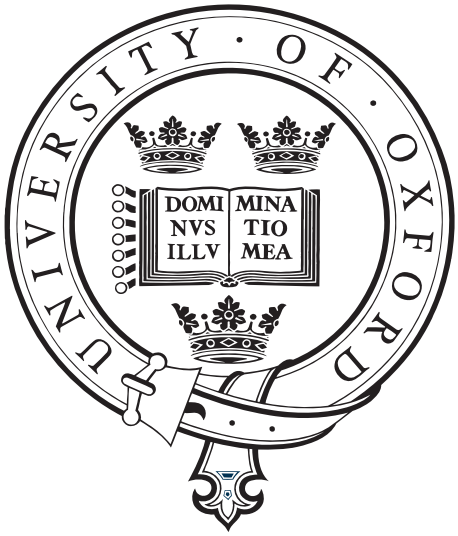
A Dynamic Logistic Model for Combining Classifier Outputs

Amber Tomas

Lady Margaret Hall

University of Oxford



A Thesis submitted for the degree of Doctor of Philosophy

Trinity Term 2008

Acknowledgements

I would like to thank the people who made my time here in Oxford possible, and those who made it what it was - a great journey. Specifically, thanks go to my parents for giving me so much freedom to choose the path which led me here that at times I thought they didn't care. Thanks also to whoever thought it would be a good idea to give young Adelaideans the financial support necessary to study at an international institution, and founded the John Crampton Travelling Scholarship. I would also like to mention the top-notch work of Prof F and team, without which I would almost surely not be in a position to write these acknowledgements today. Also big thanks to Prof Ripley for agreeing to supervise me and doing such a class job of it - I'm sure at least some of the wise words you passed on in our meetings have sunk in and will serve me in good stead for the future. Finally, big thanks to Joachim for your support, and for putting up with me as I tried to process and structure the barrage of new observations, thoughts and views encountered every day in Oxford into a workable philosophy.

A Dynamic Logistic Model for Combining Classifier Outputs

Amber Tomas, Lady Margaret Hall

DPhil thesis, Department of Statistics

Trinity Term 2008

Many classification algorithms are designed on the assumption that the population of interest is stationary, i.e. it does not change over time. However, there are many real-world problems where this assumption is not appropriate. In this thesis, we develop a classifier for non-stationary populations which is based on a multiple logistic model for the conditional class probabilities and incorporates a linear combination of the outputs of a number of pre-determined component classifiers. The final classifier is able to adjust to changes in the population by sequential updating of the coefficients of the linear combination, which are the parameters of the model.

The model we use is motivated by the relatively good classification performance which has been achieved by classification rules based on combining classifier outputs. However, in some cases such classifiers can also perform relatively poorly, and in general the mechanisms behind such results are little understood. For the model we propose, which is a generalisation of several existing models for stationary classification problems, we show there exists a simple relationship between the component classifiers which are used, the sign of the parameters and the decision boundaries of the final classifier. This relationship can be used to guide the choice of component classifiers, and helps with understanding the conditions necessary for the classifier to perform well.

We compare several "on-line" algorithms for implementing the classification model, where the classifier is updated as new labelled observations become available. The predictive approach to classification is adopted, so each algorithm is based on updating the posterior distribution of the parameters as new information is received. Specifically, we compare a method which assumes the posterior distribution is Gaussian, a more general method developed for the class of Dynamic Generalised Linear Models, and a method based on a sequential Monte Carlo approximation of the posterior. The relationship between the model used for parameter evolution, the bias of the parameter estimates and the error of the classifier is explored.

Declaration

I declare that this thesis is wholly my own work unless otherwise stated. No part of this thesis has been accepted or is currently being submitted for any degree or diploma or certificate or other qualification in this university or elsewhere.

Candidate: Amber TOMAS

Signed: _____ Date: _____

Contents

Chapter 1

Introduction

In this thesis we are concerned with developing a classifier for problems where the population of interest is not assumed to be stationary. Roughly speaking, we say a population is non-stationary (or "dynamic") if some feature of the population may change from one time-point to the next. Although the majority of work on classification algorithms is applicable only when the population of interest is stationary, there are many real-world problems where this assumption is not appropriate. For example, the average income of applicants for a bank loan who go on to default on their loan may increase with changing economic circumstances, and the features of junk emails will change as spammers attempt to evade the latest spam filters. If we were to ignore the possibility of such change, then all past observations would be wrongly assumed to have equal relevance to the current situation, so affecting the accuracy of a classification algorithm.

In this thesis, we develop an approach to the problem of dynamic classification. Our classification algorithm is based upon a logistic model which combines the outputs of a number of pre-determined classifiers. Combining the outputs of a set of classifiers has often been shown empirically to be capable of producing

good classification performance, often better than using a single classifier. However, sometimes combining classifier outputs can also result in poor classification rules, and the mechanisms behind when this will be the case are relatively little understood. Analysis of the model we use yields a result which helps to give an intuitive understanding of why this may be the case, and also the conditions under which the ability of the model to achieve good classification accuracy will breakdown. We can use this analysis to assist with specifying the model components for different problems, based on some prior knowledge of the population dynamics expected. In addition, it is necessary to have an algorithm for implementing the model and updating the parameters over time. In this thesis we compare three such algorithms, considering their suitability, efficiency and performance on some artificial test scenarios.

1.1 Background and Literature Review

The basic assumption underlying many classification algorithms is that the population of interest is "stationary". To define this term, we first introduce some basic notation. Suppose the population is composed of K sub-populations or "classes", labelled $1, 2, \ldots, K$. Certain features of the population can be measured, and the measurements taken at some time t are stored in a feature vector $\boldsymbol{X}_t$. Given an observed value $\boldsymbol{x}_t$ of $\boldsymbol{X}_t$, the aim of classification is to estimate the label of the class which generated this observation. In general, we denote the random variable representing the label of the class associated with $\boldsymbol{X}_t$ by Y_t, and an observed class label by y_t. We define a population as *stationary* if the conditional distribution of $Y_t|\boldsymbol{X}_t$ does not change with time. A population is referred to as *dynamic* if it is not assumed to be stationary. Hence a dynamic classification problem may include

periods of time for which $Y_t|\boldsymbol{X}_t$ is not changing, but there may also be periods where this distribution does change.

Much of the work which has been done previously on dynamic classification problems has assumed that the data is received sequentially. That is, at some time t an observation $\boldsymbol{x}_t$ is received, and then at a later time $t+1$ a further observation $\boldsymbol{x}_{t+1}$ is received. We also make this assumption throughout this thesis, as it is relevant to many practical problems. Further, we assume the class label y_t is revealed after $\boldsymbol{x}_t$ is classified, but before $\boldsymbol{x}_{t+1}$ is observed. Algorithms which receive data in this manner are often referred to as *on-line* algorithms, as the data observed at time t is used to update the classifier "on-line" before the observation $\boldsymbol{x}_{t+1}$ is received.

As well as being focused on the dynamic classification scenario, the other significant feature of the model we propose in this thesis is that it is based on combining the outputs of a number of classifiers. We denote the number of classifiers combined by M, and refer to these classifiers as *component classifiers*. The only restriction placed on the type of component classifier is that each classifier must output an estimate of the conditional distribution of Y_t when given an input $\boldsymbol{x}_t$. We denote the estimate output by the ith component classifier of $\text{Prob}\{Y_t = k|\boldsymbol{X}_t = \boldsymbol{x}_t\}$ by $\hat{p}_i(k|\boldsymbol{x}_t)$, for $k = 1, 2, \ldots, K$. The final classifier, which combines the outputs of the component classifiers, is referred to as the *combined classifier*. The estimate of $\text{Prob}_t\{Y_t = k|\boldsymbol{X}_t = \boldsymbol{x}_t\}$ output by the combined classifier is denoted $\hat{p}_t(k|\boldsymbol{x}_t)$, for $k = 1, 2, \ldots, K$.

There has been earlier work done on applying combined classifiers to dynamic scenarios, much of which is summarised in Kuncheva (2004). A distinguishing feature of our approach is that it is model-based, and the parameters are updated on-line within a Bayesian framework.

We first review the relevant literature on dynamic classification, and then con-

sider the work done on classifier combination, as our work combines aspects of both.

1.1.1 Dynamic Classification

Recently there has been increased interest in the problem of classifying sequential observations from dynamic populations. Many of the methods developed for on-line classification in dynamic scenarios have been published in the machine learning literature, and are heuristic in nature. A common feature of all dynamic classifiers is that they must incorporate some form of "forgetting", which is often referred to as having a "limited memory". That is, there must exist a mechanism which reflects the fact that some observations will be more relevant to the present population scenario than others. Often, it is assumed that recent observations are most relevant to the current situation, in which case a "moving window" of the most recent observations is used to train the classifier. Algorithms such as FLORA (Widmer & Kubat, 1996) are based on using a moving window whose width varies to reflect the rate of perceived population change. FLORA updates the classifier after each observation, but alternatively re-training can be performed only when it is deemed necessary (for example Cunningham et al. (2003)). Other heuristics incorporate a method by which the classifier is sequentially updated based on the most recent observations. For example STAGGER (Schlimmer & Granger, 1986) is based on this approach, and to reflect the speed by which the population may be changing only modifies the classifier if the most recent observation was misclassified. Other methods incorporate "forgetting" through the use of decay terms on certain features of the classifier. An early example of this approach is the weighted majority algorithm of Littlestone & Warmuth (1989). They use a weighted combination of the outputs of several classifiers to make a final classification, and after each new observation the

weights assigned to the classifiers which made an incorrect classification are reduced according to a decay parameter.

Contrasting to the heuristic nature of the work described above, the theoretical properties of classification algorithms in the presence of population non-stationarity have been studied in the field of computational learning theory. The goal of this work is generally to investigate how the best achievable accuracy of classification is related to the rate of change of the population and the rate of new information being received. For example, Helmbold & Long (1994) derive a bound on how fast the population can change in order for the error rate of the classifier to be below a certain threshold. More recent work (Bartlett et al., 2000) provides tighter bounds. Similarly, Freund & Mansour (1997) derive an upper bound on the error of a simple classification algorithm as a function of the complexity of the classification problem and the rate of change of the population. They also make the comment that, without a lower bound on such error, it is very hard to evaluate the performance of a classifier compared to the best it is theoretically possible to obtain. Therefore, unlike in the stationary scenario when we can compare the error rate of the classifier to the Bayes error (or to an approximation of the Bayes error), there exists no such simple comparison in the dynamic case. However, the algorithms used in these analyses are often very basic, and the bounds are loose. Therefore, although the concepts presented provide a good framework for thinking about the performance of dynamic classification algorithms, the results are not used directly in this thesis.

Other methods for dynamic classification are based on modifying existing methods used in the stationary scenario. Examples include logistic regression (Penny & Roberts, 1999), decision trees (Hulten et al., 2001) and self-organising networks (Fritzke, 1997). The approach of Penny & Roberts (1999) is notable in that it is based on a parametric model in which the parameters are updated over time

according to a strict Bayesian framework. It is unusual because, in general, the model-based approach to dynamic classification has received relatively little attention. A clear exception to this statement is the class of Dynamic Linear Models (Harrison & Stevens, 1976), which we define here because of their relevance to our approach.

The general normal dynamic linear model (DLM) can be defined as follows (West & Harrison, 1997): Firstly, the observation vector $\boldsymbol{X}_t$ is related to the parameter vector $\boldsymbol{\beta}_t$ at time t via the model

$$\boldsymbol{X}_t = F_t^T \boldsymbol{\beta}_t + \boldsymbol{\nu}_t,\ \boldsymbol{\nu}_t \sim N(\mathbf{0}, W_t),$$

where F_t is a known evolution matrix and W_t is a known variance matrix. The parameter evolution follows a first-order Markov process modelled by

$$\boldsymbol{\beta}_t = G_t \boldsymbol{\beta}_{t-1} + \boldsymbol{\omega}_t,\ \boldsymbol{\omega}_t \sim N(\mathbf{0}, V_t),$$

where the parameter evolution matrix G_t and the parameter evolution variance V_t are assumed to be known. The distributions are assumed to be conditional on all prior information, and the errors $\boldsymbol{\nu}_t$ and $\boldsymbol{\omega}_t$ are assumed to be independent for all t. At time $t-1$, all the information about the model is summarised in the distribution of the parameters

$$\boldsymbol{\beta}_{t-1} | D_{t-1},$$

where D_{t-1} denotes the set of all information which has become available up to the current time. Under this framework, there exists an efficient and exact method for updating the distribution of the parameters after having observed $\boldsymbol{X}_t$, using Bayesian updating equations (West & Harrison, 1997, section 4.3). However, for

classification problems we observe the class labels y_t rather than a direct observation of $\boldsymbol{X}_t$. That is, for classification the observation distribution is multinomial rather than Gaussian as assumed by the DLM. West et al. (1985) addressed this problem by extending the class of DLMs to the class of Dynamic Generalised Linear Models (DGLMs). In this class the observation distribution is assumed only to belong to the exponential family, so includes the multinomial case encountered in classification problems.

Because Bayesian updating allows simple incorporation of model uncertainty and new information, it is often considered naturally suited to the task of sequential updating of parametric models. Furthermore, as opposed to the heuristic methods discussed above, Bayesian updating supplies a rigorous framework for incorporating new information and updating the parameters of the model over time.

1.1.2 Combining Classifiers

Methods for combining classifiers in stationary scenarios have seen much attention since the early 1990s. Inspired by results showing the effectiveness of combining regression estimates to reduce prediction error (for example Perrone & Cooper (1992); Breiman (1996)), researchers began to experiment with combining classifiers. Whilst there was much interest in the topic, research developed almost parallel in several different communities, such as neural networks, machine learning and statistics.

It was quickly established amongst the neural network community that combining a number of neural networks into an "ensemble" could produce better classifications than any of the individual networks (Hansen & Salamon, 1990). In machine learning, Wolpert (1992) introduced the concept of feeding the outputs of a number of classifiers into a further classifier, in a process he called "stacked generalization".

This approach was examined by Breiman (1993) and LeBlanc & Tibshirani (1996), which raised interest in combining classifiers amongst the broader statistical community.

Whilst it was quickly established that combining classifier outputs was a very powerful technique for classification, understanding why this was so has taken longer to develop. It is well known that the ability of combining regression models to reduce prediction error can be explained in terms of a simple bias-variance relationship. Specifically, suppose we have M regression estimates $\hat{f}_i(x),\ i = 1, 2, \ldots, M$, and the combined regression estimate is of the form

$$\hat{f}(x) = \sum_{i=1}^{M} w_i \hat{f}_i(x), \tag{1.1}$$

where the weights $w_1, w_2, \ldots, w_M$ sum to one. In the special case that the weights are all equal, the estimates $\hat{f}_i(x)$ are unbiased and the errors $\hat{f}_i(x) - f(x)$ are independent for all i and x, it can be shown (Perrone, 1993) that the mean square error of the combination is $1/M$th of the average mean square error of the regression estimates which were combined. Further, for general combinations of the form (1.1) it is easily shown that

$$\mathrm{MSE}[\hat{f}(x)] = \mathrm{Var}(\hat{f}(x)) + \mathrm{bias}^2(\hat{f}(x)). \tag{1.2}$$

If all the regression estimates are equally biased, then the bias term of this expression is the same for the combination and the estimates of the combination. Therefore the reduction in mean squared error obtained when combining can be attributed to a reduction in the variance term of (1.2). A further explanation for the success of combining regression estimates was provided by Krogh & Vedelsby (1995). They showed that the mean squared error of the combination (1.1) is equal to the average

mean squared error of the component estimates, minus the "ensemble ambiguity". The ensemble ambiguity is the average expected squared difference between the prediction of a single regression estimate and the prediction of the combined estimate. Hence, roughly speaking, the more different the component regression estimates are to one another for fixed average mean squared error, the lower the mean squared error of the combination.

The existence of such results for regression has inspired researchers to look for similar relationships to explain the behaviour of classifier combinations. However, such a relationship has remained largely elusive (Kuncheva & Whitaker, 2003; Kuncheva, 2003). One difficulty encountered is that for classification the concepts of bias and variance do not have a commonly accepted definition. This has led to researchers seeking to define these terms based on a form which has a convenient decomposition, rather than on a theoretical or philosophical basis. James (2003) provides a comparison of many of these definitions.

Related to this work, and again inspired by the results of Krogh & Vedelsby (1995), there has been much research into trying to understand the relationship between the "diversity" of the component classifiers and the "accuracy" of the combined classifier. The diversity of a set of component classifiers is some measure of how different the component classifiers are to one another, similar to the ambiguity of a set of regression estimates. However, as for bias and variance, the diversity of a set of component classifiers is a concept without a commonly accepted definition. This has led to a multitude of empirical studies trying to find a definition of diversity which can be shown to be directly linked to accuracy. The relationships found are often weak, and due to the nature of the studies lack any theoretical justification (Kuncheva & Whitaker, 2003).

One of the factors complicating the analysis of combined classifiers is that there

exist a very large number of possible combining rules, and ways to choose the number and type of component classifiers which are used. A commonly used combining rule is majority vote, which is used in the well-known algorithm AdaBoost (Freund & Schapire, 1997), for example. Majority vote methods produce a classification based on the labels output by the component classifiers when given an input $\boldsymbol{x}_t$. However, in this thesis we will restrict ourselves to using component classifiers which output an estimate of the conditional class distribution $Y_t|\boldsymbol{X}_t$. Assuming that we have component classifiers which output such information, then it seems sensible to use all this information, rather than use only the labels output by the component classifiers. For this reason we do not consider majority vote algorithms in this thesis.

Although most research on the performance of combined classifiers has focused on empirical studies examining the relationship between the error rate of the combined classifier and measures of diversity, there has been some theoretical analysis performed. The combining rule most studied is called *weighted averaging*; given M component classifiers, the outputs of the combined classifier are given by

$$\hat{p}(k|\boldsymbol{x}_t) = \sum_{i=1}^{M} \beta_i \, \hat{p}_i(k|\boldsymbol{x}_t), \;\; k = 1, 2, \ldots, K, \tag{1.3}$$

where $\sum \beta_i = 1$. In the case that all the weights $\beta_i, i = 1, 2, \ldots, M$ are equal, this combining rule is referred to as *simple averaging*. Tumer & Ghosh (1996) analyse the performance of simple averaging by studying the expected location and variance of the decision boundary of the combined classifier on problems with a one-dimensional feature space and two classes. They show that the variance of the decision boundary of the combined classifier is less than the variance of the decision boundary of a single classifier, which results in the combined classifier having lower expected error. This work was extended to weighted averaging with

non-negative weights by Fumera & Roli (2002), who also studied when this form of weighted averaging is likely to significantly outperform simple averaging (Fumera & Roli, 2003). Although the results are enlightening, they are based on rather strict conditions which the authors acknowledge can not always be met in practice. For example, it is assumed that the decision boundary of the component classifiers will be close enough to the Bayes boundary that

a) a linear approximation to the conditional class densities in the region between these decision boundaries is reasonable, and

b) the density $p(x)$ can be approximated by $p(x^*)$ for all x, where x^* is the Bayes boundary.

Hence it may be questionable to extend this analysis to cases where some of the component classifiers may have decision boundaries far from optimal. In addition, the results obtained are relevant only prior to the component classifiers being trained or selected. They do not apply to the case that, *given* a set of component classifiers, we wish to combine their outputs.

Therefore, there is no simple interpretation of when combining the outputs of a set of classifiers may outperform the standard approach of selecting a single classifier, and no intuitive explanation for why combining can be so effective. Hence there remain many open questions and no simple explanations for the performance of a classifier combination, even in stationary scenarios.

1.2 Further Definitions and Assumptions

We now present the basic framework we use to discuss the classification problem, and outline some of the assumptions made in later analyses.

Firstly, we need to define the criteria we use to assess a classifier. For stationary

classification problems, the goal is generally to design a classifier with minimum expected loss. A loss function $L(j,k)$ specifies the penalty incurred when an observation from class k is classified as having come from class j. Generally $L(j,j) = 0$ for all $j = 1, 2, \ldots, K$, and $L(j,k) > 0$ for all $k \neq j$. If we denote the label output by a classifier when given input $\boldsymbol{x}_t$ by $c(\boldsymbol{x}_t)$, then the expected loss is defined as

$$\sum_{k=1}^{K} \text{Prob}\{Y = k\} \sum_{j=1}^{K} L(j,k) \text{Prob}\{c(\boldsymbol{X}_t) = j | Y_t = k\}. \tag{1.4}$$

Often we will use the "0–1" loss function,

$$L(j,k) = \begin{cases} 1 & \text{if } j \neq k, \\ 0 & \text{if } j = k \end{cases},$$

in which case the expected loss is equal to the error rate.

In dynamic classification problems, it is not so straightforward to specify a criteria for assessing classifiers, because a classifier that may perform relatively well in some time periods may not perform so well in others. To decide which of two classifiers is "better", we would need a loss function specified as a function of time. To simplify this issue, we will assume that the loss function used does not change with time. Hence we can use the average loss over some fixed time period to rank the performances of different classifiers. If we use 0–1 loss, then this is equivalent to saying that the "best" classifier after having classified t observations is that with the lowest cumulative error at that time.

Many of the results of this thesis will be based on analysis of the "decision boundaries" of a classifier. The *decision boundaries* of a classifier partition the feature space into labelled regions such that any observation which falls within a region will be classified with the corresponding label. Therefore, there is a one-to-

one relationship between a classification rule and the decision boundaries of that classifier[1]. Suppose we knew the true conditional class probabilities at time t, $p(k|\boldsymbol{x}_t)$. Then the classification rule which minimises the expected loss (1.4) would be

$$c(\boldsymbol{x}_t) = \text{argmin}_k \sum_{j=1}^{K} L(k,j)p(j|\boldsymbol{x}_t), \tag{1.5}$$

(see for example Ripley (1996)). This is sometimes referred to as the *Bayes rule*, and the corresponding expected loss as the *Bayes risk*. Further, the decision boundaries corresponding to the rule (1.5) are called the *Bayes boundaries*. The expected loss of a classifier will be equal to the Bayes risk only if the decision boundaries of that classifier are equal to the Bayes boundaries.

Throughout the thesis we will often make several simplifying assumptions. Although these assumptions are restrictive, they allow us to gain insights into the problem which will hopefully allow the work to be generalised. Unless explicitly stated otherwise, we will assume the use of a 0–1 loss function. Often, we will assume that $K = 2$, i.e. there exist only two classes. Additionally, we sometimes restrict the form of the component classifiers to being linear discriminant analysis (LDA) classifiers. If the observations from each class $k = 1, 2, \ldots, K$ are distributed normally with mean $\boldsymbol{\mu}_k$ and common covariance matrix Σ, and the prior probability an observation is from class k is denoted π_k, then under 0–1 loss the optimal rule (1.5) can be written

$$c(\boldsymbol{x}_t) = \text{argmin}_k\{-2\boldsymbol{\mu}_k^T\Sigma^{-1}\boldsymbol{x} + \boldsymbol{\mu}_k^T\Sigma^{-1}\boldsymbol{\mu}_k - 2\log\pi_k\} \tag{1.6}$$

(see for example Ripley (1996)). When we refer to a "linear discriminant analysis

[1]For stochastic or "fuzzy" classification rules this remark does not apply, but in this thesis we consider only deterministic rules.

classifier" (LDA classifier), we refer to a classifier that estimates the parameters $\boldsymbol{\mu}_k$, $\pi_k, k = 1, 2, \ldots, K$ and Σ^{-1} from a training set, and plugs these estimates into the rule (1.6). An important feature of such a classifier is that the decision boundary between any two classes is linear.

1.3 Structure of the Thesis

In chapter 2 we begin by introducing the dynamic classification model which forms the basis of this thesis. We then discuss the interpretation and significance of various components of the model. Of significant importance to the performance of the final classifier is the choice of component classifiers. This issue is discussed in detail in chapter 3, where we show that the choice of component classifiers should depend on the expected degree of movement in the optimal decision boundary over time, and the constraints placed on the parameter values. We discuss how this relates to the topic of "accuracy" and "diversity" for component classifier selection. In chapter 4 we discuss issues relating to implementation of the classifier, and several algorithms for parameter updating. The results of applying these algorithms to several artificial classification problems are presented in section 4.5. Finally, in chapter 5 we present our conclusions and ideas for further work.

Chapter 2

The Classification Model

In this chapter we define and discuss the model upon which our classifier will be based. The model is defined in terms of two inter-related components: a parametric model of the conditional class distribution, and a model for how the parameters change over time. In this chapter we focus mainly on the first of these components, as well as discussing interpretation of changes in the parameters.

The main results of this thesis are presented in chapter 3, and are based on analysing the decision boundaries of the component and combined classifiers. In this chapter we examine how the form of the model and the values of the parameters relate to the decision boundaries of the combined classifier, hence laying the foundation for later analysis.

2.1 Conditional Class Distribution

As mentioned in chapter 1, we wish to model the conditional class distribution as a combination of the outputs of a number of component classifiers. We assume the number of component classifiers, M, and the component classifiers themselves

have been pre-specified and are fixed. The only restriction on the form of the component classifiers is that when given an input $\boldsymbol{x}_t$ they output an estimate of the probabilities $\text{Prob}\{Y_t = k|\boldsymbol{X}_t = \boldsymbol{x}_t\}$, for $k = 1, 2, \ldots, K$. The estimates are denoted $\hat{p}_i(k|\boldsymbol{x}_t), i = 1, 2, \ldots, M$. Firstly, consider adapting the weighted averaging model (1.3) to the dynamic scenario by allowing the weights to vary with the time index t. This would yield the model

$$p_t(k|\boldsymbol{x}_t; \boldsymbol{\beta}_t) = \sum_{i=1}^{M} \beta_{ti} \hat{p}_i(k|\boldsymbol{x}_t), \;\; k = 1, 2, \ldots, K, \tag{2.1}$$

where the weight assigned to the ith component classifier at time t is denoted β_{ti}. In order for (2.1) to be a properly defined probability distribution, the weights $\beta_{ti}, i = 1, 2, \ldots, M$, must sum to one at all times t. This restriction limits the way in which we can model the evolution of the weights over time.

To avoid this problem, we instead use a multiple logistic model for the class probabilities. Generally, such a model has the form

$$\log\left(\frac{p_t(k|\boldsymbol{x}_t)}{p_t(1|\boldsymbol{x}_t)}\right) = g_k(\boldsymbol{x}_t; \boldsymbol{\beta}_t), \;\; k = 1, 2, \ldots, K, \tag{2.2}$$

for some functions $g_k(\boldsymbol{x}_t; \boldsymbol{\beta}_t)$ of the observations $\boldsymbol{x}_t$, with $g_1(\boldsymbol{x}_t; \boldsymbol{\beta}_t) = 0$ for all $\boldsymbol{x}_t$. Consider some vectors

$$\boldsymbol{\eta}_k(\boldsymbol{x}_t) = (\eta_{k1}(\boldsymbol{x}_t), \eta_{k2}(\boldsymbol{x}_t), \ldots, \eta_{kM}(\boldsymbol{x}_t))^T, \;\; k = 1, 2, \ldots, K,$$

where the ith component, $\eta_{ki}(\boldsymbol{x}_t)$, is a function of the output only of the ith component classifier when given input $\boldsymbol{x}_t$. That is,

$$\eta_{ki}(\boldsymbol{x}_t) = f(\hat{p}_i(1|\boldsymbol{x}_t), \hat{p}_i(2|\boldsymbol{x}_t), \ldots, \hat{p}_i(K|\boldsymbol{x}_t)), \;\; k = 2, \ldots, K,$$

for some function $f(\cdot)$ and $i = 1, 2, \ldots, M$. For the case $k = 1$, we define $\eta_{1i}(\boldsymbol{x}_t) = 0$ for all i and $\boldsymbol{x}_t$. Then we choose to define the functions $g_k(\boldsymbol{x}_t; \boldsymbol{\beta}_t)$ in (2.2) by

$$g_k(\boldsymbol{x}_t; \boldsymbol{\beta}_t) = \boldsymbol{\beta}_t^T \boldsymbol{\eta}_k(\boldsymbol{x}_t), k = 1, 2, \ldots, K. \tag{2.3}$$

Hence, we model the log ratio of probabilities (2.2) as a linear combination of the components $\boldsymbol{\eta}_k(\boldsymbol{x}_t)$, with weights $\boldsymbol{\beta}_t$. On rearranging (2.2) and substituting (2.3), our final model for the conditional class distribution at time t can be written

$$p_t(k|\boldsymbol{x}_t; \boldsymbol{\beta}_t) = \frac{\exp[\boldsymbol{\beta}_t^T \boldsymbol{\eta}_k(\boldsymbol{x}_t)]}{\sum_{j=1}^{K} \exp[\boldsymbol{\beta}_t^T \boldsymbol{\eta}_j(\boldsymbol{x}_t)]}, k = 1, 2, \ldots, K. \tag{2.4}$$

Using this model, the probability estimates will sum to one for any value of the weights $\boldsymbol{\beta}_t$, which we refer to as the parameters of the model.

