University of Exeter

Department of Computer Science

A Bayesian expected cost reduction approach to active learning

Richard Fredlund

April 2011

Submitted by Richard Fredlund, to the University of Exeter as a thesis for the degree of Doctor of Philosophy in Computer Science, April 2011.

This thesis is available for Library use on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

I certify that all material in this thesis which is not my own work has been identified and that no material has previously been submitted and approved for the award of a degree by this or any other University.

(signature) .....

Declaration

Chapter 3 is partially based on the previously published work:

"A Bayesian framework for active learning" in proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCNN 2010) by R.M. Fredlund, R.M. Everson and J.E. Fieldsend.

Richard Fredlund

To my son Daniel Fredlund who is never far from my mind.

Acknowledgements

I would especially like to thank my supervisors Richard Everson and Jonathan Fieldsend who always managed to exceed my expectations. I would also like to thank the various friends and family who were helpful and supportive along the way.

Abstract

There has been growing recent interest in the field of active learning for binary classification. This thesis develops a Bayesian approach to active learning which aims to minimise the objective function on which the learner is evaluated, namely the expected misclassification cost. We call this approach the expected cost reduction approach to active learning. In this form of active learning queries are selected by performing a 'lookahead' to evaluate the associated expected misclassification cost.

Firstly, we introduce the concept of a *query density* to explicitly model how new data is sampled. An expected cost reduction framework for active learning is then developed which allows the learner to sample data according to arbitrary query densities. The model makes no assumption of independence between queries, instead updating model parameters on the basis of both which observations were made *and* how they were sampled. This approach is demonstrated on the probabilistic high-low game which is a non-separable extension of the high-low game presented by Seung et al. [1992]. The results indicate that the Bayes expected cost reduction approach performs significantly better than passive learning even when there is considerable overlap between the class distributions, covering 30% of input space. For the probabilistic high-low game however narrow queries appear to consistently outperform wide queries. We therefore conclude the first part of the thesis by investigating whether or not this is always the case, demonstrating examples where sampling broadly is favourable to a single input query.

Secondly, we explore the Bayesian expected cost reduction approach to active learning within the pool-based setting. This is where learning is limited to a finite pool of unlabelled observations from which the learner may select observations to be queried for class-labels. Our implementation of this approach uses Gaussian process classification with the expec-

tation propagation approximation to make the necessary inferences. The implementation is demonstrated on six benchmark data sets and again demonstrates superior performance to passive learning.

Contents

1. Introduction

One of the central problems in machine learning is the classification of data. The simplest type of classification is binary classification where an algorithm (learner) attempts to label observations into one of two classes. In the training phase the learner is presented with a set of labelled examples, called *training data*. Having learned from the training data the learner then goes on to make inferences about the class labels of new unlabelled observations called *test data*.

While unlabelled observations, which we call *inputs*, are often free or inexpensive providing class labels for the training data may incur a cost; either monetary or in terms of resources. An example where providing labels for training data is costly is determining whether a biopsy slide is cancerous or not. Here the time of an expert is required to label each slide by hand. In the extreme case of predicting where to drill for oil, training data requires actually drilling down to see if there is oil or not. Hence, we are often limited to existing data, or the amount of data we can afford to label.

The traditional approach to selecting training data for classification, called *passive* learning, selects inputs for labelling which are independently and identically drawn (IID) from the underlying distribution of the data. However, it may be the case that the class label of one input is more informative than another. *Active* learning seeks to minimise the labelling cost by allowing the learner to select which inputs to query for class labels. There has therefore been growing interest in the field of *active* learning and a number of algorithms and heuristics have recently been developed; prominent references are: [Seung et al., 1992, Lewis and Gale, 1994, Roy and McCallum, 2001, Tong and Koller, 2000, Schohn and Cohn, 2000, Campbell et al., 2000, Baram et al., 2004].

We first introduce two approaches which have been predominant in the field of active learning before discussing the expected cost reduction approach to active learning which is advanced in this thesis.

Uncertainty sampling [Lewis and Gale, 1994] is one of the most intuitive approaches to active learning. Here the learner has some measure of confidence in its predicted class label and queries the input about which it is currently most uncertain. The idea being that inputs for which the learner is already relatively certain of the class label are likely to be less informative. Unfortunately there are also inputs about which the learner may be uncertain, but are not informative. For example, a queried input may be an outlier, meaning that it is not close in observation space to other inputs and is therefore uninformative about the overall decision boundary. Uncertainty sampling has the advantage of being computationally efficient and easy to implement. It tends to perform well and has demonstrated significant improvement over passive learning in a number of settings.

The *query by committee*(QBC) approach to active learning was introduced by Seung et al. [1992]. QBC is based on the assumption that data is *separable*, meaning that there exists a decision boundary within the model which completely separates the two classes and would always assign the correct class label to any unlabelled input. In this ideal situation the 'true' decision boundary will be consistent with every class label observed. The set of solutions which are consistent with all previous observations is called *version space*. In this context each new labelled input has the potential to eliminate a portion of version space because it rules out some decision boundaries which are not consistent with the labelled input and active learning can be seen as the process of selecting queries which maximally reduce version space. The best query here is one which precisely halves version space. The QBC approach views each possible decision boundary as a committee member and the halving of version space happens when an input is predicted as being in one class, by as many committee members in version space, as predict it being in the other class. Clearly when there is disagreement among the committee regarding the class label of a given input the input has the potential to reduce version space regardless of which class label is assigned. The QBC approach uses the principle of 'maximal disagreement'

in order to select new queries in order to minimise the volume of version space at each step. This is done by sampling committee members from version space and selecting for query the input for which there is the most disagreement. Somewhat surprisingly this approach is robust to the number of committee members sampled and even a committee of two randomly drawn committee members performs reasonably well.

The QBC assumption that the data is separable is a strong assumption which in general is not true. This is because there will often be some overlap between class distributions preventing 'perfect' classification. However, it has presented a useful starting point for approaches in active learning for two reasons. Firstly, the QBC approach allows strong theoretical results for active learning [Seung et al., 1992, Freund et al., 1997]. Secondly, the QBC approach has been successfully extended to loosen the separability assumption. This has been done in two directions. It has inspired support vector machine (SVM) approaches to active learning such as [Schohn and Cohn, 2000, Campbell et al., 2000, Tong and Koller, 2000] where a 'soft' margin is used to loosen the separability assumption. In the other direction the QBC approach has also inspired approaches such as [Balcan et al., 2006, Argamon-Engelson and Dagan, 1999] which attempt to extend the QBC approach to the probabilistic setting.

1.1. Expected cost reduction

In active learning there is a trade-off between minimising the number of labels required while maximising performance of the classifier on the unseen test data. In contrast to uncertainty sampling and the QBC approach, the expected cost reduction approach to active learning selects queries which are most likely to improve performance of the classifier. This approach is *greedy* in the sense that the query selected is the one which immediately most maximises the performance of the learner.

In this thesis the empirical measure of performance for a classifier is the expected misclassification cost. In general the exact misclassification cost will depend on the nature of the data being labelled and need not be a financial cost; rather it corresponds to the set of outcomes associated with misclassification, both financial and otherwise. Therefore

the issue of evaluating the actual costs associated with misclassification are complex and problem specific and in practice a fixed objective function called a cost or loss function is usually defined. The most common cost function is the $0-1$ loss function, which simply assigns equal cost to any misclassified data. In this thesis we also consider the cost function which allows a different cost to be assigned for misclassified inputs in one class than for misclassified inputs in the other. We do not assign any cost to correctly assigned inputs. The expected misclassification cost refers to the average misclassification cost over all unlabelled observations in the test set.

For the purposes of an active learner in the expected cost reduction approach to active learning, performance is measured by a cost function. This cost function is the objective function used by the learner to approximate the expected misclassification cost. Implementation of this approach requires a classifier which is capable of producing a probability of class membership. For each potential query the learner must update its beliefs about the probabilities of class membership before using these updated beliefs to estimate the expected misclassification cost, according to the chosen cost function. While any probabilistic classifier may be used for this approach, the estimate of the expected misclassification cost is dependent on the accuracy with which these probabilities are updated and a Bayesian classifier represents the ideal choice.

The first authors to suggest the expected cost reduction approach to active learning were Roy and McCallum [2001], who implemented the cost reduction approach using naïve Bayes classifiers. Naïve Bayes classifiers make a strong independence assumption about the data, updating beliefs as though the observations were IID. This assumption is clearly not true in active learning where observations are selected *actively* rather than *passively* from the underlying distribution. Despite this the Roy and McCallum [2001] implementation has been shown to perform well and was cited by Baram et al. [2004], who dubbed the algorithm SELF-CONF, as being one of the two top performing active learning algorithms.

Other classifiers have also been applied to the cost reduction approach, logistic regression [Guo and Greiner, 2007] and Gaussian random fields [Zhu et al., 2003]. However only Zhu

et al.'s work could be said to be a Bayesian approach.

In this thesis we present an expanded framework for the expected cost reduction approach to active learning which allows the learner to select not only which observations to query but also how each new observation is sampled from the underlying distribution. This approach is largely theoretical, being difficult to implement in practice for anything other than quite simple problems. We therefore also present a pool-based implementation of the cost reduction approach to active learning using Gaussian processes to perform the necessary Bayesian inferences.

1.2. Outline of thesis

Chapter 2 presents the necessary background and methodology for the thesis before reviewing previous work in the field of active learning. We start by introducing the context, which is binary classification within the supervised learning setting (section 2.1) before describing maximum likelihood (ML), which is a commonly used point estimate approach to parameter estimation (section 2.1.1). The Bayesian approach to binary classification is then described (section 2.1.2). The advantages of the Bayesian approach over point estimation are discussed and various commonly used methods for approximating the necessary posterior distributions are briefly described; namely, Monte Carlo approaches and variational Bayes.

Chapter 3 presents a general Bayesian framework for active learning in which the learner selects an arbitrary query density from which to sample new observations. In this scheme the learner requests new observations from a *sample oracle* which samples unlabelled inputs from the underlying distribution according to the query density defined by the learner. The main advantage of this framework is that it incorporates both passive learning and *full* active learning (where the label for a specific input is requested by the learner, without reference to the underlying distribution) within the same Bayesian scheme. The only difference between active and passive learning in this context being the choice of query density used.

The framework is implemented in section 3.3 on the probabilistic high-low game, which is a non-separable extension of the high-low game presented by Seung et al. [1992]. However, the optimal query density in this case appears to be a Dirac delta query, which selects a specific query without reference to the underlying distribution. This therefore presents the question: "*are point-queries always the most informative query or are there instances where a wide query is favourable?*"

Section 3.5 shows two examples where a query somewhere between either extreme is optimal. That is there are optimal queries which are neither passive nor *fully* active.

The approach laid out in chapter 3 is a general framework which is capable of encompassing both active and passive learning approaches. Because of the integrals involved in completing the Bayesian update, it is, however, computationally too expensive for models of any significant complexity.

Chapter 4 presents the Bayesian cost reduction approach in the pool-based setting for active learning, which we then implement using Gaussian processes. This is seen as a simplification of the model presented in chapter 3. We start by examining what happens to the model in chapter 3 when the query density is narrowed to a single point (section 4.2). In the pool-based setting all available information regarding the underlying distribution of inputs is contained within the pool of unlabelled inputs. This presents further simplifications to the model, which are described in section 4.3. The implementation of the expected cost reduction algorithm for active learning in the pool-based setting is then described in section 4.4. This is presented for any probabilistic classifier capable of estimating the necessary class conditional probabilities. We apply this algorithm in the Bayesian setting using Gaussian processes classification. Gaussian processes and the specific classifier used are described in section 4.4, before showing results on a number of benchmark data-sets in section 4.6. Two performance measures for active learning are then used to summarise these results (section 4.7). The chapter is then summarised in section 4.9.

Chapter 5 presents a discussion of the approaches taken, suggesting ways in which the query density framework of chapter 3 might be made more feasible. Methods of extending

the implementation in chapter 4 to larger data sets are also discussed. Finally section 5.3 makes some closing remarks.

2. Background and Related Work

This Chapter is split into two main sections. Section 2.1 introduces methods for learning parametrised models, in particular Maximum Likelihood and Bayesian inference. Section 2.2 reviews some of the literature on active learning, setting a broader context for the thesis.

2.1. Supervised Learning

A common machine learning task is to predict, or forecast, an outcome on the basis of some data. This data set may be any set of pertinent measurements or observations, on which the algorithm is expected to make predictions. The data that an algorithm uses as input is entirely dependent on the particular task at hand. There are many different areas in which machine learning can be applied, and each one requires its own relevant data. Examples of input data used by machine learning algorithms include: images of faces for face recognition [Li and Jain, 2005]; sound bites, for voice recognition [Rabiner, 1989]; and word frequencies in document classification tasks [Sebastiani and Ricerche, 2002]. Whatever form they take, we can encode this input data so that the set of observations can be represented by a single vector $\mathbf{x}$. Depending on the data, these values may be discrete, continuous or some combination of the two. In some cases $\mathbf{x}$ may simply be scalar.

Similarly, the type of prediction made by an algorithm falls into two main categories. Firstly, *classification* tasks, where the aim is to classify the observation into one of two or more categories. For example in medical imaging we may be attempting to label biopsy slides as being in one of two classes, 'cancer' or 'not cancer'. When there are only two possible classes this is called *binary classification.* An example of a *multi-class classification*

problem, in which there are more than two classes, is English character recognition, where an algorithm tries to determine which one of 26 letters has been written. In this case the predicted output will be one of a finite set of discrete values. The second type of prediction made is *regression*, where the *target* being predicted is some continuous value. For example the learner may be trying to predict life expectancy, height, or the volume of water flow in a river. In each of these cases the target predicted by the learner is a continuous scalar value y. In classification the targets are called *class labels*. We may denote each class label by a specific discrete value, so that the target y, which in this case is the class label, may be represented by a single scalar value, taking on only these discrete values. In binary classification, for example, the two classes are usually denoted as class 0 and class 1. In regression y will usually be scalar, for example when trying to predict height, or life expectancy. In general y could be a vector as in the case where we are trying to predict more than one value at the same time, however, this thesis is principally concerned with binary classification, so we will take y to be a scalar labelling the two classes.

In the learning phase the learner attempts to discover a mapping from input vectors $\mathbf{x}$ to the corresponding target y. Learning falls into three main categories, *unsupervised learning* [eg Hinton and Sejnowski, 1999], *reinforcement learning* [eg Sutton and Barto, 1999] and *supervised learning* [eg Kotsiantis, 2007].

In unsupervised learning the learner is not given any information about the true target values; for example, some classification problems can be learned without any feedback. Instead the algorithm attempts to find structure in the data, for example classifying data on the basis of how that data is clustered. These sort of algorithms are often referred to as clustering algorithms and are self organizing. Most problems cannot be solved in this way, and for the algorithm to learn to make predictions effectively it is necessary to train the algorithm in some way.

In reinforcement learning [Kaelbling et al., 1996, Sutton and Barto, 1999] there is feedback given to the algorithm but it is not necessarily direct. The feedback usually takes the form of some cost function or reward which indicates when it is performing well or

when the learner is performing badly. However, the true target values are not given to the algorithm. In this case the algorithm must learn on the basis of past feedback. A good example of reinforcement learning is motor control for a robot arm [Peters et al., 2007], where performance of some task (like picking up a ball), can be fed back to the machine but the exact motor actions necessary to grip the ball must be inferred. Both unsupervised learning and reinforcement learning are major topics in machine learning, but are not the subject of this thesis.

Finally, in supervised learning the algorithm is given direct feedback about the true target values during the training phase. There are many instances where it is possible to provide the algorithm with some training data for which the outcomes are known. In general this *training data* $\mathcal{D}$ consists of N *inputs*, or observations $\mathbf{x}_i$ together with their corresponding targets, or class labels, y_i:

$$\mathcal{D} = \{\mathbf{x}_1, y_1, \ldots, \mathbf{x}_N, y_N\}. \tag{2.1}$$

For example, if we are trying to predict the height of a river on the basis of some observations, then we might compile a set of N observation vectors $\mathbf{x}_i$, each containing pertinent measurements such as the river's width, the rate of flow, soil type, slope of the river bank, monthly rainfall etc, together with a target vector y_i, containing the actual depth of the river corresponding to $\mathbf{x}_i$.

It is often the case that training data for supervised learning is costly to produce, because it requires knowing the true target values. For example, in the binary classification task of deciding whether a biopsy slide is cancerous or not, training data requires the time of an expert to correctly label some data by hand. In the case of predicting life expectancy data takes a long time to collect and we may well be limited to existing data. In the extreme case of predicting where to drill for oil, training data requires actually drilling down to see if there is oil or not. Hence, we are often limited to existing data, or the amount of data we can afford to collect.

In general for supervised learning we have some function $y = F(\mathbf{x}, \boldsymbol{\theta})$ which models y for a given $\mathbf{x}$ and depends on some parameters $\boldsymbol{\theta}$. In the training phase, the algorithm is given some *training data* $\mathcal{D} = \{\mathbf{x}_1, y_1, \ldots, \mathbf{x}_N, y_N\}$ which contains a set of *inputs* $\mathbf{x}_i$ and their corresponding targets y_i. In non-Bayesian schemes a specific parametrization $\boldsymbol{\theta}_*$ is 'learned' from the training data, usually by minimising an error function measuring the discrepancy between y_i and $F(\mathbf{x}_i, \boldsymbol{\theta})$. Given the learned parameters, $F(\mathbf{x}, \boldsymbol{\theta}_*)$ can then be used to predict the target y on some new previously unseen *test data* during the test phase. The exact form of $F(\mathbf{x}, \boldsymbol{\theta})$ depends on the supervised learning algorithm being used. There are many different supervised learning algorithms, each with a different functional form $F(\mathbf{x}, \boldsymbol{\theta})$ and different methods of learning $\boldsymbol{\theta}$ from the training data. For example, $F(\mathbf{x}, \boldsymbol{\theta})$ might be a multilayer perceptron (MLP), a radial basis function (RBF) network, a logistic function of a linear combination of the inputs (in the case of logistic regression), a support vector machine (SVM), or any other supervised learning algorithm. How $\boldsymbol{\theta}$ is adjusted, based on the training data, depends on the algorithm being used.

In general performance is measured by a *cost function* $\mathcal{C}(F(\mathbf{x}, \boldsymbol{\theta}), y)$. Depending on the context, this is also called the *error function* or *risk function* and compares function predictions $F(\mathbf{x}, \boldsymbol{\theta})$ with the true targets y on the (previously unseen) test data.

One of the important questions in supervised learning is, given the limited set of training data, how best to generalise from this training data in order to predict labels to previously unseen values of $\mathbf{x}$, called *test data*. If our function $F(\mathbf{x}, \boldsymbol{\theta})$ is too complex then it may fit the training data well, but generalise poorly. This is because it will tend to over fit the training data, learning any noise, and not just the underlying structure of the data. A less complicated model may not fit the training data as well but may generalise better. This trade-off between complexity and generalization is one of the central themes of supervised learning, and applies in both regression and classification [Bishop, 2006, pp 6].

In the next section we look at one statistically principled way in which the parameters $\boldsymbol{\theta}$ might be learned.

2.1.1. Maximum likelihood

Maximum likelihood (ML), is an approach, based on the question, 'given some training data $\mathcal{D} = \{\mathbf{x}_1, y_1, \ldots, \mathbf{x}_N, y_N\}$, which parameters $\boldsymbol{\theta}$ are most likely to have produced these observations?'. To answer this question we need to model $p(\mathcal{D} \,|\, \boldsymbol{\theta}) = p(y_1, \mathbf{x}_1, \ldots, y_N, \mathbf{x}_N \,|\, \boldsymbol{\theta})$, the likelihood of observing $\mathcal{D}$ given $\boldsymbol{\theta}$. The ML parameters are then those that maximise this likelihood, $\boldsymbol{\theta}_{ML} = \text{argmax}_{\boldsymbol{\theta}}\, p(\mathcal{D} \,|\, \boldsymbol{\theta})$. However, the distribution $p(y_1, \mathbf{x}_1, \ldots, y_N, \mathbf{x}_N \,|\, \boldsymbol{\theta})$ is a joint distribution across all observed data $\mathcal{D} = \{\mathbf{x}_1, y_1, \ldots, \mathbf{x}_N, y_N\}$ and in general may be difficult to evaluate.

When the input-target pairs $(\mathbf{x}_i, y_i)$ are independent of each other, and drawn from the same fixed but unknown distribution $p(\mathbf{x}, y)$ it is said that $(\mathbf{x}, y)$ are independently and identically distributed (IID). In this case the joint density of observing the data can be modelled as the product of $p(y_i, \mathbf{x}_i \,|\, \boldsymbol{\theta})$ for each $\mathbf{x}_i$. The density $p(y_i, \mathbf{x}_i \,|\, \boldsymbol{\theta})$ can be further simplified by noting that the target value y_i depends only on the particular value of $\mathbf{x}_i$ to which it corresponds, and so $p(y_i, \mathbf{x}_i \,|\, \boldsymbol{\theta}) = p(y_i \,|\, \mathbf{x}_i, \boldsymbol{\theta}) p(\mathbf{x}_i \,|\, \boldsymbol{\theta})$ for a given $\boldsymbol{\theta}$. Putting these together we have:

$$p(y_1, \mathbf{x}_1, \ldots, y_N, \mathbf{x}_N \,|\, \boldsymbol{\theta}) = \prod_{n=1}^{N} p(y_n \,|\, \mathbf{x}_n, \boldsymbol{\theta}) p(\mathbf{x}_n \,|\, \boldsymbol{\theta}) \tag{2.2}$$

However, it is sometimes easier to work with the log of the likelihood which has the advantage of being a summation rather than a product. When the likelihood is an exponential, taking logs also has the advantage of simplify the resulting equations. Equation (2.2) then becomes:

$$\log\left(p(y_1, \mathbf{x}_1, \ldots, y_N, \mathbf{x}_N \,|\, \boldsymbol{\theta})\right) = \sum_{n=1}^{N} \log(p(y_n \,|\, \mathbf{x}_n, \boldsymbol{\theta}) p(\mathbf{x}_n \,|\, \boldsymbol{\theta})) \tag{2.3}$$

$$= \sum_{n=1}^{N} \log(p(y_n \,|\, \mathbf{x}_n, \boldsymbol{\theta})) + \sum_{n=1}^{N} \log(p(\mathbf{x}_n \,|\, \boldsymbol{\theta})) \tag{2.4}$$

This works because the log function is monotonic, so maximizing the log-likelihood in equation (2.3) is equivalent to maximizing the original likelihood in equation (2.2). Notice that only the first term in (2.4) is dependent on the observed targets. Hence, finding the maximum likelihood for the joint distribution $p(y_1, \mathbf{x}_1, \ldots, y_N, \mathbf{x}_N \,|\, \boldsymbol{\theta})$ is equivalent

to finding the maximum log-likelihood for the conditional distribution $p(y_n \,|\, \mathbf{x}_n, \boldsymbol{\theta})$ and the underlying distribution $p(\mathbf{x}_n \,|\, \boldsymbol{\theta})$ separately. In many cases we only model the conditional distribution $p(y_n \,|\, \mathbf{x}_n, \boldsymbol{\theta})$ and it is enough simply to maximise $\sum_{n=1}^{N} \log(p(y_n \,|\, \mathbf{x}_n, \boldsymbol{\theta}))$; equivalently often an error function $\mathcal{E}(\boldsymbol{\theta}) = -\sum_{n=1}^{N} \log(p(y_n \,|\, \mathbf{x}_n, \boldsymbol{\theta}))$ is minimised.

Kullback-Leibler divergence

The Kullback-Leibler divergence, which is otherwise known as the KL-divergence or the relative entropy, is a measure of the divergence between two probability density functions, $p(\mathbf{z})$ and $q(\mathbf{z})$. For example, in the context of maximum likelihood, $\mathbf{z}$ denotes an input-label pair $(\mathbf{x}, y)$. KL-divergence was first introduced by Kullback and Leibler [1951] and later by Kullback [1959]. It has its roots in information theory [Cover and Thomas, 1991] and is defined as:

$$D_{KL}(p \,||\, q) = -\int p(\mathbf{z}) \log \big(\frac{q(\mathbf{z})}{p(\mathbf{z})}\big) d\mathbf{z}. \tag{2.5}$$

Here the vertical bars are representative of a *divergence*. KL-divergence is like a measure of distance in that it is always non-negative, with $D_{KL}(p \,||\, q) \equiv 0$ if and only if the two distributions are the same almost everywhere. However, unlike a distance measure it is not symmetric, because in general $D_{KL}(p \,||\, q) \neq D_{KL}(q \,||\, p)$, and it does not obey the triangle inequality.

In coding theory, if $p(\mathbf{z})$ is the true distribution from which some data $\mathcal{D}_{\mathbf{z}}$ are drawn and $q(\mathbf{z})$ is an approximation to $p(\mathbf{z})$, then the KL-divergence, $D_{KL}(p \,||\, q)$ tells us something about the compression rates achievable when using $q(\mathbf{z})$, rather than $p(\mathbf{z})$, to compress $\mathcal{D}_{\mathbf{z}}$. The Shannon optimal compression rate, when using the true distribution $p(\mathbf{z})$, is given by the entropy $H[p(\mathbf{z})] \coloneqq -\int p(\mathbf{z}) \log p(\mathbf{z}) d\mathbf{z}$. The more dissimilar $q(\mathbf{z})$ is to $p(\mathbf{z})$ the worse the compression rate will be and the KL-divergence corresponds to the minimum extra information which would be required to encode using $q(\mathbf{z})$, rather than $p(\mathbf{z})$.

There is a relationship between maximum likelihood and KL-divergence. If $p(\mathbf{z})$ is to be modelled by some family of distributions $q(\mathbf{z} \,|\, \boldsymbol{\theta})$, then maximum likelihood on $\boldsymbol{\theta}$ will tend to minimize the KL-Divergence between $p(\mathbf{z})$ and $q(\mathbf{z} \,|\, \boldsymbol{\theta})$, when the number of observations

is large [Eguchi and Copas, 2006]. The reason for this is that the KL-Divergence is defined by (2.6) which can be rearranged to give (2.7):

$$D_{KL}(p \,||\, q) = -\int p(\mathbf{z}) \log \big(\frac{q(\mathbf{z} \,|\, \boldsymbol{\theta})}{p(\mathbf{z})}\big) d\mathbf{z} \tag{2.6}$$

$$= \int p(\mathbf{z}) \log(p(\mathbf{z})) d\mathbf{z} - \int p(\mathbf{z}) \log(q(\mathbf{z} \,|\, \boldsymbol{\theta})) d\mathbf{z}. \tag{2.7}$$

The first term in (2.7) is dependent only on $p(\mathbf{z})$ and is unaffected by maximizing over $\boldsymbol{\theta}$. The second term is approximated by the average log likelihood of the sample, from the data:

$$\int p(\mathbf{z}) \log(q(\mathbf{z} \,|\, \boldsymbol{\theta})) d\mathbf{z} \approx \frac{1}{N} \sum_{i=1}^{N} \log(q(\mathbf{z}_i \,|\, \boldsymbol{\theta})). \tag{2.8}$$

The summation is the likelihood of observing the data $\mathcal{D} = \{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$ sampled from $p(\mathbf{z})$, for a given $\boldsymbol{\theta}$. At the limit as $N \to \infty$ this approximation becomes equality. Therefore, at the limit, selecting $\boldsymbol{\theta}$ to maximize the log-likelihood, from the summation in (2.8) will also minimize the KL-divergence $D_{KL}(p \,||\, q)$. This implies that:

1. For a *closed* model, where there exists a parametrization θ such that $p(\mathbf{z}) = q(\mathbf{z} \,|\, \boldsymbol{\theta})$, then the ML estimate will equal the true distribution $p(\mathbf{z}) = q(\mathbf{z} \,|\, \boldsymbol{\theta}_{ML})$.

2. For an *open* model, where the true distribution lies outside the parametrized model, the KL-divergence $D_{KL}(p(\mathbf{z}) \,||\, q(\mathbf{z} \,|\, \boldsymbol{\theta}_{ML}))$ is minimised.

This is reassuring because, in the limit of an infinite number of observations, the ML estimate $\boldsymbol{\theta}_{ML}$ of the parametrized model $q(\mathbf{z} \,|\, \boldsymbol{\theta}_{ML})$ results in a distribution which is in some sense 'close' to the true distribution $p(\mathbf{z})$.

