Extracting Motion Primitives from Natural Handwriting Data

Ben H Williams

ben.williams@ed.ac.uk

Doctor of Philosophy

Institute for Adaptive and Neural Computation

School of Informatics

University of Edinburgh

2008

Abstract

Humans and animals can plan and execute movements much more adaptably and reliably than current computers can calculate robotic limb trajectories. Over recent decades, it has been suggested that our brains use *motor primitives* as blocks to build up movements. In broad terms a primitive is a segment of pre-optimised movement allowing a simplified movement planning solution. This thesis explores a generative model of handwriting based upon the concept of motor primitives. Unlike most primitive extraction studies, the primitives here are time extended blocks that are superimposed with character specific offsets to create a pen trajectory. This thesis shows how handwriting can be represented using a simple fixed function superposition model, where the variation in the handwriting arises from timing variation in the onset of the functions. Furthermore, it is shown how handwriting style variations could be due to primitive function differences between individuals, and how the timing code could provide a style invariant representation of the handwriting. The spike timing representation of the pen movements provides an extremely compact code, which could resemble internal spiking neural representations in the brain. The model proposes an novel way to infer primitives in data, and the proposed formalised probabilistic model allows informative priors to be introduced providing a more accurate inference of primitive shape and timing.

Acknowledgements

Many thanks to my first and second supervisors, respectively Amos Storkey, and Marc Toussaint, without whom this research would not have taken course.

Thanks go also to Chris Williams for providing helpful feedback and comments on the research, and suggesting related material.

Financial support was gratefully received from the EPSRC and MRC, which jointly fund the Neuroinformatics Doctoral Training Centre within the School of Informatics, and support several post graduate students each year.

Finally, thanks go to my parents for financial and moral support, and moreover to Val Williams for proof-reading this work.

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.

(*Ben H Williams*
ben.williams@ed.ac.uk
)

Table of Contents

Chapter 1

Introduction

This project aims to draw from two large areas of current research to form a probabilistic model of handwriting, based upon the biological/robotics concept of motor primitives.

Recently there have been many biological studies looking at how our brains plan and control movement. These can be broken down into electrophysiological studies (Section 2.1.1) on various animal models, and psychophysical studies (Section 2.1.2) performed on human subjects. Almost all of these studies focus on reaching and grasping tasks, or more general limb extension tasks.

In the domain of data modelling, there have been several recent models of handwriting, mostly prompted by the invention and popularity of hand held personal organisers incorporating digitisation tablets for input. Most of these models use some form of modularisation, requiring a pre-segmentation of handwriting which influences the model, and can often be unreliable. This project proposes a primitive based model that modularises handwriting without requiring pre-segmentation.

Handwriting is an almost universal form of human communication which, before the invention of printing, was a unique skill that allowed ideas and experience to be passed on in a much more reliable format than with speech. Reading and writing other people's handwriting is a difficult task for computers and in some cases for humans as well. This is mostly due to different styles of handwriting which lead to very different markings meaning exactly the same thing. Also there is a great degree of variation in a single person's handwriting, which is normally attributed to noisy motor control. Automated handwriting recognition is therefore a very unreliable process, and is greatly aided by some higher level language model, as in speech recognition. However, if the variability in handwriting is better understood, then the low level recognition could

become more robust.

The aim of this project is to explore a model of handwriting based upon motor primitives. Assumptions about the nature of motor primitives translate into assumptions about the type of variability that might be present in handwriting data.

Movement planning and control is a very difficult problem in real-world applications. Current robots have very good sensors and actuators, allowing accurate movement execution, however the ability to organise complex sequences of movement is still far superior in biological organisms, despite being encumbered with noisy sensory feedback, and requiring control of many non-linear and variable muscles.

The advantage that biological control has is therefore likely to be due to a superior internal representation and a more robust sequencing. There is much evidence to suggest that biological movement generation is based upon *motor primitives,* with discrete muscle synergies found in frog spines, (Bizzi et al., 1995; d'Avella and Bizzi, 2005; d'Avella et al., 2003; Bizzi et al., 2002), evidence of primitives being locally fixed (Kargo and Giszter, 2000), and modularity in human motor learning and adaptation (Wolpert et al., 2001; Wolpert and Kawato, 1998). Compact forms of representation for any biologically produced data should therefore also be based upon primitive sub-blocks.

1.1 Problem statement

Handwriting has much variability, arising from biological noise in the planning and execution of the necessary movements to produce a character. Many muscles must be activated in a coherent, coordinated fashion to control a complex system of joints in order to achieve robust control of a pen. The possibility of modelling such a non-linear system using a linear model would be very attractive if the variability in the data were well captured. This project examines the use of a simple model of superimposed offset functions to model pen trajectory data. The shape of the functions is not constrained or limited to a particular set, and as such there are many parameters that need to be learnt. This project explores the use of a probabilistic framework to infer the most likely shapes of these functions, and their positions of occurrence.

If motor primitives can be approximated by fixed functions, at least for a specific task, then modelling of the data is greatly simplified, and it provides a useful framework for recognition tasks. This project explores whether the primitives present in handwriting are consistent enough to be modelled by a function superposition model.

Furthermore, if the primitives inferred are adequate to model handwriting data, then the timing of these primitives could provide an abstraction of the character, or a timing code. Examination of whether this timing code itself can be modelled and usefully generalised across multiple character samples will provide further analysis of the appropriateness of such a model of primitives.

1.2 Model overview

The model investigated in this project envisages handwriting data as a superposition of sparsely activated motion primitives. This approach can intuitively be compared to a Piano Model which has also been called a Piano roll model (Cemgil et al., 2005). Just as piano music can approximately be modelled as a superposition of the sounds emitted by each key, biological movement can be represented as a superposition of pre-learnt motion primitives. This implies that the whole movement can be compactly represented by the timing of each primitive by analogy to a score of music. The formulated model presented in this project reflects these assumptions. The model can be described on two levels, corresponding to Chapters 4 and 6, where on the lower level a factorial Hidden Markov Model (fHMM) (see Ghahramani and Jordan (1997)) is used to model the output as a combination of signals emitted from independent primitives, where each primitive corresponds to a factor in the fHMM. On the higher level there is a model for the primitive timing dependent upon character class. The same primitive functions are shared across characters, only their timings differ. This model is trained on handwriting data using an EM-algorithm and thereby infers the primitives and the primitive timings inherent in these data. The inferred timing posterior for a specific character is a compact representation for the specific character which allows for a good reproduction of this character using the learnt primitives. Furthermore, using the timing model learnt on the higher level, new samples of characters can be generated which are in the same writing style as the data, and also scribblings that exhibit local similarity to written characters when the higher level timing control is omitted.

1.3 Chapter overview

Chapter 2 reviews scientific literature relating to the project. Chapter 3 defines how primitives are approximated in this project, and some model variable definitions. Chapters 4 and 5 provides details of the low-level primitive model, and chapter 6 examines

several possible timing models. Chapter 7 details how the primitive and timing models are coupled together during learning. Chapter 8 assesses the performance of the model, and compares it to some alternative models. Chapter 9 reviews the implications of the project, and possible uses and extensions.

Chapter 2

Background

Simply reaching out and grasping an object is a task that we as humans generally take for granted, and normally requires minimal conscious control. For decades the study of human and animal movements has been the focus of much scientific interest, partially motivated by the still comparatively poor performance of modern robotics and also by a need to understand our own control systems, and their failings, for medical reasons.

In (Wolpert, 2007), Wolpert points out a significant contrast in the ability of computers to beat humans at chess, with the inability of a robot arm to move the actual chess pieces with as much dexterity as even a young child. (Wolpert, 2007; Wolpert et al., 2001) have reviews of Bayesian decision theory as applied to biological motor planning and control. The difference between playing chess theoretically, and controlling a robot arm in the real world, is that all the variables are known, or at least calculable and discrete in the chess game, whereas in the real world the knowledge of the world is based upon noisy sensors that only partially measure a subset of the world's state. Additionally, the actuators (motors or muscles) have a degree of noise, leading to unpredictable movements. Finally the world is a dynamic state, and so any internal model of the world needs to be continually updated and revised.

There are two possible extremes of control strategies. Firstly, the traditional control approach is to have a fast feedback loop, and some controller that acts to minimise an error signal. This works with simple control problems, however in the real world the feedback is too slow to be useful due to the complexity of any potential error signal calculation. Secondly, there could be a pre-learnt repertoire of movements in which entire task related movements are optimised and then selected by a central controller at a given time. The problem with storing so many movements is that for any real world situation the storage and selection capabilities become inadequate. Motor primitives

can be thought of as a compromise between these two extremes. They are loosely defined as sub-blocks of muscle activation synergies that are fitted together to make up a complete movement. Therefore, the same primitive might be used for many different movements. Over the past decade there has been much biological evidence in support of the existence of motor primitives.

2.1 Motor primitives

2.1.1 Electrophysiological evidence

Strong evidence for motor primitives was first found in frogs (Bizzi et al., 1995) where stimulation of a single spinal motor afferent triggered a complete sweeping movement of the frog's leg. In fact the stimulation of specific sites on the spinal column induced spatial force fields for the limb. These force fields were linearly superimposed when triggered concurrently. Thinking of these force fields as motor primitives implies that the motor control and hence behaviour of the frog may be built up of similar superimposed motor primitives.

There has been much work investigating these motor primitives, or muscle activation synergies. (d'Avella and Bizzi, 2005; d'Avella et al., 2003) used electromygraphic recording techniques to record the natural activation of frog leg muscles. Using component factorisation techniques, they showed evidence of modularisation of the motor control system. Extending this model to higher levels of motor control, (d'Avella and Bizzi, 1998) stimulated the vestibular nerve in several different places, and performed PCA on the resultant force fields. They showed that 94% of the total variation of the data could be explained using only four principal components, suggesting that the movements were built up in a modular way.

There is also a strong body of evidence to support a sharing of resources between motor control and action perception. (Gallese et al., 1996) recorded from neurons in the pre-motor cortex, and found that they responded similarly when a specific action was performed by the subject, and also when the same action was observed. These neurons have become known as mirror neurons, and they are the strongest evidence that the brain contains structures that deal both with action and perception. This suggests that if the movement is generated using some spectrum of primitive activations, then perceived movements would be represented in a similar primitive activation space.

For an overview of the inverse dynamics problem that must be solved for motor

planning and control, see (Mussa-Ivaldi and Bizzi, 2000). They propose a possible solution through the use of motor primitives and an extension of the established spinal cord basis of muscle synergies to higher motor planning areas. For a review of the modularisation of motor control in the spine, see (Bizzi et al., 2002).

2.1.2 Psychophysical evidence

The many strategies by which humans and animals can accomplish any single motor task in the real world have an infinite number of solutions. Despite this, we show consistency both in varying situations, and across subjects (Todorov and Jordan, 2002; Matarić, 2004; Wolpert et al., 2001; Wolpert and Kawato, 1998). This suggests people use similar movement strategies. A central assumption is that such a strategy should be in some way optimal, due to the pressures of natural selection. What exactly is being optimised however is unclear. Wolpert has investigated this question using mostly reaching and grasping tasks.

Wolpert's central hypothesis, becoming increasingly accepted is that the brain uses an approximate form of Bayesian inference to best predict what is happening in the world, and also to plan movement in such a way as to minimise end-point position noise in important dimensions. There must be some internal model for the purposes of planning and prediction, and Wolpert suggests that this model is a combination of multiple paired modules, modelling both consequence and cause (see (Wolpert and Kawato, 1998)).

There has been much work by Wolpert and colleagues (Wolpert et al., 2001; Davidson and Wolpert, 2003, 2004; Hamilton et al., 2004; Körding et al., 2004; van Beers et al., 2004; Witney and Wolpert, 2003; Caithness et al., 2004) looking at how our internal forward models are structured and applied in motor adaptation tasks. They suggest that there are multiple forward and inverse internal models in the brain, and that the movement strategies are selected so as to minimize muscle activation noise, and therefore end point error. For an overview of their research see (Wolpert, 2007; Wolpert and Kawato, 1998). Their model tries to explain both motor adaptation experimental results, and motor optimisation strategies. It has long been noted that learning different tasks in a similar environment is more difficult than learning two unrelated tasks. The tasks interfere with each other. However it is possible to switch motor behaviour in different environments, for instance motor skills when playing tennis, or driving a car. They believe that there is a modularisation of internal models, which can be in-

dependently activated for specific circumstances. This can go some way to explaining the phenomenon of motor learning interference. Furthermore, they hypothesise that movement strategies are evolved to minimise end-point error. This is a novel alternative to the common idea of minimising some path distance metric, such as absolute distance, or average muscle force needed, or jerk minimisation. They have obtained many experimental results to support this theory, which takes into account the fact that muscle noise is proportional to muscle activation.

Wolpert's experiments generally are concerned with reaching and grasping tasks. Although these tasks require far larger movements than for pen-control tasks such as handwriting, it has been suggested that a similar modelling framework could be adapted for both types of task, for instance (Meulenbroek et al., 1996) explore the capability for a person to write on many different scales, and even with different limbs.

During a reaching and grasping experiment, if the target is suddenly changed whilst a movement is underway, the manner in which people are capable of reacting reveals characteristics of the underlying system. (Kargo and Giszter, 2000) showed that rather than simply correcting the movement, the subjects would superimpose the original movement with another that had the effect of correction. This suggests that the primitives have a time-extended and fixed component, at least once a movement has been commenced.

The locally fixed, time-extended nature of primitives is a central assumption in this project, and is supported by the distributed computation and representations that must exist in the brain. If primitives are considered to be time slices, then there must be a regular and very fast clock signal in the brain. There is no evidence to support this, and therefore it is much more likely that primitives are 'triggered' at specific times, and then are active for an extended time period, probably overlapping with several other primitives for complex and adaptable movements.

The hypothesis that complete movements are made up of superimposed, time extended blocks of movement implies that somewhere, there must be a timing circuit that 'fires' the appropriate primitive at the appropriate point. There is evidence that the cerebellum is involved with motor function, and more specifically with timing of motor function, and perception. (Meegan et al., 2000) showed that perceptual learning of rhythms improved performance in motor tasks which involved the learned rhythm. (Dennis et al., 2004) showed a relationship between cerebellar volume in children and performance in motor timing tasks. (Penhume et al., 1998) used PET imaging to show that the cerebellum provides a supramodal contribution to motor timing tasks. They

hypothesise that the cerebellum provides the necessary circuitry to extract timing information for the sensory and motor system.

If movements are repeatably composed of locally fixed sub-blocks, then with enough samples of movement, it should be possible to infer what these sub-blocks might be, and thus extract the primitives from data.

2.2 Modelling primitives

Modelling primitives for the purposes of robotic control has become popular recently, with the hope of simplifying the kinematics-dynamics transformation problem (Mussa-Ivaldi and Bizzi, 2000; Padoa-Schioppa et al., 2002). The central concept being that the primitives are in some way optimised for a segment of the transformation from muscle control to movement, and so rather than having to calculate the entire kinematic-dynamic transformation for a new movement, the brain has to activate relevant primitives.

Assuming that the primitives help the brain with the movement planning problem, it would also be useful to make use of them in robotics, not least for the purposes of copying gestures. The problem therefore is to define what a primitive might be, and to infer a set of useful primitives.

There are several different approaches to modelling and extracting motor primitives in robotics. (Ijspeert et al., 2003; Schaal et al., 2004) learn non-linear attractor systems defined by differential equations, which are adapted to perform new tasks. (Amit and Mataric, 2002) propose a two-layered system of primitives, which creates a single attractor point, which is modulated to produce a movement. Similarly, (Mussa-Ivaldi and Bizzi, 2000) support the equilibrium-point hypothesis, where a movement is defined as a series of attractor points, and (Drumwright et al., 2004; Fod et al., 2002) segment movement into events, and cluster the movement segments to define primitives.

An interesting primitive based study of arm movements was conducted by (D. del Vecchio, 2003), where movements were segmented into *movemes*. These movemes were taken from an alphabet of dynamical systems, and the segmentation allowed classification of drawing tasks. The segmentation points were global transitions from one moveme to the next, disallowing overlapping of movemes. This view of primitives sees movement as a series of switching events, rather than an ongoing, parallel process of timed events as is considered in this thesis.

These approaches define a primitive as a segment of movement, rather than allow-

ing a movement to be built up by multiple over-lapping primitives. The primitives are generally assumed to be active either for the entire movement, or between pre-segmented intervals. In real world tasks, the variety of movements that is required is enormous, and each movement does not have a clear starting point, but follows on fluidly from the last. This suggests that the primitive sub-blocks making up the movement can probably be superimposed. This is supported by (Bizzi et al., 2002) which suggests that the primitives in a frog's spine when activated in combination produce a force field that is a linear superposition of the force fields produced by the separate primitives.

A related area to limb control is robotic navigation, and the force field summation principle of limb control proposed in Bizzi's work can also be found in navigation studies such as (S.R. Lindemann, 2005). There are some differences between the problems of limb control and navigation, such as the constraints on the task. In general navigation constraints relate to obstacle avoidance, whereas limb control is more concerned with speed, accuracy of path, and in particular accuracy of end point position. However it is interesting to note that a similar model can be used for both tasks. Primitive based navigation policies can also be found, such as (D.C. Conner, 2006), where sequentially composed movements are constructed using switching of local feedback control policies. Similarly to (D. del Vecchio, 2003), there is no overlapping of the policies, thus the movements are segmented by global switching points. This implies a single central synchronous controller rather than large parallel operation of multiple asynchronous controlling elements, as is likely to be the case in the brain. Most robotics studies consider primitives in such a fashion, effectively as a method of segmenting movements. This thesis considers primitives to be individually segmented into their onset times, but independent from each other and without any global switching of primitives.

Pre-segmentation followed by PCA/ICA is often used to extract components that are referred to as primitives (Fod et al., 2002). These primitives are therefore strongly defined by the segmentation points within the movement. For certain movements with clearly defined start points, this technique is effective for movement classification, mainly due to the speed of ICA algorithms. For a review of ICA techniques, see (Hyvärinen, 1999), and an interesting extension to ICA is proposed by (Karklin and Lewicki, 2005) focussing on image patches. However in the real world, movements cannot be segmented into start points, and so using a windowed technique is difficult.

2.2.1 A piano model of primitives

Assuming that the primitives can overlap, that they have no common start point and that they cannot be instantly switched off, there is an analogous model that is used in music. A simple piano can be modelled as a series of key presses at different times. These key presses give rise to sound waves which are overlaid to create the music that the audience hear. (Cemgil et al., 2005) name such a model a 'piano-roll' model, due to its similarities with the pianola or reproducing piano. The idea of this analogy is to use such a model to infer *which* notes are being played *when* in a piece of music. This model is further discussed in Chapter 3.

Assuming the data is made up of time-extended functions with specific offsets, one way to extract them is by using a pattern extraction algorithm. (Chiu et al., 2003) propose an efficient algorithm that finds recurrent patterns within a time series. This model uses tolerance levels to determine how similar the data must be to define a motif, rather than formally modelling noise in the data. If several motifs are superimposed with different offsets, it is uncertain how successful such a pattern-matching based approach would be in reconstructing the original separate motifs.

The offset function model is also attractive for auditory encoding, as is suggested in (Lewicki and Sejnowski, 1999; Smith and Lewicki, 2005). Although they are not trying to model motor primitives, the concept is similar to the piano model proposed in Chapter 3. A sound wave can be decomposed into a set of basis functions with various offsets. Once a reasonable reconstruction is obtained, the basis can be adapted to better represent the signal, thus effectively extracting primitives from the signal. More details and a comparison with this model can be found in Section 8.5.2.3.

2.3 Handwriting

Handwriting is an extremely useful skill that has revolutionised human communication over the past 6000 years, allowing ideas and discoveries to be communicated more precisely and repeatably than by spoken word. Evolutionarily speaking, it is a relatively new motor skill, as compared to walking, grabbing, or even speech. These older skills are likely to have evolved a specialised brain area over a longer period of time, while handwriting is probably controlled by a generalised motor planning area in the brain, responsible for generic adaptable behaviour. The biological control of handwriting, despite its enormous importance, and high symbolic information content is probably

treated similarly to any another hand movement by the brain. Handwriting was chosen as the dataset to model for this reason, and because it is now easy to get very accurate pen trajectory data from digitisation tablets.

Most handwriting studies over the past three decades have been focussed entirely on recognition. For a review of handwriting recognition see (Meulenbroek and Gemmert, 2003; Beigi, 1993). Handwriting and more general movement studies have found a consistent relationship between angular velocity and curvature of movements, known as the two-thirds law. This is explored in a novel way by (T. Flash, 2007) who propose that this relationship is due to a non-Euclidean internal geometrical representation of space. They show how successive application of geometrical transformations can be used to construct movements. It would be interesting to examine whether geometrical primitives could be used to produce movements that have similar properties as biologically produced movements.

An inverse dynamics approach to handwriting was explored by (Singer and Tishby, 1994), in which cursive handwriting was treated as a system of oscillators, which could be modulated to create characters. This approach has similarities with the primitives proposed by (Ijspeert et al., 2003; Schaal et al., 2004), which combine oscillatory and transient primitives.

An interesting framework for modelling handwriting was proposed by (Hinton and Nair, 2005), which uses a spring system to model the pen dynamics, where the stiffness of the springs over time dictates what character is being written. Inference of a motor program is done for a single character image, and then noise is added to the image and the motor program in such a way as to train a neural network to learn the distribution over motor programs that can create such an image. With a few additional algorithmic features, a good recognition performance is obtained.

Examination of human learning of handwriting can potentially reveal aspects of the internal representation. (Marquardt et al., 1999) support the idea of primitives in handwriting by varying the visual feedback. They find that varying the size of characters will produce an adaptation in the handwriting only after the first character written. In a second experiment they vary a target during a handwriting stroke, and find that rather than aborting the initial movement, a second stroke is added. This is similar to the findings of (Kargo and Giszter, 2000) where the experiment is a reaching and grasping task.

(Kharraz-Tavakol et al., 2000) found that when learning two new but similar characters, there is a significant transfer in learning between the two characters. This

supports the idea of subroutines being involved in character production, rather than learning a distinct motor program for each character.

An alternative to PCA/ICA decomposition of handwriting is proposed by (Ramsay, 2000), where they use spline fitting to decompose handwriting trajectories into derivative functions. These functions are assumed to be active over the whole handwriting sample, and the samples are linearly interpolated, and registered prior to analysis, similarly to fixed-dimension techniques such as PCA/ICA. The argument is that any physical system must have derivatives due to Newton's third law, and so this is a logical way in which to decompose movements. It would be interesting to examine whether primitives could be extracted with derivatives, or potentially a set of first derivative primitives, and a set of second, etc. An extension to this thesis could be to include a derivative based system in the output, and have the primitives parameterise the system.

2.4 Probabilistic time series models

If biologically produced movement data are made up of primitives, then a time series analysis of such data should take into account the modular nature of their generation. A common probabilistic framework for modelling time series data with hidden underlying structure is the Hidden Markov Model (HMM). This model, and a specialised subset is discussed further in Chapter 4. HMM models have been used extensively in speech recognition, mostly as language models, which provide a prior for which characters or words are likely to come next, given the current context. Such an informed prior can help recognition tasks enormously. A tutorial on HMMs and their uses, focussing on speech recognition can be found in (Rabiner, 1989).

Also in the field of speech, (Roweis and Alwan, 1997) uses a constrained HMM to model articulatory information in the hidden state. By constraining the hidden state transitions to be smooth, the transformation from spectra to articulation is based upon multiple time steps. The smooth underlying articulation trajectory could then be used for recognition purposes.

(Vinciarelli and Bengio, 2002) use an HMM system for character recognition. A sliding window scans across the image containing the characters, providing a feature vector. PCA/ICA is used to de-correlate the data, before using continuous density HMMs to model the transitions between feature vectors within a character in a word. This system works well, and demonstrates the flexibility of HMMs, as they are not modelling time transitions here, rather than dependencies between adjacent parts of

the image.

To create a handwriting model based upon motor primitives, a system with multiple concurrent modules is required. This is similar to the piano model (Cemgil et al., 2005), or the atomic decomposition model (Lewicki and Sejnowski, 1999; Smith and Lewicki, 2005), which are both based upon offset, overlapping functions. A probabilistic framework for modelling such situations exists as a constrained version of an HMM. Assuming the hidden state is made up of several independent factors, the transition matrix can be split up, leading to a Factorial Hidden Markov Model (fHMM). This subset of an HMM is discussed in detail in (Ghahramani and Jordan, 1997). In its exact form, it is identical to a very large HMM, with a specialised transition matrix. However the factorial nature of the hidden state allows a special variational method to be used during inference. The main drawback of HMMs is the extent of the computational requirements for inference. Using a variational approximation such as the one proposed in (Ghahramani and Jordan, 1997) greatly improves the speed. The fHMM model is used as the basis for the low level primitive model in this project, and more detail on the model and its implementation can be found in Chapter 4.

2.5 Placement and contrasts

The approach of this thesis is similar in many ways to Lewicki's atomic signal decomposition. The two methods are discussed and contrasted in further detail in Section 8.5.2.3. Indeed it could be argued that this thesis treats movements more like sounds, as there are no dynamical feedback systems being modelled.

A fixed function approximation of biological primitives was chosen as an attempt to model the variation in movements based upon an assumption of timing noise being present in the brain. Clearly there will be other sources of variation, such as that arising from dynamical systems. This is modelled simply as Gaussian output covariance, and an extension to this work would be to include a dynamical system, such as the spring system proposed by Hinton for the purposes of character production.

The offset function model was chosen here because of the assumption of asynchrony and massively parallel operation of the brain. If the brain organises movement in a parallel fashion, it can be assumed that there will be much timing noise present in the sequencing stage. This simple noise will then give rise to complex variation in the output due to the superposition of the many primitives. This thesis attempts to explain the variation in handwriting in terms of timing noise within such a model. No dynam-

ical system was explored so as to keep the primitives as simple as possible within the offset model, and to examine exactly the extent that timing noise gives rise to typical handwriting variation.

The timing code of the primitives may well be mirrored in the spiking activity of biological brains, a similar hypothesis to Lewicki's spikegram as a representation of cochlear encoding. In that sense the primitives here can be thought of as biologically inspired, however, without a model of the physical system of the hand and arm, our primitives remain in 'pen-space'. This is really an engineering compromise, as implementing such a model would further complicate the inference procedure. This thesis is motivated by biological studies into primitives, and hopefully provides a method in which biological primitives may be studied, and an insight into how the brain could be planning movement. Engineering motivations also exist, such as the possibility of using primitives for robotic control or handwriting representation. The assumption is that an accurate model of the biological primitives will also be a very good model for robotic control, or in the very least, a good model for perception of biological movements.

Essentially this thesis attempts to explain the variation in handwriting by finding hidden structure in handwriting data noise, where the types of structure of this noise are based upon biologically inspired concepts of motor primitives.

Chapter 3

Assumptions and model overview

The biological evidence for motor primitives has been discussed in Chapter 2, where a broad spectrum of evidence indicating modularity of the motor system was presented. This chapter formalises how we define primitives, and how we base a model of handwriting upon the concept of biological motor primitives.

3.1 Biological primitives

Chapter 2 discussed various studies about motor primitives in biology. The central idea behind motor primitives is a possible simplification of the motor planning problem. This problem is that biological (and indeed robotic) bodies have many joints, and muscles, which must be controlled in a coherent manner to perform a physical task. The brain must output a spectrum of muscle activation commands, given a large amount of sensory input sampling the state of the world, including the state of the body being controlled. This sensory information is very noisy, and incomplete, whilst the world itself is very variable and even the body being controlled is likely to have a degree of unpredictability, such as muscle tiredness, muscle and limb growth, clothing, and so on. Furthermore, the delay between muscle activation and sensory feedback is extremely long, at least in terms of traditional control theory.

Despite all these difficulties, which become apparent when trying to control robotic movement in real world environments, biological organisms perform movement control tasks reliably and repeatably, and are also able to adapt quickly to novel environments. The idea of motor primitives constitutes a proposed method by which the brain may be achieving these goals. A motor primitive can be thought of as a subblock, or subroutine of motion, which controls a segment of the spectrum of muscle

activations to perform a movement. The brain then has to fit these motor primitives together in a coherent fashion, rather than calculate the appropriate levels of muscle activation.

3.2 Modelling primitives

The exact nature of a motor primitive is unknown. Various possible definitions exist, for instance a motor primitive can be thought of as muscle activations which generate an end-point force field (Bizzi et al., 1995; d'Avella and Bizzi, 2005; d'Avella et al., 2003; Bizzi et al., 2002), dynamical systems defining attractor dynamics (Ijspeert et al., 2003; Schaal et al., 2004; Amit and Mataric, 2002), modular models of limb dynamics (Wolpert et al., 2001; Wolpert and Kawato, 1998), and independent component decomposition (Matarić, 2004).

Primitives are unlikely to be time-indexed, fixed dimensional constructs such as those defined by an ICA analysis of data. For the brain to control such primitives, it would need some universal clock prompting the swapping of one set of primitives for another at each time step. Assuming that the brain operates as an asynchronous machine, it is more likely that the primitives are time-extended blocks that define a section of muscle activations over this time-period.

The range and adaptability of biological movement suggests that the manner in which the primitives can be combined must be fairly flexible. Instead of having a sequence of primitives, it is more likely that they can be combined together, such as in superposition. This theory is suggested in (Bizzi et al., 1995), where the concurrent activation of primitives leads to roughly linearly superimposed force fields. It is therefore assumed that primitives may be superimposed with non-concurrent activation.

The length of a primitive is unknown, but is assumed to be shorter than the length of the task being performed, and could be different for different primitives. It is also assumed that once a primitive has been initiated, it must *play out* its entire length, as suggested by (Kargo and Giszter, 2000). Therefore instead of switching off a primitive mid-length, the brain can only modify a movement mid-length by superimposing another primitive.

3.3 Assumptions

The data being modelled in this project is generated from handwriting samples. The data is captured as pen-tip trajectories. The primitives are modelled in the same space as the pen-tip data, rather than trying to create a model of the physical system of the hand and arm where primitives would consist of high dimensional muscle activations.

Primitives are assumed to be locally fixed, time-extended arbitrary functions in pen trajectory space. These functions are superimposed with specific offsets to create the trajectory giving rise to a particular character.

3.4 Piano Model

The model described above has been called a *Piano Model* (also called Piano roll model (Cemgil et al., 2005)), due to its similarities with a simplistic model of a Piano being played. In this model, the score contains the timing information dictating when the notes should be played, and the music is a superposition of the waveforms over time. The variation in the sound produced is due to variations in the timing of the notes, rather than variations in the waveforms of the notes themselves. Such a model can be formalised as a fixed function offset model,

$$\mathbf{y}(t) = \sum_{i=1}^{I} \sum_{j=1}^{J_i} \alpha_{i,j} \mathbf{w}_i(t - \tau_{i,j}) \ , \tag{3.1}$$

where $\mathbf{y}(t)$ is the observable output, which in the case of a piano is the music as heard by a listener. In this formulation, there are I fixed functions, $\mathbf{w}_i(t)$, which occur at specific offsets, $\tau_{i,j}$, and with specific scaling factors, $\alpha_{i,j}$. Each function occurs J_i times within the sample $\mathbf{y}(t)$.

We define our primitives in the space of pen trajectories as the functions $\mathbf{w}_i(t)$. There are no constraints on where the primitives might occur in the characters, and there are no constraints on the shapes of the primitives. To simplify the problem, the scaling factors, $\alpha_{i,j}$ are set to unity, thus removing them from the model.

To learn the shapes of the primitives, and the timings of the primitives in a data sample, we assume that there is a degree of noise in both of these quantities. This variation can be modelled in a probabilistic framework.

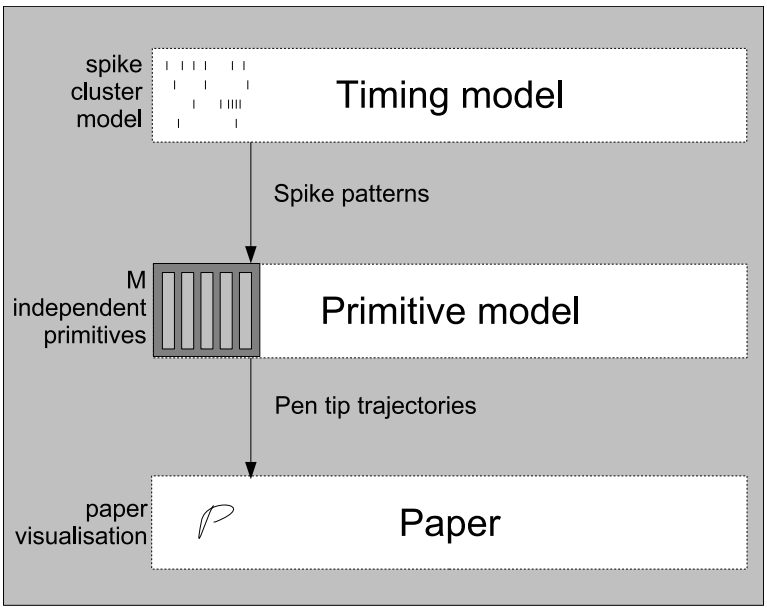


Figure 3.1: Block diagram of the model, giving an overview of generative dependency flow, and data representation in different parts of the model.

3.5 Probabilistic model overview

The probabilistic model can be split up into two sections. The Piano Model described above is implemented in the lower level model, which captures the shape and timing of the primitives in a non-character-specific dataset. A higher level model represents the timing distribution of primitives in a character-specific dataset. An overview of the model organisation can be gained from Figure 3.1, showing the layered structure of the model, and the generative dependency flow. The data is represented in different ways on different levels of the model, as will be discussed in more detail later.

The lower level model is based on a Factorial Hidden Markov Model and is described in detail in Chapter 4. This lower level model is referred to as the *primitive model*. The primitive model represents the shape of the primitives as parameters of the model, and learns a distribution over the states of the primitives for a specific pen trajectory sample. For a specific dataset, the primitive model can learn a set of primitive functions, and use them to reconstruct the data as best they can. During this reconstruction, the model infers a distribution over the onset probabilities of the primitives. The mode of this distribution gives rise to a spike timing representation of where the primitives are most likely to be active in any given sample from the dataset. This spike timing representation can then be used in the higher level model, to generalise across character samples.

The higher level model is based on a set of Hidden Markov Models, as detailed in Chapter 6, and referred to as the *timing model*. The timing model takes the spike timing information generated by the primitive model as the data to be modelled. It is assumed that the spikes come from hidden Gaussian components, which can produce at most one spike, and are not always present. The timing model captures variation in the spike timing data, whilst providing an adequate template to produce a coherent character sample.

The two models are coupled together by use of a *spike prior*, which is the prior probability of primitive onset at any one time point, as determined by the timing model. The method of coupling is detailed in Chapter 7. Intuitively, the primitive model learns what the primitives look like, and where they are active in all character samples. Given the timing of the primitives, the timing model then determines the likely timing of primitives based upon all samples of a particular character. The prior probability of primitive onset is then used to improve the primitive model's inference of primitive shape and timing.

The transformation of the trajectory data into spike timing data via the primitive model is a strong compression of the data, and allows efficient higher level models that generalise across character samples much faster than could be done using the untransformed trajectory data. This compression is the key advantage of using primitives to represent the data, and reflects the advantage of motor primitives as a biological movement planning tool in the brain.

3.6 Definitions

There are multiple parameter matrices in the model, and state variables. To clarify what they are representing, an overview is presented here. There are other parameters, such as Markov transition matrices and covariance matrices that are introduced where appropriate in the model descriptions.

$\mathbf{Y}_{d,t}$ Characters such as those seen in Figure 3.2 were collected using a digitisation tablet. $\mathbf{Y}$ represents the normalised first differential of the data, which can be seen in Figure 3.3. The differentiation was done numerically, such that

$$\mathbf{Y}'_t = \mathbf{Y}_t - \mathbf{Y}_{t-1} \,. \tag{3.2}$$

As the dataset is normalised, there is no need to divide by a δt. The dataset normalisation ensures that the variance over time samples of each dimension of

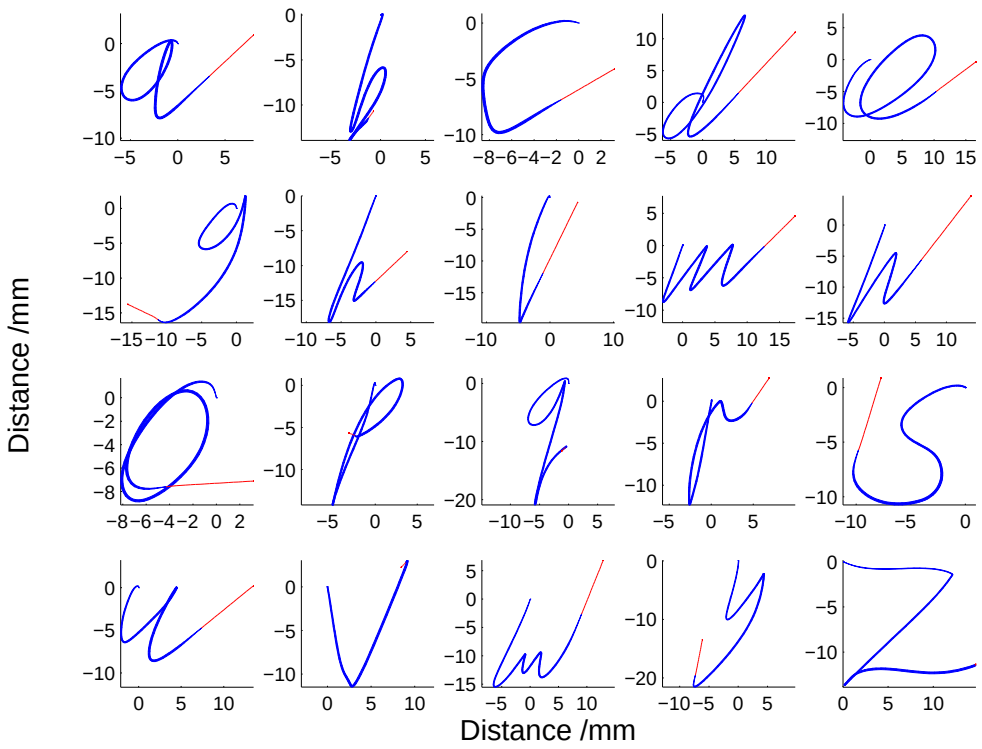


Figure 3.2: Examples of the characters collected from the digitisation tablet. Only characters without *PEN-UP* segments are currently considered to simplify the model. Blue sections represent pen movements in contact with the paper, and red sections represent pen movements off the paper. The thickness of the line represents pen pressure (pen tip force).

the data is unity. This normalisation makes primitives from different datasets more comparable. For more details of how the data is collected, and pre-treated, see Section 5.1. $\mathbf{Y}$ is a $D \times T$ dimensional matrix of real numbers, where D is the dimensionality of the data, and T is the length of the sample.

$\mathbf{W}^m_{d,k}$ The shapes of the primitives are stored as parameters in the matrices $\mathbf{W}^m$, such as those seen in Figure 3.4, where m enumerates the primitive. $\mathbf{W}^m$ is a $D \times K^m$ matrix of real numbers, where K^m is the length of primitive m. The number of states in a primitive is referred to as the *length* of the primitive as the primitives are constrained to transition monotonically through their states, as detailed in Section 4.4.

$\mathbf{S}^m_t$ The state of each primitive is represented by the discrete state variable, $\mathbf{S}^m_t$, as seen in Figure 3.5. The state of a primitive m at time t determines whether or not it is active at that time point, and if it is active, the state defines a point along the length of the primitive. If the primitive is K^m states long, $\mathbf{S}^m_t$ can take $K^m + 1$

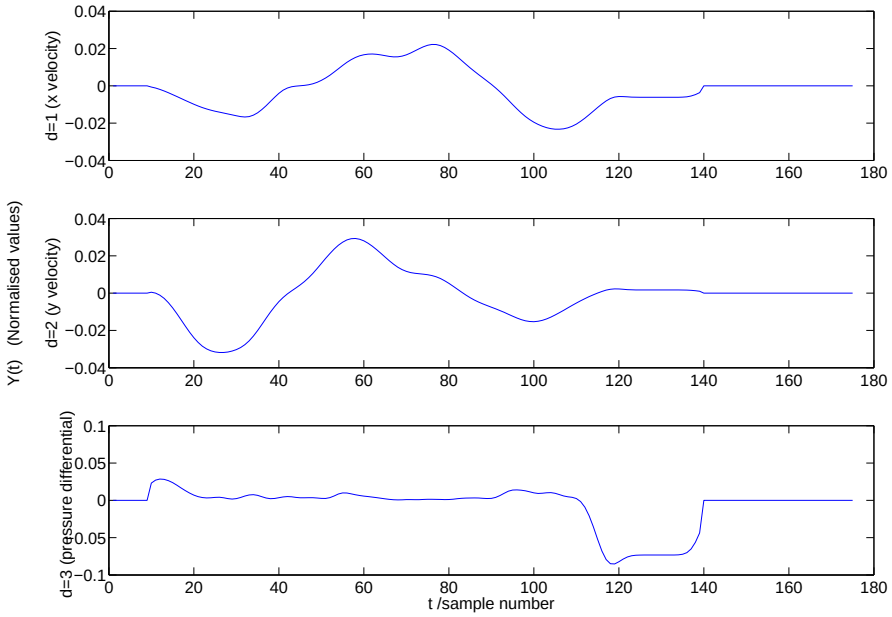


Figure 3.3: $D = 3$ dimensions of **Y** plotted as a function of t. **Y** contains the observable data, as seen here. The observable data is the first differential of the character samples seen in Figure 3.2. The sampling rate is 200Hz, so sample number multiplied by 0.005 represents time in seconds.

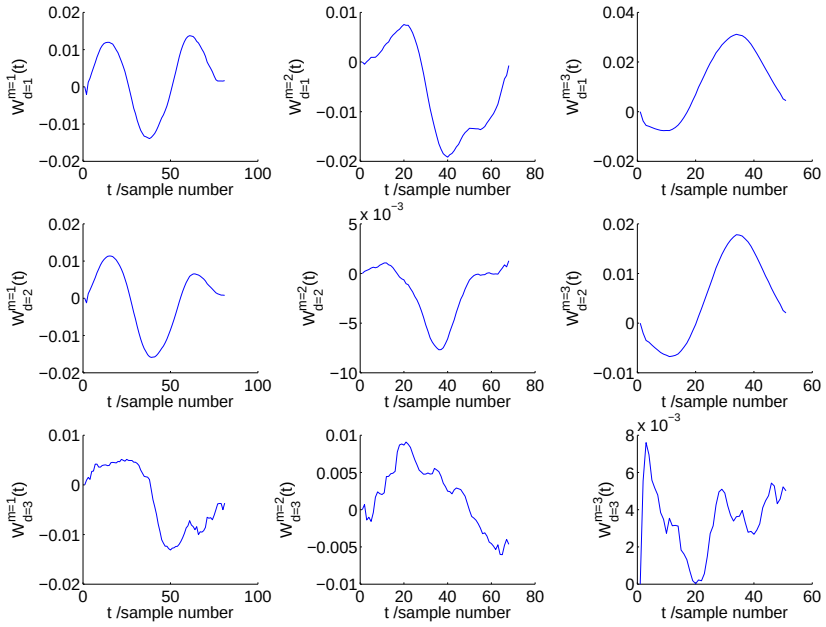


Figure 3.4: $D = 3$ dimensions of $\mathbf{W}^m$ plotted as a function of t. $\mathbf{W}^m$ contain the shapes of the primitives. A sample of 3 primitives can be seen here, in the 3 columns of this figure, with each row showing a separate dimension.

discrete states. $\mathbf{S}_t^m \in \{0, 1, ..., K^m\}$

$\boldsymbol{\lambda}_t^m$ The primitive onset times for a specific character sample can be taken from the mode of the posterior distribution over $\mathbf{S}_t^m$. These onset times are then modelled using the binary variable $\boldsymbol{\lambda}_t^m$, which can be expressed as a set of spike times, as can be seen in Figure 3.6. $\boldsymbol{\lambda}_t^m \in \{0, 1\}$.

$\mathbf{G}_i^m$ Generalising across multiple character samples, the spikes are assumed to come from particular generative Gaussian components. The component giving rise to the ordered spike i from a particular sample is represented by the discrete state variable $\mathbf{G}_i^m$, as can be seen in Figure 3.7. $\mathbf{G}_i^m \in \{1, 2, ..., K\}$, where K is the number of generative components giving rise to the spikes for primitive m. This describes a mixture of Gaussians model, however dependencies can be introduced between the components, creating more complex models. Generalisation across different samples of characters, and the various modelling possibilities is discussed in Chapter 6.

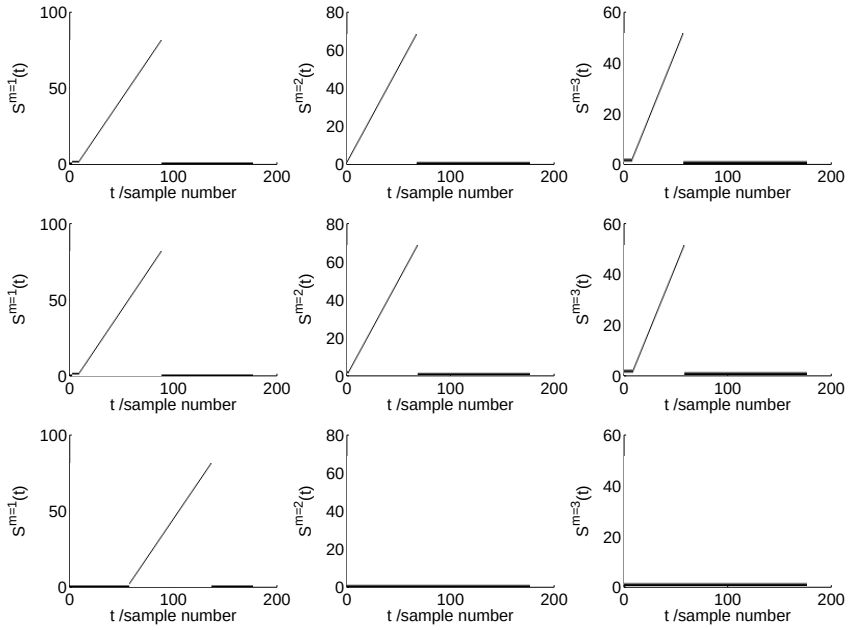


Figure 3.5: $\mathbf{S}_t^m$ is shown here as a function of time t. $\mathbf{S}^m(t)$ contains the state of the primitives over time. Here the monotonicity of the state transitions can be seen. Three primitives from three samples are shown in this figure. Each column shows a primitive, and each row a sample.

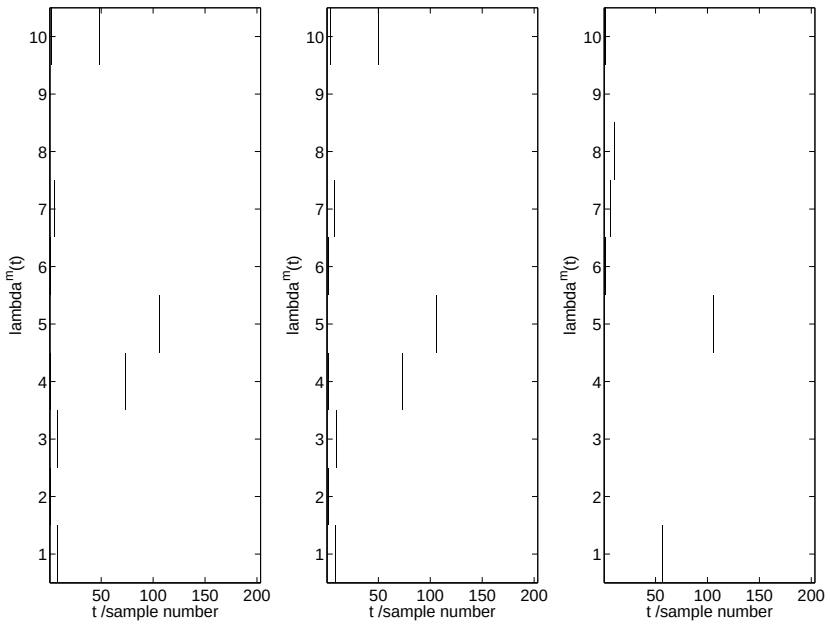


Figure 3.6: $\boldsymbol{\lambda}^m(t)$ is a binary variable dictating the onset positions of the primitives. This figure shows $\boldsymbol{\lambda}^m(t)$ for 10 primitives in 3 samples, demonstrating the natural representation in terms of spike times.

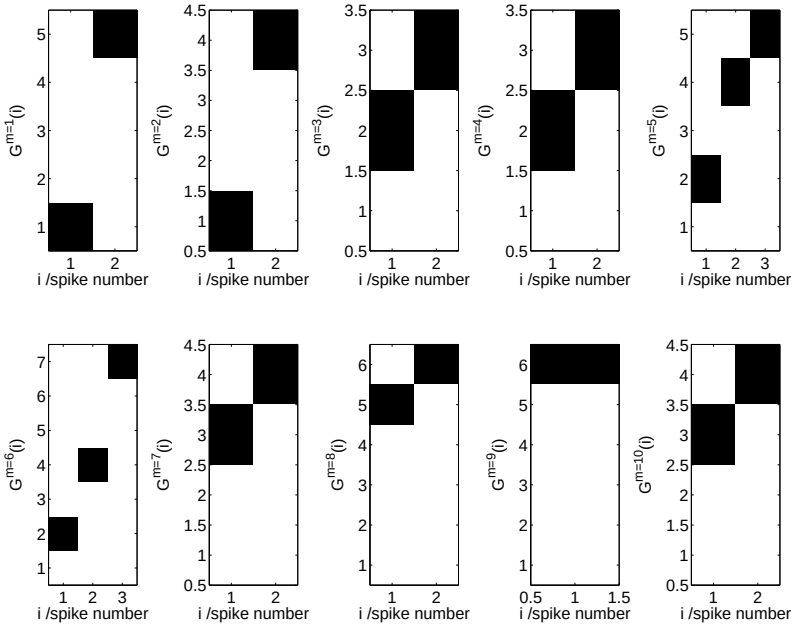


Figure 3.7: $\mathbf{G}^m(i)$ depicts a hidden state that is indexed by spike count, as seen here. The hidden state can take values $\{1..K\}$, and determines which Gaussian component is responsible for the associated spike i.

3.7 Operational overview

The detailed operation of the subsections of the model are described in the following chapters, along with inference methods to learn the parameters. It will help to keep in mind an overview of how the whole model operates whilst generating a character, as described here.

Given a character class, there are M parameterised Markov chains $\mathbf{G}_i^m$. These are constrained such that each state can be visited at most once, and there is a termination state at the end of the chain. Imagine a machine that enters one of a string of states, but can only travel in one direction along the string. It is possible to skip one or many states, but at the end of the string, the machine stops. (In this analogy there are M strings, each with its own machine.) Upon entering a state along the string, the machine will emit exactly one spike, the timing of which is sampled from a Gaussian distribution. So now, on a separate piece of string, there are L spikes, where L is the number of states that the machine entered on its journey. The second spike string (or *train* to use a more common term) is indexed not by discrete state, but by time. This analogy describes the operation of the timing model.

The primitive model can be thought of as M machines able to enter states on *loops* of string. The size of each loop is referred to as the length of the primitive. Initially all M machines are in a rest state on the loops (think of a knot perhaps), where they temporarily remain. Each loop of string is associated with one of the spike trains. As time progresses, the appearance of a spike causes the associated machine sitting in its rest state on the loop to start progressing around the loop, at a constant speed, until the rest state is once again reached, and the machine must wait until the next spike. As the machine steps around the loop, it outputs a varying weight which is associated with each state around the loop. (In the rest state, this weight is zero.) Therefore, at any one time point, there are M machines on M loops, giving M weights. The sum of these weights parameterises a Gaussian output distribution, describing the pen tip velocities at each time point. Therefore sampling from this distribution at each time point gives the complete velocity trajectory of the pen for a single character.

It should be noted at this stage that the timing model machine is indexed by spike count, whereas the primitive model machine is indexed by time. This complicates the inference procedure as we shall see later on, and necessitates a mixture of Gaussians distribution to approximate the timing model for coupling purposes.

Chapter 4

Factorial Hidden Markov Model

In Chapter 3, a simple deterministic model is described, the Piano Model, where primitives are defined as fixed arbitrary functions. These functions are superimposed with time offsets, to create the output signal giving rise to the data. The data we are concerned with is handwriting trajectory data, although the model could be applied to any repeatable biologically generated data. This model is the motivation behind the complete probabilistic model described in Section 3.5. The complete model was described in a top-down fashion, to give an idea of the generative procedure. It is possible to decouple the model into two levels, and run these stages separately. The two stages of the model are named the Primitive model, and the Timing model. The Primitive model can be thought of as the output, or low-level stage of the model, and the Timing model as a higher level controlling part of the model. The Primitive model is learnt using a Factorial Hidden Markov Model framework, as described in this chapter. The Timing model is learnt using several Hidden Markov Models, as described in Chapter 6.

Some of the work in this chapter is published in (Williams et al., 2006), with more detail here. In this chapter, we look at the probabilistic implementation of the Piano Model, and the correspondences and differences of the two models. We then go through the details of the model and the method of inference, and finally the constraints and modifications to the algorithm.

4.1 Probabilistic Implementation of the Piano Model

In Chapter 3 the motivating model was described, which was referred to as the Piano Model. In this model, a set of M primitives are superimposed with various offsets. The primitives in this model are defined as fixed, time extended functions, of varying

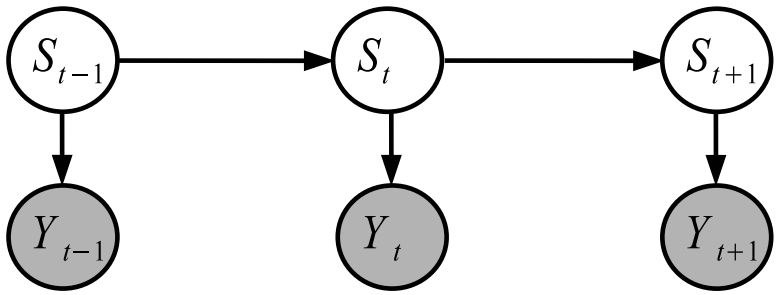


Figure 4.1: Graphical model of a Hidden Markov Model. $\mathbf{S}_t$ is a discrete random variable representing the hidden state of the model at time t. $\mathbf{Y}_t$ can be either a discrete or, as in this case, a continuous random variable representing the observable data at time t. Horizontal dependencies run across time, and vertical dependencies show the dependency of the observable output upon the hidden state.

lengths. It is difficult to learn such a model, as the shape, number, or length of these primitives are not known, nor the timing of their offsets. Assuming there is a degree of noise in the timing of the offsets, and a degree of noise in the output of the model, a probabilistic framework was chosen to learn the parameters of the model from data.

A Factorial Hidden Markov model was chosen because of an implicit assumption of independence between primitives, and the need to infer the shape and timing of the primitives in the presence of noise. From an implementation point of view, the fHMM is a convenient modelling framework due to the presence of efficient learning algorithms and the possibility of using faster variational approximations to full posterior inference. The Piano Model can be thought of as the motivating model, and the fHMM as a framework for this model. The fHMM class of models are more general than the Piano Model, and to learn appropriate primitives, several constraints were imposed on the fHMM and are discussed later in this chapter.

4.1.1 Factorial Hidden States

A commonly used framework to model time series data generated from underlying processes is the Hidden Markov Model, (HMM) (Rabiner, 1989). A graphical representation of such a model can be seen in Figure 4.1. In these models, a hidden state variable, $\mathbf{S}_t$ controls the distribution of the output, or observable state variable, $\mathbf{Y}_t$. In the case of handwriting, therefore, the observable variable would model the pen trajectory data, and the hidden states would somehow model the states of the primitives. This would be a very difficult problem if the primitives are allowed to interact with each other, and influence the behaviour of others during operation. In fact, if the prim-

itives are assumed to be interdependent, the benefits of having discrete pre-determined subroutines is questionable, as the task for the brain to then piece them together in such a manner as to create coherent, predictable movements is much more difficult. There is evidence from biological studies such as (d'Avella and Bizzi, 1998) that suggests biological movement can be linearly decomposed into primitives, and (Kargo and Giszter, 2000) suggests that the primitives are uninterruptable time extended blocks. It is assumed therefore that the primitives should be largely independent of each other, given the observable data. In Markov Models, if the hidden state is composed of several independently evolving factors, it is possible to decompose the model into factors, as seen in Figure 4.2. This is now a Factorial Hidden Markov Model, (fHMM). A thorough treatment of this class of model, and the various options for learning the parameters of such models can be found in (Ghahramani and Jordan, 1997), and was briefly discussed in Chapter 2. In summary, the most efficient learning method for this class of model is that of structured variational Expectation-Maximisation (EM), where the output dependent coupling between factors, during learning, is replaced by a single responsibility factor that is updated so as to minimise the KL divergence between the approximating distribution and the true distribution.

Referring to Figure 4.2, where the nodes of the graph denote states and arrows denote dependencies, the observable states are shown at the bottom of the figure, and the hidden states are shown above. For our purposes, the hidden states are modelling the states of the primitives. $\mathbf{S}_t^m$ represents the state of primitive m at time t. As this is a probabilistic model, these states are modelled with discrete probability distributions over all their possible values. The estimated probability distribution over the hidden states is $Q(\mathbf{S}_t^m)$. Each factor can take one of K states, $\mathbf{S}_t^m \in \{1,2,...,K\}$, where, at a particular state k, an output contribution of $\mathbf{W}_k^m$ is observed. Therefore, assigning a primitive m to a factor, the length of the primitive is defined by the number of states K in the factor. As we do not wish to constrain the length of the primitives, we must therefore introduce variable length factors. This is described in Section 4.4.2.1. The length of factor m is now a function of m, i.e. K^m. The timing offsets in the Piano Model are modelled therefore by the state trajectories in the factors of the fHMM. The shapes of the primitives are modelled by the consecutive output contributions, $\mathbf{W}_{\{1..K\}}^m$. These are depicted by the vertical dependencies in Figure 4.2. The one aspect of the Piano Model that is not captured with a fHMM is the scaling of the primitives, as there is only a single multivariate mean contribution associated with each hidden state. A higher level model would be needed to include such a scaling factor, as a concept of

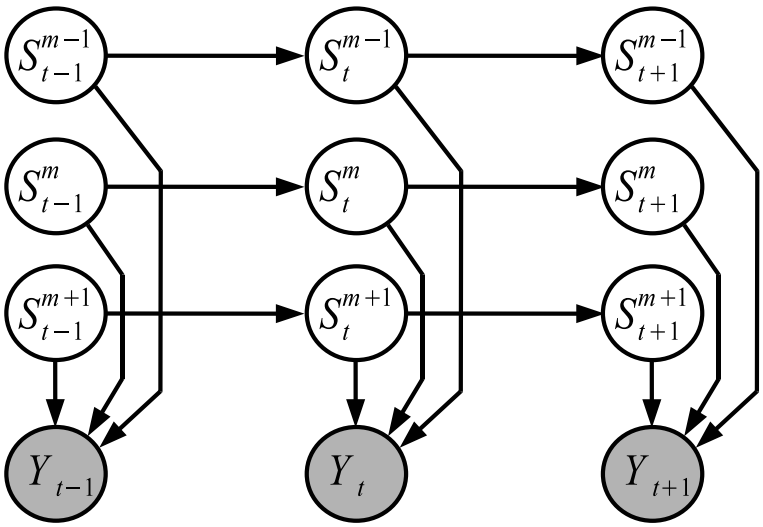


Figure 4.2: Graphical model of a Factorial Hidden Markov Model. The difference to the HMM shown in Figure 4.1 is that the hidden state is split up into M independent factors. $\mathbf{S}^m_t$ is a discrete random variable representing the state of factor m at time t, which in terms of the primitive model described here indicates a length along primitive m. Horizontal dependencies run across time, and vertical dependencies represent contributions from all primitives to the observable output at a particular time step. The fHMM is a special case of an HMM with a large hidden state where the transition matrix is constrained in such a way as to produce M independent factors. The benefit of reformulating the model in this manner is that there are variational inference procedures that efficiently exploit the independence of the factors.

primitive activation enumeration would be needed. It is assumed that a scaling factor is not absolutely necessary as two factors could be used to model the same primitive on different scales if required by the data.