Note that the model is defined in terms of the components $\boldsymbol{\eta}_k(\boldsymbol{x}_t)$. The form of these components is left open to choice, in order to increase the flexibility of the model. However, some choices for the form of these functions seem more sensible than others, and we discuss this issue in section 2.5.

In order to simplify the notation we will often drop the subscript t from the left hand side of (2.4) and write instead $p(k|\boldsymbol{x}_t; \boldsymbol{\beta}_t)$.

2.2 Model of Parameter Evolution

The logistic model (2.4) for the conditional class distribution is implicitly conditional on the value of the parameters $\boldsymbol{\beta}_t$, as well as the observation $\boldsymbol{x}_t$. Because we assume the component classifiers are determined in advance and the $\boldsymbol{\eta}_k(\boldsymbol{x}_t)$ are functions of the component classifier outputs, the functions $\boldsymbol{\eta}_k(\boldsymbol{x}_t)$ are also fixed. Therefore, any changes in the conditional class distribution over time must be mod-

elled through changes in the parameters $\boldsymbol{\beta}_t$. Note that an alternative method for adapting a combined classifier to the dynamic scenario would be to update the component classifiers over time. However, frequent re-training of component classifiers can be computationally intensive, and in addition there is not as yet a generally accepted method for training or selecting the component classifiers even for stationary problems with lots of available data. Therefore, in this thesis we prefer to explore a model-based approach in which the parameters are sequentially updated.

Unlike in a stationary logistic model, we require a model for how the parameters change over time. In this thesis, we consider the random walk model

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t + \boldsymbol{\omega}_t,\ \boldsymbol{\omega}_t \sim N(\mathbf{0}, V_t). \tag{2.5}$$

Hence, the parameter vector $\boldsymbol{\beta}_{t+1}$ is equal to the parameter vector at the previous time point, $\boldsymbol{\beta}_t$, plus a normally distributed random component $\boldsymbol{\omega}_t$ which has mean zero and variance-covariance matrix V_t. Thus we are indirectly modelling the changes in the population over time through changes in the parameters.

2.2.1 Relationship to Population Change

Generally, we will not expect the model (2.5) to be "correct" in the sense that we do not really expect the parameter evolution to follow a random walk. Of interest when investigating the suitability of this model is to determine the changes in the optimal parameters induced by the kinds of changes in the population which might be expected.

From the logistic model (2.4) we can see that $p(k|\boldsymbol{x}_t; \boldsymbol{\beta}_t)$ is a continuous function of the parameters $\boldsymbol{\beta}_t$. Hence if the population undergoes a sudden change which results in a change in the conditional class distribution, we would expect the optimal

parameter values to also undergo a sudden change[1]. Conversely, slow changes (or "drift") of the population should result in slow changes in the conditional class probabilities and therefore in the optimal parameters. However, there is no reason to expect all parameter values to change at the same rate. For example, in the two-class case, if we denote $p(2|\boldsymbol{x}_t;\boldsymbol{\beta}_t)$ by P_t, then

$$\begin{aligned}\frac{\partial P_t}{\partial \beta_{ti}} &= \frac{\exp\{\boldsymbol{\beta}_t^T\boldsymbol{\eta}_2(\boldsymbol{x}_t)\}\eta_{2i}(\boldsymbol{x})}{1+\exp\{\boldsymbol{\beta}_t^T\boldsymbol{\eta}_2(\boldsymbol{x}_t)\}}\left[1-\frac{\exp\{\boldsymbol{\beta}_t^T\boldsymbol{\eta}_2(\boldsymbol{x}_t)\}}{1+\exp\{\boldsymbol{\beta}_t^T\boldsymbol{\eta}_2(\boldsymbol{x}_t)\}}\right] \\ &= \eta_{2i}(\boldsymbol{x}_t)P_t(1-P_t). \end{aligned} \tag{2.6}$$

Hence the change in $p(2|\boldsymbol{x}_t;\boldsymbol{\beta}_t)$ for a fixed change in β_{ti} will be proportional to the value of $\eta_{2i}(\boldsymbol{x}_t)$. In addition, because $\boldsymbol{\eta}_2(\boldsymbol{x}_t)$ is a function of the component classifier outputs, this demonstrates that to determine the optimal parameter change induced by a specific population change, we must condition on the set of component classifiers used.

A major drawback to modelling changes in $p(k|\boldsymbol{x}_t)$ through changes in the parameters is that our prior knowledge will generally be about the types and rate of changes in the population, and this is not necessarily easy to convert to information about the parameter change. This problem is compounded by the fact that the effect of any parameter change is conditional on the component classifiers which are used.

However, although the model we have chosen may not be most efficient in light of prior information about population changes, it is likely to be flexible and able to adapt to most types of change. In addition, we can incorporate prior knowledge

[1]Changes in the population will usually, but not always, result in a change in the conditional class probabilities $p(k|\boldsymbol{x})$. This is discussed in Kelly et al. (1999). In addition, if the component classifiers are such that there does not exist a $\boldsymbol{\beta}_t$ such that $\hat{p}(k|\boldsymbol{x}_t,\boldsymbol{\beta}_t) = p(k|\boldsymbol{x}_t)$ for all $\boldsymbol{x}_t$ and $k = 1, 2, \ldots, K$, then population changes may not necessarily mean that the optimal value of $\boldsymbol{\beta}_t$ changes.

about the rate of change or volatility of the population through the variance of the random walk for parameter evolution, V_t. The implementation of this aspect of the model is discussed in more detail in chapter 4.

2.3 Classification Rule

Unless explicitly stated, we will assume the use of the 0–1 loss function (1.4). In this case, the optimal classification rule (1.5) can be written as

$$c(\boldsymbol{x}_t) = \text{argmax}_k \; p(k|\boldsymbol{x}_t).$$

Therefore, we assume use of the classification rule

$$c(\boldsymbol{x}_t) = \text{argmax}_k \; p(k|\boldsymbol{x}_t; \boldsymbol{\beta}_t), \tag{2.7}$$

where the probabilities $p(j|\boldsymbol{x}_t; \boldsymbol{\beta}_t)$ are given by the logistic model (2.4). Note that in this case, the decision boundary between classes j and j' can be written as the set of points

$$\{\boldsymbol{x}_t : p(j|\boldsymbol{x}_t; \boldsymbol{\beta}_t) = p(j'|\boldsymbol{x}_t, \boldsymbol{\beta}_t), \; p(j|\boldsymbol{x}_t, \boldsymbol{\beta}_t) > p(k|\boldsymbol{x}_t, \boldsymbol{\beta}_t), k \neq j, j'\}. \tag{2.8}$$

2.4 Notes on the parameters $\boldsymbol{\beta}_t$

The logistic model (2.4) has M parameters, $\beta_{t1}, \beta_{t2}, \ldots, \beta_{tM}$. Considered as "mixing weights" for the components $\eta_{k1}(\boldsymbol{x}_t), \eta_{k2}(\boldsymbol{x}_t), \ldots, \eta_{kM}(\boldsymbol{x}_t), k = 1, 2, \ldots, K$, the weight assigned to the ith component of $\boldsymbol{\eta}_k(\boldsymbol{x}_t)$ is the same for all k. Although allowing a different parameter for each i and k would result in a classifier with min-

imum error no bigger than the error of (2.7), using the same weight for each class k simplifies the model and allows for some interpretation of the parameter values. In addition, it reduces the number of parameters to be estimated and hence reduces the computational complexity of parameter updating.

Ideally, the "true" value of the parameters will be that for which the probabilities $p(k|\boldsymbol{x}_t, \boldsymbol{\beta}_t)$ are equal to the true conditional class probabilities. However, depending on the choice of component classifiers and functions $\boldsymbol{\eta}_k(\boldsymbol{x}_t)$, such a value may not exist. Therefore, roughly speaking, we can think of the unknown parameter value as that which minimises the error rate of the classification rule (2.7), subject to the conditional class probabilities being as close as possible to the true probabilities.

Note 2.4.1. *Either the components of $\boldsymbol{\beta}_t$ or $\boldsymbol{\eta}_k(\boldsymbol{x}_t)$ should be allowed to take both negative and positive values.*

Proof. From the equation of the decision boundary (2.8) and the logistic model for the conditional class probabilities (2.4), it can be seen that the decision boundary between class 1 and class k is a subset of the points

$$\begin{aligned} \left\{\boldsymbol{x}_t : \exp[\boldsymbol{\beta}_t^T \boldsymbol{\eta}_1(\boldsymbol{x}_t)] \right. &= \left. \exp[\boldsymbol{\beta}_t^T \boldsymbol{\eta}_k(\boldsymbol{x}_t)]\right\}, \text{ i.e. of} \\ \{\boldsymbol{x}_t : \boldsymbol{\beta}_t^T \boldsymbol{\eta}_k(\boldsymbol{x}_t) &= 0\}. \end{aligned}$$

Aside from the trivial case $\boldsymbol{\beta}_t = \mathbf{0}$, this will be non-empty only if the components of at least one of $\boldsymbol{\beta}_t$ or $\boldsymbol{\eta}_k(\boldsymbol{x}_t)$ can take both positive and negative values. Hence if the components of the $\boldsymbol{\eta}_k(\boldsymbol{x}_t)$ are strictly non-negative, for example, then we must not constrain the parameter values to be only positive or only negative. If the components of $\boldsymbol{\eta}_k(\boldsymbol{x}_t)$ can take both negative and positive values, then we can constrain the parameters $\boldsymbol{\beta}_t$ to be either negative or non-negative without restricting the labels the classifier is able to output. □

Note 2.4.2. *The classifier (2.7) with parameters $\boldsymbol{\beta}_t$ is equivalent to the classifier with parameters $\alpha\boldsymbol{\beta}_t$, for any constant $\alpha > 0$.*

Proof. With parameters $\boldsymbol{\beta}_t$, the decision boundary between classes j and j' is the set of points

$$\{\boldsymbol{x}_t : \exp[\boldsymbol{\beta}_t^T \boldsymbol{\eta}_j(\boldsymbol{x}_t)] = \exp[\boldsymbol{\beta}_t^T \boldsymbol{\eta}_{j'}(\boldsymbol{x}_t)], \\ \exp[\boldsymbol{\beta}_t^T \boldsymbol{\eta}_j(\boldsymbol{x}_t)] > \exp[\boldsymbol{\beta}_t^T \boldsymbol{\eta}_k(\boldsymbol{x}_t)], \ \forall\ k \neq j, j'\}. \tag{2.9}$$

If we use parameters $\alpha\boldsymbol{\beta}_t$, then the decision boundary between classes j and j' is the set of points

$$\{\boldsymbol{x}_t : \exp[\alpha\boldsymbol{\beta}_t^T \boldsymbol{\eta}_j(\boldsymbol{x}_t)] = \exp[\alpha\boldsymbol{\beta}_t^T \boldsymbol{\eta}_{j'}(\boldsymbol{x}_t)], \\ \exp[\alpha\boldsymbol{\beta}_t^T \boldsymbol{\eta}_j(\boldsymbol{x}_t)] > \exp[\alpha\boldsymbol{\beta}_t^T \boldsymbol{\eta}_k(\boldsymbol{x}_t)], \ \forall\ k \neq j, j'\},$$

which is clearly the same as (2.9). □

Note that this result does not mean that the conditional class probability estimates $\hat{p}(k|\boldsymbol{x}_t, \boldsymbol{\beta}_t)$ output by the classifier with parameters $\boldsymbol{\beta}_t$ will be the same as those of the classifier using $\alpha\boldsymbol{\beta}_t$ (and in general they won't be). However, because the decision boundaries of the two classifiers are the same, the classifications made by each of the two classifiers will be identical. This relationship will become important when we consider parameter estimation in chapter 4.

Note 2.4.3. *In the two-class case, the labels output by the classifier with parameters $\boldsymbol{\beta}_t$ are the opposite of those output by the classifier with parameters $-\boldsymbol{\beta}_t$.*

Proof.

$$\begin{aligned} p(1\,|\boldsymbol{x}_t; \boldsymbol{\beta}_t) &= 1 - p(2\,|\boldsymbol{x}_t; \boldsymbol{\beta}_t) \\ &= 1 - \frac{\exp[-\boldsymbol{\beta}_t^T \boldsymbol{\eta}_2(\boldsymbol{x}_t)]}{1 + \exp[-\boldsymbol{\beta}_t^T \boldsymbol{\eta}_2(\boldsymbol{x}_t)]} \\ &= \frac{1}{1 + \exp[-\boldsymbol{\beta}_t^T \boldsymbol{\eta}_2(\boldsymbol{x}_t)]} \\ &= \frac{\exp[\boldsymbol{\beta}_t^T \boldsymbol{\eta}_2(\boldsymbol{x}_t)]}{1 + \exp[\boldsymbol{\beta}_t^T \boldsymbol{\eta}_2(\boldsymbol{x}_t)]} \\ &= p(2\,|\boldsymbol{x}_t; \boldsymbol{\beta}_t). \end{aligned}$$

Therefore, when using the classifier (2.7), the result holds. □

Corollary 2.4.1. *In the two-class case, multiplying the ith component of $\boldsymbol{\beta}_t$ by -1 is equivalent to replacing the ith component classifier with a component classifier which outputs the opposite labels to the ith component classifier.*

Proof. This follows from note 2.4.3 when we set β_{tj} to zero, for all $j \neq i$. □

Note 2.4.4. *In the two-class case, for any constant $\alpha' < 0$, the combined classifier with parameters $\alpha'\boldsymbol{\beta}_t$ outputs the opposite labels to the combined classifier with parameters $\boldsymbol{\beta}_t$.*

Proof. This follows directly from applying Note 2.4.2 to Note 2.4.3. □

Corollary 2.4.1 helps us to understand the effect of allowing negative parameter values. If a component of the parameter vector $\boldsymbol{\beta}_t$ changes in sign, then the influence of the corresponding component classifier is essentially reversed. Thinking about it the other way, replacing the ith component classifier with one which outputs the opposite labels is equivalent to multiplying the ith parameter β_{ti} by -1. This property will be important in chapter 3, when we consider how constraints on the parameters will influence the way we should select the component classifiers.

2.5 Specification of functions $\boldsymbol{\eta}_k(\boldsymbol{x}_t)$

As already discussed, the outputs of the component classifiers come into the logistic model (2.4) for the conditional class probabilities via the functions

$$\boldsymbol{\eta}_k(\boldsymbol{x}_t),\ k = 1, 2, \ldots, K.$$

For each k, we restrict the ith component of $\boldsymbol{\eta}_k(\boldsymbol{x}_t)$, $\eta_{ki}(\boldsymbol{x}_t)$, to be a function of the output only of the ith component classifier. This means that the ith term in the linear combination $\boldsymbol{\beta}_t^T \boldsymbol{\eta}_k(\boldsymbol{x}_t)$ is a function of only β_{ti} and the output of the ith component classifier. Although there are many choices for the form of the $\boldsymbol{\eta}_k(\boldsymbol{x}_t)$, we primarily consider two possibilities:

1.
$$\eta_{ki}(\boldsymbol{x}_t) = \log\left(\frac{\hat{p}_i(k|\boldsymbol{x}_t)}{\hat{p}_i(1|\boldsymbol{x}_t)}\right),\ \text{and} \tag{2.10}$$

2.
$$\eta_{ki}(\boldsymbol{x}_t) = \hat{p}_i(k|\boldsymbol{x}_t) - \hat{p}_i(1|\boldsymbol{x}_t). \tag{2.11}$$

Note that (2.10) takes values in $(-\infty, \infty)$ and (2.11) takes values in $[-1, 1]$. Therefore, as discussed in section 2.4, we can impose a non-negativity constraint on the parameters $\boldsymbol{\beta}_t$ without restricting the labels which can be output by the classifier.

Although we motivated our use of a logistic model by the drawbacks of the dynamic weighted averaging model (2.1), it is interesting to consider the similarities between that model and the model we obtain when using options (2.10) or (2.11).

Define

$$\begin{aligned} G(k|\boldsymbol{x}_t) &= \log\left(\frac{p(k|\boldsymbol{x}_t)}{p(1|\boldsymbol{x}_t)}\right), \text{ and} \\ \hat{G}_i(k|\boldsymbol{x}_t) &= \log\left(\frac{\hat{p}_i(k|\boldsymbol{x}_t)}{\hat{p}_i(1|\boldsymbol{x}_t)}\right). \end{aligned}$$

Then if we use option (2.10) for $\boldsymbol{\eta}_k(\boldsymbol{x}_t)$ we can write the model (2.4) as

$$G(k|\boldsymbol{x}_t) = \sum_{i=1}^{M} \beta_{ti}\ \hat{G}_i(k|\boldsymbol{x}_t),\ k = 1, 2, \ldots, K. \tag{2.12}$$

This has the same form as the weighted averaging model (2.1), i.e. a linear combination, but in (2.12) the components $G(k|\boldsymbol{x}_t)$ and $\hat{G}_i(k|\boldsymbol{x}_t)$ for $i = 1, 2, \ldots, M$ are not constrained to the interval $[0, 1]$, as are the probabilities in (2.1). Writing the model in this form allows comparison with the large body of work on linear combinations of regression estimates, for instance Perrone (1993), Breiman (1993) and LeBlanc & Tibshirani (1996). The latter considered models for classification very similar to the weighted averaging model (2.1) and the logistic model (2.12), the only difference being that they allowed the parameter values to vary over k as well as i, so that the number of parameters to estimate was KM rather than M. For the stationary case, they found empirically that when using non-negative parameter values the weighted averaging model similar to (2.1) gave better classification results than that using log ratios as in (2.12), though their experiments were quite limited.

If we instead define $\boldsymbol{\eta}_k(\boldsymbol{x}_t)$ by (2.11), we do not obtain a linear model as for the choice (2.10). However, in this case the decision boundary between classes j and j',

$$\{\boldsymbol{x}_t : \boldsymbol{\beta}_t^T \boldsymbol{\eta}_j(\boldsymbol{x}_t) = \boldsymbol{\beta}_t^T \boldsymbol{\eta}_{j'}(\boldsymbol{x}_t),\ \boldsymbol{\beta}_t^T \boldsymbol{\eta}_j(\boldsymbol{x}_t) > \boldsymbol{\beta}_t^T \boldsymbol{\eta}_k(\boldsymbol{x}_t), k \neq j, j'\},$$

can be written as

$$\{\boldsymbol{x}_t : \sum_{i=1}^{M} \beta_{ti}\, \hat{p}_i(j|\boldsymbol{x}_t) = \sum_{i=1}^{M} \beta_{ti}\, \hat{p}_i(j'|\boldsymbol{x}_t),\ \sum_{i=1}^{M} \beta_{ti}\, \hat{p}_i(j|\boldsymbol{x}_t) > \sum_{i=1}^{M} \beta_{ti}\, \hat{p}_i(k|\boldsymbol{x}_t), k \neq j, j'\}.$$

It can be seen that this is exactly the same as the decision boundaries obtained by using the classification rule of (2.7) but when the conditional class probabilities are given by the dynamic weighted averaging model (2.1). Therefore, from note 2.4.2, the dynamic logistic model (2.4) with $\boldsymbol{\eta}_k(\boldsymbol{x}_t)$ defined by (2.11) produces classifications which are the same as when using the dynamic weighted averaging model (2.1) with the parameters normalised to sum to one. Hence this choice for $\boldsymbol{\eta}_k(\boldsymbol{x}_t)$ can be viewed as equivalent to using the weighted averaging rule for combining classifier outputs. Accordingly, any result which holds for the dynamic logistic model (2.4) will also hold for the dynamic weighted averaging model (2.1), which in the stationary case is equivalent to weighted averaging.

Both choices (2.10) and (2.11) have the property that

$$\eta_{ki}(\boldsymbol{x}_t) \begin{cases} < 0 & \text{if } \hat{p}_i(k|\boldsymbol{x}_t) < \hat{p}_i(1|\boldsymbol{x}_t) \\ = 0 & \text{if } \hat{p}_i(k|\boldsymbol{x}_t) = \hat{p}_i(1|\boldsymbol{x}_t) \\ > 0 & \text{if } \hat{p}_i(k|\boldsymbol{x}_t) > \hat{p}_i(1|\boldsymbol{x}_t) \end{cases} . \tag{2.13}$$

At least in the two-class case, this allows some interpretation of the effect of including the ith component classifier. We have already stated that the parameters $\boldsymbol{\beta}_t$ should change over time to reflect changes in the population. Therefore, we might be interested in what changes in $\boldsymbol{\beta}_t$ tell us about the changing influence of the individual component classifiers. Through the logistic model (2.4), we have assumed that

$$p(k|\boldsymbol{x}_t) = \frac{\exp[\boldsymbol{\beta}_t^T \boldsymbol{\eta}_k(\boldsymbol{x}_t)]}{\sum_{j=1}^{K} \exp[\boldsymbol{\beta}_t^T \boldsymbol{\eta}_j(\boldsymbol{x}_t)]}, k = 1, 2, \ldots, K, \tag{2.14}$$

where $\boldsymbol{\eta}_1(\boldsymbol{x}_t) = \mathbf{0}$. Now let us denote $p(j|\boldsymbol{x}_t)$ by P_{jt}, and $\boldsymbol{\eta}_j(\boldsymbol{x}_t)$ by $\boldsymbol{\eta}_j$. Then differentiating (2.14) with respect to t yields the relation

$$\dot{P}_{kt} = P_{kt}\dot{\boldsymbol{\beta}}_t^T(\boldsymbol{\eta}_k - \sum_{j=1}^{K} P_{jt}\boldsymbol{\eta}_j), \tag{2.15}$$

where we have used the notation $\dot{X} = \partial X/\partial t$. In the two-class case, (2.15) simplifies to

$$\begin{aligned} \dot{P}_{2t} &= P_{2t}(1 - P_{2t})\dot{\boldsymbol{\beta}}_t^T \boldsymbol{\eta}_2 \\ &\propto \sum_{i=1}^{M} \dot{\beta}_{ti}\eta_{2i}. \end{aligned} \tag{2.16}$$

From (2.13), if $\dot{\beta}_{ti} > 0$, then

$$\begin{aligned} \dot{\beta}_{ti}\eta_{2i} &> 0 \text{ if } \hat{p}_{it}(2|\boldsymbol{x}_t) > \hat{p}_{it}(1|\boldsymbol{x}_t), \text{ and} \\ \dot{\beta}_{ti}\eta_{2i} &< 0 \text{ if } \hat{p}_{it}(1|\boldsymbol{x}_t) > \hat{p}_{it}(2|\boldsymbol{x}_t), \end{aligned}$$

and if $\dot{\beta}_{ti} < 0$ then

$$\begin{aligned} \dot{\beta}_{ti}\eta_{2i} &> 0 \text{ if } \hat{p}_{it}(1|\boldsymbol{x}_t) > \hat{p}_{it}(2|\boldsymbol{x}_t), \text{ and} \\ \dot{\beta}_{ti}\eta_{2i} &< 0 \text{ if } \hat{p}_{it}(2|\boldsymbol{x}_t) > \hat{p}_{it}(1|\boldsymbol{x}_t). \end{aligned}$$

Hence from (2.16), if $\dot{\beta}_{ti}$ is greater than zero and all other parameters are constant, then P_{2t} increases if the ith component classifier favours class two, and decreases if the ith component classifier favours class 1. If $\dot{\beta}_{ti}$ is less than zero, then P_{2t} increases if the ith component classifier favours class 1 and decreases if the ith component classifier favours class 2. Thus by examining changes in β_{ti} over time, one can see

how the predictions of the combined classifier have become either less ($\dot{\beta}_{ti} < 0$) or more ($\dot{\beta}_{ti} > 0$) like the predictions of the ith component classifier over time.

Suppose that one of the component classifiers has a much larger error rate than the other component classifiers at some time t. Then if we constrain the parameters to be non-negative, the parameters corresponding to the more accurate component classifiers should be relatively large in comparison to that corresponding to the less accurate component classifier. In this case we would expect the parameter estimate corresponding to the weak component classifier to be close to zero, so that it won't have much influence on the combined classifier. However, knowing that a component classifier is inaccurate provides useful information that could potentially be used to improve the combined classifier. By the results in section 2.4, we can see that such information can be usefully incorporated through allowing the use of negative parameter values.

2.6 Shape of the Decision Boundary

In this section we consider how the form of the $\boldsymbol{\eta}_k(\boldsymbol{x}_t)$ can restrict the flexibility of the decision boundary of the classifier (2.7). At any point in time, the decision boundary will depend on the value of the parameters $\boldsymbol{\beta}_t$, as well as the choice of functions $\boldsymbol{\eta}_k(\boldsymbol{x}_t)$ and the component classifiers. Here we only consider the decision boundary as a function of $\boldsymbol{\beta}_t$, when the component classifiers are LDA classifiers which have been specified in advance, and the $\boldsymbol{\eta}_k(\boldsymbol{x}_t)$s are as defined in (2.10) or (2.11).

Let $\hat{\pi}_{ik}, \hat{\boldsymbol{\mu}}_{ik}$ and $\hat{\Sigma}_i$ be the estimates of the prior probability, mean and variance respectively of class k output by the ith component classifier. Assuming that the ith component classifier is an LDA classifier, the estimated class posterior probabilities

can be written as

$$\begin{aligned}
\hat{p}_i(k|\boldsymbol{x}_t) &= \frac{\hat{p}_i(\boldsymbol{x}_t|k)\hat{\pi}_{ik}}{\sum_{j=1}^{K}\hat{p}_i(\boldsymbol{x}_t|j)\hat{\pi}_{ij}}, k = 1, 2, \ldots, K \\
&= \frac{\hat{\pi}_{ik}\exp\left(-\frac{1}{2}(\boldsymbol{x}_t-\hat{\boldsymbol{\mu}}_{ik})^T\hat{\Sigma}_i^{-1}(\boldsymbol{x}_t-\hat{\boldsymbol{\mu}}_{ik})\right)}{\sum_{j=1}^{K}\hat{\pi}_{ij}\exp\left(-\frac{1}{2}(\boldsymbol{x}_t-\hat{\boldsymbol{\mu}}_{ij})^T\hat{\Sigma}_i^{-1}(\boldsymbol{x}_t-\hat{\boldsymbol{\mu}}_{ij})\right)} \qquad (2.17) \\
&= \Big[1+\sum_{\substack{j=1\\j\neq k}}^{K}\hat{\pi}_{ij}/\hat{\pi}_{ik}\times\exp\big(-\frac{1}{2}(\boldsymbol{x}_t-\hat{\boldsymbol{\mu}}_{ij})^T\hat{\Sigma}_i^{-1}(\boldsymbol{x}_t-\hat{\boldsymbol{\mu}}_{ij}) \\
&\quad +\frac{1}{2}(\boldsymbol{x}_t-\hat{\boldsymbol{\mu}}_{ik})^T\hat{\Sigma}_i^{-1}(\boldsymbol{x}_t-\hat{\boldsymbol{\mu}}_{ik})\big)\Big]^{-1} \\
&= \left[1+\sum_{\substack{j=1\\j\neq k}}^{K}\hat{\pi}_{ij}/\hat{\pi}_{ik}\exp\left((\hat{\boldsymbol{\mu}}_{ij}-\hat{\boldsymbol{\mu}}_{ik})^T\hat{\Sigma}_i^{-1}(\boldsymbol{x}_t-\hat{\bar{\boldsymbol{\mu}}}_{ikj})\right)\right]^{-1}, \qquad (2.18)
\end{aligned}$$

where $\hat{\bar{\boldsymbol{\mu}}}_{ikj} = (\hat{\boldsymbol{\mu}}_{ik}+\hat{\boldsymbol{\mu}}_{ij})/2$.