2.1.2. Bayesian approach

In a Bayesian setting we model not just $F(\mathbf{x}, \boldsymbol{\theta})$, but also our prior beliefs about the parameters $\boldsymbol{\theta}$, which are now treated as random variables. The parameters are modelled by a prior distribution $p(\boldsymbol{\theta})$, which encodes the modeller's beliefs about the parameters before the data are observed [Bernardo and Smith, 1994]. Bayes theorem, which is one of the cornerstones of machine learning, then allows us to update this prior belief about $\boldsymbol{\theta}$

on the basis of new data. In its simplest form this is:

$$\underbrace{p(\boldsymbol{\theta} \,|\, \mathcal{D})}_{Posterior} \propto \underbrace{p(\mathcal{D} \,|\, \boldsymbol{\theta})}_{Likelihood} \underbrace{p(\boldsymbol{\theta})}_{Prior} \tag{2.9}$$

Put simply, the *prior* belief about the parameters $p(\boldsymbol{\theta})$ multiplied by the *likelihood* of observing the data given the parameters $p(\mathcal{D} \,|\, \boldsymbol{\theta})$, is proportional to the *posterior* beliefs about the parameters having observed the data, $p(\boldsymbol{\theta} \,|\, \mathcal{D})$. The constant of proportionality (or normalization constant) $p(\mathcal{D})$ is found by integrating the right hand side with respect to $\boldsymbol{\theta}$ and is called the *evidence*:

$$p(\boldsymbol{\theta} \,|\, \mathcal{D}) = \frac{p(\mathcal{D} \,|\, \boldsymbol{\theta}) p(\boldsymbol{\theta})}{\underbrace{\int p(\mathcal{D} \,|\, \boldsymbol{\theta}) p(\boldsymbol{\theta}) \, d\boldsymbol{\theta}}_{Evidence}} \tag{2.10}$$

Note that, when $\boldsymbol{\theta}$ is integrated out, the evidence is simply the model's average prediction of $p(\mathcal{D})$, the overall likelihood of the data or in some contexts, the *marginal likelihood*.

The Bayesian setting differs from the maximum likelihood approach in that given the data one does not optimize to find a particular parametrization, such as $\boldsymbol{\theta}_{ML}$. Rather, given a prior distribution for the parameters and some data, the *posterior* distribution of the parameters having observed the data is found. Parameter uncertainty is then integrated out in order to make predictions. In the context of classification, if we have a likelihood model $p(y \,|\, \hat{\mathbf{x}}, \boldsymbol{\theta})$ for a previously unobserved input $\hat{\mathbf{x}}$ then, the model predicts the probability of the corresponding class label $\hat{y}$ being in class 1, $p(\hat{y} = 1 \,|\, \hat{\mathbf{x}})$, by:

$$p(\hat{y} = 1 \,|\, \hat{\mathbf{x}}) = \int p(\hat{y} = 1 \,|\, \hat{\mathbf{x}}, \boldsymbol{\theta}) p(\boldsymbol{\theta} \,|\, \mathcal{D}) d\boldsymbol{\theta} \tag{2.11}$$

This estimate of the class conditional probability averages over all parametrisations of the model; it therefore compensates for, and is more robust to, model uncertainty. The Bayesian approach has the distinct advantage of being able to incorporate prior beliefs into the model, and is a principled way to update beliefs about the parameters given new data, even a single observation.

The relative ease or difficulty of finding or drawing samples from the posterior distribution $p(\boldsymbol{\theta} \mid \mathcal{D})$ in equation (2.10) is dependent on the functional form the prior and likelihood take. In general, particularly when $\boldsymbol{\theta}$ is high dimensional, the *evidence* $\int p(\mathcal{D} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta})\, d\boldsymbol{\theta}$ which is the normalisation constant on the R.H.S of equation (2.10) may be difficult to evaluate.

However, when the prior and the likelihood are *conjugate* we have the particularly nice property that the posterior distribution is of the same functional form as the prior and can be written in an algebraically closed form. For example, in the case of a Gaussian likelihood function with known variance and unknown mean μ; the conjugate prior on the mean $p(\mu)$ is another Gaussian. In this case the posterior distribution $p(\mu \mid \mathcal{D})$ is a Gaussian and of the same functional form as the prior. A particularly important class of likelihood functions known to have conjugate priors is the exponential family of probability density functions [Bishop, 2006, pp 117]. For a relatively complete list of conjugate prior distributions see [Fink, 1997].

When conjugacy does not hold then finding the posterior distribution is more difficult because it involves evaluating the evidence. This involves integrating out the parameters $\boldsymbol{\theta}$ and when the dimension of $\boldsymbol{\theta}$ is large (because the model has lots of parameters) the evidence may be difficult to evaluate. There have been various approaches to overcome these issues, which are now briefly discussed.

Maximum a posteriori learning

The first, and probably simplest, approach is the *maximum a posteriori* (MAP) estimate. As with maximum likelihood, this does not provide a full posterior distribution, only a single point estimate for the true parameters. In this case the point estimate chosen, which we call $\boldsymbol{\theta}_{MAP}$, is the mode of the posterior distribution. Because, the mode of the distribution remains unchanged when the distribution is multiplied by a constant, it is enough to find an un-normalized distribution $\hat{p}(\boldsymbol{\theta} \mid \mathcal{D})$ which is proportional to the true distribution $p(\boldsymbol{\theta} \mid \mathcal{D})$. The un-normalized distribution can be found without having to evaluate the evidence, from equation (2.9), by setting $\hat{p}(\boldsymbol{\theta} \mid \mathcal{D}) = p(\mathcal{D} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta})$.

MAP learning has the advantage over maximum likelihood, as described above, as it can incorporate a prior $p(\boldsymbol{\theta})$ on the parameters. When this prior is uniform the maximum likelihood estimate and the MAP estimate are the same. A disadvantage of this approach is that it is not invariant to non-linear transformations of parameter space [Bernardo and Smith, 1994, Duda et al., 2000]. In other words two models which are essentially identical except that the parameter space is defined differently can produce different MAP estimates. It is also limited in that it only makes a single point estimate and does not model the full posterior distribution, and therefore does not obtain the advantages of integrating out parameter uncertainty.

Monte Carlo approaches

An alternative to obtaining an explicit formula for the posterior distribution, is to draw samples from the posterior, which can then be used to approximate integrals like (2.11). When $\boldsymbol{\theta}$ is high dimensional, in general, both drawing samples from $p(\boldsymbol{\theta} \,|\, \mathcal{D})$ and evaluating the evidence $\int p(\mathcal{D} \,|\, \boldsymbol{\theta}) p(\boldsymbol{\theta})\, d\boldsymbol{\theta}$ are difficult problems. An alternative to evaluating the evidence is to draw samples from the un-normalised distribution:

$$\hat{p}(\boldsymbol{\theta} \,|\, \mathcal{D}) = p(\mathcal{D} \,|\, \boldsymbol{\theta}) p(\boldsymbol{\theta}) \tag{2.12}$$

There are several different approximation approaches to sampling from high dimensional distributions which broadly come under the heading of Monte Carlo approaches. Several approaches to sampling from high-dimensional distributions are now briefly described, namely *rejection sampling*, *Gibbs sampling*, the *Metropolis algorithm* and *Metropolis Hastings*. In order to remain general these approaches are described for an arbitrary probability distribution $g(\boldsymbol{\theta})$, however, for Bayesian inference the distribution of interest is $p(\boldsymbol{\theta} \,|\, \mathcal{D})$. For a good introduction to Monte Carlo approaches see MacKay [1998].

Sampling from a distribution

One of the most straightforward approaches to sampling from an un-normalised distribution is *rejection sampling*, MacKay [1998]. Here we are attempting to draw from some distribution $g(\boldsymbol{\theta}) = \hat{g}(\boldsymbol{\theta})/Z$ where Z is some unknown, and in general difficult to calculate, normalization constant. The idea here is to approximate $\hat{g}(\boldsymbol{\theta})$ by some distribution

$\hat{Q}(\boldsymbol{\theta})$ which we can sample from and which has the property that for some real number c, $c\hat{Q}(\boldsymbol{\theta}) > \hat{g}(\boldsymbol{\theta})$ for all $\boldsymbol{\theta}$. To sample from $\hat{g}(\boldsymbol{\theta})$ using rejection sampling, we first draw a sample $\boldsymbol{\theta}$ from $\hat{Q}(\boldsymbol{\theta})$, and a real number u which is drawn uniformly from the interval $[0, c\hat{Q}(\boldsymbol{\theta})]$. If $u > \hat{g}(\boldsymbol{\theta})$ then $\boldsymbol{\theta}$ is rejected. The process is repeated until a u is found for which $u \leq \hat{g}(\boldsymbol{\theta})$. It can be seen that this process does indeed sample from $\hat{g}(\boldsymbol{\theta})$. However when $\boldsymbol{\theta}$ is high dimensional, even a small difference between $g(\boldsymbol{\theta})$ and $Q(\boldsymbol{\theta})$ can lead to large values of c; in which case the interval $[0, c\hat{Q}(\boldsymbol{\theta})]$ is large and many samples of $\boldsymbol{\theta}$ may be needed before the criterion $u \leq \hat{g}(\boldsymbol{\theta})$ is met.

Gibbs sampling attempts to reduce the problem of sampling from a high dimensional distribution by sampling from one (or several) dimensions at a time, while keeping the others fixed, and repeating the process many times until convergence is reached. For example if $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)$ then we would initialize the process with some $\boldsymbol{\theta}^*$. We then sample θ_1 from $g(\theta_1 \,|\, \theta_2^*, \theta_3^*)$ while keeping θ_2 and θ_3 fixed. Similarly θ_2 is then sampled from $g(\theta_2 \,|\, \theta_1, \theta_3^*)$ keeping θ_1 from the previous step and θ_3 fixed. The same is done for θ_3 and the process is then repeated by updating θ_1 etc.

A Markov chain is a sequence of non-independent samples, $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ...\}$, such that each sample is dependent only on the previous sample. That is $p(\boldsymbol{\theta}_t \,|\, \boldsymbol{\theta}_{t-1}, \boldsymbol{\theta}_{t-2}, ..., \boldsymbol{\theta}_1) \equiv p(\boldsymbol{\theta}_t \,|\, \boldsymbol{\theta}_{t-1})$. In a Markov Chain Monte Carlo (MCMC) approach, a distribution $Q(\boldsymbol{\theta}_t \,|\, \boldsymbol{\theta}_{t-1})$ which is called a *jump or proposal distribution* is used to generate $\boldsymbol{\theta}_t$ from $\boldsymbol{\theta}_{t-1}$. This distribution may depend on $\boldsymbol{\theta}_{t-1}$ so that $Q(\boldsymbol{\theta}_t \,|\, \boldsymbol{\theta}_{t-1})$ may be a very different distribution to $Q(\boldsymbol{\theta}_{t+1} \,|\, \boldsymbol{\theta}_t)$. At each step a sample is selected from Q and is either accepted or rejected based on comparison with $g(\boldsymbol{\theta})$. The Markov Chain starts at a particular $\boldsymbol{\theta}^*$ and makes a 'random walk'. The idea is that after a certain 'burn in' time these walks tend towards a stationary distribution $g(\boldsymbol{\theta})$, and $\boldsymbol{\theta}_t$ represents a random draw from $g(\boldsymbol{\theta})$. MCMC approaches can also be used to sample from a distribution $g(\boldsymbol{\theta}) = \hat{g}(\boldsymbol{\theta})/Z$ where normalization constant Z is not known and may be difficult to evaluate. The two main MCMC approaches are the *Metropolis algorithm* and an extension of this which is called *Metropolis Hastings algorithm.*

The Metropolis algorithm, [Metropolis et al., 1953] makes the assumption that the jump distribution is symmetric. That is $Q(\boldsymbol{\theta}_{t+1} \,|\, \boldsymbol{\theta}_t) = Q(\boldsymbol{\theta}_t \,|\, \boldsymbol{\theta}_{t+1})$. The algorithm initializes with some value $\boldsymbol{\theta}_0$ such that $\hat{g}(\boldsymbol{\theta}_0) > 0$. At each step a candidate point $\boldsymbol{\theta}^*$ is sampled from the jump distribution $Q(\boldsymbol{\theta}_t \,|\, \boldsymbol{\theta}_{t-1})$ and is accepted with probability α_t (given in equation (2.14)) or rejected. If $\boldsymbol{\theta}^*$ is accepted then $\boldsymbol{\theta}^*$ is assigned to $\boldsymbol{\theta}_t$, if $\boldsymbol{\theta}^*$ is not accepted then $\boldsymbol{\theta}_{t-1}$ is added to the list instead, that is:

$$\boldsymbol{\theta}_t = \begin{cases} \boldsymbol{\theta}^* & \text{if accepted with probability } \alpha_t \\ \boldsymbol{\theta}_{t-1} & \text{otherwise.} \end{cases} \tag{2.13}$$

where α_t is given by:

$$\alpha_t = \min\left(\frac{\hat{g}(\boldsymbol{\theta}^*)Q(\boldsymbol{\theta}_{t-1} \,|\, \boldsymbol{\theta}^*)}{\hat{g}(\boldsymbol{\theta}_{t-1})Q(\boldsymbol{\theta}^* \,|\, \boldsymbol{\theta}_{t-1})}, 1\right) \tag{2.14}$$

The Metropolis Hastings algorithm [Hastings, 1970] extends the Metropolis algorithm to jump distributions which are not necessarily symmetric. There is considerable theory on MCMC which establishes the conditions in which the sequence $\boldsymbol{\theta}_t$ will eventually converge to an independent sample from $g(\boldsymbol{\theta})$, for example [Cowles and Carlin, 1996, Gamerman and Lopes, 2006, Gilks et al., 1996]. One of the main problems though, is that even when these conditions are met it is unclear how many iterations are necessary before $\boldsymbol{\theta}_t$ becomes, effectively, an independent sample from $g(\boldsymbol{\theta})$. In general MCMC takes many thousands of iterations and it is difficult to assess the accuracy of the predictions. However, it is one of the most effective and widely used methods of sampling from complex and high dimensional distributions.

2.1.3. Monte Carlo integration

The ability to sample from a distribution $g(\boldsymbol{\theta})$ can also be used to perform a Monte Carlo integration of some non-trivial function $h(\boldsymbol{\theta})$. In particular drawing independent samples from $p^*(\boldsymbol{\theta} \,|\, \mathcal{D})$ also allows integrals such as the equation (2.11) to be evaluated. To perform a Monte Carlo integration on some arbitrary distribution $h(\boldsymbol{\theta})$ the distribution

is first rewritten in the form $h(\boldsymbol{\theta}) = f(\boldsymbol{\theta})g(\boldsymbol{\theta})$ and the integration becomes:

$$\int h(\boldsymbol{\theta})d\boldsymbol{\theta} = \int f(\boldsymbol{\theta})g(\boldsymbol{\theta})d\boldsymbol{\theta} = E_{g(\boldsymbol{\theta})}[f(\boldsymbol{\theta})] \approx \frac{1}{M}\sum_{m=1}^{M} f(\boldsymbol{\theta}^{(m)}) \tag{2.15}$$

Here, the integral has been rewritten as an expectation of $f(\boldsymbol{\theta})$ over $g(\boldsymbol{\theta})$ and is approximated by the sum of observations $\boldsymbol{\theta}^{(m)}$ sampled from $g(\boldsymbol{\theta})$. It is known that the variance of the estimate is $\frac{\sigma^2}{M}$, where σ^2 is the variance of $f(\boldsymbol{\theta}^{(m)})$. Notice that this is not dependent on the dimension of $\boldsymbol{\theta}$ and decreases in proportion to $\frac{1}{M}$. Therefore, somewhat surprisingly, relatively few samples may be required for the integral to be evaluated, sometimes even as few as a dozen [MacKay, 1998].

Variational inference

The variational approach, [MacKay, 1995, Attias, 2000, Jordan et al., 1999], may be used to approximate an otherwise intractable posterior distribution $p(Z \mid X)$ by a known parametrized distribution $q(Z)$. Here Z may refer to both the model parameters $\boldsymbol{\theta}$ and hidden variables within the model. The choice of which functional form $q(Z)$ should take represents a trade-off between tractability and how closely $q(Z)$ is able to match the true distribution $p(Z \mid X)$. Typically, it is assumed that $q(Z)$ is factorisable into groups; or single variables Z_i. That is:

$$q(Z) = \prod_{i=1}^{m} q_i(Z_i). \tag{2.16}$$

The approximation is made by minimizing the KL-divergence between the parametrized distribution $q(Z)$ and the posterior distribution $p(Z \mid X)$. We start by decomposing $\ln p(X)$ as:

$$\ln p(X) = \int q(Z) \ln \left(\frac{p(X,Z)}{q(Z)} \right) dZ - \int q(Z) \ln \left(\frac{p(Z \mid X)}{q(Z)} \right) dZ \tag{2.17}$$

and noting that this is of the form:

$$\ln p(X) = \mathcal{L}(q(Z)) + KL(q(Z) \,||\, p(Z \mid X)) \tag{2.18}$$

Here $\mathcal{L}(q(Z))$ is a functional in $q(Z)$, given by [1]:

$$\mathcal{L}(q(Z)) = \int q(Z) \ln \left(\frac{p(X,Z)}{q(Z)} \right) dZ \tag{2.19}$$

The key point being that, because $p(X)$ does not depend on Z, maximizing $\mathcal{L}(q(Z))$ will therefore minimize the KL-divergence. By substituting in the factorization assumption from equation (2.16), this becomes:

$$\mathcal{L}(q) = \int \left(\prod_i q_i(Z_i) \right) \left(\ln p(X,Z) - \sum_i \ln q_i(Z_i) \right) dZ \tag{2.20}$$

The maximization of $\mathcal{L}(q(Z))$ is achieved by iteratively optimizing with respect to each Z_j in turn while keeping Z_i, $i \neq j$ fixed. We start by rearranging the lower bound to:

$$\mathcal{L}(q) = \int q_j(Z_j) \left(\int \ln p(X,Z) \prod_{i \neq j} q_i(Z_i) dZ_i \right) dZ_j - \int \prod_i q_i(Z_i) \sum_i \ln q_i(Z_i) dZ \tag{2.21}$$

We note that $\int \ln p(X,Z) \prod_{i \neq j} q_i(Z_i) dZ_i = E_{i \neq j}[\ln p(X,Z)]$, and define a new distribution $\tilde{p}(X, Z_j)$ by:

$$\ln \tilde{p}(X, Z_j) = E_{i \neq j}[\ln p(X,Z)] + constant \tag{2.22}$$

where the constant is necessary to normalize $\tilde{p}(X, Z_j)$. The second term in (2.21) can be simplified when we consider that we are maximizing the lower bound only with respect to $q_j(Z_j)$, and keeping all other $q_i(Z_i)$ fixed. Therefore:

$$\begin{aligned} \int \prod_i q_i(Z_i) \sum_i \ln q_i(Z_i) dZ = & \int \prod_i q_i(Z_i) \ln q_j(Z_j) dZ \\ & + \int \prod_i q_i(Z_i) \sum_{i \neq j} \ln q_i(Z_i) dZ, \end{aligned} \tag{2.23}$$

[1] Note the decomposition in equation 2.17 can be verified by substituting the product rule for probability, $\ln p(X,Z) = \ln p(X) + \ln p(Z \mid X)$ in to $\mathcal{L}(q(Z))$. For details see Bishop [2006, pp 451].

which can be rearranged to give:

$$\int \prod_i q_i(Z_i) \sum_i \ln q_i(Z_i) dZ = \int \left(\prod_{i \neq j} \int q_i(Z_i) dZ_{i \neq j} \right) q_j(Z_j) \ln q_j(Z_j) dZ_j + \int \left(\int q_j(Z_j) dZ_j \right) \prod_{i \neq j} q_i(Z_i) \sum_{i \neq j} \ln q_i(Z_i) dZ_{i \neq j}. \quad (2.24)$$

The R.H.S in equation (2.24) simplifies to $\int q_j(Z_j) \ln q_j(Z_j) dZ_j + constant$ by noting that $\int q_i(Z_i) dZ_i = 1$ for all distributions $q_i(Z_i)$ and noting that because of this the second term depends only on $q_i(Z_i)$ where $i \neq j$. Putting these together we can write equation (2.21) as:

$$\mathcal{L}(q) = \int q_j(Z_j) \ln \tilde{p}(X, Z_j) dZ_j - \int q_j(Z_j) \ln q_j(Z_j) dZ_j + constant \quad (2.25)$$

which is the negative KL-divergence between $\tilde{p}(X, Z_j)$ and $q_j(Z_j)$. Hence the minimum $\mathcal{L}(q)$ occurs when $q_j(Z_j) = \tilde{p}(X, Z_j)$. We therefore have an expression for the optimal $q_j^*(Z_j)$ given by:

$$\ln q_j^*(Z_j) = E_{i \neq j}[\ln p(X, Z)] + constant \quad (2.26)$$

Equation (2.26) is used to iteratively update each $q_j(z_j)$ in turn. In many cases the expectations on the R.H.S are analytically tractable and, therefore, variational Bayes is often computationally efficient.

In this section we have looked at various commonly used approximation approaches to either evaluate the Bayesian posterior, by sampling from a distribution, or approximating the posterior distribution directly, as in the case of variational inference. We now look specifically at active learning which is the focus of the thesis.

2.2. Active Learning

In this section we make a brief review of literature and current approaches to active learning in order to set a broader context for the thesis. The emphasis here is on binary classification, where some set of inputs is to be labelled as being in one of just two classes.

Algorithm 1 Active Learning

Require: Initial set of n input-label pairs, $\mathcal{D}_0$

1: Time step: $t = 1$
2: `repeat:` Until stopping criterion.
3: Find: $p(\boldsymbol{\theta} \,|\, \mathcal{D}_{t-1})$ — Update model parameter distribution
4: Select: ϕ_t given $p(\boldsymbol{\theta} \,|\, \mathcal{D}_{t-1})$ — Learner selects query parameters
5: $\mathbf{x}_t \sim Query\, Oracle(\phi_t)$ — Given ϕ_t query oracle returns $\mathbf{x}_t$
6: $y_t \sim Label\, Oracle(\mathbf{x}_t)$ — Given $\mathbf{x}_t$ label oracle returns y_t
7: $\mathcal{D}_t \coloneqq \mathcal{D}_{t-1} \cup \{\mathbf{x}_t, y_t\}$ — Add $\{\mathbf{x}_t, y_t\}$ to the labelled data
8: $t \coloneqq t + 1$ — Increment time-step
9: `end`

For now and in Chapter 3 we will consider the class labels as being 0 and 1, so that for a given $\mathbf{x}$ we are attempting to assign a class label $y \in \{0, 1\}$[2]. While we limit the discussion to binary classification, it should be noted that much of what is said could in principle be straightforwardly extended to multi-class problems, where there are more than two class labels. Indeed, the formal structure of the Bayesian approach laid out in Chapter 3 is couched in terms of cost functions. As such it is, in principle, possible to construct sensible cost functions both for multi-class problems and also for problems in regression, where the target y is continuous rather than discrete.

In binary classification we start with some training data $\mathcal{D} = \{\mathbf{x}_1, y_1, \ldots, \mathbf{x}_N, y_N\}$, where $\mathbf{x}_i \in \mathcal{X}$, which is a set of inputs $\mathbf{x}_i$ together with their corresponding labels y_i. These are used by the learner to refine its beliefs about the model parameters, which we usually denote by $\boldsymbol{\theta}$. In the case of a maximum likelihood model, or some 'deterministic' model such as an SVM or an MLP, some specific parametrization $\boldsymbol{\theta}^*$ is learned, and in the case of a Bayesian model the posterior distribution for the parameters, having seen the data, $p(\boldsymbol{\theta} \,|\, \mathcal{D})$ is inferred. These parameters are then used to make inferences about the class labels for previously unseen test data, $\mathcal{T} = \{\hat{\mathbf{x}}_m\}_{m=1}^{M}$, where $\hat{\mathbf{x}}_m \in \mathcal{X}$.

The difference between active learning and passive learning, also called *passive learning*, is that in passive learning, the labelled training inputs are drawn independently and identically from an underlying distribution $p(\mathbf{x})$. The learner then has to wait for an adequate set of labelled inputs in order to be able to make its classification. There are

[2] In Chapter 4 it will be more convenient to denote the class labels as $+1$ and -1 so $y \in \{-1, +1\}$.

many applications where obtaining the class labels y_i incurs a cost. A common example of this is when the labels have to be assigned by hand, for example, in medical image classification where data must be labelled by an expert in the field [Hoi et al., 2006]. With passive learning the learner has no choice about which labelled examples it is given. As it can be costly to obtain the class labels, the aim is to learn using as little data as possible. In *active* learning rather than being presented with the data all in one go, the learner is able to select its own training data to learn from. The idea of active learning is that by allowing the learner to select relevant inputs $\mathbf{x}_i$ it is often possible to considerably reduce the number of labelled inputs necessary in order to classify the data.

The process by which active learning is performed is shown in algorithm 1. Generally, we start with a small number of randomly chosen input-label pairs from each class (often just one), represented by $\mathcal{D}_0$.

In line 3 of algorithm 1 the classifier learns from previous data $\mathcal{D}_{t-1}$. For a Bayesian classifier parametrised by $\boldsymbol{\theta}$ the updated 'beliefs' are represented by the distribution $p(\boldsymbol{\theta} \,|\, \mathcal{D}_{t-1})$. However, in general the appropriate representation is dependent on the classifier in question. For example Gaussian process (GP) classifiers are non-parametric, in which case it would be more accurate to represent the update by $p(y \,|\, \mathbf{x}, \mathcal{D}_{t-1})$ the probabilities of class membership predicted by the GP. For a non-Bayesian classifier a specific parametrisation $\boldsymbol{\theta}^*$ may be learned from the data $\mathcal{D}_{t-1}$ rather than a probability distribution.

Often in active learning a specific input $\mathbf{x}_t$ is selected by the learner, however active learning as presented in algorithm 1 is kept more general by the introduction of a *query oracle*, parametrised by ϕ_t. As will be seen in chapter 3, this allows the learner the flexibility of sampling inputs from the underlying distribution $p(\mathbf{x})$. The query parameters ϕ_t are selected by the learner in line 4 of algorithm 1. The parameters ϕ_t are then given to the *sample oracle* in line 5 of algorithm 1. The query oracle draws an input label $\mathbf{x}_t$ from the underlying distribution $p(\mathbf{x})$ which is then given to the *label oracle* in line 6. The label oracle returns a class label y_t which is drawn from the true class conditional distribution $p(y \,|\, \mathbf{x})$. The input-label pair $\{\mathbf{x}_t, y_t\}$ are then added to the labelled data set

$\mathcal{D}_t$ in line 7. This completes one time-step of active learning. The process is then repeated with one new input-label pair drawn in each step. In some cases the learner may request a small batch of inputs to be labelled, in each step t. However, in this thesis we consider specifically the case where the learner is selecting just one new input to query in each step.

Active learning is distinct from online-learning where the learner *updates* its knowledge on the basis of new data, because the active learner also has a choice as to which new data to observe.

Fitting into the general framework, there are three main types of active learning: *pool-based*, *membership query* and *stream-based*. The first, *pool-based* active learning is where the underlying distribution $p(\mathbf{x})$ is not known, but we are supplied with a (finite) pool of unlabelled data. In pool-based active learning the query oracle cannot sample from the underlying distribution and the learner simply selects which input $\mathbf{x}$ to query. Many practical applications are pool-based. For example in medical imaging we might be given a finite number of unlabelled medical slides, on which to learn a classification. Here the medical slides represent a finite set of inputs, which the learner may select for labelling. Pool-based active learning is of considerable importance, because of availability of large amounts of unlabelled data in many domains. For example, Guyon et al. [2010] cite pool-based learning as having application in "Pattern Recognition (handwriting, speech, airborne or satellite images etc), text processing (Internet document archives), chemo-informatics (untested molecules from combinatorial chemistry) and also marketing".

The second type is the *membership query* approach also known as *selective learning*. In this scenario the learner cannot draw from the underlying probability distribution $p(\mathbf{x})$ but may construct arbitrary inputs $\mathbf{x}$ to be queried. Here the query oracle may construct artificial examples of inputs or *membership queries* from the sample space, to be labelled by the label oracle. In pool-based learning, if the pool is small, then it may not be possible to query the input which is most informative to the learner, because it is absent from the pool, so membership query approaches can circumvent this difficulty. The two main advantages of the *membership query* approach are that it allows the learner to select *any* query which will be informative to the learner. Secondly, it can be computationally faster

than pool-based learning, because a query instance may be constructed without having to evaluate every input in the pool. Ling and Du [2008] describe a *membership query* approach to active learning using decision trees to construct instances (inputs) about which the learner is most uncertain. They demonstrate this approach on 12 UCI data sets, and show a significant improvement over two comparison pool-based active learning algorithms on 11 out of the 12 data sets. However, the membership query approach is not always appropriate. For example, in text classification an arbitrary membership query is unlikely to be an allowable natural language expression and would be difficult to label for a human teacher [Lewis and Gale, 1994].

The final broad category of active learning is *stream-based* active learning where it is assumed that drawing unlabelled points from the underlying distribution is free or inexpensive. In this case inputs are drawn sequentially and the learner decides whether to discard the input or to query for its class label; however, an input cannot be retrieved once discarded. In this type of active learning the query oracle may define the probability with which any given input from the stream is accepted. Argamon-Engelson and Dagan [1999] apply active learning in this scenario by evaluating whether to query or not based on a measure of the unlabelled input's 'informativeness'. The focus of chapter 4 is primarily pool-based active learning although the framework described in Chapter 3 is more general.

