This thesis has been submitted in fulfilment of the requirements for a postgraduate degree (e.g. PhD, MPhil, DClinPsychol) at the University of Edinburgh. Please note the following terms and conditions of use:

This work is protected by copyright and other intellectual property rights, which are retained by the thesis author, unless otherwise stated.
A copy can be downloaded for personal non-commercial research or study, without prior permission or charge.
This thesis cannot be reproduced or quoted extensively from without first obtaining permission in writing from the author.
The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the author.
When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given.
Machine Learning for Soft Matter Image Analysis

Emily Gould

Doctor of Philosophy
The University of Edinburgh
February 2021
Abstract

Over the last few years there has been a great increase in interest in the application of machine learning to problems in physics. Within machine learning there is a wide range of methods that can be used for a huge variety of different problems applicable to soft matter such as the classification of data and the processing of images, but the use of these methods in the field is still in its infancy. This thesis aims to explore how well-established machine learning methods can be put to use to gain more information from historical image data of soft matter systems.

The data investigated were historical confocal images of particle-stabilised emulsion systems: bicontinuous interfacially jammed emulsion gels (bijels) and emulsion droplets stabilised by oppositely charged particles. The first bijel data were a collection of images taken from a series of successful and failed bijel syntheses. Machine learning classification algorithms were trained on this data to create a tool for predicting whether an unseen image is from a successful or unsuccessful bijel. A variety of techniques were trialled on a range of variables derived from the autocorrelation function and structure factor of the images, and the best-performing combination of these formed the final model for use in classifying future images. Further data was then added to the model in the form of a second imaging channel and the process was repeated: this model had an improved predictive accuracy. It was found that variables associated with a characteristic length scale in the image were the most useful for classification.

The second data were confocal images of droplets stabilised by a mixture of positively and negatively charged colloidal particles. Unsupervised machine learning was used to look for trends in the data that could not be found by traditional methods. These traditional methods were also applied to the data to ascertain how droplet roundness changed as the ratio of positive to negative particles increased. Again, a few different algorithms and variable sets were trialled in order to find the most useful combination. I found that the features I
considered follow a different trend to that seen for the size of the droplets in the literature.

Finally, machine learning was applied to images of the path of air bubbles through bijel samples after centrifugation in order to ascertain the extent of the effect of the bubble’s passage on the bijel structure. I used a regression algorithm to investigate how the distance from the track left by the bubble could be predicted based on the autocorrelation function of sections of the image. The results can be compared to very recent research using bubbles for quantitative micro-rheology.

These examples demonstrate a range of machine learning techniques and their application to soft matter image analysis. One output is a bijel classification tool. With this, and by demonstrating a deeper understanding of the system through data exploration, machine learning is shown to have a wide range of potential uses within the field. The examples also highlight a persistent problem created by the small amounts of data usually available, and some strategies for using these data as effectively as possible are demonstrated.
Lay summary

Machine learning has become somewhat of a buzzword recently, but at its roots it is simply a collection of methods for finding answers from data. There are many different machine learning methods for different tasks, such as figuring out which group something belongs to (‘is this a photo of a cat or a dog?’) and predicting the numerical value of something (‘what rating will this person give this movie?’).

In this thesis I apply machine learning to different emulsion systems. An emulsion is a mixture of two liquids that do not want to mix, and it usually takes the form of small droplets of one liquid floating around in the other liquid. Because the two liquids do not want to mix, emulsions usually separate quickly into separate layers of the two liquids: this is commonly seen when salad dressings are left for too long after being mixed together. We can prevent this by adding something called an emulsifier into the mixture, which makes the emulsion more stable so it stays mixed for longer. These emulsifiers can either be surfactants, which are a type of molecule where one part wants to be in water and the other part wants to be in the oil, or very small solid particles (e.g. tiny plastic spheres).

When choosing particles to stabilise an emulsion, there are a lot of different options to consider, including mixtures of different types of particles. In one chapter of this thesis I look at emulsion droplets that have been stabilised by a mixture of positively charged and negatively charged particles in different ratios. I use machine learning to decide which images of the particles are similar to each other, in order to find out what makes the droplets similar or different.

One specific type of emulsion that I look at is called a bijel. Instead of small droplets of one of the liquid in the other, a bijel has both liquids intertwined in a structure rather like a sponge: one liquid takes the role of the solid part of the sponge and the other liquid takes the role of the air inside the sponge. Lots of people are interested in using bijels to make materials for things like energy
storage and medical applications. In two of the chapters in this thesis I use machine learning to make a tool for deciding whether or not this structure has formed correctly. In my final chapter I use a different form of machine learning to predict how far a patch of a bijel is from an area where a bubble has broken the bijel, and use this prediction to find out how strong the bijel is.
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.

Parts of this work have been published in [1]. All of the content from the publication that is presented in this thesis is my own work.

(Emily Gould, February 2021)
Acknowledgements

First and foremost, I want to thank Paul Clegg for his patience and support throughout the last few years, going above and beyond to help me reach the end of my PhD study. I would also like to thank Job Thijssen for encouraging me to even consider pursuing a PhD, and for continually letting me know that he was there to help with any questions I had even if I rarely took him up on the offer.

This thesis would have not been possible without the work of those who provided me with experimental data, so I extend great thanks to Katherine Macmillan and David French for providing me with data from their experiments, and for answering my many questions about their work long after they were done with that work themselves.

During the last few years I have made some fantastic friends who have made the time I spent on this PhD all the more enjoyable. The people who worked in my office, came to PPA events and Firbush trips, or hung out in the coffee room at lunch contributed to an atmosphere that was a joy to be a part of. In particular I’d like to thank Iain Muntz, Eoin Ó Laighléis, and David Crosby for reminding me of all the physics I should already know, and Katherine Brown for always being there when I needed a chat. I am so glad that spending this time at Edinburgh meant that I could continue to be a part of the fabulous Exmoor Pony Trekking Section and to join the Female Voice Choir, who welcomed me with open arms despite my busy schedule.

Finally I thank Sam Henderson for always being there for me. Not only did he keep me sane through the ups and downs of this whole process and agree to marry me during it, but he was also mad enough to proof read this entire thesis.
## Contents

Abstract  
Lay summary  
Declaration  
Acknowledgements  
Contents  
List of Figures  
List of Tables  

1 Introduction  

1.1 Machine learning ................................................................. 1  
1.2 Uses of machine learning in physics and beyond ...................... 2  
1.2.1 Machine learning in physics .............................................. 2  
1.2.2 Machine learning for image analysis .................................... 4  
1.3 Pickering emulsion systems .................................................... 4  
1.3.1 Particles at interfaces ...................................................... 4  
1.3.2 Emulsions ..................................................................... 6  
1.3.3 The physics of partially miscible liquid mixtures ................... 7
1.3.4 Bijels

1.4 Aims and outline of this thesis

2 A guide to machine learning

2.1 Fundamentals of machine learning

2.1.1 Supervised vs. unsupervised algorithms

2.1.2 Parametric vs. non-parametric models

2.1.3 Eager vs. lazy learners

2.1.4 Errors and cross-validation

2.1.5 Model validation

2.2 Machine learning algorithms

2.2.1 K-nearest neighbours classification

2.2.2 Linear regression

2.2.3 Logistic regression

2.2.4 Decision trees

2.2.5 Support vector machines

2.2.6 Dimension reduction methods

2.2.7 Clustering methods

3 Autonomous identification of bijels from a two-dimensional image

3.1 Introduction

3.1.1 The growing interest in bijel fabrication

3.1.2 Limitations of characterising bijels

3.2 Data
5 Using unsupervised learning to analyse the structure of oppositely charged particles on the surface of a sphere

5.1 Introduction

5.1.1 Emulsions stabilised by oppositely charged particles

5.2 Data

5.3 Traditional image analysis methods

5.4 Machine learning methods

5.4.1 Machine learning approach

5.4.2 Clustering algorithms

5.4.3 Preparing the data for machine learning

5.4.4 Extracting variables for machine learning

5.4.5 Quantifying clustering performance

5.4.6 Visualisation of results

5.5 Results and discussion

5.5.1 Traditional image analysis results

5.5.2 Basic clustering

5.5.3 Finding the best clustering performance

5.5.4 Interpretation of clustering

5.6 Conclusions
6 Investigating the effects of local forces within bijels under centrifugation

6.1 Introduction

6.1.1 Forces in bijels

6.1.2 Microrheology using bubbles

6.1.3 Aims of this chapter

6.2 Data

6.3 Methods

6.3.1 Image segmentation

6.3.2 Preparing the data

6.3.3 Investigating changes with distance from bubble track

6.3.4 Support vector regression

6.4 Results and discussion

6.4.1 Image segmentation

6.4.2 Relationship between bijel orientation and distance from bubble track

6.4.3 Support Vector Regression to find distance from bubble track

6.5 Conclusions

7 Conclusions and future outlook

Bibliography
# List of Figures

1.1 Free energy of detachment of a spherical particle calculated with $r = 20\text{nm}$ ........................................... 5

1.2 Particles with an interfacial contact angle of $< 90^\circ$ (left) and $> 90^\circ$ (right) tend to form emulsions of differing internal phase due to the spherical particles sitting more in one phase than the other and inducing a curvature in the interface ......................................................... 7

1.3 An example phase diagram of a system of two partially miscible liquids. The behaviour of the system depends on the composition $\phi$, which describes how much of each liquid there is, and the temperature. Although the example here is symmetric, this is not always the case ................................................................. 8

1.4 An example of the shape of a system de-mixing via spinodal decomposition. The darker region is one liquid and the lighter region is the other. Taken from [91] under fair use ........................................... 9

1.5 The evolution of the structure factor of a system undergoing spinodal decomposition. At each timestep, there is a single peak in the structure factor indicating a single characteristic lengthscale in the system. This peak moves as the system coarsens. Taken from [99] under fair use ........................................... 10

2.1 A summary of the machine learning algorithms that will be discussed in this chapter and how they fit into the paradigms of supervised or unsupervised learning .................................................. 15

2.2 An illustration of 5-fold cross-validation. The data is randomly split into 5 equally-sized sections (folds), and each fold takes a turn being used as a test set for validation while the others are used for training ................................................................. 21

2.3 Example of a Receiver Operator Characteristic (ROC) curve. The area under the curve (AUC) is a measure of the quality of the result .................................................. 23
2.4 An example histogram showing how the machine learning model performs well only when the samples are correctly labelled. The histogram shows the distribution of error rates for models trained on randomly labelled data, and the line shows the error for the model trained on the true data. 24

2.5 The effect of the value of $k$ in a k-nearest neighbour classification. Here each point in variable space has been predicted as either the blue or red class based on the data given, and the number of neighbours ($k$) used in the algorithm has a strong effect on the predictions. 26

2.6 An example of KNN classification accuracy as a function of number of neighbours. 27

2.7 A demonstration of the curse of dimensionality. The same 30 random points have been plotted in (a) one, (b) two and (c) three dimensions. As the number of dimensions increases, the points become more spaced out and therefore a larger region is required to capture the same number of points. 27

2.8 An example single-variable logistic regression result. In red we see the estimated probability that a point with input value $x$ belongs to the given class, and in grey we have a histogram of the true classifications as a function of $x$. 33

2.9 An example of the decision boundary drawn by a decision tree classifier. In this case, the classifier is applied to palmerpenguins dataset. Taken from [123] under fair use. 35

2.10 An example decision tree classification using the Iris dataset. Each branch shows how the algorithm chose to split the variable space during recursive binary splitting. Each node shows the predicted class, the probabilities of a point in that node belonging to each possible class, and the percentage of all data sorted into that node. 35

2.11 An example of 2-class (red/green) data that is perfectly separable by a number of different hyperplanes, such as the ones shown in black here. 38

2.12 A plot to demonstrate support vector regression. Only the points outside the dotted lines are considered when choosing the line of best fit, shown by the solid line, and the error in the fit considers only the distance from each point to the dotted lines rather than the solid line. 41

2.13 The effect of (a) PCA and (b) t-SNE applied to the MNIST data set [132]. 46
2.14 An example dendrogram showing hierarchical clustering, created using the Iris data set. The height at which two clusters are joined is the dissimilarity between the two clusters, as measured by the chosen linkage method. The red lines show cutoff values of dissimilarity where the data could easily be split into clusters.

3.1 Experimental data used in this project. Panel (a) shows the outcome of 135 attempted bijel samples indicating whether a successful bijel was created as a function of particle wettability and nitromethane mass fraction. Where samples 1 and 2 have the same composition, they are made from the same sample batch split in half. Panels (b) and (c) show examples of a successful and an unsuccessful bijel respectively. The ethanediol phase (liquid channel) is shown in magenta on the left, and the stabilising silica particles (particle channel) are in cyan on the right.

3.2 Representative example bijels (in red) and non-bijels (in black) in terms of the radial average of their structure factors and autocorrelation functions. Panel (a) shows the structure factors of the liquid channel images, and panel (b) shows the autocorrelation functions of the same images.

3.3 Box-and-jitter plots of three of the variables considered in the initial model. These plots show how the values of the variable are distributed for bijels and non-bijels. Jitter (random offset) is added in the x-axis so all points can be seen. Panel (a) demonstrates a variable that is particularly good for separating bijels from non-bijels, with no overlap between the boxes representing the majority of the distributions of the two classes. Panel (b) demonstrates a variable that is poor for separating bijels from non-bijels, with a great deal of overlap between the two classes. Panels (c) and (d) demonstrate a variables where it is unclear whether it will be useful for separating bijels from non-bijels.

3.4 Box-and-jitter plots for the seven variables considered in the initial model. It is clear that some of these variables separate bijels from non-bijels more successfully than others: the number of turning points in the autocorrelation function (a) is a particularly poor example, and the position of the first turning point in the autocorrelation function (b) is a particularly good example. With other variables like the position of the biggest jump in the structure factor (g) it is unclear what the predictive performance will be.
3.5 A plot showing how the classification error changes as the number of variables in the KNN model is reduced by removing the least significant variable in the model each time. In this case, the optimal model is the smallest, with only one predictive variable.  

3.6 Cross-validation results for the final single-variable KNN fit to the liquid channel images, and how the cross-validated accuracy of the model changes with the number of neighbours used in the classification. Here the optimal value chosen by the algorithm was $k = 5$, the first peak in this plot.  

3.7 The output and classification performance of the final model. The plot shows the predicted (open circle) and true (filled circle) bijel classifications for each sample in the dataset, with $y$-axis jitter added because only one variable was used in the model. Predictions were output from the final liquid channel KNN fit.  

3.8 Some examples of the two-dimensional autocorrelation function of images used in this chapter. For some images, such as in panel (a), the assumption of radial symmetry is only valid for the central region of this function.  

4.1 Confocal micrographs of (left) the liquid channel of a bijel and (right) the particle channel of the same area of bijel. These two channels are often imaged separately using different fluorescent dyes.  

4.2 Top: Examples of the structure factor (a) and autocorrelation function (b) of the particle channel of representative bijel (in red) and non-bijel (in black) confocal images. Bottom: plot of all examples of the structure factor (c) and autocorrelation function (d), showing the broad distribution of these functions, particularly the structure factor. 12.5 μm corresponds to approximately 10 pixels in the image.  

4.3 Box-and-jitter plots for the five variables considered in the initial particle channel model. None of these variables appear to be particularly good at separating bijels from non-bijels, but a combination of them may fare better.  

4.4 A plot showing how the classification error changes as the number of variables in the particle channel model is reduced. The optimum performance is achieved when using the two most significant variables.
4.5 Results from the best particle channel logistic regression model. Predicted (open circle) and true (filled circle) bijel classification results for each sample in the dataset. Predictions were output from the final particle channel model and plotted as a function of the two variables in the model, using the same standardised values on which the model was trained. The line where a sample is equally likely to be a bijel or a non-bijel is plotted in blue. Here, all the variables have been standardised so are in arbitrary units.

4.6 A plot to show how the classification errors of the logistic regression and k-nearest neighbours models change for all possible combinations of the variables in the combined model. The points corresponding to the optimal models using variables from only the particle channel or the liquid channel are labelled for comparison. The optimal performance is achieved using the logistic regression algorithm with all three variables.

4.7 Results from the final logistic regression model using both liquid and particle channels. Predicted (open circle) and true (filled circle) bijel classification results for each sample in the data set plotted for each possible pair of the three variables used in the model. Each plot is a projection of all data onto the given plane.

4.8 Receiver operator characteristic curve for the final combined channel model fit, used to measure the quality of the fitted model. This shows how the true and false negative results change as a function of the probability threshold at which a sample is classed as a non-bijel. The area underneath the curve (AUC) is 0.91, indicating an excellent model.

4.9 Histogram of the error rate for 1000 iterations of the final model fitted with random bijel labelling. These errors are all significantly higher than the error achieved with the model fit to the true data, shown as a purple line.

4.10 Examples of the autocorrelation function of the particle channel of representative bijel (in red) and non-bijel (in black) confocal images. The location of the 10th and 20th points in the image are marked by blue and green vertical lines, respectively. 12.5 µm corresponds to approximately 10 pixels in the image.

4.11 A plot of bijel classifications for each sample in the training dataset, with the true classifications represented by filled circles and the classifications predicted by the particle channel model represented by open circles. The decision boundary defined by Equation 4.1 with \( p(\text{Bijel}) = 0.5 \) is shown as a blue dotted line. Here, all the variables have been standardised so are in arbitrary units.
4.12 A comparison of the predictions of the liquid channel model from Chapter 3 to a projection of the combined particle and liquid channel model presented in this chapter. The inclusion of more variables leads to a better performance in the combined model, with fewer samples classified incorrectly.

5.1 Two examples of 10× magnification images suggesting increasing circularity of the droplet with increasing $\phi_r$. The scale bar in each image is 100 µm.

5.2 An example of manually defined droplets in an image at 10× magnification. Each droplet was identified by eye based on information from all channels and the outline was drawn manually then smoothed. The scale bar in each image is 100 µm.

5.3 Two examples of 63× magnification images suggesting increasing circularity of the droplet with increasing $\phi_r$. Each image is a 2D projection of a z-stack through the droplet. It is difficult to distinguish individual particles on the droplet surface.

5.4 An example of how a droplet image was split into 12 sub-images.

5.5 The green (a) and red (b) channels of an example slice of a sub-image presented in rectangular form. These sub-images are both taken from one slice of the droplet depicted in Figure 5.4.

5.6 A demonstration of the processes of dilation (b), erosion (c), and closing (d) on an example image (a). Taken from [148] under fair use.

5.7 Examples of plots used to visualise a clustering result.

5.8 An example visualisation of all the data points fed into a machine learning algorithm to represent two example images. Each horizontal slice in the image represents one of the 12 slices that were used as separate data points during the clustering. The small white numbers denote which cluster that slice was grouped into, and the same number is repeated at multiple x values to ensure it can be seen.

5.9 An example of how a dataset can be visualised using two t-SNE dimensions to see where the data might be split into clusters.
5.10 An overview of how circularity changes as the fraction of negative particles in the system (\(\phi_r\)) increases. Panels (a) and (b) show the trends of four different measures of droplet roundness and circularity as the fraction of negative particles increases in the case where \(\phi_p = 0.5\%\) and \(\phi_p = 1.0\%\), respectively. There is an indication that droplets become more round as more negative particles are added to the system, and there is no evidence of symmetry about \(\phi_r = 50\%\).

5.11 An overview of the evidence used to decide how many clusters to use. Panel (a) shows a truncated example of a hierarchical clustering dendrogram for all the data with PCA retaining 99% of the variance, where each node may contain multiple image slices. Panel (b) shows a gap statistic plots for a range of input data and variables. Panels (c–f) show the data plotted on the t-SNE components calculated for different sets of input data and variables: the raw rectangular form (c), and after applying PCA retaining 99% variance to the rectangular forms of the whole dataset (d) and the two reduced datasets (e–f). In the graph titles, rect: rectangular data, all: all data, oneA/oneB: reduced dataset A/B (Section 5.5.3), pca99: PCA applied with 99% variance, p: perplexity of t-SNE.

5.12 Stacked bar plots showing how slices from each individual image are grouped into clusters based on the raw rectangular data. The bar for each image is coloured based on the frequency with which slices from that image were grouped into each cluster. Two of the images are missing a slice, this is because the connected-region variables (discussed later) could not be calculated for these slices. In the graph titles, rect: rectangular data, all: all data.

5.13 A plot of the \(\theta\)-averaged data in terms of its two t-SNE components. A perplexity value of 10 was used to calculate the components shown here, and other values of this parameter gave similar results. In the graph title, th-av: \(\theta\)-averaged data, all: all data, p: perplexity of t-SNE.

5.14 The effects of PCA on the data and clustering performance. Panel (a) shows how an example image slice is represented by different numbers of PCA components that explain different amounts of the variance within an image. Panels (b–c) show how the mean time taken for the clustering algorithms to be trained changes as the variance retained and therefore the number of components is changed. Each result is the mean of 10 repeats of 10 runs of the algorithm. Panels (d–e) show how the use of PCA to reduce the images to components explaining 97% or 99% of variance effects the clustering performance in terms of Gini score.
5.15 The green channel of the four images at composition $\phi_r = 0.5\%$, $\phi_r = 10\%$. (a) and (b) show the images that were removed, (c) and (d) were retained and considered in the reduced-data part of the analysis.

5.16 Stacked bar plots showing how slices from each individual image are grouped into clusters based on the reduced data with one image per experimental composition. The bar for each image is coloured based on the frequency with which slices from that images were clustered into each cluster. One of the images in set B is missing a slice, this is because the connected-region variables (used later) could not be calculated for this slice. In the graph titles, rect: rectangular data, oneA/oneB: reduced dataset A/B (Section 5.5.3).

5.17 Stacked bar plots showing how slices from each individual image are grouped into clusters based on the rectangular data both alone (a) and combined with the connected-region variables (b). The bar for each image is coloured based on the frequency with which slices from that images were clustered into each cluster. A few of the images are missing a slice, this is because the connected-region variables could not be calculated for this slice. In the graph titles, rect: rectangular data, all: all data, combi: combination of normal and connected-region variables.

5.18 Plots of the connected-region data in terms of two t-SNE components, calculated for two different perplexity values (top=5, bottom=30) and two different random number seeds. None of the plots show any indication of sensible clusters for these data. In the graph titles, rect: rectangular data, all: all data, wedge: connected-region data, p: perplexity of t-SNE.

5.19 Examples of images that have been grouped into clusters 1 (panels (a–d)), 3 (panels (e–h))m and 0 (panels i–l)). Each horizontal line represents one slice of the image, and the $\theta$-averaged value is shown for each value of $r$. Some slices were not sorted into the same cluster as the rest of the image, these have been removed.

The colour scale is the same for every image.
5.20 Compositional plots of hierarchical clustering results based on the values of experimental parameters $\phi_p$ (x-axis) and $\phi_r$ (y-axis). The frequency of each cluster at each composition is represented by the size of the circle. Panel (a) shows the result from the raw rectangular data. Panel (b) shows the result from the $\theta$-averaged data, and it is unclear but the $\phi_p = 0.5\%$, $\phi_r = 60\%$ sample is split equally between clusters 3 and 0. Panel (c) shows the result from the data with PCA applied to retain 99\% of the variance in the images, and has a very similar result to panel (a). Panel (d) is a key for comparing the clusters in different panels. In the graph titles, rect: rectangular data, th-av: $\theta$-averaged data, all: all data, oneA/oneB: reduced dataset A/B, pca99: PCA applied with 99\% variance.

6.1 Two examples of the images used in this chapter. The passage of a bubble has left a track through the bijel, visible here as the dark region across the centre of the image. In some cases, such as in panel (b), the ethanediol leaked from the bijel into the hole leaving a bright region at the edge of the bubble track.

6.2 Examples of some of the different image features calculated by the Trainable Weka Segmentation plugin for ImageJ. Selected pixels in the original image are represented by their counterparts in these and similar representations to form a set of features to use for training the classifier.

6.3 Examples of the different stages of processing the original bijel image. Panel (a) shows an example image with the hole removed. Panel (b) shows the first 64-pixel square sub-image of that image, from the top left of the original, which is 161 pixels from the hole. Panel (c) shows the autocorrelation function of that sub-image. Panels (d–f) show more examples of the autocorrelation function of different sub-images from the same original image.

6.4 Examples of the process and end result of calculating the orientation of a bijel sub-image or a segmented image. Panels (a), (b), and (c) show the contours, autocorrelation function, and original image, respectively, of a bijel sub-image. Panels (d) and (e) show the original image and autocorrelation function, respectively, of a segmented image.

6.5 Examples of a train/test split on my data. Panels (a–c) show the distance distribution of each set when split without stratification, and (d–f) with stratification. Stratification has prevented the over-representation of the smallest distances in the training data and therefore their under-representation in the test data.
6.6 Three examples of the results of two attempts at Weka segmentation trained on two different images. Panels (a–c) show the original images. Panels (c–f) show the corresponding segmented images based on training pixels selected from image (a). Panels (h–j) show the corresponding segmented images based on training pixels from image (b). The green and red regions are those classed as bijel and hole, respectively.

6.7 Three examples of the results of image segmentation via seeded region growth. Panels (a–c) show the original images, and panels (d–f) show the corresponding segmented images. The red regions are those classed as bijel and the green, yellow, and blue regions are classed as dark, intermediate, and bright regions of hole, respectively.

6.8 Scatter plots showing how the angle of each sub-image relates to its distance from the bubble track. Three different sizes of sub-image are shown: (a) 128 × 128 pixels, (b) 64 × 64 pixels, and (c) 32 × 32 pixels.

6.9 Scatter plots showing how the average absolute magnitude (in degrees) of the orientation of sub-images at different distances (in µm). Panel (a) shows the result for 64 × 64 pixel sub-images. Panels (b) and (c) both show the result for 32 × 32 pixel sub-images, but panel (c) includes the standard deviation of the angle in each distance bin shown as the pale blue area behind the scatter points.

6.10 Plots of the true vs. predicted distance of each 32 × 32-pixel bijel sub-image from the bubble track, in µm. The first panel shows the result for the entire dataset, the second panel shows the result for the training data, and the final panel shows the result for the test data. The training data is expected to be the closest to the perfect result, shown by the red line. The training and test data were selected from the full dataset using a stratified train/test split with 5 bins. Each point is plotted with transparency so the darker regions are those where more points overlap.

6.11 A plot of the range of true distances present in each 25 µm bin of predicted distances. The red strip shows the point at which the range of the distances is the same size as the distance itself, and is plotted to give a reference for the scale of the variation within each bin.
6.12 Areas of bubble track (purple), distorted bijel (green), and unaffected bijel (yellow) superimposed on three example images. In particular, the distorted region defined as the area less than 125 µm into the bijel from the bubble track, as determined by the distance past which the autocorrelation function can no longer predict the distance from the bijel.
List of Tables

3.1 The effect of the removal of individual variables on the classification performance of a trained k-nearest neighbours classification model. The cases where removing the variable has reduced the error are highlighted in bold. ........................................... 66

3.2 F-statistic values indicating the significance of each variable in the KNN liquid channel model. .................................................. 67

3.3 A breakdown of the predictions of the final model when applied to three different data sets. Each combination of true and predicted bijels and non-bijels is shown with a count of how many data points fit into that category. The numbers of true bijels predicted as bijels and the true non-bijels predicted as non-bijels (i.e. the correctly classified data points) are shown in plain text, and the numbers of incorrectly classified data points are shown in italics. The false positive rate (FPR) and false negative rate (FNR) are included for each data set. .......................................................... 70

4.1 A summary of the logistic regression fit to the particle channel. All values are quoted to 4 significant figures. The odds ratio for each variable is the exponential of the logistic regression coefficient for that variable. Where the odds ratio is less than one, it is expressed in terms of its inverse so the values can more easily be compared. 84

4.2 A breakdown of the predictions of the optimised particle channel model when applied to three different data sets. Each combination of true and predicted bijels and non-bijels is shown with a count of how many data points fit into that category. The numbers of correctly classified data points are shown in plain text, and the numbers of incorrectly classified data points are shown in italics. The false positive rate (FPR) and false negative rate (FNR) are included for each data set. .......................................................... 86
4.3 A breakdown of the predictions of the optimised combined channel model when applied to three different data sets. Each combination of true and predicted bijels and non-bijels is shown with a count of how many data points fit into that category. The numbers of correctly classified are shown in plain text, and the numbers of incorrectly classified data points are shown in italics. The false positive rate (FPR) and false negative rate (FNR) are included for each data set.

4.4 A summary of the classification performance of the three different optimised models on different data sets. The errors obtained by applying the models to the data on which they were trained are consistently a little lower than the cross-validated errors quoted in this chapter, but both allow for easy comparison of model performance.

5.1 Summary of the Gini scores for different unsupervised clustering models (k-means and hierarchical clustering) on different data and variables. A lower Gini score indicates that the model is more successful at keeping together data points that originate from the same image.
Chapter 1

Introduction

1.1 Machine learning

Machine learning is a research area that has gone through phases of expansion as well as more quiescent periods [2–4]. It is currently in a phase of dramatic expansion in which is is proving its worth to a wide range of disciplines [5–9]. The field is very close to that of statistical modelling, to the extent that it can be difficult at first to identify the difference between the two. Both machine learning algorithms and statistical models are used for the same goal; to gain meaningful information from data. The fundamental difference between the two is that, where statistical modelling involves fitting a deterministic function to relate a dependent variable to a number of independent variables, the goal of machine learning is simply to predict an output based on one or more inputs [10].

Machine learning is split into three categories of algorithm: supervised, unsupervised, and reinforcement learning [10]. In supervised learning, the user provides known inputs and outputs, and the computer learns to predict future unknown outputs based on their inputs. This can provide the user with a powerful and versatile prediction tool. Unsupervised learning is different because the machine is not provided with known outputs, but instead groups entries based on inputs only. Despite the lack of supervising outputs, unsupervised learning can reveal information about relationships between the inputs, and on the structure of the data in variable space. Reinforcement learning is somewhat of a halfway point between the other two categories, where outputs are known but are not used in
the initial training of an algorithm. Instead, they are used in between iterations of training to improve the algorithm: this information is often given to the algorithm in the form of a score to be optimised over multiple training runs or by telling the algorithm when its output is wrong but not what the correct output should be.

1.2 Uses of machine learning in physics and beyond

1.2.1 Machine learning in physics

The use of machine learning in physics has a long history and in fact the origin of some parts of machine learning is closely linked with statistical physics. Neural networks are a good example of such links [7, 11, 12], and statistical physics is still very relevant in machine learning today [13]. Now that the field is well established, there are a number of areas of physics in which machine learning is used extensively.

In particle physics and cosmology, machine learning has helped researchers to confront large challenges in the field. Both of these fields tend to work with complex computer simulations in order to understand large amounts of data, and signals of interest are very subtle deviations from the relevant null models to describe the background. Machine learning helps with these challenges because it provides a more data efficient approach, can help avoid the problem of heavy reliance on a simulation that contains a degree of uncertainty, and allows for this uncertainty to be accounted for when the simulation is used. The use of machine learning can also allow for the detection of anomalies from the null model that may indicate a signal without relying on a specific hypothesis about the properties of the signal. Machine learning can also be used to improve the simulations used in these fields or to replace them entirely with, for example, neural networks that can much more quickly synthesise realistic data [7].

In particle physics, a number of machine learning techniques have been used since their first adoption in the early 1990s [14, 16] and particularly since their first successful applications led to their wide acceptance as a useful toolkit. The algorithms used in this field can generally be separated into two different
classes: ‘multivariate analysis methods’ \cite{17} and neural networks \cite{18}, although the two are often used in tandem to solve a single problem. Multivariate analysis methods are forms of supervised machine learning techniques, and tend to be trained on data from Monte Carlo simulations due to the expensive nature of real experiments. Multivariate analysis has been used successfully in a number of applications such as the measurement of the top quark mass and the search for the Higgs boson \cite{17}.

In cosmology, machine learning is used for tasks such as calculating the photometric redshift of distant luminous bodies without the need for expensive spectroscopic surveys, using either classification and regression techniques \cite{19,21} or neural networks \cite{22,23}. Neural networks are also extensively used in the investigation of gravitational lensing for the task of identifying gravitational lenses in a set of galaxy clusters \cite{24,26}.

**Machine learning in soft matter**

The use of machine learning for soft matter applications is in its infancy compared to the widely accepted uses in particle physics and cosmology. However, a core set of techniques is becoming established in the field \cite{6}, most often in the design of soft materials \cite{27}. Examples include the design of self-assembled materials \cite{28,29} and the prediction of the properties of polymers \cite{30,33}, although this type of application is somewhat more developed in the fields of hard condensed matter \cite{34,36} and chemistry \cite{37}. Machine learning has also been used to predict the properties of soft matter systems such as liquid crystals \cite{38,41} and nanoparticle assemblies \cite{42,43}, and to analyse images of soft matter systems \cite{44,46}.

Principal component analysis is a commonly used machine learning tool in soft matter \cite{6}, from its early use in protein folding simulations \cite{47,48} to a wide range of modern applications including the analysis of blood for forensic \cite{49} and medical \cite{50,52} purposes, and the investigation of the free energy landscapes of proteins \cite{53,56}. Support vector machines are another common tool used for classification in soft matter and medical applications, such as the search for peptides with particular properties \cite{57,58} and the prediction of new drug candidates \cite{59,61}.
1.2.2 Machine learning for image analysis

Perhaps one of the most widely discussed uses of machine learning in the current time is the development of ‘artificial intelligence’ for applications such as identifying objects from images. These are very advanced projects that require a lot of computational power, but they have achieved great success due to investment in tools and research by large companies such as Google \cite{62,63} and Amazon \cite{64,65}, among many others.

There are, however, a number of simpler ways to use machine learning for image analysis without the need for millions of example images on which to train an algorithm. Perhaps one of the most pertinent fields where this is common is in medical imaging, where machine learning is used to identify signs of cancer \cite{66,68}, malaria \cite{69,71}, and Alzheimer’s disease \cite{72,73} from images of cells and tissue. These applications use a wide range methods to detect features in these images including support vector machines, neural networks, logistic regression, and k-nearest neighbours classification.