Theorem 2.6.1. *The decision boundary of the classifier (2.7) when using LDA classifiers as the component classifiers and $\boldsymbol{\eta}_j(\boldsymbol{x}_t)$ defined by (2.10) is piecewise linear.*

Proof. In this case

$$\hat{p}(k|\boldsymbol{x}_t;\boldsymbol{\beta}_t) \propto \exp\left(\sum_{i=1}^{M}\beta_{ti}\log\left(\frac{\hat{p}_i(k|\boldsymbol{x}_t)}{\hat{p}_i(1|\boldsymbol{x}_t)}\right)\right),\ k = 1, 2, \ldots, K,$$

so the equation of the decision boundary between classes j and j' is a subset of the set

$$\begin{aligned}
\left\{\boldsymbol{x}_t : \sum_{i=1}^{M}\beta_{ti}\log\left(\frac{\hat{p}_i(j|\boldsymbol{x}_t)}{\hat{p}_i(1|\boldsymbol{x}_t)}\right)\right. &= \left.\sum_{i=1}^{M}\beta_{ti}\log\left(\frac{\hat{p}_i(j'|\boldsymbol{x}_t)}{\hat{p}_i(1|\boldsymbol{x}_t)}\right)\right\}, \text{ i.e.} \\
\left\{\boldsymbol{x}_t : \sum_{i=1}^{M}\beta_{ti}\log\left(\frac{\hat{p}_i(j|\boldsymbol{x}_t)}{\hat{p}_i(j'|\boldsymbol{x}_t)}\right)\right. &= \left. 0\right\}. \qquad (2.19)
\end{aligned}$$

On substituting expression (2.17) into (2.19) for $k = j$ and j' and rearranging, it can be shown that (2.19) can be written in the form

$$\{\boldsymbol{x}_t : (A\boldsymbol{\beta}_t)^T \tilde{\boldsymbol{x}}_t = 0\},$$

where, using a colon to denote matrix concatenation, $\tilde{\boldsymbol{x}}_t^T = (1 : \boldsymbol{x}_t^T)$ and

$$A = [\tilde{\boldsymbol{a}}_1 : \tilde{\boldsymbol{a}}_2 : \ldots : \tilde{\boldsymbol{a}}_M], \tag{2.20}$$

where

$$\tilde{\boldsymbol{a}}_i^T = \left(\log\left(\frac{\hat{\pi}_{ij}}{\hat{\pi}_{ij'}}\right) - (\hat{\boldsymbol{\mu}}_{ij} - \hat{\boldsymbol{\mu}}_{ij'})^T \hat{\Sigma}_i^{-1} \, \frac{\hat{\boldsymbol{\mu}}_{ij} + \hat{\boldsymbol{\mu}}_{ij'}}{2} \;:\; (\hat{\boldsymbol{\mu}}_{ij} - \hat{\boldsymbol{\mu}}_{ij'})^T \hat{\Sigma}_i^{-1} \right).$$

Hence in this case the decision boundary between any two classes is linear, and so the decision boundary is either linear ($K = 2$) or piecewise linear ($K > 2$). □

Theorem 2.6.2. *The decision boundary of classifier (2.7) when using LDA component classifiers and functions $\boldsymbol{\eta}_k(\boldsymbol{x}_t)$ defined by (2.11) is non-linear.*

Proof. In this case the decision boundary between classes j and j' is a subset of

$$\left\{ \boldsymbol{x}_t : \sum_{i=1}^{M} \beta_{ti} \left(\hat{p}_i(j|\boldsymbol{x}_t) - \hat{p}_i(1|\boldsymbol{x}_t) \right) = \sum_{i=1}^{M} \beta_{ti} \left(\hat{p}_i(j'|\boldsymbol{x}_t) - \hat{p}_i(1|\boldsymbol{x}_t) \right) \right\},$$

which is equivalent to

$$\left\{ \boldsymbol{x}_t : \sum_{i=1}^{M} \beta_{ti} \, \hat{p}_i(j|\boldsymbol{x}_t) = \sum_{i=1}^{M} \beta_{ti} \, \hat{p}_i(j'|\boldsymbol{x}_t) \right\}. \tag{2.21}$$

In the 2-class case, this simplifies to

$$\left\{ \boldsymbol{x}_t : \beta_{t1}\ \hat{p}_1(1|\boldsymbol{x}_t) + \beta_{t2}\ \hat{p}_2(1|\boldsymbol{x}_t) = \frac{\beta_{t1} + \beta_{t2}}{2} \right\}. \tag{2.22}$$

Substituting (2.18) into (2.22) yields

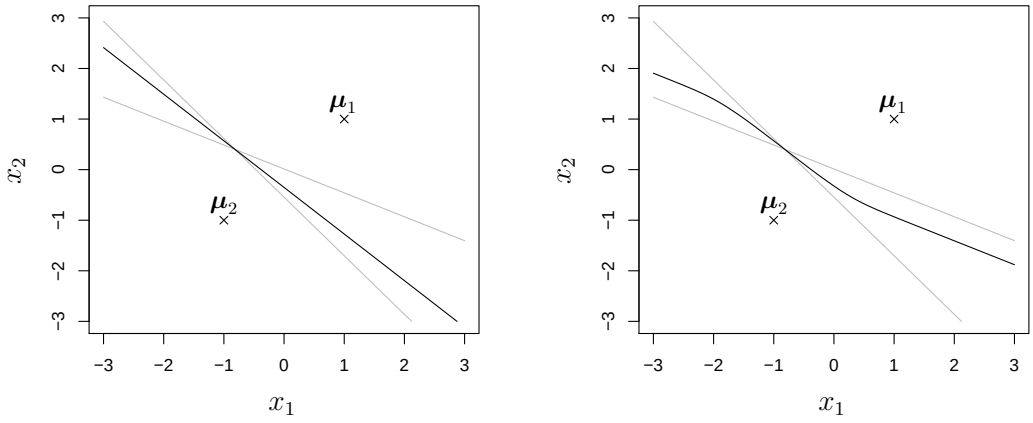
$$\left\{ \boldsymbol{x}_t : \frac{\beta_{t1}}{1 + \tilde{\pi}_1 \exp(Q_{1t})} + \frac{\beta_{t2}}{1 + \tilde{\pi}_2 \exp(Q_{2t})} = \frac{\beta_{t1} + \beta_{t2}}{2} \right\}, \tag{2.23}$$

where $\tilde{\pi}_i = \frac{\pi_{i2}}{\pi_{i1}}$ and $Q_{it} = (\hat{\boldsymbol{\mu}}_{i2} - \hat{\boldsymbol{\mu}}_{i1})^T \hat{\Sigma}_i^{-1} (\boldsymbol{x}_t - \hat{\bar{\boldsymbol{\mu}}}_i)$. Thus the decision boundary is clearly non-linear and has no simple form. □

Figures 2.1(a) and 2.1(b) show an example of the decision boundaries of the classifier (2.7) when using $\boldsymbol{\eta}_k(\boldsymbol{x}_t)$ defined by (2.10) and (2.11) respectively. In each example there are two classes, and each of the two component classifiers are trained on a random sample of four observations from each class. The grey lines represent the decision boundaries of the two component classifiers (LDA classifiers), and the solid black line is the decision boundary of the combined classifier. The means of each of the two classes are indicated.

If we use $\boldsymbol{\eta}_k(\boldsymbol{x}_t)$ defined by (2.11), small changes in the parameters $\boldsymbol{\beta}_t$ can result in large changes in the shape of the decision boundary. Figures 2.2(a) to 2.2(d) show the shape of the combined decision boundary for several values of the parameters for the same case as considered in figure 2.1. Notice that the decision boundary changes dramatically as the parameters change from $(1, -1)$ to $(1, -1.01)$. This is because as $|\beta_{t2}|$ becomes larger than $|\beta_{t1}|$, the second component classifier becomes "dominant"[2]. Due to the effect of the negative sign of β_{t2}, this will have a large effect on the decision boundary of the combined classifier. In a real application it

[2]The exact meaning of what we mean here by "dominant" is explained in more detail in section 3.4.

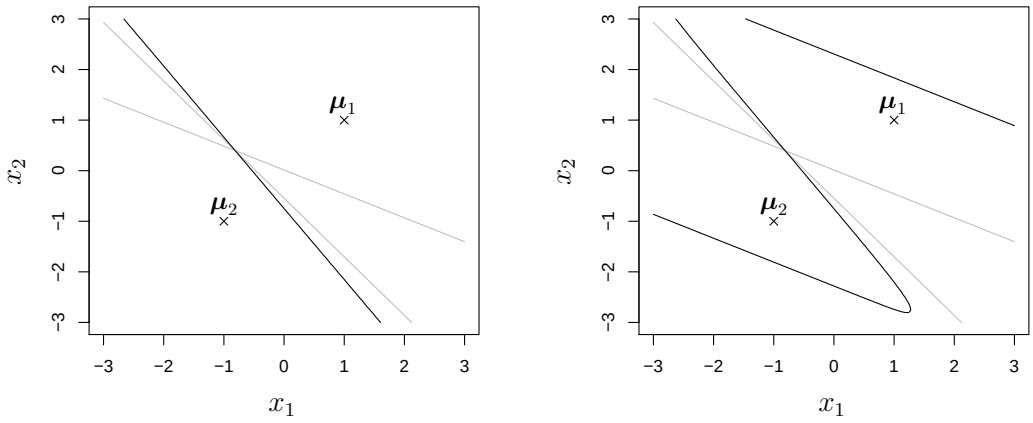


(a) $\boldsymbol{\eta}_j(\boldsymbol{x}_t)$ is defined as in (2.10)

(b) $\boldsymbol{\eta}_j(\boldsymbol{x}_t)$ is defined as in (2.11)

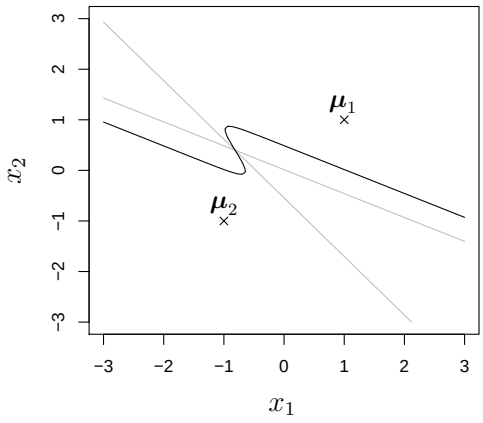
Figure 2.1: Decision boundaries of the two component classifiers (grey) and the combined classifier (black) for two choices of the functions $\boldsymbol{\eta}_j(\boldsymbol{x}_t)$, with parameters $\boldsymbol{\beta}_t = (1, 2)$.

is unlikely that such a scenario would occur; this example is simply to show that abrupt changes in the decision boundary for small changes in the parameters *can* occur. However, in general it seems that small changes in the parameters will result in small changes in the shape of the decision boundary.

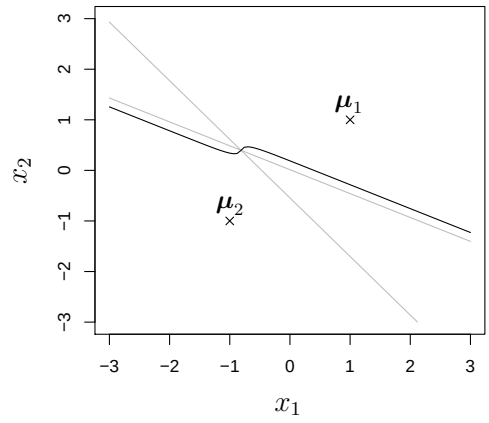


(a) $\boldsymbol{\beta}_t = (1, -1)$

(b) $\boldsymbol{\beta}_t = (1, -1.01)$



(c) $\boldsymbol{\beta}_t = (1, -2)$



(d) $\boldsymbol{\beta}_t = (1, -5)$

Figure 2.2: Decision boundary of the combined classifier for several values of the parameters $\boldsymbol{\beta}_t$ when the $\boldsymbol{\eta}_j(\boldsymbol{x}_t)$ are defined by (2.11) and the component classifiers are LDA classifiers. The decision boundaries of the component classifiers are shown in grey, and the boundary of the combined classifier in black.

Chapter 3

On the Choice of Component Classifiers

In chapter 2 we introduced the logistic model for the conditional class distribution (2.4), and discussed the influence of the form of the functions $\boldsymbol{\eta}_k(\boldsymbol{x}_t)$ and the parameter values. We assumed the set of component classifiers had already been specified. We also noted that the decision boundary and hence error rate of the final classifier are, through the $\boldsymbol{\eta}_k(\boldsymbol{x}_t)$, conditional on the component classifiers which are used. Because the choice of component classifiers has such an influence on the performance of the combined classifier, it is important to have some understanding of the mechanisms responsible for this. This knowledge can then be used to help with specifying which component classifiers should be used, or at least what properties the set of component classifiers should possess.

In this thesis we are not concerned with precisely how the component classifiers are trained. All that is important is that the set of features and the set of class labels correspond to those in the population of interest. If there is no data available *a priori* from the population on which to train the component classifiers, then artificial

data can be used. Indeed, even if real data is available, it may often be the case that artificial data is preferable, as this allows more control over training of the component classifiers.

The aim of the work in this chapter is to better understand the relationship between the component classifiers and the combined classifier. The approach we take, similarly to that of Tumer & Ghosh (1996), is to study the location of the decision boundary of the combined classifier. However, unlike the work of Tumer & Ghosh (1996) and Fumera & Roli (2002), the results we obtain can be related back to the actual physical location of the component classifier decision boundaries. This allows for a relatively intuitive interpretation of the result and further extension, which we then relate back to earlier results and empirical findings in the literature.

3.1 Selecting the number of component classifiers

Before we can consider which component classifiers should be used, we first consider the question of how many component classifiers should be included, i.e. the value of M. Clearly this should not be considered entirely independently of which M classifiers are chosen. For example, choosing an identical component classifier to one already used will increase the value of M by one but be totally redundant. However, it is still an important question to consider, and we show that for the case of linear decision boundaries at least, it is possible to specify a lower bound on the number of component classifiers required. Firstly, we make some comments on the general problem of specifying M, and then consider the case when the decision boundary of the combined classifier is restricted to being linear.

3.1.1 General Case

Consider combining two component classifiers via the model (2.4). Then for some value of the parameters β_{t1} and β_{t2} the error rate of the combined classifier will be at least as low as the error rate of the best component classifier. If we then include another component classifier, for some values of the parameters the error rate will be at least as low again. Following this argument, it would be optimal to include as many component classifiers as possible. The reason that this argument breaks down at some point is computational - as the number of component classifiers increases, so does the number of parameters. Hence implementing the model will become more computationally demanding, and the precision of the estimates will decrease. Selecting the number of classifiers is hence a trade-off between the flexibility of the decision boundary of the combined classifier, i.e. the extent to which the decision boundary can change when the parameter values are changed, and the computation and regularisation of the parameter estimates. Therefore, a reasonable approach to selecting the number of component classifiers would seem to be the following: determine both the desired amount of flexibility in the decision boundary and the maximum number of component classifiers, $M^{\max}$ say, that is computationally feasible for the computing power, time and amount of data available. The component classifiers should then be chosen either to reach the ideal degree of flexibility with as few classifiers as possible, or if that is not possible with only $M^{\max}$ classifiers, to maximise the flexibility of the decision boundary. However, determining a suitable threshold $M^{\max}$ and measuring the flexibility of the decision boundary are both likely to be non-trivial problems. In this thesis we consider only the case that we wish to restrict the decision boundary to being linear.

3.1.2 Linear Boundaries

In the special case that we want to consider only linear decision boundaries, it is possible to determine the minimum number of component classifiers which will be needed. That is, under the restriction that the boundary of the combined classifier be linear, it is possible to represent all such boundaries with a limited number of component classifiers. In this case, using more component classifiers only serves to increase the computational complexity of implementation.

Consider a classification problem with an N-dimensional feature space. Suppose each of the component classifiers is an LDA classifier, so that the decision boundary between any two classes is linear. In section 2.6, we showed that if we use $\boldsymbol{\eta}_k(\boldsymbol{x}_t)$ defined by (2.10), then the decision boundary of the combined classifier is also linear. In this case we can write the equation of the decision boundary at time t as

$$\{\boldsymbol{x}_t : (A\boldsymbol{\beta}_t)^T \tilde{\boldsymbol{x}}_t = 0\},$$

where $\tilde{\boldsymbol{x}}_t^T = (1 : \boldsymbol{x}_t^T)$ and the matrix A is defined by (2.20). Then in this case, the equation of the decision boundary of the combined classifier is equal to the line with equation $\boldsymbol{b}^T \tilde{\boldsymbol{x}}_t = 0$ if

$$A\boldsymbol{\beta}_t = \boldsymbol{b}. \tag{3.1}$$

This implies that if there exists a solution to (3.1) for all $\boldsymbol{b}$, and the optimal decision boundary is linear, then the optimal classifier lies within the model space. There will exist such a solution if A is square (in which case $M = N+1$) and non-singular.

Note that A is singular if and only if either the decision boundaries of all the component classifiers are "parallel", or they all intersect at a single point. In the latter case, the decision boundary of the combined classifier will also pass through

the point of intersection. In fact, this is true for all definitions of $\boldsymbol{\eta}_k(\boldsymbol{x})$, regardless of whether the resulting decision boundary is linear or not. To see this, we can write the equation of the decision boundary of the combined classifier between classes j and j' at time t as

$$\{\boldsymbol{x}_t : \boldsymbol{\beta}_t^T(\boldsymbol{\eta}_j(\boldsymbol{x}_t) - \boldsymbol{\eta}_{j'}(\boldsymbol{x}_t)) = 0\}. \tag{3.2}$$

If all the decision boundaries of the component classifiers between classes j and j' pass though the point $\boldsymbol{x}_t^*$ say, then

$$\beta_{ti}(\eta_{ji}(\boldsymbol{x}_t^*) - \eta_{j'i}(\boldsymbol{x}_t^*)) = 0, \text{ for } i = 1, 2, \ldots, M.$$

Hence $\boldsymbol{\beta}_t^T(\boldsymbol{\eta}_j(\boldsymbol{x}_t^*) - \boldsymbol{\eta}_{j'}(\boldsymbol{x}_t^*)) = 0$, and so from (3.2) we can see that $\boldsymbol{x}_t^*$ also lies on the decision boundary of the combined classifier. Clearly this limits the flexibility of the decision boundary we can obtain by combining such component classifiers.

To summarise, for problems with an N-dimensional feature space we can represent any linear decision boundary by using $\boldsymbol{\eta}_k(\boldsymbol{x}_t)$ defined by (2.10) and $N+1$ LDA component classifiers whose decision boundaries are not parallel and do not pass through a common point. Hence if the optimal decision boundary is also linear, we need not combine more than $N + 1$ component classifiers.

3.2 Bounding the Decision Boundary

In this section we present our main results relating the decision boundary of the classifier (2.7) to the decision boundaries of the component classifiers.

Theorem 3.2.1. *When using a 0–1 loss function, $\boldsymbol{\eta}_k(\boldsymbol{x}_t)$ defined by (2.10) or (2.11) and non-negative parameter values $\boldsymbol{\beta}_t$, the decision boundary of the classifier (2.7) must lie in regions of the feature space where the component classifiers "disagree".*

Proof. Assuming 0–1 loss, the decision boundary of the ith component classifier between the jth and j'th classes is a subset of the set

$$\{\boldsymbol{x}_t : \hat{p}_i(j|\boldsymbol{x}_t) = \hat{p}_i(j'|\boldsymbol{x}_t)\}. \tag{3.3}$$

Define $\mathcal{R}_j^i$ as the region of the feature space in which the ith component classifier would classify an observation as class j. That is,

$$\mathcal{R}_j^i = \{\boldsymbol{x}_t : j = \text{argmax}_k\ \hat{p}_i(k|\boldsymbol{x}_t)\},\ j = 1, 2, \ldots, K.$$

Hence for all $\boldsymbol{x}_t \in \mathcal{R}_j^i$,

$$\hat{p}_i(j|\boldsymbol{x}_t) > \hat{p}_i(j'|\boldsymbol{x}_t) \text{ for } j \neq j'.$$

Define

$$\mathcal{R}_j^* = \cap_i \mathcal{R}_j^i.$$

Then for all i, for all $\boldsymbol{x}_t \in \mathcal{R}_j^*$,

$$\hat{p}_i(j|\boldsymbol{x}_t) > \hat{p}_i(j'|\boldsymbol{x}_t),\ j \neq j'. \tag{3.4}$$

From (3.4), for $\beta_{ti} \geq 0, i = 1, 2, \ldots, M$, it follows that for all $\boldsymbol{x}_t \in \mathcal{R}_j^*$,

$$\sum_{i=1}^{M} \beta_{ti}\{\hat{p}_i(j|\boldsymbol{x}_t) - \hat{p}_i(1|\boldsymbol{x}_t)\} > \sum_{i=1}^{M} \beta_{ti}\{\hat{p}_i(j'|\boldsymbol{x}_t) - \hat{p}_i(1|\boldsymbol{x}_t)\}. \tag{3.5}$$

Similarly, from (3.4), we can show that for $\beta_{ti} \geq 0, i = 1, 2, \ldots, M$, for $\boldsymbol{x}_t \in \mathcal{R}_j^*$,

$$\sum_{i=1}^{M} \beta_{ti} \log\left(\frac{\hat{p}_i(j|\boldsymbol{x}_t)}{\hat{p}_i(1|\boldsymbol{x}_t)}\right) > \sum_{i=1}^{M} \beta_{ti} \log\left(\frac{\hat{p}_i(j'|\boldsymbol{x}_t)}{\hat{p}_i(1|\boldsymbol{x}_t)}\right). \tag{3.6}$$

For the classifier (2.7), the decision boundary between the jth and j'th classes is contained in

$$\{\boldsymbol{x}_t : \boldsymbol{\beta}_t^T \boldsymbol{\eta}_j(\boldsymbol{x}_t) = \boldsymbol{\beta}_t^T \boldsymbol{\eta}_{j'}(\boldsymbol{x}_t)\}. \tag{3.7}$$

Therefore, for the definitions of $\boldsymbol{\eta}_j(\boldsymbol{x}_t)$ considered in section 2.5, we can see from (3.5) and (3.6) that $\mathcal{R}_j^*$ does not intersect with the set (3.7). That is, there is no point on the decision boundary of our final classifier that lies in the region where all component classifiers agree. □

From (3.4) it is easy to show that this result also holds for classifiers based on the weighted averaging model (2.1), in which case the result can be extended to any loss function.

As an example, figure 3.1 shows the decision boundaries of three component classifiers in a two-class problem with two-dimensional feature space. The shaded area represents the region of the feature space where not all of the component classifiers agree. Our result shows that the decision boundary of the combined classifier must lie within this shaded region.

Roughly speaking, this result shows we would like the decision boundaries of the component classifiers to straddle the Bayes boundary, at least in the regions of highest probability density. By "straddle", we mean that the decision boundaries of the component classifiers should be positioned such that the Bayes boundary is in the region of disagreement of the component classifiers. If this is the case, then for some value of the parameters it is likely that the decision boundary of the combined classifier will be closer[1] to the Bayes boundary than the decision boundary of the

[1]With respect to some region of the feature space, we mean that one decision boundary is "closer" to the Bayes boundary than another if the probability of an observation falling in the region of disagreement between that decision boundary and the Bayes boundary is smaller than the probability of an observation falling in the region of disagreement between the other decision boundary and the Bayes boundary.

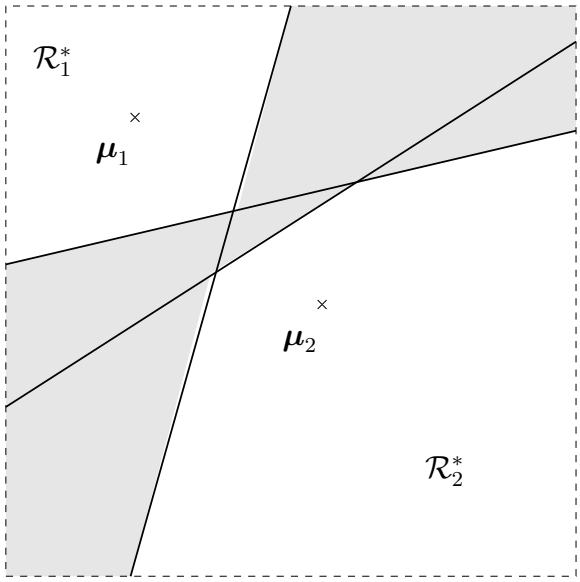


Figure 3.1: The decision boundaries of the component classifiers are shown in black, and the regions in which they disagree are shaded grey. $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ denote the means of classes one and two respectively. When using non-negative parameter values, the decision boundary of the combined classifier must lie within the shaded region.

best component classifier.

3.3 Diversity and Accuracy

There have been many attempts to infer a definitive relationship between the accuracy (or error rate) of a combined classifier and the diversity of the component classifiers. As mentioned in chapter 1, many empirical studies have demonstrated some kind of relationship between the two, with various authors using different definitions of diversity. In addition, the theoretical developments of Kleinberg (1990) and Tumer & Ghosh (1996) both mention the importance of some kind of diversity amongst the component classifiers. However, clearly the error rate of the final classifier depends both on the component classifiers being used and the method of combination. Therefore, it seems reasonable that one should consider the concept of diversity with respect to a particular method of combination and classification

rule.

In this section we consider the relevance of theorem 3.2.1 to the concept of diversity amongst the component classifiers. This discussion will be relevant to methods of combining for which that theorem holds, that is, for the classifier (2.7) as well as simple and weighted averaging. It is important to remember, however, that theorem 3.2.1 holds only when the parameters $\boldsymbol{\beta}_t > 0$. Firstly, we consider the implications of this result in the special case that the population is stationary, and then in the case that the population is dynamic.

3.3.1 Stationary Case

If the population is stationary, then the Bayes boundary does not change with time. Suppose that, due to the data available for training for example, the decision boundaries of the component classifiers differ from the Bayes boundary in a systematic fashion. Then the more similar the component classifiers are to one-another, the smaller the region of disagreement and hence the less likely it will be that the region of disagreement contains the Bayes boundary[2]. Therefore, if the decision boundaries of the component classifiers are systematically different to the Bayes boundary, it is best that they are as diverse (different to one another) as possible in order to maximise the region of disagreement and hence the probability that the region of disagreement contains the Bayes boundary. Because in this case the Bayes boundary is likely to be relatively close to the edge of the region of disagreement, it is likely that the minimum classification error is obtained when those component

[2]For the purposes of this discussion we only consider the regions of the feature space with non-negligible probability density. Hence, for example, when we say the region of disagreement contains the Bayes boundary, we mean that the region of disagreement contains the Bayes boundary in the regions of highest probability density. The location of the Bayes boundary with respect to the region of disagreement in other regions of the feature space is unlikely to have a significant impact on the error rate of the classifier.

classifiers with decision boundaries closest to the Bayes boundary are given the most weight i.e. have relatively large corresponding parameter values. If the Bayes boundary lies outside the region of disagreement then giving the other component classifiers substantial weight is only likely to result in the decision boundary of the combined classifier moving further from the Bayes boundary.

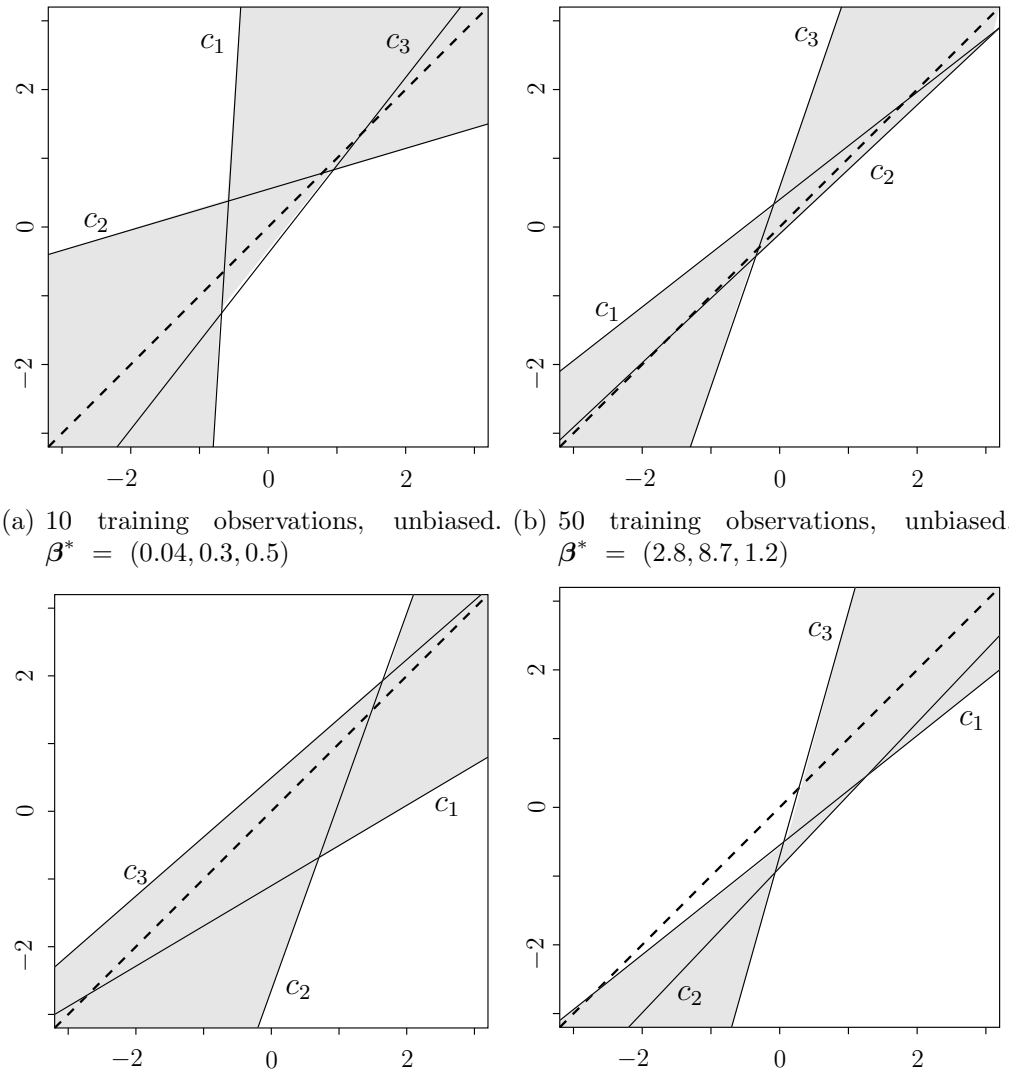
If the decision boundaries of the component classifiers are not systematically different to the Bayes boundary, then compared to the case that there is a systematic difference, less diversity amongst the component classifiers will be necessary for it to be as likely that the region of disagreement contains the Bayes boundary. Subject to the region of disagreement containing the Bayes boundary, shrinking this region can be seen as a sort of regularisation: as well as reducing the maximum possible error rate of the combined classifier, it should also be easier to estimate the optimal weights. This agrees with empirical findings which have shown that when the decision boundaries of the component classifiers are expected to lie "evenly" about the Bayes boundary, the more accurate the component classifiers are the lower the error rate of the combined classifier (see for example the results of Fumera & Roli, 2005).