An advantage of using a fHMM in our case is that the inference can be simplified by using a variational approximation to the posterior distribution over the hidden states. This will be addressed in Section 4.3.1, and a justification for such an approximation along with inference speed benefits can also be found in (Ghahramani and Jordan, 1997).

4.1.2 Markov Properties

The graphical representation of a fHMM can be seen in Figure 4.2. This is a generalised model capable of representing time series data generated from several underlying independent and hidden discrete processes, which have Markovian dynamics. This means that each hidden state $\mathbf{S}^m_{t=\mathrm{t}}$ is independent of past states $\mathbf{S}^m_{t<(\mathrm{t}\text{-}1)}$, given the

immediately preceding state $\mathbf{S}^m_{t=(\text{t-1})}$. Formally,

$$P(\{\mathbf{S}^m\}) = P(\mathbf{S}^m_1)\prod_{t=2}^{T} P(\mathbf{S}^m_t \,|\, \mathbf{S}^m_{t-1}) \; . \tag{4.1}$$

The factors are also independent, meaning that

$$P(\{\mathbf{S}\}) = \prod_m P(\{\mathbf{S}^m\}) \; . \tag{4.2}$$

In the Piano Model, it is assumed that the pen output is made up of several independent processes, or primitives. In a fHMM representation of the Piano Model therefore, the hidden states represent the states of the primitives, with each factor representing one primitive. More specifically, the hidden state of factor $\mathbf{S}^m_t$ represents a length along primitive m, at time t, therefore the set of hidden factors determine the temporal activity of the primitives. The observable states represent the pen tip data.

4.2 Model Definition

The graphical model for a Factorial Hidden Markov Model can be seen in Figure 4.2. Here it will be described how the model operates, and how it models the data. Figure 4.3 shows a sample of the data being modelled, both in 'pen space' and velocity space. The velocity data is defined therefore as the observable data. The observable data is the first differential of pen position and pen tip pressure, and in our case is 3 dimensional, although I shall refer to D dimensions of the observable data for ease of generalisation. In Figure 4.4, we can see there is a large degree of variation in the datasets. We will assume that this variance comes both from noise in the timing of the primitives, and from the output of the model. We assume that the noise on the output of the model is Gaussian for modelling convenience. $\mathbf{Y}_t$ is the random variable representing the observable output, so if using the $D = 3$ dimensional data, as described in Section 5.1, the pen tip velocity $\mathbf{Y}_t$ is defined as

$$\mathbf{Y}_t = (\dot{x}, \dot{y}, \dot{z})^T \tag{4.3}$$

$$\mathbf{Y}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \mathbf{C}) \; , \tag{4.4}$$

where $\mathcal{N}$ denotes a Gaussian distribution. $\mathbf{C}$ is a $D \times D$ parameter matrix of output covariance, and

$$\boldsymbol{\mu}_t = \sum_{m=1}^{M} \mathbf{W}^m_{\mathbf{S}^m_t} \; . \tag{4.5}$$

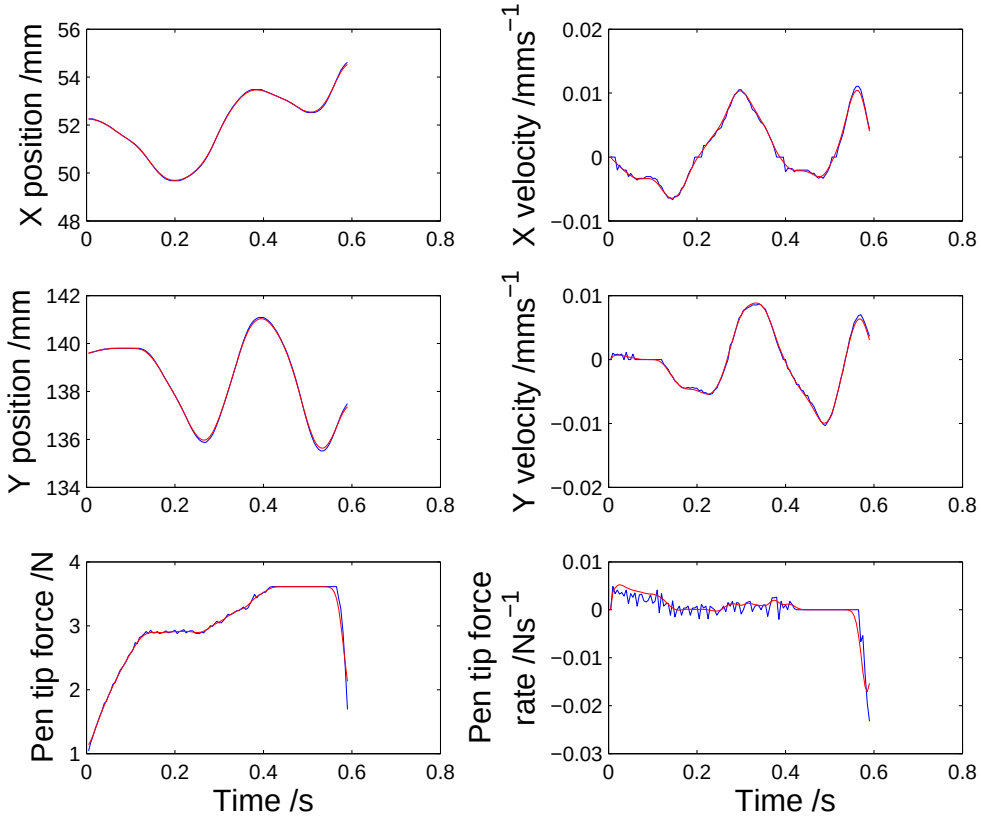


Figure 4.3: Single character sample shown in position (left column) and velocity (right column) space. Data is overlaid showing the effect of Gaussian smoothing on the data as a pre-treatment. The blue plot shows the raw, untreated data, whilst the overlain red plot shows the Gaussian smoothed data, with $\sigma = 0.01s$. This is sufficient to remove most of the noise in the data, which is amplified by the differentiation, as can be seen on the right. Gaussian smoothing was implemented by simply convolving the signal with the appropriate Gaussian function.

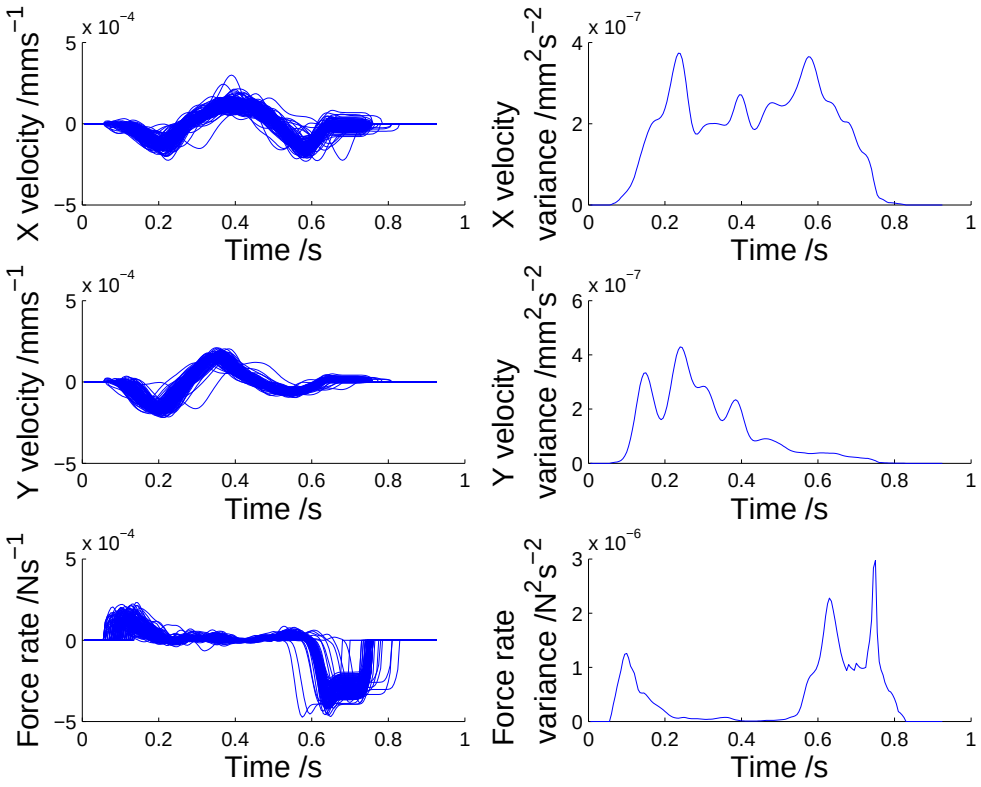


Figure 4.4: Left column shows an overlay plot of 131 character samples of the letter *'p'* in velocity space, demonstrating the variation in the data. The variance profile of the dataset is shown on the right. The variance profile is specific to a particular character class and writer, where the implicit hypothesis of the primitive model is that part or all of this variation, can be modelled by primitive timing noise, at least for a single writer.

So $\mathbf{W}^m_{\mathbf{S}^m_t}$ is the output contribution of the factor m, and $\mathbf{S}^m_t$ is the state of factor m at time t. In other words, the output at any time point is a sum of the factor contributions, each of which is determined by the state of that factor at that time. This means the probability density function for the model output, at time t, is defined as

$$P(\mathbf{Y}_t \,|\, \mathbf{S}_t) = |\mathbf{C}|^{-1/2}(2\pi)^{-D/2}\exp\{\frac{1}{2}(\mathbf{Y}_t - \boldsymbol{\mu}_t)^T\mathbf{C}^{-1}(\mathbf{Y}_t - \boldsymbol{\mu}_t)\} \ . \tag{4.6}$$

This multivariate Gaussian distribution is parameterised by the $\boldsymbol{\mu}_t$ and $\mathbf{C}$ variables, where $\boldsymbol{\mu}_t$ represents the expected pen-tip velocity, and $\mathbf{C}$ represents the stationary covariance of the pen-tip velocity. $\boldsymbol{\mu}_t$ is dependent upon the hidden states of the model at any one time point, as defined above. Referring to Figure 4.2 again, the parameters $\mathbf{W}^m$ or output contributions are represented by the vertical dependencies leading to the observable output, and originating from M hidden factors. The factors are conditionally independent of each other given the output, which is shown by using separate horizontal dependencies for each factor. These horizontal dependencies are parameterised by the M transition matrices $\mathbf{P}^m$.

The M factors have standard Markov state transition assumptions, with a set of parameters defining the hidden state priors of the model. Initially, at $t = 1$, the prior over the hidden states is defined by the parameter $\boldsymbol{\pi}$,

$$P(\mathbf{S}^m_1 = i) = \boldsymbol{\pi}^m_i \ , \tag{4.7}$$

where $\boldsymbol{\pi}^m$ is a vector length K^m. As the factors are independent, the complete state prior distribution at time $t = 1$ can be written as,

$$P(\mathbf{S}_1) = \prod_{m=1}^{M} \boldsymbol{\pi}^m \ . \tag{4.8}$$

The subsequent hidden state prior distributions are defined by the transition matrices $\mathbf{P}^m$,

$$\begin{aligned} P(\mathbf{S}^m_t = i \,|\, \mathbf{S}^m_{t-1} = j) &= \mathbf{P}^m_{i,j} \\ P(\mathbf{S}_t \,|\, \mathbf{S}_{t-1}) &= \prod_{m=1}^{M} \mathbf{P}^m \ , \end{aligned} \tag{4.9}$$

where $\mathbf{P}^m$ is a $K^m \times K^m$ matrix containing the hidden state transition priors.

The factors are independent, as described in Section 4.1.1. This independence means that the probability of factor 1 to be in state x, whilst factor 2 is in state y, and factor 3 is in state z, and so on, is the product of the separate marginal probabilities.

The joint probability distribution can therefore be factorised as

$$P(\{\mathbf{Y}_t, \mathbf{S}_t\}) = P(\mathbf{S}_1)P(\mathbf{Y}_1|\mathbf{S}_1)\prod_{t=2}^{T} P(\mathbf{S}_t|\mathbf{S}_{t-1})P(\mathbf{Y}_t|\mathbf{S}_t) \tag{4.10}$$

$$= \prod_{m=1}^{M} \boldsymbol{\pi}^m P(\mathbf{Y}_1|\mathbf{S}_1)\prod_{t=2}^{T}\prod_{m=1}^{M} \mathbf{P}^m P(\mathbf{Y}_t|\mathbf{S}_t)\ . \tag{4.11}$$

4.2.1 Free parameters

To summarise, there are four sets of parameters to be learnt in this model, each having many dimensions, as listed here

$$\mathbf{W} \ : \ D \times \sum_{m}^{M} K^m \tag{4.12}$$

$$\mathbf{C} \ : \ D \times (D+1)/2 \tag{4.13}$$

$$\boldsymbol{\pi} \ : \ \sum_{m}^{M} K^m \tag{4.14}$$

$$\mathbf{P} \ : \ \sum_{m}^{M} K^m \times K^m\ . \tag{4.15}$$

The output contributions of the factors, **W**; the stationary output covariance, **C**; the initial hidden state priors, $\boldsymbol{\pi}$; and the hidden state transition matrices, **P**. However, as we shall see in Section 4.4, most of the **P** parameters do not need to be learnt, and similarly, the $\boldsymbol{\pi}$ parameters are unnecessary due to assumptions about the primitives.

4.3 Inference

A common technique for learning parameters in Dynamical Bayesian Models (DBMs) such as the model introduced here, is Expectation Maximisation (EM), a two step iterative procedure, whereby in the E-step, the conditional posterior distribution over the model states, $P(\mathbf{S}\,|\,\mathbf{Y}, \theta)$, is inferred, then in the M-step, the parameters of the model, θ, are updated to their maximum-likelihood values, given the inferred state of the model. This procedure is guaranteed to converge to a local maximum of the likelihood function in the parameter search space. The main problem with the procedure is therefore ensuring that the local maximum found is in fact the global maximum, or at least close to it. This is dependent upon the initialisations of the parameters, 4.5, and the constraints upon the algorithm.

4.3.1 Expectation Step

Exact inference of the distribution over the hidden states, $P(\mathbf{S} \mid \mathbf{Y}, \theta)$, is a computationally intensive process, and for this reason there are a number of approximate inference solutions that have been explored. Faster solutions to the inference of $P(\mathbf{S} \mid \mathbf{Y}, \theta)$ involve introducing an approximate distribution, $Q(\mathbf{S})$, which corresponds to a simpler model structure. Several structural approximations to the fHMM are explored in (Ghahramani and Jordan, 1997). Here the structured variational approximation in which the factors are decoupled is adopted. In this approximate distribution, $Q(\mathbf{S})$, corresponds to multiple uncoupled Markov Chains, exploiting the efficient forward-backward algorithm. The aim of the variational inference is to minimise the KL divergence between this approximate distribution and the exact distribution $P(\mathbf{S})$. The approximate distribution simplifies the inference problem by treating the factors separately during the E-step and using a responsibility factor, $\mathbf{h}_t^m$, in place of the observable probabilities, $P(\mathbf{Y} \mid \mathbf{S})$ during the inference. To minimise the KL divergence, the responsibility factor $\mathbf{h}_t^m$ is updated thus,

$$\mathbf{h}_t^m = \exp\left\{ (\mathbf{W}^m)^T \mathbf{C}^{-1} \tilde{\mathbf{Y}}_t^m - \frac{1}{2} \triangle^m \right\} \tag{4.16}$$

$$\triangle_k^m \equiv ((\mathbf{W}_k^m)^T \mathbf{C}^{-1} \mathbf{W}_k^m) \tag{4.17}$$

$$\tilde{\mathbf{Y}}_t^m \equiv \mathbf{Y}_t - \sum_{l \neq m}^{M} \mathbf{W}^l Q(\mathbf{S}_t^l) \ , \tag{4.18}$$

where $\tilde{\mathbf{Y}}_t$ is the residual error. $\mathbf{h}_t$ is updated, as a function of $Q(\mathbf{S}_t)$, the expected hidden state distribution. However $Q(\mathbf{S}_t)$ is inferred by splitting the factors up into M standard Hidden Markov Models, and using $\mathbf{h}_t^m$ in place of $P(\mathbf{Y}_t \mid \mathbf{S}_t)$ for standard Baum-Welch inference techniques (Baum et al., 1970). Therefore the E-step becomes an iterative process, with $\mathbf{h}_t^m$ updated with each iteration i, and $m = i \mod M$. This iterative E-step is justified by the speed increase of treating the conditionally independent factors separately, allowing a forward-backward Baum-Welch algorithm to infer the hidden state distribution for each Markov Chain.

4.3.2 Likelihood calculation

The observable data likelihood is defined as,

$$\begin{aligned}\log P(\{\mathbf{Y}_t\}) &= \log \sum_{\{\mathbf{S}_t^m\}} P(\{\mathbf{S}_t^m, \mathbf{Y}_t\}) \\ &= \log \sum_{\{\mathbf{S}_t^m\}} Q(\{\mathbf{S}_t^m\}) \left[\frac{P(\{\mathbf{S}_t^m, \mathbf{Y}_t\})}{Q(\{\mathbf{S}_t^m\})}\right] \\ &\geq \sum_{\{\mathbf{S}_t^m\}} Q(\{\mathbf{S}_t^m\}) \log \left[\frac{P(\{\mathbf{S}_t^m, \mathbf{Y}_t\})}{Q(\{\mathbf{S}_t^m\})}\right] \end{aligned} \tag{4.19}$$

The inequality defines a lower bound on the likelihood, called the variational log likelihood (VLL), which is equivalent to the negative variational free energy.

$$VLL = \sum_{\{\mathbf{S}_t^m\}} Q(\{\mathbf{S}_t^m\}) \log P(\{\mathbf{S}_t^m, \mathbf{Y}_t\}) - \sum_{\{\mathbf{S}_t^m\}} Q(\{\mathbf{S}_t^m\}) \log Q(\{\mathbf{S}_t^m\}) \,. \tag{4.20}$$

A prior over the values of $\mathbf{W}$ was included in the model, to help with numerical instability issues when updating $\mathbf{W}$ parameters. This prior, and the derivation of the variational likelihood function can be found in Appendix 10.1. For ease of reference, here is the resultant regularised VLL

$$\begin{aligned} VLL_{reg} = & \sum_m \sum_k \gamma^m_{t=1,k} \log \boldsymbol{\pi}^m_k \\ & + \sum_m \sum_{t=2}^{T} \sum_{k,j} \xi^m_{t,k,j} \log \mathbf{P}^m_{k,j} \\ & - \frac{T}{2} \log(|\mathbf{C}|) - \frac{TD}{2} \log(2\pi) - \frac{1}{2} \sum_{t=1}^{T} \mathbf{Y}_t \mathbf{C}^{-1} \mathbf{Y}_t \\ & + \sum_{t=1}^{T} \sum_{m,k} \gamma^m_{t,k} \mathbf{Y}_t \mathbf{C}^{-1} \mathbf{W}^m_k \\ & - \frac{1}{2} \sum_{t=1}^{T} \sum_{m,n \neq m} \sum_{k=1}^{K^m} \sum_{j=1}^{K^n} \gamma^m_{t,k} \gamma^n_{t,j} \mathbf{W}^m_k \mathbf{C}^{-1} \mathbf{W}^n_j \\ & - \frac{1}{2} \sum_{t=1}^{T} \sum_m \sum_{k=1}^{K^m} \gamma^m_{t,k} \mathbf{W}^m_k \mathbf{C}^{-1} \mathbf{W}^m_k \\ & - \sum_m \sum_k \gamma^m_{t=1,k} \log \gamma^m_{t=1,k} \\ & - \sum_m \sum_{t=2}^{T} \sum_{k,j} \xi^m_{t,k,j} \log \left(\frac{\xi^m_{t,k,j}}{\gamma^m_{t-1,j}} \right) \\ & + \left(\frac{KMD}{2} \right) \log \lambda - \left(\frac{KMD}{2} \right) \log(2\pi|\mathbf{C}|) - \frac{\lambda}{2} \sum_{k,m} \mathbf{W}'^m_k \mathbf{C}^{-1} \mathbf{W}^m_k \,. \end{aligned} \tag{4.21}$$

To conform to the standard notation conventions, $Q(\mathbf{S}_t^m = k)$ is written as $\gamma_{t,k}^m$, which is the probability that the hidden state of factor m takes the value k at time t. Likewise, $Q(\mathbf{S}_t^m = k, \mathbf{S}_{t-1}^m = j) = \xi_{t,k,j}^m$, which is the joint probability that the hidden state of factor m takes the value k at time t and *and* takes the value of j at time $t-1$. K_x is the number of hidden states in primitive x, and $K = \sum_x K^x$, which is the number of hidden states in the whole model.

4.3.3 Maximisation Step

The M-step updates the parameters to their maximum likelihood values.

$$\mathbf{W} \leftarrow (\sum_{t=1}^{T} \mathbf{Y}_t Q(\mathbf{S}_t)')(\sum_{t=1}^{T} Q(\mathbf{S}_t)Q(\mathbf{S}_t)' + \lambda I)^{\dagger} \tag{4.22}$$

$$\boldsymbol{\pi}^{(m)} \leftarrow Q(\mathbf{S}_1^m) \tag{4.23}$$

$$\mathbf{P}_{i,j}^m \leftarrow \frac{\sum_{t=2}^{T} Q(\mathbf{S}_t^m = i, \mathbf{S}_{t-1}^m = j)}{\sum_{t=2}^{T} Q(\mathbf{S}_{t-1}^m = j)} \tag{4.24}$$

$$\mathbf{C} \leftarrow \frac{1}{T}\sum_{t=1}^{T} \mathbf{Y}_t \mathbf{Y}_t' - \frac{1}{T}\sum_{t=1}^{T}\sum_{m=1}^{M} \mathbf{W}^m Q(\mathbf{S}_t^m)\mathbf{Y}_t' \tag{4.25}$$

$Q(\mathbf{S}_t)$ is the distribution over the hidden states of the model, or the expected value of the hidden states at time t. $\dagger$ means *pseudo-inverse*. $\mathbf{W}^m$ is the contribution of factor m to the output mean. $\mathbf{W}$ is a concatenated matrix of all these means, thus storing all the primitives. The $\mathbf{W}$ update includes a regularisation factor, λ, as described in Section 4.3.2. $\mathbf{Y}_t$ is the observable output vector. As all the M-step updates are summations over time, it is easier to implement this with separate character samples, by gathering the statistics during the E-step.

4.4 Constraints and modifications

Several constraints and modifications were made to the model. The constraints were made so that the learnt probabilistic models have the same structure as the Piano Model, as described in Chapter 3. The constraints do not affect the likelihood from one EM iteration to the next; however, without the constraints, a higher likelihood could be obtained.

The modifications to the algorithm are heuristic optimisation techniques to avoid local maxima in the likelihood function. They are heuristic because calculating the likelihood for the model takes a lot of computation, and it is not feasible to perform

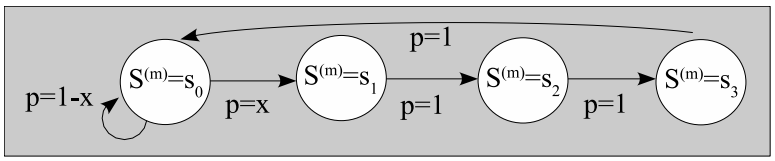


Figure 4.5: Allowed state transitions for each Markov chain, or primitive. Once out of s_0, the states transition monotonically, and deterministically, showing how the Markov chains can correspond to individual, time extended primitives.

multiple calculations for locally exhaustive parameter values. Therefore, the shifting, merging and resetting routines will normally result in a likelihood penalty in the short term, but should help to increase the likelihood in the long run by 'freeing' the algorithm from a local maxima. This is examined further in Section 5.4.

4.4.1 Modelling constraints

4.4.1.1 Transition constraints

This model will extract factors that are maximally statistically independent of each other, given the observable data. However, we are interested in factors that have repeatable time extended shapes, not merely a set of distinct values. Effectively, we want to constrain the primitives to be active along their entire length, without the possibility of interruption (Kargo and Giszter, 2000). This is accomplished by constraining the transition probabilities, $\mathbf{P}^m$, so that it is impossible for unwanted state transitions to take place. Figure 4.5 shows the possible transitions. To make the primitives more generalisable, it is possible to introduce a fixed self-transition term to the transition matrices, $\mathbf{P}^m$, allowing the primitives to be elongated if required by the data.

Due to this constraint on the possible hidden state transitions, the only parameter from the transition matrices that needs to be learnt is the onset probability of each primitive, $P(\mathbf{S}_t^m = 1 \,|\, \mathbf{S}_{t-1}^m = 0)$[1].

Self transitions can be implemented by setting $P(\mathbf{S}_t^m = i \,|\, \mathbf{S}_{t-1}^m = i)$ to a fraction of the learnt onset probability for that primitive (between $1-10\%$). A constant fraction of the onset probability was used, rather than allowing the algorithm to learn the self transition terms because local maxima in the likelihood function make these parameters difficult to learn.

[1] Hard probabilities can be used to speed up the algorithm, however normally a zero probability is defined as 1×10^{-50}, for numerical purposes.

4.4.1.2 Zero State Assumptions

The model does not provide automatically for an *off* state for the primitives, as each state has an associated output contribution. We assume that the primitives are time-extended, but not continuously active throughout the entire data sequence, (i.e. character) and so we constrain the primitives to have a zero state, with zero output contributions in all data dimensions. This is easily done by setting $\mathbf{W}_1^m = 0$ in the M-step, and not learning these parameters. It is expected that the primitives will be sparsely used, and so this rest state will be the norm, with a correspondingly large self-transition term in the $\mathbf{W}^m$ matrices, learnt during EM.

4.4.2 Heuristic optimisations

4.4.2.1 Shifting

There are many local maxima in the parameter search space, as demonstrated when running the algorithm on an artificially generated dataset, containing planted motifs, see Figure 4.6. The primitives would 'latch on' to certain parts of the motifs, and if the *start* of a learnt primitive becomes a good representation for the *end* of a planted motif in the data, the algorithm would find it difficult to learn the rest of the motif. This problem was addressed by introducing a shifting/extension heuristic, which examines the start and the end of the primitives. The reasoning behind this decision was that if a motif is being partially represented by a factor, then the shifting algorithm would be able to extend the factor to include the rest of the motif, whereas if a motif is entirely represented by a factor, then by averaging over many character samples, the learnt values outside of the motif should be equal to zero, therefore the primitive would not be extended. An example can be seen in Figure 4.7, where the dotted lines represent the specific cut offs for the primitive being examined, and are 3 dimensional.

To determine whether a particular primitive should be shifted, the start and end sections of the primitives are examined. If the mean absolute value of any dimension of the examined section is above a constant 'body definition' fraction of the mean absolute value of the whole primitive, then the section is assumed to be part of the motif, and is extended. Otherwise, if the square of all dimensions are below a 'zero definition' fraction of the primitive, then the section is assumed to be outside the motif being captured by the primitive, and the section is therefore 'pruned'. It should be noted that the length of the primitives is not fixed, and is defined therefore by this shifting

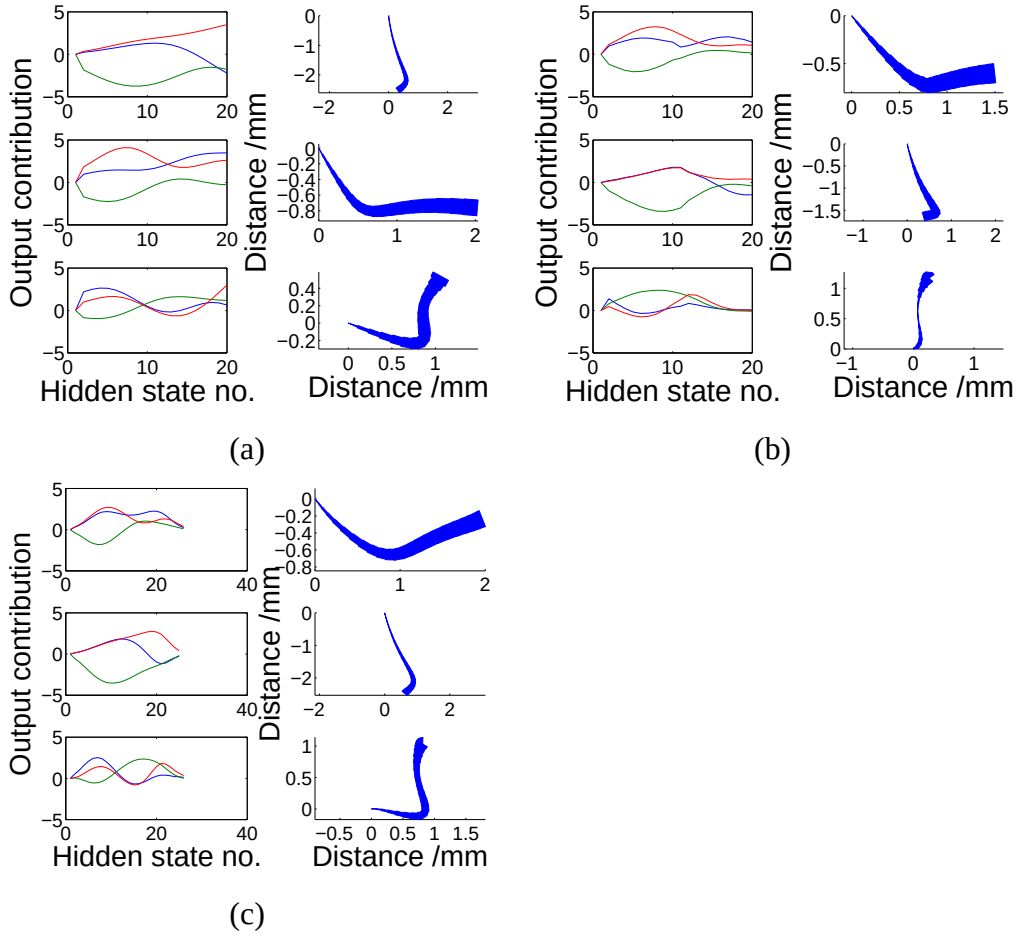
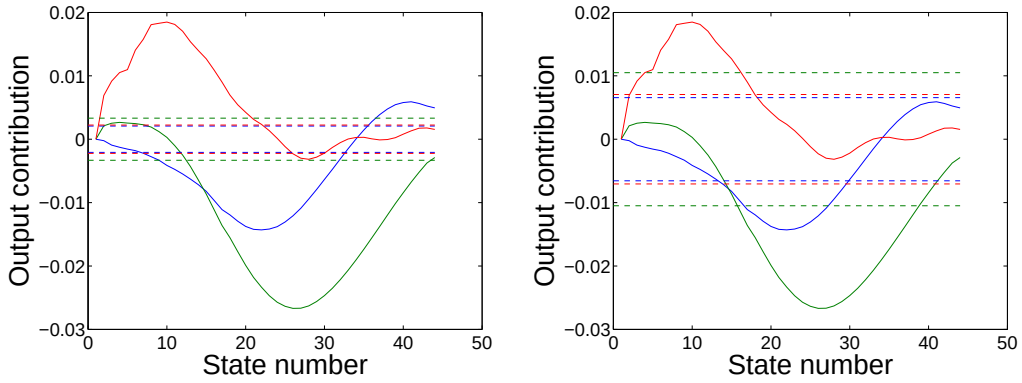


Figure 4.6: In all three sub-figures, the left column shows plots of motifs, or primitives, in normalised velocity space, and the right column shows reconstructions of these primitives as if on paper. Three artificially generated motifs are shown in (a). An attempted recovery of these motifs can be seen in (b) and (c). Both examples use identical parameter initialisations, with the only difference that in (b), no shifting was used, whereas in (c), shifting as described in Section 4.4.2.1 helped to recover the original motifs. It can be seen that without shifting, two of the original motifs were well recovered, but the recovery of the third 'latched on' to the second half of the motif. With the shifting algorithm this did not happen.



(a) (b)

Figure 4.7: A sample primitive is shown here in normalised velocity space. In (a), dotted lines are drawn at the corresponding cutoff for the zero definition, representing 10% of the variance of the primitive. In (b) the dotted lines represent 100% of the variance of the primitive, the cutoff point for lengthening the primitive. The square of the mean over 3 states at the start and the end of the primitive is compared to the square of the cutoffs shown here. In this particular case, the start of the primitive would be extended, as the pen tip pressure dimension is greater than the cutoff at the start.

algorithm. The zero and body definitions provide two more parameters that must be learnt empirically. However, as the shifting routine is not guaranteed to increase the likelihood of the model, the exact settings of these parameters was not found to effect the outcome predictably. The inclusion of shifting in general was useful, perhaps by adding noise to the inference procedure, thus effectively smoothing over the local maxima in the search space. The shifting algorithm is explored in more detail in Section 5.4, and the parameters were fixed at a zero definition of 10% of the absolute mean of the primitive, and a body definition of 90% of the absolute mean.

Likelihood cost

The aim of shifting is to increase the likelihood of the model in the ‘long run’, although an immediate increase is not guaranteed. In the ideal case, the likelihood should remain the same for any extension or reduction, as this should only happen for output contributions of zero, thus leaving the likelihood unaffected (ignoring the regularisation terms). In practice the learnt values of the primitives are never exactly zero, and so the cut-off values are necessary. This means that there is a likelihood penalty for the shifting. This is examined in more detail in Section 5.4.

Pseudo code

SHIFT

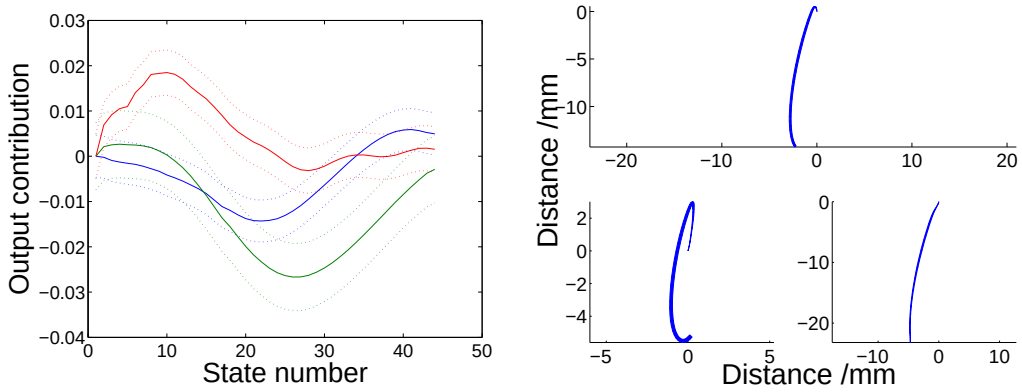
Calculate cutoff points for shifting:
1 *zerodef* $\equiv$ 10% of mean absolute value of primitive
2 *bodydef* $\equiv$ 90% of mean absolute value of primitive
3 *testsections* $=$ mean absolute value over 3 samples at start and end of primitives
4 **if** any dimension of a *testsection* $>$ *bodydef*
5 **then** extend the primitive
6 **else if** all dimensions of the *testsection* $<$ *zerodef*
7 **then** shorten the primitive

4.4.2.2 Merging

Sections of primitives can sometimes start to model similar sections of the data. If two primitives become close to identical, then they should be merged, to help with generalisation. This merging problem was resolved by examining the mean squared difference between primitives, which, if smaller than a defined fraction of the variance of the primitives (in all dimensions), then the primitives are flagged as the same, and so merged by setting one to the average of both, and resetting the other. The other variables, such as $Q(\mathbf{S}_t^m)$ were altered accordingly, preserving the onsets of the two original primitives in the new merged primitive, and resetting the onsets of the new reset primitive. Therefore the new merged primitive assumes the average shape of the two original primitives, and all the onset positions of both original primitives.

Figure 4.8 shows an example of a primitive and the ranges of potential primitives that would be merged using this algorithm, with a cutoff placed at 50% of the primitive variance. The examples given are the extremes, with the entire length of the primitive on the borderline of the merging cutoff. In practice merging does not occur as excessively as is perhaps suggested by these plots, as the EM algorithm will try to find primitives that best explain the data, therefore unless the data only contains a single motif, with very little noise, the primitives learnt will not normally be identical to each other.

It should be noted that for primitives of different lengths, the comparison is done by sliding the shorter of the two primitives along the length of the longer one, giving the average mean squared difference at several offsets. If this mean squared difference



(a) (b)

Figure 4.8: (a) Solid plot shows a sample primitive in normalised velocity space. The dotted lines show the 50% level of the variance of the primitive, which if on average another primitive lies below (in *all* dimensions), then the two would be merged. (b) Three reconstructions of primitives as they would appear on paper, the first being the original primitive, the second and the third being the extreme examples of primitives that would be merged, or considered the 'same' by the algorithm.

is smaller than the cutoff value in *all* dimensions, at *any* offset, then the primitives will be merged, by averaging them (and taking into account any offset).

4.4.2.3 Resetting

Sometimes a particular primitive will not converge towards a useful representation of the data, in which case, the onset likelihood of the primitive will tend towards zero, and the primitive will no longer be used by the algorithm. If this happens, there is a checking routine that will reset such primitives to randomised values. It would also be possible to simply 'delete' such primitives as an alternative solution to this problem.

4.5 Parameter Initialisations

As with all local optimisation algorithms, the initialisations of the parameters is very important. $\mathbf{C}$ is initialised to the covariance over time of the dimensions of the data vector $\mathbf{Y}_t$. $\mathbf{W}^m$ is initialised to sections of the mean of the data, taken at equal intervals. $\boldsymbol{\pi}^m$ is initialised so that all primitives start in state 0. In fact, it is assumed that all primitives start in state 0 for all character samples, as the start of the data is zero padded, therefore the parameter $\boldsymbol{\pi}^m$ is not needed. $\mathbf{P}^m_{ij}$ is initialised to $1/T$ for the

onset transition, and the deterministic state transitions as described above, for other transitions.

The initialisations of the primitive shapes are contained in the parameter matrix $\mathbf{W}^m$, and are extremely important, and have an effect upon the resultant primitives. It is suggested in Chapter 9 that these parameters be initialised by some fast greedy algorithm to help with finding a repeatable solution to this local optimisation problem.

Chapter 5

Factorial Hidden Markov Model Results

In Chapter 4 we discuss in detail a model for representing handwriting data using time-extended primitives. This Factorial Hidden Markov Model (fHMM) represents pen movement primitives in the the form of parameters, and the states of these primitives are represented in the hidden states of the model. The state of the model can be approximated with a set of primitive timings giving rise to a particular character reproduction, and as such does not generalise across character samples, as this timing, or onset of the primitives is represented in the posterior over the hidden states of the model. However, the shapes of the primitives themselves, and how they model the variance of the data is interesting, and will be examined in this chapter. The repeatability of the inference, and therefore reliability of primitive extraction will be assessed, and the generalisation of the model for test data, based on similar, or novel characters will be examined.

Finally the generative behaviour of the model will be demonstrated, highlighting the need for a further level of modelling that allows the primitives to generalise across different character samples.

5.1 Data

The raw data comes from an INTUOS 3 WACOM digitisation tablet `http://www.wacom.com/productinfo/9x12.cfm`, providing 200Hz pen tip position data capture. The tablet uses an electromagnetic field to detect the position and orientation of the pen `http://www.wacom-components.com/english/technology/emr.html`. The

pressure of the writing is detected using a pressure sensor in the tip of the pen, which modulates the field and is read by the tablet. The pen tip pressure (technically force, of course) information is digitised to $1,024$ discrete levels, for a desired range of $30-400$g, or $0.29-3.92$N giving a resolution of 3.53mN. When the pen is not in contact with the tablet, position information is still detected, up to a height of 6mm. The resolution of the pen position is given as $5,080$ lines per inch, however, the accuracy is given as ± 0.25mm. Assuming the latter is the case, this gives the useable resolution of the pen position as 0.5mm, or 51 lines per inch. If, however, the accuracy derives from some locally smooth offset, due to non-linearities in the electromagnetic field, as is likely to be the case, then the useable resolution for a small area is much better. Samples of the raw data can be seen in Figure 5.1, and of the same, differentiated data in Figure 5.2. The signal to noise ratio of the pen tilt and pen orientation dimensions was too low for useful information to be gained, and it was found that including this data diminished the performance of the model, therefore only the first three dimensions of the data were used. For ease of generalisation however, I will refer to D dimensions of the data.

We wish the primitives to be as generally applicable as possible, and not limited intentionally to certain parts of characters. We also need to remove any constant offset in the data (arising from drift in position of character drawing). This could be accomplished by subtracting the mean for each character sample, but it does not solve the problem within a character. To model the data using a fixed function model, we want the data to be normalised, and roughly zero centred, therefore it was decided to model the velocity, or first differential of the data. Normalisation of the dataset helps the model to be applicable over multiple scales. The differentiated (velocity) data may well be a better representation of the internal biological code for the muscle activations, as these are very unlikely to be in absolute coordinates, and would be expected to reflect either velocity or acceleration of the system. Other possible representations include affine geometrical transformations such as those proposed by (T. Flash, 2007). Before differentiating the data, it was smoothed using a Gaussian filter to reduce the amplification of noise due to numerical differentiation. There are more sophisticated smoothing methods such as those used in (Ramsay, 2000). Gaussian smoothing was chosen for its simplicity and repeatable predictability of behaviour. The difference between the raw and smoothed data can be seen in Figure 5.3. In this figure, it can be seen that without smoothing, the first differential of the data is quite noisy, especially the pen tip pressure data.

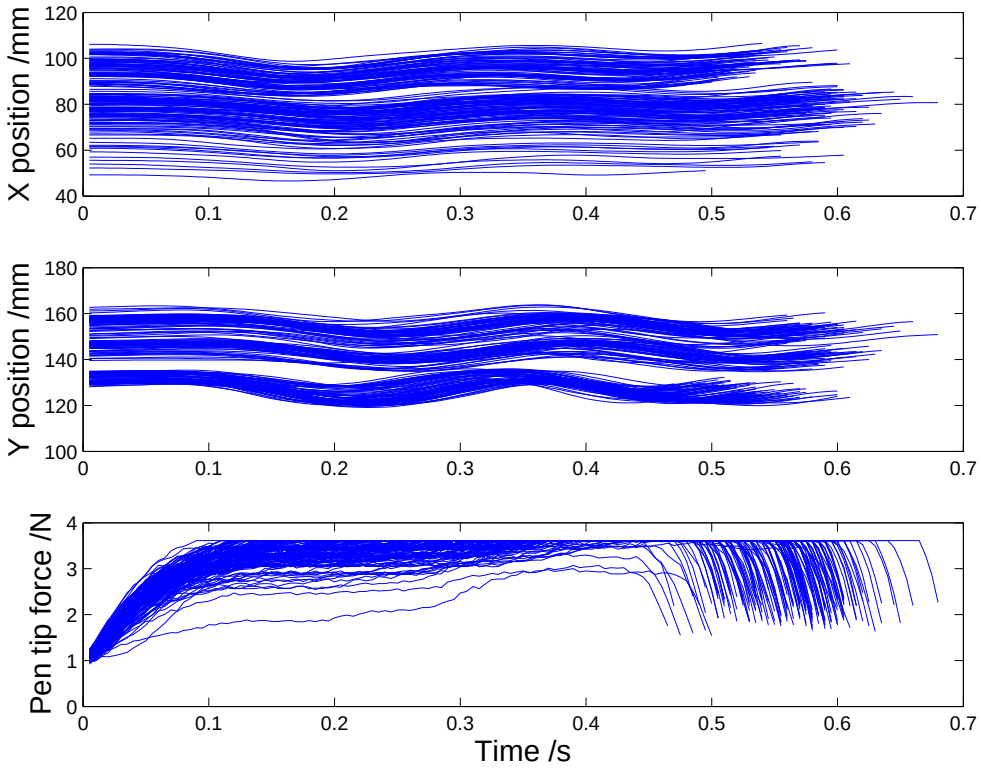


Figure 5.1: Raw data obtained from a WACOM INTUIOS 3 Data Tablet. Three dimensions of the data are shown on the three axes, X position, **Y**position and pen tip force. Several plots are overlaid on each axis. Each plot shows a separate sample of the same character.

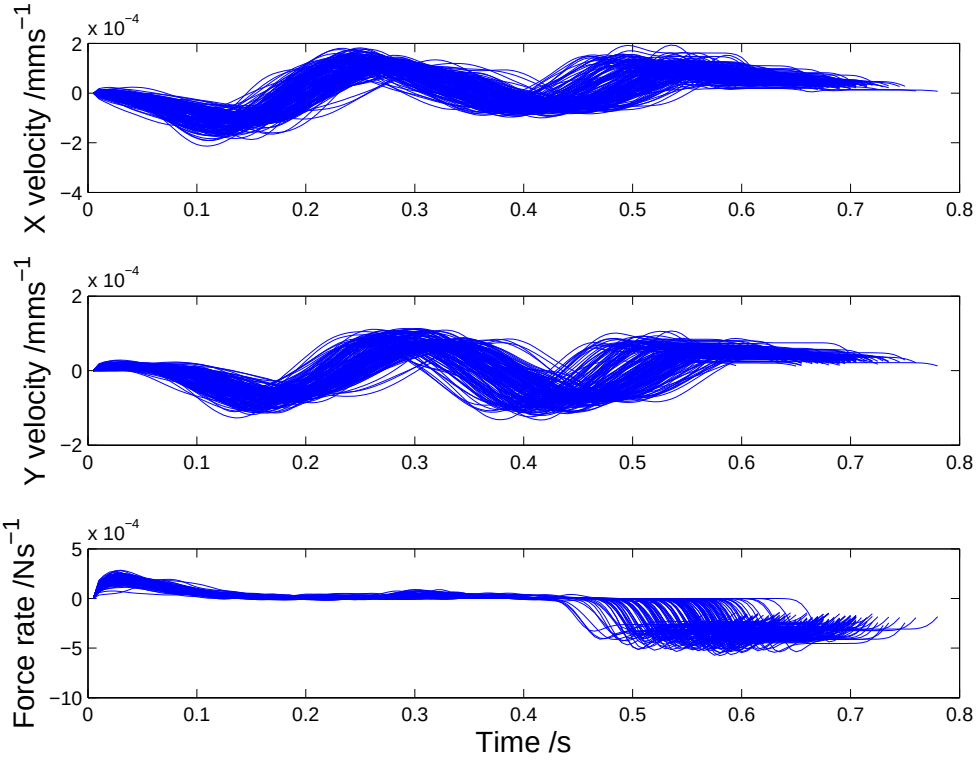


Figure 5.2: The raw data seen in Figure 5.1 after differentiation can be seen here, and the shape of the average sample can be more clearly seen than in the raw data.

Also, the data is extended at the end of the character, preserving the local gradient of the data. Without this extension, there were problems with the character reproduction, as effectively the data is indicating that the pen reaches the point of leaving the paper, and then freezes. This is very difficult for the primitives to model accurately, and so 'tails' appeared at the end of character reproduction, a sign of poor modelling of pen-paper contact. With the extension to the end of the data, the assumption is that the pen continues to move with the same velocity after it is lifted from the paper, which is easier, and more intuitive to model.

During the inference, complete state distributions need to be calculated, and stored, requiring large amounts of memory. This is alleviated by separating the data into character samples, with separate state inference for each character, then combined parameter updates with accumulated statistics gathered in an online-learning fashion (see Section 4.4). The accumulated statistics are implemented by splitting up the sums over time in the parameter update equations into N sums over shorter length of each character sample. Intuitively this segmentation makes sense, as there is a gap between each character, where the pen is off the paper. During this period it is unclear whether our

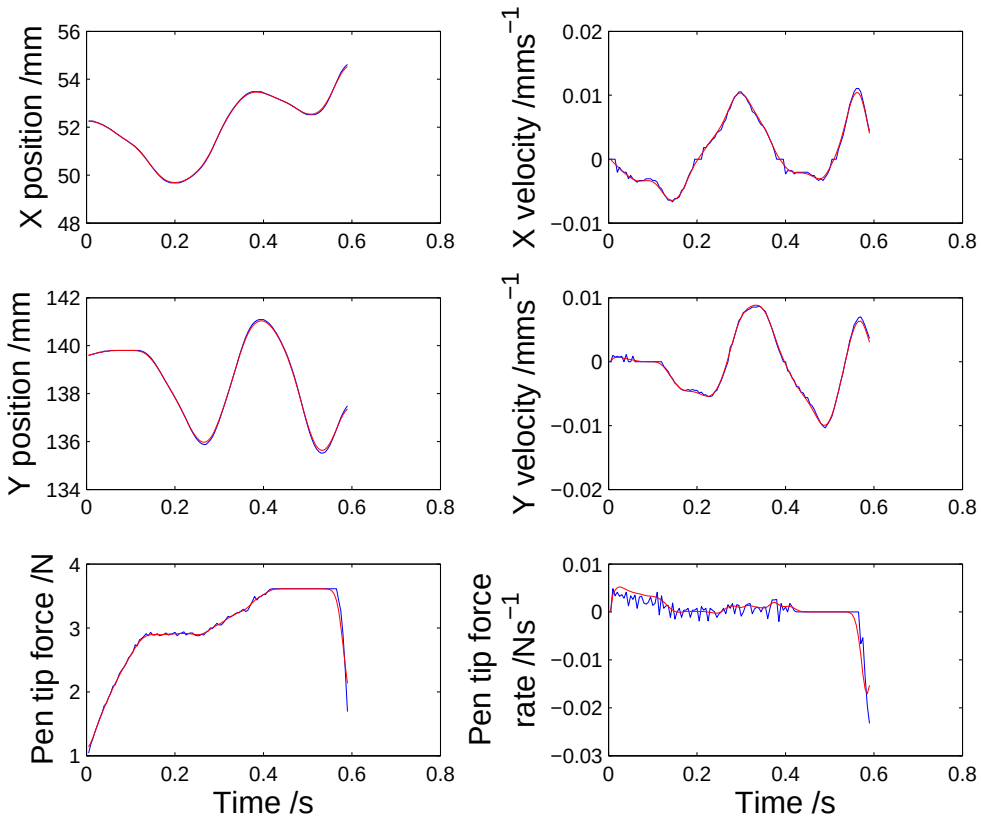


Figure 5.3: Single character sample data shown untreated in blue and Gaussian smoothed in red. Axes to the left show undifferentiated data. Axes to the right show differentiated data.

model would be sufficient for modelling the primitives involved in the pen tip motion; also the dimension of the data would be reduced, as the pen tip pressure would not give any information. Furthermore, any pen off-paper motion would be a function of the previous and subsequent characters to be drawn, and other higher level, unmodelled effects (such as breaks from data entry). The cut-off point for *pen up* regions was defined as 250, or 0.88N, empirically giving the whole character, whilst removing *pen bounce* noise. Currently, there is no provision for characters that require two separate strokes, where the pen must leave the paper within the drawing of the character. This drawback could be removed by including some random variable indicating when a character is being drawn, which would couple the two parts of the character together. This setup would be greatly improved with visual feedback, as we naïvely group pen strokes into single characters based upon spacial coherence; i.e. the dots need to be above or at least near the *i*'s, and the dashes need to cross the *t*'s.

In a given dataset, n enumerates the character samples, of varying length, so for each character sample, $\mathbf{Y}_n$, is a $D \times T_n$ matrix of observable variables, where D is the dimension of the data (3 in this application) and T_n is the number of samples in character n. For ease of notation whilst describing the model, the sample number will be subsequently omitted, and the observable data can be thought of as a concatenation of the separate samples unless stated otherwise. The data is normally time indexed, making $\mathbf{Y}_t$ a vector of length D.

5.1.1 Datasets Used

Several different datasets were used for training purposes. Here the most commonly used datasets are described:

Artificially generated dataset Five artificially generated primitives were used to generate a dummy dataset, using the fHMM as a generative model, with the constraints and assumptions used for the learning procedure. Figure 5.4 shows the primitives used, and some dataset samples as they would appear reproduced on paper. The length and number of primitives, as well as other parameters, such as self transition priors, were configurable, allowing testing of various aspects of the algorithm.

***'p'* dataset** A sub-sampled dataset composed of the character *'p'* consisted of 131 character samples. Figure 5.5 shows the variance profile of this dataset.

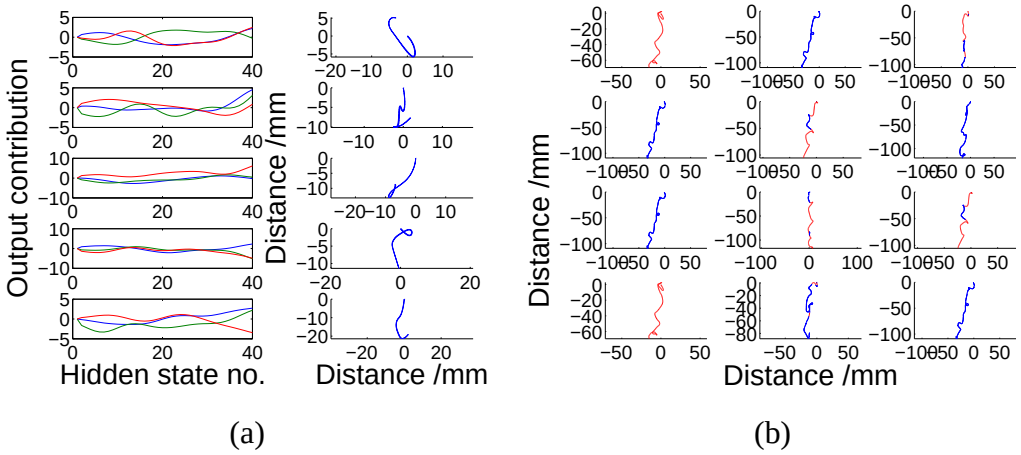


Figure 5.4: (a) Five artificially generated primitives shown in normalised velocity space. Each axis contains 3 plots, representing a dimension of the primitive data. (b) Samples of data generated using the constrained fHMM with primitives as shown on the left. Here the data is plotted as it would appear on paper, with the thickness of the line representing pen tip pressure.

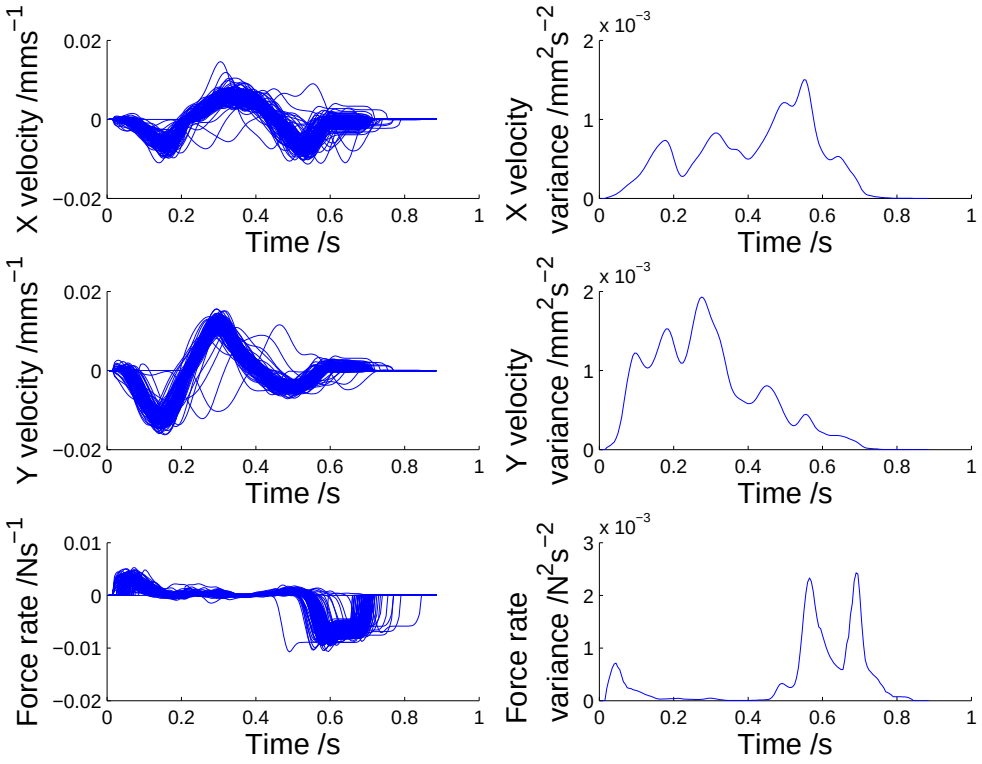


Figure 5.5: Overlay plot of 131 character samples of the letter *'p'*, demonstrating the variation in the data. The variance profile of the dataset is shown on the right. The implicit hypothesis of the primitive model is that part or all of this variation, can be modelled by primitive timing noise.