1.3 Pickering emulsion systems

Now that I have introduced the type of methods I will be using in this thesis, I shall introduce the type of systems on which I will use these methods. Although not necessary, it is useful to understand the physical systems from which data is obtained because it allows for more thorough and useful interpretation of the results.

1.3.1 Particles at interfaces

At the point where two immiscible fluids meet, there is an energetically costly interface. The system will, therefore, try to minimise the surface area of this interface in order to reduce the free energy of the system. However, there are many situations where it is desirable for the interface to be kept in a configuration with a larger surface area than would be energetically favourable (for example, many cosmetics would be rather ineffective if the oil and water in them were to separate) so the interface must be stabilised. Although it is common to stabilise an interface with molecular surfactants, particles that adsorb at the interface are
also useful for this. Where a surfactant reduces the free energy of the system by reducing the interfacial tension, and therefore the energy cost per unit area, particles reduce this energy by reducing the area of the fluid-fluid interface by replacing it with fluid-solid interfaces. The contact angle $\theta$ of a particle $p$ at an interface between fluids 1 and 2 is given by Young’s equation [74]:

$$\cos \theta = \frac{\sigma_{p1} - \sigma_{p2}}{\sigma_{12}}$$  \hspace{1cm} (1.1)

where $\sigma_{ab}$ is the interfacial tension between phases $a$ and $b$. When the contact angle is $0^\circ$ or $180^\circ$ the particle has no affinity for the interface and tends to remain dispersed in one of the bulk phases, and when it is $90^\circ$ the particle is neutrally wetting, meaning it has an equal affinity for both liquid phases and will sit with equal volume in each.

Figure 1.1: Free energy of detachment of a spherical particle calculated with $r = 20 \text{nm}$.

When a particle of radius $r$ adsorbs to an interface, the free energy of detachment is given by:

$$\Delta F_d = \pi r^2 \sigma_{12} (1 - |\cos \theta|)$$  \hspace{1cm} (1.2)

where $\sigma_{12}$ is the surface tension of the interface, and $\theta$ is the contact angle of the particle with the interface as defined in Equation 1.1. As can be seen in Figure 1.1, this free energy is much larger than $k_B T$ for all angles other than $0^\circ$ and $180^\circ$ (i.e. all conditions other than full wetting). The condition for irreversible
adsorption is therefore met whenever the particle is not fully wetted by either of the two fluid phases (it is partially wetted by both) and there is therefore a finite contact angle between the fluid-fluid interface and the surface of the particle so the particle wants to sit at the interface rather than being in bulk.

### 1.3.2 Emulsions

An emulsion is a system of two immiscible or partially miscible liquids, where one liquid is present in droplets dispersed in the continuous phase of the other. The droplets need to be stabilised by a surfactant or by particles, otherwise they will tend to coalesce in order to minimise the energetically costly interface between the two phases. Emulsions are widespread in industry and everyday life, particularly in food and cosmetics [75].

As previously discussed, particles can irreversibly adsorb onto an interface and thus reduce the interfacial energy of the system. This property can be used to create particle-stabilised emulsions, or Pickering emulsions [76, 77]. These emulsions tend to be more stable than those stabilised by a surfactant, particularly when they contain large droplets [78], as the particles form a solid barrier between droplets which is more effective at preventing coalescence [79] and Ostwald ripening. Ostwald ripening is the transfer of material from smaller to larger droplets due to differences in Laplace pressure, and particles on the interface stabilise against this as the particle layer resists expansion and shrinkage [80].

As long as the particles are not neutrally wetting, the particle will sit more in one phase than the other, and this can dictate which is the internal phase in a particle stabilised emulsion [81] as shown in Figure 1.2. This is not always the case, however. Catastrophic phase inversion is when changes in the relative volume of the two liquid phases cause an emulsion to invert, even though the contact angle of the particle would inhibit the inversion [82]. The contact angle dictates the volume fraction at which this inversion should occur.

In order to create an emulsion, an energy input of at least $4\pi \sigma r^2$ is required in order to form each droplet of radius $r$ in a system with interfacial tension $\sigma$. For emulsions, this energy is generally provided through physical mixing. The simplest method of creating an emulsion is to shake a sample of the three components (two immiscible fluids and a surfactant or particle stabiliser). When
1.2 Particles with an interfacial contact angle of $< 90^\circ$ (left) and $> 90^\circ$ (right) tend to form emulsions of differing internal phase due to the spherical particles sitting more in one phase than the other and inducing a curvature in the interface.

In the case of a particle-stabilised emulsion, the limit of the amount of interface in the final emulsion is controlled by the amount of particles present in the system and therefore the amount of interface they are able to stabilise. Since the area of the interface is directly linked to the size of emulsion droplets, the amount of particles in the system can be used to control the size of emulsion droplets. The total volume of emulsion droplets is of course controlled by changing the total volume of the liquid that forms the internal phase.

1.3.3 The physics of partially miscible liquid mixtures

Although it is common for emulsions to be fabricated by the creation of droplets via physical mixing of immiscible liquids, it is also possible to make them by utilising the properties of partially miscible liquids less commonly found in everyday life. These are combinations of liquids that can be mixed under some conditions, often at high temperatures, but under other conditions are immiscible.

A phase diagram, such as the one shown in Figure 1.3, can be drawn to show
under which circumstances a binary fluid pair mixes and de-mixes. In such a
diagram there are three distinct regions: the stable mixed region, the unstable
unmixed region, and a metastable region in which the mixture is stable only to
small energetic fluctuations [90]. In these phase diagrams, there are two important
lines: the binodal denotes where the mixed and unmixed phases can coexist, and
the spinodal is the line of points at which the curvature of the free energy function
$\frac{d^2 F}{d\phi^2}$ is zero. These two lines meet at their minima (maxima when plotted
in terms of temperature, Figure 1.3), the critical point of the system, and the
composition at which this point occurs is known as the critical composition.

Outside of the binodal, in the stable region shown in Figure 1.3, the two liquids
remain mixed. Inside the binodal they de-mix, and the method by which this
occurs depends on the spinodal. Between the binodal and the spinodal, the
curvature of the free energy function is greater than zero and the system is in a
metastable state. In this state, the mixture is stable to small fluctuations but
de-mixing occurs via nucleation when a small energy barrier is overcome. In
this process, a local fluctuation overcomes the energy barrier and creates a small
droplet of a de-mixed phase. If this droplet is large enough that the free energy
cost of creating its interface is overcome by the free energy gain from the fact
that the core of the droplet is a pure unmixed liquid, it grows until the system

Figure 1.3: An example phase diagram of a system of two partially
miscible liquids. The behaviour of the system depends on the
composition $\phi$, which describes how much of each liquid there is,
and the temperature. Although the example here is symmetric,
this is not always the case.
is completely de-mixed. Inside both the binodal and the spinodal, where the curvature of the free energy function is less than zero, the mixture is unstable and de-mixing begins immediately via spinodal decomposition \[92, 93\]. This de-mixing occurs as a negative diffusion, with molecules moving from areas of low concentration to areas of higher concentration and therefore increasing the separation of the two fluids. Fluctuations of any size can grow via this process, but not all fluctuations do so at the same rate; there is an optimal size of fluctuation that grows most quickly. Patterns created by spinodal decomposition therefore have a characteristic length scale but they are otherwise random as depicted in Figure 1.4.

1.3.4 Bijels

A bijel is a “bicontinuous interfacially jammed emulsion gel”: a special class of particle-stabilised emulsion prepared by arresting de-mixing that occurs via spinodal decomposition \[91, 94, 96\]. During spinodal decomposition particles adsorbed to the liquid-liquid interface jam as the interfacial area decreases, arresting the coarsening of the structure. The end result is a bicontinuous structure with the two phase-separated liquids intertwined and stabilised by the
adsorption of colloidal particles to the interface. This adsorption is effectively irreversible due to the high attachment energy of the particles to the interface [97], so the structure is stable against further coarsening once the particles are closely packed: they cannot move within the interface and the energy cost of leaving the interface is too high. The jamming of interfacial particles is assisted by the negative Gaussian curvature of the bijel structure, which creates attractive capillary forces between the particles [98]. The curvature of the interface is important to the bijel structure, so the interfacial particles must be neutrally wetting so that they do not affect the curvature of the interface via their affinity for one phase or the other. A bijel has a single characteristic length scale, the width of the liquid channels, which is determined by the amount of interfacial particles in the system. This can be seen in the structure factor of a system undergoing spinodal decomposition (Figure 1.5) which shows a single intensity peak, which can be related to the width of the liquid channels, for a system arrested at any given time during de-mixing via spinodal decomposition.

![Figure 1.5](image)

**Figure 1.5:** *The evolution of the structure factor of a system undergoing spinodal decomposition. At each timestep, there is a single peak in the structure factor indicating a single characteristic lengthscale in the system. This peak moves as the system coarsens. Taken from [99] under fair use.*

Bijels were first predicted using lattice Boltzmann simulations in which the coarsening of a binary liquid system undergoing spinodal decomposition was observed to be arrested by the jamming of nanoparticles in the system [94]. The first experimental realisation of a bijel came two years later, using a mixture of
2,6-lutidine and water stabilised with fluorescent silica particles that were heat treated to obtain neutral wetting \[91\]. Since then, bijels have been made from a range of materials: the main requirements bijel are a pair of partially miscible fluids of similar density and a particle that sits on the interface between them with a contact angle of 90° \[98\].

The tortuous, interconnected spinodal pattern that characterises bijels also makes them interesting materials for a number of potential applications. Their unique combination of high surface area and long-range bicontinuous connectivity make them interesting for applications involving chemical reactions and transport, such as in fuel cells \[100\] and electrodes \[101\]. The porous structure of bijels can be fabricated, or even changed post-fabrication, with a great deal of control, making them interesting materials for applications such as tissue engineering \[102\] and controlled release devices \[103\]. The high level of control over the microstructure of bijels means that they can even be used to create porous or hierarchically porous materials, either directly \[104\] or by replacing one of the fluid phases with a porous material \[105\].

The bijel structure can also be robust, and bijels can even display self-healing properties \[103\]. The structure has a negative feedback response to compressional forces because under compression the domains in the bijel become aligned perpendicular to the direction of compression \[106\], although this behaviour depends on the size of the bijel domains. The interfacial colloidal skeleton of a bijel is also reasonably robust by itself, as demonstrated by the ability to remix the liquid components while maintaining this skeleton, although its strength is notably reduced \[107\] \[108\]. This robust nature opens up the possibility for the use of bijels as templates, particularly because they have the potential to provide very effective transport of material in a wide range of applications: they perform very favourably in comparison to porous metals, inverse opals, and polymerised high internal phase emulsions \[109\].

### 1.4 Aims and outline of this thesis

In this thesis, I aim to give the reader a basic understanding of the fundamentals of machine learning via a selection of algorithms. I then use example problems from soft matter to demonstrate how machine learning can be applied to an image analysis problem and additionally I use these techniques to investigate the physics
behind different soft matter emulsion systems. In Chapter 2 I describe a range of the most commonly used machine learning techniques and outline the algorithms and principles behind each one.

In subsequent chapters I move on to demonstrating examples of using machine learning for image analysis in soft matter, using both R and Python and focussing on two-dimensional confocal images of two different emulsion systems. Although machine learning is often used on very large datasets where other methods are perhaps unsuitable, I demonstrate that it is also useful for smaller datasets such as those often available in soft matter. I take a physics-led approach to this task, where I aim to connect the machine learning results with physical interpretations and known physics concepts.

In Chapters 3 and 4 I apply machine learning to confocal images of successful and unsuccessful bijels.

In Chapter 3 I use supervised classification algorithms to create and optimise a tool for separating successful bijel samples from unsuccessful ones where the bijel structure did not properly form, based only on confocal images of the liquid components of the bijel. I demonstrate the use of the k-nearest neighbours algorithm in R, and a workflow for selecting the optimal variables to represent my data in this algorithm. I assess the physical implications of the variables that I discover to be relevant for this classification. I find that it is possible to classify bijels based on the characteristic lengthscale of the liquid channel.

In Chapter 4 I build on my work from the previous chapter to show that confocal images of the interfacial particles in the bijel also contains important information about the bijel that makes classification more successful. I again use supervised classification algorithms to separate the two types of image (successful and unsuccessful) but this time I use variables derived from the particle channel of the bijel and I demonstrate the use of the logistic regression algorithm in R. I then move on to combine the information from both the liquid and particle images into a final model and use additional methods for assessing the performance of a classification model to verify the success of this model in accurately identifying successful bijel samples. Finally I interpret the possible physical meaning of the particle channel variables that are useful for identifying bijels. I find that bijels are most successfully classified by a combination of variables from the liquid and particle channels, and that the useful particle channel variables are also related to the lengthscales in the system.
In Chapter 5 I work with a Pickering emulsion system, considering confocal image stacks of droplets stabilised by a mixture of positively and negatively charged particles. I investigate how machine learning can be used for exploratory analysis of this data by using unsupervised clustering. I also apply some traditional data analysis techniques to other images from the same system, and discuss the importance of considering these techniques alongside machine learning tools. I demonstrate the use of machine learning tools in Python, in particular k-means and hierarchical clustering, principal component analysis for dimension reduction, and t-distributed stochastic neighbour embedding for data visualisation. I use machine learning to group my image data into clusters of similar features based on different representations of the image, and try to interpret what the key features of different clusters might be. I assess whether these clusters reveal patterns relating to any known physical characteristics of the system and whether these patterns match those seen in my traditional analysis and in the literature for similar systems. I find that the particle arrangement and droplet roundness follow a different trend to the droplet characteristics investigated in the literature.

In Chapter 6 I return to the bijel system, this time considering images of bijels through which a bubble has passed during centrifugation of the samples. I demonstrate and compare the results of two ImageJ methods for segmenting images to identify the boundary between the bijel and the hole where the bubble passed through: the machine learning approach of trainable Weka segmentation and the traditional approach of seeded region growth. I then use these segmented images to process my bijel data, and use the machine learning regression tool of support vector regression to investigate how well the distance of an area of bijel from the bubble track can be predicted from the autocorrelation function of the image of that area. I demonstrate the use of support vector regression, cross-validation, and data manipulation tools in Python. I compare my preliminary findings to results in the literature where microrheology is performed on gels using bubbles, and draw some tentative conclusions about the local properties of bijels in comparison to their bulk properties.
Chapter 2

A guide to machine learning

2.1 Fundamentals of machine learning

Machine learning is, at its roots, simply a set of methods that allow us to automatically predict an output based on one or more inputs. These methods detect patterns in data and use these patterns to predict future data or otherwise assist with decision making [110]. The name comes from the way in which machine learning algorithms lead the computer to adapt their actions in order to obtain a more accurate outcome, much like humans and animals learn by remembering information, adapting knowledge based on new information and generalising knowledge to unseen problems [111].

Machine learning shares a number of similarities to statistical modelling, but there is one key difference: where statistical modelling involves rigorously fitting a deterministic function to relate a dependent variable to a number of independent variables, the goal of machine learning is simply to predict the output [10]. True statistical modelling requires the testing of hypotheses and an understanding of the significance of a result, but in machine learning such rigour is not required. The boundary between the two can be blurred: for example, linear regression is a commonly used statistical model [112, 113], but also can be viewed as a machine learning technique [8, 110].

In this chapter I outline a selection of algorithms and concepts that demonstrate the underlying principles of machine learning, including those used in this thesis. The majority of this content I learned from ‘An Introduction to Statistical
Learning’ [10], one of many excellent resources for any readers who want to understand the machine learning algorithms they might use.

2.1.1 Supervised vs. unsupervised algorithms

Machine learning is split into two main frameworks, supervised and unsupervised learning (Fig. 2.1). Both have a variety of uses depending on what data are available and what you want to learn from them.

**Figure 2.1**: A summary of the machine learning algorithms that will be discussed in this chapter and how they fit into the paradigms of supervised or unsupervised learning.

**Supervised learning**

Supervised learning is where your training data comprises of a set of examples labelled with correct responses, so the algorithm is provided with both inputs and the corresponding outputs. From this information, the algorithm aims to generalise the provided knowledge so that it can determine the correct output to all possible inputs. It is the most common type of machine learning, and is used in a variety of ways which will be discussed later. The method can be formalised as a function approximation: we assume the data fits a function, and use the labelled training data to estimate this function. This estimated function can then be used to make predictions of our output based on new inputs. Supervised learning can be split into two types, based on the type of output we are looking at. If the output variable is a set of discrete classes, we have a classification
problem; if instead we have a continuous numerical output, we have a regression problem.

Classification problems can be binary or multi-class and also include situations where labels are not mutually exclusive, i.e. each data point can have multiple labels and belong to more than one class. In the case of multi-label classification, the problem is best viewed as a series of binary class labels that can be predicted with a multiple-output model. Supervised machine learning is used for classification in a number of everyday applications, such as document classification for email spam filtering [114] and handwriting recognition for automatic sorting in postal systems [115]. In some highly risk-averse applications, it is common to introduce an extra “I don’t know” option for the output, used in ambiguous cases where the algorithm cannot classify an input with enough certainty.

Regression problems are, fundamentally, very similar to classification problems and the methods used are very similar. The key difference is that instead of a categorical output, the output is a number on a continuous scale. The algorithm predicts the most likely value of the continuous output. Supervised machine learning is used for regression in a wide range of applications, including the prediction of stock market prices, room temperatures, and the outcomes of medical procedures.

**Unsupervised learning**

Unsupervised learning is where the algorithm has no knowledge of the correct responses. Rather than searching for the correct output, the algorithm tries to identify similarities between inputs and uses this information to, for example, group together inputs that have features in common. The goal here is to discover interesting structure in the data, rather than to reach explicit predictions about new data. Unsupervised learning has a larger scope for applications than supervised learning, because it avoids the need to acquire expensive labelled data. Labelled data sets also tend to contain a much narrower set of information than unlabelled ones because the process of separating and labelling information requires selecting only those parts that can be summarised, and requires some human input.

There are a few different ways in which unsupervised learning is used to gain
information from data. One approach is to cluster data into groups that share similar characteristics. Clustering is used in applications such as star classification and targeted advertising. The first step in this approach is to assess the distribution of points across variable space, and make an estimate of the number of clusters present in the data. We can then calculate which cluster each point is most likely to belong to, sorting the data into distinct groups that share some common characteristics.

Unsupervised learning can also be used to discover latent factors in high-dimensional data (data described by a large number of variables) and therefore reduce the number of dimensions we have to work with. In real-world data there are often many fewer degrees of variability than there are variables, with a lot of correlation between different variables, and it is useful to project this data onto a lower-dimensional space that captures the same information. This is a useful exercise because it allows us to apply other statistical models to our data with better predictive accuracy, as well as learning about the relationship between the data and the different variables. The most common method used for this is called principal component analysis, which is discussed further in Section 2.2.6. Latent factor discovery is commonly used in a wide range of applications such as interpreting gene microarray data and creating computer-graphic animations from motion capture data.

The graph structure of data can also be found using unsupervised learning. This is useful when we measure correlated variables and want to know which variables are the most correlated with others. We represent variables as nodes and direct dependencies between them as edges, and from this graph we can learn about the data. In particular, this allows us to do two things: discover new knowledge about the data and how the variables are linked, and improve estimations used to model correlations and make predictions. The former motivation is of great interest in systems biology, and the latter is common in applications such as financial portfolio management and traffic flow modelling.

The final main application of unsupervised learning is in matrix completion. This is useful when we have a lot of missing data, and want to make plausible guesses for the values of missing entries. Among other applications, matrix completion is widely used to predict customer behaviours in a variety of contexts: from a very sparse matrix of what individual customers buy in a single transaction, or what they think of a particular product, we can deduce what other products they are likely to buy or what they are likely to think of other products. A notable
example of this is the competition by the movie streaming service Netflix, which
callenged research groups to beat their current algorithm for predicting how
viewers would rate unseen movies based on their opinions of the movies they
have seen along with the ratings of other viewers.

Other types of learning

There are also a few other types of machine learning that do not fall within
the supervised or unsupervised label. Reinforcement learning is somewhere
in between supervised and unsupervised learning, where the algorithm is told
whether or not its answer is correct but not what the correct answer is. Obviously
in the case of a classification problem with only two classes, this is the same as
supervised learning, but in all other cases it is a distinct approach. Evolutionary
learning is somewhat similar, and is inspired by evolution in biological systems
and the concept of survival of the fittest. Each solution the algorithm arrives
at is given a fitness score, which corresponds to how good the solution is. This
score is used to inform further iterations of the algorithm in order to improve the
outcome.

2.1.2 Parametric vs. non-parametric models

One factor to consider when choosing a machine learning model is how much data
is available, and how the model scales with the data. Non-parametric models offer
the most flexibility as they avoid making assumptions about the data. However,
these models quickly become computationally intractable for high-dimensional
inputs. Section 2.2.1 highlights an example of this “curse of dimensionality” in
the k-nearest neighbours algorithm.

Parametric models have a fixed number of parameters so are faster to use,
especially for large data sets, and avoid the “curse of dimensionality” associated
with their non-parametric counterparts. The downside is that we avoid this
problem by making assumptions about the distribution of the data, which reduces
the flexibility of the model as it can only be used for data that fit those
assumptions. The specific assumptions vary between models, for example a
simple linear regression (Section 2.2.2) is a parametric model that assumes a
linear relationship between variables.
2.1.3 Eager vs. lazy learners

Another way to distinguish between different types of machine learning algorithms is to consider at what point in the learning process the algorithm abstracts from the data. Some algorithms, which we call eager learners, start learning as soon as they receive training data. Lazy learners, on the other hand, simply store the training data after minimal processing, and save any learning until test data are provided [116].

It can clearly be seen that algorithms that are eager learners take more time to run during training, but lazy learners are slower during testing. As such, when choosing between these types of algorithms one should consider the frequency at which new data is anticipated compared to how often a trained model would be used for testing. If the known training data is updated often, there is little point in pre-training the model on the known data as this will need to be done too frequently. If the model is used for testing often but trained rarely, it takes much less computational time overall to use an eager learner. However, lazy learners can save computation time by training only on a relevant subset of the data, if a local prediction is all that is needed.

2.1.4 Errors and cross-validation

Choosing an error measure

When using machine learning, it is of the utmost importance that we understand the errors associated with our model. There are a number of different ways to measure the performance of a model, depending on the type of data available. For supervised learning, we often look at training error (how well the output fits to the training data), validation error (how well the output fits to data set aside for testing while tuning the model), and test error (how well the output fits to data set aside for assessing the performance of the final tuned model). We have to be careful which measures we choose because it is very easy to obtain a result that doesn’t truly represent the efficacy of the model, if you look at the wrong type of error. For example, when choosing the best from a number of supervised machine learning algorithms it would seem natural to calculate the errors by assessing how well each trained model fits to the data on which it was trained (the training error). However, this approach forgets one of the key components
of machine learning: generalisation. We don’t actually care how well the model fits to the data on which it was trained, but rather how well it will predict the outputs from new, unseen data (the validation or test error).

To this end, the application of machine learning to a problem generally requires splitting the available data into two sets: one for training and one for validation. The algorithm is first trained on the training data, then the final model output from this is used to predict the outcome of the validation data. The error on these predictions is sometimes called the validation error, and can be used to compare models. This approach is required in order to ensure that the model is not overfitted to the fluctuations and quirks of a specific data set and can therefore be effectively used to predict future data from outside the training set. Sometimes, for example in competitions for developing the best machine learning approach to a problem, there is a third set of data set aside for a final test of a method. In such cases, this test data set is unseen by the people working on the problem, and all the development and testing is done on the training and validation sets. This allows for further verification that there is no bias in the model towards the training and validation data.

The null error

When assessing the quality of a model’s fit to data, we must consider the fact that the level of error we can accept as ‘good’ depends on the properties of the data. In particular, when considering a classification problem where one class contains more members than the other, we can see that an error that seems reasonable could be achieved by simply sorting all examples into the larger class. We therefore introduce the null error, which is defined as the error we would get if we sorted everything into this larger class. Any classification error greater than the null error means that there is a simple solution that performs better than the model, so the model is not worth using.

Bias-variance trade-off

We can view the problem of overfitting to the training data as a balance between two properties of the model fit: bias and variance. Bias is the error that comes with applying an approximation to a real-life problem that is much more complicated than any model we can realistically use. Variance is the amount that
the model’s fit would change if we were to change the training data set. Both of these are quantities that we want to minimise, but as a general rule as we decrease one we increase the other: a more complex model requires fewer approximations but can more easily be influenced by noise in the training data. This means that when we tune parameters in our model to minimise the error rate, we are generally looking to find the balance point between bias and variance [10].

Cross-validation

![Figure 2.2](image.png)

Figure 2.2: An illustration of 5-fold cross-validation. The data is randomly split into 5 equally-sized sections (folds), and each fold takes a turn being used as a test set for validation while the others are used for training.

When the amount of data available is limited, cross-validation can be used to make the most of a smaller amount of data by avoiding the need to set aside a large chunk for model validation. K-fold cross-validation (see Figure 2.2) requires splitting the data into \( k \) equally sized sections, or folds. One of these folds is set aside for validation and the model is trained on the remaining \( k-1 \) folds. The trained model is then tested on the fold that was set aside and the predictions from this fit is stored. The number of folds \( k \) can be tuned to ensure that the testing fold contains enough data for effective testing without having to train the model too many times, and 5-fold or 10-fold cross-validation are commonly used values.

The process is repeated \( k \) times such that each fold has been used exactly once as a validation set. The results of all \( k \) folds are easily combined: we divide
the sum of all incorrect predictions across all test sets by the number of data points in the data set. As each data point is used exactly once for validation (and \( k - 1 \) times for training) cross-validation gives a final error effectively based on the whole data set as a validation set, avoiding the problem of over-fitting because we utilise random subsets of the data. In general, cross-validation allows us to easily tune the flexibility of our model in order to optimise the bias-variance trade-off.

Other types of cross-validation exist, such as leave-one-out cross-validation which is effectively \( k \)-fold cross-validation where \( k \) is equal to the number of data points in the set \( N \). The main distinction between different types of cross-validation is whether the method is exhaustive, where all possible ways to divide the data are tested, or not. Of the two examples here, leave-one-out is exhaustive whereas \( k \)-fold is non-exhaustive because we do not try all possible permutations of forming a test set of our chosen size \( N/k \). Cross-validation techniques can also be performed multiple times, averaging the results, which can help us avoid any potential problems that come with the non-exhaustive random selection of folds. Cross-validation of some kind is often used to tune model parameters within a single machine learning method, and can be included in the algorithm.

### 2.1.5 Model validation

When assessing the quality of a model there are a number of ways to further understand how well it represents your data, aside from calculating the error rate of your model on validation and testing data sets.

**ROC curve**

The receiver operating characteristic (ROC) curve is a plot that can be used to explore the diagnostic ability of a binary classification \([117]\). It is a plot of the true positive rate (or “hit rate”),

\[
\frac{\text{True Positive}}{\text{Total Positive}} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}},
\]

against the false positive rate (or “false alarm rate”),

\[
\frac{\text{False Positive}}{\text{Total Negative}} = \frac{\text{False Positive}}{\text{True Negative} + \text{False Positive}}.
\]
where the denominator in each expression is the total number of data points in the given category (e.g., total positive is the number of data points that should be classed as positive, which includes the ones correctly classed as positive and the ones that should have been classed as positive but were incorrectly classed as negative). An example ROC curve can be seen in Figure 2.3. We get a curve, rather than a single point, in the case where the output of our algorithm is a probability of the data point being a certain class. In this case, we can choose a cutoff value for this probability output at which we decide that the data point belongs in the class, and varying this cutoff gives us a series of points to form the curve. In the case of perfect classification, the curve will consist of straight lines from (0, 0) to (0, 1) to (1, 1), and in the case of random classification, it will follow the $y = x$ line.

![Figure 2.3: Example of a Receiver Operator Characteristic (ROC) curve. The area under the curve (AUC) is a measure of the quality of the result.](image)

The closer the curve is to the ideal case of perfect classification, the better the performance of the model. To this end, we can quantify the performance of the model by the area under the ROC curve: a value of 1.0 is perfect classification, and any value less than 0.5 is worse than random chance and therefore not a useful model. The curve can also help us determine which probability cutoff should be used to balance true and false positives: the value that gives the point closest to
the top left corner of the plot is usually a good choice.

This curve is useful because it allows us to understand the trade-off between the different types of incorrect answer we can have. It is important to understand how the true and false positives compare to the true and false negatives. In some cases, false negatives can be a lot more detrimental than false positives, or vice versa. For example when conducting medical tests, a false positive is often more acceptable than a false negative, as further medical investigation is more likely in the case of a positive test result so the mistake would be easier to identify.

Random re-labelling

Another method of comparing the performance of a model to random chance is to test it on random data. This can be done by randomising the labels attached to your training and/or validation data, and training the model on this new data set. Repeating this allows us to plot a histogram of the errors our model gives on randomly labelled data with the same inputs. Then we can check that our actual error from the true labels is statistically better than random chance. If not, we are likely just fitting to the noise in the data rather than real information. This can be seen in Figure 2.4, which shows that the model error trained on the properly labelled data is significantly lower than could be attributed to random chance based on the errors obtained from random labels.

![Histogram of error rates](image)

**Figure 2.4:** An example histogram showing how the machine learning model performs well only when the samples are correctly labelled. The histogram shows the distribution of error rates for models trained on randomly labelled data, and the line shows the error for the model trained on the true data.
2.2 Machine learning algorithms

In this section, we outline some of the commonly used machine learning algorithms and how they work.

2.2.1 K-nearest neighbours classification

The k-nearest neighbours (KNN) algorithm is one of the simplest and therefore most commonly used machine learning classification algorithms. It can be used in situations where we want to sort data into different categories, or classes, based on information we already know about the data: for example we could predict the variety of a flower based on the length and width of its petals. This method classifies every point in the input variable space (e.g. all possible petal lengths and widths), visualised by each small dot in Figure 2.5 based on a “vote” of its nearest k neighbours of known classification (e.g. flowers of known variety that have been measured), visualised by the points shown with open circles in Figure 2.5. The class that is the most commonly represented in these neighbours is the class that we predict a new measurement at that point in variable space to belong to: we expect points in the bottom right corner of the plots in Figure 2.5 to belong to the blue class and points in the top left corner to be red. This method can be expanded to any number of input variables, although it works better for fewer variables. A number of metrics can be used to determine the distance between points (and therefore which neighbours are the nearest), but the most common is the Euclidean distance: the distance between points \( \vec{p} = (p_1, p_2, ..., p_d) \) and \( \vec{q} = (q_1, q_2, ..., q_d) \) in \( d \) dimensional space given by

\[
d(\vec{p}, \vec{q}) = \sqrt{\sum_{i=1}^{d} (q_i - p_i)^2}.
\] (2.3)

The number of neighbours \( k \) is tuned to improve the fit of the model. As can be seen in Figure 2.5, a higher value of \( k \) results in a smoother decision boundary between the different classes. A lower value of \( k \) will always give a lower training error; this can clearly be seen in the extreme case of \( k = 1 \) (Figure 2.5(a)) where every training point is correctly classified by definition since its nearest neighbour is itself. However, as discussed in Section 2.1.4 this is not a very useful measure
of the performance of an algorithm. If instead we assess a validation accuracy (defined as the fraction of data points in a validation set that are correctly classified) as a function of \( k \), we expect a curve with a clear maximum such as in Figure 2.6. The best value of \( k \) can therefore be chosen as the one that minimises the error, or maximises the accuracy: in the example case here, a value of \( k \) between 35 and 37 would yield the best result.

In order to avoid inadvertently weighting some variables more than others, this algorithm requires that all variables are normalised so that they are of comparable scale. As long as this is done, the absolute scale of the variables is unimportant. Variables are therefore often normalised by dividing by the mean value of the variable.

K-nearest neighbours is a non-parametric model, so its usefulness can quickly diminish with high dimensional inputs. This can be understood by considering the size that a \( d \)-dimensional hypercube must be in order to contain, for example, the nearest 1% of known data points to a test point [110]. The expected edge length of this hypercube, assuming a uniform distribution, is \( L = f^{1/d} \) where \( f \) is the fraction of points within the hypercube. In a reasonably sized problem, say \( d = 10 \), to include just 1% of the data in this cube requires its length to be 63% of the entire range of the data. Our aim to find local behaviour via the use of nearest neighbours is no longer local at all. This is demonstrated in Figure 2.7 where a set of randomly distributed points become much more spread out as the number of dimensions increases and a larger region is therefore required to capture the same number of points.
Figure 2.6: An example of KNN classification accuracy as a function of number of neighbours.

Figure 2.7: A demonstration of the curse of dimensionality. The same 30 random points have been plotted in (a) one, (b) two and (c) three dimensions. As the number of dimensions increases, the points become more spaced out and therefore a larger region is required to capture the same number of points.
Overall, the k-nearest neighbours algorithm is simple and can give very good performance. With few enough dimensions, a well-chosen distance metric and enough labelled training data \((N \to \infty)\), its performance can tend to within a factor of 2 of the theoretical best performance possible \([119]\). Despite its simplicity, this algorithm can produce highly complex classification boundaries and can therefore be a deceptively powerful tool.

### 2.2.2 Linear regression

Linear regression is one of the most widely used models for regression, and will no doubt be familiar to the reader as an analysis tool if not a machine learning method. It is a parametric supervised algorithm that can be used to gain a greater understanding of relationships between variables, and to form predictive models based on these relationships. A number of machine learning approaches can be seen as simply altered versions of linear regression.

In linear regression we assert that the (continuous) output \(y\) is a linear combination of the predictive input variables \(X\) plus a residual error \(\epsilon\) to account for imperfect fitting:

\[
y = \beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n + \epsilon_g.
\]  

The residual component is usually assumed to have a Gaussian distribution with a fixed width, so here we denote it as \(\epsilon_g\). The output \(y\) could be a variable such as the predicted sales generated by an advertising campaign, where the predictive inputs \(X = (x_1, x_2, \ldots, x_n)\) might be the budgets for TV, radio and social media advertising. Although the method uses a linear equation, non-linear relationships can be modelled using new inputs that are non-linear combinations of the originals. This makes linear regression a very useful tool, one that should not be discounted just because it is simpler and more familiar than other forms of machine learning.