To illustrate these points, consider a two-class problem with two-dimensional feature space, when the observations from the ith class are normally distributed with parameters $\boldsymbol{\mu}_i$ and Σ_i. Suppose $\boldsymbol{\mu}_1 = (-1, 1), \boldsymbol{\mu}_2 = (1, -1)$, $\Sigma_1 = \Sigma_2 = I$ (the identity matrix) and the probability $\pi_1 = \text{Prob}\{Y = 1\} = 0.5$. In each of figures 3.2(a) to 3.2(d), the Bayes boundary is shown (dashed line) along with the decision boundaries of three LDA component classifiers. In figures 3.2(a) and 3.2(b) the component classifiers have been trained on random samples from the population of size 10 and 50 respectively. Therefore in these cases the decision boundaries of the combined classifiers should not differ systematically from the Bayes boundary. The

set of component classifiers which are each trained on only 10 observations would be expected to be more diverse than the set of those trained on 50 observations, due to greater sampling variation in the training data. In figures 3.2(c) and 3.2(d) the component classifiers were again trained on 10 and 50 random observations respectively, except that in these cases the training samples were biased. This was done by drawing samples from the population with $\pi_1 = 0.8$ rather than the true probability $\pi_1 = 0.5$. Hence the decision boundaries of the component classifiers should differ systematically to the Bayes boundary. The optimal parameter values $\boldsymbol{\beta}^*$ are given in each case - these are the values for the parameters of the model such that the probabilities $\hat{p}(k|\boldsymbol{x};\boldsymbol{\beta})$ are equal to the true conditional class probabilities. In both figures 3.2(a) and 3.2(b) the region of disagreement contains the Bayes boundary and the optimal parameter values are non-negative. However, in figure 3.2(d) when the component classifiers are systematically different to the Bayes boundary and less diverse than in figure 3.2(c), the region of disagreement does not contain the Bayes boundary. Accordingly, one of the optimal parameter values in this case is negative. Hence the best error rate achievable in this case is sub-optimal, and greater than for the other three cases. Also note that the component classifiers with decision boundaries closest to the Bayes boundary generally have the largest corresponding value of β_i.

3.3.2 Dynamic Case

If the population of interest is dynamic, then in general the Bayes boundary will be changing over time. However, because our model uses the same set of component classifiers for all time points, the region of disagreement is fixed. Therefore, even if the Bayes boundary is initially contained within the region of disagreement, after some time this may cease to be the case. If the Bayes boundary moves outside the



(a) 10 training observations, unbiased. $\boldsymbol{\beta}^* = (0.04, 0.3, 0.5)$

(b) 50 training observations, unbiased. $\boldsymbol{\beta}^* = (2.8, 8.7, 1.2)$

(c) 10 training observations, biased. $\boldsymbol{\beta}^* = (0.10, 0.09, 0.41)$

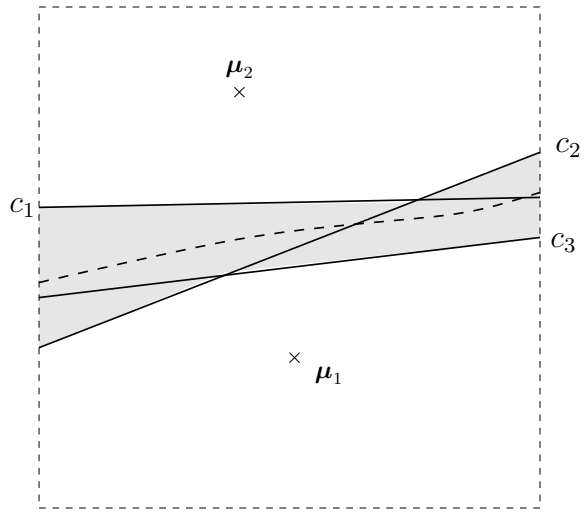
(d) 50 training observations, biased. $\boldsymbol{\beta}^* = (28.4, -30.0, 7.2)$

Figure 3.2: The Bayes boundary (dashed) and component classifier decision boundaries (solid) for the example discussed in section 3.3. The region of disagreement is shaded. $\boldsymbol{\beta}^*$ is the optimal parameter vector for the given set of component classifiers on this problem.

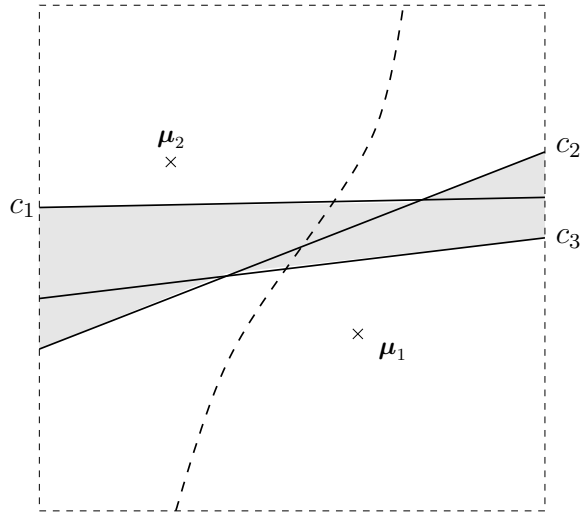
region of disagreement, then it is likely the performance of the classifier will deteriorate. Therefore, when selecting the component classifiers for a dynamic classifier it is important to ensure the region of disagreement is as large as possible, in order that it contains the Bayes boundary at all future time points. For example, figures 3.3(a) and 3.3(b) show the decision boundaries of three component classifiers on a two-class problem, and the region of disagreement for those classifiers. Initially, in figure 3.3(a), the Bayes boundary is located within the region of disagreement. However, at a later time the population means have drifted in such a way that the Bayes boundary has changed. In this case, shown in figure 3.3(b), the Bayes boundary is predominantly outside the region of disagreement. Hence although good classification performance should be possible for the situation of figure 3.3(a), because the decision boundary of the combined classifier must lie in the region of disagreement it is likely the best achievable classification accuracy of the combined classifier for the case of figure 3.3(b) would be poor.

3.3.3 On the Definition of Diversity

Consider defining the diversity of a set of classifiers as the volume of the "region of disagreement" discussed above, i.e. the integral over the feature space where at least two of the component classifiers disagree, where the integration is with respect to the probability distribution of features $\boldsymbol{X}$. Initially this may seem like a reasonable definition. However, it is easy to construct a counter example to its appropriateness. Consider a two-class problem and two classifiers c_1 and c_2 which are such that whenever one of the classifiers predicts class one, the other classifier will predict class two. Then according to the definition suggested above, the set of component classifiers $\{c_1, c_2\}$ is maximally diverse. This set is also maximally diverse according to the *difficulty* measure introduced by Kuncheva & Whitaker



(a) Initial scenario ($t = 1$)



(b) Later scenario ($t = 2$)

Figure 3.3: Decision boundaries of the component classifiers (black), the region of disagreement (shaded) and the Bayes boundary (dashed) of a dynamic classification scenario at two points in time.

(2003). However, if we use the classifier (2.7), for all values of the parameters β_{t1} and β_{t2} the final classifier will be equivalent to either c_1 or c_2 (this is proved in section 3.4). Thus although the region of disagreement is maximised, there is very little flexibility in the classifier as $\boldsymbol{\beta}_t$ varies.

The problem with considering the volume of the region of disagreement as a diversity measure is that this is a bound on the decision boundary of the combined classifier. Ideally, a measure of diversity would reflect the actual variation in decision boundaries that it is possible to obtain with a particular set of component classifiers and classification rule. However, the region of disagreement is still a useful concept for the design of dynamic classifiers. For a classifier to perform well on a dynamic scenario it is necessary that the region of disagreement is large enough to contain the Bayes boundary at all time points. We then need to consider the flexibility of the decision boundary within this region.

3.4 The Relationship to Negative Parameter Values

As discussed in section 3.3, when considering dynamic scenarios it is important to ensure the region of disagreement is sufficiently large. However, the bounds of section 3.2 hold only because we assume the parameters are constrained to be non-negative. In corollary 2.4.1 we showed that in the two-class case the effect of using negative parameter values was to reverse the labels of the component classifiers. Therefore, we can think of allowing $\beta_{ti} < 0$ as equivalent to using $\beta_{ti} > 0$ but with the labels of the ith component classifier reversed. For example, consider again the situation shown in figure 3.3(b). If we change β_{t2} to be negative, then to calculate the bound on the feature space in which the decision boundary of the combined

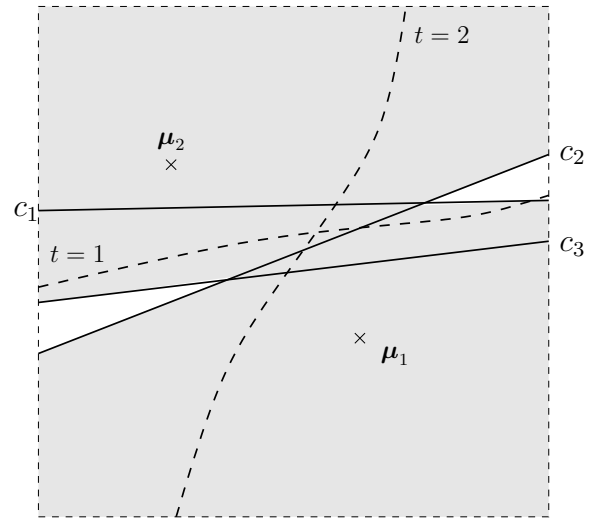


Figure 3.4: The decision boundaries of three component classifiers (black), the Bayes boundaries at times $t = 1$ and $t = 2$ (dashed) and the region of disagreement (shaded) when β_{t1} and β_{t3} are positive, and β_{t2} is negative.

classifier may lie (i.e. the region of disagreement), we simply swap the labels of the second component classifier. In the case that $\beta_{t1} > 0$, $\beta_{t3} > 0$ and $\beta_{t2} < 0$, the region of disagreement and the Bayes boundaries of figure 3.3 are shown in figure 3.4. If β_{t1}, β_{t2} and β_{t3} are all positive, the region of disagreement is as in figure 3.3(a). Therefore, using unconstrained parameter values is clearly a useful feature for dynamic problems, as the region of disagreement can be altered as the parameters change sign.

Alternatively, we may be able to achieve the same effect as allowing negative parameter values by constraining the parameters to be non-negative and including pairs of component classifiers which are such that they will always output the opposite labels, i.e. they disagree over the entire feature space. One way in which to produce such a pair of classifiers is to train two classifiers on the same data, except that the labels of the observations are reversed for the second classifier. Because of this, we will refer to such a pair of component classifiers as "label-swapped" classifiers.

In this section we show that using label-swapped component classifiers is equivalent to using unconstrained parameter values, and discuss the implications of this result. For reasons of simplicity we limit the analysis to two-class problems.

Suppose we combine M pairs of label-swapped classifiers so that we have $2M$ component classifiers in total. Then according to the classification rule (2.7) an observation $\boldsymbol{x}_t$ is classified to class 1 if $p(1|\boldsymbol{x}_t;\boldsymbol{\beta}_t) > p(2|\boldsymbol{x}_t;\boldsymbol{\beta}_t)$, i.e.

$$\sum_{i=1}^{2M} \beta_{ti}\eta_{2i}(\boldsymbol{x}_t) < 0. \tag{3.8}$$

Theorem 3.4.1. *Suppose $\eta_{2i}(\boldsymbol{x}_t) > 0$ if and only if $\hat{p}_i(2|\boldsymbol{x}_t) > \hat{p}_i(1|\boldsymbol{x}_t)$, and that*

$$\eta_{22}(\boldsymbol{x}_t) = -\eta_{21}(\boldsymbol{x}_t).$$

Then the classifier (2.7) obtained by using two label-swapped classifiers c_1 and c_2 and parameters β_{t1} and β_{t2} is equivalent to the classifier c_i, where $i = argmax_j\ \beta_{tj}$.

Proof. From (3.8) with $M = 1$, we see that $p(1|\boldsymbol{x}_t;\boldsymbol{\beta}_t) > p(2|\boldsymbol{x}_t;\boldsymbol{\beta}_t)$ whenever

$$(\beta_{t1} - \beta_{t2})\eta_{21}(\boldsymbol{x}_t) < 0. \tag{3.9}$$

Therefore, $p(1|\boldsymbol{x}_t;\boldsymbol{\beta}_t) > p(2|\boldsymbol{x}_t;\boldsymbol{\beta}_t)$ when either

$$\beta_{t1} < \beta_{t2} \text{ and } \eta_{21}(\boldsymbol{x}_t) > 0,$$
$$\text{or} \qquad \beta_{t1} > \beta_{t2} \text{ and } \eta_{21}(\boldsymbol{x}_t) < 0,$$

i.e. when

$$\begin{aligned} & \beta_{t1} < \beta_{t2} \text{ and } \hat{p}_2(1|\boldsymbol{x}_t) > \hat{p}_2(2|\boldsymbol{x}_t), \\ \text{or} \quad & \beta_{t1} > \beta_{t2} \text{ and } \hat{p}_1(1|\boldsymbol{x}_t) > \hat{p}_1(2|\boldsymbol{x}_t). \end{aligned}$$

So if $\beta_{t1} > \beta_{t2}$, the combined classifier is equivalent to using only c_1, and if $\beta_{t2} > \beta_{t1}$ the combined classifier is equivalent to using only c_2. □

Note that the conditions required by theorem 3.4.1 hold for the two definitions of $\eta_2(\boldsymbol{x}_t)$ recommended in section 2.5, namely

$$\begin{aligned} \eta_{2i}(\boldsymbol{x}_t) &= \log\left(\frac{\hat{p}_i(2|\boldsymbol{x}_t)}{\hat{p}_i(1|\boldsymbol{x}_t)}\right), \text{ and} \\ \eta_{2i}(\boldsymbol{x}_t) &= \hat{p}_i(2|\boldsymbol{x}_t) - \hat{p}_i(1|\boldsymbol{x}_t). \end{aligned}$$

This result demonstrates that if we use only one set of label-swapped component classifiers, the decision boundary of the combined classifier will be the same for all values of the parameters. If we change the parameter values, this can only alter the labelling of the regions, but not the location of the decision boundary.

Now suppose we combine M pairs of label-swapped classifiers and label them such that c_{2i} is the label-swapped partner of c_{2i-1}, for $i = 1, 2, \ldots, M$.

Theorem 3.4.2. *Using M pairs of label-swapped classifiers with parameters $\beta_{t1}, \beta_{t2}, \ldots, \beta_{t,2M}$ is equivalent to the model which uses only classifiers $c_1, c_3, \ldots, c_{2M-1}$ with parameters $\tilde{\beta}_{t1}, \tilde{\beta}_{t2}, \ldots, \tilde{\beta}_{tM}$, where*

$$\tilde{\beta}_{ti} \triangleq \beta_{t,2i-1} - \beta_{t,2i}.$$

Proof. From (3.8),

$$p(1|\boldsymbol{x}_t;\boldsymbol{\beta}_t) > p(2|\boldsymbol{x}_t;\boldsymbol{\beta}_t)$$

when

$$\sum_{i=1}^{2M} \beta_{ti}\eta_{2i}(\boldsymbol{x}_t) < 0,$$

i.e. when

$$\begin{aligned}(\beta_{t1} - \beta_{t2})\eta_{21}(\boldsymbol{x}_t) + (\beta_{t3} - \beta_{t4})\eta_{23}(\boldsymbol{x}_t)\\ + \ldots + (\beta_{t,2M-1} - \beta_{t,2M})\eta_{2(2M-1)}(\boldsymbol{x}_t) < 0\end{aligned}$$

i.e. when

$$\sum_{i=1}^{M} \tilde{\beta}_{ti}\eta_{2(2i-1)}(\boldsymbol{x}_t) < 0, \tag{3.10}$$

where

$$\tilde{\beta}_{ti} = \beta_{t,2i-1} - \beta_{t,2i}. \tag{3.11}$$

Comparing (3.10) with (3.8), we can see this is equivalent to the classifier which combines $c_1, c_3, \ldots, c_{2M-1}$ with parameters $\tilde{\beta}_{t1}, \tilde{\beta}_{t2}, \ldots, \tilde{\beta}_{tM}$, which are not necessarily greater than zero. □

Therefore, every combined classifier which uses label-swapped component classifiers and non-negative parameter values is equivalent to a classifier which uses only one component classifier from each label-swapped pair and unconstrained parameter values. Therefore, using label-swapped component classifiers should only be considered if the parameters are constrained to be non-negative. In this case, they can be used to increase the region of disagreement if the population is believed to be dynamic, or the component classifiers are not believed to be sufficiently diverse. It is also possible to include both label-swapped and non-label-swapped component

classifiers. This would then be equivalent to constraining some but not all of the parameters to be non-negative.

It is important to note that using label-swapped classifiers leads us to requiring twice as many component classifiers and hence parameters as the corresponding model with unconstrained parameters. In addition, the additional computational effort involved in enforcing the non-negativity constraint means that the label-swapped approach is significantly more computationally intensive than using standard unconstrained estimates. Because in practice we must estimate the parameter values, the performance of a classifier with label-swapped component classifiers may not be identical to the performance of the equivalent classifier for which the parameter estimates are unconstrained, due to the effect on parameter estimation of enforcing the non-negativity constraint.

3.5 Example

In this section we demonstrate the results from this chapter on an artificial dynamic classification problem. There exist two classes, class 1 and class 2, and observations from each class are distributed normally with a common covariance matrix. The probability that an observation is generated from class 1 is 0.7. At time $t = 0$ the mean of class 1 is $\boldsymbol{\mu}_1 = (1, 1)$, and the mean of class 2 is $\boldsymbol{\mu}_2 = (-1, -1)$. The mean of population 1 changes in equal increments from $(1, 1)$ to $(1, -4)$ over 1000 time steps, so that at time t, $\boldsymbol{\mu}_1 = (1, 1 - 0.005t)$.

For the following simulations we used three LDA component classifiers, each of which was trained on an independent random sample of 10 observations from the population at time $t = 0$. Figure 3.5 shows the Bayes boundary at times $t = 0$, $t = 500$ and time $t = 1000$, along with the decision boundaries of the

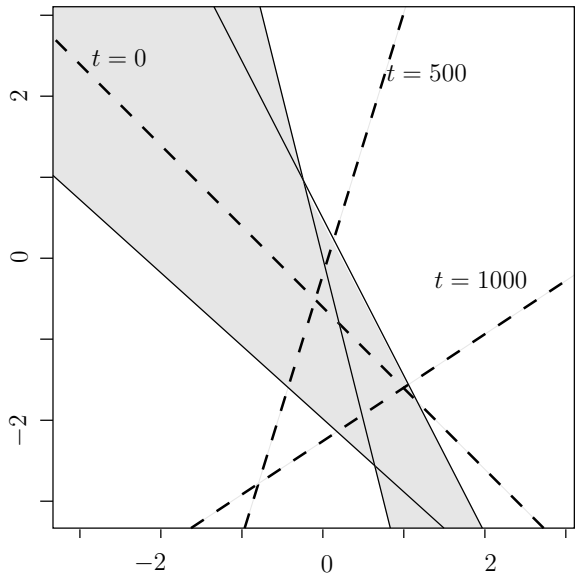


Figure 3.5: The decision boundaries of the component classifiers (solid) and the Bayes boundary (dashed) at times $t = 0$, $t = 500$ and $t = 1000$. The region of disagreement of the component classifiers is shaded grey.

three component classifiers. We choose to use $\boldsymbol{\eta}_k(\boldsymbol{x}_t)$ defined by (2.10), so for this example both the Bayes boundary and the decision boundary of the combined classifier are linear. In addition, from the result of section 3.1.2, we know that if using unconstrained parameter estimates then at all times there will exist a value for the parameters such that the error rate of the combined classifier is equal to the Bayes error.

We used a particle filter with 300 particles to update the parameter estimates. The details of the algorithm are given in chapter 4. The model for parameter evolution used was

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t + \boldsymbol{\omega}_t, \tag{3.12}$$

where $\boldsymbol{\omega}_t$ has a normal distribution with mean $\mathbf{0}$ and covariance matrix equal to 0.005 times the identity matrix. An observation $\boldsymbol{x}_t$ was classified as belonging to

class k if

$$k = \text{argmax}_j \ \hat{E}_{\boldsymbol{\beta}_t}[\hat{p}(j|\boldsymbol{x}_t; \boldsymbol{\beta}_t)]. \tag{3.13}$$

Denote by Err_{-i} the error rate of the ith component classifier on the training data of the other component classifiers, for $i = 1, 2, 3$. The value of β_{0i} was chosen to be proportional to $1 - \text{Err}_{-i}$ for $i = 1, 2, 3$. Each simulation involved repeating the data generation, classification and updating procedure 100 times, and the errors of each run were averaged to produce an estimate of the error rate of the classifier at every time t.

We repeated the simulation three times. In the first case, we constrained the parameters $\boldsymbol{\beta}_t$ to be non-negative. A smooth of the estimated error rate is shown in figure 3.6(a), along with the Bayes error (grey line) and the error of the the component classifier corresponding to the smallest value of Err_{-i} (included to demonstrate the deterioration in performance of the "best" component classifier at time $t = 0$, dashed line). The error rate of the classifier is close to the Bayes error for the first 200 updates, but then the performance deteriorates. After $t = 200$, the Bayes boundary has moved enough that it can no longer be well approximated by a linear decision boundary lying in the region of disagreement.

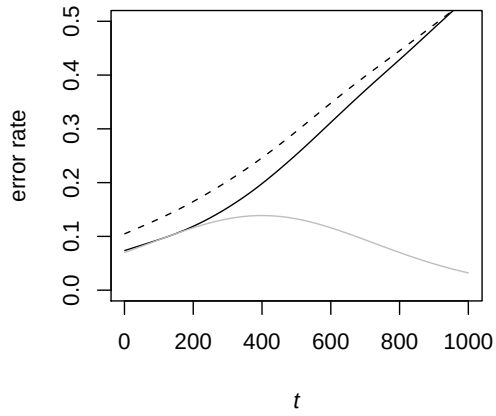
In the second case, we use the same three classifiers as above, but include their label-swapped pairs. We hence have six component classifiers in total. A smooth of the estimated error rate is shown in figure 3.6(b), and it is clear that this classifier does not succumb to the same level of decreased performance as seen in figure 3.6(a). However, for the first 200 updates the error rate of the classifier is slightly higher than that in figure 3.6(a). This could be because using label-swapped pairs increases the region of disagreement and hence there is less regularisation.

Thirdly, we used the original set of three component classifiers but without

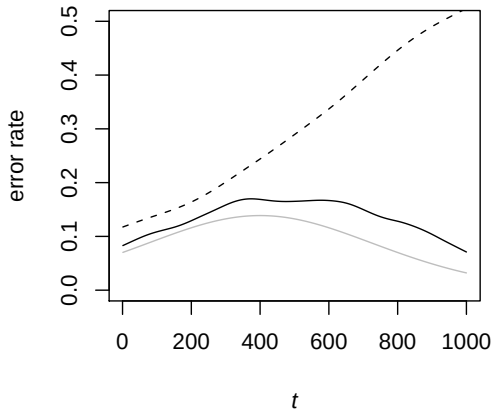
constraining the parameter values to be non-negative. The resulting error rate, shown in figure 3.6(c), is very similar to that using label-swapped classifiers. In this case the average error curve is slightly more smooth than in figure 3.6(b), possibly due to the difference in the number of parameters which are unknown.

In figures 3.7(a) and 3.7(b) we show the average expected parameter values returned by the updating algorithm when using unconstrained parameters and label-swapped component classifiers with non-negative parameters respectively. In figure 3.7(c) we show the values of $\tilde{\boldsymbol{\beta}}_{ti}$ (defined on page 52), which can be seen to be very similar to the unconstrained parameter values in figure 3.7(a). This explains the similarity of classification performance between the label-swapped and unconstrained cases. Furthermore, we can see from figure 3.7(a) that when using unconstrained parameter estimates negative parameter values are used after about 300 updates. This requirement for negative parameter values explains the deterioration in performance of the classifier using non-negative parameter values seen in figure 3.7(a).

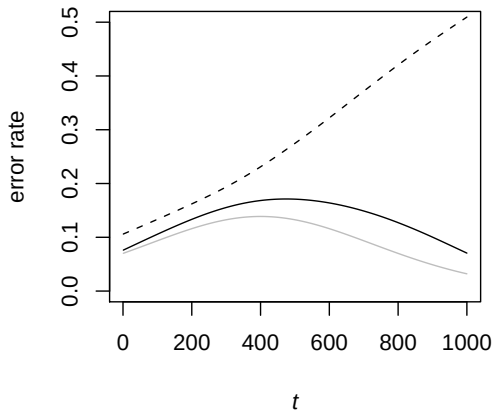
This example demonstrates that it is important to consider the region of disagreement in dynamic classification problems. Furthermore, for this simple case we found no difference in performance of the classifier using label-swapped component classifiers with non-negative parameter values, and the classifier using unconstrained parameter estimates. However, for the first 200 time-steps whilst the Bayes boundary was within the region of disagreement of the component classifiers, the method using non-negative parameter estimates and no label-swapped classifiers slightly out-performed the other two methods.



(a) Non-negative parameter values.

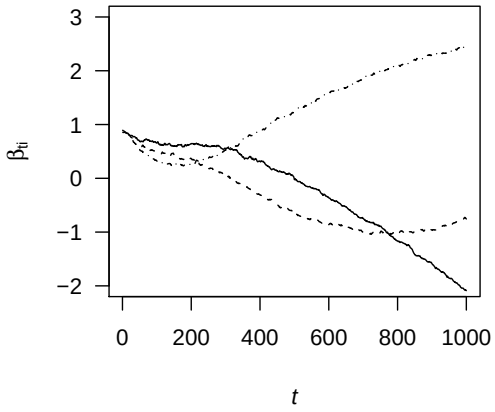


(b) Label-swapped classifiers.

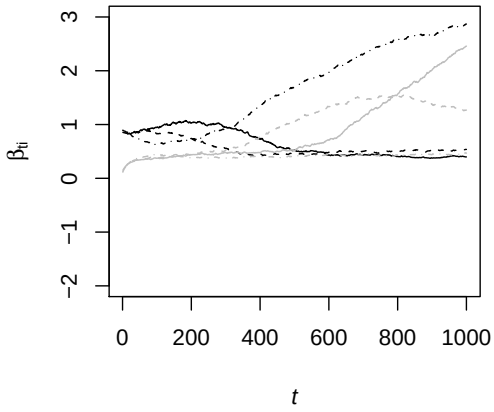


(c) Unconstrained parameter values.

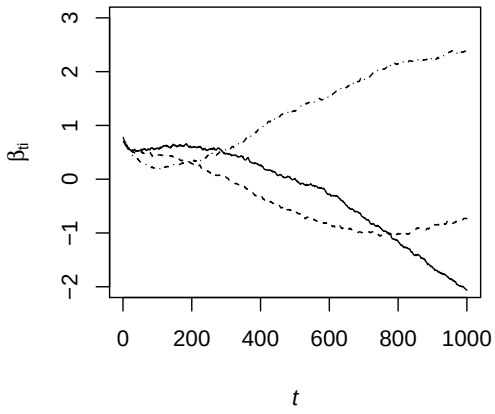
Figure 3.6: Smoothed estimated error rates (solid black line) for the three scenarios described, along with the Bayes risk (grey) and smoothed estimated error rate of the component classifier with the lowest estimated error rate at time $t = 0$ (dashed black line).



(a) Unconstrained parameter values.



(b) Label-swapped classifiers: $\boldsymbol{\beta}_t$.



(c) Label-swapped classifiers: $\tilde{\boldsymbol{\beta}}_t$.

Figure 3.7: Average expected parameter values β_{it}, for $i = 1$ (solid line), $i = 2$ (dashed) and $i = 3$ (dot-dashed). In figure 3.7(b) the grey and black lines of each type correspond to a label-swapped pair.

3.6 Summary

The results of this chapter can be summarised with respect to two cases:

1. the population of interest is believed to be stationary

2. the population of interest is believed to be dynamic.