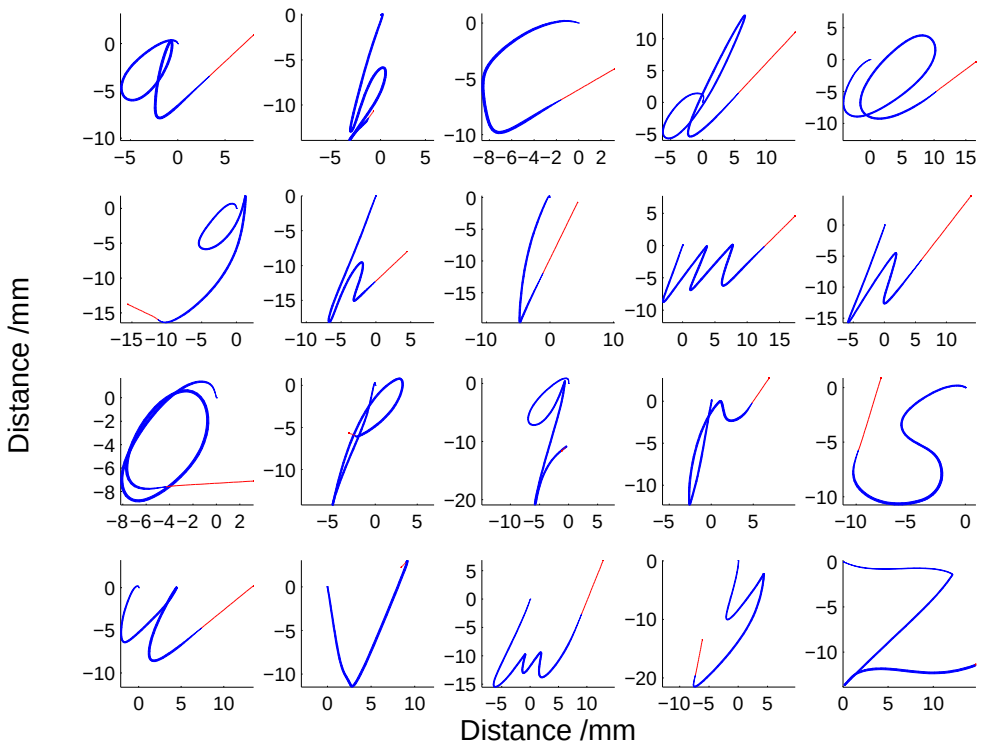


Figure 5.6: 20 samples of characters taken from the mixed dataset. Total size of dataset is 2858 characters, with over 100 samples of each character.

Multiple characters dataset A large dataset was used with samples of most characters. As there is no allowance for characters including *pen up* sections, such as a *t*, these are not included. The characters used were *abcdeghlmnopqrsuvwyz*. There were 2860 character samples, with length $\mu = 101.0$, $\sigma = 21.1$ samples. Samples of this dataset can be seen in Figure 5.6.

Multiple writers Several datasets have been contributed by Pooria Zamani, of Tehran University, Iran. These datasets contain several different characters, half from Latin script, and half Persian script.

For the purpose of examining primitives over different writers, and different character sets, this dataset was collected from six biliterate subjects fluent in Arabic (Persian) and Latin (English) script. Each subject produced around 1000 character samples, separated into 6 types of Persian character, and 2 types of English character. The characters produced were $\{g, w, daal, horouf, laam, re, saad, vaav\}$.

5.2 Variance profiles of data

The different characters are drawn with a certain degree of variation in the movements of the pen. Certain parts of a character may potentially have greater or lesser variance than others, leading to a variance profile of a character, over time. The shape of this variance profile is different for the different characters, and also for the different writers. The character samples are separated using a pen tip pressure cut-off point (see Section 5.1). This means that any delay in the onset of a character from the recording start point should be constant across different samples of a character, assuming low variance in the start of the character.

The variance profile over time for different characters can be seen in Figure 5.7. These variance profiles, from all the characters in the Mixed Character dataset, show that the variance tends to be peaked in certain areas of the characters, rather than be flat, or slowly increasing. Also, these variance peaks are in different places for different characters.

The variance profile provides an objective measure of how well the model captures the variation of the dataset, and is used in Chapter 8 to examine the performance of the model, and compare it to others.

5.3 Primitives extracted

The constrained Factorial Hidden Markov model, as described in Chapter 4 can extract primitives from data, along with the most likely timing occurrence of these primitives. For a single character, such as the letter *p*, the learnt primitives capture certain areas of this character, as can be seen in Figure 5.8. They are not pen strokes, as they can overlap with varying offsets, creating very different reproductions on paper. In fact, it is difficult to immediately see where a particular primitive might be active in the character, or whether the set seen here would be capable of reproducing a convincing character sample. Figure 5.9 shows 5 character reconstructions of the dataset, showing where the primitives are active with arrows, and with a primitive onset timing diagram, in the form of spikes. Primitives can also be inferred from datasets containing multiple characters, where the primitives are then shared across all the characters. An example of such a set of primitives can be seen in Figure 5.10, and 24 samples of reconstructed characters can be seen in Figure 5.11. When the primitives are shared across multiple characters, the shapes of the primitives are less specific to a particular character, and

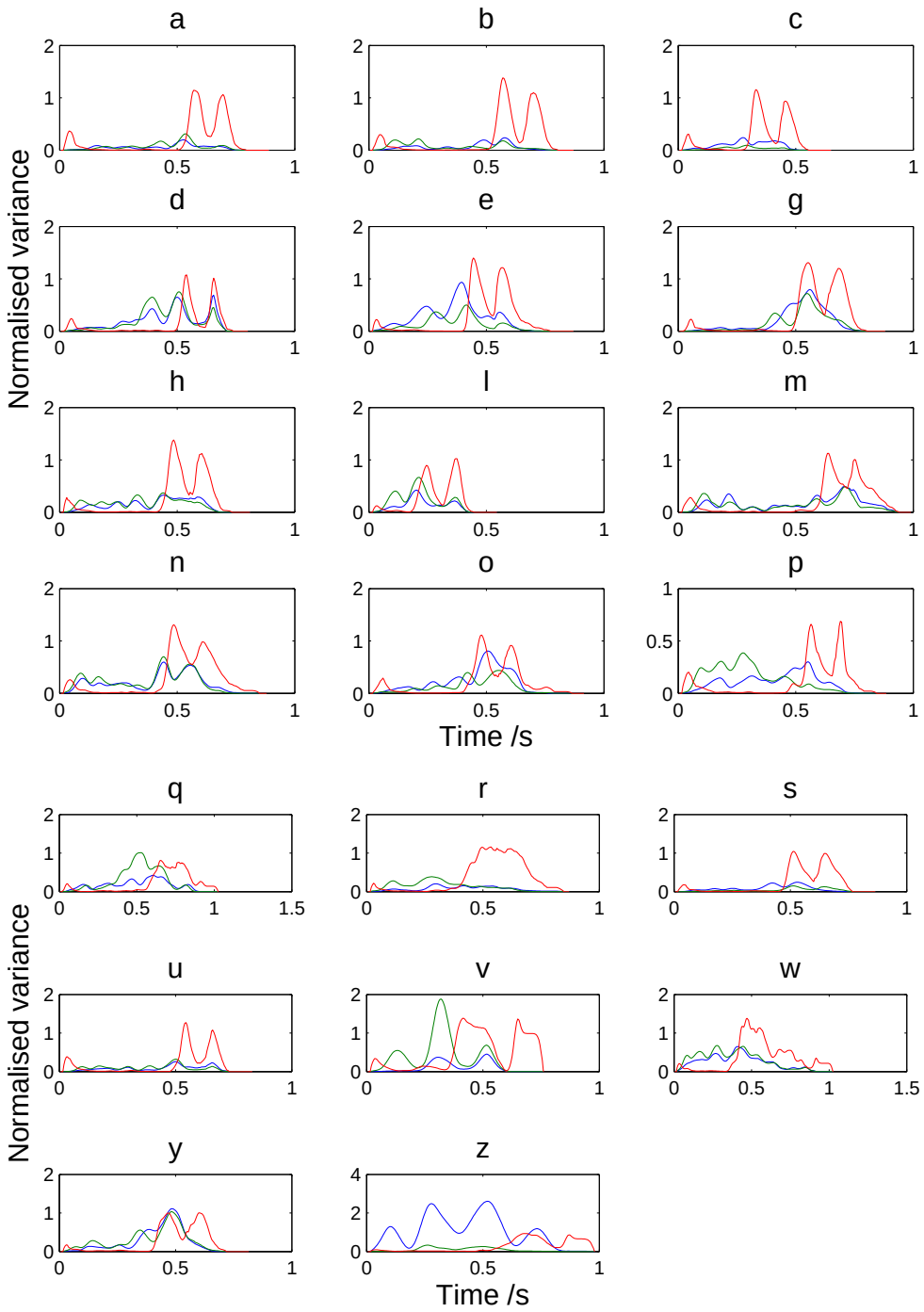


Figure 5.7: Variance profiles of all the characters in the mixed character dataset. Each axis has three plots, corresponding to each of the dimensions of the data, and the dimensions are colour-coded (blue=x, green=y, and red=z). Each dimension has been normalised so the the variance over all the data is unity. These variance profile plots show regions of some characters that vary more than for other characters, and the overall variance of each character is roughly the same, with the possible exception of the character z.

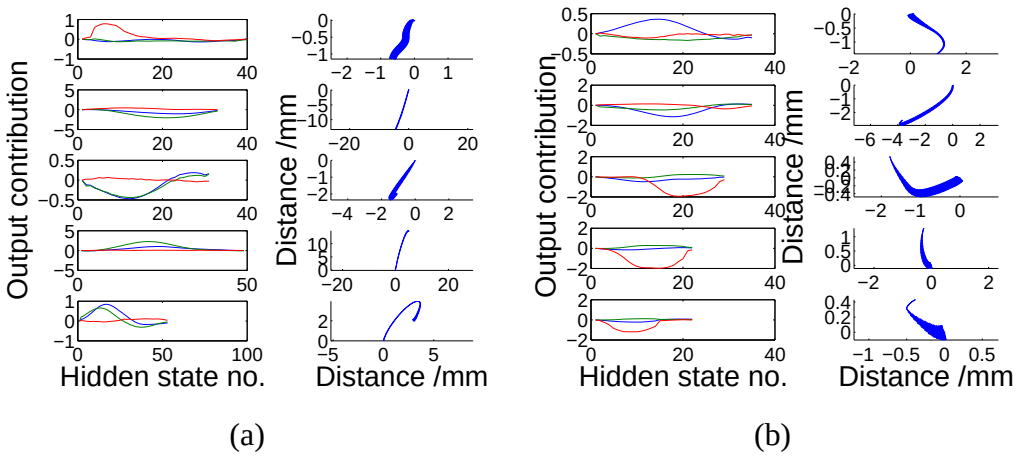


Figure 5.8: 10 examples of primitives extracted from the *p* dataset. These primitives were extracted using the uncoupled model, as described in Chapter 4. The initial length of the primitives was 20 states, and some were lengthened as described in Section 4.4.2.1.

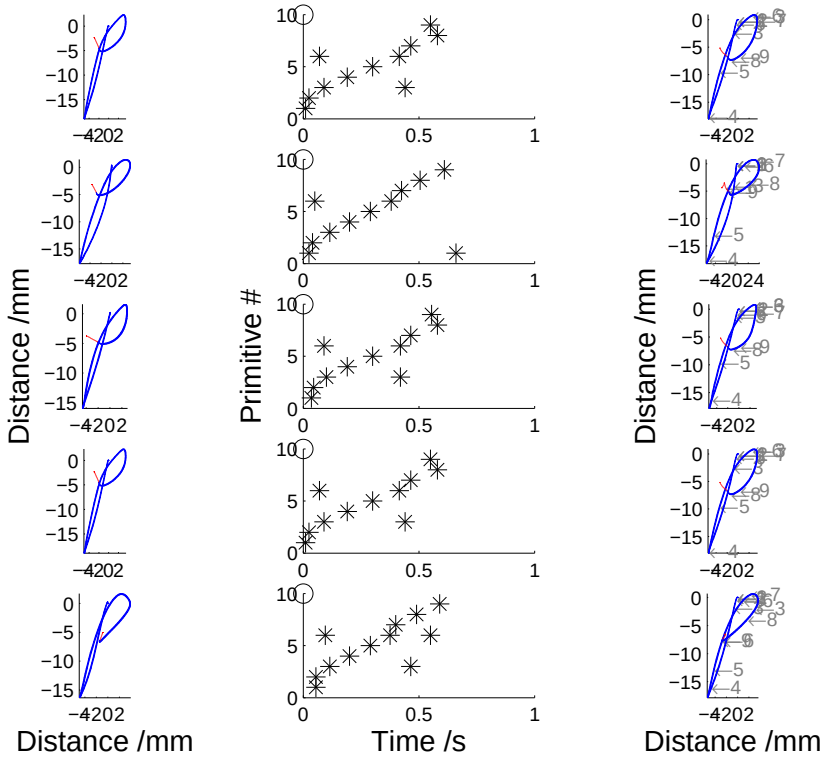


Figure 5.9: Reconstructed samples of 5 characters from the *p* dataset, using 10 primitives, shown in Figure 5.8. The leftmost column shows the original data, the centre column shows the onset timing of the different primitives, in the form of spikes, and the rightmost column shows the reconstruction of the dataset using the primitives. Arrows show where primitives become active during character production. The primitives were initialised from sections of the mean character, causing the semi-consecutive nature of the activations.

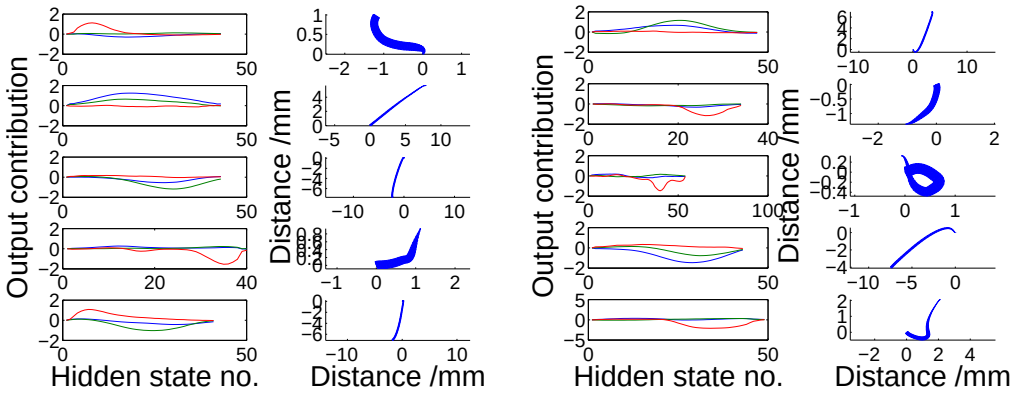


Figure 5.10: 10 primitives extracted from a dataset containing 30 samples of the characters $\{a, b, c\}$, using the uncoupled model as described in Chapter 4.

they are therefore more generalised primitives. To examine how selective a primitive is to a certain character, Figure 5.12 shows how often a particular primitive is used for each character type, and the clustering of primitive activation during the writing of a character.

5.4 Repeatability

The repeatability of the shifting algorithm is examined here. For more details on how the shifting algorithm operates, please refer to Section 4.4.2.1. To analyse the repeatability of the algorithm, the likelihood of the model is examined. A derivation of the Variational Log Likelihood (VLL) is given in Section 10.1. A plot of the likelihood over 200 EM iterations, for 5 different runs, with random parameter initialisations can be seen in Figure 5.13. In this case, no shifting of the primitives was included (see Section 4.4.2.1). The EM algorithm converges towards a local maximum in the likelihood function. This can be seen in Figure 5.13, where each run converges to a roughly fixed VLL value after around 20 EM iterations. With artificially planted motifs in the the data, the primitives can be shown to partially represent the motifs without shifting (see Section 4.4.2.1 for a description of the shifting algorithm). Intuitively, the primitives are *latching on* to certain regions of the motifs, as discussed in Section 4.4.2.1. To deal with this problem, a shifting algorithm was introduced, to examine the ends of the primitives, and extend or shorten them appropriately. In terms of the likelihood, the shifting algorithm is trying to move out of local maxima in the parameter search

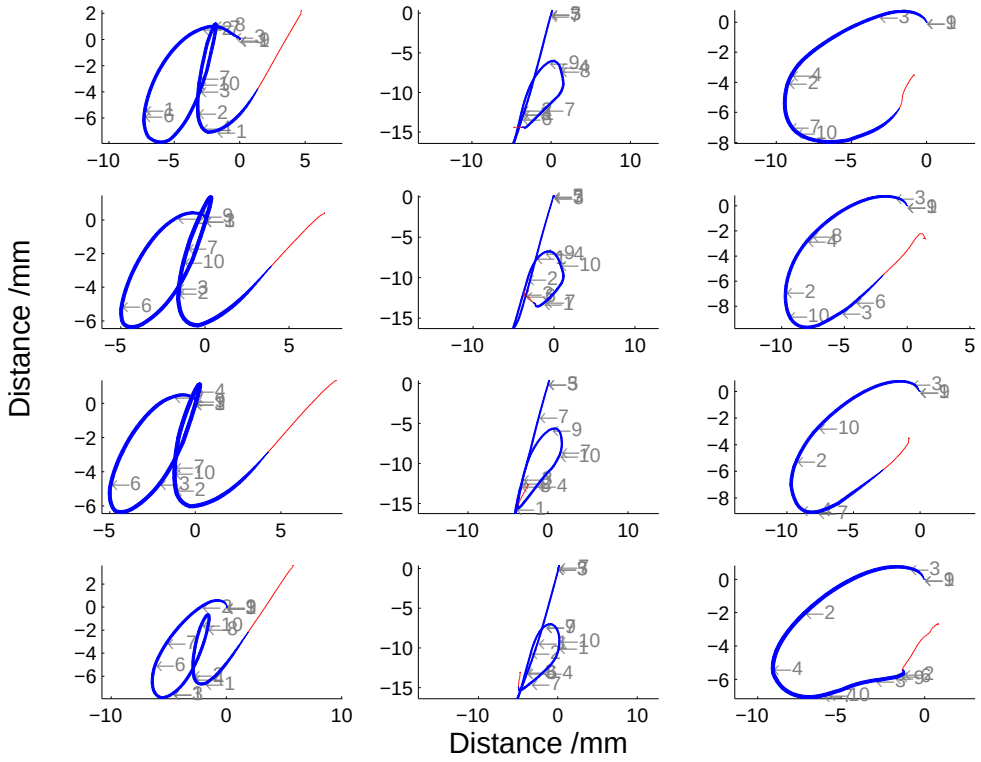


Figure 5.11: Samples of dataset reconstructions using the 10 primitives seen in Figure 5.10. Arrows show primitive onsets. Thickness of line represents pressure at pen tip, and thin red lines represent pen movement off paper. Almost all primitives are shared between all character types, as seen in Figure 5.12

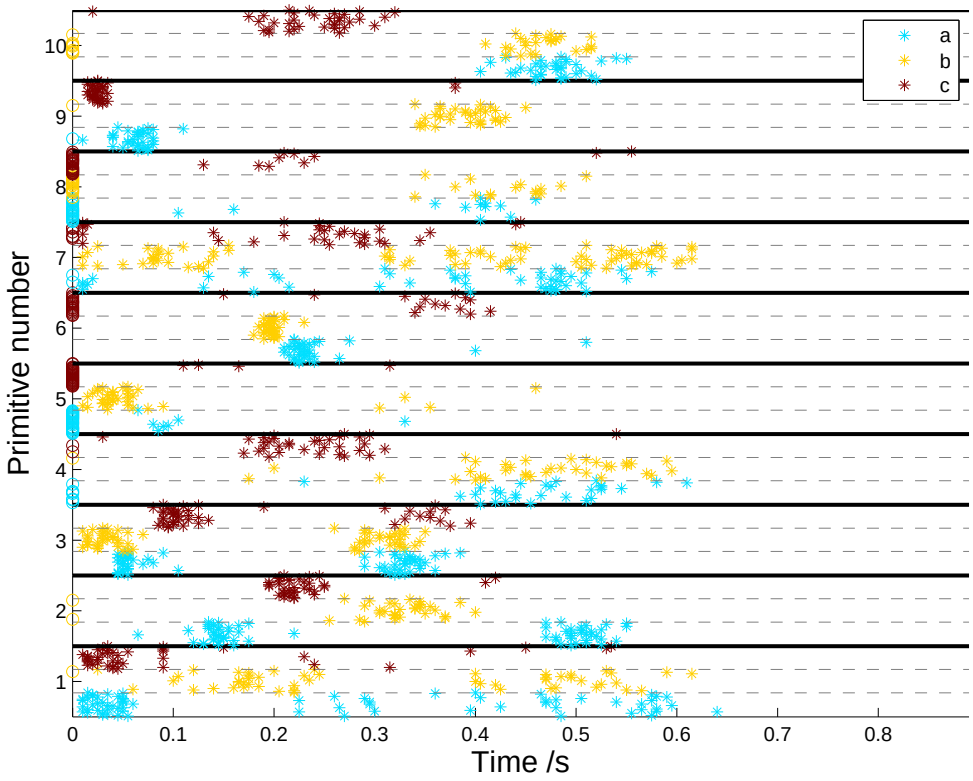


Figure 5.12: Scatter plot of all primitive activations for samples of characters $\{a, b, c\}$. Primitives used in different characters are colour-coded, and subdivided into three rows for each of the ten primitives. A star represents the onset of a primitive, and a zero at time 0 represents a sample in which that primitive was not used. In this dataset, it can be seen that all primitives were used in all characters.

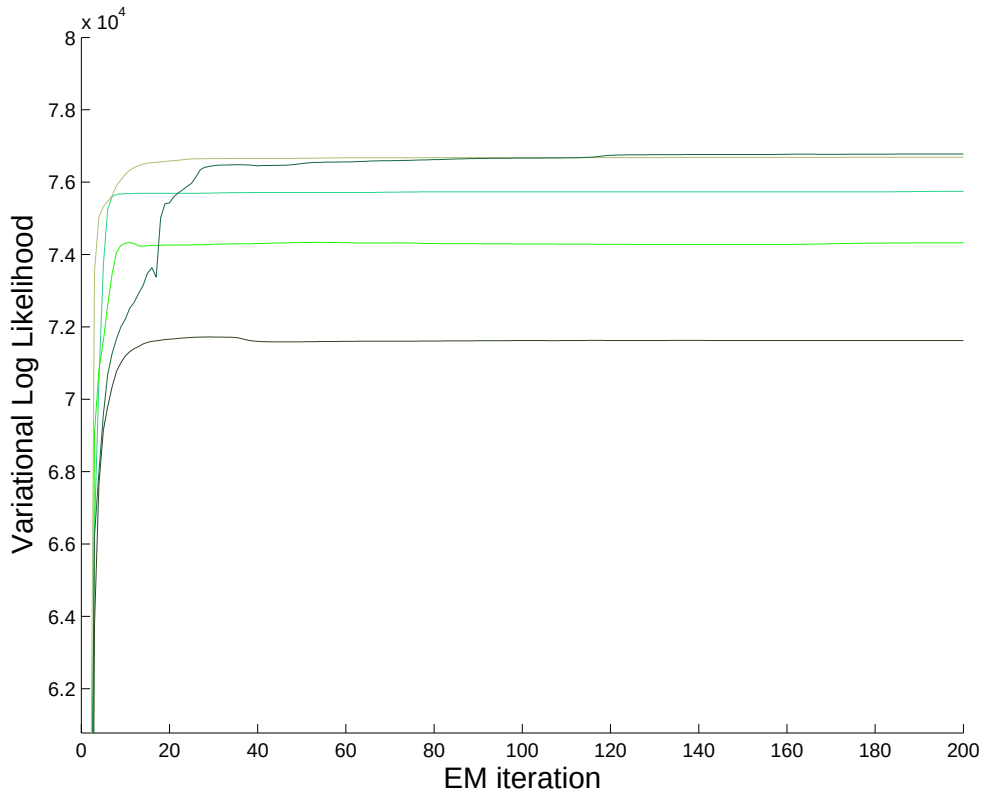


Figure 5.13: Variational Log Likelihood of 5 runs with random parameter initialisations, and no shifting of primitives, showing the convergence to different local maxima in the likelihood function. As the VLL is the objective function that is optimised during variational EM, it should be monotonically increasing until a local maximum is found. From this plot it appears that this occurs after roughly 10-30 EM iterations. The apparent decreases in VLL probably occur due to machine limits on numerical accuracy during the VLL calculation.

space; however, the algorithm is not based on any likelihood calculation, and as such, provides no theoretical guarantee that the shifting process will improve the likelihood.

The shifting algorithm has two parameters that are not learnt, rather defined as constants at the start of the learning procedure. These constants define the zero definition (z), and the body definition (b) of the primitive, as percentages of the variance of the primitive. The shifting algorithm, and these constants are discussed in Section 4.4.2.1. It was generally found that different initialisations of the model parameters had more of an effect on the outcome of the learning process than differences in the shifting parameters. However, to show that the shifting algorithm is useful, an artificially generated dataset with two motifs was created. The original motifs can be seen in Figure 5.14.

The primitive model was initialised randomly, and trained on the dummy dataset 5 times without any shifting, and 5 times with shifting every 5 EM iterations until a final

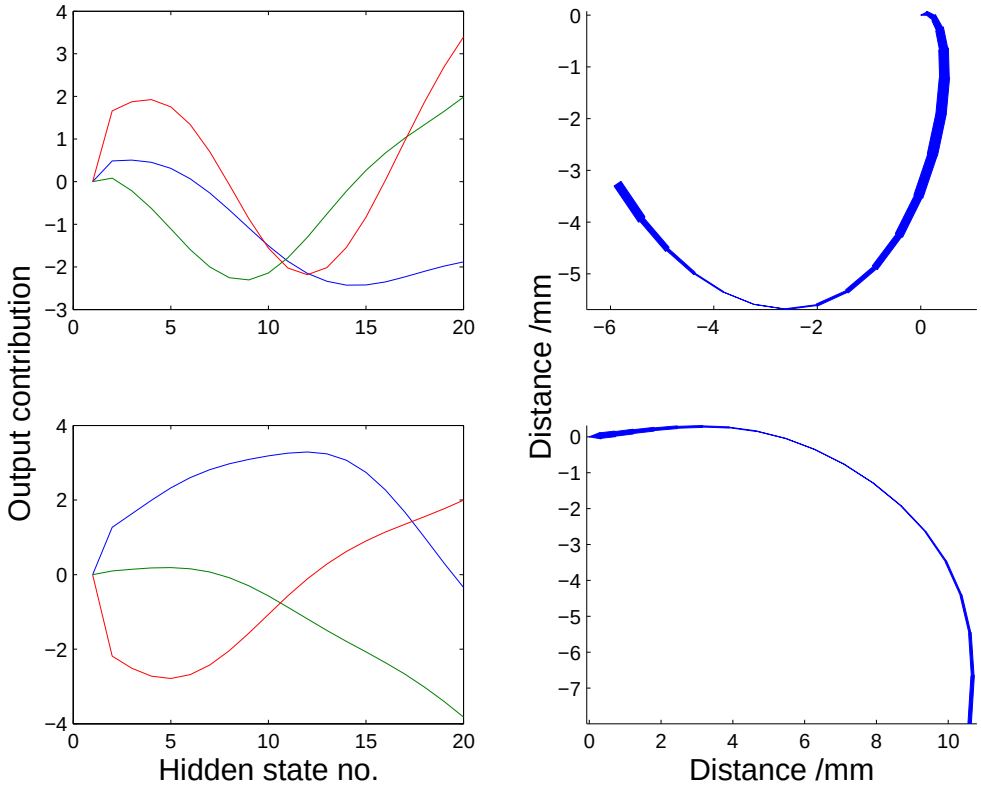


Figure 5.14: Two motifs used to construct a dataset for testing purposes.

20 iterations without shifting. Each run iterated 200 times, and the plots of the VLL can be seen in Figure 5.15 showing the convergence of the algorithm.

Without shifting, the algorithm clearly converges to positions in the likelihood function, whereas with shifting, the algorithm seems to converge to the same position, except for one run which does not. To examine what these different values of the VLL mean for the primitives extracted, a comparison of the final primitives with the original motifs can be seen in Figure 5.16.

The expectation-maximisation (EM) training procedure converges to different solutions on the same training dataset without the inclusion of shifting. Shifting does help the recovery of the original motifs, although it is not totally reliable, as can be seen from the differing VLL values of the runs. A more detailed comparison of the lower VLL primitives and a successful run can be seen in Figure 5.17. The lower VLL run has primitives that differ slightly from the original motifs; however the variation is not great, and is not due to the problem being addressed by the shifting algorithm, as the learnt primitives do not differ at their ends, rather than in the middle of the primitive. This is not the same as the latching problem that is addressed with the shifting

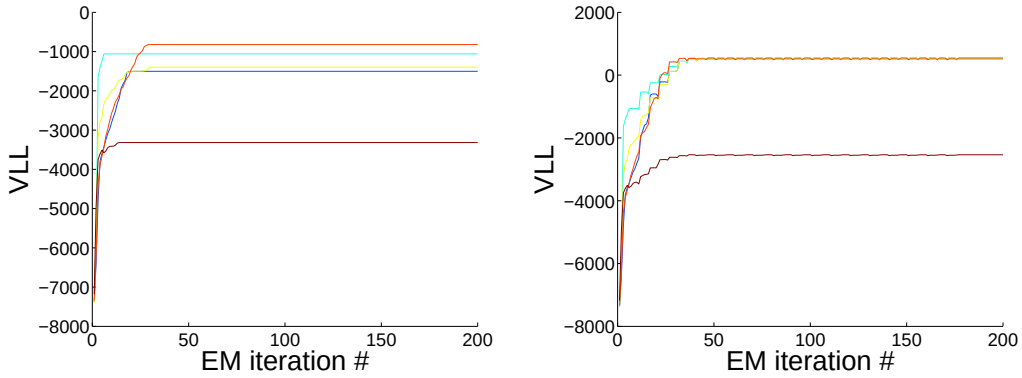


Figure 5.15: The variational log likelihood for 10 training runs of the primitive model, on an artificially generated dataset constructed using 2 motifs. 5 runs shown on the left have no shifting, and clearly converge to different local maxima in the likelihood function. The 5 runs shown on the right have shifting as described in Section 4.4.2.1 every 5 EM iterations, and it can be seen that they converge to roughly the same position in the likelihood function, except for one run.

algorithm, which is highlighted in Figure 5.18. The run with shifting that resulted in a lower VLL was possibly due to bad parameter initialisations, which, when coupled with a relatively small dataset and the presence of noise, meant that the algorithm failed to find the best parameters.

To examine the repeatability of the algorithm on a real dataset, 5 runs with 5 primitives were performed on a dataset containing the character *p*. The first two runs had the same parameter initialisations, one with shifting and one without. The remaining three runs had different parameter initialisations, all with shifting. The primitives extracted can be seen in Figure 5.19. The different initialisations have an effect on the resultant primitives extracted even with shifting, as the runs with different initialisations have differing resultant primitives. The shifting algorithm not only helps with the recovery of artificially planted motifs in the data, as shown above, but it also helps with the modelling of a real dataset, as the variance profile of the reconstructed dataset without shifting when compared to the variance profile of the original data is almost twice as bad as the shifted runs. The total absolute variance profile error (TAVPE) of the unshifted run was 43.18, whereas the TAVPE of the shifted run was 26.26 for exactly the same parameter initialisations.

Without shifting, the algorithm gets stuck in local maxima in the likelihood function. The shifting algorithm helps the system out of the local maxima caused by the *partial motif representation* problem, whereby a primitive latches on to a section of

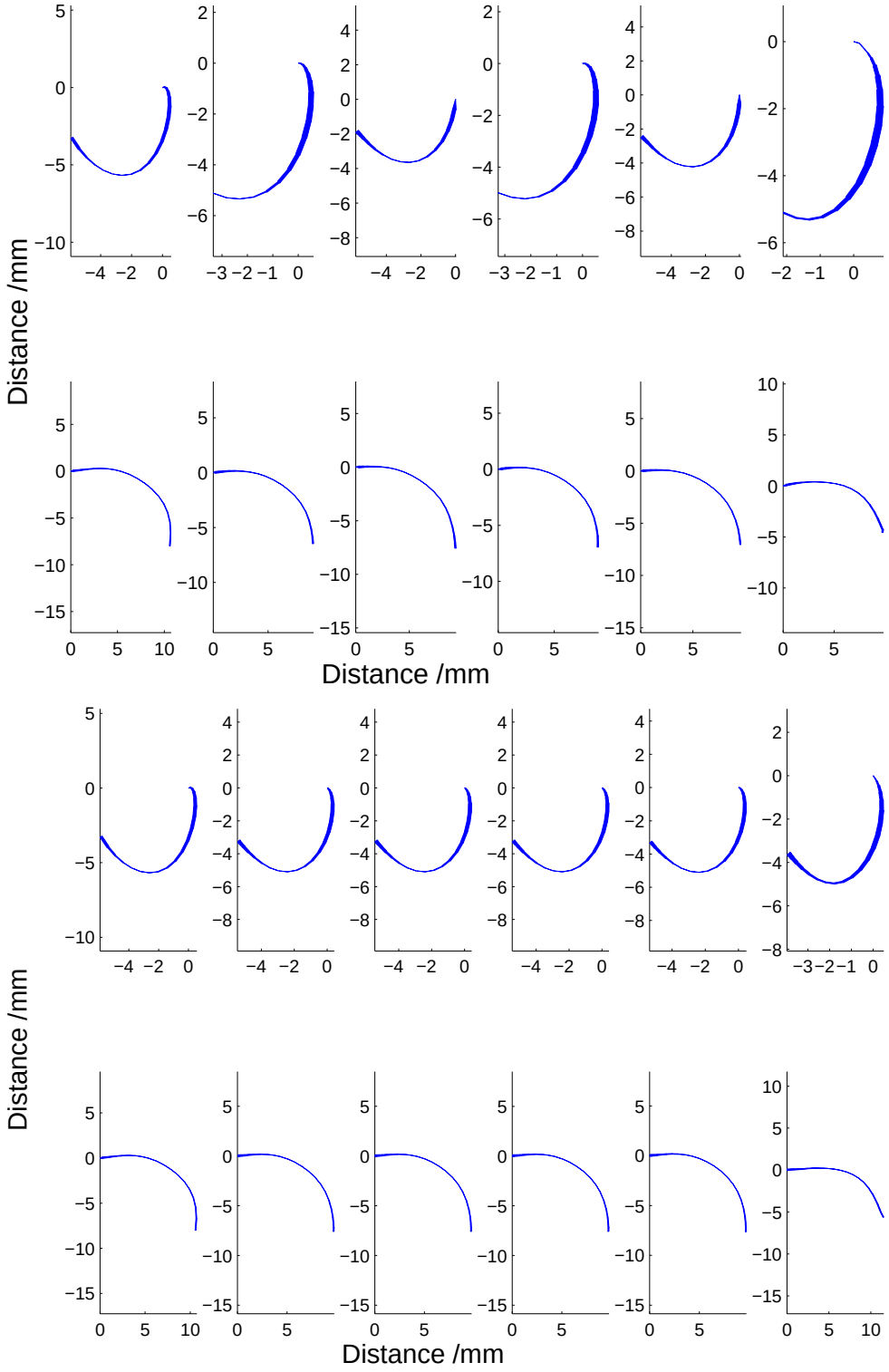


Figure 5.16: Primitives extracted are compared to the original motifs in the dummy dataset. The upper plots shows the 2 original motifs on the left, followed by the 5 runs with no shifting. The lower plots show the same two original motifs on the left, followed by the primitives extracted from the runs with shifting. The shifting run with the lower VLL as seen in Figure 5.15 is the last run, shown on the far right at the bottom. The differences in the VLL do translate to noticeable differences in the shapes of the primitives, although the differences are small.

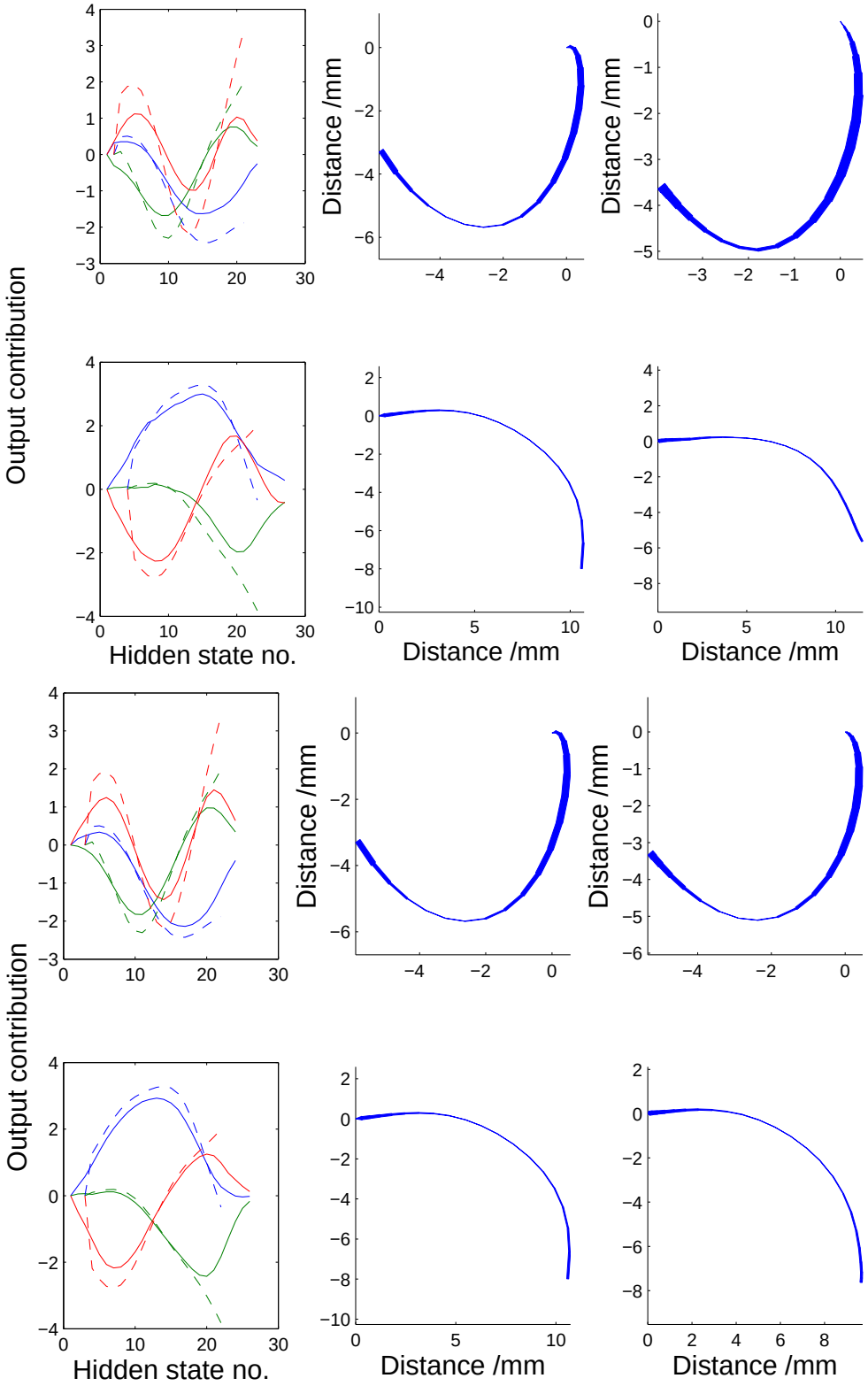


Figure 5.17: A more detailed comparison of two runs with shifting and the original motifs can be seen here. The upper plots show the run with a lower VLL, and the lower plots show a successful run with a higher VLL. The leftmost column shows the three dimensions of the primitives overlaid with the original motifs plotted as dashed lines. The middle column shows the original motifs, and the rightmost column shows the extracted primitives.

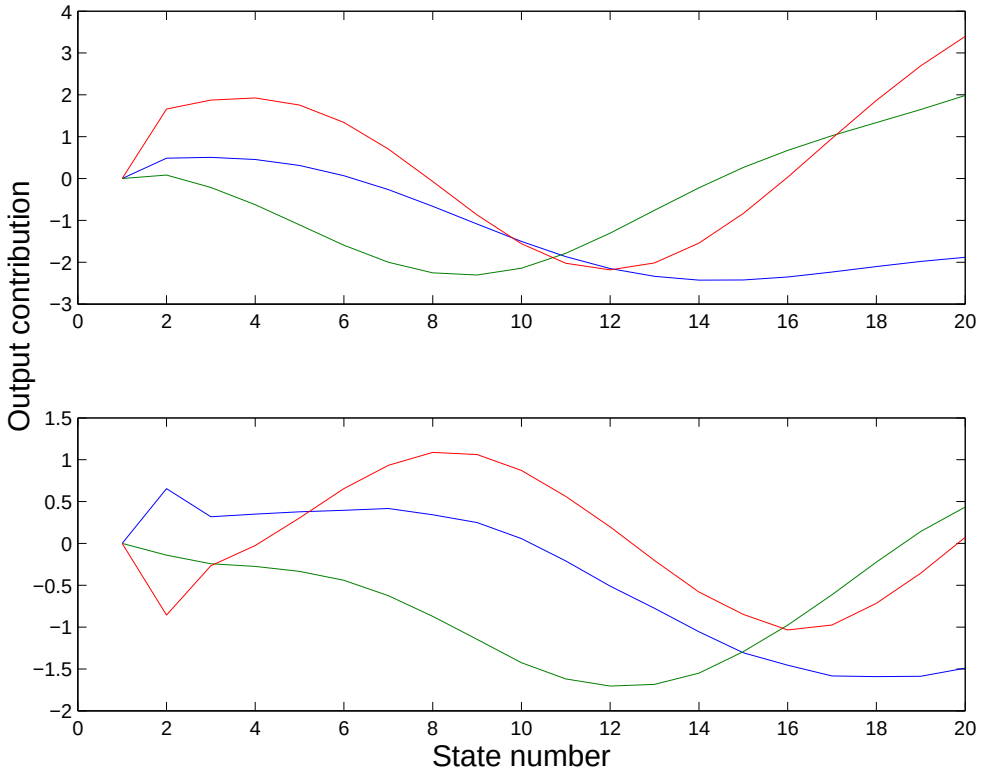


Figure 5.18: An example of the *latching problem*, whereby the primitive captures a section of the original motif, but missing a tail. The upper plot shows an original motif used to construct an artificially generated dataset, and the lower plot shows a primitive extracted from the dataset using EM without any shifting. It can be seen that the extracted primitive represents a translated version of the original motif, and so misses the end section. The shifting algorithm would extend the end of the primitive thus allowing it to capture the entire motif.

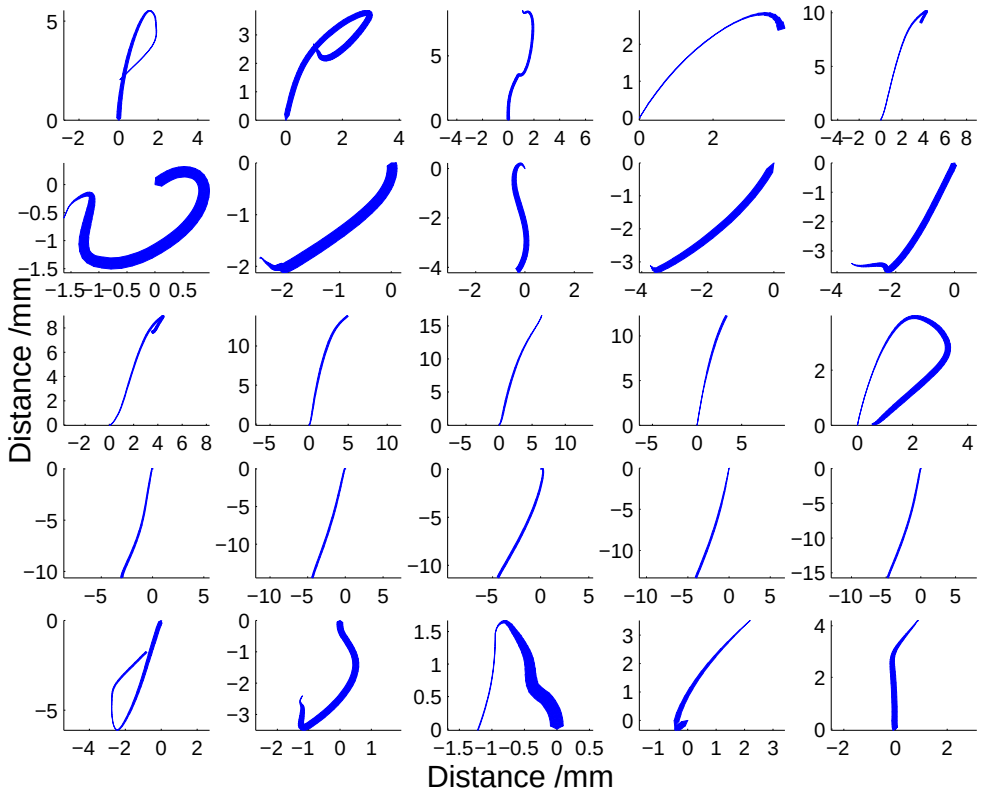


Figure 5.19: Five primitives were trained on a dataset containing the character *p* multiple times with different initialisations. Each column represents the resultant primitives of a separate run. The first 3 leftmost columns are the results of 3 runs with different parameter initialisations, showing certain similarities in the primitives extracted. The 2 rightmost columns show two runs with identical parameter initialisations, where one run has shifting and one without, showing the effect of shifting on otherwise identical runs.

a motif, but not its entire length. The shifting parameters defining when a primitive should be extended or contracted have little effect on the outcome of the learning process, although they do need to have sensible values. It was found empirically that the values $z = 60\%$, and $b = 90\%$ of the variance work well. The parameter initialisations do have a significant effect on the outcome of the learning process, so to standardise the initialisations across different runs, the initialisations of the primitive shapes are taken from sections of the mean data[1]. These are taken as defaults for all other experiments. The gap between shifting was set at 5 steps of the EM iteration.

For the artificially constructed dataset used here, it is known how many primitives should be present; however in handwriting data this parameter is unknown. For most experiments, 10 primitives were used. As more primitives are added, the fHMM can better model the data, so it is difficult to find an optimal number without taking into account computational load. The optimal number of primitives is discussed further in Section 7.5, taking into account the complete coupled model.

5.5 Generalisation of primitives

The assumption that the learnt primitives arise due to fundamental motor primitives means that the primitives should generalise across different samples of characters, and possibly even to novel character types. To examine this, a set of 15 primitives were learnt from a dataset containing the characters $\{a,b,c,d,e,g,h,l,m,n,o,p\}$. A reconstruction of some samples from this dataset can be seen in Figure 5.20, and the primitives learnt can be seen in Figure 5.21. These parameters were then fitted to some other datasets. Figure 5.22 shows samples of the reconstruction of some novel characters using the same parameters after a single E-step. Both the previously seen character types, and novel character types are well reconstructed, and easily recognisable. However, for a set of unlearnt primitives taken from sections of mean character samples, and with the same average lengths as for the learnt primitives (see Figure 5.23), the reconstruction was much worse, as seen in Figure 5.24. The VLL for the first reconstruction, using learnt primitives was -6847, whilst the second reconstruction had the lower VLL of -9893 and the reconstructions are clearly worse, despite the fact that the primitives used were taken from sections of mean character samples. This shows that these type of primitives do generalise to other characters, and are not limited purely

[1]The mean is taken over samples of the same type of character, and primitives are initialised from this using a sliding window.

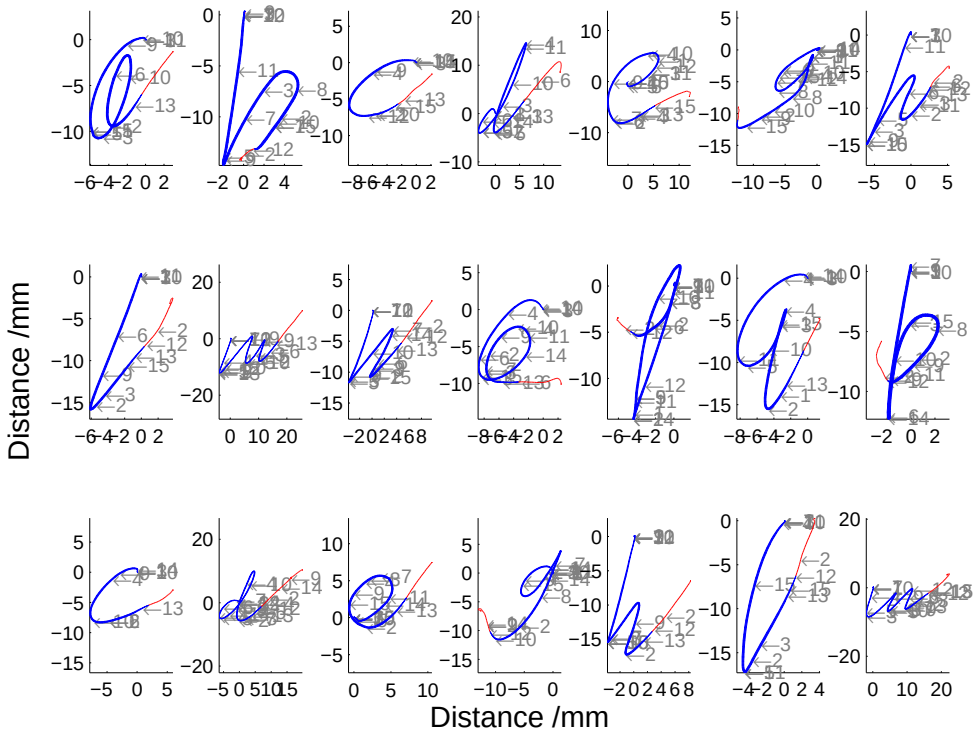


Figure 5.20: Reconstructions of a dataset containing 12 character types: $\{a,b,c,d,e,g,h,l,m,n,o,p\}$, using 15 primitives, as seen in Figure 5.21. The small numbered arrows indicate onset points of the relevant primitive. They are shown to give an impression of the extent of superimposed overlap of the primitives, and how some regions of the character require more primitives than others.

to the dataset upon which they were learnt. Also it shows that the learning procedure helps with generalisation of the primitives.

To show that the primitives extracted from characters are common to all handwriting, a dataset of simple scribbles was created (see Figure 5.25). Primitives extracted from this dataset are shown in Figure 5.26. These primitives can also be fit to a character based dataset, as seen in Figure 5.27. The VLL of this reconstruction was greater than that that of a reconstruction using randomly initialised primitives, showing that the primitives used for character production are likely to also be used for other activities, such as scribbling or drawing, and that the primitives inferred from these activities should also be useful, if not sufficient, for modelling writing.

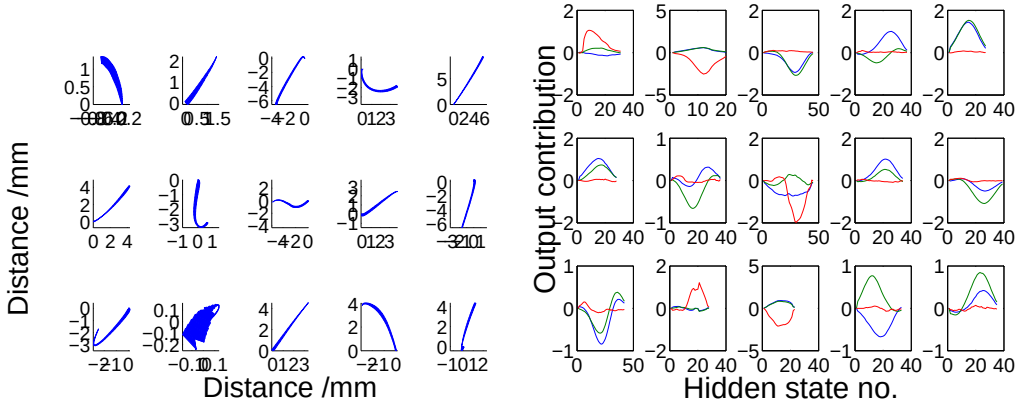


Figure 5.21: Primitives learnt from varied dataset, containing 12 character types. Reconstructions of the dataset can be seen in Figure 5.20.

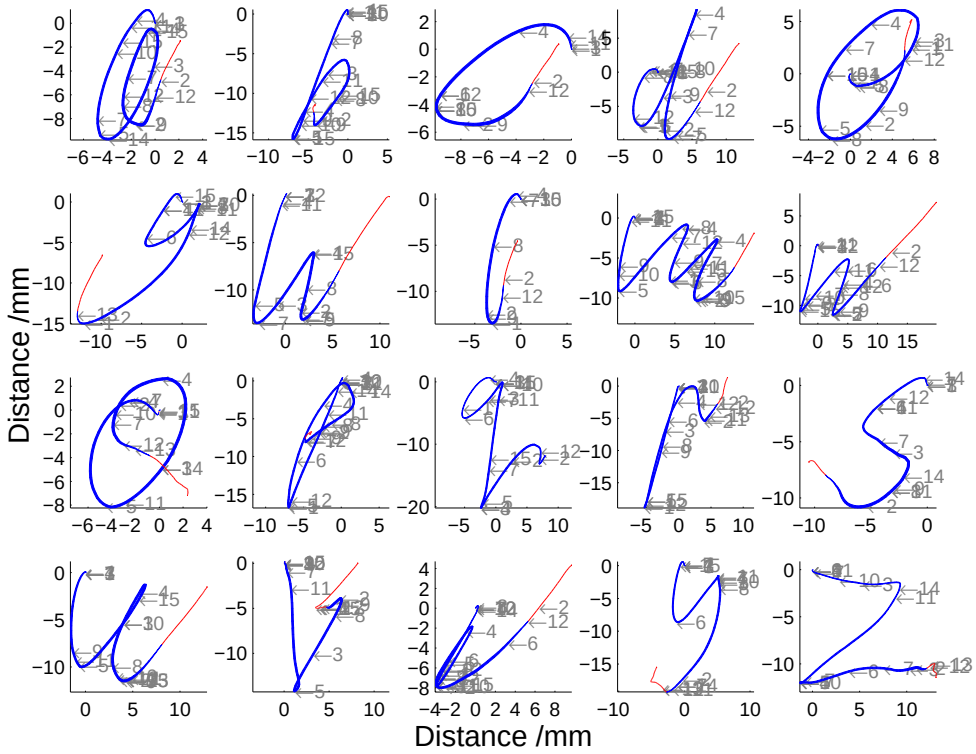


Figure 5.22: Reconstructions of a novel dataset of 20 character types, containing 8 character types not present in the training set. The primitives used can be seen in Figure 5.21.

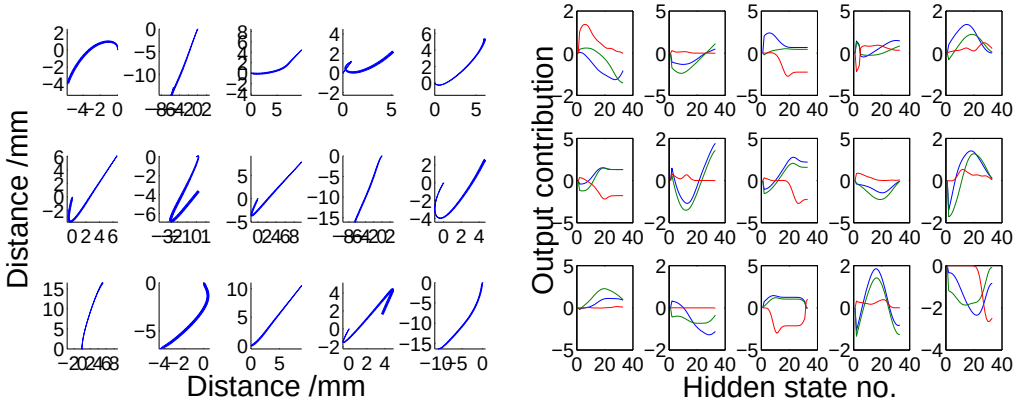


Figure 5.23: Unlearnt set of primitives initialised from the complete dataset, containing 20 character types. The length of each primitive is equal to the average lengths of the primitives seen in Figure 5.21. It should be noted that the combined length of all the primitives is considerably shorter than the combined length of all the character types.

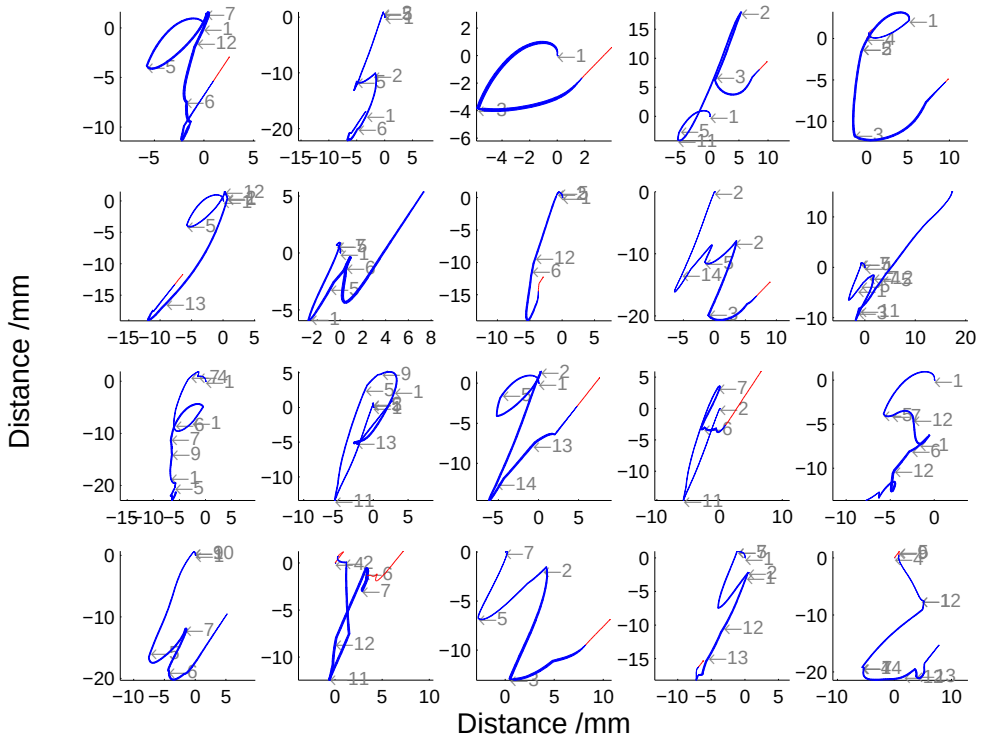


Figure 5.24: Attempted reconstruction of character samples from the complete dataset containing 20 character types, using 20 primitives taken from the dataset. This reconstruction is produced after a single E-step, similar to the data shown in Figure 5.22. Comparison of the reconstructed samples from these two datasets suggests that the learnt primitives generalise better to novel character samples (Figure 5.22), than primitives taken from an average of the data (This figure).