Algorithms and heuristics to select informative data for active learning are therefore of interest and a number of approaches have been developed for doing this. Notable among these is the work by Seung et al. [1992], Lewis and Gale [1994], Roy and McCallum [2001], Tong and Koller [2000], Schohn and Cohn [2000], Campbell et al. [2000] and Baram et al. [2004]. There are three main approaches to active learning which encompass most of the literature on active learning. The first approach is *uncertainty based sampling* which evaluates the confidence of the classifier on unseen instances and queries the data about which it is most uncertain [Lewis and Gale, 1994]. Secondly, approaches such as *query by committee* (QBC) [Seung et al., 1992, Balcan et al., 2006] depend upon the notion of a version space. Loosely, this is the set of classifiers or parametrizations of a classifier family that are consistent with the data thus far observed. In this approach new queries are selected which ideally halve the space of consistent solutions, thus providing the maximum

information gain. QBC approaches have been applied to Support Vector Machine (SVM) classification. This was proposed independently by Schohn and Cohn [2000], Campbell et al. [2000], and Tong and Koller [2000] who dubbed their version SIMPLE. This approach works well in practice with significant success in practical problems [Tong and Koller, 2000]. The third approach is called SELF-CONF, which optimizes the choice of query in order to maximize its expected utility at the next time step. Baram et al. [2004] consider SIMPLE and SELF-CONF to be the two top performing active learning algorithms.

We now briefly discuss some of the theoretical results in active learning before looking in a bit more detail at the various approaches.

2.2.1. Theoretic bounds on Active Learning

The majority of published literature on active learning certainly suggests that, empirically, active learning works [Settles, 2010]. That is, it usually performs at least as well as passive learning, where data is sampled randomly from the underlying distribution, and in many cases is considerably better. However, it would be good to have some theoretical reassurance that active learning works. The two principal questions are:

1. If the algorithm is given enough data will it eventually converge to the best possible classification achievable by the model?

2. How many queries will it take to achieve a certain error rate ϵ?

Question 1 is usually referred to as *consistency* and is clearly a desirable property for any learning algorithm. To make the choice to employ active learning rather than passive learning then we would like to have some assurance that the active learner is consistent. Question 2 is about how fast the algorithm converges to a solution. In particular, does active learning learn faster than passive learning?

A major step forward in this direction was the Query by Committee approach of Seung et al. [1992], who demonstrated that for certain separable problems active learning leads to a generalization error which decreases exponentially with the number of queries. Freund et al. [1997] improved on this by proving that for any separable data with Vapnik-Chervonenkis (VC) dimension d the QBC algorithm requires only $\tilde{\mathcal{O}}(d \log \frac{1}{\epsilon})$ queries, as

compared to $\tilde{\mathcal{O}}(\frac{1}{\epsilon})$ queries in the case of passive learning [3]. The VC dimension, which was first introduced by Vapnik and Chervonenkis [1971], is a measure of the flexibility of a classifier. The VC dimension h is the least number of inputs such that *no* training set of h inputs can always be separated for all possible combinations of class labels. It is useful because it gives an upper bound on the test error [Vapnik, 1982]. Given the training error, the following holds with probability $1 - \eta$:

$$\textit{test error} \leq \textit{training error} + \sqrt{\frac{h(\log(\frac{2N}{h}) + 1) - \log(\frac{\eta}{4})}{N}}. \tag{2.27}$$

Here N is the number of inputs in the training set.

Dasgupta et al. [2005] also demonstrate a generalization error which decreases exponentially with the number of queries using a variation of perceptron-based active learning. However, these positive theoretical results on the generalisation error are limited because of the requirement that the data be separable, and do not satisfy the *consistency* criterion if the data is not separable. Dasgupta et al. [2005] also show that there are many (including some very simple) examples where the number of queries required to *verify* an error of less than ϵ, is no better with active learning than for passive learning. For example the *standard* perceptron update performs no better than passive learning, whichever active learning scheme is applied, requiring $\tilde{\mathcal{O}}(\frac{1}{\epsilon})$ samples to verify an error of less than ϵ.

Balcan et al. [2008] argue that these results are misleading because there is a difference between how many queries are required to reach an error rate of ϵ and how many queries are needed for the active learner to verify that it has reached this error rate. Using this distinction Balcan et al. [2008] go on to make some very general statement about the efficacy of active learning. They start by distinguishing between the number of queries needed to confirm an error of less than ϵ which they call *verifiable sample complexity* and the number of queries needed to learn a particular target distribution with error less than ϵ which they call *sample complexity*. The simple illustrative example that they give is learning the bounds of a single uniform interval (a, b) such that $0 \leq a \leq b \leq 1$. This is

[3] $f(N)$ being of order $\mathcal{O}(g(N))$ means that there exists a constant $k_1 > 0$ so that $|\, f(N) \,| \leq g(N).k_1$ for all N larger than some fixed value. $f(N) \in \tilde{\mathcal{O}}(g(n))$ is looser, saying there exists a constant $k_2 > 0$ so that $f(n)$ is of order $\mathcal{O}(g(N) \log^{k_2}(g(N)))$. $\Omega(g(N))$ is the same as $\mathcal{O}(g(N))$ except that $|\, f(N) \,| \geq g(N).k_1$.

a binary classification problem where inputs within the interval are in class 1 and inputs outside the interval are in class 0. They describe an active learning algorithm which requires only $\mathcal{O}(\frac{1}{w}) + \mathcal{O}(\log(\frac{1}{\epsilon}))$ samples to learn the interval with error less than ϵ for a given interval with width $w = |b - a|$. On the other hand if the target distribution for class 1 is the empty interval (which occurs when $a = b$ and there is no volume of inputs in class 1) it will take $\Omega(\frac{1}{\epsilon})$ label requests to verify this, which is the same bound as passive learning. They prove that for *any* distribution with finite VC dimension, active learning has a lower *sample complexity* than passive learning. In other words, because the active learner has no knowledge about the distribution that it is learning, it will take $\Omega(\frac{1}{\epsilon})$ queries to verify that its error is less than ϵ. However, for any *given* distribution (with finite VC dimension) we can expect active learning to require fewer queries.

Many of the algorithms which have been presented in the literature on the bounds for active learning are not feasible in practice or have limited scope (such as requiring the data to be separable). More recently Beygelzimer et al. [2009] have presented an algorithm which is both computationally feasible and has some theoretical reassurance. They prove bounds on label complexity, namely the number of observations necessary for the classifier to learn, which are superior to supervised learning and prove the *consistency* of this algorithm. They demonstrate their algorithm in practice on the MNIST data set of handwritten digits showing empirical results which reliably beat passive learning.

2.2.2. QBC approaches

The first of the main approaches to active learning is the query by committee approach presented by Seung et al. [1992]. Seung et al. demonstrate a strong theoretical framework for the separable case. They showed that when the data is separable, that is when there exists a decision boundary for which all test data could be correctly labelled, then a committee approach can lead to a generalization error which decreases exponentially with the number of observations, as compared to random sampling which converges according to the inverse power law.

When the data is separable, repeated observations of a particular input always lead to the same class label. In this situation whenever a new observation is made, some of

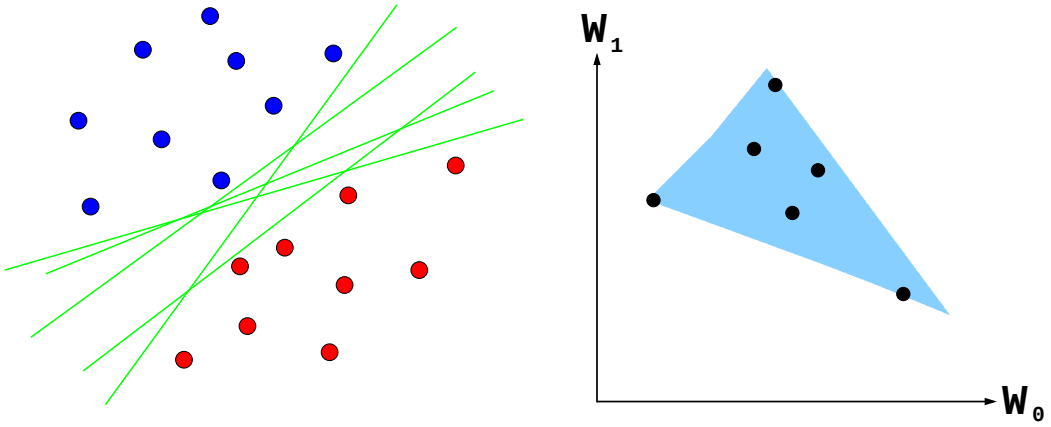


Figure 2.1.: Schematic illustration of linear decision boundaries consistent with previous observations (left) and the corresponding version space (right). Left: blue and red dots represent labelled observations. Right: w_0, w_1 represent the slope and intercept of the linear decision boundaries. Therefore dots in right represent linear decision boundaries in left. The blue region represents version space, where parametrizations correspond to consistent decision boundaries.

the model solutions can be ruled out as being inconsistent with the observation. Figure 2.1 (left) illustrates some consistent solutions when the decision boundary is linear. If these decision boundaries are parametrized then the set of parameters which correspond to consistent solutions is called *version* space. See figure 2.1 (right) for an illustration of version space corresponding to figure 2.1 (left). In the QBC view version space is viewed as a (possibly infinite) 'committee' of members each representing one possible solution consistent with previous observations. For any given input $\mathbf{x}_n$ each member of the committee will assign a particular class label $y_n \in \{0, 1\}$. The principle of query by committee is to query the input about which there is maximal disagreement, among the committee. The ideal situation being where half of the committee assign the class label 0 and half the committee assign the class label 1 to a given input. In this case a single query will eliminate half of the committee and version space will be halved. Since in general calculating the expected reduction in the volume of version space is prohibitively costly, because all possible classifiers in version space must be evaluated on each candidate query. Seung et al. [1992] suggest drawing only a sample 'committee' from version space and choosing the input for which there is maximum disagreement. Somewhat surprisingly they showed that a committee of two is quite effective.

The QBC approach has been extended by a number of authors to account for non-separable data. In this case a hypothesis cannot be removed simply because it is inconsistent with one observation. Argamon-Engelson and Dagan [1999] retain the idea of querying where there is maximal disagreement between classifiers while working with a committee of probabilistic classifiers. Committee members are drawn according to their posterior probability from previous observations, and inputs are queried for which this sample of probabilistic classifiers maximally disagree. Here the relative probability of hypotheses are updated, rather than eliminating 'inconsistent' classifiers.

The QBC approach has also been extended by Balcan et al. [2006] to account for non-separable data with an approach they call agnostic active learning. Their model is based on separable data with the addition of a certain level of noise ϵ. Instead of eliminating hypotheses on the basis of a single observation, which risks eliminating the optimal hypotheses, they estimate an upper bound and a lower bound for the generalization error of each hypothesis, removing only the hypothesis for which the lower bound is larger than the least upper bound. That is they remove the hypothesis which at best still performs worse than at least one other hypothesis in the set. In the agnostic approach data is drawn in IID batches sampled from the current region of uncertainty (the region upon which there remains disagreement among the hypotheses). This is necessary in order to estimate upper and lower bounds on the generalisation error. Balcan et al. [2006] showed that for some sets of classifiers, even with arbitrary noise levels, generalization errors which decrease exponentially with the number of observations are possible. However, despite achieving a generalization error which decreases exponentially with the number of observations, the need to sample in IID batches from the current region of uncertainty is an artefact of the agnostic approach and it is not clear why IID batches of observation should represent the ideal sequence of observations for an active learner.

The idea of doing active learning by reducing the volume of version space has also been extended to Support Vector Machines. This was done simultaneously by three groups: Campbell et al. [2000], Schohn and Cohn [2000] and Tong and Koller [2000].

Support Vector Machines were first introduced by Vapnik and Cortes [1995], and work by using a non-linear mapping Ψ from the data space $\mathcal{X}$ to a much higher dimensional space (possibly infinite dimensional) $\mathcal{H}$, which is assumed to be separable. They seek to find a linear decision boundary (or hyperplane) in $\mathcal{H}$ which corresponds to a non-linear decision boundary in the original data space $\mathcal{X}$. The motivation for this is Cover's theorem [Cover, 1965] which states that within a high dimensional space $\mathcal{H}$ data is more likely to be linearly separable. If the data is linearly separable in $\mathcal{H}$ there may be many possible dividing hyperplanes. The SVM chooses the 'maximum margin' hyperplane, for which the perpendicular distance to the nearest data-point is maximized. The idea being that the maximum margin hyperplane minimizes the risk of misclassifying future observations by ensuring that the gap between classes is as large as possible. The margin is entirely defined by the *support vectors*, which are the inputs that when mapped into $\mathcal{H}$ lie on the margins of the hyperplane.

Vapnik and Cortes [1995] note that the calculations to find the necessary hyperplane in $\mathcal{H}$ can be expressed entirely in terms of inner products in Ψ. Therefore for a kernel function defined as:

$$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \Psi(\mathbf{x}_i) \cdot \Psi(\mathbf{x}_j). \tag{2.28}$$

computing the hyperplane becomes a quadratic optimization problem, which can be expressed entirely in terms of the kernel $\mathcal{K}$. This is called the *kernel trick* [Schlkopf and Smola, 2002] and means that all distances in the high dimensional space $\mathcal{H}$ can be calculated directly in the original much lower dimensional space $\mathcal{X}$. When the kernel function satisfies Mercer's condition, there exits a mapping Ψ such that equation (2.28) holds. In this case the quadratic optimization can be implemented without even having to define Ψ directly. SVMs are computable in $\tilde{\mathcal{O}}(N^3)$ time, where N is the number of inputs, though faster algorithms have been written, such as Sequential Minimal Optimization, Platt [1999], which takes $\tilde{\mathcal{O}}(N^2)$ time. For a good introduction to SVMs and kernel functions see Campbell and Ying [2011].

For active learning the assumption of separability in $\mathcal{H}$ allows the idea of version space from the query by committee approach to be applied. When $\mathcal{H}$ is separable the set of dividing hyperplanes consistent with current observations corresponds to version space. The approach, which was independently suggested by Campbell et al. [2000], Schohn and Cohn [2000] and Tong and Koller [2000], is to minimize this version space by querying unlabelled points which are as close to the current decision boundary as possible. In practice when this algorithm is applied the assumption of separability can be loosened by seeking hyperplanes which correctly classify *most* of the data. This is done by seeking to minimize a soft margin loss function, which includes a regularized penalty for mislabelled inputs.

As mentioned earlier in section 2.2, this algorithm was dubbed SIMPLE by Tong and Koller [2000] and Baram et al. [2004] consider SIMPLE to be one of the two top performing active learning algorithms. It was applied to text classification by Tong and Koller [2000] and has had significant success in practical problems.

2.2.3. Uncertainty based approaches

There are a number of criteria which, loosely speaking, stem from the idea of selecting inputs for which the model produces the least certain classification. This requires a probabilistic classifier, or some other prediction of classifier uncertainty. This idea should favour data close to the estimated decision boundary which are expected to be informative. In their paper on uncertainty sampling Lewis and Gale [1994] demonstrate good results on training text classifiers by querying points for which the predicted probability of class membership is close to a half. In fact they managed to reduce by 500-fold the amount of training data needed as compared to random sampling. This is because the fraction of negative samples was similarly high (500 to 1). The entropy $H[p(y \mid \mathbf{x})]$ of the class prediction has also been suggested by [MacKay, 1991] and Settles [2010]. This is a similar criterion because the point of maximum posterior uncertainty coincides with the point of maximum entropy [Chen and Mani, 2010].

In general, the difficulty with uncertainty sampling is that it tends to be 'noise seeking' and may select outliers about which the classifier is uncertain but do not inform overall

classification [Roy and McCallum, 2001]. To compensate for this Settles and Craven [2008] propose a form of uncertainty sampling which also tries to choose inputs 'representative' of the unlabelled data. Here, representative means that the average distance to other unlabelled inputs is small. There is a weighting parameter which balances this criterion with the input uncertainty. Reported results for this approach have shown it to perform better than uncertainty sampling where representativeness is not considered. More recently this idea has been applied to active learning with SVMs [Xu et al., 2009]. Because SVMs are a hard classifier (simply predicting class labels without giving a probability) uncertainty is approximated by the distance to the current decision hyperplane. They combine the criterion of choosing points closest to the hyperplane [Campbell et al., 2000, Schohn and Cohn, 2000, Tong and Koller, 2000] with a measure of how representative they are of the data. Xu et al. [2009] report a significant reduction in learning costs while achieving the desired performance.

2.2.4. Expected cost reduction approaches

This approach aims to select the query which minimizes the estimated generalization error. Roy and McCallum [2001] were the first to describe a scheme which seeks to directly minimize classification cost in the next time-step by estimating these costs on the basis of the current parameter estimate. They hint that this approach is 'optimal' for active learning, but the main focus is on finding practical sampling techniques which can be applied to real-world applications such as text classification. However, they use naïve Bayes classifiers and make the assumption that the data selected are independently and identically distributed. This unwarranted independence assumption tends to produce overly sharp posterior distributions, which they overcome by bagging and averaging over the posteriors. This approach was dubbed SELF-CONF by Baram et al. [2004].

Roy and McCallum's [2001] work was followed by Guo and Greiner [2007] who applied a similar approach using logistic regression. In order to minimize computation time they employ an 'optimistic' guess. That is, they assume that the true class label is given by the most likely predicted label. If this turns out to be incorrect when the class label is revealed, they compensate in the next time-step by using the uncertainty sampling

approach of querying the input nearest to the current decision boundary. This algorithm performed well when compared to uncertainty sampling on 17 UCI datasets. Interestingly it compared favourably to uncertainty sampling even on data-sets where the 'optimistic' guess approach (with no correction) performed poorly in relation to uncertainty sampling. When the 'optimistic' guess fails this tends to indicate that the current predicted boundary matches poorly with the true boundary. It therefore makes sense that a query near the boundary such as uncertainty sampling, to compensate for failed guesses, performs better than the same algorithm with no 'correction'.

However, while this approach is computationally simplified by the optimistic guess, it does not fully utilise the learners current class predictions. The relative certainty of each class label for a potential query gives information which is lost when the potential query input is simply evaluated on the most likely class label. The posterior prediction of the expected cost may crucially depend on the class label of the queried input, and the relative likelihood of each outcome informs the overall expected cost associated with that query. It is this property of the expected cost reduction approach which allows the learner to automatically balance queries which are 'explorative' (exploring unqueried regions) and 'exploitative' (refining the current decision boundary) without the need for combining approaches.

The emphasis of this thesis is to apply the expected cost reduction approach in a fully Bayesian setting which does not make the IDD assumption of Roy and McCallum [2001]. Notably there has been some progress in this direction by Zhu et al. [2003] who implement the expected cost reduction approach using Gaussian random fields. Their approach also incorporates semi-supervised learning by training the classifier on both the labelled and unlabelled inputs and shows significant improvement over uncertainty sampling. We develop Zhu et al.'s [2003] approach to active learning in the supervised learning setting, in chapter 3, where Zhu et al.'s [2003] paper is discussed further.

2.2.5. Other approaches

Other criterion have also been used for selecting inputs in active learning. For example, in regression, Cohn et al. [1996] seek to minimize the variance of the prediction over the data

$\mathcal{D}$ by selecting inputs whose target predictions have maximum variance. They show that minimizing the variance of the prediction over $\mathcal{D}$ is equivalent to minimizing the expected mean squared error (MSE) between the predicted target and the true targets (which are continuous in the case of regression) when the target estimates are unbiased. Here the expectation for the MSE is over both the class conditional probabilities and the data $\mathcal{D}$.

Settles et al. [2008] introduce an approach, which they call *greatest change*, as a criterion for query selection. Their approach is specific to the multiple-instance logistic regression which they use to train the classifier. Their definition of greatest change is the input whose class label causes the maximum expected change in the gradient of the objective function within the logistic regression model. Settles et al. [2008] demonstrate that this approach has comparable learning rates to uncertainty sampling, outperforming uncertainty sampling on some data sets, while under performing on others.

Note that the maximum change idea could be thought of more generally as selecting the input which maximally changes the posterior model predictions of $p(y \mid \mathbf{x})$ although this is not what is being directly minimized by Settles et al. [2008].

2.2.6. Performance measures for active learning

Another question associated with active learning is how best to compare the performance of two active learners. Tong and Koller [2001] simply measure the performance of an active learner after some predefined number of observations. This is straightforward but only measures the rate of learning at one fixed point on the learning curve. If in practice if we know how many observations we are likely to make then this is a perfectly acceptable criterion.

More sophisticated measures have also been used. Baram et al. [2004] define a measure of an active learner's performance, which they call the *deficiency*. This is based on the average learning rate of an active learner when compared to the average learning rate of a passive learner and the maximum classification rate achievable given unlimited data or, in the case of pool-based learning, all the observations in the pool. This is illustrated in figure 2.2 which shows classification rate versus the number of inputs queried. If A is defined as

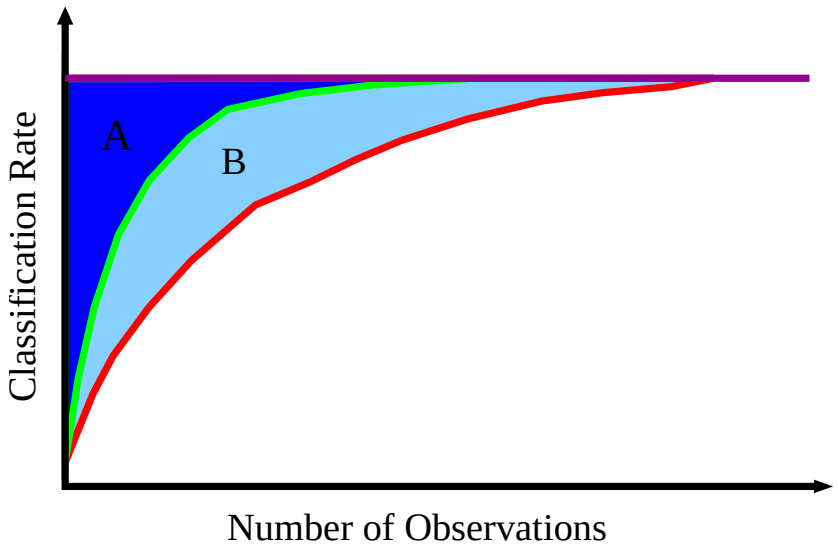


Figure 2.2.: Illustration of the 'deficiency' as defined by Baram et al. [2004] (on which this figure and caption are based). Here the red line indicates the average learning rate for a passive learner receiving IID observations from $p(\mathbf{x})$. The green line represents the average learning rate for an active learner. The magenta line indicates the maximum classification rate achieved from having seen all of the data, in the case of pool-based learning, or having seen a large volume of data in other active learning settings. A is the area between the maximum classification rate and the active learning rate. B is the area between the active learning rate and the passive learning rate. When the green line is above the red line, the deficiency of the active learning algorithm is defined as $\frac{A}{A+B}$.

the area of an active learner's classification rate below the maximum classification rate, and B is defined as the area between the classification rate of an active learner and the average classification rate attained by random sampling, then the global score is measured as $\frac{A}{A+B}$. This deficiency is thus a score in the range $[0, 1)$, where 1 constitutes no improvement over passive learning and scores closer to 0 represent more 'efficient' active learning.

More recently, in the 2010 IJCNN active learning challenge Guyon et al. [2010] used a performance comparison based on the area under the curve (AUC) of the receiver operating characteristic (ROC) curve. The AUC is evaluated after each labelled query to form a curve in a similar way to the learning curve for the classification rate. They use a baseline of a classifier which labels observations randomly, and has a fixed AUC of 0.5 in each step. They call the area under the baseline ARAND. Their top-line is a classifier which makes

perfect label classifications on all data, having AUC of 1 in each step. The area under the top-line is called AMAX. If the area under the learning curve of the active learner is called ALC then they arrive at a global score:

$$\text{globalscore} = \frac{\text{ALC} - \text{ARAND}}{\text{AMAX} - \text{ARAND}} \tag{2.29}$$

Here again the global score is in the range $[0, 1]$ although the best performance is 1 when ALC = AMAX and the worst score is 0 when ALC = AMIN.

The *deficiency* and the ROC performance measures are clearly similar types of measure, in that they both attempt to evaluate the overall learning performance of an active learner rather than simply evaluating performance at a single time-step. Either measure is appropriate for comparison between algorithms on a particular data-set. However, the baseline and top-line for the ROC approach give no indication about the relative ease or difficulty of classification on the particular data-set on which the learner was tested. In contrast the passive learning baseline, and the maximum achievable classification rate, top-line, for the deficiency are specific to the data-set on which the learner is evaluated. It is therefore a better indicator for comparison of active learners, which have been evaluated on different data-sets.

Whether or not the ROC or the classification rate represent the better choice for evaluating the active learner depends on what is known about the classification threshold. When the relative misclassification cost is known with certainty then the classification rate would seem to be the better indicator. However, the ROC evaluates performance across all thresholds and is therefore a measure of robustness to uncertainty in the relative misclassification cost.

2.2.7. Stopping criteria

One of the issues explored in the literature on active learning is when to stop querying new observations. The aim being to stop making new queries when the cost of a new query outweighs the improvement in classification. For some approaches a natural stopping criterion arises. For example the SVM approach to active learning based on Campbell

et al. [2000], Schohn and Cohn [2000], Tong and Koller [2000] has a natural stopping criterion when the decision margin is empty.

The issue of an *automated* stopping criteria is not explored in this thesis, instead learning is simply continued for a fixed number of iterations. In practice the stopping criterion is often affected by external factors and an automated criterion may not be appropriate. For example Settles [2010] states the following about stopping criterion, *"in my own experience, the real stopping criterion for practical applications is based on economic or other external factors, which likely come well before an intrinsic learner-decided threshold"*.

2.3. Discussion

To summarise, we have seen that active learning can be approached in several different ways. It can be viewed as a minimization of the set of consistent solutions, as is the case for QBC approaches, or as the minimization of the uncertainty about individual inputs, as in the uncertainty sampling approach. Finally, there is the expected cost reduction approach which seeks to minimize the expected classification cost directly by effectively performing a 'lookahead', which is the main focus of this thesis.

Uncertainty sampling is a computationally efficient way to perform active learning which can handle large data sets and can be implemented in real-time for many applications. However, as mentioned previously, one of the difficulties with uncertainty based sampling is that it is 'noise seeking' and can select points about which the learner may be uncertain, but which do not inform the overall classification [Roy and McCallum, 2001]. This is somewhat compensated for by Settles and Craven [2008] and Xu et al. [2009] by choosing queries which are also 'representative' of the data. While this method can improve the performance of uncertainty sampling it is not clear how best to balance between the two criteria of 'representativeness' and 'uncertainty'.

The main obstacle to implementation for the QBC approach is the assumption that the data is separable. This assumption is clearly not true for most applications, where there is likely to be considerable overlap between the classes. However, the two main approaches which extend QBC to separable data have both been shown to work well in practice:

1. SVM approaches based on Campbell et al. [2000], Schohn and Cohn [2000], Tong and Koller [2000] which map inputs into a high dimensional space where data is more likely to be linearly separable (by Cover's theorem).

2. Probabilistic versions of QBC such as Argamon-Engelson and Dagan [1999] which draws each committee member according to its posterior probability conditioned on previous observations.

For example, in their paper on the 2010 IJCNN active learning challenge, which was a competition between many active learners on six data sets, Guyon et al. [2010] note that among the active learners presented, QBC and SVM approaches tended to outperform uncertainty based approaches. The winning entry was a probabilistic version of QBC similar to Argamon-Engelson and Dagan [1999] which was based on the QBC paper of Freund et al. [1997].

The reason QBC approaches tend to outperform uncertainty sampling may be because of the need in uncertainty sampling for a trade-off between queries which are 'uncertain' and queries which are 'representative' of the data.

However, the main issue with the QBC approaches (and uncertainty sampling approaches) is that they work directly on parameter space without reference to the specific cost function. This can lead to learning which eliminates areas of parameter space which do not affect misclassification cost. In this thesis we apply the cost reduction approach in a Bayesian setting, which naturally incorporates the cost function by directly seeking to minimise the expected misclassification cost.

One of the difficulties with active learning is that because data are not passively drawn from the underlying distribution, the observed inputs $\mathcal{F} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ cannot be assumed to be IID draws from $p(\mathbf{x})$. In pool-based active learning this is not an issue because it is assumed that the initial pool of unlabelled inputs is an IID draw from the underlying distribution and no further information about $p(\mathbf{x})$ is gleaned from queries to the label oracle. However, in the more general setting where IID inputs are not given, how queried inputs are sampled will affect their resulting distribution. Only in the case of

passive learning can it be assumed that the distribution of the observed inputs $\mathcal{F}$ matches the underlying distribution $p(\mathbf{x})$.

It is for this reason that Balcan et al.'s [2006] agnostic approach (where upper and lower bounds on the generalisation error are based on the model's estimate of $p(\mathbf{x})$) requires inputs to be drawn from regions of the underlying distribution $p(\mathbf{x})$ in IID batches.

In chapter 3 we describe a framework for active learning which extends the cost reduction approach in the Bayesian setting to incorporate a 'query density' which explicitly models the selection procedure of $\mathbf{x}$. This is performed by adjusting the posterior model of $p(\mathbf{x} \mid \mathcal{D})$ based on the observed input *and* how the input was sampled.

In chapter 4 we apply the Bayesian cost reduction approach to pool-based active learning, using Gaussian processes to make inferences. This can be seen as a simplification of the more general framework described in chapter 3.