The values of the coefficients \(\beta\) are estimated such that the linear model fits the data as well as possible. Although there are many ways to measure how well the model fits the data, this is most commonly done using the least squares method,
in which we seek to minimise the residual sum of squares (RSS), defined as

\[ \text{RSS} = \varepsilon_1^2 + \varepsilon_2^2 + \ldots + \varepsilon_N^2, \quad (2.5) \]

where

\[ \varepsilon_i = y_i - \hat{y}_i \quad (2.6) \]

is the residual on the \( i \)th data point where \( \hat{y}_i \) is the predicted value of the output with a known value \( y_i \). It can be shown that, for a simple linear regression with one predictive input variable, the RSS is minimised by

\[ \hat{\beta}_1 = \frac{\sum_{i=1}^{N}(x_i^1 - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{N}(x_i^1 - \bar{x})^2}; \quad \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}_1, \quad (2.7) \]

where \( x_i^1 \) and \( y_i \) are the known values of the input \( x_1 \) and the output \( y \), respectively, for the \( i \)th data point. \( \bar{x}_1 \) and \( \bar{y} \) are the mean values (across all data points) of the known inputs and outputs. This is the calculation performed in all simple line-fitting packages. For problems where we have more than one predictor this simple analysis can be extended to a multivariate case, but the estimates for these values of \( \beta \) are much more complicated and so are not discussed here. It is unsatisfactory to simply perform a separate single-variable regression for each variable, as this makes it very difficult to give useful predictions and, more importantly, neglects the effects of any correlations between the different input variables. Fortunately, these coefficients are estimated computationally so we need not worry about their algebraic complexity.

**Variable reduction for multivariate regression**

In the case that there are too many predictive variables in a problem it is useful to have tools that allow us to reduce the number of variables without adversely affecting the predictive performance of the final model. This allows us to avoid the overfitting that is common when using a model with too many variables. There are a number of ways to select which variables to include.

Most simply, if we have fewer predictive variables than training data points, we can fit the model multiple times using different combinations of variables and then assess the performance of each one. To quantify this performance we can use one
of variety of measures. One approach is to use the cross-validated prediction error (Section 2.1.4), which measures the predictive accuracy of the fitted model. Alternatively, we can use something like the F-statistic:

\[
F = \frac{(\text{TSS} - \text{RSS})/d}{\text{RSS}/(n - d - 1)}
\]

(2.8)

where \( n \) is the number of data points and \( d \) is the number of predictive variables. This involves computing the total sum of squares:

\[
\text{TSS} = \sum_i (y_i - \bar{y})^2,
\]

(2.9)

where \( \bar{y} \) is the mean value of the output \( y \), and the residual sum of squares (Eq. 2.5). The F-statistic can be thought of as the ratio between the explained and unexplained variance in the data, or the ratio between (what the model classes as) signal and noise in the data.

When there is no relationship between the response and the predictors we expect a value of \( F \approx 1 \), and larger values of \( F \) (often in the hundreds) indicate stronger relationships between the response and the predictor. This measure is similar to the \( R^2 \) value of a linear fit, but is much more useful for comparing the performance of models of different complexity, such as the case presented here where we are comparing a multiple linear regression with different numbers of variables.

We can use our chosen measure to compare the models and determine which is the best suited to the problem. However, to most thoroughly use this approach, we need to train and evaluate \( 2^d \) models for \( d \) variables, which quickly becomes intractable for large numbers of variables.

More automated approaches can offer a better solution if there are more predictive variables than training points, or if we wish to reduce the amount of human input required. Shrinkage methods, including ‘the lasso’ [120] and its predecessor ‘ridge regression’, involve fitting a single model with all \( d \) predictive variables, then reducing the coefficients of some of them towards zero. In the case where we allow the coefficients to reach zero (the lasso), this enables us to reduce the number of variables used to fit the model.

The lasso is an altered version of a least squares fit, wherein we seek to minimise
an adapted version of the residual sum of squares (RSS):

\[
\text{RSS} + \lambda \sum_{j=1}^{d} |\beta_j| = \sum_{i=1}^{n} \left( y_i - \beta_0 - \sum_{j=1}^{d} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{d} |\beta_j| \quad (2.10)
\]

by choosing the optimal values of \( \beta \) (which we call \( \hat{\beta}^R \)). We use \( \lambda \) as a tuning parameter, which determines the strength of the shrinkage penalty \( \lambda \sum_{j=1}^{d} |\beta_j| \) that encourages the coefficients of each variable towards zero. In practice, the balance between the shrinkage penalty and the need to fit the data means that the less important variables (the ones that explain less of the variation in the data) are pushed towards zero first.

The value of \( \lambda \) dictates how quickly the coefficients of the less important variables are forced towards zero, and is most easily tuned using cross-validation. If there exists a sparse model that explains the data with a very small subset of the available variables, the lasso is a very efficient way of finding it. However, if no such model is possible it may not reduce the number of variables as much as desired.

A final method for reducing the number of variables in a linear model is to reduce the dimensions of the problem by projecting the predictors into a lower-dimensional subspace, meaning that we represent the same data points with fewer variables. We then fit our model using this smaller number of composite variables which are chosen to capture the most variance in the data in the fewest variables. This is usually done via an unsupervised learning technique called principal component analysis, which is discussed in Section 2.2.6.

2.2.3 Logistic regression

Logistic regression [121] is very similar to linear regression but is used for binary classification, generally in the case where the question is ‘does this item belong in this given class or not?’’. Despite its name, it is not a tool for regression. In logistic regression, we classify points based on the probability \( p(X) \) that a data point with input variables \( X = (x_1, x_2, ..., x_n) \) belongs to a given class. Using the same example case as earlier, we could be looking to determine how likely a flower with petal length \( x_1 \) and width \( x_2 \) is to be one of variety \( A \). We achieve this by by fitting a linear equation (i.e. performing a linear regression) and then
passing it through a logit function \( \logit(p) = \log \left( \frac{p}{1-p} \right) \) that ensures the output is bounded between 0 and 1, as a probability should be. The combined function,

\[
\log \left( \frac{p(X)}{1-p(X)} \right) = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n + \epsilon_b, \tag{2.11}
\]

or

\[
p(X) = \frac{e^{\beta_0 + \beta_1 x_1 + ... + \beta_n x_n}}{1 + e^{\beta_0 + \beta_1 x_1 + ... + \beta_n x_n}} \tag{2.12}
\]

is fitted to the training data using the maximum likelihood method (Equation 2.13) to find values of \( \beta \) that fit the training data provided \([10]\), and therefore the probability \( p(X) \). Both logistic and linear regression belong to a broader class of ‘generalised linear models’, a generalisation of linear regression that allows us to assume different probability distributions for the output variable and thus different forms for the residual error. Here we assume a Bernoulli distribution for the residual error, which is a discrete distribution that takes the value 1 with probability \( p \) and 0 with probability \( 1-p \). This is much more appropriate than a Gaussian in classification applications such as this, as our response variable is binary.

The maximum likelihood method is an alternative to the least squares method (in fact, when considering linear regression, the least squares approach is a special case of maximum likelihood \([10]\)), and has better statistical properties than the least squares method when applied to logistic regression. We find values for the coefficients \( \beta \) such that the predicted output for each data point corresponds as closely as possible to its known classification (1 or 0). This can be done by finding values that maximise the likelihood function:

\[
I(\beta_0, \beta_1...\beta_n) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1 - p(x_{i'})) \tag{2.13}
\]

where \( I \) is the likelihood function, \( x_i \) is the estimated probability that data point \( i \) belongs to the class, and \( y_i \) is the true probability of data point \( i \) belonging to the class. The first sum is over all points that truly belong to the class, and the second sum is over all points that truly do not belong to the class. By definition, \( y_i \) must equal either 1 or 0 because the point is known to be either in the class or not. Therefore, the first product is a measure of how well the true positives are classified by the algorithm, and the second product a measure of how well the true negatives are classified.
Figure 2.8: An example single-variable logistic regression result. In red we see the estimated probability that a point with input value $x$ belongs to the given class, and in grey we have a histogram of the true classifications as a function of $x$.

Figure 2.8 shows an example output from a logistic regression classification with a single input variable and a single output class. For each value of the input variable (x-axis), there is an associated probability (shown in red) that the data entry belongs in the given class, based on the input data given to the algorithm (shown in the grey histograms, e.g. values of $x$ between 28 and 29 belonged to the class on two occasions and didn’t belong on four occasions). As this is a supervised method and our training data are labelled, we can also plot a histogram of the true frequency of each classification (belonging to the class or not), shown in grey. Although this example only shows a one-dimensional input variable space, the same approach can easily be extended to multi-variate inputs: it is just not possible to visualise them in such a clear way.

The logistic regression algorithm is in some ways more versatile than k-nearest neighbours because it can give a probability of the sample belonging to one class or not, which allows a classification to be tuned to achieve a suitable balance of false positives and false negatives depending on the needs of any application. Logistic regression is therefore one of the algorithms where ROC curves (Section 2.1.5) can be useful. The regression coefficients $\beta_i$ for each input variable $x_i$ can also be used to consider the importance of each of the variables in the classification decision:
this can be done directly from the coefficients or by taking the exponential of the coefficient to find the odds ratio. The odds ratio for each input variable describes the probability that a high value of the variable is associated with membership of the given class and ranges from 0 to $\infty$, with these extreme values suggesting that the input variable is a strong indicator of membership in the class and values close to 1 suggesting a weak indicator.

### 2.2.4 Decision trees

The decision tree algorithm can be used for both classification and regression \[122\], and involves splitting the available variable space into different regions that share similar outcomes. To make predictions for a new data point, we look at which region it is in and predict its outcome to be the mean value or modal class of the training observations in that region. As the general basis of decision trees can be applied to either classification or regression problems with very little alteration, here we focus on classification trees and consider the mode rather than the mean. Decision trees are useful because they are simple and very easy to interpret. Unfortunately they do tend to under-perform compared to other supervised learning approaches, often because by design they must always split the data orthogonal to the axes, as illustrated in Figure 2.9. We will discuss methods for improving their performance (at the cost of some interpretability) by combining multiple trees.

In order to form a classification tree, we take an approach known as recursive binary splitting, wherein we start at the ‘top’ of the tree and split the predictor space in two, such that the classes are best separated. We then repeat the splitting process down both possible branches, and stop ‘growing’ the tree when further branching leads to no significant improvement in the classification error \[10\]. In the example shown in Figure 2.9, where we are trying to classify different species of penguin based on their flipper length and bill length, the first split based on flipper length separates the Gentoo penguins from the other two species, creating a good separation between ‘Gentoo’ and ‘not Gentoo’. The other two species are then separated from each other by the second split based on bill length, so now the species are separated and we have three regions which are each very rich in a single species of penguin. The final tree can be plotted to show the splits that have been chosen as well as additional useful information about the contents of each node, for example in Figure 2.10. This figure was plotted using the iris
Figure 2.9: An example of the decision boundary drawn by a decision tree classifier. In this case, the classifier is applied to palmerpenguins dataset. Taken from [123] under fair use.

Figure 2.10: An example decision tree classification using the Iris dataset. Each branch shows how the algorithm chose to split the variable space during recursive binary splitting. Each node shows the predicted class, the probabilities of a point in that node belonging to each possible class, and the percentage of all data sorted into that node.
dataset from the UCI Machine Learning Repository [124], which is commonly used for tutorials and examples. In this dataset, four measurements (the petal and sepal length and width) of many examples of three species of iris flowers (setosa, versicolor and virginica) are recorded. These measurements can be used to predict which iris species a particular flower belongs to.

As the algorithm chooses the best option at each decision step without considering the wider picture, it is known as a greedy algorithm [125]. This means that decision trees often miss the best overall combination of variable space splits, as they settle instead into a more local best solution. This leads to a tendency for decision tree algorithms to have very high variance, so the results are very dependent on the training data. If we were to split a training data set into two random halves, the results from training a decision tree on each half would likely be very different. In order to combat this problem, we can use more advanced decision tree techniques. Examples include bagging, random forests, and boosted decision trees, all of which add to the basic method of recursive binary splitting.

**Bagging**

The simplest of these methods is called bootstrap aggregation or bagging [126]. In this method, for a training set with \(N\) data points, we randomly select, with replacement, \(N\) entries from the training set and use this to train the model. When selecting with replacement, some entries are selected more than once and some are left out, allowing us to avoid the problem of high variance (by repeating this process and combining the results of different random data sets) without reducing the size of each training data set. The final step of this method is aggregation, where the multiple trees are used to give one result by taking the average (mean for regression, mode for classification) of the prediction of each tree.

**Random forests**

The random forests method [127] takes the randomisation introduced by bagging one step further by also randomly selecting a small subset of the available input

\footnote{Not to be confused with the eager learners discussed in Section 2.1.3, which are often also called greedy learners. Here we aim to avoid the confusion by referring only to eager learners and greedy algorithms, but never greedy learners, but those undertaking further research into the topic should take care.}
variables at each node and making the splitting decision only on this subset of variables. At subsequent nodes, the number of variables in the subset remains unchanged but another random selection is made to choose which ones are included. We can tune the performance of the random forest model by tuning the size of this variable subset. A smaller subset reduces the correlation between any two trees in the forest, which improves the performance of the method, but it also reduces the performance of each individual tree and therefore the forest. There is a trade-off to be made between the benefits of both larger and smaller variable subsets, and this can be tuned (for example using cross-validation) if desired.

**Boosted decision trees**

Boosting decision trees works in a similar way to bagging, in that they both involve creating multiple versions of the original training data and training decision trees on each one before combining the results to create a predictive model. However, where bagging involves training each tree independently, to make a boosted decision tree we train the trees sequentially, using information from the previous models to help grow the next tree. The end effect is similar to the outcome if we were to fit a large single decision tree to the data, but bypasses the problem of over-fitting to the training data building up a large tree from a series of smaller ones trained on slightly different versions of the training data.

When using a boosted decision tree for classification, information is passed to the new tree using a weight function from the previous tree which describes its accuracy in classifying each node in the data. This allows us to focus on improving performance in areas where it was previously poor. In general, approaches that learn more slowly, like this one, tend to perform better so we often slow down the learning process even further by shrinking each new tree before adding it to the model.

2.2.5 **Support vector machines**

**Maximal margin classifier**

The support vector machine (SVM) algorithm is a generalisation of a simple maximal margin classifier, so this is where we start our discussion. This classifier
separates the $d$ dimensional variable-space of a problem into different classes using a flat $(d - 1)$ dimensional hyperplane, where $d$ is the number of input variables. There are often a large number of ways in which a single hyperplane can be drawn while still separating two classes, as can be seen in Figure 2.11 so the problem is not so simple as finding any separating hyperplane. In fact, if our data can be perfectly separated by a hyperplane, then there are an infinite number of hyperplanes that can separate the classes. In order to choose which one best separates the two classes, we define the maximal margin hyperplane. We calculate the perpendicular distance from each data point to the hyperplane, and choose the hyperplane that maximises the width of margin that can be drawn either side of the hyperplane without meeting any data points, i.e. we maximise the distance to the closest point on either side. This approach can be generalised to more than two classes by drawing multiple hyperplanes to separate the different classes. Either each category has an associated hyperplane that separates it from the points that are not in that category, or each pair of categories has a hyperplane that separates them.

![Figure 2.11: An example of 2-class (red/green) data that is perfectly separable by a number of different hyperplanes, such as the ones shown in black here.](image)

We can use this optimally separating hyperplane to classify new data, under the assumption that the classifier with a large margin on the training data will also have a large margin on the new data. This is not always the case, and the
maximal margin classifier can therefore have a tendency to overfit when we have a large number of input variables. This classifier can also be very susceptible to noisy data, where adding a single point to the training data could drastically change the location of the maximal margin hyperplane. Another clear limitation of this classifier is the fact that it is only useful when the data are separable by a hyperplane.

**Support vector classifier**

To avoid some of these issues, we can relax the requirement that the hyperplane perfectly separates the variable space. We introduce a soft margin, in which we allow some points to exist on the wrong side of the margin or even the hyperplane. In this approach, called the support vector classifier, we give ourselves a ‘budget’ $C$ that dictates the number and severity of misclassifications (i.e. points on the wrong side of the margin) allowed, and find the maximal margin hyperplane subject to this allowance.

As well as allowing us to classify data that aren’t perfectly linearly separable, this approach can have a reduced tendency to overfit to the training data, because by increasing $C$ we make the position of the plane more stable. Conversely, the smaller $C$ is the narrower the margin, making more accurate predictions possible (at least on the training data) as we allow for fewer points to be on the wrong side of the margin. This is a classic example of the bias-variance trade-off discussed in Section 2.1.4: the more stable plane introduces a low variance, but the narrower plane allows for fewer misclassifications and therefore has a lower bias. In practice, the value of $C$ is often chosen using cross-validation.

Interestingly, only observations that either lie on the margin or are on the wrong side of it have any impact on the position of the hyperplane. The data points that are clearly correctly classified have no effect on the support vector classifier. We call the points that do have an effect ‘support vectors’, and the fact that the classification is made based on the support vectors only is what gives this method its name. The importance of support vectors makes sense in the context of the bias-variance trade-off in this approach: when we have a large $C$ and a wide margin, we also have a lot of support vectors influencing the position of the hyperplane, so the variance is lower but the bias is potentially higher (and vice-versa for small $C$).
Support vector machine

The support vector machine is a further generalisation of this method that introduces the ability to capture non-linear decision boundaries \[129\]. We do this by enlarging the variable space, much like how we can add higher-power terms to perform a linear regression on non-linear data (Section \[2.2.2\]). We can choose from a variety of different kernels: functions that we use in place of our observed data points, and that quantify the similarity between two observations. We can actually specify the linear support vector classifier as one such kernel:

\[
K(x_i, x_{i'}) = \sum_{j=1}^{d} x_{ij} x_{i'j},
\]

(2.14)

where \(x_i\) is the \(i\)th training observation, \(x_{ij}\) is the component of that observation in the \(j\)th variable, and we sum over all the variables in the problem. Some other widely-used examples of kernels include the polynomial kernel of degree \(p\):

\[
K(x_i, x_{i'}) = (1 + \sum_{j=1}^{d} x_{ij} x_{i'j})^p
\]

(2.15)

and the radial kernel:

\[
K(x_i, x_{i'}) = \exp\left(-\gamma \sum_{j=1}^{d} (x_{ij} - x_{i'j})^2\right),
\]

(2.16)

both of which (along with many others) add a great deal of flexibility to an otherwise rigidly linear method, making the support vector machine much more useful for a wide range of applications. We can formalise our support vector machine for any kernel \(K\) as the function:

\[
f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i K(x, x_i)
\]

(2.17)

that classifies a test point \(x\) based on the sign of \(f(x)\), where \(\beta_0\) and \(\alpha_i\) are the parameters used to fit the model and \(\mathcal{S}\) represents the set of observations that are support vectors. Strictly we would sum over all observations, but the value of \(\alpha_i\) is non-zero only for support vectors \[10\] so we can reduce the number of calculations.
required by only including support vectors in the sum. While the sign of $f(x)$
gives us our classification, its magnitude is also useful as it gives information on
how far away the point is from the decision boundary and therefore how sure we
are of the classification.

Support vector regression

Support vector machines can be adapted for use in regression problems. In the
same way that support vector machines find the optimally separating hyperplane
while allowing a margin of error of a certain size, support vector regression
finds the hyperplane that best fits the data within a margin of error as shown in Figure 2.12. In the case of the support vector machine we wanted to
ensure maximal separation between classes by maximising the distance from the
hyperplane to the data of different categories, subject to a given allowance for
some data points to be misclassified. To convert this approach to regression we
want to minimise the distance from the hyperplane to each data point, and we set
an allowance for imperfect fitting by setting a margin either side of the hyperplane
in which data points are viewed as being on the hyperplane, as demonstrated in
Figure 2.12. Each point outside of this margin is not on the hyperplane and its
distance from the hyperplane is measured as the shortest distance from the point
to the edge of the margin, rather than to the plane itself.

Figure 2.12: A plot to demonstrate support vector regression. Only the
points outside the dotted lines are considered when choosing the line of best fit, shown by the solid line, and the error in the
fit considers only the distance from each point to the dotted lines rather than the solid line.
One key difference between this method and linear regression is the fact that, by introducing the margin in which errors are not considered in the minimisation, the method is minimising a different type of loss function. This means that it can more easily fit smooth functions to noisy data and, most importantly, it gives a result that is less susceptible to changing if small changes in the data are made. We might be tempted to think that the use of kernels also differentiates between linear regression and support vector regression, but such kernels could actually be applied to a linear regression if desired.

2.2.6 Dimension reduction methods

There are a number of situations where we may want to reduce the number of dimensions in a problem. In the case where we have more predictive variables than data points, many of the algorithms described above cannot be used, and others will severely overfit to the training data. The previously discussed methods such as lasso and ridge regression cannot be applied in such cases because they require the fitting of an algorithm that cannot cope with more variables than data points. Dimension reduction is also useful for the visualisation of data, as the multivariable space can be mapped into two dimensions for plotting.

Component analysis methods

So far, our discussion has mainly focussed on supervised learning methods, but a lot of these require the selection of a set of variables that characterise the features in the data. In some applications, this selection can be accelerated by using unsupervised component analysis methods to select a representative basis set of variables, the most common of which is principal component analysis (PCA) \[131\].

When we have many variables but some of them are correlated, PCA allows us to select a smaller set of variables that represent the same information and can be used to further understand the data. The method allows us to represent our data in a lower-dimensional space that is much easier to visualise and can help to avoid overfitting, by seeking a small number of dimensions that are as ‘interesting’ as possible. In effect, the process is equivalent to finding the eigenvectors and eigenvalues of the covariance matrix of the variables. We measure how interesting a dimension is based on how much the observations vary along that dimension,
and find the linear combinations of features \((X_1, \ldots, X_d)\) with the largest variance. Unlike in the other sections, when discussing component analysis we denote each input variable as \(X_i\) rather than \(x_i\), as the latter is used for another purpose later in this explanation. We normalise the linear combinations

\[ Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + \ldots + \phi_{d1}X_d, \quad (2.18) \]

where \(\phi_{11}, \ldots, \phi_{d1}\) are called the loadings of \(Z_1\) the first principal component, such that:

\[ \sum_{j=1}^{d} \phi_{j1}^2 = 1. \quad (2.19) \]

This normalisation ensures that we do not accidentally create an arbitrarily large variance simply by scaling up all of the variables.

The easiest way to find the principal components, i.e. the linear combinations of variables that provide maximal variance in the observations, is to first scale the data such that every variable has a mean value of 0. This is a valid approach to take because we are interested only in the variance of each variable, not the absolute values. We then calculate the value of a linear combination of features for each data point \(i\):

\[ z_{i1} = \phi_{11}x_{i1} + \phi_{21}x_{i2} + \ldots + \phi_{d1}x_{id}, \quad (2.20) \]

which we refer to as the scores of the first principal component. We choose \(\phi\) such that the variance in this score is maximised:

\[ \max_{\phi_{11}, \ldots, \phi_{d1}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left( \sum_{j=1}^{d} \phi_{j1}x_{ij} \right)^2 \right\} \quad (2.21) \]

subject to the previous constraint (Eq. 2.19). When we then find the second principal component \(Z_2\) we consider only combinations that are uncorrelated with \(Z_1\), and so on to find subsequent principal components. This constraint turns out to be equivalent to only considering a principal component loading vector \(\phi_2\) that is orthogonal to \(\phi_1\).

At some point, we need to decide how many principal components to use.
Unfortunately, there is no clear methodology for this and the answer can depend on the application. Often, we look at a plot of the proportion of variance explained (PVE):

\[
\sum_{j=1}^{d} \text{Var}(X_j) = \sum_{j=1}^{d} \frac{1}{n} \sum_{i=1}^{n} x_{ij}^2
\]

for each principal component, and assess by eye the point at which the PVE for the next principal component becomes too small to be worth including.

Often, when we are using PCA to look for interesting patterns and features in the data, we use only the first few principal components initially. If there is nothing interesting to see there, it is very unlikely that subsequent components will contain more interesting features so we can end our analysis there. If there are interesting patterns, we can keep searching through the components until we find nothing more of interest.

PCA can also be used as a precursor to unsupervised techniques, such as clustering (Section 2.2.7). Similarly to their use in supervised regression (Section 2.2.2), the use of principal components can lead to less noisy results because the signal tends to be concentrated in the first few principal components, with the noise becoming more dominant later when the signal is smaller. Again, the question of how many principal components to use has no well-defined answer in these types of application.

It is easier to decide which principal components to include when we use PCA to create composite variables for use in a supervised algorithm, because then we can treat the number of components as a parameter to optimise in the model, and use methods such as cross-validation to choose the best value. It makes sense that it is easier to define a ‘best’ number of components in these types of applications, because supervised learning problems are generally much better defined than unsupervised ones, and the labelling of data allows for an objective analysis of a model’s performance.

**t-distributed stochastic neighbour embedding**

t-distributed stochastic neighbour embedding (t-SNE) is a machine learning algorithm for visualising high-dimensional data in two dimensions. It is a non-
parametric method, and as such involves no assumptions about the data: it simply aims to plot the most similar data points closest to each other. It achieves the same end goal as the component analysis methods discussed previously, but those methods fail to effectively preserve local structures in some datasets so t-SNE is increasingly used to visualise high-dimensional data [8].

This algorithm is a form of stochastic embedding, in which each training data point is non-linearly mapped to lower-dimensional coordinates such that the local structure in the data is preserved. For the region close to each data point, a Gaussian probability distribution

\[ p_{ij} = \frac{\exp(-||X_i - X_j||^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-||X_i - X_k||^2/2\sigma_i^2)} \] (2.23)

that describes the likelihood that data point \( X_j \) is in the neighbourhood of \( X_i \), where \( ||X - Y|| \) denotes the Euclidean distance between \( X \) and \( Y \). The value of \( \sigma_i \) is chosen by setting the value of the perplexity

\[ P = 2^{H(p_i)} \] (2.24)

\[ H(p_i) \equiv \sum_j p_{ji} \log_2(p_{ji}). \] (2.25)

The perplexity \( P \) is a constant across all data points, and its value is an input parameter to the algorithm. A lower value makes the definition of ‘local’ more strict: points in regions of high density will have smaller values of \( \sigma_i \), so the neighbourhoods in high density regions are smaller than those in low density regions.

The problem with using a Gaussian probability distribution is that the mapping of outlying data points to the new variable space can be ambiguous. In t-SNE this problem is avoided by defining the new variable space using a long tail distribution:

\[ q_{ij} = \frac{(1 + ||Y_i - Y_j||^2)^{-1}}{\sum_{k \neq i} (1 + ||Y_i - Y_k||^2)^{-1}}. \] (2.26)

This preserves short distance information while repelling points that are far apart
in the original variable space, because at short distances the distributions in Equations 2.23 and 2.26 are very similar but at longer distances the difference increases dramatically. In order to find the coordinates $Y_i$ for the data points $X_i$, the t-SNE algorithm finds the $Y_i$ that minimises the Kullback-Leibler divergence between the two distributions

$$D_{KL}(p||q) \equiv \sum_{ij} p_{ij} \log \left( \frac{p_{ij}}{q_{ij}} \right), \quad (2.27)$$

where we use $p_{ij} \equiv (p_{ij} + p_{ji})/2N$ to preserve symmetry.

This minimisation is repeated for each point $x_i$ in the dataset, and is equivalent to finding a balance between an attractive force between points in the same neighbourhood and a repulsive force between points in different neighbourhoods. The end result is a plot in which groups of similar data can be visually separated into clear clusters, as shown in Figure 2.13(b) and the method is therefore commonly used to visualise clustering or classification problems.

![Figure 2.13: The effect of (a) PCA and (b) t-SNE applied to the MNIST data set] 

2.2.7 Clustering methods

K-means clustering

Sometimes, we want to group data into classes even though we don’t know what those classes are. This is where unsupervised learning can be used to cluster the data. One common algorithm used for this is $K$-means clustering, a method with
roots similar to k-nearest neighbours (KNN) classification. We aim to split the
data into $K$ clusters $C_1, C_2, ... C_K$, which between them contain all of the data
points and do not overlap (i.e. each data point is in exactly one cluster). A good
clustering will be one for which the variation between observations in a single
cluster is very small. As in the KNN algorithm, we use the Euclidean distance
(Eq. 2.3) as an indicator of how close together observations are. In this case
we define our within-cluster variance $W(k)$ as the sum of the squared Euclidean
distance in our $d$-dimensional variable space:

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,j \in C_k} \sum_{j=1}^{d} (x_{ij} - x_{ij'}^\prime)^2,$$

(2.28)

where $|C_k|$ is the number of observations in the $k$th cluster and we sum the
squared Euclidean distance over all pairs of particles in the cluster. The variance
is often expressed as the inertia of the cluster, which is the within-cluster sum of
squares $|C_k|W(C_k)$. We aim to partition the observations to minimise the sum
of these within-cluster variations over all clusters:

$$\min_{C_1, ..., C_K} \left\{ \sum_{k=1}^{K} W(C_k) \right\}.$$

(2.29)

In order to achieve this, we first randomly assign a number from 1 to $K$ to each
of the observations to initialise the system. Then, for each cluster, we compute
the centroid $(\bar{x}_1, ..., \bar{x}_j, ..., \bar{x}_d)$ where

$$\bar{x}_{kj} = \frac{1}{|C_k|} \sum_{i \in C_k} x_{ij},$$

(2.30)

giving us a $d$-dimensional vector to define each cluster’s centroid. We then
reassign each observation to a new cluster based on its closest centroid (where
again, we use Euclidean distance to determine closeness), and reiterate until the
within-cluster variation (Eq. 2.29) is no longer decreasing. The within-cluster
variation cannot increase using this method, so when the result no longer changes
we are guaranteed to have reached a local minimum.

This method finds only a local optimum solution, and the results depend on the
initial random clustering at the start of the method. Therefore, we must run the
algorithm multiple times using different initial cluster assignments, in order to get closer to a global minimum. We can then simply select the best solution as the one for which Eq. 2.29 is smallest.

One disadvantage of K-means clustering is the need to choose the number of clusters before we begin to fit the model. There is no obvious way to choose the optimal value of \( k \), other than by trying different values. Unfortunately, because we generally use unlabelled data for unsupervised learning such as this, we cannot use methods such as cross-validation to help with this task.

Hierarchical clustering

There are other, more complex, clustering methods that we can use instead of K-means clustering in order to avoid the need to find the correct number of clusters via trial and error. One of these is hierarchical clustering, where we start by assuming that each data point is in its own cluster. We therefore have \( n \) clusters and can use a measure such as Euclidean distance to determine the distance between every pair of clusters. We then look through this list of similarity measures and find the two clusters which are most alike (e.g. the pair separated by the smallest Euclidean distance). Although Euclidean distance is one option for a measure of the similarity of points within a cluster, we choose our similarity measure based on the nature of our data: in some applications, measures such as a correlation-based distance can much better capture the similarities we wish to capture.

We re-calculate the similarity between the new cluster and all the other clusters, and repeat until there is only one cluster. There are a few different ways in which we can choose how to measure the similarity between multi-observation clusters. Complete, single, and average linkage are the most common types of linkage, which defines the dissimilarity between two groups of observations. They involve computing all pairwise dissimilarities between observations in the two clusters and using the largest one for complete linkage (in effect using the maximum dissimilarity), the smallest for single linkage (using the minimum dissimilarity), or the mean of them all for average linkage. Average and complete linkage tend to result in more balanced clusters. Centroid linkage, where we calculate the dissimilarity from the centroid of each cluster, is another option but is less commonly used as it can result in inversions which can make the result difficult to visualise and interpret. A final commonly used approach, the Ward method,
calculates and minimises the variance within each cluster rather than the distance between clusters. In this method, the linkage between two clusters is defined as the increase in the within cluster variance that would occur if those clusters were merged.

![Cluster Dendrogram](image)

**Figure 2.14:** An example dendrogram showing hierarchical clustering, created using the Iris data set. The height at which two clusters are joined is the dissimilarity between the two clusters, as measured by the chosen linkage method. The red lines show cutoff values of dissimilarity where the data could easily be split into clusters.

This method is visualised by a dendrogram, such as in Figure 2.14. Starting at the bottom of the graph, whenever we fuse together two clusters we mark their fusion at a height corresponding to the dissimilarity between the clusters. The end result can be used to split the data into any number of clusters, and the choice of how many is usually done by eye based on the heights of fusion: in the example here, we might choose to split the data into 4 or 6 clusters, indicated by red lines on the graph. If we have labelled data, we can try a few different candidates to find the best number of clusters to split that data into, as well as trying different linkage methods and comparing the error rates. Otherwise, we have to accept that our choice may not be the best one. Unsurprisingly, hierarchical clustering performs poorly in cases where the data itself is not hierarchical. This means that in some situations it can lead to less accurate results than $K$-means clustering.

As with KNN classification, it can be beneficial to scale the variables so that some are not artificially viewed as more important than others. We should take care
to normalise the standard deviation, otherwise variables with a higher variance (either due to more actual variance or simply to a choice of units) will dominate the decisions.
Chapter 3

Autonomous identification of bijels from a two-dimensional image

In this chapter I present an example of using machine learning for a soft matter image analysis problem. I use machine learning to classify the outcome (successful or failed) of bijel fabrication experiments based on a single two-dimensional image of one of the liquid channels. I investigate the use of machine learning methods to quickly separate successful bijel samples from failed ones in order to automate one part of the difficult process of bijel production. In order to take a physics-led approach to the machine learning, I try to represent each image using variables that can be linked to a physical interpretation so that the model can highlight which features of the image are most useful for bijel classification, as well as being useful for classification itself.