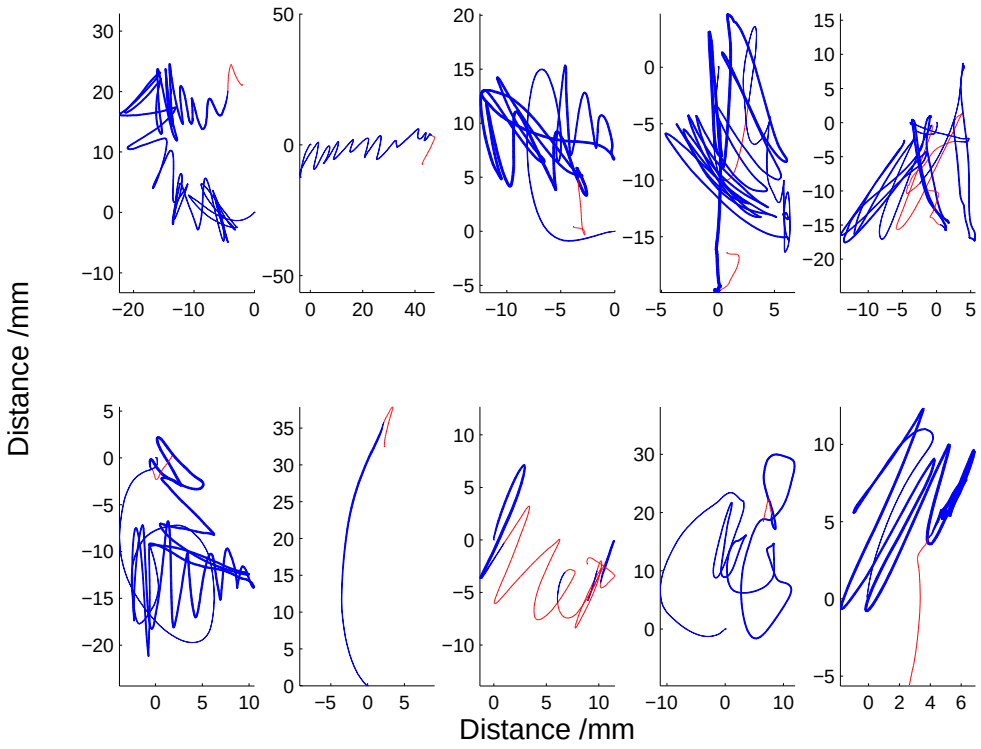


Figure 5.25: Reconstructed samples from the scribble dataset, showing the varied nature of the different samples.

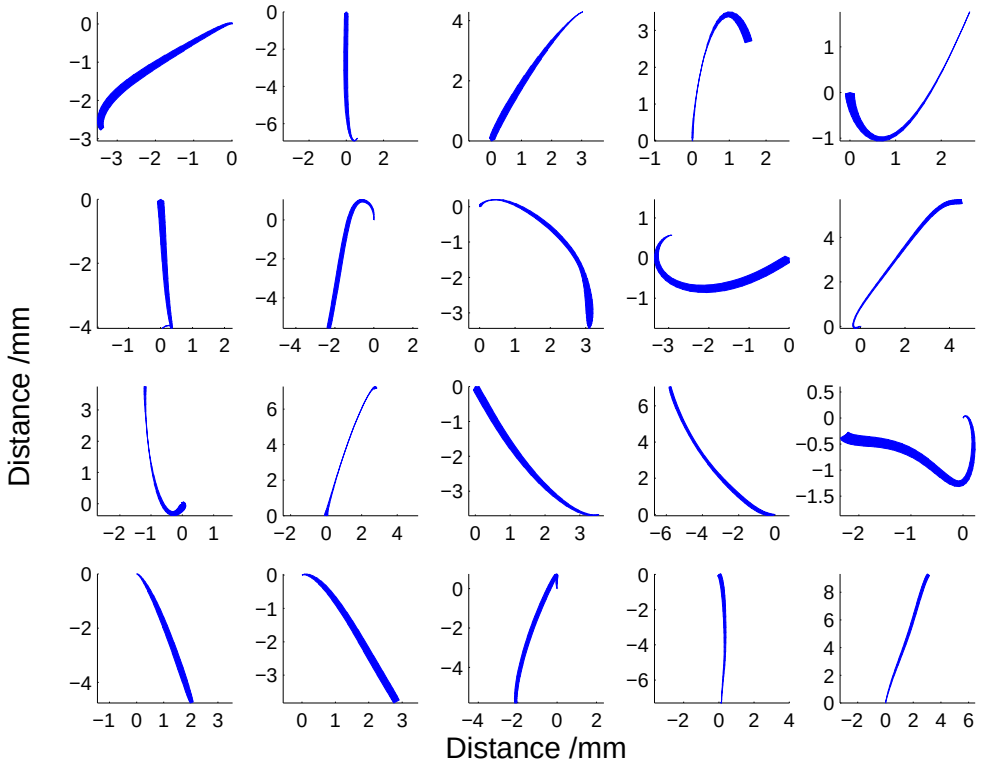


Figure 5.26: Primitives extracted from the scribble dataset, seen in Figure 5.25. The primitives are plotted as reconstructions on paper.

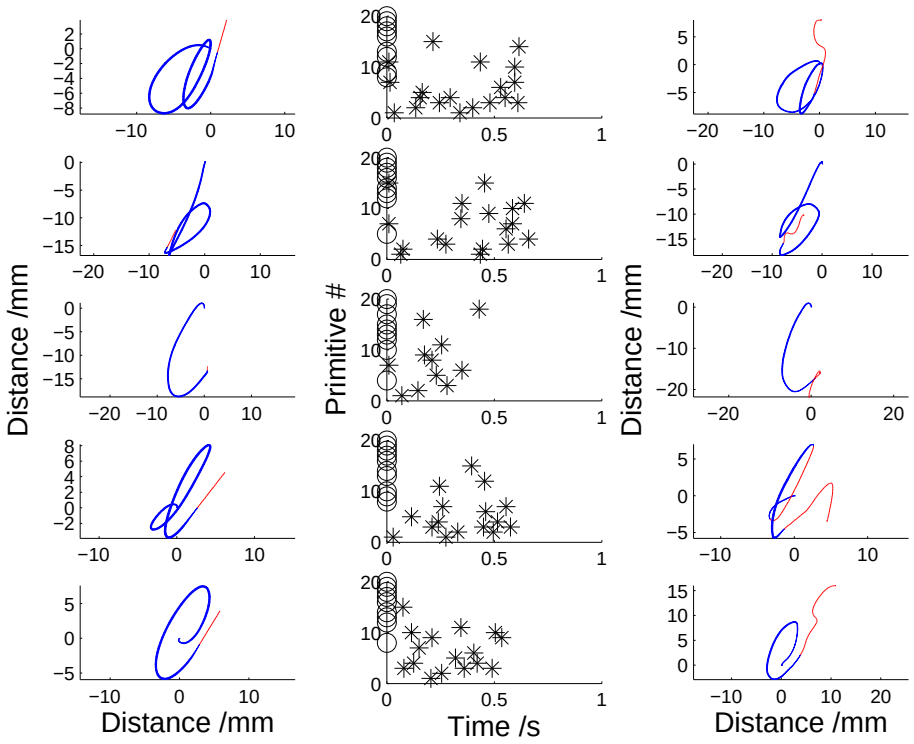


Figure 5.27: Scribble primitives can be fit to single character samples, as seen here. The original data samples are shown on the left, whilst the primitive timings, and character reconstructions shown on the right. These reconstructions are produced after a single E-step of the algorithm, using the primitives shown in Figure 5.26. The reconstructions show that primitives learnt from scribbles are suitable for character production even without adaptation. Although after a few EM iterations, the fit would improve.

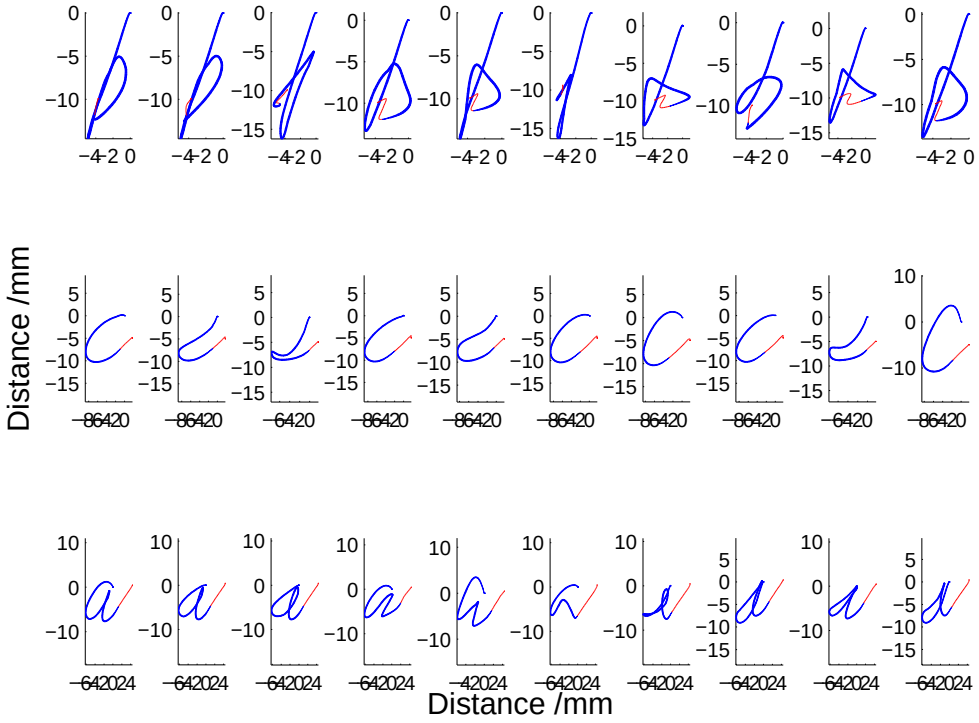


Figure 5.28: 5 samples of character reconstructions from the $\{a, b, c\}$ set. On the left are the original reconstructions, while in each subsequent column are the same reconstructions, but with progressively more primitive timing noise added. The noise is Gaussian, and the variance ranges from 1% of the total spike variance, up to 10%. The total spike variance is high, as the clustering of spikes was not taken into account when calculating this naive measure. Although the noise does sometimes affect the reproduction, most samples are readable, showing a robustness of the model to timing noise (or a robustness of our recognition capabilities to dealing with this particular form of noise).

5.6 Character reconstructions and model robustness

One assumption in our model is that the variation in handwriting is due, at least in part, to the timing variation of the primitive activations. Therefore, it is assumed that noise added to primitive timings should not greatly affect the reconstruction of a character. Figure 5.28 shows character reconstructions, with progressively more primitive timing noise added. This shows the model is at least to some extent robust to timing noise. In Chapter 6 we examine the variance of individual primitives, and whether certain primitives have more timing variation than others.

5.7 Spike timing

The primitives extracted using the model described in Chapter 4 can reconstruct the dataset, given the appropriate timing information, as demonstrated throughout this chapter. The process of character reconstruction produces a set of primitive onset timings or spikes. These spikes are approximations to the full posterior distribution over the hidden states. Ignoring any self transitions in the hidden states, if allowed, the spikes are derived from a Viterbi alignment of the hidden states, given a set of primitives. The posterior distribution is referred to as $Q(\mathbf{S})$, so let the Viterbi alignment of this be $Q_{\text{MAP}}(\mathbf{S})$. Therefore the spikes that correspond to the onset of primitives are taken from $Q_{\text{MAP}}(\mathbf{S}_t^m = 1)$, where the first non-rest state of the primitives is indexed as 1. Samples of the spike timing data produced can be seen in Figure 5.27 in the central column.

5.7.1 Generative sampling

So far the model has been used as a framework to extract likely primitives given a particular dataset, however, as the model is a generative one, it can also be used to produce data samples. In this case, the hidden states are sampled from the prior distribution, which is dependent only upon the previous time step, as it is a first order Markov model. These samples have the form of scribbles, capturing only an aspect of the character the model was trained upon, as seen in Figure 5.29.

The generative behaviour of the model gives a good impression of what the type of prior distribution the model captures. In this case, the model is capturing a prior over pen movements which look locally coherent, and this prior is perhaps suitable for representing scribbling behaviour. For a character, the local coherence of the primitives is not enough however, and therefore to make a generative model of characters we must have a model of the timing of the primitives. This is addressed in Chapter 6.

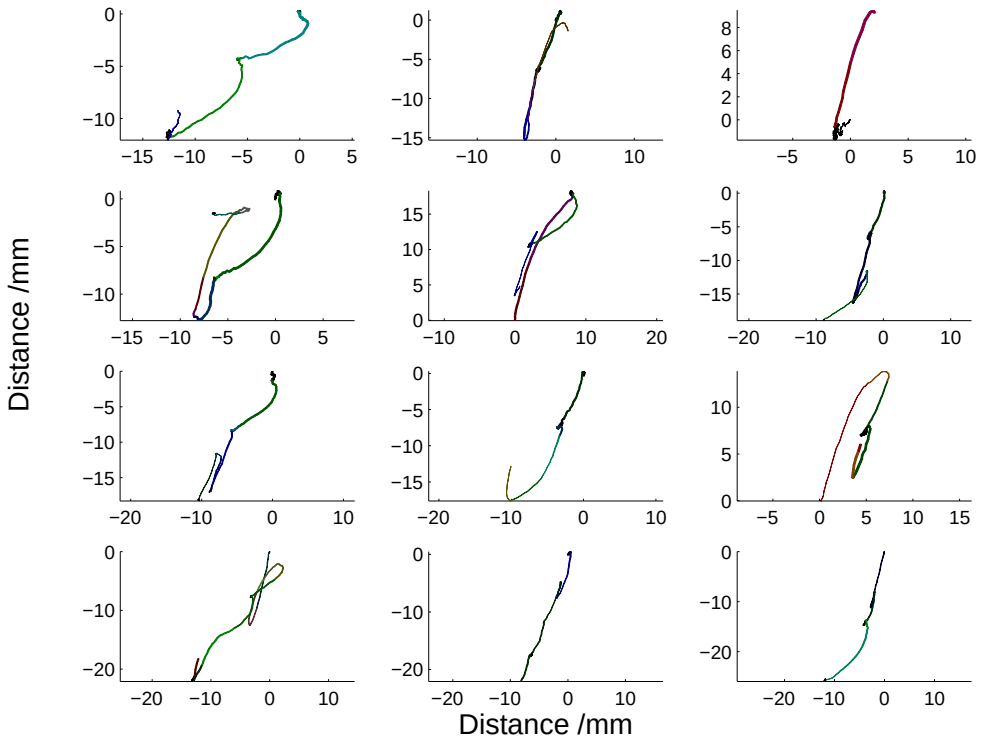


Figure 5.29: Generative samples from the uncoupled model, showing scribbling behaviour. The parameters were learnt from the $\{p\}$ dataset, and corresponding reconstructed samples can be seen in Figure 5.9 showing the importance of global primitive timing information to the reconstruction of a character.

Chapter 6

Timing Model

Chapters 4 and 5 have dealt with the low level model, which provides a prior over what the pen can do, with short term time dependencies. For producing a generative output that resembles a human controlling a pen, but not actually writing a coherent character, the low level model is suitable. To extend this model so that a generative model of character production is possible, we need to examine the timing of the primitives, and generalise over different samples of the same character. This chapter focusses on the primitive activation, or spike timing data that arises from the hidden state representation in the low level model. Some areas explored in this chapter have been published in (Williams et al., 2007).

6.1 Primitive-onset representation of the data

In Chapter 5, the data being modelled was the first differential of the pen tip position, and pen tip pressure. Sampled trajectory information of this kind is a very inefficient way of representing the character being drawn. If the pen trajectory arises directly from some inherent set of motor programs, or motor primitives, as suggested in Chapter 2, then a more efficient form of representation would take into account these primitives. If the output is built upon a fixed set of primitives, as described in Chapter 3, then the data can be transformed into the space of primitive activation timings, as described in Section 5.7.

To illustrate this transformation, some reconstructed character samples can be seen in Figure 6.1, along with the primitive activation times for each sample, represented in the form of spike timings. These spikes can be pooled across all character samples, and the scatter plot of these spikes can be seen in Figure 6.2, showing the occurrence

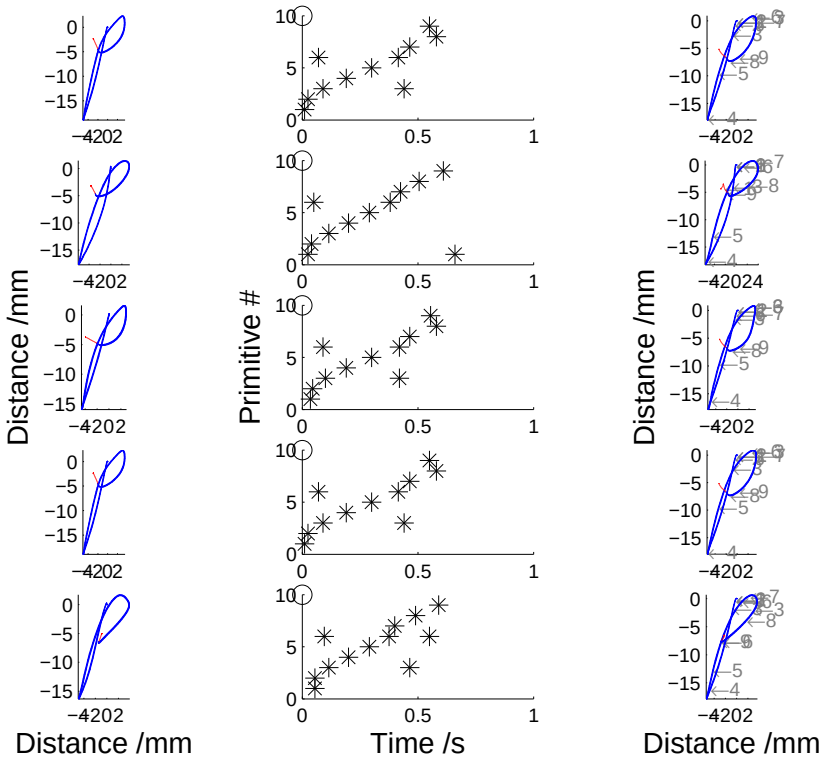


Figure 6.1: Five reconstructions of the character *p*. Leftmost column shows original data, and rightmost column the reconstructions. The central column shows the activation times of the primitives. These activation times can be stored as spike times, and act as a code for the particular character reconstruction, given a common set of output primitives.

of a primitive is likely to occur in a particular part of a character, as would be expected in a consistently produced dataset.

Without modelling this timing information, this is nevertheless a generative model of pen control, but requiring top-down, task-specific timing information to produce a useful output. The lower level model can be thought of as placing a prior over what the pen can do. To produce generative samples of characters, a model is required that learns a prior over the timing information for a specific character type.

An obvious simple approximation to the spike timing distributions would be a Mixture of Gaussians model, however there are two problems that limit such simpler models. Firstly, the number of spikes for a particular primitive may vary across samples. Secondly, the presence of a spike at time t is heavily dependent upon the presence at time $t-1$, due to the fact that if a primitive becomes active, then it cannot become active again until it plays out its entire length.

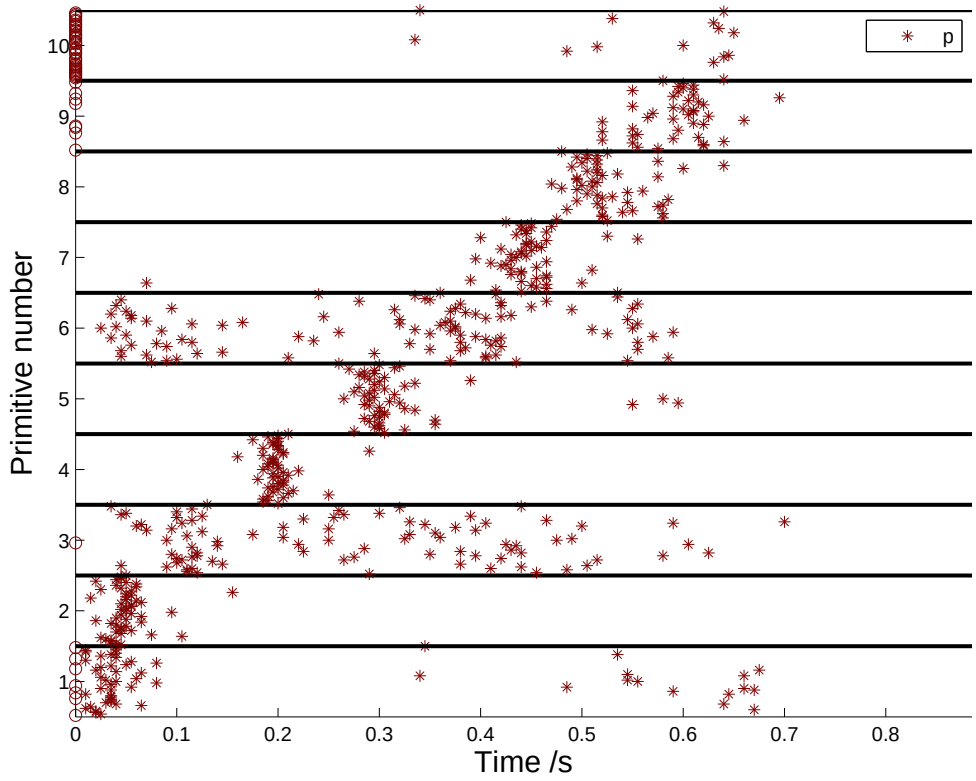


Figure 6.2: Scatter plot of all the spikes for a complete reconstruction of a *p* dataset, containing 50 character samples. Ten primitives were used for this particular reconstruction. The scatter plot is divided into 10 rows, for each primitive. Samples without any spikes are shown with a $\circ$ at $t = 0$. The sequential nature of the activation times is due to the way the primitives were initialised from sections of the data.

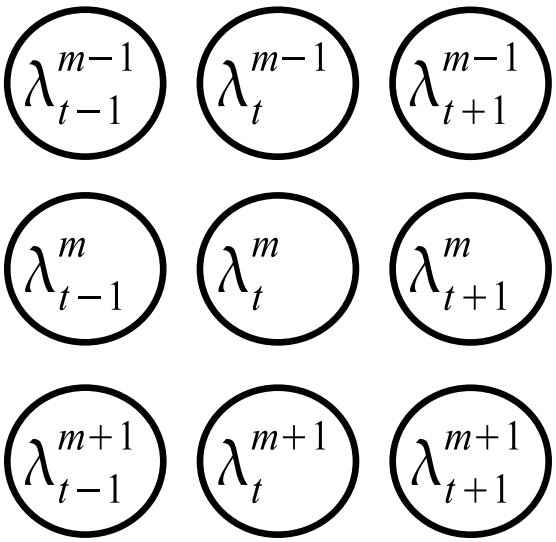


Figure 6.3: Graphical model of an independent spiking model. Each node represents a binary stochastic variable, indicating the presence of a spike. There are no dependencies or hidden states, making the joint distribution over the expected states a product of the marginal distributions, which are simply the average number of spikes observed over the sample size.

6.2 Potential timing models

Several different models have been explored, each with various benefits and shortcomings. This chapter looks at the details of some of these models, along with their performance in terms of character generation.

6.2.1 Independent spiking model

The simplest model of the spike timings is that in which we consider the presence of a spike at time t to be independent of the presence at other time points,

$$P(\lambda) = \prod_{t,m} P(\lambda_t^m) = \prod_{t,m} \mathbf{P}_t^m \,, \tag{6.1}$$

where λ_t^m is a stochastic binary variable indicating the presence of a spike at time t in primitive m and λ is the collection of λ_t^m. Figure 6.3 shows the simple graphical representation of this model.

This model is very easy to learn, as the parameters, $\mathbf{P}_t^m$, are simply the expected probability of seeing a spike, $\bar{\lambda}_t^m$, which is the average of λ over all samples. There are no hidden states, so there is no iterative learning procedure.

The expected distribution, $\bar{\lambda}_t^m$ can be used to parameterise the fHMM model, producing samples such as seen in Figure 6.4. This timing model is quite bad due to the

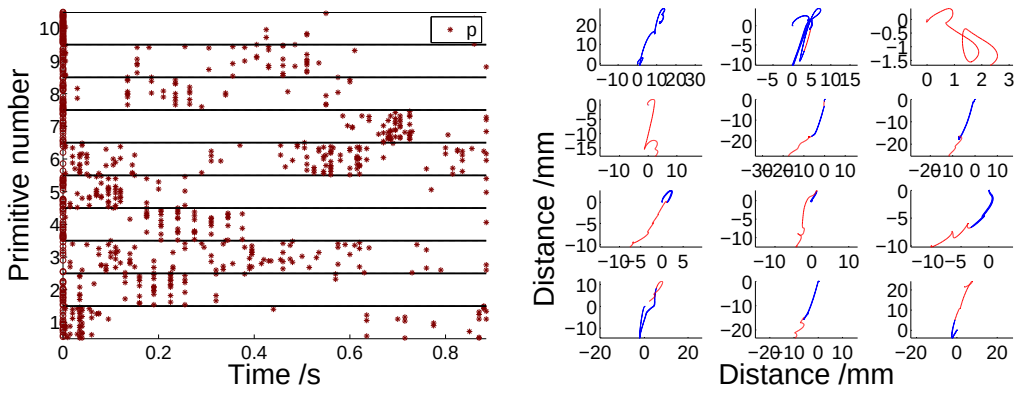


Figure 6.4: Generative samples from the independent spiking model. The scatter plot of spike times is shown on the left, and some samples of character reproductions are shown on the right, showing the inability of this timing model to produce legible character samples.

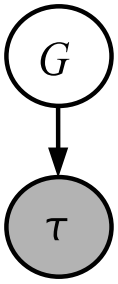


Figure 6.5: Mixture of Gaussians (MoG) model. **G** enumerates a hidden component, which determines the distribution over the continuous, observable variable τ.

time independence assumption, which is clearly not true in the spike data. Furthermore, this model has $M \times T$ parameters for each character type, meaning the efficiency of such a model is quite low; in fact, the space requirements are similar to that needed to store T samples, where T is the maximum length of a sample. This means that over fitting is likely for small sample sizes. Although this model is very fast to learn, its potential for providing a useful prior for spike position is not very promising.

6.2.2 Gaussian model

The spike distribution seen in Figure 6.2 suggests that the spikes are clustered into multiple clusters within each primitive. Intuitively therefore, each such cluster could be reasonably modelled with a Gaussian distribution.

There are two possible ways to represent the spike timings with a standard Mixture of Gaussians (MoG) model. The first option is a single Gaussian distribution, where for a particular character sample each spike is treated as a separate dimension of a

continuous multivariate Gaussian distribution. The problem with this model is that the number of spikes per character sample is inconsistent, thereby altering the dimensionality of the Gaussian distribution, and forcing an approximation such as several Gaussians of different dimensions. Alternatively, the spikes can be pooled, and treated as independent samples, from a one-dimensional mixture of Gaussians. The second option will be examined here.

Using this model, the original separate samples must be pooled, and a single sample, for the purposes of learning the model, be considered to be a single spike. Figure 6.5 shows a MoG model, consisting of K components,

$$\begin{aligned} \mathbf{G} &\in \{1,...,K\} \\ \tau &\in \mathbb{R} \ , \end{aligned} \tag{6.2}$$

therefore, the marginal probability density over the output, $P(\tau)$, is defined by a weighted sum of these components,

$$P(\tau) = \sum_{k}^{K} P(\mathbf{G}=k) P(\tau \,|\, \mathbf{G}=k) \ , \tag{6.3}$$

where each component is a discrete normal distribution,

$$P(\tau \,|\, \mathbf{G}=k) = \mathcal{N}(\boldsymbol{\mu}_k, \sigma_k^2) \ . \tag{6.4}$$

The parameters $\boldsymbol{\mu}$, σ^2 and the prior distribution over the components, $P(\mathbf{G})$, are learnt using a standard EM procedure.

Generatively, as there is no distribution over the number of spikes per sample included in this model, the easiest way to sample spikes, and disallow multiple spikes from the same component is to sample from each component in turn, with the presence of a spike from component k equal to $P(\mathbf{G}=k)$. If a spike is present from component k, this component parameterises the output distribution $P(\tau \,|\, k)$, being a single Gaussian, which can then be sampled from, producing a spike time. This is repeated K times. This approximation relies on the probability of presence of a particular component being the same in any one sample as it is across the whole dataset, and that there can be no dependence between components. Samples of characters generated in such a manner can be seen in Figure 6.6, with the same dataset containing a single character, p is shown, along with another dataset containing 3 characters, $\{a,b,c\}$. The generative samples for the dataset with a single character are much better than those from the mixed dataset, as the primitives used in the mixed dataset need to be shared between

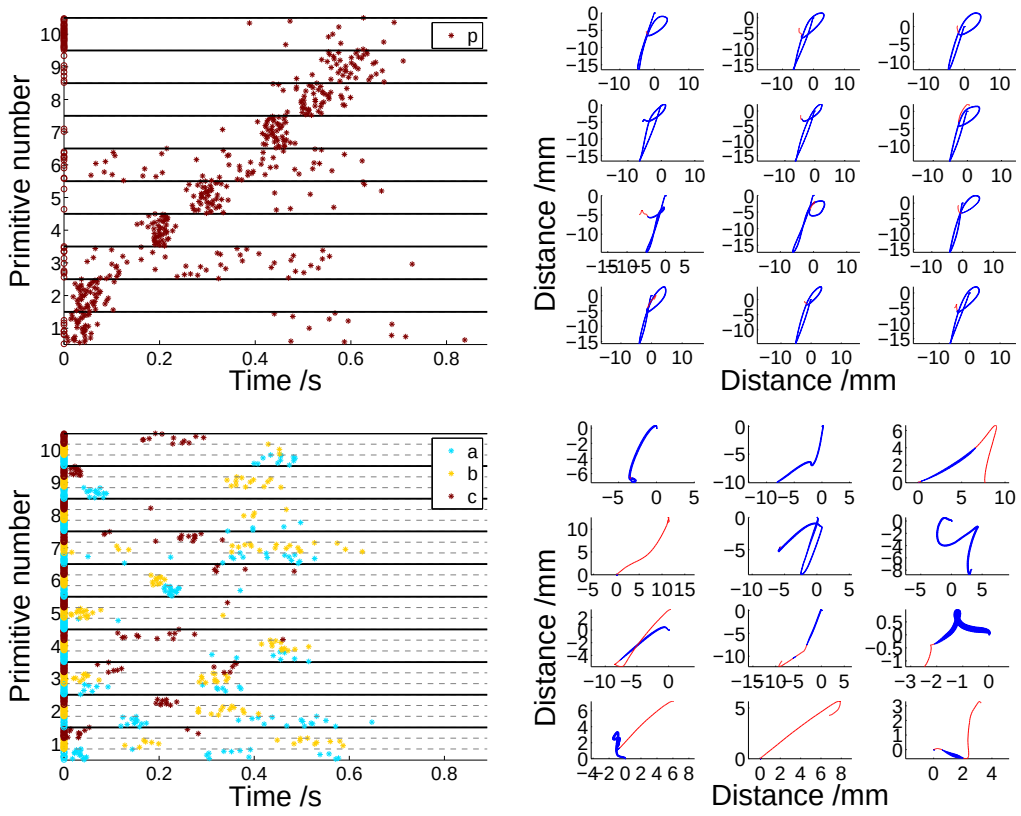


Figure 6.6: Samples from the MoG timing model. Generative spikes are shown on the left, and character reconstructions are shown on the right. The top row shows the model fit to a dataset consisting of a single character, which was easier to model than the dataset shown below, consisting of 3 characters, and requiring a more complex timing model.

different characters, and require greater coordination in their activations. Also, the primitives learnt for the single character tend to be used once only during the character production, simplifying the timing model.

The main problem with the MoG model is that each spike is sampled independently of the others, meaning that more than one spike can come from the same component. This does not occur in the data, as once a particular primitive has started, it cannot recommence until it has 'played out' its entire length. The model can be sampled in a slightly different manner, as described above, disallowing multiple spikes from the same cluster, however the data upon which the model is learnt still must be pooled, losing the spike ordering information present in a single sample. Sampling from the model in this way is effectively assuming that each cluster is only being fit to spikes from separate samples. As this is not a constraint of learning the model, there is no guarantee that this is true. One Gaussian may have been fit to multiple spikes from the

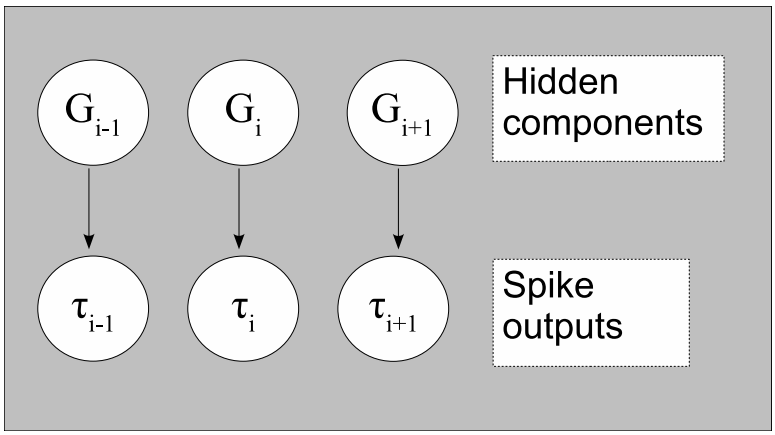


Figure 6.7: Set of Gaussians model. Components are hidden binary stochastic variables representing presence or absence of an associated spike.

same character sample if the spikes are considered as separate samples.

An alternative Gaussian model is one with multiple, independent Gaussians, or a set of Gaussians (see Figure 6.7. Each Gaussian component, *i*, has a hidden state indicating the probability of presence of a spike from that cluster, $P(\mathbf{G}_i)$, where $\mathbf{G}_i \in \{1,0\}$. Only spikes from components in state 1 are then sampled. Such a model is difficult to learn, as the spikes are not labelled with which component they come from.

6.2.3 Integrate and Fire model

An Integrate and Fire (IF) neuron is a different type of model, that takes some inspiration from biological models. A stochastic leaky IF neuron was used to produce samples of spikes. The Integrate and Fire (IF) model originates from simplified models of biological neurons. It treats the neuron as a leaky capacitor, upon which a charge is built up by the inputs, over time. Once the voltage across the capacitor reaches a threshold level, the neuron fires, producing a spike at its output, and discharging the capacitor. This means that, due to the leak term, over a long time scale, the inputs at different times are independent; however, on a short time scale, they are not, as it is the short-term running sum of the inputs that causes the neuron to fire. This is desirable for the primitive model, because the timing of a necessary primitive can be variable in the character samples; however, the IF neuron will still fire as long as it receives enough inputs during its temporal memory window. A probabilistic model of an IF neuron includes a noise term in the threshold.

The most straight forward model using IF neurons is to attribute one IF neuron to one primitive. The inputs to the neurons will then determine the timing of the

primitives. For a particular primitive, m, the probability of a spike at time t, $P(\lambda_t^m)$ is given by:

$$P(\lambda_t^m|\lambda_{t-1}^m = 0) = P(\lambda_{t-1}^m) + I_t^m - L_t^m, \tag{6.5}$$

$$P(\lambda_t^m|\lambda_{t-1}^m = 1) = I_t^m - L_t^m, \tag{6.6}$$

$$L_t^m = \nu P(\lambda_{t-1}^m), \tag{6.7}$$

where I_t^m are the input excitations, and L^m is a leak term proportional to the accumulated probability. Therefore, given a common set of primitives, a character is defined by its *temporal excitation matrix*, I_t^m, which parameterises the IF model. This matrix is learnt from the spiking statistics of the character training set. In fact it contains exactly the same parameters as for the independent spiking model.

Samples of spikes and generative character samples from such a model can be seen in Figure 6.8. This model produces reasonable samples due to the inherent time dependence of the observation of a spike, and the near future of a spike observation. This simple model lacks a proper learning procedure and probabilistic output distribution. A formal probabilistic spiking neural network could be implemented, but would be computationally expensive to train, and it is unclear if it would offer any benefits over the simpler Markov models discussed later in this chapter. It would however be an interesting alternative to the approach taken in this project.

6.2.4 HMM Timing model

An extension to the set of Gaussians model presented in Section 6.2.2, is to include coupling between the hidden states. This transforms the model into a standard HMM, where the hidden state encodes cluster number, and the observable state encodes spike timing. Figure 6.9 shows a graphical representation of an HMM. A separate HMM is used for each primitive. The hidden state $\mathbf{G}_i \in \{1,...,K\}$ encodes cluster index. The final index, K, is a null state indicating the end of a sequence, without an associated spike being emitted. A single sample of spikes is produced from a single state sequence, ending in the null state. Only forward transitions are allowed in the hidden states, ensuring that between 0 and $K-1$ spikes are emitted per sample, and also that a maximum of 1 spike is emitted from each cluster. The number of spikes in a particular sample is I.

Formally, for a single HMM, the joint distribution over all the states is,

$$P(\mathbf{G},\tau) = P(\mathbf{G}_1)\prod_{i=2}^{I} P(\mathbf{G}_i\,|\,\mathbf{G}_{i-1})\prod_{i=1}^{I} P(\tau_i\,|\,\mathbf{G}_i)\ . \tag{6.8}$$

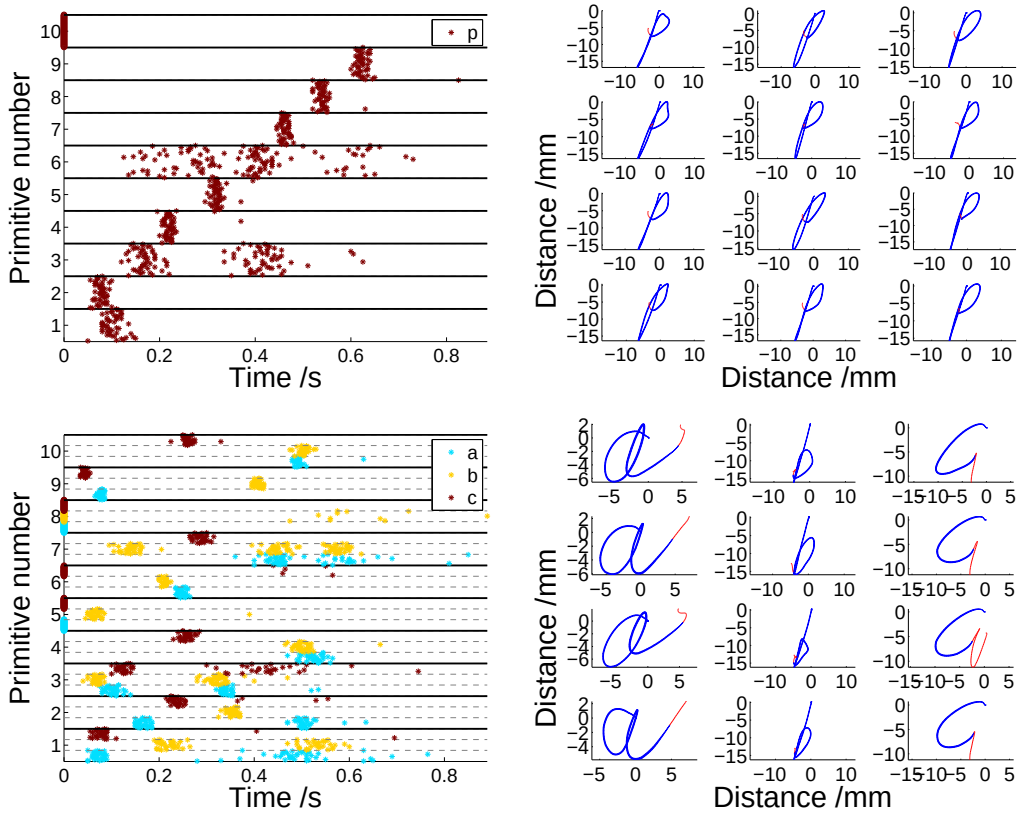


Figure 6.8: Samples from the Integrate and Fire timing model. Generative spikes are shown on the left, and character reconstructions are shown on the right. Generative samples from the Integrate and Fire model are clearly legible, even when there are multiple characters in the dataset, as seen here.

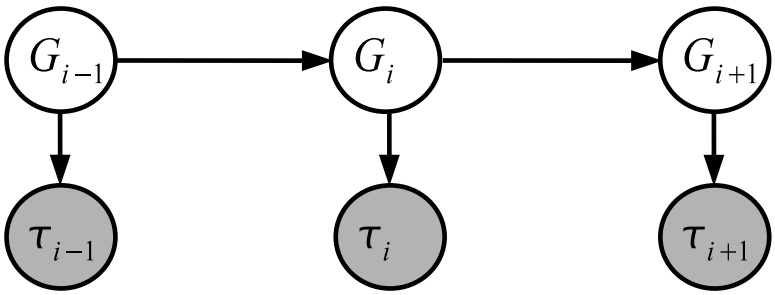


Figure 6.9: Standard Hidden Markov Model (HMM). Hidden states $\mathbf{G}_i$ enumerate primitive state, and parameterise the distribution over the continuous observable output state, τ_i. In the simplest case, there is a separate HMM for each primitive.

The parameters of the model are the output mean and variance, $\boldsymbol{\mu}_k, \sigma_k^2$, the initial prior, $P(\mathbf{G}_1 = k) = \boldsymbol{\pi}_k$, and the hidden state transition prior, $P(\mathbf{G}_i = k \,|\, \mathbf{G}_{i-1} = j) = \mathbf{P}_{j,k}$. The observable outputs are Gaussian distributed,

$$P(\tau_i \,|\, \mathbf{G}_i = k) \sim \mathcal{N}(\boldsymbol{\mu}_k, \sigma_k^2) \; . \tag{6.9}$$

The lower bound of the log likelihood function of this model is,

$$\begin{aligned}
\log P(\{\mathbf{G}_i\}) &\geq \sum_{\{\mathbf{G}_i\}} Q(\{\mathbf{G}_i\}) \log \left[\frac{P(\{\mathbf{G}_i, \tau_i\})}{Q(\{\mathbf{G}_i\})} \right] \\
&= \sum_k Q(\mathbf{G}_1 = k) \log \boldsymbol{\pi}_k \\
&+ \sum_{k,j} \sum_{i=2}^{I} Q(\mathbf{G}_i = k, \mathbf{G}_{i-1} = j) \log \mathbf{P}_{k,j} \\
&- \sum_k \sum_i Q(\mathbf{G}_i = k) \frac{1}{2} \log \left(2\pi\sigma_k^2 \right) \\
&- \sum_k \sum_i Q(\mathbf{G}_i = k) \frac{(\tau_i - \boldsymbol{\mu}_k)^2}{2\sigma_k^2} \\
&- \sum_k \sum_i Q(\mathbf{S}_i = k) \log Q(\mathbf{S}_i = k) \; .
\end{aligned} \tag{6.10}$$

Differentiating this quantity w.r.t. the parameters gives the standard HMM parameter update equations,

$$\boldsymbol{\pi}_k = \frac{Q(\mathbf{G}_1 = k)}{\sum_j Q(\mathbf{G}_1 = j)} \tag{6.11}$$

$$\mathbf{P}_{k,j} = \frac{\sum_{i=2}^{I} Q(\mathbf{G}_i = k, \mathbf{G}_{i-1} = j)}{\sum_k \sum_{i=2}^{I} Q(\mathbf{G}_i = k, \mathbf{G}_{i-1} = j)} \tag{6.12}$$

$$\boldsymbol{\mu}_k = \frac{\sum_i Q(\mathbf{G}_i = k) \tau_i}{\sum_i Q(\mathbf{G}_i = k)} \tag{6.13}$$

$$\sigma_k^2 = \frac{\sum_i Q(\mathbf{G}_i = k)(\tau_i - \boldsymbol{\mu}_k)^2}{\sum_i Q(\mathbf{G}_i = k)} \; . \tag{6.14}$$

The inference of the hidden state distribution is done with the standard Baum-Welch forward backward algorithm.

The advantage of this model is that it considers all the spikes from a single sample jointly, but without needing to place *a priori* constraints as to which Gaussian component they may have come from. Samples from this model can be seen in Figure 6.10.

There are two main problems with this model. Firstly, there is still no coupling between primitives, meaning that if a certain primitive tended to be used *in place* of

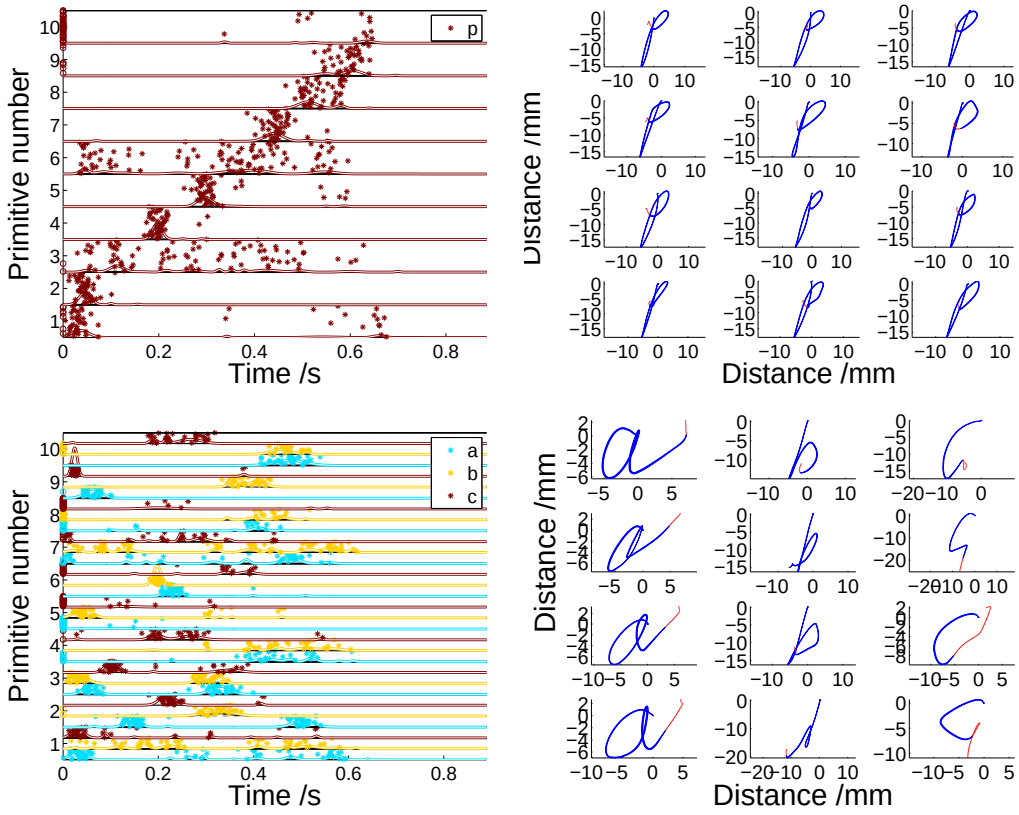


Figure 6.10: Samples from the HMM timing model. Generative spikes are shown on the left, and character reconstructions are shown on the right. This timing model works reasonably well for both the single character dataset, and the multiple character dataset. Problems occur due to the independence assumption on the primitives, causing a lack of correlation in primitive timing offsets for simultaneously triggered primitives, and no accommodation for *primitive substitution*.

another primitive, this anti-correlation in primitive presence would not be captured by this model. Secondly, the Gaussian outputs are independent, which is necessary because spikes are not present in all samples from all components, making it impossible to learn a joint Gaussian. This is a problem because there may be spike timing correlations between spikes from the same primitive, and also between spikes from different primitives. Two extensions to the basic independent HMM timing model were explored, and are detailed here. One uses a global offset to perform dynamic time warping, and the other attempts to introduce coupling between the presence or absence of primitives, by extending the model to a higher level, called character modes.

6.2.4.1 Dynamic time warping

To examine any correlations between spike timings, the offsets of the observed spikes from the model expectation can be seen in Figure 6.11. Some characters appear to have correlations between spike offsets in certain areas, suggesting that there should be some common offset that captures these correlations. It would be difficult to build such a parameter into the model, but a simple 'running offset' can be calculated by fitting a best fit spline function to the pooled offsets, as in Figure 6.11. By limiting the spline function to be a polynomial function of limited degree, over fitting can be controlled. Using such a time warping procedure to estimate the drifting time offset of each sample, and adjust the spike times accordingly before fitting the timing model improves the the fit of the model. Figure 6.12 shows the fit of the Gaussian distributions to spike data before and after time warping of the spikes. In this example, the sum of the variances of the output distributions was roughly halved due to the time warping providing a better fit.

The problem with using dynamic time warping is that the length of the characters tend to reduce, so that the total error of the fit is reduced. Also it further complicates the inference procedure by adding a further non-probabilistic part to the model, which means that the likelihood function is no longer valid from one iteration to the next. The technique is generally unsuitable to be used during inference, but it does provide a convenient way to reduce variance in the final model before generative samples are produced, if necessary.

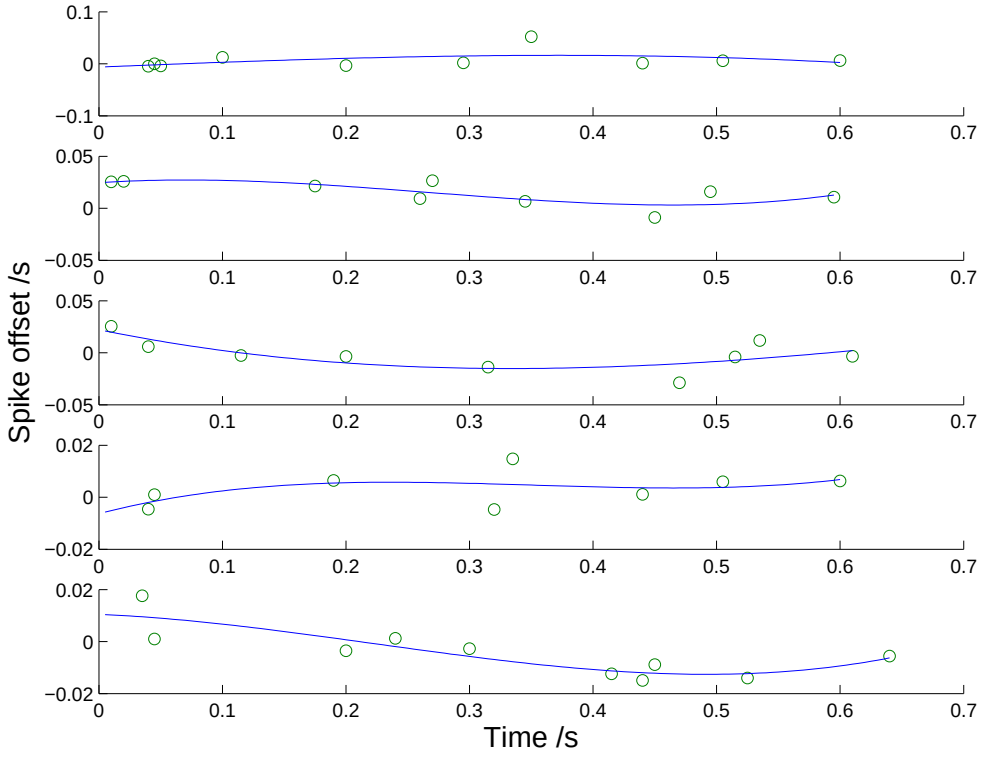


Figure 6.11: Scatter plots of spike offsets, for 5 character samples. Spike offsets from MAP cluster means are shown. Third degree polynomial best fits are also shown, which can be used as time warping functions for the spike times.

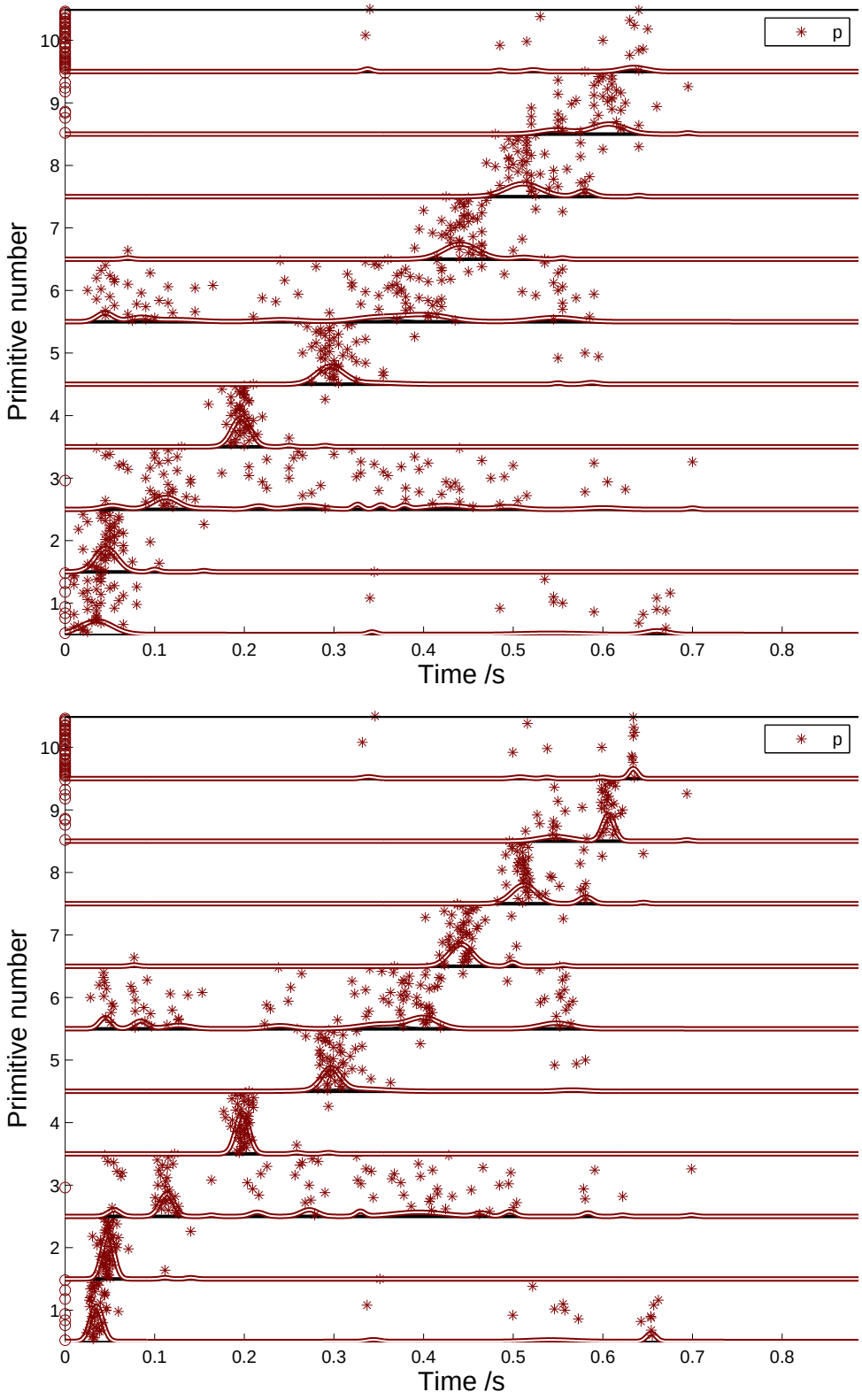


Figure 6.12: Effect of dynamic time warping on the fit of the HMM timing model. Top plot shows the fit of the model without DTW, whilst the bottom shows it included, both for the position of the spikes, and the resultant Gaussians for the model.

6.2.4.2 Character Modes

The hidden state trajectories of the M HMMs determines which clusters produce spikes, but not the exact position of these spikes. Therefore, in terms of the character produced, these hidden state trajectories determine not the exact pen trajectory, but the *form* of the character produced. This will be referred to as the *mode* of the character. The mode of the character is important, because although the HMM architecture captures local structure for the mode, it does not capture dependencies between non-neighbouring clusters. Moreover, it is assumed that the primitives are conditionally independent, therefore the modes of the different HMMs are not coupled. It is possible to calculate the modes of the different character samples in the training set by taking the Viterbi alignment of the hidden states of the HMMs. It is interesting to note that this is similar in procedure to the method of extracting the spike timings from the underlying primitive model, allowing re-representation of the data into spike timing space, as detailed in Section 6.1. Likewise, the modes could form a further level of abstraction. To examine this possibility, the easiest approach is to store all the modes for the training sample set. Each mode is a $M \times L_{max}$ matrix of integers, representing the hidden state trajectories of each HMM, as seen in Figure 6.13. Instead of generatively sampling the hidden state trajectories of the independent HMMs, a mode is selected at random, dictating the hidden states of the HMMs. The spikes are then sampled independently, giving rise to samples as seen in Figure 6.14. Although this method captures any correlations between presence or absence of components from different primitives for a particular dataset, it is unclear how to incorporate it into the generative model and to cluster modes together.

6.2.5 Combined hidden state HMM

The primitives in the fHMM model are considered to be conditionally independent, given the observable data. This is perhaps not the case, as certain primitives could be 'substituted' for others in certain characters, capturing specific areas of variance in that character. The dependencies in the HMM capture primitive presence dependencies between neighbouring clusters within a primitive. Another, and in some ways simpler, model would be a single HMM for all primitives. In this case, the hidden state represents the presence or absence of all the clusters. The observable probability density for a sample of spikes from primitive m is artificially limited to hidden states associated with that primitive. This model has been called a combined HMM, as the hidden state

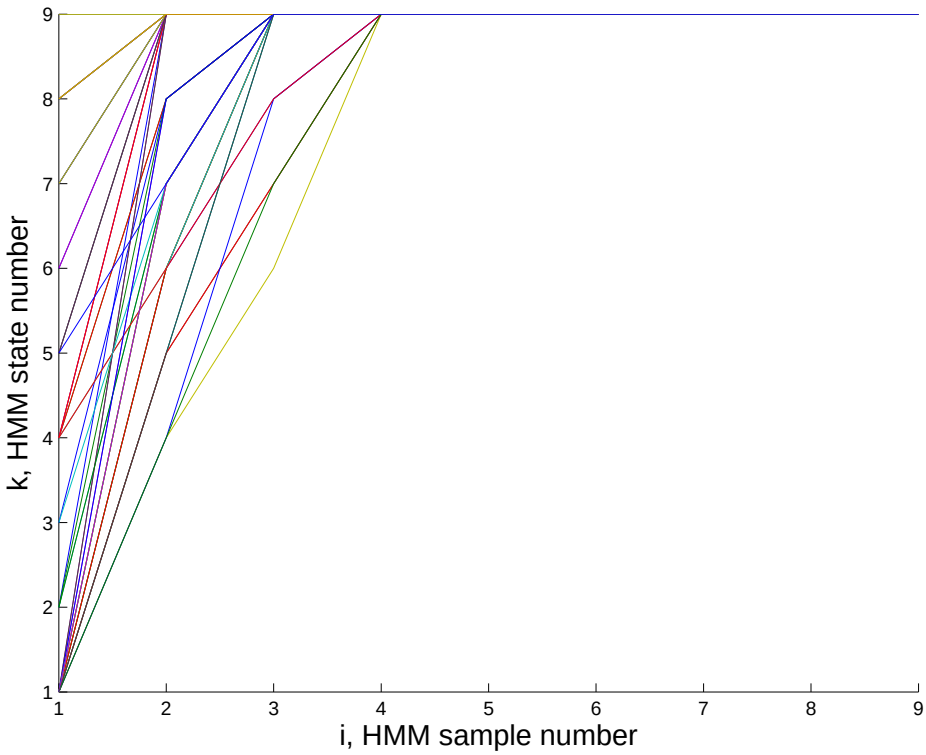


Figure 6.13: Overlaid plot of hidden state progression for 50 samples of the character *p*. Here one primitive is shown, demonstrating the variance across samples of which generative components are used for the reconstructions. In this case there are $K = 9$ hidden states, and the most number of spikes generated in any sample is $L_{max} = 4$. Although these trajectories might be representable using the Markov transition matrix, it does not capture any correlation with other primitives, referred to as 'primitive substitution'.