3. A Bayesian Framework for Active Learning

3.1. Introduction

In this chapter we describe a Bayesian framework for active learning which incorporates a *query density* to explicitly model how new data is to be sampled. The query density is selected by the learner and defines how new inputs are drawn from the underlying distribution. This is therefore a flexible model which can range all the way from passive learning to the type of active learning where the learner makes a *point query* at a specific input. In this model passive learning corresponds to a query density which is uniform across all inputs $\mathbf{x}$ and therefore draws an input from the underlying distribution $p(\mathbf{x})$, while active learning, where a specific input is selected by the learner (with no reference to the underlying distribution), corresponds to a query density which is simply a delta function at the selected input.

The model makes no assumption that queried inputs are IID. Rather, it updates model parameters on the basis of both observations and how those observations were made. This is an expected cost reduction approach to active learning, which seeks select queries which minimise the expected misclassification cost in the next time-step.

3.2. A framework for Bayesian active learning

In this section we present the query density framework for active learning. Although the framework presented is straightforwardly extended to other supervised learning problems in this thesis we focus on binary classification. In section 3.2.1 we describe the query density in detail, before looking at how the parameters of the model are updated for a

single time-step in section 3.2.2. Section 3.2.3 describes the process by which the algorithm evaluates the expected cost for a given query. This is the main step in the algorithm where the idea is to draw hypothetical observations from the model in order to estimate the expected misclassification cost for a given query. In section 3.2.4 we bring these together to form an overview of the algorithm.

We start by briefly introducing some notation for the n^{th} step in the active learning process, assuming that $n-1$ previous observations have already been made. The data from previous steps being denoted by $\mathcal{D}_{n-1} = \{\mathbf{x}_1, y_1, \ldots, \mathbf{x}_i, y_i, \ldots, \mathbf{x}_{n-1}, y_{n-1}\}$, where $\mathbf{x}_i$ and y_i represent the i^{th} observation and corresponding *class label* respectively. For simplicity we make the class labels 0 and 1 so that $y_i \in \{0, 1\}$. The aim is to assign a new input $\mathbf{x}_i$ to C_0 if $y_i = 0$ and C_1 if $y_i = 1$. We also assume from previous steps that $p(\boldsymbol{\theta} \,|\, \mathcal{D}_{n-1})$ is known, where $p(\boldsymbol{\theta} \,|\, \mathcal{D}_{n-1})$ encapsulates the algorithms beliefs about the parameters, $\boldsymbol{\theta}$, given previous observations. Notice that in the first step $p(\boldsymbol{\theta} \,|\, \mathcal{D}_{n-1})$ will simply be our original prior for the model $p(\boldsymbol{\theta})$.

3.2.1. Sample oracle and query density

Rather than selecting a particular datum to query, the learner chooses the query density $q(\mathbf{x}; \phi)$ by selecting its parameters ϕ. This query density is a probability density function and therefore has the properties that: $q(\mathbf{x}; \phi) \geq 0$ for all $\mathbf{x} \in \mathcal{X}$ and $\int q(\mathbf{x}; \phi)\, d\mathbf{x} = 1$. A new input $\mathbf{x}$ is then sampled from the underlying distribution $p(\mathbf{x})$ according to $q(\mathbf{x}; \phi)$. The resulting distribution which we call the *sample oracle* is:

$$Q(\mathbf{x}; \phi) = \frac{q(\mathbf{x}; \phi) p(\mathbf{x})}{\int q(\mathbf{x}; \phi) p(\mathbf{x})\, d\mathbf{x}}. \tag{3.1}$$

To make the nth query the *sample oracle* accepts a parametrisation ϕ_n and returns an $\mathbf{x}_n$ sampled according to (3.1). $Q(\mathbf{x}; \phi)$ is the likelihood of the sample oracle returning a particular $\mathbf{x}$ given a particular query density (as defined by ϕ) and underlying distribution $p(\mathbf{x})$. Since $p(\mathbf{x})$ itself is unknown to the learner the distribution $Q(\mathbf{x}; \phi)$ is modelled by:

$$Q(\mathbf{x} \,|\, \boldsymbol{\theta}, \phi) = \frac{q(\mathbf{x}; \phi) p(\mathbf{x} \,|\, \boldsymbol{\theta})}{\int q(\mathbf{x}; \phi) p(\mathbf{x} \,|\, \boldsymbol{\theta})\, d\mathbf{x}} \tag{3.2}$$

where $p(\mathbf{x} \mid \boldsymbol{\theta})$ is the learner's model for $p(\mathbf{x})$ and $Q(\mathbf{x} \mid \boldsymbol{\theta}, \phi)$ is the learner's model of the sample oracle.

If $q(\mathbf{x})$ is very narrow then the sample oracle only draws $\mathbf{x}$ from a very specific region of $p(\mathbf{x})$, on the other hand if it is very wide then the sample oracle draws $\mathbf{x}$ much more broadly from the underlying distribution. As described above, the extreme cases of a wide query and narrow query are passive learning and *point query* active learning respectively.

To summarise, the query density $q(\mathbf{x}; \phi)$ combines with the underlying distribution $p(\mathbf{x})$ to form the *sample oracle*, which is analogous to the *label oracle*. While the label oracle returns a class label y for a given input $\mathbf{x}$, the sample oracle returns a datum $\mathbf{x}$ given the query density parameters ϕ chosen by the learner.

3.2.2. Parameter update

We now look at how the parameters are updated in the model, and inferences are made. Because $p(\mathbf{x})$ and $p(y \mid \mathbf{x})$ are not known these are modelled by $p(\mathbf{x}_n \mid \boldsymbol{\theta})$ and $p(y_n \mid \mathbf{x}_n, \boldsymbol{\theta})$ respectively, and the sample oracle for a given ϕ is then modelled by $Q(x \mid \boldsymbol{\theta}, \phi)$ as in equation (3.2). This models the distribution from which the observation $\mathbf{x}_n$ is drawn, and forms the *likelihood* of observing $\mathbf{x}_n$ in the Bayesian update of the parameters:

$$p(\boldsymbol{\theta} \mid \mathbf{x}_n, \mathcal{D}_{n-1}, \Phi_n) \propto Q(\mathbf{x}_n \mid \boldsymbol{\theta}, \phi_n) p(\boldsymbol{\theta} \mid \mathcal{D}_{n-1}, \Phi_{n-1}) \tag{3.3}$$

where the constant of proportionality is found by integrating the r.h.s. over $\boldsymbol{\theta}$. Here we include the specific dependency of the prior and posterior on the queries made through the sequence of query density parameters $\Phi_n = \{\phi_n, \phi_{n-1}, \ldots, \phi_1\}$.

The class label $y_n \in \{0, 1\}$ is supplied by the label oracle which provides a sample from $p(y_n \mid \mathbf{x}_n)$. The true label oracle is not known to the learner. Instead, the model distribution $p(y_n \mid \mathbf{x}_n, \boldsymbol{\theta})$ forms the *likelihood* of observing y_n in the Bayesian update:

$$p(\boldsymbol{\theta} \mid \mathcal{D}_n, \Phi_n) \propto p(y_n \mid \mathbf{x}_n, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \mathbf{x}_n, \mathcal{D}_{n-1}, \Phi_n) \tag{3.4}$$

		Predicted	
		0	1
Actual	0	0	λ_{01}
	1	λ_{10}	0

Table 3.1.: Cost Matrix

where again the constant of proportionality may be found by integration. Clearly (3.3) and (3.4) may be combined as:

$$p(\boldsymbol{\theta} \,|\, \mathcal{D}_n, \Phi_n) \propto p(y_n \,|\, \mathbf{x}_n, \boldsymbol{\theta}) Q(\mathbf{x}_n \,|\, \boldsymbol{\theta}, \phi_n) p(\boldsymbol{\theta} \,|\, \mathcal{D}_{n-1}, \Phi_{n-1}) \tag{3.5}$$

where $p(y_n \,|\, \mathbf{x}_n, \boldsymbol{\theta}) Q(\mathbf{x}_n \,|\, \phi_n, \boldsymbol{\theta})$ is the combined likelihood of observing $\{\mathbf{x}_n, y_n\}$.

Inferences are made about the probability of an input $\mathbf{x}$ belonging to class C_1 by averaging the predictive likelihood $p(y = 1 \,|\, \mathbf{x}, \boldsymbol{\theta})$ weighted by the learned posterior:

$$p(y = 1 \,|\, \mathbf{x}, \mathcal{D}_n, \Phi_n) = \int p(y = 1 \,|\, \mathbf{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta} \,|\, \mathcal{D}_n, \Phi_n) \, d\boldsymbol{\theta}. \tag{3.6}$$

Given the probability of class membership predicted by the learner in equation (3.6) the optimal assignment of class labels, and the corresponding expected cost is dependent on the specific cost function for the model. In general the cost associated with misclassifying y_n may be dependent on $\mathbf{x}_n$, however, in this thesis we use the cost function (shown in table 3.1) which assigns a cost λ_{01} to inputs which are actually in class 0 but are predicted to be in class 1 and λ_{10} to inputs which are actually class 1 but predicted to be in class 0. Correctly classified inputs are assumed to have zero cost.

For this cost function the assignment that minimises the expected misclassification cost [Duda et al. [2000] pp25] is given by the following decision function:

$$\mathcal{A}(\mathbf{x} \,|\, \mathcal{D}_n) = \begin{cases} 1 & \text{if } p(y = 1 \,|\, \mathbf{x}, \mathcal{D}_n, \Phi_n) \geq \lambda \\ 0 & \text{otherwise.} \end{cases} \tag{3.7}$$

where λ is the decision threshold defined by $\lambda = \lambda_{01}/(\lambda_{01} + \lambda_{10})$. The estimated expected misclassification cost or risk for an input $\mathbf{x}$ is therefore:

$$R(\mathbf{x} \,|\, \mathcal{D}_n, \Phi_n) = \delta_{0,\mathcal{A}(\mathbf{x})} p(y = 1 \,|\, \mathbf{x}, \mathcal{D}_n, \Phi_n)\lambda_{10} + \delta_{1,\mathcal{A}(\mathbf{x})} p(y = 0 \,|\, \mathbf{x}, ...\mathcal{D}_n, \Phi_n)\lambda_{01} \quad (3.8)$$

where δ is the Kronecker delta and we have suppressed the explicit dependence of $\mathcal{A}(\mathbf{x})$ on $\mathcal{D}_n$.

3.2.3. Expected misclassification cost

In order for the active learner to query an observation, according to the query density framework, it is necessary for the learner to select the query parameters ϕ_n given to the query oracle. The expected misclassification cost associated with given query parameters ϕ_n is estimated by performing a lookahead, based on the learner's current beliefs which are encapsulated by the prior $p(\boldsymbol{\theta} \,|\, \mathcal{D}_{n-1}, \Phi_{n-1})$. In the lookahead step the hypothetical observations are drawn from the model, the update based on the hypothetical observations is performed and the posterior expected cost is evaluated, which permits the query with the lowest expected misclassification cost to be located.

The process of drawing a 'hypothetical' observation begins with a sample $\boldsymbol{\theta}'$ from the prior $p(\boldsymbol{\theta} \,|\, \mathcal{D}_{n-1}, \Phi_{n-1})$. The query oracle $Q(\mathbf{x} \,|\, \phi_n, \boldsymbol{\theta}')$ is then defined and a query sample $\mathbf{x}'_n$ is drawn from it. This is then given to the label oracle $p(y \,|\, \mathbf{x}'_n, \boldsymbol{\theta}')$ from which y'_n is drawn. The dash is used to emphasise that these variables are 'hypothetical', in the sense that they are generated from the model as opposed to being externally observed. The hypothetical observation is then added to the data to give $\mathcal{D}'_n = \mathcal{D}_{n-1} \cup \{\mathbf{x}'_n, y'_n\}$.

The lookahead applies the same update as described in section 3.2.2 in order to find $R(\mathbf{x} \,|\, \mathcal{D}'_n, \Phi'_n)$ for the hypothetical observation $(\mathbf{x}'_n,\ y'_n)$. However, since $R(\mathbf{x} \,|\, \mathcal{D}'_n, \Phi'_n)$ depends on the particular $\mathbf{x}'_n$ and y'_n returned by the oracles, the estimated cost should be averaged over the query density:

$$R(\mathbf{x} \,|\, \Phi'_n) = \iint R(\mathbf{x} \,|\, \mathcal{D}'_n, \Phi'_n) p(y'_n \,|\, \mathbf{x}'_n) Q(\mathbf{x}'_n \,|\, \phi)\, d\mathbf{x}'_n dy'_n. \quad (3.9)$$

The average misclassification cost anticipated when making a query with query parameters ϕ'_n is therefore:

$$R(\Phi'_n) = \int R(\mathbf{x} \,|\, \Phi'_n) p(\mathbf{x}) \, d\mathbf{x} \tag{3.10}$$

$$= \iiint R(\mathbf{x} \,|\, \mathcal{D}'_n, \Phi'_n) p(y'_n \,|\, \mathbf{x}'_n) Q(\mathbf{x}'_n \,|\, \phi'_n) p(\mathbf{x}) \, d\mathbf{x}'_n dy'_n d\mathbf{x} \tag{3.11}$$

Since $p(\mathbf{x})$ and $p(y \,|\, \mathbf{x})$ are unknown we estimate $R(\Phi'_n)$ by $\tilde{R}(\Phi'_n)$ using models based on $\mathcal{D}_{n-1}$:

$$\tilde{R}(\Phi'_n) = \iiint R(\mathbf{x} \,|\, \mathcal{D}'_n, \Phi_n) p(y'_n, \mathbf{x}'_n \,|\, \mathcal{D}_{n-1}, \Phi_n) p(\mathbf{x} \,|\, \mathcal{D}_{n-1}) \, d\mathbf{x}'_n dy'_n d\mathbf{x} \tag{3.12}$$

where

$$p(y'_n, \mathbf{x}'_n \,|\, \mathcal{D}_{n-1}, \Phi_n) = \int p(y'_n, \mathbf{x}'_n \,|\, \boldsymbol{\theta}, \phi_n) p(\boldsymbol{\theta} \,|\, \mathcal{D}_{n-1}, \Phi_{n-1}) \, d\boldsymbol{\theta}. \tag{3.13}$$

The joint density can be factorised as:

$$p(y'_n, \mathbf{x}'_n \,|\, \boldsymbol{\theta}, \phi_n) = p(y'_n \,|\, \mathbf{x}'_n, \boldsymbol{\theta}) Q(\mathbf{x}'_n \,|\, \boldsymbol{\theta}, \phi_n) \tag{3.14}$$

and

$$p(\mathbf{x} \,|\, \mathcal{D}_{n-1}, \Phi_{n-1}) = \int p(\mathbf{x} \,|\, \boldsymbol{\theta}) p(\boldsymbol{\theta} \,|\, \mathcal{D}_{n-1}, \Phi_{n-1}) \, d\boldsymbol{\theta}. \tag{3.15}$$

The integral in equation (3.12) is estimated by performing multiple hypothetical observations from the model. In this way an estimate of the expected misclassification cost $\tilde{R}(\Phi'_n)$ is evaluated for each potential ϕ'_n and the parameter ϕ_n is chosen which minimises the estimated expected cost:

$$\phi_n = \underset{\phi}{\operatorname{argmin}} \, \tilde{R}(\Phi'_n) \tag{3.16}$$

Algorithm 2 Bayesian active learning

Require: Prior on parameters $p(\boldsymbol{\theta})$

1: **for** $n = 1, \ldots, N$ — N learning iterations
2: $\quad \phi_n = \text{argmin}_\phi \text{ESTIMATED-RISK}(\phi)$ — Algorithm selects ϕ_n: *see algorithm* (3)
3: $\quad \Phi_n = \Phi_{n-1} \cup \phi_n$ — Add ϕ_n to the list
4: $\quad \mathbf{x}_n \sim Q(\mathbf{x}; \phi_n)$ — Sample oracle returns $\mathbf{x}_n$: *see equation* (3.1)
5: $\quad y_n \sim p(y \mid \mathbf{x}_n)$ — Label oracle returns y_n
6: $\quad \mathcal{D}_n = \mathcal{D}_{n-1} \cup \{\mathbf{x}_n, y_n\}$ — Add $\{\mathbf{x}_n, y_n\}$ to the data
7: $\quad p(\boldsymbol{\theta} \mid \mathcal{D}_n, \Phi_n) \propto p(y_n, x_n \mid \boldsymbol{\theta}, \phi_n) p(\boldsymbol{\theta} \mid \mathcal{D}_{n-1}, \Phi_{n-1})$ — Update parameters $\boldsymbol{\theta}$
8: **end for**

3.2.4. Algorithm overview

Algorithm 2 shows an overview of active learning in the query density framework. In each time-step the learner starts with the prior on the parameters $p(\boldsymbol{\theta} \mid \mathcal{D}_{n-1}, \Phi_{n-1})$. This prior defines the learners current beliefs about the model and is used by the learner to estimate the expected risk associated with a particular query parameter ϕ. The evaluation of the expected risk is the main step and is described separately in algorithm 3. The query parameter ϕ_n for which the expected risk is minimised is then selected (line 2). The query parameter ϕ_n is then given to the sample oracle $Q(\mathbf{x}; \phi_n)$ (see equation (3.2)) which returns an input $\mathbf{x}_n$ (line 4). This input in turn is given to the label oracle which returns a class label y_n from the true distribution $p(y \mid \mathbf{x}_n)$ (line 5). The new input-label pair $\{\mathbf{x}_n, y_n\}$ is then added to the data, and finally the parameters are updated given the new observation (line 7). This posterior then becomes the prior in the next time-step and completes one iteration for the active learner.

Algorithm 3 shows the procedure by which $\tilde{R}(\Phi'_n)$ is evaluated. This is the learner's estimate of the risk associated with a given query parameter ϕ. As described in section 3.2.3, hypothetical input-label pairs, $\mathbf{x}_n^{(k)\prime}$ and $y_n^{(k)\prime}$, are sampled from the model (lines 3 and 4) by first sampling a $\boldsymbol{\theta}^{(k)\prime}$ from the prior distribution (line 2). This hypothetical input-label pair is then added to the data (line 5) to form a hypothetical data set $\mathcal{D}_n^{(k)\prime}$. Line 6 calls algorithm 4 which estimates the expected risk for an arbitrary input $\mathbf{x}$ given the hypothetical input-label pair. The likelihood for the hypothetical input-label pair is then evaluated by integrating over the prior distribution of the parameters (line 7). The specific estimated risk and the likelihood are then combined to find the average risk for all

Algorithm 3 ESTIMATED-RISK(ϕ)

Require: $p(\boldsymbol{\theta} \,|\, \mathcal{D}_{n-1}, \Phi_{n-1})$ $\quad$ $\Phi'_n := \phi \cup \Phi_{n-1}$

Draw K hypothetical samples:

1: `for` $k = 1$ to K

2: $\quad \boldsymbol{\theta}^{(k)'} \sim p(\boldsymbol{\theta} \,|\, \mathcal{D}_{n-1}, \Phi_{n-1})$ $\quad$ Sample $\boldsymbol{\theta}^{(k)'}$

3: $\quad \mathbf{x}^{(k)'} \sim Q(\mathbf{x} \,|\, \phi, \boldsymbol{\theta}^{(k)'})$ $\quad$ Sample $\mathbf{x}^{(k)'}$

4: $\quad y^{(k)'} \sim p(y \,|\, \mathbf{x}', \boldsymbol{\theta}^{(k)'})$ $\quad$ Sample $y^{(k)'}$

Add samples to the data:

5: $\quad \mathcal{D}_n^{(k)'} = \mathcal{D}_{n-1} \cup \{\mathbf{x}^{(k)'}, y^{(k)'}\}$

Evaluate specific estimated risk for a given sample:

6: $\quad \tilde{R}(\mathbf{x} \,|\, \mathcal{D}^{(k)'}, \Phi'_n) = \text{SPECIFIC-ESTIMATED-RISK}(\mathbf{x}, \mathcal{D}_n^{(k)'}, \Phi'_n)$ $\quad$ See algorithm 4

Integrate over prior to find likelihood:

7: $\quad p(\mathbf{x}^{(k)'}, y^{(k)'} \,|\, \mathcal{D}_{n-1}, \Phi'_n) = \int p(\mathbf{x}^{(k)'}, y^{(k)'} \,|\, \boldsymbol{\theta}, \phi) p(\boldsymbol{\theta} \,|\, \mathcal{D}_{n-1}, \Phi_{n-1}) d\mathbf{x}$ $\quad$ eq (3.13)

8: `end`

Average over samples to find risk for arbitrary $\boldsymbol{x}$:

9: $\tilde{R}(\mathbf{x} \,|\, \phi) = \sum_{k=1}^{K} R(\mathbf{x} \,|\, \mathcal{D}_n^{(k)'}, \Phi'_n) p(\mathbf{x}^{(k)'}, y^{(k)'} \,|\, \mathcal{D}_{n-1}, \Phi'_n)$ $\quad$ eq (3.9)

Integrate over prior to find underlying distribution:

10: $p(\mathbf{x} \,|\, \mathcal{D}_{n-1}, \Phi_{n-1}) = \int p(\mathbf{x} \,|\, \boldsymbol{\theta}) p(\boldsymbol{\theta} \,|\, \mathcal{D}_{n-1}, \Phi_{n-1}) d\boldsymbol{\theta}$ $\quad$ eq (3.15)

Integrate over $\boldsymbol{x}$ to find risk for ϕ:

11: $\tilde{R}(\phi) = \int \tilde{R}(\mathbf{x} \,|\, \phi) p(\mathbf{x} \,|\, \mathcal{D}_{n-1}, \Phi_{n-1}) d\mathbf{x}$ $\quad$ eq (3.12)

12: `return:` $\tilde{R}(\phi)$

Algorithm 4 SPECIFIC-ESTIMATED-RISK($\mathbf{x}, \mathcal{D}_n^{(k)'}, \Phi'_n$)

Require: $\Phi'_n := \phi \cup \Phi_{n-1}$, prior $p(\boldsymbol{\theta} \,|\, \mathcal{D}_{n-1}, \Phi_{n-1})$ and data $\mathcal{D}_n^{(k)'}$

Draw samples from posterior:

1: $p(\boldsymbol{\theta} \,|\, \mathcal{D}_n^{(k)'}, \Phi'_n) \propto p(\mathbf{x}^{(k)'}, y^{(k)'} \,|\, \boldsymbol{\theta}, \phi)\, p(\boldsymbol{\theta} \,|\, \mathcal{D}_{n-1}, \Phi_{n-1})$ $\quad$ eq (3.5)

Estimate posterior class predictions:

2: $p(\mathbf{y} = 1 \,|\, \mathcal{D}_n^{(k)'}, \Phi'_n) \approx \sum_{\boldsymbol{\theta}} p(\mathbf{y} = 1 \,|\, \mathbf{x}, \boldsymbol{\theta})\, p(\boldsymbol{\theta} \,|\, \mathcal{D}_n^{(k)'}, \Phi'_n)$ $\quad$ eq (3.6)

Evaluate posterior decision function:

3: $A(\mathbf{x} \,|\, \mathcal{D}_n^{(k)'}, \Phi'_n)$ for $p(\mathbf{y} = 1 \,|\, \mathbf{x}, \mathcal{D}_n^{(k)'}, \Phi'_n)$ $\quad$ eq (3.7)

Estimate risk for given sample:

4: $\tilde{R}(\mathbf{x} \,|\, \mathcal{D}^{(k)'}, \Phi'_n)$ for $A(\mathbf{x} \,|\, \mathcal{D}_n^{(k)'}, \Phi'_n)$ and $p(\mathbf{y} = 1 \,|\, \mathbf{x}, \mathcal{D}_n^{(k)'}, \Phi'_n)$ $\quad$ eq (3.8)

5: `return:` $\tilde{R}(\mathbf{x} \,|\, \mathcal{D}^{(k)'}, \Phi'_n)$

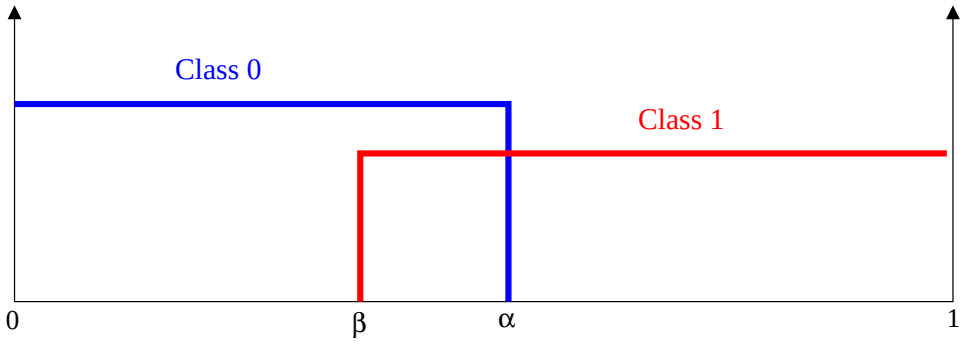


Figure 3.1.: Schematic of the probabilistic high-low game, consisting of two overlapping uniform densities for classes C_0 and C_1 in the interval $[0, 1)$, each defined by one parameter, α and β respectively.

hypothetical observations (line 9). The underlying distribution is evaluated by integrating over the prior distribution for the parameters (line 10). The risk for an arbitrary input $\tilde{R}(\mathbf{x} \mid \phi)$ is then integrated over the models estimate of the underlying distribution to give the over all expected risk for a given query $\tilde{R}(\mathbf{x})$.

Algorithm 4 describes the procedure by which the specific estimated risk is evaluated. For a given hypothetical sample $\mathcal{D}_n^{(k)'}$ the posterior class predictions are estimated (line 2) by first drawing samples from the posterior distribution associated with $\mathcal{D}_n^{(k)'}$ (line 1). Assuming the same cost function as in section 3.2.3 the decision function is then evaluated based on the posterior class predictions (line 3). The specific estimated risk is then found as in equation (3.8) (line 4).

Implementation of this active learning scheme requires successive calculations of the parameter posterior and the expected cost. Although the expression (3.5) for the posterior is a straightforward application of Bayes' rule, the presence of the query function renders finding conjugate families unlikely in all but very special circumstances and Monte Carlo sampling or approximate methods are likely to be required in general.

3.3. Probabilistic high-low game

We now demonstrate the framework on a simple toy problem where parameter space is just two dimensional. The problem is an extension of the high-low game on which Seung et al.

[1992] demonstrate their QBC model. The original high-low game is a one dimensional problem containing two uniform classes which meet at some unknown threshold γ, where form of the high-low game is completely separable and the target given by the *label* oracle is always correct, allowing hypotheses to be kept or ruled out completely on the basis of each new queried input. We consider a straightforward non-separable extension of this model, which we call the probabilistic high-low game. This is similar to the original high-low game with the extension of allowing some overlap between the classes. Our results indicate that the active Bayes algorithm performs significantly better than passive learning even when the overlap region is wide, covering over 30% of the input space.

For the probabilistic high-low game we assume that class C_0 is uniform on $[0, \alpha)$ and class C_1 is uniform on $[\beta, 1)$, see figure 3.1. For simplicity we assume that the relative frequency of the two classes, w_0 and w_1, are known and $P(C_0) = w_0$ and $P(C_1) = w_1$. This model has only two unknown parameters, α and β, and for particular α, β we have:

$$p(x \,|\, \alpha, \beta) = \frac{w_0}{\alpha} \mathcal{I}_{[0,\alpha)}(x) + \frac{w_1}{1-\beta} \mathcal{I}_{[\beta,1)}(x). \tag{3.17}$$

where $\mathcal{I}_s(x)$ is the indicator function that is 1 if $x \in s$ and 0 otherwise. Equation (3.17) defines $p(x \,|\, \boldsymbol{\theta})$ in (3.2) with the parameters $\boldsymbol{\theta} = \{\alpha, \beta\}$. In order to ensure that the two classes at least meet we add the condition that $\beta < \alpha$. When $\alpha = \beta$ there is no overlap between the classes and the problem simplifies to the separable high-low game, with $\alpha = \beta = \gamma$. Given this model $p(y = 1 \,|\, x, \alpha, \beta)$ is also known. As can be seen from figure 3.1 an observation may lie in one of three regions: When $x < \beta$ it is definitely in class 0, when $x \geq \alpha$ it definitely lies in class 1. The third region is the overlap region where:

$$p(y = 1 \,|\, x, \alpha, \beta) = \frac{\alpha w_1}{\alpha w_1 + (1-\beta) w_0} \quad \text{for } \beta < x < \alpha \tag{3.18}$$

which depends on the relative frequency of the two classes, w_0 and w_1. We define the query density $q(x; \phi)$ to be uniform between q_- and q_+, and zero elsewhere. $Q(x \,|\, \boldsymbol{\theta}, \phi)$ will therefore sample data from $p(x \,|\, \boldsymbol{\theta})$, but only in the interval $[q_-, q_+)$.