3.1 Introduction

3.1.1 The growing interest in bijel fabrication

Bijels were first theorised in 2005 [94] and successfully created in 2007 [91]. Since then a number of research groups have investigated their fabrication using different materials and methods, their physical characteristics, and how they can be processed for different potential applications such as in fuel cells [100], controlled release devices [103], and tissue engineering [102].
The traditional method of making a bijel requires heating partially miscible liquids and interfacial particles up to a point where they mix then subjecting them to a temperature quench that takes them into the region where they will separate via spinodal decomposition (as described in Section 1.3.4). The success of this process relies on very fine tuning of the wettability of the stabilising particles, to ensure neutral wetting and therefore avoid inducing a preferred direction of curvature on the interface \[98\]. The method also requires the use of pairs of partially miscible liquids with similar densities which significantly limits the materials that can be used, especially because most such pairs include toxic or explosive materials. Novel methods of creating the same structure but avoiding these drawbacks are therefore an active topic of research, which creates a wide range of potential uses for a bijel classification tool.

One method, known as solvent transfer induced phase separation (STRIPS), uses a third liquid that acts as a solvent allowing the mixing of two otherwise immiscible liquids \[104\]. The mixture of three liquids is injected into a continuous phase, and as this happens the solvent is extracted into this continuous phase leading to demixing of the two immiscible liquids. The system is tuned such that this demixing occurs via spinodal decomposition and, in the presence of nanoparticles suspended in the ternary mixture, a bijel is formed.

To avoid the challenging task of tuning the surface treatment of particles to make them neutrally wetting in a given system, a mixture of commercially available particles with a mixture of wettabilities can be used to create the same overall neutrally wetting effect \[133\]. This approach requires using two populations of particles, each suitable for adsorption to the interface but preferentially wet by one of the two phases. By tuning the proportion of these two particle populations, the overall effect of neutral wetting can be achieved more easily.

Another novel method is the fabrication of bijel structures by mixing only, without inducing a phase change in the system. This can be done by using a mixture of functional polymers with different molecular weights dispersed in toluene that act as nanoparticle surfactants \[134\]. When this dispersion is mixed with water and agitated, the system forms a bijel structure with a characteristic length scale of 10\(\mu\)m or less. Alternatively, immiscible pairs of high viscosity liquids can be mixed into a bijel structure when stabilised by interfacial nanoparticles and a surfactant \[135\]. This bijel structure is created by tuning the composition and using a multi-stage mixing protocol in which the system is initially mixed slowly to form droplets, which are then turned into a particle-stabilised bicontinuous
structure via fast mixing.

The development of each of these bijel fabrication approaches will have required the manufacture of many samples which must be assessed to determine if the bijel structure has been successfully fabricated. A simple tool for bijel classification could be widely used to verify if a bijel fabrication has been successful, particularly because of this wide range of research is ongoing into easier methods for making bijel structures.

3.1.2 Limitations of characterising bijels

Most bijel publications tend to focus solely on successful samples, so there is little public record of methods used for determining whether a bijel fabrication has been successful or not. It is likely that most samples are assessed by the experimenter by eye both macroscopically and (predominantly) microscopically during imaging.

There are a number of published methods to quantify different aspects of a bijel, some of which can be used to identify failed bijels as well as to characterise successful ones. The local Gaussian ($K$) and mean ($H$) curvatures of a bijel structure have been found to have strong peaks at $K < 0$ and $H = 0$ \cite{107,136}. An alternative three-dimensional image analysis method of ‘region growing’ \cite{136} is used to determine whether the structure is bicontinuous: an important requirement of a bijel.

These methods could be used to help identify successful bijel samples. However, as these methods all require a suitable three-dimensional image of the bijel (such as a confocal stack or a CT scan of a polymerised bijel), as well as significant computational workload, there is much scope for a more versatile, high throughput alternative. Some initial attempts have been made to evaluate potential bijel structures in two dimensions using an empirical cost function \cite{137}, but this approach still requires significant image analysis in order to calculate the required quantities.

In this chapter I apply machine learning classification algorithms to confocal micrographs of only two dimensions, in order to create a tool that can be used to predict whether a micrograph is from a successful or unsuccessful bijel fabrication. This tool could therefore be used to simplify the process of assessing bijel samples.
to determine whether or not they have achieved the desired structure, as well as potentially speeding up the process by removing the need for the experimenter to assess each sample manually. In addition, I hope that the process of creating the classification tool will give an insight into what features of the micrograph are most important for the classification of bijels, which could give information on what distinguishes a bijel from other particle stabilised emulsion systems.

3.2 Data

I make use of data provided by Katherine Macmillan from a previous study of the mechanical properties of bijels under compression [106], in which there was a great need to verify the quality of bijel samples before they were subjected to centrifugal compression. This verification involved assessing each sample under a confocal microscope and I therefore have access to a large number of confocal micrographs (which I shall also refer to as images) of both successful and failed bijel samples on which different classification methods can be trained and tested.

The confocal images are from a selection of successful and unsuccessful bijel samples fabricated using the traditional quenching method. Each image is composed of a liquid (fluorescein-doped ethanediol) and a particle (silica modified with hexamethyldisilazane, doped with rhodamine B) channel. These two different fluorescent dyes were used so that the particles and the liquid could be imaged separately [106]. The second liquid phase (nitromethane) was not dyed but could be identified from the lack of the other two signals if needed. The images from the two fluorescent channels could be processed independently to obtain variables from image and channel, and in this chapter I work solely with the dyed liquid channel. The dataset comprises of 135 images that were chosen to represent the overall structure of the sample from which they were taken, each at a resolution of 512×512 pixels for each channel. Each sample and therefore each image was hand-classified as a bijel or a non-bijel by the experimenter, and these classification labels are included in the data.

Aside from the previously mentioned limitations of the current methods for identifying a bijel without requiring the experimenter to classify samples by eye, the data used in this chapter present an additional factor that makes it hard to predict the outcome of an experiment: the compositional parameters cannot be used to predict the success of the bijel fabrication with any accuracy.
Figure 3.1: Experimental data used in this project. Panel (a) shows the outcome of 135 attempted bijel samples indicating whether a successful bijel was created as a function of particle wettability and nitromethane mass fraction. Where samples 1 and 2 have the same composition, they are made from the same sample batch split in half. Panels (b) and (c) show examples of a successful and an unsuccessful bijel respectively. The ethanediol phase (liquid channel) is shown in magenta on the left, and the stabilising silica particles (particle channel) are in cyan on the right.
Figure 3.1(a) shows the outcome (successful bijel or not) of the representative set of 135 bijel experiments used in this analysis, and their dependence on key compositional parameters. The HMDS (hexamethyldisilazane) content controls the particle wettability, and the nitromethane mass fraction (relative to the ethanediol content) acts as a proxy for the distance of the system from the critical composition (the composition at which the critical point occurs, as described in Section 1.3.3), which is at a mass fraction of 0.640 \[^{106}\]. Many of the successful samples are clustered around this composition, but it is important to notice that the outcome of the experiment (bijel or non-bijel) is not guaranteed to be the same for the same composition because the fabrication process is highly sensitive to changes in the experimental process that may be within measurement tolerances.

This problem can be seen in Figures 3.1(b) and (c), which show images from two compositionally identical samples. These were created by splitting one sample in half and applying the same experimental protocol to each half. The sample in Fig. 3.1(b) was classified as a bijel, and Fig. 3.1(c) as a non-bijel as is evident from the abundance of droplets in the image. In this situation where identical composition parameters do not always give identical outcomes, they cannot be used to classify the success of a bijel fabrication. A classification approach based entirely on the analysis of confocal micrographs is therefore particularly valuable.

### 3.3 Methods

#### 3.3.1 Image processing

In order to represent the image data in a simpler way, I derived two functions from the Fourier transform of each image. Firstly, I calculated the structure factor by radially averaging the Fourier transform of the image. This is the same function that would be achieved by performing light scattering on a sample, but that approach is unsuitable due to the opacity of the bijel structure. Secondly, I calculated the autocorrelation function also by multiplying the Fourier transform of the image with itself under the transformation \( r \rightarrow -r \), then performing an inverse Fourier transform on the result. This effectively convolves the image with its reflection, which gives the correlation function of the image with itself, i.e. the
autocorrelation function:

\[ I(\vec{r}) \ast I(\vec{r}) = I(\vec{r}) \ast I^*(-\vec{r}) = \mathcal{F}^{-1}[\mathcal{F}(I(\vec{r}))\mathcal{F}(I^*(-\vec{r}))]. \]

I performed this image processing in Python using the skimage package to read in the two separate channels for each image and using the fast Fourier transform methods available in the standard scipy package. My scripts for this analysis were adapted from previous code created by Iain Muntz. I saved these functions as .csv files so I could use them in the rest of my analysis which was performed using R rather than python.

Figure 3.2 shows examples of the particle channel structure factor (a) and liquid channel autocorrelation function (b) of a bijel and a non-bijel. Both of these functions are two-dimensional, so I radially averaged them to create a one-dimensional plot: this is suitable for my data because bijels are isotropic over the span of the image so radially averaging should not remove much information. As I will show and discuss in section 3.5.2, this assumption holds for the majority of each image. As the structure factor is much noisier than the autocorrelation function, it was less useful for providing a wide range of variables to consider for classification predictors but could still provide some interesting variables to try. I could derive variables from these functions and equivalent ones from other images, and use these variables as inputs for machine learning algorithms.

![Figure 3.2: Representative example bijels (in red) and non-bijels (in black) in terms of the radial average of their structure factors and autocorrelation functions. Panel (a) shows the structure factors of the liquid channel images, and panel (b) shows the autocorrelation functions of the same images.](image-url)
3.3.2 Extracting and choosing variables

For this application it is generally the overall shape of these functions that I want to capture, so it made sense to derive variables that characterise the shapes of the functions. This also means that I can use many fewer variables than if I were to use the functions without further processing, making it much easier to find a minimal model without using variable reduction methods such as principal component analysis or ridge regression. Although these methods can be very useful, the use of additional algorithms can make it more difficult to interpret the results so it is nice not to rely on them where the problem allows. I used R to read in the previously calculated structure factor and autocorrelation functions of each image and to manipulate the data in order to calculate the different variables I wanted to consider.

The process of extracting variables from the functional representations of images started with an assessment of what information I could potentially use to characterise them. I considered that when researchers classify bijel samples manually they will be using their experience of bijels to identify them based on features such as the patterns and shapes that the different imaging channels typically make, and the presence of non-bijel elements such as bubbles. Generally, these features are difficult for a computer-based approach to identify and even more difficult to quantify, but I made sure to bear them in mind because they represent the process I am trying to reproduce.

Ideally my chosen variables would be easy to relate to a physical feature, because this would lead to a more interpretable model that would allow me to gain a greater understanding of the system as well as simply classifying the bijel, but this is not a requirement for the model to be useful as a classifier. I considered a range of approaches including the order and coefficients of a polynomial (or polynomial plus exponential) fit to the function; the number and position (both in x and y) of turning points in the function; the gradient at key points in the function, such as at the start or end; and the depth and position of trough/peak pairs. The selection of variables to try is the main source of human input to this analysis, which can be both a good and a bad thing. A lot of computational effort can be saved by using knowledge of the physics behind the functions and by assessing ways in which bijel and non-bijel functions look different, but there is always a risk that I may miss a variable that could be a great predictor because I cannot identify it. However, if a model can identify bijels with an acceptable
level of accuracy then it is not very important that some variables may have been missed.

I also considered using the whole function as a set of variables, rather than manually selecting and quantifying individual features. However, using the whole function as a set of variables meant that there were more variables than data points (images), so it was always possible to perfectly separate the bijels from the non-bijels. This meant that shrinkage methods such as the lasso and ridge regression could not be used. I also did not want to use dimension reduction methods because I wanted the results to be as easy to interpret as possible, so I focused on using specific variables as previously described.

To eliminate variables with poor predictive power, I assessed each candidate variable using a box-and-jitter plot such as the ones shown in Figures 3.3. A plot such as the one in Figure 3.3(b) where the two boxes, showing the 25th and 75th percentile values of the variable for bijels and non-bijels, overlap a great deal is an indicator of a variable that is unlikely to be useful for distinguishing between bijels and non-bijels. A plot such as in Figure 3.3(a) suggests the opposite: that the variable will be very useful for distinguishing between the two classes. Such variables are, however, uncommon and a more usual indicator that a variable may be useful for my purposes is a difference in the mean values of the variable shown by the lines within the boxes (Figure 3.3(c, d)), or in the overall distribution of the variable (Figure 3.3(d)).

I also trialled the use of some additional variables which were not directly derived from the autocorrelation function or structure factor, such as the predictions of alternative classification algorithms (for example using the output of a logistic regression classifier as an input variable in a KNN classifier). I chose to do this, particularly in my initial trials before I started optimising the variables within my model, because I wanted to include enough variables to describe all the key features of each function. I was also aware that variables could be poor indicators for classifying bijels when viewed alone as in the box-and-jitter plots but then prove to be very useful when combined with other variables. I therefore trialled these variables when choosing the initial set of variables to work from.
Figure 3.3: Box-and-jitter plots of three of the variables considered in the initial model. These plots show how the values of the variable are distributed for bijels and non-bijels. Jitter (random offset) is added in the x-axis so all points can be seen. Panel (a) demonstrates a variable that is particularly good for separating bijels from non-bijels, with no overlap between the boxes representing the majority of the distributions of the two classes. Panel (b) demonstrates a variable that is poor for separating bijels from non-bijels, with a great deal of overlap between the two classes and no real difference between the two classes. Panels (c) and (d) demonstrate a variables where it is unclear whether it will be useful for separating bijels from non-bijels.
3.3.3 Machine learning approach

When applying machine learning to a problem the first step is to decide what information is desired and what data are available, and then to choose what type of algorithm is most suited to the problem. Here I had labelled categorical data, and my goal was to identify which category future images belong to, so the most obvious choice was to use supervised classification techniques. I could also consider unsupervised clustering: although this would not make use of the fact that my data were labelled it could help me to understand underlying structures in the data, and the labels could be used later to verify whether the clustering correlated with the known bijel classification.

Both supervised classification and unsupervised clustering use similar types of data, so I could trial both approaches using the same data. I could use either continuous or categorical variables, but there is little sense in creating categorical variables in this instance when I have continuous numerical data in the form of the functions described above. As well as deriving numerical variables that describe the shape of the function, I could input the functions directly into an algorithm by treating each q- or r- value as a variable: this variable space would likely need reducing using a method such as ridge regression or PCA.

This problem seems to lend itself to the application of supervised classification algorithms to a small number of variables derived from the structure factor and autocorrelation function of the image data, and chosen such that they captured the key features of each function, but for completeness I performed small exploratory tests using unsupervised clustering and using the raw function as an input to the algorithm. I found that these were not suitable, so I chose a small number of supervised classification algorithms to work with and optimise in more depth.

I compared the performance of k-nearest neighbours classification, logistic regression, classification (decision) trees and support vector machines in order to determine which was the most suitable for accurately classifying these data. I selected the best-performing of these models to use as a base upon which I would work to improve the result. Although I focused the majority of my analysis on one algorithm, I revisited some of the others at various points in the project to check that I was still working with the most optimal algorithm. I used R to perform all of the machine learning in this chapter, primarily using the caret package [139].
which contains functions for all of these algorithms along with a wide array of
machine learning tools.

3.3.4 Errors and Cross-validation

In order to find the optimal model, I needed to decide how I was going to measure
the performance of each model I considered. As my aim was to accurately
predict whether an image was from a successful or unsuccessful bijel, I measured
performance based on the model’s error rate when classifying images with known
labels. Here, the classification error rate is defined as the percentage of images
that are misclassified during testing. The optimal model is the one with the
smallest classification error.

Although my data set of 135 samples was large in terms of soft matter
experimentation, this is a small number in terms of machine learning. I therefore
used cross-validation, as described in Section 2.1.4, to make the most of these
data without having to set aside a large chunk for testing the models separately.
I used 10-fold cross-validation in all of the model fits, so each portion of the data
was used for training 9 times and for testing once. The classification error rate
I used to compare models was therefore the cross-validation error rate which, as
discussed previously, quantifies how well the model classifies every data point in
the data set.

When fitting any model, the first comparison to make is to the null error rate, as
discussed in Section 2.1.4. In this case the most common class is the successful
bijels, with 92 of the 135 samples belonging to this class. The null error rate is
therefore 31.9%, the error achieved if all of the 43 non-bijels are classed as bijels.
Any model performing worse than this error rate is entirely useless.

3.3.5 Optimising the variables within a model

Once I had decided on an algorithm and an initial set of variables to use for a base
model, I worked to improve its classification performance. I revisited the variables
that were included in the model, and assessed their success in differentiating
between bijel and non-bijel samples within the context of my base model. As I
was working with only a few predictive variables, I could assess the impact of each
variable individually rather than relying on methods such as principal component
analysis. This gave me the benefit of knowing exactly how each variable affected the final performance of the model, making it more likely that the final model will have some scope for physical interpretation.

Starting from an initial set of 7 variables as outlined below, I first looked for any variables that might be misleading the algorithm into false categorisations and should therefore be removed from the model. To do this, I removed each variable in turn and trained the model on the remaining variables. I compared the error rate for each new version of the model with the error rate of my initial model, and if the removal of a variable reduced this error rate then I could conclude that it was negatively impacting the performance of the model and therefore should not be included going forward.

Once I had this new set of variables that was free from those that decreased the classification performance, I worked to reduce the model by removing any variables that had little to no effect on the final outcome to create a simpler model and reduce the chance of over-fitting to the data. Even with a modest number of variables, testing every possible version of the model with different number of variables would have been very labour intensive. I therefore calculated the F-statistic for each variable in the model in order to quantify how significant it was to the final result. I used this statistic to rank the predictive variables in order of significance then removed variables from the model one at a time, starting with the least significant, to make a series of models each with one variable fewer than the last. Comparing the error rates of each of these models allowed me to find the number of variables that gave the smallest error. I therefore arrived at a final model containing the minimal subset of variables required to successfully separate bijels from non-bijels.

3.4 Results

3.4.1 Initial model

For my initial model, I settled on seven initial predictive variables describing the shapes of the autocorrelation function and structure factors of the liquid channel images. These variables were chosen from a larger set of candidates based on their box-and-jitter plots as described above. The variables are:
• The number of turning points present in the autocorrelation function.
  – As identified using the turnpoints function in the pastecs R package, which is very sensitive and can pick out turning points that are not visible to the eye.
• The value of \( r \) at which the first turning point in the autocorrelation function occurs.
• The depth of the first peak/trough pair in the autocorrelation function.
  – The correlation value at the second turning point minus the value at the first point.
• The value of the structure factor at the point \( r = 30 \) pixels.
  – Calculated as the average of points 29-31 to smooth some of the effects of noise in the data.
• The gradient of the first 20 points of the structure factor.
• The height of the largest jump in the structure factor.
  – The maximum of the value \( SF[r + 5] - SF[r] \) when calculated across all \( r \).
• The location of the largest jump in the structure factor.
  – The value for \( r \) at which the value \( SF[r + 5] - SF[r] \) is maximised.

The box-and-jitter plots for each of these variables are shown in Figure 3.4. I also trialled the use of the output of one classification model as an additional input to another, but this did not help the performance of the model in the initial testing phase.

As I investigated different variables, I also tested and compared the performance of the four machine learning algorithms (k-nearest neighbours, logistic regression, support vector machines, and decision trees). I found that the k-nearest neighbours (KNN) algorithm consistently gave the best results for these images over a range of variable combinations, so I continued to use this algorithm for further refinement. The error achieved with this model, using the seven variables outlined above and a value of \( k \) chosen via cross-validation to be 55 nearest neighbours, was 21.5%. As this is a significant improvement on the null error, this result was a promising starting point for further optimisation.
Figure 3.4: Box-and-jitter plots for the seven variables considered in the initial model. It is clear that some of these variables separate bijels from non-bijels more successfully than others: the number of turning points in the autocorrelation function (a) is a particularly poor example, and the position of the first turning point in the autocorrelation function (b) is a particularly good example. With other variables like the position of the biggest jump in the structure factor (g) it is unclear what the predictive performance will be.
3.4.2 Performance improvement

I started the process of improving on the base model by changing the variables input to the k-nearest neighbours algorithm. As previously discussed, I removed each variable one at a time and compared the errors of each 6-variable model to the full 7-variable version, as shown in Table 3.1. Each version of the model was optimised using cross-validation to choose the optimal number of neighbours. These results show that one particular variable, the position of the largest jump in the structure factor, was making the performance of the model worse. I therefore removed this variable from the model, along with the height of this jump which also made the model slightly worse. A couple of other variables also made the model marginally worse but since they made less than 0.1% difference to the error I retained them for the time being, expecting that they were likely to be removed in the later stages of model optimisation anyway.

<table>
<thead>
<tr>
<th>Removed variable</th>
<th>Error on trained model</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>21.5%</td>
</tr>
<tr>
<td>Number of ACF turning points</td>
<td>21.5%</td>
</tr>
<tr>
<td>Depth of first ACF peak/trough pair</td>
<td>21.5%</td>
</tr>
<tr>
<td>Position of first turning point in ACF</td>
<td>24.4%</td>
</tr>
<tr>
<td>Value of SF at r=30 pixels</td>
<td>21.5%</td>
</tr>
<tr>
<td>Gradient at start of SF</td>
<td>21.8%</td>
</tr>
<tr>
<td>Height of largest jump in SF</td>
<td><strong>21.2%</strong></td>
</tr>
<tr>
<td>Position of largest jump in SF</td>
<td><strong>19.0%</strong></td>
</tr>
</tbody>
</table>

Table 3.1: The effect of the removal of individual variables on the classification performance of a trained k-nearest neighbours classification model. The cases where removing the variable has reduced the error are highlighted in bold.

Working with this reduced set of variables, I compared their significance based on the F-statistic, as outlined in Table 3.2. From these results it is clear that the position of the first turning point in the autocorrelation function is the most significant predictor, followed by the initial gradient of the structure factor. I iteratively re-fit the KNN model removing the least significant variable each time. Figure 3.5 shows that the removal of each less significant variable led to a decrease in the classification error, and that the best-performing model has only one variable: the position of the first turning point in the autocorrelation function.
Table 3.2: F-statistic values indicating the significance of each variable in the KNN liquid channel model.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position of first turning point in ACF</td>
<td>61.6</td>
</tr>
<tr>
<td>Gradient at start of SF</td>
<td>49.8</td>
</tr>
<tr>
<td>Value of SF at r=30 pixels</td>
<td>27.3</td>
</tr>
<tr>
<td>Depth of first ACF peak/trough pair</td>
<td>12.2</td>
</tr>
<tr>
<td>Number of turning points in ACF</td>
<td>0.09</td>
</tr>
</tbody>
</table>

Figure 3.5: A plot showing how the classification error changes as the number of variables in the KNN model is reduced by removing the least significant variable in the model each time. In this case, the optimal model is the smallest, with only one predictive variable.
3.4.3 Final model

Figure 3.6: Cross-validation results for the final single-variable KNN fit to the liquid channel images, and how the cross-validated accuracy of the model changes with the number of neighbours used in the classification. Here the optimal value chosen by the algorithm was $k = 5$, the first peak in this plot.

Figure 3.6 shows the outcome of the final k-nearest neighbours model fitted to the liquid channel images, specifically how the accuracy varies depending on the number of nearest neighbours used in the model. It shows that $k = 5$ gives the model fit with the highest accuracy, so this is the value I used for the final model. Figure 3.7 shows the predictions made by this model compared to the true classifications. The error rate for this prediction model is 16.4%.

As an additional validation of this model and its usefulness for identifying future bijels, I applied the trained model to new data that were not used at any point during the testing and training above. I assessed two different sets of data: one with images from samples with similar compositions to those the model was trained on, and one with different compositions and therefore slightly different channel widths. In the case where the compositions were similar to some present in the training data the error was 15.8%, an improvement compared to a null error rate of 42.1%. In the case where the compositions were more different the
error was 38.7%, a worse performance than the null error rate of 29.0%.

![Figure 3.7: The output and classification performance of the final model. The plot shows the predicted (open circle) and true (filled circle) bijel classifications for each sample in the dataset, with y-axis jitter added because only one variable was used in the model. Predictions were output from the final liquid channel KNN fit.](image)

Table 3.3 shows a breakdown of the classification performance on the original data set used for training and on two additional unseen data sets, one including only a selection of samples with a similar composition to the original data and one including samples of a different composition. For each of these three data sets, the correctly (plain text) and incorrectly (italic text) classified samples are further separated based on whether they were truly bijels or non-bijels. The balance of between these different populations of data points can be understood by looking at the false positive rate (FPR) and false negative rate (FNR), which summarise the performance of the model on each of the two classes individually.
<table>
<thead>
<tr>
<th>Training/test data</th>
<th>True non-bijel</th>
<th>True bijel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted non-bijel</td>
<td>28</td>
<td>5</td>
</tr>
<tr>
<td>Predicted bijel</td>
<td>15</td>
<td>87</td>
</tr>
<tr>
<td></td>
<td><strong>FPR: 34.9%</strong></td>
<td><strong>FNR: 5.4%</strong></td>
</tr>
<tr>
<td>New data, similar composition</td>
<td>True non-bijel</td>
<td>True bijel</td>
</tr>
<tr>
<td>Predicted non-bijel</td>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>Predicted bijel</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td><strong>FPR: 0%</strong></td>
<td><strong>FNR: 37.5%</strong></td>
</tr>
<tr>
<td>New data, different composition</td>
<td>True non-bijel</td>
<td>True bijel</td>
</tr>
<tr>
<td>Predicted non-bijel</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>Predicted bijel</td>
<td>7</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td><strong>FPR: 77.8%</strong></td>
<td><strong>FNR: 22.7%</strong></td>
</tr>
</tbody>
</table>

Table 3.3: A breakdown of the predictions of the final model when applied to three different data sets. Each combination of true and predicted bijels and non-bijels is shown with a count of how many data points fit into that category. The numbers of true bijels predicted as bijels and the true non-bijels predicted as non-bijels (i.e. the correctly classified data points) are shown in plain text, and the numbers of incorrectly classified data points are shown in italics. The false positive rate (FPR) and false negative rate (FNR) are included for each data set.

3.5 Discussion

3.5.1 Model performance

The main result of this chapter is that I have created a classification model that can predict whether a single two-dimensional image is from a bijel sample or a non-bijel sample with up to 83.6% accuracy. The classification error is around half that of the null error, meaning it will mis-classify half as many samples as in the case where everything is assumed to be a bijel. It should not be assumed, however, that the model follows the pattern of the null error where only the non-bijels are misclassified. Looking at a breakdown of the results shown in Table 3.3, although most of the misclassified samples were indeed non-bijels (i.e. false positives) some of them were a false negative result on true bijels.

Although the model performs very well on the original data set, its performance somewhat is unreliable when the trained model is applied to unseen images. When used to classify images similar to those on which it was trained, the model performs very well, with an error much lower than the null error for that dataset.
and even lower than the error for the data on which the model was trained. This improvement in performance is likely down to luck: the images in this small dataset are probably ones in which the bijels and non-bijels are unusually distinct in terms of the variable used in this model. When used to classify other bijel images with a slightly different experimental composition, the model performs very poorly: worse than the null error rate for these data. The poor performance is dominated by a very high false positive rate which I discuss below, but the false negative rate is also much higher than for the original data. Interestingly, the similar-composition result actually performs the worst of the three in terms of false negative rate which could be an issue for the usefulness of the model because this set of data represents the case in which it is most useful to have a low false negative rate.

Overall, the classification model usually has a bias towards a higher false positive rate and a lower false negative rate. Taking the results from all three datasets into account, the false positive rate is 34.9% and the false negative rate is only 10.7%. This bias is probably due to the fact that the training data contains many more examples of bijels than non-bijels, which could introduce a tendency to class an image as a bijel in cases where the outcome is unclear. It is possible that this bias has had some impact on the effectiveness of the model when applied to the different-composition dataset, because it contains a higher proportion of non-bijels than the data on which the model was trained.

There is also the possibility that the characteristic value of the variable in the model is different in the new data sets. This seems likely to be the reason for at least some of the decrease in performance. The new data set that was different in composition from the original data (with a much greater concentration of HMDS) performed worse than the other new data set which had the same composition as two of the samples in the original data. This indicates that a model with more variables would be worth investigating, because additional variables may make it easier to account for variances between different bijel samples. I did not revisit the versions of my model with more variables in because this would decrease the performance of the model on the original data, and although the effects of over-fitting to the specific data are worth considering, these have been somewhat mitigated by the use of cross-validation. Instead I accept that the final variable in the model, the position of the first turning point in the autocorrelation function, gives the model the most information it can use to differentiate between bijels and non-bijels, out of the variables I have considered so far. I will now move on.
to consider different variables such as ones related to the particle channel which was not considered in this chapter.

This chapter demonstrates the potential for the use of machine learning algorithms for supervised classification of bijel images, with the ability to beat the null error rate on unseen data with a similar composition to the training data, but also that there is a lot of room for improvement when the classifier is applied to less similar unseen data. In order to apply my current classification approach to a completely new set of bijel data for a different experiment, it would require training the model on an initial set of labelled data before it could be used to skip the manual labelling step to assess any number of future images. My final model takes only a matter of seconds to run, including training it on the data with cross-validation, so once an experimenter has generated enough labelled data to effectively train the model they can then save a significant amount of time classifying further images. Even taking time to periodically re-train the model on new data, this approach could be used to create a valuable tool for researchers creating large numbers of bijel samples that may fail to form the correct structure.

Although I did not test all different possibilities to confirm that my final model has the best performance possible, by testing different combinations of selected algorithms and variables I could at least optimise the most promising approaches and converge on a local maximum of performance. It would be impractical to even test all possible combinations of the variables I used, let alone all possible variables that could describe the images. As long as the model was accurate enough to be useful for the problem at hand, i.e. the identification of successful bijels from two-dimensional images, then the process could be deemed a success.

As this model optimisation has converged on the use of only one variable, it might be tempting to consider classifying bijels based simply on a threshold value of this variable. However, the application of machine learning is still required for two reasons. Firstly, the benefit of developing a machine learning method is that once a good model is chosen it can be re-trained on a different set of data in order to make predictions about a different bijel system. If I aimed to identify bijels by simply setting a threshold for the value of this autocorrelation turning point, this may not be easy to generalise to other bijel systems. Secondly, the non-linear nature of the KNN algorithm has allowed for a more complex decision boundary than a single split at a certain value. This can be seen in Figure 3.7, with non-bijels predicted both below and above what might initially seem like a cut-off point around 50 µm. The fact that such a decision boundary is possible is
partly related to the number of nearest neighbours chosen for the model via cross-validation. The initial model before optimisation used the nearest 55 neighbours to determine whether a point in variable space would be a bijel or not, which as discussed in Chapter 2 (Figure 2.5) means that the decision boundary is much less complex than when there are only 5 nearest neighbours considered, as in this final model.

### 3.5.2 Model interpretation

The approach I have taken, starting with a set of variables and then reducing it to the best-performing model, can reveal which physical features of the bijel are most important for their successful classification. Clearly the most important variable is the position of the first turning point in the autocorrelation function. This variable can be interpreted physically as the predominant characteristic length scale of features in the image. In the case of bijels, it is known that the structure has a single characteristic length scale which corresponds to the size of the two liquid domains. A close look at the result presented in Figure 3.7 above shows that this particular group of bijels is identified by the model as having a characteristic length scale of less than $50 \mu m$ but not $25–27 \mu m$. This result aligns with the evidence in the original data (e.g. Figure 3.1), where the characteristic length scale of the bijel example is significantly less than $50 \mu m$ but the length scales visible in the non-bijel example are generally greater than $50 \mu m$.

I can also look at the variables that weren’t included in the final model, and I am particularly interested in the ones that I removed in the first variable reduction phase because they made the classification performance worse: perhaps they capture a feature that is shared by some bijels and some non-bijels and this is the reason they act against successful classification. These variables were the position and height of the largest jump in the structure factor of the image, and a look at their box-and-jitter plots in Figure 3.4(f–g) suggests that the problem is a high concentration of data points with a value at or around zero for both bijels and non-bijels. This is the case for both of these confounding variables as well as for the depth of the first peak/trough pair in the autocorrelation function, which was the first variable to be removed based on its significance as measured by the F-statistic. This gives me more understanding of features to consider in the box-and-jitter plots in future, as it suggests that this high concentration of data points at or around a single value (zero in this case) is another indicator
of a poor variable for separating the two classes. These two variables were also difficult to link to a physical interpretation, which may be related to their poor classification performance.

Another variable with poor classification performance is the number of turning points in the autocorrelation function. There are two likely reasons for this: firstly the Python function I used to calculate the number of turning points in the autocorrelation function has a tendency to pick up many more turning points than a human would identify by eye, which suggests that the variable may mostly be capturing noise in the function. The second reason becomes clear upon closer inspection of the assumption previously made that the autocorrelation function has radial symmetry due to the isotropic nature of bijels. As shown in Figure 3.8, the autocorrelation functions in two dimensions are mostly radially symmetrical but this breaks down in the outer region of some of the images, such as in panel (a). The number of turning points in the autocorrelation function is the only variable that is not derived solely from the region where the assumption of radial symmetry is valid, so this is another explanation for why it does not effectively characterise the original images.

3.6 Conclusions and next steps

In conclusion, this approach of using unsupervised learning to separate successful bijels from unsuccessful ones has proved to be effective. I achieved an error rate
of only 16.4%, meaning the model will mis-classify around half as many non-bijels as in the case of the null error. The model has a tendency to mis-classify true bijels less often than true non-bijels, which could be seen as an advantage for its use as a practical tool for filtering out poor samples. When considering the model's performance on unseen data, it is clear that the performance is not as good as it first seems. Although the model performs very well on one set of data, it is worse than the null error on the other set. This may be due to the fact that the final model relies on only one variable, and it highlights significant room for improvement that could be achieved by considering information from the particle channel images as well as the liquid channel considered here.

The process of starting from a larger model and reducing the number of variables involved proved to be an effective approach to the problem, and in this case resulted in a model with only one variable. Where possible, I chose to work with variables that could be related to the physical characteristics of the system, and this allowed me to understand what physical features were relevant for this classification. In this case, the classification decision is based solely on the position of the first turning point in the autocorrelation function, which is an indicator of the characteristic length scale in the image. This result matches nicely with knowledge that all bijels have a single characteristic length scale, and reveals that it is possible to identify bijels with some accuracy based on this feature alone. The physics-led approach to this task has therefore allowed me to conclude that the characteristic lengthscale is a key defining feature of a bijel image, in addition to lending credibility to the classification tool itself because its criteria for classification are easy to understand. I think this part of my aim for this chapter has been a particular success.

