al. (2022), we filter out days when less than 40 stations collected measurements, and only use the 'Roadside' sensors, resulting in 214 training problems and 365 test problems. We log-transformed the data, and standardised it using the mean and standard deviation from the training set. Example problems are shown in Fig. 4.

The LAQN-BO benchmark has several advantages. Firstly, it uses real data. Secondly, we present a *set* of related problems, so that methods can be trained on the training set, and evaluated on the test set. Finally, the provided scripts can be easily adapted to generate new Bayesian optimisation problems using other pollutants or other years of the extensive available LAQN data. None of the other benchmarks combine all of these advantages. A disadvantage is that data is not available everywhere, only at locations with existing sensors, limiting the available search space. This corresponds to having to choose from a sparse set of locations. For instance, if we are mounting sensors to lampposts, we can only choose between the locations with lampposts.

- **Benchmark:** `https://github.com/sighellan/LAQN-BO`
- **Features:** Spatial coordinates: 2 dimensions
- **Data type:** Python class
- **Sampling:** Place pollution sensor. In benchmark replaced by table look-up
- **Objective:** NO_2 concentration
- **Impact:** Better monitoring of air pollution, enabling targeted interventions

6 Benchmark comparison

The four benchmarks identified in our survey have different characteristics, as is summarised in Table 1. They represent different key challenges for Bayesian optimisation deployments, and provide breadth for evaluating different types of Bayesian optimisation methods. The key challenge of the materials benchmark is its small size, meaning there is little opportunity to fine-tune the methods. For LAQN-BO it is that of constructing priors from the training data, as the problems have as little as 40 evaluations each, requiring very sample-efficient learning. The wind farm layout benchmark also requires efficient use of data. But

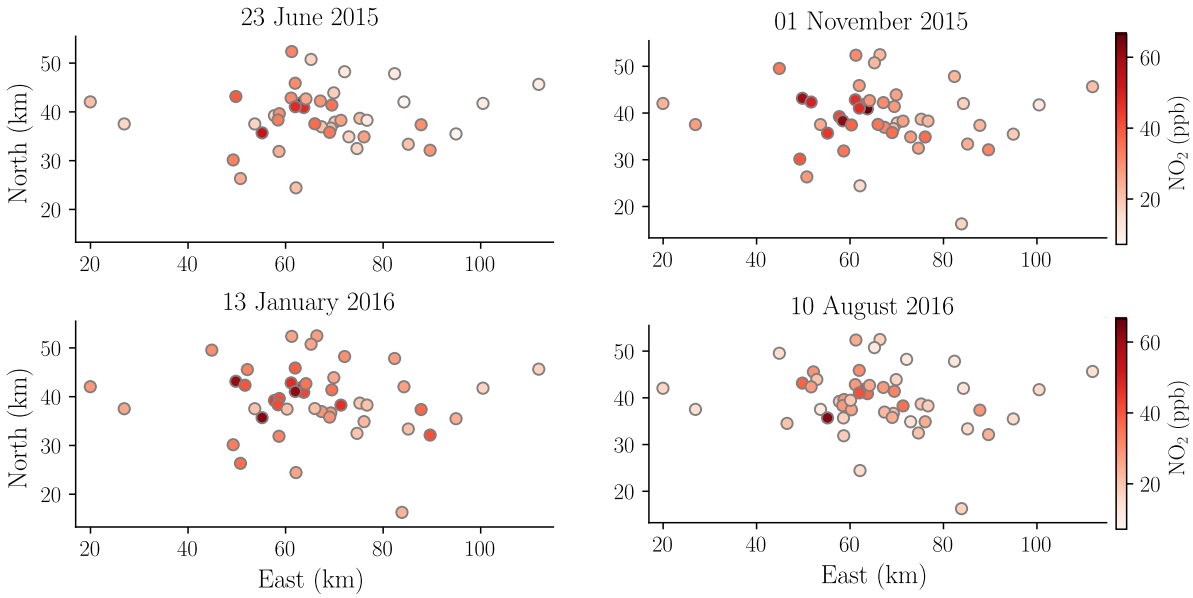


Figure 4: Example LAQN-BO problems from the training set (2015, top) and test set (2016, bottom). Not all sensors are available on every day. The overall pollution level varies, compare the top left and right plots. The locations of maxima also varies, compare the right top and bottom plots. Most of the sensors are clustered together in central London.

in contrast to LAQN-BO, it does not provide training data for learning priors. The key challenge for the renewable control benchmark is dealing with large numbers of samples, as the panel direction is optimised regularly over many days. It is therefore a good candidate for testing Bayesian optimisation built on scalable Gaussian processes (Liu et al., 2020). It also requires keeping track of context, e.g. time, as the sun moves across the sky. An additional contrast to the other benchmarks is that the performance throughout the optimisation matters, as it directly impacts the amount of electricity generated.

Table 1: Comparison of benchmarks.

Benchmark	Type	Key challenge	Dimensionality	# problems
Materials (Liang et al., 2021)	Real	Small size	3/5	3
Wind farm layout (Bliek et al., 2021)	Simulation	Sample efficiency	10	1
Renewable control (Abel et al., 2018)	Simulation	Scaling, context	2	10
Environmental monitoring (LAQN-BO)	Real	Priors	2	214+365

The materials benchmark is the easiest to start using, as the data is provided in CSV files, followed by that for environmental monitoring, LAQN-BO, which provides a simple Python interface to the data. The benchmarks for wind farm layout and renewable control require more setup, as methods need to be interfaced with the simulators. Their advantage is that they are easier to extend, and can be evaluated for more input values. LAQN-BO can be easily extended to more problems using the LAQN data (Imperial College London, 1993), but extensions to other base data sets would require more work. The materials benchmark is the hardest to extend, as it relies on results from synthesising and evaluating materials.

7 Conclusion and other applications

Bayesian optimisation and related methods have many more climate-related applications: learning energy consumptions of individual household appliances (Jia et al., 2019); optimising charging protocols for electric vehicle batteries (Attia et al., 2020); scheduling smart appliances to smooth out demand curves (Tabakhi et al., 2020); determining vibration suppression parameters for floating wind turbines (Zhang et al., 2022); producing policies to reduce the impact of livestock diseases made more prevalent by global warming (Spooner et al., 2020); and tuning parameters for HVAC system and building models to reduce cooling and heating energy demands (Zhan et al., 2022; Chakrabarty et al., 2021a;b). Bayesian optimisation also has applications in improving climate models, by targeting informative training data (Watson-Parris, 2021). And it has been used for parameter tuning of a global land surface model (Druel et al., 2017), a landslide model (Pradhan et al., 2021) and forecasting models for electricity generation and demand (Trivedi & Khadem, 2021).

To encourage more Bayesian optimisation practitioners to work on climate change applications, an important next step is to establish more benchmarks, standard data sets or simulators, as we do with the LAQN-BO benchmark. More challenging benchmarks should also be established, e.g. for constrained or multi-objective optimisation, to encompass the full complexity of real problems. For example: outside a simulation, some of the ground might be unsuitable for building wind turbines; and when building renewable energy generators there are multiple competing objectives (Wu et al., 2018; Flecker et al., 2022). Additionally, there might be constraints for controlling the generators, due to increased energy costs and fatigue when changing settings. The energy cost of changing the panel angles is included in the renewable control benchmark (Abel et al., 2018), so Bayesian Ascent (Park & Law, 2016a) should be tested, as it limits the difference in input — panel angles — between iterations. As a further extension, the layout and control of wind farms can be optimised jointly (Chen et al., 2022).

In the future, all these benchmarks should be brought together and their interfaces aligned, to create a standardised, climate-themed benchmark suite. By identifying four benchmarks representing the main application domains, introducing a new benchmark, and highlighting the potential for Bayesian optimisation, we take an important step in facilitating future research.

Acknowledgments

We would like to thank Linus Ericsson and Michael Gutmann at the University of Edinburgh for helpful suggestions for improving the paper.

This work was supported by the EPSRC Centre for Doctoral Training in Data Science, funded by the UK Engineering and Physical Sciences Research Council (grant EP/L016427/1) and the University of Edinburgh.

References

Hany Abdelrahman, Felix Berkenkamp, Jan Poland, and Andreas Krause. Bayesian optimization for maximum power point tracking in photovoltaic power plants. In *2016 European Control Conference (ECC)*, pp. 2078–2083. IEEE, 2016.

David Abel. simple_rl: Reproducible reinforcement learning in python. 2019.

David Abel, Edward C Williams, Stephen Brawner, Emily Reif, and Michael L Littman. Bandit-based solar panel control. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

B. Ainslie, C. Reuten, D. G. Steyn, Nhu D. Le, and James V. Zidek. Application of an entropy-based Bayesian optimization technique to the redesign of an existing monitoring network for single air pollutants. *Journal of Environmental Management*, 90(8):2715–2729, June 2009. ISSN 0301-4797. doi: 10.1016/j.jenvman.2009.02.016. URL `https://www.sciencedirect.com/science/article/pii/S0301479709000711`.

Leif Erik Andersson and Lars Imsland. Real-time optimization of wind farms using modifier adaptation and machine learning. *Wind Energy Science*, 5(3):885–896, 2020.

Emil Annevelink, Rachel Kurchin, Eric Muckley, Lance Kavalsky, Vinay I Hegde, Valentin Sulzer, Shang Zhu, Jiankun Pu, David Farina, Matthew Johnson, et al. Automat: Automated materials discovery for electrochemical systems. *MRS Bulletin*, pp. 1–9, 2022.

Martin Asenov, Marius Rutkauskas, Derryck Reid, Kartic Subr, and Subramanian Ramamoorthy. Active Localization of Gas Leaks Using Fluid Simulation. *IEEE Robotics and Automation Letters*, 4(2):1776–1783, April 2019. ISSN 2377-3766. doi: 10.1109/LRA.2019.2895820.

Peter M Attia, Aditya Grover, Norman Jin, Kristen A Severson, Todor M Markov, Yang-Hung Liao, Michael H Chen, Bryan Cheong, Nicholas Perkins, Zi Yang, et al. Closed-loop optimization of fast-charging protocols for batteries with machine learning. *Nature*, 578(7795):397–402, 2020.

Arun Kumar AV, Santu Rana, Alistair Shilton, and Svetha Venkatesh. Human-ai collaborative Bayesian optimisation. *Advances in Neural Information Processing Systems*, 35:16233–16245, 2022.

Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*, 2020. URL `http://arxiv.org/abs/1910.06403`.

Daniil Bash, Yongqiang Cai, Vijila Chellappan, Swee Liang Wong, Xu Yang, Pawan Kumar, Jin Da Tan, Anas Abutaha, Jayce JW Cheng, Yee-Fun Lim, et al. Multi-fidelity high-throughput optimization of electrical conductivity in P3HT-CNT composites. *Advanced Functional Materials*, pp. 2102606, 2021.

Laurens Bliek. A survey on sustainable surrogate-based optimisation. *Sustainability*, 14(7):3867, 2022.

Laurens Bliek, Arthur Guijt, Rickard Karlsson, Sicco Verwer, and Mathijs de Weerdt. Expobench: Benchmarking surrogate-based optimisation algorithms on expensive black-box functions. *arXiv preprint arXiv:2106.04618*, 2021.

Ankush Chakrabarty, Claus Danielson, Scott A Bortoff, and Christopher R Laughman. Accelerating self-optimization control of refrigerant cycles with Bayesian optimization and adaptive moment estimation. *Applied Thermal Engineering*, 197:117335, 2021a.

Ankush Chakrabarty, Emilio Maddalena, Hongtao Qiao, and Christopher Laughman. Scalable Bayesian optimization for model calibration: Case study on coupled building and HVAC dynamics. *Energy and Buildings*, 253:111460, 2021b.

Kaixuan Chen, Jin Lin, Yiwei Qiu, Feng Liu, and Yonghua Song. Joint optimization of wind farm layout considering optimal control. *Renewable Energy*, 182:787–796, 2022.

Tinkle Chugh and Endi Ymeraj. Wind farm layout optimisation using set based multi-objective Bayesian optimisation. In *Genetic and Evolutionary Computation Conference Companion (GECCO '22 Companion)*, pp. 695–698. ACM, New York, NY, USA, 07 2022. URL `https://doi.org/10.1145/3520304.3528951`.

George De Ath, Richard M Everson, Alma AM Rahat, and Jonathan E Fieldsend. Greed is good: Exploration and exploitation trade-offs in Bayesian optimisation. *ACM Transactions on Evolutionary Learning and Optimization*, 1(1):1–22, 2021.

Bart M Doekemeijer, Daan C Van Der Hoek, and Jan-Willem van Wingerden. Model-based closed-loop wind farm control for power maximization using Bayesian optimization: a large eddy simulation study. In *2019 IEEE Conference on Control Technology and Applications (CCTA)*, pp. 284–289. IEEE, 2019.

Hongyang Dong, Jingjie Xie, and Xiaowei Zhao. Wind farm control technologies: From classical control to reinforcement learning. *Progress in Energy*, 2022.

Priya L Donti and J Zico Kolter. Machine learning for sustainable energy systems. *Annual Review of Environment and Resources*, 46:719–747, 2021.

Arsène Druel, Philippe Peylin, Gerhard Krinner, Philippe Ciais, Nicolas Viovy, Anna Peregon, Vladislav Bastrikov, Natalya Kosykh, and Nina Mironycheva-Tokareva. Towards a more detailed representation of high-latitude vegetation in the global land surface model ORCHIDEE (ORC-HL-VEGv1. 0). *Geoscientific Model Development*, 10(12):4693–4722, 2017.

Marcello Fiducioso, Sebastian Curi, Benedikt Schumacher, Markus Gwerder, and Andreas Krause. Safe contextual Bayesian optimization for sustainable room temperature PID control tuning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 5850–5856. International Joint Conferences on Artificial Intelligence Organization, 7 2019. doi: 10.24963/ijcai.2019/811. URL `https://doi.org/10.24963/ijcai.2019/811`.

Alexander S Flecker, Qinru Shi, Rafael M Almeida, Héctor Angarita, Jonathan M Gomes-Selman, Roosevelt García-Villacorta, Suresh A Sethi, Steven A Thomas, N LeRoy Poff, Bruce R Forsberg, et al. Reducing adverse impacts of amazon hydropower expansion. *Science*, 375(6582):753–760, 2022.

Jose Pablo Folch, Shiqiang Zhang, Robert Lee, Behrang Shafei, David Walz, Calvin Tsay, Mark van der Wilk, and Ruth Misener. SnAKe: Bayesian optimization with pathwise exploration. *Advances in Neural Information Processing Systems*, 35:35226–35239, 2022.

Peter I Frazier and Jialei Wang. Bayesian optimization for materials design. In *Information science for materials discovery and design*, pp. 45–75. Springer, 2016.

Daniel Frey, K.C. Neyerlin, and Miguel A. Modestino. Bayesian optimization of electrochemical devices for electrons-to-molecules conversions: The case of pulsed CO2 electroreduction. *ChemRxiv*, 2022. doi: 10.26434/chemrxiv-2022-n09hc.

Tianyu Gao and Xiaoli Bai. Bayesian optimization-based three-dimensional, time-varying environment monitoring using an UAV. *Journal of Intelligent & Robotic Systems*, 105(4):1–21, 2022.

Sajad Haghanifar, Bolong Cheng, Michael McCourt, and Paul Leu. Using Bayesian optimization to improve solar panel performance by developing antireflective, superomniphobic glass. 2021. URL `https://www.climatechange.ai/papers/icml2021/84`.

Sigrid Passano Hellan, Christopher G. Lucas, and Nigel H. Goddard. Optimising Placement of Pollution Sensors in Windy Environments, December 2020. URL `http://arxiv.org/abs/2012.10770`. Presented at the NeurIPS 2020 Workshop on AI for Earth Sciences.

Sigrid Passano Hellan, Christopher G. Lucas, and Nigel H. Goddard. Bayesian optimisation for active monitoring of air pollution. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(11):11908–11916, Jun. 2022. doi: 10.1609/aaai.v36i11.21448. URL `https://ojs.aaai.org/index.php/AAAI/article/view/21448`.

Carl Hvarfner, Frank Hutter, and Luigi Nardi. Joint entropy search for maximally-informed Bayesian optimization. In *Advances in Neural Information Processing Systems*, 2022.

Environmental Research Group Imperial College London. London Air Quality Network. `https://www.londonair.org.uk/LondonAir/Default.aspx`, 1993. Accessed: 2021-08-31.

Yiling Jia, Nipun Batra, Hongning Wang, and Kamin Whitehouse. Active collaborative sensing for energy breakdown. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 1943–1952, 2019.

Steven Kleinegesse and Michael U Gutmann. Bayesian experimental design for implicit models by mutual information neural estimation. In *International Conference on Machine Learning*, pp. 5316–5326. PMLR, 2020.

Fang-Yie Leu and Jia-Sheng Ho. Air pollution source identification by using neural network with Bayesian optimization. In *Innovative Mobile and Internet Services in Ubiquitous Computing: Proceedings of the 13th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS-2019)*, pp. 514–524. Springer, 2020.

Qiaohao Liang, Aldair E Gongora, Zekun Ren, Armi Tiihonen, Zhe Liu, Shijing Sun, James R Deneault, Daniil Bash, Flore Mekki-Berrada, Saif A Khan, et al. Benchmarking the performance of Bayesian optimization across multiple experimental materials science domains. *npj Computational Materials*, 7(1):1–10, 2021.

Haitao Liu, Yew-Soon Ong, Xiaobo Shen, and Jianfei Cai. When Gaussian process meets big data: A review of scalable GPs. *IEEE transactions on neural networks and learning systems*, 31(11):4405–4423, 2020.

S Lyden, W Olding, and Z Darbari. Bayesian optimization for maximum power point tracking in photovoltaic systems. In *2018 IEEE Power & Energy Society General Meeting (PESGM)*, pp. 1–5. IEEE, 2018.

Anastasia Makarova, Huibin Shen, Valerio Perrone, Aaron Klein, Jean Baptiste Faddoul, Andreas Krause, Matthias Seeger, and Cedric Archambeau. Automatic termination for hyperparameter optimization. In *International Conference on Automated Machine Learning*, pp. 7–1. PMLR, 2022.

R. Marchant and F. Ramos. Bayesian optimisation for Intelligent Environmental Monitoring. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2242–2249. IEEE, October 2012. doi: 10.1109/IROS.2012.6385653.

Flore Mekki-Berrada, Zekun Ren, Tan Huang, Wai Kuan Wong, Fang Zheng, Jiaxun Xie, Isaac Parker Siyu Tian, Senthilnath Jayavelu, Zackaria Mahfoud, Daniil Bash, et al. Two-step machine learning enables optimized nanoparticle synthesis. *npj Computational Materials*, 7(1):1–10, 2021.

John Mern, Anil Yildiz, Zachary Sunberg, Tapan Mukerji, and Mykel J Kochenderfer. Bayesian optimized Monte Carlo planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11880–11887, 2021. URL `https://ojs.aaai.org/index.php/AAAI/article/view/17411`.

Philippe Morere, Roman Marchant, and Fabio Ramos. Sequential Bayesian optimization as a POMDP for environment monitoring with UAVs. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6381–6388. IEEE, May 2017. doi: 10.1109/ICRA.2017.7989754.

Nikolaos Moustakis, Sebastiaan Paul Mulders, Jens Kober, and Jan-Willem van Wingerden. A practical Bayesian optimization approach for the optimal estimation of the rotor effective wind speed. In *2019 American Control Conference (ACC)*, pp. 4179–4185. IEEE, 2019.

SP Mulders, AK Pamososuryo, and JW Van Wingerden. Efficient tuning of individual pitch control: a Bayesian optimization machine learning approach. In *Journal of Physics: Conference Series*, volume 1618, pp. 022039. IOP Publishing, 2020.

Quan Nguyen, Kaiwen Wu, Jacob Gardner, and Roman Garnett. Local Bayesian optimization via maximizing probability of descent. *Advances in neural information processing systems*, 35:13190–13202, 2022.

NREL. FLORIS. version 2.1.1, 2020. URL `https://github.com/NREL/floris`.

Rafael Oliveira, Louis Tiao, and Fabio T Ramos. Batch Bayesian optimisation via density-ratio estimation with guarantees. *Advances in Neural Information Processing Systems*, 35:29816–29829, 2022.

Diana L Ortiz-Montalvo, Aaron Gilad Kusne, Austin McDannald, Daniel Siderius, Kamal Choudhary, and Taner Yildirim. Machine learning for climate change: Guiding discovery of sorbent materials for direct air capture of CO2. In *ICML 2021 Workshop on Tackling Climate Change with Machine Learning*, 2021. URL `https://www.climatechange.ai/papers/icml2021/84`.

Leonard Papenmeier, Luigi Nardi, and Matthias Poloczek. Increasing the scope as you learn: Adaptive Bayesian optimization in nested subspaces. In *Advances in Neural Information Processing Systems*, 2022.

Jinkyoo Park. Contextual Bayesian optimization with trust region (CBOTR) and its application to cooperative wind farm control in region 2. *Sustainable Energy Technologies and Assessments*, 38:100679, 2020.

Jinkyoo Park and Kincho H Law. A Bayesian optimization approach for wind farm power maximization. In *Smart Sensor Phenomena, Technology, Networks, and Systems Integration 2015*, volume 9436, pp. 46–57. SPIE, 2015.

Jinkyoo Park and Kincho H Law. Bayesian ascent: A data-driven optimization scheme for real-time control with application to wind farm power maximization. *IEEE Transactions on Control Systems Technology*, 24(5):1655–1668, 2016a.

Jinkyoo Park and Kincho H Law. A data-driven, cooperative wind farm control to maximize the total power production. *Applied Energy*, 165:151–165, 2016b.

Jinkyoo Park, Soon-Duck Kwon, and Kincho H Law. A data-driven approach for cooperative wind farm control. In *2016 American Control Conference (ACC)*, pp. 525–530. IEEE, 2016.

Jinkyoo Park, Soon-Duck Kwon, and Kincho Law. A data-driven, cooperative approach for wind farm control: a wind tunnel experimentation. *Energies*, 10(7):852, 2017.

Jinkyoo Park, Max Ferguson, and Kincho H Law. Data driven analytics (machine learning) for system characterization, diagnostics and control optimization. In *Workshop of the European Group for Intelligent Computing in Engineering*, pp. 16–36. Springer, 2018.

Mads M Pedersen, Paul van der Laan, Mikkel Friis-Møller, Jennifer Rinker, and Pierre-Elouan Réthoré. Dtuwindenergy/pywake: Pywake. *Zenodo [code]*, 10, 2019.

Biswajeet Pradhan, Maher Ibrahim Sameen, Husam AH Al-Najjar, Daichao Sheng, Abdullah M Alamri, and Hyuck-Jin Park. A meta-learning approach of optimisation for spatial prediction of landslides. *Remote Sensing*, 13(22):4521, 2021.

Tom Rainforth, Adam Foster, Desi R Ivanova, and Freddie Bickford Smith. Modern Bayesian experimental design. *arXiv preprint arXiv:2302.14545*, 2023.

Shyam Sundhar Ramesh, Pier Giuseppe Sessa, Andreas Krause, and Ilija Bogunovic. Movement penalized Bayesian optimization with application to wind energy systems. *arXiv preprint arXiv:2210.08087*, 2022.

David Rolnick, Priya L Donti, Lynn H Kaack, Kelly Kochanski, Alexandre Lacoste, Kris Sankaran, Andrew Slavin Ross, Nikola Milojevic-Dupont, Natasha Jaques, Anna Waldman-Brown, et al. Tackling climate change with machine learning. *ACM Computing Surveys (CSUR)*, 55(2):1–96, 2022.

Federico Peralta Samaniego, Daniel Gutiérrez Reina, Sergio L Toral Marín, Mario Arzamendia, and Derlis O Gregor. A Bayesian optimization approach for water resources monitoring through an autonomous surface vehicle: The Ypacarai lake case study. *IEEE Access*, 9:9163–9179, 2021.

Nayat Sanchez-Pi, Luis Marti, André Abreu, Olivier Bernard, Colomban de Vargas, Damien Eveillard, Alejandro Maass, Pablo A Marquet, Jacques Sainte-Marie, Julien Salomon, et al. Artificial intelligence, machine learning and modeling for understanding the oceans and climate change. In *NeurIPS 2020 Workshop-Tackling Climate Change with Machine Learning*, 2020.

Kristen A Severson, Olivia P Pfeiffer, Jie Chen, Kai Gong, Jeremy Gregory, Richard Goodwin, and Elsa A Olivetti. Amortized inference of Gaussian process hyperparameters for improved concrete strength trajectory prediction. *Tackling Climate Change with Machine Learning: workshop at NeurIPS 2021*, 2021.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015. ISSN 1558-2256. doi: 10.1109/JPROC.2015.2494218.

A. Singh, F. Ramos, H. D. Whyte, and W. J. Kaiser. Modeling and decision making in spatio-temporal processes for environmental surveillance. In *2010 IEEE International Conference on Robotics and Automation*, pp. 5490–5497, May 2010. doi: 10.1109/ROBOT.2010.5509934.

Lei Song, Ke Xue, Xiaobin Huang, and Chao Qian. Monte carlo tree search based variable selection for high dimensional Bayesian optimization. *arXiv preprint arXiv:2210.01628*, 2022.

Thomas Spooner, Anne E Jones, John Fearnley, Rahul Savani, Joanne Turner, and Matthew Baylis. Bayesian optimisation of restriction zones for bluetongue control. *Scientific reports*, 10(1):1–18, 2020.

Shijing Sun, Armi Tiihonen, Felipe Oviedo, Zhe Liu, Janak Thapa, Yicheng Zhao, Noor Titan P Hartono, Anuj Goyal, Thomas Heumueller, Clio Batali, et al. A data fusion approach to optimize compositional stability of halide perovskites. *Matter*, 4(4):1305–1322, 2021.

Atena M Tabakhi, William Yeoh, and Ferdinando Fioretto. The smart appliance scheduling problem: A Bayesian optimization approach. In *International Conference on Principles and Practice of Multi-Agent Systems*, pp. 100–115. Springer, 2020.

Alexander Thebelt, Calvin Tsay, Robert Lee, Nathan Sudermann-Merx, David Walz, Behrang Shafei, and Ruth Misener. Tree ensemble kernels for Bayesian optimization with known constraints over mixed-feature spaces. *Advances in Neural Information Processing Systems*, 35:37401–37415, 2022.

Peter Tillmann, Klaus Jäger, and Christiane Becker. Minimising the levelised cost of electricity for bifacial solar panel arrays using Bayesian optimisation. *Sustainable Energy & Fuels*, 4(1):254–264, 2020.

Kevin Tran and Zachary W Ulissi. Active learning across intermetallics to guide discovery of electrocatalysts for CO2 reduction and H2 evolution. *Nature Catalysis*, 1(9):696–703, 2018.

Rohit Trivedi and Shafi Khadem. Peak demand management and schedule optimisation for energy storage through the machine learning approaches. In *IEEE EUROCON 2021-19th International Conference on Smart Technologies*, pp. 157–163. IEEE, 2021.

Ben Tu, Axel Gandy, Nikolas Kantas, and Behrang Shafei. Joint entropy search for multi-objective Bayesian optimization. *arXiv preprint arXiv:2210.02905*, 2022.

Simon Valentin, Steven Kleinegesse, Neil R. Bramley, Peggy Seriès, Michael U. Gutmann, and Christopher G. Lucas. Designing optimal behavioral experiments using machine learning, 2023.

Duncan Watson-Parris. Machine learning for weather and climate are worlds apart. *Philosophical Transactions of the Royal Society A*, 379(2194):20200098, 2021.

Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

Xiaojian Wu, Jonathan Gomes-Selman, Qinru Shi, Yexiang Xue, Roosevelt Garcia-Villacorta, Elizabeth Anderson, Suresh Sethi, Scott Steinschneider, Alexander Flecker, and Carla Gomes. Efficiently approximating the pareto frontier: hydropower dam placement in the amazon basin. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Shanghui Yang, Xiaowei Deng, Zilong Ti, Bowen Yan, and Qingshan Yang. Cooperative yaw control of wind farm using a double-layer machine learning framework. *Renewable Energy*, 193:519–537, 2022.

Sicheng Zhan, Gordon Wichern, Christopher Laughman, Adrian Chong, and Ankush Chakrabarty. Calibrating building simulation models using multi-source datasets and meta-learned Bayesian optimization. *Energy and Buildings*, pp. 112278, 2022.

Xiantao Zhang, Da Lu, Hongyang Dong, Xiaowei Zhao, Feargal Brennan, and Yibo Liang. Vibration suppression of multi-component floating structures via passive TMDs and Bayesian ascent. *Ocean Engineering*, 259:112088, 2022.

Miao Zhong, Kevin Tran, Yimeng Min, Chuanhao Wang, Ziyun Wang, Cao-Thang Dinh, Phil De Luna, Zongqian Yu, Armin Sedighian Rasouli, Peter Brodersen, et al. Accelerated discovery of CO2 electrocatalysts using active machine learning. *Nature*, 581(7807):178–183, 2020.

3.5 Discussion

In the paper we surveyed applications of Bayesian optimisation and related methods to problems related to climate change. In doing so we identified four main clusters of work, and suggested a benchmark for each of them. These benchmarks are intended as starting points for practitioners and researchers coming from machine learning who want to evaluate their methods on climate-related problems.

The paper does not include any experiments, and we did not run any comparisons of methods or implementations. But we tested the benchmarks before recommending them, making the contributions listed below.

Material discovery (Liang et al., 2021): This benchmark was provided as CSV files, so no testing was necessary.

Wind farm layout (Bliek, Guijt, Karlsson, Verwer, & de Weerdt, 2023): After the code was published the reference link to the necessary turbine and wind farm specification file had broken. This happened due to changes in a library dependency. We provided a pull request on GitHub to fix the problem. Due to new changes in the dependency, this happened again and another pull request was provided.

Optimal renewable control (Abel, Williams, Brawner, Reif, & Littman, 2018), originally introduced for reinforcement learning: We debugged the dependencies, provided a requirements file and ported the code to Python 3. We also provided a basic Bayesian optimisation implementation in BoTorch (Balandat et al., 2020), making our contributions available at `https://github.com/sighellan/solar_panels_rl`. The Bayesian optimisation implementation was not meant to be competitive with the reinforcement learning methods, but rather to show how a BO library can be interfaced with the simulation-based benchmark. Fig. 3.1 shows how our implementation compares to some of the methods used in the original paper.

Environmental monitoring: We did not find a suitable benchmark for Bayesian optimisation within environmental monitoring. So we cleaned up and improved the documentation for the code we used to generate one of the data sets in Chapter 4[1]. The code is available at `https://github.com/sighellan/LAQN-BO`.

1. The generated optimisation tasks in Chapter 3 differ from those in Chapter 4 in the preprocessing. In Chapter 4 the data from each task were mean-centred using data from that task. In Chapter 3 both years are standardised using the tuning data.

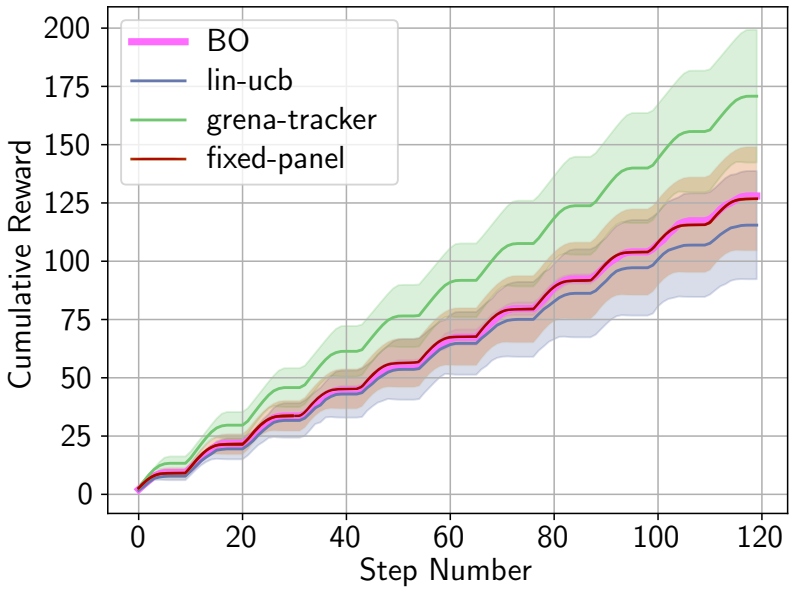


Figure 3.1: Example of integrating Bayesian optimisation with the optimal renewable control benchmark (Abel et al., 2018). BO does very similarly to 'fixed-panel', where the panel faces a constant direction. 'lin-ucb' is a contextual bandit approach and 'grena-tracker' an algorithm for tracking the sun's position in the sky (Grena, 2012). The lines for BO and 'fixed-panel' almost overlap.

3.6 Future work

An obvious next step is to integrate the benchmarks into a common framework, making it easier to collect results for a new method. And then to compare some standard Bayesian optimisation methods and report their results. This will show how easy or challenging the problems are for Bayesian optimisation, and therefore how much more work is needed. Once these benchmarks have been solved convincingly, they can be extended and supplemented, for instance with planning the layout of larger wind parks. This should be done in collaboration with domain experts, to make the problems as realistic as possible and focus research attention on problems with real-world impact.

Although we have not conducted experiments, we expect the surveyed benchmarks to be challenging. This is a positive, as trivial benchmarks would not be interesting to machine learning researchers. We summarise below how Bayesian optimisation compared to other methods on the benchmarks, which explains why we expect the benchmarks to be challenging:

- For *material discovery* Bayesian optimisation out-performs random search.
- For *wind farm layout*, Bliek et al. note that Bayesian optimisation is out-performed by all the other methods.
- The BO implementation for the *optimal renewable control* benchmark is very simple, but it does not beat the sun-tracking benchmark 'grena-tracker', see Fig. 3.1. In the absence of clouds and reflective surroundings, tracking the sun should be very competitive. But the BO implementation also does not have meaningfully better performance than a fixed panel. A better implementation should be able to do better than a fixed policy.
- In Chapter 4 we show that for the *environmental monitoring* benchmark Bayesian optimisation requires many tweaks to beat random sampling. This benchmark is difficult because of the small search space, and non-smooth optimisation landscape. The small search space means that random sampling finds the optimum in 40–52 evaluations, so Bayesian optimisation must be very sample efficient to beat it.

The benchmarks also have different challenges: training BO's surrogate model on all the collected samples will not be an issue for the material discovery benchmark, but will be for the renewable control one. Once we have results for competitive Bayesian optimisation methods we can look at automatically deciding which one to apply.

3.7 Limitations

The impressive results obtained by machine learning easily lead to hype, exaggerations of its current capabilities, and inflated expectations. At the same time, climate change is a very important field, which brings with it a temptation to emphasise connections even when these links are weak.

The survey is intentionally directed at Bayesian optimisation and machine learning researchers. It is up to us to demonstrate that Bayesian optimisation can contribute to solutions, rather than for people already working on climate change mitigation to pause their current projects in order to learn machine learning. This collection of four benchmarks is a good place for us to start. If we can beat state-of-the-art on these benchmarks, that will justify more investments in applications of Bayesian optimisation, and that in turn can produce benefits to the climate. But Bayesian optimisation is in no way a solution in itself: in the four application domains identified, Bayesian optimisation is always used to support an underlying technology.

- *Material discovery*: It is used to find more efficient solar panel materials, so that the solar panels can more effectively displace coal or gas for electricity production.
- *Wind farm layout*: It is used to find more efficient wind farm layouts, so that the wind turbines can displace coal or gas for electricity production.
- *Optimal renewable control*: It is used to find more efficient control settings, so that the solar panels (or wind turbines) can displace coal or gas for electricity production.
- *Environmental monitoring*: It is used to monitor the environment, so that interventions can be brought in to improve the environment.

4

Bayesian Optimisation for Air Pollution Monitoring

1 animal escaped.

– Mac Passano listing the number of crabs surviving his thesis experiments

This chapter introduces air pollution monitoring as an application for Bayesian optimisation, preparing two new data sets for this purpose. It also develops a new method for transfer learning, using related data to learn priors for the Gaussian process hyperparameters. Content from this chapter has been presented at the AI for Earth Sciences workshop at NeurIPS 2020 (Hellan et al., 2020) and at AAAI 2022 (Hellan et al., 2022). Some of the content in this chapter has been taken verbatim from these two publications, which are also included in Appendices A.1 and A.2. The new transfer learning method is developed further in Chapter 5.

4.1 Introduction

Air pollution is a globally important health issue, with several million deaths attributed to it every year (Cohen et al., 2017; Lelieveld, Evans, Fnais, Giannadaki, & Pozzer, 2015). In this chapter we consider the problem of air pollution monitoring, and the possibility of using Bayesian optimisation to plan a static monitoring network. Low-cost sensors have become available, but they are still expensive enough that we want to deploy them efficiently. We could plan the networks using historical data or domain expertise to identify the areas that are most polluted, and therefore most in need of

monitoring. But this relies on past monitoring networks — which are not available in many locations — or on consulting domain experts — which can be expensive. By automating the decision-making process for sensor placement, we can help lower-resource groups make the most of their limited resources and sensors.

This goal, of simplifying the process for setting up new monitoring networks in a low-cost way, motivates our decision to focus on simple feature sets, and models that are relatively inexpensive in terms of computational costs. Although models are generally improved by using data on road traffic and other pollution sources, these data can be difficult or expensive to obtain for developing nations or citizen scientists. We demonstrate that even a minimal feature set containing only sensor locations and readings can be useful. Additionally, all the computations are done on CPUs, with the time to choose the next sensor location being 31 seconds or less, so access to expensive data centres is not needed.

The problem of air pollution monitoring also serves as an interesting deep dive into an application of automatically learning hyperparameter priors from related data. We found that even for relatively simple optimisation tasks an informative prior was needed for good results. Simple priors based on maximum likelihood estimators on the tuning data worked better than no priors, and more sophisticated priors using Markov chain Monte Carlo to learn distributions of the hyperparameters worked even better. The main contribution of this chapter is showing that Bayesian optimisation has the potential to automate planning of pollution sensor networks, and that this is made possible by using prior transfer to learn hyperparameter priors. Specifically, we consider the problem of iteratively placing stationary sensors and locating the maximum average pollution in the given area.

The chapter begins with an introduction to air pollution, its impact and how it is monitored. We then discuss related work applying Gaussian processes and Bayesian optimisation to air pollution. Next, we present the data sets used. This is followed by introducing new GP kernels for modelling air pollution and especially the impact of wind. We run three sets of experiments, gradually moving from no priors to developing a new method for prior transfer. The chapter ends with a discussion of the results and future work.

4.2 Background

4.2.1 Air pollution and monitoring

Air pollution refers to harmful gases and particles introduced into the air by humans. It is grouped into two categories: indoor and outdoor, also called ambient, air pollution. Major sources of indoor air pollution are cooking, lighting and heating, using solid fuels such as coal, wood, charcoal, animal dung and domestic waste (Gordon et al., 2014). Ambient air pollution comes from a range of sources including heating, manufacturing, and transportation (Mukherjee & Agrawal, 2017), and can travel between countries (The Lancet, 2016). Throughout this thesis we only consider ambient air pollution. An accessible introduction to air pollution is given by Smedley (2019).

Health impacts

The World Health Organization (WHO) recommends limits for carbon monoxide (CO), sulphur dioxide (SO_2), nitrogen dioxide (NO_2), ozone (O_3), and particulate matter of different sizes ($PM_{2.5}$ and PM_{10}) (World Health Organization et al., 2021, pp. xvii–xviii). $PM_{2.5}$ refers to particles suspended in the air with diameter less than $2.5\,\mu m$, PM_{10} to such particles with diameter less than $10\,\mu m$. Particulate matter is a joint term for a range of different types of particles, the sources of which vary between locations (Mukherjee & Agrawal, 2017).

Ambient air pollution is one of the leading causes of death globally, with $PM_{2.5}$ causing 3–4 million deaths each year (Cohen et al., 2017; Lelieveld et al., 2015). Of the different pollutants, the effects of particulate matter are best understood (Landrigan et al., 2018, p. 475), but another commonly monitored pollutant is NO_2[1], as it is used to indicate the presence of traffic-generated air pollution (Katsouyanni, 2003). NO_2 is also part of an O_3-generating process, and short-term increases in O_3 concentration are followed by short-term increases in mortality (Katsouyanni, 2003). In addition to the recommendations from the WHO, many countries, including the EU, USA, India and China, have adopted air pollution regulations (World Health Organization, 2006, pp. 174–175). These regulations include NO_2 and particulate matter.

1. It is difficult to quantify the impact of NO_2, because its presence is often correlated with other pollutants. For instance, vehicle exhaust includes both NO_2 and fine particles. See Atkinson, Butland, Anderson, and Maynard (2018) for a discussion of the evidence. But monitoring is useful for gaining a better understanding of the health impacts of NO_2.

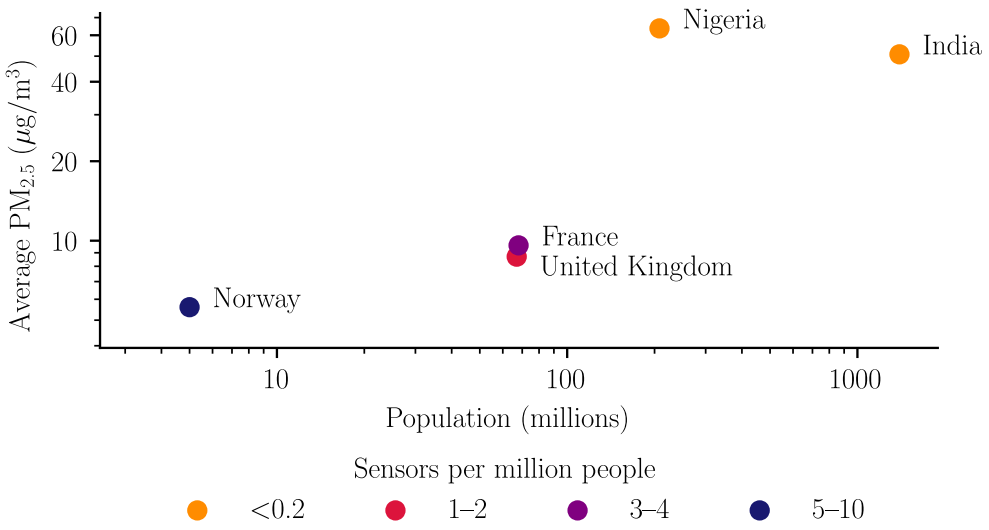


Figure 4.1: Comparison of densities of traditional monitoring sensors for selected countries. Population estimates from The World Bank (2023), pollution estimates from Van Donkelaar et al. (2021) and sensor density estimates from Martin et al. (2019).

Monitoring

Monitoring is important for multiple reasons. It can be used to estimate health impacts, guide town planning[2], target interventions, motivate low emission zones and enforce regulatory compliance. The more specific task of locating the maximum can be used to verify compliance more efficiently: if the area of the city with the most pollution is adhering to the limits, then the rest of the city is too. By identifying the worst hotspot, it also identifies the area most in need of intervention.

Traditional air pollution monitoring uses large and expensive sensors that are typically managed by national or municipal authorities. The sensor locations are decided based on domain knowledge and constraints posed by the bulky nature of the sensors (Carminati, Ferrari, & Sampietro, 2017). Six examples are shown in Figs. 4.2d to 4.2i. As can be seen, these are not easy to move, and cannot be placed in an arbitrary location. The density of monitoring varies greatly, see Fig. 4.1. This problem is exacerbated by the density being higher in high-resource countries, which in general are also the ones with less air pollution (World Health Organization, 2021, pp. 8–10).

2. Research on air pollution in London has shown that children exposed to more air pollution develop smaller lungs (Mudway et al., 2019).

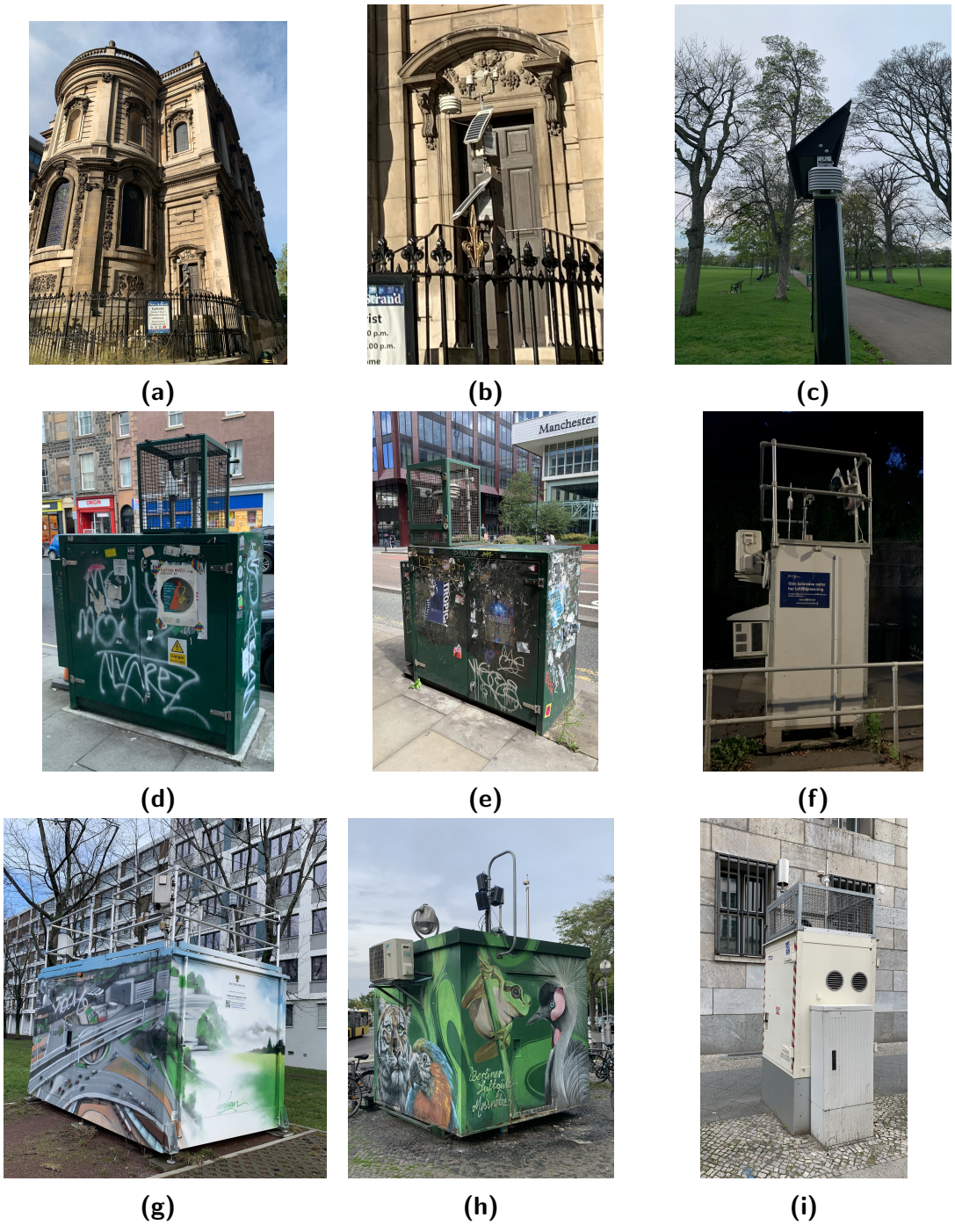


Figure 4.2: Examples of monitoring sensors. a) a low-cost sensor in London (June 2023). b) a close-up of the same sensor. c) a low-cost sensor in Edinburgh (May 2023). d–i) traditional monitoring stations in Edinburgh (May 2022), Manchester (July 2023), Oslo (August 2022), Halle (February 2023), near Berlin Zoo (September 2022) and Berlin Potsdamer Platz (September 2022). Photographs by author.

More recently, low-cost air pollution sensors have become available, often developed for personal exposure monitoring (Plume Labs, 2023; Smedley, 2019, pp. 71–74), citizen science (Kelly et al., 2017; PurpleAir, 2020) or research (Lin et al., 2015; X. Liu et al., 2020; University of Cambridge, 2019) as well as for increasing monitoring density (Castell et al., 2017; Johnston et al., 2019; Spinelle, Gerboles, Villani, Aleixandre, & Bonavitacola, 2015) and extending monitoring to new regions (Wilkinson, Álvarez López, & Bainomugisha, 2019). Low-cost sensors allow groups with limited budgets and domain expertise to create sensor networks, including local governments in developing nations as well as community groups around the world. They can also be deployed in mobile environments, as in open-seneca team (2018); Smith, Ssematimba, Álvarez, and Bainomugisha (2019). Fig. 4.3 illustrates the size and type of locations that are made possible by low cost sensors. Figs. 4.2a and 4.2b show a low-cost sensor installed on the St Mary le Strand church in London and Fig. 4.2c shows one in Edinburgh.

Another example of a low-cost sensor is the KOALA sensor from X. Liu et al. (2020). It is small enough to be attached to a fence, and comes with a solar panel and battery, so it does not need to be connected to the electricity grid. It monitors two types of pollutants: $PM_{2.5}$ and NO_2. In contrast, the St Leonard's monitoring site[3] in Edinburgh, which uses a traditional sensor, measures CO, nitrogen oxide (NO), NO_2, O_3, PM_{10}, $PM_{2.5}$ and SO_2 (Air Quality in Scotland, 2021). The quality and price of low-cost sensors is very variable, but a high-level qualitative comparison of low-cost and reference sensors is given in Table 4.1. A limitation of low-cost sensors is that while they might work well on one type of particulate matter, they might not perform as well on other types. This might be due to the calibration having focused on one type. For the KOALA sensor, differences were found between the quality of measurements for old and fresh traffic emissions (X. Liu et al., 2020).

The low-cost sensors motivate the usefulness of automated planning of monitoring networks, but another approach would be to let the networks grow organically. Companies such as PurpleAir (PurpleAir, 2020) allow individuals to purchase sensors and decide where to place them independently of others. Anyone can buy one of their sensors, and then share the recordings publicly. The advantage of this approach is that anyone interested can purchase one and install it where they find it important to monitor the pollution. However, the sensors are still not cheap, at $229 to $299 (as of November 2023), which is not affordable to everybody. And with air pollution

3. Not the one pictured in Fig. 4.2d, but a larger one.

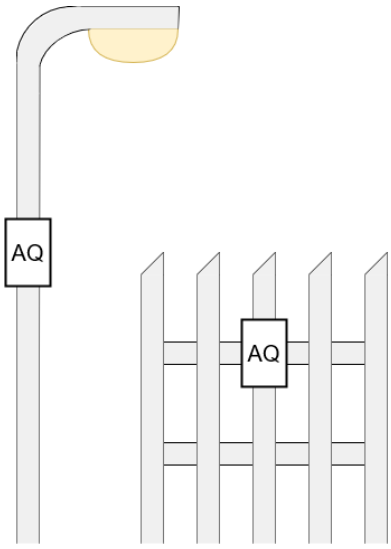


Figure 4.3: Example of low-cost sensor size and installation options. The boxes labelled 'AQ' represent the sensors.

	Reference sensors	Low-cost sensors
Price	High	Lower
Installation	Extensive	Simple
Maintenance	Expensive	Little
Accuracy	Very high	Variable, less known
Size	Large	Small

Table 4.1: Qualitative comparison of reference sensors and low-cost sensors. See also examples in Fig. 4.2.

disproportionally affecting poorer neighbourhoods (Balakrishnan et al., 2019), there is a risk that the areas of most concern will go unmonitored. As there is no centralised planning of the monitoring there is also a risk of sensor inefficiency. Two neighbours could independently decide to buy sensors at the same time, giving mostly duplicate readings. Putting one of them on the other side of town would be more useful but might not be of interest to either of the neighbours. Network planning is therefore useful for maximising the utility of a given number of deployed sensors.

4.2.2 Air pollution and Gaussian processes

In this chapter we explore using Bayesian optimisation with Gaussian process models to plan monitoring networks. Efficient sensor placement requires accurate and sample efficient models that can handle uncertainty, which is a good fit for Gaussian processes. To make them more sample efficient we introduce inductive biases. We do this partly through introducing covariance functions that model the wind direction, and partly through learning priors for the Gaussian process hyperparameters.

Gaussian processes have been successfully applied to air pollution monitoring in a range of settings for prediction and regression (Cheng, Li, Li, Jiang, & Jiang, 2014; Grašič, Mlakar, & Božnar, 2006; J. J. Li, 2014; H. Liu, Yang, Huang, Wang, & Yoo, 2018; X. Liu, Xi, & Ngai, 2016; Reggente et al., 2014; Tanaka et al., 2019; Zhou, Zhao, Lin, Wang, & Li, 2019). Suitable covariance functions are explored in H. Liu et al. (2018), while a new hyperparameter tuning algorithm is proposed in Zhou et al. (2019). Grašič et al. (2006) emphasise the importance of feature selection, while J. J. Li (2014) takes a different approach by introducing a large number of land use, road and traffic characteristics as features. In Tanaka et al. (2019) multiple data sets are combined by using a mixture of latent GPs. In Zheng et al. (2019), GPs are used to calibrate low-cost sensors and detect malfunctioning ones, similarly to the plan in Smith et al. (2019).

4.2.3 Automated planning of monitoring networks

Much less work has been done on using Bayesian optimisation for planning monitoring networks. Previous work has tended to either estimate hyperparameters or their distributions in ways that do not generalise to new sites in a sample efficient way (Ainslie, Reuten, Steyn, Le, & Zidek, 2009) or to focus on robot trajectories (Marchant & Ramos, 2012; Morere, Marchant, & Ramos, 2017; Singh, Ramos, Whyte, & Kaiser, 2010), a distinct problem from the one we consider here. Work has also been done on monitoring gas leaks (Asenov, Rutkauskas, Reid, Subr, & Ramamoorthy, 2019; Reggente & Lilienthal, 2009), but again by using a single moving agent. Folch et al. (2022); Samaniego, Reina, Marín, Arzamendia, and Gregor (2021) use BO for water quality monitoring, but again focusing on path planning, and they use synthetic data. Balandat et al. (2020) include an example in their appendix of applying the

related method of active learning to decide sampling locations for disease monitoring. Our motivations are similar to Smith et al. (2019), who use Gaussian processes for calibrating low-cost pollution sensors, but they do not address the problem of sensor placement.

The novelty of our approach comes from combining Bayesian Optimisation with air pollution monitoring, using kernels informed by wind transportation, see Sections 4.2.4 and 4.4. Our approach differs from the bulk of GP applications to air pollution monitoring by planning the sensor placement, not just modelling the pollution based on a given set of samples. Prediction and regression are only two of the required components when planning a monitoring network. It is also necessary to determine which new sampling locations are most informative about the pollution distribution.

4.2.4 Air pollution and wind

Wind is known to carry pollutants (Samson, 1988, pp. 78–80), and work on gas mapping has shown the importance of including this effect (Asenov et al., 2019; Reggente & Lilienthal, 2009). For instance, Asenov et al. (2019) used a gas localisation task to show that a method incorporating wind velocity outperforms BO using wind-insensitive kernels. To our knowledge, previous work on pollution monitoring using BO and related methods, e.g. Marchant and Ramos (2012), has not considered wind.

By leveraging wind, fewer samples should be needed to build a reasonable model of the pollution landscape. Because we are focusing on time-averaged pollution concentrations, detailed wind information and fluid simulations are not essential: the covariance structure of pollution is driven by the prevailing wind direction, which can be characterised with a single parameter. The kernels presented in Section 4.4 allow such information to be incorporated into BO, and have the advantage that wind need not be measured separately, as is done in Asenov et al. (2019). This is desirable from a cost perspective, as additional wind sensors are not needed.

In Reggente and Lilienthal (2009) a Gaussian kernel is part of the model used to estimate gas distributions. An important difference to our work is that the sampling was done along a predefined trajectory, i.e. there was no active sampling. However, the kernel used to introduce the influence of wind is still interesting. The kernel is taken to be a Gaussian squeezed along the direction of the wind according to the wind intensity, leading to a similar kernel to the Product kernel introduced in Section 4.4.

However, their approach requires both the wind speed and direction to be measured directly, and assumes a linear relationship between the kernel's long axis and the wind speed. In contrast, our kernels can learn the wind direction from the data, and do not rely on a known relationship between the kernel and the wind speed.

4.2.5 Suitability of Bayesian optimisation

Gaussian processes and Bayesian optimisation are well matched to locating air pollution maxima. BO focuses on sample efficiency, in exchange for more computations between each sampling. This is very useful for the air pollution monitoring application, since taking a new sample consists of going out and physically installing a sensor. That makes it an excellent example of an expensive to evaluate, black-box function. Gaussian processes are useful because they naturally capture uncertainty, modelling epistemic and aleatoric uncertainty separately. And they allow inductive biases to be introduced through priors on the hyperparameters or the selection of mean and covariance functions. The models also help with interpretability. Compared to a neural network, the handful of GP hyperparameters are a much smaller number of values to consider, and they have interpretable values, such as the noise parameter which models the measurement noise.

4.3 Data

We use two air pollution monitoring data sets. We refer to the first, and simpler, one as 'Satellite'. It comes from the EU Copernicus project (Copernicus Sentinel-5P (processed by ESA), 2018), consists of measurements taken from a satellite, and gives pollution concentrations in a column through the troposphere. The second comes from the London Air Quality Network (Imperial College London, 1993), is referred to as 'LAQN', and has been collected by the councils of London and the surrounding area. It consists of ground-level measurements — 1.5 to 4 meters above the ground for the subset used — of various pollutants, and the sensors have been grouped into categories based on their location, e.g. kerbside or suburban.

The differences between the data sets are summarised in Table 4.2, and can be seen from comparing Figs. 4.4 and 4.6. In addition to the difference in altitude, the data sets differ in that the readings are spread out on a regular grid for the satellite data but for LAQN are unevenly spread across London, clustering around central London.

	Satellite	LAQN
Altitude	Troposphere	Ground-level
Regularity	Regular	Irregular
# measurements	High	Low
Spatial resolution	Low	High
Spatial changes between tasks	Yes	No
Starting samples	10	5

Table 4.2: Comparison of the two data sets used for air pollution monitoring.

Additionally, each data snapshot, corresponding to an optimisation task, has more measurements, up to 784, in the satellite case, as compared to up to 52 for the LAQN data. On the other hand, the spatial resolution is higher for the LAQN data, as the whole London area covers about 36 by 92 kilometres, while each data point for the satellite data covers about 7 by 7 kilometres. So the London area corresponds to a satellite image with 5 by 13 pixels. Finally, the locations stay mostly the same between data snapshots in the LAQN data, although some sensors go offline or come online, while the satellite data snapshots are of new locations.

The two data sets allow us to explore different things. The satellite data set was used first, and constitutes a simpler set of tasks. Because of the low resolution, and the high altitude, we can ignore effects of individual road intersections, fine-grained surface topography including buildings[4], and the distance to the nearest road. Additionally, the data set includes a range of pollution concentrations and differences in the ratios between the maximal and average concentrations, as can be seen in Fig. 4.4, where the satellite data have been split into subsets using the maximal pollution concentrations in each image. We use these concentrations as proxies for the difficulty of the optimisation task, allowing us to compare performance at different difficulties. Comparing the examples in Fig. 4.4, we see that the Strong example has the highest pollution concentration and the smoothest background level of pollution relative to the peak. It also has few local maxima of comparable magnitude to the global maximum. In contrast to this, the maximum concentration of the Weak example is more than a magnitude less than for the Strong example, and there is no clear

4. Roads surrounded by tall buildings can build up high pollution concentrations, due to the buildings stopping the pollutants dispersing. This effect is called the urban canyon effect (Pirjola et al., 2012).

global maximum. The easiest subset, Strong, allowed us to check the feasibility of the method, since it was unlikely to work on harder optimisation tasks if it did not solve easier ones. The LAQN data set was then brought in as a more realistic test of the method.

4.3.1 Satellite data

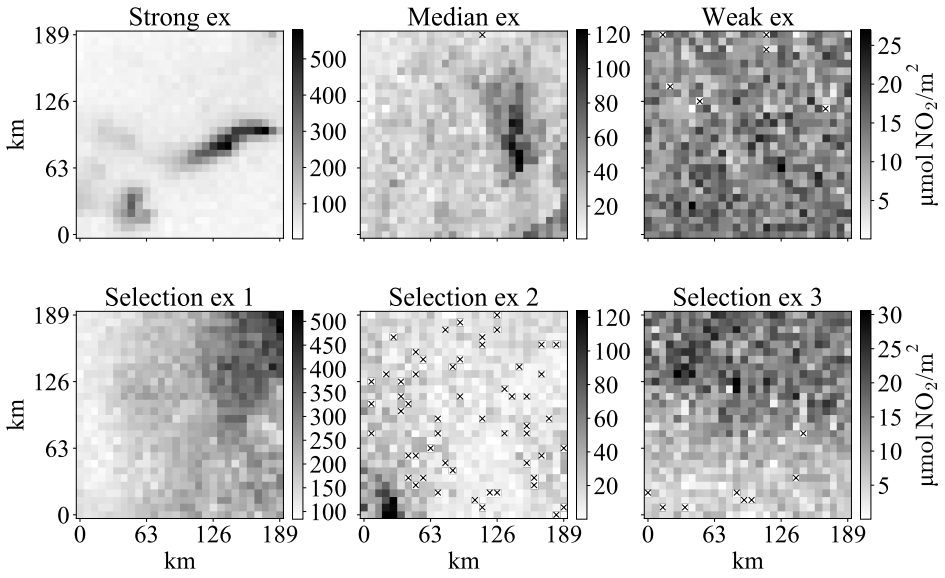


Figure 4.4: Example NO_2 concentration images from the Strong, Median, Weak and Selection subsets of the satellite data set. Crosses indicate missing data. Note the difference in scale between the subsets, and the pollution plumes from point sources in the Strong and Median examples. Such plumes were present in many images, particularly in the Strong subset, and are what the wind-informed kernels are meant to capture.

Data set origin: The data were collected by the TROPOspheric Measuring Instrument (TROPOMI) aboard the Sentinel-5P satellite (Copernicus Sentinel-5P (processed by ESA), 2018), but the subset we used was provided by Dr Douglas Finch of the School of Geosciences, University of Edinburgh. The subset was constructed to include more images with interesting pollution levels. The data set consists of images of 28x28 pixels, with each pixel giving the NO_2 concentration in mol/m^2 within a ca. 7x7km box on the ground reaching the upper troposphere. The images are from October and November 2018 and cover landmasses across the world.

Preprocessing: We removed duplicate images and images with more than 10 % missing values, resulting in 1083 images. We then log transformed the data, as the measurements — and air pollution concentrations in general (Cats & Holtslag, 1980) — are approximately log-normal. Therefore, the log transform allowed us to satisfy the assumption of Gaussian marginal and residual distributions in GP models. This is a simpler version of the transform recommended in Cowen-Rivers et al. (2022). The images were standardised using means and standard deviations from the associated tuning set[5]. The rationale for this was that the true mean and standard deviation for a deployment region would likely be unknown, but values from similar areas would be available.

Data split: After the basic filtering in the preprocessing step, the remaining images were split into subsets, based on their 'strength'. By strength we mean the maximum concentration of NO_2 present. The split was done as NO_2 concentrations varied greatly between images, leading to images with more or fewer local maxima of comparable concentrations to the global maximum. Due to unimpressive initial results we wanted to identify subsets with varying difficulty. In that way, the method could first be tested and improved on simpler optimisation tasks. The optimisation tasks in Strong are also easier to visually inspect and interpret, as there are clear maxima, as opposed to Weak where there are many local maxima and the images look 'noisy'. Therefore, Strong is better suited to debugging the optimisation, while Selection is suitable for collecting representative results.