In the first case, assuming the decision boundaries of the component classifiers are not systematically different to the Bayes boundaries, the result of section 3.2 implies that we should use component classifiers that are as accurate as possible, and constrain the parameters to be non-negative. The region of disagreement should then contain the Bayes boundary, and be relatively small. Minimising the region of disagreement about the Bayes boundary has the advantage of limiting the space of possible decision boundaries. For this reason, using the non-negativity constraint can be seen as a type of regularisation; as long as the Bayes boundary still remains within the region of disagreement, it is likely we would obtain a better combined classifier than if using unconstrained estimates. This corresponds to the empirical findings of Breiman (1993) and LeBlanc & Tibshirani (1996).

If the decision boundaries of the component classifiers may be systematically different to the Bayes boundary, then we should use relatively diverse component classifiers in conjunction with non-negative parameter values in order to increase the probability that the Bayes boundary lies within the region of disagreement. If it is considered likely that the region of disagreement will still be too small, we could allow unconstrained parameter estimates or include some pairs of label-swapped classifiers.

If the population is assumed to be dynamic, then in general we should use unconstrained parameter estimates. However, if the changes are expected to be

small, it may be better to use non-negative parameter estimates but with diverse component classifiers to ensure that the region of disagreement is sufficiently large.

If we have a reasonable amount of training data available, one way to influence the diversity of the component classifiers is to change the size of the training sets. In general the smaller the training sets used, the more diverse the component classifiers will be, due to increased sample variation. According to the reasoning in section 3.1, we should use as many component classifiers as is computationally feasible. This should also increase the flexibility of the decision boundary within the region of disagreement.

If we restrict ourselves to combined classifiers with linear decision boundaries, then we do not need any training data from the population in order to obtain suitable component classifiers. As shown in section 3.1.2, we know that if we use almost any $N + 1$ component classifiers for an N dimensional problem, then with that set of component classifiers we can represent any linear decision boundary. New observations can be used to update the parameters over time. However, if the population is not expected to change much, better performance may result from using non-negative parameter estimates, and an appropriate method of selecting the component classifiers as discussed above, as this will reduce the size of the region of disagreement.

The above comments are only guidelines, based on a study of the location of the combined decision boundary in relation to the decision boundaries of the component classifiers. Ultimately, the performance of the combined classifier will depend heavily on the algorithm used to implement the classifier, and its ability to accurately estimate the optimal values for the parameters. We consider algorithms for implementation in chapter 4.

Chapter 4

Implementation

In previous chapters we have explored the properties of the classification model introduced in chapter 2. This was important in order to understand the model and the situations to which its application may be appropriate. In addition, the work presented in chapter 3 helps to understand the influence of the component classifiers which are used, and shows how the component classifiers can affect the performance of the combined classifier. In this chapter, we discuss how one might develop a classification algorithm to implement this model. By "classification algorithm", we simply mean an algorithm which when given an input $\boldsymbol{x}_t$ outputs a label $\hat{y}_t = c(\boldsymbol{x}_t)$. Although there are several issues to consider before finally implementing the model, such as as which component classifiers to use and the form of the functions $\boldsymbol{\eta}_k(\boldsymbol{x}_t)$, the main issue to be addressed is how to deal with the unknown parameters $\boldsymbol{\beta}_t$. Rather than use "plug-in" estimates of the parameter values, the algorithms we consider are based on the predictive approach to classification, which we believe to be a better suited approach to this problem. In this chapter we discuss and compare three algorithms which could be used for implementing our model. Two of the algorithms we consider make assumptions about the form of the

prior distribution of the parameters $\boldsymbol{\beta}_t$. The first of these assumes the distribution is Gaussian and the second is based on a slightly less strict approach developed for dynamic generalised linear models. The third algorithm we consider allows the distribution of $\boldsymbol{\beta}_t$ to take any form, but uses sequential Monte Carlo methods to approximate this distribution.

For many classification problems there will be a classification algorithm which will perform very well on that particular problem, but perform relatively poorly on others. Due to the form of the random walk model for parameter evolution (equation (2.5), page 18), our model is best suited for classification in the presence of steady changes in the population. Therefore, this is the scenario of population change on which we focus our comparisons. We also compare the performance of the algorithms when the population is stationary. Hence our analysis is relevant to performance of the algorithms on populations which may have periods of change, interspersed with periods of stationarity.

The analysis of this chapter is by no means exhaustive, and it does not answer all questions about the performance we can expect from implementations of this model on a dynamic classification problem. It does, however, give an idea of the sorts of issues which are important to consider in implementation of models of this type, and will hopefully serve as a starting point for more in-depth analysis of the performance of algorithms designed for dynamic classification.

We begin with an outline of the predictive approach to classification, and then introduce the scenarios of population change which will be used for all simulations. We then introduce the three algorithms we will consider, and compare their performance.

4.1 The Predictive Approach to Classification

As discussed in section 2.4, due to the difficulty of converting prior knowledge about the population to knowledge about the optimal parameter values there will be quite a lot of uncertainty associated with the parameter values at any time. This uncertainty is enhanced by the assumption that the population is dynamic, which has the consequence that the optimal parameters will be changing (or "evolving") over time. One method of classification is to estimate the parameter values directly, and plug these estimates into the classification rule (2.7). However, for classification our primary concern is with estimating the conditional class probabilities, rather than the parameters themselves. The predictive approach to classification produces estimates of the conditional class distribution and deals with uncertainty about the parameters by averaging over all possible values. We therefore adopt this approach, which can be described as follows:

Suppose that at time $t-1$ our knowledge about the parameters $\boldsymbol{\beta}_{t-1}$ is summarised by the posterior distribution $p(\boldsymbol{\beta}_{t-1}|\boldsymbol{z}_{1:t-1})$, where we use the notation

$$\boldsymbol{z}_{1:t-1} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_{t-1}, y_{t-1})\}.$$

After observing $\boldsymbol{X}_t$, we are interested in the probabilities

$$\begin{aligned} \tilde{p}(Y_t = k|\boldsymbol{x}_t, \boldsymbol{z}_{1:t-1}) &= \int_{\boldsymbol{\beta}_t} p(Y_t = k, \boldsymbol{\beta}_t|\boldsymbol{x}_t, \boldsymbol{z}_{1:t-1}) d\boldsymbol{\beta}_t \\ &= \int_{\boldsymbol{\beta}_t} p(Y_t = k|\boldsymbol{x}_t, \boldsymbol{\beta}_t) p(\boldsymbol{\beta}_t|\boldsymbol{z}_{1:t-1}) d\boldsymbol{\beta}_t, \end{aligned} \tag{4.1}$$

where $p(\boldsymbol{\beta}_t|\boldsymbol{z}_{1:t-1})$ is given by

$$p(\boldsymbol{\beta}_t|\boldsymbol{z}_{1:t-1}) = \int_{\boldsymbol{\beta}_{t-1}} p(\boldsymbol{\beta}_t|\boldsymbol{\beta}_{t-1})p(\boldsymbol{\beta}_{t-1}|\boldsymbol{z}_{1:t-1})d\boldsymbol{\beta}_{t-1}. \tag{4.2}$$

We will use $\tilde{p}(k|\boldsymbol{x}_t)$ to denote $\tilde{p}(Y_t = k|\boldsymbol{x}_t, \boldsymbol{z}_{1:t-1})$. The probabilities $p(Y_t = k|\boldsymbol{x}_t, \boldsymbol{\beta}_t)$ and $p(\boldsymbol{\beta}_t|\boldsymbol{\beta}_{t-1})$ on the right hand sides of (4.1) and (4.2) are specified by the form of the model through (2.4) and (2.5) respectively, and the density $p(\boldsymbol{\beta}_{t-1}|\boldsymbol{z}_{1:t-1})$ is assumed known from time $t-1$. Hence in theory we have all the information needed to compute (4.1) and the probabilities $\tilde{p}(k|\boldsymbol{x}_t), k = 1, 2, \ldots, K$. Having calculated these probabilities, we classify $\boldsymbol{x}_t$ to the class with the largest value of $\tilde{p}(k|\boldsymbol{x}_t)$, so

$$\hat{c}(\boldsymbol{x}_t) = \text{argmax}_k \ \tilde{p}(k|\boldsymbol{x}_t).$$

Suppose that we then observe the true label y_t. We can update the distribution of $\boldsymbol{\beta}_t$ according to the relationship

$$p(\boldsymbol{\beta}_t|\boldsymbol{z}_{1:t}) \propto p(y_t|\boldsymbol{x}_t, \boldsymbol{\beta}_t)p(\boldsymbol{\beta}_t|\boldsymbol{z}_{1:t-1}),$$

where $p(\boldsymbol{\beta}_t|\boldsymbol{z}_{1:t-1})$ is given by (4.2). This classification and updating process can then be repeated for each new observation.

In general, it will not be possible to evaluate the integrals in (4.1) and (4.2) exactly. The exception to this is if the distribution of $\boldsymbol{\beta}_t|\boldsymbol{z}_{1:t}$ is assumed to be Gaussian, which is the case we consider in section 4.4.1. The approach of section 4.4.2 relaxes the strict assumptions of normality, but still assumes the posterior distribution to be unimodal. In section 4.4.3 we consider a sequential Monte Carlo algorithm based on a discrete approximation to the distribution of $\boldsymbol{\beta}_t|\boldsymbol{z}_{1:t}$.

4.2 Specifying the Parameter Evolution Variance

Before we discuss the classification algorithms it is important first to consider the role played by the parameter evolution variance V_t. As given in section 2.2, the model of parameter evolution is given by

$$\boldsymbol{\beta}_t = \boldsymbol{\beta}_{t-1} + \boldsymbol{\omega}_t,\ \boldsymbol{\omega}_t \sim N(\mathbf{0}, V_t). \tag{4.3}$$

Because V_t specifies the distribution of $\boldsymbol{\beta}_t|\boldsymbol{\beta}_{t-1}$, we will refer to its components as the hyper-parameters of the model. If the components of V_t are non-zero, then it is clear that the assumed parameter evolution adds to the uncertainty of the parameter value $\boldsymbol{\beta}_t$ when compared to the case that the optimal parameters are assumed not to change over time. Unlike for stationary problems, even if we knew the optimal value of $\boldsymbol{\beta}_{t-1}$ then under (4.3) we still have uncertainty about the value of $\boldsymbol{\beta}_t$. By specifying the value of V_t we are expressing how much information the current parameter value gives us about the parameter value at the time when we receive the next observation. In this sense, the value of V_t can be considered as specifying the degree of "memory" of the process, i.e. for how long past observations are still informative about the present. Specifying a value for V_t therefore plays a similar role to choosing the size of the window for a moving-window classifier. If the population is stationary for a period of time, then optimally the value of V_t for that time period should be zero as all the observations received during that period are equally as informative about the current value of the parameters. However, if the population is not stationary and V_t is zero, then we take into account misleading information about the current situation from past observations. This can result in the classifier being slow to respond to changes in the population.

The value of V_t can have a large influence on the performance of the classifier.

The importance of specifying V_t has long been recognised, and many researchers have suggested ways to estimate this hyper-parameter. West & Harrison (1997, ch 6.3) note a number of drawbacks with the model of parameter evolution (4.3), and instead advocate a reformulation in terms of discount factors (Ameen & Harrison, 1985). Other approaches to specification of V_t are given in Penny & Roberts (1999) and de Freitas et al. (2000a), who consider V_t in the context of an on-line regularisation parameter.

Often, in order to simplify the problem, it is assumed that

$$V_t = v_t I, \tag{4.4}$$

where I is the $M \times M$ identity matrix and v_t is a scalar (see for example de Freitas et al. (2000a); Penny & Roberts (1999); Højen-Sørensen et al. (2000)). In this case the problem of specifying the variance-covariance matrix V_t reduces to that of estimating the scalar v_t.

Before we adopt this convention, we should consider the implications of the assumptions made. Firstly, by assuming that V_t is diagonal, we assume that the noise components $\omega_1, \omega_2, \ldots, \omega_M$ are independent. Consider using two component classifiers which are very similar. In this case, we would expect the optimal changes in parameter values corresponding to those component classifiers to be similar as well. By assuming independence we introduce more variation than is necessary. In general, however, this assumption seems reasonable, particularly if we avoid using component classifiers which are very similar. By assuming V_t is diagonal when the noise components are not truly independent we introduce extra variation, but on the other hand we have fewer parameters to estimate.

The other assumption made by adopting form (4.4) is that all diagonal compo-

nents of V_t are equal. Intuitively, the value of V_t on the ith diagonal is related to the amount by which it is expected β_{ti} will change from one time to the next. To see this, consider the probabilities

$$p(\beta_{t+1,i}|\beta_{ti}) = \frac{1}{\sqrt{2\pi V_{ti}}} \exp\{-\frac{1}{2V_{ti}}\delta_i^2\}, i = 1, 2, \ldots, M \tag{4.5}$$

specified by the model for parameter evolution (4.3), where $\delta_i^2 = (\beta_{t+1,i} - \beta_{ti})^2$, and V_{ti} denotes the ith component of the diagonal of V_t. The value of V_{ti} which maximises equation (4.5) for each i is

$$V_{ti} = \delta_{ti}^2 = (\beta_{t+1,i} - \beta_{ti})^2.$$

Therefore, V_{ti} should increase as the squared difference δ_{ti}^2 increases. Depending on which component classifiers are used, the values of δ_{ti}^2 for $i = 1, 2, \ldots, M$ could be quite different. By overestimating V_{ti} we increase the variance introduced by parameter evolution, whereas underestimation can result in the estimate of $\beta_{t+1,i}$ being biased towards β_{ti}.

In general, it is hard to assess the impact of assuming the form (4.4) for V_t without knowing the relationship between the optimal parameter values and the component classifiers. However, in all simulations performed in this chapter we adopt this assumption for simplicity. We hence have M unknown parameters $\beta_{1t}, \beta_{2t}, \ldots, \beta_{Mt}$ and one hyper-parameter v_t.

4.3 Simulation Methodology

All the simulations we carry out to compare the classification algorithms are performed on artificial data, in order that an estimate of the true classifier error rate at

Classifier 1	
$\boldsymbol{x}$	y
(0,0)	1
(-1.4,1.4)	1
(1.4,1.4)	1
(0,3)	2
(-1.4,1.6)	2
(1.4,1.6)	2

Classifier 2	
$\boldsymbol{x}$	y
(0,0)	1
(0,-2)	1
(2,0)	1
(2.2,-0.2)	2
(3.2,-2.2)	2
(-1.2,-2.2)	2

Classifier 3	
$\boldsymbol{x}$	y
(-2,0)	1
(0,0)	1
(0,-2)	1
(-2.2,-0.2)	2
(-3.2,-2.2)	2
(-1.2,-2.2)	2

Table 4.1: Training data used to train each of the LDA component classifiers.

all time points can be obtained. We use two scenarios of population change, in each case with two classes each normally distributed with common covariance matrix. The mean of the kth class at time t is denoted $\boldsymbol{\mu}_{kt}$, and the covariance by Σ_t. The prior probability that at time t an observation is generated from the kth class is denoted π_{kt}.

In all the simulations we adopt the form of $\boldsymbol{\eta}_k(\boldsymbol{x})$ given by (2.10). As shown in section 3.1.2, we know that in such scenarios we need only three component classifiers with linear boundaries in order that there exists a value of $\boldsymbol{\beta}_t$ such that the resulting decision boundary of the classifier is optimal. We use three LDA classifiers trained on the data shown in table 4.1, which results in the component classifiers with decision boundaries shown in figure 4.1. By adopting this framework it is possible to calculate the optimal parameter values, which we can then compare to information output by the algorithms about the parameters.

The two scenarios we consider are as follows:

1. **No change**

   In this scenario the population is stationary. $\boldsymbol{\mu}_{1t} = (-1, 1)$, $\boldsymbol{\mu}_{2t} = (1, 1)$, $\pi_{1t} = 0.7$ and $\Sigma_t = I$.

2. **Slow change**

   In this scenario, the parameters of the population change steadily over time.

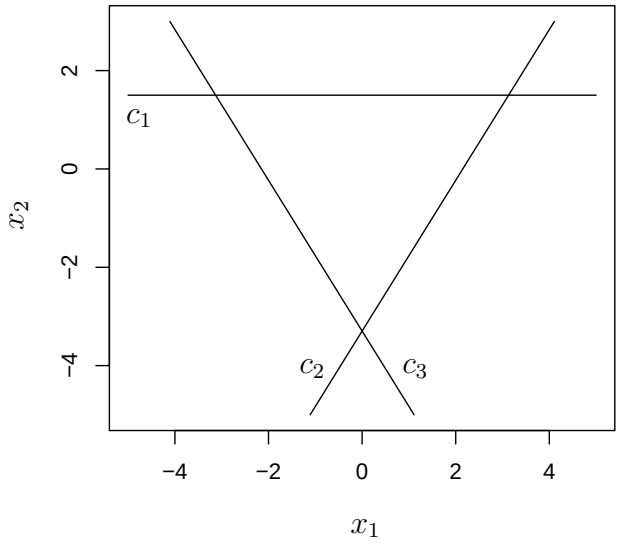
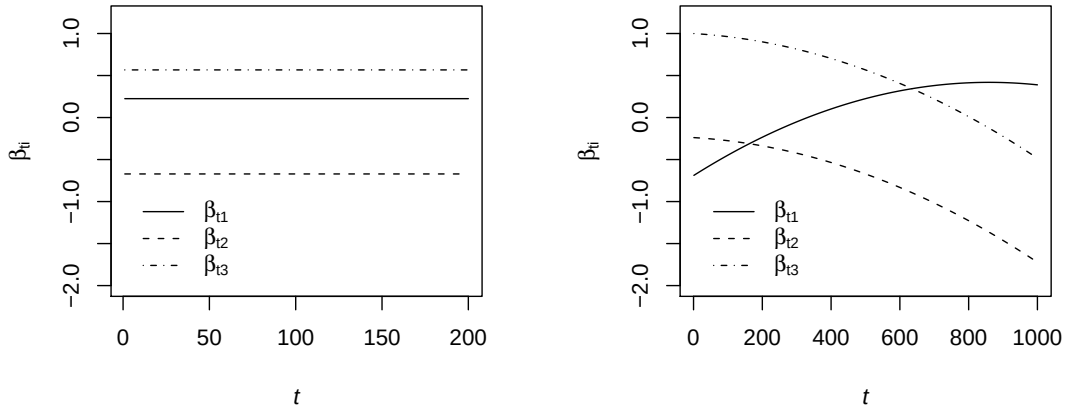


Figure 4.1: The decision boundaries of the three component classifiers which are used for all simulations.

The speed of the change is controlled by how many observations are received between time $t = 0$ and time $t = T$: $\boldsymbol{\mu}_{1t} = (-1, 1), \pi_{1t} = 0.7$ and $\Sigma_t = I$ for all t. $\boldsymbol{\mu}_2$ changes steadily over time from $\boldsymbol{\mu}_{20} = (1, 1)$ to $\boldsymbol{\mu}_{2T} = (1, -3)$. We consider $T \in \{200, 1000\}$.

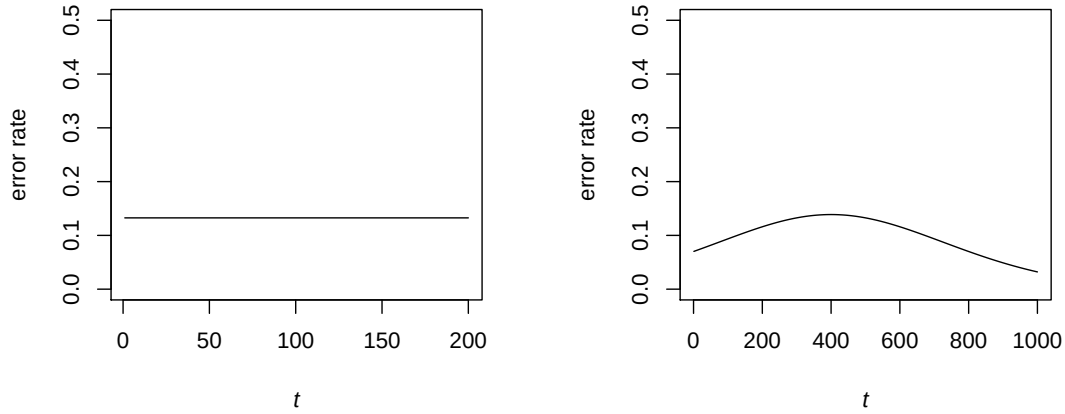
The optimal parameter values when using the component classifiers described above are shown for each of the two scenarios in figure 4.2. The Bayes error rates are shown in figure 4.3.

To obtain an estimate of the error rate of a classification algorithm, we run the algorithm on each scenario 200 times. On each run, for times $t = 1, 2, \ldots, T$, an observation is randomly generated from the population at that time. It is then classified by the algorithm, the true label of the observation is revealed and the parameter distributions updated before the next observation is received. The estimated error rate of an algorithm at time t is the proportion of runs on which



(a) No change (b) Slow change

Figure 4.2: Optimal parameter values for the two scenarios used in simulations, when using the component classifiers described.



(a) No change (b) Slow change

Figure 4.3: Bayes risk for the two scenarios used in simulations.

the observation generated at time t was incorrectly classified by that algorithm. Because we expect the true error rate of the classification algorithms to be around 0.2 or less and we are performing 200 runs, we can approximate the standard error of the estimated error rate at any time by 0.025. Hence two standard errors will be approximately 0.05. To make the simulation output clearer, we plot a smooth of the estimated error rate. This was obtained by fitting a smoothing spline to the average error using the R function smooth.spline() with the number of knots set to five for the scenarios with $T = 200$ and ten for the dynamic scenario with $T = 1000$.

4.4 Updating Algorithms

We now describe the algorithms we will use for estimating the probabilities

$$\tilde{p}(Y_t = k|\boldsymbol{x}_t) = \int_{\boldsymbol{\beta}_t} p(Y_t = k|\boldsymbol{x}_t, \boldsymbol{\beta}_t) p(\boldsymbol{\beta}_t|\boldsymbol{z}_{1:t-1}) d\boldsymbol{\beta}_t, \;\; k = 1, 2, \ldots, K \tag{4.6}$$

as given in (4.1). Because these estimates are based on updating the distribution of $\boldsymbol{\beta}_t|\boldsymbol{z}_{1:t}$ after every new observation is received, we refer to the algorithms as "updating algorithms".

The first two algorithms we consider, described in sections 4.4.1 and 4.4.2, are based on making assumptions about the form of the distribution of $\boldsymbol{\beta}_t|\boldsymbol{\beta}_{t-1}$ which simplify the updating. In section 4.4.3 we use a finite number of weighted points to approximate the distribution of $\boldsymbol{\beta}_t|\boldsymbol{\beta}_{t-1}$, but make no assumptions about the form of this distribution. Because the examples we consider in this chapter consist of a population with only two classes, for simplicity of exposition we present the algorithms for this case only.

4.4.1 Gaussian Approximation

As the name suggests, in this case we assume that $\boldsymbol{\beta}_{t-1}|\boldsymbol{z}_{1:t-1}$ has a Gaussian distribution. Specifically, suppose

$$\boldsymbol{\beta}_{t-1}|\boldsymbol{z}_{1:t-1} \sim N(\boldsymbol{m}_{t-1}, C_{t-1}).$$

Then under the model of parameter evolution (4.3),

$$\boldsymbol{\beta}_t|\boldsymbol{z}_{1:t-1} \sim N(\boldsymbol{a}_t, R_t), \tag{4.7}$$

where

$$\boldsymbol{a}_t = \boldsymbol{m}_{t-1},\ R_t = C_{t-1} + V_t. \tag{4.8}$$

As we only consider the two-class case, the quantity we wish to approximate can be written

$$\begin{aligned} \tilde{p}(2|\boldsymbol{x}_t) &= E_{\boldsymbol{\beta}_t|\boldsymbol{z}_{1:t-1}}[p(2|\boldsymbol{x}_t, \boldsymbol{\beta}_t)] \\ &= E_{\boldsymbol{\beta}_t|\boldsymbol{z}_{1:t-1}}\left[\frac{\exp[\lambda_t]}{1+\exp[\lambda_t]}\right], \end{aligned} \tag{4.9}$$

where λ_t is defined as

$$\lambda_t = \boldsymbol{\beta}_t^T \boldsymbol{\eta}_t, \tag{4.10}$$

and we once again use $\boldsymbol{\eta}_t$ to denote $\boldsymbol{\eta}_2(\boldsymbol{x}_t)$. From (4.7),

$$\lambda_t|\boldsymbol{z}_{1:t-1}, \boldsymbol{x}_t \sim N(\boldsymbol{a}_t^T \boldsymbol{\eta}_t, \boldsymbol{\eta}_t^T R_t \boldsymbol{\eta}_t).$$

However, if we simply plug $E[\lambda_t|\boldsymbol{z}_{1:t-1}, \boldsymbol{x}_t] = \boldsymbol{a}_t^T \boldsymbol{\eta}_t$ into (4.9), the resulting estimate of $\tilde{p}(2|\boldsymbol{x}_t)$ will be biased. Spiegelhalter & Lauritzen (1990) suggest the approxima-

tion

$$\hat{p}(2|\boldsymbol{x}_t) = \frac{\exp[c\ \boldsymbol{a}_t^T \boldsymbol{\eta}_t]}{1+\exp[c\ \boldsymbol{a}_t^T \boldsymbol{\eta}_t]}, \tag{4.11}$$

where $c = (1+0.368\ \boldsymbol{\eta}_t^T R_t \boldsymbol{\eta}_t)^{-1/2}$. We adopt this approximation[1], together with the dynamic version of the updating rules given in Spiegelhalter & Lauritzen (1990), as used by Penny & Roberts (1999). This gives the following algorithm:

Updating Algorithm (Gaussian - known V_t)

1. Suppose

$$\boldsymbol{\beta}_{t-1}|\boldsymbol{z}_{1:t-1} \sim N(\boldsymbol{m}_{t-1}, C_{t-1}).$$

2. After observing $\boldsymbol{x}_t$, calculate $\boldsymbol{a}_t$ and R_t from (4.8), and

$$c = (1+0.368\ \boldsymbol{\eta}_t^T R_t \boldsymbol{\eta}_t)^{-1/2}.$$

   Output $\hat{c}(\boldsymbol{x}_t) = 2$ if

$$\hat{p}(2|\boldsymbol{\beta}_{t-1}, \boldsymbol{x}_t) = \frac{\exp[c\ \boldsymbol{a}_t^T \boldsymbol{\eta}_t]}{1+\exp[c\ \boldsymbol{a}_t^T \boldsymbol{\eta}_t]} > 0.5, \tag{4.12}$$

   and $\hat{c}(\boldsymbol{x}_t) = 1$ otherwise.

3. Calculate

$$\check{p}_t = \frac{\exp[\boldsymbol{a}_t^T \boldsymbol{\eta}_t]}{1+\exp[\boldsymbol{a}_t^T \boldsymbol{\eta}_t]},$$

   and the updated mean and variance parameters

$$C_t = R_t - \left(\frac{\check{p}_t(1-\check{p}_t)}{1+\check{p}_t(1-\check{p}_t)\boldsymbol{\eta}_t^T R_t \boldsymbol{\eta}_t}\right)(R_t\boldsymbol{\eta}_t)(R_t\boldsymbol{\eta}_t)^T, \tag{4.13}$$

$$\boldsymbol{m}_t = \boldsymbol{a}_t + (y_t - 1 - \check{p}_t)C_t\boldsymbol{\eta}_t. \tag{4.14}$$

[1]The value of c is obtained by twice applying the approximation $e^x/(1+e^x) \approx \Phi(0.607x)$.

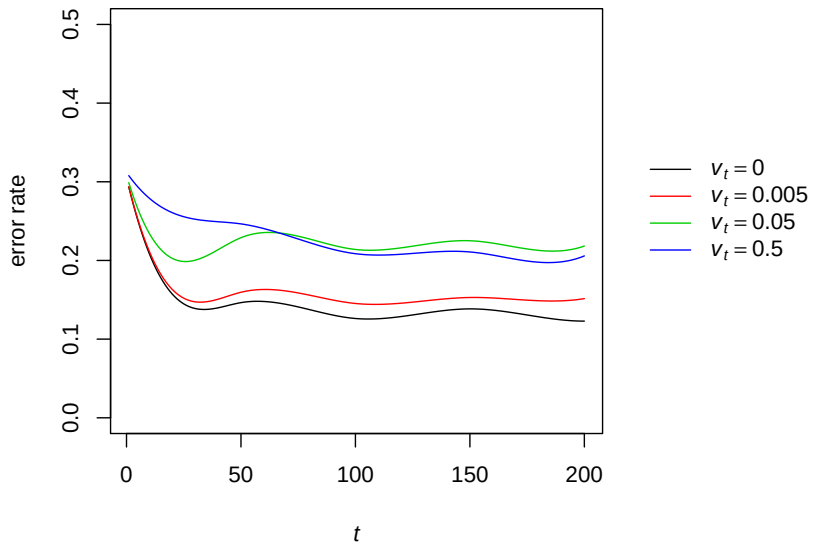


Figure 4.4: Smoothed average error rates of the Gaussian algorithm on the stationary scenario, for several values of the parameter evolution variance $V_t = v_t I$.