As the use of machine learning for this application has resulted in a very fast and somewhat accurate classification tool, I conclude that it is worth investigating this approach further by adding in more variables. I hope that the addition of these to the model will improve the classification performance without compromising on classification speed.
Chapter 4

Analysing and improving bijel classification performance

In this chapter I expand on the work from the previous chapter and investigate how the addition of variables from a second imaging channel can improve the classification performance of machine learning algorithms applied to the problem of identifying successfully fabricated bijels. Continuing to take a physics-led approach to this machine learning task, I determine the importance of different variables derived from both the liquid phase and the interfacial particle phase of a bijel and use this information to draw conclusions about the defining features of a bijel. I also demonstrate the use of a different classification algorithm as well as some additional techniques for validating the success of a classification model.

4.1 Introduction

The bicontinuous nature of bijels is a key feature of the structure, but the use of this term somewhat neglects the role of the particle network that supports the bijel and forms a third continuous phase in the system. Of course, the liquid channels are a key component and their structure is key to defining a bijel. The majority of the novel bijel fabrication methods outlined in Chapter 3 aim to widen the range of liquids that can be used to fabricate a bijel. There is also a great deal of research focused on functionalising one or both of the liquid channels, usually involving the polymerisation of one channel to create a robust template.
in which the other channel can be replaced with a material of choice \cite{107}. These templates can be used for a wide range of applications, for example to create a hierarchically porous material for use in electrodes \cite{140}, or a biomaterial for applications such as tissue regeneration \cite{141}.

The particle channel of a bijel is, however, a key component of the structure because it is only by the jamming of interfacial particles that the bijel structure can be stabilised \cite{94}. The morphology of a bijel can be directly controlled by the concentration of interfacial particles in the system \cite{91} because the spinodal decomposition that forms the bijel structure is arrested at the point where the particles jam on the interface so adding more particles to the system will arrest the demixing earlier, leading to a larger interface and narrower channels. It has also been shown \cite{96, 142} that it is possible to make the particle network robust enough to retain its shape upon remixing of the two liquid phases, creating a new form of gel which is referred to as a monogel. The strength of the interactions between interfacial particles has also been shown to have a direct impact on the ability to process bijels without compromising their structure \cite{108}, so the applications mentioned above that involve polymerising a liquid channel of the bijel are directly impacted by the stability of the particle channel. The same polymer templates can also be used to effectively replace the particle network with a new material, by coating the template in a thin layer of the desired material \cite{101}.

Figure 4.1: Confocal micrographs of (left) the liquid channel of a bijel and (right) the particle channel of the same area of bijel. These two channels are often imaged separately using different fluorescent dyes.
A previous attempt to identify bijels from a 2D image relies solely on information from the liquid channel \[137\], but in three dimensions a bijel can be defined solely by the curvature of its interface \[136\], which is stabilised by the particle channel. I therefore aim in this chapter to investigate the importance of the particle channel for identifying bijels from a two dimensional image with machine learning. I apply the same machine learning classification algorithms as used in Chapter 3 to variables derived from the particle channel of confocal bijel images, first alone and then in combination with variables from the liquid channel of the same images. In doing this, I aim to find the optimal tool for predicting whether a dual-channel confocal micrograph is from a successful or unsuccessful bijel fabrication, and hopefully to improve on the performance of the model defined in Chapter 3 in terms of classifying unseen data. I also use the opportunity to further investigate and demonstrate the use of physics-led machine learning techniques for image analysis in soft matter by performing additional validation on my most optimal model and by demonstrating a different algorithm from the one highlighted previously.

### 4.2 Data and methods

In this chapter I used the same data as in Chapter 3: two-dimensional confocal images of a mixture of successful and unsuccessful bijel samples. As discussed in that chapter, the images included two fluorescent channels but I only used one, the ethanediol liquid phase. In this chapter I use the data from that channel along with the data from the fluorescent particle channel, in order to understand what additional information can be gained from the particle structure.

The particle channel images were processed in the same way as the liquid channel images were, using Fourier transforms to find the structure factor and autocorrelation of each image. Examples of these functions are shown in Figure 4.2. In this case I found that the shape of the structure factor was extremely variable within each of the two classes of bijel and non-bijel, as can be seen in Figure 4.2(c), so I used only the autocorrelation function for this analysis.

My approach to machine learning was also similar to that in the previous chapter. I started from a base set of variables derived from the autocorrelation function of the particle channel images and standardised them to have a mean of zero and a standard deviation of one. I then ascertained which algorithm (out of k-nearest
Figure 4.2: Top: Examples of the structure factor (a) and autocorrelation function (b) of the particle channel of representative bijel (in red) and non-bijel (in black) confocal images. Bottom: plot of all examples of the structure factor (c) and autocorrelation function (d), showing the broad distribution of these functions, particularly the structure factor. 12.5 µm corresponds to approximately 10 pixels in the image.
neighbours, logistic regression, classification trees and support vector machines) gave the lowest classification error for these variables. My classification error was again defined as the percentage of all the 135 images that were incorrectly classified during training and testing with 10-fold cross-validation. I then reduced the number of variables in the model by removing the least significant one and repeating the training process using the best-performing algorithm, until I found the optimal number of most significant variables to include in the model.

I then had a model equivalent to the one presented in Chapter 3 and could directly compare them to determine which image channel contained the most pertinent information for identifying a successful bijel. The final step I took, in terms of applying machine learning to this problem of bijel classification, was to repeat this process of selecting and refining a model but this time starting from an initial set of variables comprised only of those that were used in the final model for each channel. After refining this model, I compared its performance to both of the single-channel models to see if the combination allowed for an increase in classification accuracy.

In this chapter I made more use of logistic regression, which allowed me to use the coefficients of the regression model to determine the significance of each variable, rather than the F-statistic I used in Chapter 3. These fit coefficients can be interpreted either as they appear in the model, or I can calculate the odds ratio by taking the exponential of each coefficient. As described in Section 2.2.3, an odds ratio close to one (coefficient close to zero) indicates a less significant variable, whereas one close to zero (coefficient large and negative) indicates a variable where a high value signifies a sample is not a bijel, and one much greater than one (coefficient large and positive) indicates a variable where a high value signifies a sample is a bijel. Because the odds ratio operates on a logarithmic scale, it can be difficult to directly compare values greater than and less than one, so I generally preferred to use the logistic regression coefficients directly but I include both in my results.

After optimising a model based on the particle channel images, I created a combined model using the variables from both the liquid and particle channel images. I restricted my investigation of which variables to use to those that were in the optimised model for each channel, so that I could quickly converge on an optimal combined model. Because I only used this restricted set of variables, and because I had already tried out a number of the combinations of these variables when I optimised models for the individual channels, it was reasonable...
to assess the performance of my chosen algorithms for all possible combinations of variables. I therefore did this instead of removing the variables in order of significance. As with the individual channel models, I assessed the performance of all four of my candidate algorithms to check I was still working with the best one. I paid closest attention to the algorithms that led to the best performance in the two individual channel models. I then compared the performance of the combined model to each of the individual channel models, and chose which of the three was the most suitable to use as a final result.

In addition to the machine learning methods used in Chapter 3, I used two further validation methods to assess the quality of my the classification performance of my final model. I plotted receiver operator characteristic (ROC) curves which, as described in Section 2.1.5, provide a deeper understanding of how well a binary classifier performs in terms of a balance between true positives and false positives. Where I used logistic regression for classification I could create the full curve demonstrated in that explanation, showing how the performance of the model changes as I change the probability threshold at which I decide an image should be classed as a bijel.

The second additional verification method I used was random re-labelling, also outlined in Section 2.1.5, to verify that my classification model was picking up features that were truly associated with the features of bijels and non-bijels. The model was trained on the same images that were used for the original training but with random (and thus likely to be incorrect) labels of bijel and non-bijel. The total number of each label was not changed.

4.3 Particle channel results

4.3.1 Initial model

As previously discussed, I decided to focus on the autocorrelation function when choosing variables for the particle channel images, particularly knowing that the final optimised version of the liquid model included only a variable from the autocorrelation function. The variables I considered for this channel are:

- The gradient of the first 10 points of the autocorrelation function.
Calculated from a straight line fit to these 10 points.

- The gradient of the first 20 points of the autocorrelation function.

Calculated from a straight line fit to these 20 points.

- The location of the first turning point in the autocorrelation function.

- The value of the first turning point in the autocorrelation function.

- The number of turning points in the autocorrelation function.

The box-and-jitter plots for these five variables are shown in Figure 4.3. None of these variables show much separation between the distributions for bijels and non-bijels, so I chose not to remove any of them based on the box-and-jitter plot. There is no clear single separator for these data, but a combination of variables may still be able to successfully identify bijels.

Because the work in Chapter 3 gave me a good idea of what kind of variables were suitable for this task, I used the results from these five variables to choose which model to work with, rather than performing more exploratory analysis. Of the four classification algorithms I tried, logistic regression gave the best performance with a classification error of 20.3%. The support vector machine was the only other algorithm that came close to this performance, at 21.5%.

Logistic regression has a benefit over the other algorithms because the fits can be directly used to determine the significance of each variable without the need to calculate additional quantities such as the F-statistic. The logistic regression generates a probability of a sample being a bijel and being a non-bijel, and the sample is classified as a bijel if it is more likely that is a bijel than not, i.e. $\Pr(\text{bijel}) \geq \Pr(\text{not bijel})$ or $\Pr(\text{bijel}) \geq 0.5$. As discussed in Section 2.2.3, this probability boundary does not need to be set at 0.5 and can be varied in order to optimise the classification performance for the desired application, but I will first work with this fixed value in order to focus on optimising the inputs to the model. I therefore move on to optimising the choice of variables in this model to reduce the 20.3% error which is already an improvement on the null error of 31.9%.
Figure 4.3: Box-and-jitter plots for the five variables considered in the initial particle channel model. None of these variables appear to be particularly good at separating bijels from non-bijels, but a combination of them may fare better.
4.3.2 Optimising the model

The output of the logistic regression fit is shown in Table 4.1. Because I made the decision to standardise the variables before applying any classification algorithm, the coefficients and odds ratios for each variable can be directly compared without having to consider the scale of each variable. The result here shows that the most significant of these predictors were the gradients of the first 10 and first 20 points of the autocorrelation function, followed by the value of the autocorrelation function at its first turning point.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Coefficient</th>
<th>Odds ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>1.653</td>
<td>5.221</td>
</tr>
<tr>
<td>Gradient of first 10 points</td>
<td>-6.621</td>
<td>(750.7)⁻¹</td>
</tr>
<tr>
<td>Gradient of first 20 points</td>
<td>5.581</td>
<td>265.3</td>
</tr>
<tr>
<td>Location of first turning point</td>
<td>-0.3741</td>
<td>(1.454)⁻¹</td>
</tr>
<tr>
<td>Value of first turning point</td>
<td>0.1875</td>
<td>1.206</td>
</tr>
<tr>
<td>Number of turning points</td>
<td>-0.5361</td>
<td>(1.709)⁻¹</td>
</tr>
</tbody>
</table>

Table 4.1: A summary of the logistic regression fit to the particle channel. All values are quoted to 4 significant figures. The odds ratio for each variable is the exponential of the logistic regression coefficient for that variable. Where the odds ratio is less than one, it is expressed in terms of its inverse so the values can more easily be compared.

Figure 4.4: A plot showing how the classification error changes as the number of variables in the particle channel model is reduced. The optimum performance is achieved when using the two most significant variables.
Figure 4.4 shows how the error changes when the number of variables in the model is reduced by removing the least significant variable each time, and it is clear that the error is minimised when the model contains only the two most significant variables. Further reduction of the model to use only one variable increases the error to 31.8%, very close to the null error, meaning that a model with only one of these variables is not worth using. I confirmed that this is the case regardless of which of the two most significant variables is removed.

### 4.3.3 Final model

![Figure 4.5: Results from the best particle channel logistic regression model. Predicted (open circle) and true (filled circle) bijel classification results for each sample in the dataset. Predictions were output from the final particle channel model and plotted as a function of the two variables in the model, using the same standardised values on which the model was trained. The line where a sample is equally likely to be a bijel or a non-bijel is plotted in blue. Here, all the variables have been standardised so are in arbitrary units.](image)

The best performing model using the particle channel is therefore one where I use logistic regression with two predictive variables: the gradients of a straight line fit to the first 10 and first 20 points of the autocorrelation function of the
This model has a classification error of 18.0%. This result is summarised in Figure 4.5, which shows that bijels are generally located at high values of the 10-point gradient and low values of the 20-point gradient. The line of equal probability (plotted in blue) shows the decision boundary used by the logistic regression algorithm to predict whether the sample is a bijel or not.

The equation for the final logistic regression fit to the particle is

$$\log \left( \frac{p(\text{Bijel})}{1 - p(\text{Bijel})} \right) = 1.616 + 6.152(20 \text{ point gradient}) - 6.989(10 \text{ point gradient})$$

or

$$p(\text{Bijel}) = \frac{e^{1.616+6.152(20 \text{ point gradient})-6.989(10 \text{ point gradient})}}{1 + e^{1.616+6.152(20 \text{ point gradient})-6.989(10 \text{ point gradient})}}.$$  \hfill (4.2)

<table>
<thead>
<tr>
<th>Training/test data</th>
<th>True non-bijel</th>
<th>True bijel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted non-bijel</td>
<td>28</td>
<td>8</td>
</tr>
<tr>
<td>Predicted bijel</td>
<td>15</td>
<td>84</td>
</tr>
<tr>
<td>FPR: 34.9%</td>
<td></td>
<td>FNR: 8.7%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>New data, similar composition</th>
<th>True non-bijel</th>
<th>True bijel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted non-bijel</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>Predicted bijel</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>FPR: 10.0%</td>
<td></td>
<td>FNR: 12.5%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>New data, different composition</th>
<th>True non-bijel</th>
<th>True bijel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted non-bijel</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Predicted bijel</td>
<td>9</td>
<td>22</td>
</tr>
<tr>
<td>FPR: 100%</td>
<td></td>
<td>FNR: 0%</td>
</tr>
</tbody>
</table>

Table 4.2: A breakdown of the predictions of the optimised particle channel model when applied to three different data sets. Each combination of true and predicted bijels and non-bijels is shown with a count of how many data points fit into that category. The numbers of correctly classified data points are shown in plain text, and the numbers of incorrectly classified data points are shown in italics. The false positive rate (FPR) and false negative rate (FNR) are included for each data set.

This model out-performs the null error rate on only one of the two additional validation data sets. For the different-composition data, the model performance matches the null error rate of 42.1%. For the similar-composition data the model
error rate of 10.5% clearly outperforms the null error rate of 29.0%. Table 4.2 outlines these results in more detail and shows that the model matches the null error rate on the new data with a different composition because it provides the same classification as the null error rate. It also shows that the rate of false positives and false negatives is much more balanced when the model is applied to the new data with a similar composition.

### 4.4 Combined dual-channel results

When I combined the optimal variables from both the liquid and particle channels, I unsurprisingly found that the KNN and logistic regression methods gave the best performance. Of these, logistic regression gave the lower error so I explored this model a little further by assessing the impact of non-linear dependencies of the logit function on the predictive variables. This consistently decreased the performance confirming that the linear logistic regression was the most suitable version.

![Figure 4.6: A plot to show how the classification errors of the logistic regression and k-nearest neighbours models change for all possible combinations of the variables in the combined model. The points corresponding to the optimal models using variables from only the particle channel or the liquid channel are labelled for comparison. The optimal performance is achieved using the logistic regression algorithm with all three variables.](image-url)

87
In order to confirm that the inclusion of all three previously chosen variables gave for the best result, I tested all previously untried combinations of pairs of these variables using both the KNN and logistic regression algorithms. As shown in Figure 4.6 I found that, with both the KNN and logistic regression algorithms, the inclusion of all three variables gave the best performance. The final model is therefore a logistic regression described by the following equation:

$$\log \left( \frac{p(\text{Bijel})}{1 - p(\text{Bijel})} \right) = 1.643 + 5.077(\text{Particle channel 20 point gradient}) - 6.048(\text{Particle channel 10 point gradient}) - 1.167(\text{Liquid channel first turning point}).$$  (4.3)

Figure 4.7 shows the predictions of the best performing model fitted using variables from both the liquid and particle image channels, and compares them to the true classifications. In lieu of a three-dimensional plot, the results are plotted in three different projections to show how the classification decision depends on each combination of the three variables in the model. This model achieved a cross-validated error of 14.6%.

Table 4.3 shows a breakdown of this result as well as the performance of this final combined model on the two unseen data sets. The addition of the liquid channel
information has led to an improvement on the particle channel model, and now the null error is out-performed for both data sets. The model error rate on the different composition data is 25.8%, now better than the null error of 29.1%. The model error rate on the same composition data is the same as the model containing only the particle channel, at 10.5%, but the balance of false positives and false negatives has changed.

<table>
<thead>
<tr>
<th>Training/test data</th>
<th>True non-bijel</th>
<th>True bijel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted non-bijel</td>
<td>31</td>
<td>7</td>
</tr>
<tr>
<td>Predicted bijel</td>
<td>12</td>
<td>85</td>
</tr>
<tr>
<td></td>
<td>FPR: 27.9% FNR: 7.6%</td>
<td></td>
</tr>
<tr>
<td>New data, similar composition</td>
<td>True non-bijel</td>
<td>True bijel</td>
</tr>
<tr>
<td>Predicted non-bijel</td>
<td>11</td>
<td>2</td>
</tr>
<tr>
<td>Predicted bijel</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>FPR: 0% FNR: 25.0%</td>
<td></td>
</tr>
<tr>
<td>New data, different composition</td>
<td>True non-bijel</td>
<td>True bijel</td>
</tr>
<tr>
<td>Predicted non-bijel</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>Predicted bijel</td>
<td>3</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>FPR: 33.3% FNR: 22.7%</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.3: A breakdown of the predictions of the optimised combined channel model when applied to three different data sets. Each combination of true and predicted bijels and non-bijels is shown with a count of how many data points fit into that category. The numbers of correctly classified are shown in plain text, and the numbers of incorrectly classified data points are shown in italics. The false positive rate (FPR) and false negative rate (FNR) are included for each data set.

4.5 Validation of the final results

Once I had chosen the combined model as my final model, based on its classification performance, I used a number of tests to assess the performance of the model further. This allowed me to gain more information on the quality of the final model besides the classification error, and to verify that it is a useful classification tool.

Figure 4.8 is a ROC (receiver operator characteristic) curve showing the balance between the true and false positive results as I change the probability threshold above which a sample is classified as a bijel. The area under the curve (AUC) is 0.91, a significant improvement on the null error case of AUC = 0.5.
Figure 4.8: Receiver operator characteristic curve for the final combined channel model fit, used to measure the quality of the fitted model. This shows how the true and false negative results change as a function of the probability threshold at which a sample is classed as a non-bijel. The area underneath the curve (AUC) is 0.91, indicating an excellent model.

Figure 4.9: Histogram of the error rate for 1000 iterations of the final model fitted with random bijel labelling. These errors are all significantly higher than the error achieved with the model fit to the true data, shown as a purple line.
As a test of whether the images carry useful information about the classification of the samples, I randomised the classification of the images. Figure 4.9 shows the error rate for 1000 iterations of the final model with random labelling rather than the true bijel/non-bijel labels, and compares it to the error for the model fit to the true labels. The model error is significantly lower than any of the randomly labelled examples.

4.6 Discussion

4.6.1 Model performance

In this chapter, I have investigated the performance of two different approaches for identifying successful bijel samples using information from the particle channel. Using the result from Chapter 3 as a reference point, I can assess these performances in more detail.

Particle channel model

Using variables from the particle channel alone, I was able to create a model that categorises successful and unsuccessful bijels based on a single two-dimensional image with 82.0% cross-validated accuracy, a significant performance increase on the null error. This model is, in general, better at identifying successful bijels than unsuccessful ones, with a false negative rate of only 8.7% in contrast with the false positive rate of 34.9%, as shown in Table 4.2. In part, it is likely that this is an effect of the distribution of bijels and non-bijels in the training data, as the algorithm has more examples of successful bijels from which it can learn to identify them correctly. The table also shows that, when the model was validated on unseen data, it performed much better on data with a more similar experimental composition to the training data. When applied to data with a different composition, the model classified all images as a bijel and therefore could not outperform the null error. This indicates that the variables derived from the particle channel images are highly susceptible to changes in the composition of the bijel samples, and that the model would require frequent re-training in order to be useful as a bijel identification tool.

This model depends on two variables derived from the autocorrelation function
of the images, both measures of the gradient at the start of the function. As outlined in Equation 4.1, a higher value of the gradient of the first 20 points of the function indicates an increased likelihood that the image is a successful bijel, and a higher value of the gradient of the first 10 points indicates an increased likelihood that the image is an unsuccessful bijel. Looking at examples of the autocorrelation function in Figure 4.10, we can guess why this might be the case: the non-bijel example continues its initial downward slope for significantly longer than the bijel example, so its gradient over the first 20 points (approx 25 µm, marked by the green line in the figure) is much steeper. On the other hand, the bijel example is slightly steeper at the very start of the function, which is picked up by the 10-point gradient (the first 12.5 µm, marked by the blue line in the figure).

Figure 4.10: Examples of the autocorrelation function of the particle channel of representative bijel (in red) and non-bijel (in black) confocal images. The location of the 10th and 20th points in the image are marked by blue and green vertical lines, respectively. 12.5 µm corresponds to approximately 10 pixels in the image.

Figure 4.5 shows how the final particle model classifies each image in the training data. Because I used logistic regression with two variables to obtain this result, I could use the model parameters outlined in Equation 4.1 to plot the decision boundary that separates the predicted bijels from the predicted non-bijels. In my case, because I used a probability threshold of 0.5, this line occurs where the model dictates that a sample is equally likely to belong to either category. If desired I could change this probability threshold in order to adjust the predictions and perhaps improve the accuracy of the model’s predictions. This would not affect the gradient of the decision boundary, but it would shift the intercept and therefore change the predictions of the points near the boundary. The ability
to plot the decision boundary with such certainty allows me to think about the different reasons for a sample to be wrongly classified. First, there is evidence that the linear decision boundary assumed by my use of logistic regression is suitable for these data, as there are no trends in the incorrectly classified points to indicate any tendency towards a curved decision boundary. Some samples on the wrong side of the decision boundary are very close to the boundary, and therefore are most likely misclassified for the very reason that this project is useful at all: the fact that some bijels aren’t perfect and some non-bijels are close to being successful. Other incorrectly classified samples are further from the decision boundary, suggesting that they are somewhat anomalous for their class, at least in terms of the shape of their autocorrelation function. These samples are the ones most likely to indicate some shortfall of the model: it does not include a key feature that, to the human eye, makes these samples look like other members of their class.

**Combined particle and liquid channel model**

A model using variables from both the particle and liquid channels has proved to be the best at classifying successful and unsuccessful bijels, with a cross-validated classification accuracy of 85.4%. Unlike the model using variables only from the particle channel, this model also outperforms the null error rate on both unseen data sets as well. Its performance on the unseen data with a similar experimental composition is identical to the particle channel overall, but the split between false positives and false negatives has shifted: there are now only false negatives, rather than one of each. With only two misclassified images in total it would not be sensible to place much importance on this observation, particularly when the results on the other two data sets are both skewed towards false negatives despite the fact that one contains predominantly bijels and the other is predominantly non-bijels.

The plot of the results in Figure 4.7 shows that, as expected, the decision boundary is no longer linear in any of the projections plotted. This is because the decision boundary is a plane rather than a line so it is impossible to plot a linear boundary on any of these projections unless the coefficient of one of the parameters in Equation 4.3 is effectively zero. It was therefore not possible to plot the decision boundary on these plots as I did with the particle channel model.

I can plot the particle channel decision boundary on the final panel of the
combined channel results plot to compare the performance of the two models in detail, as shown in Figure 4.11. From this figure it is clear that the addition of the liquid channel variable to the particle channel model has had both a positive and a negative effect: some of the samples classified incorrectly by the combined model would have been classified correctly by the particle channel model, and vice versa. Of course, the greatly improved classification accuracy of the model overall shows that the improvements outweigh the downsides of the few newly-misclassified samples.

Figure 4.11: A plot of bijel classifications for each sample in the training dataset, with the true classifications represented by filled circles and the classifications predicted by the particle channel model represented by open circles. The decision boundary defined by Equation 4.1 with $p(\text{Bijel}) = 0.5$ is shown as a blue dotted line. Here, all the variables have been standardised so are in arbitrary units.

The additional validation performed on this model can reveal more about its performance. The ROC curve in Figure 4.8 shows that the model performs well. The curve goes close to the error-free point at $(0, 1)$ and is very far from the straight $y = x$ line that would indicate the null error case. The area under the curve of 0.91 is in the range generally viewed as excellent, as it is much closer to the perfect test case of AUC = 1 than the null error case of AUC = 0.5. The ROC curve also shows that at my chosen probability threshold of 0.5, I am accepting
quite a high level of false positives in order to increase the true positive rate. However, if I decreased the probability threshold then the false positive fraction would increase much more than the true positive fraction, and if I increased the probability threshold then the true positive fraction would start to decrease a lot without much change in the false positive fraction until a cutoff of 0.7. Therefore, although the value of 0.5 may not be the optimal value, it is certainly close enough. The high false positive rate compared to the false negative rate (1 – the true positive fraction) is just another way of representing the explicitly calculated result shown in Table 4.3 that the false positive rate is 27.9% and the false negative rate is 7.6%. In this application, it is more desirable to minimise the false negative rate because that minimises the chances that a successful bijel sample is incorrectly rejected.

**Comparison to previous result**

At the end of Chapter 3, I concluded that a bijel classification tool based solely on one liquid channel of the bijel, although it performed well on the data on which it was trained, was not useful when applied to unseen data. Table 4.4 shows that the particle channel includes information that makes the classification model much better at identifying bijels from an unseen data set, provided that the new data represents bijels with a similar experimental composition to the training data. The table clearly shows that the model using variables from both the liquid and particle channel gains the benefits from both channels while avoiding the downsides: the combined model performs better than either of the single-channel models.

<table>
<thead>
<tr>
<th>Error</th>
<th>Liquid channel</th>
<th>Particle channel</th>
<th>Both channels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cross-validated</td>
<td>16.4%</td>
<td>18.0%</td>
<td>14.6%</td>
</tr>
<tr>
<td>Training data</td>
<td>14.8%</td>
<td>17.0%</td>
<td>14.1%</td>
</tr>
<tr>
<td>Similar data</td>
<td>15.8%</td>
<td>10.5%</td>
<td>10.5%</td>
</tr>
<tr>
<td>Different data</td>
<td>38.7%</td>
<td>29.0%</td>
<td>25.8%</td>
</tr>
</tbody>
</table>

**Table 4.4:** A summary of the classification performance of the three different optimised models on different data sets. The errors obtained by applying the models to the data on which they were trained are consistently a little lower than the cross-validated errors quoted in this chapter, but both allow for easy comparison of model performance.

The analysis in this chapter also provides a little more insight into the
performance of different machine learning algorithms for this task of identifying successful bijels. Because the optimisation of the combined channel model involved briefly revisiting the liquid channel model, but this time considering the logistic regression algorithm as well as k-nearest neighbours, I have seen that logistic regression actually performs slightly better once the choice of variables in the liquid channel model is fully optimised. In Chapter 3 I chose not to compare multiple models throughout the long process of choosing the optimal variables but the results in Figure 4.6 show that once the choice of variables has been somewhat optimised then the picture of which algorithm performs best becomes more consistent. This suggests that if this approach is used in future it might be worth assessing the performance of different classification models once the optimal variables have been chosen. Figure 4.6 also suggests that the difference in performance between different models increases as the number of variables in the model increases. This makes sense because, as the number of variables decreases to the limiting case of a single variable, there are fewer different ways for a decision boundary to be drawn so it is logical that algorithms would converge on a similar result.

I can also compare the complexity of the decision boundaries allowed by each of my models. The particle channel logistic regression model gives a straight line boundary which is arguably the simplest of the three, despite the liquid channel using only one variable, because the k-nearest neighbours algorithm allows for a more complex boundary than a single straight line even with few variables. Comparing the liquid channel model to the combined model, shown in Figure 4.12, it is clear that the use of even one additional variable in the model allows for samples with the same value of liquid channel turning point to be sufficiently separated to tell them apart without the need to introduce a jitter to the plot. This separation helps the logistic regression algorithm to be successful, and the addition of a third variable makes it easier to achieve an accurate decision boundary. The particle channel information adds a little more structure to the data, with the non-bijels having a generally higher value of the particle channel 10-point gradient, which allows for more distinction between the two classes.

Finally, I can briefly compare these results to those obtained by another method on the same data [1]. This method involved decision trees combined with a global optimisation method applied to variables captured by directly processing the micrograph images, such as the area per perimeter of liquid domain shapes and the fractal dimensionality of a pre-processed version of the image. The global
Figure 4.12: A comparison of the predictions of the liquid channel model from Chapter 3 to a projection of the combined particle and liquid channel model presented in this chapter. The inclusion of more variables leads to a better performance in the combined model, with fewer samples classified incorrectly.
optimisation method was used to optimise the pre-processing steps applied to each image, to find the combination of processing steps that gave the lowest classification error. This approach reduced the classification error to 12.0% with a classifier based on the fractal dimensionality, area per perimeter, Feret mean diameter, and the solidity of each processed image. As with the methods outlined in this chapter, most of these descriptors characterise the lengthscale of the bijel domains in some way. Although this alternative method gave a lower classification error, the methods outlined in this thesis make use of the autocorrelation function to drastically reduce the amount of time and human input to the process of obtaining variables to input to the machine learning algorithm by avoiding the need to optimising any image pre-processing. On balance, both methods have their benefits depending on the priorities of the task at hand.

4.6.2 Model interpretation

It is clear from the results in this chapter that information from both the particle and liquid channels is required in order to most successfully classify bijels, but that of the two the liquid channel is more important. This indicates that the most important defining feature of a bijel is the characteristic length scale of the liquid channels, as discussed in Chapter 3, but that the particle channel still adds useful information.

Unlike in the case of the liquid channel, the position of the first turning point in the particle channel autocorrelation function, representing the characteristic length scale of the particle channel, proved not to be useful. This makes sense because it is the liquid channel of a bijel that has a defined characteristic length scale, whereas the particles form a monolayer coating the interface. Instead, the variables of interest in the particle channel are the gradients of the first 10 and 20 points (approximately 12.5 µm and 25 µm) of the autocorrelation function. Because the autocorrelation function always starts at a value of 1 by definition, this gradient is approximately equal to the value of the function at 12.5 µm and 25 µm, scaled by the gradient range and translated by the intercept. The only caveat is that, in the straight line fit to the relevant section of the function, the intercept is free to change. This value could be constrained in future repetitions of this process, but in this example the intercepts are all between 0.93 and 1.01, so they are quite similar and it was not worth repeating the process to constrain these values. Physically, therefore, the result from both models including the
particle channel suggests that bijels are likely to be more uniform over a range of 25µm than non-bijels, but less uniform over a range of 12.5µm. It is a little harder than in the previous chapter to link these to physical features of the bijel, but they seem to be capturing information about lengthscales of the bijel in a slightly different way.

It is interesting to note that, despite the fact that the liquid channel model consistently outperforms the particle channel model, the liquid channel variable in the combined model is a factor of 5 smaller which indicates that it is less significant in the model. As all the variables are standardised, this cannot be due to differences in the scale of the variables, and this result is the same even if logistic regression is used for both models. It might therefore be the case that the particle channel model was quite close to correctly classifying a significant number of samples, and just needed a small adjustment from the liquid channel to shift these from an incorrect classification to a correct one. This highlights the fact that the combination of different variables can often be a more powerful classifier than might be expected from the predictive power of each variable alone, or from a subset of the variables. This conclusion highlights a key benefit of using logistic regression instead of k-nearest neighbours when taking a physics-led approach to this classification task, because the parameters of the model itself can be interpreted more thoroughly. In this case the choice of algorithm gives a clear insight into the relative importance of each of the variables in the final model, which would not be as easy to replicate using the k-nearest neighbours algorithm.

4.7 Conclusions and future outlook

In this chapter I have shown that variables from the autocorrelation functions of both the liquid and particle channels of a bijel image are needed to most effectively identify bijels. With this approach, using logistic regression and three variables, I reduced the classification error rate from 16.4% to 14.6% by adding variables from the particle channel information to the liquid channel variable, despite the fact that the particle channel variables alone could only classify bijels with an error of 18.0%. The use of a combined channel model has had an even greater effect on the classification performance of the model on unseen data, where the combined model performs consistently as well as the better of the two single channel models for each data set. This addresses one of the key flaws of the liquid channel model presented in Chapter 3.
I have also shown that, assuming a sample is always classified into the most likely category, the false positive rate is greater than the false negative rate. This means that the classification tool is probably most useful for filtering out most of the failed bijels from a data set, leaving an experimenter with a smaller subset of images from which to remove the few remaining non-bijels. The optimal model presented in this chapter uses the logistic regression algorithm which outputs a probability that each image depicts a bijel, which could be used as additional information to help an experimenter classify their samples.

I have concluded that, in addition to the characteristic length scale of the liquid channel, bijel samples can be identified by the uniformity of the particle channel image over certain length scales. In other words, how uniform the brightness of an image is over a patch of a given size. Unlike the characteristic length scale, the variables used to describe this feature depend on the specific length scales of the images in the data set so, now that I have arrived at a potential physical interpretation for the variables, it would be valuable to investigate the use of other variables that can capture the same feature without specifying a length scale. If this is possible it would add evidence to my physical interpretation of the particle channel variables used in this chapter, and it would also make the final model more useful for understanding what features set a bijel apart from a non-bijel. It might also give more insight into how variables derived from the liquid and particle channels can give similar and different information about the bijel structure.
Chapter 5

Using unsupervised learning to analyse the structure of oppositely charged particles on the surface of a sphere

In this chapter I present an example of how machine learning can be used for exploratory data analysis. I use unsupervised machine learning to cluster image data, and I evaluate the membership of each cluster to probe the physical basis for the observed behaviour. By this route, I may be able to overcome some problems caused by the surface particles being difficult to identify individually. I also hope to investigate what features of the image are important to the clustering decision, and therefore which physical features set each cluster of images apart.