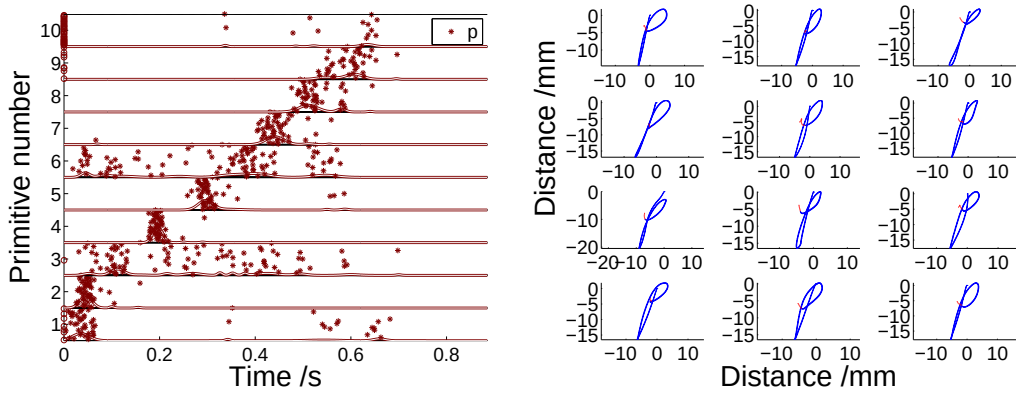


Figure 6.14: Samples generated from data-conditioned modes. The timing model is exactly the same as in Figure 6.10, except that in this case the hidden state trajectory of the samples is not generated, rather than inferred from the training data. The exact position of the spikes is still sampled from the Gaussian distribution.

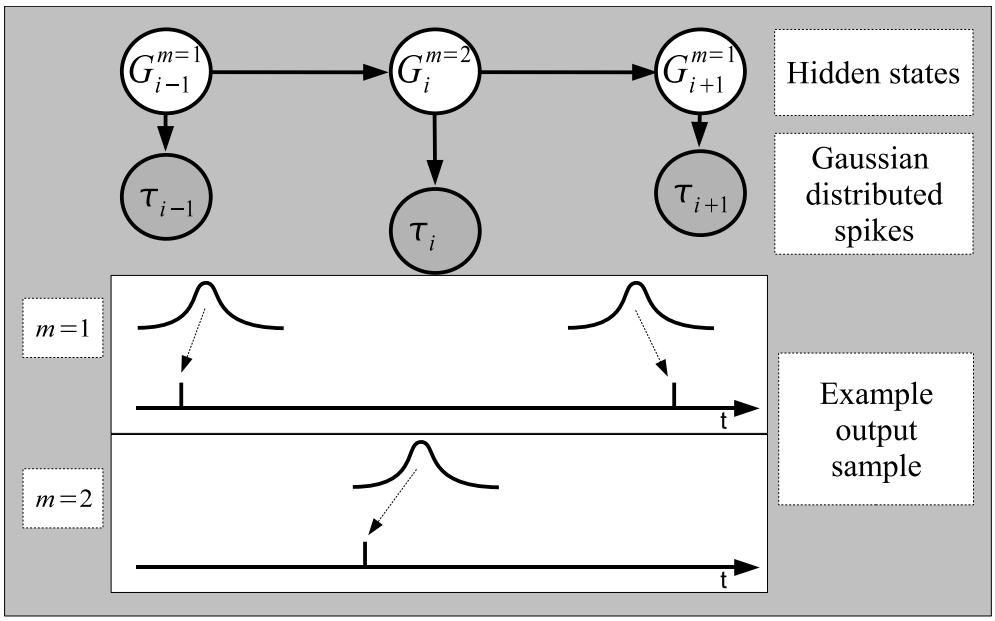


Figure 6.15: Combined Hidden Markov Model. Hidden states are concatenation of hidden states from all M HMMs detailed in Section 6.2.4. Each state k has a primitive associated with it.

represents a concatenation, or combination of the standard HMM states. A graphical representation of this model can be seen in Figure 6.15.

Therefore, the hidden state now has more possible values,

$$\mathbf{G} \in \{1,...,K\} \tag{6.15}$$

where K is the sum of the previous K^m's from the previous section (6.2.4). For each state k, there is an associated indicator constant, $\phi_k \in \{1,...,M\}$, recording which primitive the output cluster k is associated with. The observable conditional distribution is now,

$$P(\tau_i^m \,|\, \mathbf{G}_i = k) = \begin{cases} \mathcal{N}(\boldsymbol{\mu}_k, \sigma_k^2) & \text{if } \phi_k = m \\ 0 & \text{if } \phi_k \neq m\ . \end{cases} \tag{6.16}$$

This model only needs a single HMM, with a slightly modified calculation of the observable likelihood function, making it more compact than the separate HMMs model. Also, as the hidden states represent presence or absence of the clusters from all primitives, the model should better capture primitive substitution. Only substitution between neighbouring clusters is captured, as the Markov chain is first-order.

The problem with this model is that the spikes lose their ordering because there is no longer a guarantee that spike τ_i comes after spike τ_{i-1}, as they could be associated with different primitives. Therefore the forward only constraint on state transitions

must be relaxed for the model to be learnt, which makes the generative samples inconsistent, as a cluster might be activated more than once.

6.2.6 Single combined HMM

A subtly different HMM timing model solves the above problems, by dividing the parameters of the model into two sets. In Section 6.2.4 a timing model with M separate HMMs was described, therefore using M sets of transition parameters $(\boldsymbol{\pi}_k, \mathbf{P}_{k,j})$, and M sets of output parameters $(\boldsymbol{\mu}_k, \sigma_k^2)$. In Section 6.2.5, the hidden state was combined, implying a single set of transition parameters, however the output parameters were still associated with a particular primitive, limiting the possible transitions between states in a manner inconsistent with the constraints needed. This can be better understood by considering that the k's and the i's may not come in the same temporal order.

In this final model, called the single combined HMM, there are M different sets of transition parameters[1], but the output parameters are shared for all primitives and characters. Figure 6.16 shows a graphical representation of this model.

There is no longer a clear definition of the maximum number of components in the model, as only a subset of the components might be used for any one data sample. Therefore, to determine the number of components, K, a simple local maxima finding algorithm is used. Procedurally, all the spike timing data is pooled together, and tallied into bins, as if for a histogram. These tally counts are Gaussian smoothed, and then it is a simple matter to find local peaks in this function. An example can be seen in Figure 6.17. If the number of components inferred is lower than the maximum number of spikes in a single sample, then the resolution of the Gaussian smoothing function is increased (σ is decreased) and the method is repeated. Not only does this technique provide a useful estimate of the maximum number of components necessary to model the data, but it also provides an estimate of the positions of these components, which is used to initialise the parameters before performing EM on the HMM.

The original motivation for this model was from experiments in character recognition, which are described in Chapter 8. The benefit of this model is that each character type has the same number of components, i.e. has the same size model. This means that the log likelihood can be compared between character models without a bias towards one with a larger model size. The likelihood is calculated as if there were M separate HMMs, as there are M different transition parameters learnt for each primitive.

[1] In fact there are $M \times C$ transition parameters, where C is the number of character types in the dataset being learnt.

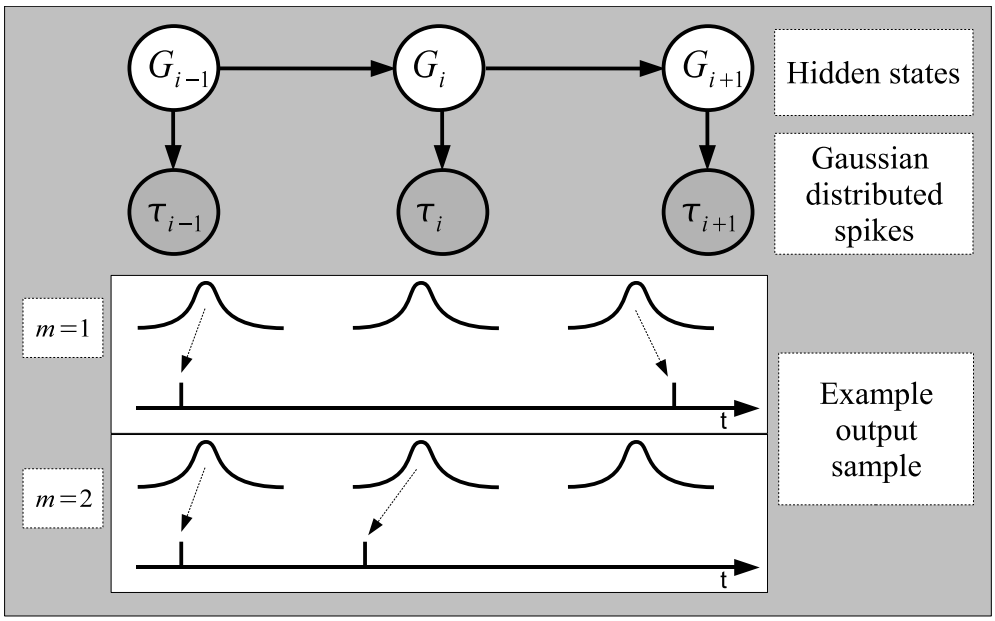


Figure 6.16: Graphical model of the single combined HMM model. Again, a single HMM, as in Figure 6.15, however, in this model, all the Gaussian components are possible generative components for all primitives, and separate transition matrices are learnt for each primitive and character type. This model works better as the spike ordering is retained, so the transition matrix can be constrained to have exclusively forward transitions.

In summary, in this model there is a single HMM model, with K hidden states, and a final rest state. For each primitive and each character type, there is a learnt prior $\boldsymbol{\pi}_k^{m,c}$ and transition matrix, $\mathbf{P}_{j,k}^{m,c}$ allowing only forward transitions through these states. Each state has a corresponding output Gaussian component, with a mean and variance that is the same for all characters and primitives, $\boldsymbol{\mu}_k$ and σ_k^2.

This model works well because the transition matrix can be constrained to only have forward transitions, therefore it can generatively produce a maximum of one spike per component, and although the presence or absence of a particular primitive is still independent of other primitives, the potential sharing of generative components between primitives allows limited coupling between temporally local spikes. This coupling is not accounted for during training, as the components are still one dimensional Gaussians, ignoring the covariance between primitives. During sampling however, coupling can be simulated by taking a single time sample from each Gaussian, and using this time sample for all primitives that use the component in question. Therefore, if a primitive is 'triggered' late, then other primitives that use the same component (i.e. are active at the same time, with the same modelling distribution) are also triggered late. Samples from this model can be seen in Figure 6.18.

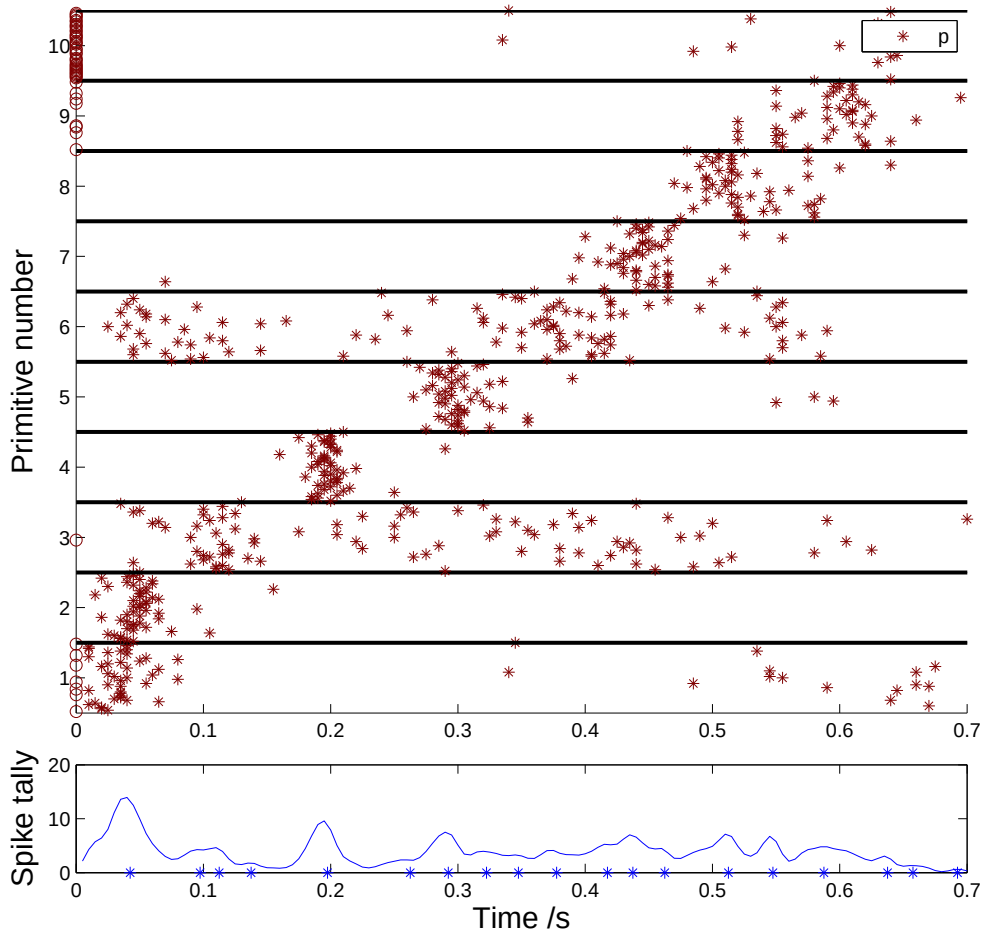


Figure 6.17: Scatter plot of all spikes for a dataset containing 50 samples of the character *p*. Below can be seen the smoothed histogram function used to initialise the number of components in the single combined HMM timing model, described in Section 6.2.6. Stars mark the initial positions of these components, which are then optimised using a process of EM on the whole HMM. In this model, it is possible for multiple primitives to share the same component, as is most likely the case for the majority of spikes in primitives 1 and 2 shown here. This allows limited coupling between primitives, and improves the generative samples.

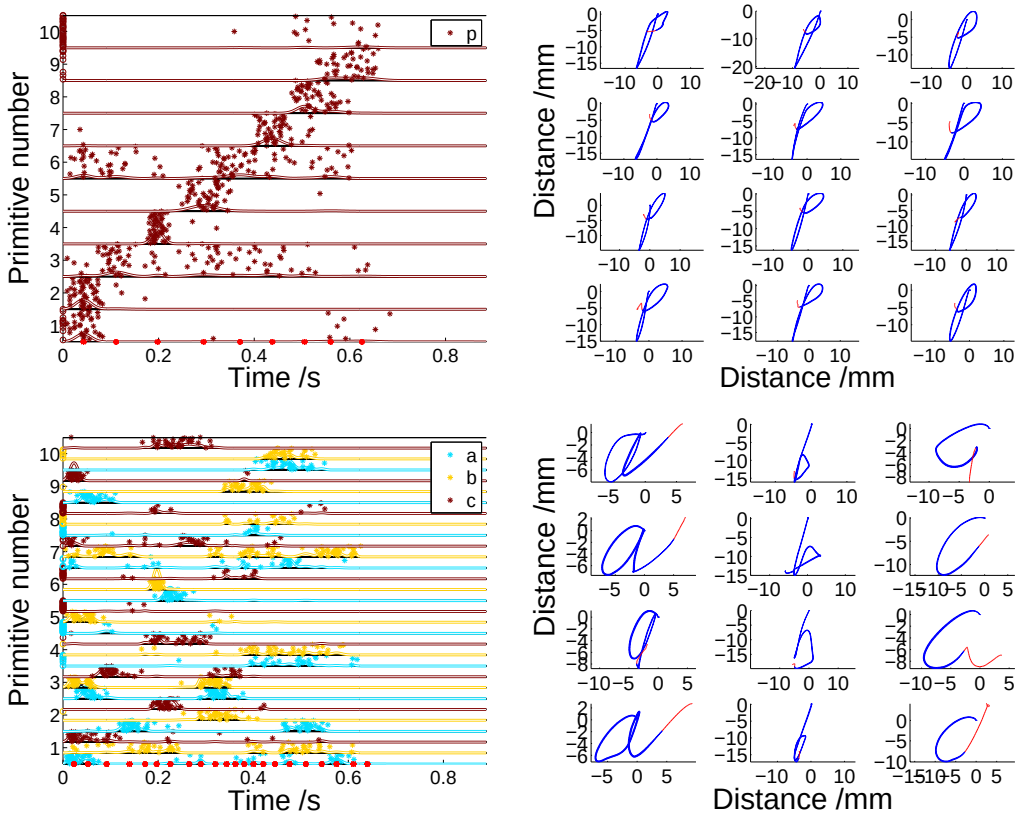


Figure 6.18: Samples generated from the single combined HMM timing model. The model is a single HMM, where the output components are shared between all primitives and characters. The hidden state transition priors are learnt separately for each primitive however, explaining how different classes of character can be produced. The top two axes show generative samples using 10 primitives to model a single character, whilst the bottom two plots show 10 primitives modelling 3 character types. In the scatter plots showing primitive onset timings, Gaussians are shown to represent the shared components, or spike clusters. Star markers are shown at the bottom of these plots to indicate the means of all of the components. The number of components used to model a single character in this case was 9, whilst for 3 characters, it was 23. The method by which this number is determined is described in Section 6.2.6.

6.3 Generative comparison

The generative performance of these models can be assessed in several different ways. Firstly, the most straightforward criteria, is simply the quality of the generative character samples produced. However, a simple mean trajectory of character samples would be a reasonable model by this assessment. The variation of the character samples is important, as it should capture the variation of the dataset. Furthermore, the models should be assessed on the size of their parameter spaces.

The dataset containing the character *p* and using 10 primitives, as used throughout this chapter is used here to test the different models for their generative mean and variance. The original data, as seen in Figure 6.1 has the mean and variance profile seen in Figure 6.19. Ten primitives were learnt as described in Chapter 4 without the use of a timing model. A single E-step infers the posterior over the fHMM, giving the primitive onset times or spikes. This posterior based reconstruction has the mean and variance profile seen in Figure 6.20, along with the original data, showing a good reconstruction. It should be noted that these data-conditioned samples are not generative samples. Generative samples are samples from the prior distributions of the model, as described below.

Without any timing model present, scribbling behaviour is produced, as seen in Figure 5.29 in the last chapter. The mean and variance profile from this behaviour can be seen in Figure 6.21, which as expected, does not match the dataset very well, except in average magnitude. The mean and variance profile for the independent spiking model as described in Section 6.2.1 can be seen in Figure 6.22, showing correspondingly poor profiles for this simple model. The timing model using Gaussian mixtures to model the spike timings as described in Section 6.2.2 produces better mean and variance profiles from its generative behaviour, as seen in Figure 6.23. The integrate and fire model as described in Section 6.2.3 produced very good generative samples, from a readability point of view; however, the mean and variance profiles seen in Figure 6.24 show that this model does not model the variance of the dataset.

There are three variations of the HMM timing model examined here. Firstly the separate HMM model, in which there are M independent HMMs modelling the M primitives, as described in Section 6.2.4. This model produces reasonable character samples from a single character dataset, and the mean and variance profiles of the samples are correspondingly good, as seen in Figure 6.25. Using the character modes as described in Section 6.2.4.2 to condition the hidden states of the separate HMM

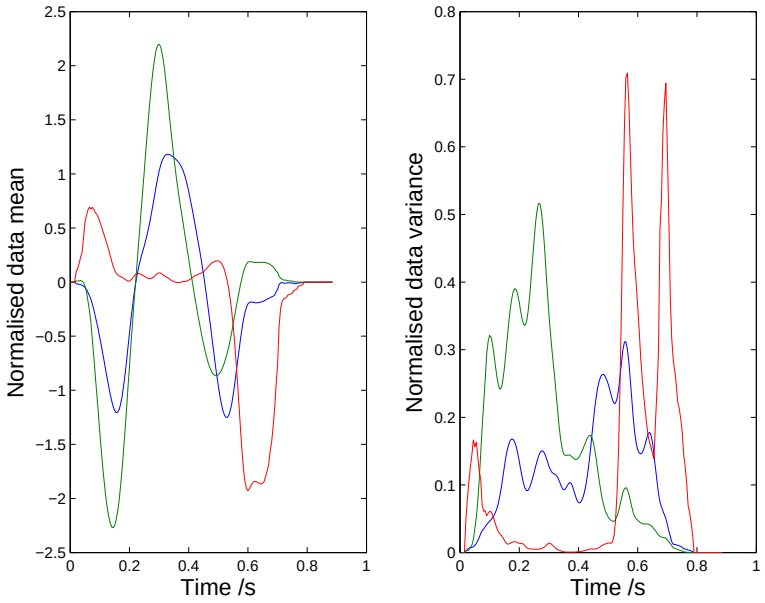


Figure 6.19: Mean and variance profile plot of the dataset containing a single character, *p*. Features in the variance profile do not correspond obviously to the mean character profile. The data has undergone a simple linear shifting process to better fit the samples together, which has the effect of tightening the peaks in the mean profile, and reducing the overall variance profile.

model improves the character samples along with the mean and variance profiles, as seen in Figure 6.26. However, as mentioned earlier, samples produced in this fashion are not truly generative, as the modes are effectively data-conditioned, and are not currently generalised with a suitable model. Finally, using the single combined HMM, as described in Section 6.2.6 to produce generative samples, produces reasonably good mean and variance profiles, as seen in Figure 6.27.

6.4 Timing constraints

There are two reasons that the samples produced might not be as good as the posterior reconstruction of the data. Either the timing variation of the primitives is not appropriate, or the wrong primitives are getting triggered in the generative samples. If we are in fact using too many primitives to try to model the data, the presence and timing distribution of a particular primitive may not be generalised across different samples. One way to address this is to reduce the number of primitives, thereby enforcing sparsity on the model. Figure 6.28 shows generative samples from the HMM model, but only using 5 primitives to model the data. With fewer possible primitives, the model learns primitives that are more generalised to the training set in question. This is because the

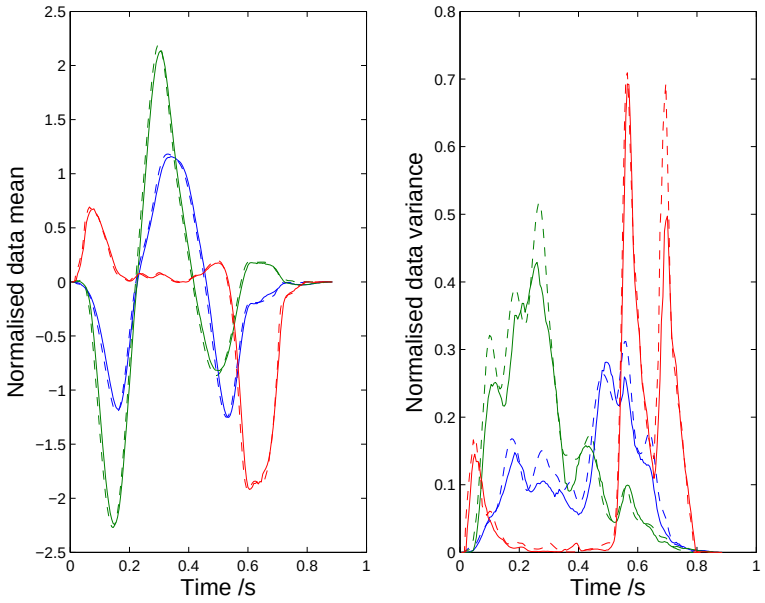


Figure 6.20: Mean and variance profile plot of the reconstructed dataset containing a single character, *p*. The original data is shown with a dashed line for comparison. The posterior based reconstruction produces a very close reconstruction of the dataset.

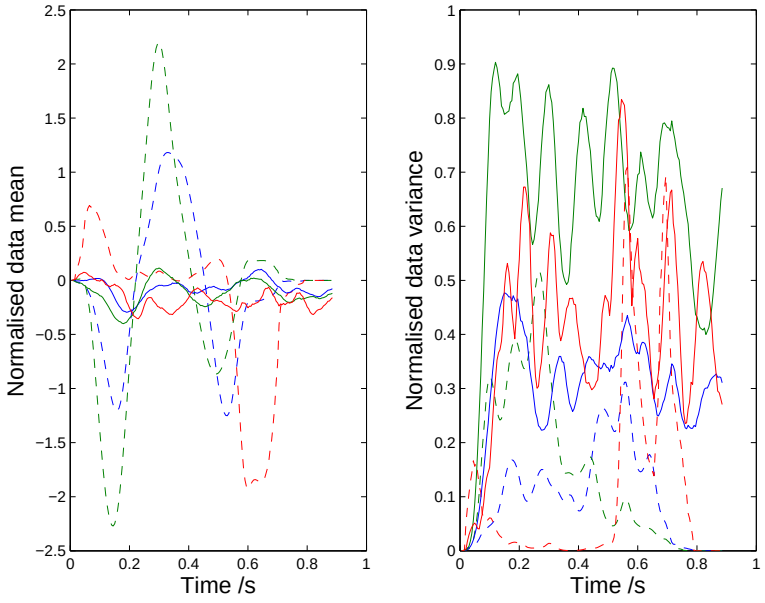


Figure 6.21: Mean and variance profile plot of the samples produced by sampling from the model, without any prior on where primitives should occur. The samples produced in this manner have the aspect of scribbles, and so intuitively they do not match the original data which is shown here with a dashed line for comparison.

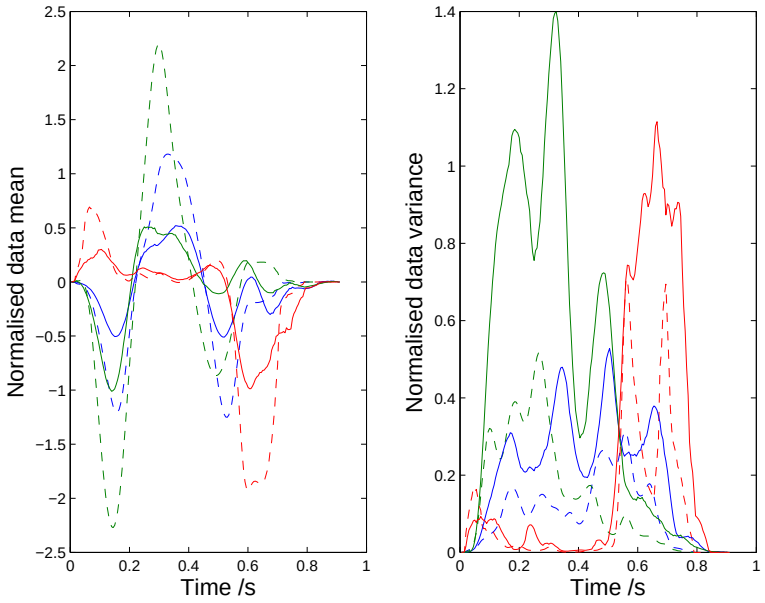


Figure 6.22: Mean and variance profile plot of the samples produced using the independent spiking model to model primitive onset times. The original data is shown with a dashed line for comparison.

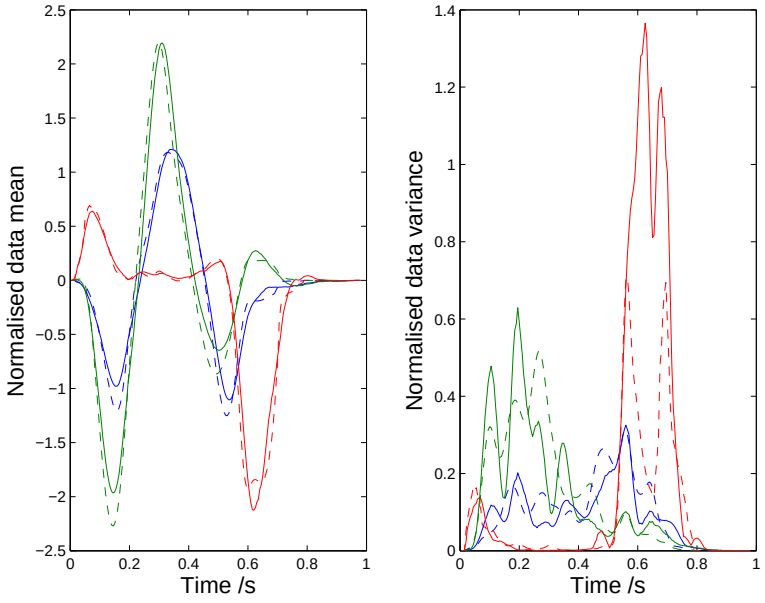


Figure 6.23: Mean and variance profile plot of the samples produced using the Gaussian mixture model to model primitive onset times. The original data is shown with a dashed line for comparison. The match is very close for this model to this dataset, however the model performs worse on a dataset with multiple characters, as described in Section 6.2.2

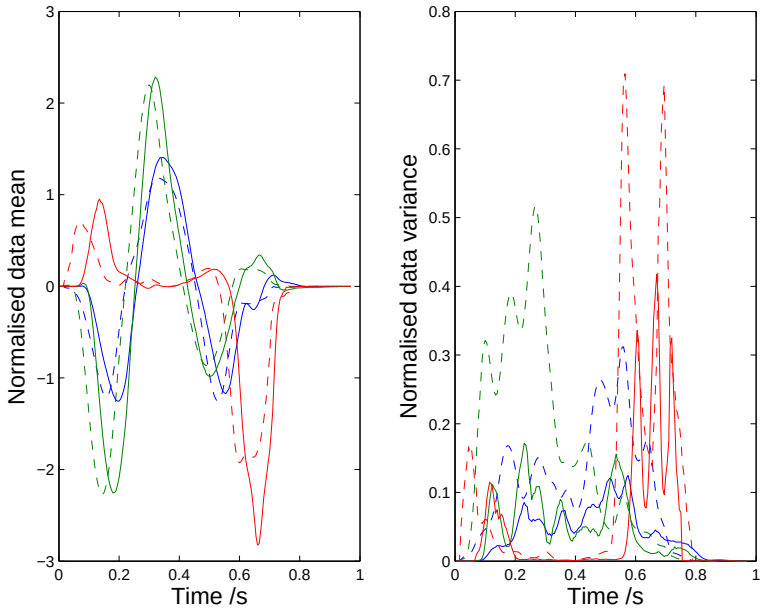


Figure 6.24: Mean and variance profile plot of the samples produced using the integrate and fire model to model primitive onset times. The original data is shown with a dashed line for comparison. The temporal offset is not very important, however, this model does not model produces samples with too little variance in general, along with other problems as described in Section 6.2.3.

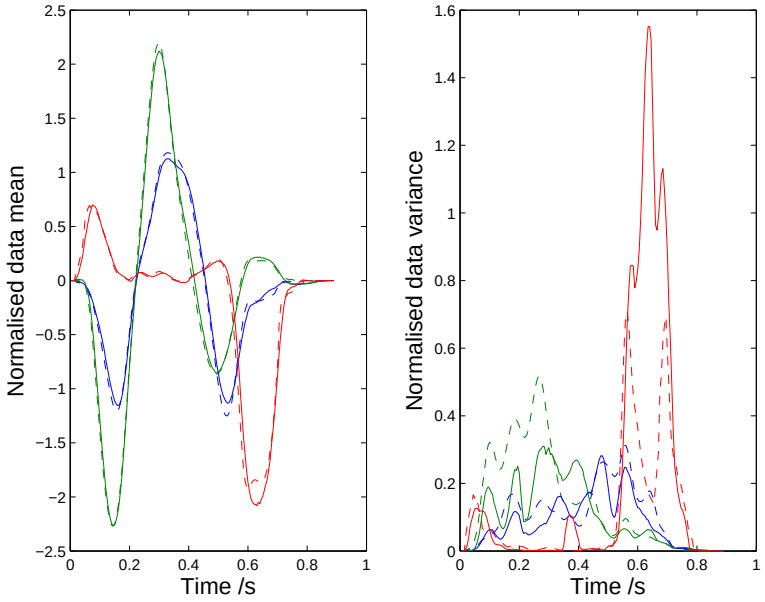


Figure 6.25: Mean and variance profile plot of the samples produced using the separate HMM model to model primitive onset times. The original data is shown with a dashed line for comparison. Mostly the profiles are well modelled, except for a single dimension of the variance profile.

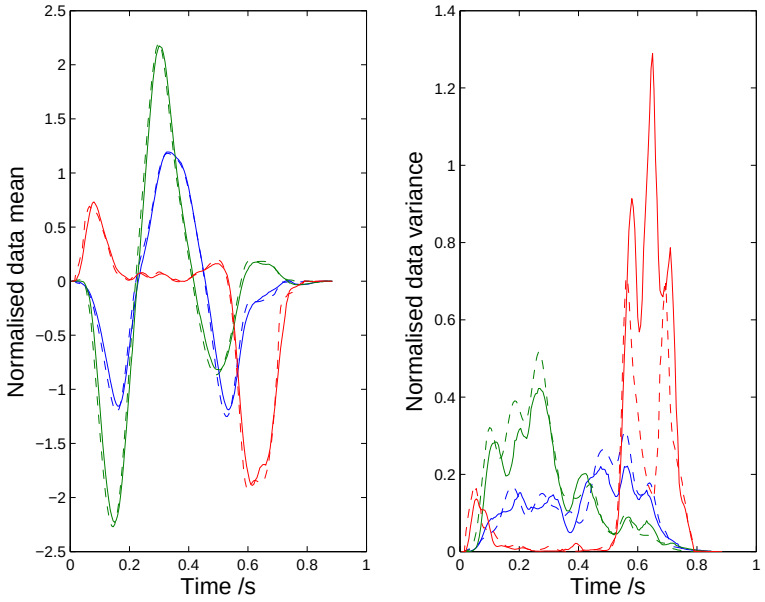


Figure 6.26: Mean and variance profile plot of the samples produced using the separate HMM model but using data-conditioned modes as described in Section 6.2.4.2 to model primitive onset times. The original data is shown with a dashed line for comparison. Using the modes of the characters to couple the presence or absence of the primitives together helps the HMM model produce characters, however this is not strictly generative behaviour, as the modes are data-conditioned.

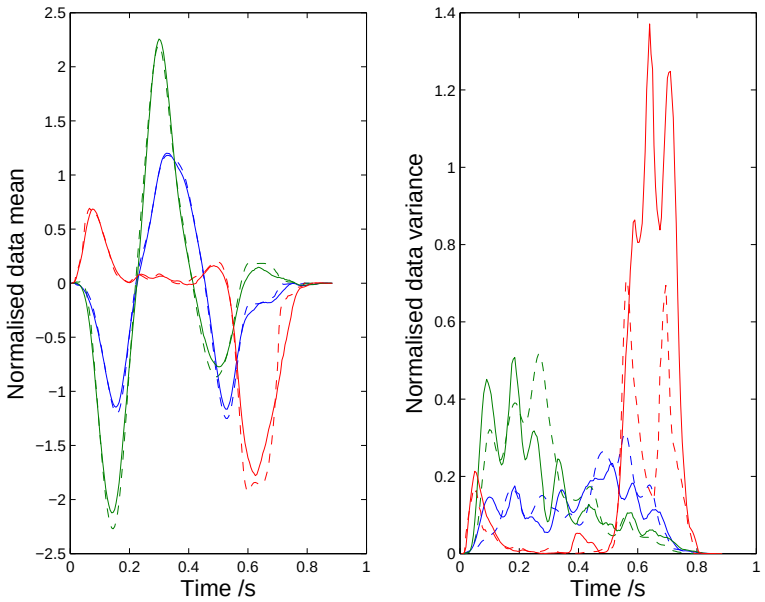


Figure 6.27: Mean and variance profile plot of the samples produced using the single combined HMM model to model primitive onset times. The original data is shown with a dashed line for comparison.

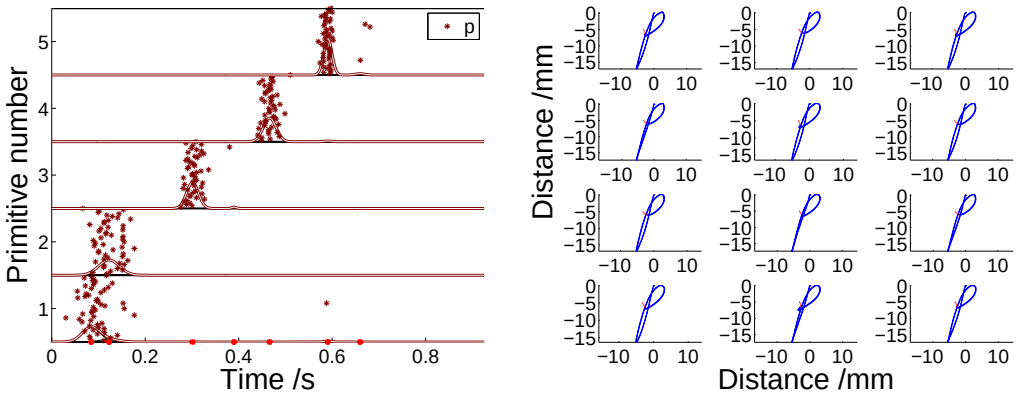


Figure 6.28: Samples from the single combined HMM timing model, but only based upon 5 primitives. Fewer primitives mean that the model must use them more sparingly and specifically for certain character regions. This sparsity improves the performance of the generative model.

limited set of primitives represent shapes present in more samples. The primitives are representative of the character being drawn, rather than trying to model noise in the data which is not represented by primitive timing variation. This makes the task easier for the timing model, because the distribution of the spikes is more clustered.

Instead of reducing the number of primitives used in the model, it is possible to introduce a clustering prior in a principled manner, by coupling the two models together during learning, as is detailed in Chapter 7. This coupling is effected by introducing an informative sparsity prior, based upon where spikes are likely to occur. This prior is therefore called a ‘spike prior’.

Chapter 7

Coupled Model

In Chapters 4 and 6, two levels of model have been discussed. Firstly, the low level model was presented, which captures the overlapping primitive segments, and provides a prior over general pen-movements, as can be seen in Figure 5.29. To provide a generative model for writing, the next stage is to model the timing of the primitives and generalise across different samples of the same character. Several different timing models were discussed in Chapter 6, along with their advantages and shortcomings. Although the HMM model is a nice solution, producing reasonable generative character samples, when using many primitives, it has a tendency to over fit the data, as the primitives are learnt as though the data is meaningless scribble, without timing dependence upon primitive onset. This means that the primitives can often be fit to inappropriate parts of the character for specific character samples, making a generalised timing model difficult to fit.

A better way to learn the primitives would be to use an informed prior for the primitive onset probabilities. This prior comes from a timing model, which is in turn learnt from the spike data extracted using the low level fHMM primitive model. This coupling of the models during learning is an iterative process, which can be implemented during the EM inference of the fHMM.

Due to the compression of the data into the form of spike times, the inference of the timing model is considerably faster than for the primitive model, therefore there is not a huge difference in inference time for the coupled model as opposed to the uncoupled fHMM.

This chapter details the method of coupling of the low level primitive model, and the higher level timing model. The full coupled model can be learnt in an iterative variational manner, and is discussed in this chapter, along with the difference that

the coupling makes to the parameters learnt. The coupled model presented here is published in (Williams et al., 2008).

7.1 Coupling method

The single combined HMM timing model introduced in Section 6.2.6 is adopted to model the spike data representing the primitive onsets. These spike timings are derived from the posterior over the hidden states of the primitive model, as discussed in Section 6.1, and provide an intermediate representation of the data. This representation is much more compact than the observable trajectory data, and allows more compact higher level models to be introduced that can more easily generalise across different character samples. To learn the models in a coupled fashion, we firstly look at the output distribution of the HMM timing model.

The HMM timing model is used to provide a mixture of Gaussians distribution where the presence or absence of a component is inferred from the data. A graphical form of the timing model can be seen in Figure 7.1. There is one timing model with separate transition parameters for each primitive. The timing model is a standard HMM with one dimensional Gaussian outputs representing the time of an indexed spike. Each spike, which represents the onset of a primitive, is assumed to come from a specific generating Gaussian component; however the components do not reliably generate spikes, as evidenced by the varying number of spikes per character sample.

The data being modelled by the HMM is a set of I spikes, $\{\tau_1, \tau_2, ..., \tau_I\}$. The number of spikes per sample varies, but they are assumed to originate from consistent generative components, modelled by Gaussian distributions. Each component is assumed to give rise to at most one spike. The component generating a particular spike is modelled in the hidden state of the model, $\mathbf{G}_i \in \{1...K\}$, where $P(\mathbf{G}_i = k)$ is the probability that spike i is generated by the k^{th} component. It should be noted that K is therefore the maximum value of I taken over all samples.

The timing model provides a distribution over the timing of an indexed spike

$$P(\tau_i) = \sum_{\mathbf{G}_i=1}^{K} P(\tau_i \,|\, \mathbf{G}_i) P(\mathbf{G}_i) \;, \tag{7.1}$$

where τ_i is a continuous variable representing the time of spike i in a sequence of I spikes (from a single primitive). $\mathbf{G}_i \in \{1..K\}$ is a discrete variable, indicating that

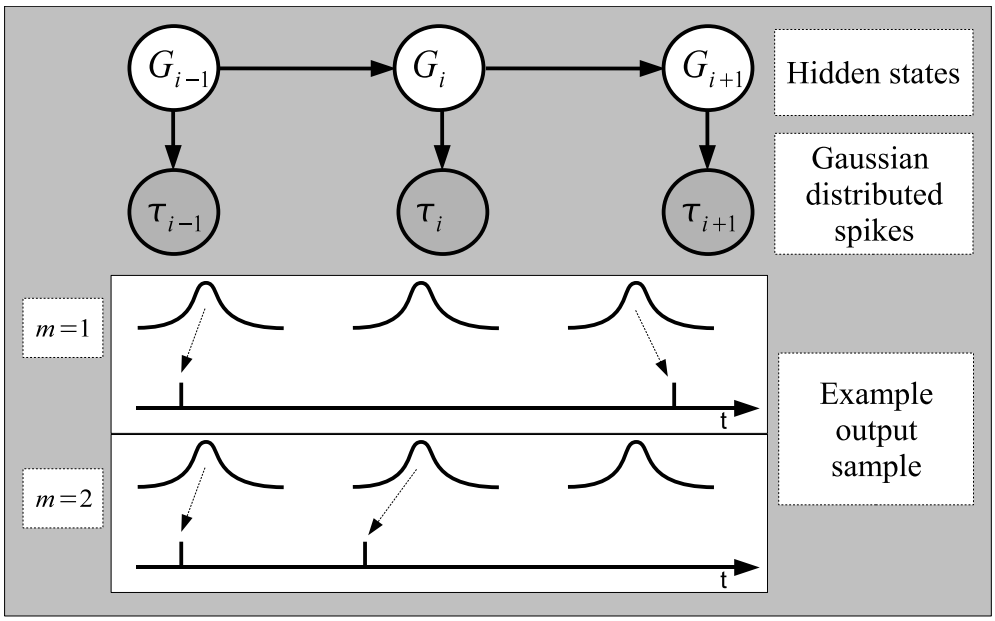


Figure 7.1: Graphical model of the single combined HMM with continuous, one dimensional, observable states representing spike times. The transition parameters are specific for primitive and character type, whereas the output parameters are shared.

spike i comes from the k^{th} generative component.

$$P(\tau_i \,|\, \mathbf{G}_i = k) = \mathcal{N}(t, \boldsymbol{\mu}_k, \sigma_k) \tag{7.2}$$

$$P(\mathbf{G}_i) = P(\mathbf{G}_1) \prod_{i=2}^{I} P(\mathbf{G}_i \,|\, \mathbf{G}_{i-1}) \;, \tag{7.3}$$

where $\boldsymbol{\mu}_k$, σ_k, $P(\mathbf{G}_1)$, and $P(\mathbf{G}_i \,|\, \mathbf{G}_{i-1})$ are standard learnt parameters of the HMM class of models, with certain constraints. The values of $\boldsymbol{\mu}_k$ are ordered so that $\boldsymbol{\mu}_k > \boldsymbol{\mu}_{k-1}$. The minimum value of σ_k is limited to one, as the spikes come from a discrete time system (the fHMM described in Chapter 4). The transition matrix, $P(\mathbf{G}_i \,|\, \mathbf{G}_{i-1})$, is constrained to be a strictly lower diagonal matrix so that each component can produce only one spike.

The mixture of Gaussians is a multi-modal distribution that is simply an additive mixture of independent Gaussian distributions. The HMM is used to model this distribution because its hidden states can represent local dependencies between the presence or absence of a particular component in the mixture. The distribution for the Gaussian mixture is simply the sum over the sample indices, i.

$$P(\tau) = \sum_{i=1}^{I} \sum_{\mathbf{G}_i=1}^{K} P(\tau_i \,|\, \mathbf{G}_i) P(\mathbf{G}_i) \;. \tag{7.4}$$

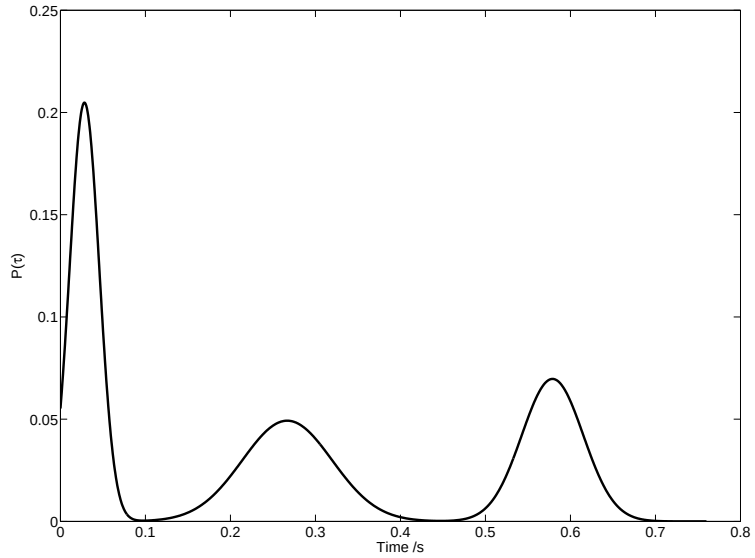


Figure 7.2: Example mixture of Gaussians distribution, showing where all spikes in a particular sample are likely to occur.

The reason we must sum over the separate samples to produce this mixture distribution is due to the difference in indexing between the lower level fHMM based primitive model, and the higher level HMM timing models. The timing model is indexed by spike count, and produces a distribution *over time* whereas the primitive model is indexed by time, and requires a distribution *over primitive onset*, which is a binary variable. During inference of the lower level model, we don't know the state of the timing model at any particular time point, so we cannot use the timing model to provide a prior over the *next spike*, which would be more informative. Instead we must use the timing model to provide a prior over all the spikes, which is the multi-modal distribution shown in Figure 7.2. This *spike prior* gives the prior probability of seeing a spike at a particular time, without knowledge of the state of the timing model.

In the uncoupled primitive model, at each time step, there is assumed to be a prior for onset of any particular primitive, which is a single term taken from the Markov transition matrices,

$$P(\mathbf{S}_t^m = 1 \,|\, \mathbf{S}_{t-1}^m = 0) = \mathbf{P}_{2,1}^m \,. \tag{7.5}$$

This term in the coupled model is replaced by a binary variable, λ_t^m, indicating the onset of primitive m at time t, as dictated by the top-down timing model. The mixture of Gaussians distribution provides a continuous distribution over $P(\tau)$. The equivalent

discrete distribution is precisely the distribution over the variables λ_t^m that we require,

$$P(\lambda_t^m) = \int_{t-0.5\delta t}^{t+0.5\delta t} P(\tau)\, d\tau\,, \tag{7.6}$$

where t indexes a discrete time, with steps of δt, so that integrating the probability density function $P(\tau)$ over this interval approximately gives the set of binary distributions $P(\lambda_t^m)$. The $P(\lambda_t^m)$ binary distributions then provide the onset prior for the primitives at time t. These are then substituted into the transition matrix $\mathbf{P}_{i,j}^m$ for each primitive, and at each time point.

$$P(\mathbf{S}_t^m = b \,|\, \mathbf{S}_{t-1}^m = a, \lambda_t^m) = \begin{cases} P(\lambda_t^m) & \text{for } a = 0 \text{ and } b = 1 \\ 1 - P(\lambda_t^m) & \text{for } a = 0 \text{ and } b = 0 \\ 1 & \text{for } a \neq 0 \text{ and } b = (a+1) \bmod K^m \\ 0 & \text{otherwise} \end{cases} . \tag{7.7}$$

This means there is now a non-stationary prior for the transitions in the fHMM. This reflects the non-stationarity of the probability of occurrence of the primitives during the writing of a character. The spike prior couples the higher level timing model, and the low level primitive model together.

It should be noted that for the sampling procedure, it is not the variables λ_t^m that are sampled from, rather the distributions $P(\tau_i)$ which then parameterise the λ_t^m variables. If the transition matrices in the fHMM are entirely deterministic given the values of λ_t^m, then sampling the states of the primitives is a trivial procedure. If the constraints on the transition matrices have been relaxed, for example by introducing self-transitions, so as to allow *stretchy* primitives as discussed in Section 4.4, then a single forward pass along the Markov chains produces a sample of the states of the primitives over time.

7.2 Generative sampling

In this section, we assume that appropriate parameters have been learnt for the whole coupled model (parameter learning is addressed in Section 7.3). For generative sampling, we sample in a top-down fashion from the prior distributions of the model. Generative samples of data are therefore produced in the observable output, which can be used to assess the appropriateness of the model. The entire, coupled model can be seen in Figure 7.3. Assuming appropriate parameters have been learnt for the model, sampling is done in a top-down fashion.

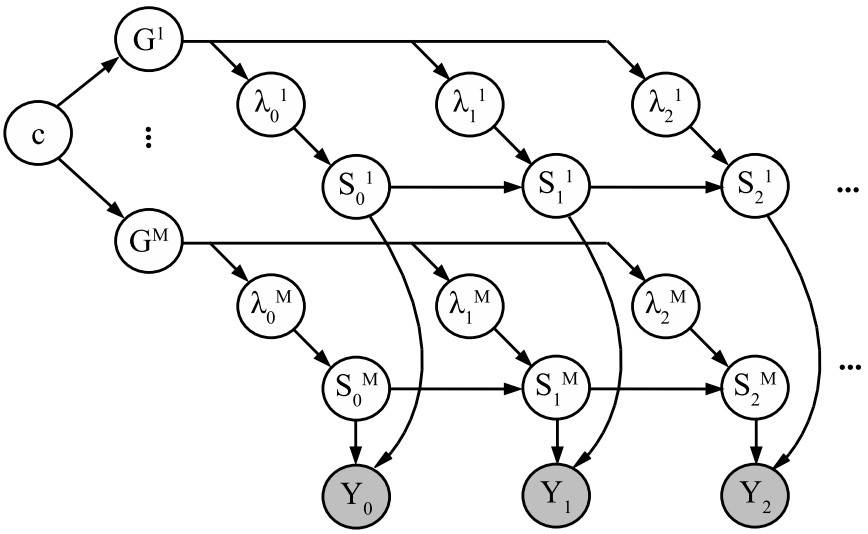


Figure 7.3: Bayesian graphical model showing both the timing model, and the primitive model coupled together.

c Firstly, the character class, c is sampled, indicating the type of character to be written. Given this class label, the timing model is parameterised in a character-specific manner.

$\mathbf{G}_i^m$ The Markov chains $\mathbf{G}_i^m$ are sampled using a single forward pass along the chains, giving the presence or absence of the Gaussian components that generate spikes.

λ_t^m A single spike time is sampled from each present Gaussian component. For ease of coupling, these spike times parameterise the complete set of coupling variables λ_t^m.

$\mathbf{S}_t^m$ The coupling variables parameterise the non-stationary transition matrix for the Markov chains $\mathbf{S}_t^m$, as described in Section 7.3.2. A single forward pass along these chains produces a sample of the states of all the primitives over time, giving a length along each primitive at each time point.

$\boldsymbol{\mu}_t$ A superposition of the appropriate output contributions of the primitives over time parameterises the mean of the Gaussian output distribution, $\boldsymbol{\mu}_t$, which is assumed to be independent at each time point. The covariance of the output distribution is assumed to be stationary, and is learnt using EM, therefore does not need to be parameterised before sampling.

$\mathbf{Y}_t$ Sampling from the Gaussian output distribution produces a sample of observable data - pen-tip trajectories, $\mathbf{Y}_t$, as described in Section 5.1. Numerical integration

of these trajectories produces character samples, such as those seen in Section 7.4.

7.3 Inference

The parameters for the coupled model are difficult to learn as the model is quite large, and multi layered. A variational approximation to the posterior distribution is used to infer the state of the model, whilst the parameter updates are done as if the model consisted of the two separate models, the fHMM primitive model and the HMM timing model, where the whole model is coupled together with the spike prior. The inference is an iterative procedure, with several adjustable constants. This section gives a description of how the inference was done algorithmically.

To clarify, the low level primitive model is a single fHMM model, and will be referred to as the primitive model. The higher level timing model is a set of M HMM models, one for each primitive. This collection of HMMs will be referred to as the timing model.

The inference of the timing model is much faster than the primitive model inference, due to the difference in indexing. The primitive model is indexed by time, and therefore, in a typical character sample, the primitive model will have around 100 samples, whereas the timing model is indexed by spike count, and for a typical character sample, there are around 2 spikes for each primitive.

7.3.1 Inference order

Parameter initialisations

All parameters are initialised to likely values. Output covariance is initialised to data covariance. Primitive shapes are taken from sections of data averaged over the character samples. Onset probabilities are initialised to $1/T$ where T is the average length of a character sample. The timing model component means are initialised using a simple one dimensional local maxima finding algorithm based on the pool of all the spike timing data. The variance of the clusters is initialised to the variance of a greedy absolute classification of the data. The transition parameters are initialised to values reflecting this initial data classification.

Initial uncoupled EM

Firstly the primitive model is inferred, by performing variational EM without a spike prior, but with a learnt primitive onset parameter, as described in Section 4.3. Twenty iterations of EM are performed to learn sensible shapes for the primitives. If training on a large dataset, this stage of inference can be performed on a representative subset of the data for speed.

Spike prior inference

A single E-step must be performed on the entire dataset to infer the posterior over λ_t^m. This is represented in terms of spike times by taking the Viterbi alignment of the hidden states of the fHMM (see Chapter 6). The spike data is organised into character class, and for each class, the inference of a timing model is done using standard EM for HMM models, with certain constraints, as described in Section 7.3.3. One spike prior per character class is produced by integrating over the timing model, as in Equation 7.1.

Coupled inference

Using the spike priors to parameterise the transition matrix for the primitive model to create a more informed prior on where the primitives are likely to be active, variational EM is used to infer this model, as in Section 7.3.2. As above, this inference need not be on the entire dataset, if the size is inhibitory. A constant spike prior is used for 20 iterations, at which point the **spike prior inference** is repeated.

7.3.2 Primitive fHMM inference

The primitive model is described in detail in Chapter 4. The only difference for the inference in the coupled version is that the spike prior is used to parameterise the transition matrix, as in Equation 7.1. In practice this is done by multiplying the likelihood function with the spike prior, and setting the transition parameters to unity, before

performing the Baum-Welch forward backward algorithm.

$$h'^{m}_{t,0} = h^{m}_{t,0} \cdot (1 - P(\lambda^{m}_{t})) \tag{7.8}$$

$$h'^{m}_{t,1} = h^{m}_{t,1} \cdot P(\lambda^{m}_{t}) \tag{7.9}$$

$$\mathbf{P}'^{m}_{0,0} = 1 \tag{7.10}$$

$$\mathbf{P}'^{m}_{1,0} = 1 \ , \tag{7.11}$$

where $\mathbf{h}^{m}_{t}$ is the variational factor taking the place of the observational likelihoods, $P(\mathbf{Y}_t \,|\, \mathbf{S}^{m}_{t})$, and $\mathbf{P}^{m}$ is the transition matrix for the M Markov chains in the primitive model. Algorithmically, multiplying the likelihood in this manner is equivalent to using a non-stationary transition matrix.

7.3.3 Timing HMM inference

The inference of the timing model can be done sensibly only when reliable spike timing information has been inferred in the lower level primitive model. This is why there is an initial 20 EM steps of uncoupled inference. Also, with large datasets, containing several different characters, the time taken to infer the lower level primitive model for all data samples can be inhibitory. The primitives are assumed to be used in all character samples, however, the timing models are character type specific, and to have enough data to learn the model, a single E-step must be performed on the entire dataset, to generate enough spike data to fit the timing models.

Ideally, the character class would be inferred, and there would be a single timing model, where the parameterisation depends upon the character class. Character recognition is currently not reliable enough to be implemented during model inference, and so this generalisation is left for the future.