The algorithm is initialised with a prior $p(\boldsymbol{\theta}) = p(\alpha, \beta)$. We take this to be uniform on the region $\beta \leq \alpha$:

$$p(\alpha, \beta) = \begin{cases} 2 & \text{if} \quad \beta \leq \alpha \\ 0 & \text{otherwise.} \end{cases} \tag{3.19}$$

Having formed the basis of the model by defining $p(x \,|\, \alpha, \beta)$ and $p(y = 1 \,|\, x, \alpha, \beta)$ and having defined a prior $p(\alpha, \beta)$, we now reach the major part of the algorithm in which the learner must decide on a suitable ϕ. Having made $n-1$ observations, the learner performs a lookahead to estimate the expected cost associated with the nth observation, choosing the interval that has the minimum expected cost (line 2 of Algorithm 2). This is achieved by sampling a hypothetical observation (x', y') from the sample and label oracles, and finding the average expected cost (3.12) associated with each $q(\cdot)$ parametrised by $\phi \equiv [q_-, q_+)$. For each ϕ the aim is to find the corresponding expected cost $\tilde{R}(\Phi_n)$, in order to find $\phi_n = \operatorname{argmin}_\phi \tilde{R}(\Phi'_n)$.

Following the same steps as algorithm 3, the first step (which corresponds to line 3 of algorithm 3) is to sample hypothetical parameters, α', β' from the prior $p(\alpha, \beta \,|\, \mathcal{D}_{n-1}, \Phi_{n-1})$. Given these parameters a *model* of the sample oracle, $Q(x' \,|\, \alpha', \beta', \phi'_n)$ is defined:

$$Q(x' \,|\, \alpha', \beta', \phi'_n) = \frac{q(x; \phi'_n)\, p(x \,|\, \alpha', \beta')}{\int q(x; \phi'_n)\, p(x \,|\, \alpha', \beta')\, dx} \tag{3.20}$$

where the underlying distribution $p(x)$ is approximated by $p(x \,|\, \alpha', \beta')$. A hypothetical input x' is then drawn from the sample oracle. Similarly, given x' a hypothetical target label y' is drawn from $p(y' \,|\, x', \alpha', \beta')$, our *model* of the label oracle (line 4 of algorithm 3).

Having drawn x' and y' we now perform a Bayesian update as in equation (3.5) to find the posterior distribution, this corresponds to line 6 of algorithm 3:

$$p(\alpha, \beta \,|\, x', y', \phi) \propto p(y' \,|\, x', \alpha, \beta) Q(x' \,|\, \alpha', \beta', \phi'_n) p(\alpha, \beta) \tag{3.21}$$

For this hypothetical posterior we now aim to find the corresponding 'optimal classifier' in order to evaluate the corresponding expected cost. Following the general case we aim

to find the decision boundary and corresponding decision function which minimises the expected cost for a given λ_{01} and λ_{10}. The class conditional probabilities are averaged over the posterior distribution of the parameters as in equation (3.6):

$$p(y = 1 \,|\, x = \eta) = \iint p(y = 1 \,|\, x = \eta, \alpha, \beta) p(\alpha, \beta \,|\, y', x', \phi) d\alpha d\beta \tag{3.22}$$

and the integral is estimated by using a Monte Carlo sample. In the work reported here we use 1600 hypothetical samples of x' and y'.

To minimise the expected misclassification cost, as described in equations (3.7) and (3.8), we find the boundary η between the decision regions such that:

$$p(y = 1 \,|\, x = \eta, x', y') \approx \iint p(y = 1 \,|\, x = \eta, \alpha, \beta) p(\alpha, \beta \,|\, y', x', \phi) d\alpha d\beta = \lambda, \tag{3.23}$$

where $p(y = 1 \,|\, x = \eta, x', y')$ is evaluated by equation (3.6). The corresponding decision function $\mathcal{A}(x)$ is given by:

$$\mathcal{A}(x) = \begin{cases} 1 & \text{if } x \geq \eta \\ 0 & \text{otherwise} \end{cases} \tag{3.24}$$

which classifies $y = 1$ for all $x \geq \eta$ and 0 otherwise. Given the posterior distribution $p(\alpha, \beta \,|\, x', y', \phi)$ equation (3.24) represents the optimal decision function for class conditional probabilities given by equation (3.23). Having obtained $\mathcal{A}(x)$ the estimated expected misclassification cost $\tilde{R}(x \,|\, \mathcal{D}'_n, \Phi'_n)$ is then given by equation (3.8) (line 9 of algorithm 3). Finally the average cost $\tilde{R}(\Phi_n)$ over $p(x)$ and the hypothetical observations $\{x', y'\}$ is found by equation (3.10) (line 12 of algorithm 3). The query parameter is then set to $\phi_n = \text{argmin}_\phi \tilde{R}(\Phi'_n)$ (completing algorithm 3). Returning to algorithm 1 the query parameters are then given to the sample oracle and label oracle, in order to make the n^{th} query $\{x_n, y_n\}$. The model parameters are updated according to equation (3.5) and this completes a single iteration in the learning process.

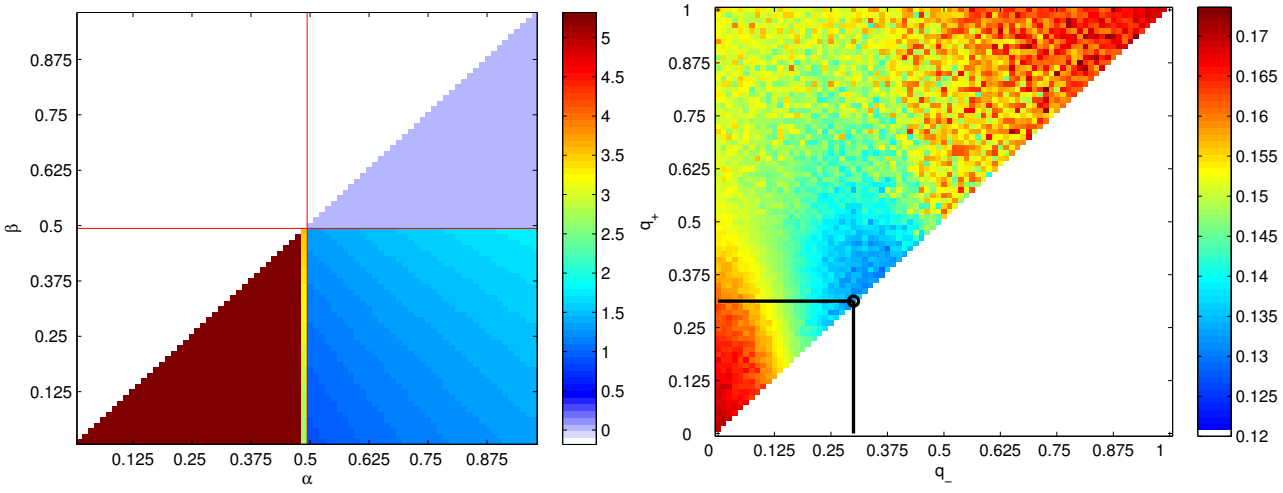


Figure 3.2.: (a) Posterior density $p(\alpha, \beta \,|\, x_1 = 0.5, y_1 = 1)$ after a single observation $x_1 = 0.5$, indicated by the red cross-hair. (b) Expected cost $\tilde{R}(\Phi_1)$ as a function of q_- and q_+, using $p(\alpha, \beta \,|\, x_1 = 0.5, y_1 = 1)$ as the prior. The black circle indicates the minimum.

3.4. Results for the high-low game

First we give an example of the prior and expected cost plots for a single time-step. Here, the algorithm was initialised with a uniform prior (equation (3.19)) and one input $x_1 = 0.5$ was queried by the learner and labelled by the oracle as being in class 1. Figure 3.2(a) shows the posterior distribution $p(\alpha, \beta \,|\, \mathcal{D}_1, \Phi_1)$ having made this single observation, (for a uniform prior $p(\alpha, \beta)$ and a single *unlabelled* input $\mathbf{x}_1$, the analytic form of $p(\alpha, \beta \,|\, x_1, \Phi_1)$ is found in appendix B and was used to confirm that the sampling approach approximates the true distribution). The posterior $p(\alpha, \beta \,|\, \mathcal{D}_1, \Phi_1)$ then becomes the prior in the next time-step and represents the learner's updated beliefs about α, β. As β is the lower boundary for class 1, then $\beta \leq 0.5$ and the posterior drops to zero for all $\beta > 0.5$.

Given these current beliefs, the algorithm now performs a lookahead to evaluate the expected cost. This is done by sampling hypothetical 'true' values α', β', from figure 3.2(a) and hypothetical observations x', y' from the model in order to evaluate the expected cost $\tilde{R}(q_-, q_+)$ associated with each choice of sample region $[q_-, q_+)$. These are shown in figure 3.2(b). As is clear from the figure, the $[q_-, q_+)$ for which the expected cost is minimised lies on the main diagonal, where $q_- = q_+ \approx 0.3$. This figure was generated on an 80 by

80 pixel grid of possible values for q_- and q_+. With the constraint that $q_- \leq q_+$ this corresponds to 3240 possible parametrisations ϕ. For each possible ϕ, 1600 hypothetical samples were drawn to estimate the expected cost. Despite this number of samples figure 3.2(b) still shows signs of noise in the estimate and more samples would be required for a smoother estimate.

For the probabilistic high-low game, in every example we encountered, the minimum expected cost was found to occur when $q_- = q_+$, implying that the learner should query at a single input rather than probabilistically from a region. In the following we therefore assume that $q_- = q_+ \equiv q^{\pm}$ which allows a very significant narrowing of the space to be searched to find the optimum query. When queries are restricted to single inputs the cost surface 3.2(b) becomes one-dimensional, as shown in figure 3.3(a).

This does however raise the question of whether there are examples of binary classification problems where it is advantageous to select from a region, rather than simply query data at a single input. To answer this question affirmatively we describe in section 3.5 examples where querying over a region gives some advantage over selection at an input.

We illustrate the learning algorithm on two versions of the probabilistic high-low game, with the two classes spanning the finite region $[0, 1)$. In one case the overlap region was narrow, $\alpha = 0.2375$ and $\beta = 0.1875$, while in the other the overlap region is much wider, $\alpha = 0.6875$ and $\beta = 0.375$. In both cases we choose $w_0 = w_1 = 0.5$ and $\lambda_{01} = \lambda_{10} = 1$ which gives a decision threshold $\lambda = 0.5$. In the overlap region the probability of choosing C_1, given by equation (3.18), is 0.23 and 0.52 in the narrow and wide overlap cases respectively.

Figures 3.3-3.13 show steps 1 to 10 and step 30 respectively, of a single run in the case of a wide overlap region. In each step we see; the prior distribution $p(\boldsymbol{\theta} \,|\, \mathcal{D}_{n-1}, \Phi_{n-1})$ (panel (b)), the predicted cost for a given query $\tilde{R}(q^{\pm})$ (panel (a)), the posterior distribution having made the observation $p(\boldsymbol{\theta} \,|\, \mathcal{D}_n, \Phi_n)$ (panel (d)) and the query location (panel (c)). Blue and red circles indicate data from class 0 and class 1 respectively, and the vertical lines indicate the true class boundaries. In each step the posterior from the previous step becomes the prior for the next step.

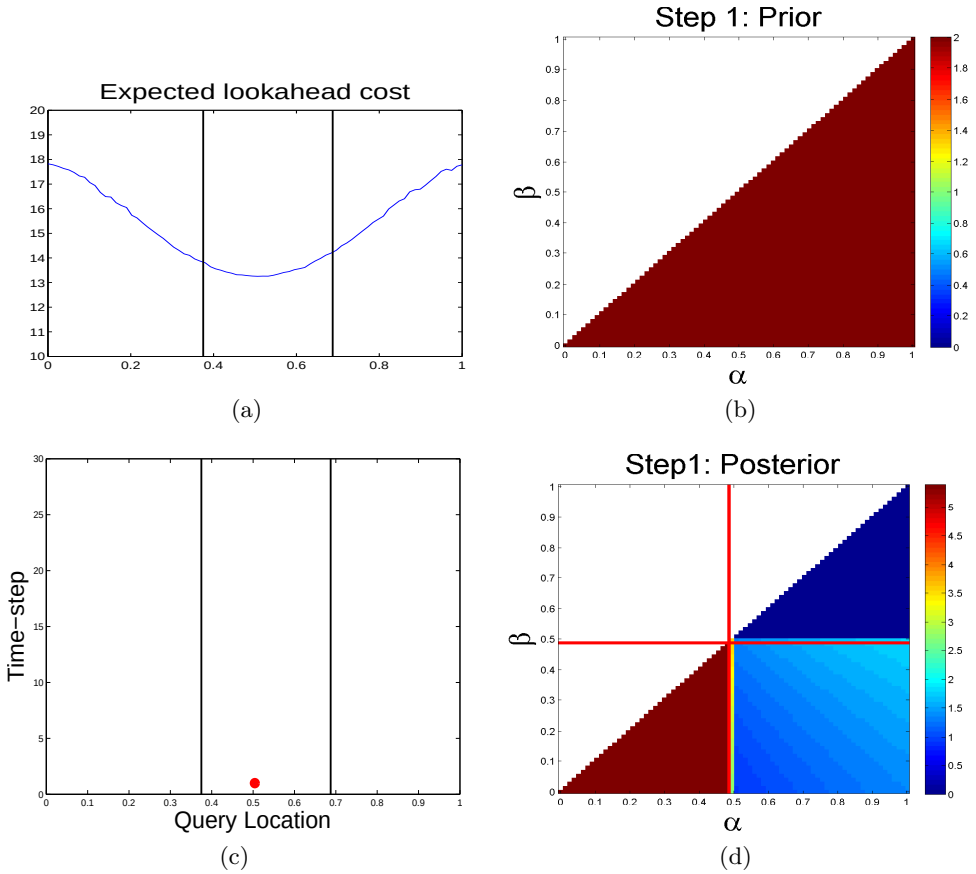
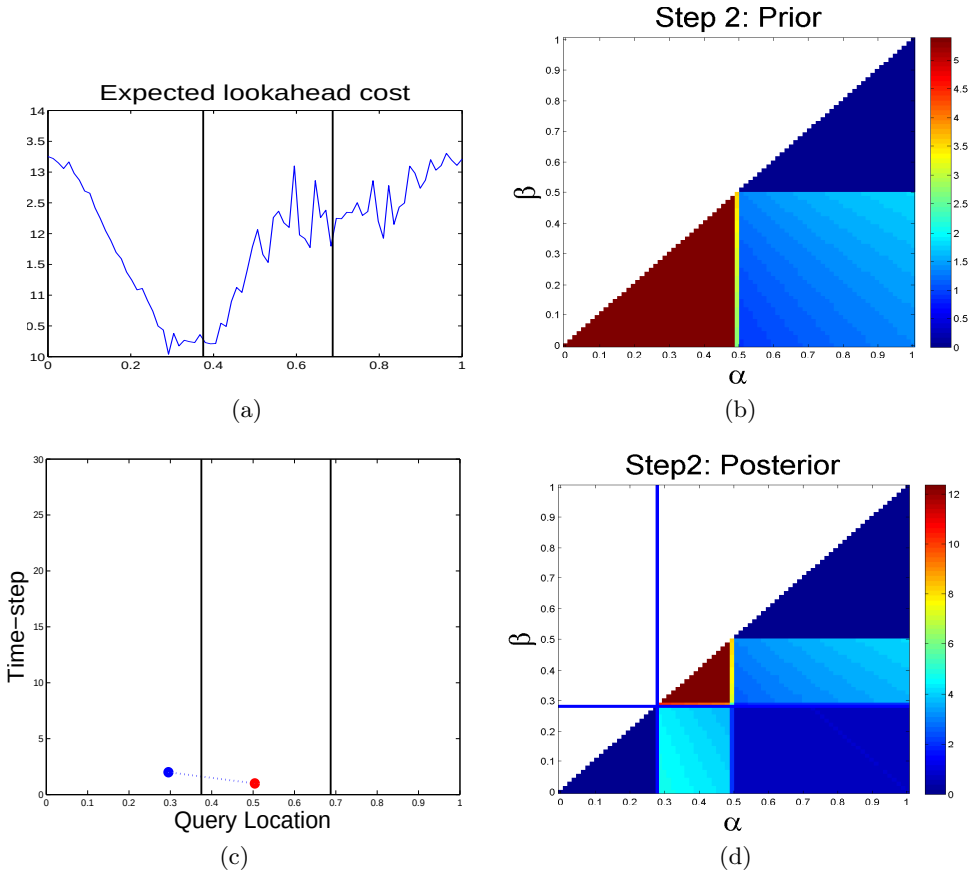
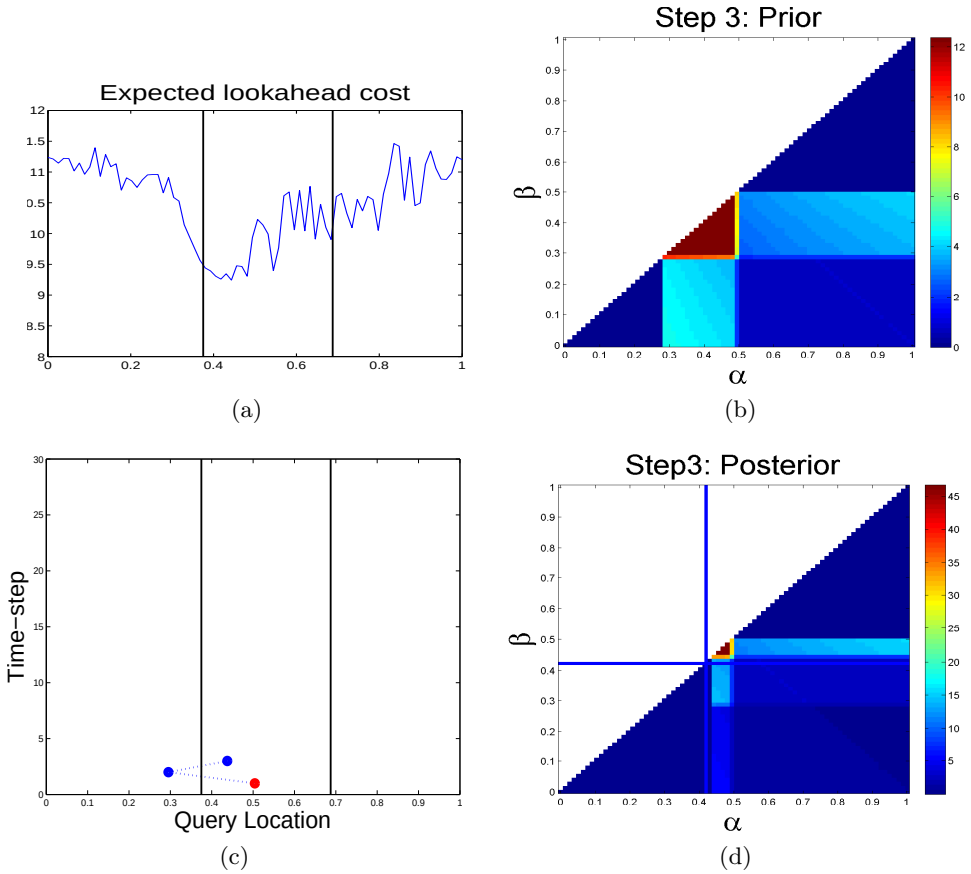


Figure 3.3.: Step 1: (b) In the first step the prior distribution $p(\alpha, \beta)$ is uniform. (a) the cost predicted at each query location. In this step the minimum cost is predicted at 0.5. (c) This plot shows the location of a query. A red dot indicates that the data was labelled as being in class 1. (d) The posterior distribution having observed $\{\mathbf{x}_1 = 0.5, \mathbf{y} = 1\}$. Notice $p(\beta > 0.5)$ has dropped to zero. The horizontal and vertical red line indicates the position of the class label query.



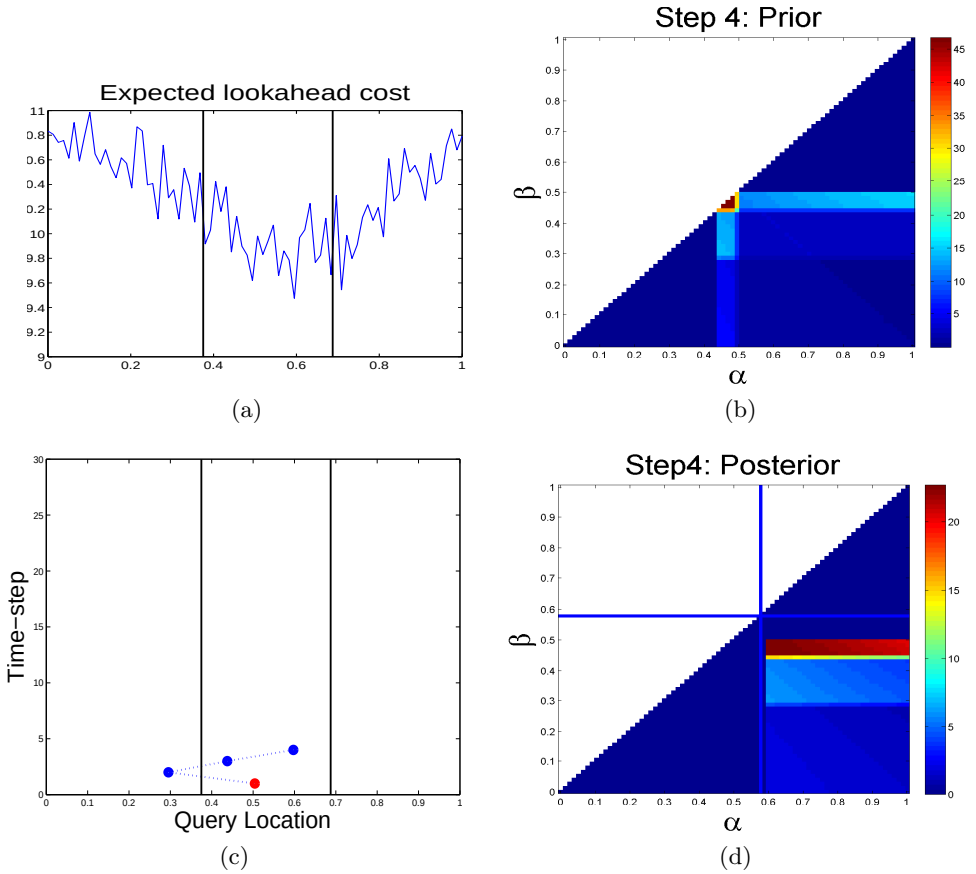
(a) (b) (c) (d)

Figure 3.4.: Step 2:(b) Posterior from previous step has become prior for this step. (a) Expected cost is minimum at 2.9 (c) This point is queried and found to be in class 0, blue. (d) The posterior distribution, $p(\alpha < 2.9)$ has dropped to zero.



(a) (b) (c) (d)

Figure 3.5.: Step 3: This step tests at 0.43 between previous queries. Most of the probability mass in the poserior density (d) is now in the small triangle where α, β are both in $[0.43, 0.5]$. The true values for α,β are 0.6875 and 0.375 respectively. The reason for this is the misleading class labels in the overlap region.



(a) (b) (c) (d)

Figure 3.6.: Step 4: This step eliminates the erronious triangular region in the posterior from step 3, by testing further across at $x = 0.6$.

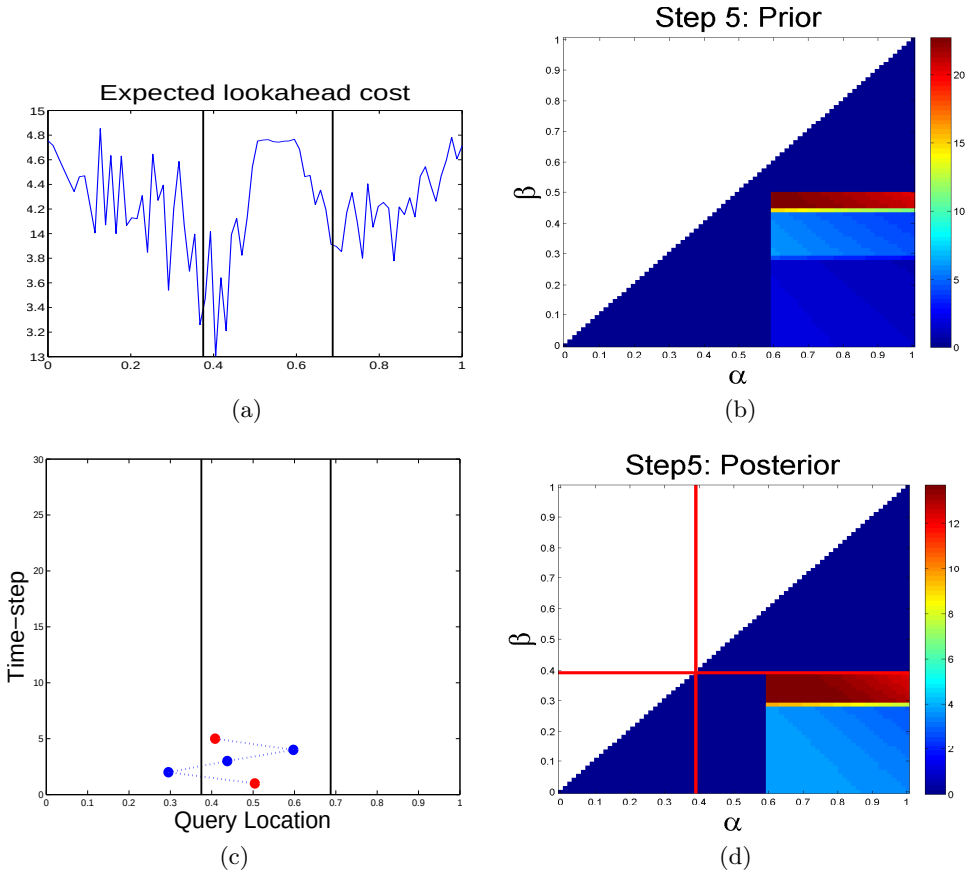


Figure 3.7.: Step 5: This the first step in which the expected cost shows noticable increase within some part of the overlap region. The observation in class 1 eliminates $\beta \geq 0.4$.

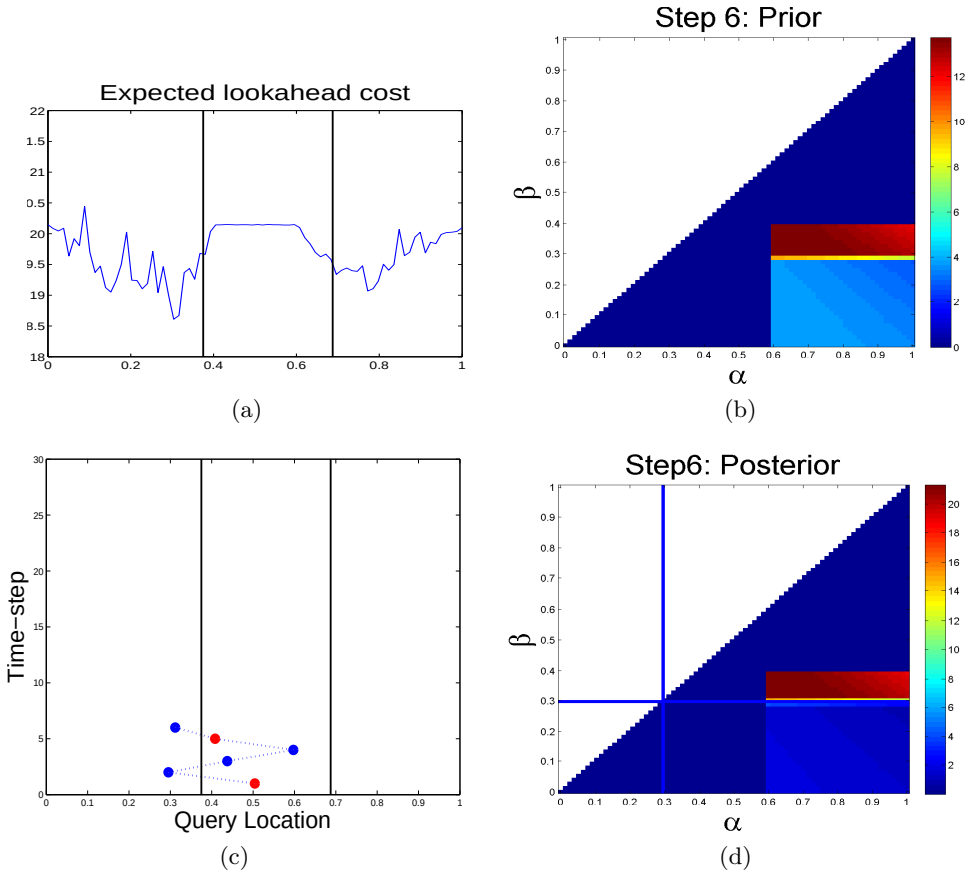


Figure 3.8.: Step 6: We can see from the expected lookahead cost that the algorithm is beining to predict the location of the overlap region.