The main advantage of this approximation is the computational simplicity. At each time one has only to make a prediction using (4.12), and then calculate the expressions for the updated mean and variance of $\boldsymbol{\beta}_t$ via (4.13) and (4.14). This approximation is likely to be good if the true distribution of $\boldsymbol{\beta}_t|\boldsymbol{z}_{1:t-1}$ is well approximated by a normal distribution for all t, which may well be the case for simple problems. However, for more complex problems it is less likely that the normal approximation and the associated assumptions of symmetry and unimodality will be appropriate.

The results for applying this algorithm to the scenarios of no population change and slow population change are shown in figures 4.4 and 4.5 respectively. In each scenario we compared the results when using $v_t = 0$, $v_t = 0.005$, $v_t = 0.05$ and $v_t = 0.5$.

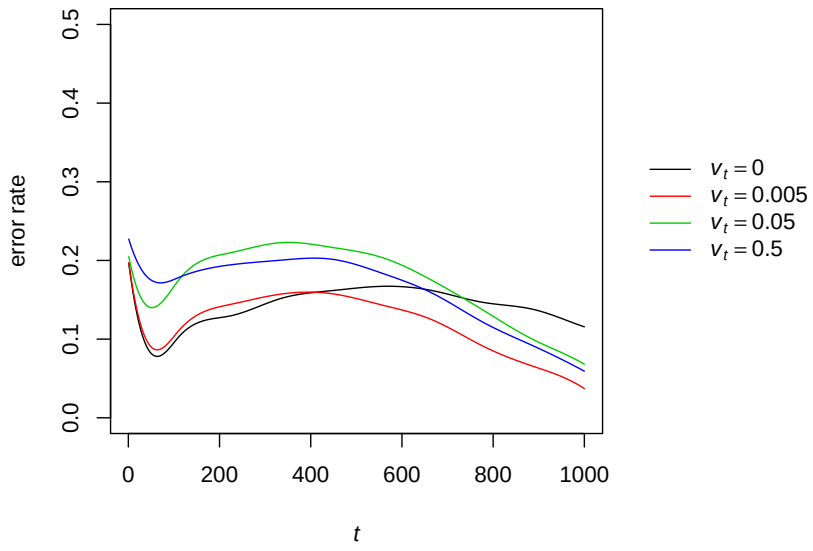


Figure 4.5: Smoothed average error rates of the Gaussian algorithm on the dynamic scenario, for several values of the parameter evolution variance $V_t = v_t I$.

From the results of figure 4.4, as expected for the stationary case, the lowest error rate is obtained when $v_t = 0$, which does slightly better than $v_t = 0.005$. Although $v_t = 0.05$ results in a lower error rate initially than when using $v_t = 0.5$, after about 100 updates their performance is comparable. This is because the error rate in the case $v_t = 0.5$ decreases slowly over time.

From the results of the dynamic scenario, shown in figure 4.5, the lowest error rate is obtained when using $v_t = 0.005$. Although using $v_t = 0$ does well initially, the performance deteriorates over time. The relative performance of the algorithm when using $v_t = 0.05$ and $v_t = 0.5$ follows the same pattern as observed in the stationary case.

4.4.2 Dynamic Generalised Linear Model

The Dynamic Linear Model (DLM) (Harrison & Stevens, 1976) was defined in section 1.1 (page 6), and can be seen to have many similarities to the model we propose for dynamic classification. In the DLM, the observations are modelled as normally distributed about the true parameter values, and in this case there exist exact methods of updating the posterior distribution of $\boldsymbol{\beta}_t$. Because for classification the observation distribution is multinomial rather than Gaussian, our model instead belongs to the class of Dynamic Generalised Linear Models (DGLM) (West et al., 1985). The DGLM is defined by a parameter vector $\boldsymbol{\beta}_t$, a regression vector $\boldsymbol{\eta}_t$, a link function $g(\cdot)$, a known observation variance V_t and known evolution matrix G_t. The observation model is

$$g(\rho_t) = \lambda_t = \boldsymbol{\beta}_t^T \boldsymbol{\eta}_t, \tag{4.15}$$

and the evolution equation

$$\boldsymbol{\beta}_t = G_t \boldsymbol{\beta}_{t-1} + \boldsymbol{\omega}_t, \ \boldsymbol{\omega}_t \sim (\mathbf{0}, V_t),$$

where the notation $\boldsymbol{\omega}_t \sim (\mathbf{0}, V_t)$ means the distribution of $\boldsymbol{\omega}_t$ has first moment $\mathbf{0}$ and second moment V_t. Our model (2.7) satisfies this framework when $G_t = I$, $g(\cdot)$ is the identity function, and

$$\rho_t = \log\left(\frac{p_t(2|\boldsymbol{x}_t)}{p_t(1|\boldsymbol{x}_t)}\right). \tag{4.16}$$

Note that (4.16) implies

$$p_t(2|\boldsymbol{x}_t) = \frac{\exp\{\rho_t\}}{1+\exp\{\rho_t\}}.$$

Due to the non-normality of the observation model in this case, there is no longer an exact method for updating the distribution of $\boldsymbol{\beta}_t|\boldsymbol{z}_{1:t}$. However, West et al. (1985) developed an approximate updating procedure which we adopt and describe here.

Updating Algorithm (DGLM - known V_t)

Rather than specify the form for the posterior distribution of $\boldsymbol{\beta}_t$, only the first two moments are used in order to update and make predictions. Therefore, we assume

$$\begin{aligned} \boldsymbol{\beta}_{t-1}|\boldsymbol{z}_{1:t-1} &\sim (\boldsymbol{m}_{t-1}, C_{t-1}), \text{ which implies} \\ \boldsymbol{\beta}_t|\boldsymbol{z}_{1:t-1} &\sim (\boldsymbol{a}_t, R_t), \end{aligned}$$

where $\boldsymbol{a}_t = \boldsymbol{m}_{t-1}$ and $R_t = C_{t-1} + V_t$ as in section 4.4.1. Prediction and updating proceed in four steps:

1. Prior:

$$\begin{pmatrix} \lambda_t \\ \boldsymbol{\beta}_t \end{pmatrix} |\boldsymbol{z}_{1:t-1} \sim \left[\begin{pmatrix} f_t \\ \boldsymbol{a}_t \end{pmatrix}, \begin{pmatrix} q_t & \boldsymbol{\eta}_{2t}^T R_t \\ R_t \boldsymbol{\eta}_{2t}^T & R_t \end{pmatrix} \right], \tag{4.17}$$

   where

$$f_t = \boldsymbol{a}_t^T \boldsymbol{\eta}_t \text{ and } q_t = \boldsymbol{\eta}_t^T R_t \boldsymbol{\eta}_t.$$

2. Prediction:

   In order to make a prediction and update the posterior distribution of ρ_t (4.16), we need to make assumptions about the form of the prior for ρ_t. For ease of computation the conjugate form is used. The conjugate prior, expressed in terms of $p_t(2|\boldsymbol{x}_t)$, is

$$p(2|\boldsymbol{x}_t, \boldsymbol{z}_{1:t-1}) \sim \text{Beta}(r_t, s_t). \tag{4.18}$$

Under this assumption, the mean and variance of λ_t (which from (4.15) is equal to ρ) are given by

$$E[\lambda_t|\boldsymbol{z}_{1:t-1}] = \gamma(r_t) - \gamma(s_t), \text{ and} \tag{4.19}$$

$$V[\lambda_t|\boldsymbol{z}_{1:t-1}] = \dot{\gamma}(r_t) - \dot{\gamma}(s_t), \tag{4.20}$$

where $\gamma(x)$ is the digamma function $\dot{\Gamma}(x)/\Gamma(x)$, and $\dot{\gamma}(x)$ is the trigamma function. The parameters r_t and s_t are chosen such that (4.19) and (4.20) are consistent with the prior (4.17). Therefore, s_t and r_t are the solutions to the equations

$$\begin{aligned} f_t &= \gamma(r_t) - \gamma(s_t) \text{ and} \\ q_t &= \dot{\gamma}(r_t) - \dot{\gamma}(s_t). \end{aligned}$$

This can be solved numerically, but we use the approximate solution suggested in West et al. (1985), namely

$$\begin{aligned} r_t &\approx q_t^{-1}(1 + \exp(f_t)) \text{ and} \\ s_t &\approx q_t^{-1}(1 + \exp(-f_t)). \end{aligned}$$

Finally, under the form of the prior (4.18), the forecast distribution of Y_t is beta-binomial. Therefore,

$$p(y_t|\boldsymbol{x}_t, \boldsymbol{z}_{1:t-1}) = \begin{cases} \frac{\Gamma(r_t+s_t)\Gamma(r_t+1)}{\Gamma(r_t)\Gamma(r_t+s_t+1)}, & y_t = 2 \\ \frac{\Gamma(r_t+s_t)\Gamma(s_t+1)}{\Gamma(s_t)\Gamma(r_t+s_t+1)}, & y_t = 1 \end{cases}$$

and we use the classification rule $\hat{c}(\boldsymbol{x}_t) = 2$ if

$$p(2|\boldsymbol{x}_t, \boldsymbol{z}_{1:t-1}) = \frac{\Gamma(r_t + s_t)\Gamma(r_t + 1)}{\Gamma(r_t)\Gamma(r_t + s_t + 1)} > 0.5.$$

3. Updating for $g(\rho_t)$:

   After observing Y_t we update the posterior

$$p(2|\boldsymbol{x}_t, \boldsymbol{z}_{1:t}) \sim \text{Beta}(r_t + Y_t - 1, s_t + 2 - Y_t).$$

   Then the posterior mean and variance of λ_t are

$$\begin{aligned} f_t^* = E[\lambda_t|\boldsymbol{z}_{1:t}] &= \gamma(r_t + Y_t - 1) - \gamma(s_t + 2 - Y_t), \text{ and} \\ q_t^* = V[\lambda_t|\boldsymbol{z}_{1:t}] &= \dot{\gamma}(r_t + Y_t - 1) - \dot{\gamma}(s_t + 2 - Y_t). \end{aligned}$$

4. Finally, we update the moments of the distribution of $\boldsymbol{\beta}_t$:

$$\begin{aligned} \text{Var}(\boldsymbol{\beta}_t|\boldsymbol{z}_{1:t}) = C_t &= \text{Var}[\hat{E}(\boldsymbol{\beta}_t|\lambda_t, \boldsymbol{z}_{t-1})|\boldsymbol{z}_{1:t}] + E[\widehat{\text{Var}}(\boldsymbol{\beta}_t|\lambda_t, \boldsymbol{z}_{1:t-1})|\boldsymbol{z}_{1:t}] \\ &= R_t - (1 - q_t^*/q_t)/q_t R_t \boldsymbol{\eta}_t (R_t \boldsymbol{\eta}_t)^T, \end{aligned} \tag{4.21}$$

$$\begin{aligned} E[\boldsymbol{\beta}_t|\boldsymbol{z}_{1:t}] = \boldsymbol{m}_t &= E_\lambda[\hat{E}(\boldsymbol{\beta}_t|\lambda_t, \boldsymbol{z}_{1:t-1})|\boldsymbol{z}_{1:t}] \\ &= E_\lambda[\boldsymbol{a}_t + R_t \boldsymbol{\eta}_t (\lambda_t - f_t)/q_t|\boldsymbol{z}_{1:t}] \\ &= \boldsymbol{a}_t + R_t \boldsymbol{\eta}_t (f_t^* - f_t)/q_t, \end{aligned} \tag{4.22}$$

   where linear Bayes estimation is used to obtain the estimates $\hat{E}$ and $\widehat{\text{Var}}$. Details are given in West & Harrison (1997, ch. 4.9).

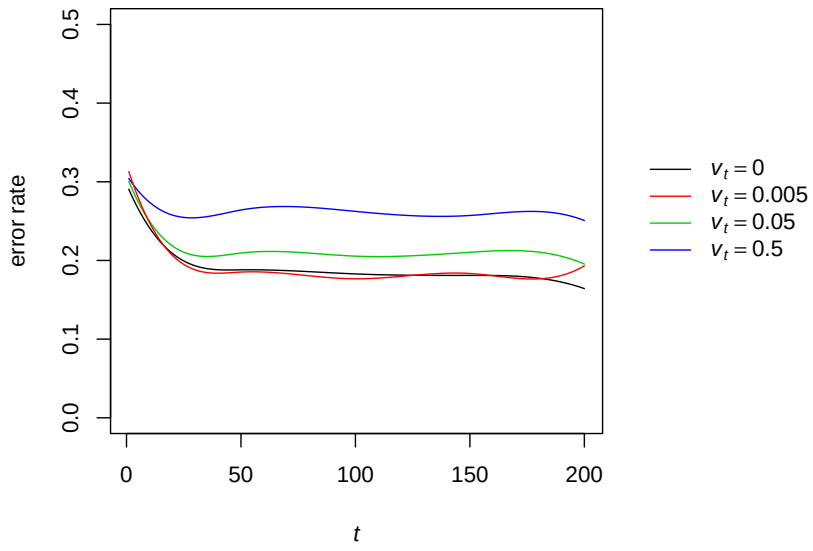


Figure 4.6: Smoothed estimated error rates of the DGLM algorithm on the stationary scenario, for several values of the parameter evolution variance $V_t = v_t I$.

Comparing (4.22) and (4.21) to to the normal update equations (4.14) and (4.13), one can see that they have a similar form. Although in this case we have dropped the assumption that $\boldsymbol{\beta}_t|\boldsymbol{z}_{1:t}$ is normally distributed, it is still assumed implicitly to be unimodal. The other similarity between the two algorithms is that it is assumed the variance of parameter evolution V_t is known.

The results of using the DGLM algorithm on the scenarios of no change and slow change are shown in figures 4.6 and 4.7 respectively. Once again, we compare the results when using $v_t = 0$, $v_t = 0.005$, $v_t = 0.05$ and $v_t = 0.5$.

For the stationary scenario, the results obtained with $v_t = 0$ and $v_t = 0.005$ are very similar, and better than when using $v_t = 0.05$ or $v_t = 0.5$. The results on the dynamic scenario are similar to those for the stationary scenario - again $v_t = 0$ and $v_t = 0.005$ perform similarly and do better than $v_t = 0.05$ and $v_t = 0.5$. After 500 updates $v_t = 0.005$ seems to do slightly better than $v_t = 0$, but the difference is

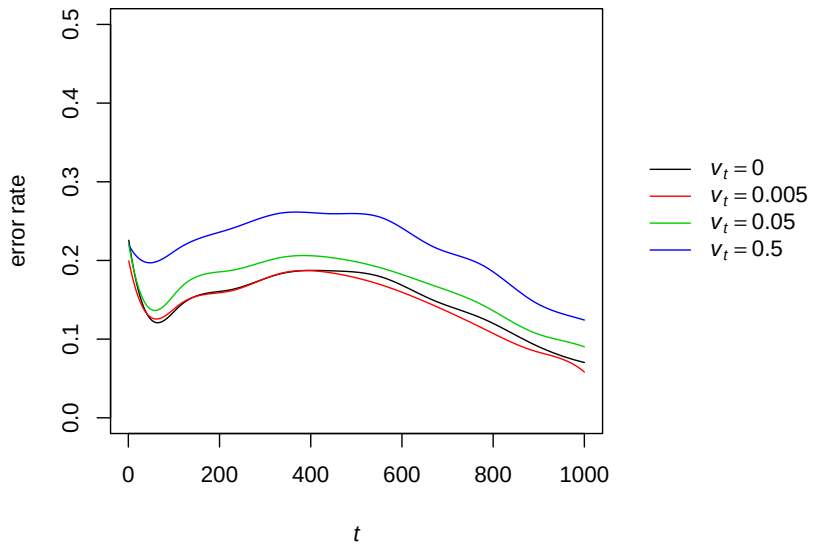


Figure 4.7: Smoothed estimated error rates of the DGLM algorithm on the dynamic scenario, for several values of the parameter evolution variance $V_t = v_t I$.

only small and certainly not significant.

4.4.3 Sequential Monte Carlo

Monte Carlo methods for sequential estimation of distributions such as $\boldsymbol{\beta}_t | \boldsymbol{z}_{1:t-1}$ have been studied for many years. They have the advantage of requiring few distributional assumptions about the process, but are also more computationally intensive than the algorithms given in sections 4.4.1 and 4.4.2. The on-line classification problem we are dealing with fits easily into the general framework of SMC algorithms; for this reason its application to classification problems has been studied before (for example Højen-Sørensen et al., 2000; de Freitas et al., 2000b).

The basic idea underlying sequential Monte Carlo (SMC) methods is that at any time t, we represent the conditional distribution of $\boldsymbol{\beta}_t$ by a number of discrete, weighted points. These points (or "particles") evolve over time to reflect parameter

evolution and the information obtained from observations. We denote the number of particles used in the approximation by N (fixed), and the particles themselves by $\boldsymbol{\beta}_t^{(1)}, \boldsymbol{\beta}_t^{(2)}, \ldots, \boldsymbol{\beta}_t^{(N)}$. The weight associated with the ith particle is denoted $\tilde{w}_t^{(i)}$, where $\sum_{i=1}^{N} \tilde{w}_t^{(i)} = 1$ and each $\tilde{w}_t^{(i)} > 0$. At time t the distribution is represented by the set of all particles and weights $\{\tilde{w}_t^{(i)}, \boldsymbol{\beta}_t^{(i)}\}_{i=1}^{N}$. The Monte Carlo approximation to $p(\boldsymbol{\beta}_t|\boldsymbol{z}_{1:t})$ is then given by

$$\hat{p}(\boldsymbol{\beta}_t|\boldsymbol{z}_{1:t}) = \sum_{i=1}^{N} w_t^{(i)} \delta(\boldsymbol{\beta}_t - \boldsymbol{\beta}_t^{(i)}), \tag{4.23}$$

where $\delta(\cdot)$ is the Dirac-delta function. Expected values of functions of the parameters can be approximated by

$$\hat{E}[f(\boldsymbol{\beta}_t)] = \sum_{i=1}^{N} w_t^{(i)} f(\boldsymbol{\beta}_t^{(i)}). \tag{4.24}$$

For the algorithms considered in this section, the approximation (4.24) approaches the true expected value $E[f(\boldsymbol{\beta}_t)]$ as $N \to \infty$ (Geweke, 1989). However, this approximation suffers from the "curse of dimensionality"; in order to maintain the same density of particles $\boldsymbol{\beta}_t^{(i)}$ in the parameter space and therefore maintain the quality of approximation, the number of particles must increase exponentially in relation to the dimension of the parameter space.

The updating algorithm we use is based on those contained in Doucet et al. (2000), which gives an overview of the basic SMC algorithms as well as some extensions. There has been more recent work suggesting improvements to the basic algorithms (for example the resample-move algorithm of Gilks & Berzuini (2001)), but because we are assuming a relatively "nice" model for the evolution of the parameters $\boldsymbol{\beta}_t$ we do not expect to encounter the kinds of sample degradation problems

the improvements are designed to overcome. Hence we restrict ourselves to applying sequential importance sampling with resampling, according to the algorithm now described.

Updating Algorithm (SMC - known V_t)

Firstly, we define the relations

$$p(Y_t = k|\boldsymbol{\beta}_t, \boldsymbol{x}_t) = \frac{\exp\{\boldsymbol{\beta}_t^T \boldsymbol{\eta}_k(\boldsymbol{x}_t)\}}{1 + \sum_{j=2}^{K} \exp\{\boldsymbol{\beta}_t^T \boldsymbol{\eta}_j(\boldsymbol{x}_t)\}}, k = 1, 2, \ldots, K, \quad (4.25)$$

$$p(\boldsymbol{\beta}_t|\boldsymbol{\beta}_{t-1}, V_t) \sim N(\boldsymbol{\beta}_{t-1}, V_t), \text{ and} \quad (4.26)$$

$$\pi(\boldsymbol{\beta}_t|\boldsymbol{\beta}_{t-1}, V_t, \nu) \sim T(\boldsymbol{\beta}_{t-1}, V_t, \nu), \quad (4.27)$$

where V_t is assumed known. The importance density (4.27) is chosen to be that of a noncentral multivariate T-distribution (see Genz & Bretz, 2002) with noncentrality parameter $\boldsymbol{\beta}_{t-1}$, correlation matrix V_t and ν degrees of freedom, where ν is specified in advance. We then choose a value for the number of particles, N, and define the threshold for resampling, N_{thres}. Define

$$\tilde{w}_0^{(i)} = 1/N, \text{ for } i = 1, 2, \ldots, N,$$

and sample $\boldsymbol{\beta}_0^{(i)}, i = 1, 2, \ldots, N$ from a chosen prior distribution $p(\boldsymbol{\beta}_0)$.

For $t = 1, 2, \ldots$

1. (Particle Evolution) For $i = 1, 2, \ldots, N$, draw

$$\boldsymbol{\beta}_t^{(i)} \sim \pi(\boldsymbol{\beta}_t|\boldsymbol{\beta}_{t-1}^{(i)}, V_t, \nu),$$
$$\boldsymbol{\beta}_t^{*(i)} \sim p(\boldsymbol{\beta}_t|\boldsymbol{\beta}_{t-1}^{(i)}, V_t),$$

and calculate

$$p(Y_t = k|\boldsymbol{x}_t, \boldsymbol{\beta}_t^{(i)}),\ k = 1, 2, \ldots, K,\ \text{from (4.25)},$$
$$p(\boldsymbol{\beta}_t^{(i)}|\boldsymbol{\beta}_{t-1}^{(i)}, V_t)\ \text{from (4.26), and}$$
$$\pi(\boldsymbol{\beta}_t^{(i)}|\boldsymbol{\beta}_{t-1}^{(i)}, V_t, \nu)\ \text{from (4.27).}$$

2. (Prediction) Calculate

$$\begin{aligned} \hat{p}(2|\boldsymbol{x}_t) &= \hat{E}_{\boldsymbol{\beta}_t|\boldsymbol{z}_{1:t-1}}[p(2|\boldsymbol{\beta}_t, \boldsymbol{x}_t)] \\ &= \sum_{i=1}^{N} p(2|\boldsymbol{\beta}_t^{*(i)}, \boldsymbol{x}_t)\tilde{w}_{t-1}^{(i)}, \end{aligned} \tag{4.28}$$

and make the prediction

$$\hat{c}(\boldsymbol{x}_t) = \begin{cases} 2 & \text{if } \hat{p}(2|\boldsymbol{x}_t) > 0.5 \\ 1 & \text{otherwise.} \end{cases}$$

3. (Updating) After observing Y_t, for $i = 1, 2, \ldots, N$ calculate

$$w_t^{*(i)} = \tilde{w}_{t-1}^{(i)}\ \frac{p(y_t|\boldsymbol{x}_t, \boldsymbol{\beta}_t^{(i)})p(\boldsymbol{\beta}_t^{(i)}|\boldsymbol{\beta}_{t-1}^{(i)}, V_t)}{\pi(\boldsymbol{\beta}_t^{(i)}|\boldsymbol{\beta}_{t-1}^{(i)}, V_t, \nu)}, \tag{4.29}$$

then normalise the weights

$$\tilde{w}_t^{(i)} = \frac{w_t^{*(i)}}{\sum_{i=1}^{N} w_t^{*(i)}},\ i = 1, 2, \ldots, N.$$

4. (Re-sampling) Calculate

$$\hat{N}_{eff} = \frac{1}{\sum_{i=1}^{N}(\tilde{w}_t^{(i)})^2}.$$

If $\hat{N}_{eff} < N_{thres}$, for $i = 1, 2, \ldots, N$,

- sample an index $j(i) \sim Pr\{j(i) = l\} = \tilde{w}_t^{(l)}, j = 1, 2, \ldots, N$
- set $\boldsymbol{\beta}_t^{(i)} = \boldsymbol{\beta}_t^{(j(i))}$.

Unknown Parameter Evolution Variance V_t

The above algorithm, as for the Gaussian and DGLM algorithms presented earlier, assumes that the parameter evolution variance V_t is known. Although we may have some information about how fast the population is changing, in general we will not know the optimal value of V_t. One advantage of using a SMC algorithm to update the parameters is that we can easily incorporate V_t into the set of unknown parameters. This can be seen as an advantage of our model-based approach over heuristic methods, for which it is often difficult to know how to update the memory parameter.

Under the assumption that V_t can be written $v_t I$, we can model the evolution of v_t as a random walk:

$$v_t = v_{t-1} + \nu_{t-1}, \ \nu_{t-1} \sim N(\mathbf{0}, W_t), \tag{4.30}$$

where here W_t is assumed known. Although we must still specify W_t, misspecification of this hyper-parameter is likely to be less influential than misspecification of v_t itself. Hence we can amend step 1 in the above algorithm to draw

$$\begin{aligned}
v_t^{*(i)} &\sim p(v_t | v_{t-1}^{(i)}, W_t) \\
v_t^{(i)} &\sim \pi(v_t | v_{t-1}^{(i)}, W_t, \nu) \\
\boldsymbol{\beta}_t^{*(i)} &\sim p(\boldsymbol{\beta}_t | \boldsymbol{\beta}_{t-1}^{(i)}, v_t^{*(i)} I) \\
\boldsymbol{\beta}_t^{(i)} &\sim \pi(\boldsymbol{\beta}_t | \boldsymbol{\beta}_{t-1}^{(i)}, v_t^{(i)} I, \nu).
\end{aligned}$$

If we assume that V_t is independent of $\boldsymbol{\beta}_t$ (which may often be a reasonable assumption), then after observing Y_t we can update the weights

$$w_t^{*(i)} \propto w_{t-1}^{*(i)} \frac{p(y_t|\boldsymbol{x}_t, \boldsymbol{\beta}_t^{(i)}) p(\boldsymbol{\beta}_t^{(i)}|\boldsymbol{\beta}_{t-1}^{(i)}, v_t^{(i)} I) p(v_t^{(i)}|v_{t-1}^{(i)}, W_t)}{\pi(\boldsymbol{\beta}_t^{(i)}|\boldsymbol{\beta}_{t-1}^{(i)}, v_t^{(i)} I, \nu) \pi(v_t^{(i)}|v_{t-1}^{(i)}, W_t, \nu)}. \tag{4.31}$$

Further, if we were to assume that V_t is unknown but constant, i.e. $W_t = 0$, then (4.31) reduces to

$$w_t^{*(i)} \propto w_{t-1}^{*(i)} \frac{p(y_t|\boldsymbol{x}_t, \boldsymbol{\beta}_t^{(i)}) p(\boldsymbol{\beta}_t^{(i)}|\boldsymbol{\beta}_{t-1}^{(i)}, v_t^{(i)} I)}{\pi(\boldsymbol{\beta}_t^{(i)}|\boldsymbol{\beta}_{t-1}^{(i)}, v_t^{(i)} I, \nu)}.$$

Because estimating v_t increases the number of unknown parameters from M to $M + 1$, to maintain the same density of particles as when assuming V_t is known we should use $N^{\frac{M+1}{M}}$ particles. Note that this can be significantly less than the N^2 particles which would be required if we assumed only that V_t was diagonal.

Hence, using a SMC algorithm allows several choices for how to model the parameter evolution variance. We consider the following three possibilities:

1. $V_t = v_t I$, where v_t is pre-determined and assumed known for all t,
2. $V_t = v_t I$, where v_t is a constant but unknown scalar ($W_t = \mathbf{0}$),
3. $V_t = v_t I$, where v_t is a time-varying and unknown scalar ($W_t \neq \mathbf{0}$).

Firstly, in figure 4.8 we compare the performance of the algorithm with V_t assumed constant and known for several choices of v_t. In all cases, the parameter ν was set equal to 3 and the prior distribution of $\boldsymbol{\beta}_0$ was chosen to be Gaussian with mean $\mathbf{0}$ and variance $0.2I$. We used 500 particles with a resampling threshold of 300. From the results in figure 4.8(a) it can be seen that $v_t = 0$ gave the best results on the stationary scenario. Once again, as for the Gaussian and DGLM algorithms,

due to increased variance of the decision boundary the error rate in this case is seen to increase with increasing v_t. For the dynamic cases with $T = 200$ and $T = 1000$, although $v_t = 0$ gave the best results initially it was slow to react to the population change compared to $v_t = 0.05$ and $v_t = 0.5$, both of which performed better than $v_t = 0$ after a number of updates. The relative performance of the choices of v_t in the dynamic scenario with $T = 200$ and $T = 1000$ are very similar. However, with $T = 200$ the error rates of the algorithms are slightly higher than for $T = 1000$, presumably due to the relative speed of change.