7.4 Results of coupling

Coupling the models together during learning has two effects on the timing distribution of the primitives learnt. Firstly, the clusters of spikes are slightly tighter, and secondly, outliers from the clusters will be suppressed. Using exactly the same dataset, consisting of 50 samples of the character *p*, two runs were performed. Firstly an uncoupled primitive model was used to learn the primitives, then after 200 EM iterations a timing model was fit to the spike timing data. A separate run was performed on the same

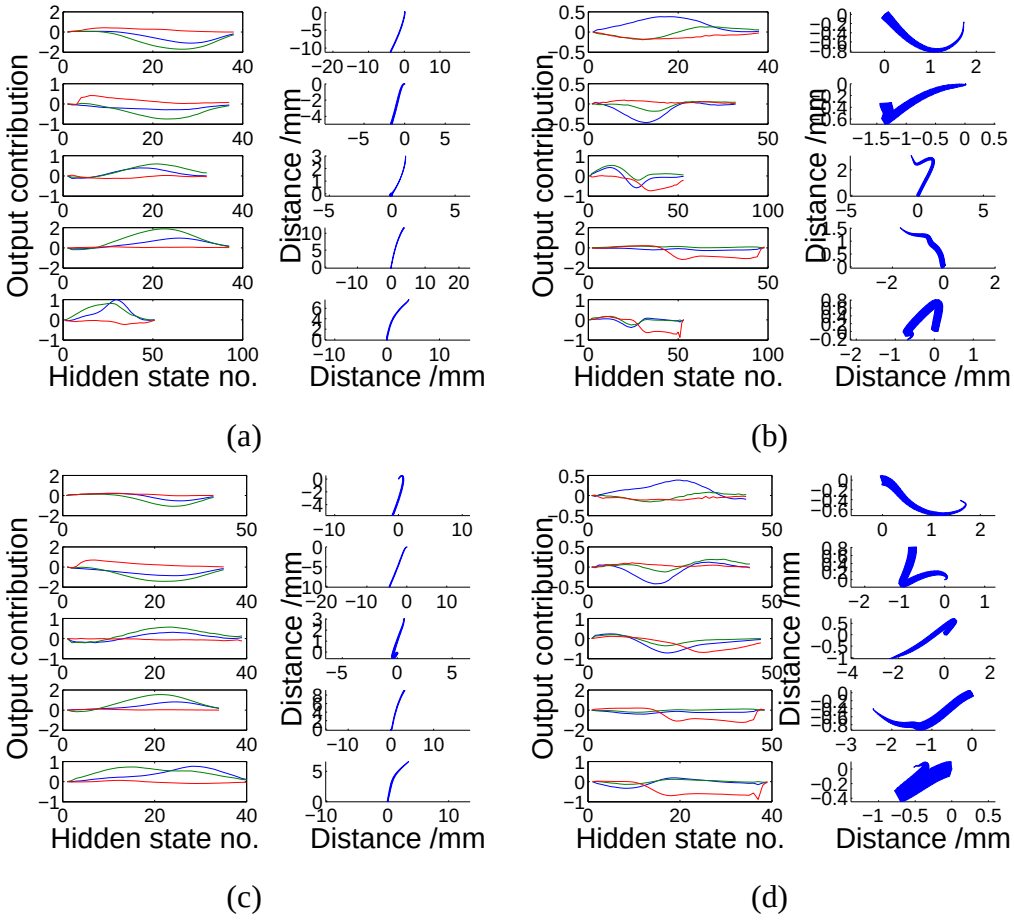


Figure 7.4: Primitives extracted from the same dataset using the same initialisations, once using an uncoupled model, seen in plots (a) and (b), and once using a coupled model, shown on plots (c) and (d). The differences between the resultant primitives (after 200 EM iterations, including shifting) shows that the coupling has a subtle effect on the resultant primitives. Most of the primitives appear the same, with small variations, and one or two appear significantly different.

dataset, but using a coupled model as described in this chapter to learn a set of primitives. The number of EM iterations was the same, and the number of primitives, and their initialisations were the same for both runs. The two sets of primitives learnt can be compared in Figure 7.4. The timing distributions can be compared in Figure 7.5, and some generative samples from each run can be seen in Figure 7.6. The coupling process means that the primitives learnt are better suited to the coupled model, where the primitives are 'encouraged' to occur in Gaussian timing distributions. As these components are part of the generative process, it means that the primitives learnt from the coupled model produce more reliable generative samples.

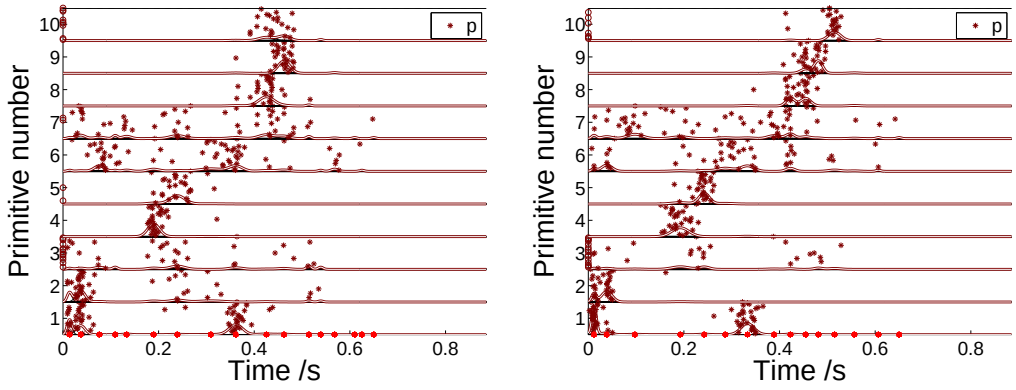


Figure 7.5: Timing distribution of the primitives seen in Figure 7.4. The left plot shows the timing distribution of primitives inferred for the uncoupled run, and the right plot for the coupled run. The effect of coupling, as can be seen here, is to tighten the spike scatter towards the Gaussian components, and to suppress outliers. It should be noted that this will only occur if the model is 'unsure' that the outlier should exist. There are still outliers present in the coupled model, but the occurrence of the primitive at that particular point was important enough for the model to keep the outlier.

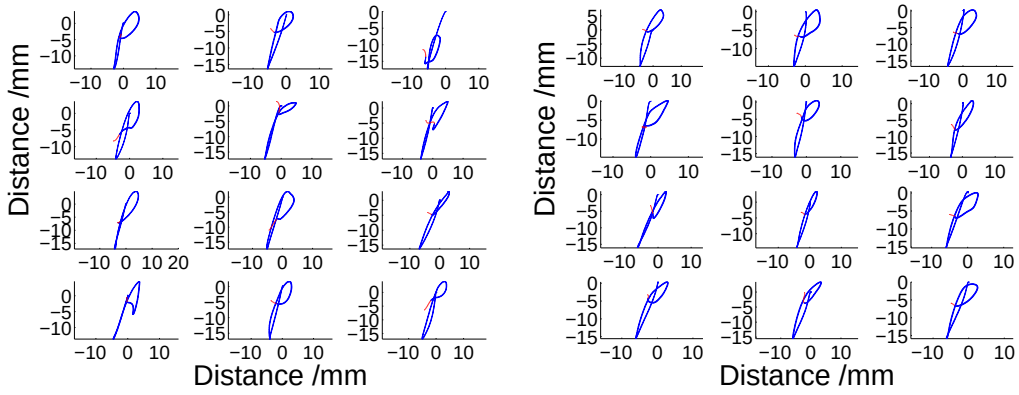


Figure 7.6: 12 generative samples from two separate runs with the same initialisations. The left plot shows an uncoupled run, and the right plot shows a coupled run. The process of coupling means that the generative samples, (which are produced from a coupled model in any case) are generally better.

7.5 Number of primitives

The number of primitives used was discussed in Chapter 5, but without the timing model, it is difficult to decide upon an optimal number of primitives.

To examine the effect of varying the number of primitives, 20 runs were performed on a dataset containing the character *p*. Each run uses 1 more primitive than the previous, and all runs are initialised from the data, with shifting parameters $z = 60\%$, $b = 90\%$ as described above. The initial length of the primitives was inversely proportional to the number of primitives, to maintain the same number of initial state size of the model. The primitives extracted from these runs can be seen in Figure 7.7. Samples of posterior reconstructions of the dataset from the various models can be seen in Figure 7.8. The more primitives used, the better the reconstructions of the training dataset become, as can be seen in Figure 7.9. This is to be expected, as the more primitives available, the more exactly the model can represent the data. If large numbers of primitives are present, some primitives might only be used in very specific circumstances, for example in a single character sample. This over-fitting of the data does not affect the posterior reconstruction error as it is derived from the specific dataset upon which the model was trained. It might have an adverse effect on the reconstruction error of an unseen test set, as is explored for a single set of primitives in Section 5.5; however further problems arise when we consider the generative behaviour of the model.

Although the posterior reconstruction of the training dataset would seem to suggest that larger numbers of primitives are better, there are additional problems associated with using more primitives. Firstly, and most obviously, the computational load increases with more primitives, slowing down the inference process. More importantly however, as more primitives become available, the primitive usage becomes more coupled. Multiple primitives are used to model the same motifs in the data, or they are substituted, depending upon the particular sample being modelled. This coupling is not well modelled in the current timing model, and so the optimal number of primitives w.r.t. the generative samples provides a good measure for how many primitives should be used.

Generative samples from the 20 runs with increasing numbers of primitives can be seen in Figure 7.10. The low numbers of primitives generally produced better quality samples, but with less variance. Objective measures for assessing the generative performance of the model are detailed in Section 8.5.1.1. There are two measures defined, the TAMPE and the TASDPE. The TAMPE defines the Total Absolute Mean Profile

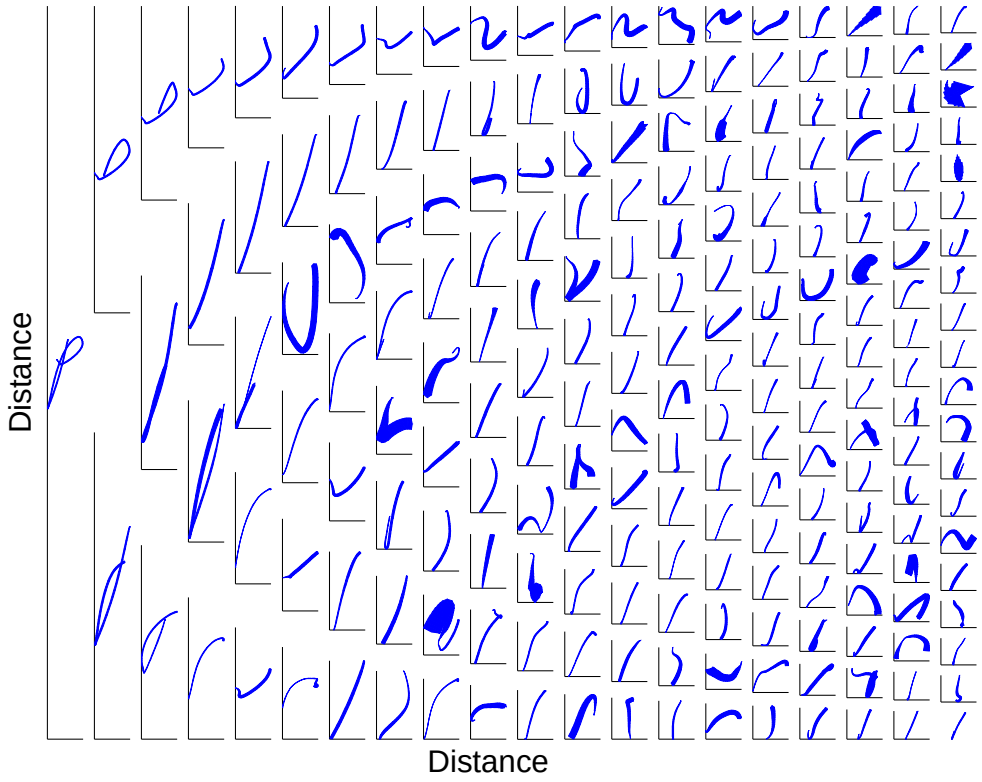


Figure 7.7: Twenty runs with increasing numbers of primitives, trained on the *p* dataset. Each column shows a separate run, and the rows are sorted so that similar primitives are aligned. It is hard to see any clear trends across the runs, although there are some primitives that seem to occur repeatably. The plots were aligned so that adjacent plots have the least mean squared error.

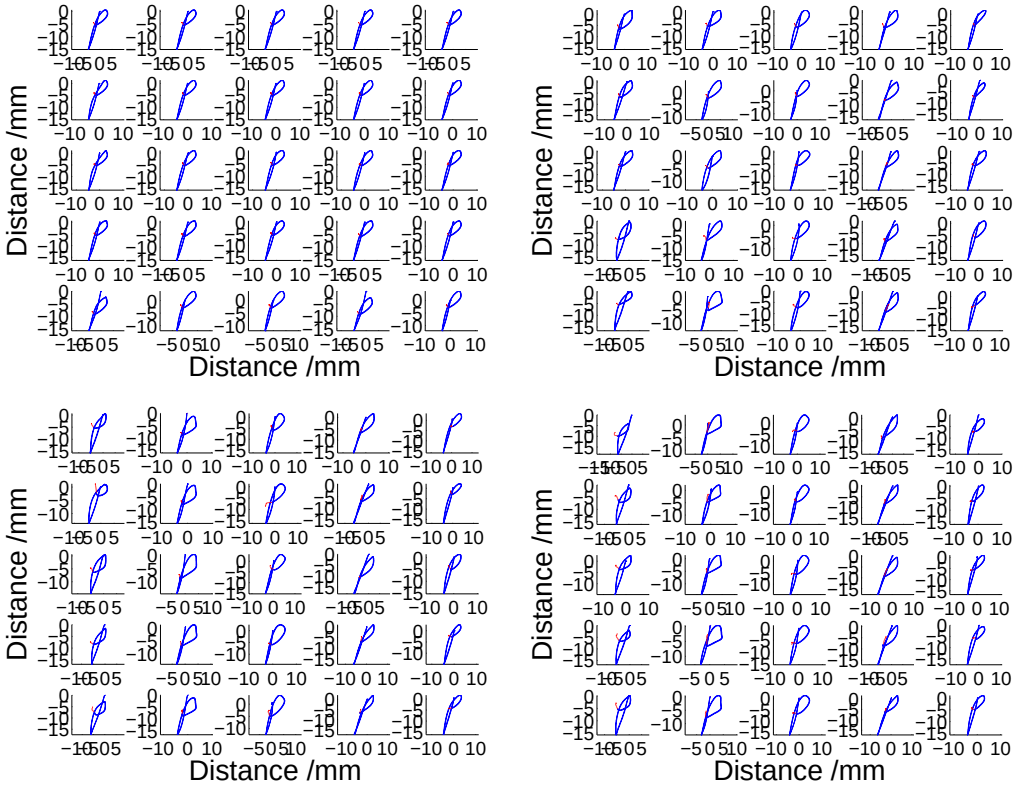


Figure 7.8: Samples of character reconstructions from multiple runs with increasing numbers of primitives. Each row from one of the four quarters shows samples with the same number of primitives. It can be seen that the variance in the reproductions increases with the number of primitives, as would be expected. In the special case of just one primitive, the mean of the dataset is modelled, as can be seen in the top left plot in the top row, where all the reproductions are identical.

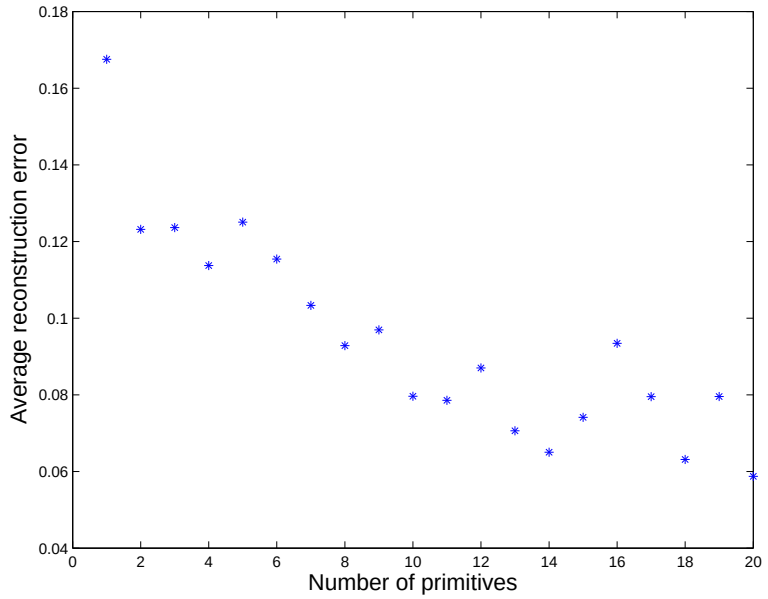


Figure 7.9: Reconstruction error for 20 separate runs, with different numbers of primitives. The reconstruction error is that of the posterior reconstruction of the training set, hence more primitives will improve the reconstruction.

Error, and the TASDPE defines the the Total Absolute Standard Deviation Profile Error. These measures compare the generative samples produced by a model with a set of unseen test samples, and compare the average character from each set. Lower values mean that the model can produce generative samples that better match the test set.

Figure 7.11 shows the TAMPE and the TASDPE plotted against the number of primitives available, based upon a set of models trained in an uncoupled fashion. These plots show that on average the generative samples are less good as the number of primitives increases, or with very few primitives.

Initially the uncoupled approach to training was chosen for the purpose of selecting the optimal number of primitives, with no bias from a coupling prior. Additionally the same experiment was repeated, but using a coupled model. The reconstruction error follows the same trend as in the uncoupled case, which is to decrease as the number of primitives increases. Figure 7.12 shows the TAMPE and the TASDPE plotted against the number of primitives available, using coupled models to learn the parameters for each model. Here we can see that the generative performance of the coupled model seems to not decrease as the number of primitives increases, in the same manner as the uncoupled model. This is because the coupling prior helps to limit the over-fitting of the primitives to the training data described above. The coupled model learns primitives that are more appropriate for character generation.

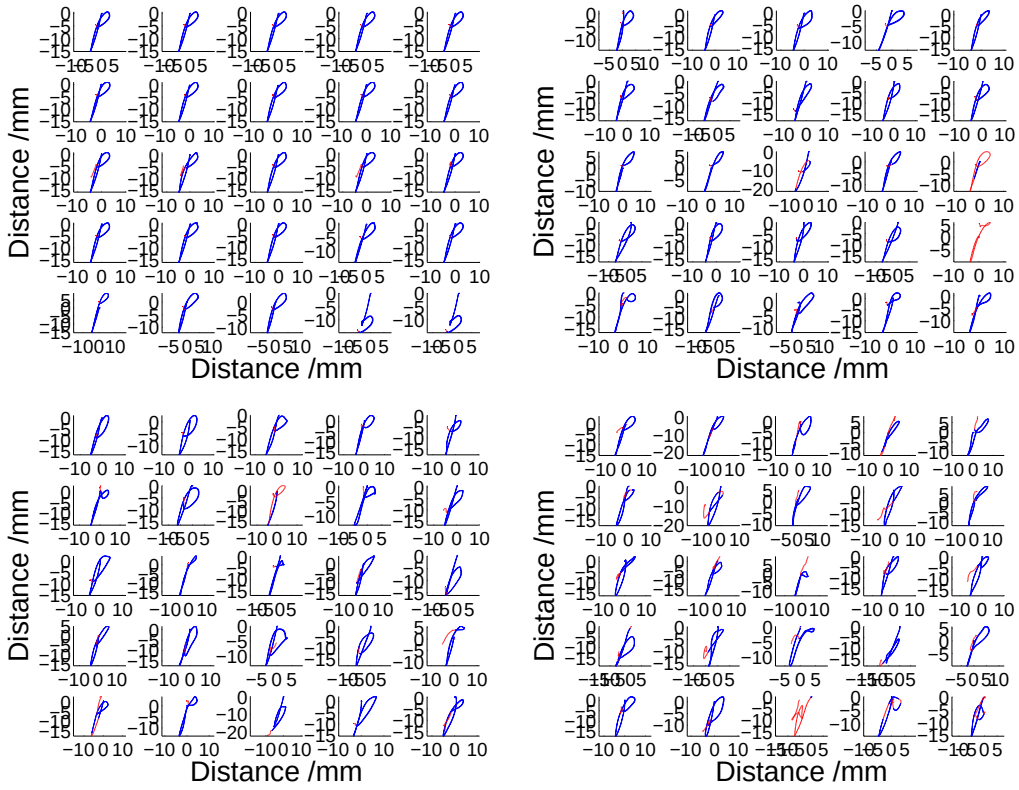


Figure 7.10: Generative samples of characters using uncoupled runs with increasing numbers of primitives. Each row from one of the four quarters shows samples with the same number of primitives. It can be seen that the variance in the reproductions increases with the number of primitives, as is the case for the posterior reconstructions seen in Figure 7.8, however, as the number of primitives continues to increase, the quality of the samples starts to deteriorate. This is because the spike timing representation becomes more complex with more primitives, and so the independence assumptions of the timing model become inappropriate for large numbers of primitives.

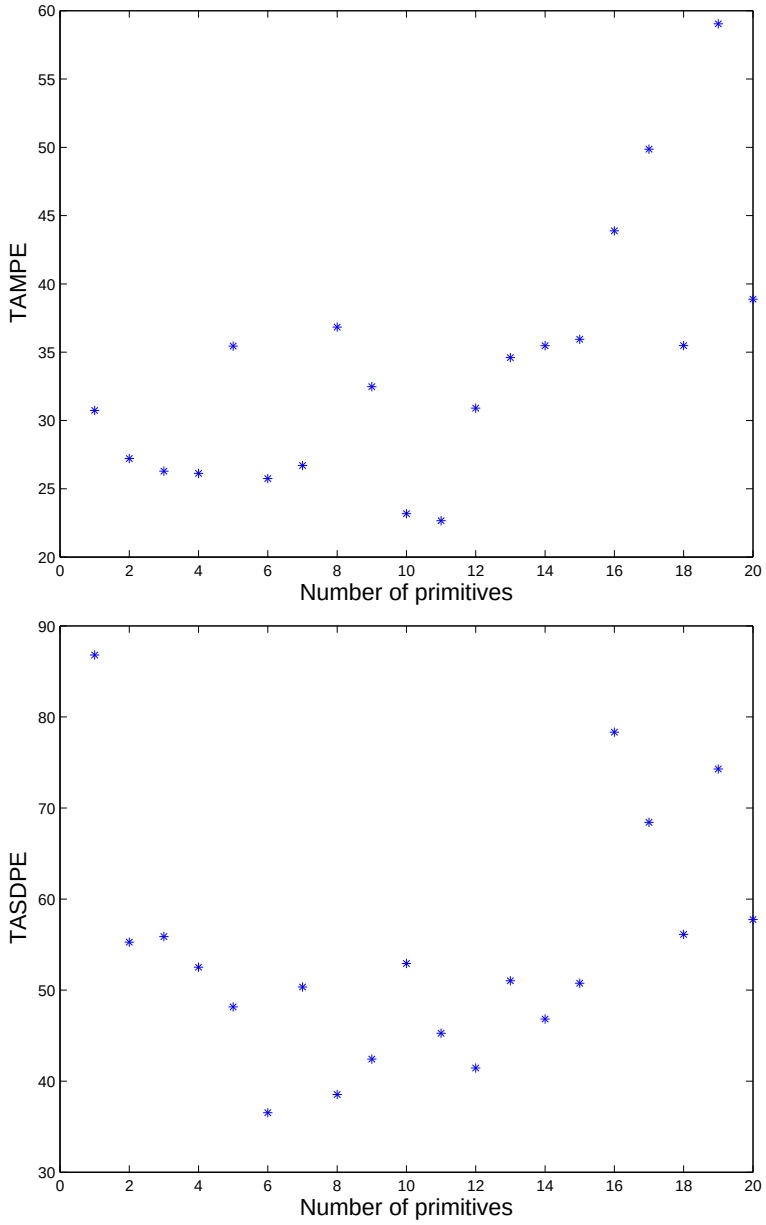


Figure 7.11: Generative performance of 20 uncoupled models trained with different numbers of primitives. The average character from 50 generative samples is compared with the average character from 50 test samples in each case.

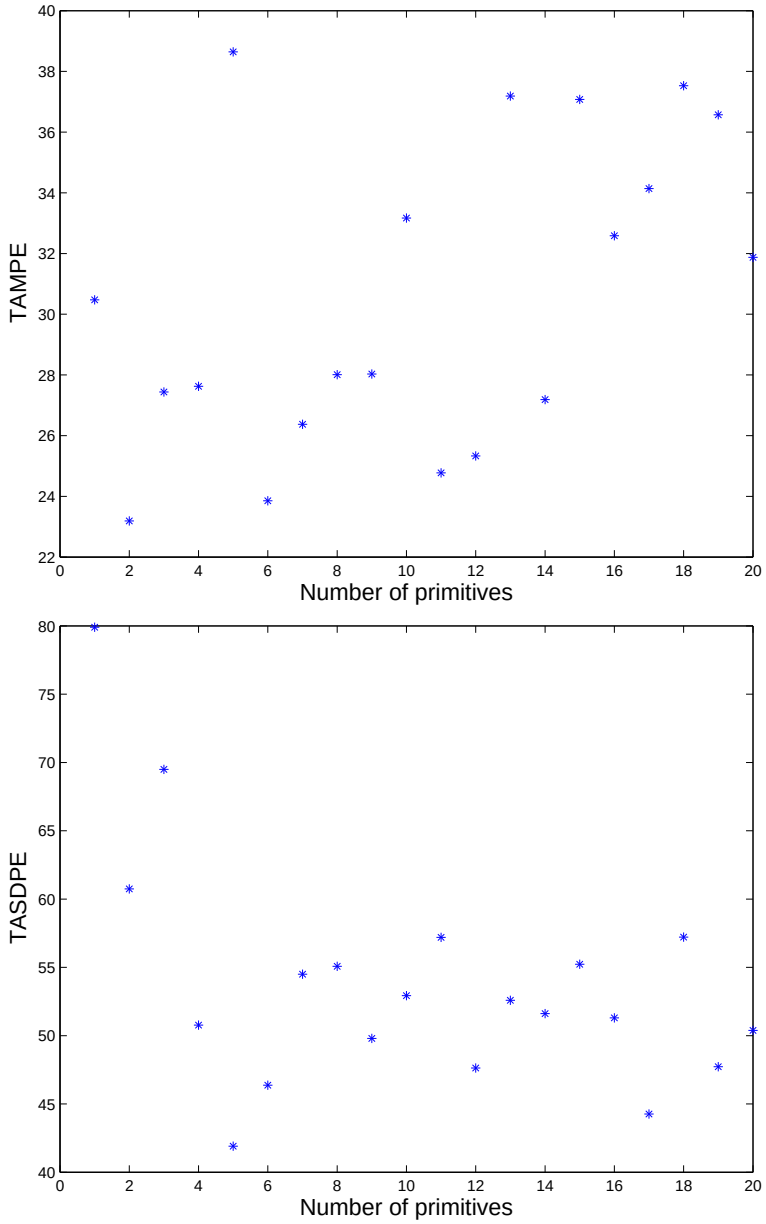


Figure 7.12: Generative performance of 20 coupled models trained with different numbers of primitives. The average character from 50 generative samples is compared with the average character from 50 test samples in each case.

All these results contain a lot of noise. This is partly because the training set sizes were relatively small, to speed up the experiment, which required separate training on each dataset. In general however, it can be seen that the optimal number of primitives for a single character should be around 6-15. With multiple characters in the dataset, the number of primitives required increases, but sub-linearly to the number of character types. It was found empirically that 5-10 primitives model a single character well, whilst 10-15 should be used for sets of 3 to 5 characters. The largest set of character types used was 20, which can be modelled with 20 primitives.

Chapter 8

Results

The past four chapters have described a handwriting model based upon the concept of primitives. Chapters 4 and 5 give details of the lower level parts of this model, and Chapters 6 and 7 examine possible higher level models and how the two models can be coupled together allowing generalisation across character samples.

This chapter examines the results of generative experiments using the coupled model. There are several ways to assess such experiments, and these are discussed. The types of primitives extracted will be shown, and the possibility of these primitives modelling the *style* of a person's handwriting will be assessed in a simple case.

Additionally, the appropriateness of the model for different character types can be used as a measure for character recognition. The discriminative performance of the model is not taken into account during training or model design, and as such can be thought of as a 'free' aspect of the model. The recognition accuracy for a simple case is assessed.

Finally this chapter includes a discussion and assessment of how this model compares to other handwriting models, and more general time series modelling systems. A comparison to some alternative models is performed, and in particular an auditory model based upon atomic signals will be examined.

8.1 Generative results

The model described in previous chapters is a generative model, which has a set of parameters which describe the probabilistic operation of the model. These parameters are learnt from a training dataset, using a variational expectation maximisation procedure as described in Section 4.3. Given a set of learnt parameters, the model then can gener-

ate data samples, and the probability of generating a particular sample under the model is calculable. This probability, $P(data \mid model\,parameters)$, is known as the *likelihood* of the parameters given the data, and is the objective function that is maximised during the expectation maximisation learning.

There are several objective methods of assessing the generative performance of the model. Firstly, the likelihood function computed for the generated samples gives an objective measure, although for the coupled model it is not possible to calculate the complete likelihood. This is discussed further in the recognition section, 8.4. An obvious way of assessing a generative character model is readability of characters, although this is a subjective rather than objective function. A more thorough study could be conducted whereby multiple subjects were required to judge whether some handwriting had been produced by a human or a machine, and then to use this measure as an assessment of the performance of the model, and can be thought of in terms of a handwriting Turing Test.

Reconstruction of the dataset can be assessed by examining the mean squared error of the samples, or alternatively the sum of the log probability of the data at each time point. For generative samples, a similar measure is obtained by examining the variance profile of the dataset. Assuming an independent Gaussian on each time point, a simple model can then be fit to the dataset. This simple Gaussian output model can then be used to assess the probability of the generative samples produced by the coupled model.

8.1.1 Readability

In this section, and throughout this chapter, several character reconstructions are presented. In these reconstructions, the thick blue line represents the track of the pen movements across the paper, where the thickness of the line represents the pressure of the pen tip. The thin red lines represent the path of the pen after it was lifted off the paper. Although this project is not explicitly trying to model this part of the character, sometimes in poor reconstructions, the pen is lifted off the paper too early, so this super-paper path is shown for clarity.

Several datasets were used, of different sizes and different numbers of characters for the training of different models. The number of primitives used in the model was in general set to 10, as this number provides a compromise between speed of learning and model complexity. In general, the more primitives, the better the reconstruction,

but the longer the training process. If too many primitives are used, the timing model takes a long time to learn, due to the complexity of the spike data, and a poor model produces poor generative samples. Too many primitives can produce over-fitting in the low level primitive model, which makes generalisation more difficult for higher level models. For large datasets containing many different character types, more primitives are used. An examination of the optimal number of primitives can be found in Section 7.5.

The initial length of the primitives is set to 20 states, although this is usually extended by the shifting algorithm.

Firstly, the simplest case is a dataset with a single character. Figure 8.1 shows 12 generative samples of the character *p*. These samples were produced from a model that was trained using a single character dataset. Effectively, the model only *knows* about the character *p*. This model consists of 10 primitives, shown in Figure 8.2

It is possible to train the model on a dataset of multiple characters, as can be seen in Figure 8.3. In this case, the class of the characters is pre-labelled before training, so that the timing model is trained upon appropriate data. It would also be possible to infer during learning which character class is the most likely given a particular data sample, although this would further complicate the inference procedure, and has not been attempted. The ability to infer character class is addressed in Section 8.4, and is not 100% reliable, therefore with the current timing model used as the character class assessor, this unlabelled method would probably not work very well.

Finally, a training dataset containing all the character types in Latin script which contain a single *pen-down* segment was used to train a model with 20 primitives, see Figure 8.5. The posterior reconstruction of the dataset was fairly good, as can be seen in Figure 8.4. The generative samples for this dataset are not so reliable, as can be seen in Figure 8.6. This is because more primitives are required to model the different characters, thus making the timing data more complex, with more unmodelled dependencies. The single combined HMM timing model attempts to model some of the local correlation between spike offsets, as detailed in Section 6.2.6, however the primitive substitution problem is not solved well. Primitive substitution is an anti-correlation of the presence of two (or more) primitives across samples. Assuming two primitives overlap, anti-correlation occurs due to variance in the data being modelled by swapping one primitive for the other. This is discussed in Section 6.2.4.2, where the *mode* of a character is defined as the sample specific progression through the hidden states of the timing model. Picking a mode at random, and then sampling from the

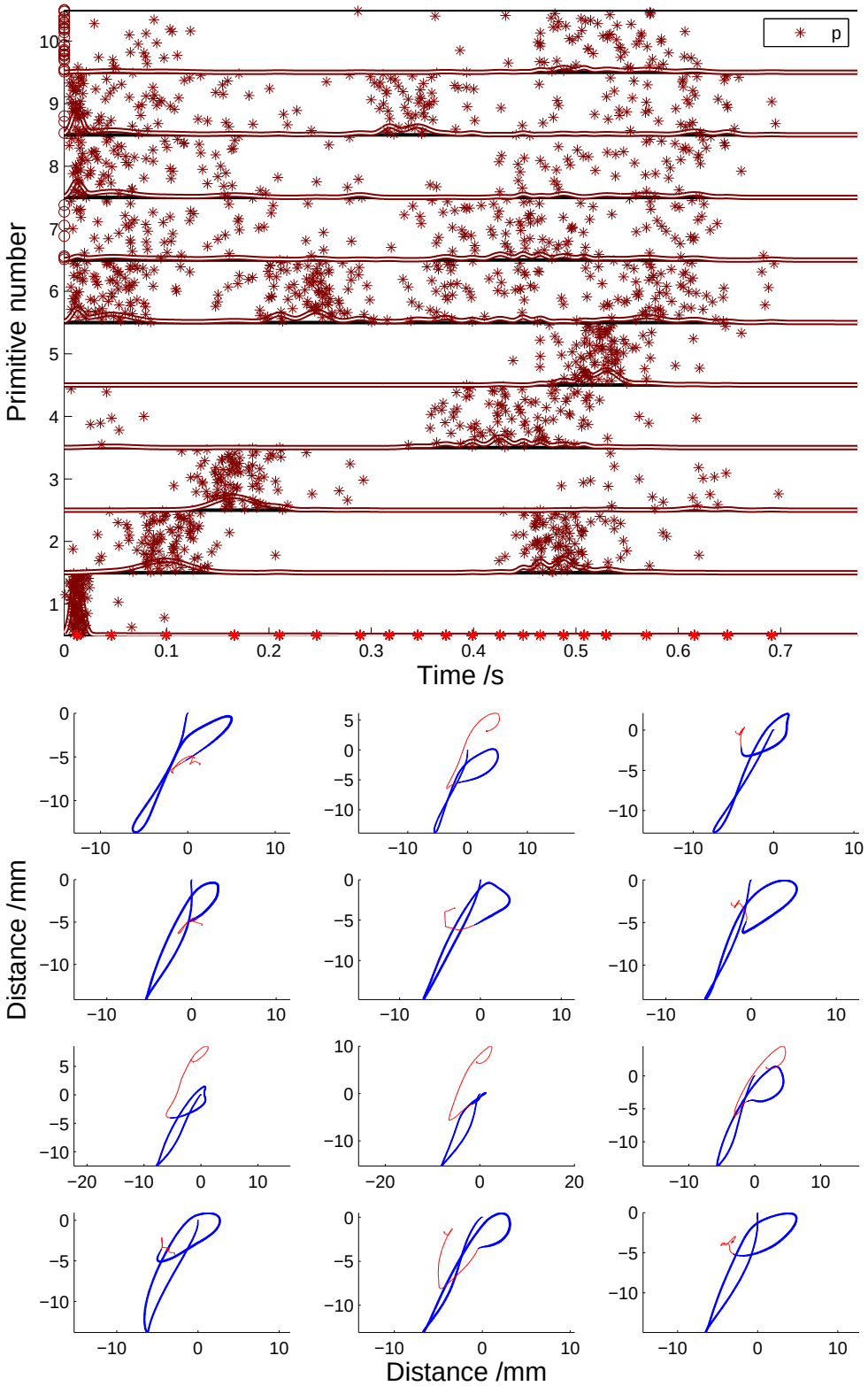


Figure 8.1: 12 generative character samples produced from a model trained upon a dataset composed of the letter *p*. The upper plot shows the generative spike times for the samples. The lower plots show *paper-space* reconstructions of the generative samples, for the purpose of readability assessment.

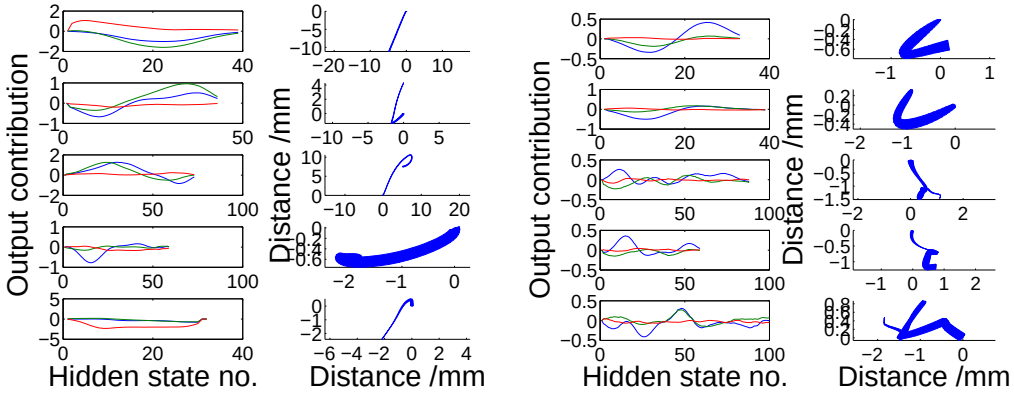


Figure 8.2: Primitives used to generate the samples seen in Figure 8.1. The initial length of the primitives was 20 states, and they were initialised from the data.

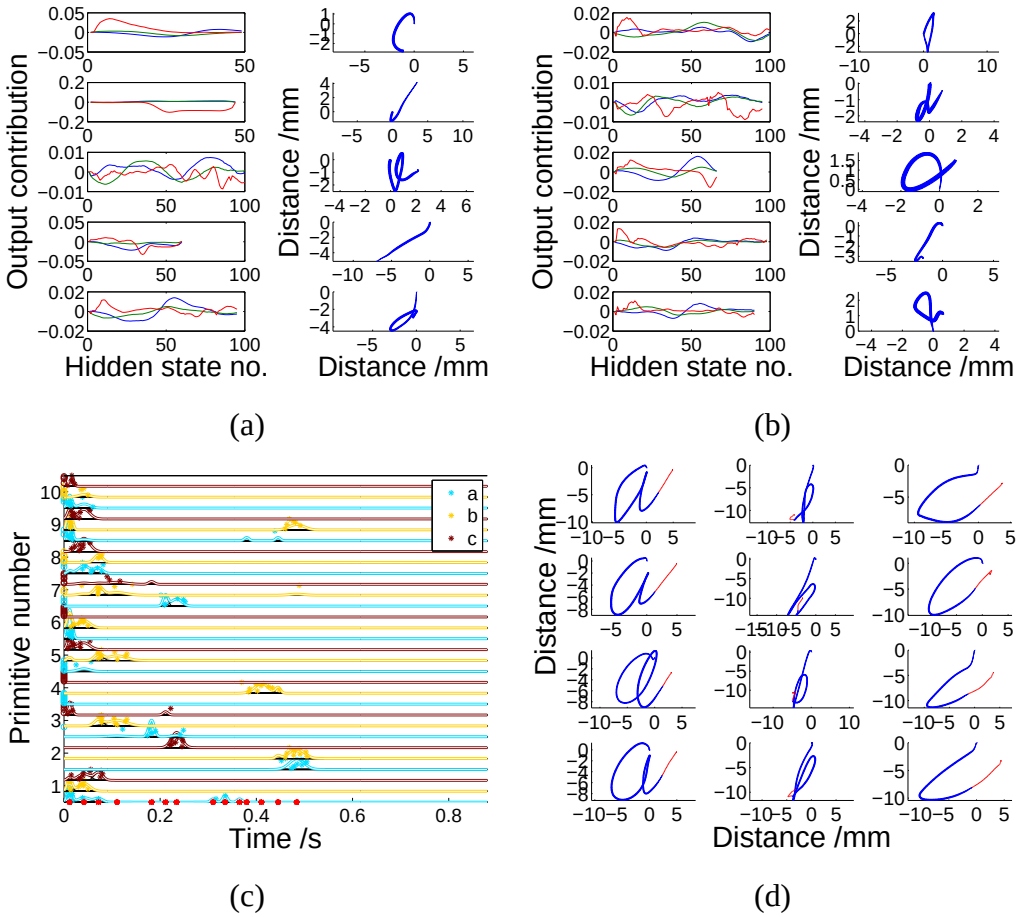


Figure 8.3: A single set of primitives can be used to generate distinctive character samples. Ten primitives are shown in (a) and (b), and were learnt from a training dataset consisting of 30 samples of the characters $\{a, b, c\}$. Generative sampling from the model produces the spikes seen in (c), and the samples seen in (d).

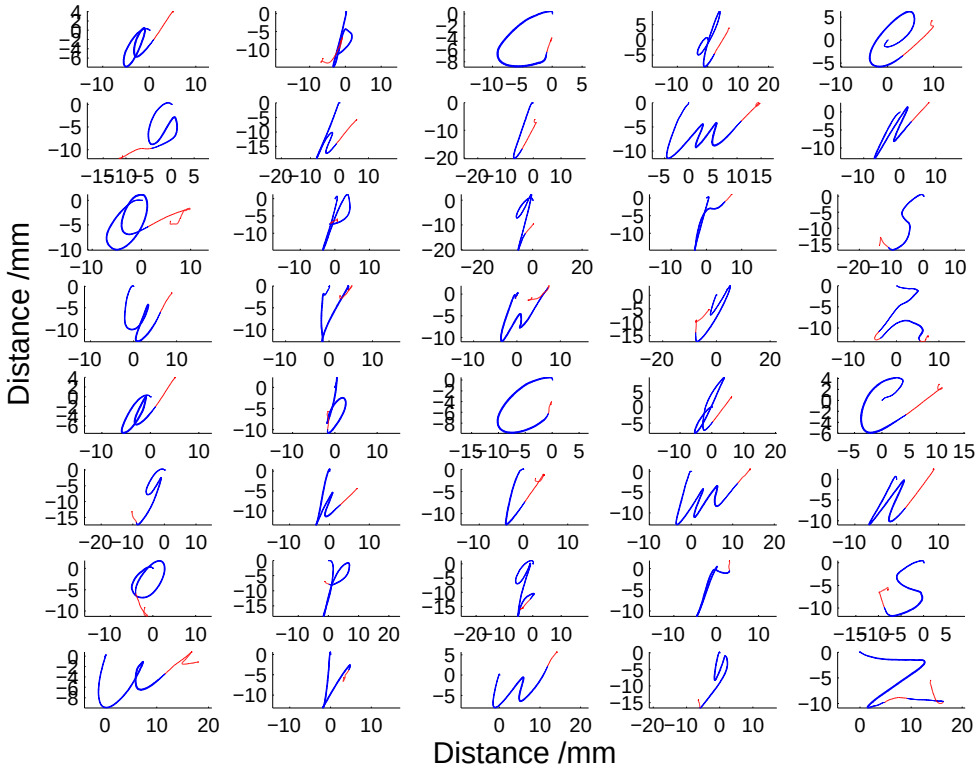


Figure 8.4: 40 posterior reconstructions of a character set containing 20 character types. 20 primitives were learnt, which can be seen in Figure 8.5.

rest of the model produces samples such as those seen in Figure 8.7. These are better, more readable samples and indicate that a further, higher level extension of the model modelling the *mode* of a character would be useful, at least when large numbers of primitives are required.

8.2 Objective measures

One simple objective measure evaluates a generative sample in terms of the probability of fit to the mean character sample, assuming that each time point is independent, and that the distribution of the data is Gaussian. Taking the sum of the log probability over time gives a log probability density measure for the generative character sample. Another measure for the appropriateness of a particular generative sample is that of the variational log likelihood of the model given the sample. These are interesting heuristic measures of the quality of a single character; however they mean little on their own, and it is difficult to compare models, as the state sizes vary depending upon the shifting history. Interestingly, there is not a direct correspondence between these

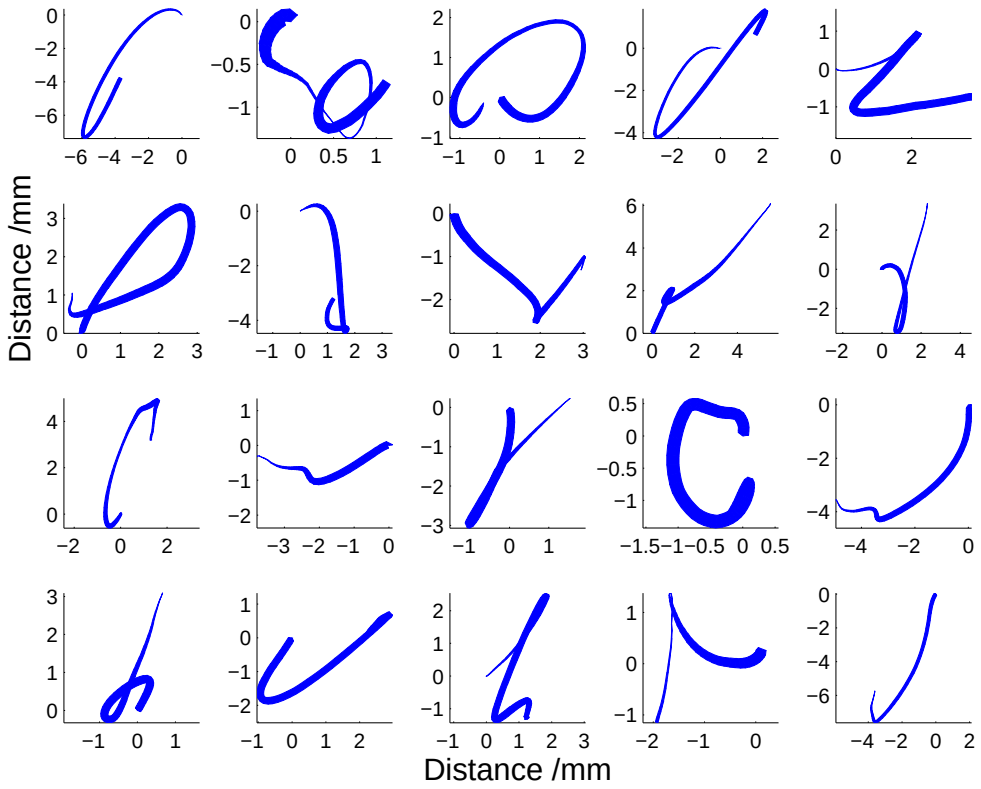


Figure 8.5: 20 primitives learnt from a dataset containing 20 character types. The characters with very little variance seem to be modelled more directly with individual primitives, such as the *c* and the *l*.

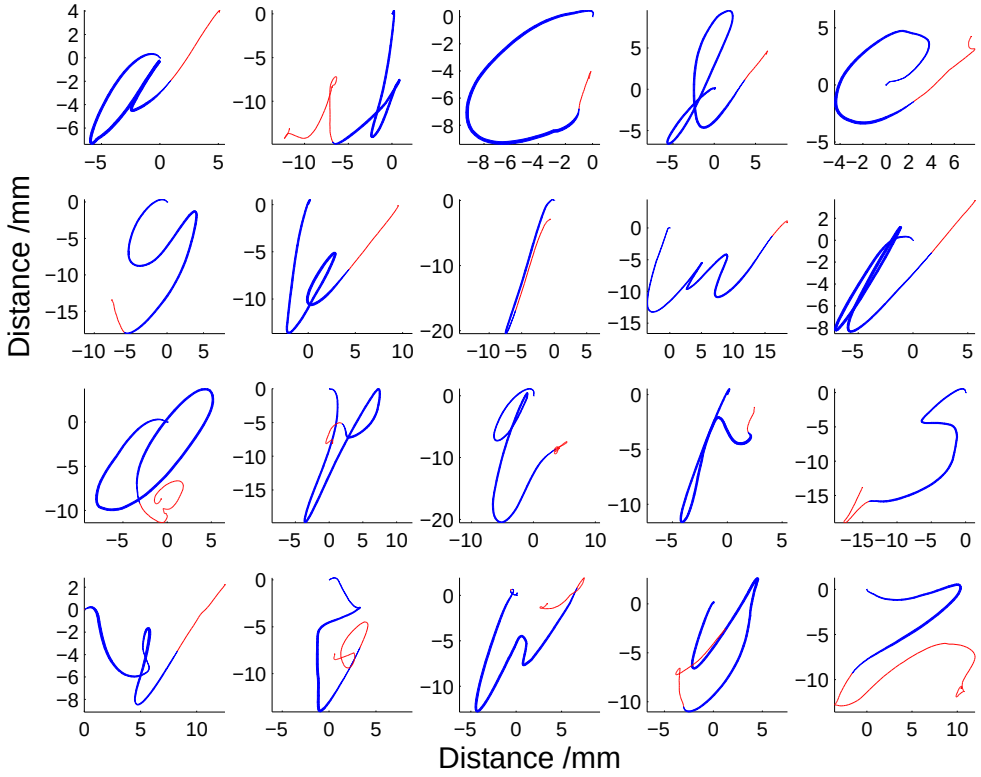


Figure 8.6: The primitives seen in Figure 8.5 are used to generated the character samples seen here. Clearly the characters are not as reliable as in the case where the primitives are modelling a single character, such as those seen in Figure 8.1. This is because the large number of primitives require a more complex timing model to model more of the dependencies between primitives.

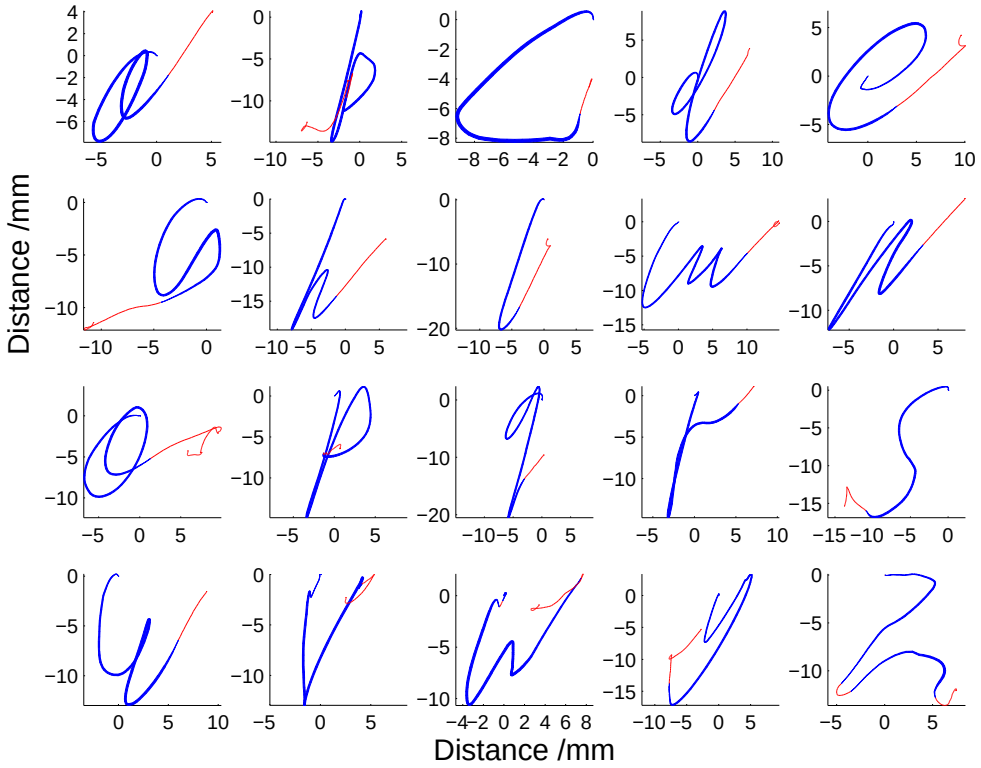


Figure 8.7: Generative samples, as in Figure 8.6, but picking an appropriate hidden state trajectory for the timing model, referred to as a *mode*. This assures that any coupling between primitive presence is captured. These samples are not fully generative, as the modes are conditioned on the training dataset.

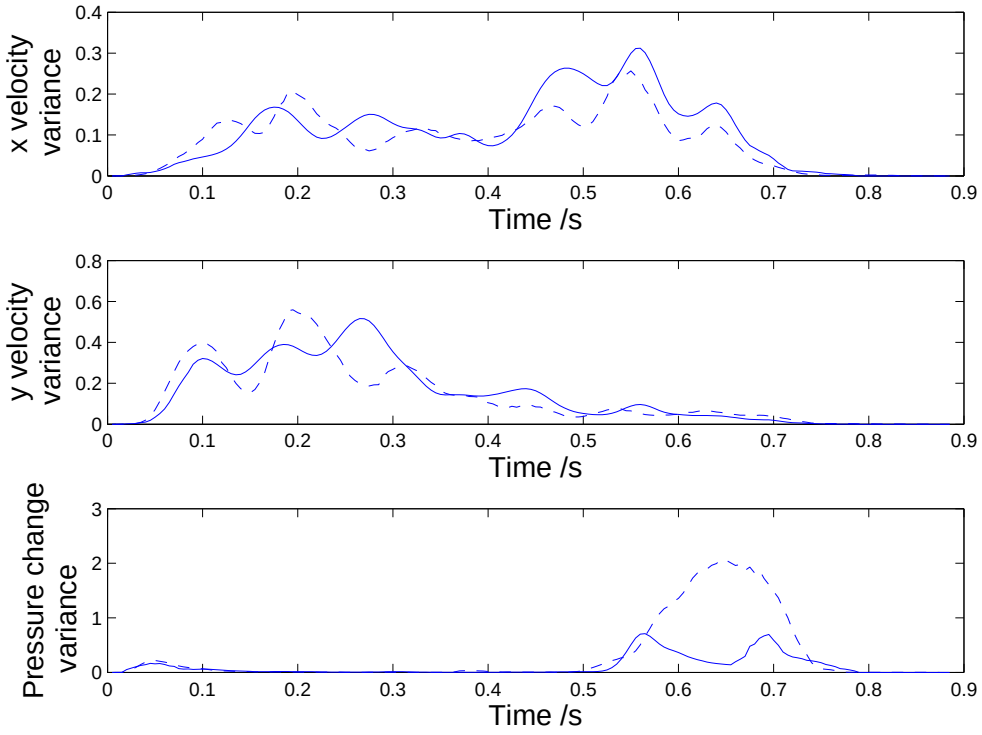


Figure 8.8: Variance profile of generative samples compared to variance profile of dataset for a single character. Three dimensions of data shown in separate axes. Solid line plots the variance over dataset samples at each time point. Dashed line shows the variance over an identical number of generated samples at each time point. Values are in normalised units. The dataset contains 50 samples of the character *p*, and generative samples can be seen in Figure 8.1.

measures and the readability of a sample.

A better measure of the generative performance of the model is to consider the variance profile of the samples produced, and to compare it to the variance profile of the dataset. This can be seen for a single character in Figure 8.8. The dataset used for this example is that shown in Figure 8.1. The variance of the generative samples seems to capture the general shape of the variance of the dataset. All the variance present in the generative samples is due either to timing jitter in the location of primitive activation, or the probabilistic nature of primitive activation.

The velocity profile for a dataset with multiple characters is not captured so well with this model, as can be seen in Figure 8.9. This could be due to a larger number of primitives making the task for the timing model more difficult, effectively meaning the independence assumptions are less valid.

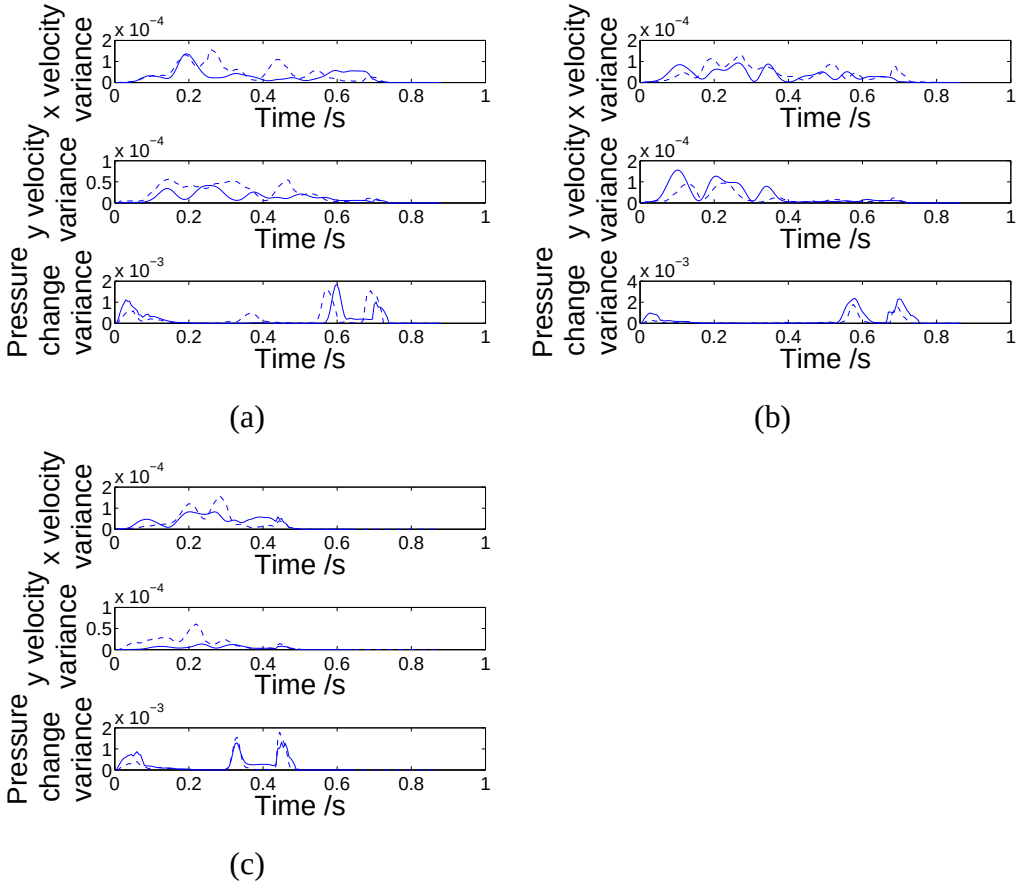


Figure 8.9: Variance profiles for three different characters, {a,b,c}, corresponding to figure labelling. Similar to Figure 8.8, solid line plots the dataset variance, and the dashed line plots the generated samples variance.

8.3 Primitive swapping

The style of a person's handwriting is generally something that remains constant over a long period of time, and is also usually characteristic of the person, despite identical teaching materials. It is possible that the primitive functions learnt using the model presented here might capture the style of a person's handwriting. There are therefore two distinct forms of variation in handwriting, the variation between character samples produced from a single writer, and the variation in style across different writers. A model that distinguishes between these forms of variation would be very useful for encoding of handwriting. If primitives can capture the style of handwriting of a person completely, then the timing code of the primitives is effectively a style invariant code for the characters written. To examine the possibility of *style primitives*, datasets were collected from several different writers. [1]

Normally the coupled model is learnt from a dataset containing multiple character types from a single writer. In this case, there is a single low level primitive model, which is shared for all characters, and multiple high level models, with one for each character. When attempting to compare primitives between writers, we examine a single character type, but have two separate primitive models. In detail, two datasets are compared, one from each writer, and two fHMM models are initialised to values taken from the separate datasets, with the primitive shape initialisations coming from the mean dataset in corresponding positions. The spike data is concatenated from both datasets, and a single timing model is learnt from this data. A single spike prior is produced that couples this timing model with the two primitive models during learning.

The original datasets can be seen in Figure 8.10, showing the difference between style variation, and the variation present in a single dataset. Training a coupled model as described above, produces two separate sets of primitives to be learnt, which can be seen in Figure 8.11.