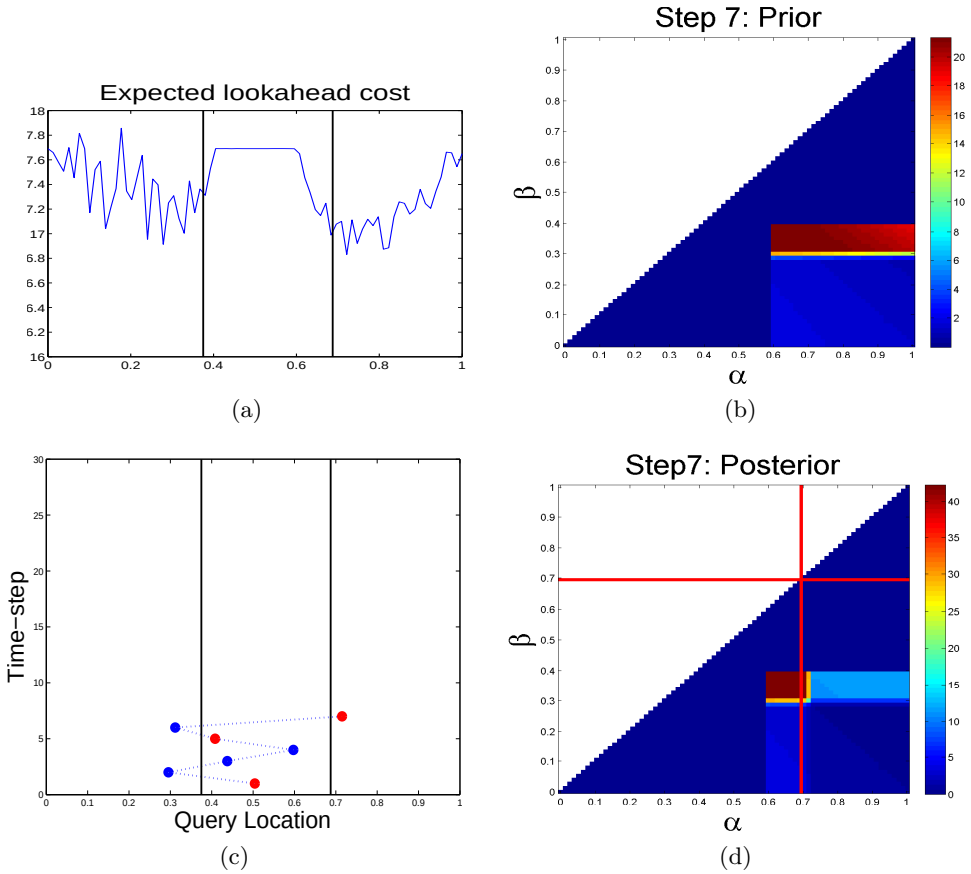


Figure 3.9.: Step 7: In this step the posterior is refined in the α direction, reducing the probability that $\alpha \geq 0.7$ in the posterior distribution.

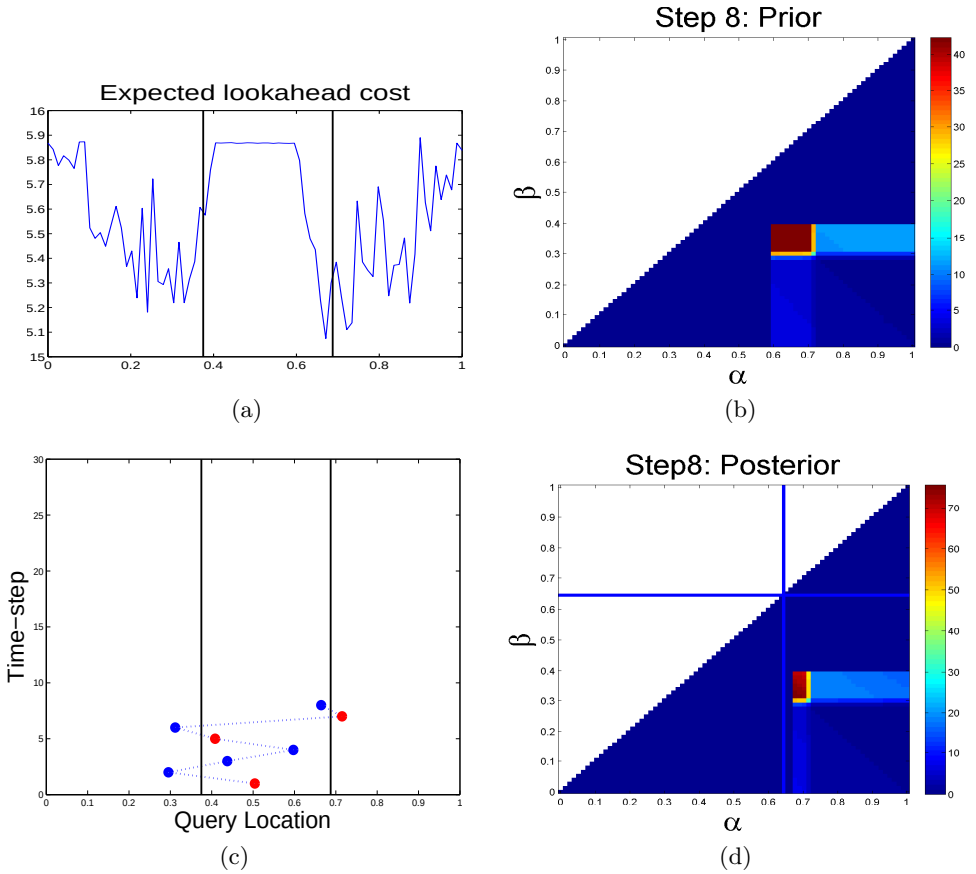
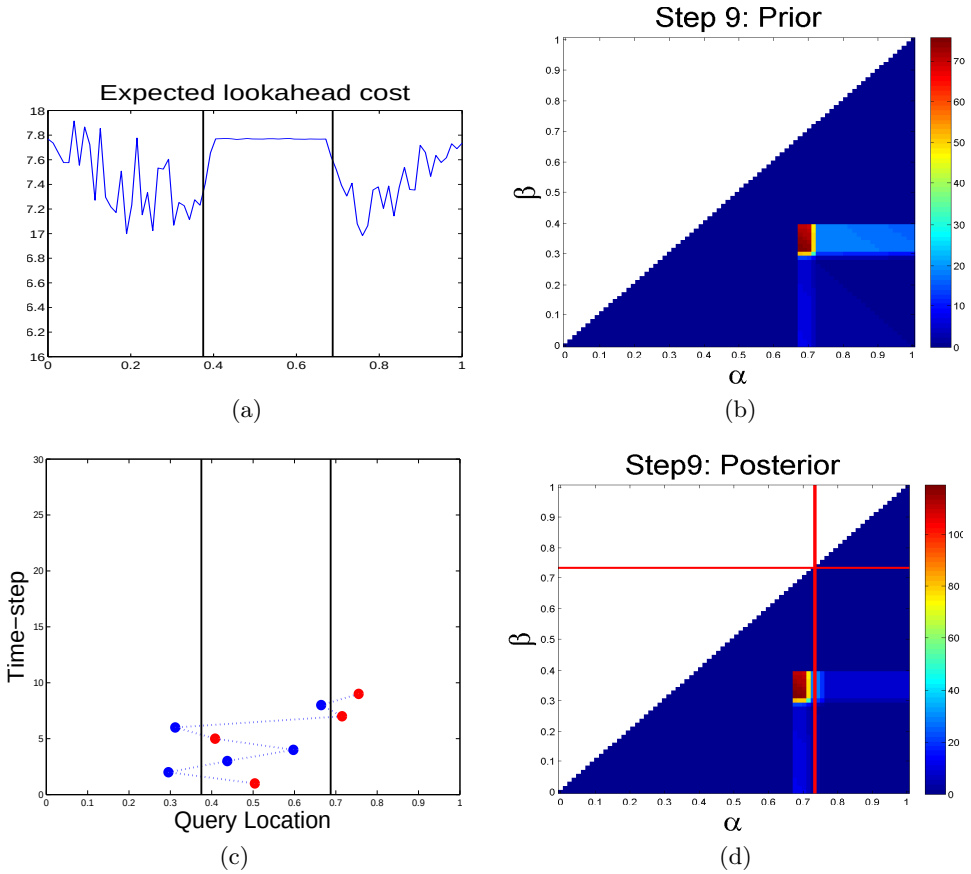
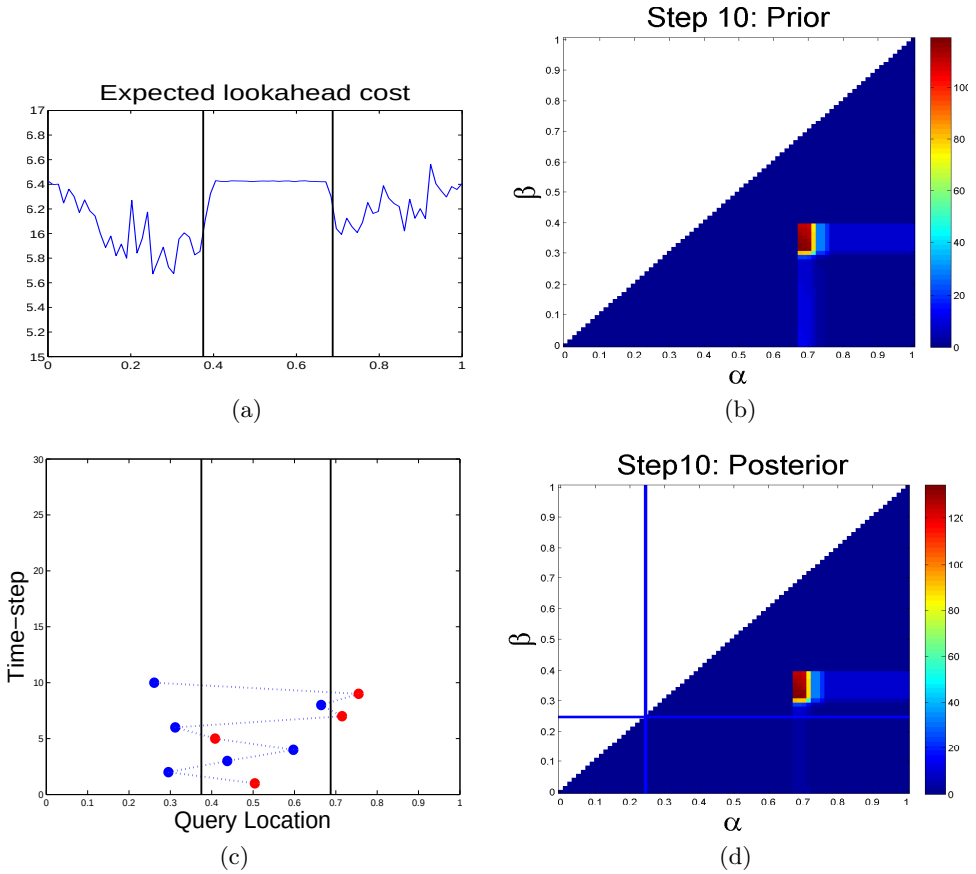


Figure 3.10.: Step 8: Following the query location from step 7 the algorithm continues to 'test' the right hand side of the overlap region.



(a) (b) (c) (d)

Figure 3.11.: Step 9: We can see from the expected lookahead cost and the posterior distribution that the algorithm is begining to predict the limits of the overlap region.



(a) (b) (c) (d)

Figure 3.12.: Step 10: After 9 observations main region of density is now near to the correct solution. We can see from the expected cost (a) that the model now has some idea of where the true overlap region is, and is predicting higher costs for queries in the overlap region.

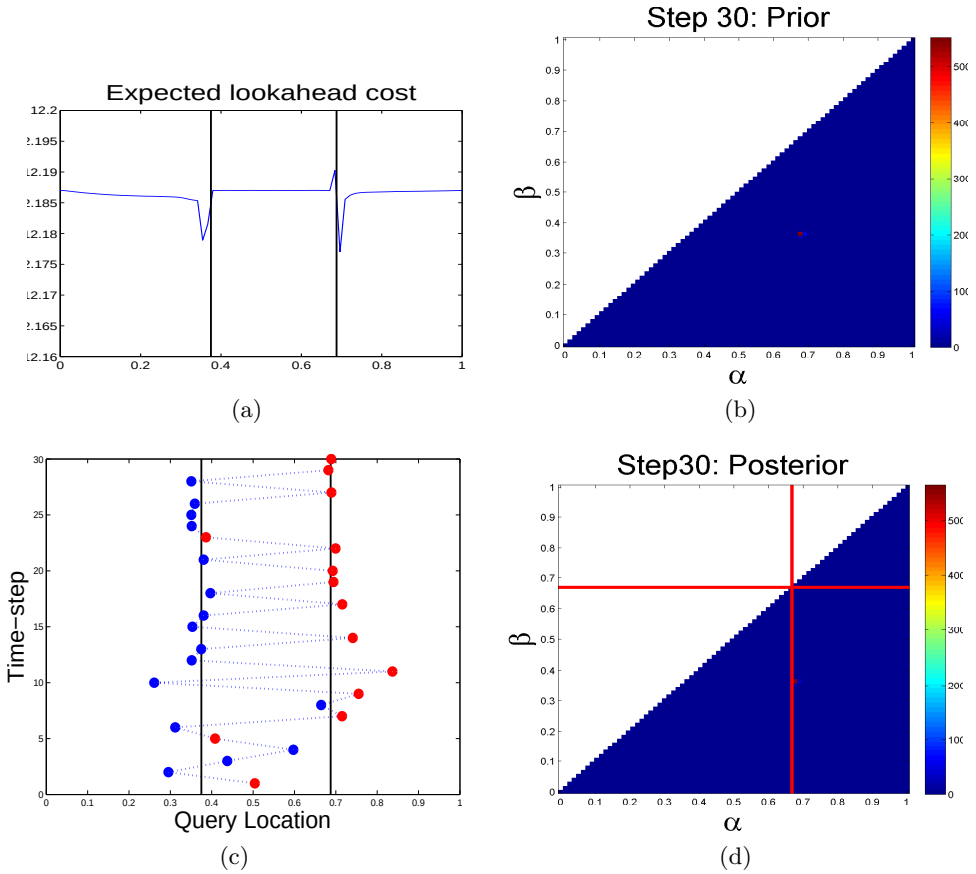
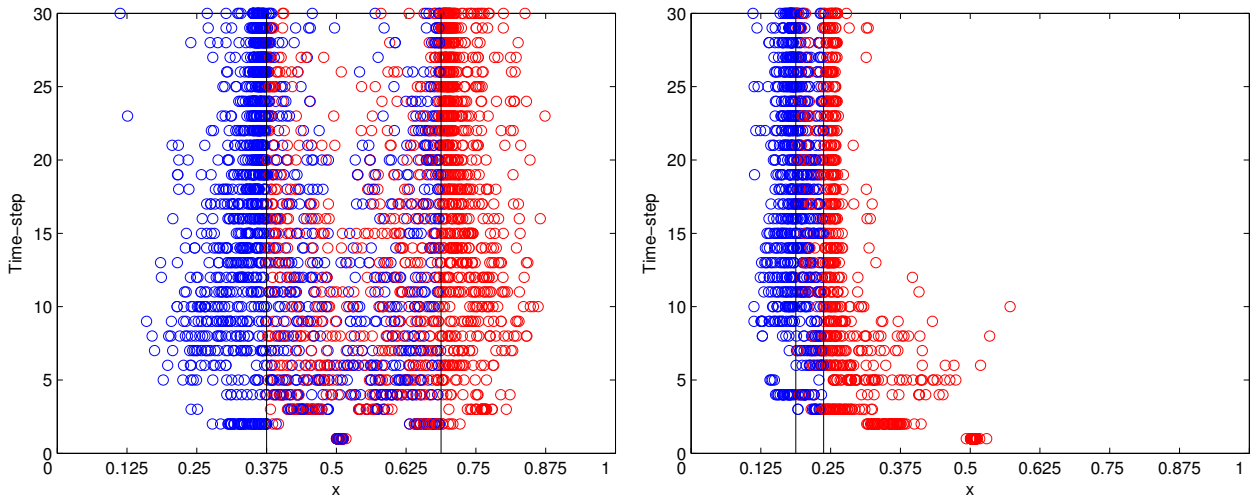


Figure 3.13.: Step 30: By step 30 the algorithm shows a clear preference for queries just outside the overlap region (a). The prior has narrowed arround the correct solution, and we can see from the query locations (c) that the algorithm has started to systematically test the class boundaries on either side.

Learning is initialised in step 1 with a uniform prior (panel (b)), and the minimum expected cost predicted by the lookahead is for a query of $x_1 = 0.5$ (panel (a)). This query location is then represented as a red dot, to show that its class label $y_1 = 1$ (panel (c)). Finally, we see that the posterior distribution (panel (d)) has dropped to zero for $\beta > 0.5$, as we would expect given the observation of an input in class 1 at 0.5. Here the red, horizontal and vertical lines represent where the query was made and that it was in class 1.

The posterior from step 1 becomes the prior in step 2 and the process is repeated (figure 3.4). This time the query is made around 0.29 and is in class 0. This causes the probability of $\alpha < 0.29$ to drop to zero. We can already see that after just 2 observations most of the probability mass is concentrated in the region where both $\alpha \in [0.29, 0.5]$ and $\beta \in [0.29, 0.5]$. Step 3 (figure 3.5) queries at about 0.43 and we can see from the posterior that the algorithm predicts that the most likely scenario is that both α and β are between 0.43 and 0.5 although there is posterior mass outside this region. Clearly this is not actually the case, as the true values are $\alpha = 0.6875$ and $\beta = 0.375$. Interestingly in step 4 (figure 3.6) the algorithm queries at 0.6 corresponding to the minimum predicted cost to test this hypothesis and observes an input in class 1. We can see from the posterior distribution that the small triangular region of prediction in step 3 has been ruled out entirely. So far the algorithm has ascertained that $\alpha > 0.59$ and $\beta < 0.5$.

In steps 5 to 9 (figures 3.7 to 3.11) the algorithm continues to test the boundary of the overlap region and by step 10 (figure 3.12) we can see that after just 9 previous observations the algorithm is already starting to predict α, β with reasonable accuracy. This is apparent in the expected lookahead cost plot (panel (a)), where we can see a marked increase in the expected cost for queries within the overlap region. This implies that queries within the overlap region are inherently less informative than queries on either side, where the class label returned will always be the same. We can see from the query locations (panel (c)) that queries 7, 8 and 9 test the upper bound for class 0 before moving across in query 10 to test the lower bound for class 1.



(a) (b)

Figure 3.14.: Query locations over multiple runs, for the wide overlap problem (a) and the narrow overlap (b). The class label returned by the oracle is represented by red circles for class 1 and blue circles for class 0. "Vertical" lines mark the true extent of the overlap regions.

By step 30 (figure 3.13) we can see the prior and posterior has become much more concentrated around the correct solution and the predicted cost figure (panel (a)) shows sharp spikes, which favour queries close to, but not inside, the overlap region. This may be because queries outside the overlap region which are close to the boundary are more informative than those further away while queries inside the overlap region give unreliable class labels. From the query location plot (panel (c)) we can clearly see how the algorithm has homed in on, and tested, the boundary on either side of the overlap region.

There is an analogy between version space and the parameter distributions. In the separable case where particular parameter combinations are either 'consistent' or not the posterior distribution for α, β would simply be uniform on all consistent solutions and zero elsewhere. In this case eliminating areas of the parameter distribution would be identical to reducing version space. Notice though that while the area of the posterior distribution which has non-zero probability may be reduced the volume of a probability density function remains constant. In the probabilistic setting a query may significantly reduce the likelihood for a particular region of parameter space without the probabilities dropping to zero. The aim here is to concentrate most of the probability mass into one small region. A good measure of the 'spread' of the distribution is the entropy. In section

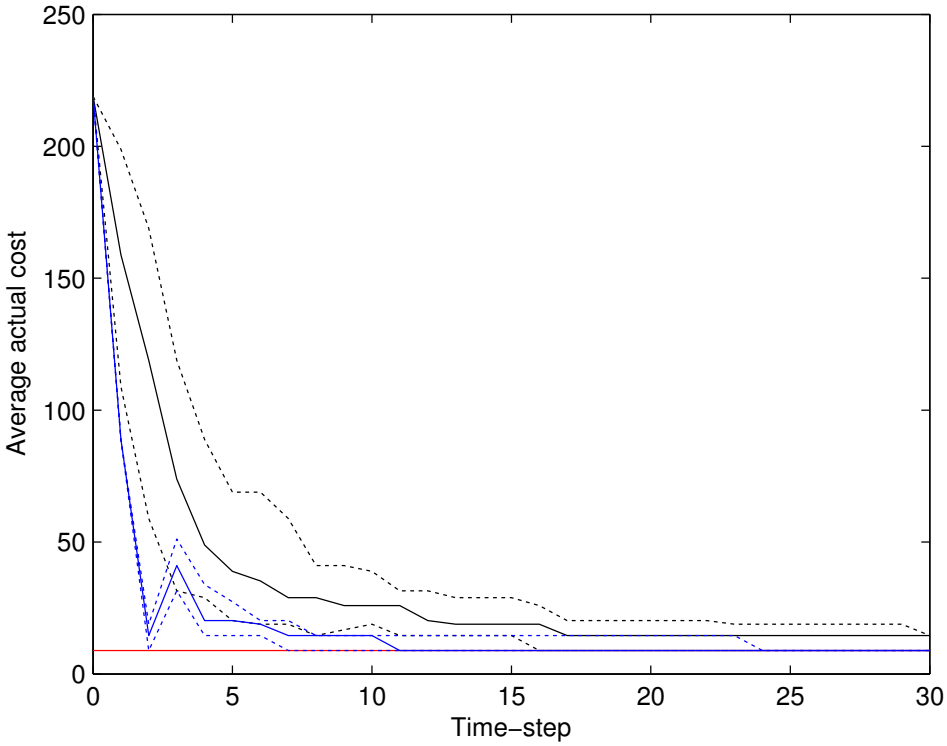


Figure 3.15.: This figure shows results for narrow overlap problem. Median cost after n observations, for random sampling (black), averaged over 200 runs and for active sampling (blue) averaged over 100 runs. The red line is the minimum cost for the problem. Dotted error lines show the 25th and 75th percentiles.

3.5.1 we introduce an example where entropy is used as an alternative to performing the full lookahead. The assumption being that where probability mass is well concentrated the model has a high degree of certainty and expected misclassification cost can be assumed to be low.

Figure 3.14 shows a scatter plot of the query locations in each time-step, over multiple runs, for the wide overlap (panel (a)) and the narrow overlap (panel (b)). We can see that in both cases our observation that the algorithm tends to favour inputs just outside the overlap region, and avoids queries near to the middle of the overlap region, is confirmed. This is particularly noticeable in the wide-overlap problem where queries become increasingly close to the class boundaries, with noticeably fewer queries within the overlap region as learning continues. Notice also that in the narrow-overlap problem, which is an easier problem, the algorithm makes most of its queries between $x = 0.1$ and $x = 0.3$. In other words it fairly rapidly homes in on the vicinity of the overlap region.

There is a stochastic element in how the expected lookahead cost is evaluated. The estimated cost for a given query ϕ is based on the posterior expected cost average over 1600 hypothetical samples from the model. In the first time-step where the prior distribution on $\{\alpha, \beta\}$ is uniform the optimum query is at 0.5. However, we can see from the query location of the first time-step, in both panels of figure 3.14 that over 100 runs there is some noise on the algorithm's estimate for the minimum expected cost. In the wide overlap case (panel (a)) the first query at 0.5 is within the overlap region. It therefore has an equal probability of being labelled in either class. We can observe that in this case queries in the second time-step centre around $\frac{1}{3}$ and $\frac{2}{3}$. Because the underlying process which we are approximating is deterministic, apart from the class labels assigned within the overlap region, it seems reasonable to conclude that if there were no noise on the model estimate then the second time-step queries would always land exactly at $\frac{1}{3}$ or $\frac{2}{3}$ depending on the class label returned by the label oracle for the $x = 0.5$ query. Similarly in the narrow case, where the class label assigned to the $x = 0.5$ query will always be in class 1 the second query would always be at $\frac{1}{3}$. This gives some intuition of how the model estimate compares to the 'true' optimal and it is interesting to note the contrast to the high-low game where class labels can be 'believed' with certainty and the optimal query in the second time-step would be at $\frac{1}{4}$ or $\frac{3}{4}$, halving the remaining region. For a better estimate more hypothetical samples would be required, however each hypothetical sample requires a full Bayesian lookahead, and evaluation.

To evaluate the performance of the algorithm we measure the 'actual cost' namely the average misclassification cost that would be incurred by a Bayesian classifier encountering examples from $p(x)$. This is:

$$R_{\text{actual}} = \int R(x \,|\, \mathcal{D}_n, \Phi_n) p(x)\, dx \tag{3.25}$$

where $R(x \,|\, \mathcal{D}_n, \Phi_n)$ is given by equation (3.7). Figure 3.15 and figure 3.16 (blue line) show the median cost versus the number of queries averaged over 100 runs, for the narrow and wide problems respectively. Also shown is the median cost for random selection of queries from $p(x)$ (black line). It is clear that the active learning algorithm learns faster than passive learning. This is particularly pronounced for the narrow run where the interquartile

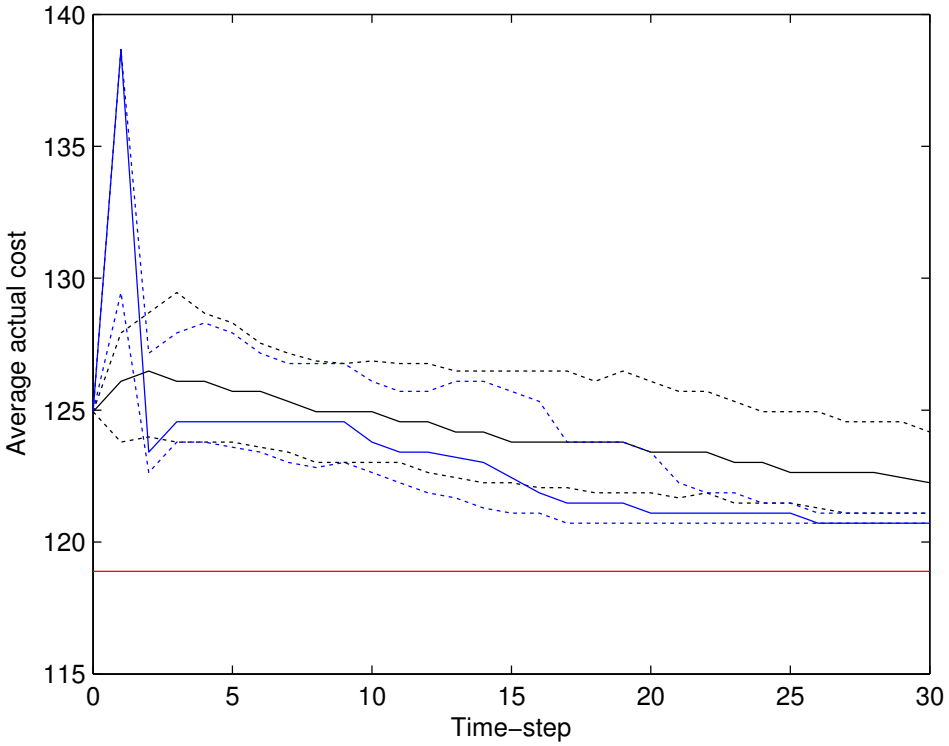


Figure 3.16.: This figure shows results for wide overlap problem. Median cost after n observations, for random sampling (black), averaged over 200 runs and for active sampling (blue) averaged over 100 runs. The red line is the minimum cost for the problem. Dotted error lines show the 25th and 75th percentiles.

range for active learning is narrow, indicating a consistently better performance than for passive learning. For the wide run, where the initial observation is always 'misleading' active learning does 'recover' by around 4 observations but the interquartile range remains relatively wide slowly narrowing with each time-step. By around 22 observations the interquartile range has narrowed considerably and performance is again consistently better than for passive learning.

Note that in both graphs there is an early spike in the actual cost for the active learner. It is noticeable that this occurs where the algorithm first queries inputs in the overlap region. In the wide overlap case, the actual cost after a single observation is greater than the cost using only the uninformative prior. This is because for the parameter values of the wide overlap problem an initial query at $x = 0.5$ is always within the overlap region where the label information is inherently misleading. However, once the learner has gained information to model where the overlap region lies this is more than compensated for.

In this section we have demonstrated the query density framework on the probabilistic high-low game. The results indicated that it performs well compared to passive learning even when the region of overlap between the classes is large, covering over 30% of input space. The model implemented had the capacity to apply uniform query densities of varying width, however in the case of the probabilistic high-low game it appears that narrow queries work best. This may be because more information is contained in the class label than in the location of the input sampled. This therefore raises the question "*are narrow queries always better than broad queries?*". In the next section we show that narrow queries are not always better. In particular we demonstrate that in some instances neither active learning, in the form of a delta function query, nor passive learning are optimal.

3.5. Examples where wide queries are better

Because in the probabilistic high-low game simulations, the minimum estimated cost always appeared at $q_- = q_+$ (see figure 3.2), which corresponds to selecting x at a point, the natural question arises: is sampling at a single point rather than from a broad query function always optimal? We now give two simple examples where a broad query outperforms a point query.