Three subsets were created as follows: the images were ordered by the highest NO_2 concentration present, and the top, median and bottom 50 images put into sets called 'Strong', 'Median' and 'Weak', respectively. See Fig. 4.4 for examples and Fig. 4.5 for an illustration of the process. Three adjacent sets of 10 images each were created for tuning and data standardisation. Two final sets of 100 images, dubbed 'Selection' and 'Selection test', were created by picking every ninth of the remaining ordered images. By having the tuning sets for Strong, Median and Weak be adjacent to, and not within, the training sets we are introducing a potential bias. The motivation was that the pollution distributions might be different between the tuning and test data in a deployment setting, and so we wanted to make sure the method was robust against

5. Median tune was used for the Selection and Selection test subsets.

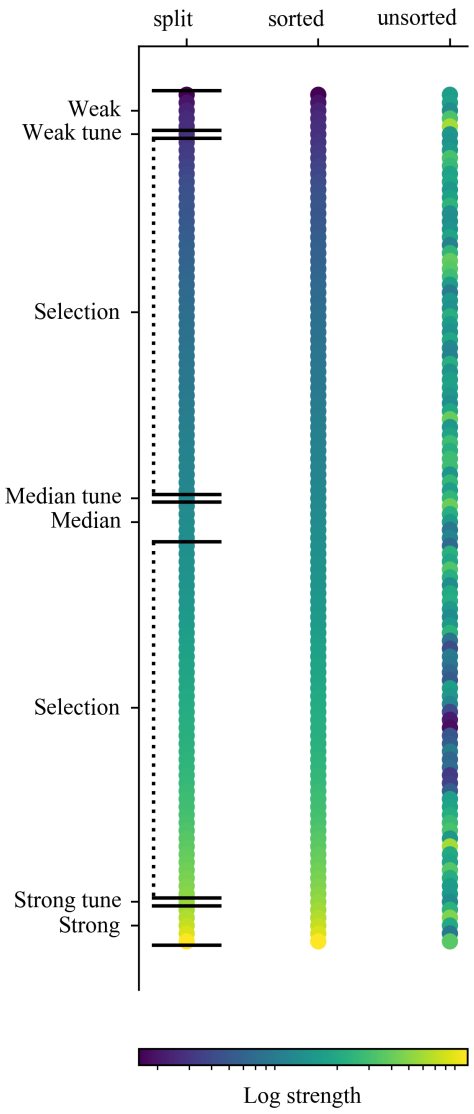


Figure 4.5: Illustration of the process of splitting the satellite data into subsets. Each circle represents 10 data snapshots or images. The rightmost column shows the unsorted data. Then in the middle column the data snapshots have been ordered by 'strength' (maximum NO_2 concentration), and in the left column this ordering has been used to split the data into subsets.

small shifts in the data distribution. The bias is larger for the Selection set as we use the Median tuning set (see Fig. 4.5). But this should make our methods perform worse compared to the baseline, not better, so again good results are an indication of robustness.

Subset	Description
Strong	50 strongest images.
Strong tune	10 images adjacent to Strong subset.
Median	50 images of median strength.
Median tune	10 images adjacent to Median subset.
Weak	50 weakest images.
Weak tune	10 images adjacent to Weak subset.
Selection	100 images not in above subsets selected representatively.
Selection test	100 images selected as for 'Selection', without overlap.
London test	365 days from 2016.
London tune	214 days from 2015.

Table 4.3: Overview of data subsets for the satellite and London data. The satellite subsets are described in Section 4.3.1 in the 'Data split' section.

The first three subsets were created to inspect performance on different degrees of difficulty, and the last two to test the overall performance. The eight subsets are summarised in Table 4.3. In the thesis we present results on the Strong, Selection and Selection test subsets.

Data examples: Examples from the subsets can be seen in Fig. 4.4.

Data characteristics: The data set has a large variation in the pollution concentrations present, from the pollution plumes[6] present in some of the Strong images to the seemingly white noise of some of the Weak images. In addition to the clearer visual image, the pollution concentrations present vary greatly, from over $500\,\mu\text{mol}/\text{m}^2$ to a maximum of about $25\,\mu\text{mol}/\text{m}^2$. A characteristic of this data set across the varying pollution levels is the quantity of measurements available. Some of the images have all 784 pixel values present, and all of the images used have at least 700 pixel values.

4.3.2 Ground-level London data

Data set origin: The London data were downloaded from the Imperial College London (1993) website, and are a subset of the total data available, which cover many more years, pollutants and site types. We restricted ourselves to the NO_2 readings of the 'roadside' sensors. We chose NO_2 and roadside as they were the combination with the most sensors in 2015, see Fig. 4.7, and optimisation tasks with more data are more suitable for showing the benefits of BO. Different pollutants and different

6. 'Clouds' or columns of pollution, see Finch, Palmer, and Zhang (2022).

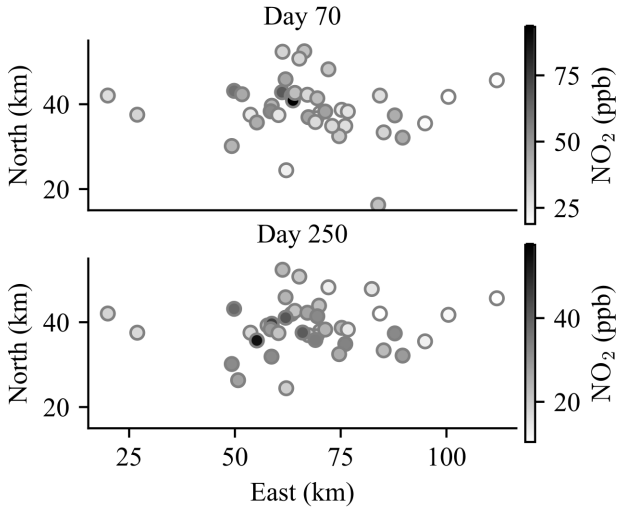


Figure 4.6: Examples of data snapshots from the LAQN data set. The distances are from the most south-westerly monitoring station, in Beech outside Alton in Hampshire. Note that not all sensors are available each day, the strong clustering in central London, and that the location of the maximum varies.

site classifications have different properties, so combining multiple site types or pollutants would have complicated the modelling. Using only the roadside sensors is a compromise between wanting more available sensors and not having to model the impact of site type on the measurements. While there are enough roadside sensors that we can model the pollution distribution, in the future the model should be extended to include all site types. There are 55 stations used in total, 51 in 2015 and 53 in 2016. Of these, 45 have sampling heights listed (Imperial College London, 1993)[7]. The sampling heights range from 1.5 to 4 metres.

Preprocessing: The values given are daily averages, and we discarded days with fewer than 40 sensors. We log-transformed and mean-centred the data[8].

Data split: We constructed a tuning set of 214 data snapshots from the 2015 data and a test set of 365 data snapshots from the 2016 data. This is summarised in Table 4.3. On average, the days in 2016 had measurements from a few more stations than the days in 2015. More information is available in Appendix B.1.

Data examples: Examples for the London data can be seen in Fig. 4.6.

---

7. Accessed online by looking at the web pages of individual stations, e.g. `https://www.londonair.org.uk/london/asp/publicdetails.asp?region=0&site=BT4&details=general` for the site with code BT4.
8. Each task was mean-centred using data from that task.

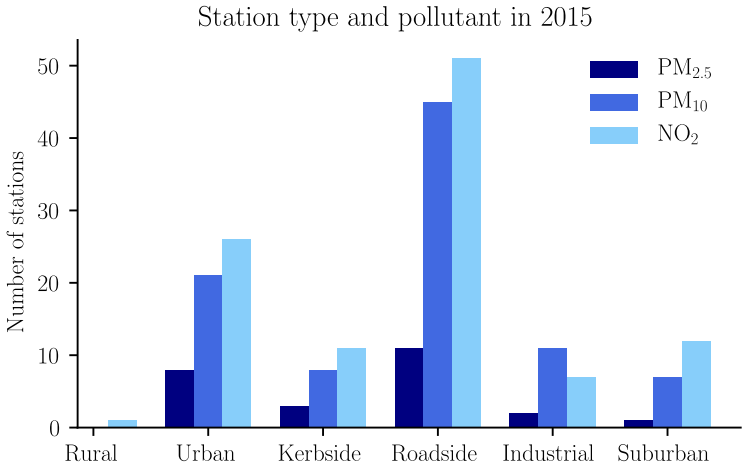


Figure 4.7: Comparison of LAQN sensors available for different station type and pollutant combinations. The combination roadside-NO_2 had the most stations available at 51.

Data characteristics: These data snapshots are quite different to the satellite data. There are many fewer samples per snapshot, and they are not spaced out evenly in a grid.

4.4 Gaussian process models

In this chapter, we use five different covariance functions, or kernels, for the Gaussian processes. They are based on the standard RBF kernel[9], Eq. (4.1), and a directed wind version, Eq. (4.2). The motivation for the wind kernel is that wind carries pollutants, e.g. there will be high pollution downwind from a coal plant (Duque & Gilraine, 2022). As a simplification, this kernel assumes that the pollution stays constant along the wind direction, and then we combine the wind kernel with the RBF kernel to get kernels that take into account both variation along, and orthogonal to, the wind direction.

The hyperparameters in Eqs. (4.1) and (4.2) are summarised in Table 4.5. $\boldsymbol{\tau}$ is the difference between the feature vectors of two locations, i.e. $\boldsymbol{\tau} = \boldsymbol{x}^{(1)} - \boldsymbol{x}^{(2)} = \begin{bmatrix} x^{(1)}_{\text{North}} - x^{(2)}_{\text{North}} \\ x^{(1)}_{\text{East}} - x^{(2)}_{\text{East}} \end{bmatrix}$. The part of $\boldsymbol{\tau}$ that is orthogonal to the wind direction γ is denoted $\tilde{\boldsymbol{\tau}}$, as illustrated in Fig. 4.8, and we can use the matrix $\boldsymbol{A}$, Eq. (4.3), to calculate $\boldsymbol{\tau}^T \boldsymbol{A} \boldsymbol{\tau} = \tilde{\boldsymbol{\tau}}^T \tilde{\boldsymbol{\tau}}$.

$$k_{\text{R},i}(\boldsymbol{\tau}) = \sigma^2_{\text{r},i} \exp\left(\frac{-\boldsymbol{\tau}^T \boldsymbol{\tau}}{l^2_{\text{r},i}}\right) \tag{4.1}$$

$$k_{\text{W},i}(\boldsymbol{\tau}) = \sigma^2_{\text{w},i} \exp\left(\frac{-\boldsymbol{\tau}^T \boldsymbol{A} \boldsymbol{\tau}}{l^2_{\text{w},i}}\right) \tag{4.2}$$

$$\boldsymbol{A} = \begin{bmatrix} \sin(\gamma)^2 & -\sin(\gamma)\cos(\gamma) \\ -\sin(\gamma)\cos(\gamma) & \cos(\gamma)^2 \end{bmatrix} \tag{4.3}$$

Based on these two base kernels, Eqs. (4.1) and (4.2), five kernels are used in this chapter: RBF, Product, Sum, RBF-RBF and RBF-Product, given in Eqs. (4.4) to (4.8). These kernels are sums and products of RBF kernel functions that are valid for all inputs, and are thus valid functions themselves (Rasmussen & Williams, 2006, pp. 83,95). At first the simpler kernels, RBF, Product and Sum, are used, but later RBF-RBF and RBF-Product are introduced, see Section 4.8.1. All five kernels are summarised in Table 4.4.

9. Eq. (4.1) differs from Eq. (2.4) by a factor of two in the denominator. This can be counteracted by scaling the lengthscale $l_{\text{r},i}$ by $\sqrt{2}$.

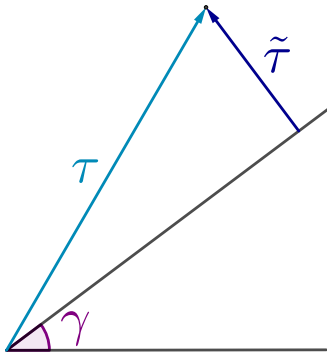


Figure 4.8: Wind kernel: $\boldsymbol{\tau}$ is the distance between two locations. $\tilde{\boldsymbol{\tau}}$ is the part of $\boldsymbol{\tau}$ orthogonal to the wind direction γ.

Kernel	Equation	Hyperparameters	# Hyperparameters
RBF	Eq. (4.4)	$\sigma^2_{\mathrm{r},1}$, $l_{\mathrm{r},1}$, σ^2_{n}	3
Product	Eq. (4.5)	$\sigma^2_{\mathrm{r},1}$, $l_{\mathrm{r},1}$, $l_{\mathrm{w},2}$, γ, σ^2_{n}	5
Sum	Eq. (4.6)	$\sigma^2_{\mathrm{r},1}$, $\sigma^2_{\mathrm{w},2}$, $l_{\mathrm{r},1}$, $l_{\mathrm{w},2}$, γ, σ^2_{n}	6
RBF-RBF	Eq. (4.7)	$\sigma^2_{\mathrm{r},1}$, $\sigma^2_{\mathrm{r},2}$, $l_{\mathrm{r},1}$, $l_{\mathrm{r},2}$, σ^2_{n}	5
RBF-Product	Eq. (4.8)	$\sigma^2_{\mathrm{r},1}$, $\sigma^2_{\mathrm{r},2}$, $l_{\mathrm{r},1}$, $l_{\mathrm{r},2}$, $l_{\mathrm{w},3}$, γ, σ^2_{n}	7

Table 4.4: Summary of the hyperparameters of each kernel.

The three simpler kernels are visualised in Fig. 4.9. In the top row the covariances are plotted as a function of distance in the x_1 and x_2 dimensions. The RBF kernel is radially symmetric, while both the Sum and Product kernels have elongated covariance along the wind direction γ. The Sum kernel also has a weaker covariance radiating out, while the Product kernel is more dominated by the wind direction. The bottom row shows example images sampled from the kernels. For Sum and Product we see that the example images show the expected directional structure.

$$k_{\mathrm{RBF}}(\boldsymbol{\tau}) = k_{\mathrm{R},1}(\boldsymbol{\tau}) \tag{4.4}$$

$$k_{\mathrm{Product}}(\boldsymbol{\tau}) = k_{\mathrm{R},1}(\boldsymbol{\tau}) k_{\mathrm{W},2}(\boldsymbol{\tau}) \tag{4.5}$$

$$k_{\mathrm{Sum}}(\boldsymbol{\tau}) = k_{\mathrm{R},1}(\boldsymbol{\tau}) + k_{\mathrm{W},2}(\boldsymbol{\tau}) \tag{4.6}$$

$$k_{\mathrm{RBF-RBF}}(\boldsymbol{\tau}) = k_{\mathrm{R},1}(\boldsymbol{\tau}) + k_{\mathrm{R},2}(\boldsymbol{\tau}) \tag{4.7}$$

$$k_{\mathrm{RBF-Product}}(\boldsymbol{\tau}) = k_{\mathrm{R},1}(\boldsymbol{\tau}) + k_{\mathrm{R},2}(\boldsymbol{\tau}) k_{\mathrm{W},3}(\boldsymbol{\tau}) \tag{4.8}$$

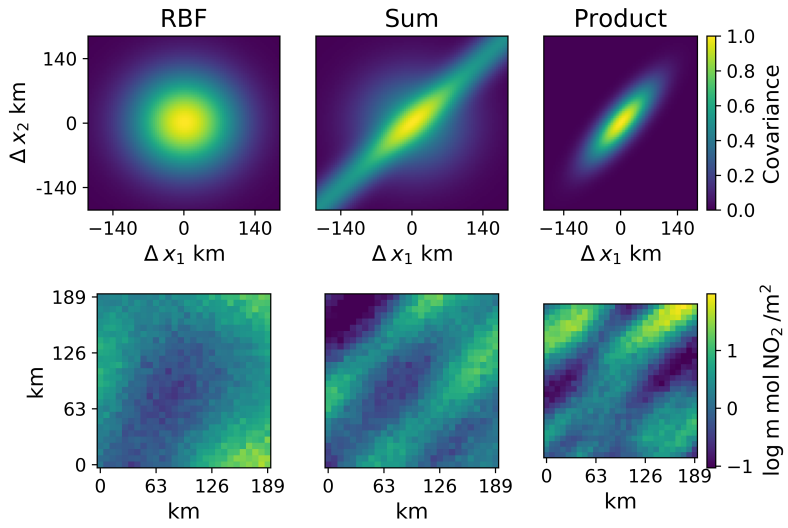
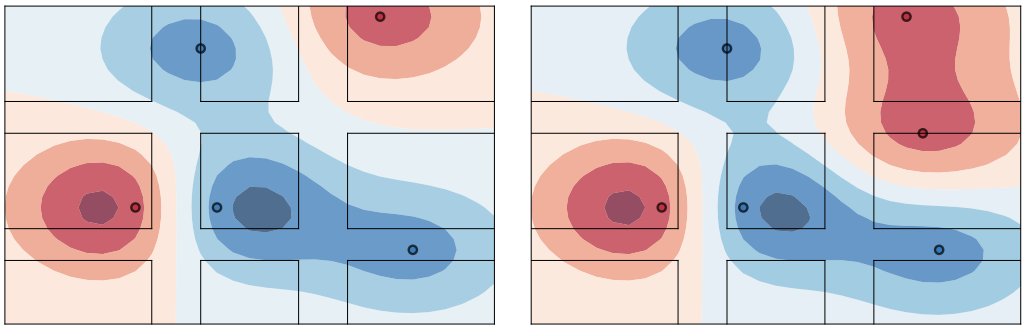


Figure 4.9: Visualisation of the kernels from Eqs. (4.4) to (4.6). The top row shows covariance as a function of distance, and the bottom row shows example images sampled from the kernels. We used $\sigma^2_{\mathrm{w},i} = 0.5$, $l_{\mathrm{r},i} = 105$, $l_{\mathrm{w},i} = 35$, $\gamma = \pi/4$, $\sigma^2_{\mathrm{n}} = 0.01$. For Sum $\sigma^2_{\mathrm{r},i} = 0.5$, while for RBF and Product $\sigma^2_{\mathrm{r},i} = 1$.

Hyperparameter	Definition
$l_{\mathrm{r},i}$,$l_{\mathrm{w},i}$	lengthscales
$\sigma^2_{\mathrm{r},i}$,$\sigma^2_{\mathrm{w},i}$	signal variances
γ	wind direction
σ^2_{n}	noise parameter

Table 4.5: Overview of hyperparameters in the GP kernels.

Fig. 4.10 illustrates one of the fundamental simplifying assumptions we adopt: that the detailed topology of the city landscape does not need to be modelled explicitly. We expect the presence of a building between two points to hinder the movement of pollutants between these two points. However, we assume that we can obtain a reasonably accurate model even without accounting for this effect directly. The covariance functions can somewhat account for this effect through the choice of a shorter lengthscale to indicate that all points further away are less correlated.



(a) Five samples. **(b)** Six samples.

Figure 4.10: Visualisation of a Gaussian process model in an urban grid setting. From (a) to (b) one additional sample has been added. Red indicates high pollution concentrations and blue low concentrations.

4.5 Experimental set-up

4.5.1 Problem statement

There are many possible applications within air pollution monitoring. We could estimate pollution densities within cities, where and how often legal limits are exceeded, or population-weighted exposures. Instead, we have chosen to focus on the problem of finding the pollution maximum in a given area. This is better suited to the data we have from London: the network has not been set up at random, and we cannot access pollution levels at arbitrary locations. So we would not be able to calculate an unbiased error estimate of the pollution density, limiting the usefulness of our experiments.

Similarly, estimating how often legal limits are exceeded in different parts of a city, or the population-weighted exposure, is difficult with the available data. To estimate the number of exceedances we would first need to estimate pollution densities within the area, with the above difficulties. Then we need to combine this with estimating the probabilities of being below or above the limit, introducing added complexity. For the population-weighted exposure we would need to either equip individuals with portable sensors, or combine a dense monitoring network with footfall data, e.g. how many people walk down each street daily. Both of these data sources would be hard to find, and would have privacy implications.

Finding the pollution maximum is a simpler problem than mapping out the pollution throughout the area, because we can disregard areas with low pollution. Solving this simpler problem is useful as a proof of concept, since an accurate estimate of the location and magnitude of the maximum is needed for a good pollution map. It is also useful directly, as higher pollution has a greater health impact, so finding the maximum both identifies the area of greatest concern, and puts a bound on how bad the other regions in an area can be.

4.5.2 Metrics

For evaluation, we compare the estimated maximiser $\hat{\boldsymbol{x}}$ and its true concentration $\hat{y}$, with the true maximiser $\boldsymbol{x}^*$ and maximum y^*. We use the subscript p to denote the p^{th} data snapshot, and subscript i to denote the i^{th} iteration of the optimisation. The two metrics we use are the maximum ratio R_i, Eq. (4.9), and the maximiser distance D_i, Eq. (4.10). They are summarised in Table 4.6. When reporting results we **bold** values where the mean $\pm$ one standard error overlap with the mean $\mp$ one standard error for the best method. So for the ratio metric (ideally 1) we find the method with the highest mean - one standard error, and bold the values of all methods for which the mean + one standard error is greater than the highest mean - one standard error. For the distance metric (ideally 0), it is the other way round.

Maximum ratio
$$R_i = \frac{1}{P}\sum_{p=1}^{P} \hat{y}_{p,i}/y_p^* \tag{4.9}$$

Maximiser distance
$$D_i = \frac{1}{P}\sum_{p=1}^{P} ||\hat{\boldsymbol{x}}_{p,i} - \boldsymbol{x}_p^*||_2 \tag{4.10}$$

The metrics are calculated at each iteration step, and the different methods are compared by observing how their performance on the metric evolves with iteration steps. The maximum ratio allows us to see how high the pollution is in the chosen location compared to the maximum, while the distance metric tells us how far we are from the true maximiser. The metrics are calculated on the preprocessed data, i.e. after the log transform and standardisation / mean-centring. Since we consider P different data snapshots, we average the performance over them.

Metric	Equation	Ideally	Domain
Maximum ratio	Eq. (4.9)	1	$-\infty$ to 1
Maximiser distance	Eq. (4.10)	0	0 to ∞

Table 4.6: Summary of the metrics used.

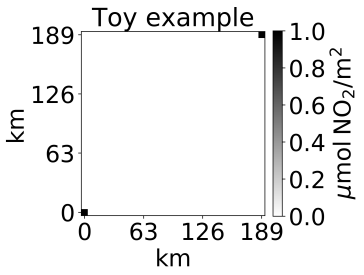


Figure 4.11: Toy example of when the distance metric can be misleading. The two points in the bottom left and top right have near identical pollution concentrations.

We want to use both metrics together, because either one of them on its own does not give us the full picture. For example, the distance metric can be very high when there are two very similar maxima far away from each other, such as in Fig. 4.11. If only the marginally lower maximum is found then this metric will score badly. On the other hand, the ratio metric does not tell us how far we are from the true maximiser. So if we want to know whether we have really found the correct maximiser, it is useful to also consider the distance metric.

4.5.3 Baselines

We use a random baseline for comparison. At each iteration, a random location is chosen as the next location to sample. This gives an entirely unguided exploration of the search space, but means that the search cannot be hampered by an overly confident model. As the results of random sampling are highly variable, the process is repeated 100 times for each data snapshot, and the average taken of the resulting metrics.

We also consider non-repeating random sampling for the London data. This is because there are few enough locations to sample that the exploration is noticeably faster when thus limited. For the satellite data there are enough locations that it makes little difference.

4.5.4 Experiments outline

We run three sets of experiments. We start with some initial experiments in Section 4.6, which do not involve learning hyperparameter priors. Then, in the experiments in Section 4.7 we learn a simple prior term from the tuning data. The final experiments, in Section 4.8, present a new transfer learning method using MCMC sampling to generate a prior.

4.6 Initial experiments

These initial experiments show the need for informative hyperparameter priors. In these experiments, the tuning sets are not consulted, instead, either no priors or very uninformative priors are used. We run two versions: 'single initialisation' only uses a single hyperparameter initialisation, while 'multiple initialisations' uses 100 initialisations. Having 100 different initialisations presupposes having a distribution to draw them from, and for that reason we introduce a very broad prior which is not extracted from the tuning data.

- Single initialisation (SI): No priors. The hyperparameters were initialised to 1, and optimised a single time.
- Multiple initialisations (MI): Very broad priors used for hyperparameter initialisations. 100 hyperparameter initialisations were drawn from the prior and optimised.

4.6.1 Methodology

Acquisition function: Upper confidence bound (UCB), Eq. (2.8), with $\beta = 1$ (Srinivas et al., 2012).

Kernels: The Product, RBF and Sum kernels, Eqs. (4.4) to (4.6).

Noise treatment: The noise parameter σ_{n}^2 was tuned in the same way as the other hyperparameters.

Hyperparameter	Low	High
γ	0	π
σ_{n}^2	1e-10	1e4
$\sigma_{\mathrm{r}}^2, \sigma_{\mathrm{w}}^2$	1e-6	1e3
$l_{\mathrm{r}}, l_{\mathrm{w}}$	1e-4	1e4

Table 4.7: Loose hyperparameter bounds used for hyperparameter initialisations in MI version of the initial experiments.

Priors: SI does not use priors. The following relates to MI. Aside from γ, the hyperparameter initialisation is done[10] by taking a log transform of the bounds, sampling uniformly in the interval and transforming back from the log transform. For γ the sampling is done uniformly without a log transform. The bounds are shown in Table 4.7. For γ we only consider values between 0 and π, because we do not differentiate between movement downwind and upwind. Therefore, there is no difference between γ and $\gamma + \pi$.

Model tuning: The model tuning consisted of maximising the log likelihood in Eq. (2.6).

4.6.2 Results

SI and MI were compared on the Strong subset. The results are shown in Fig. 4.12 and Table 4.8.

4.6.3 Discussion

Inspecting Fig. 4.12 and Table 4.8, there is clearly scope for improvement. Despite this data subset having been extracted to be simpler, and to have smoother and clearer maxima, making it more suitable for BO, the performance flattens out for both metrics. And for SI the random baseline catches up on the maximum ratio metric by 100 iterations while for MI it is about to. On the distance metric none of the BO models manage to outperform random sampling.

10. Except for the first initialisation which is set to 1 as in SI.

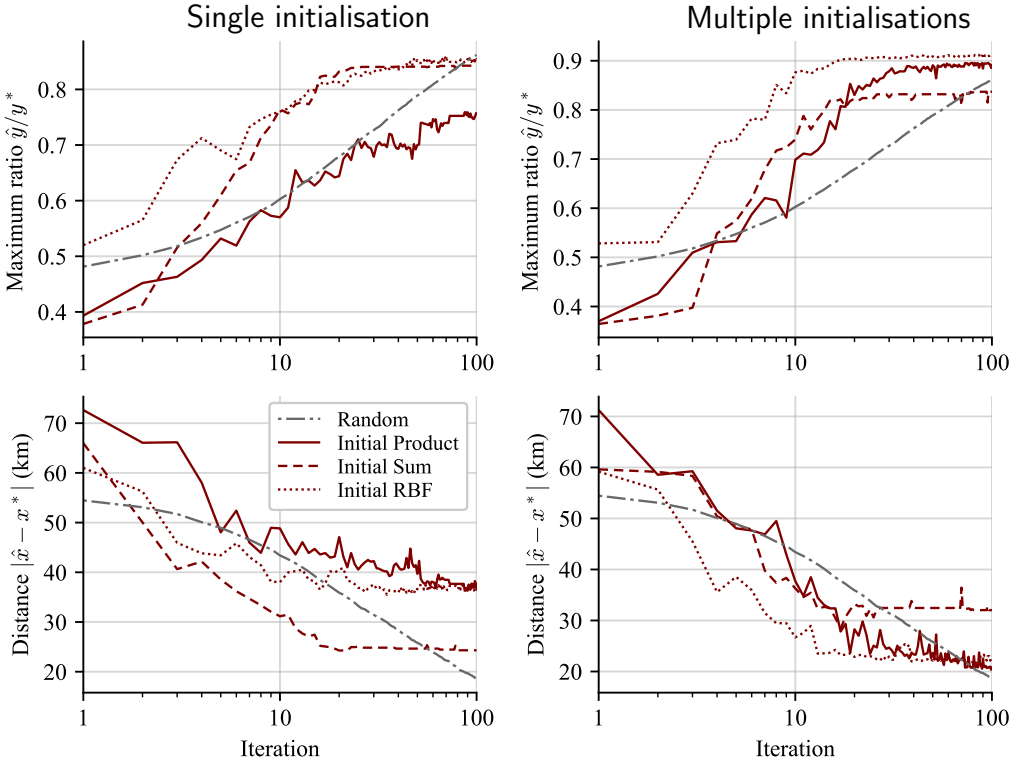


Figure 4.12: Initial experiments: comparison of version and kernel performance on the Strong subset. SI on the left and MI on the right. We see that MI, with multiple initialisations, does better, but both versions suffer from the curve flattening, and are only able to learn slightly faster than random sampling on the Strong subset, which is the simplest.

By introducing the coarse hyperparameter bounds in MI, we were able to improve performance, and based on that developed the hypothesis that better hyperparameter treatment would lead to better performance. This is shown next in Section 4.7 and even more so in Section 4.8. MI was included to explore the possibility that the problem was merely a matter of the models finding local optima, and so the optimisation tasks could be solved by a better hyperparameter optimisation algorithm instead of using priors. But even when initialising 100 times we still did not see great performance.

Maximum ratio	RBF SI 0.85 (0.83–0.88)	Sum SI 0.84 (0.81–0.88)	Product SI 0.76 (0.73–0.79)
	RBF MI **0.91 (0.89–0.93)**	Sum MI 0.84 (0.80–0.87)	Product MI **0.89 (0.87–0.92)**
	Random 0.86 (0.86–0.86)		
Distance (km)	RBF SI 36.46 (28.84–44.08)	Sum SI 24.31 (19.45–29.17)	Product SI 36.30 (30.51–42.10)
	RBF MI **22.96 (18.24–27.69)**	Sum MI 32.03 (25.55–38.51)	Product MI **20.94 (16.13–25.76)**
	Random **18.49 (18.14–18.85)**		

Table 4.8: Initial experiments: confidence intervals for the maximum ratio and distance metrics at the final iteration for the Strong subset. We show means $\pm$ one standard error. The best values are shown in **bold**.

4.7 Simple prior experiments

In these experiments we learned simple priors from the tuning data. This led to improved performance over the initial experiments, but for the Selection subset the performance is not better than random sampling. The simple prior experiments were presented in Hellan et al. (2020).

4.7.1 Methodology

The basic methodology remained the same as for Section 4.6. Here, the differences to the initial and sampling approaches are highlighted.

Acquisition function: UCB, Eq. (2.8), as for the initial experiments. We also tried an entropy-based acquisition function (Hernández-Lobato et al., 2014) and other values of β without seeing marked improvement.

Kernels: The Product, RBF and Sum kernels, Eqs. (4.4) to (4.6), as for the initial experiments.

Noise treatment: The noise parameter σ_{n}^2 was tuned in the same way as the other hyperparameters, again as for the initial experiments.

Priors: We used a uniform prior for γ. For the other hyperparameters the tuning sets were used to design simple priors. For the Strong subset the Strong tune subset was used, and for the Selection subset the Median tune subset. The idea behind the simple priors was to do a thorough search for the best settings for each optimisation task in the tuning data, and then use these as a basis for the priors. We computed priors separately for each of the GP kernels, i.e. Product, RBF and Sum.

To do so, we maximised the log marginal likelihood, Eq. (2.6), for each of the tuning images. We used 200 hyperparameter initialisations for each image, drawn from a wide distribution of possible values. For each, the gradient-based optimisation was run for up to 100 iterations. We then picked the set of hyperparameters that maximised the log marginal likelihood for each image, resulting in ten sets of hyperparameters.

Having obtained these ten sets of hyperparameters, we transformed them into prior distributions which could be included in the hyperparameter fitting for the test optimisation tasks. We used the maximum likelihood estimator with Bessel's correction to fit to a log normal distribution, expressed through the parameters $\boldsymbol{\mu}_{\mathrm{p}}$ and $\boldsymbol{\sigma}_{\mathrm{p}}$, each of dimension G_{nc}, where G_{nc} is the number of hyperparameters aside from γ. The log normal distribution $\mathcal{N}(\log(\boldsymbol{\theta}); \boldsymbol{\mu}_{\mathrm{p}}, \boldsymbol{\sigma}_{\mathrm{p}}^2 \boldsymbol{\mathcal{I}})$ was used as the prior.

Model tuning: As for the initial experiments we optimised the log likelihood, but added a prior term: $\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) + \log p(\boldsymbol{\theta})$. To avoid only finding local optima we ran the optimisation using 100 hyperparameter initialisations drawn from the prior.

4.7.2 Results

The models were tested on the Strong and Selection subsets of the satellite data. The results are shown in Fig. 4.13. The confidence intervals are shown in Table 4.9 (maximum ratio) and Table 4.10 (maximiser distance) together with those for Section 4.8.

4.7.3 Discussion

Inspecting Fig. 4.13, we easily spot the difference in difficulty between the Strong and Selection subsets. The random baseline reaches about 0.86 of the true maximum after 100 iterations on the Strong subset, compared to about 0.78 for the Selection subset. Similarly, the mean distance after 100 iterations is about 18km for the Strong subset and about 37km for the Selection subset. For the Strong subset the simple

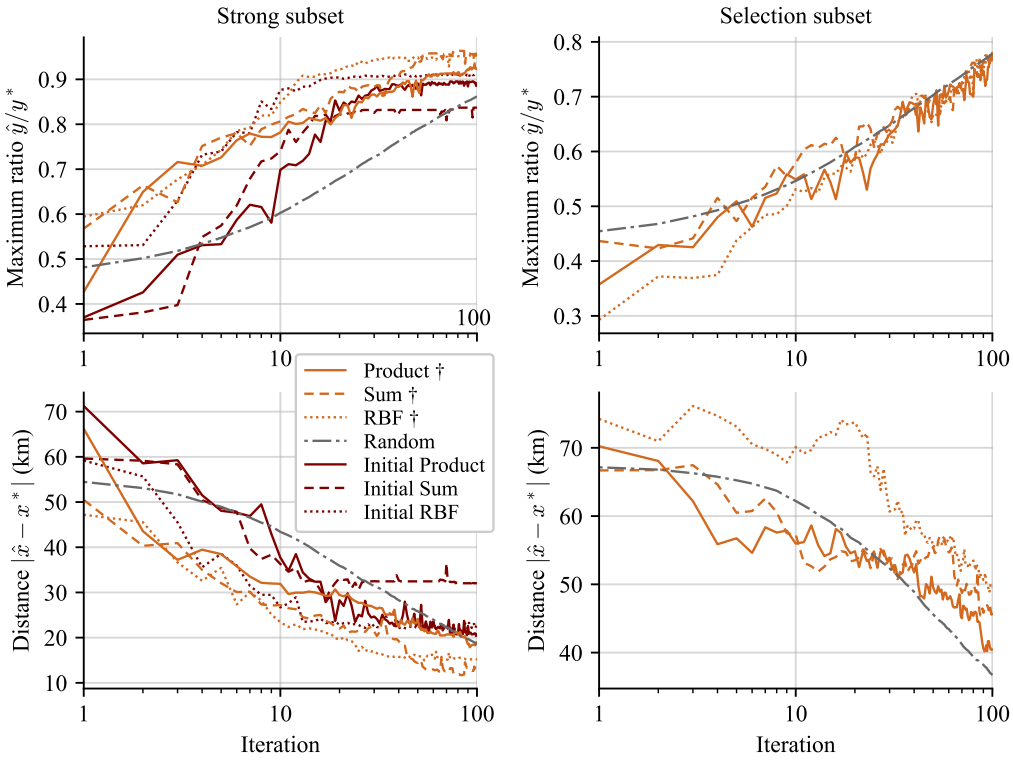


Figure 4.13: Simple prior experiments: comparison of kernel performance on Strong and Selection subsets. For the Strong subset we also include the MI results from Section 4.6. The Sum kernel reaches slightly better values than the RBF kernel on the Strong subset, and the Product kernel outperforms the other kernels on the distance metric on the Selection subset. † indicates results using simple priors.

prior models beat the random baseline on both metrics, but for the Selection subset the performance is more or less the same for the ratio metric and actually worse for the distance metric. Comparing the initial models and the simple prior models, the latter are clearly better. We conclude that the simple priors have led to improved performance on the Strong subset, but more work is needed to solve the optimisation tasks in the Selection subset.

4.8 Sampled prior experiments

The results using simple priors in Section 4.7 motivated an improved approach. We modelled the data set using a hierarchical model and used MCMC to generate samples for the hyperparameter priors. Problem-appropriate hyperparameter settings are essential for Bayesian optimisation to be efficient. Too large a lengthscale $l_{\mathrm{r},i}$ and the process is assumed to vary too slowly and relevant locations are not explored. Too

small a lengthscale and resources are wasted checking locations that are unlikely to be high and provide little new information. These pitfalls can be avoided using a prior that is informed by relevant data from other contexts. The sampled prior experiments were presented in Hellan et al. (2022).

4.8.1 Methodology

Several changes were made from the simple prior approach discussed in Section 4.7. These are visualised in Fig. 4.14 and discussed further below. We made multiple changes, despite this complicating direct comparison of the prior approaches, because the focus in this chapter was on demonstrating the usefulness of Bayesian optimisation to placing pollution sensors. Later, in Chapter 5, we will perform further experiments so different prior approaches can be compared directly. A major change was the adoption of the hierarchical structure shown in Fig. 4.15. It allows the hyperparameters of the different data snapshots to be related through the prior distribution $\boldsymbol{\eta}$. The methodology detailed here is relevant for this chapter: in Chapter 5 the method is developed further, and the implementation simplified, especially for the prior generating step.

Acquisition function: Expected improvement, Eq. (2.9), (Jones et al., 1998). EI was found to perform better than UCB, which motivated the change.

Kernels: The Sum, RBF-RBF and RBF-Product kernels, Eqs. (4.6) to (4.8). This change was made because we wanted the kernels to be able to model both slowly varying and faster varying signals.

Noise treatment: The noise parameter was clamped to a small value. This change was made to align the modelling with the evaluation metrics. When evaluating, the measured values were taken to be ground truth. If the model instead interpreted an isolated high value as measurement error and hence not the real maximum, then it would score badly using our metrics.

Priors: The priors were generated using MCMC and the hierarchical structure in Fig. 4.15, by collecting joint samples of $\boldsymbol{\eta}$ and $\{\boldsymbol{\theta}_1 \cdots \boldsymbol{\theta}_N\}$. Then the samples of $\boldsymbol{\eta}$ can be used to generate candidate samples of $\boldsymbol{\theta}_j$ as a prior for the model tuning in the BO loop. The lowest level of the hierarchical model is the data snapshots, which are the only variables that are observed directly. GP models are fitted to the data snapshots, and the hyperparameters $\boldsymbol{\theta}$ of the GP models constitute the middle layer of the model. The top level is the parameters defining

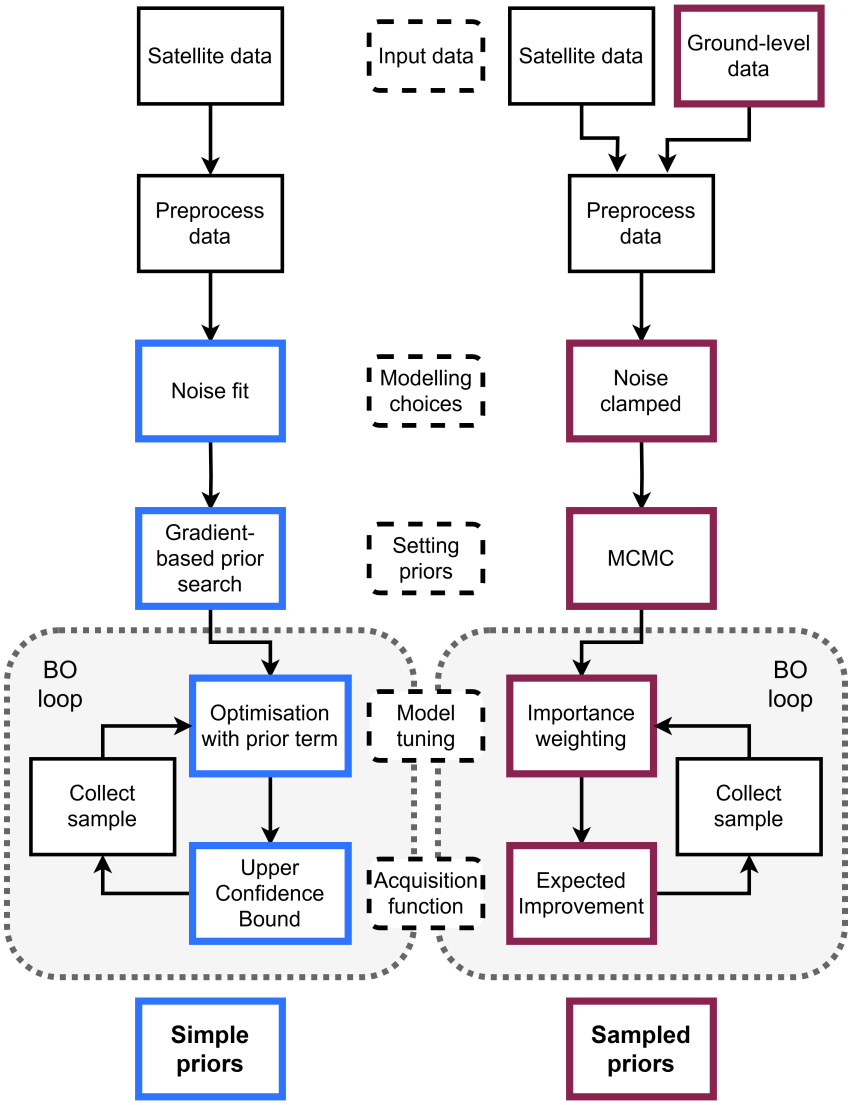


Figure 4.14: Diagram of differences between the model used with simple priors (left) and sampled priors (right). The coloured boxes show where changes have been made. Only one of the satellite and ground-level data sets is used at a time, the figure is meant to show that the sampled prior approach is evaluated on more data sets.

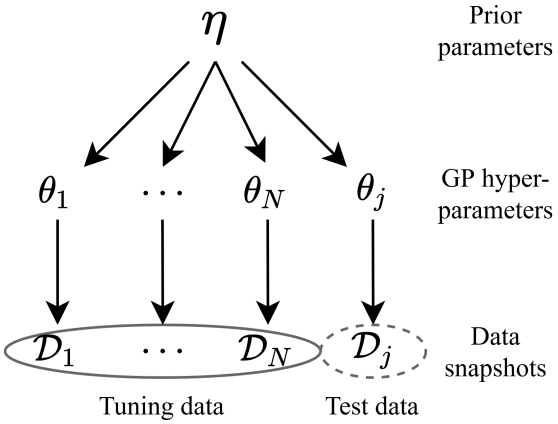


Figure 4.15: Visualisation of the hierarchical structure adopted for the sampled priors. The bottom level is the observed data $\boldsymbol{\mathcal{D}}$. It is modelled using GPs, which are defined by their hyperparameters $\boldsymbol{\theta}$. The distribution of the GP hyperparameters is captured by $\boldsymbol{\eta}$. The hyperparameters $\boldsymbol{\theta}$ are independent given $\boldsymbol{\eta}$.

the distribution of the GP hyperparameters. This can be expressed mathematically as $p(\boldsymbol{\eta}, \boldsymbol{\theta}, \boldsymbol{\mathcal{D}}) = p(\boldsymbol{\eta})p(\boldsymbol{\theta}|\boldsymbol{\eta})p(\boldsymbol{\mathcal{D}}|\boldsymbol{\theta}) = p(\boldsymbol{\eta}) \prod_n p(\boldsymbol{\theta}_n|\boldsymbol{\eta})p(\boldsymbol{\mathcal{D}}_n|\boldsymbol{\theta}_n)$. The structure reflects the assumption that the data snapshots are generated by distinct but similar underlying processes. Therefore, the tuning hyperparameters $(\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_N)$ cannot be used directly, but their distribution $\boldsymbol{\eta}$ can be. We assume the hyperparameters have a gamma distribution, i.e. $p(\boldsymbol{\theta}_n|\boldsymbol{\eta}) = \Gamma(\boldsymbol{\theta}_n; \boldsymbol{\psi}, \boldsymbol{\phi})$. The gamma distribution is a flexible distribution for positive, real-valued variables. It is commonly used for GP hyperparameters, e.g. by Balandat et al. (2020).

The MCMC sampling is done according to Algorithm 1. It is a coupled sampling based on Metropolis-Hastings (Chib & Greenberg, 1995), where we alternate between sampling $\boldsymbol{\theta}$ given $(\boldsymbol{\eta}, \boldsymbol{\mathcal{D}})$ and sampling $\boldsymbol{\eta}$ given $\boldsymbol{\theta}$. We initially tried resampling all $\boldsymbol{\theta}$ values jointly and all $\boldsymbol{\eta}$ values jointly, but found that this gave a very low acceptance ratio. We therefore changed to resampling each model hyperparameter separately. More details on the sampling is given in Appendix B.4. However, the sampling is simplified in Chapter 5, so the details become redundant.

Having used MCMC to generate samples of $\boldsymbol{\eta}$, we sample new values $\boldsymbol{\theta}_h$ from them, which we use in the model tuning. We discard the first 200 samples, and then randomly pick $H = 100$ of the remaining $\boldsymbol{\eta}$ samples, and use them to generate H candidate hyperparameter values $\boldsymbol{\theta}_h$.

Algorithm 1 Sampling $(\boldsymbol{\eta}, \boldsymbol{\theta})$

Initialise $\boldsymbol{\eta}^{(0)}$, $\boldsymbol{\theta}^{(0)}$ (Appendix B.4.1)
for $h = 1 \ldots \mathsf{H}_{\mathsf{MCMC}}$ **do**
    — *Sample* $\boldsymbol{\theta}^{(h)}$ ————————
    for $n = 1 \ldots \mathsf{N}$ **do**
        for $g = 1 \ldots \mathsf{G}$ **do**
            # Create a vector of $\boldsymbol{\theta}_n^{(h)}$ *except* $\theta_{n,g}^{(h)}$
            $\boldsymbol{\theta}_{n,\not g}^{(h)} \Leftarrow \left\{\theta_{n,1}^{(h)}, \ldots, \theta_{n,g-1}^{(h)}, \theta_{n,g+1}^{(h-1)}, \ldots, \theta_{n,G}^{(h-1)}\right\}$
            $\theta_{n,g}^{(h)} \Leftarrow \mathrm{sample}_\theta\left(\boldsymbol{\eta}^{(h-1)}, \boldsymbol{\theta}_{n,\not g}^{(h)}, \theta_{n,g}^{(h-1)}\right)$ (Appendix B.4.2)
        end for
        $\boldsymbol{\theta}_n^{(h)} \Leftarrow \left[\theta_{n,1}^{(h)}, \ldots, \theta_{n,G}^{(h)}\right]$
    end for
    $\boldsymbol{\theta}^{(h)} \Leftarrow \left[\boldsymbol{\theta}_1^{(h)}, \ldots, \boldsymbol{\theta}_N^{(h)}\right]$
    — *Sample* $\boldsymbol{\eta}^{(h)}$ ————————
    $\boldsymbol{\psi}_{:,0}^{(h)} \Leftarrow \boldsymbol{\psi}_{:}^{(h-1)}$, $\boldsymbol{\phi}_{:,0}^{(h)} \Leftarrow \boldsymbol{\phi}_{:}^{(h-1)}$
    for b $= 1 \ldots$ B **do**
        for $g = 1 \ldots G_{\mathrm{nc}}$ **do**
            $\psi_{g,b}^{(h)} \Leftarrow \mathrm{sample}_\eta\left(\boldsymbol{\theta}^{(h)}, \phi_{g,b-1}^{(h)}, \psi_{g,b-1}^{(h)}\right)$ (Appendix B.4.3)
            $\phi_{g,b}^{(h)} \Leftarrow \mathrm{sample}_\eta\left(\boldsymbol{\theta}^{(h)}, \psi_{g,b}^{(h)}, \phi_{g,b-1}^{(h)}\right)$ (Appendix B.4.3)
        end for
    end for
    $\boldsymbol{\eta}^{(h)} \Leftarrow \left[\psi_{1,B}^{(h)}, \phi_{1,B}^{(h)}, \ldots \psi_{G_{\mathrm{nc}},B}^{(h)}, \phi_{G_{\mathrm{nc}},B}^{(h)}\right]$ @ $\boldsymbol{\eta}_{G_{\mathrm{nc}}:}^{(0)}$
    Store sample $(\boldsymbol{\eta}^{(h)}, \boldsymbol{\theta}^{(h)})$
end for
return $\mathsf{H}_{\mathsf{MCMC}}$ samples of $(\boldsymbol{\eta}, \boldsymbol{\theta})$

@ denotes list concatenation. ψ is the shape parameter of the Γ distribution, ϕ the scale parameter.

Variable definitions:

- H_{MCMC}: Number of MCMC samples
- N: Number of tuning data snapshots
- G: Number of GP hyperparameters
- G_{nc}: Number of GP hyperparameters with non-constant $\boldsymbol{\eta}$
- B: Number of times to sample $\boldsymbol{\eta}$ per $\boldsymbol{\theta}$ sample. We used $B = 10$

The $\boldsymbol{\eta}$ parameters for the directional hyperparameter γ and the noise hyperparameter are constant.

Model tuning: For the BO loop we want to calculate the expectation of the acquisition function on data snapshot $\boldsymbol{\mathcal{D}}_j$ at point $\boldsymbol{x}$ given the distribution of GP hyperparameters. Our acquisition function, EI (Eq. (2.9)), presupposes a single set of hyperparameter values $\boldsymbol{\theta}$, so we need a way to combine the different hyperparameter samples $\boldsymbol{\theta}_h$. We use importance weighting to do so. We write $a_{j,i,h} = \alpha_{\mathrm{EI}}(\boldsymbol{x}; \boldsymbol{\mathcal{D}}_j^{:i}, \boldsymbol{\theta}_h)$ for the acquisition function evaluated on snapshot j at iteration i using hyperparameter sample $\boldsymbol{\theta}_h$ on the point $\boldsymbol{x}$. The quantity we want is

$$a_{j,i}(\boldsymbol{x}) = \mathop{\mathbb{E}}_{p(\boldsymbol{\theta}_h|\boldsymbol{\mathcal{D}}_{1:N}, \boldsymbol{\mathcal{D}}_j^{:i})} [a_{j,i,h}(\boldsymbol{x})] \tag{4.11}$$

where $\boldsymbol{\mathcal{D}}_{1:N}$ are the tuning data snapshots. It can be approximated as

$$a_{j,i}(\boldsymbol{x}) = \mathop{\mathbb{E}}_{p(\boldsymbol{\theta}_h|\boldsymbol{\eta})p(\boldsymbol{\mathcal{D}}_j^{:i}|\boldsymbol{\theta}_h)/Z} [a_{j,i,h}(\boldsymbol{x})] \tag{4.12}$$

$$\approx \frac{1}{H} \sum_{h=1}^{H} a_{j,i,h}(\boldsymbol{x}), \;\; \boldsymbol{\theta}_h \sim \frac{1}{Z} p(\boldsymbol{\theta}_h|\boldsymbol{\eta}) p(\boldsymbol{\mathcal{D}}_j^{:i}|\boldsymbol{\theta}_h) \tag{4.13}$$

$$\approx \frac{1}{W} \sum_{h=1}^{H} a_{j,i,h}(\boldsymbol{x}) p(\boldsymbol{\mathcal{D}}_j^{:i}|\boldsymbol{\theta}_h), \;\; \boldsymbol{\theta}_h \sim p(\boldsymbol{\theta}_h|\boldsymbol{\eta}) \tag{4.14}$$

where $W = \sum_{h=1}^{H} p(\boldsymbol{\mathcal{D}}_j^{:i}|\boldsymbol{\theta}_h)$ is the sum of the importance weights and Z is a normalising constant. In Eq. (4.12) $\boldsymbol{\eta}$ is used to summarise $\boldsymbol{\mathcal{D}}_{1:N}$. In Eq. (4.13) the expectation is approximated using Monte Carlo. In Eq. (4.14) importance weighting is used so samples can be generated from the prior $\boldsymbol{\eta}$, and weighted by their fit to the specific data snapshot $\boldsymbol{\mathcal{D}}_j^{:i}$. By taking MCMC samples from $p(\boldsymbol{\theta}_h|\boldsymbol{\eta})$ and not $p(\boldsymbol{\theta}_h|\boldsymbol{\eta})p(\boldsymbol{\mathcal{D}}_j^{:i}|\boldsymbol{\theta}_h)$ the computations are sped up greatly because we can approximate the distribution once and then reuse it at each iteration of the BO loop and for each test data snapshot.

4.8.2 Results

The models were evaluated on the satellite and London data. The results are shown in Tables 4.9 to 4.12 and Figs. 4.16 and 4.17. As for the Selection subset, we used the Median tuning set for the Selection test subset.

Strong	RBF-RBF **0.99 (0.99–1.00)**	Sum **1.00 (0.99–1.00)**	RBF-Product **0.99 (0.99–1.00)**
	RBF † 0.95 (0.94–0.97)	Sum † 0.96 (0.94–0.97)	Product † 0.92 (0.91–0.94)
Selection	RBF-RBF **0.83 (0.80–0.87)**	Sum **0.85 (0.82–0.89)**	RBF-Product **0.86 (0.83–0.89)**
	RBF † 0.78 (0.75–0.80)	Sum † 0.78 (0.76–0.80)	Product † 0.78 (0.75–0.81)

Table 4.9: Sampled prior experiments: confidence intervals for the maximum ratio at the final iteration for the satellite data. We show means ± one standard error. The best values are shown in **bold**. † indicates results using simple priors.

Strong	RBF-RBF **5.31 (2.38–8.23)**	Sum **2.99 (1.29–4.69)**	RBF-Product **3.41 (0.78–6.04)**
(km)	RBF † 15.20 (11.39–19.01)	Sum † 12.71 (8.45–16.98)	Product † 19.23 (14.51–23.94)
Selection	RBF-RBF 53.38 (44.46–62.30)	Sum **46.57 (36.96–56.18)**	RBF-Product **45.55 (36.13–54.98)**
(km)	RBF † 49.58 (44.14–55.01)	Sum † 46.23 (40.55–51.92)	Product † **40.28 (35.04–45.52)**

Table 4.10: Sampled prior experiments: confidence intervals for the maximiser distance at the final iteration for the satellite data. We show means ± one standard error. The best values are shown in **bold**. For Selection the random baseline (not listed, see Fig. 4.16) does best on average, but the standard error overlaps with some of the BO kernels. † indicates results using simple priors.

Execution times

The execution times are shown in Table 4.13. For MCMC, it is the time needed to collect the joint $(\boldsymbol{\theta}, \boldsymbol{\eta})$ samples. For BO, it is the average over data snapshots of the time needed for 100 iterations. For the London data, the MCMC takes between 21 and 41 hours, and the BO loops for 2 minutes. For the satellite data, the MCMC takes between 38 and 78 minutes and the BO loops for 44–52 minutes.

BO is slower for the satellite data due to there being more candidate sampling locations. MCMC is slower for the London data due to there being more tuning data. The MCMC time is less critical as it can be precomputed, while the BO time is encountered while placing sensors. The BO time is for 100 iterations, so even the

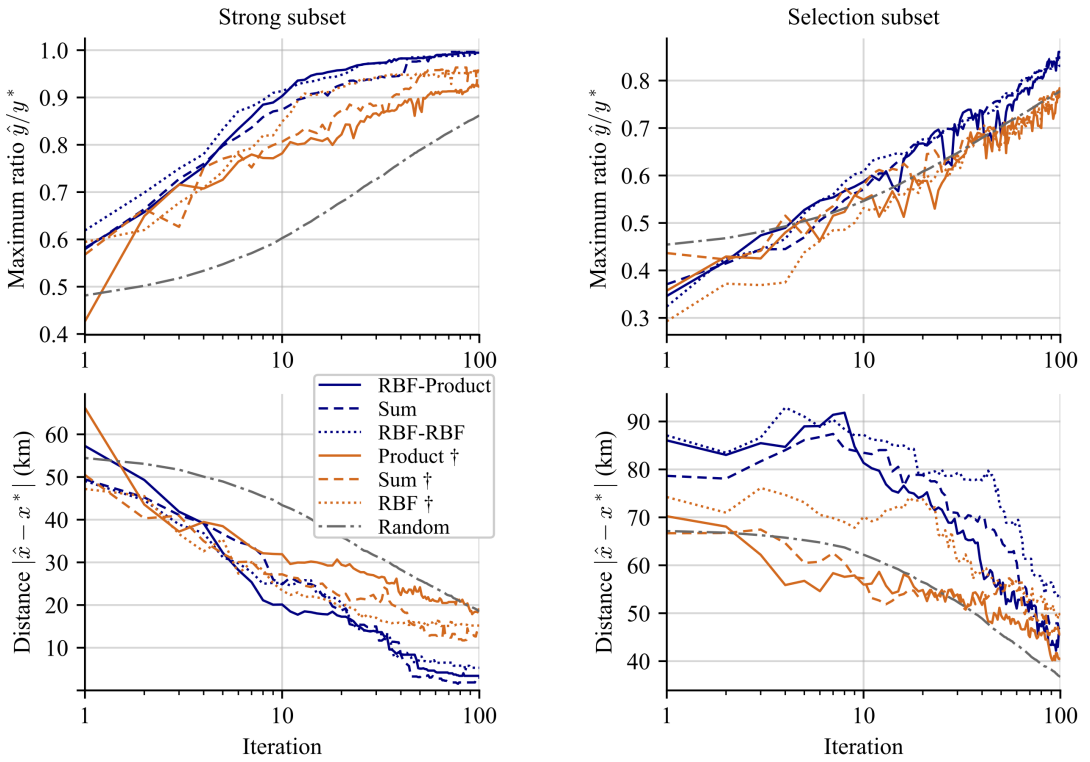


Figure 4.16: Sampled prior experiments: results on the Strong (left) and Selection (right) subsets of the satellite data. Sampled prior values in navy blue compared to simple prior values from Section 4.7 in orange marked with †. The baseline 'Random' is shown in grey.

slowest time, for the Sum kernel on the Selection subset — 52 minutes — averages to 31 seconds per iteration. This is negligible compared to the time required to seek out a location and install a sensor. The BO complexity is linear in the number of hyperparameter samples $\boldsymbol{\theta}_h$ from the prior. The complexity of the method is discussed further in Section 5.7.4.

4.8.3 Discussion

We start by examining the results from the sampled prior, first on the satellite data and then the London data. Then we use the London data to examine how the methods are exploring the search space looking for pollution maxima, and how we can interpret the model hyperparameters. More details for reproducibility are given in Appendix B.3.

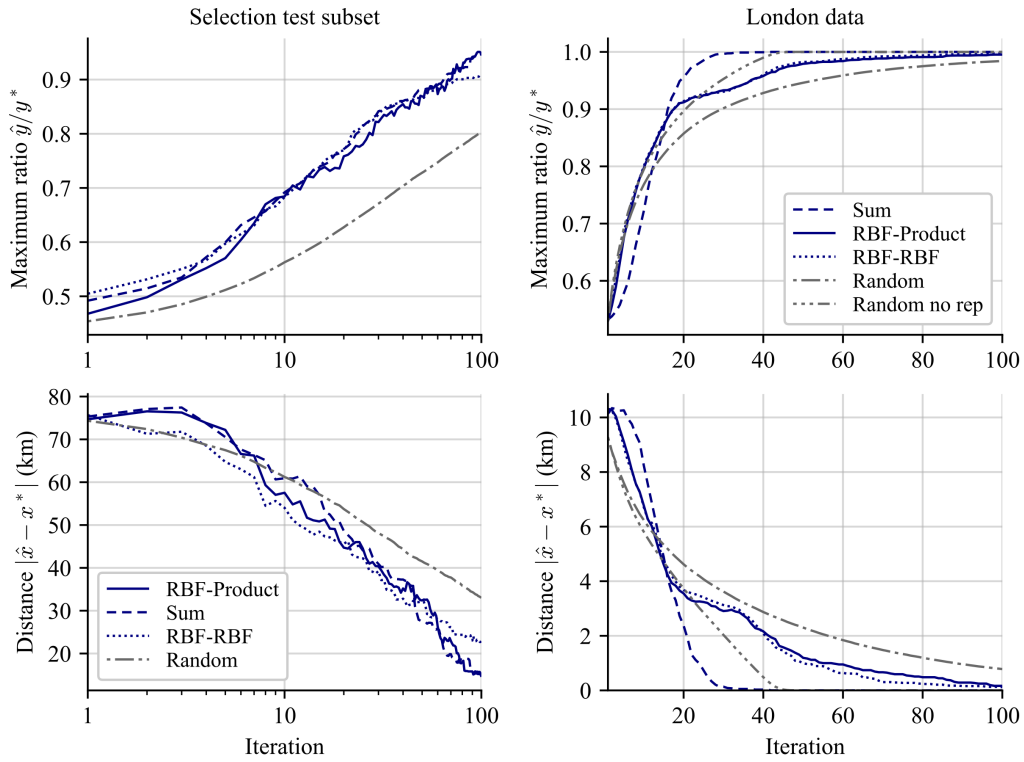


Figure 4.17: Sampled prior experiments: results on the Selection test subset of the satellite data (left) and on the London data (right). For the London data, the RBF-Product and RBF-RBF lines follow each other closely and sometimes overlap. The Sum model is slower to start learning, but then finds the maximum much faster than the other models and the baselines.

Satellite data

The results on the satellite data, see Figs. 4.16 and 4.17 and Tables 4.9 to 4.11, show that the sampled prior gives improved results compared to the simple prior, even on the challenging Selection subset. The exception is the distance metric on the Selection subset, where not even the sampled prior beats random sampling. As discussed in Section 4.5.2 and illustrated in Fig. 4.11, bad (high) scores for the distance metric can be a result of similar local optima in different parts of the images. By inspection of Fig. 4.4 we can see that the third Selection example has many local optima, and so would suffer from this. As multiple experiments had been run on the Selection subset by now, we also introduced a new subset, Selection test, to verify the results. As can be seen in Fig. 4.17, the sampled priors do better here than on the Selection subset, showing that the improved performance is not just a matter of overfitting. It seems that Selection test is a slightly easier subset than Selection, as the random sampling also does better.

	RBF-RBF	Sum	RBF-Product
Maximum	**0.91 (0.89–0.93)**	**0.95 (0.93–0.96)**	**0.94 (0.93–0.96)**
ratio	Random		
	0.80 (0.80–0.81)		
	RBF-RBF	Sum	RBF-Product
Distance	**22.02 (17.41–26.64)**	**15.08 (11.12–19.03)**	**14.53 (11.60–17.47)**
(km)	Random		
	33.02 (32.62–33.42)		

Table 4.11: Sampled prior experiments: confidence intervals for the maximum ratio and distance metrics at the final iteration for the Selection test subset. We show means ± one standard error. The best values are shown in **bold**.