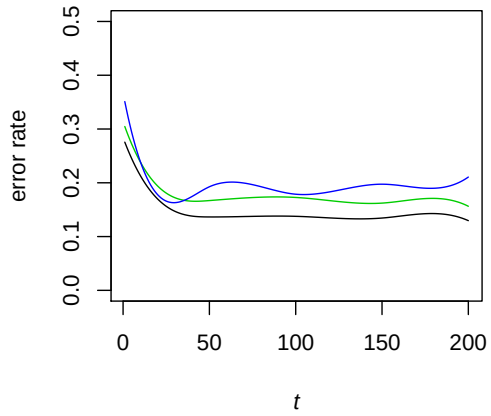
In figure 4.9 we compare the performance of the SMC algorithm when

a) $v_t = 0$ (stationary) or $v_t = 0.05$ (dynamic) for all t, with $N = 500/300$

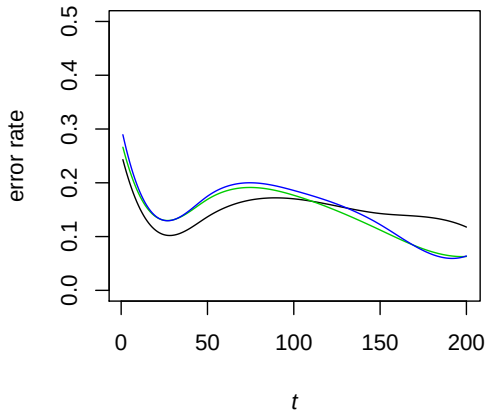
b) v_t is assumed to be unknown and constant, $N = 1000/1000$

c) v_t is assumed to be unknown and possibly time-varying, $N = 1000/600$.

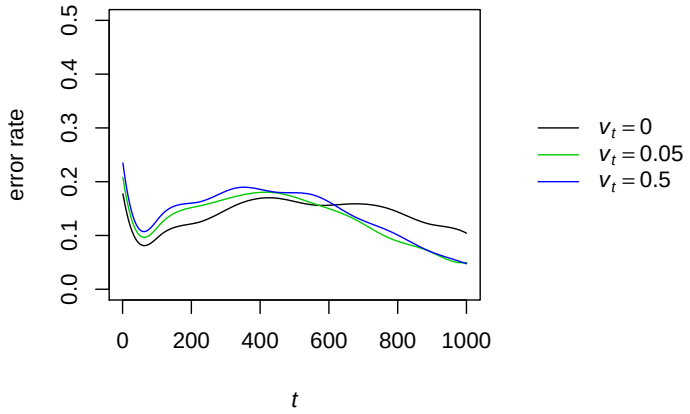
Here we use the notation $N = 500/300$ for example to mean we used 500 particles in the approximation with a resampling threshold of 300. In all cases the initial particles $\boldsymbol{\beta}_0^{(i)}$ were randomly sampled from a Gaussian distribution with mean $\mathbf{0}$ and variance $0.2I$. For the cases b) and c), $v_0 \sim N(0.1, 0.005)$. In case b) $W_t = 0$ for all t, and in c) $W_t = 0.005$ for all t. Other values for these parameters were tried, and these chosen because they seemed to give relatively good results. From figure 4.9(a) we can see that in the stationary scenario assuming that $v_t = 0$ gave the best results. This is what one would expect because we know that the optimal parameters do not change with time, and assuming otherwise only introduces an additional source of variation. In the dynamic scenarios of figures 4.9(b) and 4.9(c), all methods seemed to perform equally. Note that this implies incorporating v_t into the vector of unknown parameters results in performance equivalent to the best-case scenario of assuming v_t known. Once again, comparing the case $T = 200$ in figure 4.9(b) with the case $T = 1000$ in figure 4.9(c), we observe that the relative performances



(a) Stationary scenario

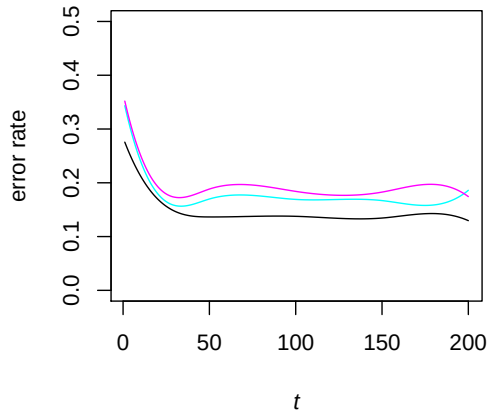


(b) Dynamic scenario - $T = 200$

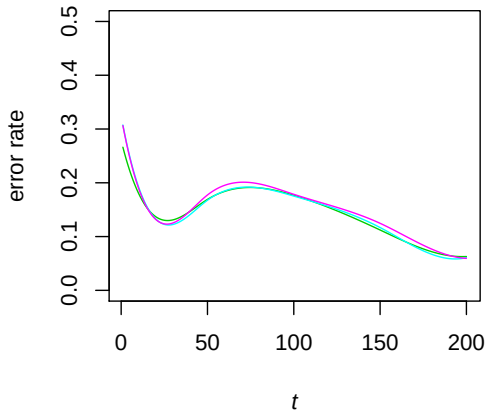


(c) Dynamic scenario - $T = 1000$

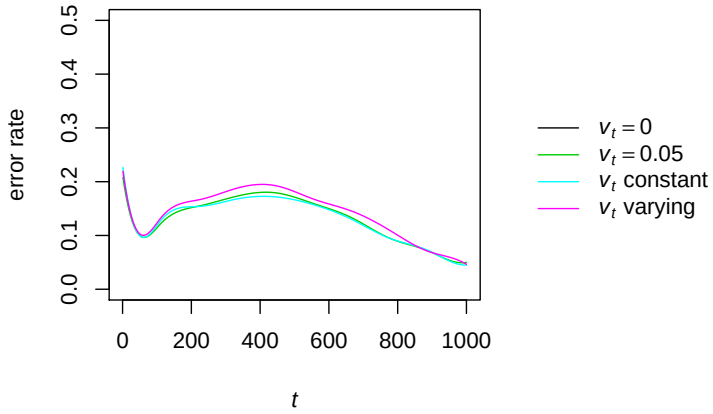
Figure 4.8: Smoothed average error rates when $V_t = v_t I$ is assumed to be known and constant, for several values of v_t.



(a) Stationary scenario



(b) Dynamic scenario - $T = 200$



(c) Dynamic scenario - $T = 1000$

Figure 4.9: Smoothed average error rates for the SMC algorithms with different assumptions about the parameter evolution variance $V_t = v_t I$.

of the algorithms is roughly the same, although the error rate is slightly higher for the case $T = 200$.

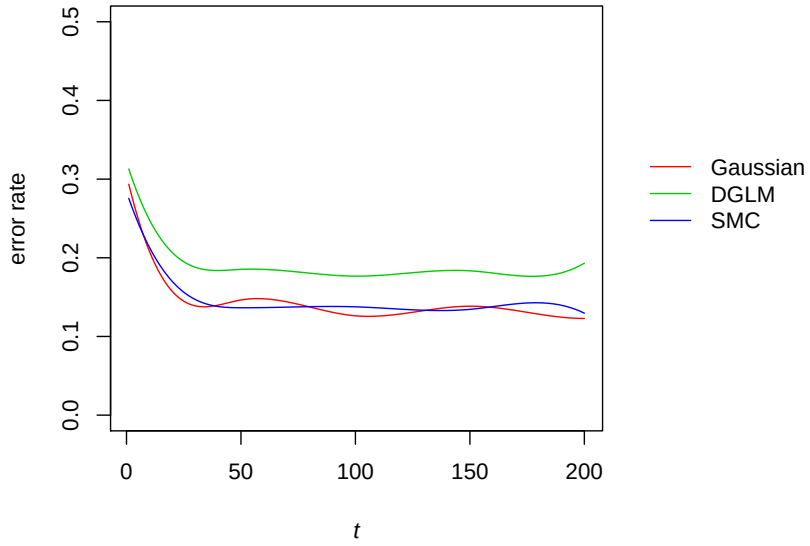
4.5 Comparison of Updating Algorithms

We now compare the best results obtained for each of the three updating algorithms considered. The results for the stationary scenario and the dynamic scenario with $T = 1000$ are shown in figures 4.10(a) and 4.10(b) respectively. From these results we can see that in general the SMC and Gaussian methods performed similarly. The Gaussian method seemed to do slightly better than the SMC for the middle part of the dynamic scenario. For both scenarios, the DGLM algorithm had the highest error rate at all time points.

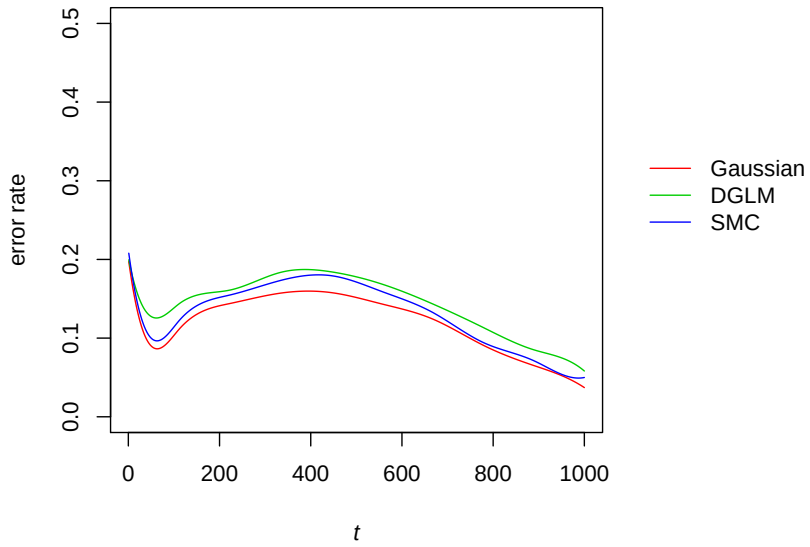
The performance of all methods was highly dependent on the value of v_t which was used. In addition, the value of v_t optimal for one updating algorithm was not necessarily optimal for the others. This can be seen from the dynamic scenario, where the best performance of the SMC algorithm was with $v_t = 0.05$, whereas the Gaussian and DGLM algorithms both performed best with $v_t = 0.005$.

4.6 Parameter Inflation

For the scenarios of population change considered in this chapter, we can calculate the optimal parameter values by solving equation (3.1) on page 37 for $\boldsymbol{\beta}_t$. Hence, we can compare the average estimated mean parameter values output by the algorithms to the optimal values. That is, we compare $\overline{\hat{E}[\boldsymbol{\beta}_t]}$ with $\boldsymbol{\beta}_t^*$. If we do this, we observe that the estimated parameter values output by the classification algorithms are



(a) Stationary scenario



(b) Dynamic scenario

Figure 4.10: Smoothed estimated error rates when using the Gaussian, DGLM and SMC updating algorithms. For each algorithm and scenario, the parameters used were those which gave the lowest error in previous simulations.

approximately inflated by a factor K, such that

$$\overline{\hat{E}[\boldsymbol{\beta}_t]} \approx K\boldsymbol{\beta}_t^*. \tag{4.32}$$

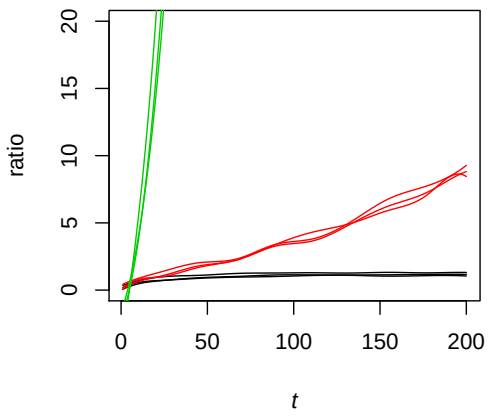
Figure 4.11 shows the ratio $\overline{\hat{E}[\boldsymbol{\beta}_t]}/\boldsymbol{\beta}_t^*$ for several fixed values of v_t for the Gaussian, DGLM and SMC algorithms on the stationary scenario. In each case we see the relationship of (4.32), where the factor K increases with v_t. Although this occurred for each updating algorithm, it was most pronounced for the Gaussian algorithm and then SMC algorithm (notice that the limits of the y-axis vary between sub-plots). In the case of the DGLM algorithm, the approximate value of K was less than one for small v_t. Similar results are obtained in figure 4.12 for the dynamic scenario. In this case, when β_i^* was close to zero the ratio became unstable, so we omitted values $t \in \{320, \ldots, 370, 800, \ldots, 815\}$ from the plots.

For the SMC algorithm when v_t was not assumed known, the average estimated expected values of v_t are shown in figure 4.13. For the cases that v_t was assumed to be unknown but constant, $\overline{\hat{E}[v_t]}$ steadily decreased and then levelled off. When v_t was assumed to be unknown and time-varying, we observe that $\overline{\hat{E}[v_t]}$ increases rapidly but then levels off, similarly to the pattern observed for $\overline{\hat{E}[\boldsymbol{\beta}_t]}$.

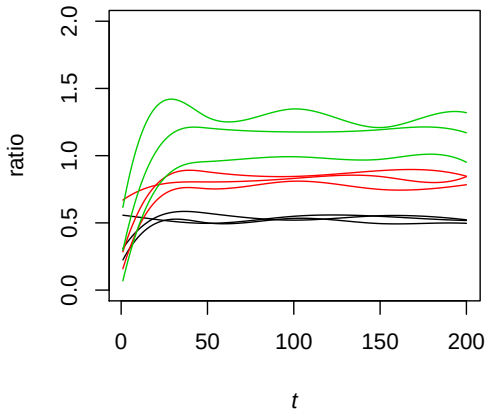
The reason for this effect is likely related to the behaviour of the likelihood function for logistic models in the presence of separation or near-separation of the data. It is well known that in the presence of separation between the observations from any two classes the maximum likelihood estimates (MLEs) are infinite. In addition, in finite samples the MLEs are biased, and the bias can be expressed as

$$b(\boldsymbol{\beta}) = \frac{b_1(\boldsymbol{\beta})}{n} + \frac{b_2(\boldsymbol{\beta})}{n^2} + \ldots, \tag{4.33}$$

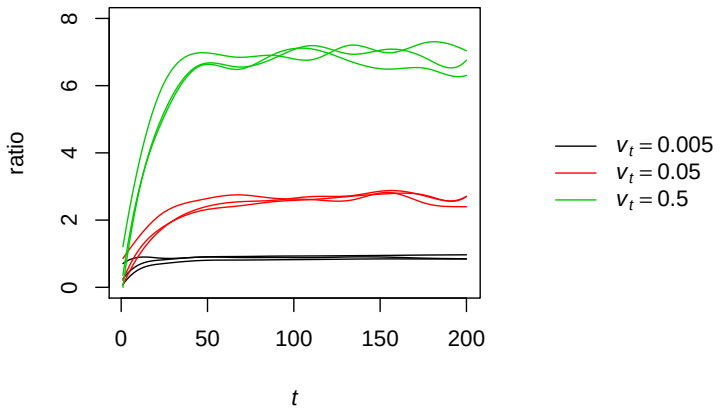
where n can be interpreted as the number of observations (Firth, 1993). Therefore,



(a) Gaussian algorithm

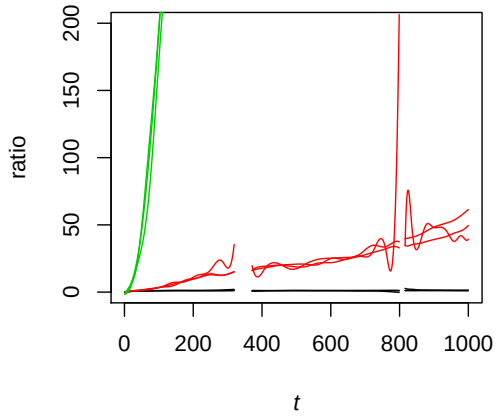


(b) DGLM algorithm

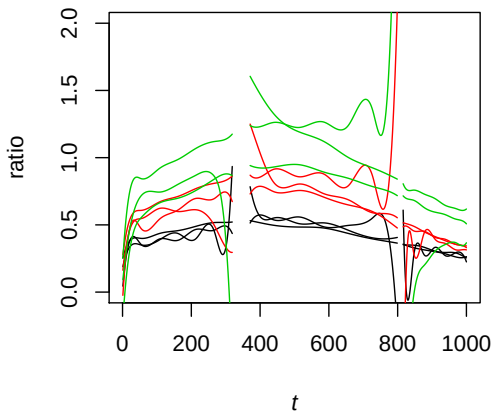


(c) SMC algorithm

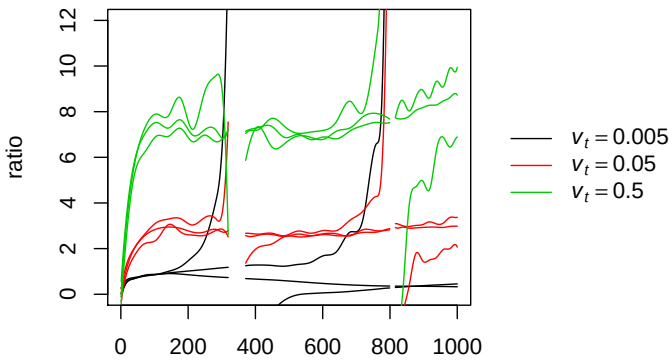
Figure 4.11: Ratios of the average estimated expected parameter values output by the updating algorithms to the optimal parameters in the stationary scenario. Note that the scale of the y-axis differs between sub-plots.



(a) Gaussian algorithm

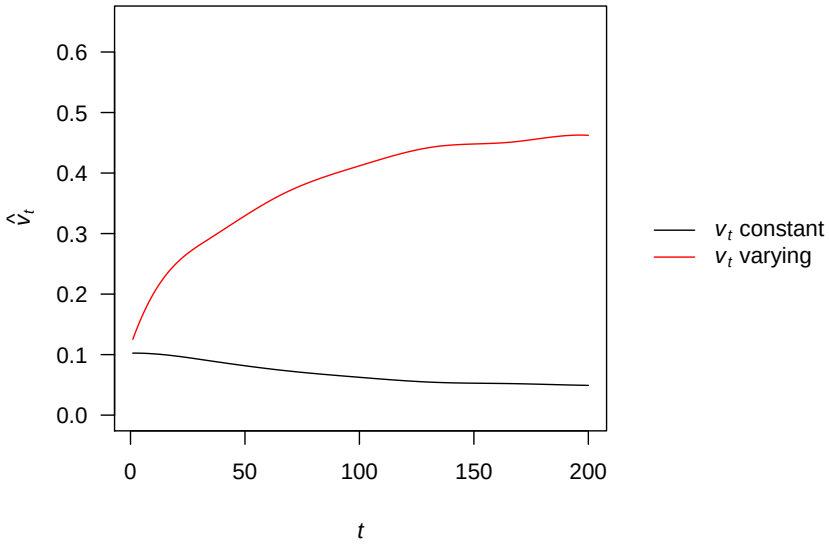


(b) DGLM algorithm

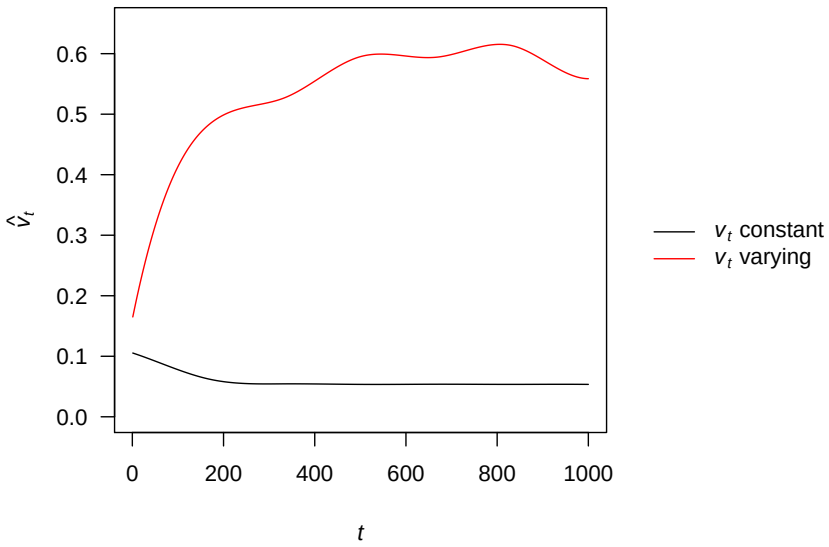


(c) SMC algorithm

Figure 4.12: Ratios of the average estimated expected parameter values output by the updating algorithms to the optimal parameters in the dynamic scenario. Note that the scale of the y-axis differs between sub-plots.



(a) Stationary scenario



(b) Dynamic scenario

Figure 4.13: Average estimated expected values of the variance parameter v_t output by the SMC updating algorithm, for the cases that v_t is assumed to be unknown but constant, and unknown and time-varying.

if the sample size is small, this bias can be significant. For the SMC updating algorithm, the relationship to maximum likelihood estimation is clear. For simplicity, consider the case that the importance density $\pi(\boldsymbol{\beta}_t|\boldsymbol{\beta}_{t-1})$ is chosen to be equal to the evolution density $p(\boldsymbol{\beta}_t|\boldsymbol{\beta}_{t-1})$. Then we can see from the weight updating equation (4.29) that the weight of the ith particle at time t

$$\begin{aligned} \tilde{w}_t^{(i)} &\propto \tilde{w}_{t-1}^{(i)}\, p(y_t|\boldsymbol{x}_t, \boldsymbol{\beta}_t^{(i)}), \text{ so} \\ \tilde{w}_t^{(i)} &\propto \tilde{w}_0^{(i)} \prod_{j=1}^{t} p(y_j|\boldsymbol{x}_j, \boldsymbol{\beta}_j^{(i)}). \end{aligned} \tag{4.34}$$

The particle $\boldsymbol{\beta}_t^{(i)}$ with most weight at time t is that with the maximum value of (4.34). This differs to the likelihood in the standard stationary case only because of the effect of the parameter evolution variance V_t. As discussed in section 4.2, in the dynamic scenario we can think of the parameter evolution variance V_t as specifying the "memory" of the process. Assuming $V_t = v_t I$, if v_t is zero then $\boldsymbol{\beta}_0^{(i)}, \ldots, \boldsymbol{\beta}_{t-1}^{(i)}$ are each equally informative about $\boldsymbol{\beta}_t^{(i)}$. If $v_t > 0$, then $\boldsymbol{\beta}_{t-1}^{(i)}$ contains more information than $\boldsymbol{\beta}_{t-2}^{(i)}$, so the more recent observations are more informative. In this way, increasing v_t is in some sense equivalent to reducing the sample size used. Hence, from (4.33), we would expect that by increasing v_t our parameter estimates become more biased, which is the effect observed in figures 4.11 and 4.12.

We have concluded then that if we use non-zero v_t we should expect the parameter estimates to be biased. The question of interest then, is will this affect the accuracy of the classifier? From the simulation outputs of section 4.5, we saw that both the parameter bias and the error rates of the algorithms tended to increase with v_t. There are two possible sources of this added error: one is the bias of the parameters, and the other is the increase in variance of the decision boundary of the classifier caused by increasing v_t.

Using v_t greater than optimal will increase the error of the classifier by increasing the variance of the location of the decision boundary. This is because changes in the parameters $\boldsymbol{\beta}_t$ over time result in changes in the decision boundary. Adding a random component to the value of $\boldsymbol{\beta}_t$ (which we do when $v_t > 0$) hence increases the variance of the parameter values and the decision boundary. In dynamic problems, the optimal v_t will be greater than zero, because this added movement in the parameter values is required to capture the movement of the decision boundary. However, if v_t is too large, the increase in variance of the decision boundary leads to extra variation in the decision boundary and hence increased error.

From note 2.4.2 (page 22), as long as the ratio of the parameters remains constant, inflated parameter estimates by themselves should not change the location of the decision boundary. Hence it is most likely the increase in variance which results in the observed increase in error rates. In theory, for stationary problems it should be possible to calculate the component of the error due to non-zero parameter evolution variance. This would allow us to decompose the difference between the observed error and the Bayes error into a component due to using a greater than optimal value for v_t and another component. However, the calculations involved are non-trivial, and we do not investigate the solution here.

4.6.1 Bias Reduction

Another way in which we could test our assertion that parameter inflation does not adversely affect the classifier is by trying to reduce this inflation. Bias of likelihood estimates for logistic regression models in small samples is a well-known phenomenon, and many methods for reducing this bias have been proposed. Firth (1993) considers methods as either "corrective", where one first calculates the MLEs and then corrects for the bias, or "preventative". Both methods we consider can

be classed as "preventative". Firstly we consider adding a term to the likelihood which penalises large parameter values, and then we consider an alternative form of modified likelihood.

In the penalisation approach, we multiply the likelihood at each time t by a factor which penalises large parameter values (Duffy & Santner, 1989). That is, the weight update equation (4.29) becomes

$$w_t^{*(i)} = w_{t-1}^{*(i)} \frac{p(y_t|\boldsymbol{x}_t, \boldsymbol{\beta}_t^{(i)}) p(\boldsymbol{\beta}_t^{(i)}|\boldsymbol{\beta}_{t-1}^{(i)}, V_t)}{\pi(\boldsymbol{\beta}_t^{(i)}|\boldsymbol{\beta}_{t-1}^{(i)})} g(\boldsymbol{\beta}_t^{(i)}),$$

where $g(\cdot)$ is a penalty function. We consider two options for this penalty function:

1. $g(\boldsymbol{\beta}_t) = ||\boldsymbol{\beta}_t||^{-\gamma}$, and
2. $g(\boldsymbol{\beta}_t) = \exp\left(-\gamma||\boldsymbol{\beta}_t||^2\right)$,

where $||\boldsymbol{\beta}_t|| = (\sum_i \beta_{ti}^2)^{1/2}$ is the norm of $\boldsymbol{\beta}_t$, and $\gamma > 0$ is a tuning parameter. The smoothed average error rates for the penalised SMC algorithm for several values of the parameter γ are shown in figure 4.14. It can be seen that for both penalty functions considered the error rate increased with γ, and no choice of $\gamma > 0$ did better than the case of using no penalty ($\gamma = 0$). This can be explained by considering the form of the penalty functions: each is a function of $||\boldsymbol{\beta}_t||$, so the size of the penalty depends only on the sum of the squared parameter values. For example, particles in which all components are inflated by a factor $K > 1$ will be penalised more than those for which only some components are inflated. However, we know that for our model the decision boundary depends only on the ratio of the parameters to one-another. The penalty terms used upset this ratio, which leads to favouring sub-optimal decision boundaries and hence increased error rates. This can be seen in the ratio plots of figure 4.15. When compared to the non-penalised

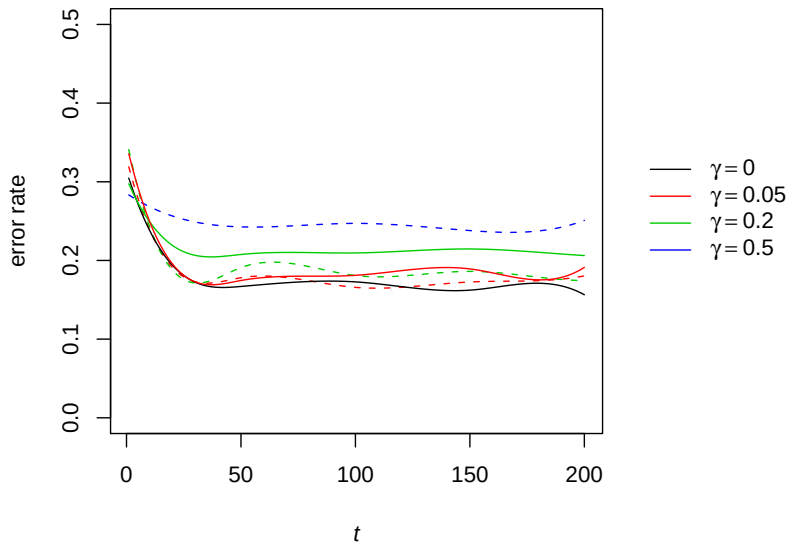
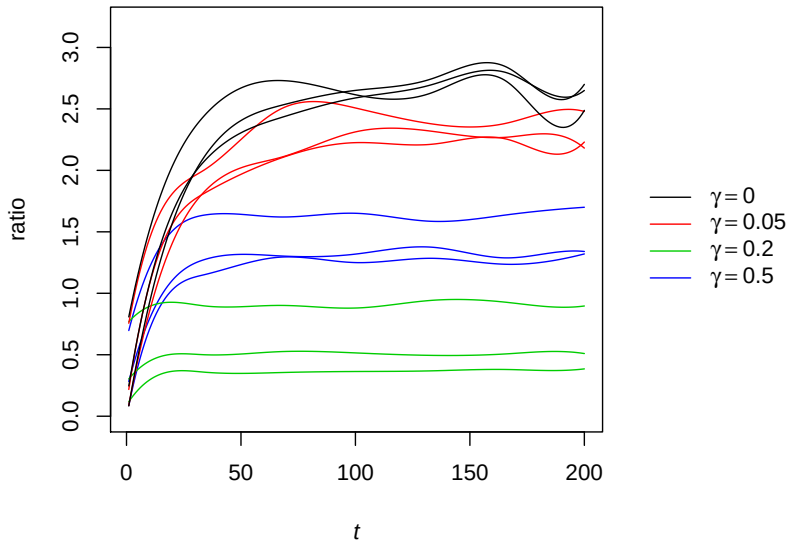


Figure 4.14: Smoothed average error rates when using the SMC algorithm with penalised likelihood. Results are shown when using the penalty function 1 (solid) or 2 (dashed), for several values of the parameter γ.