The first example, which we call the 'three Gaussian' problem is a simple example involving three one-dimensional Gaussian distributions which shows that a constrained query density for which the optimum is not necessarily infinitely narrow or infinitely wide but has some finite width.

3.5.1. Example 1: Three Gaussians

Here we present a simple example where, for a constrained query density, the optimum is not necessarily infinitely narrow or infinitely wide but has some finite width. The problem, which we call the 'three Gaussians' problem, is illustrated in Figure 3.17. This is a binary classification problem, where there are two classes, red for C_1 and green for C_0. We keep the model as simple as possible by assuming that the distribution of C_1 (red) has already been learned. Its distribution is $p(x \mid C_1) = (\mathcal{N}(x \mid 3, 1) + \mathcal{N}(x \mid 7, 1))/2$ as shown by the

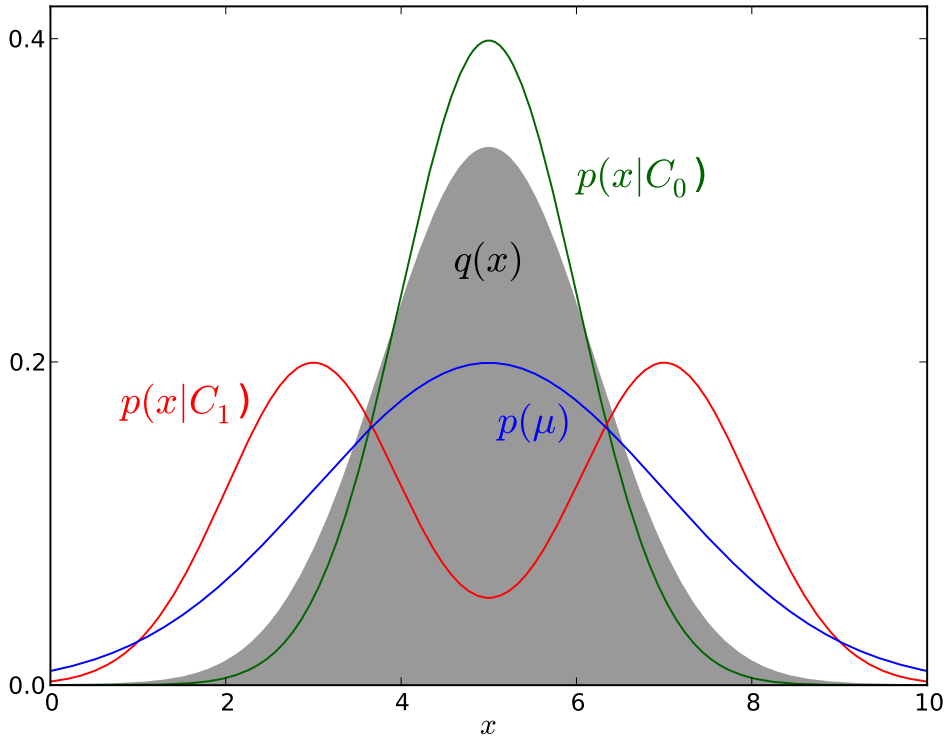


Figure 3.17.: Illustration of the three Gaussian problem. Class conditional densities $p(x \mid C_i)$ are shown in green and red. All the parameters of $p(x \mid C_1)$ are known, but the mean μ of C_0 class is to be learned; the prior $p(\mu)$ is shown in blue. Data is queried from a query density $q(x) = \mathcal{N}(x \mid 5, \sigma_q)$ whose width σ_q may be varied; the shaded distribution shows a query density of approximately the optimal width.

red line in Figure 3.17. We also assume that $p(C_0) = p(C_1) = 1/2$.

To even further simplify the model we assume that it is known that class 1 is Gaussian distributed with unit variance, $p(x \mid C_0) = \mathcal{N}(x \mid \mu, 1)$ as shown by the green line in figure 3.17. Its mean μ then is the only unknown parameter to be learnt in the model. To make the model Bayesian we also assume the prior $p(\mu) = \mathcal{N}(\mu \mid 5, 2)$ as indicated by the blue curve.

We now assume that the set of possible query densities is constrained to be Gaussian of the form, $q(x) = \mathcal{N}(x \mid 5, \sigma_q^2)$, where the mean is centred on $x = 5$ (corresponding to the centre of symmetry) and the standard deviation σ_q is the only adjustable parameter.

In this example we estimate the full lookahead cost by the entropy of the expected posterior density. High entropy corresponds to a posterior density $p(\mu \mid x_1)$ which is 'spread

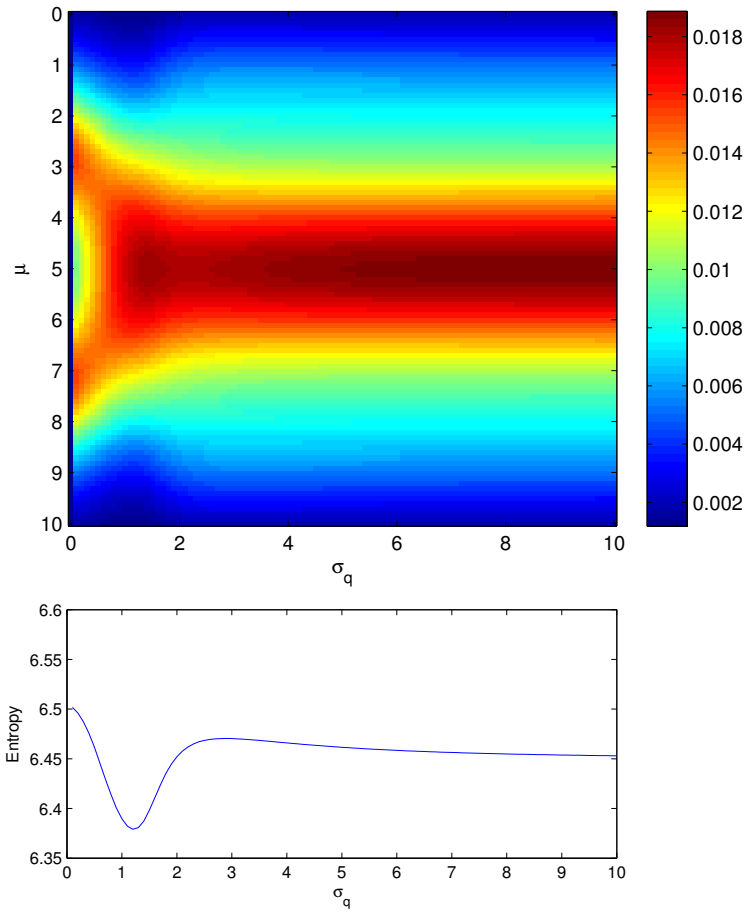


Figure 3.18.: Top: expected posterior density $p(\mu \,|\, x, y, \Phi = \sigma_q)$ in the next time-step as a function of σ_q. This is found by averaging across all hypothetical observations (x', y') with x' drawn from $Q(x \,|\, \mu, \Phi)$. Bottom: entropy of the expected posterior density for each σ_q. This has a minimum when $\sigma_q = 1.2$, which suggests that in the three Gaussian problem $q(x) = \mathcal{N}(x \,|\, 5, 1.2)$ is the optimum query density for the first observation.

out' in parameter space and implies a low level of certainty by the model about μ. On the other hand lower entropy corresponds to a more compact distribution in parameter space where the probability mass of the posterior distribution is concentrated at one location in parameter space. In this case the model shows a high level of certainty about the parameters. This is analogous to reducing version space when the model is separable and it seems reasonable that when there is higher certainty about the posterior distribution of μ the expected misclassification cost will also be less. Figure 3.18(a) shows the expected posterior density of μ for values of $\sigma_q \in [0, 10]$. These are found using:

$$p(\mu \,|\, \phi = \sigma_q) = \int p(\mu \,|\, x', y') p(y' \,|\, x', \mu) Q(x' \,|\, \sigma_q) \, dx' dy', \tag{3.26}$$

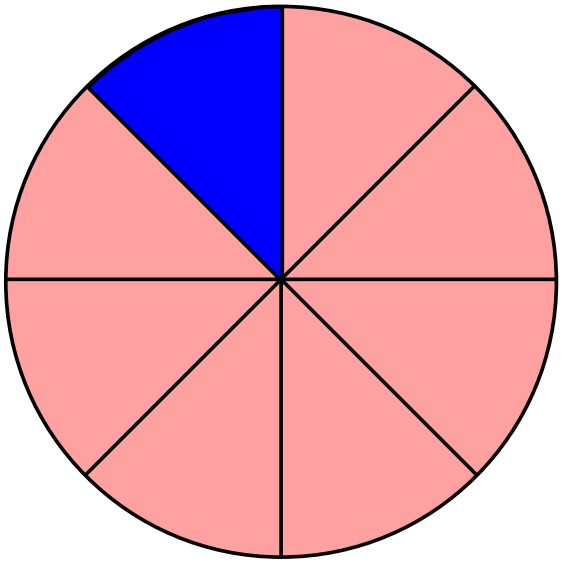


Figure 3.19.: Pie example: In this problem there are N fixed regions (or slices). It is known to the learner that there is only one slice in class 1 (blue) and all other slices are in class 0 (pink). It is also known that $p(x \mid C_1) = M \cdot p(x \mid C_0)$ where the ratio of density in class 1 is M times higher than in class 0. Before any observations have been made each slice has an equal probability of being in class 1. The question here is, how best to sample an unlabelled input in order to minimise the posterior expected cost, where the same cost function is used as in section 3.3.

where $p(\mu \mid x', y')$ is the posterior density of μ having observed a single hypothetical pair (x', y'). Figure 3.18(b) shows the entropy of these posterior distributions as a function of σ_q. As can be seen in this case the minimum cost is given by sampling with $\sigma_q \approx 1.2$. Therefore, the constrained Gaussian query density which minimizes the uncertainty about μ in the next time-step is $q(\mathbf{x}) = \mathcal{N}(x \mid 5, (1.2)^2)$.

A broad sample from $p(x \mid C_0)$ gives more information than a narrow query. To see why this is probably the case consider the extreme case where $p(x \mid C_0)$ is very narrow (i.e. has very low variance). In this case a single broad sample would be very likely to return an x near to the true mean and therefore provide a good estimate of the mean. However in the three Gaussian problem, queries which are too wide have a high probability of drawing observations in class 1 which is uninformative to the model. A query density with an intermediate standard deviation 1.2 therefore represents an intuitive compromise between these two factors.

3.5.2. Example 2: Pie example

In the previous example we were able to show that the query density which minimises the posterior entropy of the parameters is neither a point query nor a passive sample from the underlying distribution. However, this example is somewhat artificial because the query is always centred on the point of symmetry in the problem. We now give an example where the expected misclassification cost for any point query is higher than for a broad query. This example is contrived, but demonstrates a counterexample to the possibility that 'single point queries are always best'. It is also simple enough to shed some intuition on why a broad query is better in this case.

We call this the pie example because the problem is defined on a circular region in $\mathbb{R}^2$, divided into N fixed regions (or slices) as illustrated in figure 3.19, for $N = 8$. It is known to the learner that each of these slices is either a blue slice (class 1) or a pink slice (class 0), but only one slice is blue (in class 1). The prior probability known to the learner is that each segment has a $\frac{1}{N}$ chance of being in class 1. It is also known to the learner that $p(x \mid C_1) = M \cdot p(x \mid C_0)$ where the ratio of density in class 1 is M times higher than in class 0. Finally, M and N are also known to the learner and $M > 1$.

We can now see that when using the same cost function as in section 3.3, regardless of the misclassification costs λ_{01} and λ_{10}, a query at a single point leads to a higher predicted misclassification cost than a broad query across the whole circular region. This is because, a point query by definition is confined to a single segment, and therefore has a $\frac{1}{N}$ chance of being labelled as being in class 1. However, a broad query which is uniform over all slices has a $\frac{M}{M+N-1}$ chance of being in class 1, which, when $M > 1$, is more than $\frac{1}{N}$.

Notice that in either case (point query or broad query) the misclassification cost associated with the target label is the same. When $y_q = 1$ the learner has complete information about the model and subsequently perfect classifications will be made. When $y_q = 0$, whether or not this was the result of a point query or the broad query, no new information has been learnt about the remaining $N - 1$ slices. The subsequent misclassification cost will therefore be the same for either point query or broad query.

The only difference between a broad query and a point query is that it has a higher chance of returning a class label $y_q = 1$, where the misclassification error drops to zero. It is therefore apparent, without evaluating the exact misclassification costs or even needing to know the classification threshold λ, that a random draw from the underlying distribution results in a lower misclassification cost than a single point query (this argument is confirmed in appendix A where the actual misclassification costs for the pie example are evaluated).

We can also see that it is not always the case that the best broad query is the query made across the whole of underlying distribution by observing that when the class label 0 is returned in the first time-step, a query in the second time-step which samples uniformly across the $N - 1$ unlabelled slices is preferable.

This observation demonstrates that there are indeed examples where the optimal query is neither to sample uniformly across the whole underlying distribution nor to sample at a point. Therefore in principle arbitrary query densities of the form a query density present a theoretical advantage over both single point active queries and passive learning.

3.6. Discussion

We have presented a framework for probabilistic active learning of non-separable data which is based upon the principle of reducing the estimated misclassification cost of each datum queried. The Bayesian formulation permits averaging over model parameters to reduce parameter uncertainty in predictions. This approach specifically models how inputs are sampled from the underlying distribution and avoids making any IID assumption about the inputs. To our knowledge this is the first time active selection from a sampling distribution, rather than specifically at a point has been considered.

This algorithm was illustrated on the non-separable problem of the probabilistic high-low game, which is a generalisation of the high-low game presented by Seung et al. [1992] and demonstrates that the active learner is able to compensate for considerable overlap between the class distributions. However single point queries proved to be favourable for the probabilistic high-low game.

The three Gaussians problem and the pie example described in sections 3.5.1 and 3.5.2 show that there exist situations in which the optimal way of selecting is not necessarily to select at a single point or randomly across the whole region, but to select from a localised distribution. We note that the cost benefit of selecting from an interval is not very large in the three Gaussians problem, which was constructed to test this, and in the probabilistic high-low game it appears that selecting at a point is optimal anyway. However, the fact this may not always be the case is pointed to by the pie example in which the expected misclassification cost is heavily weighted in favour of a broad query across unlabelled slices when the ratio of class densities is large.

An important question which informs the debate of when active learning should be preferred to passive learning is: *how to characterise those problems in which sampling broadly is better than sampling from a point?*

The principle which seems to apply here is that sampling broadly can in some situations increase the probability of drawing from one class or another. This is beneficial if for example an input label in one class reduces the expected misclassification cost more than an input label in the other class. In the pie example this is taken to an extreme where a class label 1 fully informs the model while a class label 0 only eliminates one slice from the model.

The integrations required to implement this framework do however come with considerable computational costs and are not yet feasible in real applications. This process is however easily 'parallelised' and different machines could draw hypothetical samples before combining the average cost at the end. In the next section we examine what happens to the Bayesian lookahead algorithm when the query density collapses to a single delta function and the algorithm draws a single point to be labelled by the label oracle, without reference to the query oracle. We then use Gaussian processes to implement a specific version of the algorithm on real data in the pool based setting.

4. Pool-based Active Learning

4.1. Introduction

In this chapter we make some simplifications to the framework presented in chapter 3 in order to implement active learning in the pool-based setting. In section 4.2 we describe what happens to the model when the query density collapses to a point query. In this case the active learner simply selects an input to be given to the label oracle, without drawing from the sample oracle. Another way of seeing this is as a query density which has collapsed to have probability 1 on a single input. In section 4.3 we examine how the model is simplified for the pool-based setting, where information about the underlying distribution $p(\mathbf{x})$ is restricted to a finite pool of unlabelled inputs. Section 4.4 brings these together to present a framework for the expected cost reduction approach to active learning in the pool-based setting.

In order for the model to estimate the expected misclassification cost for a given query, the model needs to be able to predict the probability of class membership given the data. Currently, Gaussian processes represent one of the most straightforward and tractable ways of performing Bayesian inference. Therefore, in this chapter we use Gaussian processes (GPs) to make the necessary predictions, though in theory any other method of making Bayesian inference could be used. We use the Rasmussen and Nickisch [2006] software to perform binary classification using the expectation propagation (EP) algorithm. For classification with GPs some approximation or Monte Carlo approach needs to be used to estimate the full posterior distribution. We choose the expectation propagation (EP) algorithm because of the balance between computational feasibility and how well it approximates the true posterior.

Section 4.5 starts by briefly discussing how GPs have been used in active learning, before describing GPs and the EP algorithm as it is implemented by the Rasmussen and Nickisch software. This is described in [Rasmussen and Williams, 2006, ch3,ch5] but for completeness we give a brief overview in sections 4.5.1 and 4.5.2. As a preliminary to discussing classification, section 4.5.1 gives a brief overview of regression with GPs, after which classification with the EP algorithm is described in section 4.5.2.

In section 4.6 we describe how the algorithm presented in section 4.4 is implemented with GPs, a single run through of the algorithm on a simple data-set is then described, before examining the results of implementing the algorithm on six benchmark data sets (three synthetic and three non-synthetic). Section 4.7 summarises overall performance measures on the benchmark sets. Section 4.8 presents a comparisons of the GP active Bayes, lookahead algorithm with 3 other active learning algorithms. Final, section 4.9 presents a brief summary of the chapter.

4.2. Point queries

In this section we look at what happens to the model from chapter 3 when the query density collapses to a Dirac delta function. This scenario corresponds to full 'active' learning where the algorithm selects specifically which input to be labelled without drawing from the sample oracle. Here, the query density has non-zero probability only at the specific $\mathbf{x}_q$ selected by the algorithm (where q stands for 'query') and rather than defining a whole distribution, the query density parameters define simply which $\mathbf{x}$ is to be queried, $\boldsymbol{\phi}_n = \mathbf{x}_q$. We will sometimes omit $\boldsymbol{\phi}_n$ and just write $\mathbf{x}_q$. Looking at equation (3.1) for the sample oracle:

$$Q(\mathbf{x};\boldsymbol{\phi}) = \frac{q(\mathbf{x};\boldsymbol{\phi})p(\mathbf{x})}{\int q(\mathbf{x};\boldsymbol{\phi})p(\mathbf{x})\,d\mathbf{x}} \tag{4.1}$$

we observe that the sample oracle also collapses to a Dirac delta function, so long as $p(\mathbf{x}_q)$ is non-zero. In this case the parameter update equation (3.5) simply reduces to:

$$p(\boldsymbol{\theta}\,|\,\mathcal{D}_n,\Phi_n) \propto p(y_n\,|\,\mathbf{x}_n,\boldsymbol{\theta})p(\boldsymbol{\theta}\,|\,\mathcal{D}_{n-1},\Phi_{n-1}) \tag{4.2}$$

This is the standard Bayesian update for the observation of an input-label y_n and does not depend on $Q(\mathbf{x}_n; \boldsymbol{\phi})$, the way in which $\mathbf{x}_n$ was sampled. Therefore, the query oracle no longer needs to be modelled and the update in equation 4.2 can be straightforwardly re-written as:

$$p(\boldsymbol{\theta} \,|\, \mathcal{D}_n, \Phi_n) \propto \prod_{i=1}^{n} p(y_i \,|\, \mathbf{x}_i, \boldsymbol{\theta}) p(\boldsymbol{\theta}) \tag{4.3}$$

where the standard independence assumption holds and $p(\boldsymbol{\theta} \,|\, \mathcal{D}_n, \Phi_n)$ can be evaluated in one step from the original prior $p(\boldsymbol{\theta})$. Notice that $p(\boldsymbol{\theta} \,|\, \mathcal{D}_n, \Phi_n)$ is no longer dependent on Φ_n because $\boldsymbol{\phi}_n$ and $\mathbf{x}_n$ are equivalent. We therefore drop the dependency on Φ_n in the remainder of this chapter. Because the dependency on *how* the inputs are sampled has been removed, any classifier capable of producing a probability of class membership for unlabelled inputs in the test set given a labelled set of training examples may be used to evaluate the class conditional probabilities.

When the model is parametrised by $\boldsymbol{\theta}$ the class conditional probabilities are evaluated as in equation 3.6 by:

$$p(y = 1 \,|\, \mathbf{x}, \mathcal{D}_n) = \int p(y = 1 \,|\, \mathbf{x}, \boldsymbol{\theta})\, p(\boldsymbol{\theta} \,|\, \mathcal{D}_n)\, d\boldsymbol{\theta}. \tag{4.4}$$

For a non-parametric Bayesian classifier, such as a Gaussian process, there are no parameters $\boldsymbol{\theta}$, rather the prior is defined over the space of possible target distributions. In this section and the next two sections we keep the formulation general by referring only to the class conditional probabilities, for example $p(y = 1 \,|\, \mathbf{x}, \mathcal{D}_n)$. It is assumed that the class conditional probabilities are inferred by an arbitrary probabilistic classifier.

To evaluate the estimated risk $R(\mathbf{x}_q)$ for a specific query $\mathbf{x}_q$ the same process is followed as in the chapter 3. For both possible query-labels $y_q \in C_i$ we start by evaluating the posterior class conditional probabilities $p(y = 1 \,|\, \mathbf{x}, \mathbf{x}_q, y_q, \mathcal{D}_n)$. The decision function then assigns 'optimal' class labels for the given cost function. Here, the same cost function is

assumed as in chapter 3 and the decision function is given by:

$$\mathcal{A}(\mathbf{x} \,|\, \mathbf{x}_q, y_q, \mathcal{D}_n) = \begin{cases} 1 & \text{if } p(y = 1 \,|\, \mathbf{x}, \mathbf{x}_q, y_q, \mathcal{D}_n) \geq \lambda \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

where the threshold $\lambda = \lambda_{01}/(\lambda_{01} + \lambda_{10})$. The expected risk associated with an arbitrary $\mathbf{x}$, when $y_q \in C_i$, is then evaluated:

$$R(\mathbf{x} \,|\, \mathbf{x}_q, y_q) = \begin{cases} p(y = 1 \,|\, \mathbf{x}, \mathbf{x}_q, y_q, \mathcal{D}_n)\lambda_{01} & \text{if } \mathcal{A}(\mathbf{x} \,|\, \mathbf{x}_q, y_q, \mathcal{D}_n) = 1 \\ p(y = 0 \,|\, \mathbf{x}, \mathbf{x}_q, y_q, \mathcal{D}_n)\lambda_{10} & \text{otherwise.} \end{cases} \tag{4.6}$$

Next, this is integrated over $\mathbf{x}$ to give:

$$R(\mathbf{x}_q, y_q) = \int R(\mathbf{x} \,|\, \mathbf{x}_q, y_q) p(\mathbf{x}) d\mathbf{x} \tag{4.7}$$

the risk associated with $\mathbf{x}_q$ being in C_i. The expected risks associated with $y_q \in C_0$ and $y_q \in C_1$ are then combined to find the overall expected risk associated with a particular choice of $\mathbf{x}_q$:

$$R(\mathbf{x}_q) = \sum_{y_q \in C_0, C_1} R(\mathbf{x}_q, y_q)\, p(y_q \,|\, \mathbf{x}_q, \mathcal{D}_n) \tag{4.8}$$

In order to choose the next query point the $\boldsymbol{\phi}_n = \mathbf{x}_q$ is selected which minimizes the expected risk in the next time-step.

$$\boldsymbol{\phi}_n = \operatorname*{argmin}_{\mathbf{x}_q} R(\mathbf{x}_q) \tag{4.9}$$

The main difference between the point query and the general query density is that all information regarding previous observations is retained in $\mathcal{D}_n$. The dependence on Φ_n can therefore be dropped and inference of class conditional probabilities follows the 'standard' format.

4.3. Pool-based active learning

In the previous sections we have assumed that the learner is supplied with a set of labelled data $\mathcal{D}$ comprising N inputs $\mathbf{x}_n \in \mathcal{X}$, $n = 1, \ldots, N$ and corresponding labels $y_n \in \{0, 1\}$. We also assume that any new $\mathbf{x}$ to be queried can be drawn from the underlying distribution $p(\mathbf{x})$. In many practical problems this is not the case, and the data available consists of some labelled data $\mathcal{D}$ and a pool of unlabelled inputs $\mathcal{U}$. In this case the possible values of $\mathbf{x}$ which may be queried by the learner are constrained to those in the set $\mathcal{U}$. In this setting our knowledge of the underlying distribution $p(\mathbf{x})$ is confined *a priori* to the inputs $\mathbf{x} \in \mathcal{U}$ and $\mathbf{x} \in \mathcal{D}$. Because knowledge of $p(\mathbf{x})$ is not being actively learned in this setting, there is no advantage in considering a broad query density. The simplifications laid out in section 4.2 can therefore be followed, with the addition that the set of possible values for which $\mathbf{x}$ can be observed is constrained to the finite set $\mathcal{P} = \mathcal{D} \cup \mathcal{U}$ and only specific queries $\mathbf{x}_q \in \mathcal{U}$ are considered.

In the case of pool-based learning the integrals in equations (4.5) to (4.9) are replaced by summations evaluated on the finite set of inputs in the pool, $\mathbf{x} \in \mathcal{P}$. The expected risk evaluated in equation (4.7) is simply averaged over the remaining unlabelled data:

$$\tilde{R}(\mathbf{x}_q, y_q) \approx \frac{1}{|\mathcal{U}|} \sum_{\mathbf{x} \in \mathcal{U}} R(\mathbf{x} \,|\, \mathbf{x}_q, y_q) \tag{4.10}$$

where $|\mathcal{U}|$ is the number of unlabelled inputs and $\tilde{R}$ represents an *approximate* risk. Notice that the estimated risks are now dependent on the finite pool of data $\mathcal{U}$. The quality of the risk estimates and the generalization of the classification are therefore contingent on $\mathcal{U}$ being representative of the true underlying distribution $p(\mathbf{x})$.

In terms of practical implementation the key advantage is that there is only a finite set of inputs $\mathbf{x}_q$ which could be queried, and because all $\mathbf{x}$ are drawn from the finite set $\mathcal{U}$, we only need evaluate the decision function, and the probabilities of class membership on this set.

Algorithm 5 Pool-based active learning

Require: Start with one labelled point from each class $\mathcal{D}_2 = \{\mathbf{x}_1, y_1 \in C_0, \mathbf{x}_2, y_2 \in C_1\}$

```
repeat: Until stopping criterion.
    Find: p(y | x, D_n)                          Using the Bayesian classifier
    Find: x_q := SELECT-QUERY(D_n, U_n)          See Select Query algorithm
    y_q ~ LabelOracle(x_q)                       Give x_q to label oracle which returns y_q
    D_{n+1} := D_n ∪ {x_q, y_q}                  Add {x_q, y_q} to the labelled data
    U_{n+1} := U_n \ {x_q, y_q}                  Remove {x_q, y_q} from unlabelled data
    n := n + 1
end
```

Algorithm 6 SELECT-QUERY($\mathcal{D}_n, \mathcal{U}_n$)

Require: $\mathcal{D}_n$, $\mathcal{U}_n$

```
for each: x_q ∈ U_n
    for i = 0, 1
        for each: x ∈ U_n \ {x_q}
            Find: p(y = 1 | x, D_n, x_q, y_q = C_i)      Using the Bayesian classifier
            Evaluate: A(x | x_q, y_q = C_i)             From equation (4.5)
            Find: R̃(x | x_q, y_q = C_i)                 From equation (4.6)
        end
        Find: p(y_q = C_i | x_q, D_n)                   Using the Bayesian classifier
        Find: R̃(x_q, y_q = C_i)                         From equation (4.7)
    end
    Find: R̃(x_q)                                       From equation (4.8)
end
return: x_q^* := argmin_{x_q} R̃(x_q)
```

4.4. Bayesian cost reduction approach to pool-based active learning

Putting together the simplifications to the query density framework of chapter 3, presented in section 4.2 and section 4.3, algorithm 5 and algorithm 6 show the implementation of the query density framework in the pool-based setting. This is presented for any Bayesian classifier capable of making probabilistic inference, though for our implementation, in section 4.6, this will be a Gaussian process classifier using the expectation propagation approximation.

Algorithm 5 presents an overview of the active learning algorithm, where in each step the algorithm selects the query $\mathbf{x}_q$ from the unlabelled inputs which has the lowest predicted misclassification cost (shown in algorithm 6). The query is given to the label oracle which

returns a class label y_q. These are then added to the set of labelled data, and removed from the set of unlabelled data. At each step the set of labelled data $\mathcal{D}_n$ grows by a single observation. The model makes predictions on the basis of its current labelled set $\mathcal{D}_n$ and at any stage the current decision boundary and class predictions are given by $p(y \mid \mathbf{x}, \mathcal{D}_n)$, which is an output of the Bayesian classifier.