	RBF-RBF	Sum	RBF-Product
Maximum	0.93 (0.93–0.94)	**1.00 (1.00–1.00)**	0.93 (0.93–0.94)
ratio	Random	Rand. no rep	
	0.90 (0.90–0.91)	0.96 (0.96–0.96)	
	RBF-RBF	Sum	RBF-Product
Distance	3.09 (2.84–3.33)	**0.09 (0.04–0.13)**	2.92 (2.69–3.15)
(km)	Random	Rand. no rep	
	3.54 (3.51–3.56)	1.88 (1.86–1.90)	

Table 4.12: Sampled prior experiments: confidence intervals for the maximum ratio and distance metrics at the 31st iteration for the London data. We show means ± one standard error. The best values are shown in **bold**.

London data

For the London data, see Fig. 4.17 and Table 4.12, we introduced a second random sampling baseline, this one without replacement. On the London data one of the GP models does much better than either baseline, but the other two perform worse. Why this happens is examined in 'Exploration', below. But it shows that BO can be much faster, even if care must be taken when choosing the models and priors. BO has the added benefit of providing models of the underlying process, with interpretable hyperparameters. An example of this is given in 'Interpretability', below.

	Kernel	Strong	Selection	London
MCMC	RBF-RBF	38 min	38 min	21 h
	Sum	56 min	56 min	30 h
	RBF-Product	71 min	78 min	41 h
BO	RBF-RBF	44 min	44 min	2 min
	Sum	50 min	52 min	2 min
	RBF-Product	49 min	49 min	2 min

Table 4.13: Run times for generating hyperparameter samples (MCMC) and locating maxima (BO), for the sampled prior experiments.

Exploration

The London results in Fig. 4.17 can be examined in light of the exploration-exploitation trade-off. We use an exploration score to do so, calculated as the minimum Euclidean distance between the new sample and any one of the already collected samples, $e_{i,j} = \min ||\boldsymbol{x}^{(i,j)} - \boldsymbol{x}^{(a,j)}||_2, \quad a < i$. This is done for each iteration i and data snapshot j and the average taken across the data snapshots; see Fig. 4.18. It shows that the reason the Sum model improves more slowly in the beginning could be that it emphasises exploration. Then, in the middle stage (5 to 20 iterations) it is able to learn faster, as it has already narrowed down what area to explore further, and manages to find the maximum faster. Conversely, and by chance, some of the random baselines' early iterations will be checking neighbouring points of those already seen, which can give small-scale improvement. The two other GP kernels start off with an exploration score between Sum and the random baselines, and the performance is also between them.

Interpretability

Part of our rationale for using comparatively simple GP kernels is so that the hyperparameters can be interpreted, and the distribution patterns of pollution in an area analysed, e.g. its smoothness and variability in space and time. This lets us perform 'at-a-glance' comparisons of different areas, discern whether extrema are more attributable to local sources or long-range patterns, and importantly lets domain experts assess the plausibility of a model's hyperparameters.

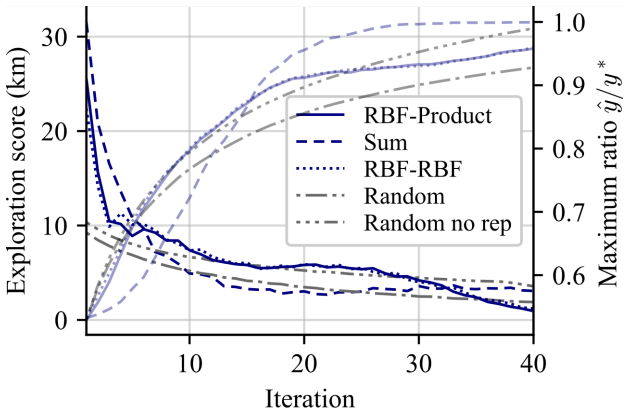


Figure 4.18: Exploration scores on the London data. Overlaid and paler are the ratio scores from Fig. 4.17. Note that Sum needs longer for the maximum ratio to increase, and has a higher exploration score at the start.

To illustrate, we consider the lengthscales $l_{\mathrm{r},i}$ of the RBF-RBF model in Eq. (4.7) on the London data. For simplicity, we take the mean of the samples generated for the prior, as that shows the trends better than the posterior on a single day. Converting the values to the standard RBF expression gives $l_{\mathrm{r},1} = 2.00$ km, $l_{\mathrm{r},2} = 241$ km, $\sigma^2_{\mathrm{r},1} = 2.05$ and $\sigma^2_{\mathrm{r},2} = 2.04$. This corresponds to one local component and one regional component contributing roughly equally to the overall variability of NO_2 concentrations. At a distance of 100 metres the expected correlation between NO_2 concentrations will be close to 1, driven by both local and long-range effects, whereas at a distance of 10 km it falls to 0.5, and local effects play a negligible role. The correlations are also visualised in Fig. 4.19. The LAQN provides a pollution map online at their website (Imperial College London, 1993). It shows pollution across all of London, but with more concentrated in central London radiating outwards and around Heathrow airport, as well as along major roads. This supports the modelling hypothesis of a local and a regional component.

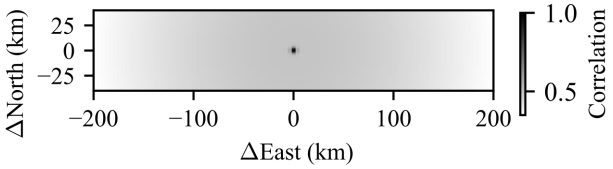


Figure 4.19: Expected correlation as function of distance for the mean of the RBF-RBF hyperparameter prior samples for the London data.

4.9 Limitations and future work

Given the relatively small difference in performance between Bayesian optimisation and the best random baseline on the London and Selection data, one might be tempted to ask whether the extra effort needed for BO is justified. In this application it is: when deployed, each extra iteration will mean one extra sensor having to be procured and installed, requiring both money and labour. In that context, using more resources when planning the placement is preferable. For other application domains, it becomes a trade-off between the cost of implementation and the cost of acquiring extra samples. If new samples can be acquired cheaply enough, then random sampling might be preferable.

4.9.1 Simplifying assumptions in our experiments

Throughout this chapter, several simplifying assumptions have been made. Firstly, although the London data are temporal, this has been simplified by constructing spatial optimisation tasks and treating them as independent of one another. Secondly, the measurements are treated as ground-truth readings. This simplifies the error metric calculations, but also makes irrelevant one of the advantages of probabilistically modelling the data, as this allows the measurement uncertainty to be modelled directly. And for the intended purpose of using low-cost sensors, handling uncertainty is important. Furthermore, we only considered the 'roadside' site type. This simplified our modelling, but meant we excluded available data. Additionally, we have only considered the problem of finding the pollution maximum, which is particularly suited to BO, and not other related monitoring problems such as developing a high-accuracy map throughout the area. We have also been limited to only considering measurement locations where there already is a sensor (London), or in the specified grid (Satellite). For an application, it would of course be more interesting to know whether the sensors in London are in the right place, but this is much harder to evaluate. To summarise, the simplifications are:

- Temporal dimension ignored.
- Readings treated as ground truth.
- Only roadside sensors considered.
- Only optimised for maxima.
- Only considered existing sensor locations.

4.9.2 Future work

Our motivation has been to show that BO can successfully be used for planning air pollution monitoring, which we have done. However, there are a number of factors we could change to make this planning more sophisticated, and improve the performance further.

- Myopic planning: We have not taken into account the number of samples left when planning the next location to sample. This can lead to sub-optimal decisions, but simplifies the problem.
- Single sensor planning: We have only deployed one sensor at a time, but you could have a situation where multiple locations need to be chosen at the same time.
- Kernel choices: We have stuck to relatively simple kernels, but there might be easy performance gains available by experimenting with kernel families (e.g. Matérn, Eq. (2.5)) and combinations, such as more complex compositional kernels to model correlations at different scales.
- Seasonal and weekly pattern modelling: Our kernels do not include any temporal data. By adding this we might be able to exploit seasonal and weekly patterns.
- Acquisition functions: The results indicated a benefit from early exploration, so the methods could be biased towards this by modifying the acquisition function. Berk, Nguyen, Gupta, Rana, and Venkatesh (2018) present a modification to expected improvement that increases early exploration without needing manual tuning.

There are two main extensions needed before deploying the method: temporal modelling and accommodating sensors with varying precision and reliability. Temporal modelling is important as pollution levels vary temporally, with some days bringing high pollution levels throughout a city. The temporal aspect can be avoided by only considering averages, but this ignores the legal limits on hourly averages (World Health Organization, 2006, pp. 174–175) and makes adding data from new sensors harder. The uncertainty of a pollution reading is important when using low-cost sensors to avoid misleading statements. Smith et al. (2019) propose a solution for sensor calibration, but do not discuss how to place the sensors. Unifying the active placement with the uncertainty treatment is necessary for successfully planning a network of low-cost sensors.

The performance of the learned priors should also be evaluated on tuning and test data from different cities, ideally using multiple combinations. Currently we are exploiting that air pollution in London in one year behaves similarly to air pollution in London the next year, so we can learn our model hyperparameters on the previous year. But in a deployment we should learn the hyperparameter distributions from multiple cities, and apply it to a new one. Additionally, experiments should be run with actual sensor placement, to avoid being limited to locations where domain experts have already placed sensors. These two changes to the evaluation will bring it much closer to showing whether the approach will work in a deployed setting. The reason this has not been done yet is that it requires large amounts of data, and access to a large number of pollution monitors.

4.10 Conclusion

Starting from no hyperparameter priors and progressively adopting more Bayesian approaches to modelling, we have shown that Bayesian optimisation with hierarchical models can be successfully applied to ground-level urban pollution data. Our final method learns distributions of hyperparameter priors, capturing the uncertainty of the prior as well as the variation between tasks. We also presented a pragmatic method for approximate inference when some of the work can be precomputed, which we believe can be useful in other applications. In Bayesian optimisation the GP tuning step needs to be repeated many more times than in standard regression, as new data points are added iteratively. Therefore, using standard MCMC on the full hierarchical model for each new data point would slow down the process. By using importance weighting in the BO tuning step the process can be sped up. This is even more useful in other applications where collecting samples is less time consuming than for pollution.

As expected, the models performed worse on the ground-level, London data set. This is due in part to the data snapshots having fewer available samples. Some exploration is needed before the model can usefully discern the areas of interest, and where the data are sparse compared to the local variations this is harder. The strength of BO lies in being able to exploit structure in data, so it works better for slowly varying processes. While the results show that the approach has promise, more evaluation is needed. In particular, due to data constraints, London was used both for developing

the prior and for evaluation. To test whether the prior can be constructed from other cities additional data needs to be used. It would also be beneficial to test the methods as they would be used, i.e. iteratively placing sensors, but this would be much more resource-intensive.

Air pollution monitoring is an interesting application of Bayesian optimisation, combining both important potential real-world impact and a challenging machine learning problem. In particular, we have used air pollution monitoring to explore the effect of hyperparameter priors. The prior learning method introduced in Section 4.8 is developed further in Chapter 5. There we provide a simpler implementation using NumPyro (Phan et al., 2019), making it more robust and easier to modify. We also compare the method to a range of other transfer learning methods, and put it in the context of the transfer learning literature.

5

Data-driven Prior Learning for Bayesian Optimisation

> *God does not care about our mathematical difficulties; He integrates empirically.*
>
> – Albert Einstein, as quoted by Leopold Infeld in *Quest: An Autobiography*

This chapter improves upon and analyses the sampled prior method developed in Chapter 4. Content from this chapter was presented at the NeurIPS 2023 Workshop on Adaptive Experimental Design and Active Learning in the Real World, in a paper also called Data-driven Prior Learning for Bayesian optimisation *(Hellan, Lucas, & Goddard, 2023b). A copy of this paper is included in Appendix A.5.*

5.1 Introduction

The previous chapter focused on the application of air pollution monitoring, and only briefly discussed the new method developed for transfer learning. In this chapter we change focus to examine the sampled prior method in more detail, and put it in the context of other transfer learning methods. We call the method Prior Learning for Bayesian Optimisation, or PLeBO.

As discussed in Chapter 2, most earlier transfer learning methods have worked by identifying (regions of) good input values to previous problems, i.e. doing the transfer in the input domain, which we call direct transfer. Instead, PLeBO performs the transfer in the Gaussian process hyperparameter domain, which we call prior transfer. This is a weaker type of transfer — we learn how the function varies, but not where it is optimal. But it also requires weaker assumptions. While direct transfer is based on the

assumption that the optima for different tasks are in similar locations, prior transfer only assumes that the different optimisation tasks have similar structures. Note that we call this category of methods *prior transfer*, and our method *prior learning* for Bayesian optimisation or PLeBO.

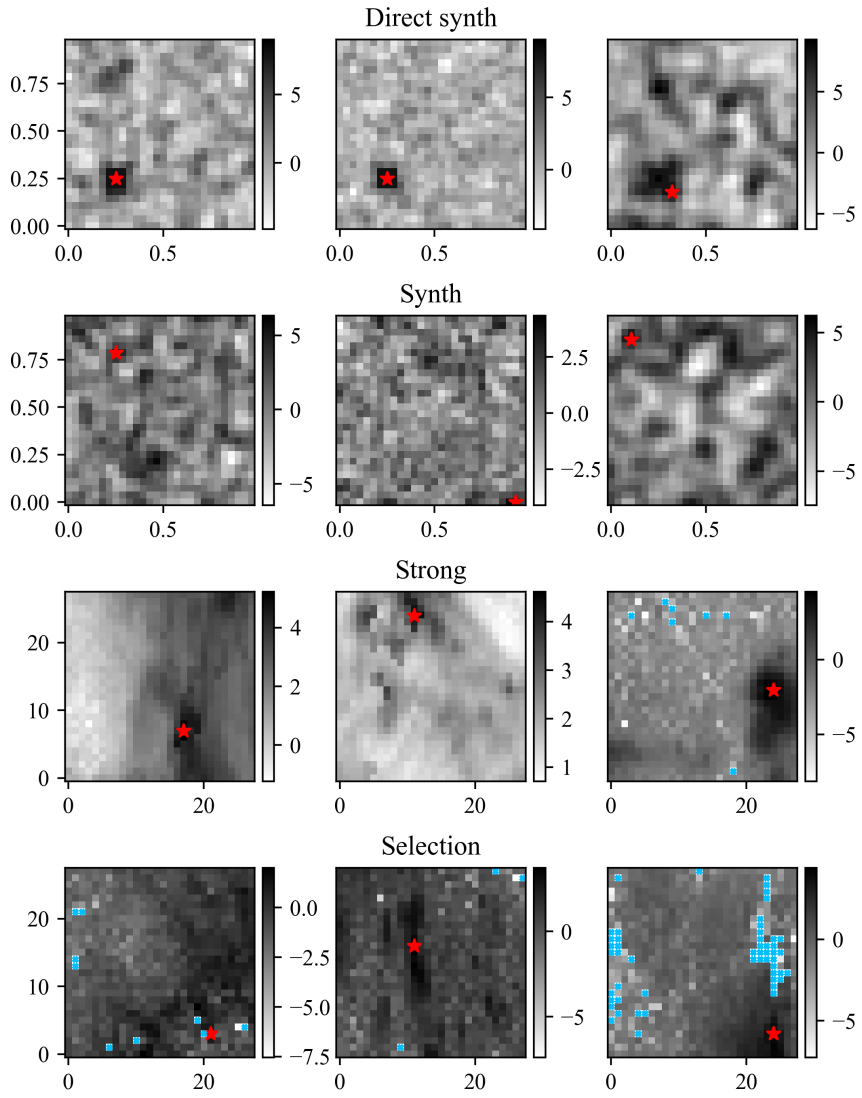


Figure 5.1: Example optimisation tasks. The top row shows synthetic tasks suitable for direct transfer. The next row shows synthetic tasks suitable for prior transfer. The bottom two rows show air pollution optimisation tasks from the satellite subsets in Chapter 4. Red stars indicate optimal points and small blue crosses missing values.

Fig. 5.1 shows synthetic tasks suitable for direct transfer in the top row, and for prior transfer in the next row. In the top row the optima are clustered around (0.25, 0.25), while in the second row the optima are dispersed but the optimisation landscapes have similar structure. The bottom two rows show satellite optimisation tasks from Chapter 4. As we can see, the optima are not clustered but dispersed as in the second row. Therefore, we expect prior transfer to work better than direct transfer for these tasks.

An alternative to using data to learn the priors is to use hand-crafted priors. Hand-crafting hyperparameter priors based on other optimisation tasks can also be seen as prior transfer, but it is useful to learn these priors automatically. Otherwise, we need to specify new priors manually for every hyperparameter each time we change the kernel. PLeBO does this by learning the hyperparameter prior given the tuning data, and then approximating the posterior given this prior. The basic method was developed in Chapter 4, but relied on a custom-built implementation of MCMC, did not analyse the quality of the developed priors, and did not compare to standard BO transfer learning baselines. In this chapter we fill this gap, and also improve upon the method. We compare PLeBO to a range of non-transfer, direct and prior transfer baselines on synthetic data and on the satellite data from Chapter 4 (Copernicus Sentinel-5P (processed by ESA), 2018).

5.2 Method description and improvements

In this section we describe the PLeBO method and highlight the differences to the closely related sampled prior method in Chapter 4. PLeBO was based on the sampled prior method, but we introduced some improvements, and it is this improved metod that we call PLeBO. The PLeBO method is an example of prior transfer, intended to work even with small amounts of tuning data. We assume a hierarchical structure where each optimisation task has its own set of GP hyperparameters $\boldsymbol{\theta}$, and these hyperparameters come from a shared distribution $\boldsymbol{\eta}$. The method has two stages: in the preprocessing step, we use MCMC to learn $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$ given observations $\boldsymbol{\mathcal{D}}$, and then sample new values of $\boldsymbol{\theta}$ based on $\boldsymbol{\eta}$ to generate a set of candidate hyperparameters $\boldsymbol{\theta}^{\text{cand}}$. In the optimisation phase we use importance weighting to fit these candidates to the observed test data $\boldsymbol{\mathcal{D}}_j$. This structure is illustrated in Fig. 5.2. The formal structure is shown in Eq. (5.1).

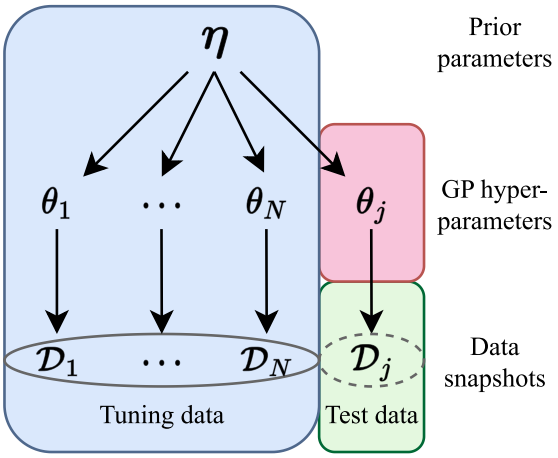


Figure 5.2: The hierarchical structure and inference used in the PLeBO method:
MCMC: Generate samples of $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$ given observations $\mathcal{D}$.
Sample candidate hyperparameters $\boldsymbol{\theta}^{\text{cand}}$ given the $\boldsymbol{\eta}$ samples.
Importance weighting: In the optimisation phase we weight candidates by their data likelihood.

$$p(\boldsymbol{\eta}, \boldsymbol{\theta}, \mathcal{D}) = p(\boldsymbol{\eta})p(\boldsymbol{\theta}|\boldsymbol{\eta})p(\mathcal{D}|\boldsymbol{\theta}) = p(\boldsymbol{\eta}) \prod_n p(\boldsymbol{\theta}_n|\boldsymbol{\eta})p(\mathcal{D}_n|\boldsymbol{\theta}_n) \tag{5.1}$$

In our experiments we use randomly sampled tuning data $\mathcal{D}$ in PLeBO, but $\mathcal{D}$ could also come from previous optimisation tasks. The modelling distributions can be chosen separately for each hyperparameter; we use gamma distributions $p(\boldsymbol{\theta}|\boldsymbol{\eta}) = \Gamma(\boldsymbol{\theta}; \boldsymbol{\eta})$. The gamma distribution is a flexible and continuous distribution for positive numbers (Murphy, 2022, p. 64). It is commonly used for GP hyperparameters, e.g. in Balandat et al. (2020); Cowen-Rivers et al. (2022); Müller et al. (2023). The choice of gamma distributions means we need to learn N sets of $\boldsymbol{\theta}$ and $2k$ parameters of $\boldsymbol{\eta}$, where k is the dimensionality of $\boldsymbol{\theta}$ and N is the number of tuning tasks. For simplicity, we restrict ourselves to two hyperparameters[1] in $\boldsymbol{\theta}$, so learn 4 parameters of $\boldsymbol{\eta}$. We also need to choose the number H of candidate samples; we use $H = 200$. In the cases we consider that provides a good approximation to taking the expectation given $\boldsymbol{\eta}$, without being prohibitively computationally expensive. For a new optimisation problem the value of H should be adapted based on the dimensionality of $\boldsymbol{\theta}$ and the number of modes in the hyperparameter landscape.

1. The two hyperparameters are the lengthscale and signal variance of the RBF kernel, see Section 5.4 for more details.

PLeBO is an approximation to jointly inferring $\boldsymbol{\theta}_j$ with $\boldsymbol{\eta}$ and $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N$. The joint inference would be very computationally intensive, as $\boldsymbol{\theta}_j$ depends on $\boldsymbol{\mathcal{D}}_j$, which increases with each optimisation step. So the full inference would have to be done at each step. As N increases — and assuming $\boldsymbol{\theta}_j$ and $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N$ come from the same distribution — $\boldsymbol{\eta}|\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N\}$ and $\boldsymbol{\eta}|\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N, \boldsymbol{\theta}_j\}$ should converge to the same distribution. Then provided a sufficiently high H we get an accurate estimate of $\boldsymbol{\theta}_j$ without redoing the inference at each optimisation step.

5.2.1 Preprocessing — learning the prior samples with MCMC

The first phase of the preprocessing is to use MCMC to approximate the distributions of $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$ given $\boldsymbol{\mathcal{D}}$. In Chapter 4 we used custom code for the MCMC sampling, using a variant of Metropolis-Hastings which resampled $\boldsymbol{\eta}$ and $\boldsymbol{\theta}$ one variable at a time, and which resampled $\boldsymbol{\eta}$ ten times for each resample of $\boldsymbol{\theta}$. This code had the disadvantage of not using the state-of-the-art in MCMC methods, and requiring a lot of custom code which would have to change if a different prior distribution than Gamma was wanted, or a different covariance function, making it harder to adapt for new projects. It also had several hyperparameters that needed to be specified, such as the width of the proposal distribution for each parameter in $\boldsymbol{\eta}$.

Therefore, the MCMC sampling code was rewritten using NumPyro (Phan et al., 2019). That greatly reduces the amount of custom code needed, and allows us to use the already implemented NUTS sampler (Hoffman & Gelman, 2014). Instead of implementing the sampling and acceptance/rejection steps, we simply define the relationship between each of the variables. This gives increased stability, makes further work easier, and makes it easier to change between inference methods.

Robustness

Not all of the MCMC runs converge, since exploration is made more difficult by regions with zero likelihood (within the available precision). Therefore, we improved upon the version of PLeBO in Chapter 4. We now use multiple runs of MCMC and postprocess the generated samples, discarding any sets of $(\boldsymbol{\eta}, \boldsymbol{\theta})$ for which the $\boldsymbol{\theta}$ samples have zero likelihood on the tuning data.

Sampling candidate values

Once we have the filtered $\boldsymbol{\eta}$ values, we choose H of them without replacement and sample one value of $\boldsymbol{\theta}$ from each of them. These are the candidate hyperparameter values $\boldsymbol{\theta}^{\text{cand}}$.

5.2.2 Optimising — using the prior samples

We use the candidate samples within the optimisation loop to calculate the acquisition function: $a_i(\boldsymbol{x}) \approx \frac{1}{W}\sum_{h=1}^{H} w_h\, a_i^{\theta}(\boldsymbol{x}, \boldsymbol{\theta}_h^{\text{cand}})$ where $w_h = p(\boldsymbol{\mathcal{D}}_j^{:i}|\boldsymbol{\theta}_h^{\text{cand}})$, $W = \sum_{h=1}^{H} w_h$ and $a_i^{\theta}(\cdot)$ is the base acquisition function, for which we use EI, see Eq. (2.9). i indexes the optimisation steps. At each optimisation step we calculate the likelihood of every hyperparameter candidate, and use it to weight the corresponding base acquisition function.

This approximates the expected value of the acquisition function given the tuning observations $\boldsymbol{\mathcal{D}}_{1:N}$ and test observations $\boldsymbol{\mathcal{D}}_j^{:i}$, as shown in Eqs. (5.2) and (5.3). An alternative would have been to approximate $\boldsymbol{\theta}^{\text{cand}}$ using MCMC directly at each iteration, but this would have been prohibitively computationally expensive[2]. The quality of the approximation can be controlled through H. Higher values give better approximations but also longer runtime. $\mathbb{E}_{p(\boldsymbol{\theta}_h^{\text{cand}}|\boldsymbol{\mathcal{D}}_{1:N}, \boldsymbol{\mathcal{D}}_j^{:i})}$ denotes taking the expectation over $p(\boldsymbol{\theta}_h^{\text{cand}}|\boldsymbol{\mathcal{D}}_{1:N}, \boldsymbol{\mathcal{D}}_j^{:i})$.

$$a_i(\boldsymbol{x}) = \mathbb{E}_{p(\boldsymbol{\theta}_h^{\text{cand}}|\boldsymbol{\mathcal{D}}_{1:N}, \boldsymbol{\mathcal{D}}_j^{:i})}[a_i^{\theta}(\boldsymbol{x}, \boldsymbol{\theta}_h^{\text{cand}})] \tag{5.2}$$

$$\approx \frac{1}{W}\sum_{h=1}^{H} w_h\, a_i^{\theta}(\boldsymbol{x}, \boldsymbol{\theta}_h^{\text{cand}}), \qquad \boldsymbol{\theta}_h^{\text{cand}} \sim p(\boldsymbol{\theta}_h^{\text{cand}}|\boldsymbol{\eta}) \tag{5.3}$$

5.3 Related work

We previously discussed direct transfer in Section 2.6.1 and existing versions of prior transfer in Section 2.6.2. Here we revisit the topic, and compare PLeBO explicitly with these methods.

2. We make some estimates in Section 5.7.4.

5.3.1 Ensembles

Similarly to Feurer et al. (2018), we use an ensemble of Gaussian processes. But while they use an ensemble of models using observations from the tuning $\boldsymbol{\mathcal{D}}_n$ and test $\boldsymbol{\mathcal{D}}_j$ tasks, see Eqs. (5.4) to (5.6), we use an ensemble of different models all using the test observations $\boldsymbol{\mathcal{D}}_j$, see Eq. (5.7).

$$\mu_{\text{joint}}(\boldsymbol{x}) = w_j \mu_j(\boldsymbol{x}; \boldsymbol{\mathcal{D}}_j) + \sum_{n=1}^{N} w_n \mu_n(\boldsymbol{x}; \boldsymbol{\mathcal{D}}_n) \tag{5.4}$$

$$\sigma^2_{\text{joint}}(\boldsymbol{x}) = w_j^2 \sigma_j^2(\boldsymbol{x}; \boldsymbol{\mathcal{D}}_j) + \sum_{n=1}^{N} w_n^2 \sigma_n^2(\boldsymbol{x}; \boldsymbol{\mathcal{D}}_n) \tag{5.5}$$

$$\alpha_{\text{RGPE}}(\boldsymbol{x}) = \alpha_{\text{base}}(\mu_{\text{joint}}(\boldsymbol{x}), \sigma^2_{\text{joint}}(\boldsymbol{x})) \tag{5.6}$$

$$\alpha_{\text{PLeBO}}(\boldsymbol{x}) = \sum_{h=1}^{H} w_h \alpha_{\text{base}}(\mu_h(\boldsymbol{x}; \boldsymbol{\mathcal{D}}_j), \sigma_h^2(\boldsymbol{x}; \boldsymbol{\mathcal{D}}_j)) \tag{5.7}$$

The weights used are also calculated differently. While they use generalisation performance to come up with the weights, we use the likelihood of the test data.

The expression in Eq. (5.7) looks very similar to the RGPE(MoGP) method in Feurer et al. (2022) (ranking-weighted Gaussian process ensemble — mixture-of-GPs), see Eq. (5.8).

$$\begin{aligned} \alpha_{\text{RGPE(MoGP)}}(\boldsymbol{x}) = {} & w_j \alpha_{\text{base}}(\mu_j(\boldsymbol{x}; \boldsymbol{\mathcal{D}}_j), \sigma_j^2(\boldsymbol{x}; \boldsymbol{\mathcal{D}}_j)) \\ & + \sum_{n=1}^{N} w_n \alpha_{\text{base}}(\mu_n(\boldsymbol{x}; \boldsymbol{\mathcal{D}}_n), \sigma_n^2(\boldsymbol{x}; \boldsymbol{\mathcal{D}}_n)) \end{aligned} \tag{5.8}$$

But while Eq. (5.8) and Eq. (5.7) both use weighted averages of the base acquisition function, the base models used — $(\mu_n(\boldsymbol{\mathcal{D}}_n), \sigma_n(\boldsymbol{\mathcal{D}}_n))$ and $(\mu_h(\boldsymbol{\mathcal{D}}_j), \sigma_h(\boldsymbol{\mathcal{D}}_j))$ — are trained on different data, and the weights are different.

The RGPE methods are direct transfer methods, while PLeBO does prior transfer and does not require the optima to be in similar locations. Additionally, the parameter H is not tied to the number of past tasks N, giving us more flexibility.

5.3.2 Deep kernels

The idea of learning the kernel from related data has been explored through deep kernels (Wilson, Hu, Salakhutdinov, & Xing, 2016), both within (Wistuba & Grabocka, 2021) and outwith (Patacchiola, Turner, Crowley, O'Boyle, & Storkey, 2020) the Bayesian optimisation paradigm. This is done by learning a feature transformation that converts the inputs before they are used in the standard kernels, a process which produces a more expressive kernel. Learning the hyperparameters, as done in PLeBO, can also be seen as a type of kernel learning, but in a much less expressive search space. The standard GP kernels make sense for the benchmarks we consider — we do not have large amounts of tuning data to learn feature transformations from, and expect points to be most correlated to those close to them. And the simpler kernels are more interpretable and easier to explain and adapt with domain experts.

5.3.3 Gaussian process hyperparameters

The idea of learning the GP hyperparameters from related data has been explored in Wang et al. (2022); Fan et al. (2022); Müller et al. (2023) and Corani et al. (2021). In Wang et al. (2022) a single set of hyperparameters $\boldsymbol{\theta}$ is learned to cover all of the related tasks, in contrast to in PLeBO where we assume a different set $\boldsymbol{\theta}_n$ for each task and learn a distribution $\boldsymbol{\eta}$ of the hyperparameters. Learning a single set of hyperparameters is easier, but assumes that the problems have very similar structures. Having $\boldsymbol{\theta}$ as a distribution of $\boldsymbol{\eta}$ is useful for modelling the uncertainty of $\boldsymbol{\theta}$, as well as the fact that we can have a range of $\boldsymbol{\theta}$ values across the tasks. We adopt this approach of using a single set of hyperparameters as the baseline Shared, see Section 5.4. Earlier work (Lawrence & Platt, 2004; Schwaighofer et al., 2004) has also used the assumption of shared GP kernels.

Fan et al. (2022) build on Wang et al. (2022), generalising it to variable-size search spaces. They assume all functions with the same search space share a single set of GP hyperparameters, but that the hyperparameters of different search spaces are related through a prior. For each search space, the hyperparameters are learned independently, by minimising the negative log likelihood. Then, the priors are fitted using the maximum likelihood estimator. Comparing to PLeBO, this means that each $\boldsymbol{\theta}_n$ is learned independently from $\boldsymbol{\mathcal{D}}_n$, and then the $\{\boldsymbol{\theta}_1 \dots \boldsymbol{\theta}_N\}$ are fixed and $\boldsymbol{\eta}$ learned.

In contrast, in PLeBO $\{\boldsymbol{\theta}_1 \dots \boldsymbol{\theta}_N\}$ and $\boldsymbol{\eta}$ are learned jointly from $\{\boldsymbol{\mathcal{D}}_1 \dots \boldsymbol{\mathcal{D}}_N\}$. This makes the preprocessing step in Fan et al. (2022) similar to that in the simple prior approach from Section 4.7. In the optimisation, the same acquisition function weighting as in PLeBO is used.

Fan et al. (2022) evaluate their method on a synthetic data set and HPO, using much more data than we do. While we use 10 tasks with 20 (synthetic) or 100 (air pollution) samples, so 200 or 1000 data points, they use 48000 data points for their synthetic data set and several million data points for the HPO data set (Arango, Jomaa, Wistuba, & Grabocka, 2021). For this amount of data, the MCMC approach in PLeBO would be computationally prohibitive, while for our amount of data the approach in Fan et al. (2022) would not extract all the available information.

As in PLeBO, Müller et al. (2023) learn priors for the GP hyperparameters. But there are key differences, both in how the priors are produced and used. Müller et al. learn the priors by randomly sampling candidate prior parameters, and using cross-validation to choose the best one. This contrasts to PLeBO where we use MCMC. And Müller et al. use the priors in the training of prior-data fitted networks. These are neural networks trained to predict the posterior predictive distribution of a GP. While the ways the priors are learned and used in PLeBO and Müller et al. (2023) are very different, the idea of using related data to learn the priors is present in both approaches.

In Corani et al. (2021) GP priors are developed to improve forecasting. They adopt a similar structure to that used in PLeBO (see Fig. 5.2), but use log-normal priors and learn them through variational inference instead of MCMC. They prioritise speed of model fitting, so use the priors to find a point estimate of the model hyperparameters, instead of weighting multiple candidate hyperparameters. Their approach of learning the prior, and then using it to fit a point estimate of the hyperparameters is similar to our baseline Gamma, see Section 5.4.

5.4 Experimental set-up

This section presents the experimental set-up, including the baselines used. In our experiments we use a synthetic benchmark and the satellite optimisation tasks from Chapter 4 (Copernicus Sentinel-5P (processed by ESA), 2018). In Section 5.5 the synthetic benchmark is introduced, and the results presented, including an analysis of the generated prior. In Section 5.6 the satellite air pollution benchmarks are recapped, and results presented. Discussion, including an analysis of the computational complexity of PLeBO, is given in Section 5.7. We discuss limitations and future work in Section 5.8.

To simplify analysis we use the RBF kernel (Eq. (2.4)) and limit ourselves to two hyperparameters: the lengthscale and the signal variance. Note that this is different to Chapter 4, where we used custom kernels and learned 4–5 hyperparameters. Our base metric is the best observed value at each iteration i, normalised by the max value of the task, $y^{(j)}_{\max}$: $r^{(j)}_i = \max \boldsymbol{Y}^{(j)}_{:N_{\text{start}}+i} / y^{(j)}_{\max}$ where $\boldsymbol{Y}^{(j)}_{:N_{\text{start}}+i}$ are the observations and N_{start} is the number of starting evaluations. When averaged over tasks, this metric is the same as the maximum ratio in Chapter 4. We assume negligible noise.

Methods

We run experiments comparing three types of methods. The first group has no transfer learning, and consists of random search as well as three versions of Bayesian optimisation. The next group is direct transfer and has two methods: one using a shared model and the other learning initial points to evaluate. Finally, we compare four methods for prior transfer. The first is our method, PLeBO. For the synthetic tasks we also consider a variant which has access to the correct hyperparameters for the new tasks, TruePLeBO. Next, we consider a variant Gamma which uses the mean gamma prior learned in PLeBO when fitting the hyperparameters using gradient-based methods. Finally, in Shared we use the same set of hyperparameters for all of the tasks, learned from gradient-based optimisation on the tuning tasks.

- No transfer
  - **RandomSearch:** Randomly select the inputs at each iteration.
  - **BoTorch:** Default settings from the BoTorch library, which include using a Matérn 5/2 kernel (Balandat et al., 2020).
  - **EI:** BO without transfer using expected improvement (Jones et al., 1998).
  - **UCB:** BO without transfer using upper confidence bound and setting the scaling parameter to 1 (Srinivas et al., 2012).

- Direct transfer
  - **DirectTrans:** Shared GP for past and new tasks. Capped at 100 past evaluations.
  - **Initial:** First evaluate good points from previous tasks, then do **EI**.
- Prior transfer
  - **PLeBO:** The new prior learning method detailed in Section 5.2.
  - **TruePLeBO:** Using EI with the true hyperparameters (only for synthetic data).
  - **Gamma:** Include the mean prior $\boldsymbol{\eta}$ from PLeBO when fitting the hyperparameters.
  - **Shared:** A single set of hyperparameters $\boldsymbol{\theta}$ learned from the tuning tasks (Wang et al., 2022).

5.5 Synthetic benchmark experiments

We introduce a synthetic benchmark that uses the same hierarchical structure as assumed in PLeBO, see Fig. 5.2. It allows us to evaluate the quality of the derived priors since we have access to the true values of $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$. We use GPs with RBF kernels and a known low level of noise. We define gamma distributions on the two hyperparameters: the lengthscale $l \sim \Gamma(5, 0.01)$, and the signal variance $\sigma_r^2 \sim \Gamma(2, 2)$. We use the shape, scale parameterisation of $\Gamma(\cdot)$. We generate a set of 10 tuning optimisation tasks and 100 test tasks. The tuning tasks have 20 evaluations each, and we start each test task with ten known evaluations. The tasks are discretised to match the satellite benchmarks. Examples of the synthetic optimisation tasks can be seen in the second row of Fig. 5.1.

Fig. 5.3 compares the performance of the different methods on the synthetic optimisation tasks. As this plot has a lot of lines, we also plot the performance compared to PLeBO in Fig. 5.4 (left). And we show the mean improvement over PLeBO in Fig. 5.5 (left). We see that TruePLeBO does the best, and is the only method which on average is better than PLeBO on the synthetic benchmark. However, it relies on knowing the true hyperparameters and so is not possible for real problems. It is included to upper bound how well we can do by learning the hyperparameters, and to validate the soundness of the approach. Comparing to the non-transfer baselines, PLeBO beats all the methods, but the difference to EI without transfer is smaller

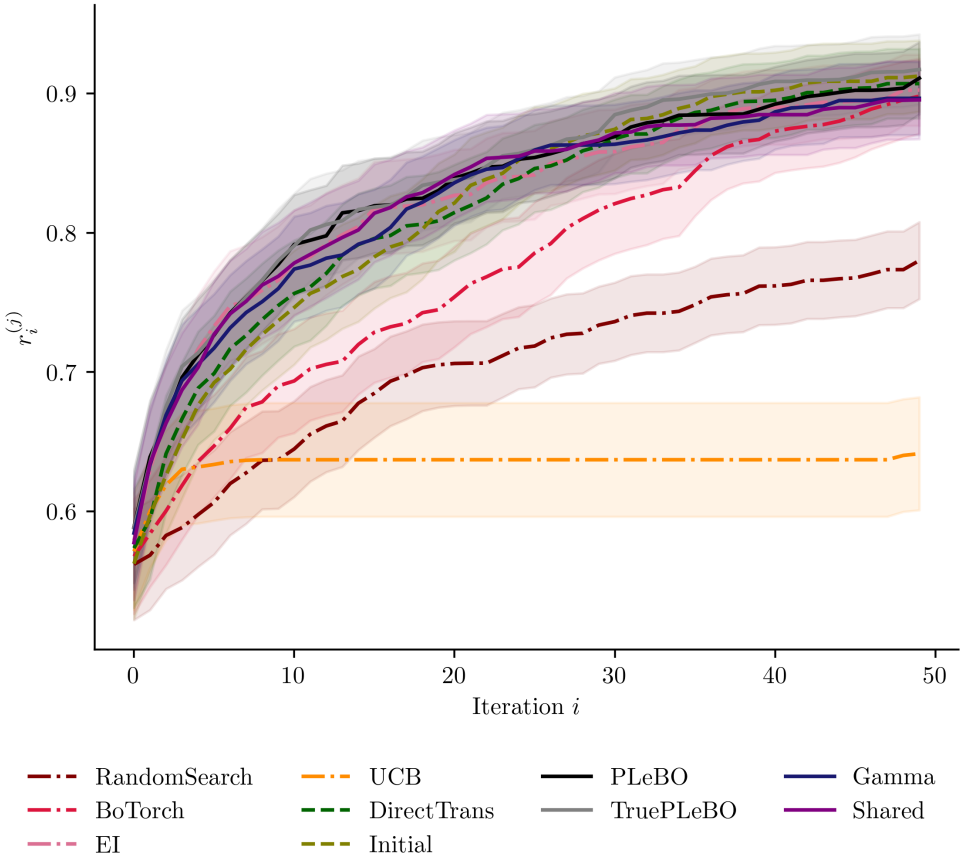


Figure 5.3: Results for synthetic optimisation tasks. We show the mean $\pm$ 2 standard errors.

than expected. The direct transfer baselines find worse inputs in the beginning, but after about 30 iterations begin finding slightly better values than PLeBO, especially Initial. The difference between PLeBO and Gamma and Shared is small, but at most iterations PLeBO has found a slightly better value.

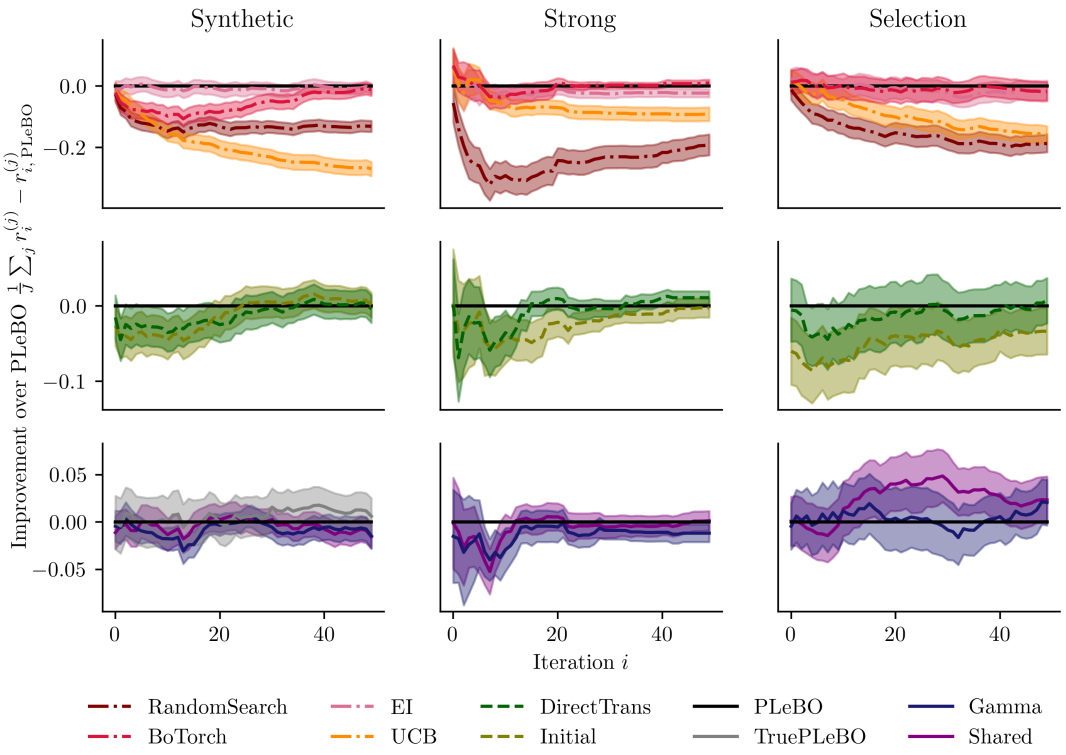


Figure 5.4: Performance compared to PLeBO at same iteration (above zero is improvement), mean ± one standard error. J is the number of test tasks. The top row compares PLeBO to non-transfer baselines, the middle to direct transfer and the bottom row to prior transfer methods. Note the difference in scales between the rows.

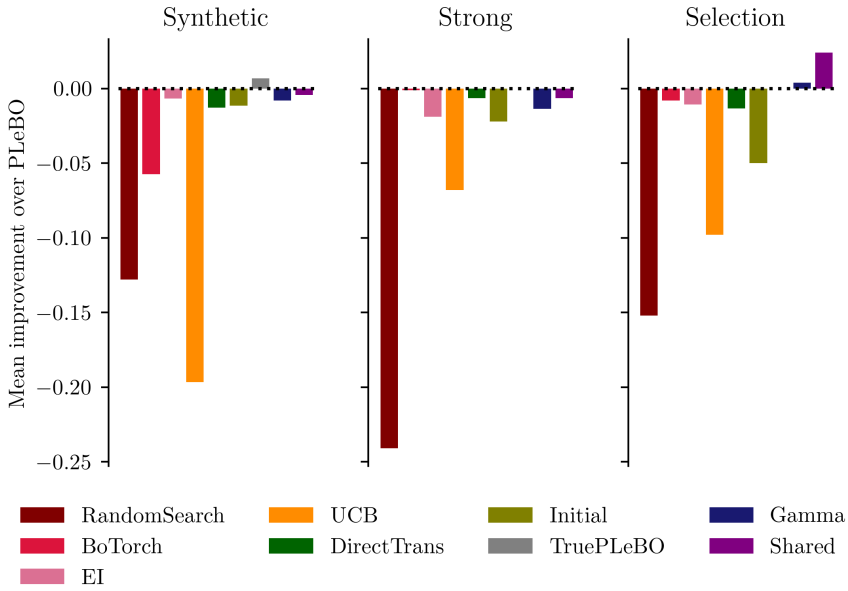


Figure 5.5: Mean performance compared to PLeBO (above zero is improvement). We show the mean over optimisation tasks and iterations: $\frac{1}{I}\sum_i \frac{1}{J}\sum_j r_i^{(j)} - r_{i,\text{PLeBO}}^{(j)}$.

Prior quality

An advantage of using synthetic data is that we can explore how well the learned prior $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$ agrees with the true prior. Fig. 5.6 compares the true hyperparameters $\boldsymbol{\theta}$ for each of the tuning tasks to the generated MCMC samples[3]. We see that the fits are generally good. Fig. 5.7 compares the likelihood of the true hyperparameters with that of the samples, and shows that the learned hyperparameters are reasonable, and in some cases (especially for tuning task 5) explain the data better. We can also compare the fit of the hyperparameter priors $\boldsymbol{\eta}$. We compare the learned closed-form prior, used in Gamma, to the true one used for generating the tasks in Fig. 5.8. Although they are not identical, they are reasonably similar. Since we only used 20 observations per tuning task to learn the prior we cannot expect to regain the true distribution $\boldsymbol{\eta}$ exactly.

3. This is after filtering out the samples with zero likelihood, see Section 5.2.1

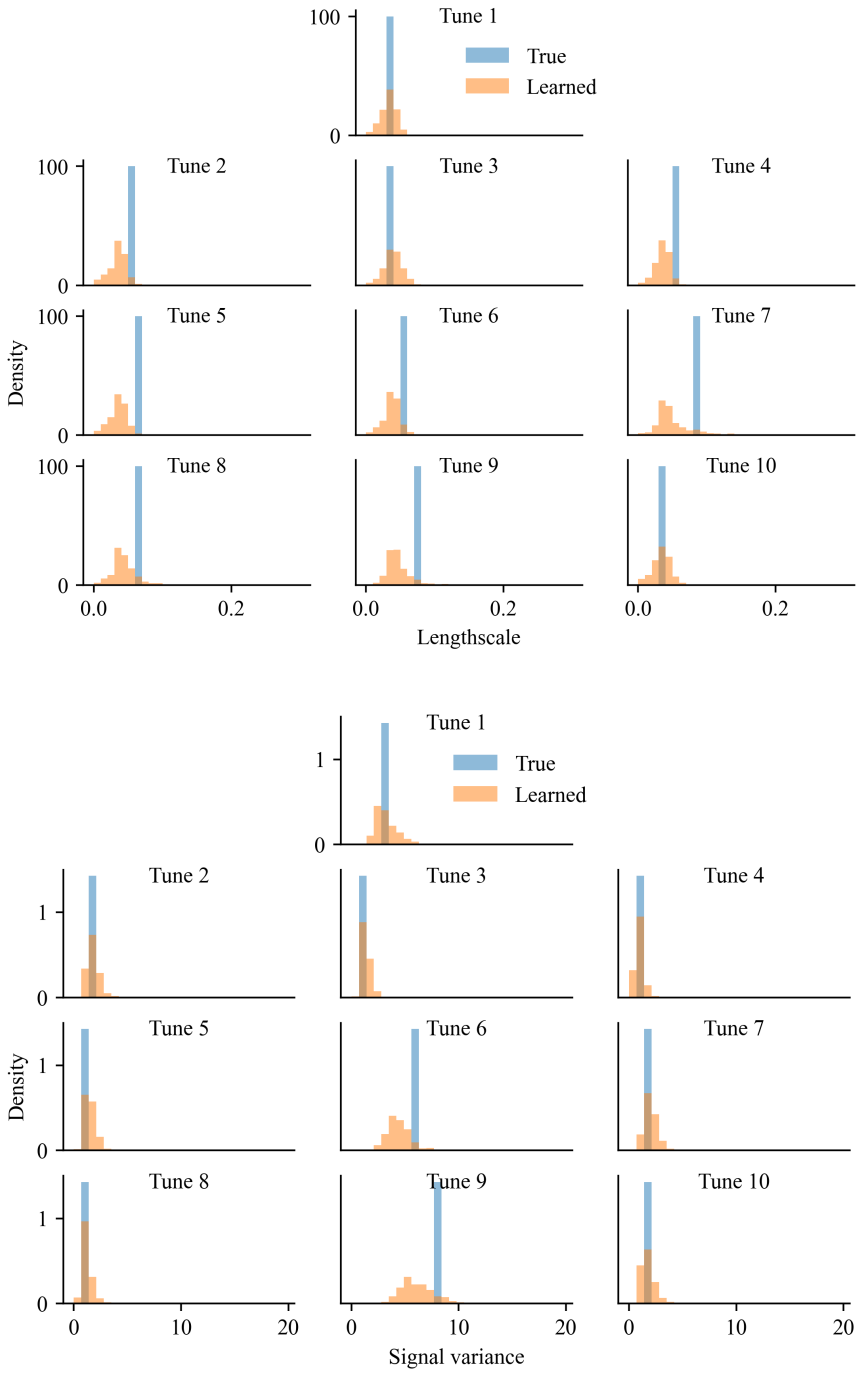


Figure 5.6: Comparing the true and learned values for the GP hyperparameters on the synthetic optimisation tasks. Tune 1 to 10 are the ten tuning tasks.

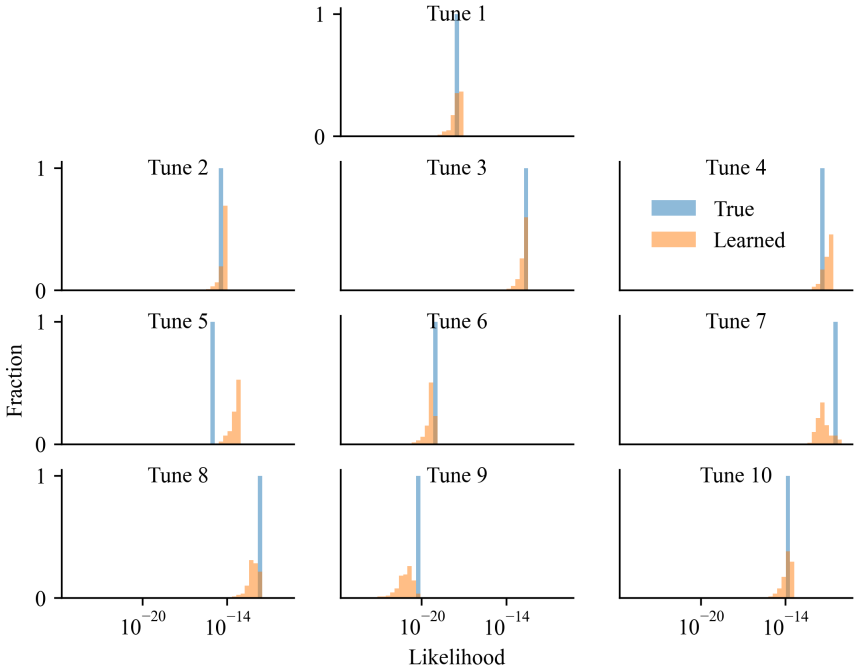


Figure 5.7: Comparing the likelihood of the true and learned values for the GP hyperparameters on the synthetic optimisation tasks. Tune 1 to 10 are the ten tuning tasks.

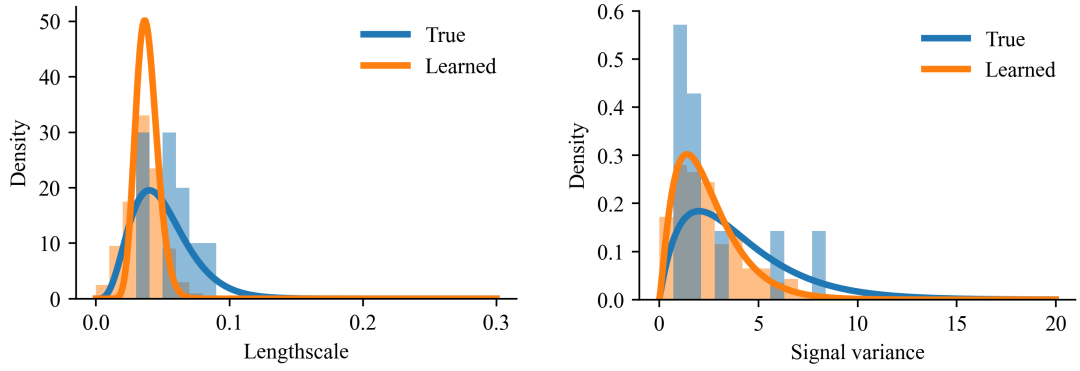


Figure 5.8: Comparing the true and learned priors for the GP hyperparameters on the synthetic optimisation tasks.

5.6 Air pollution experiments

We also compare the methods on the satellite benchmarks from Chapter 4. To simplify the analysis and comparison to baselines we made some changes to the set-up from Chapter 4, so the results are not directly comparable. The main changes are that

- We use a simple RBF kernel[4], instead of the combinations of kernels used in Chapter 4 including the new directed wind kernel. This allows us to use the GPy library (GPy, since 2012) instead of custom GP implementations, and makes it easier to inspect the generated priors. But it also simplifies the preprocessing in PLeBO, as we are only learning 2 hyperparameters, instead of 4–5.
- We only consider the satellite data. This is because all the London data come from the same city, and so we expect the direct transfer approaches to do much better than they would in a real deployment transferring between cities.

Using two-dimensional data, simple kernels and only tuning two hyperparameters simplifies the modelling and analysis, and allows us to visualise the optimisation landscape. But informative priors are more important for more expressive kernels. For instance, if we have five-dimensional input features, and fit a separate lengthscale to each of the features, there are many more ways of explaining the data correlations than if we have only one lengthscale. Therefore, we expect the impact of prior transfer to be greater for kernels with more hyperparameters. We discuss this further in Section 5.8.2 as future work.

We use the same subsets developed in Chapter 4, see Section 4.3.1. So 'Strong' refers to the 50 tasks with the maximal pollution concentrations present, and 'Selection' to 100 tasks representing the range of tasks, without overlap with the other subsets. The tuning set for Strong is 10 similar tasks. The tuning set for Selection is 10 tasks with maximal pollution concentrations at the middle of those for Selection. We use 100 randomly sampled values from each tuning task in the PLeBO preprocessing. We do not collect results for TruePLeBO as the data were not generated synthetically, so there are no true hyperparameters $\boldsymbol{\theta}$ to use.

4. Aside from BoTorch, which still uses its default kernel.

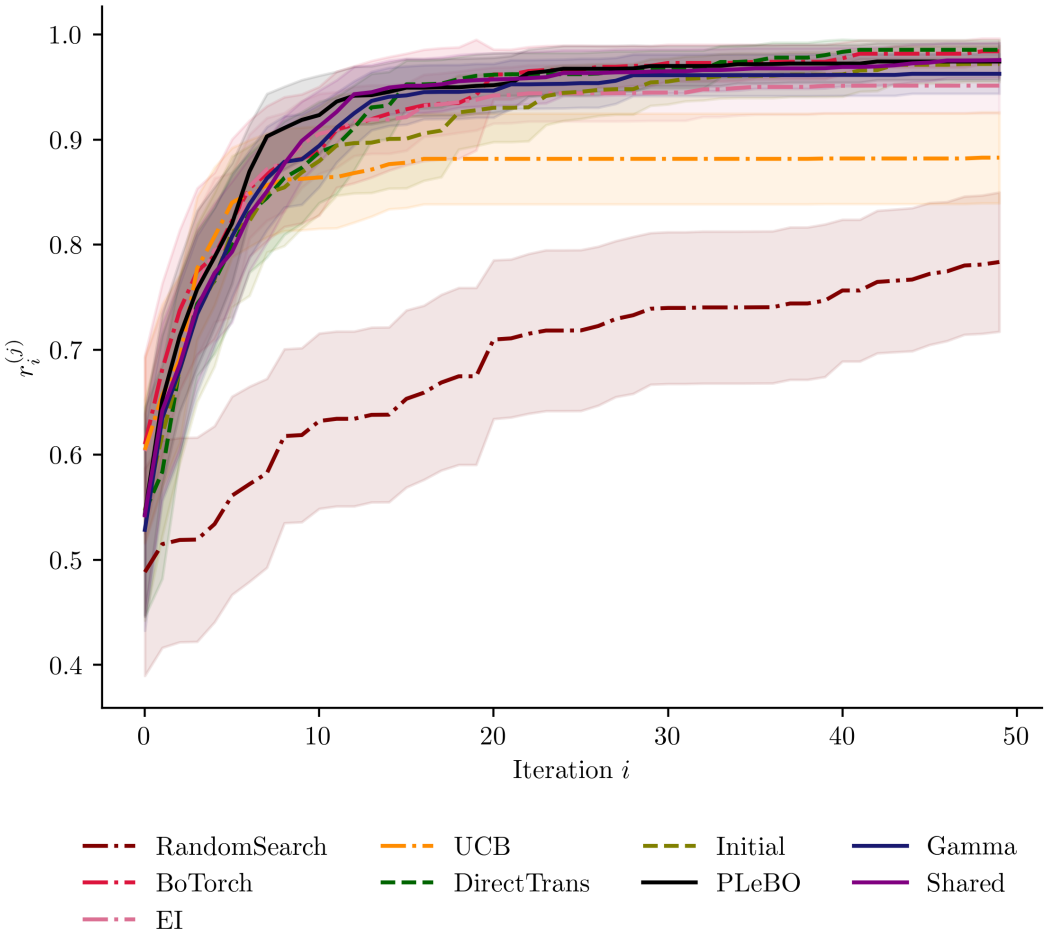


Figure 5.9: Results for Strong subset. We show the mean $\pm$ 2 standard errors.

The performance of all the methods is shown in Figs. 5.9 and 5.10. As in Chapter 4 the Selection tasks are harder, so the overall performance achieved is lower. The performance relative to PLeBO is shown in Figs. 5.4 and 5.5. For the Strong tasks PLeBO has found the best input at most of the iterations. But aside from UCB and RandomSearch the difference is small. For the Selection tasks Shared does best, with Gamma and PLeBO the runners-up.

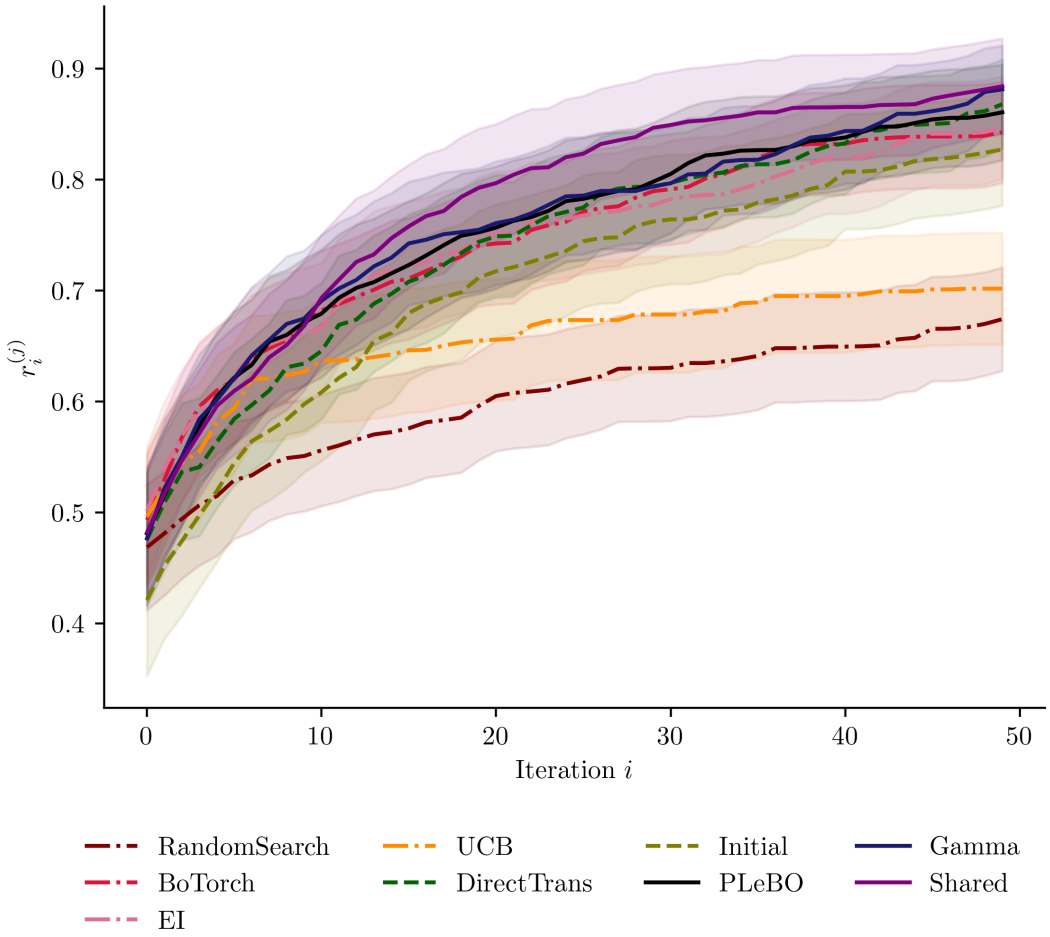


Figure 5.10: Results for Selection subset. We show the mean $\pm$ 2 standard errors.

5.7 Discussion

5.7.1 Upper confidence bound acquisition function

Looking at Figs. 5.3, 5.9 and 5.10, the UCB baseline does the worst of all the baselines aside from RandomSearch, and on the synthetic benchmark it even does worse than RandomSearch. One reason is likely to be that we are not tuning its hyperparameter β which balances exploration and exploitation (see Eq. (2.8)). But we did try other values of β in our experiments in Chapter 4, without any marked improvement. Therefore, it seems that EI does much better than UCB on these tasks. This suggests that a large part of the performance boost we got between the simple priors and the sampled priors in Chapter 4 came from swapping acquisition functions.