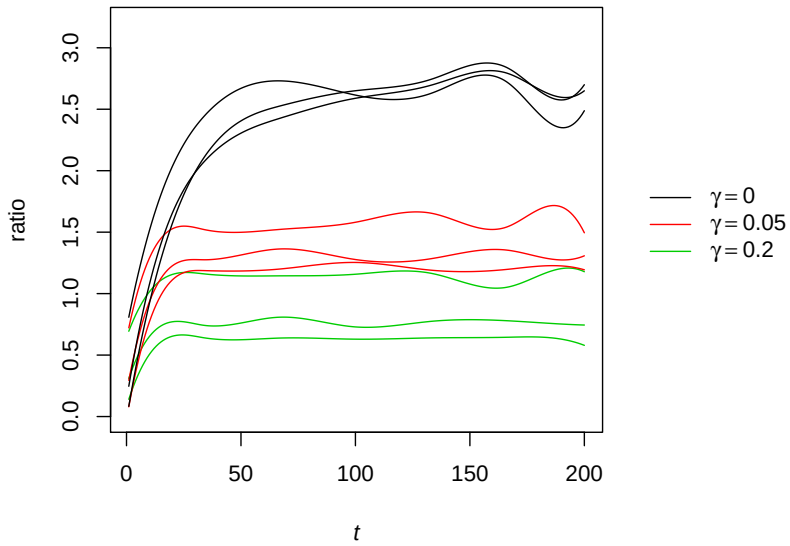
case ($\gamma = 0$), one can see that the ratio $\overline{\hat{E}[\beta_{tj}]}/\beta^*_{tj}$ is not the same for all j, which means $\overline{\hat{E}[\boldsymbol{\beta}_t]} \not\approx K\boldsymbol{\beta}^*_t$ for any K. Duffy & Santner (1989) showed that using the penalised likelihood reduces bias, but they measured the bias as the sum of the absolute bias of each component. Clearly this measure is not appropriate for the classification model we consider.

The second approach we tried was to modify the likelihood. We followed the work of Rousseeuw & Christmann (2003) who proposed a modified likelihood function which always has a finite maximum and is claimed to be robust against separation. For the stationary regression problem they consider, Rousseeuw & Christmann (2003) suggest using the parameter values which maximise the function

$$L(\boldsymbol{\beta}|\tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_n) = \prod_{l=1}^{n} p(Y = 1|\boldsymbol{\beta}, \boldsymbol{x}_l)^{\tilde{y}_l} p(Y = 2|\boldsymbol{\beta}, \boldsymbol{x}_l)^{1-\tilde{y}_l},$$



(a) Penalty function 1.



(b) Penalty function 2.

Figure 4.15: Ratios of $\overline{\hat{E}[\beta_{ti}]}/\beta^*_{ti}$ for $i = 1, 2, 3$ when using the SMC algorithm with penalised likelihood for several values of the parameter γ.

where the "pseudo-observations"

$$\tilde{y}_l = (2 - y_l)\delta_0 + (y_l - 1)\delta_1, \tag{4.35}$$

and $\delta = (\delta_0, \delta_1)$ is a vector of tuning parameters. We integrate this into the SMC updating rule by modifying the weight update equation (4.29) to

$$w_t^{*(i)} \propto w_{t-1}^{*(i)} \frac{p(Y_t = 1|\boldsymbol{\beta}_t^{(i)}, \boldsymbol{x}_t)^{\tilde{y}_t} p(Y_t = 2|\boldsymbol{\beta}_t^{(i)}, \boldsymbol{x}_t)^{1-\tilde{y}_t} p(\boldsymbol{\beta}_t^{(i)}|\boldsymbol{\beta}_{t-1}^{(i)}, V_t)}{\pi(\boldsymbol{\beta}_t^{(i)}|\boldsymbol{\beta}_{t-1}^{(i)})}, \tag{4.36}$$

where the $\tilde{y}_t$ are defined by (4.35). Rousseeuw & Christmann (2003) suggest methods for choosing the parameters δ_0 and δ_1, but when running the simulations we tried out several values to see which gave the best results.

From the output in figure 4.16, we see that this method is very effective at reducing the bias. Moreover, we can see that the modified likelihood approach preserves the optimal ratio between the parameters. However, from the plot of average error rates for this method shown in figure 4.17, we see that the lowest error rate was again obtained in the case that $\delta = (1, 0)$, i.e. no modification.

This is an interesting observation, as from figure 4.16 we would expect the algorithm with $\delta = (0.9, 0.1)$ to perform well. However, we can deduce that the added error of using $\delta = (0.9, 0.1)$ over $\delta = (1, 0)$ is due to added variation of the decision boundary. This is because for all values of δ we assumed that v_t was known and equal to $v_t = 0.05$. As stated before, in the stationary scenario the optimal $v_t = 0$. Reducing the bias in this case can increase the effect of using a larger than optimal value for v_t.

As an example, suppose we have only two component classifiers, and consider

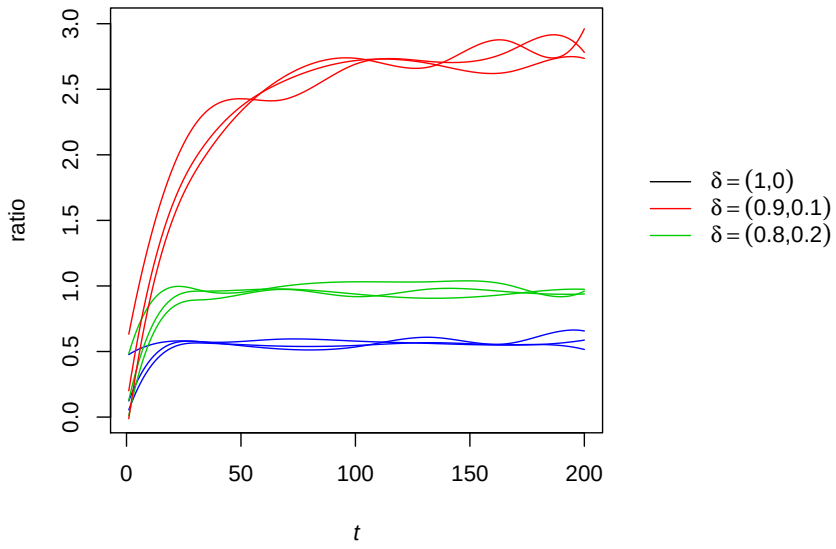


Figure 4.16: Ratio of $\overline{\hat{E}[\beta_{ti}]}/\beta^*_{ti}$ for $i = 1, 2, 3$ when using the SMC algorithm with modified likelihood, for several values of the parameters δ_0 and δ_1.

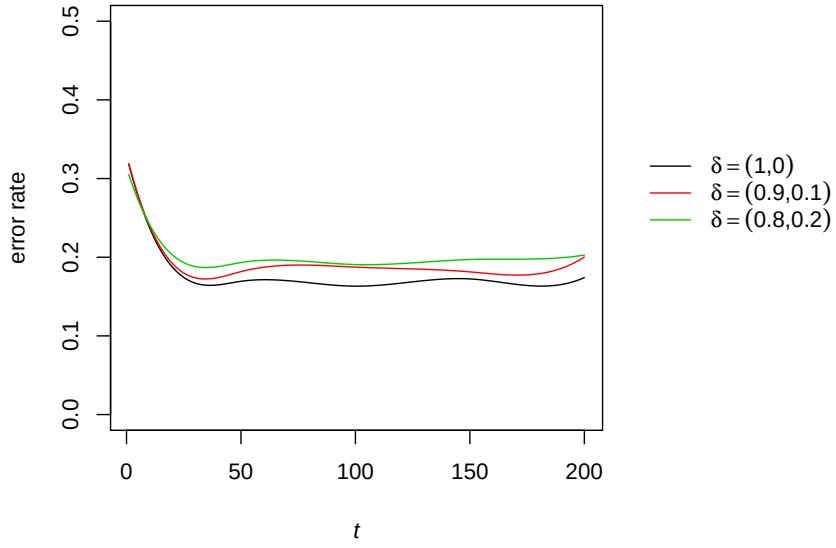


Figure 4.17: Average expected parameter values for the SMC updating algorithm when using the modified likelihood function, for several values of the parameters δ_0 and δ_1.

two particles $\boldsymbol{\beta}_t^{(i)}$ and $\boldsymbol{\beta}_t^{(j)}$. Suppose

$$\boldsymbol{\beta}_t^{(j)} = K\boldsymbol{\beta}_t^{(i)},$$

so that the decision boundaries corresponding to each particle are the same. Then

$$\frac{\beta_{t1}^{(j)}}{\beta_{t2}^{(j)}} = \frac{K\beta_{t1}^{(i)}}{K\beta_{t2}^{(i)}} = \frac{\beta_{t1}^{(i)}}{\beta_{t2}^{(i)}} \triangleq R_t,$$

say. At time $t+1$ the ratio of the component parameters of particle i will be

$$\frac{\beta_{t+1,1}^{(i)}}{\beta_{t+1,2}^{(i)}} = \frac{\beta_{t1}^{(i)} + \omega_{t1}^{(i)}}{\beta_{t2}^{(i)} + \omega_{t2}^{(i)}} \triangleq R_{t+1}^{(i)},$$

and the ratio of the component parameters of particle j will be

$$\frac{\beta_{t+1,1}^{(j)}}{\beta_{t+1,2}^{(j)}} = \frac{\beta_{t1}^{(j)} + \omega_{t1}^{(j)}}{\beta_{t2}^{(j)} + \omega_{t2}^{(j)}} = \frac{K\beta_{t1}^{(i)} + \omega_{t1}^{(j)}}{K\beta_{t2}^{(i)} + \omega_{t2}^{(j)}} \triangleq R_{t+1}^{(j)},$$

where $\omega_{t1}^{(i)}, \omega_{t2}^{(j)}, \omega_{t1}^{(j)}$ and $\omega_{t2}^{(j)}$ are independently drawn from a $N(0, v_t)$ distribution. The distribution of $R_{t+1}^{(i)} - R_t$ has some variance which is a function of v_t. The distribution of $R_{t+1}^{(j)} - R_t$ will be a function of both K and v_t. Clearly as K gets larger, $R_{t+1}^{(j)} \to R_t$, so the variance of $R_{t+1}^{(j)} - R_t$ should decrease for fixed v_t. Thus, increasing K for constant v_t results in decreased variance of the parameter ratio, which in turn will result in a more stable decision boundary and lower error rate.

To demonstrate this relationship, suppose $\boldsymbol{\beta}_t^{(i)} = (2, 1)$, $\boldsymbol{\beta}_t^{(j)} = K\boldsymbol{\beta}_t^{(i)}$ and $v_t = 0.5$. A random noise vector $\boldsymbol{\omega}_t^{(j)}$ was generated, and the difference between $R_{t+1}^{(j)}$ and R_t was calculated. This was done 1000 times and density plots of the results for $K = 0.5, 1, 2$ and 5 are shown in figure 4.18. From this example we observe that, for fixed v_t, as K increases the variance of $R_{t+1}^{(j)} - R_t$ tends to decrease. This can

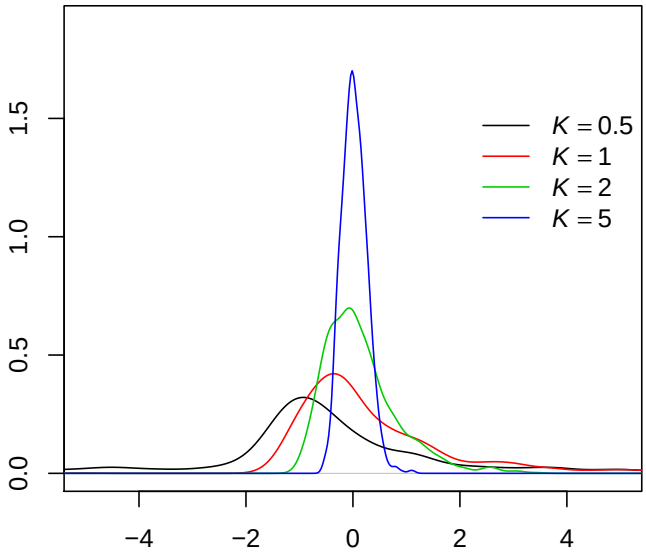


Figure 4.18: Density plots of the difference between the ratio of component parameters at time $t+1$ and t, for $v_t = 0.5$ and several values of K.

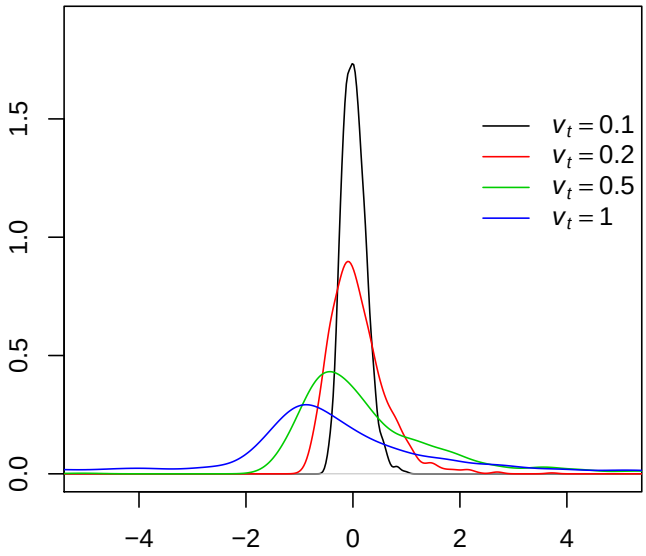


Figure 4.19: Density plots of the difference between the ratio of component parameters at time $t+1$ and t, for $K = 1$ and several values of v_t.

explain why, although using the modified likelihood with parameter $\delta = (0.9, 0.1)$ resulted in less biased estimates than when using $\delta = (1, 0)$, the error rate of the classifier in the latter case was less. From figure 4.16 we can see that the value of K was greater for $\delta = (1, 0)$ (approximately equal to 3) than it was for $\delta = (0.9, 0.1)$ (approximately equal to 1).

However, clearly another way to control the variance introduced by the random noise component of parameter evolution is to change v_t. Figure 4.19 again shows a density plot of the difference $R_{t+1}^{(j)} - R_t$, but in this case K is held constant at $K = 1$ and v_t is varied. In this case we see that the variance of $R_{t+1}^{(j)} - R_t$ increases with increasing v_t when K is held constant.

Therefore, one would think that increasing v_t (which increases the variance of

$R_{t+1}^{(j)} - R_t$) may be able to offset the effect of increasing bias (which decreases the variance of $R_{t+1}^{(j)} - R_t$). This is possibly why the parameter v_t was seen to increase in figure 4.13 when assumed to be unknown and varying. Alternatively, in the cases that we assumed v_t to be known and constant, the SMC algorithm may have been able to regulate v_t indirectly through adjusting the bias. Hence, the bias observed in the stationary scenarios may be a result of v_t being larger than optimal, and the parameter inflation trying to account for this.

However, there is not a simple relationship between the parameter bias K and the value of the parameter evolution variance v_t. What can be said, however, is that this relationship complicates the problem of interpreting the value of v_t used and the observed bias in parameter estimates.

From the observation that no form of trying to reduce the bias led to improved classification performance (and in most cases resulted in worse performance), we can conclude that parameter bias itself does not seem to be a problem. In addition, it seems that in cases where V_t is greater than optimal, parameter bias can even be beneficial. Therefore, it is probably best not to use any form of bias reduction or other likelihood modification.

4.7 Conclusions

In this chapter we presented three algorithms for implementing the classification model introduced in chapter 2. We then compared their performances on both a stationary and a dynamic scenario. In terms of classification performance, the DGLM algorithm of section 4.4.2 was always outperformed by the Gaussian algorithm and the SMC algorithm. Because the assumptions made for the Gaussian algorithm are a special case of the assumptions for the DGLM algorithm, we would

have expected the DGLM to perform at least as well. It is possible that on a problem where the posterior distribution of $\boldsymbol{\beta}_t|\boldsymbol{\beta}_{t-1}$ is highly skewed the DGLM algorithm (which assumes only that the distribution is unimodal) would outperform the Gaussian algorithm (which assumes the distribution is Gaussian).

From figures 4.10(a) and 4.10(b) we can see that the Gaussian and SMC algorithms performed comparably. Therefore, on these problems it would be recommended to use the Gaussian algorithm if v_t is known, as it is much less computationally intensive than the SMC algorithm. However, the SMC algorithm has the advantage that it will in theory cope with any form of distribution for $\boldsymbol{\beta}_t|\boldsymbol{\beta}_{t-1}$, assuming that enough particles are used. In addition, it allows for more flexible modelling assumptions about the form of the parameter evolution variance V_t to be made. Although all algorithms produced some kind of biased parameter estimates, this effect by itself did not seem to affect the error rates of the classification algorithms. However, its relationship to the parameter evolution variance V_t should be further explored.

In conclusion, it seems that for problems where it is likely the assumptions of the Gaussian algorithm approximately hold, this updating method should be used. If it is believed that the distribution of $\boldsymbol{\beta}_t|\boldsymbol{\beta}_{t-1}$ is likely to be skewed or irregular, and computational time is not a significant factor, then the SMC algorithm should be used instead. However, performance of the algorithms was sensitive to the choice of parameter evolution variance. Therefore in cases where there is little information about a suitable choice of V_t, it would seem best to use the SMC algorithm for v_t unknown.

Chapter 5

Conclusions and Ideas for Further Work

In this chapter we draw together the main conclusions of the work presented in this thesis, and suggest areas for further work.

One of the main contributions of this thesis is to understanding the relationship between the component classifiers and the performance of the combined classifier. For the dynamic logistic model presented in chapter 2 (which we showed to be a generalisation of several existing models for classifier combination in stationary scenarios) we proved a bound on the location of the classifier decision boundary as a function of the location of the decision boundaries of the component classifiers. Specifically, we showed that the decision boundary of the combined classifier must lie within the region of disagreement of the component classifiers. Hence a necessary condition for the decision boundary of the combined classifier to be equal to the Bayes boundary, and hence the error rate equal to the Bayes risk, is that the region of disagreement contain the Bayes boundary. A nice property of this result is that no assumptions need to be made about the component classifiers. Further, it applies

to any given set of component classifiers, regardless of the method by which they were trained.

However, because this result is only a bound on the location of the decision boundary of the combined classifier, there is still a lot of uncertainty about the location of this decision boundary within the bounded region. It would be useful to explore further how the shape and flexibility of the decision boundary is affected by the form of $\boldsymbol{\eta}_k(\boldsymbol{x})$ and the number of component classifiers M, so that suitable choices for these can be made based on the expected complexity of the Bayes boundary.

The result discussed above has been derived only from consideration of the model for the conditional class probabilities. For dynamic problems, it is also necessary to consider the influence of the model for parameter evolution. In chapter 2 we defined the model for parameter evolution to be a Gaussian random walk. The main reasons for using this model were because it is commonly used in dynamic modelling, and allows use of the Gaussian updating algorithm. The assumption that the random components have zero mean seems reasonable, which leaves only the variance of the random walk, V_t, to be specified. This is generally recognised as being a difficult yet important problem. In our case this difficulty is enhanced, because the optimal value of V_t will depend not only on changes in the population, but also on the component classifiers which are used. Therefore, the effect of the assumption we adopted that V_t is of the form $v_t I$ will depend on the choice of component classifiers. Although we did not explore this relationship in our work, it is likely to be important to consider when choosing the component classifiers for a dynamic problem.

The simulations we performed in chapter 4 confirmed that the specification of v_t has a significant influence on the error rate of the classifier. If v_t is too small, the

parameter values can not change fast enough to adapt to the changing population, and if v_t is too large the added variance results in increased error. However, when the logistic model is used for classification, it is the ratio of the parameters to one another which determines the decision boundary and hence error rate of the classifier. Therefore, it is the variance of the parameter ratio which contributes to the error rate of the classifier rather than the variance of the parameters themselves.

When comparing the expected parameter values to the optimal values, we observed that the parameter estimates were biased away from zero, and that this bias increased with v_t. In addition, we demonstrated that for fixed v_t increasing this bias will reduce the variance of the parameter ratio. Hence, increasing the bias of the parameter estimates by over-estimating v_t may actually reduce the error rate of the classifier in some cases. More work is needed to understand this relationship more thoroughly, and its effect on specification and interpretation of v_t.

A further contribution of the work in this thesis is the adoption of the predictive approach to dealing with the unknown parameters of the classifier combination, and the comparison in this context of several sequential updating algorithms for classifying observations. The Gaussian and SMC updating algorithms consistently out-performed the DGLM algorithm on the problems we considered. The Gaussian algorithm has the advantage that updates are faster to perform than for the SMC algorithm, but the disadvantage that V_t is assumed to be known. On the other hand, the SMC algorithm has the advantage that V_t can be incorporated into the vector of unknown parameters, but it is more computationally demanding, particularly as the dimension of the feature space or the number of parameters is increased. It would be interesting to know how the proposed algorithms perform on more difficult, real problems, and in comparison to other dynamic classification algorithms. In addition, more work is needed to compare the performance of the

SMC and Gaussian updating algorithms on higher-dimensional problems, and scenarios where the distribution of $\boldsymbol{\beta}_t|\boldsymbol{\beta}_{t-1}$ is not well approximated by a Gaussian distribution.

The classifier we proposed is an extension to the dynamic scenario of several model-based methods for combining classifier outputs. Analysis of this model has helped us to better understand the influence of the set of component classifiers on the performance of the final classifier. However, there are many aspects relating to specification of the model components which must be considered and understood before the model can be effectively implemented. Further work is required to understand the influence of these components and how they interact, in order that the limits of the classifier are known and to guide effective strategies for implementation on real-world problems.

Bibliography

J. Ameen & P. Harrison (1985). 'Normal Discount Bayesian Models'. In J. Bernado, M. DeGroot, D. Lindley, & A. Smith (eds.), *Bayesian Statistics 2: Proceedings of the Second Valencia International Meeting*, pp. 271–298. Elsevier Science.

P. Bartlett, et al. (2000). 'Learning Changing Concepts by Exploiting the Structure of Change'. *Machine Learning* **41**:153–174.

L. Breiman (1993). 'Stacked Regression'. Tech. Rep. 367, Dept of Statistics, University of California, Berkeley.

L. Breiman (1996). 'Bagging Predictors'. *Machine Learning* **24**:123–140.

P. Cunningham, et al. (2003). 'A case-based approach to spam filtering that can track concept drift'. Tech. Rep. TCD-CS-2003-16, Trinity College Dublin.

J. de Freitas, et al. (2000a). 'Hierarchical Bayesian Models for Regularization in Sequential Learning'. *Neural Computation* **12**:933–953.

J. de Freitas, et al. (2000b). 'Sequential Monte Carlo Methods to Train Neural Network Models'. *Neural Computation* **12**:955–993.

A. Doucet, et al. (2000). 'On sequential Monte Carlo sampling methods for Bayesian filtering'. *Journal for Statistics and Computing* **10**:197–208.

D. Duffy & T. Santner (1989). 'On the small sample properties of norm-restricted maximum likelihood estimators for logistic regression models'. *Communications in Statistics - Theory and Methods* **18**(3):959–980.

D. Firth (1993). 'Bias reduction of maximum likelihood estimates'. *Biometrika* **80**(1):27–38.

Y. Freund & Y. Mansour (1997). 'Learning Under Persistent Drift'. In *Proceedings of the third European conference on Computational Learning Theory*, vol. 1208, pp. 109–118. Springer.

Y. Freund & R. Schapire (1997). 'A decision-theoretic generalization of online learning and an application to boosting'. *Journal of Computer and Systems Sciences* **55**:119–139.

B. Fritzke (1997). 'A self-organizing network that can follow non-stationary distributions'. In *Proceedings of ICANN-97, International Conference on Artificial Neural Networks*, pp. 613–618. Springer.

G. Fumera & F. Roli (2002). *Performance Analysis and Comparison of Linear Combiners for Classifier Fusion*, vol. 2396/2002 of *Lecture Notes in Computer Science*, pp. 47–64. Springer.

G. Fumera & F. Roli (2003). *Linear Combiners for Classifier Fusion: Some Theoretical and Experimental Results*, vol. 2709/2003 of *Lecture Notes in Computer Science*, pp. 74–83. Springer.

G. Fumera & F. Roli (2005). 'A theoretical and experimental analysis of linear combiners for multiple classifier systems'. *IEEE Trans. on Pattern Analysis and Machine Intelligence* **27**(6):942–956.

A. Genz & F. Bretz (2002). 'Comparison of Methods for the Computation of Multivariate t-Probabilities'. *Journal of Computational and Graphical Statistics* **11**(4):950–971.

J. Geweke (1989). 'Bayesian inference in econometric models using Monte Carlo integration'. *Econometrica* **57**(6):1317–1339.

W. Gilks & C. Berzuini (2001). 'Following a moving target - Monte Carlo inference for dynamic Bayesian models'. *Journal of the Royal Statistical Society B* **63**:127–146.

L. Hansen & P. Salamon (1990). 'Neural Network Ensembles'. *IEEE Trans. on Pattern Analysis and Machine Intelligence* **12**:993–1001.

P. Harrison & C. Stevens (1976). 'Bayesian Forecasting'. *Journal of the Royal Statistical Society B* **38**(3):205–247.

D. Helmbold & P. Long (1994). 'Tracking Drifting Concepts by Minimizing Disagreements'. *Machine Learning* **14**(1):27–45.

P. Højen-Sørensen, et al. (2000). 'On-line probabilistic classification with particle filters'. *Neural Networks for Signal Processing X, 2000. Proceedings of the 2000 IEEE Signal Processing Society Workshop* **1**:386–395.

G. Hulten, et al. (2001). 'Mining time-changing data streams'. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 97–106.

G. James (2003). 'Variance and Bias for General Loss Functions'. *Machine Learning* **51**:115–135.

M. Kelly, et al. (1999). 'The Impact of Changing Populations on Classifier Performance'. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 367–371.

E. Kleinberg (1990). 'Stochastic Discrimination'. *Annals of Mathematics and Artificial Intelligence* **1**:207–239.

A. Krogh & J. Vedelsby (1995). 'Neural Network Ensembles, Cross Validation and Active Learning'. *Advances in Neural Information Processing Systems* **7**:231–238.

L. Kuncheva (2003). 'That Elusive Diversity in Classifier Ensembles'. In *Pattern Recognition and Image Analysis*, vol. 2652/2003 of *Lecture Notes in Computer Science*, pp. 1126–1138. Springer.

L. Kuncheva (2004). 'Classifier Ensembles for Changing Environments'. In F. Roli, J. Kittler, & T. Windeatt (eds.), *Proc. 5th Int. Workshop on Multiple Classifier Systems, Cagliari, Italy*, pp. 1–15. Springer.

L. Kuncheva & C. Whitaker (2003). 'Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy'. *Machine Learning* **51**(2):181–207.

M. LeBlanc & R. Tibshirani (1996). 'Combining estimates in regression and classification'. *Journal of the American Statistical Association* **91**:1641–1650.

N. Littlestone & M. Warmuth (1989). 'The Weighted Majority Algorithm'. In *Proceedings, 30th Annual Symposium on Foundations of Computer Science*, pp. 256–261.

W. Penny & S. Roberts (1999). 'Dynamic Logistic Regression'. In *Proceedings, International Joint Conference on Neural Networks*, vol. 3, pp. 1562–1567.

M. Perrone (1993). *Improving Regression Estimation: Averaging methods for variance reduction with extensions to general convex measure optimization.* Ph.D. thesis, Brown University.

M. Perrone & L. Cooper (1992). 'When networks disagree: Ensemble methods for hybrid neural networks'. Tech. Rep. A540062, Institute for Brain and Neural Systems, Brown University.

B. Ripley (1996). *Pattern Recognition and Neural Networks.* Cambridge University Press.

P. Rousseeuw & A. Christmann (2003). 'Robustness against separation and outliers in logistic regression'. *Computational Statistics and Data Analysis* **43**(3):315–332.

J. Schlimmer & R. Granger (1986). 'Incremental Learning from Noisy Data'. *Machine Learning* **1**(3):317–354.

D. Spiegelhalter & S. Lauritzen (1990). 'Sequential Updating of Conditional Probabilities on Directed Graphical Structures'. *Networks* **20**(5):579–605.

K. Tumer & J. Ghosh (1996). 'Analysis of Decision Boundaries in Linearly Combined Neural Classifiers'. *Pattern Recognition* **29**:341–348.

M. West & J. Harrison (1997). *Bayesian Forecasting and Dynamic Models*. Springer, 2nd edn.

M. West, et al. (1985). 'Dynamic Generalized Linear Models and Bayesian Forecasting'. *Journal of the American Statistical Association* **80**(389):73–83.

G. Widmer & M. Kubat (1996). 'Learning in the Presence of Concept Drift and Hidden Contexts'. *Machine Learning* **23**(1):69–101.

D. Wolpert (1992). 'Stacked Generalization'. *Neural Networks* **5**:241–259.