5.1 Introduction

5.1.1 Emulsions stabilised by oppositely charged particles

As discussed in Chapter [1] emulsion droplets can be stabilised in a very robust way by the presence of particles on the oil-water interface, making a Pickering emulsion. Particles can successfully stabilise a droplet but many common colloids are inherently hydrophilic so are repelled by the oil-water interface. Reducing the
charge of such particles increases their oil wettability and therefore their ability to adsorb onto the surface, and this is often achieved by the addition of electrolytes or surfactants.

Another, more novel, approach is to use a mixture of positively and negatively charged particles to create the same effect as changing the charge. This was first done using nanoparticles [143, 144], with the mixture of particles forming mixed aggregates with a charge closer to zero that allowed them to successfully stabilise emulsions. It was suggested that these aggregates therefore have a wetting angle closer to neutral wetting, and that this was the reason they could easily stabilise the interface.

A further study confirmed that the effectiveness of this approach is not limited to nanoparticles: mixtures of sub-micron and micron sized oppositely charged particles can also stabilise droplets effectively [145]. This study goes further to investigate the effect of varying various parts of the system, including the oil:water ratio and the ratio of positive:negative particles. The variation of the oil:water ratio found a phase inversion from oil-in-water to water-in-oil emulsions when the mixture is 80% oil or more. The variation of the positive:negative particle ratio found that droplets can be stabilised provided that at least 10% of the particles are of the minority composition, and that these droplets stabilise the most collective volume of droplets when there are 30-50% positive particles. Additionally, it was discovered that the diameter of the stabilised droplet changes as the particle ratio is varied, with a minimum when positive and negative particles are present in equal amounts and a symmetrical increase either side of this value. In other words, the behaviour of a system with 20% positive particles was largely the same as one with 20% negative particles.

In this chapter I investigate whether this symmetrical behaviour applies to a chemically different emulsion and to other features of this new system. I investigate the droplet roundness using traditional image analysis methods, and I use machine learning for exploratory analysis of the particles on the surface of the droplets. As well as looking for symmetry, I investigate how effective machine learning can be in general, and whether the results can be interpreted to give insight into what features of the particle distribution are important for the clustering decision.
5.2 Data

David French collected and kindly provided the images for me to analyse here. They are a series of confocal micrographs of emulsion droplets stabilised by a mixture of positive and negative particles, where each image is the two-dimensional projection of a z-stack through part of the droplet. The aqueous phase of the emulsions was distilled water, and the oil phase was anhydrous dodecane. The negatively charged particles were silica labelled with rhodamine-B and the positively charged particles were polystyrene labelled with fluorescein methacrylate. The two different particle types were imaged separately based on their fluorescent dyes, so each image has two separate channels to consider: one for the positive particles and one for the negative. For some images, there was also a brightfield channel.

There are two parameters varied throughout the experiment: the total fraction of particles in the system \( \phi_p \) and the fraction of those particles that is dyed red (i.e. the negative particles) \( \phi_r \). \( \phi_p \) takes values of either 0.5% or 1%, and \( \phi_r \) varies between 10% and 75%. For each different combination of these parameters, I had access to between one and four images at 63× magnification (15 in total) and one image at 10× magnification (9 in total).

5.3 Traditional image analysis methods

An initial qualitative assessment of the images at 10× magnification suggests that as \( \phi_r \), the proportion of particles that are labelled with rhodamine-B and imaged as red, increases then the droplets become more round. With these images, it was difficult to automatically distinguish the individual droplets that could be identified by eye and after attempting a number of methods to achieve this I settled on a manual approach. I used ImageJ \cite{146, 147} to analyse the roundness and circularity of the droplets in the 10× images. ImageJ was an ideal tool for this task because it allowed for easy exploration of the various image processing steps, and since there were very few images in the dataset it was feasible to process them one at a time. It was particularly useful when dealing with the fact that the exact level of magnification often changed between images: I ensured that the true scale in \( \mu m \) was saved in the image, and these real units were automatically used for all my measurements.
Figure 5.1: Two examples of 10× magnification images suggesting increasing circularity of the droplet with increasing $\phi_r$. The scale bar in each image is 100$\mu$m.

Figure 5.2: An example of manually defined droplets in an image at 10× magnification. Each droplet was identified by eye based on information from all channels and the outline was drawn manually then smoothed. The scale bar in each image is 100$\mu$m.
For each image I methodically selected around 30 droplets, starting in the top left-hand corner of the image and outlining droplets that had a clear edge in at least one of the imaging channels (preferably more). An example is shown in Figure 5.2. I measured each droplet outline using ImageJ to return the area $A$, perimeter $P$, and circularity ($4\pi \times A/P^2$) of the droplet, as well as the roundness of an ellipse fitted to the outline. I plotted the mean and minimum circularity and roundness to see how each different measure changed with $\phi_r$, the results of which I show later.

5.4 Machine learning methods

5.4.1 Machine learning approach

![Two examples of 63× magnification images suggesting increasing circularity of the droplet with increasing $\phi_r$. Each image is a 2D projection of a z-stack through the droplet. It is difficult to distinguish individual particles on the droplet surface.](image)

Figure 5.3: Two examples of 63× magnification images suggesting increasing circularity of the droplet with increasing $\phi_r$. Each image is a 2D projection of a z-stack through the droplet. It is difficult to distinguish individual particles on the droplet surface.

Although I perform some traditional analysis on these images, there are limitations to this approach. In the case of the 63× magnification data it is difficult to identify the location of each individual particle on the surface of the droplet. This means that I am unable to investigate the local arrangement of these particles to see whether this changes with different experimental parameters i.e. the total number of particles and the fraction of those particles that are positively
charged. The analysis is also limited to features I can identify in the image, but machine learning may be able to pick up other patterns in the data.

Unlike in the Chapters 3 and 4, these data are not sorted into any categories: they are unlabelled. This impacts my choice of machine learning approach because it limits me to the realm of unsupervised learning. My aim is to analyse the 63× magnification images and find out if there are any trends that can be seen in the data. I would like to discover whether there is any particle-level structural information that is hard for a human to see but can be picked up by machine learning algorithms.

I am interested to see if macroscopic trends in the behaviour of similar systems such as the similarities between a system with 25% positive particles and one with 25% negative particles [145] are reproduced by the outcome of unsupervised machine learning looking at the system at a greater magnification. In general, I would like to find out which droplets are most similar in terms of particle arrangement on the surface of the droplet. I therefore decided to focus on clustering algorithms, as they are suited to finding which data points are most similar and grouping them together. They are also generally useful for exploratory data analysis, which is what I aim to do: I will see what patterns emerge in the data and try to understand what these patterns might mean.

5.4.2 Clustering algorithms

In this chapter I used two different methods of unsupervised clustering. I chose to use relatively straightforward algorithms because they make it more likely that I will be able to delve into the process of how the result was achieved; this in turn makes it easier to understand the results in terms of a physical interpretation. With a more black-box approach, this would be impossible. I can also afford to use less efficient algorithms than many of the common applications of machine learning because I am working with a very small set of data, so the computational efficiency in terms of scaling for large amounts of data is of little concern in this application.

I used k-means clustering which is loosely related to the k-nearest neighbours classifier I used in Chapters 3 and 4. It simply groups the data into the specified number of clusters by finding the solution that minimises the variance within each cluster, as described in Chapter 2. I used the KMeans function within
the sklearn.cluster package for k-means clustering, which requires the user to define a model with input parameters such as the number of clusters to use, then to use a fitting function to fit that model to the data. The function outputs variables including labels describing which cluster each data point is in, and the inertia measure of within-cluster variance. As there is a random component in the model, I set a constant random seed to make sure I got the same result each time I applied the model to the same data so that I could produce different graphs representing the same results.

As discussed in Section 2.2.7, one downside of k-means clustering is that it is difficult to choose the number of clusters to split the data into. I therefore also applied hierarchical clustering to the same data: although it still requires some input to select the number of clusters, it gives the user some additional information with which to make that decision via the use of dendrograms to display the results. It is also useful to try another clustering method to see if the results are similar: if they are it will give me more confidence in the result. As with the k-means analysis, I used Euclidean distance as the measure of similarity in this algorithm. I used the Ward method to determine the measure upon which the decision to merge together two clusters is based, so the within-cluster variance was minimised at each stage.

5.4.3 Preparing the data for machine learning

Unlike the data used in Chapters 3 and 4 in this case I did not have access to hundreds of images with which I could train a machine learning algorithm. I therefore needed to create more data points from the limited number of images available. As the images are of droplets, and therefore generally have radial symmetry, I chose to slice each image into 12 wedges around the centre of the droplet, as shown in Figure 5.4.

I went about this in two ways, to gain two different formats of data to which I could apply different analysis methods. Before using either of the following approaches, I used ImageJ to identify the centre of each droplet by drawing an ellipse around the droplet and measuring the ellipse to obtain the coordinates of the estimated centre. First, I wanted to present each wedge as a rectangular array that I could treat as a typical image, as shown in Figure 5.5. For each value of \( r \), the distance (in 0.1\( \mu m \) intervals) between a point and the centre of the droplet, I wanted 100 values of \( \theta \) for each slice (i.e. 100 points on an arc across the
Figure 5.4: An example of how a droplet image was split into 12 sub-images.

For each combination of $r$ and $\theta$ I found the nearest pixel by rounding the corresponding $x$ and $y$ coordinates to the nearest integer, and I took the intensity value of this pixel as the value for the new image slice. I did this for $r$ values ranging from 0 to a little smaller than the radius of the smallest droplet, so that no slices included information from outside the droplet. Although this method creates a nice rectangular image with which to work, it has the obvious downside that the image is significantly stretched close to the centre of the droplet so it is unsuitable for analysis that requires the absolute size of features to be preserved, or when features at different radii might be directly compared.

For these types of analysis, I took a second approach to slicing the images into 12 wedges like the ones shown in Figure 5.4, rather than stretching them out into a rectangular shape. I took the image as an array and created two masks: one a circle of given radius $R$ centred about the centre of the droplet, and one an isosceles triangle with sides of length $1.5 \times R$ (in order to ensure they extend past the outer edge of the circle) and an angle of $\frac{\pi}{6}$ radians at the centre of the droplet (one twelfth of the circle). Combining these two masks with an AND operator gave me a wedge-shaped mask that enclosed a twelfth of the droplet. I then applied this mask to the image 12 times, rotating the underlying images by $\frac{\pi}{6}$ radians between applications. The pixels outside of the desired wedge were set to a null value, and those within the wedge were preserved at their original value.
As each droplet image is a projection of a z-stack into the droplet, all particles visible are on the edge of the droplet. This means that there is no particular meaning to the position of a particle in the image, so the radial slicing described above is certainly not the only suitable method for taking multiple sub-images from each droplet image. I chose this approach because all the images show a round droplet centred at different and it was therefore more difficult to slice the images into, for example, square sub-images while ensuring that these sub-images only contain the droplet and not the background. Although the radial symmetry of the droplet has little meaning in terms of the physical system, I could still exploit it to make it easier to treat each image in the same way to gain sub-images suitable for my analysis.

With both of the methods I used, as the image consisted of two fluorescent particle channels, the end result was 24 sliced images for each original image. I therefore had 360 data points to investigate. I would expect the 24 slices from each image to be clustered together, but this is not guaranteed to happen so this expectation will be a good check of how believable my clustering results are.
5.4.4 Extracting variables for machine learning

In this application of machine learning, I had no prior information on which images were similar to each other (except the knowledge of which slices came from the same image). It was therefore not possible to assess the suitability of any individual variables that I might extract from the data, so I used a much greater number of variables in order to ensure I captured the features of each image fully.

I first chose to use the image data directly as a set of variables. I went about this in two ways: first I used each pixel in the rectangular image slice as its own variable, and I also used a $\theta$-averaged version in which I flattened the 2-dimensional image slice to a 1-dimensional profile with a variable for each value of $r$. In either case, I scaled the data to ensure that it was normalised with zero mean and unit standard deviation. I used standard scaling (via the StandardScaler function in Python) to ensure that the algorithms I used didn’t weight any of the variables as more important simply due to them being on a larger scale.

I added to these variables by deriving an additional set based on the properties of connected regions in the wedge-shaped slices. I thresholded each slice and applied the closing function to it: this process is achieved by first dilating each pixel of signal (replacing each pixel in the original image, Figure 5.6(a), with a larger one of a chosen shape, Figure 5.6(b)) and then eroding the resulting image (the inverse process of dilating, Figure 5.6(c)) in order to (mostly) preserve the original size of the signal area. The end result is a form of noise removal in which small holes in a region of signal will be removed, as demonstrated in Figure 5.6(d). I then used the measure.label function from the skimage Python package to identify and measure the area, perimeter and centroid position of each connected region within

Figure 5.6: A demonstration of the processes of dilation (b), erosion (c), and closing (d) on an example image (a). Taken from [148] under fair use.
the image. From this I calculated the following variables: average and standard deviation of the regions’ areas, average and standard deviation of the distance between the regions (measured as the difference between the centroids), the total area of all the regions, and the average circularity of the regions. These variables can give some insight into features of the image such as its homogeneity and the quality of particle dispersion. They may also pick up on some experimental parameters such as the concentration of each type of particle. These additional variables had very different magnitude to those derived directly from the image intensity, so again I made sure to standardise the data so each variable had the same mean and standard deviation. This means that the algorithms won’t accidentally count one variable as more or less important than the others just because it is scaled differently.

Principal component analysis

I investigated the use of principal component analysis (PCA) for two particular aims: to reduce the noisiness of the data, and to reduce the number of variables in the machine learning models (therefore reducing the likelihood of overfitting).

As previously discussed in Chapter 2, PCA works by retaining only the components of the data that explain the most variance in the data. It can therefore be used to reduce noise, which often affects each of the principal components equally. The more components which are removed, the more noise is removed. Sometimes the noise is concentrated in the less important components that are removed first. In these cases, the noise can be almost entirely removed by PCA. I used the PCA function in the sklearn Python package to represent each rectangular image slice (not θ-averaged) by the principal components that explained a chosen proportion of the variance in the image, and I compared the clustering results on the data with PCA applied at different levels of variance to select the most suitable level.

Another potential use for PCA is to use the principal components in order to understand the data better. In this case with two-dimensional image data, the principal components can be reconstructed into an image to show which areas of the image contain the most variance. I performed a brief investigation into this but found it difficult to ascertain any physical meaning from the principal components of this particular data so I do not present these results in this thesis.
5.4.5 Quantifying clustering performance

In order to help choose a suitable number of clusters to split the data into, as well as to compare the performance of different clustering attempts, I considered a number of methods to quantify the performance of the clustering. I chose to base the measure on how effectively an algorithm groups together data points that originated from the same confocal image, because they are images of the same droplet so should display similar physical properties. This means that the clustering is more likely to be based on real physical characteristics of the system because the clusters will have been optimised with this in mind.

In essence, the measures I discuss below all quantify the purity of a group of data points. This unfortunately means that it is never possible to find a clear first-order turning point at the optimal number of clusters: by definition, a model with more clusters available will always perform more favourably. However, by looking for higher-order changes in the plot of measure vs. number of clusters, I can gain some insight into the approximate range of a sensible number of clusters. The measures are also useful for comparing the performance of different models with the same number of clusters, for example comparing different clustering algorithms or different input variables.

First, I took inspiration from decision tree algorithms and used a Gini score to quantify how frequently slices that came from the same original image were clustered together. It is defined as:

\[
\text{Gini score} = \sum p(1 - p),
\]

where, in this case, \( p \) is the fraction of slices from one image that are in a given cluster, and the sum is over all clusters. I therefore obtain a score for each original image, and I quantify the overall performance of a model by taking the mean of these scores across the images. The score is close to zero for a result that clusters together slices that are from the same original image and are therefore known to be (in general) similar for my purposes. Another alternative is the entropy:

\[
\text{Entropy} = -\sum p \ln(p),
\]

where I use the same definition for \( p \) and again sum over all clusters. The entropy
and Gini score actually give very similar results numerically \cite{10}, so I only used the Gini score.

I also used the gap statistic: a measure used to help choose a suitable number of clusters \cite{149}, commonly used with k-means clustering. It is calculated from the within-cluster variance (inertia) of the clustering result, normalised by the inertia of a reference result:

\[
\text{Gap statistic} = \ln(I_{\text{ref}} - I_{\text{result}}).
\]  

(5.3)

The reference result is obtained by applying the clustering to a number of random data sets with the same dimensions as the real data, and taking the mean of the within-cluster variance of these results. The gap statistic will be maximised at the optimal number of clusters, and can be more useful than the measures above because it accounts for the improvement in score associated with simply increasing the number of clusters. Unlike the others, this measure is not based on any information about which image a data point originally came from, so if it agrees with the Gini score then that is additional evidence that I have chosen a suitable number of clusters. To calculate the Gap statistic, I used code from an Anaconda contributor \cite{150}.

5.4.6 Visualisation of results

In this project I am working with a large number of variables, which makes it particularly difficult to visualise the results. I used a number of different approaches to visualising the data, depending on what information I wanted to analyse.

Similarly to the information contained in the Gini score measure discussed above, I first wanted to visualise the clusters based on the original image each slice came from. I used a stacked bar plot for this purpose (as shown in Figure 5.7(a)), using different colours to represent the different clusters into which the images were separated. These plots were useful for two purposes: ascertaining the difference in the results given by different models and gaining an understanding of how ‘pure’ each original image was in terms of the cluster membership of its slices. The latter purpose was particularly useful when I compared the results of a number of different models, because the existence of any images that were consistently more
or less pure could provide an interesting result in itself. I could also compare the results in terms of the experimental composition or in terms of which particles were depicted in the image.

Figure 5.7: Examples of plots used to visualise a clustering result.

In order to more easily assess how the clustering results depended on composition, I plotted the results in terms of the two compositional parameters: the total particle fraction $\phi_p$ and the fraction of the particles which were red $\phi_r$. On these axes I plotted a circle for each cluster with its size proportional to the cube of the number of data points of the given composition in that cluster. An example plot is shown in Figure 5.7(b). These plots allowed me to see if there were any clusters that dominated at any particular compositions, and if any compositions showed similarities in terms of the clusters present.

It was also useful to understand which features were shared by images that were grouped into the same cluster, essentially to understand if any variables were a strong indicator of belonging in a particular cluster. Physical meaning could be derived from knowing what features were the most different between clusters. In order to do this, I visualised the data that the clustering algorithm ‘sees’ by
plotting them as an image, as shown in Figure 5.8. In these plots, each of the 12 slices that make up the original was represented by a row of pixels and I plotted the cluster number as text on top of each pixel to indicate which cluster that particular slice belonged to. I made these plots for examples of images that were consistently grouped in the same cluster, in order to look for features that were representative of that cluster and therefore hopefully allow me to find a physical interpretation.

![Figure 5.8: An example visualisation of all the data points fed into a machine learning algorithm to represent two example images. Each horizontal slice in the image represents one of the 12 slices that were used as separate data points during the clustering. The small white numbers denote which cluster that slice was grouped into, and the same number is repeated at multiple x values to ensure it can be seen.](image)

(a) $\phi_p = 1\%$, $\phi_r = 60\%$, red  
(b) $\phi_p = 0.5\%$, $\phi_r = 40\%$, red

**t-distributed stochastic neighbour embedding**

As discussed in Section 2.2.6, t-SNE can be unreliable as a clustering tool. It is, however, very useful for visualising multi-dimensional data in two dimensions. I therefore used it as an additional method for visualising the results of clustering using other algorithms. I plotted the data in 2D based on its two t-SNE components, and if desired I could colour-code the points based on the feature of interest such as a compositional parameter or the results of a clustering method: an example t-SNE plot is shown in Figure 5.9. I can use these plots to look for overlap between features and clusters, or to simply see how many clusters might be most suitable for the data.

I also took some information from the clustering suggested by the t-SNE visualisation, focusing only on features that were robust to changes in the parameters in the t-SNE function and taking care to derive no meaning from the relative position of data points in different clusters. I used the visualisations to help me choose how many clusters to split the data into. This was useful.
because the measures described in Section 5.4.5 gave only an indication of a range of reasonable values for the number of clusters. I used t-SNE to choose a single number of clusters that was most likely to be suitable, providing it lay within the range indicated by the other methods.

5.5 Results and discussion

5.5.1 Traditional image analysis results

As mentioned in Section 5.3, the data used in this chapter proved difficult to analyse via automated processing, and a lot of manual input was required. This is probably because the droplets overlapped a lot and their edges were therefore not easily discernible. By defining the outline of each droplet manually, I was able to use the measured features of the droplets to look for trends in the shape of the droplets, as shown in Figure 5.10 with the outlined droplets shown in panels (a–g). For each composition, I calculated four possible measures: the mean and minimum roundness, and the mean and minimum circularity. To avoid basing the minimum measure on a single droplet that may be anomalous, I defined the minimum as the mean of the smallest 10% of the sample (i.e. the smallest three). Because the droplet edges were drawn manually, I prefer the use of roundness as a measure because it is obtained via the fitting of an ellipse to the shape which reduces the effect of any small inaccuracies in the shape. The circularity, on the
Figure 5.10: An overview of how circularity changes as the fraction of negative particles in the system ($\phi_r$) increases. Panels (a) and (b) show the trends of four different measures of droplet roundness and circularity as the fraction of negative particles increases in the case where $\phi_p = 0.5\%$ and $\phi_p = 1.0\%$, respectively. There is an indication that droplets become more round as more negative particles are added to the system, and there is no evidence of symmetry about $\phi_r = 50\%$.

Figure 5.10 panels (a) and (b) show how the roundness and circularity of the 0.5% and 1% particle images, respectively, change as $\phi_r$ (the fraction of the particles that are negatively charged) increases. It is perhaps unsurprising that the measures based on the minimum portion of the distribution show a stronger trend than those that are a mean of the whole distribution, but it is interesting to see that the mean circularity in particular shows basically no change as $\phi_r$ increases. This indicates that the centre of the circularity distribution doesn’t change a great deal, and it is possible that this value is dominated by the hand-drawing errors previously discussed.

Focusing on the minimum roundness and circularity, and particularly the roundness, it is clear that the droplets become more spherical (assumed from circularity in these 2D images) as $\phi_r$ increases. Comparing this finding to the conclusions of Nallamilli et al. [145], it is interesting to see an increasing trend rather than a symmetrical distribution about $\phi_r = 0.5$. This indicates that, at least in this system, high concentrations of positive polystyrene particles lead to
droplet distortion. As there is no mention in the literature of droplet distortion, only spherical droplets of differing size and phase, it is likely that this phenomenon is due to the choice of materials.

5.5.2 Basic clustering

Choosing the number of clusters

The first step in exploring the data with unsupervised clustering was to determine how many clusters the data should be split into. The dendrogram for hierarchical clustering of this data, shown in Figure 5.11(a), can easily be split into 4 clusters and could also be convincingly split into 5, indicating that some number of clusters around these values would be suitable. Figure 5.11(b) shows how the gap statistic for k-means clustering varies as the number of clusters increases, with data plotted for a variety of different versions of the data as discussed later in this chapter. The score will always increase with more clusters but this increase levels off at around 3–7 clusters, depending on the version of the data, implying that the optimal number of clusters is in this range. The Gini score for hierarchical clustering is not shown here because the dendrogram provides a clearer indication of the number of clusters in this case, but this metric suggests that 5 or 6 clusters would be best.

Turning to t-SNE visualisation, Figure 5.11(c–f) implies that these different forms of the data all fit into 4–6 clusters, generally agreeing with the results from the dendrogram and gap statistic. Panels (c) and (d), which include all of the data, give a convincing argument for the presence of 5 clusters but what looks like a cluster at the bottom of the plots may actually be part of the cluster above it and there are a few points at the tops of the plots that could justify the addition of another cluster. Panels (e) and (f), which include a subset of the data, look a little more different: panel (e) looks most comfortably grouped into 5 clusters but the two outlying points may present an issue, and panel (f) suggests that 4 or 5 clusters would be most suitable but the clusters are less clear as they merge together and lack a tightly grouped central region. The $\theta$-averaged form of the data is discussed later in Section 5.5.3 because in t-SNE space it looks different from the examples shown here, but it does not change my overall decision on the number of clusters to use. The appearance of these clusters is robust against changes in both the random seed and the perplexity of the t-SNE model, although
Figure 5.11: An overview of the evidence used to decide how many clusters to use. Panel (a) shows a truncated example of a hierarchical clustering dendrogram for all the data with PCA retaining 99% of the variance, where each node may contain multiple image slices. Panel (b) shows a gap statistic plots for a range of input data and variables. Panels (c–f) show the data plotted on the t-SNE components calculated for different sets of input data and variables: the raw rectangular form (c), and after applying PCA retaining 99% variance to the rectangular forms of the whole dataset (d) and the two reduced datasets (e–f). In the graph titles, rect: rectangular data, all: all data, oneA/oneB: reduced dataset A/B (Section 5.5.3), pca99: PCA applied with 99% variance, p: perplexity of t-SNE.
they become less obvious as the increase in perplexity causes the points to spread out.

On balance, I chose to use 5 clusters for the remainder of this chapter. This value is within the suitable range suggested by all of the methods shown here and the t-SNE plots in particular could not consistently be split into 4 or 6 clusters but there was an argument for 5 clusters for all four plots, so the case for 5 clusters was the clearest. Without the t-SNE plots I might suggest that 4 clusters may be more suitable, but adding an additional cluster gives the algorithm more flexibility to separate image slices that behave differently. This result suggests that there may be 5 distinct ways in which these oppositely charged particles are arranged on the droplet surface.

**Initial clustering results**

Working with this value of 5 clusters, I created stacked bar plots of the k-means and hierarchical clustering results, shown in Figure 5.12, to get an overview of how each image was clustered and to compare the performance of the two algorithms. The Gini scores for these cluster groupings of the data are 0.205 and 0.143 for k-means and hierarchical clustering respectively, indicating that the hierarchical clustering performs slightly better in this case.

![Figure 5.12: Stacked bar plots showing how slices from each individual image are grouped into clusters based on the raw rectangular data. The bar for each image is coloured based on the frequency with which slices from that image were grouped into each cluster. Two of the images are missing a slice, this is because the connected-region variables (discussed later) could not be calculated for these slices. In the graph titles, rect: rectangular data, all: all data.](image-url)
Looking in detail at the plots, the performance for most of the original images is quite similar between the two algorithms. For example, the red channels of all samples with $\phi_p = 0.5\%$ and $\phi_r = 10\%$ are predominantly in the same cluster (cluster 3 for k-means clustering and cluster 1 for hierarchical clustering), with little else in that cluster. Another example is that the green channels of three of the $\phi_p = 0.5\%, \phi_r = 10\%$ images are clustered together with the red channels of the $\phi_p = 1\%, \phi_r = 60\%$ and $\phi_p = 1\%, \phi_r = 75\%$ images. This agreement is good because it indicates that the clustering is based on real features in the data, not just random grouping based on the decisions of the algorithms.

There are of course some differences too, for example the green channel of $\phi_p = 0.5\%, \phi_r = 10\%$ sample number 4 is split into three different clusters with k-means clustering but only two with hierarchical clustering. In general it seems that the main difference between the clustering results is that, where an image is split between multiple clusters, hierarchical clustering tends to sort more of the images into a single cluster than k-means clustering, leading to the lower Gini score. This trend of hierarchical clustering outperforming k-means clustering continues throughout my later results, so in general I focus more on the hierarchical clustering results.

### 5.5.3 Finding the best clustering performance

Having chosen the basic parameters for my clustering problem, I investigated how using different representations and subsets of the data might change the clustering performance. In this context there are two relevant types of performance: cluster purity as defined by the Gini score, and computational performance which dictates how much time it takes to cluster the data.

#### Reducing the number of variables via averaging

As discussed in Section 5.4.3, my raw rectangular data was in the form of a $100 \times 146 \ (\theta \times r)$ pixel array for each slice of each image, which I treated as a 14600 component vector, but I also created a version where I averaged over the 100 values of $\theta$ to leave a $1 \times 146 \ (1 \times r)$ array. This means that the clustering on this data is based on 146 variables instead of 14600, and that the clustering is based solely on how the intensity of each image slice changes with the distance from the centre of the droplet.
The t-SNE visualisation for this representation of the $\theta$-averaged data shows that the structure of this data is rather different to the rectangular data. Figure 5.13 shows a much broader distribution of the data points, and it is less clear how many clusters exist in the data. There is some clumping of points but it is difficult to discern how many distinct clusters are present. This may suggest that 5 clusters might not be the best way of grouping this data, but it also doesn’t suggest that any other number of clusters would be particularly suitable. The gap statistic plot in Figure 5.11(b) also shows that this form of the data behaves quite differently to the others considered and has a less clear point at which the rate of increase of the gap statistic slows down. Considering the fact that this form of the data includes less information about the images, this result did not change my decision to run the clustering algorithms with 5 clusters.

With this data, k-means clustering gave a Gini score of 0.183 and hierarchical clustering gave a score of 0.146. Comparing these values to the scores for the rectangular data (0.205 for k-means clustering and 0.143 for hierarchical clustering), the averaging has significantly improved the k-means clustering performance, indicating that there was more variance within each cluster but that a greater amount of this within-cluster variance is now explained by differences between different slices from the same image. The Gini score for hierarchical clustering has been made slightly worse by the $\theta$-averaging so this positive result.
is restricted to the k-means algorithm and therefore less relevant because, as
was the case with the rectangular data, the hierarchical clustering algorithm
consistently outperforms k-means clustering.

On balance, these scores indicate that the information gained from considering
different angles at the same radius (i.e. using the raw data rather than the \(\theta\)-
averaged version) gives a little additional information that hierarchical clustering
can use to more easily recognise slices from the same images as being similar, but
this information leads k-means clustering to separate slices from the same image.
I will therefore consider how both forms of the data can be useful in the later
parts of this chapter.

A major benefit of reducing the number of variables in this way has been in
drastically decreasing the amount of time it takes to run the clustering algorithms.
In either case a single run of each algorithm takes little time compared to the
processing of the data, but when I run them multiple times then the difference
between the two data representations becomes clear. The multiple runs required
to make each curve of gap statistic vs. cluster number, such as the ones in Figure
5.11(b), took around 400s to run with the original rectangular data whereas the
same information was obtainable in only 15s using the theta-averaged data. This
highlights how useful it can be to reduce the number of variables I am working
with, so I continued to explore other ways of achieving the same effect.

**Reducing the number of variables via principal component analysis**

I investigated whether the clustering performance could be improved by using
principal component analysis (PCA) to de-noise the images and reduce the
number of variables fed into the machine learning algorithm. Figure 5.14(a)
shows how an example image slice can be represented by a reduced number of
components derived via PCA, and it is clear that the more detailed features of
the image disappear as the retained variance decreases. However, when I keep
95–99% of the variance there is not much difference between the reconstructed
image and the original, and the number of variables has been reduced by two
orders of magnitude from 14600 to 113–225. Considering the potential for noise
reduction, there are now many fewer variables in which noise can be introduced
to the image so it is likely that this will have a positive effect on the clustering
performance.
Figure 5.14: The effects of PCA on the data and clustering performance. Panel (a) shows how an example image slice is represented by different numbers of PCA components that explain different amounts of the variance within an image. Panels (b–c) show how the mean time taken for the clustering algorithms to be trained changes as the variance retained and therefore the number of components is changed. Each result is the mean of 10 repeats of 10 runs of the algorithm. Panels (d–e) show how the use of PCA to reduce the images to components explaining 97% or 99% of variance effects the clustering performance in terms of Gini score.
As in the case of $\theta$-averaging, this reduction in the number of variables leads to a reduction in the time taken for the clustering algorithms to run. Figure 5.14(b–c) shows that the introduction of PCA immediately reduces the time taken per runs of the algorithm by around 1 second. For hierarchical clustering the time then remains relatively stable at the lower value, although there is a lot of random fluctuation. For k-means clustering the time taken is reduced by a similar amount when PCA is first introduced, with an improvement of around 1 second for 97\% and 99\% variance compared to no PCA. In contrast to hierarchical clustering, the trend of increasing speed continues all the way to only the 4 variables that explain 50\% of the variance. In both of these graphs there is an anomalous peak at 92\% variance; this is likely due to another process occurring in the computer and slowing down the execution of the code. Since the point is within error of the points either side of it in both plots I find it reasonable to assume that PCA, in general, reduces the time taken for either algorithm to run. Although the time reduction is only a second or so this difference quickly adds when the clustering algorithm is run multiple times, for example when creating plots of Gini scores of different clustering approaches, so it is a worthwhile reduction to make.

Finally, Figure 5.14(d–e) shows that the introduction of PCA has a small and varied effect on the performance of the clustering as measured by the Gini score for image purity. For hierarchical clustering, PCA usually makes the performance worse but PCA at 99\% variance makes it slightly better. For k-means clustering with less than 6 clusters, the scores are generally within error of each other so are effectively unchanged. The errors on this plot show that the time taken for k-means clustering is also more variable than with hierarchical clustering but this is expected due to the nature of the algorithms. With k-means clustering the initial random seed dictates the position in Euclidean space at which the 5 clusters are seeded, which can lead to different results each time the algorithm is run.