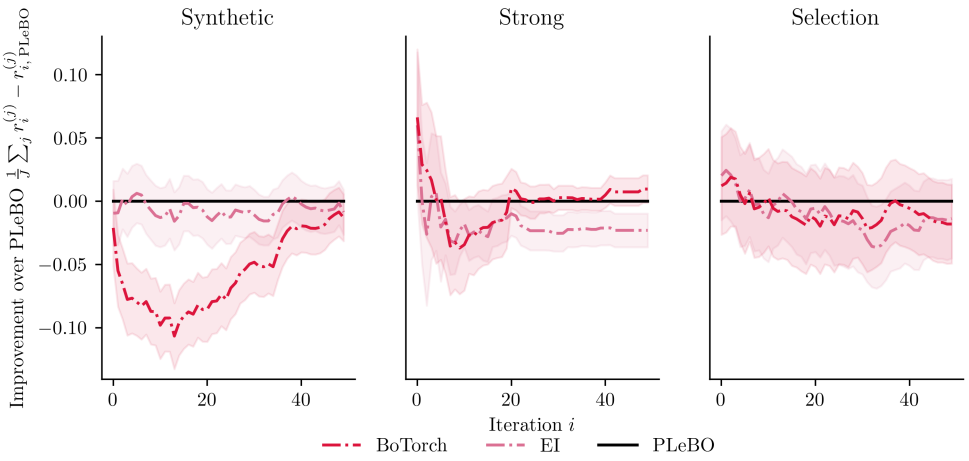


Figure 5.11: Extract of results from Fig. 5.4, comparing PLeBO to EI and BoTorch. Mean ± one standard error.

5.7.2 Limitations of prior transfer

While prior transfer is more generally applicable than direct transfer, it is also a weaker type of transfer learning. This is illustrated in Fig. 5.11, which compares PLeBO to the top two non-transfer baselines, EI and BoTorch. The BoTorch baseline does well on the air pollution benchmarks, but not the synthetic benchmark. This might be because its default kernel is a better match for the structures in the real-world data, as opposed to the synthetic data generated using RBF kernels, see Fig. 5.1. EI does very similarly to PLeBO on the synthetic benchmark, but on average slighly worse on the air pollution benchmarks.

Prior transfer cannot be a miracle pill, as it makes weaker assumptions about the relationship between tasks. For direct transfer, the best case is that the optima are in the same location, meaning that the entire optimisation process can be eliminated for the new task. But for prior transfer with GPs the best case is that we perfectly estimate the GP hyperparameters. That means we can reduce the number of function evaluations necessary by correctly disregarding regions around suboptimal points, but we still need to explore the search space.

5.7.3 PLeBO hyperparameter estimates

In Section 5.5 we considered the priors generated in the PLeBO preprocessing. But we can also examine the hyperparameter estimates generated by the importance weighting. We do this by computing hyperparameter means using the same weights as in the optimising step, as described in Section 5.2.2. *Note that we do not use these hyperparameter estimates in PLeBO, since we compute a weighted average of the acquisition function, not the hyperparameters.* But it gives a sense of what the hyperparameters are taken to be by PLeBO.

In Figs. 5.12 and 5.13 we compare the hyperparameter estimates of EI and PLeBO with the true values for the synthetic benchmark. For the lengthscale we see that the PLeBO estimate is clustered at a central value, with reasonable estimates even for very few data points. The estimates do not change much as we add more data, but EI is also not able to get close estimates. For the signal variance, both methods are better able to reconstruct the hyperparameter, but PLeBO does much better for few observations. The fact that neither EI nor PLeBO are able to learn the exact lengthscale, but instead have similar estimates for when the true lengthscales are large and small (the distributions of estimates are mostly flat for both in Fig. 5.12), indicates that there are not enough observations to learn the exact lengthscale. The mean square error (MSE) of the estimates is shown in Fig. 5.14. This shows that PLeBO is consistently able to produce better hyperparameter estimates.

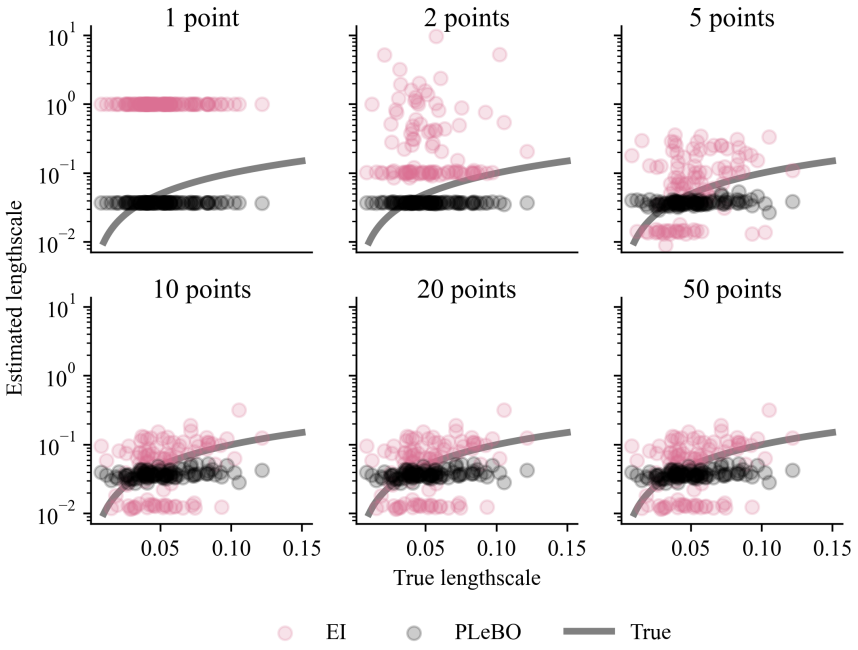


Figure 5.12: Comparison of lengthscale estimates of PLeBO and EI to the true values on the synthetic benchmark.

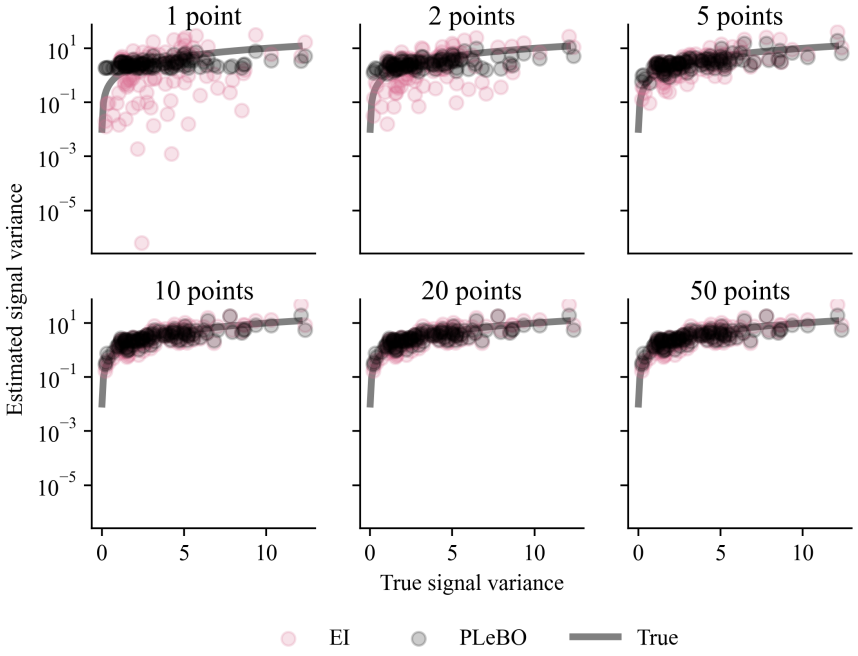


Figure 5.13: Comparison of signal variance estimates of PLeBO and EI to the true values on the synthetic benchmark.

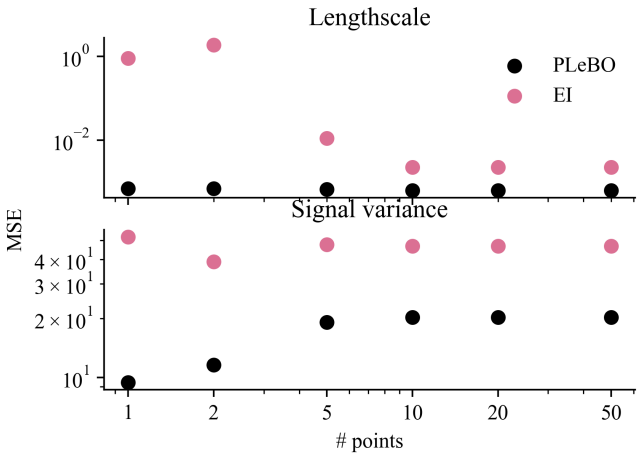


Figure 5.14: Mean square error of hyperparameter estimates of PLeBO and EI using a variable number of data points. Synthetic benchmark.

5.7.4 Computational complexity

PLeBO is computationally more expensive than the baselines. But it is much cheaper than doing a full inference of $\boldsymbol{\theta}_j|\{\boldsymbol{\mathcal{D}}_1, \dots \boldsymbol{\mathcal{D}}_N, \boldsymbol{\mathcal{D}}_j\}$, see Fig. 5.2. That would add the MCMC preprocessing to each optimisation step. We can estimate[5] the cost of this approach as being between 82 and 1759 times slower, using the values in Tables 5.1 and 5.2. Below we detail the computational cost of PLeBO and discuss ways of reducing it.

Preprocessing

In the preprocessing we run MCMC to collect samples to approximate the distributions of $\boldsymbol{\eta}$ and $\boldsymbol{\theta}$. The time complexity depends on the number of related tasks N, the number of evaluations per related task N_D, as well as the number of hyperparameters k. In this chapter we use $k = 2$: fitting the lengthscale and the signal variance. For each hyperparameter we have two parameters in $\boldsymbol{\eta}$, the shape ψ and scale ϕ of the gamma distribution. As we fit separate GPs to each tuning task we also have N sets of hyperparameters $\boldsymbol{\theta}_n$, which requires fitting Nk values.

The complexity of the NUTS sampler (Hoffman & Gelman, 2014) is not deterministic, but depends on the number of steps needed before a turn occurs. However, it does depend on the complexity of calculating the likelihoods of $\boldsymbol{\eta}$ and $\boldsymbol{\theta}$:

- To calculate the likelihood of $\boldsymbol{\eta}$ we need to calculate the probability of $\boldsymbol{\theta}$ given $\boldsymbol{\eta}$. That requires evaluating $\Gamma(\theta; \psi, \phi)$ for Nk values of θ.
- To calculate the likelihood of $\boldsymbol{\theta}$ we need to compute $\mathcal{N}(\boldsymbol{\mathcal{D}}_n; 0, \boldsymbol{K}_y(\boldsymbol{\theta}_n))$. This requires inverting the matrix $\boldsymbol{K}_y(\boldsymbol{\theta}_n)$ which is $O(N_D^3)$ (Rasmussen & Williams, 2006, p. 19). We need to do this for N different sets of $\boldsymbol{\theta}_n$.

The time complexity also depends on the hyperparameter choices for the NUTS sampling, e.g. the number of samples per chain $N_{\text{samples/chain}}$, the number of warm-up (burn-in) samples $N_{\text{burn-in}}$ and the number of chains N_{chains}. We use $N_{\text{samples/chain}} = 200$, $N_{\text{burn-in}} = 500$ and $N_{\text{chains}} = 5$.

---

5. This is an approximation for two reasons. Firstly, we are using empirical estimates of the preprocessing and optimisation durations. Secondly, doing the full inference at each step would introduce the hyperparameters $\boldsymbol{\theta}_j$ and observations $\boldsymbol{\mathcal{D}}_j$ of the test task into the inference, increasing the number of variables to estimate.

Empirically, we found that the preprocessing took 9 minutes for the synthetic benchmark, and 2.6–3.4 hours for the air pollution benchmarks on a standard desktop, see Table 5.1. We use $N_D = 20$ for the synthetic benchmark and $N_D = 100$ for the air pollution benchmarks, so expect the latter to be slower. The preprocessing is only done once, so a higher computational expense can be tolerated, especially when optimising expensive tasks. The preprocessing time can be reduced by reducing one or more of $N_{\text{samples/chain}}$, N_{chains} and $N_{\text{burn-in}}$.

Benchmark	Preprocessing time
Synthetic	9 min 11 sec
Strong	2 h 38 min
Selection	3 h 22 min

Table 5.1: PLeBO preprocessing times.

Optimising

Within each loop of the optimisation, we evaluate the acquisition function and likelihood for H hyperparameter values. Both the likelihood and acquisition function require inverting the covariance matrix which is $O(N_i^3)$ where N_i is the number of observations available at iteration i (Rasmussen & Williams, 2006, p. 19). So the complexity of a PLeBO optimisation step is $O(HN_i^3)$. For EI and UCB this inversion is instead done for each of the N_{grad} steps of the gradient-based method, which scales as $O(N_{\text{grad}}N_i^3)$.

Empirically, we see that PLeBO takes about 6.7 seconds per optimisation steps, compared to 0.13 seconds for EI and 0.04 seconds for Shared. For very expensive optimisation tasks this is still only a tiny fraction of the total cost, e.g. when installing an air pollution sensor. Runtimes for all methods are given in Table 5.2. We have capped the number of past evaluations available to DirectTrans at 100, otherwise it would be much slower than the listed times. PLeBO's runtime can be adjusted through H based on the available budget and task evaluation speed. Higher H has a higher runtime, but also a higher quality of the posterior estimate. At one extreme we could set H to 1, which should give a similar runtime to Shared.

Method	Synthetic	Strong	Selection
RandomSearch	0.006	0.005	0.005
BoTorch	0.20	0.31	0.27
EI	0.16	0.11	0.11
UCB	0.17	0.11	0.11
DirectTrans	1.35	1.20	1.20
Initial	0.16	0.10	0.11
PLeBO	6.73	6.60	6.89
TruePLeBO	0.05	-	-
Gamma	0.18	0.13	0.12
Shared	0.05	0.04	0.04

Table 5.2: Mean durations in seconds of an optimisation step.

Possible improvement: As N_i increases, the weights will become more dissimilar, and the weighted average more dominated by the top fitting hyperparameter candidates. This is illustrated in Fig. 5.15, which shows the distribution of weights for the first and fiftieth iteration of PLeBO. As we can see, there are many more highly unlikely hyperparameter candidates for the fiftieth iteration, when the highest weights are also higher.

To speed up the optimisation, candidates could be discarded once their contributions to the weighted sum are negligible. A similar idea is used in Feurer et al. (2022) to avoid weight dilution. While discarding candidates would speed up later optimisation iterations, it would also reduce the opportunity of recovering from model misspecification: if new evaluations are much more likely using the to-be-discarded hyperparameters, these hyperparameters would start impacting the acquisition function again.

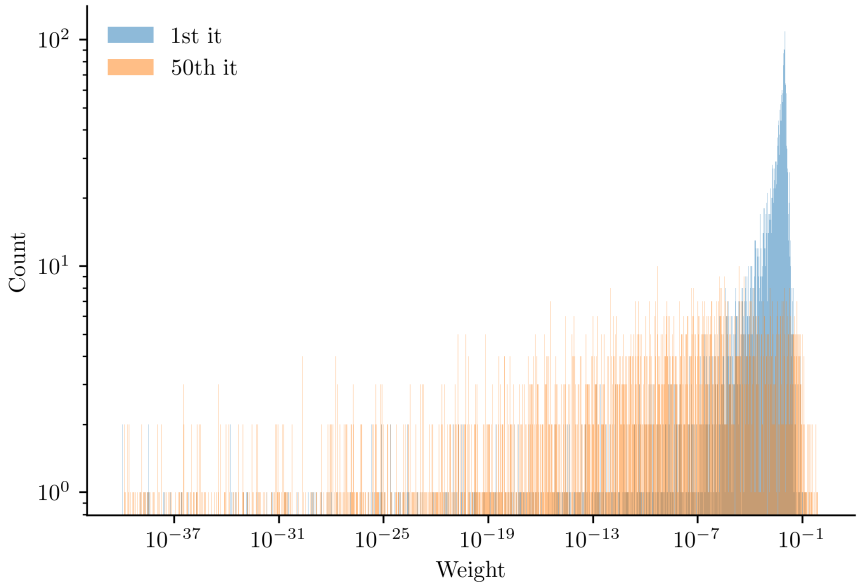


Figure 5.15: Distribution of weights for synthetic tasks for the first and fiftieth iteration.

5.7.5 Comparing prior transfer methods

We collect results for four prior transfer methods: PLeBO and its version with access to the true hyperparameters, TruePLeBO; Gamma, and Shared. Gamma uses the same preprocessing as PLeBO, but instead of using the MCMC samples of $\boldsymbol{\eta}$ to generate candidate hyperparameters as in PLeBO, in Gamma we calculate the mean of $\boldsymbol{\eta}$ and use that as a prior when fitting the hyperparameters using gradient-based optimisation. So the computational complexity of the preprocessing of PLeBO and Gamma is the same, but the optimisation steps are faster using Gamma, see Table 5.2. Shared learns only one set of GP hyperparameters across all tasks.

The prior transfer results are shown in Fig. 5.16. For TruePLeBO we can only collect results for the synthetic benchmark, but it outperforms the other methods. For the synthetic and Strong benchmarks PLeBO does slightly better than Gamma and Shared on average, but the differences are small, except for the first ca. 15 iterations of Strong, where PLeBO is clearly preferable.

For the Selection benchmark, Gamma does slightly better than PLeBO, and Shared is clearly preferable. This is also the hardest of the benchmarks, as can be seen by comparing the top performance obtained in Figs. 5.3, 5.9 and 5.10. The tuning set used for Selection is also less helpful than for Strong and synthetic, as the tuning tasks are clustered around the median test tasks, and do not show the range of optimisation

tasks. But this is the same tuning set used by Shared and Gamma. The results show that there is still work to do in improving the PLeBO method. It is possible that PLeBO is more reliant on using the correct kernel, as PLeBO did relatively better on the synthetic benchmark.

As discussed in Section 5.7.4, PLeBO is more expensive than Gamma and Shared. This can be improved by reducing H, but Shared is a cheaper method to use. If the function evaluations are cheap, e.g. taking less than a second to run on a computer, then Shared is the obvious choice. But there is less need for prior transfer overall for these cheap optimisation objectives, and we would quickly collect enough data for the new task to make prior transfer superfluous as we could just fit a kernel to evaluations of the test task.

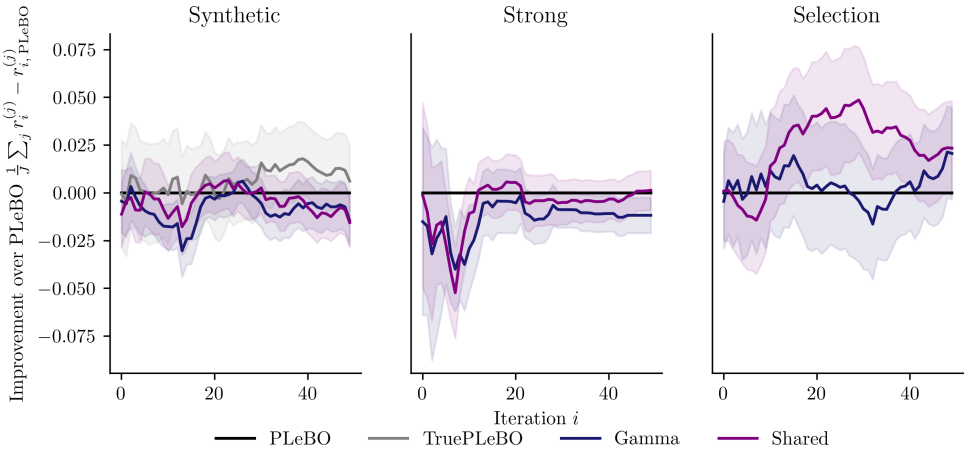


Figure 5.16: Extract of results from Fig. 5.4, comparing PLeBO to TruePLeBO, Gamma and Shared. Mean $\pm$ one standard error.

5.8 Conclusion

In this chapter we have improved upon, analysed and compared the method developed in Chapter 4 for air pollution monitoring. This is a new transfer learning method for BO, and in this chapter we compared it with related transfer learning methods. We provided a new implementation using NumPyro (Phan et al., 2019), which is much shorter and makes it easier to change design decisions such as the MCMC sampler or the distributions for $p(\boldsymbol{\theta}|\boldsymbol{\eta})$.

Our method PLeBO can be combined with other approaches. For instance, we can use it with acquisition functions other than expected improvement, including ones explicitly learning hyperparameter values such as Hvarfner et al. (2023). PLeBO can also be combined with other transfer learning methods. For instance, we could combine direct and prior transfer and use past tasks to both learn a set of initial points to evaluate and to learn an informative prior.

The current method only uses hyperparameters learned from previous tasks. We use the new task to decide which of these candidate hyperparameters to give the most weight, but we do not learn any new ones. But the candidate hyperparameters could be combined with learning a new set of hyperparameters on the new task, and adding this into the weighed combination, as is done in other ensemble methods, e.g. Feurer et al. (2022). We also have not compared different values of H. This could be done to speed up the optimisation step for PLeBO, which is currently much slower than for the baselines.

5.8.1 Limitations

Our experiments, see e.g. Fig. 5.4, show that while PLeBO and prior transfer in general is preferable to direct transfer in the examined settings, there are very strong non-transfer baselines. This highlights the limitations of prior transfer — it cannot teach us everything we need to know about the function we are optimising, just improve the model fitting step. Compared to other prior transfer methods, PLeBO also has limitations. Choosing each function evaluation takes much longer than for the baselines, and the performance is similar to the other methods, especially Shared. Finally, we only compare on one type of real-world data, while a more thorough comparison of the methods would use multiple types of real-world optimisation problems.

5.8.2 Future work

As mentioned in Section 5.6, it would be interesting to apply PLeBO to more expressive kernels. As the kernels become more expressive, and have more hyperparameters that need tuning, we need more evaluations to explain the data. For instance, consider the toy data in Table 5.3. If we use a kernel with a separate lengthscale for each dimension[6], then the evaluations can be explained in different ways. The similarity in

6. Called automatic relevance determination, see Rasmussen and Williams (2006, pp. 106–107).

pollution levels between evaluations 1 and 3, and dissimilarity to evaluation 2, could be explained by changes in latitude, longitude, measurement height or distance to the nearest road. Or by any number of interactions between the features. However, if we have narrow priors on the lengthscales, for instance specifying that the correlation falls off after 0.5 kms in either latitude or longitude, and that any measurement height between 0 and 10 metres is highly correlated, then we can explain the changes in terms of the road distance feature.

	Lat. (km)	Lon. (km)	Time (h)	Height (m)	Road dist. (m)	Pollution (ppb)
Eval. 1	0	0	0	1.0	1	25
Eval. 2	8	10	0	1.2	10	5
Eval. 3	1	2	1	1.0	1	24

Table 5.3: Toy higher-dimensional air pollution data.

Because more hyperparameters mean more possible explanations of the data, we expect the difference between prior transfer and no transfer to be greater for larger numbers of hyperparameters. As an early exploration, a limited number of hyperparameters is sensible, and it allowed us to build up intuition about the method and inspect the priors more easily. The good news is that while changing to more expressive kernels, higher-dimensional data, or both, would have been time-consuming using the implementation in Chapter 4, it should be much faster with our new implementation.

If we adopt more expressive kernels, we would also expect to need more tuning data. And we expect an increase in the dimensionality of both $\boldsymbol{\theta}$ and $\mathcal{D}$ to result in an increase in the time needed for learning $\boldsymbol{\eta}$ in the preprocessing step. For common problem settings, we could then run the preprocessing once centrally, and publish the resulting hyperparameter priors for re-use.

6

Ordered Transfer Hyperparameter Optimisation

Take them one by one.

– Jorunn Hellan on the topic of overtaking slower cars

This chapter is based on the paper Obeying the Order: Introducing Ordered Transfer Hyperparameter Optimisation*, which was presented in the workshop track of the AutoML 2023 conference (Hellan, Shen, et al., 2023). It is the result of an internship conducted at Amazon Web Services.*

6.1 Introduction

We now move on to considering direct transfer, i.e. transfer problems where we expect good input values on past tasks to work well on a new task. We will consider the application of hyperparameter optimisation for machine learning models.

This chapter is an exploration of a simple idea: when developing and deploying a machine learning model, we often optimise the hyperparameters of many similar set-ups, and we are likely to make changes in a temporally directed fashion. Our running example is that we are gradually increasing the amount of training data for a model. For instance, we can imagine a movie recommendation system, which is retrained every week to include the latest ratings. Then the training data for week 2 are an expanded version of the training data for week 1, and the data for week 3 are an expanded version of the data for week 2. When we have this gradual change from one optimisation task to the next we can apply *ordered transfer learning*. This is a version of direct transfer which prioritises the historical evaluations from the most recent

tasks. By taking the task ordering into account we can reduce the number of input evaluations needed for the new task. We implemented our methods and benchmarks in Syne Tune (Salinas et al., 2022), an open-source HPO library, to facilitate future work.

Note that we have now reached another use of the term 'hyperparameter'. While in Chapters 4 and 5 we were talking about the GP hyperparameters, so parts of the *optimisation model*, we will now be applying optimisation to the hyperparameters of other machine learning models.

6.2 Hyperparameter optimisation for machine learning

The main application in this chapter is hyperparameter optimisation for machine learning. That is, for a machine learning model and a data set, we want to find the hyperparameter values that optimise performance. We consider five machine learning models: XGBoost, random forest, approximate k-nearest neighbours, elastic net and support vector machines. For XGBoost we train and evaluate our own models. For the rest we use the YAHPO Gym surrogate models (Pfisterer, Schneider, Moosbauer, Binder, & Bischl, 2022). The training data for the surrogate models are hyperparameter evaluations: the features are hyperparameter configurations, and the labels the performance of an ML model trained using those hyperparameters. The surrogate models therefore predict hyperparameter performance, and can estimate labels for unseen hyperparameter configurations.

Hyperparameter optimisation is tricky — it is a research field in itself, and papers are written about optimising the hyperparameters of individual methods, e.g. Probst, Wright, and Boulesteix (2019). Each machine learning method has its own set of hyperparameters, so intuition for reasonable values has to be built up separately. At the same time, we want to know whether the method will work on our data set before we commit to learning the details of its hyperparameters. Using automated hyperparameter optimisation, we can tune the method to our data set without

consulting the documentation for the method's software implementation. This is much faster, so we can do it for multiple methods, and choose the one that works best. If further tuning is necessary, we can afterwards focus our reading on our chosen method.

Other motivations for HPO are to free up developer and research time for other activities and reduce the amount of total compute needed. A key motivation for automating HPO is therefore to improve productivity. Another motivation is to make machine learning more accessible to non-experts, by removing the need to understand the methods used.[1] For HPO, transfer learning means using evaluations from other HPO runs. Often, this comes from applying the same method to a different data set. We consider a subcategory of this, where the next data set is a superset of the previous data set.

The machine learning methods to which we applied HPO were treated as black boxes[2]. But brief summaries of the methods are given here. Most of the hyperparameters are method-specific, but the YAHPO benchmarks all have an imputation hyperparameter for missing data (Binder, Pfisterer, & Bischl, 2020).

6.2.1 XGBoost

Extreme gradient boosting (XGBoost, T. Chen and Guestrin (2016)) is a version of tree boosting (Murphy, 2022, p. 617). In boosting we build up an ensemble of base models from the errors of the previous models: we weight the data points based on whether the previous ensemble got them wrong. The individual models might not do well, but the combined model does a lot better.

We compare the performance of XGBoost classifiers[3] trained on MNIST (Vanschoren, van Rijn, Bischl, & Torgo, 2013), optimising four hyperparameters:

- max depth (integer)
- number of estimators (integer)

---

1. Removing the need for users to understand the methods has both advantages and disadvantages. While it means more people are able to use them, it also risks more uncritical applications of machine learning.
2. Other HPO methods, e.g. PASHA (Bohdal et al., 2023), can use structural properties of the problem to speed up the HPO. For instance, the relative performance early in the training runs is highly correlated with the relative performance after the full training runs, so some hyperparameter configurations can be discarded after only part of the training run.
3. XBGClassifier `https://xgboost.readthedocs.io/en/stable/python/python_api.html#xgboost.XGBClassifier`.

- minimum child weight (continuous)
- learning rate (continuous)

Max depth is the maximum depth of the tree base models. The number of estimators is the number of models we use. The minimum child weight specifies the minimum sum of weights of the data points in a node: if a split would result in a node with too low a weight the split is not carried out. The learning rate scales the new feature weights in boosting to avoid overfitting.

6.2.2 Random forest

Random forest is an ensemble method where we learn a set of weak learners — trees — and combine them to make a better learner. We randomly subsample the data and features used to learn each tree (Murphy, 2022, p. 608). The ranger implementation (Wright & Ziegler, 2017) was used to train the YAHPO surrogate model. We optimise a mixed set of eight hyperparameters:

- Number of trees (integer)
- Splitting rule (categorical: gini impurity or extra trees)
- Number of random splits (integer, only for extra trees splitting rule)
- Sample fraction (continuous)
- Minimal node size for splitting (integer)
- Respect unordered factors (categorical: ignore, order or partition)
- Mtry power (integer)
- Imputation (categorical: mean, median or histogram)

The splitting rule is how we choose to split a node (Probst et al., 2019), which can be either through minimising the gini impurity, or by choosing among randomly sampled splits (Geurts, Ernst, & Wehenkel, 2006). The sample fraction is the fraction of observations to use for each tree. The minimal node size for splitting determines how small we are willing to make our end nodes. The respect unordered factors hyperparameter is how to deal with categorical variables, and Mtry power is how many features to consider when choosing a splitting rule.

6.2.3 Approximate k-nearest neighbours

K-nearest neighbours (KNN) is a method that classifies a new point $\boldsymbol{x}_{new}$ by finding the k nearest neighbours of it within the training data, and using their classes to define a distribution or to predict the most common one (Murphy, 2022, p. 545). Which points are nearest is defined by the similarity metric used, for instance the Euclidean distance $d_{Euclid} = \sqrt{(\boldsymbol{x}_{new} - \boldsymbol{x})^T(\boldsymbol{x}_{new} - \boldsymbol{x})}$. The exact solution requires comparing the new point to all the training points, which scales linearly with the number of training points. The idea behind approximate k-nearest neighbours (KANN, Malkov and Yashunin (2018))[4] is to speed up the process of finding the neighbouring points by using an approximation. Malkov and Yashunin use an approximation based on multiple levels of graphs, so that the search can first be done on a graph representing fewer data points, and then refined on more detailed graphs.

We optimise a mixed set of six hyperparameters:

- M (integer)
- distance metric (categorical: squared L2, cosine similarity or inner product)
- ef (integer)
- ef construction (integer)
- k (integer)
- Imputation (categorical: mean, median or histogram)

'ef' and 'ef construction' are related to the accuracy/speed-up trade-off, and M is a parameter of the approximation structure. k is the number of neighbours to use.

6.2.4 Elastic net

Elastic nets are a type of regularisers for linear regression which combine the L1 and L2 loss (Murphy, 2022, p. 394). The YAHPO surrogate model is based on the glmnet package (Friedman, Hastie, & Tibshirani, 2010).

We optimise a mixed set of three hyperparameters:

- Alpha (continuous)
- s (continuous)
- Imputation (categorical: mean, median or histogram)

Alpha is a hyperparameter balancing the L1 and L2 loss. s (also called lamdba) scales the regularisation.

4. `https://github.com/nmslib/hnswlib`.

6.2.5 Support vector machines

Support vector machines (SVMs, Schölkopf and Smola (2002); Murphy (2022, pp. 583–595)) are a machine learning method used chiefly for classification. We use a kernel to transform the features to a potentially more expressive space, and then learn a simple classifier in that space.

The surrogate model used (Pfisterer et al., 2022) builds on the e1071 R package implementation[5]. We optimise a mixed set of six hyperparameters:

- Cost (continuous)
- Kernel (categorical: linear, polynomial or radial)
- Termination tolerance (continuous)
- Degree (integer, only for polynomial kernel)
- Gamma (continuous, kernel parameter for polynomial and radial)
- Imputation (categorical: mean, median or histogram)

The cost hyperparameter is a regularisation term. The kernel is the feature transformation, which can have extra hyperparameters degree and gamma. The termination tolerance determines when to stop optimising the decision boundary.

6.2.6 Summary

The previous summaries illustrate the difficulty of hyperparameter optimisation. The five methods have a total of 24 hyperparameters that need to be learned, and understanding each requires learning about details of the method. A black-box optimisation method providing good results is therefore very useful.

6.3 NewsVendor benchmark

In addition to the HPO benchmarks we also demonstrated the effect of ordering on a pricing benchmark. The NewsVendor benchmark is a modification of the Dynamic News problem from the SimOpt library (Eckman, Henderson, & Shashaani, 2023). The problem is to maximise profit by setting the prices for three items given a changing and unknown demand. We change the demand between tasks using a random walk, giving a gradually changing problem.

5. `https://CRAN.R-project.org/package=e1071`.

6.4 Author contributions

The project was led by me, mentored by Shen and had weekly input from Aubet, Salinas and Klein. Aubet also helped with identifying the NewsVendor benchmark, and Salinas and Klein with the Syne Tune library. Shen helped with identifying and plotting the YAHPO benchmarks. I led the writing, but everyone contributed, especially to the Related work and Problem definition sections.

6.5 The paper

Obeying the Order: Introducing Ordered Transfer Hyperparameter Optimisation

Sigrid Passano Hellan[†1] **Huibin Shen**[*2] **François-Xavier Aubet**[2] **David Salinas**[2] **Aaron Klein**[2]

[1]University of Edinburgh
[2]AWS AI Labs

Abstract We introduce *ordered transfer hyperparameter optimisation* (OTHPO), a version of transfer learning for hyperparameter optimisation (HPO) where the tasks follow a sequential order. Unlike for state-of-the-art transfer HPO, the assumption is that each task is most correlated to those immediately before it. This matches many deployed settings, where hyperparameters are retuned as more data is collected; for instance tuning a sequence of movie recommendation systems as more movies and ratings are added. We propose a formal definition, outline the differences to related problems and propose a basic OTHPO method that outperforms state-of-the-art transfer HPO. We empirically show the importance of taking order into account using ten benchmarks. The benchmarks are in the setting of gradually accumulating data, and span XGBoost, random forest, approximate k-nearest neighbor, elastic net, support vector machines and a separate real-world motivated optimisation problem. We open source the benchmarks to foster future research on ordered transfer HPO.

1 Introduction

All modern machine learning (ML) pipelines contain many hyperparameters that are critical for final performance, as they govern key parts of the training such as the optimisation (learning rate), the capacity of the model (number of layers or regularisation weights) or data augmentation. Hyperparameter optimisation (HPO) — see e.g. the recent book by Feurer and Hutter (2019) — aims to find the optimal hyperparameters of a machine learning method by casting it as an optimisation problem: for each iteration a new set of hyperparameters is used to train and validate the method.

In practical scenarios, hyperparameters are not tuned once but many times. Consider a movie recommendation ML system being deployed. The model hyperparameters must be tuned frequently, given that the data set consistently evolves with new movies and users, which increases the data set size and gives a continuous shift in the optimal hyperparameter values. In particular, we expect some hyperparameters that define the regularisation or the capacity of the model to change as more data is observed. Smaller models might be initially superior for little data, as only simple rules can be learned, but, as more data becomes available, more expressive models start to become competitive, as they are able to identify smaller differences between the inputs. This point is illustrated in Fig. 1 which plots the validation performance of XGBoost with respect to the number of estimators and the maximum depth of each tree. As the data set size increases (from left to right) the optimal hyperparameter values change smoothly and more expressive models (i.e more and deeper trees) become superior. We call each hyperparameter optimisation of a model a *task*.

A popular family of approaches that can exploit information from previous tasks is transfer HPO (Wistuba et al., 2015b; Feurer et al., 2018; Salinas et al., 2020). Such methods exploit data collected from the HPO of previous tasks to warm-start the optimisation on the current task.

[†]Work done during an internship at AWS AI Labs.
[*]Correspondence to: Huibin Shen < >.

 © 2023 the authors, released under CC BY 4.0

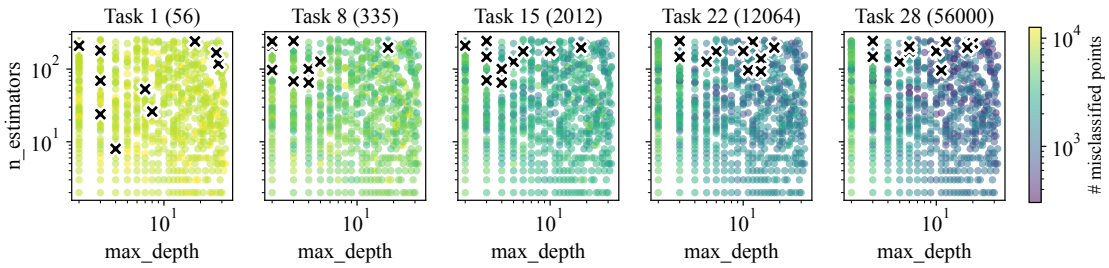


Figure 1: Evolution of the optimal XGBoost hyperparameters on MNIST for gradually increasing training set sizes (from 56 to 56000 points). The black crosses indicate the 10 best hyperparameters, and can be seen to shift upwards to more estimators as more data is added.

However, transfer HPO methods treat tasks as a set and ignore any intrinsic order. In practice, data is often collected in a sequence, for instance when a production system is tuned at frequent intervals. We introduce *ordered transfer HPO* (OTHPO), as a special case of transfer HPO that exploits this sequential nature of tasks, enabling a better transfer of knowledge across tasks. See Fig. 2 for an illustration. Our contributions are:

- We propose a formal definition of ordered transfer HPO and outline the differences to related problems, such as standard transfer HPO and continual learning.
- We provide ten benchmarks for this setup, integrated in the recently open-sourced HPO library Syne Tune (Salinas et al., 2022) to compare existing approaches for transfer HPO and foster the development of future methods. These benchmarks include XGBoost (Chen and Guestrin, 2016), support vector machines, approximate k-nearest neighbor (Malkov and Yashunin, 2018), random forest (Wright and Ziegler, 2017) and elastic net (Friedman et al., 2010) on various data sets, as well as a blackbox optimisation task based on SimOpt (Eckman et al., 2023).
- Our results in the setting of accumulating training data over time suggest that, in this setting, OTHPO methods taking order into account are simple and performant, which provides guidance for HPO practitioners in a deployed system.

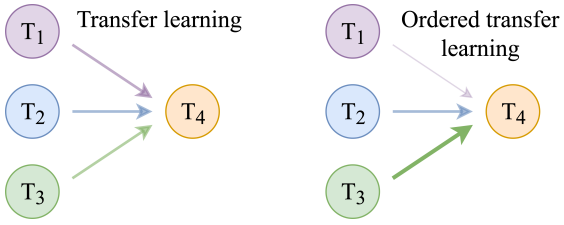


Figure 2: Contrasting standard transfer HPO and OTHPO. In the former (left), when learning the next task, T_4, each previous task (T_1, T_2, T_3) is either assumed equally important or weighted according to meta-features or hyperparameter rank matching. For OTHPO (right), more recent tasks are assumed more relevant, as illustrated by the difference in arrow widths.

2 Related work

OTHPO is related to but distinct from standard transfer HPO (Bai et al., 2023), continual learning (Van de Ven and Tolias, 2019; Chaudhry et al., 2019) and multi-fidelity HPO (Jamieson and Talwalkar, 2016; Li et al., 2017).

Standard transfer HPO (Perrone et al., 2019; Salinas et al., 2020; Wistuba et al., 2015b; Horváth et al., 2021) typically transfers knowledge between data sets, and, compared to OTHPO tasks, does not have an inherent ordering that one can exploit. For example, Wistuba et al. (2015b) used

hyperparameter configurations evaluated on previous tasks to create a portfolio of well-performing configurations, which are sequentially evaluated on a new task. Perrone et al. (2019) reduced the search space for a new task by defining a bounding box around the best performing configurations on all previous tasks. To account for the variability in objective scales of different tasks, Salinas et al. (2020) learned a semi-parametric Gaussian Copula distribution across tasks. Springenberg et al. (2016) used a Bayesian neural network to model the correlation between tasks for multi-task Bayesian optimisation. In a similar vein, Perrone et al. (2019) trained neural networks to learn basis functions across tasks and combined this with Bayesian linear regression to obtain reliable uncertainty estimates for Bayesian optimisation. This idea was extended by Horváth et al. (2021), which regularised the basis functions to account for the changing complexity during the optimisation process.

In Wistuba and Grabocka (2021), the authors considered HPO as a few-shot learning problem where a Deep GP (Gaussian process) model was trained jointly on a set of meta-tasks by few-shot learning. For a target task they then started from the initialised kernel parameters, before fine-tuning the model with a few hyperparameter evaluations.

Some transfer learning HPO methods such as Wistuba et al. (2015a) proposed leveraging meta-features of previous tasks to exploit task similarities. However, those methods rely on manual engineering of task features which are critical for final performance. To tackle this issue, Jomaa et al. (2021) proposed using Deep GPs to learn meta-features in an end-to-end fashion, which shows encouraging results in combating negative transfers. In what follows, we use standard transfer HPO as a shorthand for non-ordered transfer learning.

Both OTHPO and **continual learning** consider sequences of tasks. But continual learning is concerned with maintaining model performance on previous tasks, typically keeping the hyperparameters constant. We, on the other hand, want to optimise the hyperparameters for the current task and do not need the new model to perform well on the previous tasks.

In contrast to **multi-fidelity HPO**, OTHPO cares about the performance at each level. While a subset of the training data could be used in the multi-fidelity setting as a heuristic for later performance (Li et al., 2017; Klein et al., 2017), we consider the performance on the earlier task as a goal in itself. The idea (Zappella et al., 2021) of multi-fidelity HPO has been also extended to the standard transfer HPO setting.

Previous work has also been motivated by the idea of considering HPO of a model under change as a sequence of tasks (Golovin et al., 2017; Zhang et al., 2019; Stoll et al., 2020), but to our knowledge none have explicitly evaluated the importance of the ordering and they exhibit a few notable differences to our work. In Stoll et al. (2020), the ordering is implicitly used by only transferring from the latest task with a different search space and underlying algorithm. Golovin et al. (2017) motivate their work through HPO on different tasks in a sequence, but do not present results on HPO or compare to standard transfer HPO. The open source version of the software does not include transfer learning, so is not included as a baseline in this paper.

Zhang et al. (2019) is most related to our work as they identify the practical problem of slowly evolving data sets and the need to perform transfer HPO. However, they do not explicitly compare using the task ordering to standard transfer HPO. And while their evaluation considers the best possible performance on a new task, we show the large potential speed-ups possible by using OTHPO, since we show improved results after only one hyperparameter evaluation. The OTHPO method we propose is much simpler than those in Golovin et al. (2017); Zhang et al. (2019). This means we can directly evaluate the benefit of taking the ordering into account.

Before going further, we want to remind our readers that our goal is not to propose another transfer HPO method but rather to introduce a setting that is relevant for practitioners using HPO in a deployed system, and demonstrate the potential of utilising the sequential nature of the tasks.

3 Problem definition

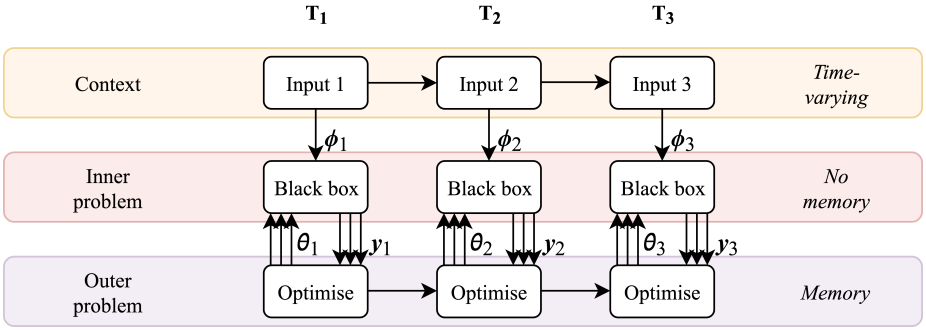


Figure 3: Structure of an ordered transfer HPO problem. The tasks T_1, T_2, T_3 have an inherent order in the outer problem stemming from the drift in the context. There is no connection between the inner problems. For the notation see Section 3.

Let $f : \Theta \to \mathbb{R}$ denote the validation performance of a machine learning algorithm after training with hyperparameters $\boldsymbol{\theta} \in \Theta$. HPO treats the search for the optimal hyperparameters as a global optimisation problem $\boldsymbol{\theta}^* = \operatorname{argmin}_{\boldsymbol{\theta} \in \Theta} f(\boldsymbol{\theta})$. The space of all possible hyperparameter configurations $\boldsymbol{\theta} \in \Theta$ is called the configuration space. Due to the intrinsic randomness of most machine learning methods, for example random weight initialisation or mini-batch sampling, we observe f only with noise: $\boldsymbol{y} = f(\boldsymbol{\theta}) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

In practice, we often face the same HPO problem repeatedly on different tasks, where the configuration space and the underlying machine learning algorithm are the same, but training and validation data sets change. To share knowledge across HPO tasks, we treat the objective functions as a series of related global optimisation problems, see Fig. 3. More formally, we augment the definition of our objective function $f : \Theta \times \mathbb{V} \to \mathbb{R}$ by another input $\boldsymbol{\phi}_i \in \mathbb{V}$ that denotes the current task i. Now, OTHPO assumes that tasks come in a sequence, such that task i is more similar to task $i-1$ than to task $i-2$.

For task i, we collect evaluations $\boldsymbol{y}_{i,m} = f(\boldsymbol{\theta}_{i,m}; \boldsymbol{\phi}_i) + \epsilon$ through the optimiser in the outer problem. The subscript m denotes which of the M_i evaluations of the inner problem i is given. When deciding what configuration to try next, the outer optimiser has access to the evaluations of previous tasks $\{\boldsymbol{\theta}_{j,m}, \boldsymbol{y}_{j,m}\}_{j=1,m=1}^{i-1,M_j}$ and all the N finished evaluations in the current task $\{\boldsymbol{\theta}_{i,m}, \boldsymbol{y}_{i,m}\}_{m=1}^{N}$. We assume that the search space Θ remains the same between tasks.

4 Benchmarks

We propose 10 benchmarks to evaluate methods on OTHPO, summarised in Table 1. We implement our benchmarks in Syne Tune (Salinas et al., 2022), making them easily available to everyone. In this section we describe each benchmark in more depth.

	XGBoost	YAHPO	NewsVendor
Context	Training data size	Training data size	Environment settings
Inner problem	Minimise error	Maximise AUC	Simulate profit
Outer problem	HPO	HPO	Parameter optimisation
Number of tasks	28	20	9
Number of benchmarks	1	8	1

Table 1: Overview of benchmarks, using concepts from Fig. 3.

XGBoost on MNIST. In a deployed setting, one collects more training data as time passes. When refitting the model on more data it is likely that the optimal hyperparameters on earlier data sets are not optimal anymore. We propose an evaluation benchmark for this setting by training an XGBoost classifier (Chen and Guestrin, 2016) on the MNIST data set (Vanschoren et al., 2013) with increasing training set sizes. We tune four hyperparameters: max_depth, n_estimators, min_child_weight and learning_rate . We have 28 tasks, with training set sizes regularly selected in the log space, ranging from 56 to 56000 training examples (see Appendix B). Some of these tasks are shown in Fig. 1. We use surrogate models fit on 1000 hyperparameter evaluations to avoid any model training during HPO. We ensure all ten classes are represented in every task. Our optimisation metric is the number of misclassified points in a validation set comprising 14000 examples.

YAHPO. The next 8 benchmarks are drawn from YAHPO Gym (Pfisterer et al., 2022), a recently published HPO benchmark suite containing a variety of HPO problems. We focus on a subset of scenarios from RandomRobot version 2 (rbv2) because they have different training set sizes available. Our initial investigation suggests that, depending on the ML model and the data set, the top performing hyperparameters are either smoothly changing over increasing training set size or stay in a similar region. We select smoothly changing ones for our benchmarks. An example is given in Fig. 4 (top), for more see Appendix F.2. In YAHPO, surrogate models are also used to predict hyperparameter performances for faster experimentation.

We consider four diverse ML models with two data sets each, so eight data sets in total, and optimise the AUC. The models considered (Binder et al., 2020) are SVM (support vector machines), AKNN (approximate k-nearest neighbor, Malkov and Yashunin (2018)), ranger (random forest, Wright and Ziegler (2017)) and glmnet (elastic net, Friedman et al. (2010)). The data set ID will be shown next to the algorithm, e.g. SVM 1220. We create 20 tasks by gradually increasing the size of the training data set from 5% to 100 %.

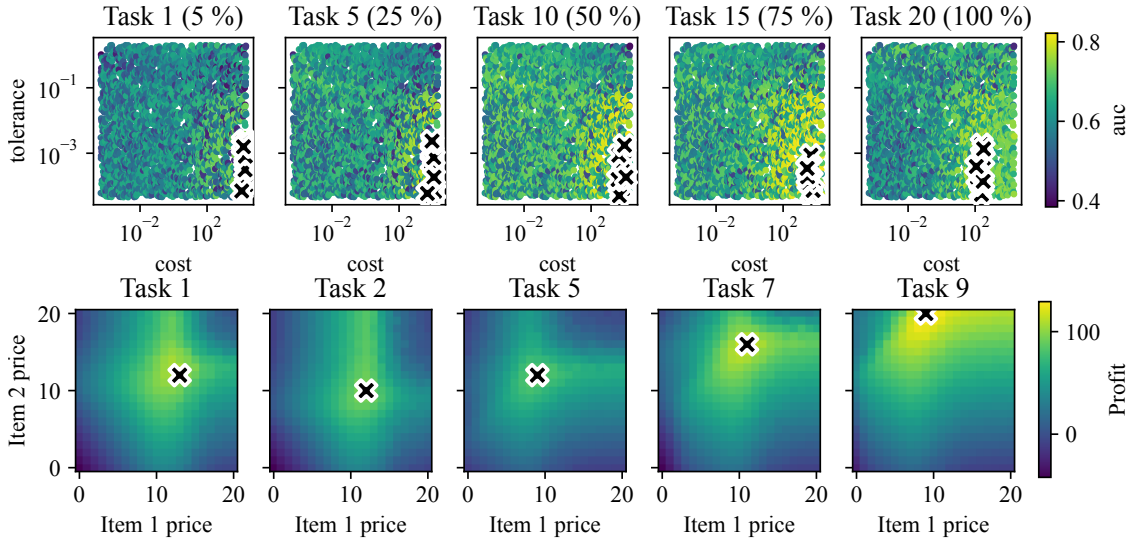


Figure 4: Hyperparameter landscapes of YAHPO SVM 1220 (top) and NewsVendor (bottom). Crosses indicate the 10 best configurations per task for YAHPO and the best one for NewsVendor.

NewsVendor. The benchmarks presented so far are focused on the change in the optimal hyperparameters of ML models as data set sizes increase. However, there are many other kinds of systems which need to be optimised periodically in evolving environmental conditions. The NewsVendor benchmark adds one such system to our set of benchmarks. In NewsVendor, the aim is to maximise profit by setting the prices for three item categories given the current uncertain demand for the

items. This benchmark is based on the Dynamic News problem from the SimOpt library that contains many simulation-optimisation problems and solvers (Eckman et al., 2023). Over time the item demand, or utility, is influenced by external factors and evolves. Here we simulate the change by following a random walk on the utility, resulting in the sequence of tasks. Note that this means the context is not necessarily changing in a single direction as for XGBoost and YAHPO. This is illustrated in Fig. 4 (bottom). We consider a sequence of 9 tasks, with three item categories, making the outer problem 3-dimensional with an integer search space limited between 0 and 20.

5 Experiments

To show the potential gain available in transfer HPO from taking the task order into account, we compare simple OTHPO methods to non-transfer Bayesian optimisation and standard transfer HPO methods. Our code is available at `https://github.com/sighellan/syne-tune/tree/othpo-results`.

5.1 Baselines

We consider several non-transfer and transfer HPO methods from the literature as baselines:

- **RandomSearch**: Sample configurations uniformly at random from the search space.
- **BO** (Snoek et al., 2012): Run Bayesian optimisation with no transfer between tasks.
- **BoundingBox** (Perrone et al., 2019): Shrink the search space of BO to the bounding box of optima on previous tasks. Note that this means the search space cannot increase for future tasks and it requires two finished tasks so a box can be computed.
- **ZeroShot** (Wistuba et al., 2015b): Learn a portfolio of complementary hyperparameters with greedy selection based on previous tasks performance. The configurations of the portfolio are then evaluated sequentially.
- **CTS** (Copula Thompson Sampling, Salinas et al. (2020)): Map the evaluations to quantiles within each task and learn a probabilistic model to predict the quantiles. For a new task, the method then samples the performance of each candidate configuration and picks the configuration with the lowest sampled value.

We use implementations in Syne Tune for our baselines and BoTorch (Balandat et al., 2020) for BO and the transfer learning methods relying on BO. We use a Matérn 5/2 kernel and a Monte Carlo version of expected improvement (Jones et al., 1998), see Appendix E. The HPO tasks are sequentially evaluated, with the evaluations collected in one task available for the subsequent tasks.

5.2 Simple ordered transfer HPO methods

- **TransferBO**: *Extend the BO surrogate model to also take the task order as an input feature.* All the evaluations from previous tasks are used to train a GP where the task order is explicitly modelled through a task feature: For NewsVendor, we use the task index as the feature; For the other benchmarks, we use the training set size. When the tasks are close in the task feature, they have more impact on each other than the tasks that are distant. The idea of modelling training set sizes in the surrogate model have been widely used (Klein et al., 2017, 2020). We use the simplest form where the same kernel function is applied on both hyperparameters and training set sizes.
- **SimpleOrdered**: *Standard BO, but start by evaluating top-performing hyperparameters from the previous tasks.* For a new task, the first N hyperparameter configurations come from the top hyperparameter configurations of each of the previous N tasks, starting with the most recent one and continuing in reverse order of time. If there are not N previous tasks yet, we continue by taking second-optimal points from those available and so on. If there are repeated optimal configurations we skip to earlier tasks. Then we continue with standard BO. We set $N = 5$ in our experiments. We also test a variant **SimpleOrderedShuffled** in ablations where we still take top configuration from previous tasks, but in random order.

- **SimplePrevious**: *Same as SimpleOrdered, but only using the last previous task.* The initial $N = 5$ hyperparameters are the best N hyperparameters from the previous task. It is a simplified version of Feurer et al. (2015) where meta-feature computation can be avoided due to our assumption that the closest task in the sequence is the most similar task. We also test a variant **SimplePreviousNoBO** which uses all the hyperparameters from the previous task, sorted by decreasing performance, without any BO.

5.3 Experimental setup and metric

Experimental setup: For each benchmark described in Section 4, we sequentially apply HPO and transfer HPO methods for each tuning task with the number of hyperparameter evaluations restricted to 25. We rerun each experiment with 50 seeds and report average performance ± 2 standard errors when plotting method results. The transfer learning methods assume evaluations from at least one previous task. All these methods therefore use BO to collect evaluations on the first task. BoundingBox also uses it on the second task, see Appendix C.

Metric: For ease of aggregation, we use a version of the Normalised Score from (Cowen-Rivers et al., 2022, eq. 3). Let i index tasks, m index iterations within a HPO task and M be the maximum number of hyperparameter evaluations for each task. The score is defined as $100 * (L_{i,m} - L_{i,M}^{\text{best}})/(L_{i,M}^{\text{RS}} - L_{i,M}^{\text{best}})$, where $L_{i,m}$ is the mean loss across replications for a method on task i at iteration m, $L_{i,M}^{\text{best}}$ is the estimated best solution for the task and $L_{i,M}^{\text{RS}}$ is the mean performance of RandomSearch on the task by the final iteration. For $L_{i,M}^{\text{best}}$ we use the best mean obtained across the compared methods.

The Normalised Score computes the loss distance at an iteration to the best solution, normalised by the loss distance between RandomSearch at the final iteration M and the best solution, thus the smaller the better, ideally 0. It allows easy comparison between minimisation (XGBoost) and maximisation (NewsVendor, YAHPO) benchmarks, and also normalises tasks by difficulty and away from the scale of the optimisation metric. We present results after 1 and 10 iterations. Fig. 5 compares all ten methods on NewsVendor, XGBoost and SVM 1220 after 1 iteration. Fig. 7 compares the best OTHPO methods, SimpleOrdered and SimplePrevious, on the eight YAHPO combinations after 1 iteration. Fig. 6 compares SimpleOrdered and TransferBO to the top-performing standard transfer HPO method CTS after 1 and 10 iterations. Further results are given in Appendix G.

5.4 First evaluation: SimpleOrdered and SimplePrevious beat standard transfer HPO

Using ordering gives a large benefit at the first evaluation. This can be seen in Fig. 5 and Fig. 6 (top). We first note that all the transfer HPO methods, including ours, outperform non-transfer methods. SimpleOrdered mostly beats — or at least is on par compared to — all the baselines. For NewsVendor, SimpleOrdered does slightly worse than for the other benchmarks. This might be because the number of previous tasks that can be used is relatively small, and the changes between tasks is not a gradual shift in one direction like in the other benchmarks. We note that SimpleOrdered is a very simple method in comparison to the standard transfer HPO baseline methods. TransferBO performs worse than SimpleOrdered. This shows that TransferBO is not able to use the task index effectively. We investigate the sampling pattern in Appendix G.6.

In Fig. 5 we see that SimpleOrdered and SimplePrevious outperform standard transfer HPO. We compare SimpleOrdered and SimplePrevious in Fig. 7, and see that the performance is very similar, with SimpleOrdered slightly preferable on early tasks and SimplePrevious on later tasks. This highlights the large impact of the most recent task.

We also further analyse the impact of ordering and the surrogate in our ablations in Appendix G.1, which shows that the lack of ordering in SimpleOrderedShuffled is detrimental, as is the lack of BO in SimplePreviousNoBO.

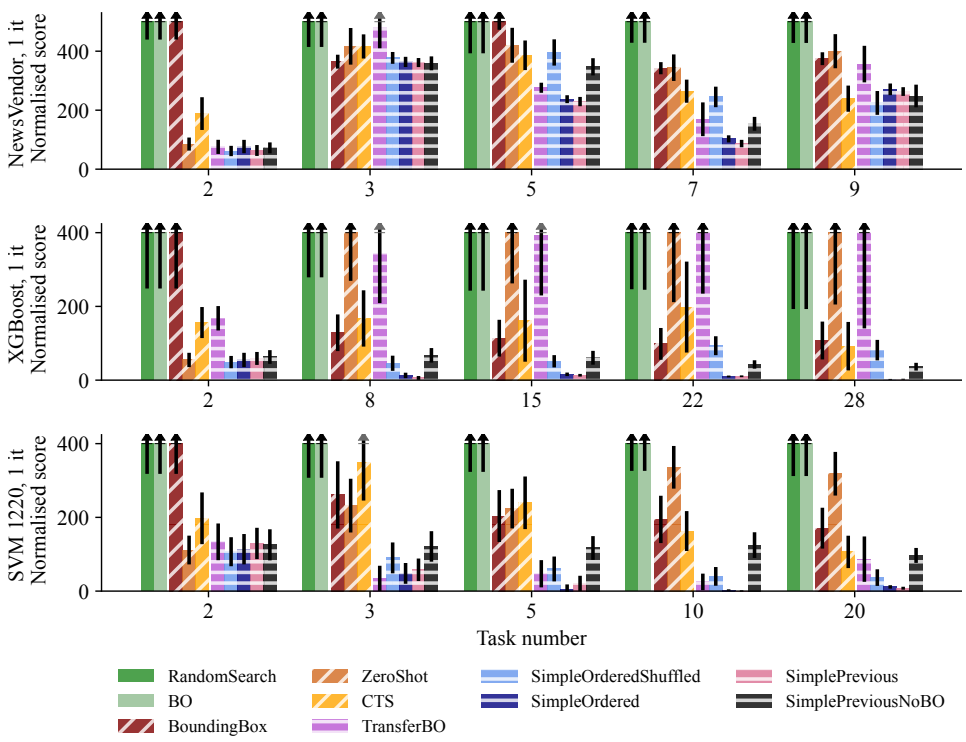


Figure 5: Mean normalised scores after first configuration (lower is better). NewsVendor (top), XGBoost (middle) and SVM 1220 (bottom). Black arrows indicate that the mean was above the plotted range, grey arrows that the standard error range was above. The methods are grouped by colour with OTHPO methods in blue, purple and pink, standard transfer HPO methods in yellow and red, and methods with no transfer in green. Task 2 is the first transfer task.

5.5 Ordered advantage reduces with more evaluations

While SimpleOrdered and SimplePrevious are clearly better after the first evaluation, this becomes benchmark-specific after the fifth evaluation. As can be seen in Fig. 6 (bottom), TransferBO is best for NewsVendor, CTS is best for XGBoost and SimpleOrdered is best for SVM 1220. We investigate this further in Appendices G.2 to G.4. It is difficult to declare a method best because the variance between runs and between benchmarks becomes too high. The general trend is that SimpleOrdered and SimplePrevious pick better first configurations, and the other methods catch up with more evaluations. But SimpleOrdered and SimplePrevious remain reliable choices even for greater number of evaluations. While TransferBO and CTS beat them on individual benchmarks, they do worse on other ones. From this we conclude that; a) there is no universal method to prefer after five evaluations, it depends on the set of tasks; and b) there is scope for improved methods combining SimpleOrdered with either TransferBO or CTS to come up with a stronger method. We also expect other more sophisticated OTHPO methods to be able to outperform SimpleOrdered.

5.6 How much better will the trained models be?

The normalised score is very useful for comparing methods across tasks and benchmarks. But it abstracts away the potential performance gain, leaving the question: *how much better will my model perform after the first configuration if I use SimpleOrdered instead of CTS?*

There are two benefits: lower variance and improved mean. Using the metrics of the underlying tasks, we get a mean improvement of 21.7% in profit for NewsVendor, 22.5 % in number of misclassified points in XGBoost and 5.8 % in AUC for SVM 1220. The lower variance is also very

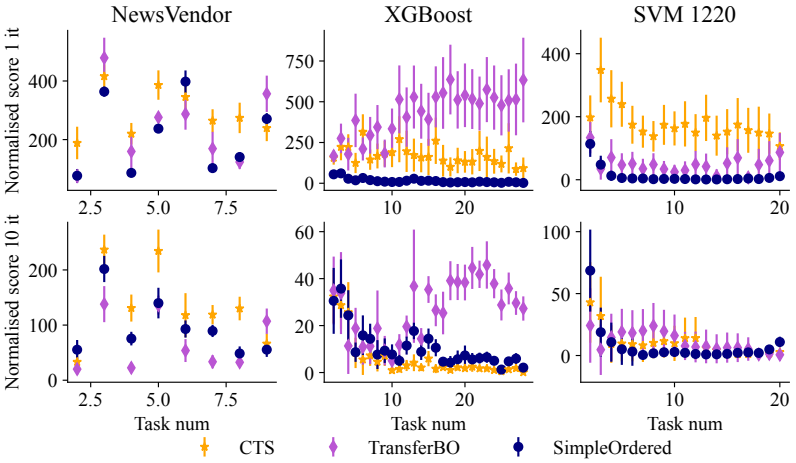


Figure 6: Mean normalised scores after first (top) and tenth (bottom) configuration. Top: SimpleOrdered clearly outperforms TransferBO and CTS. Bottom: For NewsVendor and XGBoost we see that TransferBO and CTS, respectively, are able to find better solutions than SimpleOrdered.