To examine the inter-changeability of the primitives between sets, the primitives were progressively swapped from one set to the other, but maintaining the timing information. This can be seen in Figure 8.12, where each set of axes shows cumulatively another primitive being swapped to the inappropriate set. This does not result in the characters becoming unreadable; rather the style of the handwriting gradually changes, and with all the primitives swapped resembles closely the handwriting style of the sec-

[1] Thanks to Pooria Zamani of Tehran University for the use of these datasets, which were in fact collected to explore the similarities between Arabic script and Latin script, and whether primitives are shared between the two.

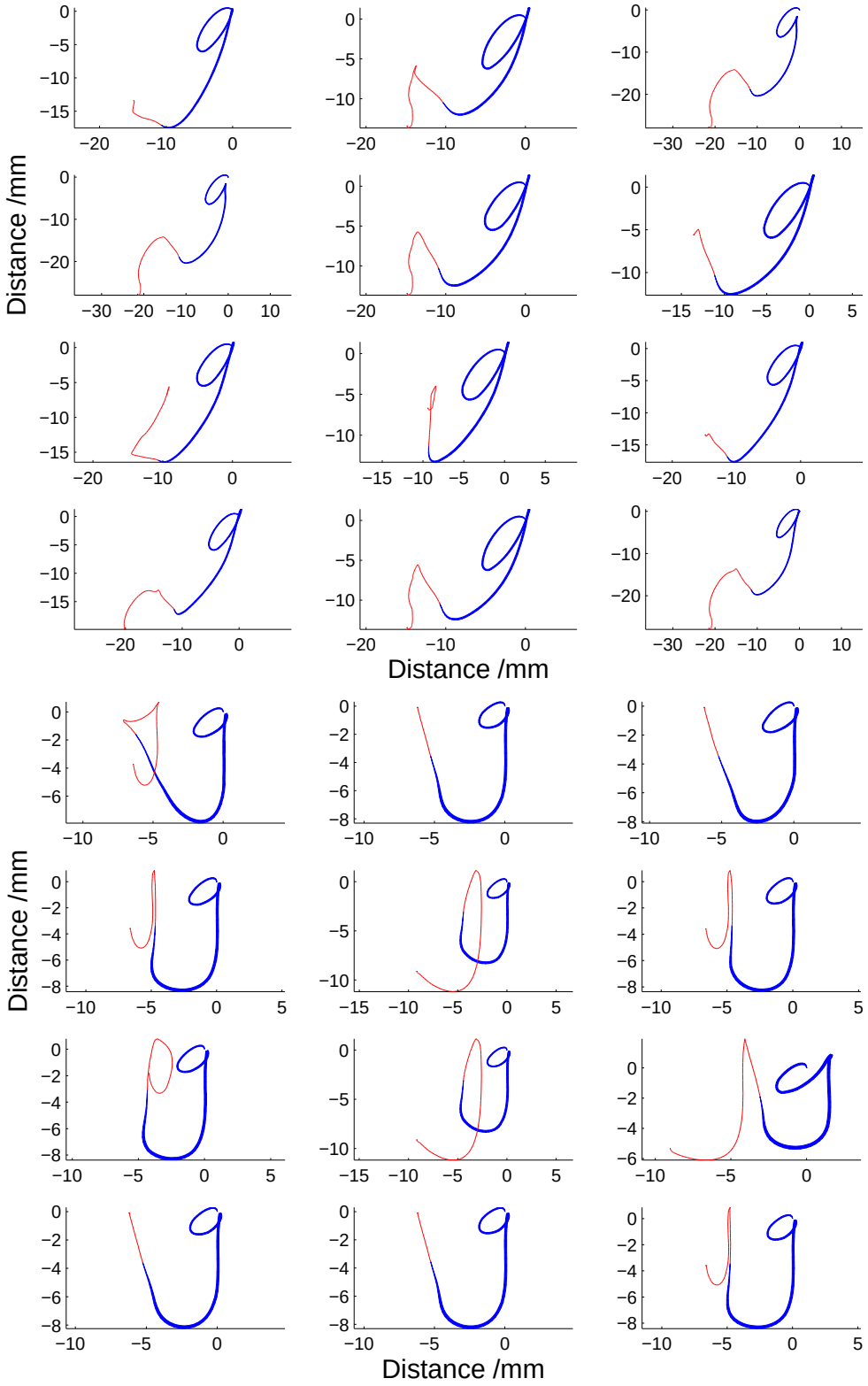


Figure 8.10: Plots of samples from two datasets, collected from different individuals with obvious differences in handwriting style. These reconstructions show the differences between the original datasets, when reconstructed using two different sets of primitives. The upper 12 plots are from writer 1, and the bottom 12 plots are from writer 2.

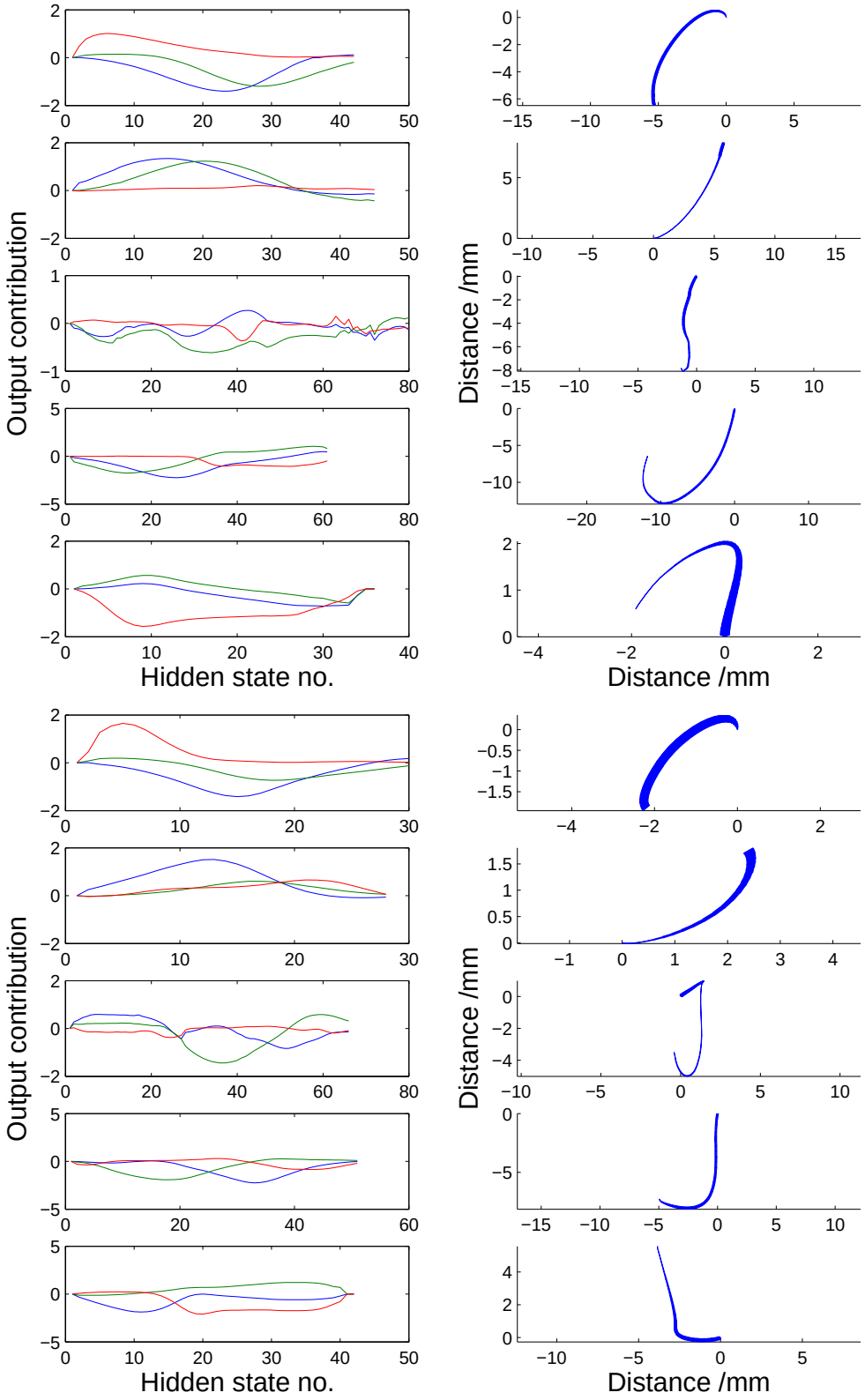


Figure 8.11: Primitives extracted from the datasets shown in Figure 8.10 using a coupled model with 2 primitive models, and a single timing model. There are similarities between the primitives from the two sets because the primitives were initialised in the same way for both models, a single timing model was used, and the data comes from the same character in both cases.

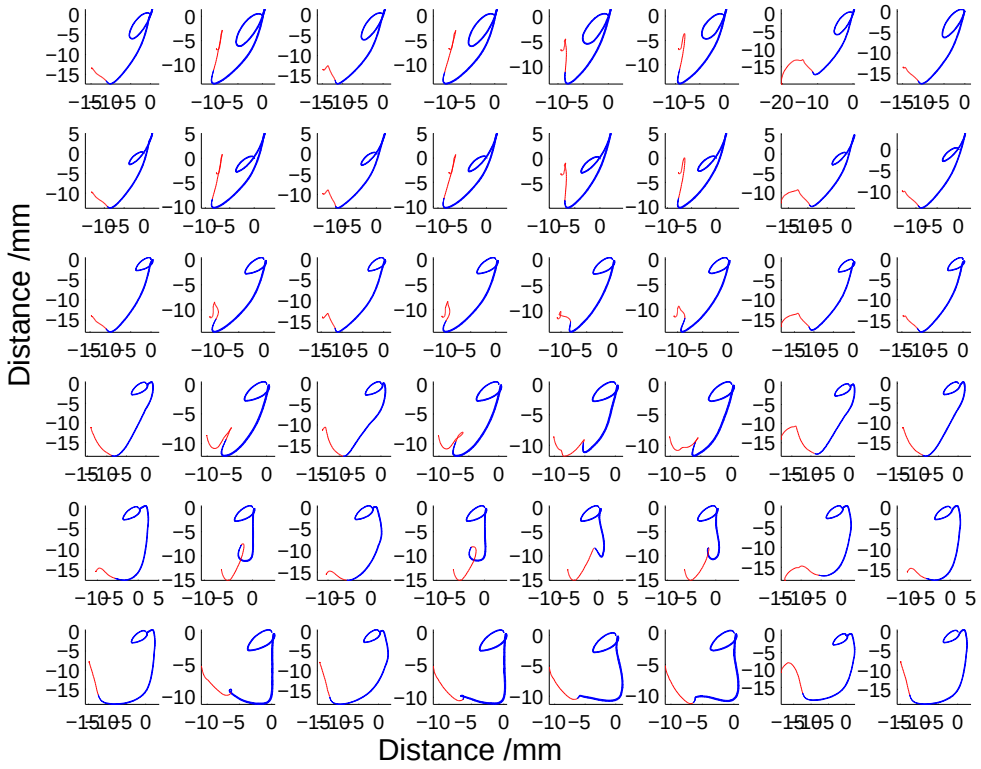


Figure 8.12: Reconstructions of samples, progressively swapping the primitives used from one writer to another the further down the figure. Each row shows the same reconstructions, with cumulatively another primitive being swapped. The top row shows reconstructions of samples from a dataset learnt from the first writer, using the first writer's primitives. The bottom row shows the same character samples, but reconstructed using the primitives from the second writer.

ond writer, despite preserving the timing information from the first writer's dataset.

In this limited example, the primitives are capturing the style of the handwriting, and the timing code therefore provides a style-invariant code for the character being produced, but does account for the handwriting variation present in the dataset.

8.4 Recognition

The handwriting model presented here is a generative model, rather than discriminative, and so is not designed to explicitly distinguish one character type from another. It is possible to perform an inefficient form of character recognition by assessing the likelihood of a particular data sample under different timing models.

The attractiveness of this model is that the low level primitive model can be inferred

without knowledge of the character class (an uncoupled model - see Chapter 7). Given a set of spike times corresponding to the shared primitive activations in the data sample, the timing model can be fit quickly, and its likelihood evaluated. This in principle means that for continuous handwriting, the primitives could be fit in an online fashion without any requirement for knowledge of character partitioning, then the recognition problem would be performed in primitive timing space, which is much more sparse than pen trajectory space. Currently, the recognition is only implemented for single character samples.

To step through the details of the recognition process, a simple case will be examined, in which there are only two characters in the dataset, {p,q}. The training set consisted of 50 character samples, such as those seen in Figure 8.3, along with the 5 primitives inferred. The test dataset consists of 255 character samples, divided between the two characters.

The first step of the recognition process is to learn the uncoupled timing distributions, as these are the appropriate distributions for inferring which character is the most likely when using an uncoupled model. The difference between the timing distributions can be seen in Figure 8.13. In the coupled timing distribution, the appropriate timing model is used to produce a spike prior that informs the primitive model where spikes are likely to occur. This has the effect of tightening the distribution of spikes, and limiting outliers, as discussed in Chapter 7. For recognition, the character class is unknown, and so no spike prior can be used. Therefore the appropriate spike distribution is slightly wider and contains more components, as seen in Figure 8.13.

With the appropriate timing models, learnt from spike times inferred from an uncoupled model, the process of character recognition can begin. A single character sample is picked at random, and the posterior over the uncoupled primitive model is inferred, producing a reconstruction, as seen in Figure 8.14. An interesting aspect of generative models is that a reconstruction can be produced in this fashion, allowing an insight into how the model 'visualises' the character sample. The reconstruction error and the variational log likelihood give goodness of fit measures.

The process of reconstruction using the primitive model involves inference of the posterior over λ_t^m. This can be interpreted as spike times, as seen in Figure 8.15. This spike timing data is then used to evaluate the likelihood of each timing model. The most likely model is assumed to be the appropriate one, and the difference between the likelihoods gives a confidence measure. In this particular case, the most likely character was correctly a *p*, with a confidence interval of 67.5. The lower this interval,

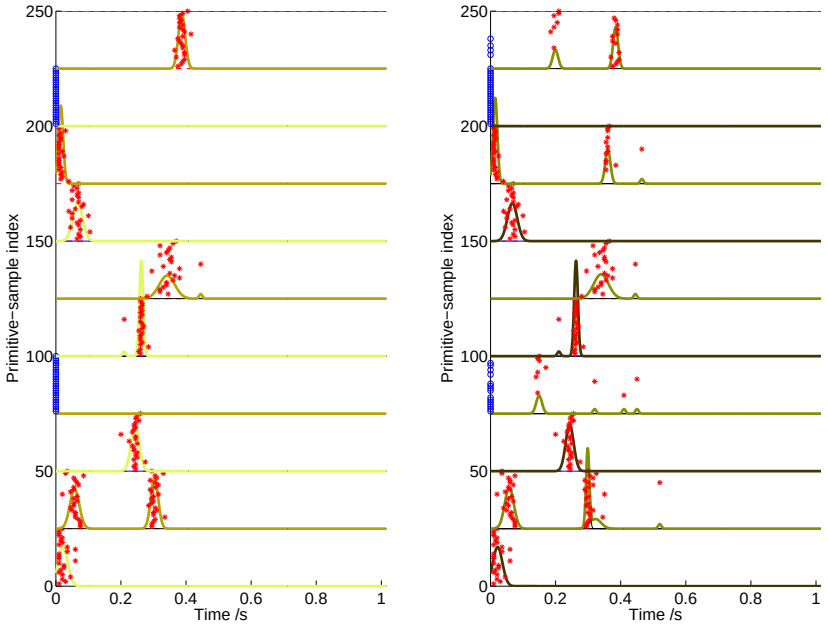


Figure 8.13: Comparison of two spike timing distributions for the same set of primitives. The scatter plot on the left shows the inferred spike pattern using the coupled model to infer spike positions, and the plot on the right shows the spike pattern inferred using the uncoupled model.

the less certain the recognition procedure is.

8.4.1 Character recognition

For the simple case, with only two characters, $\{p,q\}$, and using 10 primitives learnt using the coupled model, the recognition rate on a test dataset containing novel character samples of the same character types in the same proportion was 99%. The simple Gaussian model described in Section 8.5.2.1 was also used to assess the probability that the test sample came from either of the training distributions. The proportion correctly identified by this method was 56%.

The confidence interval of the recognition is the difference between the log likelihood of the best timing model, and the second best model. This interval gives an indication of the certainty of the recognition. The average confidence interval for correctly identified characters was 46.8, whilst the average confidence interval for incorrectly labelled characters was 20.8. Similarly, the average VLL of the uncoupled primitive model (fHMM) fit for correct trials was -31180, against -47200 for incorrect trials. As expected, the fit of the low level model was on average worse in the trials that were in-

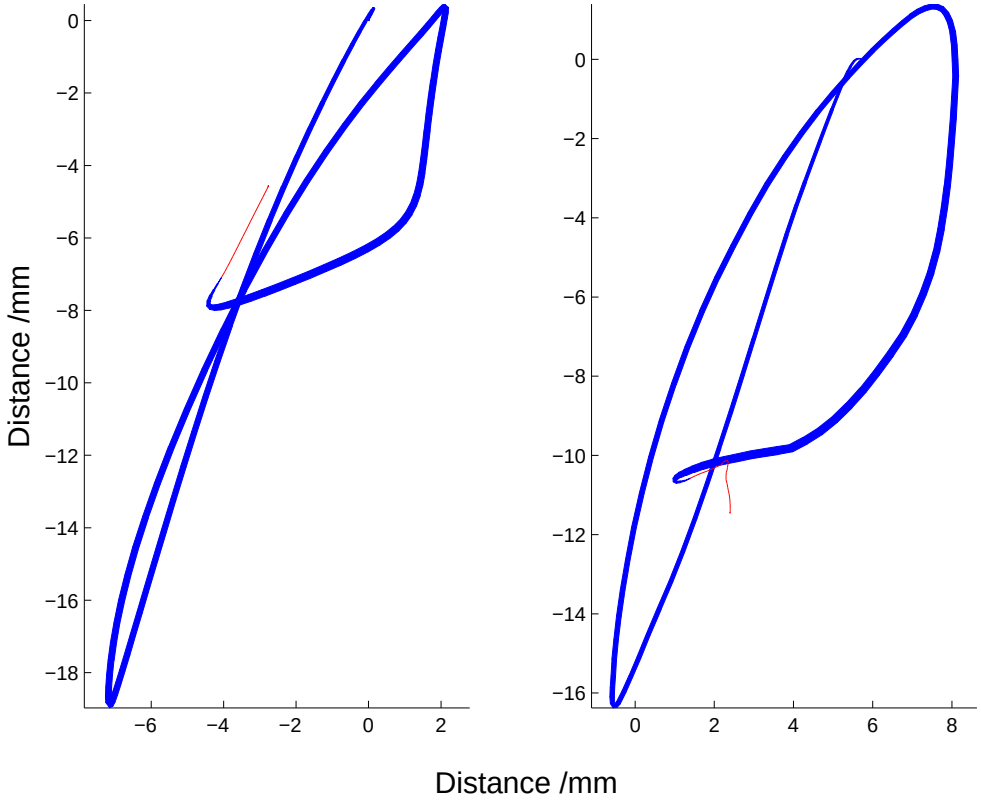


Figure 8.14: Character reconstructions showing the first phase of recognition. The sample on the left is taken from an unseen test dataset containing known characters. The sample on the right is a posterior reconstruction of the same character, using 10 primitives that were learnt from a training dataset containing samples of the characters *p* and *q*.

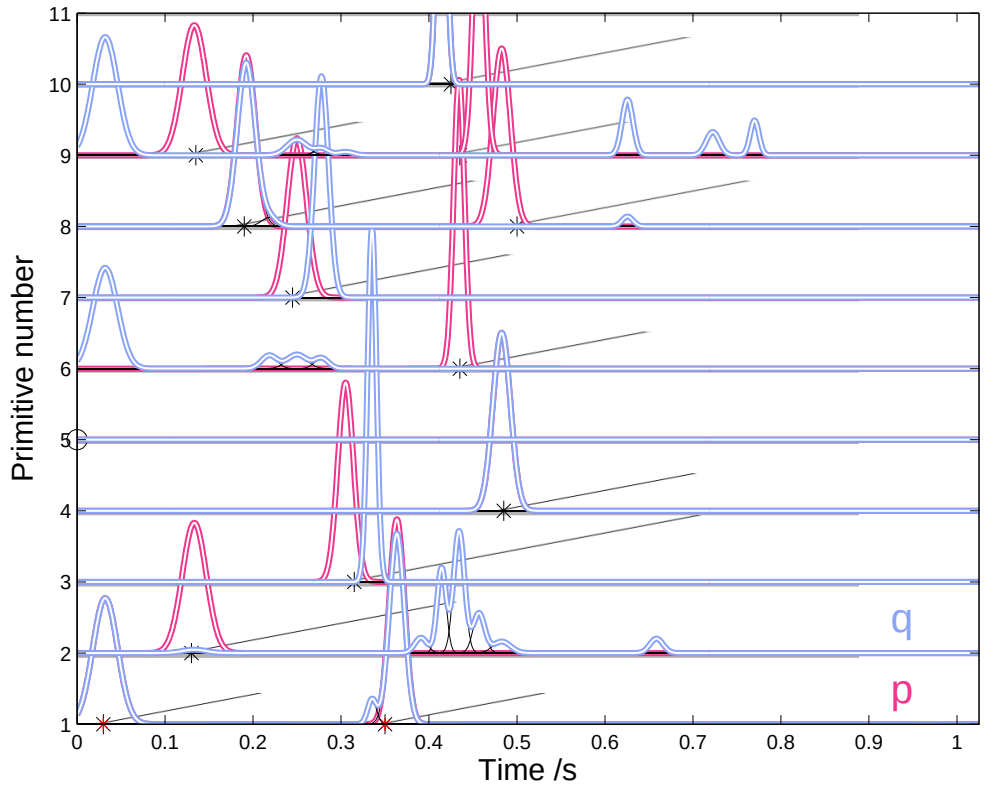


Figure 8.15: The posterior fit of the primitives to a data sample produces an internal 'visualisation' of the character, as seen in Figure 8.14. This visualisation is really an expression of a set of spike times, which can be seen here. Each row represents one of the ten primitives, and the activation of a primitive is shown with a $*$ marker. The progression through the primitive's states is then shown as a diagonal line originating from the activation point. The timing models for the two possible character types 'expect' to see certain primitives activated in certain places, and this is represented by the Gaussian distributions, where the colour of the Gaussian corresponds to the character type, as shown in the key.

correctly identified, and likewise the difference in the likelihood between the possible timing models was smaller on incorrect trials.

On a larger dataset, containing several characters, the recognition performance decreases, as the timing model becomes inadequate at modelling the variance in the data. A better timing model, with more dependencies between the presence of components would help with more generalised recognition performance.

The drawback with using the fHMM for character recognition is that the inference of the posterior takes a significant time. For a single character it took on average 13 seconds to perform the recognition, including fHMM and multiple HMM posterior inference. A greedy initialisation of the fHMM states might improve the performance. A convolution function provides one quick approach to this problem, and a similar model is discussed and compared in Section 8.5.2.3 which uses such a technique. Potentially a combination of the two approaches could provide a solution to the speed problem. Alternatively, approximate inference, or semi-convergence could be explored for speed improvement. Additionally, the factorial nature of the fHMM would lend itself towards parallelization, which would improve the performance further.

8.5 Comparison with other models

Several primitive based models were discussed in Chapter 2. Some models consider primitives to be time-slice components, rather than time extended blocks. In this section we look at some related models, and compare the performance of these models with the primitive based model presented earlier.

8.5.1 Objective measures

8.5.1.1 TAMPE and TASDPE

Model performance can be assessed in different ways; however the most common model independent assessment compares the reconstructed samples to the original ones, calculating the reconstruction error. For generative models, this cannot be done as there are no exact original samples to compare with. A similar assessment for generative samples is to look at the statistics of the generative data, over many samples. The mean and the variance over all the samples can then be compared to the mean and variance of the original dataset, or a test dataset, at each time point. These measures create error profiles for the character, the absolute sums of which provide objective measures

for the performance of the model. The total absolute mean profile error (TAMPE) and the total absolute standard deviation profile error (TASDPE) are used throughout this section. Formally,

$$\text{TAMPE} = \sum_{t,d} \text{abs}\left(\mu(X) - \mu(\mathbf{Y})\right) \ , \tag{8.1}$$

where

$$\mu_{t,d}(\mathbf{Y}) = \frac{\sum_n^N \mathbf{Y}_{t,d}^{(n)}}{N} \ , \tag{8.2}$$

and $\mathbf{Y}^{(n)}$ are the training or test samples, $X^{(n)}$ are the generated samples, and there are N of each. Similarly,

$$\text{TASDPE} = \sum_{t,d} \text{abs}\left(\sigma(X) - \sigma(\mathbf{Y})\right) \ , \tag{8.3}$$

where

$$\sigma_{t,d}(\mathbf{Y}) = \sqrt{\frac{\sum_n^N \left(\mathbf{Y}_{t,d}^{(n)} - \mu(\mathbf{Y})\right)^2}{N}} \ . \tag{8.4}$$

To summarise the results, here is a table showing the performance of the various models compared in the following section. A prefix of *tr* means a comparison of the generative results to the training dataset, and a prefix of *te* means a comparison of the generative results with an unseen test set which contains samples of the same class of character as in the training set. The size of both sets and the number of generative samples produced was equal. The model definitions are detailed in the following sections.

	Reconstruction error	trTAMPE	trTASDPE	teTAMPE	teTASDPE
Test Set	-	15.18	42.37	-	-
Simple Gaussian Model	-	14.09	14.69	21.66	50.50
Covariance Model	-	10.05	10.18	17.60	44.56
PCA/Covariance Model	-	16.52	15.56	21.13	51.01
Matching Pursuit model	0.0416	-	-	-	-
Uncoupled fHMM	0.0603	18.97	36.83	21.32	53.94
Coupled fHMM/Timing model	0.0552	10.80	29.27	16.70	58.48

It is interesting to note that the mean and standard deviation profile errors are smaller for the Gaussian models than for the test set when comparing the generative samples with the training set. This shows that the Gaussian models produce better examples of the training set than the test set does. This means they are *over fitting* the training set. Comparing the generative samples to the test set gives a better evaluation of the model performance. Here we can see that the complete covariance model performs the best, although it has other short comings which are discussed in Section 8.5.2.2.

The reconstruction error for the Matching Pursuit/Gradient descent algorithm was smaller than for the fHMM based model, due to the possibility of using a scaling factor. In this case, 10 atomic functions were used and 650 occurrences of atoms were fit to the data, equalling on average, one of each atom per character. Although the number of occurrences of the primitives in the fHMM based model are not constrained, the MAP number of spikes was of the same order as this. Without a generalising model across the character samples, the Matching Pursuit algorithm is not generative, and so the TAMPE and TASDPE measures are not calculable. (They could be calculated on the reconstructed dataset, and as such would seem to strongly over-fit the data.)

The performance of the uncoupled and coupled models was very similar by these measures, and they are both comparable to the PCA/Covariance model, where 10 PCA components were used. The main advantage of the fHMM primitive based model is that the same primitives (or components) can be used to model parts of other characters, whereas the covariance based models do not allow offsets, and so the components are only going to be suitable for a single type of character, or very similar ones such as a *b* and an *h*.

The coupled model performed slightly better than the uncoupled model, producing samples that were closer to the mean character sample from both the training set and the test set.

8.5.2 Alternative models

8.5.2.1 Simple Gaussian model

A straightforward model of a character is to assume that the data is independent at each time point, and modelled by a Gaussian distribution.

$$P(\mathbf{Y}) = \prod_t \mathcal{N}(\boldsymbol{\mu}_t, \sigma_t) \ , \tag{8.5}$$

where $\boldsymbol{\mu}_t$ and σ_t are learnt using standard estimators.

This distribution is parameterised by taking the mean and variance of the character samples. Sampling from such a model produces generative characters as seen in Figure 8.16.

The wobbly nature of the samples produced by this model is due to the independence assumption implicit in the model, meaning that the samples at adjacent time points are independent. The shape of the samples in general however is reasonably

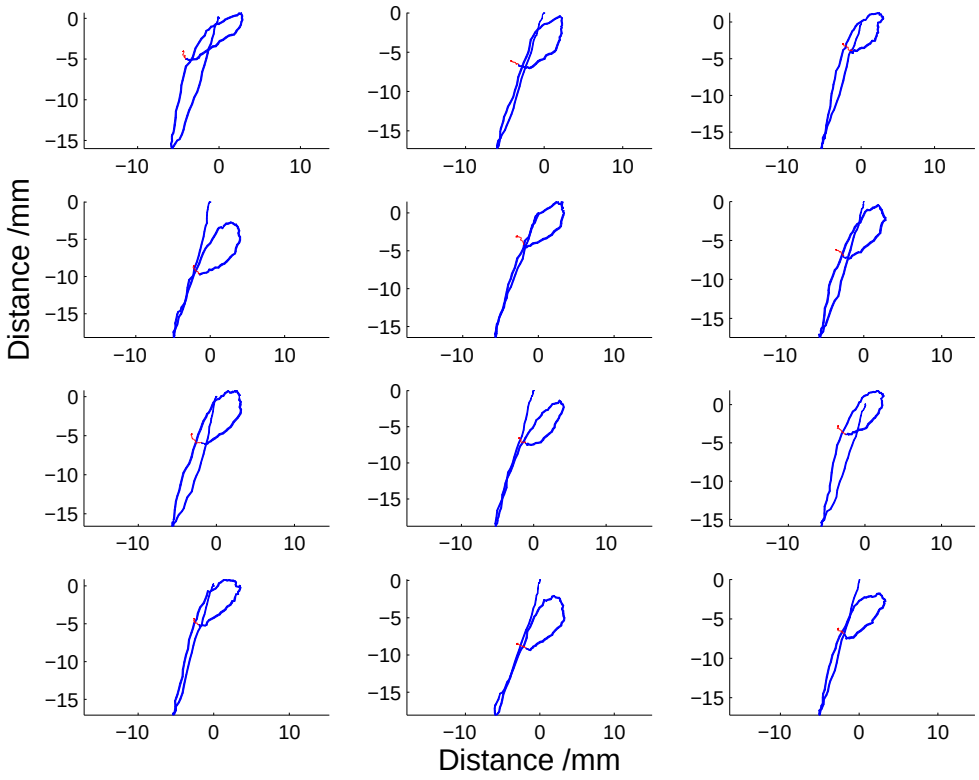


Figure 8.16: Samples from a simple Gaussian model of a character, as described in Section 8.5.2.1. The generative samples appear 'wobbly' due to the lack of dependence between time points. Although this simple model produces good results for a single character, it should be noted that it does not generalise across characters, meaning that the parameters learnt for a *p* are not useful for the modelling of a *g* in any way except average magnitudes. The primitive based model presented in this thesis uses the learnt primitive functions in multiple characters, and the inference procedure produces generalised primitives, as discussed in Section 5.5.

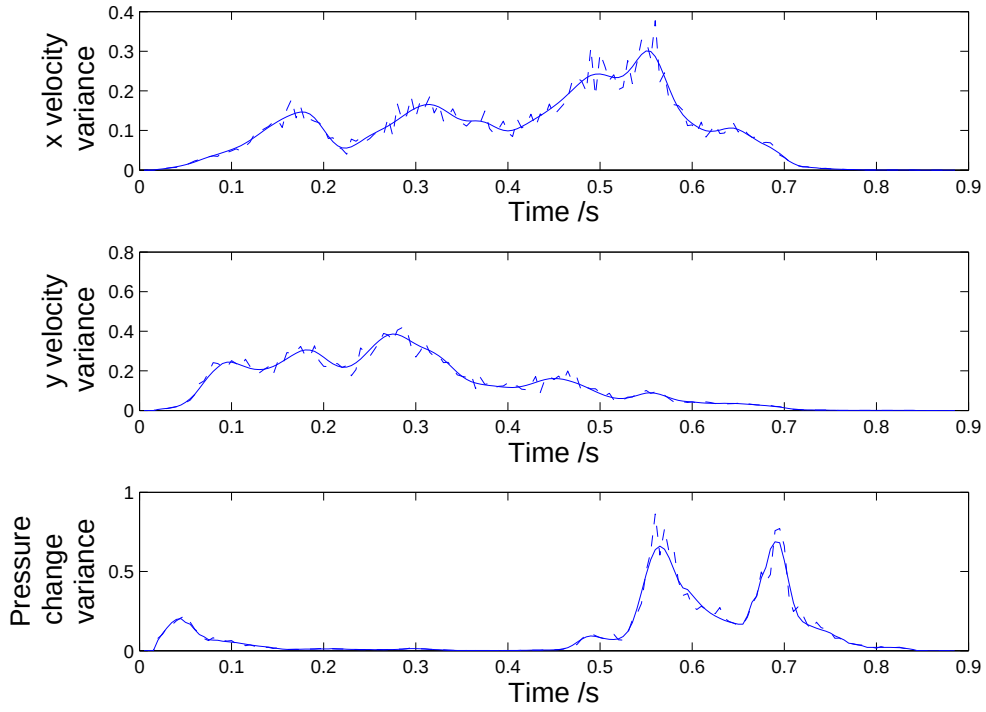


Figure 8.17: The variance profile of generative samples produced from a simple independent Gaussian model of a character. The general shape of the variance profile fits the original data, however, due to the lack of local dependencies, the generated data is very messy.

good, and they are easily readable. These results are mirrored in the variance profile of the generative samples, as is seen in Figure 8.17.

The same effect is seen when the covariance matrix is used, as can be seen in Figure 8.18

8.5.2.2 PCA/Complete covariance model

A variation on the above model is to consider a single multivariate Gaussian that captures the covariance across all time points.

$$P(\mathbf{Y}) = \mathcal{N}(\boldsymbol{\mu}, \sigma) \,, \tag{8.6}$$

where $\boldsymbol{\mu}$ is the mean of the character samples at all data points, and σ is the covariance of all data points in the character. $\boldsymbol{\mu}$ is a vector length DT, and σ is a $DT \times DT$ matrix. This is a very good model as it captures all the first order variance in the data. Samples and the variance profile from this model can be seen in Figure 8.19.

The problem with this model is that for each character, it requires $D^2 \times T^2 + D \times T$ parameters. This can be reduced by taking the principal components (PCA) of the

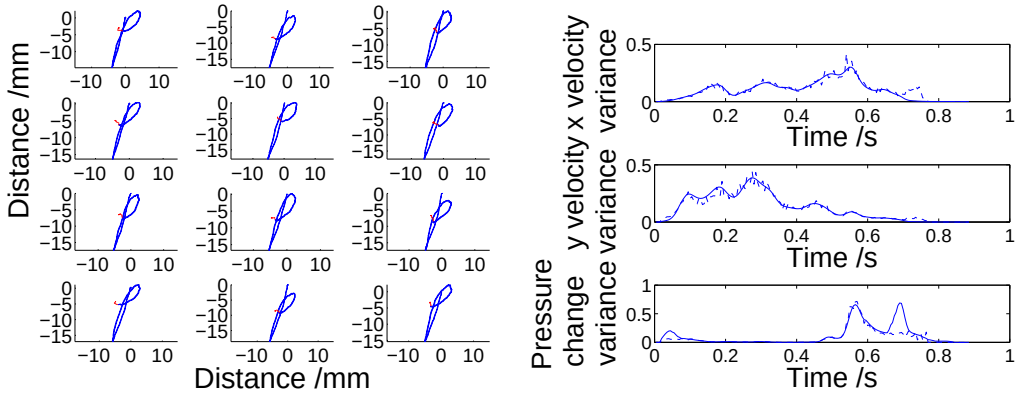


Figure 8.18: Samples from a character model capturing the covariance at each time point. The same lack of dependence between time points as in Figure 8.16 produces the 'wobbly' samples.

covariance matrix, but this reduces the variance of the samples. An example can be seen in Figure 8.20, where 5 principal components were used to reduce the dimension of the covariance matrix to 5×5. A reconstruction of the components selected can be seen in Figure 8.21.

Although the PCA/complete covariance model works well on a specific dataset, it does not generalise well. For instance, when presented with a novel character, it cannot represent it that well, as can be seen in Figure 8.22.

Using a set of PCA components from a dataset of the character *p* to model the variance of a dataset of the character *q* for instance does not work very well. However, because the PCA components are modelling only the variance of the dataset, and not the mean, the generative character samples appear quite good, although that is simply because the mean character is easily readable. Performing a similar experiment using the primitive model, whereby 10 primitives were extracted from the dataset containing the character *p*, and then fit using a single E-step to a dataset containing the character *q* produces reconstructions such as those seen in Figure 8.23. These samples are on first impression worse than those produced with the PCA model; however, the primitive model is also modelling the mean of the dataset as well as the variance, which is why the samples look messy. The variance profile of these reconstructions is closer to that of the dataset, than when using the PCA model, as can be seen in Figure 8.23.

Crucially, the PCA model must have a character specific set of mean and variance parameters, which will not generalise well to other characters. Using the primitive model however allows primitives to be trained on multiple characters, which makes them more generalised, and able to model novel characters, as shown in Chapter 5. The

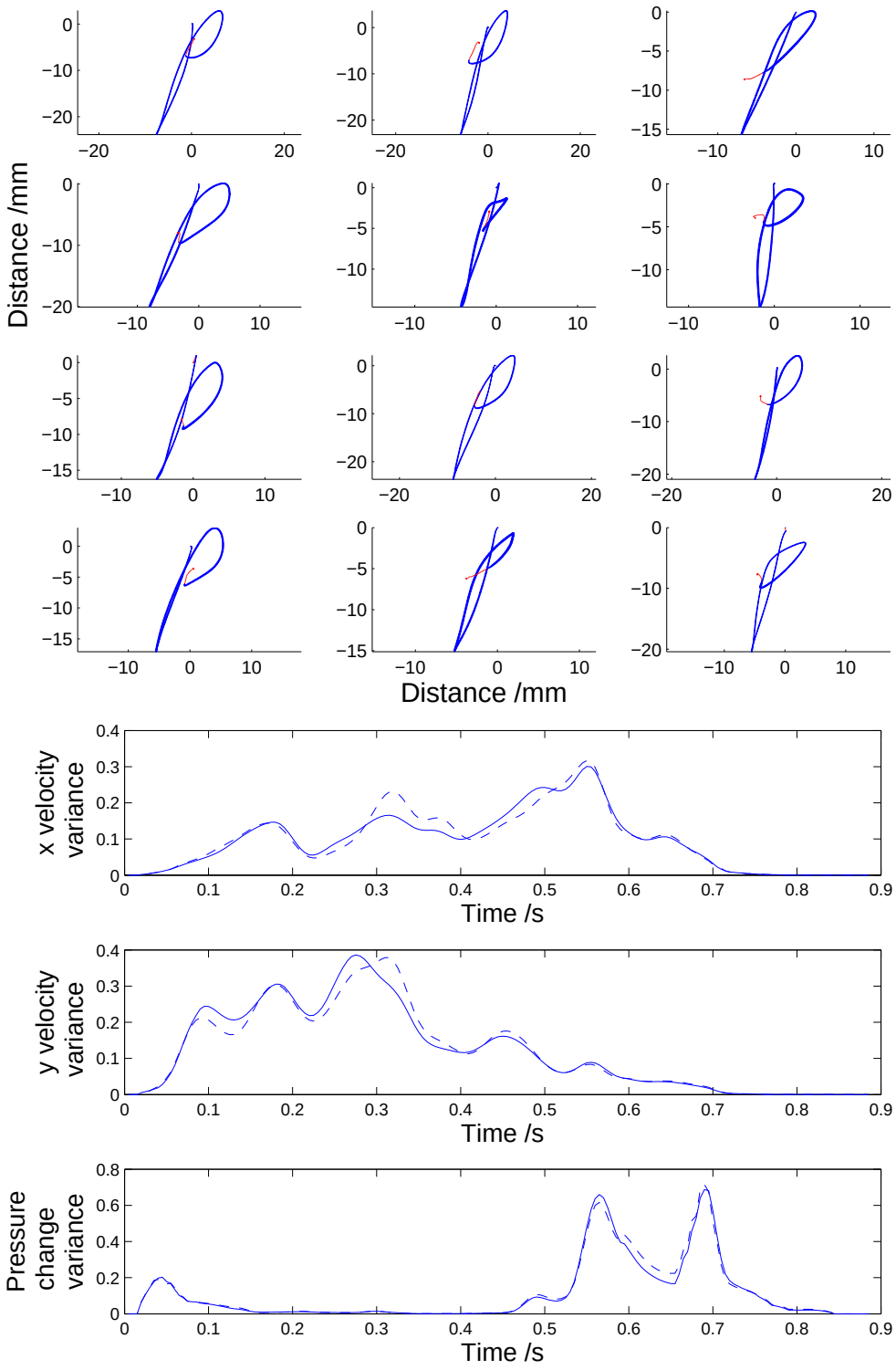


Figure 8.19: Samples produced from a complete covariance model of the data. This model captures all the first order covariance of the data, but requires a large parameter matrix.

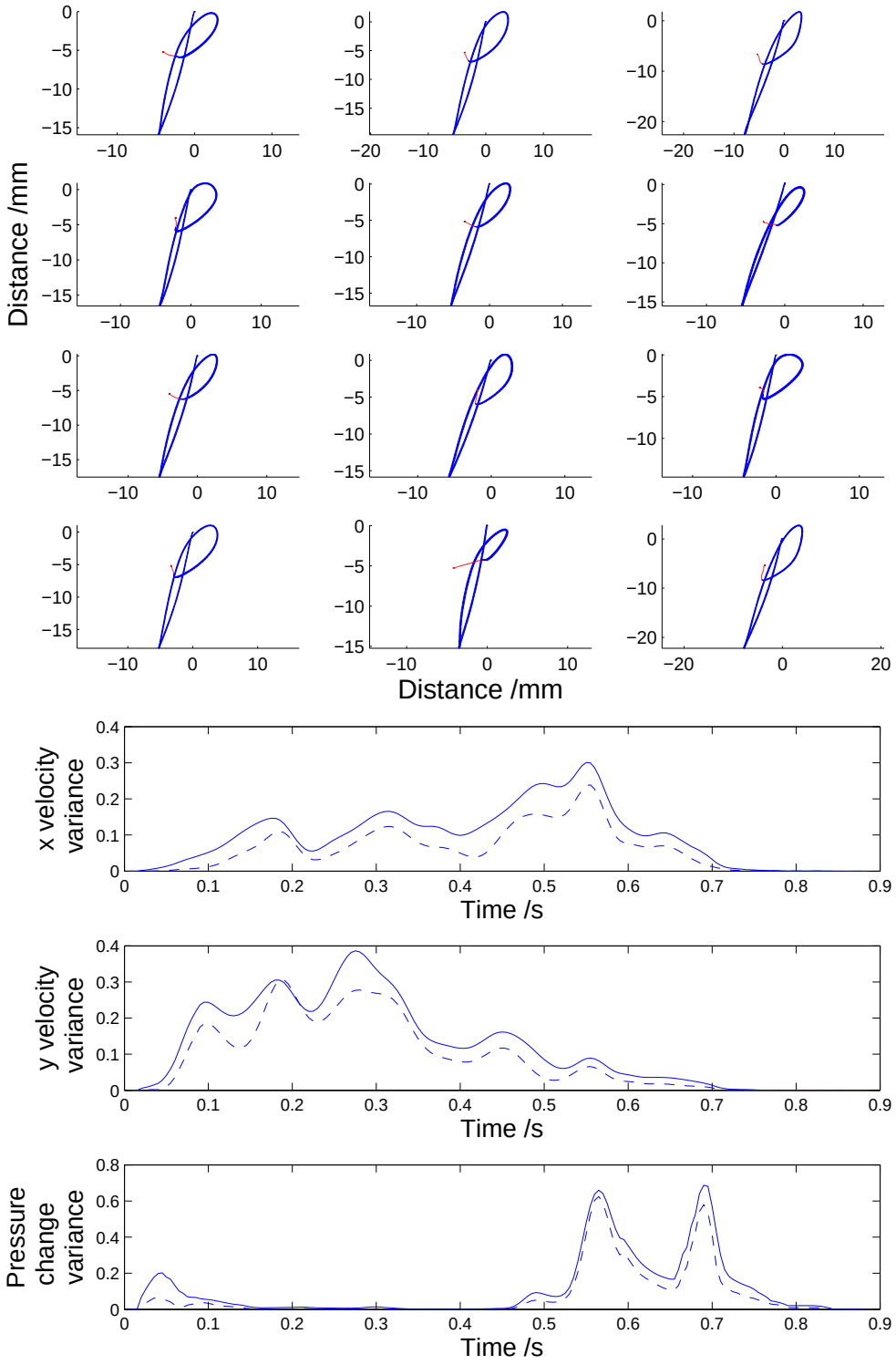


Figure 8.20: Samples produced using the complete covariance model described in Section 8.5.2.2, but using PCA to reduce the dimension of the covariance matrix to 5×5. This reduces the number of parameters needed, but also reduces the variance of the samples produced, as can be seen in the variance profile plots. A reconstruction of the principal components selected using PCA can be seen in Figure 8.21.

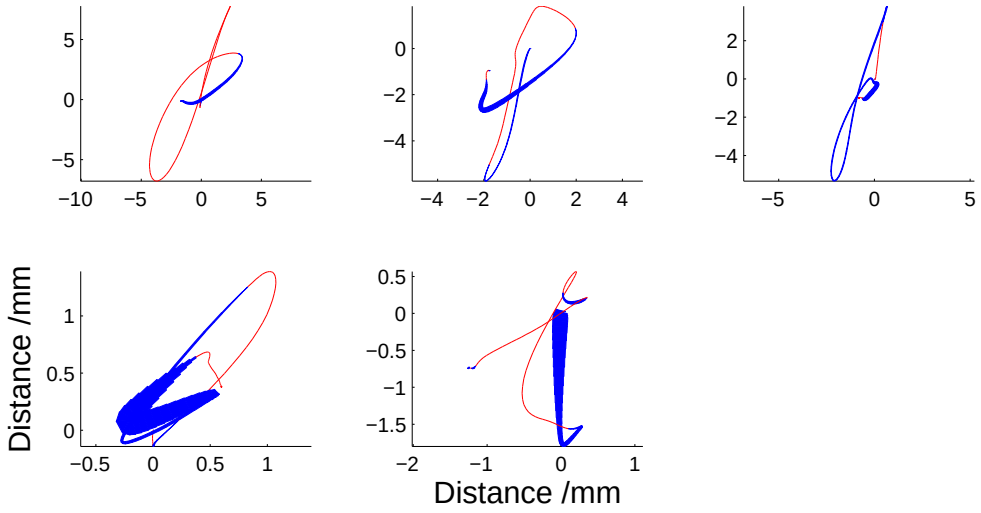


Figure 8.21: PCA components used for the samples shown in Figure 8.20. Here the components are shown as if they were used in isolation, and reconstructed on paper.

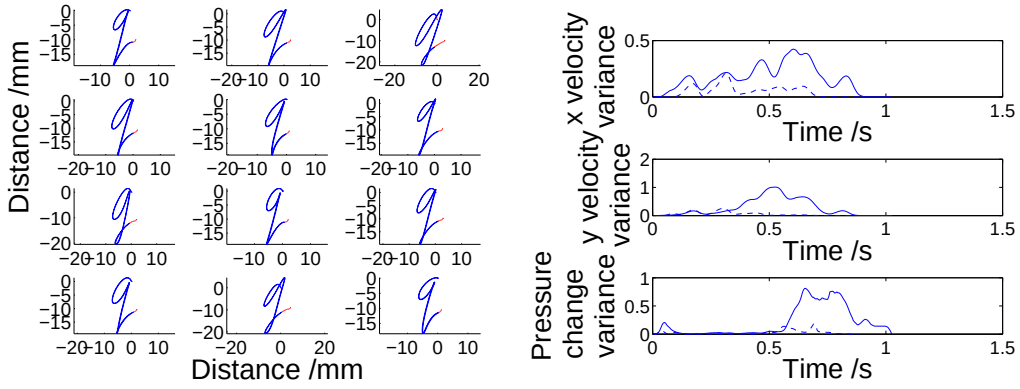


Figure 8.22: Using PCA components to model the variance on a different character does not work very well. PCA components for the character *p* were used to model the variance of a dataset containing the character *q*. The mean of the *q* dataset was used, which is why the characters are readable, however the variance profiles of the generative samples do not match those of the dataset.

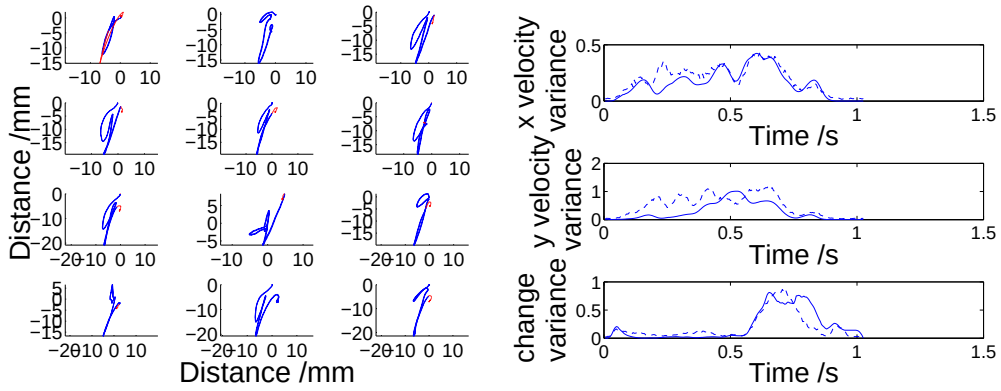


Figure 8.23: Using 10 primitives trained on the character *p* to reconstruct samples of the character *q* produces fairly messy character reproductions, however the variance profile of the reconstructions is reasonably close to the dataset variance.

primitives used in the example shown in Figure 8.23 were trained on a dataset containing only one character, for comparability with the PCA model; however, if the primitives are learnt from a dataset containing the characters $\{a,b,c,d,e,g,h,l,m,n,o,p\}$, then the reconstruction of a novel character, *q* is much better, as is the variance profile, as seen in Figure 8.24, showing the manner in which the primitives become more generalised as the training set expands. The PCA model would not be able to do this, as each set of components needs to be trained on a separate character type.

The posterior reconstructions seen in Figure 8.24 were used to train a timing model which could then produce generative samples, as seen in Figure 8.25. These generative samples are very bad because the primitives were not learnt on the dataset used to produce the timing model, and also the primitives were not fit to the dataset in a coupled manner, because no timing prior was suitable without adapting the model to the dataset.

It may seem as though the samples seen in Figure 8.22, produced by the PCA/Gaussian model are more legible than those produced in Figures 8.24 and 8.25 by the primitive model, however this is because the the PCA/Gaussian model has the mean character as a 'base-line', whereas the primitive model has a zero base-line, and therefore has to model the mean as well as the variance of the dataset. A fairer comparison would be to subtract the mean from the data and learn primitives to model the residual data. In this case however, the mean character would need to be stored along with the common set of primitives, effectively increasing the number of parameters required with the number of characters in the dataset. Also it would be bizarre for this to be the case in biology, because if it was possible for the brain to reproduce an 'average' character, it

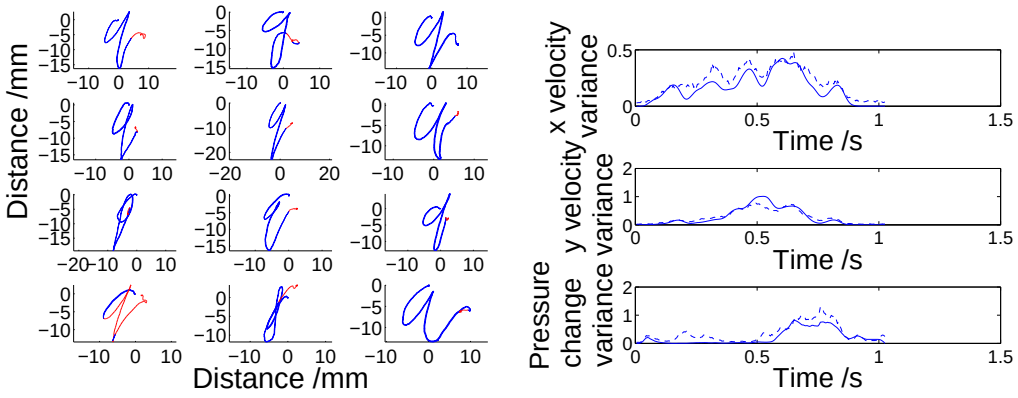


Figure 8.24: Using 10 primitives trained on a dataset containing 12 character types, reconstructions of a dataset containing the novel character *q* is possible, and the variance profile closely matches the dataset. The samples produced here should be contrasted with those in Figure 8.22, produced by the PCA/Gaussian model, which vary little from the mean character, which is not part of the generalisation experiment.

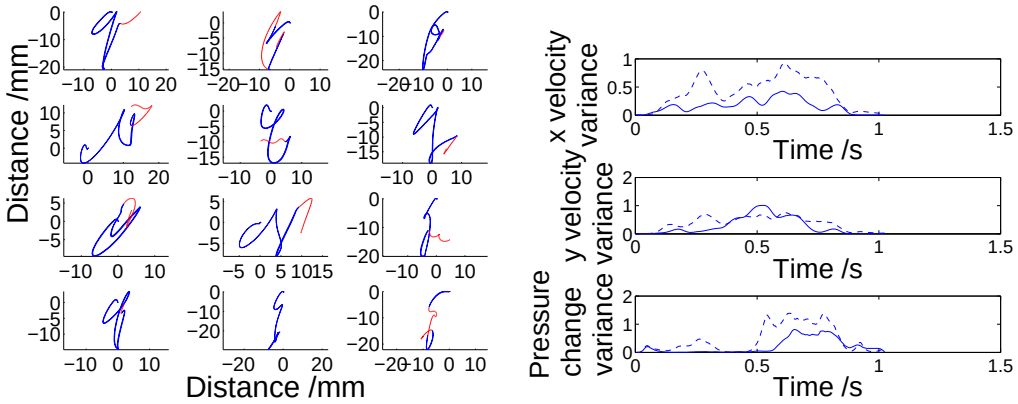


Figure 8.25: Using 10 primitives trained on a dataset containing 12 character types, a complete reconstruction of the a dataset containing the character *q* was performed. Samples of this reconstruction can be seen in Figure 8.24. A single combined HMM timing model as described in Section 6.2.6 was trained on this dataset, so that generative samples as seen here could be produced. The samples are poor because the primitives were not learnt for this particular dataset, and their fitting to the characters was done in an uncoupled fashion, without a timing prior. The variance profile of the generative samples is worse than that of the posterior reconstruction of the dataset, as seen in Figure 8.24, but is still better than that of the PCA/Gaussian model seen in Figure 8.22.

would not then need to superimpose primitives on top of the reproduction.

8.5.2.3 Matching pursuit atomic decomposition

Chapter 2 discussed several primitive inspired models, with various definitions of what exactly a primitive is. The primitives used by the model presented here can be summarised as time-extended, fixed arbitrary functions with offsets, as described in Chapter 3. This definition of primitives is very similar to the atomic functions in (Lewicki and Sejnowski, 1999; Smith and Lewicki, 2005). Lewicki *et al* define atomic functions as an adaptable basis set of functions that are fit to certain areas of a signal using a convolution function to find time offsets and scaling factors. The signals studied by Lewicki *et al* are natural sound waves, with the argument that an efficient cochlear encoding may well be based on an equivalent spike based encoding, where the timing of the spike corresponds to the timing of a particular atomic function, or cochlear filter. This is in principle the same motivation as for the offset primitive function model described in Chapter 3; however there are some differences between the models, and in particular their implementation.

The main drawback of the matching pursuit model could also be thought of as an advantage over the fHMM based model. The inclusion of a scaling factor for each occurrence of an atomic function allows more generalised atoms, thus allowing a better reconstruction of a dataset. However the timing code produced by the matching pursuit model is then far more complex, as it needs to include a scaling factor with each spike. Modelling this timing code would require a more complex class of model than the mixture of Gaussian based timing models examined here. Without such a higher level model to generalise across character samples, it is not possible to use the matching pursuit model generatively, and so it can only be compared in terms of reconstruction error.

To help with the comparison between the fHMM based model presented here, and Lewicki's atomic function model, a brief overview of Lewicki's model is included here.

The signal is assumed to be made up of M fixed functions, or atoms, $\phi_m(t)$ occurring multiple times at offsets τ_i^m, where i enumerates the occurrences. Each occurrence of an atomic function is weighted by a scaling factor s_i^m. Therefore, the signal can be expressed as the sum

$$\tilde{x}(t) = \sum_{m=1}^{M} \sum_{i=1}^{I_m} s_i^m \phi_m(t - \tau_i^m) + \varepsilon(t) \ , \tag{8.7}$$

where $\varepsilon(t)$ is a noise function. This formulation is very similar to the Piano Model described in Chapter 3, except that here the scaling factor for each occurrence of an atom is kept, whereas the primitives in the fHMM are not scalable.

The system is re-formulated as a convolution of the scaling functions with the atomic functions, creating a sparse over-complete kernel basis, (see Lewicki and Sejnowski (1999) for details). It becomes quickly intractable to learn the exact parameters for such a system with a realistic amount of data, and so they propose a hybrid algorithm consisting iteratively of a matching pursuit algorithm as in (Mallat and Zhang, 1993) and then gradient descent to improve the atomic functions.

The matching pursuit algorithm finds the maximum of the convolution function of all the atoms and the residual signal, which selects a particular atom m, its offset τ_m and its scaling factor s_m. The maximum of the function

$$C(m,\tau) = \int R^n(t)\phi_m(t-\tau)dt \tag{8.8}$$

gives the scaling factor, where $R^n(t)$ is the residual signal at the n^{th} iteration. The index of the maximum value gives the atom and the time offset. The residual is simply the difference between the original signal and the reconstruction,

$$R(t) = x(t) - \tilde{x}(t) \ . \tag{8.9}$$

This scaled atom is then subtracted from the residual and the process is repeated until the error in the reconstruction reaches an acceptable value, or a predefined number of atoms have been fit.

After this process, the atoms themselves can be adapted to better model the signal, using a gradient descent process, where the objective function is

$$L = \sum_t R(t)^2 + \lambda \sum_{i,m} s_i^m \ , \tag{8.10}$$

and λ is a regularisation constant. Therefore the gradient to descend for each atom is

$$\frac{dL}{d\phi_m(t)} = \frac{2}{I}\sum_i^I R(t-\tau_i^m)\sum_i -s_i^m \ , \tag{8.11}$$

where t ranges from 1 to the length of the atom $\phi_m(t)$. The gradient multiplied by a learning rate constant is subtracted from the atom m, improving the fit of the atom to the data, and minimising the loss function L. This process is performed iteratively on all the atoms until the change in the loss function is smaller than the learning rate.