For my chosen number of 5 clusters and using the same random seed as in the rest of this chapter, the introduction of PCA at 97\% and 99\% variance decreased the score to 0.183 and 0.179, respectively, with k-means clustering and changed it to 0.152 (an increase) and 0.141 (a decrease), respectively, with hierarchical clustering. These changes in performance are small, so I can conclude that PCA allows the reduction of computation time without negatively impacting the clustering result. This suggests that PCA has successfully reduced the information needed to represent the images while preserving the physically relevant features of each sub-image.
Reducing the number of data points

I next looked at the effect of removing data points from consideration, in particular by removing images such that there was only one image to represent each composition. I did this because, with some compositions having more data points than others, there was a risk of biasing the clustering towards picking up features that are common in those compositions even if they aren’t so relevant for other images. Firstly, I decided to remove two of the four images at $\phi_p = 0.5\%$, $\phi_r = 10\%$ (shown in Figure 5.15) so there were only one or two examples at each composition. I also removed the image at $\phi_p = 0.5\%$, $\phi_r = 50\%$ because it used a different type of negative particle to all the other images so is likely to behave differently.

![Image 2](image1.png) ![Image 3](image2.png)

(a) Image 2  (b) Image 3

![Image 1](image3.png) ![Image 4](image4.png)

(c) Image 1  (d) Image 4

**Figure 5.15:** The green channel of the four images at composition $\phi_p = 0.5\%$, $\phi_r = 10\%$. (a) and (b) show the images that were removed, (c) and (d) were retained and considered in the reduced-data part of the analysis.
I chose to remove images 2 and 3 (Figures 5.15 (a) and (b), respectively) for a number of reasons. Firstly, as can be seen immediately from the images, the droplet is cut off at the edge of the image which makes it harder to find the centre of the droplet accurately. Image 3 also looks particularly unusual, as the droplet seems to buckled and wrinkled. This makes it difficult to find features in the actual distribution of the particles on the surface, and is a problem because in all the other images in the data set, without wrinkles, the distribution of particles is the only source of features in the image. I use information from previous clustering results to confirm my decision to remove image 2: Figure 5.12 shows that, for both types of clustering, image 2 is the only one at this composition where the green channel does not contain slices in cluster 0, suggesting that it is not representative of that composition in general and therefore a more sensible image to remove than one that is more representative.

I then created two sets of data (A and B), and for the compositions where there were two images I arbitrarily put one in each data set. Where there was only one image, it was included in both sets. I then performed the same analysis as above and found that, compared to the results from all the data, set B consistently made the Gini score worse. Set A generally made the performance worse with hierarchical clustering but better with k-means clustering, with an exception for the $\theta$-averaged data where the reverse is true.

![Data set A and Data set B](image)

Figure 5.16: Stacked bar plots showing how slices from each individual image are grouped into clusters based on the reduced data with one image per experimental composition. The bar for each image is coloured based on the frequency with which slices from that images were clustered into each cluster. One of the images in set B is missing a slice, this is because the connected-region variables (used later) could not be calculated for this slice. In the graph titles, rect: rectangular data, oneA/oneB: reduced dataset A/B (Section 5.5.3).
Looking at this result in more detail via the bar plots shown in Figure 5.16 there are a number of cases where the result was the same for both sets of data. However, the act of changing some of the images in the data has also changed the clustering purity of images that are present in both sets.

**Using different variables**

The final way I tried to improve the clustering performance was by considering additional variables based on connected regions in the images as described in Section 5.4.4. These variables may provide more information to the algorithms because they aren’t based on the intensity of the pixels and are therefore uncorrelated to the other variables I was already using. They are also easy to interpret physically, so if they prove to be useful for clustering then it would be easy to understand what features characterise the different clusters.

![Stacked bar plots showing how slices from each individual image are grouped into clusters based on the rectangular data both alone (a) and combined with the connected-region variables (b). The bar for each image is coloured based on the frequency with which slices from that images were clustered into each cluster. A few of the images are missing a slice, this is because the connected-region variables could not be calculated for this slice. In the graph titles, rect: rectangular data, all: all data, combi: combination of normal and connected-region variables.](image)

The bar plots for hierarchical clustering of the raw rectangular data both with and without the additional six connected region variables is shown in Figure 5.17. This hierarchical clustering has a Gini score of 0.141, very close to that of the clustering without the additional variables. Comparing this to Figure 5.12(b) it is clear that the clustering itself is also very similar, with differences in only three
Figure 5.18: Plots of the connected-region data in terms of two t-SNE components, calculated for two different perplexity values (top=5, bottom=30) and two different random number seeds. None of the plots show any indication of sensible clusters for these data. In the graph titles, rect: rectangular data, all: all data, wedge: connected-region data, p: perplexity of t-SNE.

Images: the red channels of the $\phi_p = 0.5\%$, $\phi_r = 25\%$ and $\phi_p = 1\%$, $\phi_r = 40\%$ samples and the green channel of the $\phi_p = 1\%$, $\phi_r = 75\%$. Other images, like the red channel of the $\phi_p = 0.5\%$, $\phi_r = 40\%$ sample, look different initially but actually the clusters are just plotted differently: looking at the clusters in terms of which other images they contain makes it clear that the two results are the same. When I use either data set with one image per composition, the clustering is identical with and without the additional variables. This result indicates that almost all of the information encoded into the additional variables has already been picked up by using the image itself as a set of variables, so there is little point in including them in the algorithm.

In order to delve into why the additional variables made no difference to the final clustering result, I assessed their performance alone to determine if they contain the same information as the earlier variables or if they are simply not useful for this clustering. The Gini scores of 0.319 for k-means clustering and 0.250
for hierarchical clustering show immediately that these variables do not capture enough of the features that characterise each image for the algorithms to cluster them together, suggesting that the clusters have very little relation to the physical features of each droplet. These scores are clearly the worst in any of the analysis so far. A glance at the t-SNE representation of this data (Figure 5.18) suggests why the clustering has performed so poorly: there are no clear clusters when the images are represented only by these six variables, and the regions where data points are clustered together are a strange elongated shape. This supports the conclusion that this representation of the data is unsuitable for clustering, and that the data is better represented by a less processed version of each image.

**Summary of clustering performance**

<table>
<thead>
<tr>
<th>Data</th>
<th>Variables</th>
<th>K-means</th>
<th>Hierarchical</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Raw rectangular data</td>
<td>0.205</td>
<td>0.142</td>
</tr>
<tr>
<td>All</td>
<td>$\theta$-averaged</td>
<td>0.183</td>
<td>0.146</td>
</tr>
<tr>
<td>All</td>
<td>PCA 97% variance</td>
<td>0.183</td>
<td>0.152</td>
</tr>
<tr>
<td>All</td>
<td>PCA 99% variance</td>
<td>0.179</td>
<td><strong>0.141</strong></td>
</tr>
<tr>
<td>All</td>
<td>Added connected-region $^1$</td>
<td>0.163</td>
<td>0.141</td>
</tr>
<tr>
<td>All</td>
<td>Connected-region only</td>
<td>0.319</td>
<td>0.250</td>
</tr>
<tr>
<td>Reduced (A)</td>
<td>Raw rectangular data</td>
<td>0.189</td>
<td>0.184</td>
</tr>
<tr>
<td>Reduced (A)</td>
<td>$\theta$-averaged</td>
<td>0.197</td>
<td><strong>0.177</strong></td>
</tr>
<tr>
<td>Reduced (A)</td>
<td>PCA 99% variance</td>
<td>0.174</td>
<td>0.184</td>
</tr>
<tr>
<td>Reduced (A)</td>
<td>Added connected-region $^1$</td>
<td><strong>0.150</strong></td>
<td>0.184</td>
</tr>
<tr>
<td>Reduced (B)</td>
<td>Raw rectangular data</td>
<td>0.274</td>
<td><strong>0.160</strong></td>
</tr>
<tr>
<td>Reduced (B)</td>
<td>$\theta$-averaged</td>
<td>0.223</td>
<td>0.165</td>
</tr>
<tr>
<td>Reduced (B)</td>
<td>PCA 99% variance</td>
<td>0.245</td>
<td>0.168</td>
</tr>
<tr>
<td>Reduced (B)</td>
<td>Added connected-region $^1$</td>
<td>0.210</td>
<td>0.160</td>
</tr>
</tbody>
</table>

Table 5.1: Summary of the Gini scores for different unsupervised clustering models (k-means and hierarchical clustering) on different data and variables. A lower Gini score indicates that the model is more successful at keeping together data points that originate from the same image.

Table 5.1 shows the Gini scores for different combinations of data and variables, outlining the information already presented in this section as well as scores for combinations that were tried but not discussed in detail. The best scores are shown in bold text. Comparing these Gini scores it is first clear that hierarchical clustering consistently outperformed k-means clustering. The reduction of the

$^1$The additional connected-region variables were used alongside the raw rectangular data.
data sets by removing images to leave only one per composition had mixed results with a very broad range of Gini scores. In the rest of this chapter I will therefore focus on the results based on all of the images. Adding in variables based on connected regions identified in a thresholded version of each image usually had very little effect on the hierarchical clustering performance. It did generally make the k-means results better, but these results are still beaten by the hierarchical clustering result so this improvement is less relevant. This result indicates that these additional variables did not add any useful information that the clustering algorithms could not already determine from the images themselves, so effective clustering can be performed without the need for extensive data processing to extract variables. On the other hand, these variables are easier to link to a physical interpretation so the clustering result without them is more difficult to link to physical characteristics of the system.

Reducing the number of variables used to represent the data sometimes increased and sometimes decreased the clustering performance in terms of the Gini scores presented here, but it always improved the performance in terms of computation time because there were many fewer variables. The $\theta$-averaged data performed a little less well than PCA in terms of preserving the quality of clustering, but it has the benefit of being easy to relate back to the original image so I will utilise both sets of data when interpreting how the clustering algorithms made their decisions. This gives me more chance of understanding the features that lead to the clustering decision and their physical meaning.

5.5.4 Interpretation of clustering

Although I know how the image slices have been clustered in terms of the final result, it is less clear what features in the images are used in the clustering decision. Because there are many variables, it is impossible to visualise where each image slice lies in variable space, which is ultimately where the similarity of slices is determined based on the Euclidean distance between them. Instead I plotted an image of the information used to represent each slice in a representative selection of images. From these plots, which show the hierarchical clustering result for each slice, I can get an idea of some features that may have led to the decision to cluster these slices together.

In order to plot the most easily understandable version of the data provided to the clustering algorithms, I chose to use the $\theta$-averaged version of the data even
though it did not provide the best clustering performance. The raw data contains too many variables to show, and the PCA variables are difficult to interpret because each principal component is in effect an image itself. By averaging across $\theta$ in each slice I am effectively forcing the algorithm to look at features on a larger scale than individual particles on the droplet surface. Aside from at very low values of $r$, the particles are much smaller than the width of the slice and therefore the averaging merges together the brightness of any particles in that region. This means that the overall brightness of a pixel in Figure 5.19 is indicative of the general particle coverage in that slice at that radius, and areas with more particles, for example aggregates, within a slice will lead to brighter patches in the averaged data. Conversely, areas with less particle coverage will lead to darker patches in the averaged data.

Cluster 1, shown in Figure 5.19(a–d), is generally bright with some small dark patches at all values of $r$. The overall brightness of the cluster indicates that there are a lot of particles on the droplet surface, but this cluster does not contain only the majority channel in terms of the positive and negative particles: examples (a) and (c) depict the majority channel but (b) and (d) do not, although these are the only examples out of 7 images. The differences between the two image channels can be seen in the images, but the clustering algorithm has decided that they still belong together. This decision could be linked to the fact that all but one of these images have a high value of $\phi_r$, i.e. $\phi_r > 50\%$. This cluster also contains only slices with $\phi_p = 1\%$, suggesting that some of the basis for clustering is associated with this variable, but it is hard to discern any relevant features.

Cluster 3, shown in Figure 5.19(e–h), looks rather different to cluster 1. The slices are dark overall with only small patches of bright particle signal, which tend not to occur at high values of $r$. Similarly to the case with cluster 1, there are two examples where the image is not from the minority channel that might be expected from this result, and one of them is shown in panel (e). Unlike cluster 1, this image does not look particularly different from the other examples in this cluster. There are only two examples of images with a high $\phi_r$ in this cluster, one of which is shown in panel (f).

Cluster 0, shown in Figure 5.19(i–l) is in many ways an intermediate stage between clusters 1 and 3. It contains more particle signal than cluster 3 but less than cluster 1, and includes both large and small patches of signal. Some images (panel (i)) look similar to cluster 1 but with larger dark areas with no signal, and others (panel (j)) look similar to cluster 3 but with more patches of intermediate
Figure 5.19: Examples of images that have been grouped into clusters 1 (panels (a–d)), 3 (panels (e–h)) and 0 (panels i–l)). Each horizontal line represents one slice of the image, and the $\theta$-averaged value is shown for each value of $r$. Some slices were not sorted into the same cluster as the rest of the image, these have been removed. The colour scale is the same for every image.
brightness and smaller dark intervals between brighter signals. When looking at these images it is less clear whether they might represent the majority or minority channel of the image, but in fact they are generally from the majority channel. The images in this cluster are also predominantly images with a low value of $\phi_r$ (<50%), similar to cluster 3.

The other two clusters, not shown here, contain fewer image slices and provide less interesting results. Cluster 2 contains only two images, both channels of the $\phi_p = 0.5\%$, $\phi_r = 75\%$ sample, and it is clear that these images are anomalous compared to the rest of the dataset because the central region of the image is dark. Unfortunately there was no other data available at this composition, so the images could not be replaced with a better pair. Cluster 4 contains images with very little particle signal: the minority (green) channels of $\phi_p = 0.5\%$, $\phi_r = 10\%$ images and of one $\phi_p = 1\%$, $\phi_r = 25\%$ image.

All of these clusters have a tendency to include images with either higher or lower values of $\phi_r$, rather than ones equally distant from $\phi_r = 50\%$. This suggests that there is some distinction in the clustering between droplets with higher and lower values of $\phi_r$, which matches my observations from using traditional analysis on the lower-magnification images of these samples (Section 5.5.1), rather than grouping droplets with $\phi_r$ values which matches the observations in the literature [145]. This observation is somewhat supported by Figure 5.20 which shows compositional plots for three different versions of the data, including the $\theta$-averaged version investigated here.

These compositional plots do not show the majority cluster for each image, but rather the number of slices at each composition sorted into each cluster. They therefore reveal different information about the clustering process to the images discussed above, as well as allowing me to look at the results from the raw and PCA-reduced data alongside the $\theta$-averaged version.

Although the clusters may be labelled differently for the different representations of the data, I can deduce which ones are equivalent based on their members. For example the cluster that contains the slices from the $\phi_p = 0.5\%$, $\phi_r = 75\%$ is number 4 in the raw and PCA data and number 2 in the $\theta$-averaged data. These equivalent clusters are outlined in Figure 5.20(d) for ease of reference. I will generally refer to the clusters by their label in the raw and PCA results.

In all cases, cluster 3 predominantly contains slices with $\phi_p = 1\%$. In the $\theta$-averaged case, this cluster becomes more present as $\phi_r$ increases but in the other
Figure 5.20: Compositional plots of hierarchical clustering results based on the values of experimental parameters $\phi_p$ (x-axis) and $\phi_r$ (y-axis). The frequency of each cluster at each composition is represented by the size of the circle. Panel (a) shows the result from the raw rectangular data. Panel (b) shows the result from the $\theta$-averaged data, and it is unclear but the $\phi_p = 0.5\%$, $\phi_r = 60\%$ sample is split equally between clusters 3 and 0. Panel (c) shows the result from the data with PCA applied to retain 99% of the variance in the images, and has a very similar result to panel (a). Panel (d) is a key for comparing the clusters in different panels. In the graph titles, rect: rectangular data, th-av: $\theta$-averaged data, all: all data, oneA/oneB: reduced dataset A/B, pca99: PCA applied with 99% variance.
cases the inverse is true and it contains more examples with lower \( \phi_r \). This somewhat supports my observation that each clustering result lacks a sense of symmetry about \( \phi_r = 50\% \) but this conclusion is far from clear and it could be argued that the presence of all compositions in the same cluster is in itself evidence of some symmetry.

Looking at these plots, cluster 4 (cluster 2 in the \( \theta \)-averaged case) provides a slightly more interesting result than the plots in Figure 5.19 because it is clear that the cluster is not entirely confined to the single example of \( \phi_p = 0.5\% \), \( \phi_r = 75\% \). Particularly in the raw and PCA results, this cluster is also well-represented in the \( \phi_p = 1\% \), \( \phi_r = 75\% \) images, suggesting that the result is not entirely due to the unusual dark centre of the particular image in the \( \phi_p = 0.5\% \), \( \phi_r = 75\% \) case. There is no evidence of these features being shared with images with a lower value of \( \phi_r \).

**5.6 Conclusions**

When I looked at the overall shape of multiple droplets at a low magnification, I found that the roundness of the droplets tended to increase as the proportion of negative particles in the system increased. Despite the relative ease with which I could draw a conclusion from the results of this analysis, it required a large amount of human input, as none of the usual methods for identifying the outline of individual droplets worked on the data. I therefore turned to machine learning for exploratory data analysis to see if the higher-magnification data could give any other interesting information that was not based on the shape of the droplet.

I first found that the data seemed to be most suited for being grouped into 5 clusters (Figure 5.11), and this allowed me to investigate ways of improving the performance of the model by using this constant number of clusters so I could directly compare different results. During this investigation I found that principal component analysis (PCA) was a very effective method for reducing the number of variables used to represent each image and therefore reducing the time taken to train the clustering algorithms, as shown in Figure 5.14. Applying PCA made a small difference to the clustering result and could make the performance score either slightly better or slightly worse depending on the number of clusters, random changes in the algorithm initial conditions, and exactly which images were included in the analysis. Reducing the number of variables by averaging
over $\theta$ also had quite a small effect on the Gini score, and generally improved the performance of k-means clustering but made hierarchical clustering marginally worse.

Overall, I concluded that both the PCA-reduced and $\theta$-averaged forms of the data would be useful to consider in more detail. The additional variables I considered, derived from traditional image analysis techniques, proved not to be useful. I also rejected the use of reduced data sets including only one image example at each composition because the results of the clustering were too dependent on which images were chosen. The most constant conclusion in the optimisation of the clustering performance was that, for this data, hierarchical clustering clearly outperformed k-means clustering in all circumstances.

Looking at the clustering results in terms of the experimental composition of the images, in Figures 5.19, it was difficult to discern exactly how each clustering decision was made but it seems that the sizes of patches of bright particle signal might be important in the process, and that this is partly but not entirely aligned with whether an image depicts the majority or minority particle phase in the system. Considering the clusters into which each of these images was predominantly sorted along with the slice-level sorting shown in Figure 5.20, I found some evidence that the high-$\phi_r$ examples are more similar to each other than to the low-$\phi_r$ examples, and vice versa. This aligns with my conclusions based on traditional analysis of the lower-magnification data that the roundness of droplets increases with increasing $\phi_r$. This analysis therefore suggests that there could be a link between droplet roundness and the particle surface structure.

The results in this chapter have shown both the benefits and downsides of using unsupervised learning for exploratory data analysis. Hierarchical clustering has proved to be a powerful tool and can easily be applied to image data in many forms, and in fact was much more effective on the data in its nearly raw form than when applied to variables obtained via more traditional image pre-processing. This is in contrast to the traditional analysis which required a large amount of human input. Unsupervised learning can give some information about which images are likely to be similar and, by optimising the performance based on a measure that rewards parts of the same image being clustered together, I have encouraged a meaning of ‘similar’ that can be interpreted as ‘similar in the same way that different parts of the same image are similar’. The downside of this clustering is that is difficult to interpret the results beyond the level of seeing which images are similar to each other, and to deduce what features are being
identified by the clustering algorithm as relevant. It was easier to make such
deductions by looking at an averaged form of the data, so it was good that the
different forms of the data gave relatively similar clustering results.

These results have also shown that even when attempting to perform machine
learning in a physics-led way, it is not always possible to obtain information about
the physics of the system from the result. In this case, the results are unlikely
to be particularly useful because I was unable to find any meaningful physical
characteristics in the clustering decision. This may be in part due to the fact
that I embarked on the project with a very broad question and a broad machine
learning approach to match: unsupervised clustering is a much less targeted
method than, for example, supervised classification but it was the most suitable
approach due to the broad nature of the problem at hand.

In future, it would be interesting to apply the same unsupervised clustering
methods to more examples of these droplets stabilised by oppositely charged
particles, including a larger number of the less round ones that I observed in the
low-magnification data. This would allow me to see more clearly if sub-images
from droplets with different levels of roundness were clustered together, rather
than looking only at the underlying composition that seems to dictate roundness,
and perhaps to draw some conclusions about the causes of the distortion of the
droplets. This would be particularly interesting because in a typical emulsion
system droplets are expected to be spherical. It is unclear what mechanisms may
be causing these droplets to deviate from this shape, but it must be to do with
the particles on the surface because without them the droplet must be spherical
in order to minimise the free energy of the oil/water interface.

This proposed investigation would be more focused and more physics-led than the
one described in this chapter, because I would be looking to answer a more specific
question about the physical system, and it therefore might be more successful in
providing physically interesting results.
Chapter 6

Investigating the effects of local forces within bijels under centrifugation

In this chapter I return to a bijel system and investigate how machine learning can be used for image segmentation and to answer a specific physical question. I use a segmentation tool based on the decision tree algorithm and compare its result to that of a traditional segmentation method. I then use these segmented images to find the extent of the impact of a local shear force on the structure of a bijel by training a regression model to predict the distance of a section of bijel from the track left by a bubble. Finally I interpret the result to come to a preliminary conclusion on its implication for the local yield stress of bijels.

6.1 Introduction

6.1.1 Forces in bijels

One reason for the wide range of potential applications of a bijel is that its structure is very robust. There are a number of different approaches to investigating the effects of shear and bulk forces on bijels, in order to understand how they might behave during and after processing into materials for different applications. Bulk rheology studies have found that bijels, as should be expected,
undergo a gelation transition as they are quenched and that the response of a bijel to bulk forces depends on a number of factors including the depth of the temperature quench, the particle content (and therefore domain width), and the choice of liquids in the fluid domains [108]. In non-polar bijels, the interfacial particles form patches of high concentration which merge during bijel coarsening and are linked to a non-monotonic rheological response during the formation [142]. It was also found that there is an increase in the elastic modulus of the bijel due to the rearrangement of interfacial particles over long timescales.

The study from which my data are extracted showed that, when a force is applied to a bijel via centrifugation, it is irreversibly compressed as liquid is expelled from the structure [106]. The bijel loses its isotropy and forms sheets perpendicular to the direction of the force. The presence of small number of bubbles inside the bijel, likely formed by the escape of nitrogen dissolved in the nitromethane, aids the compression of the bijel because the movement of the bubbles forms channels parallel to the direction of compression through which liquid can escape from the structure. The yield stress of the bulk of the bijel was found to be on the order of kPa, but the passage of a bubble applies additional local forces to the structure.

The effect of such local compressional stress on bijels has been investigated by pressing a needle into the structure and imaging the structure before, during and after the passage of the needle [103]. The elastic and self-healing nature of the bijel was inferred from the fact that, despite being compressed by five domain widths, the bijel returned to nearly its original structure upon removal of the needle. The only lasting effect of the needle’s compression was an elongation of the domains that were directly adjacent to the needle. The effect of shear stress was also investigated by shearing a bijel between two glass slides, and the domains of the bijel showed a clear tendency to align towards the direction of the shear, deviating from the usual isotropic bijel structure.

### 6.1.2 Microrheology using bubbles

A novel approach to the investigation of the local effects of forces within a viscoelastic material such as a gel is to apply forces via a bubble inside the material. This can be done using ultrasound, either at a very high frequency to induce a constant acoustic radiation force on the surrounding material [151] or close to the resonant frequency of a bubble, in which case the oscillations of the bubble impose large oscillatory strain on the surrounding medium [152].
the former case, the shear modulus of the surrounding material can be found directly from the total displacement of the bubble under the acoustic radiation force, assuming that the shape of the bubble does not change. In the latter, the resonance frequency of a bubble in a hydrogel is found to increase as the shear modulus of the gel is increased and the high frequency oscillatory rheology of the material is found to be very different to results measured at lower frequencies by conventional rheometers.

Taking inspiration from such approaches, the application of forces via the simple expansion of a bubble has been shown to be effective for determining the local yield stress of a microfibrous cellulose gel \[153\]. In this case, the bubble is used to apply a non-oscillating force to the surrounding medium by changing the pressure of the system, which changes the size and therefore the buoyancy of the bubble, causing it to apply a controlled amount of force to the surrounding gel network. This method allows for sensitive measurements of small local yield stresses in the material, which are found to be very different from the bulk yield stress of the same gel. Tiny tracer particles are included in the gel matrix in order to track the distortion of the gel matrix due to the force applied by the bubble as it moves upwards through the gel because of this buoyancy force. It is found that the path of the bubble depends on whether or not the gel is able to rearrange locally in response to the force it applies: rearrangement causes the gel to become denser above the bubble and forces it to take a non-linear path. A quantitative comparison of the displacement of tracer particles at increasing distances from the bubble shows the existence of a limited region of the gel that is distorted by the bubble before the material yields entirely and allows the bubble to move. This final result is interesting in the context of this chapter, because such an approach could be taken in order to investigate how deep into a bijel the effect of a shear force is felt.

### 6.1.3 Aims of this chapter

In this chapter I aim to investigate how the passage of a bubble through a bijel can affect its local structure, and to what extent the proximity of a section of bijel to a bubble passage can be predicted using machine learning. I first compare different methods of segmenting images to separate the areas of bijel from the holes left by the passage of a bubble, to see if a machine learning approach or a traditional approach is more suited to my requirements. After this process I will
have a set of images where I know where the bubble trace is in each one, and therefore how far each part of an image is from the bubble trace, so I can use this information in the rest of my analysis.

I then investigate how the orientation of the bijel changes with increasing distance from the bubble track, and whether a measure of orientation based on the 2D autocorrelation function of the raw and segmented images is an indicator of this distance. If the orientation is linked to the distance of a bijel section from the nearest bubble trace, this information can be used to understand how the orientation of the bijel is affected by the passage of the bubble. Because previous studies have shown that the presence of shear forces can induce an orientation on an otherwise isotropic bijel, this approach directly uses known behaviours of the physical system to investigate the forces on the bijel.

I also use the autocorrelation function as inputs to a support vector regression algorithm, with the aim of predicting how far from the bubble track a given piece of bijel is. The aim here is to train and test the algorithm on the autocorrelation function of different sections of images to predict how far each section is from the bubble track in the image, in order to understand how the unprocessed autocorrelation function of the bijel is affected by the passage of the bubble. In addition, by assessing the success of this prediction at different distances I can investigate how far from the bubble trace its effects can be seen in the bijel structure. Although this approach does not directly calculate the orientation that is expected to be an indicator of the effect of a shear force, information on the orientation of each image is present in the autocorrelation function and can therefore be used by the machine learning algorithm.

Finally, I use my results to gain insight into the lengthscale of the impact of the bubble’s passage on the local bijel structure and therefore the local yield stress of the bijel, and compare this to the literature for a non-bijel material. This will demonstrate how the application of machine learning to images can be used to probe the physical properties of a system.

6.2 Data

In this chapter I use data provided by Katherine Macmillan from the previously mentioned study of the mechanical properties of bijels under compression
The data available were confocal images of ethanediol/nitromethane bijels stabilised with silica particles. The ethanediol was doped with fluorescein and the particles were treated with rhodamine-B, so they could be imaged individually. For this chapter, I used only the fluorescein-doped ethanediol channel to represent the bijel’s structure.

These data are from the same experiment as the data used in Chapters 3 and 4 but in this case I used images taken after the bijels were subject to centrifugation, rather than images of the newly fabricated samples. From this post-centrifugation dataset I selected all images in which one or more bubble tracks were clearly visible, provided that they were taken at the same magnification and were from samples with a similar composition to one another. Example images are shown in Figure 6.1 in most images the bubble has travelled all the way through the frame, as demonstrated in Figure 6.1(a), but in some cases the bubble track ends within the image as in Figure 6.1(b). In the latter case, the bubble is probably not passing through this part of the bijel and therefore it may not apply a shear force to the structure. In total I identified nine images to work with, although some were rejected for parts of my analysis.
Figure 6.1: Two examples of the images used in this chapter. The passage of a bubble has left a track through the bijel, visible here as the dark region across the centre of the image. In some cases, such as in panel (b), the ethanediol leaked from the bijel into the hole leaving a bright region at the edge of the bubble track.
6.3 Methods

6.3.1 Image segmentation

In this chapter I needed to identify two regions in each of my images: the bijel and the hole in the bijel. In order to do this quickly and accurately I compared two different image segmentation methods, one that used machine learning and one that didn’t, to see which was the most suited to my task and data.

Trainable Weka segmentation

Weka (Waikato Environment for Knowledge Analysis) is a Java-based software that contains a range of tools for machine learning and data processing, designed for the purposes of data exploration and for quickly trying different methods on a data set [154]. This software has been used in an ImageJ [146, 147] plugin, Trainable Weka Segmentation [155], to create a machine learning tool that takes a small number of manually drawn examples of different areas of an image and uses them to train a classifier that can be used to automatically segment the rest of that image and any subsequent similar images. This tool provides a graphical interface in which the user can select regions of interest for each class in an image, and creates many different versions of these regions by processing the original image in a number of different ways.

In this chapter I use the default processing options as follows. Gaussian blur (Figure 6.2(b)) applies a Gaussian filter to the image by convolving the image with a Gaussian kernel to create a blurred version. A Sobel filter (Figure 6.2(c)) can be used following a Gaussian blur, and calculates an approximation of the gradient of the image intensity at each pixel by applying a $3 \times 3$ matrix [156]. The Hessian matrix (Figure 6.2(d)) is calculated for each pixel following a Gaussian blur and describes the local curvature of the image, and features including its determinant (Figure 6.2(e)) and eigenvalues (for example Figure 6.2(f)) are also calculated and used. The difference of Gaussians (Figure 6.2(g)) calculates two Gaussian blur images with different levels of blur and subtracts one from the other. For each of the processing options just mentioned the extent of the effect of the Gaussian filter is set by the value $\sigma$, and I use values of $\sigma = 1, 2, 4, 8, 16$ to provide a range of processed images. I also use membrane projections which
Figure 6.2: Examples of some of the different image features calculated by the Trainable Weka Segmentation plugin for ImageJ. Selected pixels in the original image are represented by their counterparts in these and similar representations to form a set of features to use for training the classifier.
use directional filtering to enhance membrane-like structures of the image, for which I use a thickness of 1 and a patch size of 19 pixels. This method creates a stack of 30 images which are combined into a single image by taking the sum, mean, standard deviation, median, maximum, or minimum of the pixels in each image in the stack. Each of these 6 images is used as a feature, for example Figure 6.2(h) demonstrates the feature obtained by taking the sum of the pixels in each image. Finally, I use the hue (Figure 6.2(i)), brightness, and saturation of the image. In total, with features calculated for multiple values of $\sigma$ and some methods creating multiple features, I have 79 versions of the image including the original.

When I select regions of interest in the original image in ImageJ, each pixel in that region is characterised by the values of that pixel in each of the 79 feature images, and is labelled with the class I select for that region. If desired, the dataset can be balanced such that each class contains the same number of pixels; this is achieved by repeating some pixels from the under-represented class and omitting some pixels from the over-represented class. The plugin then uses the Weka toolkit to apply my chosen machine learning classification algorithm to this labelled dataset. I used the default algorithm which is a random forest algorithm as described in Section 2.2.4, and I use the result of 200 different trees trained with two random features at each decision node. The result is a trained classifier that can be used to classify the whole image, segmenting it into the desired regions. The classifier can also be saved and then applied to other images. For my set of images, I first created a classifier trained on one example image. I then applied the classifier to all images in my data set (both the straightforward examples shown in Figure 6.1 and other, more challenging, images) to create a segmented version of each image.

### Seeded region growing

Image segmentation can also be achieved with methods that do not use machine learning algorithms. One such method is called seeded region growing [157, 158], in which a number of initial seeds sorted into the different classes in the image are expanded until the whole image is sorted into exactly one of the classes. It is not a machine learning method because it does not create a model or rule for classifying an image that can be applied to new images, rather it is applied to each image with no prior knowledge of the data. The algorithm works as follows.
For each pixel $x$ adjacent to a seed region containing pixels $y$, the quantity

$$\delta(x) = |g(x) - \text{mean}[g(y)]|$$

is calculated as a measure of how different the pixel is from the region it adjoins, where $g(x)$ is the grayscale value of pixel $x$ and the mean is calculated across all pixels $y$ in the relevant seed class. If a pixel $x$ is adjacent to two different class, then it is assigned to the region for which $\delta(x)$ is smallest.

This process is repeated until all pixels have been assigned to one of the regions. If a class is split into two, for example in my data if the bubble track splits the bijel into two unconnected regions, then each instance of the class is given its own seed and treated as its own region in this algorithm. In the end result multiple regions can be labelled as the same class even if they are not touching each other. Because my regions of bijel include a range of greyscale intensity values, I make sure to use seed regions large enough that this variance in intensity is captured so that the algorithm can more correctly identify the bijel regions.

As with Weka segmentation, I used seeded region growing via an ImageJ plugin. For each image, I selected rectangular sections of each continuous region of bijel and hole. For images such as the one shown in Figure 6.1(a), three seed regions were needed: one for the hole and one each for the two regions of bijel each side of the hole. In cases such as in Figure 6.1(b) where dye leaked from the bijel into the hole, two additional non-bijel classes of region needed to be defined: the bright area where the dye is present in high concentrations and the grey edges of the bright area where the concentration of the dye was diminished. In this particular example, each of the four classes required only one seed as they were all continuous regions. In some cases my initial choice of seeds for an image did not yield a sensible result, so I changed the seed and repeated the segmentation. This is, for example, how I discovered the need to define a separate class for the grey area between the bright and dark areas of the bubble track. Unlike the Weka segmentation, no information on the classes or seeds was shared between images: each one was segmented individually.
6.3.2 Preparing the data

In order to investigate the effects of the passage of the bubbles through the bijels, I needed to split the bijel region into smaller sub-images, such as the one shown in Figure 6.3 (b) to use as data points. This allowed me to gain a reasonable number of data points from my small number of images, as well as to create different data points for different regions of the bijel.