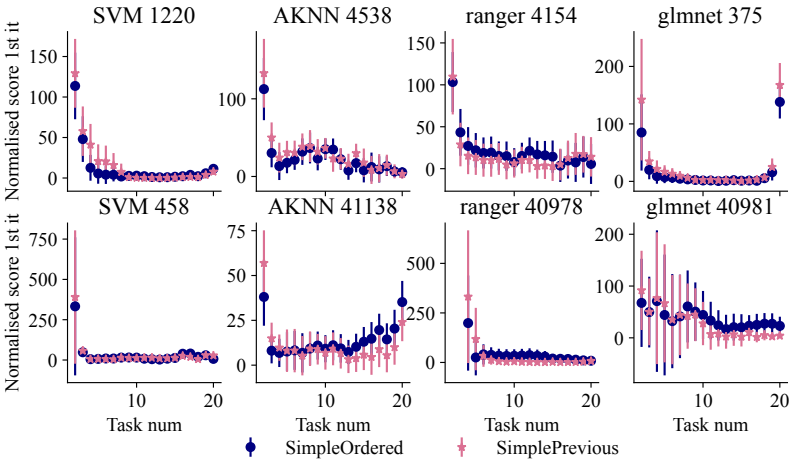


Figure 7: Mean normalised scores after first configuration evaluation for all eight YAHPO combinations considered. We see that for SimpleOrdered and SimplePrevious perform very similarly.

valuable, as it reduces the need to retrain models with multiple seeds to get a good model. The reduction in standard error of using SimpleOrdered instead of CTS is 61.3 % for NewsVendor, 92.5 % for XGBoost and 89.4 % for SVM 1220. We summarise these numbers in a table in Appendix G.5, where we also give error bounds. The higher variance of CTS is also visible in Figs. 5 and 6.

6 Conclusion

We introduced the problem of ordered transfer learning, motivated by the need of tuning regularly deployed models over time. We proposed a novel set of benchmarks to evaluate the performance of HPO methods in this setting, containing a blackbox optimisation problem, as well as HPO of XGBoost, SVM, random forest, elastic net and approximate k-nearest neighbor. We illustrated the key difference with standard transfer HPO approaches and showed how simple methods taking the order into account can outperform more sophisticated transfer methods by better tracking smooth shifts of the hyperparameter landscape. We hope that our simple methods will be useful to enable

the regular tuning of deployed methods while containing tuning costs, and that the benchmarks will enable evaluating methods in this practical setting.

Practical recommendation: Our results show that in this setting of accumulating data, SimpleOrdered performs well, especially on early iterations for a new task. We therefore recommend practitioners to start with this simple method before trying more sophisticated ones.

7 Limitations

We focused on a situation that commonly occurs for deployed models, namely that the size of the training data increases over time, but other sequences of tasks should also be explored, e.g. when the model capacity or number of classes increases over time. In addition, while we show good performance for the simple method proposed, we believe further gain can be achieved with more sophisticated methods that decide for instance whether tuning is needed on the new tasks and automatically terminates in cases where tasks do not change, similarly to Makarova et al. (2022).

8 Broader Impact Statement

The intended impact is to make a subset of transfer HPO problems more efficient, with the positive impact of reducing energy consumption from training models. There is a risk that this will instead lead to the same computational budget being used and higher accuracies obtained, but that is just the lack of a benefit, not a harm in itself. However, we also show the benefit of doing hyperparameter optimisation for subsequent tasks as data set sizes increase. This could lead to more models being trained. By contributing simulation-based benchmarks the total energy consumption of future work should be reduced as the models do not need to be retrained. We see no ethical concerns with the data sets used.

9 Submission Checklist

1. For all authors...

   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] [We also itemise our contributions in the introduction.]

   (b) Did you describe the limitations of your work? [Yes] [See Section 7.]

   (c) Did you discuss any potential negative societal impacts of your work? [Yes] [See the Broader Impact Statement.]

   (d) Have you read the ethics author's and review guidelines and ensured that your paper conforms to them? `https://automl.cc/ethics-accessibility/` [Yes] [The ethics guidelines at `https://2023.automl.cc/ethics/`.]

2. If you are including theoretical results...

   (a) Did you state the full set of assumptions of all theoretical results? [N/A]

   (b) Did you include complete proofs of all theoretical results? [N/A]

3. If you ran experiments...

   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results, including all requirements (e.g., `requirements.txt` with explicit version), an instructive `README` with installation, and execution commands (either in the supplemental material or as a URL)? [Yes] [These are given at `https://github.com/sighellan/syne-tune/tree/othpo-results`, as stated in Section 5. We provide requirement files for running the experiments both locally and remotely, and a README with step-by-step instructions.]

   (b) Did you include the raw results of running the given instructions on the given code and data? [Yes] [These are available together with the rest of the code at `https://github.com/sighellan/syne-tune/tree/othpo-results`.]

   (c) Did you include scripts and commands that can be used to generate the figures and tables in your paper based on the raw results of the code, data, and instructions given? [Yes] [For all plots containing data. Two of our figures (Figs. 2 and 3) are illustrative and do not include data; scripts are not included for these. In the README we provide a list of what scripts generate what figures.]

   (d) Did you ensure sufficient code quality such that your code can be safely executed and the code is properly documented? [Yes] [See the README for more details.]

   (e) Did you specify all the training details (e.g., data splits, pre-processing, search spaces, fixed hyperparameter settings, and how they were chosen)? [Yes] [This is also fully specified in the code to make it reproducible.]

   (f) Did you ensure that you compared different methods (including your own) exactly on the same benchmarks, including the same datasets, search space, code for training and hyperparameters for that code? [Yes] [We collect the results for the different methods on all benchmarks ourselves, and keep the settings of the benchmarks constant.]

   (g) Did you run ablation studies to assess the impact of different components of your approach? [Yes] [We provide results both for SimpleOrdered and for a non-order version, SimpleOrderedShuffled. And for SimplePrevious and a version without BO, SimplePreviousNoBO. See Appendix G.1.]

(h) Did you use the same evaluation protocol for the methods being compared? [Yes] [This was automated in the file preprocess_results .py.]

(i) Did you compare performance over time? [Yes] [We discuss this in Section 5.5, and present additional results in Appendices G.2 to G.4.]

(j) Did you perform multiple runs of your experiments and report random seeds? [Yes] [We rerun each experiment 50 times. The seeds are given in the code and in Appendix C.]

(k) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] [Our plots contain error bars. The error range for the values in Section 5.6 are given in Appendix G.5.]

(l) Did you use tabular or surrogate benchmarks for in-depth evaluations? [Yes] [We used the YAHPO surrogate benchmark, and used Syne Tune to generate a surrogate on top of our XGBoost evaluations.]

(m) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] [This is given in Appendix H.]

(n) Did you report how you tuned hyperparameters, and what time and resources this required (if they were not automatically tuned by your AutoML method, e.g. in a NAS approach; and also hyperparameters of your own method)? [N/A] [We did not tune the hyperparameters of our methods. SimpleOrdered and SimpleOrderedShuffled have a parameter N which we set to 5 without tuning.]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] [These are cited in the paper. They are also repeated in Appendix A.]

(b) Did you mention the license of the assets? [Yes] [We list the licenses of the assets used in Appendix A.]

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes] [We include all the code and results obtained as supplementary material.]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] [The only data set we handle is MNIST. There are other data sets listed in Appendix A, but we do not use these directly – they are listed for completeness. We only use the models of hyperparameter performance in YAHPO. These data sets were used to learn the published YAHPO models.]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] [This is discussed in Appendix A. It does not.]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Acknowledgements. The authors would like to thank Jan Gasthaus, Valerio Perrone, Martin Wistuba and Michael Bohlke-Schneider for help with the project.

Hellan was supported by the EPSRC Centre for Doctoral Training in Data Science, funded by the UK Engineering and Physical Sciences Research Council (grant EP/L016427/1) and the University of Edinburgh.

References

Bai, T., Li, Y., Shen, Y., Zhang, X., Zhang, W., and Cui, B. (2023). Transfer learning for Bayesian optimization: A survey. *arXiv preprint arXiv:2302.05927*.

Balandat, M., Karrer, B., Jiang, D. R., Daulton, S., Letham, B., Wilson, A. G., and Bakshy, E. (2020). BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*.

Binder, M., Pfisterer, F., and Bischl, B. (2020). Collecting empirical data about hyperparameters for data driven AutoML. *Democratizing Machine Learning Contributions in AutoML and Fairness*, page 93.

Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P. K., Torr, P. H., and Ranzato, M. (2019). On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*.

Chen, T. and Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA. ACM.

Cowen-Rivers, A. I., Lyu, W., Tutunov, R., Wang, Z., Grosnit, A., Griffiths, R. R., Maraval, A. M., Jianye, H., Wang, J., Peters, J., and Bou-Ammar, H. (2022). HEBO: pushing the limits of sample-efficient hyperparameter optimisation. *Journal of Artificial Intelligence Research*, 74:1269–1349.

Dua, D. and Graff, C. (2017). UCI machine learning repository.

Eckman, D. J., Henderson, S. G., and Shashaani, S. (2023). Simopt: A testbed for simulation-optimization experiments. *INFORMS Journal on Computing*.

Feurer, M. and Hutter, F. (2019). Hyperparameter optimization. *Automated machine learning: Methods, systems, challenges*, pages 3–33.

Feurer, M., Letham, B., Hutter, F., and Bakshy, E. (2018). Practical transfer learning for Bayesian optimization. *arXiv preprint arXiv:1802.02219*.

Feurer, M., Springenberg, J., and Hutter, F. (2015). Initializing Bayesian hyperparameter optimization via meta-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29.

Friedman, J., Hastie, T., and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22.

Golovin, D., Solnik, B., Moitra, S., Kochanski, G., Karro, J., and Sculley, D. (2017). Google Vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1487–1495.

Horváth, S., Klein, A., Richtárik, P., and Archambeau, C. (2021). Hyperparameter transfer learning with adaptive complexity. In *International Conference on Artificial Intelligence and Statistics*, pages 1378–1386. PMLR.

Jamieson, K. and Talwalkar, A. (2016). Non-stochastic best arm identification and hyperparameter optimization. In *Artificial intelligence and statistics*, pages 240–248. PMLR.

Jomaa, H. S., Arango, S. P., Schmidt-Thieme, L., and Grabocka, J. (2021). Transfer learning for Bayesian HPO with end-to-end landmark meta-features. In *Fifth Workshop on Meta-Learning at the Conference on Neural Information Processing Systems*.

Jones, D. R., Schonlau, M., and Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455.

Klein, A., Falkner, S., Bartels, S., Hennig, P., and Hutter, F. (2017). Fast Bayesian optimization of machine learning hyperparameters on large datasets. In *Artificial intelligence and statistics*, pages 528–536. PMLR.

Klein, A., Tiao, L. C., Lienart, T., Archambeau, C., and Seeger, M. (2020). Model-based asynchronous hyperparameter and neural architecture search. *arXiv preprint arXiv:2003.10865*.

Kudo, M., Toyama, J., and Shimbo, M. (1999). Multidimensional curve classification using passing-through regions. *Pattern Recognition Letters*, 20(11-13):1103–1111.

Kushmerick, N. (1999). Learning to remove internet advertisements. In *Proceedings of the third annual conference on Autonomous Agents*, pages 175–181.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. (2017). Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816.

Madeo, R. C., Lima, C. A., and Peres, S. M. (2013). Gesture unit segmentation using support vector machines: segmenting gestures from rest positions. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, pages 46–52.

Makarova, A., Shen, H., Perrone, V., Klein, A., Faddoul, J. B., Krause, A., Seeger, M., and Archambeau, C. (2022). Automatic termination for hyperparameter optimization. In Guyon, I., Lindauer, M., van der Schaar, M., Hutter, F., and Garnett, R., editors, *Proceedings of the First International Conference on Automated Machine Learning*, volume 188 of *Proceedings of Machine Learning Research*, pages 7/1–21. PMLR.

Malkov, Y. A. and Yashunin, D. A. (2018). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836.

Organizers of KDD Cup 2012 and Tencent Inc (2012). 2012 KDD Cup data set.

Perrone, V., Shen, H., Seeger, M. W., Archambeau, C., and Jenatton, R. (2019). Learning search spaces for Bayesian optimization: Another view of hyperparameter transfer learning. *Advances in Neural Information Processing Systems*, 32.

Pfisterer, F., Schneider, L., Moosbauer, J., Binder, M., and Bischl, B. (2022). YAHPO Gym – an efficient multi-objective multi-fidelity benchmark for hyperparameter optimization. In *International Conference on Automated Machine Learning*, pages 3–1. PMLR.

Quinlan, J. R. (1987). Simplifying decision trees. *International journal of man-machine studies*, 27(3):221–234.

Salinas, D., Seeger, M., Klein, A., Perrone, V., Wistuba, M., and Archambeau, C. (2022). Syne Tune: A library for large scale hyperparameter tuning and reproducible research. In *First Conference on Automated Machine Learning (Main Track)*.

Salinas, D., Shen, H., and Perrone, V. (2020). A quantile-based approach for hyperparameter transfer learning. In *International conference on machine learning*, pages 8438–8448. PMLR.

Simonoff, J. S. (2003). *Analyzing categorical data*, volume 496. Springer.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25.

Springenberg, J. T., Klein, A., Falkner, S., and Hutter, F. (2016). Bayesian optimization with robust Bayesian neural networks. In *Proceedings of the 29th International Conference on Advances in Neural Information Processing Systems (NIPS'16)*.

Stoll, D., Franke, J. K. H., Wagner, D., Selg, S., and Hutter, F. (2020). Hyperparameter transfer across developer adjustments. *arXiv preprint arXiv:2010.13117*.

Tiao, L. C., Klein, A., Seeger, M. W., Bonilla, E. V., Archambeau, C., and Ramos, F. (2021). BORE: Bayesian optimization by density-ratio estimation. In *International Conference on Machine Learning*, pages 10289–10300. PMLR.

Van de Ven, G. M. and Tolias, A. S. (2019). Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*.

Vanschoren, J., van Rijn, J. N., Bischl, B., and Torgo, L. (2013). OpenML: networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60.

Wistuba, M. and Grabocka, J. (2021). Few-shot Bayesian optimization with deep kernel surrogates. *arXiv preprint arXiv:2101.07667*.

Wistuba, M., Schilling, N., and Schmidt-Thieme, L. (2015a). Learning data set similarities for hyperparameter optimization initializations. In *Metasel@ pkdd/ecml*, pages 15–26.

Wistuba, M., Schilling, N., and Schmidt-Thieme, L. (2015b). Sequential model-free hyperparameter tuning. In *2015 IEEE international conference on data mining*, pages 1033–1038. IEEE.

Wright, M. N. and Ziegler, A. (2017). ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software*, 77:1–17.

Zappella, G., Salinas, D., and Archambeau, C. (2021). A resource-efficient method for repeated hpo and nas problems. *arXiv:2103.16111 [cs.LG]*.

Zhang, Y., Jordon, J., Alaa, A. M., and van der Schaar, M. (2019). Lifelong Bayesian optimization. *arXiv preprint arXiv:1905.12280*.

A Assets used

We use benchmarks from three families:

- YAHPO (Pfisterer et al., 2022) (Apache License 2.0)
  - We only handle the surrogate benchmarks in YAHPO, not the data used to generate them. But we list these data sets here. For the data sets citing Dua and Graff (2017) the authors of the repository request citation.

  - Data set 1220 (Organizers of KDD Cup 2012 and Tencent Inc, 2012). Requires attribution, and the data are restricted to be used for scientific research purposes only. License: Public
  - Data set 4538. Requires citation of (Dua and Graff, 2017) and a relevant paper, e.g. (Madeo et al., 2013) License: Public
  - Data set 4154. License: Public
  - Data set 375 (Dua and Graff, 2017; Kudo et al., 1999). Author asked to be informed about published work using the data. License: Public
  - Data set 458 (Simonoff, 2003). Requires attribution, and data are restricted to be used for scientific, educational and/or noncommercial purposes. License: Public
  - Data set 41138 (Dua and Graff, 2017). License: GNU GPL v3
  - Data set 40978 (Dua and Graff, 2017; Kushmerick, 1999). License: Public
  - Data set 40981 (Dua and Graff, 2017; Quinlan, 1987). License: Public
- NewsVendor, based on SimOpt (Eckman et al., 2023) (MIT license):
  - No underlying data set. We generate the changing utilities using a random walk.
- XGBoost (Chen and Guestrin, 2016) (Apache License 2.0)
  - Data set: mnist_784 downloaded from (Vanschoren et al., 2013)

The only data we handled was the MNIST data. It does not contain any personally identifiable or offensive content.

B XGBoost evaluations collection

For our XGBoost benchmark we collected evaluations of 1000 hyperparameter configurations on 28 training data set sizes, which we used as the basis of our simulation. The hyperparameter configurations were randomly selected and evaluated on each of the 28 data set sizes. The search space for the hyperparameters is given in Table 2 and the data set sizes in Table 3.

Hyperparameter	Type	Min	Max	Scaling
learning_rate	Cont.	1e-6	1	log
min_child_weight	Cont.	1e-6	32	log
max_depth	Int	2	32	log
n_estimators	Int	2	256	log

Table 2: Hyperparameter search space used for XGBoost.

56	72	93	120	155	201	259	335	433
560	723	934	1206	1558	2012	2599	3357	4335
5600	7232	9341	12064	15582	20125	25992	33571	43358
56000								

Table 3: Data set sizes evaluated for XGBoost.

C Additional details on the experimental setup

Seeds: We rerun each method on each benchmark 50 times. We use integers between 0 and 49 as seeds.

The transfer learning methods require evaluations from previous tasks. We use BO to collect these. For all methods except BoundingBox this was done for one task (task 1). For BoundingBox we do two tasks (tasks 1 and 2), as otherwise the method collapses to only trying the best evaluation from the first task on any of the future tasks. More specifically, we run BO until we have different

optima for at least two tasks, as otherwise we also get the situation of the bounding box only containing one configuration.

D SimpleOrdered implementation details

The first N configurations are used to evaluate the top configuration from each of the previous N tasks. We do this in reverse order, i.e. at task i we first evaluate the configuration from task $i-1$, then from task $i-2$ and so on until task $i-N$.

There are several situations which require modifications to this:

- There are $L < N$ previous tasks: in this case, we pick the top configuration from each of the L tasks, and then pick the second-best configuration from each of the tasks, until we reach N configurations. If we still don't have N configurations we continue with the third-best, and so on.
- There are repeated optimal configurations: in this case the configuration is skipped. That means that we might end up using the top configuration from task $i-(N+1)$.
- There are joint optima: this is the situation if two hyperparameter configurations both perform best. We attempt to pick the first of these configurations, but if it is a repeat it will be skipped. We then add the other joint optima to the back of the list we are considering. So if there are $L < N$ previous tasks it might get selected once we have included one optima from each of the L tasks.

The code for SimpleOrdered is available together with the rest of the paper code.

E BO details

We use a slightly updated version of the BoTorch-based (Balandat et al., 2020) BO in Syne Tune (Salinas et al., 2022), see `https://github.com/sighellan/syne-tune/blob/othpo-results/syne_tune/optimizer/schedulers/searchers/botorch/botorch_searcher.py`. The acquisition function is a Monte Carlo version of expected improvement (Jones et al., 1998), and the covariance function Matérn 5/2. We also apply input warping.

For TransferBO the maximum number of observations is set to 200. For later tasks, we subsample 200 of the past/current samples.

F Additional hyperparameter landscapes

F.1 XGBoost

Fig. 8 compares the top hyperparameters for the first and last tasks of the XGBoost benchmark. We show all six possible 2-dimensional combinations of the four hyperparameters.

F.2 YAHPO

Figs. 9 and 10 show additional YAHPO hyperparameter landscapes for the model – data set combinations used in Fig. 7. The final hyperparameter landscape used is shown in Fig. 4 (top).

G Additional results

G.1 Ablations

We perform two ablation experiments. In Fig. 11 we compare SimpleOrdered to SimpleOrdered-Shuffled, the version that takes top points from randomly chosen previous tasks. As can be seen, the ordered version does much better.

In Fig. 12 we compare SimplePrevious to SimplePreviousNoBO, the version that only considers configurations used in the previous task. That means that throughout the tasks only the configurations used by BO in the initial task are considered. As can be seen, the version that includes exploration through BO does much better.

	NewsVendor		XGBoost		SVM 1220	
Reduction in standard error SimpleOrdered over CTS (%)	61.3	(53.7 – 68.9)	92.5	(88.9 – 96.0)	89.4	(83.0 – 95.9)
Improvement in mean SimpleOrdered over CTS (%)	21.7	(8.0 – 35.4)	22.5	(18.2 – 26.7)	5.8	(4.9 – 6.7)

Table 4: Downstream comparison of SimpleOrdered and CTS after one configuration. SimpleOrdered gives an improvement on all benchmarks, and the standard error is smaller, making it more reliable. We present means +- two standard errors for each value.

G.2 Rankings over evaluations

Fig. 13 shows the mean internal rankings among the ten methods for a subset of the methods. We see that SimpleOrdered and SimplePrevious do very well at the beginning, and are then overtaken by either TransferBO or CTS.

G.3 Additional bar plots

Figs. 14 and 15 are versions of Fig. 5 for 10 and 25 configuration evaluations, respectively.

G.4 Optimisation curves

We show the optimisation curves for NewsVendor, XGBoost and SVM 1220 in Fig. 16.

G.5 Downstream performance

Table 4 shows that the downstream performance of SimpleOrdered is better than CTS. The reduction in standard error is calculated as $100(1 - s^{(i)}_{\text{SO}}/s^{(i)}_{\text{CTS}})$ for each task i where $s^{(i)}_{\text{CTS}}$ is the standard error of the performance across replications for CTS. $s^{(i)}_{\text{SO}}$ is the same for SimpleOrdered.

The improvement in mean is calculated as $100(1 - m^{(i)}_{\text{SO}}/m^{(i)}_{\text{CTS}})$ for maximisation, and the same multiplied by -1 for minimisation. Here $m^{(i)}_{\text{SO}}$ and $m^{(i)}_{\text{CTS}}$ are the means across replications for SimpleOrdered and CTS, respectively. Note that we do not include the between-seed variation in our error estimate of the improvement in mean, only the between-task variation.

G.6 Sampling locations

Fig. 17 shows sampling locations. As can be seen, SimpleOrdered and SimplePrevious combine very focused early exploitation with broad exploration later on.

G.7 Ordering with different surrogate model: Density-Ratio Estimation

This section presents ablation results of combining the ordered approach in SimpleOrdered with BO using density-ratio estimation for the surrogate model (Tiao et al., 2021): DREOrdered. Note that the figures in the main paper were not replotted with these new results, although the metrics of the other methods can be impacted by the introduction of a new method. We also did not update the values in Appendix H to include these extra experiments.

We present results for DREOrdered in Figs. 18 and 19. As can be seen, the performance of DREOrdered is between that of SimplePrevious and SimplePreviousNoBO. This suggests that Gaussian processes are better surrogate models for our problem. But comparing DREOrdered to standard transfer HPO we see that also with this surrogate model we outperform the non-ordered methods.

H Compute budget

This summaries the compute costs for the results included in the paper (not including Appendix G.7).

The experiments were run on AWS Sagemaker, using ml.c5.18xlarge compute instances, which have 72 vCPUs, and 144 GiB memory. We ran a total of 65 experiments: 10 for NewsVendor, 10 for XGBoost, 10 for SVM 1220 (YAHPO) and 5 for each of the 7 remaining YAHPO combinations, so 35. We also trained and evaluated a total of 28000 XGBoost models for our XGBoost benchmark, also on AWS Sagemaker.

Collecting the XGBoost evaluations took a total of 79 hours and 46 minutes of compute time.

Collecting the experiment evaluations took a total of 231 hours and 39 minutes.

In total, we used 311 hours and 26 minutes of compute time for the results presented.

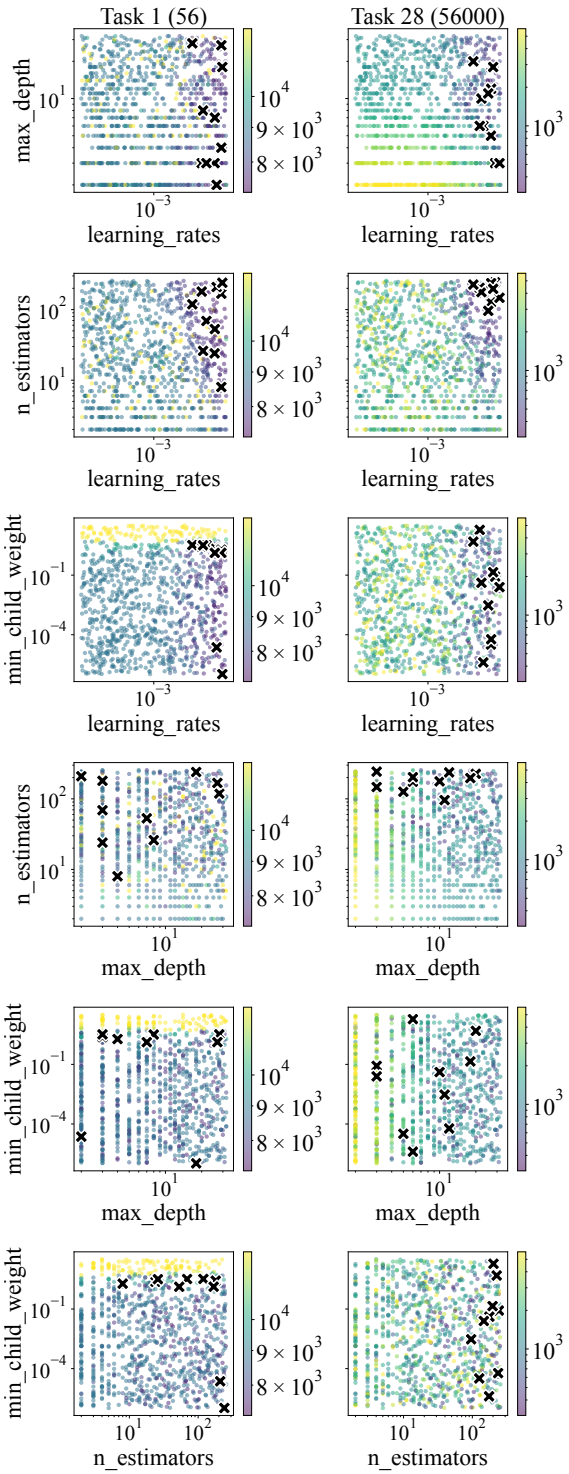


Figure 8: Hyperparameter landscape plots for all combinations of two hyperparameters of the XGBoost benchmark. Left: Task 1 with 56 data points in the training set. Right: Task 28 with 56000 data points. Black crosses indicate the top 10 configurations.

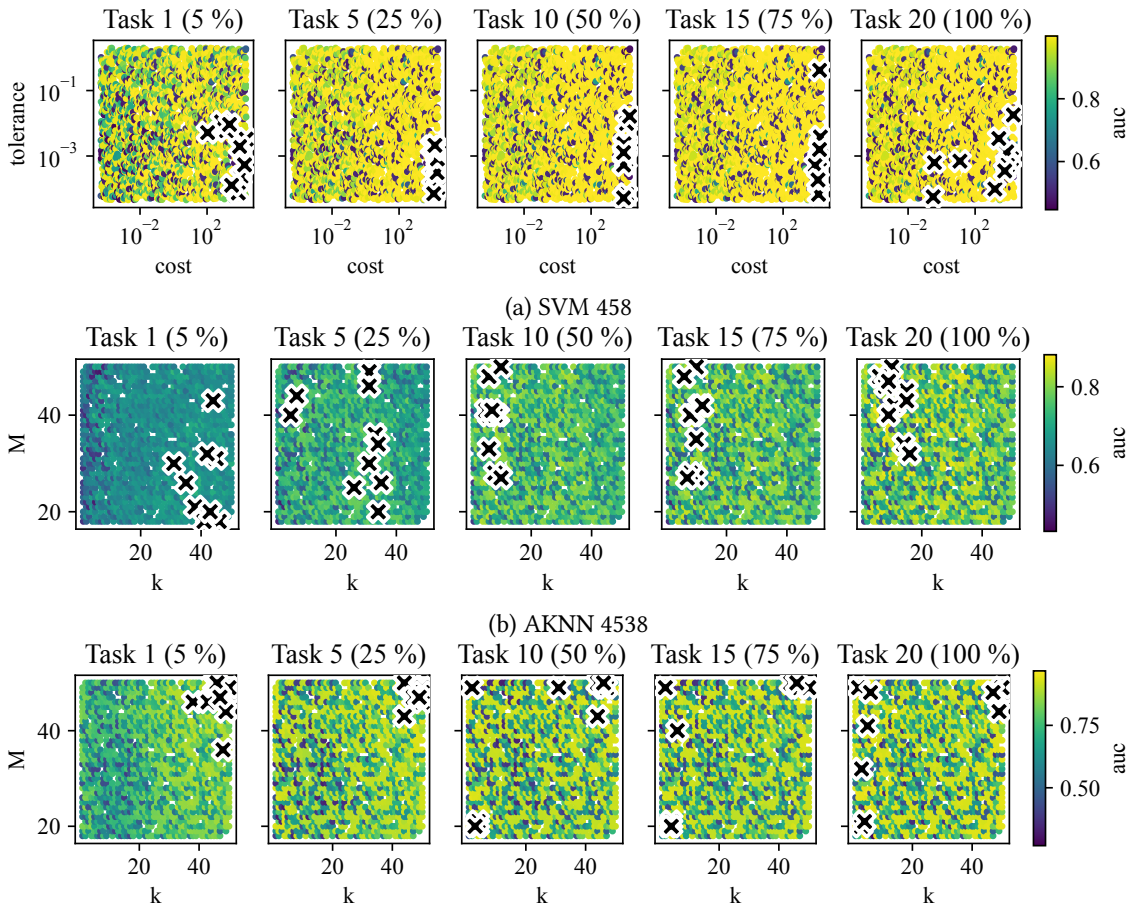


Figure 9: Further plots of YAHPO hyperparameter landscapes showing ordered behaviour on SVM and AKNN models.

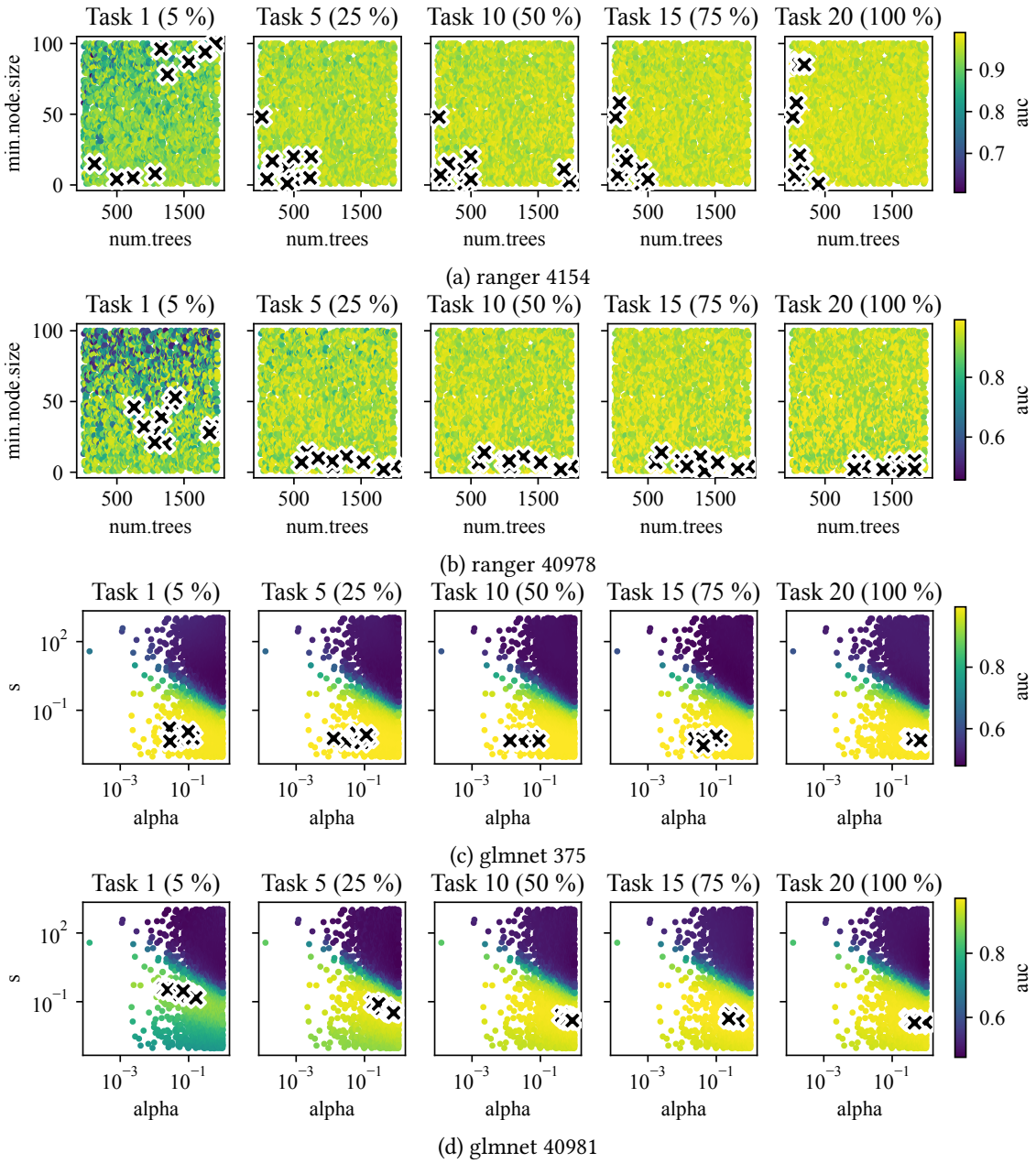


Figure 10: Further plots of YAHPO hyperparameter landscapes showing ordered behaviour on ranger and glmnet models.

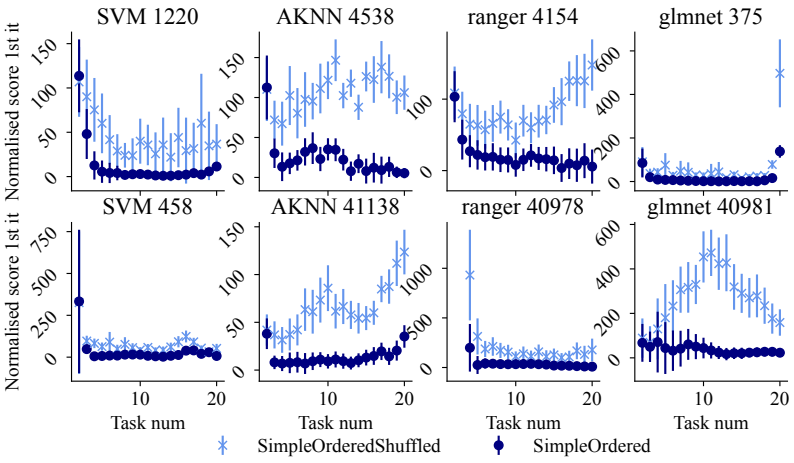


Figure 11: Mean normalised scores after first configuration evaluation for all eight YAHPO combinations considered. We see that for almost all the tasks SimpleOrdered performs better than its ablation SimpleOrderedShuffled.

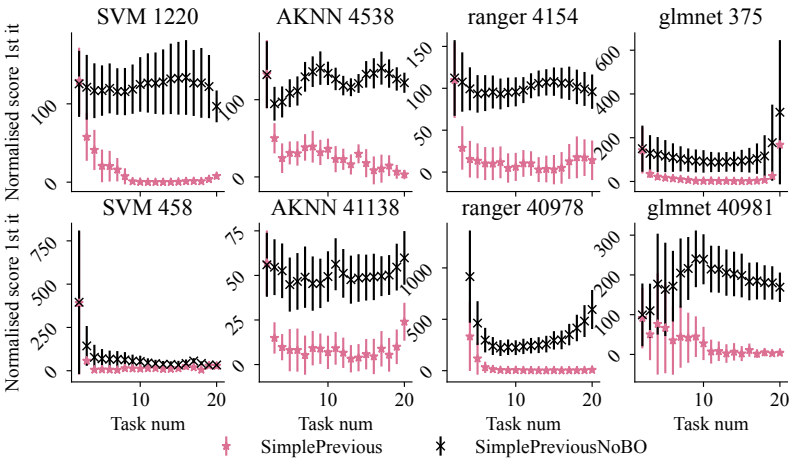


Figure 12: Mean normalised scores after first configuration evaluation for all eight YAHPO combinations considered. We see that for almost all the tasks SimplePrevious performs better than its ablation SimplePreviousNoBO.

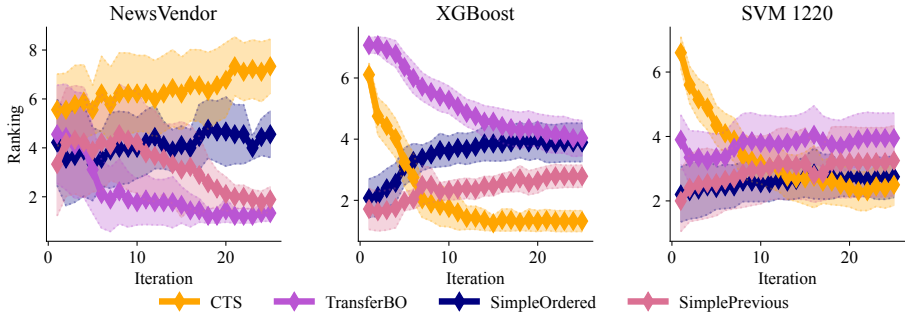


Figure 13: Mean rankings (± 2 standard error) as a function of configurations evaluated, averaged over tasks.

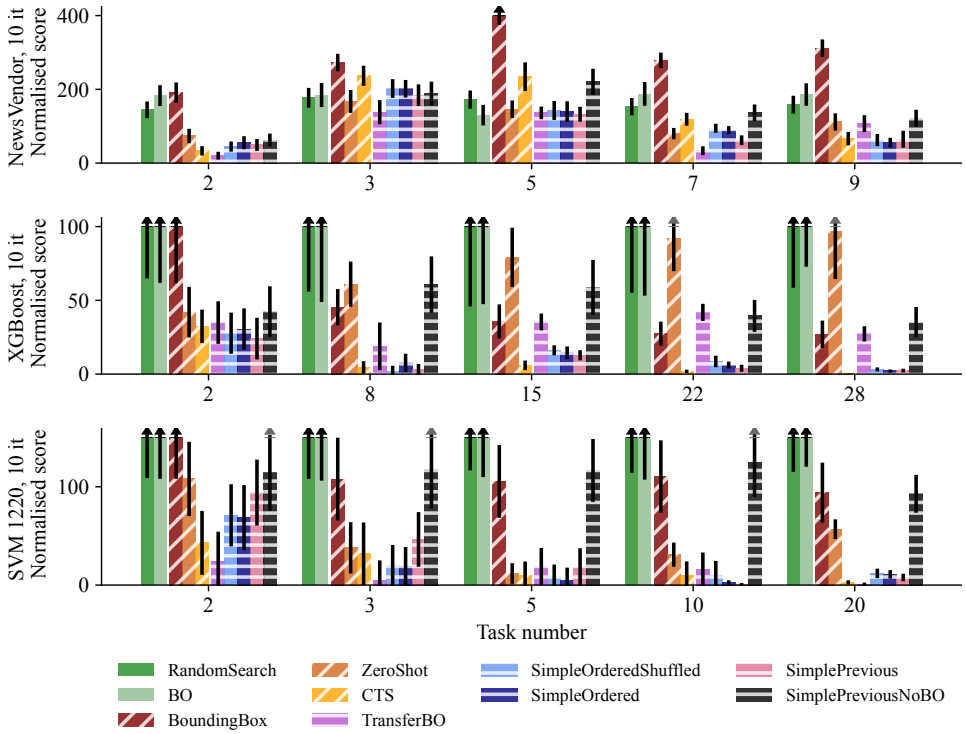


Figure 14: Mean normalised scores (+- 2 standard error) after the tenth configuration (lower is better). NewsVendor (top), XGBoost (middle) and SVM 1220 (bottom). Black arrows indicate that the mean was above the plotted range, grey arrows that the standard error range was above. Version of Fig. 5 for the tenth configuration.

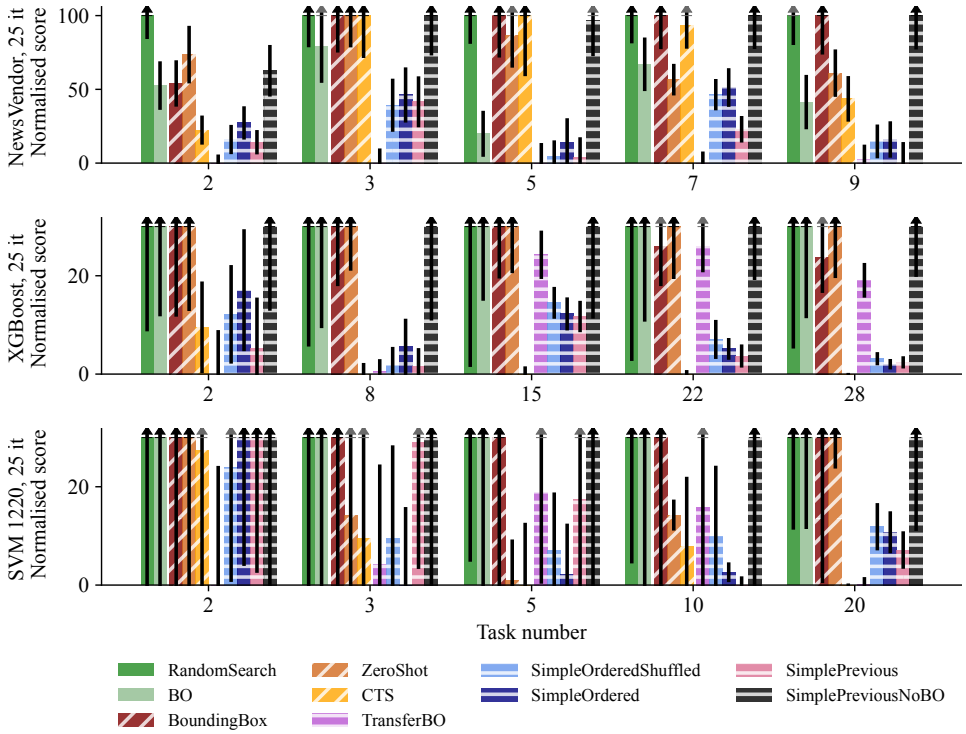
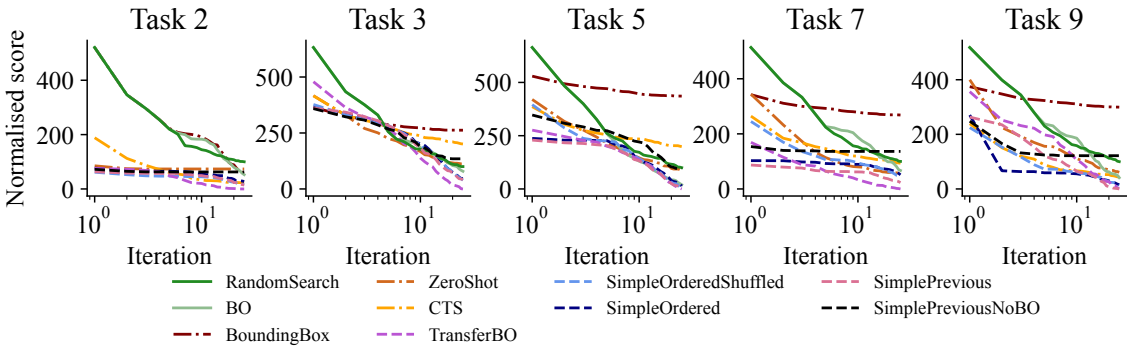
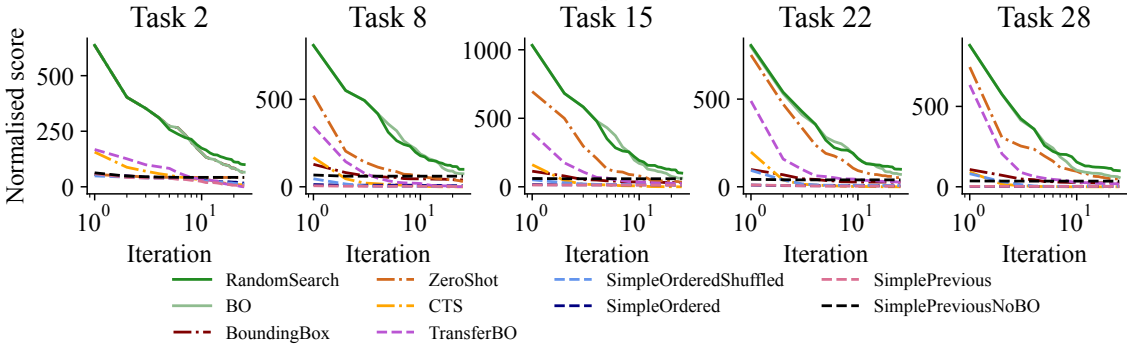


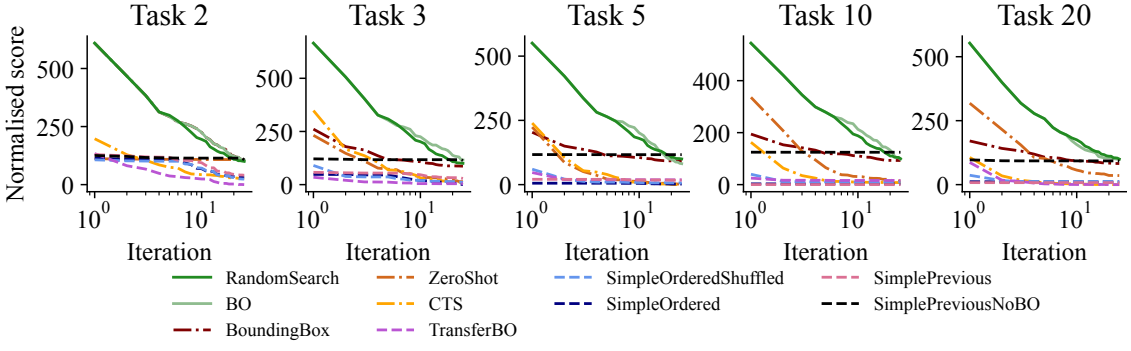
Figure 15: Mean normalised scores (+- 2 standard error) after the 25th configuration (lower is better). NewsVendor (top), XGBoost (middle) and SVM 1220 (bottom). Black arrows indicate that the mean was above the plotted range, grey arrows that the standard error range was above. Version of Fig. 5 for the 25th configuration.



(a) SimOpt



(b) XGBoost



(c) SVM 1220

Figure 16: Companion plot to Fig. 5, showing the normalised score as a function of configurations evaluated for the same subset of tasks. SimpleOrdered starts off well, but does not improve much with more iterations.

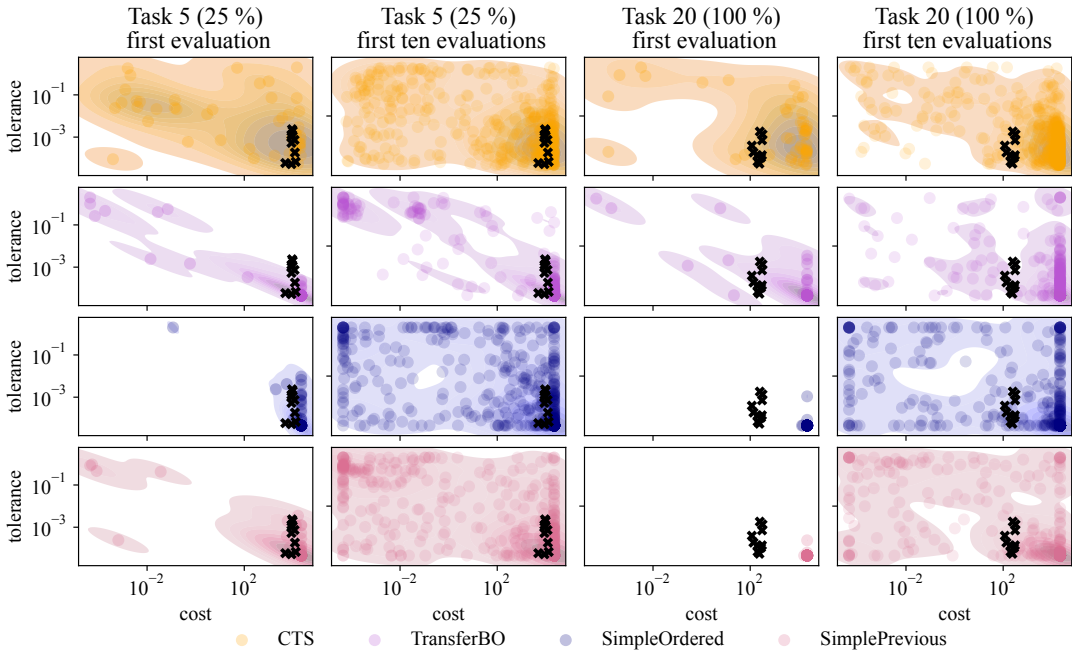


Figure 17: Sampling locations of the first and first ten evaluations for YAHPO SVM 1220 across the fifty replications. Black crosses indicate the top ten hyperparameter configurations. As can be seen, the SimpleOrdered and SimplePrevious samples are much more concentrated than the others for the first evaluation, and then is very explorative once the first five evaluations are done.

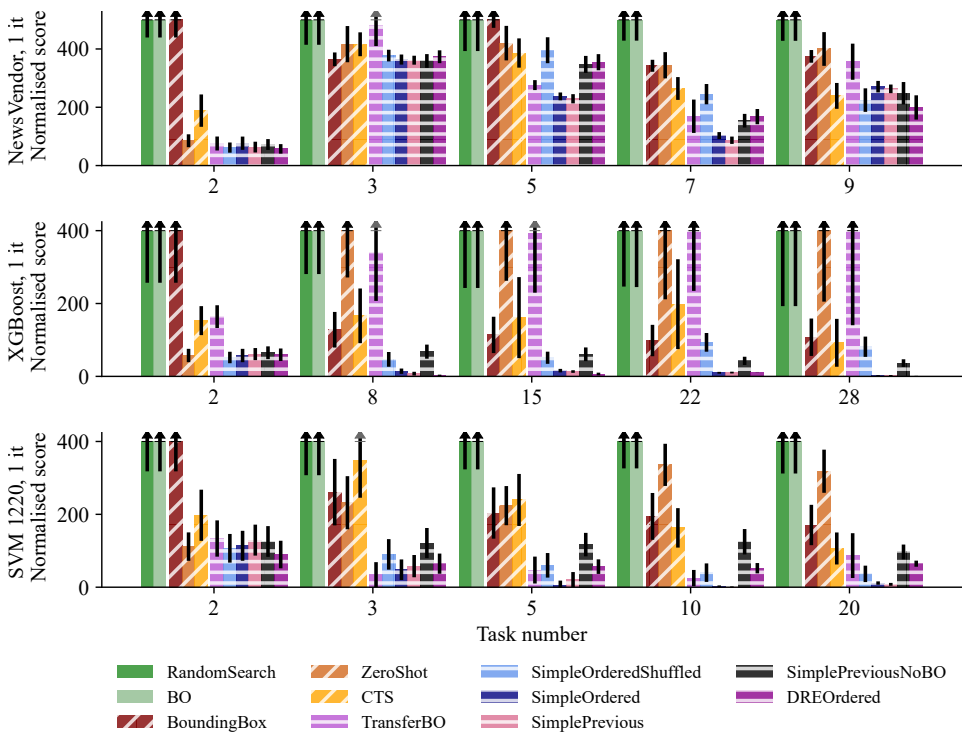


Figure 18: Mean normalised scores (+- 2 standard error) after the first configuration (lower is better). NewsVendor (top), XGBoost (middle) and SVM 1220 (bottom). Black arrows indicate that the mean was above the plotted range, grey arrows that the standard error range was above. Version of Fig. 5 with the addition of DREOrdered as a baseline.

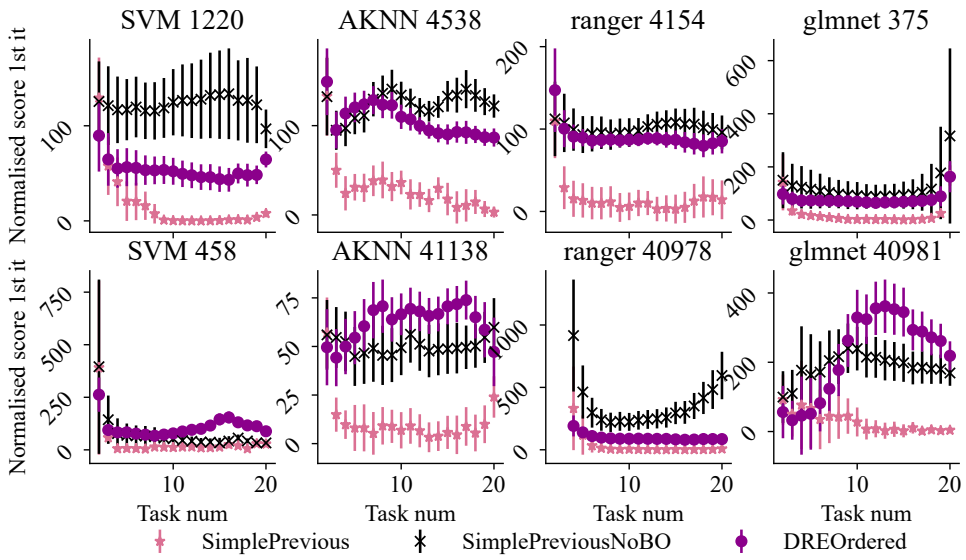


Figure 19: Comparison of new baseline DREOrdered with SimplePrevious and SimplePreviousNoBO.

6.6 Discussion

In the paper we consider a common setting where we are doing transfer learning between optimisation tasks that come in a sequence. We show that the ordering of the sequence is itself useful for the transfer.

The main methods used to introduce ordering are SimpleOrdered and SimplePrevious. The former is illustrated in Fig. 6.1. We start by identifying the best input values for each of the five previous tasks, and then start our optimisation by evaluating these before moving on to Bayesian optimisation. This Bayesian optimisation does not have access to any of the evaluations on past tasks — the only transfer is through the initial points. This makes our approach a version of transfer through initial points (see Section 2.6.1). Another example is Feurer et al. (2015), but we avoid the expensive need to learn which tasks are most related by exploiting the inherent ordering. Similarly to Feurer et al., we observe that the chosen initial points lead to an initial advantage, but that the other methods catch up with more evaluations. SimplePrevious is very similar to SimpleOrdered, but takes five points from the most recent task, instead of one point each from the five most recent tasks.

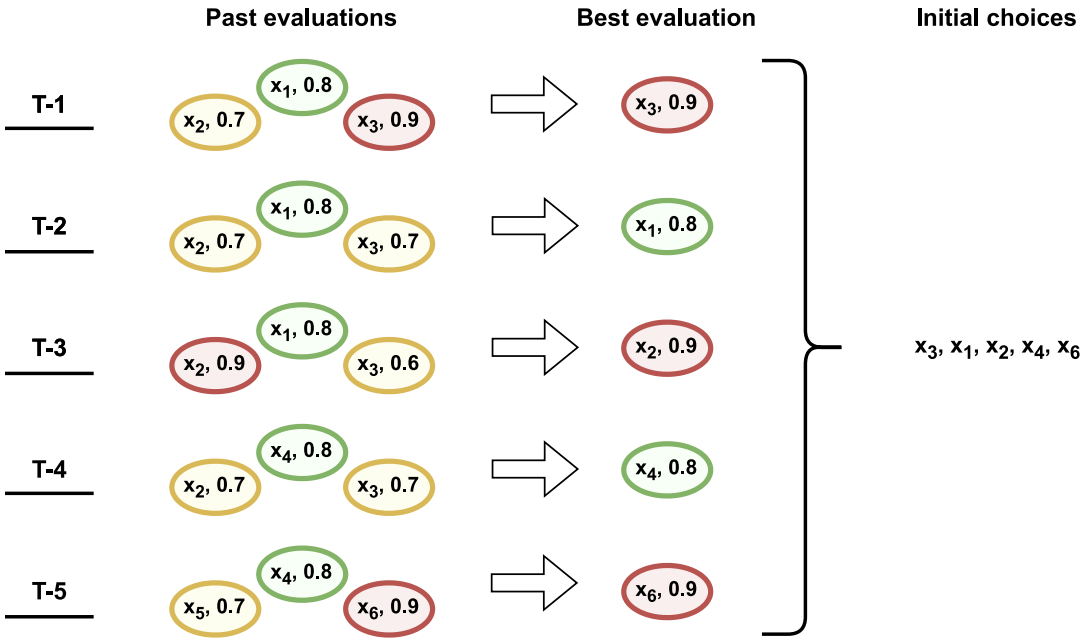


Figure 6.1: Visualisation of the SimpleOrdered method. T-1 to T-5 are the five most recent tasks. We find the best evaluation for each of these tasks, and evaluate the top configurations in order of how recent they are.

SimpleOrdered should be more robust than SimplePrevious. We assume that the optimisation tasks come in an ordered sequence. But it is reasonable to assume that there could be noise in this sequence, or that the changes are not required to be in one direction. The latter is the case for the NewsVendor benchmark. Using the five previous tasks is designed to weaken the dependence on a strict ordering assumption. For the benchmarks considered, we see that SimpleOrdered and SimplePrevious do very similarly, and their costs are the same. The memory requirements are very similar, as we only need the best points to be stored, not all the evaluations[6].

In the paper we also considered an ordered method based on a joint Gaussian process model, TransferBO. This takes the task ID as an input feature. It is a simple method that could be improved upon. For instance, the number of observations is restricted to 200 due to the cubic scaling. If there are more, observations are dropped at random from both the past and new task. Which observations are dropped changes between iterations. Better performance might be obtained by only dropping past evaluations. We also did not explore different ways of encoding the ordering and choices of covariance function. For NewsVendor and the YAHPO benchmarks the task ID was the task number. For XGBoost the task ID was the data set size.

6.7 Future work

The next step is to come up with a stronger ordered method. This can be done by combining the ordered approach with more sophisticated standard HPO transfer learning methods. For instance, we could combine the reduced search space in BoundingBox (Perrone et al., 2019) with the initial points in SimpleOrdered. Or we could combine the initial points with a multitask GP such as TransferBO. Another thread of future work is to test the ordered approach on other applications. For instance, we could apply ordered transfer to parameter tuning of environmental models (Druel et al., 2017) or the optimisation of wind turbine operation for subsequent days (Park & Law, 2015).

---

6. Strictly speaking, we sometimes need more than one point per task for SimpleOrdered, as we take steps to avoid using the same past optimum twice. But even storing the five top evaluations per task is very cheap compared to training a machine learning model.

We only considered ordering of HPO tasks coming from increases in the training data. But ordering can come from other sources. We could be gradually increasing the capacity of the model, e.g. the number of layers in a neural network. Or we could be testing different neural network architectural choices more generally: first swapping out the activation function (Goodfellow, Bengio, & Courville, 2016, p. 168), then changing the type of layer, for instance from a fully connected to a convolutional layer (Goodfellow et al., 2016, chaps. 6, 9), and so on. If these changes happen one at a time, we could still expect the most recent task to be the most relevant, even if the transfer is a lot harder in this more general setting.

7

Conclusion

In nature nothing exists alone.

– Rachel Carson, *Silent Spring*, 1962

In this thesis we have explored different transfer learning methods for Bayesian optimisation. We have categorised the methods into direct transfer and prior transfer, and discussed when one would be preferable over the other. We have also developed new transfer learning methods.

In Chapter 4 we introduced a new method for prior transfer, an early version of PLeBO. It uses MCMC and a hierarchical structure to learn GP hyperparameter priors from related data. We then use importance weighting to apply these priors to new optimisation tasks. We also compared several approaches to hyperparameter priors: we started with no priors, and moved through very broad priors and simple priors before developing our sample-based priors.

In Chapter 5 we improved upon the prior transfer method in Chapter 4, calling our improved version PLeBO. We wrote a new implementation using libraries for approximate inference and Gaussian processes, making design changes easier. We also compared PLeBO to a range of non-transfer, direct transfer and prior transfer baselines, and analysed the generated priors. The comparison was done on both synthetic and air pollution data.

In Chapter 6 we introduced SimpleOrdered, a method designed to work well for the subset of direct transfer optimisation tasks that have an ordering between the tasks. We applied our method to hyperparameter optimisation for machine learning, in the setting of gradually accumulating training data. We also compared the method to a series of ablations and to standard HPO transfer learning methods.

The potential benefit of prior transfer is smaller than that of direct transfer. So if direct transfer is working well we should not replace it with prior transfer, instead prior transfer provides a way of introducing transfer learning into new settings. In settings suitable for direct transfer we can introduce prior transfer as an additional level of transfer.

In addition to the methodological work, the thesis also makes contributions towards applying machine learning to new problems. In Chapter 3 we surveyed applications of Bayesian optimisation related to climate change, identifying four key applications and associated benchmarks. This work is intended to facilitate more machine learning specialists beginning work on these applications. And in Chapter 4 we ran a series of experiments applying Bayesian optimisation to real-world air pollution data. This work spanned: i) searching for data sets — including exploring multiple data sets that did not make it into the thesis; ii) adapting the Gaussian process surrogate models to the problem — by developing the combinations of RBF and wind kernels; iii) exploring different ways of learning hyperparameter priors — which we developed into the PLeBO method in Chapter 5; and iv) experimenting with different design choices throughout the Bayesian optimisation framework — as visualised in Fig. 4.14.

7.1 Findings

In Chapter 3 we surveyed work applying Bayesian optimisation to **climate-related problems**, and found that these could be categorised into four main clusters of work: material discovery, wind farm layout, optimal renewable control and environmental monitoring. We also identified suitable benchmarks for each of them.

In our work on **air pollution monitoring** in Chapter 4 we showed that out-of-the-box Bayesian optimisation struggled in this setting. However, we also showed that we were able to beat the baselines using an early version of our prior transfer method PLeBO. Because wind is known to transport pollutants, we derived a directed kernel to model this phenomenon. An important but time-consuming task was identifying air pollution data sets on which to apply Bayesian optimisation. We settled on two: one using satellite data from Copernicus Sentinel-5P (processed by ESA) (2018) and the other using ground-level data from London (Imperial College London, 1993).

Continuing the work on **prior transfer**, in Chapters 2 and 5 we provided intuition and examples of when to prefer prior transfer over direct transfer. Using synthetic data and some of the air pollution benchmarks from Chapter 4, we showed that prior transfer in general, and PLeBO in particular, can work for problems where direct transfer does not. An overview of these results can be seen in Figs. 5.4 and 5.5. We also showed that the learned priors match the tuning data and lead to good hyperparameter estimates on the test data. While the computational complexity of PLeBO is greater than that of the baselines, our method is designed for very expensive applications, which justify the computational burden.

In Chapter 6 we instead consider **ordered transfer**, a variant of direct transfer. We formalised and motivated the setting of ordered transfer: for instance, ordered transfer is applicable when we retune hyperparameters as the size of the training data increases. We introduced a simple method, SimpleOrdered, that can exploit ordering between optimisation tasks. It works by first evaluating the best input from each of the most recent tasks, and then performing non-transfer BO. We showed that for small optimisation budgets it outperforms standard HPO transfer learning.

7.2 Limitations

The major limitation of the survey of climate related applications in Chapter 3 is that Bayesian optimisation is not able to solve this problem on its own. Climate change is much too complex for that. Instead, Bayesian optimisation has promise in improving some of the solutions and mitigations being developed in other research fields. Additionally, we did not collect results comparing different variants of Bayesian optimisation. As we discuss in the chapter, the benchmarks identified have different characteristics, e.g. in terms of the number of samples the surrogate models must handle. Therefore, we expect different BO variants to be suitable for different benchmarks.