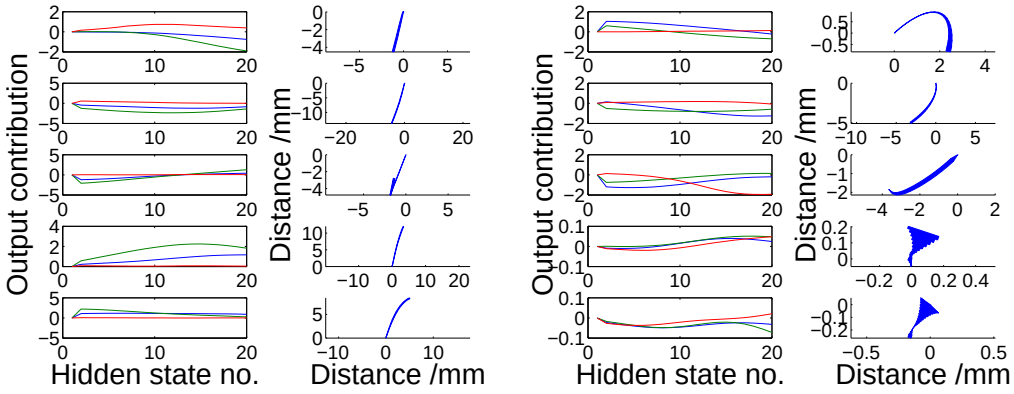


Figure 8.26: Initialisations for the atomic functions in the matching pursuit algorithm, and the primitives in the fHMM.

The main difference between the matching pursuit approach and the fHMM framework is that the atoms used as a basis for the matching pursuit are assumed to be normalised to the $\mathcal{L}_2$ norm and have an associated scaling factor with each occurrence, whereas the primitives in the fHMM are not scalable due to the computational complexity of learning such a scaling factor within a probabilistic framework.

Using a dataset consisting of the character *p*, a set of 10 primitives/atomic functions were initialised from the data, and can be seen in Figure 8.26. The final primitives using an uncoupled fHMM model with no shifting after 200 EM iterations and the final atoms after 200 matching pursuit-gradient descent iterations can be compared in Figure 8.27.

It can be seen that the matching pursuit method does not greatly alter the initialisations, so the algorithm was run for 10000 iterations to examine further changes. The results can be seen in Figure 8.28, where the shapes of the resultant functions are similar to the initialisations in Figure 8.26. To examine whether the learning rate was too small, and so the algorithm had not yet converged, the mean squared error of the reconstruction is plotted in Figure 8.29, showing that the algorithm seemed to converge after around 3000 iterations in this case.

The atomic weights allow for a greater flexibility of what the atomic functions can represent, making direct comparisons between the two models difficult. Also, the number of atoms being used is potentially limitless, and the same atom may be overlaid with itself, whereas the fHMM formulated model requires a limited number of primitives where one primitive can only be in one state at once, disallowing the possibility of overlaps of the same primitive. In this implementation, the number of

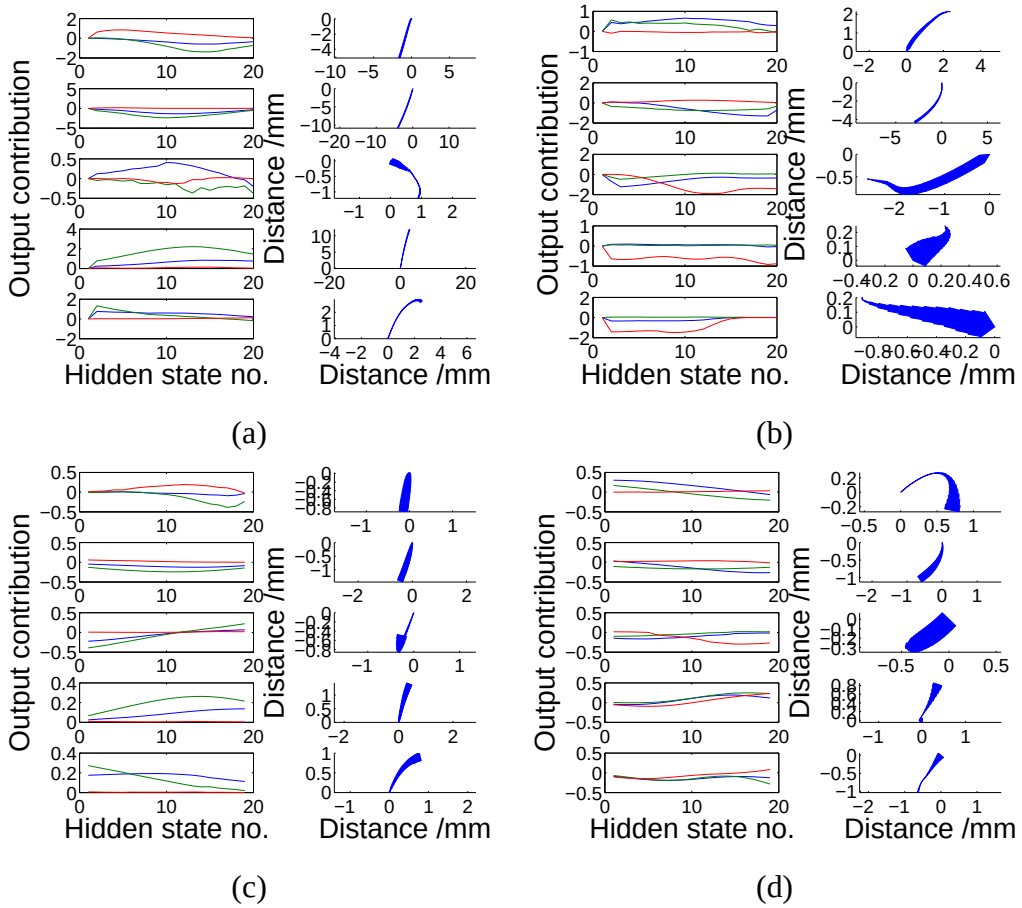


Figure 8.27: Functions derived from the fHMM model and the matching pursuit model. The functions shown in (a) and (b) are primitives from the fHMM approach after 200 EM iterations, and the functions shown in (c) and (d) are atoms from the matching pursuit approach. The atoms are normalised in $\mathcal{L}_2$ so are different sizes, however their shape is very similar to the initialisations.

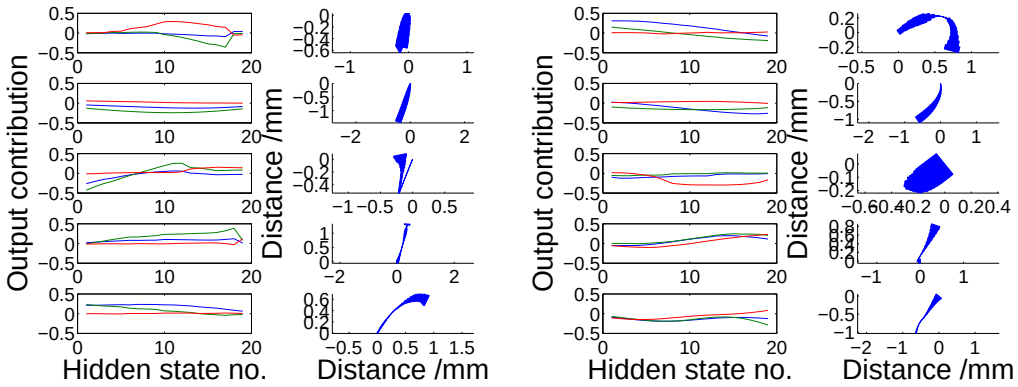


Figure 8.28: Resultant atomic functions after 10000 iterations of the matching pursuit/gradient descent algorithm. The shapes of the functions are still similar to the initialisations shown in 8.26, and can be assumed to have converged, given the plot of the mean squared error of the reconstruction in Figure 8.29.

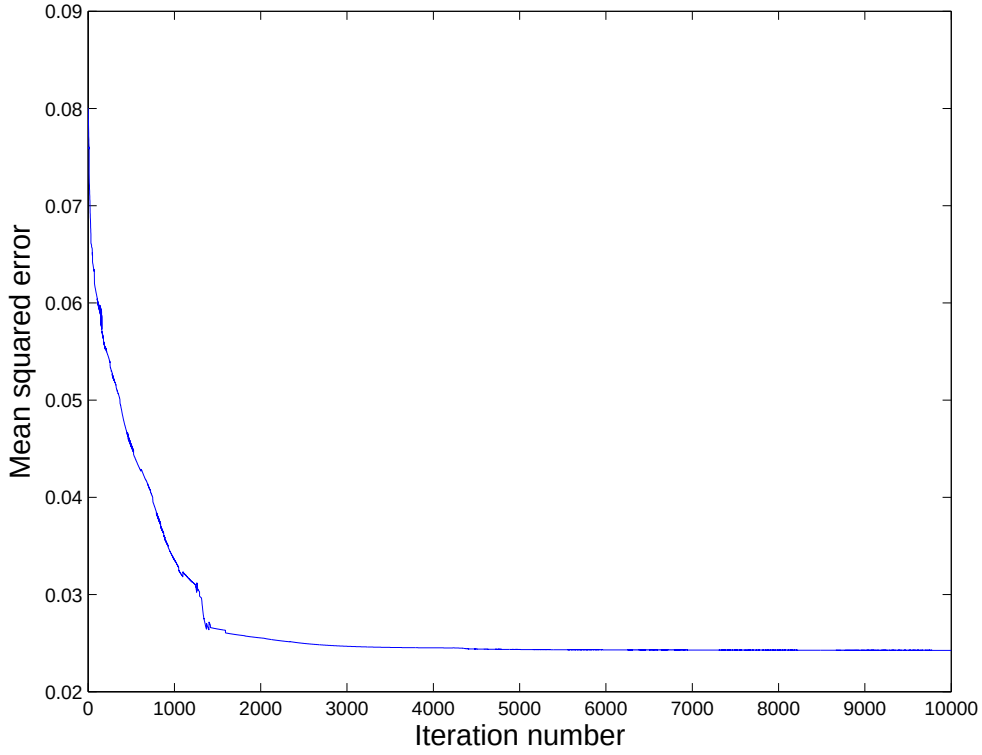


Figure 8.29: The mean squared error of the reconstruction for the matching pursuit/gradient descent method gives an indication of when the algorithm has converged, which in this case was after around 3000 iterations.

occurrences of the atoms was set to a value of 100, as this would allow on average one occurrence of each atom per sample, and is almost exactly the same as the number of spikes in the fHMM model.

Reconstructions of the dataset and a comparison of the variance profiles of the reconstructions can be seen in Figure 8.30. The matching pursuit model has a larger variance profile error, although it is of the same scale. However, it is still the case that the primitives in the matching pursuit model are scaled at each occurrence, making them naturally more flexible, and so despite the fact that they have been less adapted from their initialisations, the model is still capable of producing a convincing reproduction of the data. It is not however a generative model of a character without a higher level model of the timing and the scaling factor, and so cannot be compared in that sense.

An advantage of the fHMM model is that it can be coupled to higher level models as detailed in Chapter 7. This is accomplished by placing a prior distribution over where the primitives are likely to be active. It is unclear if an equivalent form of hierarchical coupling could be implemented using the matching pursuit algorithm. Producing generative samples from either model requires some generalisation over the timing data, although an extra degree of complexity is required for the matching pursuit data, due to the scaling factors, which are associated with each spike, effectively giving the spikes an amplitude.

8.5.3 Conclusion

There are several simple variance modelling methods that treat a character as a single datum, and can then model the variance of a dataset containing multiple characters. These models are constrained by the dimension of the datum however, meaning that the basis functions that are used to build an approximation of the dataset must be of the same dimension as the data samples, which, in this case are characters, and so the dimension varies depending upon the length of the character. Models such as these would have great difficulty operating on continuous handwriting, as the handwriting would need to be pre-partitioned into characters, and then zero-padded and any offsets removed.

In function offset models, such as the Piano model, the length of a character is unimportant, accepting computational limits on storage and processing time. The primitive functions can be learnt from individual characters and fit to unpartitioned

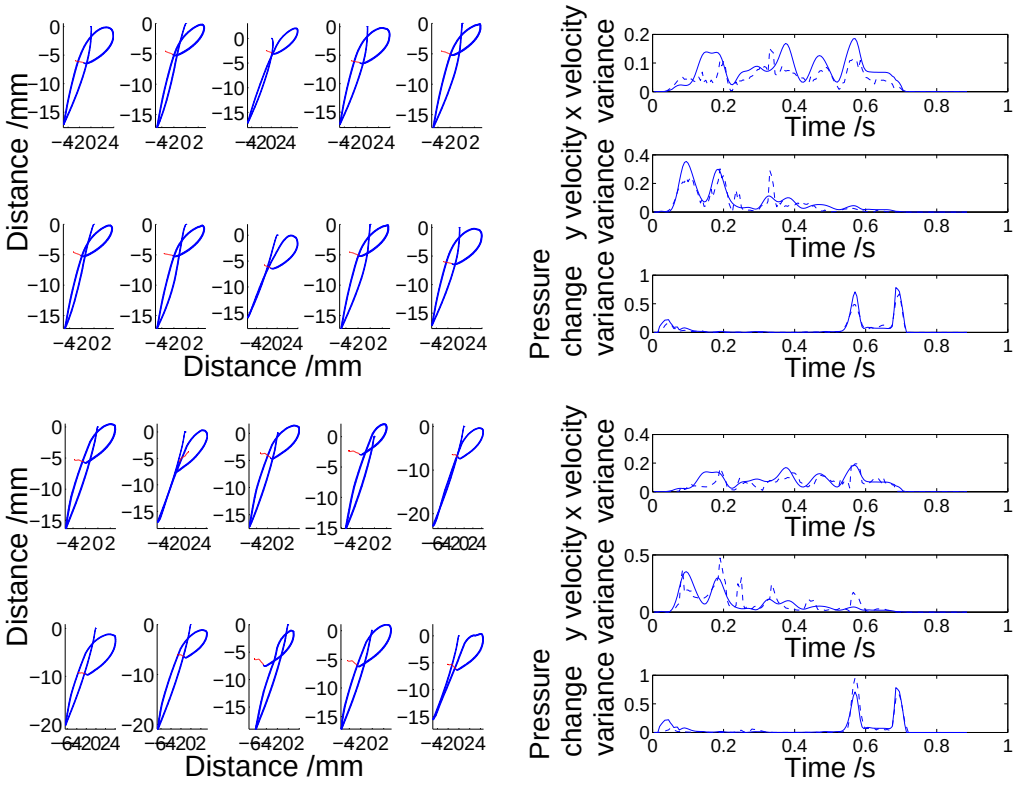


Figure 8.30: Reconstructions of a dataset containing the character *p* comparing two different models. The top row shows reconstructions and the variance profile for reconstructions using the fHMM model (with 5 primitives). The bottom row shows reconstructions and their variance profile using the matching pursuit/gradient descent model, after 10000 iterations. The total variance profile absolute error for the fHMM model was 12.02, whereas for the matching pursuit model it was 14.36. These are not generative samples, and the variance profiles are taken from the reconstructed dataset rather than a set of generated samples.

continuous segments of handwriting data, or *vice versa*. It could be argued that the timing model needs to have partitioned characters to work, and for training purposes this is indeed the case, as the timing model needs to be able to generalise across different samples of the same character rather than generic handwriting samples. Because the timing model is based upon a very sparse encoding of the characters, it is a much simpler task to infer where the start of a character would be in spike timing space. A very simple model based upon the specific combination of spikes within a short time window at the start of a character would provide a very quick potential segmentation function.

A further problem with the PCA/ICA models is that it is difficult for them to model timing drift, as it corresponds to a shift in the dimensional representation of the data when using a whole character as a datum. This type of timing jitter error is well modelled in offset function models, and in Markov models, where it is possible for a 'delay' to be modelled by self transitions, which is known as dynamic time warping.

The matching pursuit model of Lewicki *et al* is very good when there are a large number of potential atoms, such as a whole family of gamma-tone kernel functions. Using a greedy cut-off process, a subset of these functions can then be selected and adapted to better fit the dataset. The code produced is known as a spikegram, and contains not just spike positions, but also spike amplitudes. This code is more complex than the spike timing code of the fHMM, and can potentially have many more atoms/primitives. This makes it difficult for a higher level model to generalise across different character sample codes. For the case of an fHMM, it is possible to couple the timing model and the primitive model together, providing a more generalised representation. It is not clear how this would be implemented for the matching pursuit case, as there is no probabilistic inference involved in the algorithm. However, the matching pursuit algorithm is faster than EM inference performed on an fHMM. One possible use of this could be to use the matching pursuit/gradient descent algorithm to initialise the primitive shapes for the fHMM. The results of the matching pursuit algorithm shown in Figure 8.28 suggest that the resultant atoms depend heavily on their initialisations, or in the case of a whole set of functions, they would depend upon the function class chosen.

Chapter 9

Discussion

This project has explored the possibility of modelling handwriting using a fixed function offset model (Piano Model). The aim of the project was to learn a generative model for handwriting, based upon the concept of motor primitives. In the space of pen movements, primitives were approximated as fixed functions in pen tip velocity space.

Appropriate functions for such a model can be learnt within a formalised probabilistic framework, using a constrained Factorial Hidden Markov Model. This framework allows the functions to be variable lengths, and does not require traditional pre-partitioning of the characters into points of highest curvature or any feature extraction, which is often used in other handwriting models.

A generative model of characters based upon the primitive's timing within a character was also presented. Several alternative models were investigated, although the inconsistent nature of the spike timing data restricts the class of Gaussian models to be one-dimensional.

Coupling of the primitive model together with the timing model during character learning allows better representations of the data to be learnt, as the timing model provides an informed onset prior distribution for the primitive model.

9.1 Implications for biology

The existence of motor primitives as sub-blocks of movement is a widely accepted theory, partially because the definition of what exactly a primitive might be is flexible. From a biological point of view, it is an attractive idea that primitives could resemble an output basis set for controlling muscles, where the selection and timing of the prim-

itives is the encoded movement. This code would therefore naturally be representable by a distributed set of spike timings, such as is likely to be the case in the brain.

One implication for biology, which is in agreement with several movement studies, focussing on rhythm representation an movement, is that the control of biological movement may be segmented into a primitive part and a timing part. (Penhume et al., 1998; Meegan et al., 2000; Dennis et al., 2004) all explore the importance of rhythm in movement planning and generation, and the role of the cerebellum in motor timing tasks.

Understanding how people might be learning to control a pen, and the internal representations used could have useful educational implications. More efficient teaching methods, perhaps focussing on rhythmic aspects of writing, with the purpose of reinforcing the best possible timing code could be developed.

A related field to the teaching of handwriting is relearning to write, after a stroke, or physical damage to the hand or arm. With a better understanding of how the brain might have originally encoded the movements, the relearning process could be optimised, making it easier and faster for the subject.

9.2 Implications for robotics

Instead of defining the observable data as pen tip velocities, it could be defined as some parameterisation of a control system. These parameterisations could change in a fixed manner over the course of the primitive, allowing complex movements to be built up by combining primitives. Such primitives could be learnable using EM as described in Chapter 4, along with a deterministic model of the control system. Simple one-dimensional timing variation could then represent the complex variation in the movement trajectory of the system being controlled. A more specific formulation would be to model the joint torques of some robot model as the observable variables, and learn primitives based upon this data.

Robotic control is traditionally a feedback control loop problem, where some error signal is fed back and along with a control or modulating signal is used to minimise the predicted error. As the system being controlled becomes more complex, and as uncertainty is introduced, this type of control becomes very difficult to plan and optimise. More recent research into using motor primitives as a basis for control has generally focussed upon segmenting movements, and using some component extraction technique to provide a basis set of movements. These techniques normally do not assume that

the components are time extended, or that they can overlap in an unrestricted fashion (either entirely or partially) with each other. An extension of the model presented here could be used as a framework to model and learn a novel type of motor primitive that could be used to control a physical control system. The combinatorial variety of the output possibilities is higher than in other models, as the same combination of primitives can be triggered with different offsets, however the internal representation is very compact, being a set of spike times.

It remains to be seen if such a primitive-timing representation could be useful for adaptive control problems, however there are suggestions that biological, or at least human control may rely on timing specific events. For instance research by David Lee into tau theory of movement has shown how biological movements appear to be guided by coupling time intervals observed in the dynamic world. (see (Lee et al., 2001), or (Lee, 2006) for a review of this research). Tau theory suggests that the brain encodes the dynamic world in terms of timings of events, and uses ratios of these timings to adapt movements. The primitive-timing representation of movements would fit in well with such a theory of movement control and planning.

9.3 Implications for data representation

Compression of data is an extremely important subject in the modern world, where information and media are continually being stored, categorised, and transmitted over the internet. Compression of data is also important for the brain, and so if the handwriting movements are produced using a similar model to the one described in this project, it is likely that they are stored in the highly compressed form of spike times. Such a code would therefore capture the recognisable variation in the handwriting, and thus be the most efficient way of storing and transmitting the handwriting.

Many devices which use handwriting recognition, such as personal hand-held devices, have a relatively low bandwidth connection, meaning the information transmitted is normally in the form of text, rather than handwriting. However it may sometimes be more convenient or desirable to transmit the handwriting, or some illustrative sketches. A primitive-timing representation of these pen movements would provide a good compression for low bandwidth transmission. An interesting aspect of this representation is the possibility of the sketch and writing being transformed into the receiver's handwriting upon receipt.

Handwriting is gradually becoming more of an art form, as people use keyboards

more and more, although this trend may not continue now that pen tablets in the form of a small hand-held PC or personal organiser are becoming popular. Handwriting is considered a very personal skill, with many 'personality guessing tests' based upon handwriting. This means that it would be a good mode of interaction with many computing devices. If the text on a mobile phone was a replication of the owner's handwriting for instance, a greater bond, or personalisation could be achieved.

9.4 Limitations/Extensions of the model

A limitation of the model is the lack of feedback, both in terms of the behaviour of the primitives themselves, and any positional feedback to higher levels of the model. This could potentially improve the performance, and make the primitives more generalised as simple parameterised systems. A simplified model of a hand as a spring system was proposed by (Hinton and Nair, 2005). Such a system would provide low level feedback, and the primitive model would parameterise the stiffness of the springs over time.

The primitives in a piano model formulation generally have a scaling factor associated with them, however in the fHMM framework, they do not. This is because the extra parameters would make the model extremely difficult to learn, and the structured variational approximation used in this project to improve the speed of the EM inference would no longer work. An alternative form of posterior estimation such as Gibbs sampling could be used on a more complex model, however the larger parameter space would bring with it more local maxima in the likelihood function, making inference difficult.

A problem for any possible timing model is that the primitives are allowed to occur a variable number of times in a particular character sample. This means that the timing model needs a latent state modelling primitive presence, and makes it difficult to capture the covariance between the spike times. Forcing all primitives to occur exactly once in a character sample would be possible by imposing very strong priors at the start and the end of a character. This would create a sort of 'Serialist Piano Model'[1] where primitives are not as flexible as in the model proposed here. Each character would have to contain every primitive. This could work for a single character class, but for multiple character types different primitives would have to be switched on or off, po-

[1] Serialism is a 20^{th} century compositional technique whereby every semitone in a scale must be played exactly once before repetitions are allowed.

tentially achieved by using a single probabilistic binary onset parameter dependent upon the character class. With this constrained primitive model a timing model would be a simple case of placing a single multivariate Gaussian distribution or a mixture of Gaussians over all the spike times for a particular character. Preliminary experiments however showed that such a model still produced multiple clusters of spikes, with some character samples using a primitive from cluster 1, and some using the same primitive from cluster 2 for instance. The extra constraints placed on the model simply mean that more primitives are required to produce an acceptable reconstruction. Such a distribution could be modelled using a mixture of Gaussians, and a higher level model coupling the components of the various primitives together, however this is really the same situation as with the current model, but with more primitives occurring fewer times.

The low level primitive model works well to produce a reconstruction of the dataset, assuming an adequate number of primitives. However the generative performance of the timing models are not very good when large numbers of primitives are needed. By fixing the hidden state trajectory of the timing model, as was discussed in Section 6.2.4.2, it can be seen that there should be more dependencies between the primitives. As the primitives are encoded as spikes, a probabilistic spiking neural network would be an interesting model to try. Connectivity in the network would determine dependency between primitives, and multiple synaptic time delays could model the timing of the spikes. Alternatively, the current mode data could be used to extend the model to another level, by modelling this data.

A very successful and widely used method of component extract is ICA. The problem with applying this approach to primitives is that the primitives must be ‘keyed’, in that their starting point must be known in each sample. The fHMM model acts in a similar manner to ICA, trying to extract the most statistically independent components or factors. The benefit of the constrained fHMM presented in this project is that the factors have no pre-defined start points. The similarities with ICA would be interesting to explore further. Using longer primitives that must occur once in a sample, and that have an associated scaling factor would perhaps provide similar independent components to ICA. However the components derived by the fHMM would be less constrained, and self-transitions could easily be introduced. A quick exploration of this approach was implemented by using fixed length primitives of 150 states, (the equivalent of $0.75s$) which is of the same order of length as the character samples. 5 long primitives were initialised randomly and trained upon a dataset containing the character *p*, and can be

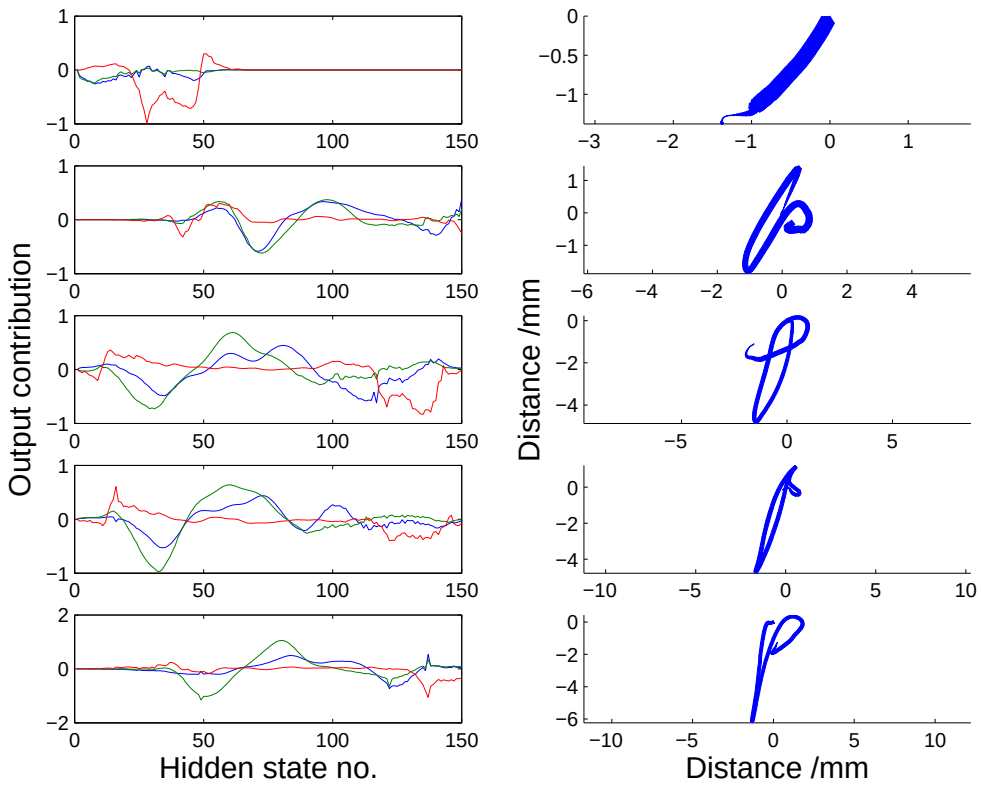


Figure 9.1: 5 primitives trained on a dataset containing the character *p*. The length of the primitives was fixed at 150 states, making them similar to an ICA-like decomposition of the character, rather than modelling sections of the character. They could be compared for instance to the PCA components seen earlier in Figure 8.21

seen in Figure 9.1. Generative samples and the timing distribution of the primitives in the dataset can be seen in Figure 9.2. These long primitives are not sections of the character, rather an ICA-like decomposition of the character, but with varying offsets.

The EM inference of a large model such as the fHMM is quite slow, and the speed and successful recovery depends upon appropriate parameter initialisations. The initialisation used in this project were taken from mean velocities taken over the character samples in the dataset. An alternative method would be to use a fast greedy algorithm to extract initial primitives, such as the matching pursuit algorithm (Lewicki and Sejnowski, 1999), as discussed in Section 8.5.2.3, or a pattern matching algorithm such as that presented in (Chiu et al., 2003), or ICA.

The model presented in this project is not limited to handwriting or pen movements. An obvious field to explore with this model would be that of music. A good model of the harmonic interaction of the notes when played on a piano, and how the force at which a note is played affects its waveform would greatly improve this model's po-

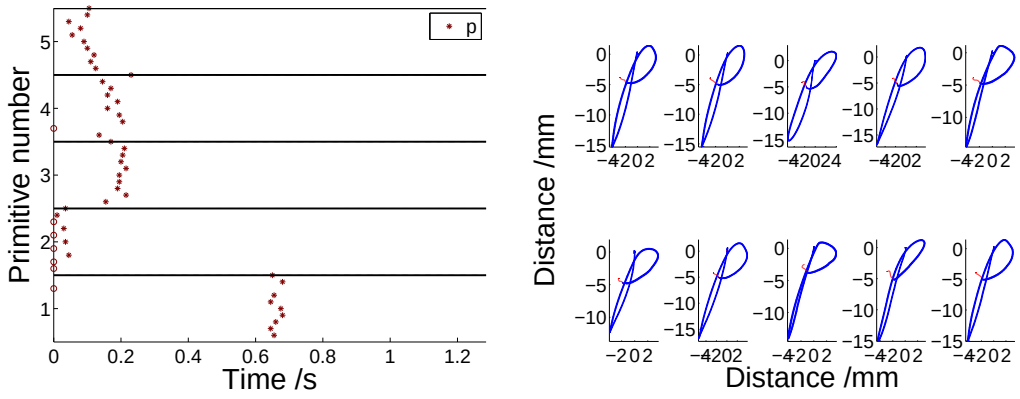


Figure 9.2: The timing of the long primitives seen in Figure 9.1 can be seen in the leftmost plot, showing that they are mostly all active for every character sample, and at most a single time. The rightmost plot shows 10 generative samples, which are expectably good, due to the simplicity of the timing distribution.

tential to represent and reproduce piano music, and potentially provide a very efficient and convincing *performance engine* for reproduction of simple MIDI scores.

9.4.1 Psychophysical experiments

Validation of the primitives and their presence in the brain could be attempted by performing some perceptual experiments. Each task would require the subject to provide handwriting samples upon which the model would be trained, and then subject specific tasks using 'their' primitives could be performed. Several psychophysical experiments would be interesting to perform in conjunction with the handwriting model presented here. Such experiments would explore the permanence of the primitive functions, and whether they are also exploited for recognition purposes.

Learning to write

The subject would be asked to learn to reproduce a novel character. Either the character could be built up using primitives extracted from their writing, or entirely different primitives. The assumption being that the task in the later case would be more difficult for the subject. A further experiment would be to then train a new model on the novel character samples, and compare the primitives extracted with the ones used to generate the character originally. The assumption being that the differences between the *planted motifs* and the extracted primitives would shed light on the limitations of the types of primitive functions that are learnable.

Finally, depending upon the rate at which a subject learns to write a new character, an analysis of the primitives present over time would be very interesting, as it is unclear when, how and if primitives might be adapted to a specific task.

Learning to read

A subject would be required to learn two or more novel characters, and then distinguish between them. One set of characters would be constructed using primitives extracted from their handwriting, whilst another set would be constructed using novel primitives. The assumption being that if primitives are exploited in some way for recognition, then the set with their own primitives would be easier to learn.

Learning to remember

A subject would be presented with a single, novel character, and then asked to recall it by using a differentiation task after a wait period. Similar to the above experiments, the character would either be made up using the subject's primitives, or novel ones.

A variation on this experiment could be to introduce noise into the character between the presentations, either as primitive shape noise, or primitive timing noise. The hypothesis here is that the timing noise would be more noticeable, as small variations in the primitive shapes might not be 'seen' by a feature detector trained upon a particular primitive.

9.5 Summary

Despite having superior motors and materials, with far less noise present in actuators and sensors, robots are still much worse at moving than biological organisms. This must be therefore due to a superior internal representation of the movements being made. This thesis has attempted to shed light on one such possible representation based upon the concept of primitives.

Primitives allow small sections of movement to be assembled to create complete task based movements. The manner in which these primitives are assembled, and the adaptability of the primitives is unknown. This thesis assumes that the brain operates in a massively parallel and asynchronous manner, meaning that the primitives are likely to be triggered at different times throughout a movement, and that the timing of these events would contain noise.

This thesis attempts to explain handwriting variation in terms of noise within the generating model, or timing jitter of primitive activation. Other sources of variation such as primitive substitution are partially modelled, and an interesting extension would be to examine how much unexplained variation could be accounted for by dynamical models of the hand.

Despite learning to handwrite in classes with a common teacher, and identical teaching materials and methods, personal handwriting develops much variation, both from a single subject, and style variation between subjects. This thesis attempts to explain the first type of variation in terms of primitive timing noise, and the larger second type of variation in terms of differences in primitive shapes between subjects.

The timing of primitives can conveniently be expressed as a spike timing pattern, which is a very compact code for a character. It holds obvious similarities to the spiking behaviour of neuronal activity, and suggests a segmentation between a timing model and a primitive model within the brain. Interesting lesion studies could be conducted by carrying out simulated lesions of the model to validate the type of primitives used here. Also this model allows several psycho-physical experiments to be conducted to further examine the nature of primitives and the operation of motor planning within the brain.

In conclusion, this thesis provides a novel manner of inferring primitives from time series movement data, in this case handwriting trajectories. This model also attempts to generalise across multiple samples to produce a fully generative model of the movement, where the complex variation from one sample to the next is captured in the higher level timing model as simple Gaussian noise. It is the hope of this author that such a model will prove useful for new types of robotic movement planning, and for efficient handwriting representation and reproduction.

Chapter 10

Appendix

10.1 Variational Log Likelihood

The observable data likelihood is defined as,

$$\begin{aligned}\log P(\{\mathbf{Y}_t\}) &= \log \sum_{\{\mathbf{S}_t^m\}} P(\{\mathbf{S}_t^m, \mathbf{Y}_t\}) \\ &= \log \sum_{\{\mathbf{S}_t^m\}} Q(\{\mathbf{S}_t^m\}) \left[\frac{P(\{\mathbf{S}_t^m, \mathbf{Y}_t\})}{Q(\{\mathbf{S}_t^m\})}\right] \\ &\geq \sum_{\{\mathbf{S}_t^m\}} Q(\{\mathbf{S}_t^m\}) \log \left[\frac{P(\{\mathbf{S}_t^m, \mathbf{Y}_t\})}{Q(\{\mathbf{S}_t^m\})}\right] \end{aligned} \tag{10.1}$$

The inequality defines a lower bound on the likelihood, called the variational log likelihood (VLL), which is equivalent to the negative variational free energy.

$$VLL = \sum_{\{\mathbf{S}_t^m\}} Q(\{\mathbf{S}_t^m\}) \log P(\{\mathbf{S}_t^m, \mathbf{Y}_t\}) - \sum_{\{\mathbf{S}_t^m\}} Q(\{\mathbf{S}_t^m\}) \log Q(\{\mathbf{S}_t^m\}) \, . \tag{10.2}$$

The various terms of this equation are derived here. The joint probability of the hidden and observable states of the fHMM can be expanded,

$$\begin{aligned}\log P(\{\mathbf{S}_t^m, \mathbf{Y}_t\}) &= \sum_m \log P(\mathbf{S}_1^m) + \sum_m \sum_{t=2}^{T} \log P(\mathbf{S}_t^m \,|\, \mathbf{S}_{t-1}^m) + \sum_{t=1}^{T} \log P(\mathbf{Y}_t \,|\, \mathbf{S}_t^1, .., \mathbf{S}_t^M) \\ &= \sum_m \log \boldsymbol{\pi}_{\mathbf{S}_1^m}^m + \sum_m \sum_{t=2}^{T} \log \mathbf{P}_{\mathbf{S}_t^m, \mathbf{S}_{t-1}^m}^m + \sum_{t=1}^{T} \log P(\mathbf{Y}_t \,|\, \mathbf{S}_t^1, .., \mathbf{S}_t^M), \end{aligned} \tag{10.3}$$

where

$$\begin{aligned}\log P(\mathbf{Y}_t \mid \mathbf{S}_t) &= -\frac{1}{2}\log(|\mathbf{C}|) - \frac{D}{2}\log(2\pi) - \frac{1}{2}\left(\mathbf{Y}_t - \sum_m \mathbf{W}^m_{\mathbf{S}^m_t}\right)\mathbf{C}^{-1}\left(\mathbf{Y}_t - \sum_n \mathbf{W}^n_{\mathbf{S}^n_t}\right) \\ &= -\frac{1}{2}\log(|\mathbf{C}|) - \frac{D}{2}\log(2\pi) - \frac{1}{2}\mathbf{Y}_t\mathbf{C}^{-1}\mathbf{Y}_t + \sum_m \mathbf{Y}_t\mathbf{C}^{-1}\mathbf{W}^m_{\mathbf{S}^m_t} - \frac{1}{2}\sum_{m,n}\mathbf{W}^m_{\mathbf{S}^m_t}\mathbf{C}^{-1}\mathbf{W}^n_{\mathbf{S}^n_t}\,. \end{aligned} \tag{10.4}$$

Taking the expectation of the joint distribution w.r.t. the variational distribution $Q(\{\mathbf{S}_t\})$,

$$\begin{aligned}&\sum_{\{\mathbf{S}^m_t\}} Q(\{\mathbf{S}^m_t\})\log P(\{\mathbf{S}^m_t, \mathbf{Y}_t\}) \\ &= \sum_m\sum_k Q(\mathbf{S}^m_t = k)\log \boldsymbol{\pi}^m_k \\ &+ \sum_m\sum_{t=2}^{T}\sum_{k,j} Q(\mathbf{S}^m_t = k, \mathbf{S}^m_{t-1} = j)\log \mathbf{P}^m_{k,j} \\ &+ \sum_{t=1}^{T} Q(\{\mathbf{S}^m_t\})\log P(\mathbf{Y}_t \mid \mathbf{S}^1_t, .., \mathbf{S}^M_t) \\ &= \sum_m\sum_k Q(\mathbf{S}^m_t = k)\log \boldsymbol{\pi}^m_k \qquad (10.5) \\ &+ \sum_m\sum_{t=2}^{T}\sum_{k,j} Q(\mathbf{S}^m_t = k, \mathbf{S}^m_{t-1} = j)\log \mathbf{P}^m_{k,j} \\ &- \frac{T}{2}\log(|\mathbf{C}|) - \frac{TD}{2}\log(2\pi) - \frac{1}{2}\sum_{t=1}^{T}\mathbf{Y}_t\mathbf{C}^{-1}\mathbf{Y}_t \\ &+ \sum_{t=1}^{T}\sum_{m,k} Q(\mathbf{S}^m_t = k)\mathbf{Y}_t\mathbf{C}^{-1}\mathbf{W}^m_k \\ &- \frac{1}{2}\sum_{t=1}^{T}\sum_{m,n\neq m}\sum_{k=1}^{K^m}\sum_{j=1}^{K_n} Q(\mathbf{S}^m_t = k)Q(\mathbf{S}^n_t = j)\mathbf{W}^m_k\mathbf{C}^{-1}\mathbf{W}^n_j \\ &- \frac{1}{2}\sum_{t=1}^{T}\sum_m\sum_{k=1}^{K^m} Q(\mathbf{S}^m_t = k)\mathbf{W}^m_k\mathbf{C}^{-1}\mathbf{W}^m_k\,. \qquad (10.6)\end{aligned}$$

The entropy of the $Q(\{\mathbf{S}_t\})$ is defined as,

$$\begin{aligned}
&\sum_{\{\mathbf{S}_t^m\}} Q(\{\mathbf{S}_t^m\}) \log Q(\{\mathbf{S}_t^m\}) \\
&= \sum_m \sum_k Q(\mathbf{S}_1^m = k) \log Q(\mathbf{S}_1^m = k) \\
&+ \sum_m \sum_{t=2}^{T} \sum_{k,j} Q(\mathbf{S}_t^m = k, \mathbf{S}_{t-1}^m = j) \log Q(\mathbf{S}_t^m = k | \mathbf{S}_{t-1}^m = j) \\
&= \sum_m \sum_k Q(\mathbf{S}_1^m = k) \log Q(\mathbf{S}_1^m = k) \\
&+ \sum_m \sum_{t=2}^{T} \sum_{k,j} Q(\mathbf{S}_t^m = k, \mathbf{S}_{t-1}^m = j) \log \left(\frac{Q(\mathbf{S}_t^m = k, \mathbf{S}_{t-1}^m = j)}{Q(\mathbf{S}_{t-1}^m = j)} \right) . \qquad (10.7)
\end{aligned}$$

The E-step calculates the $Q(\{\mathbf{S}_t\})$ distribution, and summarises it in the $\gamma_{t,k}^m$ and $\xi_{t,k,j}^m$ marginal distributions, where

$$\gamma_{t,k}^m \equiv Q(\mathbf{S}_t^m = k), \qquad (10.8)$$

and

$$\xi_{t,k,j}^m \equiv Q(\mathbf{S}_t^m = k, \mathbf{S}_{t-1}^m = j) . \qquad (10.9)$$

There is also a distribution required, $Q(\mathbf{S}_t^m, \mathbf{S}_t^n)$, but due to the independence assumption of the Markov chains, this can be expressed using the $Q(\mathbf{S}_t^m)$ distribution,

$$\begin{aligned}
&Q(\mathbf{S}_t^m = k, \mathbf{S}_t^n = j) = Q(\mathbf{S}_t^m = k) Q(\mathbf{S}_t^n = j) \ \ \forall m \neq n, \\
&Q(\mathbf{S}_t^m = k, \mathbf{S}_t^n = j) = Q(\mathbf{S}_t^m = k) \ \ \forall m = n, k = j \\
&Q(\mathbf{S}_t^m = k, \mathbf{S}_t^n = j) = 0 \ \ \text{otherwise} \qquad (10.10)
\end{aligned}$$

Therefore, in summary,

$$
\begin{aligned}
VLL = & \sum_m \sum_k \gamma^m_{t=1,k} \log \boldsymbol{\pi}^m_k \\
& + \sum_m \sum_{t=2}^{T} \sum_{k,j} \xi^m_{t,k,j} \log \mathbf{P}^m_{k,j} \\
& - \frac{T}{2} \log(|\mathbf{C}|) - \frac{TD}{2} \log(2\pi) - \frac{1}{2} \sum_{t=1}^{T} \mathbf{Y}_t \mathbf{C}^{-1} \mathbf{Y}_t \\
& + \sum_{t=1}^{T} \sum_{m,k} \gamma^m_{t,k} \mathbf{Y}_t \mathbf{C}^{-1} \mathbf{W}^m_k \\
& - \frac{1}{2} \sum_{t=1}^{T} \sum_{m,n \neq m} \sum_{k=1}^{K^m} \sum_{j=1}^{K_n} \gamma^m_{t,k} \gamma^n_{t,j} \mathbf{W}^m_k \mathbf{C}^{-1} \mathbf{W}^n_j \\
& - \frac{1}{2} \sum_{t=1}^{T} \sum_m \sum_{k=1}^{K^m} \gamma^m_{t,k} \mathbf{W}^m_k \mathbf{C}^{-1} \mathbf{W}^m_k \\
& - \sum_m \sum_k \gamma^m_{t=1,k} \log \gamma^m_{t=1,k} \\
& - \sum_m \sum_{t=2}^{T} \sum_{k,j} \xi^m_{t,k,j} \log \left(\frac{\xi^m_{t,k,j}}{\gamma^m_{t-1,j}} \right)
\end{aligned}
\tag{10.11}
$$

10.1.1 Regularised VLL

The update for the $\mathbf{W}$ parameters (see Section 4.3.3 contains a pseudo-inverse function, which becomes numerically unstable with near singular matrices. To help with this instability, a regularisation constant, λI is added before the inversion in the M-step,

$$
\mathbf{W} \leftarrow \left(\sum_{t=1}^{T} \mathbf{Y}_t Q(\mathbf{S}_t)' \right) \left(\sum_{t=1}^{T} Q(\mathbf{S}_t) Q(\mathbf{S}_t)' + \lambda I \right)^{\dagger} . \tag{10.12}
$$

This is equivalent to putting a prior on $\mathbf{W}$,

$$
P(\mathbf{W}) = \left(\frac{\lambda}{2\pi |\mathbf{C}|} \right)^{\frac{-KMD}{2}} \exp \left(\frac{-\lambda}{2} \sum_{k,m} \mathbf{W}'^m_k \mathbf{C}^{-1} \mathbf{W}^m_k \right) . \tag{10.13}
$$

This is not dependent upon t, and is included into the likelihood calculation only once.

$$
\begin{aligned}
VLL_{reg} = & \sum_m \sum_k \gamma^m_{t=1,k} \log \boldsymbol{\pi}^m_k \\
& + \sum_m \sum_{t=2}^{T} \sum_{k,j} \xi^m_{t,k,j} \log \mathbf{P}^m_{k,j} \\
& - \frac{T}{2} \log(|\mathbf{C}|) - \frac{TD}{2} \log(2\pi) - \frac{1}{2} \sum_{t=1}^{T} \mathbf{Y}_t \mathbf{C}^{-1} \mathbf{Y}_t \\
& + \sum_{t=1}^{T} \sum_{m,k} \gamma^m_{t,k} \mathbf{Y}_t \mathbf{C}^{-1} \mathbf{W}^m_k \\
& - \frac{1}{2} \sum_{t=1}^{T} \sum_{m,n \neq m} \sum_{k=1}^{K^m} \sum_{j=1}^{K_n} \gamma^m_{t,k} \gamma^n_{t,j} \mathbf{W}^m_k \mathbf{C}^{-1} \mathbf{W}^n_j \\
& - \frac{1}{2} \sum_{t=1}^{T} \sum_{m} \sum_{k=1}^{K^m} \gamma^m_{t,k} \mathbf{W}^m_k \mathbf{C}^{-1} \mathbf{W}^m_k \\
& - \sum_m \sum_k \gamma^m_{t=1,k} \log \gamma^m_{t=1,k} \\
& - \sum_m \sum_{t=2}^{T} \sum_{k,j} \xi^m_{t,k,j} \log \left(\frac{\xi^m_{t,k,j}}{\gamma^m_{t-1,j}} \right) \\
& + \left(\frac{KMD}{2} \right) \log \lambda - \left(\frac{KMD}{2} \right) \log(2\pi|\mathbf{C}|) - \frac{\lambda}{2} \sum_{k,m} \mathbf{W}'^m_k \mathbf{C}^{-1} \mathbf{W}^m_k \; . \qquad (10.14)
\end{aligned}
$$

To show that this prior will produce the regularised update, we take the partial differential of the log likelihood w.r.t. $\mathbf{W}$, and set to 0,

$$
\begin{aligned}
\frac{\partial vll}{\partial \mathbf{W}^b_l} = & \sum_{t=1}^{T} Q(\mathbf{S}^b_t = l) \mathbf{Y}'_t \mathbf{C}^{-1} - \frac{1}{2} \sum_t \sum_{n \neq b} \sum_{j \neq l} Q(\mathbf{S}^b_t = l, \mathbf{S}^n_t = j) \mathbf{W}^n_j \mathbf{C}^{-1} \\
& - \frac{1}{2} \sum_t \sum_{m \neq b} \sum_{k \neq l} Q(\mathbf{S}^m_t = k, \mathbf{S}^b_t = l) \mathbf{W}^m_k \mathbf{C}^{-1} - \sum_t Q(\mathbf{S}^b_t = l, \mathbf{S}^b_t = l) \mathbf{W}^b_l \mathbf{C}^{-1} - \lambda \mathbf{W}^b_l \mathbf{C}^{-1} \\
= & \sum_t Q(\mathbf{S}^b_t = l) \mathbf{Y}_t - \sum_t \sum_{m \neq b} \sum_{k \neq l} Q(\mathbf{S}^m_t = k, \mathbf{S}^b_t = l) \mathbf{W}^m_k - \sum_t Q(\mathbf{S}^b_t = l, \mathbf{S}^b_t = l) \mathbf{W}^b_l - \lambda \mathbf{W}^b_l \\
= & \sum_t Q(\mathbf{S}^b_t = l) \mathbf{Y}_t - \sum_t \sum_{m,k} Q(\mathbf{S}^m_t = k, \mathbf{S}^b_t = l) \mathbf{W}^m_k - \lambda \mathbf{W}^b_l \; . \qquad (10.15)
\end{aligned}
$$

In matrix form,

$$
\begin{aligned}
\frac{\partial vll}{\partial \mathbf{W}} &= \sum_t \mathbf{Y}_t Q(\mathbf{S}_t)' - \sum_t \mathbf{W} Q(\mathbf{S}_t, \mathbf{S}_t) - \mathbf{W} \lambda I = 0 \\
\mathbf{W} \left(\sum_t Q(\mathbf{S}_t, \mathbf{S}_t) + \lambda I \right) &= \sum_t \mathbf{Y}_t Q(\mathbf{S}_t)' \\
\mathbf{W} &= \sum_t \mathbf{Y}_t Q(\mathbf{S}_t)' \left(\sum_t Q(\mathbf{S}_t, \mathbf{S}_t) + \lambda I \right)^{\dagger} \qquad (10.16)
\end{aligned}
$$

Bibliography

R. Amit and M. Mataric. Parametric primitives for motor representation and control. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 863–868, 2002.

L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *Ann. Math. Statist.*, 40(1):164–171, 1970.

H. S. M. Beigi. An overview of handwriting recognition. In *1st Annual conference on technological advancements in developing countries*, pages 30–46, Columbia University, New York, July 1993.

E. Bizzi, S.F. Giszter, E. Loeb, F.A. Mussa-Ivaldi, and P. Saltiel. Modular organization of motor behavior in the frog's spinal cord. *Trends in Neurosciences*, 18(10):442–446, 1995.

E. Bizzi, A. d'Avella, P. Saltiel, and M. Trensch. Modular organization of spinal motor systems. *The Neuroscientist*, 8(5):437–442, 2002.

G. Caithness, R. Osu, P. Bays, H. Chase, J. Klassen, M. Kawato, D. M. Wolpert, and J. R. Flanagan. Failure to consolidate the consolidation theory of learning for sensorimotor adaptation tasks. *Journal of Neuroscience*, 24(40):8662–8671, 2004.

A. Cemgil, B. Kappen, and D. Barber. A generative model for music transcription. In *IEEE Transactions on Speech and Audio Processing*, volume 13, 2005.

B. Chiu, E. Keogh, and S. Lonardi. Probabilistic discovery of time series motifs, 2003.

P.Perona D. del Vecchio, R.M. Murray. Decomposition of human motion into dynamics-based primitives with application to drawing tasks. *Automatica*, 39:2085–2098, 2003.

A. d'Avella and E. Bizzi. Shared and specific muscle synergies in natural motor behaviors. In *National Academy of Sciences*, volume 102, pages 3076–3081, 2005.

A. d'Avella and E. Bizzi. Low dimensionality of supraspinally induced force fields. *Proc. Natl. Acad. Sci. USA*, 96:7711–7714, 1998.

A. d'Avella, P. Saltiel, and E. Bizzi. Combinations of muscle synergies in the construction of a natural motor behavior. *Nature Neuroscience*, 6(3):300–308, 2003.

P. R. Davidson and D. M. Wolpert. Motor learning and prediction in a variable environment. *Curr. Opinion in Neurobiology*, 13:1–6, 2003.

P. R. Davidson and D. M. Wolpert. Scaling down motor memories: de-adaptation after motor learning. *Neuroscience Letters*, 370:102–107, 2004.

A. Rizzi D.C. Conner, H. Choset. Integrated planning and control for convex-bodied nonholonomic systems using local feedback. In *Robotics: Science and Systems II, MIT Press*, pages 57–64, 2006.

M. Dennis, K. Edelstein, R. Hetherington, K. Copeland, J. Frederick, S. E. Blaser, L.A. Kramer, J.M. Drake, M. Brandt, and J. M. Fletcher. Neurobiology of perceptual and motor timing in children with spina bifida in relation to cerebellar volume. *Brain*, 2004.

E. Drumwright, O. C. Jenkins, and M. J. Matarić. Exemplar-based primitives for humanoid movement classification and control. In *In Proc. of Intl. Conf. on Robotics and Automation*, pages 140–145, 2004.

A. Fod, M.J. Mataric, and O.C. Jenkins. Automated derivation of primitives for movement classification. *Autonomous robots*, 12(1):39–54, 2002.

V. Gallese, L. Fadiga, L. Fogassi, and G. Rizzolatti. Action recognition in the premotor cortex. *Brain*, 119:593–609, 1996.

Z. Ghahramani and M.I. Jordan. Factorial hidden Markov models. *Machine Learning*, 29:245–275, 1997.

A. Hamilton, K. Jones K, and D. M. Wolpert. The scaling of motor noise with muscle size and motor unit number. *Experimental Brain Research*, 157:417–430, 2004.

G. E. Hinton and V. Nair. Inferring motor programs from images of handwritten digits. In *Advances in Neural Information Processing Systems*, pages 515–522, 2005.

A. Hyvärinen. Survey on independent component analysis. *Neural Computing Surveys*, 2:94–128, 1999.

A. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor lanscapes for motor primitives. In *Advances in Neural Information Processing Systems*, volume 15, pages 1523–1530, 2003.

W.J. Kargo and S.F. Giszter. Rapid corrections of aimed movements by combination of force-field primitives. *J. Neurosci.*, 20:409–426, 2000.

Y. Karklin and M. S. Lewicki. A hierarchical bayesian model for learning non-linear statistical regularities in non-stationary natural signals. *Neural Computation*, 17(2): 397–423, 2005.

O. D. Kharraz-Tavakol, T. Eggert, N. Mai, and A. Straube. Learning to write letters: transfer in automated movements indicates modularity of motor programs in human subjects. *Neurosci Lett.*, 17(1-2):33–36, 2000.

K. P. Körding, S. Ku, and D. M. Wolpert. Bayesian integration in force estimation. *Journal of Neurophysiology*, 92:3161–3165, 2004.

D.M. Lee. How movement is guided. http://www.perception-in-action.ed.ac.uk/PDF_s/Howmovementisguided.pdf, 2006.

D.N. Lee, A.P. Georgopoulos, M.J. Clark, C.M. Craig, and N.L. Port. Guiding contact by coupling the taus of gaps. *Exp. Brain Res.*, 139(2):151–159, 2001.

M. S. Lewicki and T. J. Sejnowski. Coding time-varying signals using sparse shift-invariant representations. *Advances in Neural Information Processing Systems*, 11: 730–736, 1999.

S. G. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, 1993.

C. Marquardt, W. Gentz, and N. Mai. Visual control of automated handwriting movements. *Exp. Brain Research*, 128:224–228, 1999.

M.J. Matarić. Primitives-based humanoid control and imitation. Technical report, DARPA MARS project, 2004.

D.V. Meegan, R.N. Aslin, and R. A. Jacobs. Motor timing learned without motor training. *Nature Neuroscience*, 3(9):860–862, 2000.

R. G. J. Meulenbroek and A. W. A. Van Gemmert. Advances in the study of drawing and handwriting. *Human movement science*, 22:131–220, 2003.

R.G.J. Meulenbroek, D.A. Rosenbaum, A.J.W.M. Thomassen, L.D. Loukopoulos, and J. Vaughan. Adaptation of a reaching model to handwriting: How different effectors can produce the same written output, and other results. *Psychological Research*, 59 (1):64–74, 1996.

F.A. Mussa-Ivaldi and E. Bizzi. Motor learning through the combination of primitives. *Phil. Trans. R. Soc. Lond.*, 355:1755–1769, 2000.

C. Padoa-Schioppa, C. S. R. Li, and E. Bizzi. Neuronal correlates of kinematics-to-dynamics transformation in the supplementary motor area. *Neuron*, 36:751–765, November 2002.

V.B. Penhume, R.J. Zatorre, and A.C. Evans. Cerebellar contributions to motor timing: A pet study of auditory and visual rythm reproduction. *Journal of Cognative Neuroscience*, 10(6):752–765, 1998.

L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Institute of Electrical and Electronic Engineers*, volume 77, pages 257–286, 1989.

J.O. Ramsay. Functional components of variation in handwriting. *Journal of the American Statistical Association*, 95:9–15, 2000.

S. Roweis and A. Alwan. Towards articulatory speech recognition: Learning smooth maps to recover articulator information. In *Proc. Eurospeech97*, pages 1227–1230, 1997.

S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert. Learning movement primitives. In *International Symposium of Robotics Research*, 2004.

Y. Singer and N. Tishby. Dynamical encoding of cursive handwriting. *Journal Biological Cybernetics*, 71(3):227–237, July 1994.

E. Smith and M. S. Lewicki. Efficient coding of time-relative structure using spikes. *Neural Computation*, 17(1):19–45, 2005.

S.M. LaVille S.R. Lindemann. Smoothly blending vector fields for global robot navigation. In *IEEE Conference Decision and Control*, 2005.

A.A. Handzel T. Flash. Affine differential geometry analysis of human arm movements. *Biological Cybernetics*, 96:577–601, 2007.

E. Todorov and M.I. Jordan. Optimal feedback control as a theory of motor coordination. *Nature Neuroscience*, 5(11):1226–1235, 2002.

R. J. van Beers, P Haggard, and D. M. Wolpert. The role of exucution noise in movement variability. *J. Neurophysiol.*, 91:1050–1063, 2004.

A. Vinciarelli and S. Bengio. Offline cursive word recognition using continuous density hidden Markov models trained with pca or ica features. In *16 th International Conference on Pattern Recognition*, volume 3, 2002.

B.H. Williams, M.Toussaint, and A.J. Storkey. Extracting motion primitives from natural handwriting data. In *International Conference on Artificial Neural Networks*, volume 2, pages 634–643, 2006.

B.H. Williams, M.Toussaint, and A.J. Storkey. A primitive based generative model to infer timing information in unpartitioned handwriting data. In *International Joint Conference on Artificial Intelligence*, pages 1119–1124, 2007.

B.H. Williams, M. Toussaint, and A.J. Storkey. Modelling motion primitives and their timing in biologically executed movements. In *Advances in Neural Information Processing Systems 20*, pages 1609–1616. 2008.

A. G. Witney and D. M. Wolpert. Spatial representation of predictive motor learning. *J. Neurophysiol.*, 89:1837–1843, 2003.

D. M. Wolpert. Probabilistic models in human sensorimotor control. *Human Movement Science*, 26:511–524, 2007.

D. M. Wolpert and M. Kawato. Multiple paired forward and inverse models for motor control. *Neural Networks*, 11:1317–1329, 1998.

D. M. Wolpert, Z. Ghahramani, and J. R. Flanagan. Perspectives and problems in motor learning. *TRENDS in Cog. Sci.*, 5(11):487–494, 2001.