![Figure 6.3: Examples of the different stages of processing the original bijel image. Panel (a) shows an example image with the hole removed. Panel (b) shows the first 64-pixel square sub-image of that image, from the top left of the original, which is 161 pixels from the hole. Panel (c) shows the autocorrelation function of that sub-image. Panels (d–f) show more examples of the autocorrelation function of different sub-images from the same original image.](image)

For a given size of sub-image, which I varied to investigate the how robust my results were, I split the original image into a grid of squares that size. I then rejected any sub-images that contained any pixels of hole as defined by the segmented image: I achieved this by setting the value of all pixels in the hole to an arbitrary unphysical number (I chose $-100$), and rejecting any sub-image that contained that value. I then treated this set of sub-images as my data points.
As well as recording the sub-images themselves, I calculated and recorded the Euclidean distance from the centre of each sub-image to the nearest pixel in the hole region of the image. This was the measure I used to define distance from the bubble track, on which I base my analysis in this chapter.

Similar to my work in Chapters 3 and 4, I based the majority of my analysis on the autocorrelation function (e.g. Figure 6.3(c)) of each sub-image. I calculated this function from the Fourier transform of the sub-image (e.g. Figure 6.3(b)) in the same way as in these earlier chapters (Section 3.3.1), but in this case I did not take a radial average to map the two-dimensional function into one dimension. This is because, after centrifugation, I do not necessarily expect the bijel structure to be isotropic and therefore a radial average is unsuitable. In fact, it would erase the very information I wish to analyse, so I instead used the autocorrelation function in its two-dimensional form.

6.3.3 Investigating changes with distance from bubble track

The literature [103] suggests that the orientation of the bijel structure is changed by the application of a shear force, such as the force applied by the bubble passing through the bijel, so I investigated how the orientation of the bijel structure changed with increased distance from the bubble track.

In order to estimate the overall orientation of an image, I took inspiration from the orientation measurement plugin in ImageJ but translated it into python for use alongside my other analysis. This method of calculating the orientation of an image [159] makes use of a form of principal component analysis often referred to as the ‘hotelling transform’ [158]. For each intensity value in an image, the covariance matrix of the \((x, y)\) coordinates of the pixels of that intensity describes how the variance in that intensity value is distributed. The eigenvectors of this matrix are the components of the image at the given intensity, and the ones with the largest eigenvalues explain the most variance and are the principal components. This approach can be applied to contours of an image, and the first principal component can be interpreted as the direction of the longest axis of the contour and therefore can be used to calculate its angle of orientation [160].

The orientation of a bijel, even when the structure is no longer isotropic due to the application of a force, is difficult to discern in real space. From the autocorrelation function of the image, however, directions of orientation in the original image can
be found from the shape of the contours \[161\]. I therefore use contours of the autocorrelation function of each image, such as the examples in Figure 6.4 as the basis from which to calculate the orientation of the structure. For each of my bijel sub-images, I calculated the orientation of each of five contours of the autocorrelation function (as in Figure 6.4(a)), and take the orientation angle of the image to be the mean of the orientation of the second to fourth brightest contours: excluding the bright (yellow) central peak because it was always a single pixel, and the darkest (blue) contour because it represents the background value of the function rather than the main part of the signal.

**Figure 6.4:** Examples of the process and end result of calculating the orientation of a bijel sub-image or a segmented image. Panels (a), (b), and (c) show the contours, autocorrelation function, and original image, respectively, of a bijel sub-image. Panels (d) and (e) show the original image and autocorrelation function, respectively, of a segmented image.

In order to account for the different orientations of some of the bubble tracks, I subtracted the bubble track orientation from the orientation of each sub-image. I calculated the bubble track orientations from the segmented images using the same method I applied to the bijel sub-images after first, where relevant, setting all non-bijel regions to the same value. Unsurprisingly, and as seen in Figure 6.4(d–e), the autocorrelation functions of the segmented images were very
different in shape to those of the bijel sub-images and I found that the most consistently representative measure of the orientation was to use the orientation of the brightest contour in the autocorrelation function. I therefore used this value rather than an average of the contours.

After calculating the orientation of each sub-image relative to orientation of the bubble track in its parent image, I could create a number of plots to investigate how this orientation angle changes with increasing distance from the bubble track. I also looked at other variables related to the profiles of the autocorrelation in the $x$ and $y$ direction to see if they had any relation to the distance from the bubble track, but none of them were promising in initial investigations so they were not considered further in this thesis.

### 6.3.4 Support vector regression

As described in Section 2.2.5, support vector regression (SVR) is a form of supervised machine learning that performs a regression using kernels to allow for the prediction of more complex relationships between the input variables and the variable to be predicted, compared to a simple linear regression. I used the SVR function in the sklearn Python package to perform this regression, along with the GridSearchCV function for cross-validation to choose parameters in the model.

First I split my data into a training set and a set for validation and testing using the train_test_split function in sklearn. Because the relationship between the autocorrelation function and the distance from the bubble track may be different at different distances, I used stratification when separating the data into these two sets. This approach involves sorting the data into a certain number of bins based on the target variable, in this case the distance, and selecting a random portion of each of these bins to go into the training and testing sets, rather than selecting this random portion from the dataset as a whole. The effect is that the distributions of distances in the testing and training sets remain more similar to the distribution of the entire data set, as shown in Figure 6.5, which should help to ensure that the trained model is able to predict all distance values as well as possible without any bias introduced by the train/test split. I repeated the regression using different numbers of bins for stratification, from 4 to 9, to ensure that my final result was not significantly impacted by the number of bins. As I had no additional data available, I retained the testing data set for the final
validation of my model rather than using it for model optimisation.

Figure 6.5: Examples of a train/test split on my data. Panels (a–c) show the distance distribution of each set when split without stratification, and (d–f) with stratification. Stratification has prevented the over-representation of the smallest distances in the training data and therefore their under-representation in the test data.

Using 5-fold cross-validation on the training set, I found the best performing combination of kernels and their relevant parameters. I considered a radial basis function kernel:

$$K(x_i, x_i') = \exp\left(-\gamma \sum_{j=1}^{d} (x_{ij} - x_{i'j})^2\right),$$  \hspace{1cm} (6.2)$$

with values of $\gamma$ ranging from $10^{-5}$ to $10^2$ incremented in powers of 10, and a linear kernel:

$$K(x_i, x_i') = \sum_{j=1}^{d} x_{ij}x_{i'j}.$$  \hspace{1cm} (6.3)$$

For both of these kernels I ranged the parameter $C$ in the support vector regressor, which dictates the width of the allowable margin of error when fitting the regression curve, from 1 to $10^5$ incremented in powers of 10.
6.4 Results and discussion

6.4.1 Image segmentation

Trained Weka segmentation

The process segmenting images using the Trainable Weka Segmentation plugin in ImageJ required some degree of trial and error, in particular relating to the source of the initial pixels chosen to train the segmentation algorithm. I trained a number of different segmentation classifiers using pixels from different images and combinations of images. Figure 6.6 shows the results of applying two example classifiers to some example images in the data.

It is immediately clear that the results are very different depending on the choice of image used to select the training pixels. Panels (d–f) show the result of a classifier trained using pixels from the image shown in panel (a) applied to images (a–c), and unsurprisingly the result of the classifier on the training image is good, with a clean boundary between bijel and hole. In the other two images, the classifier fails to identify dark regions of the bijel structure and bright regions of the bubble track, which makes sense given that such regions do not exist in the image used for training.

I therefore trained a new classifier on pixels from image (b), which includes both of the features on which the first classifier failed to identify. The segmentation results based on this classifier are shown in Figure 6.6(g–i). The segmentation now correctly excludes the bright region of bubble track from the bijel class, and performs a little better when classifying some of the dark bijel regions, but this comes at the expense of the introduction of small patches classed as hole within the bijel region of all three images. The segmentation also fails to correctly classify the grey region between the bright and dark parts of the bubble track in panel (h), despite the fact that this is the image on which the classifier was trained and examples of this region were included in the pixels selected for training. If the classifier is trained on features from both images (a) and (b), the result is very similar to when it is trained only on image (b), so there is little point in discussing it further other than to note that the features from image (b) therefore seem to dominate the classification decision by the algorithm.
Figure 6.6: Three examples of the results of two attempts at Weka segmentation trained on two different images. Panels (a–c) show the original images. Panels (c–f) show the corresponding segmented images based on training pixels selected from image (a). Panels (h–j) show the corresponding segmented images based on training pixels from image (b). The green and red regions are those classed as bijel and hole, respectively.
Figure 6.7: Three examples of the results of image segmentation via seeded region growth. Panels (a–c) show the original images, and panels (d–f) show the corresponding segmented images. The red regions are those classes as bijel and the green, yellow, and blue regions are classed as dark, intermediate, and bright regions of hole, respectively.

In the case of seeded region growing, any degree of trial and error was constrained to an individual image. In general, I found that each image required at most two iterations of seed selection in order to find a suitable result, such as the ones shown in Figure 6.7. These results (panels (d–f)) show clear regions of bijel and hole, and the edge of the bubble track seems to have been sensibly drawn. The hole regions are generally a little wider than the Weka segmentation results, but when compared to the original images in panels (a–c) this wider bubble track looks more akin to the original. This is particularly easy to see in the left-most panels of Figures 6.6 and 6.7.
Discussion

The main benefit of the Weka approach to segmenting this data was that once I spent some time training the classifier, it could be applied to any number of additional images with no additional input. This can save time and means that any resulting segmented image are based entirely on the features of that image and unaffected by human input. I tried using seeded region growing on some images where the bubble track was much less clear than in the nine examples I use in this chapter, but the method was only able to return a result almost identical to the rectangular seeds given. This is in contrast to Weka segmentation which could at least give a result based on the features of the image, but this fact does not make up for the fact that Weka segmentation performs poorly on the more ideal images.

This potential speed benefit was somewhat reduced by the fact that it took a number of seconds to load the classifier and apply it to each new image, so the time saved is small if an image needs only a small number of seeds for region growth, but the real problem with this approach is that the bijel and hole regions in the resulting images both contain small sections of the other class. This is a sign that, despite the use of Gaussian blurring to smooth the images, Weka segmentation struggles with the fact that such images of bijels are effectively full of small holes where the non-fluorescent liquid phase resides. It is possible that introducing stronger Gaussian blur or removing features with weaker blur could improve the segmentation performance, so this could be worth trying in future although it may not improve the result. An additional processing step is therefore required before using the segmented image to reject sections of the image that contain the hole: the regions of hole in the bulk of the bijel need to be removed or ignored. Whether this is achieved by further processing the image or by adapting the hole checking method to only include larger areas of hole, the additional work makes the Weka-segmented images a less favourable option.

On the other hand, although seeded region growing requires the selection of seeds for every image the end result is very clear and will be easy to use in my further analysis. I did perhaps give this method an advantage by including, where necessary, three different classes of hole so that the bijel could be correctly separated. It could therefore be worth seeing if the Weka segmentation could be improved by introducing these additional classes of hole, particularly as this method can be more widely applied to different images. Weka segmentation could
also be trained individually for each image, similar to the selection of seed regions for region growth segmentation, but the Weka approach would then be losing its main advantage which is the fact that classifying a future image requires no human input. The approach would also be very slow because training the classifier takes much longer than applying the trained classifier to a new images, since the algorithm is optimised for producing quick answers from an already trained model. Given the success of the seeded region growing method and the small number of images to segment, I decided not to spend further time on improving the Weka segmentation of this image set.

Overall, segmentation via seeded region growth is clearly the more suited approach for my needs in this case. Because I have a small number of images, and each image contains a small number of regions, the speed advantages of the machine learning approach are not useful enough to justify further optimising it for a better performance. The approach of seeded region growth works particularly well for my data because it can only create continuous regions from each seed, which means that it is not susceptible to brightness variations in the bulk of the bijel because they aren’t connected to the hole. It perhaps also benefits from the fact that it views each region on an averaged basis rather than a pixel-by-pixel basis: this may make it more suitable for bijel images since they always vary in brightness. The areas of bijel are, on average, a different intensity to the regions of hole so the method can easily tell the two apart. For the remainder of my analysis I will therefore use segmented images obtained by the seeded region growth method.

6.4.2 Relationship between bijel orientation and distance from bubble track

In this part of my investigation of the data, I first explored how much information could be gained from the data using different sizes of sub-image. Unfortunately, I was not able to calculate a reasonable estimate for the orientation of two of the segmented images because they contained multiple bubble tracks and this disrupted the calculation. I therefore omitted these images from this part of my analysis, but included them again when I was not using the orientation. My images were originally 512 × 512 pixels, so I tried splitting them into square sub-images with side lengths of 128, 64 and 32 pixels: any larger than this and there would be very few sub-images that did not contain the bubble track, and any
smaller would be too small to see any features of the bijel in a sub-image. For each sub-image, I calculated the orientation angle of the bijel structure relative to the orientation of the bubble track, and plotted this against the distance of the centre of the sub-image from the nearest part of the bubble track. The calculated angle as a function of distance is plotted for each size of sub-image in Figure 6.8, and it is clear that the data have a broad spread and are symmetrical about an angle of 0°. It is also clear that $128 \times 128$ pixel sub-images give too few data points to see any real patterns in the data, so I consider only $64 \times 64$ and $32 \times 32$ pixel sub-images going forward.

![Figure 6.8: Scatter plots showing how the angle of each sub-image relates to its distance from the bubble track. Three different sizes of sub-image are shown: (a) $128 \times 128$ pixels, (b) $64 \times 64$ pixels, and (c) $32 \times 32$ pixels.](image)

As the distributions are symmetrical about 0°, I next investigated how the magnitude of the angle changes with distance from the bubble track. I split the distance data into bins, with the bin size (and therefore the resolution of the result) set to a quarter of the length of the image chunk. For each bin, I calculated the mean of the absolute orientation angle of the sub-images in the bin. Figure 6.9 shows how this changes with distance: there is some tendency for the orientation angle to decrease with increasing distance until around 450 µm from the bubble track, and after that point there is no clear pattern to the data. This might indicate that the bijels are, in general, somewhat aligned in the same direction as the bubble tracks (due to the force applied to both by centrifugation) and that the passage of the bubble distorts the bijel by dragging it to a different orientation.

Overall, the orientation of the sub-images does seem to change with increasing distance from the bubble track. This indicates that there may be some information in this feature of the autocorrelation function of the sub-images that is linked to the effect of the passage of the bubble through the bijel. However, Figure 6.9(c) shows that there is a huge amount of variation in the orientation
Figure 6.9: Scatter plots showing how the average absolute magnitude (in degrees) of the orientation of sub-images at different distances (in µm). Panel (a) shows the result for 64 × 64 pixel sub-images. Panels (b) and (c) both show the result for 32 × 32 pixel sub-images, but panel (c) includes the standard deviation of the angle in each distance bin shown as the pale blue area behind the scatter points.

at each distance so it is difficult to draw any conclusions based on this variable. It would certainly be difficult to accurately predict the distance of a sub-image from the bubble track based on this variable alone.

Given that the orientation of a bijel has been previously shown to clearly change in response to shear forces [103], a phenomenon that is visible by eye in some of the images used in this analysis, it is possible that this difficulty in drawing a clear conclusion from the data presented in this section may be due to an inaccuracy in the method of determining the orientation angle of each sub-image. It clearly works well on the sub-images I used to test it but there is potential for a lot of variation in the sub-images from different original images. I also note that a lot of the sub-images may be close to isotropic with very little trend in orientation, and it would be very difficult to accurately determine the orientation of such data points. In light of this, I did try refining the PCA-based orientation calculation by weighting sub-images with a more obvious orientation, as well as using this and other methods (such as Sobel filters) on the sub-images themselves rather than their autocorrelation function. However, I was not able to improve on the clarity of these results.
6.4.3 Support Vector Regression to find distance from bubble track

Given the conclusions in the previous section, I focussed my use of support vector regression (SVR) on the entire autocorrelation function of each sub-image, rather than from variables derived from it such as the orientation angle. This meant that I could investigate whether the distance of a sub-image from the bubble track can be predicted from the autocorrelation function of the image, and although I cannot draw conclusions relating to the orientation of the bijel I can still see how the effect of the bubble’s passage changes with depth into the bijel bulk.

Figure 6.10: Plots of the true vs. predicted distance of each 32 × 32-pixel bijel sub-image from the bubble track, in µm. The first panel shows the result for the entire dataset, the second panel shows the result for the training data, and the final panel shows the result for the test data. The training data is expected to be the closest to the perfect result, shown by the red line. The training and test data were selected from the full dataset using a stratified train/test split with 5 bins. Each point is plotted with transparency so the darker regions are those where more points overlap.
Figure 6.10 shows how the predicted distance correlates with the true distance from the bubble track. This example is the result of stratifying the 32-pixel sub-images into 5 bins, but the results from other numbers of bins were similar. Using cross validation I found that the best regression model for this data used a radial basis function kernel with $\gamma = 0.01$ and $C = 1000$: all versions of the regression shared the fact that a radial basis function was the more suitable kernel and that generally high values of $C$ were optimal, but the optimal $\gamma$ varied from 0.001 to 0.1 depending on the data. It is clear from the plots that, for most distances, even the optimised regression model cannot accurately predict the distance of a sub-image from the bubble track, even for the training data. However, for predicted distances below 125 $\mu m$ the prediction becomes much more accurate. The regression is able to predict the training data very successfully, and provides a reasonable estimate for the test data: not an excellent performance but significantly better than for predicted distances above this 125 $\mu m$ threshold.

To investigate this result further I plotted the range of true distances present at different predicted distances, as shown in Figure 6.11. This plot confirms that the range of true distances predicted within a 25 $\mu m$ range of predicted distances increases dramatically after the predicted distance increases past the 125–150 $\mu m$ bin, implying a large decrease in the precision of the prediction.

![Figure 6.11: A plot of the range of true distances present in each 25 $\mu m$ bin of predicted distances. The red strip shows the point at which the range of the distances is the same size as the distance itself, and is plotted to give a reference for the scale of the variation within each bin.](image)

Interestingly, the regressor predicts no distances greater than around 450 $\mu m$, despite the fact that some true distances reach up to 750 $\mu m$. This may be
due to the fact that there are fewer examples of sub-images from these large distances in the data, or it may just confirm that the predictive power of the regressor is poor at larger distances. It is interesting that distances greater than roughly 450µm is also the range for which the relationship between distance and sub-image orientation becomes less clear and more chaotic. It seems likely that these phenomena are linked: presumably they are caused by the same underlying feature of the data.

Figure 6.12: Areas of bubble track (purple), distorted bijel (green), and unaffected bijel (yellow) superimposed on three example images. In particular, the distorted region defined as the area less than 125µm into the bijel from the bubble track, as determined by the distance past which the autocorrelation function can no longer predict the distance from the bijel.

The fact that the distance of a sub-image from the bubble track can only be accurately predicted up to a distance of 125µm suggests that this distance is the maximum distance at which the bijel is noticeably affected by the passage of the bubble. Comparing this result to the one based on the orientation of the bijel in Figure 6.4(b), there is perhaps some corroboration here in that the distribution of angles with respect to distance appears to be slightly less spread out below this value than above it. Although the earlier result is very unclear, the fact that it suggests something similar to this more robust result by an independent route is a good indication that the result is a true indication of the behaviour of the system. Further from the bubble track, the autocorrelation function is unrelated to the distance so this region can be viewed as the general bulk of the bijel that has no ‘knowledge’ of the passage of the bubble.

Figure 6.12 shows how the region of bijel affected by the bubble’s passage as predicted by the above discussion (shown in green) looks on some examples of the original images. In panel (a) it is very clear that this area of the bijel is indeed
more affected by the bubble than areas further from the bubble track, and this is true for many other images in the data. In panel (b) the bubble has not yet passed through the bijel, and therefore its effect is not visible in the boundary region: this is to be expected and is good validation that the behaviour that defines this region is due to the shear forces imposed on the bijel by the bubble, rather than the mere presence of a hole in the bijel. In panel (c) it is easy to see the effect of the bubble’s passage in some regions but not in others, which means that with this limited data there will always be some sub-images that are unusual for their distance. It would therefore certainly be worthwhile to investigate this further with a more extensive dataset where the effects of unusual sub-images can be considered and any outliers can be therefore removed as required.

I can now consider what this estimated distance of 125 µm from the bubble track might mean in terms of the forces on the bijel, comparing my results to some from the previous research [153] discussed in Section 6.1.2. The extent of the yielded region of a microfibrous cellulose gel (MCG) due to the passage of a bubble was reported to be 2–4 times the initial radius of the bubble used to apply force to the gel (50–100 µm). The size of the yielded region can be interpreted as the amount of material that must be moved before the structure yields and allows the bubble to move through it, and is therefore related to the local yield stress of the material. My estimate of the bijel yielding distance is towards the lower end of the reported MCG range of 100–400 µm, but the bubbles in my data are much larger (approximately 200–400 µm in radius) meaning that the yielded region of bijel is only 0.25–0.5 times the bubble radius. The initial size of the bubble is relevant because the buoyancy force is proportional to the bubble volume so the stress (force per unit area) is therefore proportional to the bubble radius.

Although I do not have enough information about the forces within the bijel to directly compare these values, they are much more similar in magnitude than the bulk yield stresses of the two different materials. The local yield stress of the MCG is 15–30 times larger than its bulk yield stress, both of which are around or below 1 Pa, whereas the bulk yield stress of a nitromethane/ethanediol bijel is on the order of $10^2$–$10^3$ Pa [106, 108] which would suggest a much smaller yield region if the local yield stress followed the same trend as in the MCG. I therefore conclude that my results suggest that the bijel is much weaker on a local scale than it is in bulk, especially compared to other gels, and it may be interesting to confirm and improve upon these preliminary results, perhaps with more experiments with the specific aim of using bubbles to investigate the local yielding behaviour of bijels.
6.5 Conclusions

In this chapter I took a more focussed approach to physics-led machine learning, using a combination of different methods to probe the effects of local shear forces on a bijel. First I needed to identify different regions of my images so I investigated how successfully different segmentation approaches could separate regions of bijel in an image from regions where a hole was left by the passage of a bubble. I found that the machine learning approach, trainable Weka segmentation, gave results that were consistently worse than those obtained using the traditional approach of seeded region growth. This highlights the fact that machine learning is not always the best tool for a job, and that simpler or more traditional image analysis techniques should also be considered. In this particular case it is likely that the pixel-by-pixel approach of Weka segmentation was less suited to the highly variable bijel structure than the area-averaged region growth approach.

I then investigated the relationship between the local orientation of the bijel structure and the distance from the bubble track. I found that the orientation was too variable to be a consistent indicator of distance from the bubble track, but there was some indication of a relationship between the two which suggests that this relationship might be worth revisiting if the orientation of each sub-image could be more reliably calculated. I considered that the autocorrelation function from which the orientation was estimated may still contain other information that correlates with distance from the bubble track, so I used the function itself in the rest of my investigation. In future it could be interesting to optimise a regression based on other variables derived from this function (ideally including at least one measure of the orientation), which would most likely require a systematic approach to choose the variables and optimise the regression model such as the one followed in Chapters 3 and 4, to see if the result is significantly different and if the variables in the model can be given further physical interpretation.

I found that the autocorrelation function of a sub-image does indeed change with increasing distance from the bubble track, and can be used to accurately predict this distance for both seen and unseen data, provided that the predicted distance is less than 125\(\mu\)m. I used this distance, which represents the distance below which the structure of the bijel is related to the passage of the bubble, to compare the local yield stress of the bijel to that of other materials, and found a very preliminary suggestion that the bijel structure is locally much weaker than its bulk behaviour might suggest, in comparison to other gel systems. There may
be another change in behaviour at a distance of around 450 µm from the bubble track, where the regression model stops predicting any data points entirely and the relationship between the bijel orientation and the distance from the bubble track appears to become more random. This result may, however, be the an artefact of the data: perhaps the lower number of data points with distances above this value is causing the result. I therefore do not interpret this result.

In general, this chapter gives promising preliminary results showing that a machine learning regression method such as support vector regression could provide an indication of how the structure of a bijel is affected by the passage of a bubble through the bijel. It shows that machine learning is an effective tool for more than just image processing, and can be used as part of a direct investigation into physical processes within a material. It would certainly be interesting to perform similar analysis on similar data where more was known about the dynamics of the bubble and the forces it applies to the bijel, so the result found here could be used to calculate, rather than vaguely estimate, the local yield stress of the bijel.
Chapter 7

Conclusions and future outlook

In this thesis I have investigated a number of situations in which machine learning can be used for the analysis of confocal images of soft matter systems. I took a physics-led approach to each machine learning task in order to obtain as much useful information about each system as possible. I used historical data for this investigation and focussed on two emulsion systems: the majority of my work considers bijels, while one chapter considers Pickering emulsions stabilised by oppositely charged particles.

In Chapter 3 I applied the k-nearest neighbours classification algorithm to a series of confocal images of the liquid channel of a bijel. I found this to be the best-performing classifier based on a number of variables derived from the radially averaged autocorrelation function and structure factor of each image. I then reduced the number of variables used as inputs to the classifier and found that the most optimal combination of variables was in fact a single one: the location of the first turning point of the autocorrelation function. Because I chose to, wherever possible, work with variables that could be interpreted physically in the system, I could conclude that this variable is an indicator of the characteristic length scale of the bijel system. It makes sense that this allows the identification of successful bijels because they have a single characteristic lengthscale which is defined by the concentration of stabilising particles in the system and thus constant across different bijels of the same composition.

The classifier I created in this chapter could classify bijels with a 16.4% error rate, as measured by cross-validation. Comparing this to the null error (in which every sample is classed as a bijel) of 31.9%, and the 21.5% error of the naïve initial
model, my classifier achieves a reasonable performance on the training data. The classifier performed well on unseen data consisting of additional images from some bijels that were already represented in the training data (with a classification error of 15.8% vs. a null error of 42.1%) but poorly on unseen images from different bijels (a classification error of 38.7% vs. a null error of 29.0%). A number of improvements could conceivably made to this classifier, and I therefore continued to explore this task in the next chapter.

In Chapter 4 I applied the logistic regression algorithm to classify images of the same bijels as in the previous chapter, but this time I used information first from the particle channel of the bijel and then from both the particle channel and the liquid channel. The particle channel classifier achieved an error rate of 18.0% on the training data and 10.5% and 29.0% on the similar and different unseen data, respectively. The classifier using variables from both the liquid and particle channels achieved improved errors of 14.6%, 10.5%, and 25.8% for the three datasets: the best performance of the models by all three measures. This combined classifier was the result of using the variables that gave the best results for each of the two single-channel classifiers: the first turning point of the autocorrelation function of the liquid channel and the gradients of the first 12.5 \mu m and 25 \mu m of the autocorrelation function of the particle channel.

I interpreted the physical meaning of the particle channel variables and deduced that they were approximately equivalent to the values of the autocorrelation function at approximately 12.5 \mu m and 25 \mu m. My model concludes that the latter is correlated to the probability of an image belonging to a successful bijel and the former is inversely correlated, which suggests that the particle channel of a successful bijel is likely to be more uniform over a range of 25 \mu m than that of an unsuccessful bijel, but less uniform over a range of 12.5 \mu m. This highlights the first change I would make to this work in future: I would repeat the particle channel analysis using the values at these two points as input variables instead of the gradients up to the points. I would therefore be working directly with the more interpretable version of these variables, and could have greater faith in this physical interpretation of my result. In fact, it would be even better if I could capture this feature without choosing specific lengthscales because different lengthscales may be more relevant in future data. Perhaps this could be achieved by considering the value of a range of points along the autocorrelation curve rather than just two: this would add more variables to the model but is likely to make it more generally applicable to other bijel images.
The results presented in Chapters 3 and 4 show that machine learning is a valuable tool for the classification of successful bijel fabrication attempts, and can reduce the need for an experimenter to manually assess whether each sample has been successful. I have also been able to investigate the key features that differentiate a successful bijel from a failed one, as well as ensuring that the decision process of the classifier is easy to understand. Even without any improvement to the model, the same algorithm with the same variables could be trained on some initial samples from a new bijel system and then used to predict whether future fabrications have been successful without having to manually analyse images of each one. The classifier gives significantly more false positives than false negatives, so the risk of incorrectly rejecting good bijel samples is small. In future it would be interesting to train and test my classifier on a different set of images of bijels of a different composition, to confirm my assumption that the three variables I have identified as useful will be equally useful when classifying different bijel images: as I have already discussed, this may not be the case for the particle channel variables.

My results suggest that the lengthscales in the bijel system are important for the successful identification of such images, which matches with the known physical properties of a bijel. This does, however, mean that the classifier has to be re-trained whenever the composition of the bijel, specifically the particle content, is changed because this changes the lengthscales of the system and therefore the key values of the variables in the classifier. It would therefore be interesting to repeat this process on a more extensive set of data including examples of lots of different types of bijel (especially those of different interface separations), with the aim of creating a more global bijel classifier.

In Chapter 5 I performed unsupervised clustering on slices from confocal images of oppositely charged particles on the surface of Pickering emulsion droplets, using the k-means and hierarchical clustering algorithms. Each of the two particle types was imaged separately and I used both imaging channels as separate images. I also applied more traditional image analysis to lower-magnification images of the same system, where I assessed the how the roundness of hand-identified droplets changes with the ratio of positive to negative stabilising particles. From this traditional analysis I found that the roundness of the least round examples increases as more negative particles are included to the system. This is in contrast to the reported behaviour of other variables in similar systems in the literature, where the size of droplets is observed to follow a symmetric distribution about
the point where positive and negative stabilising particles are present in equal amounts.

Using unsupervised clustering, I split each image into twelve slices and measured the suitability of the clusters based on how well slices from the same original image were clustered together. I found that the data fit well into five clusters, and that hierarchical clustering consistently gave better clustering results than k-means clustering. I then found that principal component analysis was a very effective tool for reducing the number of variables fed into the clustering algorithm, as was the reduction of variables by averaging over one dimension in the two-dimensional image slices to leave data based on the distance from the centre of the droplet. Neither of these approaches had much effect on my measure of cluster performance and the averaged data in particular made it easier to visualise the data and investigate the clustering decision.

Despite using the data in the form that was most easy to visualise it was difficult to see how the clustering decision was made. It seems that the sizes of bright areas in the image may be an important factor and this aligns somewhat with a trend in whether the image is of the majority or minority phase of particles. I found some evidence to suggest the same result as I found with my traditional analysis: that the examples with high proportions of negative particles are more similar to each other than those with lots of positive particles, and vice versa. This suggests that there may be a link between the roundness of a droplet and its surface structure, but further investigation would be required to confirm this. In particular, I only used a small number of distorted droplets in my clustering analysis and therefore cannot draw any firm conclusions about a link between droplet distortion and the arrangement of particles on the droplet surface. In future, I think it would be valuable to investigate a much larger number of droplets making sure to include a representative distribution of different droplet shapes for each different composition. This would allow for a much more robust investigation of any links between droplet distortion and particle-level structure.

In Chapter 6 I used confocal images of the passage of a bubble through a bijel during centrifugation to investigate the response of the bijel to the forces applied by the bubble. I first compared two different methods for segmenting an image to separate the bijel from the bubble’s track and found that, for this data, the traditional approach of seeded region growth was much more suitable than the machine learning approach of trainable Weka segmentation.
I split the bijel region of each image into smaller sub-images and used the autocorrelation function of each sub-image to represent the structure of the bijel at different distances from the bubble track. I used the hotelling transform to find the orientation of each of these two-dimensional functions and therefore the orientation of the bijel structure, and found that this variable was not consistent enough to be a reliable indicator of distance from the bubble track but there was some indication of a relationship between the two. In future it would be interesting to revisit this relationship, but this would require a more reliable method of calculating the orientation of each sub-image to ensure that the results were valid.

I used support vector regression to find the relationship between the autocorrelation function as a whole and the distance of a sub-image from the bubble track, by using the function as a set of input variables from which to predict the distance as an output variable. I found that the resulting regression model was able to predict the distance with some accuracy up to a predicted distance of 125 µm but at larger distances the predicted distance gave no real indication of the true distance of either the training or testing data. This result suggests that the effect of the bubble’s passage penetrates only 125 µm into the bijel. Comparison to similar experiments in the literature, where microrheology was performed on gels using bubbles, allowed me to draw a preliminary conclusion that the local yield stress of a bijel is much smaller than might be expected based on comparison of its bulk yield stress to other gels.

I would be very interested to repeat this analysis on data from an experiment specifically designed to investigate the local yielding behaviour of a bijel using the passage of a bubble, taking inspiration from the literature. If the behaviour of the bubble was controlled and measured, the stress required to locally yield the bijel could be calculated and this would allow the results from this chapter to be converted into a true measure of local yield stress rather than a comparative estimation. This would make my regression result much more useful because I would be able to draw firm conclusions based on it.

In conclusion this thesis has explored a range of ways in which machine learning can be applied to images of soft matter systems, specifically emulsions. I have shown that a wide range of problems, from the classification of samples during experimentation to the quantitative analysis of a system, can be approached using machine learning. I have also shown that, by taking the physics of the system into consideration when designing machine learning approaches to solve
these problems, information can be gained from the machine learning models themselves, not just their predictions. On the other hand, I have shown that in some cases machine learning is not the best way to approach a task and as such it should be viewed as an additional set of techniques to be considered for a given problem, rather than a universal solution. In general I found the exploration of data via unsupervised learning to be the least useful investigation, mostly because it was the most challenging to interpret, but the result did support the conclusions of my other analysis via an unrelated method so the attempt was worthwhile. Other forms of unsupervised learning, such as principal component analysis, proved to be very useful for dimensional reduction and my investigations using supervised learning were also rewarding. The observations in this thesis will, I hope, encourage soft matter physicists to consider how machine learning might be used to accelerate parts of their experimentation process, confirm results from other methods of analysis, or reveal more information from their data.
Bibliography


74 B. P. Binks and T. S. Horozov, Colloidal Particles at Liquid Interfaces (Cambridge University Press, 2006).


E. Fix and J. L. Hodges, *Discriminatory analysis: nonparametric discrimination, consistency properties*, tech. rep. (USAF School of Aviation Medicine, Randolph Field, Texas, 1951).