We made many simplifications when applying Bayesian optimisation to air pollution monitoring, as well as when discussing climate-related applications. These were done in order to gradually build up an understanding of the problems, and solutions to them. If our methods are unable to solve simplified versions of the optimisation problems, they are unlikely to solve the same problems with the full complexities of the real world. For instance, when preparing the London air pollution data we decided to only use

measurements from one type of site, namely roadside. Even with this simplification the resulting optimisation tasks were challenging. But to be useful in practice, Bayesian optimisation will need to handle multiple types of measurement locations, as not all of London is next to a road. Our work does not provide a ready solution for replacing the traditional way of setting up monitoring networks. Rather, it is the first step towards providing a complementary automated way, which can be used where we do not have the resources required to employ domain experts.

Prior transfer has the advantage of being more generally applicable than direct transfer, but it cannot be a miracle cure. Even in the best case, when the best hyperparameter settings are the same in the tuning tasks and the test task, we still need to explore our optimisation search space. Although we verified that the hyperparameter priors and resulting estimates for PLeBO were reasonable, EI without transfer remained a strong baseline. And while PLeBO was the best realisable method[1] for two of our three benchmarks, the improvement over the prior transfer baselines was small. And in the final benchmark, PLeBO was the worst prior transfer method. Therefore, future work is still required to determine which prior transfer method is preferable in what settings.

In Chapter 6 we introduced the framework of ordered transfer learning, as a subset of direct transfer. It is suitable when we can order the optimisation tasks, and therefore know which of the tuning tasks are most relevant for the test task. We also provided a method, SimpleOrdered, for doing ordered transfer learning. However, this is a simple method, which we expect can be improved upon. For instance, after the initial evaluations we no longer consult any observations of the previous tasks. By combining initial evaluations based on the previous optima with a joint modelling of all the tasks we should be able to speed up the optimisation further.

---

1. On the synthetic benchmark TruePLeBO performs best, but it relies on knowing the true GP hyperparameters, which is not realisable in practice.

7.3 Future work

7.3.1 Baselines for climate benchmarks

Our survey in Chapter 3 did not include collecting baseline results on the benchmarks. A natural next step would be to run some baselines on the benchmarks to compare their performance, which would also set a standard for future work to beat. The benchmarks have different characteristics: some are challenging due to the small size of the evaluation data, others due to the cost of evaluating an input. Additionally, the renewable control benchmark comes with varying context for the different optimisation iterations, which is not part of the standard BO formulation. Because of these different characteristics, we expect different versions of Bayesian optimisation to work best on the different benchmarks.

7.3.2 Temporal monitoring

In Chapter 4 we ignored the temporal aspect of air pollution, as well as the measurement error that would come from using low-cost sensors. To bring the work closer to deployment, we should introduce a third, temporal dimension to the Gaussian process models, and model the measurement error. Accurate modelling of the latter would be difficult, as the properties of these errors are still being investigated. But we could start by introducing Gaussian noise, and make the modelling more realistic once that additional challenge has been overcome.

7.3.3 New PLeBO application

In Chapter 5 we took the prior transfer learning method introduced in Chapter 4 and developed and analysed it further, calling it PLeBO. We compared it to a range of other transfer learning methods on a synthetic data set, and again on air pollution data. To ease analysis and visualisations we used two-dimensional inputs and only learned priors for two hyperparameters. But as we argue in Section 5.8.2, informative hyperparameter priors are more important for more expressive models with more hyperparameters. Therefore, an informative next step would be to apply PLeBO to a new application, using more expressive kernels. For instance, we could use it for optimising product prices. In Chapter 6 we developed a version of the product pricing problem consisting of an ordered sequence of optimisation tasks, coming from gradual

changes in the product demand. To test PLeBO we could develop a set of optimisation tasks corresponding to different product groups. They would have a greater range of optimal prices than those used in Chapter 6, so we would not expect direct transfer learning to work well.

7.3.4 Ordered air pollution monitoring

In Chapter 4 we used pollution data from London in our evaluations. Our motivation was to develop a prior transfer learning method for air pollution so we could learn GP hyperparameter priors on pollution data from cities with dense monitoring networks, and use these priors to plan a monitoring network in an unmonitored city. Therefore, we pretended that different days in London corresponded to different cities. We used data from one year to learn the hyperparameter distributions, treating the measurements from each day as separate optimisation tasks. And we used these priors to speed up the optimisation of the test tasks, again treating each day as a separate optimisation task.

This recasting of the data made sense in Chapter 4, since we wanted to evaluate the ability to plan a monitoring network in a new city. But we could also use the same data to evaluate ordered transfer learning, as introduced in Chapter 6. We would do so by treating the sequence of optimisation tasks in the test set as an ordered sequence. So when finding the pollution maximum of 5 January we would use that of 1 to 4 January, and so on. Applying ordered transfer learning, we are assuming that the optima are directly related, i.e. that there are sensors which are over-represented among the maxima throughout the year. This is a reasonable assumption, as urban pollution sources such as major roads do not often move around.

The evaluation could be done using the Syne Tune library, as was done in Chapter 6. Then we can compare the same methods as used in Chapter 6, but on the new benchmark using London air pollution data. There are two advantages of this proposed work. Firstly, it provides an additional benchmark for comparing the performance of SimpleOrdered to standard direct transfer methods. Secondly, it would be an initial investigation into whether Bayesian optimisation could be used to set up dynamic monitoring networks where the sensors are moved around to track the pollution maximum.

7.4 Direct, prior and ordered transfer

The results in Chapter 5 showed the difficulty of prior transfer. The best-case scenario for direct — and therefore also ordered — transfer is that the optima are in the same location, such that we can solve the new optimisation task in one evaluation. But for prior transfer the best-case scenario is that the ideal Gaussian process hyperparameters are the same. This simplifies the optimisation task, but the search space still needs to be explored. We can formulate a preference list for the three types of transfer learning, shown in Table 7.1. If the set of optimisation tasks fulfils the requirements for all types of transfer, we prefer ordered transfer. If we do not have an ordering for the tasks, we revert to direct transfer. And if we do not have co-located optima, we resort to prior transfer.

Preference	Transfer type	Requirements	Introduced method
1	Ordered	– co-located optima – task ordering	SimpleOrdered
2	Direct	– co-located optima	—
3	Prior	– similar optimisation landscapes	PLeBO

Table 7.1: Summary of the types of transfer learning methods discussed in the thesis. If suitable, we prefer ordered, followed by direct and finally prior transfer.

All three types of transfer learning discussed are topics for further research. Ordered and prior transfer are recent topics, where many new methods can be developed and applied to new applications. SimpleOrdered was developed as a simple example of ordered transfer. And PLeBO only beat the prior transfer baselines on two of the three benchmarks. Direct transfer is a more mature research field, with many more published methods. But it is still an area of active research, with new methods being developed. And most of these methods have focused on the application of hyperparameter optimisation for machine learning models. Other applications might require new methods, or adaptions to existing methods, e.g. to determine good sets of metadata.

Bibliography

Abdelrahman, H., Berkenkamp, F., Poland, J., & Krause, A. (2016). Bayesian optimization for maximum power point tracking in photovoltaic power plants. In *2016 European control conference (ECC)* (pp. 2078–2083).

Abel, D. (2019). simple_rl: Reproducible reinforcement learning in python. *ICLR Workshop on Reproducibility in Machine Learning*.

Abel, D., Williams, E. C., Brawner, S., Reif, E., & Littman, M. L. (2018). Bandit-based solar panel control. In *Thirty-second AAAI conference on artificial intelligence.*

Ainslie, B., Reuten, C., Steyn, D. G., Le, N. D., & Zidek, J. V. (2009). Application of an entropy-based Bayesian optimization technique to the redesign of an existing monitoring network for single air pollutants. *Journal of Environmental Management*, *90*(8), 2715–2729.

Air Quality in Scotland. (2021). *Site data, Edinburgh St Leonards.* Retrieved 2023-10-18, from `https://www.scottishairquality.scot/latest/site-info/ED3`

Ament, S. E., & Gomes, C. P. (2022). Scalable first-order Bayesian optimization via structured automatic differentiation. In *International conference on machine learning* (pp. 500–516).

Andersson, L. E., & Imsland, L. (2020). Real-time optimization of wind farms using modifier adaptation and machine learning. *Wind Energy Science*, *5*(3), 885–896.

Annevelink, E., Kurchin, R., Muckley, E., Kavalsky, L., Hegde, V. I., Sulzer, V., ... Viswanathan, V. (2022). AutoMat: Automated materials discovery for electrochemical systems. *MRS Bulletin*, 1–9.

Arango, S. P., Jomaa, H. S., Wistuba, M., & Grabocka, J. (2021). HPO-B: A large-scale reproducible benchmark for black-box HPO based on OpenML. *arXiv preprint arXiv:2106.06257*.

Asenov, M., Rutkauskas, M., Reid, D., Subr, K., & Ramamoorthy, S. (2019). Active localization of gas leaks using fluid simulation. *IEEE Robotics and Automation Letters*, *4*(2), 1776–1783. doi: 10.1109/LRA.2019.2895820

Atkinson, R. W., Butland, B. K., Anderson, H. R., & Maynard, R. L. (2018). Long-term concentrations of nitrogen dioxide and mortality: a meta-analysis of cohort studies. *Epidemiology (Cambridge, Mass.)*, *29*(4), 460.

Attia, P. M., Grover, A., Jin, N., Severson, K. A., Markov, T. M., Liao, Y.-H., ... Chueh, W. C. (2020). Closed-loop optimization of fast-charging protocols for batteries with machine learning. *Nature*, *578*(7795), 397–402.

Austen, J. (2003). *Pride and prejudice*. Collector's Library. (Originally published in 1813. Citation from Chapter 27.)

Autograd. (since 2015). *Autograd.* `https://github.com/HIPS/autograd`.

AV, A. K., Rana, S., Shilton, A., & Venkatesh, S. (2022). Human-AI collaborative Bayesian optimisation. *Advances in Neural Information Processing Systems*, *35*, 16233–16245.

Bai, T., Li, Y., Shen, Y., Zhang, X., Zhang, W., & Cui, B. (2023). Transfer learning for Bayesian optimization: A survey. *arXiv preprint arXiv:2302.05927*.

Balakrishnan, K., Dey, S., Gupta, T., Dhaliwal, R. S., Brauer, M., Cohen, A. J., ... Dandona, L. (2019). The impact of air pollution on deaths, disease burden, and life expectancy across the states of India: the Global Burden of Disease Study 2017. *The Lancet Planetary Health*, *3*(1), e26–e39. (Publisher: Elsevier)

Balandat, M., Karrer, B., Jiang, D. R., Daulton, S., Letham, B., Wilson, A. G., & Bakshy, E. (2020). BoTorch: A framework for efficient Monte-Carlo Bayesian optimization. In *Advances in neural information processing systems 33*.

Bash, D., Cai, Y., Chellappan, V., Wong, S. L., Yang, X., Kumar, P., ... Hippalgaonkar, K. (2021). Multi-fidelity high-throughput optimization of electrical conductivity in P3HT-CNT composites. *Advanced Functional Materials*, 2102606.

Benjamins, C., Raponi, E., Jankovic, A., Doerr, C., & Lindauer, M. (2023). Self-adjusting weighted expected improvement for bayesian optimization. In *AutoML conference 2023*.

Berk, J., Nguyen, V., Gupta, S., Rana, S., & Venkatesh, S. (2018). Exploration enhanced expected improvement for Bayesian optimization. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 621–637). Springer International Publishing. doi: 10.1007/978-3-030-10928-8_37

Binder, M., Pfisterer, F., & Bischl, B. (2020). Collecting empirical data about hyperparameters for data driven AutoML. *Democratizing Machine Learning Contributions in AutoML and Fairness*, 93.

Bliek, L. (2022). A survey on sustainable surrogate-based optimisation. *Sustainability*, *14*(7), 3867.

Bliek, L., Guijt, A., Karlsson, R., Verwer, S., & de Weerdt, M. (2023). Benchmarking surrogate-based optimisation algorithms on expensive black-box functions. *Applied Soft Computing*, *147*, 110744.

Bohdal, O., Balles, L., Wistuba, M., Ermis, B., Archambeau, C., & Zappella, G. (2023). PASHA: Efficient HPO and NAS with progressive resource allocation. In *Iclr.*

Bouneffouf, D., Rish, I., & Aggarwal, C. (2020). Survey on applications of multi-armed and contextual bandits. In *2020 IEEE congress on evolutionary computation (CEC)* (pp. 1–8).

Britannica, E. (1998). *Bushel.* Encyclopædia Britannica, inc. Retrieved 2023-07-13, from `https://www.britannica.com/science/bushel`

Carminati, M., Ferrari, G., & Sampietro, M. (2017). Emerging miniaturized technologies for airborne particulate matter pervasive monitoring. *Measurement*, *101*, 250–256.

Carson, R. (2020). *Silent spring*. Penguin. (Originally published in 1962. Citation from Chapter 4.)

Castell, N., Dauge, F. R., Schneider, P., Vogt, M., Lerner, U., Fishbain, B., ... Bartonova, A. (2017). Can commercial low-cost sensor platforms contribute to air quality monitoring and exposure estimates? *Environment International*, *99*, 293–302.

Cats, G., & Holtslag, A. (1980). Prediction of air pollution frequency distribution—Part I. The lognormal model. *Atmospheric Environment (1967)*, *14*(2), 255–258.

Chakrabarty, A., Danielson, C., Bortoff, S. A., & Laughman, C. R. (2021). Accelerating self-optimization control of refrigerant cycles with Bayesian optimization and adaptive moment estimation. *Applied Thermal Engineering*, *197*, 117335.

Chakrabarty, A., Maddalena, E., Qiao, H., & Laughman, C. (2021). Scalable Bayesian optimization for model calibration: Case study on coupled building and HVAC dynamics. *Energy and Buildings*, *253*, 111460.

Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P. K., Torr, P. H., & Ranzato, M. (2019). On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*.

Chen, K., Lin, J., Qiu, Y., Liu, F., & Song, Y. (2022). Joint optimization of wind farm layout considering optimal control. *Renewable Energy*, *182*, 787–796.

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 785–794). New York, NY, USA: ACM.

Cheng, Y., Li, X., Li, Z., Jiang, S., & Jiang, X. (2014). Fine-grained air quality monitoring based on Gaussian process regression. In C. K. Loo, K. S. Yap, K. W. Wong, A. Teoh, & K. Huang (Eds.), *Neural Information Processing* (Vol. 8835, pp. 126–134). Cham: Springer International Publishing. (Series Title: Lecture Notes in Computer Science)

Chib, S., & Greenberg, E. (1995). Understanding the Metropolis - Hastings algorithm. *The American Statistician*, *49*(4), 327–335.

Chugh, T., & Ymeraj, E. (2022). Wind farm layout optimisation using set based multi-objective Bayesian optimisation. In *Genetic and evolutionary computation conference companion (GECCO '22 companion)* (pp. 695–698). ACM, New York, NY, USA.

Cohen, A. J., Brauer, M., Burnett, R., Anderson, H. R., Frostad, J., Estep, K., ... Forouzanfar, M. H. (2017). Estimates and 25-year trends of the global burden of disease attributable to ambient air pollution: an analysis of data from the Global Burden of Diseases Study 2015. *The Lancet*, *389*(10082), 1907–1918.

Copernicus Sentinel-5P (processed by ESA). (2018). *TROPOMI Level 2 Nitrogen Dioxide total column products. Version 01.* European Space Agency. Retrieved 2022-06-06, from `https://sentinels.copernicus.eu/web/sentinel/data-products/-/asset_publisher/fp37fc19FN8F/content/sentinel-5-precursor-level-2-nitrogen-dioxide` doi: 10.5270/S5P-s4ljg54

Corani, G., Benavoli, A., & Zaffalon, M. (2021). Time series forecasting with Gaussian processes needs priors. In *Joint European conference on machine learning and knowledge discovery in databases* (pp. 103–117).

Cowen-Rivers, A. I., Lyu, W., Tutunov, R., Wang, Z., Grosnit, A., Griffiths, R. R., ... Bou-Ammar, H. (2022). HEBO: Pushing the limits of sample-efficient hyper-parameter optimisation. *Journal of Artificial Intelligence Research*, *74*, 1269–1349.

De Ath, G., Everson, R. M., & Fieldsend, J. E. (2021). How Bayesian should Bayesian optimisation be? In *Proceedings of the genetic and evolutionary computation conference companion* (pp. 1860–1869).

De Ath, G., Everson, R. M., Rahat, A. A., & Fieldsend, J. E. (2021). Greed is good: Exploration and exploitation trade-offs in Bayesian optimisation. *ACM Transactions on Evolutionary Learning and Optimization*, *1*(1), 1–22.

De Ath, G., Fieldsend, J. E., & Everson, R. M. (2020). What do you mean? The role of the mean function in bayesian optimisation. In *Proceedings of the 2020 genetic and evolutionary computation conference companion* (pp. 1623–1631).

Doekemeijer, B. M., Van Der Hoek, D. C., & van Wingerden, J.-W. (2019). Model-based closed-loop wind farm control for power maximization using Bayesian optimization: a large eddy simulation study. In *2019 IEEE conference on control technology and applications (CCTA)* (pp. 284–289).

Dong, H., Xie, J., & Zhao, X. (2022). Wind farm control technologies: From classical control to reinforcement learning. *Progress in Energy*.

Donti, P. L., & Kolter, J. Z. (2021). Machine learning for sustainable energy systems. *Annual Review of Environment and Resources*, *46*, 719–747.

Druel, A., Peylin, P., Krinner, G., Ciais, P., Viovy, N., Peregon, A., ... Mironycheva-Tokareva, N. (2017). Towards a more detailed representation of high-latitude vegetation in the global land surface model ORCHIDEE (ORC-HL-VEGv1. 0). *Geoscientific Model Development*, *10*(12), 4693–4722.

Dua, D., & Graff, C. (2017). *UCI machine learning repository.* Retrieved from `http://archive.ics.uci.edu/ml`

Duque, V., & Gilraine, M. (2022). Coal use, air pollution, and student performance. *Journal of Public Economics*, *213*, 104712.

Eckman, D. J., Henderson, S. G., & Shashaani, S. (2023). Simopt: A testbed for simulation-optimization experiments. *INFORMS Journal on Computing*.

Fan, Z., Han, X., & Wang, Z. (2022). HyperBO+: Pre-training a universal prior for Bayesian optimization with hierarchical Gaussian processes. *arXiv preprint arXiv:2212.10538*.

Feurer, M., & Hutter, F. (2019). Hyperparameter optimization. In F. Hutter, L. Kotthoff, & J. Vanschoren (Eds.), *Automated machine learning: Methods, systems, challenges* (pp. 3–33). Cham: Springer International Publishing.

Feurer, M., Letham, B., & Bakshy, E. (2018). Scalable meta-learning for Bayesian optimization using ranking-weighted gaussian process ensembles. In *AutoML workshop at ICML* (Vol. 7).

Feurer, M., Letham, B., Hutter, F., & Bakshy, E. (2022). Practical transfer learning for Bayesian optimization. *arXiv preprint arXiv:2304.11005*.

Feurer, M., Springenberg, J. T., & Hutter, F. (2015). Initializing Bayesian hyperparameter optimization via meta-learning. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 29).

Fiducioso, M., Curi, S., Schumacher, B., Gwerder, M., & Krause, A. (2019). Safe contextual Bayesian optimization for sustainable room temperature PID control tuning. In *Proceedings of the twenty-eighth international joint conference on artificial intelligence, IJCAI-19* (pp. 5850–5856). International Joint Conferences on Artificial Intelligence Organization.

Finch, D. P., Palmer, P. I., & Zhang, T. (2022). Automated detection of atmospheric NO_2 plumes from satellite data: a tool to help infer anthropogenic combustion emissions. *Atmospheric Measurement Techniques*, *15*(3), 721–733. (Publisher: Copernicus GmbH)

Flecker, A. S., Shi, Q., Almeida, R. M., Angarita, H., Gomes-Selman, J. M., García-Villacorta, R., ... Gomes, C. P. (2022). Reducing adverse impacts of Amazon hydropower expansion. *Science*, *375*(6582), 753–760.

Folch, J. P., Zhang, S., Lee, R., Shafei, B., Walz, D., Tsay, C., ... Misener, R. (2022). SnAKe: Bayesian optimization with pathwise exploration. *Advances in Neural Information Processing Systems*, *35*, 35226–35239.

Frazier, P. I., & Wang, J. (2016). Bayesian optimization for materials design. In *Information science for materials discovery and design* (pp. 45–75). Springer.

Frey, D., Neyerlin, K., & Modestino, M. A. (2022). Bayesian optimization of electrochemical devices for electrons-to-molecules conversions: The case of pulsed CO2 electroreduction. *ChemRxiv*. doi: 10.26434/chemrxiv-2022-n09hc

Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, *33*(1), 1–22.

Gao, T., & Bai, X. (2022). Bayesian optimization-based three-dimensional, time-varying environment monitoring using an UAV. *Journal of Intelligent & Robotic Systems*, *105*(4), 1–21.

Garnett, R. (2023). *Bayesian optimization*. Cambridge University Press.

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine learning*, *63*, 3–42.

Golovin, D., Solnik, B., Moitra, S., Kochanski, G., Karro, J., & Sculley, D. (2017). Google Vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1487–1495).

González, J., Dai, Z., Hennig, P., & Lawrence, N. (2016). Batch Bayesian optimization via local penalization. In *Artificial intelligence and statistics* (pp. 648–657).

González, J., Osborne, M., & Lawrence, N. (2016). GLASSES: Relieving the myopia of Bayesian optimisation. In *Artificial intelligence and statistics* (pp. 790–799).

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Gordon, S. B., Bruce, N. G., Grigg, J., Hibberd, P. L., Kurmi, O. P., Lam, K.-b. H., ... Martin, W., II (2014). Respiratory risks from household air pollution in low and middle income countries. *The Lancet Respiratory medicine*, *2*(10), 823–860.

GPy. (since 2012). *GPy: A gaussian process framework in python*. `http://github.com/SheffieldML/GPy`.

Grašič, B., Mlakar, P., & Božnar, M. (2006). Ozone prediction based on neural networks and Gaussian processes. *Il Nuovo Cimento C*, *29*(6), 651–661.

Grena, R. (2012). Five new algorithms for the computation of sun position from 2010 to 2110. *Solar Energy*, *86*(5), 1323–1337.

Haghanifar, S., Cheng, B., McCourt, M., & Leu, P. (2021). Using Bayesian optimization to improve solar panel performance by developing antireflective, superomniphobic glass. In *Icml 2019 workshop: Climate change: How can AI help?*

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362.

Hellan, S. P., Lucas, C. G., & Goddard, N. H. (2020). Optimising placement of pollution sensors in windy environments. *NeurIPS 2020 Workshop on AI for Earth Sciences*.

Hellan, S. P., Lucas, C. G., & Goddard, N. H. (2022). Bayesian optimisation for active monitoring of air pollution. *Proceedings of the AAAI Conference on Artificial Intelligence*, *36*(11), 11908-11916.

Hellan, S. P., Lucas, C. G., & Goddard, N. H. (2023a). Bayesian optimisation against climate change: Applications and benchmarks. *Data-centric Machine Learning Research (DMLR) Workshop at ICML 2023*.

Hellan, S. P., Lucas, C. G., & Goddard, N. H. (2023b). Data-driven prior learning for Bayesian optimisation. *NeurIPS 2023 Workshop on Adaptive Experimental Design and Active Learning in the Real World*.

Hellan, S. P., Shen, H., Aubet, F.-X., Salinas, D., & Klein, A. (2023). Obeying the order: Introducing ordered transfer hyperparameter optimisation. *AutoML 2023 Workshop track*.

Hennig, P., & Schuler, C. J. (2012). Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, *13*(6).

Hernández-Lobato, J. M., Hoffman, M. W., & Ghahramani, Z. (2014). Predictive entropy search for efficient global optimization of black-box functions. *Advances in Neural Information Processing Systems 27*, 918–926.

Hoffman, M. D., & Gelman, A. (2014). The No-U-Turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, *15*(1), 1593–1623.

Horváth, S., Klein, A., Richtárik, P., & Archambeau, C. (2021). Hyperparameter transfer learning with adaptive complexity. In *International conference on artificial intelligence and statistics* (pp. 1378–1386).

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. doi: 10.1109/MCSE.2007.55

Hvarfner, C., Hellsten, E., Hutter, F., & Nardi, L. (2023). Self-correcting Bayesian optimization through Bayesian active learning. In *Advances in neural information processing systems* (Vol. 36).

Hvarfner, C., Hutter, F., & Nardi, L. (2022). Joint entropy search for maximally-informed Bayesian optimization. In *Advances in neural information processing systems* (Vol. 35).

Imperial College London, E. R. G. (1993). *London Air Quality Network.* Retrieved 2021-08-31, from `https://www.londonair.org.uk/LondonAir/Default.aspx`

Infeld, L. (2006). *Quest : an autobiography* (2nd ed.). Providence, R.I.: AMS Chelsea Pub. Co. (Citation from p. 279.)

Jamieson, K., & Talwalkar, A. (2016). Non-stochastic best arm identification and hyperparameter optimization. In *Artificial intelligence and statistics* (pp. 240–248).

Jansson, T. (2000). *Finn family moomintroll*. Puffin Books. (Originally published in 1948 in Swedish. Translated by Elizabeth Portch. Citation from Chapter 7.)

Jia, Y., Batra, N., Wang, H., & Whitehouse, K. (2019). Active collaborative sensing for energy breakdown. In *Proceedings of the 28th ACM international conference on information and knowledge management* (pp. 1943–1952).

Johnston, S. J., Basford, P. J., Bulot, F. M. J., Apetroaie-Cristea, M., Easton, N. H. C., Davenport, C., ... Cox, S. J. (2019). City scale particulate matter monitoring using LoRaWAN based air quality IoT devices. *Sensors*, *19*(1), 209.

Jomaa, H. S., Arango, S. P., Schmidt-Thieme, L., & Grabocka, J. (2021). Transfer learning for Bayesian HPO with end-to-end landmark meta-features. In *Fifth workshop on meta-learning at the conference on neural information processing systems.*

Jones, D. R., Schonlau, M., & Welch, W. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, *13*(4), 455–492.

Katsouyanni, K. (2003). Ambient air pollution and health. *British Medical Bulletin*, *68*(1), 143–156.

Kelly, K. E., Whitaker, J., Petty, A., Widmer, C., Dybwad, A., Sleeth, D., ... Butterfield, A. (2017). Ambient and laboratory evaluation of a low-cost particulate matter sensor. *Environmental Pollution*, *221*, 491–500.

Klein, A., Falkner, S., Bartels, S., Hennig, P., & Hutter, F. (2017). Fast Bayesian optimization of machine learning hyperparameters on large datasets. In *Artificial intelligence and statistics* (pp. 528–536).

Klein, A., Tiao, L. C., Lienart, T., Archambeau, C., & Seeger, M. (2020). Model-based asynchronous hyperparameter and neural architecture search. *arXiv preprint arXiv:2003.10865*.

Kleinegesse, S., & Gutmann, M. U. (2020). Bayesian experimental design for implicit models by mutual information neural estimation. In *International conference on machine learning* (pp. 5316–5326).

Kudo, M., Toyama, J., & Shimbo, M. (1999). Multidimensional curve classification using passing-through regions. *Pattern Recognition Letters*, *20*(11-13), 1103–1111.

Kushmerick, N. (1999). Learning to remove internet advertisements. In *Proceedings of the third annual conference on autonomous agents* (pp. 175–181).

Kushner, H. J. (1964). A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, *86*(1), 97–106.

Lalchand, V., & Rasmussen, C. E. (2020). Approximate inference for fully Bayesian Gaussian process regression. *Symposium on Advances in Approximate Bayesian Inference*, 1–12. (PMLR)

Landrigan, P. J., Fuller, R., Acosta, N. J., Adeyi, O., Arnold, R., Baldé, A. B., ... Zhong, M. (2018). The Lancet Commission on pollution and health. *The Lancet*, *391*(10119), 462–512.

Lange, C., Cole, E., Van Horn, G., & Mac Aodha, O. (2023). Active learning-based species range estimation. *Advances in Neural Information Processing Systems*, *36*.

Lawrence, N. D., & Platt, J. C. (2004). Learning to learn with the informative vector machine. In *Proceedings of the twenty-first international conference on machine learning.*

Lelieveld, J., Evans, J. S., Fnais, M., Giannadaki, D., & Pozzer, A. (2015). The contribution of outdoor air pollution sources to premature mortality on a global scale. *Nature*, *525*(7569), 367–371. (Number: 7569 Publisher: Nature Publishing Group)

Leu, F.-Y., & Ho, J.-S. (2020). Air pollution source identification by using neural network with Bayesian optimization. In *Proceedings of the 13th international conference on innovative mobile and internet services in ubiquitous computing (IMIS-2019)* (pp. 514–524).

Li, J. J. (2014). Estimating urban ultrafine particle distributions with Gaussian process models. *Research@Locate'14*, 9.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2017). Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, *18*(1), 6765–6816.

Liang, Q., Gongora, A. E., Ren, Z., Tiihonen, A., Liu, Z., Sun, S., ... Buonassisi, T. (2021). Benchmarking the performance of Bayesian optimization across multiple experimental materials science domains. *npj Computational Materials*, *7*(1), 1–10.

Lin, C., Gillespie, J., Schuder, M. D., Duberstein, W., Beverland, I. J., & Heal, M. R. (2015). Evaluation and calibration of Aeroqual series 500 portable gas sensors for accurate measurement of ambient ozone and nitrogen dioxide. *Atmospheric Environment*, *100*, 111–116.

Liu, H., Ong, Y.-S., Shen, X., & Cai, J. (2020). When Gaussian process meets big data: A review of scalable GPs. *IEEE transactions on neural networks and learning systems*, *31*(11), 4405–4423.

Liu, H., Yang, C., Huang, M., Wang, D., & Yoo, C. (2018). Modeling of subway indoor air quality using Gaussian process regression. *Journal of Hazardous Materials*, *359*, 266–273.

Liu, X., Jayaratne, R., Thai, P., Kuhn, T., Zing, I., Christensen, B., ... Morawska, L. (2020). Low-cost sensors as an alternative for long-term air quality monitoring. *Environmental Research*, *185*, 109438.

Liu, X., Xi, T., & Ngai, E. (2016). Data modelling with Gaussian process in sensor networks for urban environmental monitoring. In *2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)* (pp. 457–462). (ISSN: 2375-0227) doi: 10.1109/MASCOTS.2016.45

Lyden, S., Olding, W., & Darbari, Z. (2018). Bayesian optimization for maximum power point tracking in photovoltaic systems. In *2018 IEEE power & energy society general meeting (PESGM)* (pp. 1–5).

Madeo, R. C., Lima, C. A., & Peres, S. M. (2013). Gesture unit segmentation using support vector machines: segmenting gestures from rest positions. In *Proceedings of the 28th annual ACM symposium on applied computing* (pp. 46–52).

Makarova, A., Shen, H., Perrone, V., Klein, A., Faddoul, J. B., Krause, A., ... Archambeau, C. (2022). Automatic termination for hyperparameter optimization. In *International conference on automated machine learning* (pp. 7/1–21).

Malkov, Y. A., & Yashunin, D. A. (2018). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, *42*(4), 824–836.

Marchant, R., & Ramos, F. (2012). Bayesian optimisation for Intelligent Environmental Monitoring. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 2242–2249). IEEE. doi: 10.1109/IROS.2012.6385653

Marchant, R., & Ramos, F. (2014). Bayesian Optimisation for informative continuous path planning. In *2014 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 6136–6143). (ISSN: 1050-4729) doi: 10.1109/ICRA.2014.6907763

Marini, F., & Walczak, B. (2015). Particle swarm optimization (PSO). A tutorial. *Chemometrics and Intelligent Laboratory Systems*, *149*, 153–165.

Martin, R. V., Brauer, M., van Donkelaar, A., Shaddick, G., Narain, U., & Dey, S. (2019). No one knows which city has the highest concentration of fine particulate matter. *Atmospheric Environment: X*, *3*, 100040.

Mekki-Berrada, F., Ren, Z., Huang, T., Wong, W. K., Zheng, F., Xie, J., ... Wang, X. (2021). Two-step machine learning enables optimized nanoparticle synthesis. *npj Computational Materials*, *7*(1), 1–10.

Mern, J., Yildiz, A., Sunberg, Z., Mukerji, T., & Kochenderfer, M. J. (2021). Bayesian optimized Monte Carlo planning. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 35, pp. 11880–11887).

Morere, P., Marchant, R., & Ramos, F. (2017). Sequential Bayesian optimization as a POMDP for environment monitoring with UAVs. In *2017 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 6381–6388). IEEE. doi: 10.1109/ICRA.2017.7989754

Moustakis, N., Mulders, S. P., Kober, J., & van Wingerden, J.-W. (2019). A practical Bayesian optimization approach for the optimal estimation of the rotor effective wind speed. In *2019 American control conference (ACC)* (pp. 4179–4185).

Mudway, I. S., Dundas, I., Wood, H. E., Marlin, N., Jamaludin, J. B., Bremner, S. A., ... Griffiths, C. J. (2019). Impact of London's low emission zone on air quality and children's respiratory health: a sequential annual cross-sectional study. *The Lancet Public Health*, *4*(1), e28–e40.

Mukherjee, A., & Agrawal, M. (2017). World air particulate matter: sources, distribution and health effects. *Environmental chemistry letters*, *15*, 283–309.

Mulders, S., Pamososuryo, A., & Van Wingerden, J. (2020). Efficient tuning of individual pitch control: a Bayesian optimization machine learning approach. In *Journal of physics: Conference series* (Vol. 1618).

Müller, S., Feurer, M., Hollmann, N., & Hutter, F. (2023). PFNs4BO: In-context learning for Bayesian optimization. In *International conference on machine learning*.

Murphy, K. P. (2022). *Probabilistic machine learning: An introduction*. MIT Press. Retrieved from `https://probml.github.io/book1` (Draft dated June 22, 2023)

Murphy, K. P. (2023). *Probabilistic machine learning: Advanced topics*. MIT Press. Retrieved from `http://probml.github.io/book2` (Draft dated April 1, 2023)

Murray, I., & Adams, R. P. (2010). Slice sampling covariance hyperparameters of latent Gaussian models. In *Advances in Neural Information Processing Systems* (Vol. 23).

Nguyen, Q., Wu, K., Gardner, J., & Garnett, R. (2022). Local Bayesian optimization via maximizing probability of descent. *Advances in neural information processing systems*, *35*, 13190–13202.

NREL. (2020). *FLORIS. version 2.1.1.* GitHub. Retrieved from `https://github.com/NREL/floris`

Oliveira, R., Tiao, L., & Ramos, F. T. (2022). Batch Bayesian optimisation via density-ratio estimation with guarantees. *Advances in Neural Information Processing Systems*, *35*, 29816–29829.

open-seneca team. (2018). *open-seneca website.* Retrieved 2022-05-31, from `https://www.open-seneca.org/` (University of Cambridge)

Organizers of KDD Cup 2012 and Tencent Inc. (2012). *2012 KDD Cup data set.*

Ortiz-Montalvo, D. L., Kusne, A. G., McDannald, A., Siderius, D., Choudhary, K., & Yildirim, T. (2021). Machine learning for climate change: Guiding discovery of sorbent materials for direct air capture of CO2. In *ICML 2021 workshop on tackling climate change with machine learning*.

Papenmeier, L., Nardi, L., & Poloczek, M. (2022). Increasing the scope as you learn: Adaptive Bayesian optimization in nested subspaces. In *Advances in neural information processing systems.*

Park, J. (2020). Contextual Bayesian optimization with trust region (CBOTR) and its application to cooperative wind farm control in region 2. *Sustainable Energy Technologies and Assessments*, *38*, 100679.

Park, J., Ferguson, M., & Law, K. H. (2018). Data driven analytics (machine learning) for system characterization, diagnostics and control optimization. In *Workshop of the European group for intelligent computing in engineering* (pp. 16–36).

Park, J., Kwon, S.-D., & Law, K. (2017). A data-driven, cooperative approach for wind farm control: a wind tunnel experimentation. *Energies*, *10*(7), 852.

Park, J., Kwon, S.-D., & Law, K. H. (2016). A data-driven approach for cooperative wind farm control. In *2016 American control conference (ACC)* (pp. 525–530).

Park, J., & Law, K. H. (2015). A Bayesian optimization approach for wind farm power maximization. In *Smart sensor phenomena, technology, networks, and systems integration 2015* (Vol. 9436, pp. 46–57).

Park, J., & Law, K. H. (2016a). Bayesian ascent: A data-driven optimization scheme for real-time control with application to wind farm power maximization. *IEEE Transactions on Control Systems Technology*, *24*(5), 1655–1668.

Park, J., & Law, K. H. (2016b). A data-driven, cooperative wind farm control to maximize the total power production. *Applied Energy*, *165*, 151–165.

Passano, L. M., III. (1952). *The x-organ sinus gland complex of brachyuran crustaceans, a neurosecretory molt controlling gland* (Unpublished doctoral dissertation). Yale University, New Haven CT. (Citation from Table 1, following p. 34.)

Patacchiola, M., Turner, J., Crowley, E. J., O'Boyle, M., & Storkey, A. J. (2020). Bayesian meta-learning for the few-shot setting via deep kernels. *Advances in Neural Information Processing Systems*, *33*, 16108–16118.

Pedersen, M. M., van der Laan, P., Friis-Møller, M., Rinker, J., & Réthoré, P.-E. (2019). Dtuwindenergy/pywake: Pywake. *Zenodo [code]*, *10*. Retrieved from `https://zenodo.org/records/2562662`

Perrone, V., Jenatton, R., Seeger, M. W., & Archambeau, C. (2018). Scalable hyperparameter transfer learning. *Advances in neural information processing systems*, *31*.

Perrone, V., Shen, H., Seeger, M. W., Archambeau, C., & Jenatton, R. (2019). Learning search spaces for Bayesian optimization: Another view of hyperparameter transfer learning. *Advances in Neural Information Processing Systems*, *32*.

Pfisterer, F., Schneider, L., Moosbauer, J., Binder, M., & Bischl, B. (2022). YAHPO Gym – an efficient multi-objective multi-fidelity benchmark for hyperparameter optimization. In *International conference on automated machine learning* (pp. 3/1–39).

Phan, D., Pradhan, N., & Jankowiak, M. (2019). Composable effects for flexible and accelerated probabilistic programming in NumPyro. *arXiv preprint arXiv:1912.11554*.

Pirjola, L., Lähde, T., Niemi, J., Kousa, A., Rönkkö, T., Karjalainen, P., ... Hillamo, R. (2012). Spatial and temporal characterization of traffic emissions in urban microenvironments with a mobile laboratory. *Atmospheric Environment*, *63*, 156–167.

Plume Labs. (2023). *Plume labs website.* Retrieved 2023-10-09, from `https://plumelabs.com/en/`

Pradhan, B., Sameen, M. I., Al-Najjar, H. A., Sheng, D., Alamri, A. M., & Park, H.-J. (2021). A meta-learning approach of optimisation for spatial prediction of landslides. *Remote Sensing*, *13*(22), 4521.

Probst, P., Wright, M. N., & Boulesteix, A.-L. (2019). Hyperparameters and tuning strategies for random forest. *Wiley Interdisciplinary Reviews: data mining and knowledge discovery*, *9*(3), e1301.

PurpleAir. (2020). *PurpleAir | real time air quality monitoring.* Retrieved 2020-02-10, from `https://www2.purpleair.com/`

Quinlan, J. R. (1987). Simplifying decision trees. *International journal of man-machine studies*, *27*(3), 221–234.

Rainforth, T., Foster, A., Ivanova, D. R., & Smith, F. B. (2023). Modern Bayesian experimental design. *arXiv preprint arXiv:2302.14545*.

Ramesh, S. S., Sessa, P. G., Krause, A., & Bogunovic, I. (2022). Movement penalized Bayesian optimization with application to wind energy systems. *Advances in Neural Information Processing Systems*, *35*, 27036–27048.

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. MIT Press.

Reggente, M., & Lilienthal, A. J. (2009). Using local wind information for gas distribution mapping in outdoor environments with a mobile robot. In *2009 IEEE SENSORS* (pp. 1715–1720). doi: 10.1109/ICSENS.2009.5398498

Reggente, M., Peters, J., Theunis, J., Van Poppel, M., Rademaker, M., Kumar, P., & De Baets, B. (2014). Prediction of ultrafine particle number concentrations in urban environments by means of Gaussian process regression based on measurements of oxides of nitrogen. *Environmental Modelling & Software*, *61*, 135–150.

Rolnick, D., Donti, P. L., Kaack, L. H., Kochanski, K., Lacoste, A., Sankaran, K., ... Bengio, Y. (2022). Tackling climate change with machine learning. *ACM Computing Surveys (CSUR)*, *55*(2), 1–96.

Salinas, D., Seeger, M., Klein, A., Perrone, V., Wistuba, M., & Archambeau, C. (2022). Syne Tune: A library for large scale hyperparameter tuning and reproducible research. In *First conference on automated machine learning (main track).*

Salinas, D., Shen, H., & Perrone, V. (2020). A quantile-based approach for hyperparameter transfer learning. In *International conference on machine learning* (pp. 8438–8448).

Samaniego, F. P., Reina, D. G., Marín, S. L. T., Arzamendia, M., & Gregor, D. O. (2021). A Bayesian optimization approach for water resources monitoring through an autonomous surface vehicle: The Ypacarai lake case study. *IEEE Access*, *9*, 9163–9179.

Samson, P. J. (1988). Atmospheric transport and dispersion of air pollutants associated with vehicular emissions. In A. Y. Watson, R. R. Bates, & D. Kennedy (Eds.), *Air Pollution, the Automobile, and Public Health.* National Academies Press (US).

Sanchez-Pi, N., Marti, L., Abreu, A., Bernard, O., de Vargas, C., Eveillard, D., ... Sebag, M. (2020). Artificial intelligence, machine learning and modeling for understanding the oceans and climate change. In *Neurips 2020 workshop-tackling climate change with machine learning*.

Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.

Schwaighofer, A., Tresp, V., & Yu, K. (2004). Learning Gaussian process kernels via hierarchical Bayes. *Advances in neural information processing systems*, *17*.

Schwarz, G. (1978). Estimating the dimension of a model. *The annals of statistics*, *6*(2), 461–464.

Settles, B. (2012). *Active learning*. Springer Cham.

Severson, K. A., Pfeiffer, O. P., Chen, J., Gong, K., Gregory, J., Goodwin, R., & Olivetti, E. A. (2021). Amortized inference of Gaussian process hyperparameters for improved concrete strength trajectory prediction. *Tackling Climate Change with Machine Learning: workshop at NeurIPS 2021*.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., & de Freitas, N. (2015). Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, *104*(1), 148–175. doi: 10.1109/JPROC.2015.2494218

Shilton, A., Gupta, S., Rana, S., & Venkatesh, S. (2017). Regret bounds for transfer learning in Bayesian optimisation. In *Artificial intelligence and statistics* (pp. 307–315).

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., ... Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, *362*(6419), 1140–1144.

Simonoff, J. S. (2003). *Analyzing categorical data* (Vol. 496). Springer.

Singh, A., Ramos, F., Whyte, H. D., & Kaiser, W. J. (2010). Modeling and decision making in spatio-temporal processes for environmental surveillance. In *2010 IEEE International Conference on Robotics and Automation* (pp. 5490–5497). doi: 10.1109/ROBOT.2010.5509934

Smedley, T. (2019). *Clearing the air: the beginning and the end of air pollution*. Bloomsbury Publishing.

Smith, M. T., Ssematimba, J., Álvarez, M. A., & Bainomugisha, E. (2019). Machine Learning for a Low-cost Air Pollution Network. *NeurIPS 2019 Workshop on Machine Learning for the Developing World*. (arXiv: 1911.12868)

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical Bayesian Optimization of Machine Learning Algorithms. *Advances in Neural Information Processing Systems*, *25*, 2951–2959.

Song, L., Xue, K., Huang, X., & Qian, C. (2022). Monte Carlo tree search based variable selection for high dimensional Bayesian optimization. *Advances in Neural Information Processing Systems*, *35*, 28488–28501.

Spinelle, L., Gerboles, M., Villani, M. G., Aleixandre, M., & Bonavitacola, F. (2015). Field calibration of a cluster of low-cost available sensors for air quality monitoring. Part A: Ozone and nitrogen dioxide. *Sensors and Actuators B: Chemical*, *215*, 249–257.

Spooner, T., Jones, A. E., Fearnley, J., Savani, R., Turner, J., & Baylis, M. (2020). Bayesian optimisation of restriction zones for bluetongue control. *Scientific reports*, *10*(1), 1–18.

Springenberg, J. T., Klein, A., Falkner, S., & Hutter, F. (2016). Bayesian optimization with robust Bayesian neural networks. In *Proceedings of the 29th international conference on advances in neural information processing systems*.

Srinivas, N., Krause, A., Kakade, S. M., & Seeger, M. (2012). Gaussian process optimization in the bandit setting: No regret and experimental design. *IEEE Transactions on Information Theory*, *58*(5), 3250–3265. (arXiv: 0912.3995)

Stoll, D., Franke, J. K. H., Wagner, D., Selg, S., & Hutter, F. (2020). Hyperparameter transfer across developer adjustments. *arXiv preprint arXiv:2010.13117*.

Sun, S., Tiihonen, A., Oviedo, F., Liu, Z., Thapa, J., Zhao, Y., ... Buonassisi, T. (2021). A data fusion approach to optimize compositional stability of halide perovskites. *Matter*, *4*(4), 1305–1322.

Swersky, K., Snoek, J., & Adams, R. P. (2013). Multi-task Bayesian optimization. *Advances in neural information processing systems*, *26*.

Tabakhi, A. M., Yeoh, W., & Fioretto, F. (2020). The smart appliance scheduling problem: A Bayesian optimization approach. In *International conference on principles and practice of multi-agent systems* (pp. 100–115).

Tanaka, Y., Tanaka, T., Iwata, T., Kurashima, T., Okawa, M., Akagi, Y., & Toda, H. (2019). Spatially Aggregated Gaussian Processes with Multivariate Areal Outputs. *33rd Conference on Neural Information Processing System*, 11.

Taylor, C. J., Felton, K. C., Wigh, D., Jeraal, M. I., Grainger, R., Chessari, G., ... Lapkin, A. A. (2023). Accelerated chemical reaction optimization using multi-task learning. *ACS Central Science*, *9*(5), 957–968.

The Lancet. (2016). Air pollution—crossing borders. *The Lancet*, *388*(10040), 103.

The World Bank. (2023). *World Bank DataBank health nutrition and population statistics: Population estimates and projections.* Retrieved 2023-10-09, from `https://databank.worldbank.org/source/population-estimates-and-projections`

Thebelt, A., Tsay, C., Lee, R., Sudermann-Merx, N., Walz, D., Shafei, B., & Misener, R. (2022). Tree ensemble kernels for Bayesian optimization with known constraints over mixed-feature spaces. *Advances in Neural Information Processing Systems*, *35*, 37401–37415.

Tiao, L. C., Klein, A., Seeger, M. W., Bonilla, E. V., Archambeau, C., & Ramos, F. (2021). BORE: Bayesian optimization by density-ratio estimation. In *International conference on machine learning* (pp. 10289–10300).

Tillmann, P., Jäger, K., & Becker, C. (2020). Minimising the levelised cost of electricity for bifacial solar panel arrays using Bayesian optimisation. *Sustainable Energy & Fuels*, *4*(1), 254–264.

Tran, K., & Ulissi, Z. W. (2018). Active learning across intermetallics to guide discovery of electrocatalysts for CO2 reduction and H2 evolution. *Nature Catalysis*, *1*(9), 696–703.

Trivedi, R., & Khadem, S. (2021). Peak demand management and schedule optimisation for energy storage through the machine learning approaches. In *IEEE EUROCON 2021-19th international conference on smart technologies* (pp. 157–163).

Tu, B., Gandy, A., Kantas, N., & Shafei, B. (2022). Joint entropy search for multi-objective Bayesian optimization. *Advances in Neural Information Processing Systems*, *35*.

University of Cambridge, D. o. C. (2019). *Low-cost air quality sensors help London breathe | Department of Chemistry.* Retrieved 2020-02-10, from `https://www.ch.cam.ac.uk/news/low-cost-air-quality-sensors-help-london-breathe`

Valentin, S., Kleinegesse, S., Bramley, N. R., Seriès, P., Gutmann, M. U., & Lucas, C. G. (2023). Designing optimal behavioral experiments using machine learning. *arXiv preprint arXiv:2305.07721*.

Van de Ven, G. M., & Tolias, A. S. (2019). Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*.

Van Donkelaar, A., Hammer, M. S., Bindle, L., Brauer, M., Brook, J. R., Garay, M. J., ... Martin, R. V. (2021). Monthly global estimates of fine particulate matter and their uncertainty. *Environmental Science & Technology*, *55*(22), 15287–15300.

Van Rijn, J. N., & Hutter, F. (2018). Hyperparameter importance across datasets. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 2367–2376).

Vanschoren, J. (2019). Meta-learning. In F. Hutter, L. Kotthoff, & J. Vanschoren (Eds.), *Automated machine learning: Methods, systems, challenges* (pp. 35–61). Cham: Springer International Publishing.

Vanschoren, J., van Rijn, J. N., Bischl, B., & Torgo, L. (2013). OpenML: networked science in machine learning. *SIGKDD Explorations*, *15*(2), 49-60.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., ... SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. doi: 10.1038/s41592-019-0686-2

Volpp, M., Fröhlich, L. P., Fischer, K., Doerr, A., Falkner, S., Hutter, F., & Daniel, C. (2020). Meta-learning acquisition functions for transfer learning in Bayesian optimization. In *International conference on learning representations*.

Wang, Z., Dahl, G. E., Swersky, K., Lee, C., Mariet, Z., Nado, Z., ... Ghahramani, Z. (2022). Pre-training helps Bayesian optimization too. *arXiv preprint arXiv:2207.03084*.

Wang, Z., & Jegelka, S. (2017). Max-value entropy search for efficient Bayesian optimization. In *International conference on machine learning* (pp. 3627–3635).

Wang, Z., Kim, B., & Kaelbling, L. P. (2018). Regret bounds for meta Bayesian optimization with an unknown gaussian process prior. *Advances in Neural Information Processing Systems*, *31*.

Watson-Parris, D. (2021). Machine learning for weather and climate are worlds apart. *Philosophical Transactions of the Royal Society A*, *379*(2194), 20200098.

Weiss, K., Khoshgoftaar, T. M., & Wang, D. (2016). A survey of transfer learning. *Journal of Big data*, *3*(1), 1–40.

Wilkinson, R., Álvarez López, M., & Bainomugisha, E. (2019). *Physically-informed probabilistic modelling of air pollution in Kampala using a low cost sensor network*. Retrieved 2019-09-30, from `https://gow.epsrc.ukri.org/NGBOViewGrant.aspx?GrantRef=EP/T00343X/1`

Williams, C., & Rasmussen, C. (1995). Gaussian processes for regression. *Advances in neural information processing systems*, *8*.

Wilson, A. G., Hu, Z., Salakhutdinov, R., & Xing, E. P. (2016). Deep kernel learning. In *Artificial intelligence and statistics* (pp. 370–378).

Wistuba, M., & Grabocka, J. (2021). Few-shot Bayesian optimization with deep kernel surrogates. In *International conference on learning representations*.

Wistuba, M., Schilling, N., & Schmidt-Thieme, L. (2015a). Learning data set similarities for hyperparameter optimization initializations. In *MetaSel@ PKDD/ECML* (pp. 15–26).

Wistuba, M., Schilling, N., & Schmidt-Thieme, L. (2015b). Sequential model-free hyperparameter tuning. In *2015 IEEE international conference on data mining* (pp. 1033–1038).

Wistuba, M., Schilling, N., & Schmidt-Thieme, L. (2018). Scalable gaussian process-based transfer surrogates for hyperparameter optimization. *Machine Learning*, *107*(1), 43–78.

World Health Organization. (2006). *Air quality guidelines: global update 2005: particulate matter, ozone, nitrogen dioxide, and sulfur dioxide*. Copenhagen, Denmark: World Health Organization. (OCLC: ocn162119780)

World Health Organization. (2021). *WHO ambient air quality database 2021 update* (Tech. Rep.). World Health Organization. Retrieved 2022-07-14, from https://cdn.who.int/media/docs/default-source/air-pollution-documents/air-quality-and-health/who-air-quality-database-2021_final.pdf?sfvrsn=2371d57c_5

World Health Organization, et al. (2021). *WHO global air quality guidelines: particulate matter (PM2. 5 and PM10), ozone, nitrogen dioxide, sulfur dioxide and carbon monoxide*. World Health Organization.

Wright, M. N., & Ziegler, A. (2017). ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software*, *77*, 1–17.

Wu, X., Gomes-Selman, J., Shi, Q., Xue, Y., Garcia-Villacorta, R., Anderson, E., ... Gomes, C. (2018). Efficiently approximating the pareto frontier: hydropower dam placement in the Amazon basin. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 32).

Yang, S., Deng, X., Ti, Z., Yan, B., & Yang, Q. (2022). Cooperative yaw control of wind farm using a double-layer machine learning framework. *Renewable Energy*, *193*, 519–537.

Yogatama, D., & Mann, G. (2014). Efficient transfer learning method for automatic hyperparameter tuning. In *Artificial intelligence and statistics* (pp. 1077–1085).

Zappella, G., Salinas, D., & Archambeau, C. (2021). A resource-efficient method for repeated HPO and NAS problems. *arXiv preprint arXiv:2103.16111*.

Zhan, S., Wichern, G., Laughman, C., Chong, A., & Chakrabarty, A. (2022). Calibrating building simulation models using multi-source datasets and meta-learned Bayesian optimization. *Energy and Buildings*, 112278.

Zhang, X., Lu, D., Dong, H., Zhao, X., Brennan, F., & Liang, Y. (2022). Vibration suppression of multi-component floating structures via passive TMDs and Bayesian ascent. *Ocean Engineering*, *259*, 112088.

Zhang, Y., Jordon, J., Alaa, A. M., & van der Schaar, M. (2019). Lifelong Bayesian optimization. *arXiv preprint arXiv:1905.12280*.

Zheng, T., Bergin, M. H., Sutaria, R., Tripathi, S. N., Caldow, R., & Carlson, D. E. (2019). Gaussian process regression model for dynamically calibrating and surveilling a wireless low-cost particulate matter sensor network in Delhi. *Atmospheric Measurement Techniques*, *12*(9), 5161–5181.

Zhong, M., Tran, K., Min, Y., Wang, C., Wang, Z., Dinh, C.-T., ... Sargent, E. H. (2020). Accelerated discovery of CO2 electrocatalysts using active machine learning. *Nature*, *581*(7807), 178–183.

Zhou, Y., Zhao, X., Lin, K.-P., Wang, C.-H., & Li, L. (2019). A Gaussian process mixture model-based hard-cut iterative learning algorithm for air quality prediction. *Applied Soft Computing*, *85*, 105789.

Appendix A

Publications

A.1 Optimising Placement of Pollution Sensors in Windy Environments

Citation: Hellan et al. (2020).

On page 3 a reference is made to 'gradient descent'. This should have been 'gradient-based optimisation'.

arXiv:2012.10770v2 [cs.LG] 28 Aug 2022

Optimising Placement of Pollution Sensors in Windy Environments

Sigrid Passano Hellan **Christopher G. Lucas** **Nigel H. Goddard**
School of Informatics
University of Edinburgh, UK
{s.p.hellan, c.lucas, nigel.goddard}

Abstract

Air pollution is one of the most important causes of mortality in the world. Monitoring air pollution is useful to learn more about the link between health and pollutants, and to identify areas for intervention. Such monitoring is expensive, so it is important to place sensors as efficiently as possible. Bayesian optimisation has proven useful in choosing sensor locations, but typically relies on kernel functions that neglect the statistical structure of air pollution, such as the tendency of pollution to propagate in the prevailing wind direction. We describe two new wind-informed kernels and investigate their advantage for the task of actively learning locations of maximum pollution using Bayesian optimisation.

Air pollution is one of the most important causes of death globally. Particulate matter with diameter less than $2.5\,\mu\mathrm{m}$, $PM_{2.5}$, is estimated to cause 3-4 million deaths yearly [1, 2]. Through monitoring one can learn more about the impact of air pollution on public health and identify areas requiring intervention. Traditional monitoring networks consist of a few high-quality reference sensors [3], but many low-cost pollution sensors are now available, e.g. [4], which can be used to build higher-resolution models to complement the traditional networks. This work aims to improve monitoring networks by improving the placement of low-cost, potentially noisy sensors to monitor average concentrations of $PM_{2.5}$ and identify areas of high average concentration, assuming that sensors can be placed incrementally. This problem – of finding the maxima of a black-box function given sparse and noisy observations – is well-matched to Bayesian optimisation (BO). Most machine learning research on air pollution tends not to consider the problem of efficient sensor placement, e.g. [5, 6, 7, 8, 9]. To our knowledge, previous work on environmental sensing using BO and related methods, e.g. [10, 11, 12], has not considered wind.

Wind is known to move pollutants downwind [13, p. 78-80], and work on gas mapping has shown the importance of incorporating this effect [14, 15]. For instance, [15] used a gas localisation task to show that a method incorporating wind velocity outperforms BO using wind-insensitive kernels. Because we are focusing on time-averaged pollution concentrations, detailed wind information and fluid simulations are not essential: The covariance structure of pollution is driven by the prevailing wind direction, which can be characterised with a single parameter. The kernels presented here allow such information to be incorporated into BO, and have the advantage that wind need not be measured separately. This is desirable from a cost perspective, as additional wind sensors are not needed.

The key question is whether these kernels improve our ability to find the location(s) of highest pollution in an area, with a minimum number of sensor placements, i.e., samples of time-averaged pollution concentrations. To answer this question using real data, we used satellite images of nitrogen dioxide (NO_2) concentrations (see Supplementary Material for details). NO_2 is a health-related pollutant in its own right [16], and we assume that the dispersion of $PM_{2.5}$ is comparable. While satellite data cannot replace surface-level sensor data, it provides a good testbed given the spatial regularity and density of the observations, and it allows us to set aside complex effects of urban topography.

AI for Earth Sciences Workshop at NeurIPS 2020.

1 Wind-informed kernels for Bayesian optimisation

In Bayesian optimisation [17] a probabilistic model, in our case a GP, is fitted to the set of observed data points. This model is then used to compute an acquisition function, and the next sample is taken at the point which maximises this function. The new sample is added to the set of data points and the process is repeated. In this paper, the upper confidence bound (UCB) [18] was used as the acquisition function, $\alpha_{\mathrm{UCB}}(\boldsymbol{x}; \beta, \mathcal{D}, M) = \mu(\boldsymbol{x}; \mathcal{D}, M) + \beta\sigma(\boldsymbol{x}; \mathcal{D}, M)$, with $\beta = 1$. $\mu(\boldsymbol{x}; \mathcal{D}, M)$ and $\sigma(\boldsymbol{x}; \mathcal{D}, M)$ are the posterior mean and standard deviation of a point $\boldsymbol{x}$ given data $\mathcal{D}$ and model M. An entropy-based acquisition function [19] and other values of β were tested, without marked improvement. To compute $\mu(\cdot)$ and $\sigma(\cdot)$, covariances need to be computed between the data points using a kernel function. A thorough introduction to GPs can be found in [20].

The motivation for the wind-informed kernels is to exploit the directional structure of wind-driven pollution transport. Intuitively, one would expect to see the highest levels of pollution downwind from a source, and if one recorded a high pollution concentration in a location one would expect to find a source upwind from it. In this work we do not aim to locate sources, but to look at concentrations, and so do not differentiate between upwind and downwind movement.

The standard radial basis function (RBF) was modified to create the wind-informed kernels. The RBF kernel calculates the covariance between two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ as $k_R(\boldsymbol{\tau}; l, \sigma_R) = \sigma_R^2 \exp(-\boldsymbol{\tau}^T\boldsymbol{\tau}/l^2)$ where $\boldsymbol{\tau} = \boldsymbol{x}_i - \boldsymbol{x}_j$, l is the length scale and σ_R^2 the signal variance. The new kernels $k_S(\cdot)$, *Sum*, and $k_P(\cdot)$, *Product*, are given in Eqs. (1) and (2) and visualised in Fig. 1. The kernels are a combination of a standard RBF kernel and another RBF kernel on the distance orthogonal to the wind direction. This distance is given by $\tilde{\tau}(\boldsymbol{\tau}; \gamma) = ||\boldsymbol{\tau} - (\boldsymbol{\tau}^T\boldsymbol{b})\,\boldsymbol{b}||$ where $\boldsymbol{b} = [\cos(\gamma)\ \ \sin(\gamma)]^T$ and γ is the wind direction hyperparameter. Further, l_D is the directional length scale, σ_D^2 the directional signal variance, σ_P^2 the signal variance for $k_P(\cdot)$, $A = \begin{bmatrix} \sin(\gamma)^2 & -\sin(\gamma)\cos(\gamma) \\ -\sin(\gamma)\cos(\gamma) & \cos(\gamma)^2 \end{bmatrix}$ and $E = l_D^2/l^2\mathbb{I} + A$. Finally, $\theta_S = \{l, l_D, \sigma_R, \sigma_D, \gamma\}$ and $\theta_P = \{l, l_D, \sigma_P, \gamma\}$ denote sets of hyperparameters. The kernel hyperparameters, together with the noise hyperparameter σ_n^2, are fitted in the GP tuning step.

$$k_S(\boldsymbol{\tau}; \theta_S) = k_R(\boldsymbol{\tau}; l, \sigma_R) + k_R(\tilde{\tau}(\boldsymbol{\tau}; \gamma); l_D, \sigma_D) = \sigma_R^2 \exp\left(\frac{-\boldsymbol{\tau}^T\boldsymbol{\tau}}{l^2}\right) + \sigma_D^2 \exp\left(\frac{-\boldsymbol{\tau}^T A\boldsymbol{\tau}}{l_D^2}\right) \quad (1)$$

$$k_P(\boldsymbol{\tau}; \theta_P) = k_R(\boldsymbol{\tau}; l, \sigma_P)\, k_R(\tilde{\tau}(\boldsymbol{\tau}; \gamma); l_D, 1) = \sigma_P^2 \exp\left(\frac{-\boldsymbol{\tau}^T E\boldsymbol{\tau}}{l_D^2}\right) \quad (2)$$

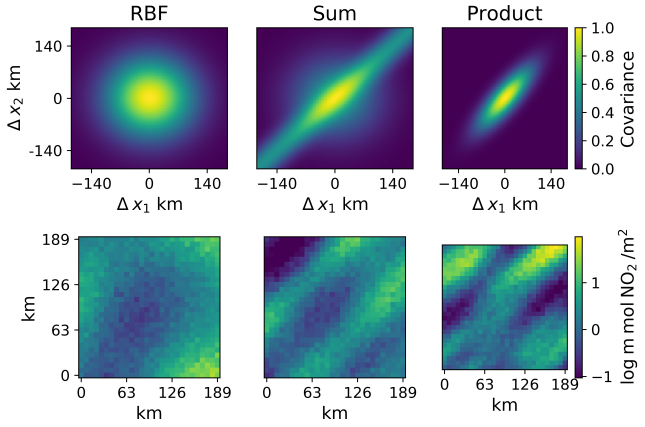


Figure 1: Visualisation of wind-informed kernels from Eqs. (1) and (2). The top row shows covariance as function of distance, and the bottom row shows example images sampled from the kernels. $\sigma_D^2 = 0.5$, $l = 105$, $l_D = 35$, $\gamma = \pi/4$, $\sigma_P^2 = 1$, $\sigma_n^2 = 0.01$. For Sum $\sigma_R^2 = 0.5$, for RBF $\sigma_R^2 = 1$.

These kernels are sums and products of RBF kernel functions that are valid for all inputs, and are thus valid kernel functions themselves [20, p. 83,95].

2 Data

The data set consists of 1083 images of 28x28 pixels, with each pixel giving the NO_2 concentration in mol/m^2 within a ca. 7x7km box on the ground reaching the upper troposphere. The images are from October and November 2018 and cover landmasses across the world. More information is given in the Supplementary Material. We removed images with more than 10 % missing values, and split the data into subsets. The split was done as NO_2 concentrations varied greatly between images, and locating peaks of high pollution is more important. Three subsets were created as follows: the images were ordered by the highest NO_2 concentration present, and the top, median and bottom 50 images put into sets called 'Strong', 'Median' and 'Weak', respectively. See Fig. 2. Three adjacent sets of 10 images each were created for tuning and normalisation. A final set of 100 images, dubbed 'Selection', was created by picking every ninth of the remaining ordered images. The first three subsets were created to inspect performance on different degrees of difficulty, and the last to test the overall performance.

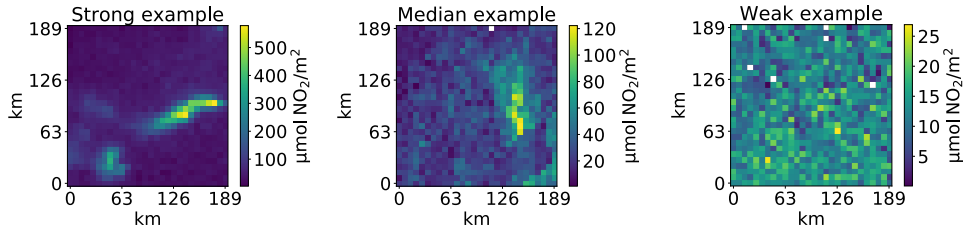


Figure 2: Example NO_2 concentration images from the Strong, Median and Weak subsets. White pixels indicate missing data. Note the difference in scale between the subsets, and the pollution plumes from point sources in two of the examples. Such plumes were present in many images, particularly in the Strong subset, and are what the wind-informed kernels are meant to capture.

3 Results

We used log concentrations rather than raw concentrations as our data – and pollution concentrations in general [21] – are approximately log-normal, satisfying the assumption of Gaussian marginal and residual distributions in GP models. Images were normalised using means and standard deviations from the associated tuning set. The rationale for this was that the true mean and standard deviation for a deployment region would likely be unknown, but values from similar areas available. The tuning sets were also used to develop priors for the hyperparameters, except for γ for which a uniform prior was used. For each tuning image, gradient descent with 200 hyperparameter initialisations was used to maximise the log marginal likelihood, $\log(p(Y^i|M))$, where M denotes the model, including kernel and hyperparameter choices, and Y^i is the pixel values in image i. A log normal distribution was then fitted to these hyperparameter samples using the maximum likelihood estimator and this was used as a prior. For the Selection subset, the Median tuning set was used for normalisation and prior computation. The models obtained on the tuning sets were compared using the Bayesian information criterion (BIC) [22]. It was highest for the Sum kernel, followed by the Product kernel, c.f. the Supplementary Material. This shows that the wind-informed kernels are able to capture more of the structure of the pollution in the images. For the Weak subset little difference was seen.

The kernels were evaluated on the task of locating the pixel with the maximum pollution concentration. Results on two of the subsets, Strong and Selection, are presented. For comparison, our baselines were (1) Bayesian optimisation with an RBF kernel (RBF) and (2) a policy of choosing points uniformly at random (Random). The latter was repeated 100 times per image and the mean taken. The experiments were initiated with 10 randomly selected pixel values. Fig. 3 shows the performance as a function of iterations done, showing the Sum kernel performs better than the RBF kernel on the Strong subset. For the selection subset, the Random baseline beats the RBF baseline. This indicates that the problem is hard for BO, as would be expected from the example images in Fig. 2. The Product kernel outperforms the other kernels on the distance metric, while the ratio metric is very similar for all methods. Fig. 4 looks more closely at the performance on the Selection subset as a function of the maximum pollution concentration present in each image. It shows that the performance improves with higher levels of pollution and also that over most of the concentration values the Product kernel is better than the RBF.

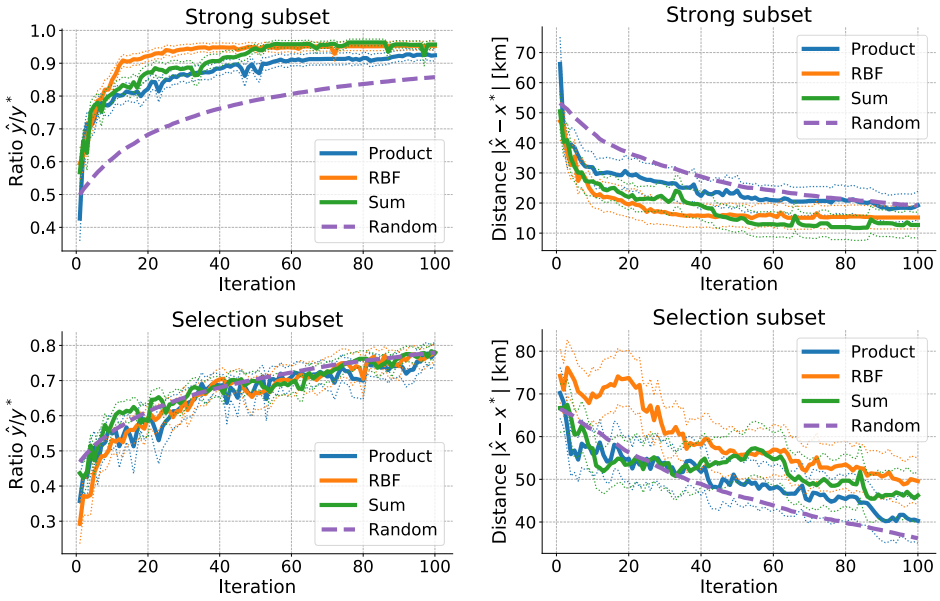


Figure 3: Comparison of kernel performance on Strong and Selection subsets. $\hat{x}$ is the estimated maximiser and x^* the true maximiser. $\hat{y}$ and y^* are the true concentration values at $\hat{x}$ and x^*, respectively. Both metrics are shown as means with standard deviation of the mean in dotted lines. The Sum kernel reaches slightly better values than the RBF kernel on the Strong subset, and the Product kernel outperforms the other kernels on the distance metric on the Selection subset.

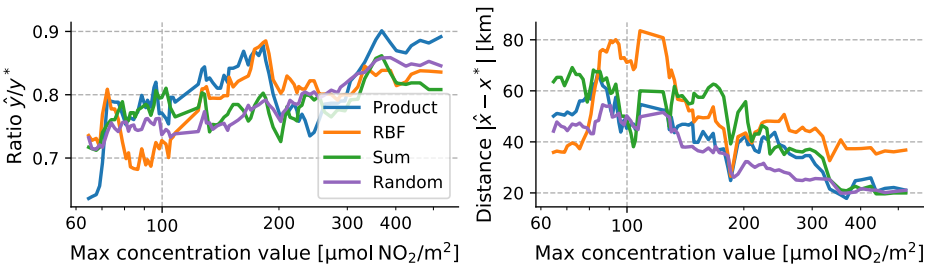


Figure 4: Inspection of performance after 100 iterations over the range of concentrations in the Selection subset. Each line shows the running average of the 20 previous values. $\hat{x}$, x^*, $\hat{y}$ and y^* as defined for Fig. 3. As can be seen the Product kernel outperforms RBF across most concentration.

4 Discussion and future work

By augmenting a standard RBF kernel with a directional element that captures the prevailing wind direction, we can better capture the covariance structure in NO_2 pollution, as indicated by the BIC scores. The results also show that this better estimates pollution maxima and the locations of those maxima for some of the images. The improvement was less clear than we expected from the BIC scores. We hypothesise that this is due to the hyperparameter tuning, which the results were sensitive to. For an example, see the Supplementary Material. Therefore, we plan to improve our priors by basing them on a larger tuning set and using hierarchical modelling. Additionally, the wind-informed kernels should be tested on more data sets, and particularly on data sets of urban air pollution. As wind is also important in moving air pollution over short distances one would expect improvement, but this might be distorted by other factors, like buildings. The approach used, of incorporating a data transformation into the covariance function, can be extended to other kernels, e.g. the set of Matérn kernels, or combined in other ways, e.g. a combination of Sum and Product kernels.

Acknowledgments and Disclosure of Funding

The authors would like to thank Dr Douglas Finch at the University of Edinburgh for extracting the data and helpful correspondence.

This work was supported in part by the EPSRC Centre for Doctoral Training in Data Science, funded by the UK Engineering and Physical Sciences Research Council (grant EP/L016427/1) and the University of Edinburgh.

References

[1] Aaron J Cohen, Michael Brauer, Richard Burnett, H Ross Anderson, Joseph Frostad, Kara Estep, Kalpana Balakrishnan, Bert Brunekreef, Lalit Dandona, Rakhi Dandona, Valery Feigin, Greg Freedman, Bryan Hubbell, Amelia Jobling, Haidong Kan, Luke Knibbs, Yang Liu, Martin Randall, and Mohammad H Forouzanfar. Estimates and 25-year trends of the global burden of disease attributable to ambient air pollution: an analysis of data from the Global Burden of Diseases Study 2015. *The Lancet*, 389(10082): 1907–1918, May 2017. ISSN 0140-6736. doi: 10.1016/S0140-6736(17)30505-6. URL https://www.sciencedirect.com/science/article/pii/S0140673617305056.

[2] J. Lelieveld, J. S. Evans, M. Fnais, D. Giannadaki, and A. Pozzer. The contribution of outdoor air pollution sources to premature mortality on a global scale. *Nature*, 525(7569):367–371, September 2015. ISSN 1476-4687. doi: 10.1038/nature15371. URL https://www.nature.com/articles/nature15371. Number: 7569 Publisher: Nature Publishing Group.

[3] Marco Carminati, Giorgio Ferrari, and Marco Sampietro. Emerging miniaturized technologies for airborne particulate matter pervasive monitoring. *Measurement*, 101:250–256, April 2017. ISSN 0263-2241. doi: 10.1016/j.measurement.2015.12.028. URL http://www.sciencedirect.com/science/article/pii/S0263224115006880.

[4] Xiaoting Liu, Rohan Jayaratne, Phong Thai, Tara Kuhn, Isak Zing, Bryce Christensen, Riki Lamont, Matthew Dunbabin, Sicong Zhu, Jian Gao, David Wainwright, Donald Neale, Ruby Kan, John Kirkwood, and Lidia Morawska. Low-cost sensors as an alternative for long-term air quality monitoring. *Environmental Research*, 185:109438, June 2020. ISSN 0013-9351. doi: 10.1016/j.envres.2020.109438. URL http://www.sciencedirect.com/science/article/pii/S0013935120303315.

[5] Michael T. Smith, Joel Ssematimba, Mauricio A. Alvarez, and Engineer Bainomugisha. Machine Learning for a Low-cost Air Pollution Network. *NeurIPS 2019 Workshop on Machine Learning for the Developing World*, December 2019. URL http://arxiv.org/abs/1911.12868. arXiv: 1911.12868.

[6] Xiuming Liu, Teng Xi, and Edith Ngai. Data Modelling with Gaussian Process in Sensor Networks for Urban Environmental Monitoring. In *2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 457–462, September 2016. doi: 10.1109/MASCOTS.2016.45. ISSN: 2375-0227.

[7] Hongbin Liu, Chong Yang, Mingzhi Huang, Dongsheng Wang, and ChangKyoo Yoo. Modeling of subway indoor air quality using Gaussian process regression. *Journal of Hazardous Materials*, 359: 266–273, October 2018. ISSN 0304-3894. doi: 10.1016/j.jhazmat.2018.07.034. URL http://www.sciencedirect.com/science/article/pii/S0304389418305648.

[8] Haolin Fei, Xiaofeng Wu, Chunbo Luo, and Yang Luo. Accurate Air Quality Prediction: A Physical-Temporal Collection Model. In *ICLR Workshop on AI for Earth Science*, 2020. URL https://www.researchgate.net/publication/341445468_ACCURATE_AIR_QUALITY_PREDICTION_A_PHYSICAL-_TEMPORAL_COLLECTION_MODEL_CONFERENCE_SUBMISSIONS.

[9] Congcong Wen, Shufu Liu, Xiaojing Yao, Ling Peng, Xiang Li, Yuan Hu, and Tianhe Chi. A novel spatiotemporal convolutional long short-term neural network for air pollution prediction. *Science of The Total Environment*, 654:1091–1099, March 2019. ISSN 0048-9697. doi: 10.1016/j.scitotenv.2018.11.086. URL http://www.sciencedirect.com/science/article/pii/S0048969718344413.

[10] R. Marchant and F. Ramos. Bayesian optimisation for Intelligent Environmental Monitoring. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2242–2249, October 2012. doi: 10.1109/IROS.2012.6385653.

[11] Philippe Morere, Roman Marchant, and Fabio Ramos. Sequential Bayesian optimization as a POMDP for environment monitoring with UAVs. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6381–6388, May 2017. doi: 10.1109/ICRA.2017.7989754.

[12] A. Singh, F. Ramos, H. D. Whyte, and W. J. Kaiser. Modeling and decision making in spatio-temporal processes for environmental surveillance. In *2010 IEEE International Conference on Robotics and Automation*, pages 5490–5497, May 2010. doi: 10.1109/ROBOT.2010.5509934.

[13] Perry J Samson. Atmospheric Transport and Dispersion of Air Pollutants Associated with Vehicular Emissions. In Ann Y. Watson, Richard R. Bates, and Donald Kennedy, editors, *Air Pollution, the Automobile, and Public Health*. National Academies Press (US), 1988. URL https://www.ncbi.nlm.nih.gov/books/NBK218142/. Publication Title: Air Pollution, the Automobile, and Public Health.

[14] Matteo Reggente and Achim J. Lilienthal. Using local wind information for gas distribution mapping in outdoor environments with a mobile robot. In *2009 IEEE SENSORS*, pages 1715–1720, October 2009. doi: 10.1109/ICSENS.2009.5398498. ISSN: 1930-0395.

[15] Martin Asenov, Marius Rutkauskas, Derryck Reid, Kartic Subr, and Subramanian Ramamoorthy. Active Localization of Gas Leaks Using Fluid Simulation. *IEEE Robotics and Automation Letters*, 4(2):1776–1783, April 2019. ISSN 2377-3766. doi: 10.1109/LRA.2019.2895820. Conference Name: IEEE Robotics and Automation Letters.

[16] Kai Luo, Runkui Li, Wenjing Li, Zongshuang Wang, Xinming Ma, Ruiming Zhang, Xin Fang, Zhenglai Wu, Yang Cao, and Qun Xu. Acute Effects of Nitrogen Dioxide on Cardiovascular Mortality in Beijing: An Exploration of Spatial Heterogeneity and the District-specific Predictors. *Scientific Reports*, 6(1): 38328, December 2016. ISSN 2045-2322. doi: 10.1038/srep38328. URL https://www.nature.com/articles/srep38328. Number: 1 Publisher: Nature Publishing Group.

[17] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proceedings of the IEEE*, 104(1):148–175, January 2016. ISSN 1558-2256. doi: 10.1109/JPROC.2015.2494218. Conference Name: Proceedings of the IEEE.

[18] Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias Seeger. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. *IEEE Transactions on Information Theory*, 58(5):3250–3265, May 2012. ISSN 0018-9448, 1557-9654. doi: 10.1109/TIT.2011.2182033. URL http://arxiv.org/abs/0912.3995. arXiv: 0912.3995.

[19] José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. Predictive Entropy Search for Efficient Global Optimization of Black-box Functions. *Advances in Neural Information Processing Systems 27*, pages 918–926, 2014.

[20] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[21] GJ Cats and AAM Holtslag. Prediction of air pollution frequency distribution—Part I. The lognormal model. *Atmospheric Environment (1967)*, 14:255–258, 1980. URL https://www.sciencedirect.com/science/article/pii/0004698180902851.

[22] Gideon Schwarz. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

A Supplementary Material

Supplementary material for *Optimising Placement of Pollution Sensors in Windy Environments* by Sigrid Passano Hellan, Christopher G. Lucas and Nigel H. Goddard presented at the AI for Earth Sciences Workshop at NeurIPS 2020.

A.1 Data

The data were collected by the TROPOspheric Measuring Instrument (TROPOMI) aboard the Sentinel 5P satellite,[1] and were extracted and shared with us by Dr Douglas Finch at the University of Edinburgh. In the original data source, the NO_2 data used is given in the column labelled 'level 2 nitrogen dioxide tropospheric'. Images covering mostly oceans were excluded.

A.2 BIC scores

The Bayesian information criterion (BIC) scores of the tuning sets referred to in Section 3 are given in Table 1. The values are given as differences because the obtained BIC scores varied a lot between problems, e.g. between 306 and -1014 for the Sum kernel on the Strong subset. As can be seen, the relative differences vary a lot less.

Table 1: Differences in Bayesian information criterion (BIC) scores obtained on tuning sets. Mean (standard deviation of mean). The wind-informed kernels were better able to explain the data for the Strong and Median subsets. There was little difference for the Weak subset, where the images were very noisy.

	Sum - Product	Sum - RBF	Product - RBF
Strong tune	13.43 (3.69)	19.92 (3.44)	6.49 (2.75)
Median tune	2.69 (1.95)	10.02 (4.30)	7.33 (3.73)
Weak tune	0.35 (0.58)	-1.12 (1.72)	-1.47 (1.34)

A.3 Implementation notes

A small amount of noise was added to sampled values in the BO loop for numerical stability. At each iteration in the BO loop 100 hyperparameter seeds were sampled from the priors and used as starting points for the GP tuning. The setting which maximised the posterior log marginal likelihood was then used. When estimating the hyperparameter priors, Bessel's correction was used for the variance estimate.

A.4 Hyperparameter prior sensitivity

Fig. 5 shows the results on the Strong subset when a slightly different prior was used for the hyperparameters, as referred to in Section 4 of the main paper.

[1] Copernicus, `https://www.copernicus.eu/en`

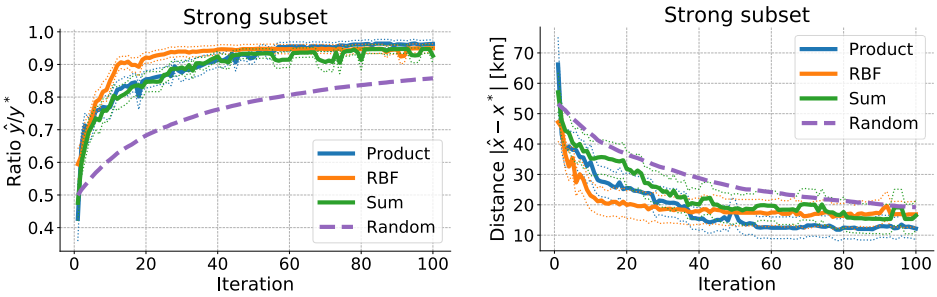


Figure 5: Comparison of kernel performance on Strong subsets, when biased estimates of variances of hyperparameter priors are used, i.e. not using Bessel's correction. Compare to Fig. 3 in body of paper. With these settings, the Product kernel performs best. Comparing the figures, the Product kernel has improved but the RBF worsened. $\hat{x}$ is the estimated maximiser and x^* the true maximiser. $\hat{y}$ and y^* are the true concentration values at $\hat{x}$ and x^*, respectively.

A.2 Bayesian Optimisation for Active Monitoring of Air Pollution

Citation: Hellan et al. (2022).

Bayesian Optimisation for Active Monitoring of Air Pollution

Sigrid Passano Hellan, Christopher G. Lucas and Nigel H. Goddard

School of Informatics
University of Edinburgh, UK
{s.p.hellan, c.lucas, nigel.goddard}

arXiv:2202.07595v2 [cs.LG] 19 Apr 2024

Abstract

Air pollution is one of the leading causes of mortality globally, resulting in millions of deaths each year. Efficient monitoring is important to measure exposure and enforce legal limits. New low-cost sensors can be deployed in greater numbers and in more varied locations, motivating the problem of efficient automated placement. Previous work suggests Bayesian optimisation is an appropriate method, but only considered a satellite data set, with data aggregated over all altitudes. It is ground-level pollution, that humans breathe, which matters most. We improve on those results using hierarchical models and evaluate our models on urban pollution data in London to show that Bayesian optimisation can be successfully applied to the problem.

Introduction

Ambient air pollution is one of the leading causes of death globally, with particulate matter with diameter less than $2.5\,\mu\mathrm{m}$, $PM_{2.5}$, causing 3-4 million deaths each year (Cohen et al. 2017; Lelieveld et al. 2015). One commonly monitored pollutant is nitrogen dioxide, NO_2, as it is used to indicate the presence of traffic-generated air pollution (Katsouyanni 2003). NO_2 is also part of an ozone-generating process, and short-term increases in ozone concentration are followed by short-term increases in mortality (Katsouyanni 2003). Due to these adverse effects the WHO have produced guidelines limiting pollution concentrations, and many countries have adopted air pollution regulations (World Health Organization 2006, p. 174-175).

Traditional air pollution monitoring uses large and expensive sensors that are typically managed by national or municipal authorities deciding where to locate sensors based on domain knowledge and constraints posed by the bulky nature of the sensors (Carminati, Ferrari, and Sampietro 2017). More recently, low-cost air pollution sensors have become available, e.g. Liu et al. (2020) and Kelly et al. (2017), which open up air pollution monitoring to more locations, and allow groups with limited budgets and domain expertise to create sensor networks, including local governments in developing nations as well as community groups around the world. By automating the decision-making process for placing air pollution sensors, we can help these groups make the most of their limited resources.

This goal, to simplify the process for setting up new monitoring networks in a low-cost way, motivates our decision to focus on computationally relatively inexpensive models and simple feature sets. Although models are generally improved by using data on road traffic and other pollution sources, these data can be difficult or expensive to obtain for developing nations or citizen scientists. We demonstrate that even a minimal feature set containing only sensor locations and readings can be useful. Additionally, all the computations are done on CPUs, so access to GPUs is not needed.

Simplicity and explainability also guide the model design. Experimental exploration showed the need for informative priors, as simpler approaches without hyperparameter priors or with only basic priors did not perform well. Instead, we found a pragmatic solution using related data to construct a prior through a hierarchical model. In that way data from other cities can be used for the prior, and the relevance of each other city inferred. Gaussian processes are used to model the data, which have interpretable hyperparameters, aiding explainability. And the number of hyperparameters is limited by keeping the covariance functions simple.

Bayesian optimisation (BO) has been used for pollution monitoring previously, but has tended either to estimate hyperparameters or their distributions in ways that do not generalise to new sites in a sample-efficient way (Ainslie et al. 2009) or to focus on robot trajectories (Morere, Marchant, and Ramos 2017; Singh et al. 2010; Marchant and Ramos 2012), a distinct problem from the one we consider here. Work has also been done on monitoring gas leaks (Reggente and Lilienthal 2009; Asenov et al. 2019), but again by using a single moving agent. An exception is Hellan, Lucas, and Goddard (2020), but it is limited to satellite data as a proxy for ground measurements. We extend on it with a more principled methodology and better results, as well as an evaluation on a more relevant data set taken from the London Air Quality Network (LAQN) (Imperial College London 1993). Our motivations are similar to Smith et al. (2019), who use Gaussian processes for calibrating low-cost pollution sensors, but do not address the problem of sensor placement.

The main contribution of this paper is showing Bayesian optimisation to be useful for automating planning of pollution sensor networks. Specifically, we consider the problem of iteratively placing stationary sensors and locating the maximum average pollution in the given area. Knowledge

of the maximum lets us know whether regulations are being adhered to. We also discuss the general usefulness of the methodology adopted.

Background

Bayesian optimisation

Bayesian optimisation (Shahriari et al. 2015) is an optimisation method based on maintaining a probabilistic model $m(x)$ of the underlying problem $f(x)$. At each iteration the model is fitted to the collected data, the next sampling location chosen according to the acquisition function $a(m(x))$ and a sample collected. The acquisition function balances the exploration/exploitation trade-off. At each iteration, the problem $\max_x a(m(x))$ is solved instead of $\max_x f(x)$. While evaluating $f(x)$ requires taking a measurement, $a(m(x))$ can be calculated from the model. One acquisition function is expected improvement (Jones, Schonlau, and Welch 1998), $a(x) = \mathbb{E}[\max(0, f(x) - f')|\, \mathcal{D}]$ where $\mathcal{D}$ is the data observed and f' is the highest sample from previous iterations. The model is usually a Gaussian process (GP) (Rasmussen and Williams 2006).

Hierarchical Bayesian modelling

Hierarchical Bayesian modelling is the principle of having layers of random variables built on top of each other (Shiffrin et al. 2008). The advantage of using a hierarchical model for BO is that it can be used to transfer information about the distribution of GP hyperparameters across different contexts – such as cities – without assuming that these contexts are identical. This is advantageous in applications like pollution monitoring where there are plentiful observations for some contexts but not others, e.g. cities with and without extensive pollution monitoring programs, and sample efficiency is paramount as each sample is expensive to collect. Hierarchical Bayesian modelling has been used for air pollution monitoring and modelling, e.g. Dawkins et al. (2020), Cocchi, Greco, and Trivisano (2007) and Sahu (2012), but not, to our knowledge, in combination with Bayesian optimisation.

Approximate inference

Although the standard approach for BO is to use a single point estimate for hyperparameters, a more Bayesian approach has several advantages, including better sample efficiency and more accurate estimates of posterior uncertainty (De Ath, Everson, and Fieldsend 2021; Snoek, Larochelle, and Adams 2012). Instead of a single sample, a set of samples (Monte Carlo approximation) or a parameterised distribution (variational inference) can be used to represent the hyperparameters. However, it makes the inference more computationally challenging. The posterior can be calculated in closed form for the standard GP model with a point estimate of the hyperparameters (Rasmussen and Williams 2006), but this is not possible for the hierarchical models. Following previous work, we use Monte Carlo approximation rather than variational inference, as De Ath, Everson, and Fieldsend (2021) found the latter to be less accurate. The Bayesian treatment of hyperparameters has been considered more widely for non-BO Gaussian processes, e.g. in Lalchand and Rasmussen (2020), Murray and Adams (2010) and Williams and Rasmussen (1995).

In general when using approximate inference for model hyperparameters, one tries to solve Eq. (1), where X^* and Y^* are the test features and values, θ_j the model hyperparameters and $\mathcal{D}_j$ the observed data. In contrast, we are solving Eq. (2) where $\mathcal{D}$ is observed data from related problems, i.e. our prior information. By factoring out the dependence on $\mathcal{D}$, we can split the inference problem into two separate ones, which we can solve separately. The first one takes into account $\mathcal{D}$ and only needs to be solved once using Markov chain Monte Carlo. The second one takes into account $\mathcal{D}_j$ and is solved each time a new observation is added through the BO iterations, using importance weighting.

$$p(Y^*|X^*, \mathcal{D}_j) = \int p(Y^*|X^*, \theta_j) p(\theta_j|\mathcal{D}_j) d\theta_j \quad (1)$$

$$p(Y^*|X^*, \mathcal{D}, \mathcal{D}_j) = \int p(Y^*|X^*, \theta_j) p(\theta_j|\mathcal{D}_j, \mathcal{D}) d\theta_j \quad (2)$$

We found a simple Metropolis-Hastings based approach (Chib and Greenberg 1995) worked well, but gradient-aware methods, e.g. Hoffman and Gelman (2014), may be more appropriate for more complex kernels and higher-dimensional parameter spaces.

Methodology

While the heart of our method is a standard Bayesian optimisation loop, its effectiveness relies on problem-specific design decisions, data pre-processing, and pre-training in order to obtain an appropriate joint prior over GP parameters.

Pollutant concentrations tend to follow a log-normal distribution (Cats and Holtslag 1980), so we log-transformed our concentration data to match the distributional assumptions behind Gaussian processes. We also standardised (satellite) or mean-centred (LAQN) our data, see the appendix. For evaluation, the observed data is considered to be the ground truth, so the variance of the noise in the GP likelihood is clamped to a small value (1e-06). The acquisition function used is expected improvement (Jones, Schonlau, and Welch 1998) and importance weighting is used to adapt the hyperparameter samples to the problem at hand. Each problem is initiated with some randomly selected samples; 10 for the satellite data and 5 for the London data. The computing resources used are described in the appendix.

GP models

The GP models are constructed from two base covariance functions, the RBF kernel k_{R} in Eq. (3) and a directed version k_{W} from Hellan, Lucas, and Goddard (2020), given in Eq. (4). The latter only considers the spatial distance orthogonal to a reference direction γ, conceptualised as the wind direction. τ is the difference between feature vectors. In Eq. (4), $A = \begin{bmatrix} \sin(\gamma)^2 & -\sin(\gamma)\cos(\gamma) \\ -\sin(\gamma)\cos(\gamma) & \cos(\gamma)^2 \end{bmatrix}$.

$$k_{\mathrm{R},i}(\tau) = \sigma^2_{\mathrm{r},i} \exp\left(\frac{-\tau^T\tau}{l^2_{\mathrm{r},i}}\right) \quad (3)$$

$$k_{\mathrm{W},i}(\tau) = \sigma^2_{\mathrm{w},i} \exp\left(\frac{-\tau^T A\tau}{l^2_{\mathrm{w},i}}\right) \quad (4)$$

From k_{R} and k_{W} three composite covariance functions are constructed, which are the ones evaluated in this paper. They are given in Eqs. (5) to (7) and consist of a sum of two terms, which are meant to capture the slowly-varying and faster-varying parts of the observed signal. Including the noise hyperparameter, the RBF-RBF model has 5 hyperparameters, the RBF-Product model 7 and the Sum model 6.

$$k_{\mathrm{RBF-RBF}}(\tau) = k_{\mathrm{R},1}(\tau) + k_{\mathrm{R},2}(\tau) \quad (5)$$

$$k_{\mathrm{RBF-Product}}(\tau) = k_{\mathrm{R},1}(\tau) + k_{\mathrm{R},2}(\tau)k_{\mathrm{W},3}(\tau) \quad (6)$$

$$k_{\mathrm{Sum}}(\tau) = k_{\mathrm{R},1}(\tau) + k_{\mathrm{W},2}(\tau) \quad (7)$$

Hierarchical structure

Problem-appropriate hyperparameter settings are essential for Bayesian optimisation to be efficient. Too large a lengthscale $l_{\mathrm{r},i}$ and the process is assumed to vary too slowly and relevant locations are not explored. Too small a lengthscale and resources are wasted checking locations that are unlikely to be high and provide little new information. These pitfalls can be avoided using a prior that is informed by relevant data from other contexts. To that end, our model is structured hierarchically as described in Fig. 1, and the prior inferred from a related tuning set $(\mathcal{D}_1, \cdots, \mathcal{D}_N)$. The lowest level of the hierarchical model is the data snapshots, which are the only variables that are observed directly. GP models are fitted to the data snapshots, and the hyperparameters θ of the GP models constitute the middle layer of the model. The top level is the parameters defining the distribution of the GP hyperparameters. This can be expressed mathematically as $p(\eta, \theta, \mathcal{D}) = p(\eta)p(\theta|\eta)p(\mathcal{D}|\theta) = p(\eta)\prod_i p(\theta_i|\eta)p(\mathcal{D}_i|\theta_i)$.

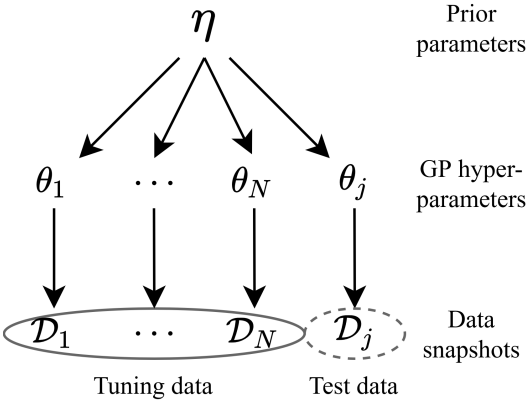


Figure 1: Visualisation of the hierarchical structure adopted. The bottom level is the observed data $\mathcal{D}$. It is modelled using GPs, which are defined by their hyperparameters θ. The distribution of the GP hyperparameters is captured by η. The hyperparameters θ are independent given η.

The structure reflects the assumption that the data snapshots are generated by distinct but similar underlying processes. Therefore, the tuning hyperparameters $(\theta_1, \cdots, \theta_N)$ cannot be used directly, but their distribution η can be.

For the BO loop we want the expectation of the acquisition function on snapshot j at point x_* given the distribution of GP hyperparameters, It can be approximated as

$$a_j(x_*) = \mathop{\mathbb{E}}_{p(\theta_i|\mathcal{D},\mathcal{D}_j)}[a_{j,i}(x_*)] \quad (8)$$

$$= \mathop{\mathbb{E}}_{p(\theta_i|\eta)p(\mathcal{D}_j|\theta_i)/Z}[a_{j,i}(x_*)] \quad (9)$$

$$\approx \frac{1}{M}\sum_{i=1}^{M} a_{j,i}(x_*), \;\; \theta_i \sim \frac{1}{Z}p(\theta_i|\eta)p(\mathcal{D}_j|\theta_i) \quad (10)$$

$$\approx \frac{1}{W}\sum_{i=1}^{M} a_{j,i}(x_*)p(\mathcal{D}_j|\theta_i), \;\; \theta_i \sim p(\theta_i|\eta) \quad (11)$$

where $W = \sum_{i=1}^{M} p(\mathcal{D}_j|\theta_i)$ is the sum of the importance weights and Z a normalising constant. $a_{j,i}(x_*)$ is the acquisition function evaluated on snapshot j using hyperparameter sample θ_i on the point x_*. In the second line η is used to summarise $\mathcal{D}$. In the third the expectation is approximated using Monte Carlo. In the fourth importance weighting is used so samples can be generated from the prior η, and weighted by their fit to the specific problem $\mathcal{D}_j$. By taking MCMC samples from $p(\theta_i|\eta)$ and not $p(\theta_i|\eta)p(\mathcal{D}_j|\theta_i)$ the computations are sped up greatly because we can approximate the distribution once and then reuse it at each iteration of the BO loop and for each test data snapshot.

MCMC sampling

The hyperparameter samples used for the prior are inferred from the tuning set using MCMC. It is done separately for each of the model kernels (RBF-Product, Sum and RBF-RBF). Joint samples are collected of the top two levels of the hierarchical structure from Fig. 1; η and $\theta = \bigcup_{n\in N}\theta_n$. Note that θ_j is not included. θ_n is the hyperparameters for the GP model of $\mathcal{D}_n$, e.g. $(\sigma^2{}_{\mathrm{r},1,n}, l_{\mathrm{r},1,n}, \sigma^2{}_{\mathrm{r},2,n}, l_{\mathrm{r},2,n})$ for the RBF-RBF kernel. Except for γ, each hyperparameter $\theta_{n,k}$, e.g. $l_{\mathrm{r},1}$, is assumed to be from a gamma distribution, $\theta_{n,k} \sim \Gamma(\psi_k, \phi_k)$, and η is the set of parameters of those gamma distributions, $\eta = \bigcup_{k\in K}\{\psi_k, \phi_k\}$. Thus $\theta_n = \bigcup_{k\in K}\{\theta_{n,k}\}$.

The sampling is done by alternatingly sampling η and θ. The tuning set $\{\mathcal{D}_1, \ldots, \mathcal{D}_N\}$ is used to calculate the likelihood of the samples. The details of this sampling process are given in the appendix. The result is H samples of $(\eta^{(h)}, \theta^{(h)})$. The M samples of θ_i for the importance weighting are generated by doing the following M times: select h at random, sample $\theta_{i,k} \sim \Gamma(\psi_k^{(h)}, \phi_k^{(h)})$ for each k. We use M=100 and H=1200 (satellite) or H=2000 (London), with a burn-in of 200 samples.

Data

Two data sets are used for evaluation. The first comes from the TROPOspheric Measuring Instrument (TROPOMI) aboard the Sentinel-5P satellite from the EU's Copernicus

programme (Copernicus 2018). The data set consists of 1083 images of 28x28 pixels, each giving the NO_2 concentration in $\mathrm{mol/m^2}$ within an area of about 7x7 km from the ground to the upper troposphere. The images are from October and November 2018, and have been selected for higher pollution concentrations. Images with more than 10 % missing data were excluded. The NO_2 concentrations vary greatly between images, so different subsets were considered. We use the same division as in Hellan, Lucas, and Goddard (2020). The 'Strong' subset consists of the 50 images with the highest maxima, the 'Median' subset of the 50 images with the median maxima and the 'Weak' subset of the 50 images with the lowest maxima. Tuning sets consist of 10 images adjacent to these. The 'Selection' subset consists of 100 images selected representatively from the whole set, without overlapping the above sets. Fig. 2 shows example images. The median tuning set is used for the selection subset. For this data set, a data snapshot $\mathcal{D}_n$ refers to one image.

The second data set was extracted from the London Air Quality Network (Imperial College London 1993). The full data set consists of years of data across multiple pollutants. For this paper, the NO_2 readings were used, and a tuning set constructed from the 2015 data and a test set from the 2016 data. Examples from the tuning set are shown in Fig. 3. The days with less than 40 readings were discarded, and only the readings from 'Roadside' sensors used. The former was done as days with more data are more suitable to show the benefits of BO. The latter was done as sensors with different classifications behaved differently, so this simplified the modelling. Each day was used as a data snapshot $\mathcal{D}_n$. The tuning set consists of 214 such days and the test set 365 days.

Results

Two metrics are used to evaluate the model performance. Maximum ratio is the ratio between the true value at the estimated maximiser, $\hat{y}$, and the true maximum y^*. Since there are multiple data snapshots, the average is used, $R = \frac{1}{P}\sum_{p=1}^{P} \hat{y}_p/y_p^*$ where P is the number of snapshots evaluated. The ratio metric is ideally one, with a higher score being better (it cannot exceed one). The second metric is the Euclidean distance between the estimated maximiser $\hat{x}$ and the true maximiser x^*. Again, the average is used, $D = \frac{1}{P}\sum_{p=1}^{P} |\hat{x}_p - x_p^*|$. The distance metric is ideally zero, with a higher score being worse. As in Hellan, Lucas, and Goddard (2020), we compare to a random baseline. At each iteration a location is chosen at random from those available. The result is given as the average over 100 runs on each of the data snapshots in each subset. For BO one run is used. The metrics are calculated on the pre-processed data, i.e. after log transform and standardisation / mean-centring.

Satellite data

The results on the satellite data are given in Figs. 4 and 5 and Tables 1 and 2, which also give the results from Hellan, Lucas, and Goddard (2020) for comparison. The performance has improved on both the strong and selection subsets. As can be seen, the previous work did not do better than the random guessing baseline on the selection subset.

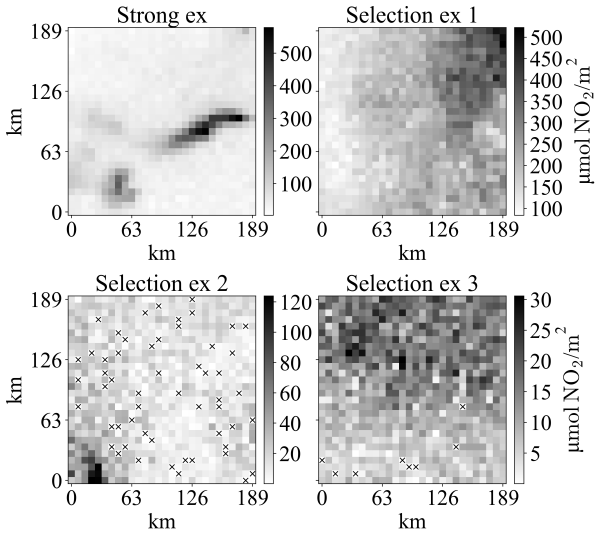


Figure 2: Examples of data snapshots from the satellite data set. Selection examples 1, 2 and 3 are the strongest, median and weakest images from the selection subset, respectively. The strong example is adapted from Hellan, Lucas, and Goddard (2020). Crosses indicate missing data.

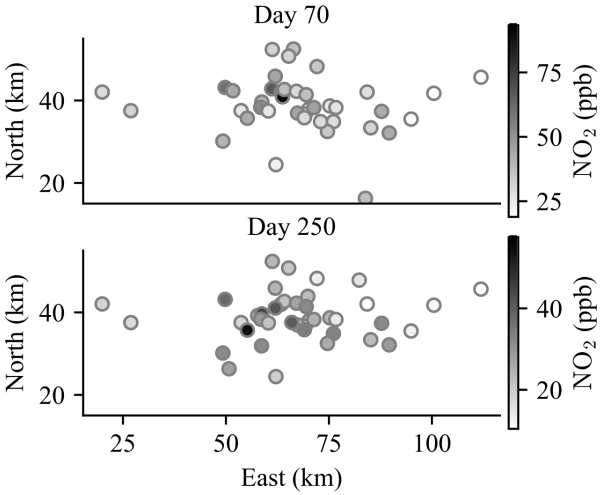


Figure 3: Examples of data from the LAQN data set. The distances are from the most south-westerly monitoring station, in Beech outside Alton in Hampshire. Note that not all sensors are available each day, that the location of the maximum varies and the strong clustering in central London.

London data

The results on the London data are given in Fig. 6 and Table 3. As there are no previously published results for comparison, an additional random baseline is provided – random selection without replacement. As all readings are assumed to be noise-free this leads to more efficient exploration. For the satellite data the difference between the baselines is small due to the larger number of available samples.

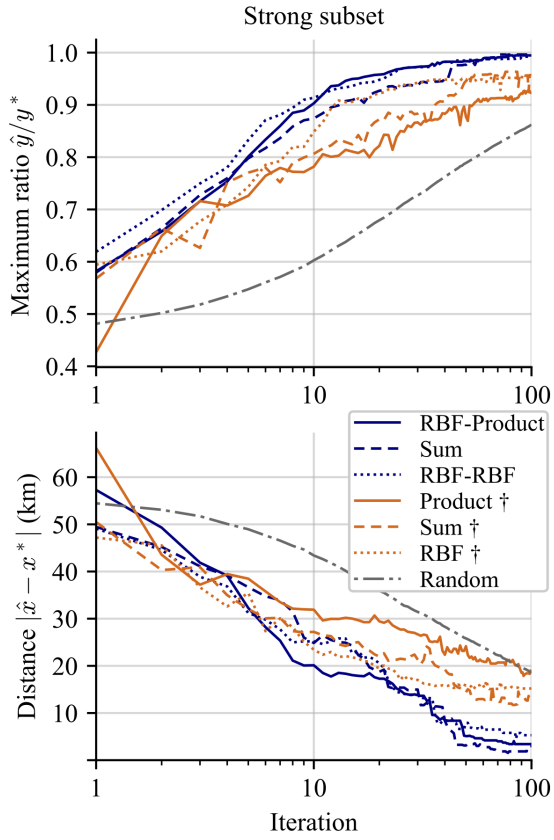


Figure 4: Results on strong subset of satellite data. Our values in navy (darkest) compared to values in Hellan, Lucas, and Goddard (2020) in orange (lightest) marked with †. The baseline 'Random' is shown in grey (medium, different pattern). The results obtained are better than those in existing work, and both out-compete the random baseline. $\hat{x}$ is the estimated maximiser and x^* the true maximiser. $\hat{y}$ and y^* are the true concentration values at $\hat{x}$ and x^*, respectively.

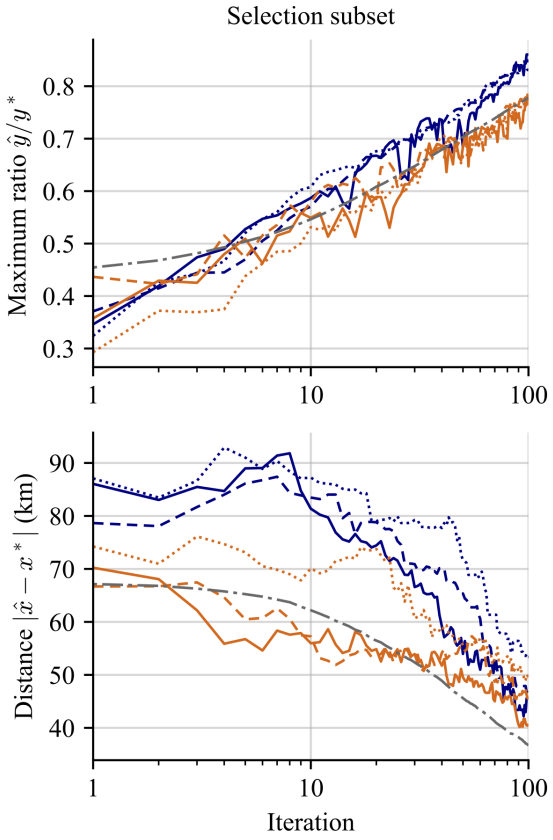


Figure 5: Results on selection subset of satellite data. The legend and variable definitions are given in Fig. 4. Values in orange marked with † are from Hellan, Lucas, and Goddard (2020). The new results are an improvement on those in existing work when considering the ratio metric; the three new results mostly lie on top of each other, and the three old results and the baseline mostly lie on top of each other. The baseline is competitive on the distance metric.

Table 1: Confidence intervals for the means of the maximum ratio at the final iteration for the satellite data. Given are means ± one standard deviation of the mean. The best values are given in **bold**. This confirms the story from Figs. 4 and 5 that improved results are obtained on the strong and selection subsets. † indicates results from Hellan, Lucas, and Goddard (2020).

Strong	RBF-RBF **0.987-0.997**	Sum **0.993-0.999**	RBF-Product **0.990-0.999**
	RBF † 0.940-0.966	Sum † 0.945-0.969	Product † 0.907-0.941
Selection	RBF-RBF **0.797-0.866**	Sum **0.816-0.886**	RBF-Product **0.826-0.894**
	RBF † 0.755-0.804	Sum † 0.756-0.802	Product † 0.750-0.810

Table 2: Confidence intervals for the means of the maximiser distance at the final iteration for the satellite data. Given are means ± one standard deviation of the mean. The best values are given in **bold**. This confirms the story from Figs. 4 and 5 that similar results are obtained on the selection subset and improved results on the strong subset. † indicates results from Hellan, Lucas, and Goddard (2020).

Strong	RBF-RBF **2.381-8.229**	Sum **1.290-4.695**	RBF-Product **0.778-6.040**
(km)	RBF † 11.386-19.010	Sum † 8.445-16.976	Product † 14.507-23.944
Selection	RBF-RBF 44.456-62.301	Sum 36.960-56.178	RBF-Product 36.126-54.977
(km)	RBF † 44.145-55.010	Sum † 40.551-51.918	Product † 35.041-45.519

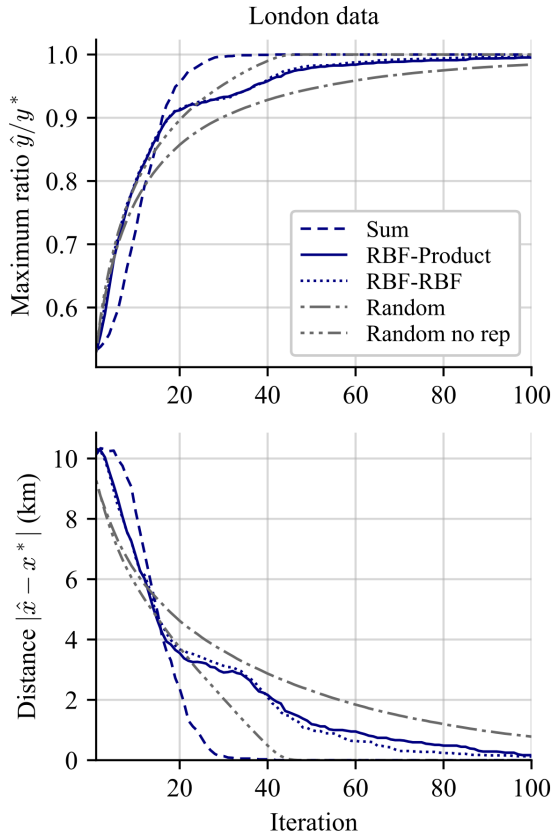


Figure 6: Results on London data. The variable definitions are given in Fig. 4. The RBF-Product and RBF-RBF lines follow each other closely and sometimes overlap. The Sum model is slower to start learning, but then finds the maximum much faster than the other models and the baselines.

Table 3: Confidence intervals for the means of the maximum ratio at the 31st iteration for the London data. Given are means ± one standard deviation of the mean. The best values are given in **bold**.

Maximum ratio	RBF-RBF 0.925-0.938	Sum **0.996-0.999**	RBF-Product 0.927-0.939
	Random 0.904-0.905	Rand. no rep 0.957-0.958	
Distance (km)	RBF-RBF 2.841-3.330	Sum **0.042-0.133**	RBF-Product 2.694-3.148
	Random 3.510-3.562	Rand. no rep 1.855-1.896	

Discussion

The results on the satellite data, Figs. 4 and 5 and Tables 1 and 2, show that our method gives improved results even on the challenging selection subset. The exception is the distance metric on the selection subset. An example of when a good ratio score corresponds to a bad distance score is an image that is zero everywhere except at the bottom right where it is 1.000, and at the top left where it is 0.999. The distance metric will be disproportionately high (bad) if only the marginally worse local optimum is found, but the ratio score will be good regardless. By inspection of Fig. 2 one can see that the selection examples have many local optima, and so would suffer from this. On the London data one of the GP models does much better than either baseline, but the other two perform worse. Why this happens is examined in the Exploration subsection. But it shows that BO can be much faster, even if care must be taken when choosing the models and priors. BO has the added benefit of providing models of the underlying process, with interpretable hyperparameters. An example of this is given in the Interpretability subsection.

Given the relatively small difference in performance between the GP models and the best random baseline one might be tempted to ask whether the extra effort needed for the former is justified. Therefore, it is worth keeping in mind the application. When deployed, each extra iteration will mean one extra sensor having to be procured and installed, requiring both money and labour. In that context, using more resources when planning the placement is preferable. For other domains, it becomes a trade-off between the cost of implementation and the cost of acquiring extra samples. If new samples can be acquired cheaply enough, then the random selection might be preferable.

Throughout this paper, several simplifying assumptions have been made. Firstly, although the London data is temporal, this has been simplified by constructing spatial problems and treating them as independent. Secondly, the measurements are treated as ground-truth readings. This simplifies the error metric calculations, but also makes irrelevant one of the advantages of probabilistically modelling the data, as this allows that uncertainty to be modelled directly. Another limitation is that the results on the satellite data are based on iteratively improving the method, and so could be considered validation results and not test results. The subsets are kept because there are published results to compare to. Because of how the data was split up it is not straightforward to generate a new Strong subset, but a new Selection subset was generated and the method evaluated on it. The results are given in the appendix, and show a strong improvement over the random baseline on both metrics.

Exploration

The results in Fig. 6 can be examined in light of the exploration-exploitation trade-off. An exploration score is used to do so, calculated as the minimum Euclidean distance between the new sample and any one of the already collected samples, $e_{i,j} = \min |x_{i,j} - x_{a,j}|, \quad a < i$. This is done for each iteration i and data snapshot j and the average taken across the data snapshots; see Fig. 7. It shows that the reason the Sum model improves more slowly in the beginning is that it emphasises exploration. Then, in the middle stage (5 to 20 iterations) it is able to learn faster, as it has already narrowed down what area to explore further, and manages to find the maximum faster. Conversely, and by chance, some of the random baselines' early iterations will be checking

neighbouring points of those already seen, which can give small-scale improvement. The two other BO kernels start off with an exploration score between Sum and the first random baseline, and the performance is also between the two.

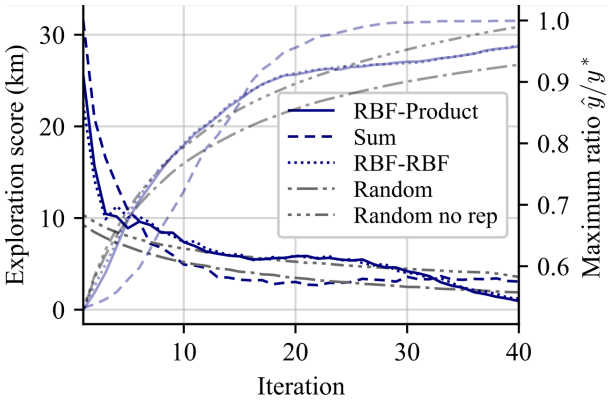


Figure 7: Exploration score of different methods on London data. Overlaid and paler are the ratio scores from Fig. 6. Note that Sum needs longer for the maximum ratio score to increase, and has a higher exploration score at the start.

Interpretability

Part of our rationale for using comparatively simple GP kernels is to facilitate interpretation of hyperparameters, and analysis of the distribution patterns of pollution in an area, e.g. its smoothness and variability in space and time. This lets us perform "at-a-glance" comparisons of different areas, discern whether extrema are more attributable to local sources or long-range patterns, and importantly lets domain experts assess the plausibility of a model's hyperparameters.

To illustrate, we consider the lengthscales $l_{\mathrm{r},i}$ of the RBF-RBF model in Eq. (5) on the London data. For simplicity, we take the mean of the samples generated for the prior, as that shows the trends better than the posterior on a single day. Converting the values to the standard RBF expression gives $l_{\mathrm{r},1} = 2.00$ km, $l_{\mathrm{r},2} = 241$ km, $\sigma^2_{\mathrm{r},1} = 2.05$ and $\sigma^2_{\mathrm{r},2} = 2.04$. This corresponds to one local component and one regional component contributing roughly equally to the overall variability of NO_2 concentrations. At a distance of 100 metres the expected correlation between NO_2 concentrations will be close to 1, driven by both local and long-range effects, whereas at a distance of 10 km it falls to 0.5, and local effects play a negligible role. The correlations are also visualised in Fig. 8. The LAQN provides a pollution map online

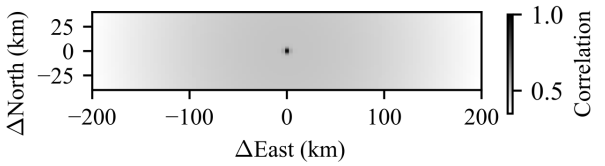


Figure 8: Expected correlation as function of distance for the mean of hyperparameter prior samples for the London data.

at their website (Imperial College London 1993). It shows pollution across all of London, but with more concentrated in central London radiating outwards and around Heathrow airport, as well as along major roads. This supports the modelling hypothesis of a local and a regional component.

Conclusion

We have shown that Bayesian optimisation with hierarchical models can be successfully applied to ground-level urban pollution data. We also presented a pragmatic method for approximate inference when some of the work can be precomputed, which we believe can be useful in other applications. In Bayesian optimisation the GP tuning step needs to be repeated many more times than in standard regression, as new data points are added iteratively. Therefore, using standard MCMC on the full hierarchical model for each new data point would slow down the process. By using importance weighting in the BO tuning step the process can be sped up. This is even more useful in other applications where collecting samples is less time consuming than for pollution.

As expected, the models performed worse on the ground-level data. This is due in part to the data snapshots having fewer available samples. Some exploration is needed before the model can usefully discern the areas of interest, and where the data is sparse compared to the local variations this is harder. The strength of BO lies in being able to exploit structure in data, so it works better for slowly varying processes. While the results show that the approach has promise, more evaluation is needed. In particular, due to data constraints, London was used both for developing the prior and for evaluation. To test whether the prior can be constructed from other cities additional data needs to be used. It would also be beneficial to test the methods as they would be used, i.e. iteratively placing sensors, but this would be much more resource-intensive. There might be easy performance gains available by experimenting with kernel families (e.g. Matérn) and combinations, such as more complex compositional kernels to model correlations at different scales. The benefits of modelling seasonal and weekly patterns should also be explored. Additionally, the results showed the benefit of early exploration, so the methods could be biased towards this by modifying the acquisition function. Berk et al. (2018) present a modification to expected improvement that increases early exploration without needing manual tuning.

There are two main extensions needed before deploying the method: temporal modelling and accommodating sensors with varying precision and reliability. Temporal modelling is important as pollution levels vary temporally, with some days bringing high pollution levels throughout a city. The temporal aspect can be avoided by only considering averages, but this ignores the legal limits on hourly averages (World Health Organization 2006, p. 174-175) and makes adding data from new sensors harder. The uncertainty of a pollution reading is important when using low-cost sensors for monitoring to avoid misleading statements. Smith et al. (2019) present a solution to this, but do not discuss how to place the sensors. Unifying the active placement with the uncertainty treatment is necessary for successfully applying it using a network of low-cost sensors.

Ethical considerations

We believe the risks are very low from our proposed method. The only data used are NO_2 concentrations, and the latitude and longitude of the stationary sensors. No images or personal data of any kinds are used. This is an advantage over other proposed solutions, e.g. using taxis to collect data. The potential benefits far outweigh the harms, by allowing local communities to keep authorities accountable.

Acknowledgements

The authors would like to thank Dr Douglas Finch at the University of Edinburgh for extracting the data and helpful correspondence, and Professor Iain Murray, also of the University of Edinburgh, for helpful advice and discussions. The authors also thank the anonymous reviewers for their time and thoughtful comments.

This work was supported by the EPSRC Centre for Doctoral Training in Data Science, funded by the UK Engineering and Physical Sciences Research Council (grant EP/L016427/1) and the University of Edinburgh.

References

Ainslie, B.; Reuten, C.; Steyn, D. G.; Le, N. D.; and Zidek, J. V. 2009. Application of an entropy-based Bayesian optimization technique to the redesign of an existing monitoring network for single air pollutants. *Journal of Environmental Management*, 90(8): 2715–2729.

Asenov, M.; Rutkauskas, M.; Reid, D.; Subr, K.; and Ramamoorthy, S. 2019. Active Localization of Gas Leaks Using Fluid Simulation. *IEEE Robotics and Automation Letters*, 4(2): 1776–1783.

Berk, J.; Nguyen, V.; Gupta, S.; Rana, S.; and Venkatesh, S. 2018. Exploration Enhanced Expected Improvement for Bayesian Optimization. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Lecture Notes in Computer Science, 621–637. Springer International Publishing. ISBN 978-3-030-10928-8.

Carminati, M.; Ferrari, G.; and Sampietro, M. 2017. Emerging miniaturized technologies for airborne particulate matter pervasive monitoring. *Measurement*, 101: 250–256.

Cats, G.; and Holtslag, A. 1980. Prediction of air pollution frequency distribution—Part I. The lognormal model. *Atmospheric Environment (1967)*, 14(2): 255–258.

Chib, S.; and Greenberg, E. 1995. Understanding the Metropolis - Hastings Algorithm. *The American Statistician*, 49(4): 327–335.

Cocchi, D.; Greco, F.; and Trivisano, C. 2007. Hierarchical space-time modelling of PM10 pollution. *Atmospheric Environment*, 41(3): 532–542.

Cohen, A. J.; Brauer, M.; Burnett, R.; Anderson, H. R.; Frostad, J.; Estep, K.; Balakrishnan, K.; Brunekreef, B.; Dandona, L.; Dandona, R.; Feigin, V.; Freedman, G.; Hubbell, B.; Jobling, A.; Kan, H.; Knibbs, L.; Liu, Y.; Randall, M.; and Forouzanfar, M. H. 2017. Estimates and 25-year trends of the global burden of disease attributable to ambient air pollution: an analysis of data from the Global Burden of Diseases Study 2015. *The Lancet*, 389(10082): 1907–1918.

Copernicus. 2018. Sentinel-5P TROPOMI Level 2 Nitrogen Dioxide total column products. Version 01. Processed by ESA. DOI: 10.5270/S5P-s4ljg54.

Dawkins, L. C.; Williamson, D. B.; Mengersen, K. L.; Morawska, L.; Jayaratne, R.; and Shaddick, G. 2020. Where Is the Clean Air? A Bayesian Decision Framework for Personalised Cyclist Route Selection Using R-INLA. *Bayesian Analysis*, 16(1): 61–91.

De Ath, G.; Everson, R.; and Fieldsend, J. 2021. How Bayesian Should Bayesian Optimisation Be? arXiv:2105.00894.

Hellan, S. P.; Lucas, C. G.; and Goddard, N. H. 2020. Optimising Placement of Pollution Sensors in Windy Environments. Presented at the NeurIPS 2020 Workshop on AI for Earth Sciences, arXiv:2012.10770.

Hoffman, M. D.; and Gelman, A. 2014. The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1): 1593–1623.

Imperial College London, E. R. G. 1993. London Air Quality Network. https://www.londonair.org.uk/LondonAir/Default.aspx. Accessed: 2021-08-31.

Jones, D. R.; Schonlau, M.; and Welch, W. 1998. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, 13(4): 455–492.

Katsouyanni, K. 2003. Ambient air pollution and health. *British Medical Bulletin*, 68(1): 143–156.

Kelly, K. E.; Whitaker, J.; Petty, A.; Widmer, C.; Dybwad, A.; Sleeth, D.; Martin, R.; and Butterfield, A. 2017. Ambient and laboratory evaluation of a low-cost particulate matter sensor. *Environmental Pollution*, 221: 491–500.

Lalchand, V.; and Rasmussen, C. E. 2020. Approximate Inference for Fully Bayesian Gaussian Process Regression. *Symposium on Advances in Approximate Bayesian Inference*, 1–12. PMLR.

Lelieveld, J.; Evans, J. S.; Fnais, M.; Giannadaki, D.; and Pozzer, A. 2015. The contribution of outdoor air pollution sources to premature mortality on a global scale. *Nature*, 525(7569): 367–371. Number: 7569 Publisher: Nature Publishing Group.

Liu, X.; Jayaratne, R.; Thai, P.; Kuhn, T.; Zing, I.; Christensen, B.; Lamont, R.; Dunbabin, M.; Zhu, S.; Gao, J.; Wainwright, D.; Neale, D.; Kan, R.; Kirkwood, J.; and Morawska, L. 2020. Low-cost sensors as an alternative for long-term air quality monitoring. *Environmental Research*, 185: 109438.

Marchant, R.; and Ramos, F. 2012. Bayesian optimisation for Intelligent Environmental Monitoring. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2242–2249. IEEE.

Morere, P.; Marchant, R.; and Ramos, F. 2017. Sequential Bayesian optimization as a POMDP for environment monitoring with UAVs. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 6381–6388. IEEE.

Murray, I.; and Adams, R. P. 2010. Slice sampling covariance hyperparameters of latent Gaussian models. In *Advances in Neural Information Processing Systems*, volume 23.

Rasmussen, C. E.; and Williams, C. K. I. 2006. *Gaussian Processes for Machine Learning*. MIT Press.

Reggente, M.; and Lilienthal, A. J. 2009. Using local wind information for gas distribution mapping in outdoor environments with a mobile robot. In *2009 IEEE SENSORS*, 1715–1720.

Sahu, S. K. 2012. Hierarchical Bayesian Models for Space–Time Air Pollution Data. In Subba Rao, T.; Subba Rao, S.; and Rao, C. R., eds., *Handbook of Statistics*, volume 30 of *Time Series Analysis: Methods and Applications*, 477–495. Elsevier.

Shahriari, B.; Swersky, K.; Wang, Z.; Adams, R. P.; and de Freitas, N. 2015. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proceedings of the IEEE*, 104(1): 148–175.

Shiffrin, R. M.; Lee, M. D.; Kim, W.; and Wagenmakers, E.-J. 2008. A Survey of Model Evaluation Approaches With a Tutorial on Hierarchical Bayesian Methods. *Cognitive Science*, 32(8): 1248–1284.

Singh, A.; Ramos, F.; Whyte, H. D.; and Kaiser, W. J. 2010. Modeling and decision making in spatio-temporal processes for environmental surveillance. In *2010 IEEE International Conference on Robotics and Automation*, 5490–5497.

Smith, M. T.; Ssematimba, J.; Álvarez, M. A.; and Bainomugisha, E. 2019. Machine Learning for a Low-cost Air Pollution Network. Presented at the NeurIPS 2019 Workshop on Machine Learning for the Developing World, arXiv:1911.12868.

Snoek, J.; Larochelle, H.; and Adams, R. P. 2012. Practical Bayesian Optimization of Machine Learning Algorithms. *Advances in Neural Information Processing Systems*, 25: 2951–2959.

Williams, C.; and Rasmussen, C. 1995. Gaussian processes for regression. *Advances in neural information processing systems*, 8.

World Health Organization. 2006. *Air quality guidelines: global update 2005: particulate matter, ozone, nitrogen dioxide, and sulfur dioxide*. Copenhagen, Denmark: World Health Organization. ISBN 978-92-890-2192-0. OCLC: ocn162119780.

Appendix

HLG20 is here used to refer to Hellan, Lucas, and Goddard (2020).

Acquisition function evaluation

At each BO iteration, importance weighting is used to calculate the acquisition function given in Eq. (13). $a_{j,i}(x_*)$ is the acquisition function for test data snapshot $\mathcal{D}_j$ using hyperparameter sample θ_i on test point x_*. We use expected improvement (Jones, Schonlau, and Welch 1998) as the base acquisition function. $f(x)$ is the underlying function and f' the highest sample from previous iterations.

$$a_{j,i}(x_*) = \mathbb{E}[\max(0, f(x_*) - f')|\, \mathcal{D}_j, \theta_i] \tag{12}$$

$$= \int_{f'}^{\infty} (f(x_*) - f')\mathcal{N}(f; \mu_{j,i}(x_*), \mathbb{V}_{j,i}(x_*))\mathrm{d}f \tag{13}$$

The hyperparameter sample θ_i is used to calculate the posterior distribution, expressed by the mean and the variance in Eq. (14) and Eq. (15), respectively. $K_j(\theta_i)$ is the covariance of the observed points from $\mathcal{D}_j$, i.e. the ones sampled at previous iterations, using hyperparameter sample θ_i. Y_j is the values from the observed points. $K_{**}(\theta_i)$ is the covariance function evaluated on $\tau = x_* - x_* = \mathbf{0}$. $K_{*,j}(\theta_i)$ is the covariance between the observed points and the test point. σ_n^2 is the variance of the Gaussian noise in the likelihood.

$$\mu_{j,i}(x_*) = K_{*,j}(\theta_i)^T (K_j(\theta_i) + \sigma_n^2 \mathcal{I})^{-1} Y_j \tag{14}$$

$$\mathbb{V}_{j,i}(x_*) = K_{**}(\theta_i) - K_{*,j}(\theta_i)^T (K_j(\theta_i) + \sigma_n^2 \mathcal{I})^{-1} K_{*,j}(\theta_i) \tag{15}$$

Then, importance weighting is used to scale the acquisition function values from each hyperparameter sample. The weight is given in Eq. (16) and the final expression in Eq. (17). $W = \sum_{i=1}^{M} p(\mathcal{D}_j|\theta_i)$ as defined in the main paper.

$$w_{j,i} = p(\mathcal{D}_j|\theta_i) = \mathcal{N}(\mathcal{D}_j; \mathbf{0}, K_j(\theta_i)) \tag{16}$$

$$a_j(x_*) \approx W^{-1} \sum_i w_{j,i} a_{j,i}(x_*) \tag{17}$$

Details of MCMC sampling

The sampling is done using MCMC to collect joint samples of (η, θ), where θ is the GP hyperparameters and η the distribution relating θ for different problems, see Fig. 1. This is done using Metropolis-Hastings (Chib and Greenberg 1995) updating one variable ψ_k, ϕ_k or $\theta_{n,k}$ at a time. The sampling is done according to Algorithm 1.

The variables in Algorithm 1 are defined as

- H: Number of MCMC samples
- N: Number of tuning data snapshots
- K: Number of GP hyperparameters
- K': Number of GP hyperparameters with non-constant η
- B: Number of times to sample η per θ sample

The η parameters for the directional hyperparameter γ and the noise hyperparameter are constant.

Algorithm 1: Sampling (η, θ)

```
Initialise η^(0), θ^(0)
for h = 1 ... H do
  for n = 1 ... N do
    for k = 1 ... K do
      θ^(h)_{n,k/} ⇐ {θ^(h)_{n,1}, ..., θ^(h)_{n,k-1}, θ^(h-1)_{n,k+1}, ..., θ^(h-1)_{n,K}}
      Sample θ^(h)_{n,k} given η^(h-1), θ^(h)_{n,k/}
    end for
  end for
  for _ = 1... B do
    for k = 1 ... K' do
      for η^(h)_{k,g} = ψ^(h)_k, φ^(h)_k do
        η^(h)_{k,g/} ⇐ ψ^(h)_k if η^(h)_{k,g} ≠ ψ^(h)_k else φ^(h-1)_k
        Sample η^(h)_{k,g} given θ^(h), η^(h)_{k,g/}
      end for
    end for
  end for
  Store sample (η^(h), θ^(h))
end for
return H samples of (η, θ)
```

Initialisation $\eta^{(0)}$ is set to a vector of ones. $\theta^{(0)}$ is set by sampling from $\eta^{(0)}$.

Sample $\theta_{n,k}^{(h)}$ given $\eta^{(h-1)}, \theta_{n,\not k}^{(h)}$ The sampling of $\theta_{n,k}^{(h)}$ given $\eta^{(h-1)}$ and $\theta_{n,\not k}^{(h)}$ is done as follows. At each iteration a proposal $\theta'_{n,k}$ is generated, and either accepted or rejected proportionally to the acceptance ratio $a_\theta(\theta'_{n,k}|\theta_{n,k}^{(h-1)})$ where $\theta_{n,k}^{(h-1)}$ is the previous sample value. If the proposal is rejected the old value is used for the new sample.

$\theta_{n,k}^{(h)}$ is the k-th GP hyperparameter fitted to data snapshot $\mathcal{D}_n$ at iteration h. The un-normalised probability function $\pi(\theta_{n,k}^{(h-1)})$ is given in Eq. (18). The proposal distribution is given in Eq. (19) and the acceptance ratio in Eq. (20). $\pi(\theta'_{n,k})$ is evaluated by replacing $\theta_{n,k}^{(h-1)}$ in Eq. (18) with $\theta'_{n,k}$.

$$\pi(\theta_{n,k}^{(h-1)}) = p(\mathcal{D}_n|\theta_{n,k}^{(h-1)}, \theta_{n,\not k}^{(h)}) p(\theta_{n,k}^{(h-1)}|\eta^{(h-1)}) \tag{18}$$

$$Q_\theta(\theta_{n,k}^{(h-1)}|\theta'_{n,k}) = p(\theta_{n,k}^{(h-1)}|\eta^{(h-1)}) \tag{19}$$

$$a_\theta(\theta'_{n,k}|\theta_{n,k}^{(h-1)}) = \min\left(1, \frac{p(\mathcal{D}_n|\theta'_{n,k}, \theta_{n,\not k}^{(h)})}{p(\mathcal{D}_n|\theta_{n,k}^{(h-1)}, \theta_{n,\not k}^{(h)})}\right) \tag{20}$$

$p(\mathcal{D}_n|\theta_{n,k}^{(h-1)}, \theta_{n,\not k}^{(h)})$ is evaluated through the standard GP expressions. $p(\theta_{n,k}^{(h-1)}|\eta^{(h-1)}) = \Gamma(\theta_{n,k}^{(h-1)}; \psi_k^{(h-1)}, \phi_k^{(h-1)})$ for k corresponding to $\{\sigma_\mathrm{r}^2, l_\mathrm{r}, \sigma_\mathrm{w}^2, l_\mathrm{w}\}$ and $p(\theta_{n,k}^{(h-1)}|\eta^{(h-1)}) = \mathrm{Uniform}(\theta_{n,k}^{(h-1)}; 0, \pi)$ for k corresponding to γ.

Sample $\eta_{k,g}^{(h)}$ given $\theta^{(h)}, \eta_{k,\not g}^{(h)}$ The sampling of $\eta_{k,g}^{(h)}$ given $\theta^{(h)}$ and $\eta_{k,\not g}^{(h)}$ is done as follows. At each iteration a proposal

$\eta'_{k,g}$ is generated, and either accepted or rejected proportionally to the acceptance ratio $a_\eta(\eta'_{k,g}|\eta^{(h-1)}_{k,g})$ where $\eta^{(h-1)}_{k,g}$ is the previous sample value. If the proposal is rejected the old value is used for the new sample.

The un-normalised probability function $\pi(\eta^{(h-1)}_{k,g})$ is given in Eq. (21). The proposal distribution is given in Eq. (22) and the acceptance ratio in Eq. (23). $\psi^{(h)}_k$ is the shape parameter of the gamma distribution, and $\phi^{(h)}_k$ the scale parameter. The prior on η, $p(\eta^{(h-1)}_{k,g})$, is set to one for valid values and zero otherwise.

$$\pi(\eta^{(h-1)}_{k,g}) = p(\eta^{(h-1)}_{k,g}) \prod_{n=1}^{N} p(\theta^{(h)}_{n,k}|\eta^{(h-1)}_{k,g}, \eta^{(h)}_{k,g}) \quad (21)$$

$$Q_\eta(\eta^{(h-1)}_{k,g}|\eta'_{k,g}) = \mathcal{N}(\eta^{(h-1)}_{k,g}; \eta'_{k,g}, \sigma^2_{\eta,k,g}) \quad (22)$$

$$a_\eta(\eta'_{k,g}|\eta^{(h-1)}_{k,g}) = \min\left(1, \frac{\pi(\eta'_{k,g})}{\pi(\eta^{(h-1)}_{k,g})}\right) \quad (23)$$

$\sigma_{\eta,k,g}$ is a parameter determining the width of the proposal. The values are

- $\sigma_{\eta,k,g}$=1.5 for $\eta^{(h-1)}_{k,g}$=$\psi^{(h)}_k$, k corresponding to l_r or l_w
- $\sigma_{\eta,k,g}$=0.5 for $\eta^{(h-1)}_{k,g}$=$\phi^{(h)}_k$, k corresponding to l_r or l_w
- $\sigma_{\eta,k,g}$=0.3 for $\eta^{(h-1)}_{k,g}$=$\psi^{(h)}_k$, k corresponding to σ^2_r or σ^2_w
- $\sigma_{\eta,k,g}$=0.1 for $\eta^{(h-1)}_{k,g}$=$\phi^{(h)}_k$, k corresponding to σ^2_r or σ^2_w

Additional results

Additional results were acquired on a novel selection test subset, given in Fig. 9 and Table 4. The subset was created in the same way as the other selection subset, but without overlap. It provides additional evidence that BO is able to effectively locate the maximum.

Table 4: Confidence intervals for the means of the maximum ratio and distance metrics at the final iteration for the selection test subset. We show means ± one standard deviation of the mean. The best values are shown in **bold**.

Maximum ratio	RBF-RBF **0.889-0.933**	Sum **0.929-0.963**	RBF-Product **0.932-0.957**
	Random 0.802-0.805		
Distance (km)	RBF-RBF **17.409-26.640**	Sum **11.125-19.026**	RBF-Product **11.604-17.465**
	Random 32.622-33.423		

Comparison to earlier model

Fig. 10 shows the changes from the approach used in HLG20. The main differences are:

i) Evaluation on more realistic data. In HLG20 only satellite data was used, while we here also use ground-level data.

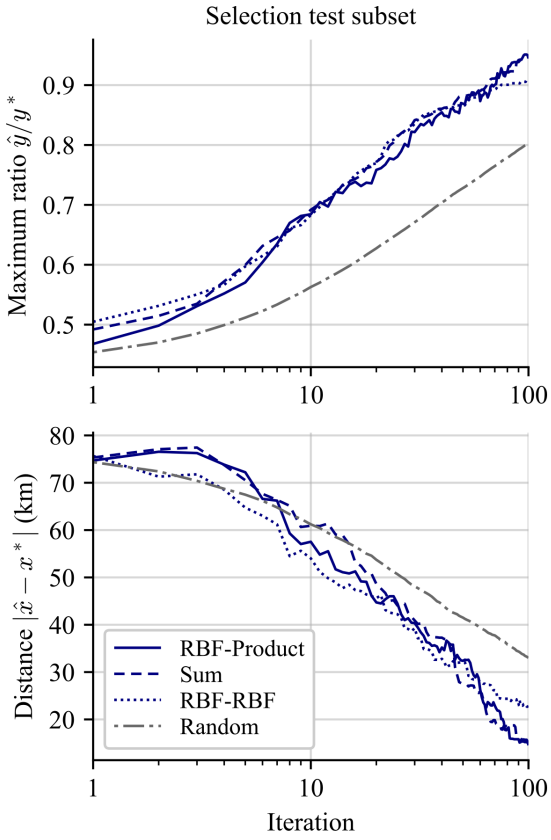


Figure 9: Results on selection test subset of satellite data. $\hat{x}$ is the estimated maximiser and x^* the true maximiser. $\hat{y}$ and y^* are the true concentration values at $\hat{x}$ and x^*, respectively.

ii) Noise modelling. In HLG20, the noise hyperparameter was optimised, like the other hyperparameters. In the new model, the noise hyperparameter was instead clamped to a very low value.

iii) Hierarchical modelling of data with MCMC for inference. HLG20 only used basic priors based on the maximum likelihood estimates (MLEs) on the tuning data. Instead, we here use a hierarchical model, obtaining samples of the hyperparameters, θ, and of the parameters defining the distribution of the hyperparameters, η.

iv) Importance weighting instead of gradient-based optimisation. In HLG20, gradient-based optimisation is used to find the MLE on the test data at each iteration in the BO loop. Instead, we use importance weighting to fit our set of hyperparameter priors to the observed data at test time.

v) Expected Improvement (EI) instead of Upper Confidence Bound (UCB). In HLG20, UCB was used as the acquisition function, but we found EI to perform better.

Reproducibility

- **Data sets**: The London data set is freely available online (Imperial College London 1993). The satellite data set is based on data freely available online at https://scihub.

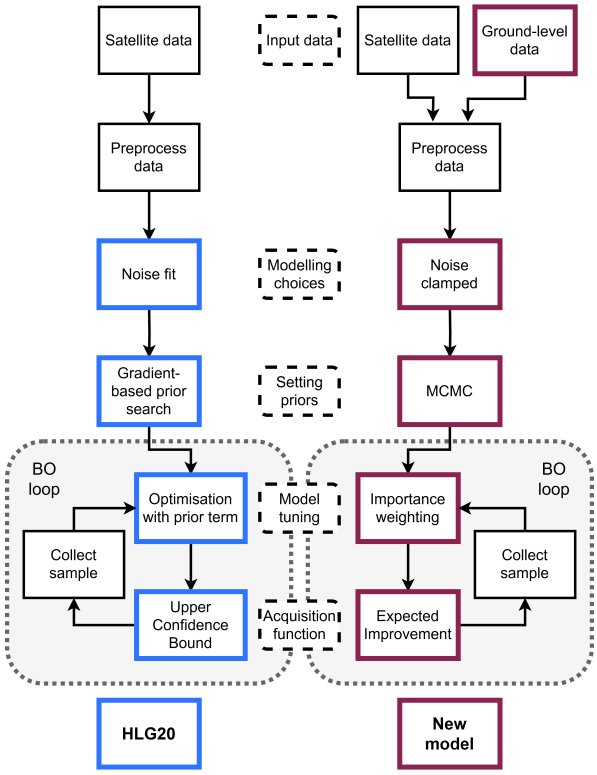


Figure 10: Diagram of differences between model used in HLG20 (left) and our new model (right). The coloured boxes show where changes have been made. Only one of the satellite and ground-level data are used at the time, the figure is meant to show that the new work is evaluated on more data.

copernicus.eu/, but not the filtered version used in this work and the previous work compared to.

- **Seeds**: The random number generator used for MCMC is seeded with the number 13, as is also specified in the code. There is little randomness in the BO loop, since the noise is set very low.
- **(Hyper-)parameter values**: The hyperparameter values needed to run the code are given, e.g. number of burn-in samples, number of BO iterations. The values for the model hyperparameters, e.g. the kernel lenghtscales for each model, are not given as they are outputs of the method and large in number.
- **Hyperparameter selection**: We do not have a full list of the hyperparameter settings tried. However, a general description of how they were set is given below. An exhaustive hyperparameter search to maximise performance was not conducted. The goal of the work was instead to evaluate the suitability of the method.
  - **MCMC hyperparameters**: Different settings were tried for the number of samples and the proposal distribution. The number of samples was first tried at 100 or 200, but this was found to be too low for mixing. Instead, 1200 and 2000 were used for the satellite data and the London data, respectively. This was found to be a good balance between the required computation time and the usefulness of the samples. The $\sigma_{\eta,k,g}$ values were experimented with to give good mixing. We first tried higher values but then reduced them to get a higher acceptance ratio. We found that higher values for the shape parameter ψ than the scale parameter ϕ worked better. We initially tried updating all of θ jointly, and then all of η jointly, but this led to much slower mixing.
  - **Noise in likelihood**: We initially tried having the noise hyperparameter be learned, but found that this did not perform well. The models set the hyperparameter much higher, which led to a disconnect with the evaluation metrics as they assume the observed values to be ground truth.
  - **GP models**: An exhaustive search of models was not done. We first tried the same models as in HLG20, (RBF, Sum and Product), but found that two kernels, one to model the slowly varying component and one for the faster varying component, worked better.
  - **Acquisition functions**: We also tried probability of improvement and upper confidence bound, the latter setting the standard deviation scaling to 1, 4 and 10. We found that expected improvement worked best.

Execution times

Table 5: Run times for maxima location (BO) and hyperparameter samples generation (MCMC).

	Kernel	Strong	Selection	London
MCMC	RBF-RBF	38 min	38 min	21 h
	Sum	56 min	56 min	30 h
	RBF-Product	71 min	78 min	41 h
BO	RBF-RBF	44 min	44 min	2 min
	Sum	50 min	52 min	2 min
	RBF-Product	49 min	49 min	2 min

The execution times are given in Table 5. For MCMC, it is the time needed to collect the joint (θ, η) samples. For BO, it is the average over data snapshots of the time needed for 100 iterations. For the London data, the MCMC takes between 21 and 41 hours, and the BO loops 2 minutes. For the satellite data, the MCMC takes between 38 and 78 minutes and the BO loops 44-52 minutes.

BO is slower for the satellite data due to there being more candidate sampling locations. MCMC is slower for the London data due to there being more tuning data. The MCMC time is less critical as it can be precomputed, while the BO time is encountered while placing sensors. The BO time is for 100 iterations, so even the slowest time, for the Sum kernel on the Selection subset – 52 minutes – averages to 31 seconds per iteration. This is negligible compared to the time required to seek out a location and install a sensor. If necessary, the BO could be sped up by using posterior estimates for the mean and variance of the pollution level, instead of

posterior estimates of the acquisition function. The BO complexity is linear in the number of hyperparameter samples from the prior.

Computing infrastructure

The experiments were run in parallel on nodes on a computer cluster running Scientific Linux 7.9 (Nitrogen). Each node had 14GB in memory and 8 CPUs. The CPU types used were:

- Xeon CPU E5-2640 v3 @ 2.60 GHz
- Xeon CPU E5-2620 v3 @ 2.40 GHz
- Xeon CPU E5-2620 v4 @ 2.10 GHz

The code was run using Python 3.7.3. The complete Python dependencies are given in the code, but the main ones are NumPy (1.16.3) and SciPy (1.2.1).

Data preprocessing

The satellite data subsets were standardised using the means and standard deviations from the tuning sets. The LAQN data snapshots were mean-centred using data from that snapshot. Both data sets were first log-transformed.

A.3 Bayesian Optimisation Against Climate Change: Applications and Benchmarks

See Section 3.4 (Hellan, Lucas, & Goddard, 2023a).

A.4 Obeying the Order: Introducing Ordered Transfer Hyperparameter Optimisation

See Section 6.5 (Hellan, Shen, et al., 2023).

A.5 Data-driven Prior Learning for Bayesian Optimisation

Citation: Hellan, Lucas, and Goddard (2023b).

Data-driven Prior Learning for Bayesian Optimisation

Sigrid Passano Hellan

Christopher G. Lucas

Nigel H. Goddard

School of Informatics
University of Edinburgh, UK

arXiv:2311.14653v2 [cs.LG] 19 Apr 2024

Abstract

Transfer learning for Bayesian optimisation has generally assumed a strong similarity between optimisation tasks, with at least a subset having similar optimal inputs. This assumption can reduce computational costs, but it is violated in a wide range of optimisation problems where transfer learning may nonetheless be useful. We replace this assumption with a weaker one only requiring the shape of the optimisation landscape to be similar, and analyse the recent method *Prior Learning for Bayesian Optimisation* — PLeBO — in this setting. By learning priors for the hyperparameters of the Gaussian process surrogate model we can better approximate the underlying function, especially for few function evaluations. We validate the learned priors and compare to a breadth of transfer learning approaches, using synthetic data and a recent air pollution optimisation problem as benchmarks. We show that PLeBO and prior transfer find good inputs in fewer evaluations.

Keywords: Bayesian optimisation, Transfer learning, Priors, Markov chain Monte Carlo

1. Introduction

Our goal in Bayesian optimisation (BO) is to find high-performing input values using few function evaluations. A powerful approach is to transfer knowledge from previous optimisation tasks. For example, when optimising many similar problems we can choose to disregard input ranges that consistently yield suboptimal outputs. Within the BO framework there are many ways of transferring knowledge (Bai et al., 2023), for instance through a shared model (Swersky et al., 2013), through the initial input values which we evaluate (Feurer et al., 2015), or through limiting the search space (Perrone et al., 2019). In this paper we develop the idea of prior transfer further, and introduce an extension to a recent prior transfer method (Hellan et al., 2022), which we call Prior Learning for BO (PLeBO).

Direct transfer: Most transfer learning for Bayesian optimisation rests on the assumption that inputs that are optimal for (some of) the tuning tasks will also be near-optimal for the new test task. For instance, a common technique is to evaluate a few good input values from previous tasks on a new task before doing Bayesian optimisation.

Prior transfer: An alternative to direct transfer is to assume only that the optimisation tasks have similar response surfaces, i.e. the shapes of their optimisation landscapes. A simple example is to assume that all optimisation tasks can be modelled using a single shared set of Gaussian process (GP) hyperparameter (HP) values, as is done in Wang et al.

(2022). Another approach is to learn a feature transformation which is applied to the input to the GP before the covariance kernel. This is done in Few-Shot BO, an example of deep kernel learning where a neural network is used for this feature transformation (Wistuba and Grabocka, 2021). Hand-crafted HP priors can also be seen as prior transfer: with PLeBO we automate this process. Müller et al. (2023) also learn HP priors from related data, but pick the best from a randomly sampled set instead of learning a posterior distribution.

Fig. 1 illustrates the difference between direct and prior transfer in two idealised settings. In the top row the optima for the tuning and test tasks coincide, and only direct transfer is able to perform useful transfer. In the bottom row the optima are far apart but the shapes are the same. In this setting prior transfer works perfectly while direct transfer does not work at all. For direct transfer we use a strategy where we evaluate the best point from the tuning data. We illustrate prior transfer with a method where we assume the shapes are identical — in practice we just assume the GP hyperparameters are similar.

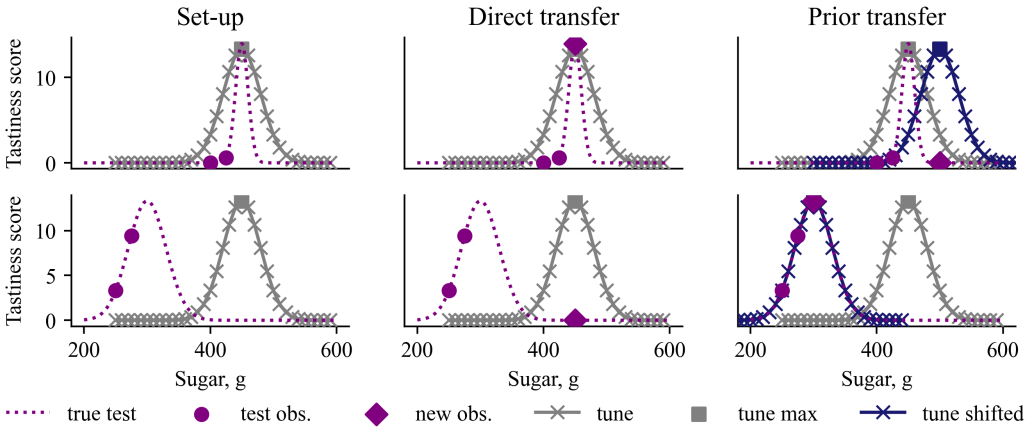


Figure 1: Comparison of direct and prior transfer. In the top row the past and new tasks share optima but have different shapes. In the bottom row they instead have the same shape. Direct transfer works well for shared optima, prior transfer for shared shapes.

Most work on Bayesian optimisation and transfer learning has been applied to hyperparameter optimisation for machine learning (HPO), e.g. Feurer et al. (2015); Perrone et al. (2019); Wistuba and Grabocka (2021). Extending the range of applications motivates the development of new methods, as assumptions that make sense for one application do not necessarily make sense for a different application. For HPO we can expect similar learning rates to work well for the same model trained on different data sets. But in the air pollution context we do not expect the same coordinates in different cities to be the most polluted — e.g. three kilometres north of the city centre. Therefore, we expect prior transfer to work well in situations unsuited to direct transfer, such as air pollution monitoring.

An earlier version of PLeBO was presented as part of Hellan et al. (2022). Here we improve on the method and compare it to other transfer learning approaches. We also provide code for a new and simpler implementation, and analyse the quality of the resulting priors. We show the advantage of prior transfer over direct transfer on both synthetic and real-world data, the latter being air pollution measurements.

2. Bayesian optimisation

Bayesian optimisation (Garnett, 2023) is a blackbox optimisation technique, which utilises a surrogate model, typically a Gaussian process, to model the underlying optimisation task and come up with informed choices of input values. It is particularly suited to problems with a complex unknown structure, where we do not know the gradients, and which are expensive to evaluate. Gaussian processes (Rasmussen and Williams, 2006) are probabilistic machine learning models which are popular in the regime of small data. They are defined by their choice of mean and covariance functions. The former is often set to zero (De Ath et al., 2020), while the latter comes with hyperparameters that need to be set. For instance, the RBF kernel $k_{RBF}(\boldsymbol{\tau}) = \sigma_r^2 \exp\left(-\frac{\boldsymbol{\tau}^T\boldsymbol{\tau}}{2l^2}\right)$ has two hyperparameters: the lengthscale l and the signal variance σ_r^2. The former defines how quickly the function changes, the latter how large its absolute values are. We also need an acquisition function, which uses the surrogate model to determine the next sampling location. We use expected improvement (EI, Jones et al. (1998)) unless otherwise noted.

The problems we want to optimise in Bayesian optimisation are of the form $\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}} f(\boldsymbol{x})$. To fit the hyperparameters of the Gaussian process we use the log marginal likelihood. A simple approach is to use gradient-based optimisation, possibly including a prior term on the hyperparameters. Alternatively, we can approximate the posterior distribution of the hyperparameters, for instance using Markov chain Monte Carlo (MCMC) (Lalchand and Rasmussen, 2020). Then we take the uncertainty of our hyperparameters into account in our model. With transfer learning we assume access to historical evaluations of other tasks, changing the problem to $\boldsymbol{x}_j^* = \arg\max_{\boldsymbol{x}} f_j(\boldsymbol{x}) \mid \{\{\boldsymbol{x}_{n,i}, y_{n,i}\}_{i=1}^{N_n}\}_{n=1}^{N}$ where n indexes tasks, i indexes samples and N_n is the number of sample evaluations for task n.

3. PLeBO

The PLeBO method is an example of prior transfer, intended to work even with a small amount of tuning data. We assume a hierarchical structure where each optimisation task has its own set of GP hyperparameters $\boldsymbol{\theta}$, and these hyperparameters come from a shared distribution $\boldsymbol{\eta}$. The method has two parts: in the preprocessing step, we use MCMC to learn $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$ given observations $\boldsymbol{\mathcal{D}}$, and then sample new values of $\boldsymbol{\theta}$ based on $\boldsymbol{\eta}$ to generate a set of candidate hyperparameters. In the optimisation phase we use importance weighting to fit these candidates to the observed test data $\boldsymbol{\mathcal{D}}_j$. This structure is illustrated in Fig. 2.

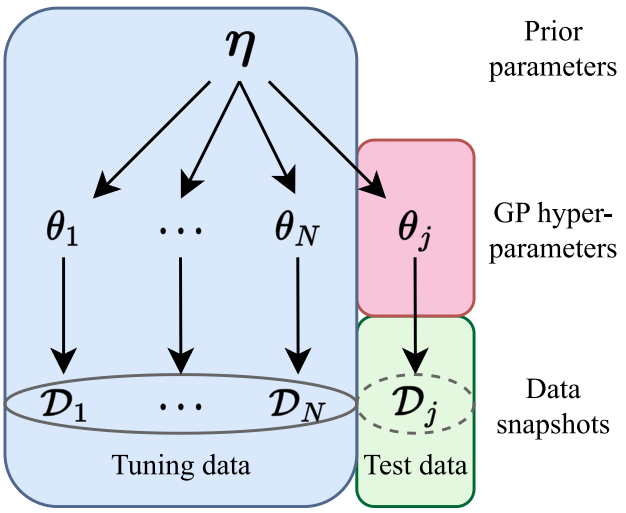


Figure 2: Hierarchical structure and inference in PLeBO.

Formally, the structure is $p(\boldsymbol{\eta}, \boldsymbol{\theta}, \boldsymbol{\mathcal{D}}) = p(\boldsymbol{\eta})p(\boldsymbol{\theta}|\boldsymbol{\eta})p(\boldsymbol{\mathcal{D}}|\boldsymbol{\theta}) = p(\boldsymbol{\eta}) \prod_n p(\boldsymbol{\theta}_n|\boldsymbol{\eta})p(\boldsymbol{\mathcal{D}}_n|\boldsymbol{\theta}_n)$. The modelling distributions can be chosen separately for each hyperparameter, we use gamma distributions $p(\boldsymbol{\theta}|\boldsymbol{\eta}) = \Gamma(\boldsymbol{\theta}; \boldsymbol{\eta})$. That means we need to learn N sets of $\boldsymbol{\theta}_n$ and $2k$ parameters of $\boldsymbol{\eta}$, where k is the dimensionality of $\boldsymbol{\theta}_n$ and N is the number of tuning tasks. For simplicity, we restrict ourselves to 2 HPs in $\boldsymbol{\theta}_n$, so learn 4 parameters of $\boldsymbol{\eta}$. We also need to choose the number H of candidate samples; we use 200 to get a good approximation to taking the expectation given $\boldsymbol{\eta}$, but future work should explore reducing H.

Preprocessing: We learn distributions for the surrogate model hyperparameters, using Markov chain Monte Carlo to collect samples from the distributions of $\boldsymbol{\eta}$ and $\boldsymbol{\theta}$. We replace Hellan et al.'s (2022) custom-built implementation with a new one (`https://github.com/sighellan/plebo`) based on NumPyro (Phan et al., 2019) and using the NUTS (Hoffman and Gelman, 2014) sampler. It is more stable and makes modelling changes, e.g. changing $p(\boldsymbol{\theta}|\boldsymbol{\eta})$, much simpler. As the sampling chains do not always find samples with non-zero likelihood we run multiple chains in parallel before a simple filtering postprocessing step.

Optimising: We use the candidate samples within the optimisation loop to calculate the acquisition function, $a_i(\boldsymbol{x}) \approx \frac{1}{W} \sum_{h=1}^{H} w_h \, a_i^{\theta}(\boldsymbol{x}, \boldsymbol{\theta}_h^{\text{cand}})$ where $w_h = p(\boldsymbol{\mathcal{D}}_j^{:i}|\boldsymbol{\theta}_h^{\text{cand}})$, $W = \sum_{h=1}^{H} w_h$ and $a_i^{\theta}(\cdot)$ is the base acquisition function, for which we use EI. At each optimisation step we calculate the likelihood of every hyperparameter candidate, and use it to weight the corresponding base acquisition function. More details are in Appendix A.2.

4. Experiments

We consider both a synthetic benchmark and real-world air pollution data (Copernicus, 2018). To simplify analysis we use the RBF kernel and limit ourselves to two hyperparameters: the lengthscale and the signal variance. Our base metric is the best observed value at each iteration i, normalised by the max value of the task, $y_{\max}^{(j)}$: $r_i^{(j)} = \max \boldsymbol{Y}_{:N_{\text{start}}+i}^{(j)} / y_{\max}^{(j)}$ where $\boldsymbol{Y}_{:N_{\text{start}}+i}^{(j)}$ are the observations and N_{start} is the number of starting evaluations.

We compare PLeBO to a collection of baselines with no, direct and prior transfer:

- No transfer
  - **RandomSearch:** Randomly select the inputs at each iteration.
  - **BoTorch:** Default settings from the BoTorch library (Balandat et al., 2020).
  - **EI:** BO without transfer using expected improvement (Jones et al., 1998).
  - **UCB:** BO without transfer using upper confidence bound (Srinivas et al., 2012).
- Direct transfer
  - **DirectTrans:** Shared GP for past and new tasks. Cap at 100 past evaluations.
  - **Initial:** First evaluate the best point from each previous task, then do **EI**.
- Prior transfer
  - **PLeBO:** The new prior learning method detailed in Section 3.
  - **TruePLeBO:** Using EI with the true hyperparameters (only for synthetic data).
  - **Gamma:** Use the mean prior $\boldsymbol{\eta}$ from PLeBO when optimising the HPs.
  - **Shared:** A single set of HPs learned from the tuning tasks (Wang et al., 2022).

The **synthetic benchmark** allows us to evaluate the extracted priors. We use the same hierarchical model as in PLeBO, see Fig. 2. We use GPs with RBF kernels and a

known low level of noise. We define gamma distributions on the two hyperparameters: the lengthscale $l \sim \Gamma(5, 0.01)$, and the signal variance $\sigma_r^2 \sim \Gamma(2, 2)$. We use the shape, scale parameterisation of $\Gamma(\cdot)$. We generate a set of 10 tuning optimisation tasks and 100 test tasks. The tuning tasks have 20 evaluations each, and we start the test tasks with ten known evaluations. The tasks are discretised to match the other benchmarks. See Fig. 3.

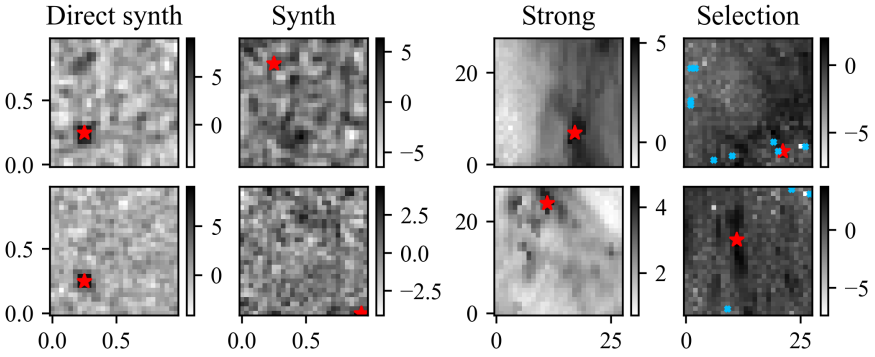


Figure 3: Example optimisation tasks. The left column shows the standard assumption that optima are near each other. Stars indicate optima and blue crosses missing values.

We also use two **air pollution benchmarks**: the satellite optimisation tasks used in Hellan et al. (2020). Each task comes from a snapshot of NO_2 air pollution levels taken from the Sentinel-5P satellite of the Copernicus programme (Copernicus, 2018). The two benchmarks come from grouping the snapshots based on the maximum pollution present. The Strong benchmark consists of 50 test tasks and 10 tuning tasks of similar problems. The Selection benchmark consists of 100 test tasks and 10 tuning tasks with maximal pollution levels at the middle of those of the test tasks. It therefore checks whether prior transfer still works if the task distributions change between the tuning and test data. The data has been preprocessed by log transform and standardisation, see Hellan et al. (2020).

5. Results and Discussion

Fig. 4 compares the performance of PLeBO to the other methods. We see that the transfer methods generally find good input values in fewer iterations. EI only does slightly worse than PLeBO. This highlights the difficulty of prior transfer, as we still need to explore the search space. We also see that prior transfer requires fewer iterations than direct transfer. The differences are smaller among the prior transfer methods. PLeBO is better than Shared and Gamma on the synthetic and Strong benchmarks, but for the Selection benchmark Shared works best. That TruePLeBO works best on the synthetic data supports our intuition that the optimisation can be sped up by better surrogate model HP estimates.

To validate PLeBO we also analyse the inferred priors. In Appendix A.4 we compare the inferred and true values for $\boldsymbol{\eta}$ and $\boldsymbol{\theta}$. While the true and estimated priors are not identical they are similar, and an exact reconstruction could not be expected from only 10 tuning tasks. We also analysed the fits on the individual tasks: the true and inferred values had very similar likelihoods, and sometimes the inferred values had higher likelihood.

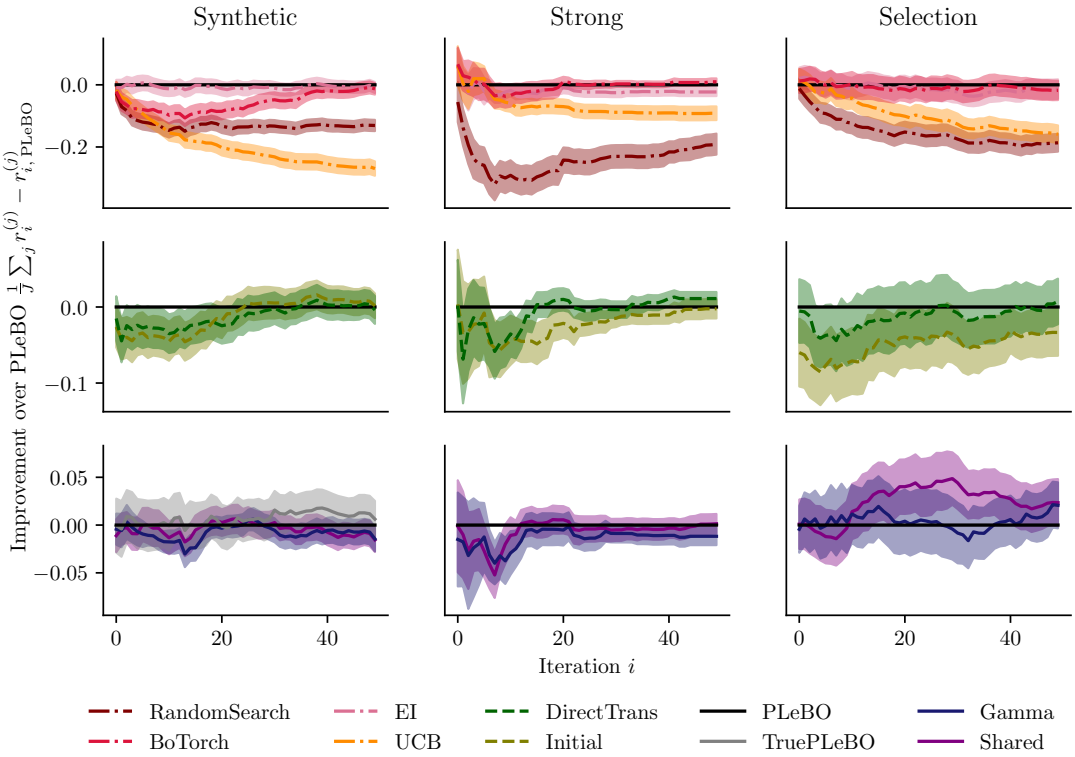


Figure 4: Performance compared to PLeBO at same iteration (above zero is improvement), mean ± one standard error. J is the number of test tasks. The top row compares PLeBO to no transfer, the middle to direct transfer and the bottom row to prior transfer methods.

PLeBO is computationally more expensive than the baselines. Its preprocessing step has a variable runtime due to the NUTS sampler. The runtime was 9 minutes for the synthetic, and 2.6–3.4 hours for the air pollution benchmarks on a standard desktop. The preprocessing is only done once, so a higher computational expense can be tolerated. In the optimisation step we replace HP optimisation with calculating the acquisition function for each of H candidate HP sets. This scales as $\mathrm{O}(H(i + N_{\text{start}})^3)$. On average, PLeBO takes about 6.7 seconds per optimisation step, compared to 0.13 seconds for EI. For very expensive optimisation tasks this is still only a fraction of the total cost, e.g. when installing an air pollution sensor. Runtimes for all methods are given in Appendix A.3. PLeBO's runtime can be adjusted through H based on the available budget and task evaluation speed.

We have presented an improved prior transfer method, PLeBO, and evaluated it on a real-world benchmark for air pollution monitoring. We showed that the generated priors align with the tuning data, and can be exploited on new tasks. The approach is modular, and is not tied to a specific acquisition function. While more general, prior transfer is also weaker than direct transfer as we do not learn about the optima directly: we showed some improvement over EI, but not a massive difference. Future work should attempt to exploit the correct HP choices more, and evaluate PLeBO on a wider range of benchmarks.

Acknowledgments

We thank Tiffany Vlaar and Linus Ericsson for comments on the paper, and G-Research for generous financial support towards conference attendance.

This work was supported by the EPSRC Centre for Doctoral Training in Data Science, funded by the UK Engineering and Physical Sciences Research Council (grant EP/L016427/1) and the University of Edinburgh.

References

Tianyi Bai, Yang Li, Yu Shen, Xinyi Zhang, Wentao Zhang, and Bin Cui. Transfer learning for Bayesian optimization: A survey. *arXiv preprint arXiv:2302.05927*, 2023.

Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: A framework for efficient Monte-Carlo Bayesian optimization. In *Advances in Neural Information Processing Systems 33*, 2020.

Copernicus. Sentinel-5P TROPOMI Level 2 Nitrogen Dioxide total column products. Version 01, 2018. URL `https://sentinels.copernicus.eu/web/sentinel/data-products/-/asset_publisher/fp37fc19FN8F/content/sentinel-5-precursor-level-2-nitrogen-dioxide`. Processed by ESA. DOI: 10.5270/S5P-s4ljg54.

George De Ath, Jonathan E Fieldsend, and Richard M Everson. What do you mean? The role of the mean function in bayesian optimisation. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, pages 1623–1631, 2020.

Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. Initializing Bayesian hyperparameter optimization via meta-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

Roman Garnett. *Bayesian optimization*. Cambridge University Press, 2023.

Sigrid Passano Hellan, Christopher G. Lucas, and Nigel H. Goddard. Optimising placement of pollution sensors in windy environments. *NeurIPS 2020 Workshop on AI for Earth Sciences*, 2020.

Sigrid Passano Hellan, Christopher G. Lucas, and Nigel H. Goddard. Bayesian optimisation for active monitoring of air pollution. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(11):11908–11916, 2022.

Matthew D Hoffman and Andrew Gelman. The No-U-Turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1): 1593–1623, 2014.

Donald R Jones, Matthias Schonlau, and William Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

Vidhi Lalchand and Carl Edward Rasmussen. Approximate inference for fully Bayesian Gaussian process regression. *Symposium on Advances in Approximate Bayesian Inference*, pages 1–12, 2020. PMLR.

Samuel Müller, Matthias Feurer, Noah Hollmann, and Frank Hutter. PFNs4BO: In-context learning for Bayesian optimization. In *International Conference on Machine Learning*. PMLR, 2023.

Valerio Perrone, Huibin Shen, Matthias W Seeger, Cedric Archambeau, and Rodolphe Jenatton. Learning search spaces for Bayesian optimization: Another view of hyperparameter transfer learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Du Phan, Neeraj Pradhan, and Martin Jankowiak. Composable effects for flexible and accelerated probabilistic programming in NumPyro. *arXiv preprint arXiv:1912.11554*, 2019.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *IEEE Transactions on Information Theory*, 58(5):3250–3265, 2012.

Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task Bayesian optimization. *Advances in neural information processing systems*, 26, 2013.

Zi Wang, George E Dahl, Kevin Swersky, Chansoo Lee, Zelda Mariet, Zachary Nado, Justin Gilmer, Jasper Snoek, and Zoubin Ghahramani. Pre-training helps Bayesian optimization too. *arXiv preprint arXiv:2207.03084*, 2022.

Martin Wistuba and Josif Grabocka. Few-shot Bayesian optimization with deep kernel surrogates. In *International Conference on Learning Representations*, 2021.

Appendix A. Supplementary material

A.1 Ethics statement

The work uses no personal data and the goal of the application is to reduce the impact of air pollution on human health. The authors have no ethical concerns.

A.2 PLeBO optimisation step

Algorithm 1 gives the procedure for calculating the acquisition function which is followed at each optimisation step. This is also expressed in Eq. (1), which shows how this corresponds to the expected acquisition function when drawing the hyperparameters from their posterior distribution. $\boldsymbol{\mathcal{D}}_{1:N}$ is the set of tuning observations. $\boldsymbol{\mathcal{D}}_j^{:i}$ is the test observations available at iteration i.

Algorithm 1 PLeBO acquisition calculation

Given $\boldsymbol{\theta}^{\text{cand}}$, $\boldsymbol{\mathcal{D}}_j^{:i}$, $\boldsymbol{X}$, $a_i^\theta(\cdot)$
$\boldsymbol{a}, W \leftarrow \mathbf{0}, 0$
for h in $1, \ldots, H$ **do**
  $w \leftarrow p(\boldsymbol{\mathcal{D}}_j^{:i} | \boldsymbol{\theta}_h^{\text{cand}})$
  $\boldsymbol{a} \leftarrow \boldsymbol{a} + w\, a_i^\theta(\boldsymbol{X}, \boldsymbol{\theta}_h^{\text{cand}})$
  $W \leftarrow W + w$
end for
Return $\boldsymbol{a}/W$

$$a_i(\boldsymbol{x}) \;=\; \mathop{\mathbb{E}}_{p(\boldsymbol{\theta}_h^{\text{cand}} | \boldsymbol{\mathcal{D}}_{1:N}, \boldsymbol{\mathcal{D}}_j^{:i})} [a_i^\theta(\boldsymbol{x}, \boldsymbol{\theta}_h^{\text{cand}})] \approx \frac{1}{W} \sum_{h=1}^{H} w_h\, a_i^\theta(\boldsymbol{x}, \boldsymbol{\theta}_h^{\text{cand}}),\;\; \boldsymbol{\theta}_h^{\text{cand}} \sim p(\boldsymbol{\theta}_h^{\text{cand}} | \boldsymbol{\eta}) \quad (1)$$

A.3 Runtime

Table 1 gives the runtime for the preprocessing step of PLeBO for each benchmark. For the air pollution benchmarks we use 100 evaluations from each of the tuning tasks. Table 2 gives the mean duration of each optimisation step for all the considered methods. We see that PLeBO is much slower than the other methods. But for expensive optimisation tasks, where each step corresponds to e.g. installing a pollution sensor or training a neural network, this cost is negligible. We have capped the number of past evaluations available to DirectTrans at 100, otherwise it would be much slower than the listed times.

The optimisation step of PLeBO scales as $\mathrm{O}(H(i + N_{\text{start}})^3)$, as opposed to $\mathrm{O}(N_{\text{grad}}(i + N_{\text{start}})^3)$ for EI where N_{grad} is the number of gradient steps (Rasmussen and Williams, 2006, p. 19). H is a hyperparameter of the method which we can adjust to balance computational cost with the quality of the posterior estimate of the GP hyperparameters. At one extreme we could set H to 1, which should give a similar runtime to Shared.

Benchmark	Preprocessing time
Synthetic	9 min 11 sec
Strong	2 h 38 min
Selection	3 h 22 min

Table 1: PLeBO preprocessing times.

Method	Synthetic	Strong	Selection
RandomSearch	0.006	0.005	0.005
BoTorch	0.20	0.31	0.27
EI	0.16	0.11	0.11
UCB	0.17	0.11	0.11
DirectTrans	1.35	1.20	1.20
Initial	0.16	0.10	0.11
PLeBO	6.73	6.60	6.89
TruePLeBO	0.05	-	-
Gamma	0.18	0.13	0.12
Shared	0.05	0.04	0.04

Table 2: Mean durations in seconds of an optimisation step.

A.4 Prior quality

Fig. 5 compares the true and inferred values of $\boldsymbol{\eta}$ and $\boldsymbol{\theta}$ for the lengthscale and signal variance. The learned distributions are more peaked than the true ones, but are similar and reasonable approximations given that we only use 10 tuning tasks.

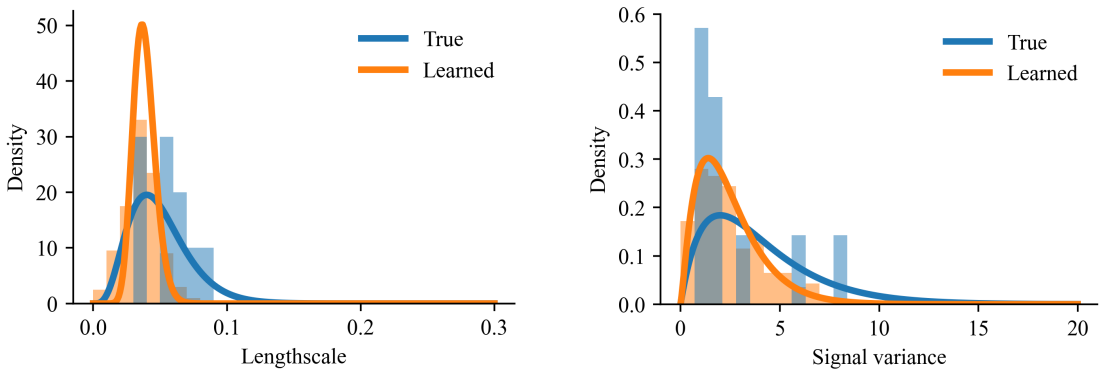


Figure 5: Comparing true and inferred hyperparameter priors for the synthetic benchmark.

Appendix B

Chapter 4 Supplementary Material

B.1 London data

The sampling heights for the sensors used are plotted in Fig. B.1. Fig. B.2 shows the number of measurements available per day for the test (2016) and tuning data (2015). More stations had measurements available in 2016.

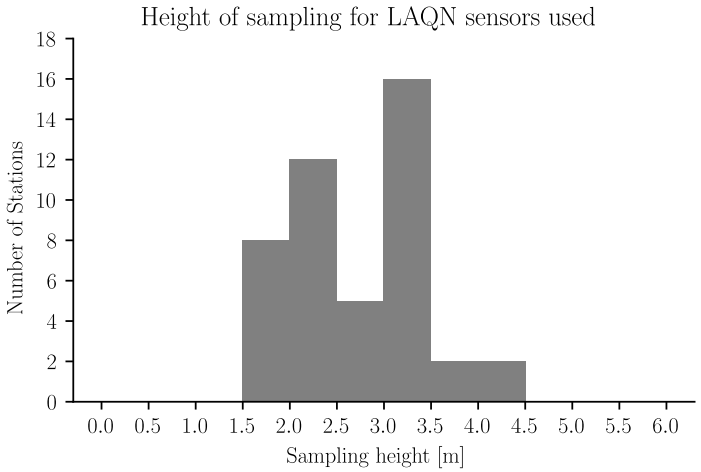


Figure B.1: Sampling heights for LAQN stations used, for the 45 out of 55 stations for which sampling height is listed.

B.2 Hyperparameter optimisation

We optimised the hyperparameters directly in the initial experiments and the simple prior experiments. We used the `minimize` function from `scipy.optimize` with the L-BFGS-B method (Virtanen et al., 2020). For the model tuning in both sets of experiments we ran `minimize` for up to 1000 iterations.

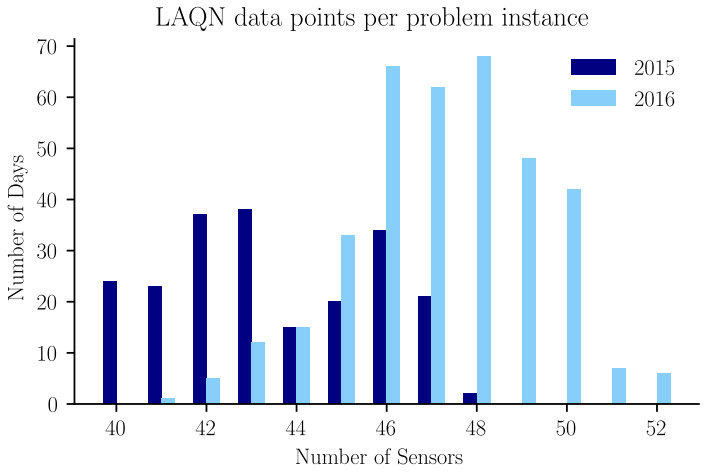


Figure B.2: Number of days with different number of stations reporting measurements for the tuning (2015) and test (2016) data.

When fitting the tuning images in the simple prior experiments we ran the optimisation for up to 100 iterations.

B.3 Reproducibility

- **Data sets**: The London data set is freely available online (Imperial College London, 1993). A slightly updated version[1] of the preprocessing is available at `https://github.com/sighellan/LAQN-BO`, as prepared for Hellan, Lucas, and Goddard (2023a), see Chapter 3. The satellite data set is based on data freely available online (Copernicus Sentinel-5P (processed by ESA), 2018), but not the filtered version used in this work.
- **Seeds**: The random number generator used for MCMC is seeded with the number 13, as is also specified in the code. There is little randomness in the BO loop, since the noise is set very low.
- **(Hyper-)parameter values**: The hyperparameter values needed to run the code are given, e.g. number of burn-in samples, number of BO iterations. The values for the model hyperparameters, e.g. the kernel lenghtscales for each model, are not given as they are outputs of the method and there is a large number of them.

1. In Chapter 4 we mean-centre the tasks using data from the same task. In the updated version we use the tuning data to standardise the tasks from both years (mean-centring and division by the standard deviation).

- **Hyperparameter selection**: We do not have a full list of the hyperparameter settings tried. However, a general description of how they were set is given below. An exhaustive hyperparameter search to maximise performance was not conducted: the goal of the work was instead to evaluate the suitability of the method.
  - **MCMC hyperparameters**: Different settings were tried for the number of samples and the proposal distribution. The number of samples was first tried at 100 or 200, but this was found to be too low for mixing. Instead, 1200 and 2000 were used for the satellite data and the London data, respectively. This was found to be a good balance between the required computation time and the usefulness of the samples. The $\sigma_{\eta,g,c}$ values in Appendix B.4.3 were experimented with to give good mixing. We first tried higher values but then reduced them to get a higher acceptance ratio. We found that higher values for the shape parameter ψ than the scale parameter ϕ worked better. We initially tried updating all of $\boldsymbol{\theta}$ jointly, and then all of $\boldsymbol{\eta}$ jointly, but this led to much slower mixing.

B.4 MCMC sampling

This section gives more details of the MCMC sampling, as done in Chapter 4. It does not apply to the improved MCMC sampling in Chapter 5.

B.4.1 Initialisation

$\boldsymbol{\eta}^{(0)}_{:G_{\text{nc}}}$ is set to a vector of ones. $\boldsymbol{\eta}^{(0)}_{G_{\text{nc}}:}$ is the parameters for the distributions of σ_n^2 and γ (if present), which stay constant throughout the sampling process. $\boldsymbol{\theta}^{(0)}$ is set by sampling from $\boldsymbol{\eta}^{(0)}$.

B.4.2 Sampling theta

The sampling of $\theta^{(h)}_{n,g}$ given $\boldsymbol{\eta}^{(h-1)}$, $\boldsymbol{\theta}^{(h)}_{n,\not g}$ and $\theta^{(h-1)}_{n,g}$ is done as follows: At each iteration a proposal $\theta'_{n,g} \sim Q_\theta(\theta'_{n,g}|\theta^{(h-1)}_{n,g})$ is generated, and either accepted or rejected proportionally to the acceptance ratio $a_\theta(\theta'_{n,g}|\theta^{(h-1)}_{n,g})$ where $\theta^{(h-1)}_{n,g}$ is the previous sample value. If the proposal is rejected the old value is used for the new sample, see Eq. (B.4). $\theta^{(h)}_{n,g}$ is the g-th GP hyperparameter fitted to data snapshot $\boldsymbol{\mathcal{D}}_n$ at MCMC iteration h. The unnormalised probability function $\pi(\theta'_{n,g})$ is given in Eq. (B.1), the proposal distribution in Eq. (B.2) and the acceptance ratio in Eq. (B.3).

$$\pi(\theta'_{n,g}) = p(\boldsymbol{\mathcal{D}}_n|\theta'_{n,g}, \boldsymbol{\theta}^{(h)}_{n,\not g})p(\theta'_{n,g}|\boldsymbol{\eta}^{(h-1)}) \tag{B.1}$$

$$Q_\theta(\theta'_{n,g}|\theta^{(h-1)}_{n,g}) = p(\theta'_{n,g}|\boldsymbol{\eta}^{(h-1)}) \tag{B.2}$$

$$a_\theta(\theta'_{n,g}|\theta^{(h-1)}_{n,g}) = \min\left(1, \frac{p(\boldsymbol{\mathcal{D}}_n|\theta'_{n,g}, \boldsymbol{\theta}^{(h)}_{n,\not g})}{p(\boldsymbol{\mathcal{D}}_n|\theta^{(h-1)}_{n,g}, \boldsymbol{\theta}^{(h)}_{n,\not g})}\right) \tag{B.3}$$

$$\theta^{(h)}_{n,g} = \begin{cases} \theta'_{n,g} & R_v \leq a_\theta(\theta'_{n,g}|\theta^{(h-1)}_{n,g}) \qquad R_v \sim \text{Uniform}(0,1) \\ \theta^{(h-1)}_{n,g} & \text{otherwise} \end{cases} \tag{B.4}$$

$p(\boldsymbol{\mathcal{D}}_n|\theta'_{n,g}, \boldsymbol{\theta}^{(h)}_{n,\not g})$ is evaluated through the standard GP expression in Eq. (2.6), and $p(\theta'_{n,g}|\boldsymbol{\eta}^{(h-1)})$ is defined in Eq. (B.5). For computational efficiency, we limit the number of samples for each tuning data snapshot $\boldsymbol{\mathcal{D}}_n$ to 100.

$$p(\theta'_{n,g}|\boldsymbol{\eta}^{(h-1)}) = \begin{cases} \Gamma(\theta'_{n,g}; \psi_g^{(h-1)}, \phi_g^{(h-1)}) & g \text{ for } \{\sigma_{\text{r}}^2, l_{\text{r}}, \sigma_{\text{w}}^2, l_{\text{w}}\} \\ \text{Uniform}(\theta'_{n,g}; 0, \pi) & g \text{ for } \gamma \end{cases} \tag{B.5}$$

	$\eta'_{g,b,c} = \psi^{(h)}_{g,b}$	$\eta'_{g,b,c} = \phi^{(h)}_{g,b}$
g for l_r or l_w	1.5	0.5
g for σ^2_r or σ^2_w	0.3	0.1

Table B.1: Values of $\sigma_{\eta,g,c}$ for different hyperparameters (rows) and the two η variables per hyperparameter (columns).

B.4.3 Sampling eta

We have two $\boldsymbol{\eta}$ parameters, ψ and ϕ, for each learned hyperparameter prior. They are sampled in the same way, so we introduce the notation $\eta^{(h)}_{g,b,c}$ for the $\boldsymbol{\eta}$ variable we are resampling, and $\eta^{(h)}_{g,b-1,c}$ for its previous value. We use $\eta^{(h)}_{g,b,c'}$ for its partner, i.e. the one we are not currently resampling. In the two relevant lines of Algorithm 1 this becomes:

- Line 17: $\eta^{(h)}_{g,b,c} = \psi^{(h)}_{g,b}, \quad \eta^{(h)}_{g,b,c'} = \phi^{(h)}_{g,b-1}, \quad \eta^{(h)}_{g,b-1,c} = \psi^{(h)}_{g,b-1}$
- Line 18: $\eta^{(h)}_{g,b,c} = \phi^{(h)}_{g,b}, \quad \eta^{(h)}_{g,b,c'} = \psi^{(h)}_{g,b}, \quad \eta^{(h)}_{g,b-1,c} = \phi^{(h)}_{g,b-1}$

The sampling of $\eta^{(h)}_{g,b,c}$ given $\boldsymbol{\theta}^{(h)}$, $\eta^{(h)}_{g,b,c'}$ and $\eta^{(h)}_{g,b-1,c}$ is done as follows: At each iteration a proposal $\eta'_{g,b,c} \sim Q_\eta(\eta'_{g,b,c}|\eta^{(h)}_{g,b-1,c})$ is generated, and either accepted or rejected proportionally to the acceptance ratio $a_\eta(\eta'_{g,b,c}|\eta^{(h)}_{g,b-1,c})$. If the proposal is rejected, the old value is used for the new sample, see Eq. (B.9). The unnormalised probability function $\pi(\eta'_{g,b,c})$ is given in Eq. (B.6). The proposal distribution is given in Eq. (B.7) and the acceptance ratio in Eq. (B.8). $\psi^{(h)}_{g,b}$ is the shape parameter of the gamma distribution, and $\phi^{(h)}_{g,b}$ the scale parameter. The prior on $\boldsymbol{\eta}$, $p(\eta'_{g,b,c})$, is set to one for valid values and zero otherwise. $\sigma_{\eta,g,c}$ in Eq. (B.7) is a parameter determining the width of the proposal distribution Q_η, with values given in Table B.1.

$$\pi(\eta'_{g,b,c}) = p(\eta'_{g,b,c}) \prod_{n=1}^{N} p(\theta^{(h)}_{n,g}|\eta'_{g,b,c}, \eta^{(h)}_{g,b,c'}) \tag{B.6}$$

$$Q_\eta(\eta'_{g,b,c}|\eta^{(h)}_{g,b-1,c}) = \mathcal{N}(\eta'_{g,b,c}; \eta^{(h)}_{g,b-1,c}, \sigma^2_{\eta,g,c}) \tag{B.7}$$

$$a_\eta(\eta'_{g,b,c}|\eta^{(h)}_{g,b-1,c}) = \min\left(1, \frac{\pi(\eta'_{g,b,c})}{\pi(\eta^{(h)}_{g,b-1,c})}\right) \tag{B.8}$$

$$\eta^{(h)}_{g,b,c} = \begin{cases} \eta'_{g,b,c} & R_v \leq a_\eta(\eta'_{g,b,c}|\eta^{(h)}_{g,b-1,c}) \qquad R_v \sim \mathrm{Uniform}(0,1) \\ \eta^{(h)}_{g,b-1,c} & \text{otherwise} \end{cases} \tag{B.9}$$