Algorithm 6 describes the process by which the expected misclassification cost $\tilde{R}(\mathbf{x}_q)$ is estimated and an input query $\mathbf{x}_q$ is selected. The algorithm follows the same procedure described in section 4.2 except for line 9 where the integral in equation (4.7) is replaced by a summation as in equation (4.10). For both possible query-labels $y_q \in C_i$ we start by evaluating the posterior class conditional probabilities $p(y = 1 \mid \mathbf{x}, \mathbf{x}_q, y_q, \mathcal{D}_n)$ (line 4). In line 5 the decision function then assigns the 'optimal' class labels according to the decision function, as in equation (4.5). The predicted class labels are then used to evaluate the expected risk for each $\mathbf{x}$ within the pool of unlabelled observations (line 6) as in equation (4.6). In line 8 the class conditional probability for the input label $p(y_q \mid \mathbf{x}_q, \mathcal{D}_n)$ is evaluated by the classifier. This is required to combine the expected costs $\tilde{R}(\mathbf{x}_q, y_q = C_0)$ and $\tilde{R}(\mathbf{x}_q, y_q = C_1)$ to find the overall expected cost $\tilde{R}(\mathbf{x}_q)$ associated with a query $\mathbf{x}_q$ in line 11. To evaluate $\tilde{R}(\mathbf{x}_q, y_q)$ in line 9 the integral in equation (4.7) is replaced by a summation as in equation (4.10). In line 11 the overall expected risk is evaluated (equation 4.8) which combines the expected risks associated with $y_q \in C_0$ and $y_q \in C_1$ for a particular choice of $\mathbf{x}_q$. Finally, to choose the next query point the $\boldsymbol{\phi}_n = \mathbf{x}_q$ is selected which minimizes the expected risk in the next time-step.

In order to perform this lookahead the model needs two things. Firstly, to evaluate the decision function in equation (4.5) the algorithm needs to evaluate the posterior class predictions $p(y \mid \mathbf{x}, \mathbf{x}_q, y_q = C_1, \mathcal{D}_n)$ and $p(y \mid \mathbf{x}, \mathbf{x}_q, y_q = C_0, \mathcal{D}_n)$. In line 6 the class predictions are also used to evaluate the expected risk associated with $\mathbf{x}_q$ and y_q, in equation (**??**). Secondly, to combine the expected risks from both cases in equation (4.8) an estimate of the probability of class membership of $p(y_q \mid \mathbf{x}_q, \mathcal{D}_n)$ is required.

Algorithm 5 and algorithm 6 present the expected cost reduction approach to active learning in the pool-based setting. We implement this algorithm with Gaussian processes

using the EP algorithm for classification. In the next section we introduce GPs before demonstrating the algorithm on six benchmark data-sets.

4.5. Gaussian process active learning

Gaussian processes (GPs) present a non-parametric approach to Bayesian learning which is tractable for moderately-sized data-sets, having a computational complexity of $\mathcal{O}(n^3)$, where n is the number of points in the training set. For this reason there has been considerable recent interest in Gaussian processes which have been widely used in machine learning, for example; *voice activity detection*, [Park and Choi, 2008]; *object categorization* [Kapoor et al., 2007]; *robot control systems* [Nguyen-tuong and Peters, 2008], and *robot path planning* [Martinez-Cantin et al., 2007].

There is a strong link between sparse GPs and active learning. In their work on sparse GPs Lawrence et al. [2003] seek to speed up GPs by finding a sparse representation of the data. In their context (unlike active learning) the true class labels are already known and the problem is to select a representative sub-sample of labelled points on which to perform a classification without jeopardizing the classification. The method used is similar to the active learning approach of MacKay [1991] which seeks to minimize the posterior predicted entropy of $p(y \mid \mathbf{x})$. By selecting a sparse representation of preselected size d they were able to show comparable performance and run-times to SVMs.

It is unlikely that we can perform active learning *and* use sparse GPs in conjunction, as active learning has already preselected the data points for their relative informativeness about the classification. However, because active learning seeks to minimise the number of observations necessary to make a classification, this minimisation of the number of inputs in the training set also serves to reduce computational complexity. Intuitively therefore, GPs represent a good choice for performing active learning in the Bayesian setting.

Some work has been done on using GPs for active learning. For example, Kapoor et al. [2007] use a least squares classification[1] for binary classification in the case of object

[1]This is where classification is purely on the basis of the sign of the target value. This has the advantage of being straightforward to implement but does not fully model $p(y \mid \mathbf{x})$.

classification. They actively select new queries using a measure of uncertainty which considers both the mean and the variance of the predicted target value and is similar to the *Fisher linear discriminant* [Duda et al. 2000 pp 119]. Seo et al. [2000] use GPs to actively sample in the case of regression selecting the query which minimizes the posterior variance. More recently Krause and Guestrin [2007] have used active learning to select sensor readings when monitoring spatial phenomena. They point out that when the hyper-parameters of a GP are either fixed or known, the variance of the posterior distribution and therefore any criterion which depends solely on the variance, such as entropy or mutual information, does not depend on the target labels. Actual observations of the target labels are therefore only required in order to optimize the unknown hyper-parameters in the model. They give bounds on how much information can be gained by observing class labels for a given prior distribution on the hyper-parameters. Their approach allows *a priori* selection of query points – which is clearly useful in the case of selecting where to put sensors – and indicates when an active query is likely to outperform an *a priori* design.

However, none of the above approaches directly minimizes the objective function on which the active learner will ultimately be evaluated. In contrast Zhu et al. [2003] employ an expected cost reduction approach using Gaussian random fields (which is another form of Gaussian process). Their approach combines active learning with semi-supervised learning where the classifier is trained on both the labelled and unlabelled examples. They show significant improvement over uncertainty sampling.

Zhu et al.'s approach to active learning uses the same objective function as ours; the minimum expected misclassification cost. However, the classifier applied is equivalent to a least squares classification with a Gaussian process (the details of which are described in section 4.5.2), where the posterior class predictions are estimated by a sigmoid on the regression function.

In contrast we use the expectation propagation algorithm [Minka, 2001] to estimate the true Bayesian posterior predictive density. This therefore represents a fully Bayesian approach to the expected cost reduction form of active learning.

4.5.1. GP regression

We start by briefly introducing Gaussian processes as they are applied in the case of regression. This is useful as it forms the basis for understanding classification with GPs. We then describe in the next section how GPs are used in classification. It is also worth starting with regression because in the case of a Gaussian prior and Gaussian likelihood function the posterior predictions are Gaussian and can be written in closed analytic form. This is not the case for classification where some approximation approach is usually necessary. This section and the next section are based on Rasmussen and Williams [2006], which is the definitive book on Gaussian processes.

Following similar notation to Rasmussen and Williams [2006], given some training data $\mathcal{D} = \{(\mathbf{x}_i, y_i) \,|\, i = 1, \ldots, n\}$ where $\mathbf{x}_i$ has dimension D we can compile these into a $D \times n$ design matrix $\mathbf{X}$ and a target vector $f \in \mathbb{R}^n$ so that $\mathcal{D} = \{\mathbf{X}, f\}$. For a similarly constructed design matrix $\mathbf{X}_*$ of test data, the Gaussian process is then able to make predictions of the corresponding target means f_* and its covariance matrix $cov(f_*)$.

The basic idea behind the GP is to construct a high dimensional Gaussian distribution where each dimension corresponds to a single data point $\mathbf{x}$. This Gaussian is fully defined by a covariance function $\mathbf{k}(\mathbf{x}, \mathbf{x}')$ and a mean function $\mathbf{m}(\mathbf{x})$. For the finite set of observation points $\mathbf{x}$ on which it is defined, the Gaussian represents a distribution over function space:

$$f(\mathbf{x}) \sim GP(\mathbf{m}(\mathbf{x}), \mathbf{k}(\mathbf{x}, \mathbf{x}')) \tag{4.11}$$

In other words if the GP is defined on t values of $\mathbf{x}$ then the scalar value of $f(\mathbf{x})$ at these points can be seen as a single vector drawn from the t-dimensional Gaussian $GP(\mathbf{m}(\mathbf{x}), \mathbf{k}(\mathbf{x}, \mathbf{x}'))$, where $\mathbf{k}(\mathbf{x}, \mathbf{x}')$ is the covariance between $\mathbf{x}$ and $\mathbf{x}'$ for some kernel $\mathbf{k}$. The covariance function defines the distance between inputs and affects how much an observation at one input affects the likelihood for another. In this way the GP represents a distribution over the function space of $f(\mathbf{x})$. In terms of the design matrix $\mathbf{X}$ we can rewrite the covariance function in matrix form as $\mathbf{K}(\mathbf{X}, \mathbf{X})$. Here we will only consider the zero-mean prior where $\mathbf{m}(\mathbf{x}) = \mathbf{0}$. This is not a significant constraint as data can be

mean centred before to being given to the GP. The joint distribution over all targets f and f_* is then:

$$\begin{bmatrix} f \\ f_* \end{bmatrix} \sim \begin{bmatrix} \mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma^2\mathbf{I} & \mathbf{K}(\mathbf{X},\mathbf{X}_*) \\ \mathbf{K}(\mathbf{X}_*,\mathbf{X}) & \mathbf{K}(\mathbf{X}_*,\mathbf{X}_*) \end{bmatrix} \tag{4.12}$$

where we have added the term $\sigma^2 I$ which corresponds to noise on the observations f. To perform regression for the test data $\mathbf{X}_*$ and find the posterior distribution for f_* we then find the conditional distribution consistent with the observations f. Writing $\mathbf{K}(\mathbf{X},\mathbf{X})$ as $\mathbf{K}$, $\mathbf{K}(\mathbf{X}_*,\mathbf{X})$ as $\mathbf{K}_*$, and $\mathbf{K}(\mathbf{X}_*,\mathbf{X}_*)$ as $\mathbf{K}_{**}$, Rasmussen and Williams [2006] show that the GP predictions are then given by:

$$f_* = \mathbf{K}_*[\mathbf{K} + \sigma^2\mathbf{I}]^{-1} f \tag{4.13}$$

$$cov(f_*) = \mathbf{K}_{**} - \mathbf{K}_*[\mathbf{K} + \sigma^2\mathbf{I}]^{-1}\mathbf{K}_*^T \tag{4.14}$$

where f_* is the mean of the conditional density, and $cov(f_*)$ its covariance.

4.5.2. GP classification

While for regression where the observations are continuous and real valued, a Gaussian likelihood function is plausible and leads to an analytically tractable posterior distribution, in general this is not the case for classification. To use GPs for binary classification the simplest approach and easiest to understand is to label the two classes $+1$ and -1 so that $y \in \{-1, +1\}$ and then perform regression on the observations as though they were real valued observations [2]. When the misclassification cost is the same for either class we could simply take the sign(f) to be the class prediction. This turns out to work fairly well and is sometimes called least squares classification in the literature, see section 6.5 of Rasmussen and Williams [2006]. At the limit of an infinite number of observations least squares classification will tend to the Bayes classifier [Rasmussen and Williams, 2006]. However, it does not provide the posterior probability of $\mathbf{x}_*$ being in either class, without which we cannot accurately estimate the expected misclassification costs on the unlabelled data. To give a value which can be interpreted as a probability the regression output f can

[2] In least squares classification the target labels can be changed from $\{-1, +1\}$ to other values in order to accommodate imbalanced classification costs.

be mapped to $(0, 1)$ with a sigmoidal link function $\sigma(f(\mathbf{x}))$, such as the logistic function or probit function to give:

$$p(y = +1 \mid \mathbf{x}) = \sigma(f(\mathbf{x})) \tag{4.15}$$

However, this is somewhat *ad hoc* because in its current state the model is treating the target observations $\{-1, +1\}$ as though they were Gaussianly sampled continuous real values.

The full Bayesian approach, described in chapter 3 of Rasmussen and Williams [2006], is a more principled approach which treats f as a latent variable, where equation (4.15) defines the likelihood of observing a particular class label given the latent variable. Notice that because $y \in \{-1, +1\}$ we can write the probability of class membership as $p(y_i \mid f_i) = \sigma(f_i \cdot y_i)$. Then $p(f_* \mid \mathbf{X}, \mathbf{x}_*, f)$ is still found in the same way as the regression case (equation (4.13)), because it does not depend on the class labels y. The posterior density of the latent variables f_* given the class labels is then found by integrating out the latent variable f:

$$p(f_* \mid \mathbf{X}, y, \mathbf{x}_*) = \int p(f_* \mid \mathbf{X}, \mathbf{x}_*, f) p(f \mid \mathbf{X}, y) df. \tag{4.16}$$

Class predictions are made using:

$$p(y_* = +1 \mid \mathbf{X}, y, \mathbf{x}_*) = \int \sigma(f_*, y_*) p(f_* \mid \mathbf{X}, y, \mathbf{x}_*) df_* \tag{4.17}$$

where $\sigma(f_* \cdot y_*)$ is the likelihood of observing y_* given f_*, and $p(f \mid \mathbf{X}, y)$ in equation (4.16) is the likelihood of the latent variable given the observed class labels. This is given by Bayes' rule:

$$p(f \mid \mathbf{X}, y) \propto p(f \mid \mathbf{X}) \prod_{i=1}^{n} \sigma(f_i, y_i) \tag{4.18}$$

where $p(f \mid \mathbf{X})$ is Gaussian and the product of sigmoids is the combined likelihood $p(y \mid f)$. However, because of the presence of the sigmoid function, in general equations (4.17) and (4.18) are not analytically tractable. When $p(f \mid \mathbf{X}, y)$ is not a Gaussian, equation (4.16) is also intractable.

One approach to Bayesian inference is to use Markov chain Monte Carlo (MCMC) approaches [Gamerman, 2002]. These, however, are not feasible here because the active learning algorithm requires multiple runs in each time-step. Rasmussen and Williams [2006] describe two algorithms which make analytical approximations to the posterior distributions. The Laplace approximation which approximates the true posterior by a single Gaussian at its mode, and the Expectation propagation (EP) algorithm from Minka [2001]. They compare the Laplace approximation with the EP algorithm on a 3s v 5s digit discrimination task, and found that, while the difference in test errors between EP and the Laplace approximation was small, the posterior distribution of the EP was considerably closer to the full MCMC approximation than the Laplace approximation.

Expectation propagation works by approximating $p(f \mid \mathbf{X}, y)$ by a Gaussian distribution $q(f \mid \mathbf{X}, y)$:

$$q(f \mid \mathbf{X}, y) = \frac{1}{Z_{EP}} p(f \mid \mathbf{X}) \prod_{i=1}^{n} \tilde{Z}_i \mathcal{N}(f_i \mid \tilde{\mu}_i, \tilde{\sigma}_i^2) \tag{4.19}$$

Here $\tilde{Z}_i \mathcal{N}(f_i \mid \tilde{\mu}_i, \tilde{\sigma}_i^2)$ are un-normalized Gaussians which we call t_i, and stand in for the sigmoid functions $p(y_i \mid f_i)$. Starting with an initial estimate of $q(f \mid \mathbf{X}, y)$ the values of $\tilde{Z}_i$, $\tilde{\mu}_i$ and $\tilde{\sigma}_i$ are determined by an iterative process. Each t_i is updated in turn, and the process is repeated until convergence is reached. t_i is updated by first removing it from the marginal distribution $q(f_i \mid \mathbf{X}, y)$, which is also Gaussian and can be found by matrix algebra from $q(f \mid \mathbf{X}, y)$. The *cavity distribution* for t_i which is called $q_{-i}(f_i)$ is then found by dividing $q(f_i \mid \mathbf{X}, y)$ by t_i (this again is Gaussian and can be found with matrix algebra). The cavity distribution is combined with the true likelihood $p(y_i \mid f_i)$ to give an improved estimate of the true marginal likelihood for f_i, which is then approximated by a new Gaussian $\hat{q}(f_i)$:

$$\hat{q}(f_i) \approx q_{-i}(f_i) p(y_i \mid f_i) \tag{4.20}$$

This approximation is made by matching the first and second moments of the distributions, which is equivalent to minimizing the KL divergence between the R.H.S and L.H.S of equation (4.20) (to see how the moments of the R.H.S are found see section 3.6 and 3.9

of Rasmussen and Williams [2006]). Because $\hat{q}(f_i)$ is un-normalized the zero-th moment or normalizing constant is also matched. Finally, t_i is then updated by matching the moments of t_i^{new} to ensure that:

$$t_i^{new} q_{-i}(f_i) = \hat{q}(f_i) \tag{4.21}$$

This process is repeated for all t_i until convergence is reached. Having found a Gaussian estimate $q(f \mid \mathbf{X}, y)$ for $p(f \mid \mathbf{X}, y)$ this can then be substituted in equation (4.16) which now becomes tractable because everything is Gaussian, giving a Gaussian estimate of $p(f_* \mid \mathbf{X}, y, \mathbf{x}_*)$. This is then substituted in equation (4.17) to give probabilities for class predictions $p(y_* = +1 \mid \mathbf{X}, y, \mathbf{x}_*)$, and in the case of a probit sigmoid function (4.17) has an analytic solution. For full details of the EP algorithm the reader is referred to Chapter 5 of Rasmussen and Williams [2006].

4.6. Implementation and results

The two algorithms in section 4.4 show the expected cost reduction approach to active learning in the pool-based setting. This is implementable for any computationally feasible classifier capable of producing probabilities of class membership. The classifier used to implement this approach here is a Gaussian process classifier using the expectation propagation approximation to the posterior distribution. This has the advantage of being relatively fast to implement and is therefore feasible to implement on reasonably sized data-sets.

With the GP the inputs $\mathbf{x}$ can be compiled into a single matrix $\mathbf{X}$ and the output probabilities calculated in one step. To evaluate $\tilde{R}(\mathbf{x}_q)$ for one potential query $\mathbf{x}_q$, the time taken to evaluate the posterior class predictions $p(y = 1 \mid \mathbf{x}, \mathbf{x}_q, y_q, \mathcal{D}_n)$ for all inputs $\mathbf{x}$ in the unlabelled set is of order $\mathcal{O}(m^3)$ where m is the total number of inputs in the pool $\mathcal{P} = \mathcal{D}_n \cup \mathcal{U}_n$. This must be evaluated twice, once for each class label y_q. The estimate of the probability of class membership for the query $\mathbf{x}_q$, $p(y_q \mid \mathbf{x}_q, \mathcal{D}_n)$ must also be evaluated. This takes time $\mathcal{O}(n^3)$ where n is the total number of observations in the training set.

We implement this approach on six benchmark data-sets and demonstrate that it consistently outperforms random queries. In each case we use a probit likelihood function:

$$\sigma(\mathbf{x}) = \frac{1}{2}\left(1 + \text{erf}\left(\frac{\mathbf{x}}{\sqrt{(2)}}\right)\right) \tag{4.22}$$

and the squared exponential (SE) covariance function which can be written as:

$$\mathbf{k}(\mathbf{z}_p, \mathbf{z}_q) = \nu \exp(-\frac{1}{2}(\mathbf{z}_p - \mathbf{z}_q)'\mathbf{L}^{-1}(\mathbf{z}_p - \mathbf{z}_q)). \tag{4.23}$$

The hyper-parameters for the covariance function in equation (4.25) are ν, which scales the overall magnitude of the function, and the length-scale matrix, $\mathbf{L}$. For $\mathbf{z}$ of dimension d the matrix $\mathbf{L}$ is a diagonal matrix of dimension $d \times d$, whose diagonal elements are the square of the characteristic length-scale in each dimension. The length-scale is a measure of how rapidly the function value changes across input space. These are the hyper-parameters for the covariance function and the only other hyper-parameter for the GP is a scalar parameter which defines the mean of the prior on the function values. The SE covariance function is often used because it has a relatively small number of hyper-parameters and makes a plausible continuity assumption.

The Rasmussen and Nickisch implementation also has the ability to tune the hyper-parameters. It does this by maximizing the posterior likelihood of the data given the hyper-parameters. On each data-set we tuned the hyper-parameters on the full labelled data-set. The same hyper-parameters were then used to test both random query and the active learning algorithm.

We have not addressed the issue of how the hyper-parameters should be tuned 'actively', when the true labels are not known. Tuning the hyper-parameters is itself computationally intensive, and it's not clear how a continually changing set of hyper-parameters would affect posterior predictions in the active learning process. However, in our experiments we found that the results were relatively robust against changes in the hyper-parameters.

We start with a simple illustration to show how the decision boundary changes at each step in a run through of several time-steps of the algorithm. We show this on a

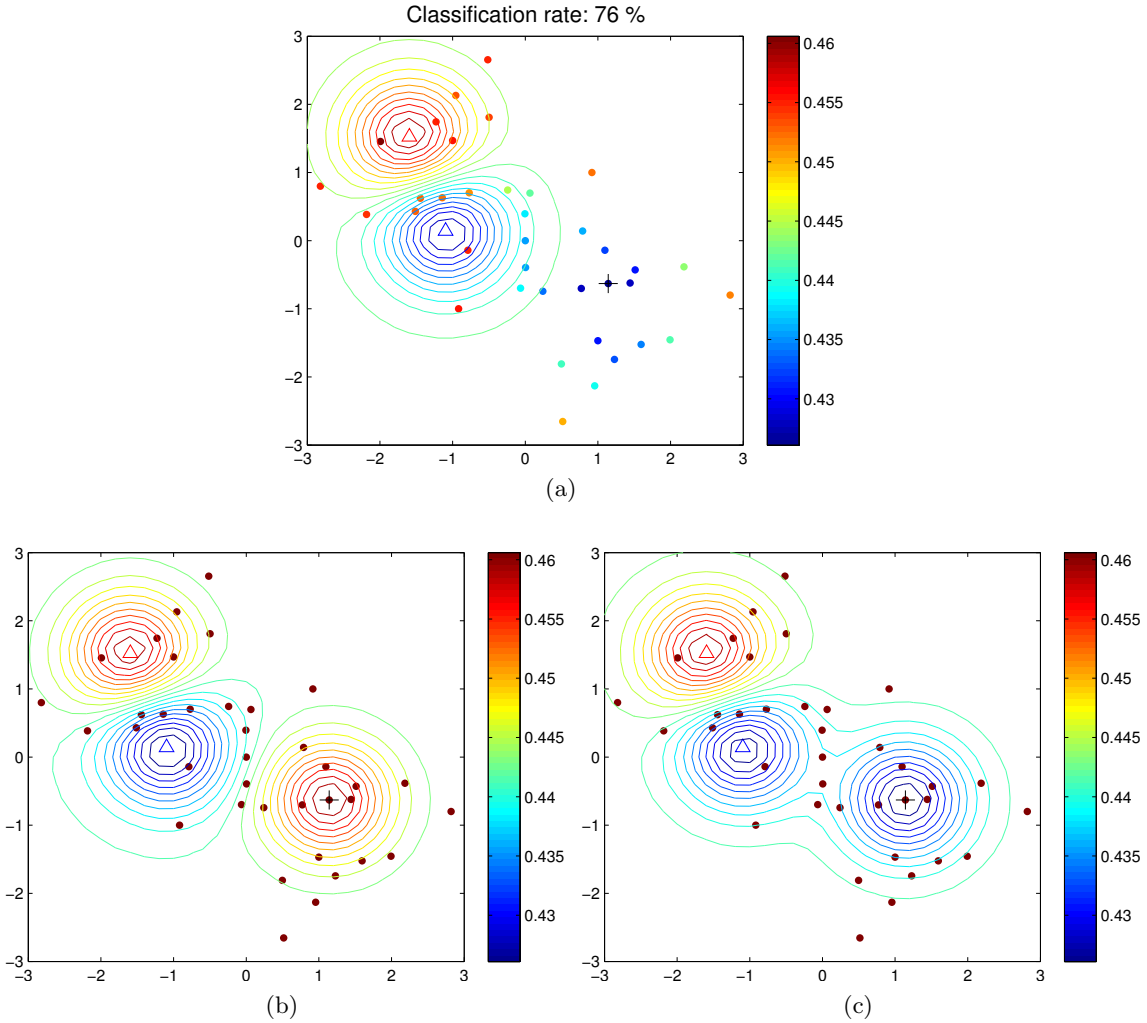


Figure 4.1.: Step 1 in algorithm: The single blue and red triangles in (a) indicate the initial observations, one from class -1 and $+1$ respectively. The initial observation in class -1 is in fact misleading as we can see from later steps (see step 36). (a) indicates the current decision boundary. The colour of the dots in (a) shows the mean misclassification cost estimated for a query at each of the unlabelled inputs. The black $+$ indicates the unlabelled point with the minimum predicted misclassification cost. (b) shows the posterior decision boundary if the query at $+$ is in class 1 (red). (c) shows the posterior decision boundary if the query is in class -1 (blue).

simple symmetric data-set with only a small number of training points in each class which was used to initially test the algorithm. This data-set has 40 inputs in the training set, and 100 inputs in the test set, with an equal number of observations from each class. The distribution of each class is a spherical Gaussian with unit variance. The means of class 1 and class -1 are $(-1, 1)$ and $(1, -1)$ respectively. For this data-set the maximum achievable classification rate is $\approx 90\%$.

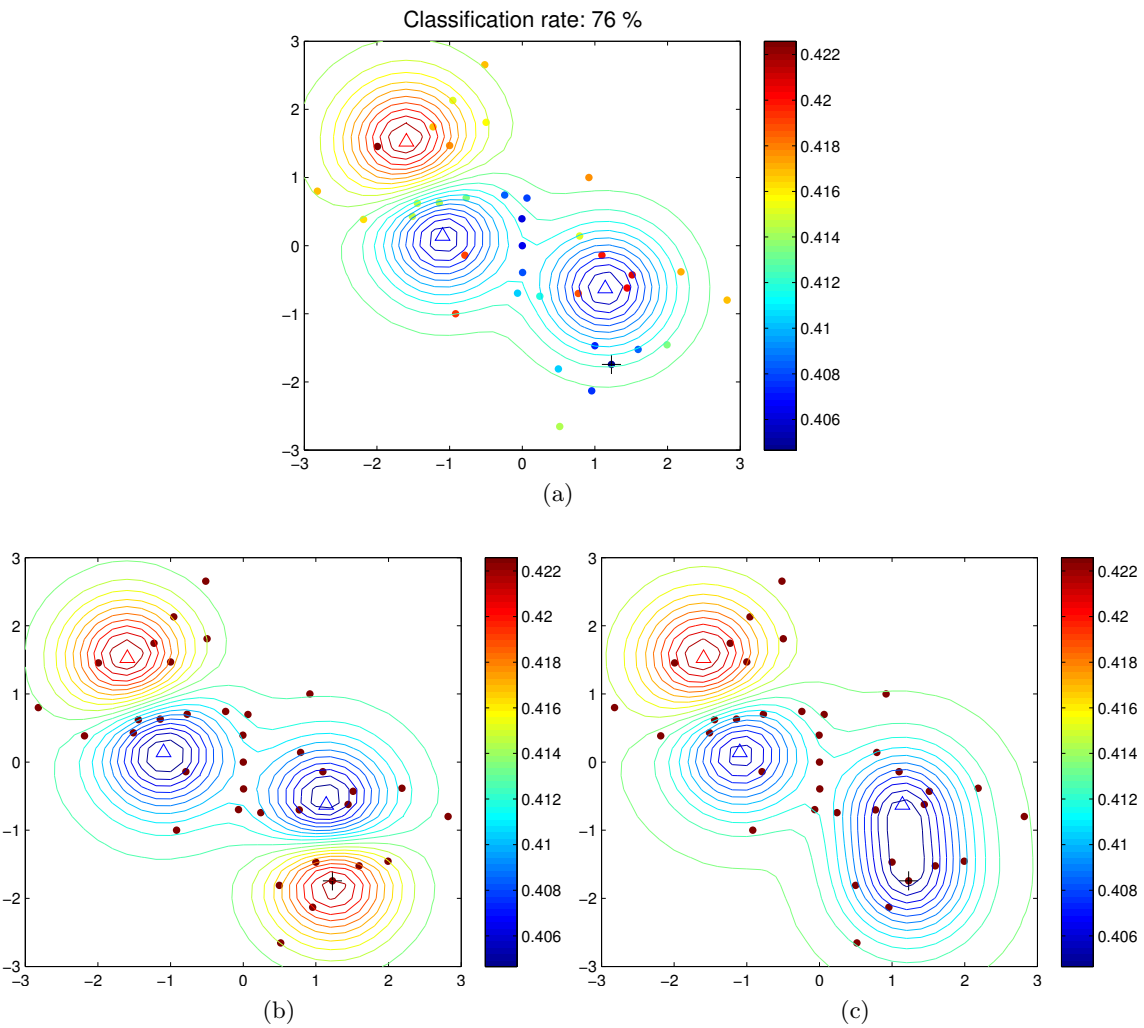


Figure 4.2.: Step 2: Similar to step 1. The extra blue triangle indicates that the label oracle has returned a class label -1 at the query point from step 1

At initialization the algorithm starts with one labelled observation from each class as shown in figure 4.1. This is an interesting example because the initial labelled point from class -1 is an outlier of its class, lying on the 'wrong' side of the decision boundary, which is not representative of the underlying distribution and is therefore misleading. Figures 4.1 to 4.6 show time-steps of the algorithm. In each figure panel (a) shows the current decision boundary, as shown by the contour lines. Previous labelled observations are shown with a coloured triangle, blue for class -1 and red for class 1. The dots represent unlabelled inputs. In the panel (a) these are coloured according to the average expected misclassification cost associated with a query at that point, $\tilde{R}(\mathbf{x}_q)$. The black + shows the unlabelled point with the minimum expected cost. This is the $\mathbf{x}_q$ selected by the algorithm in this time-step. The panel (b) shows the predicted decision boundary if the class label

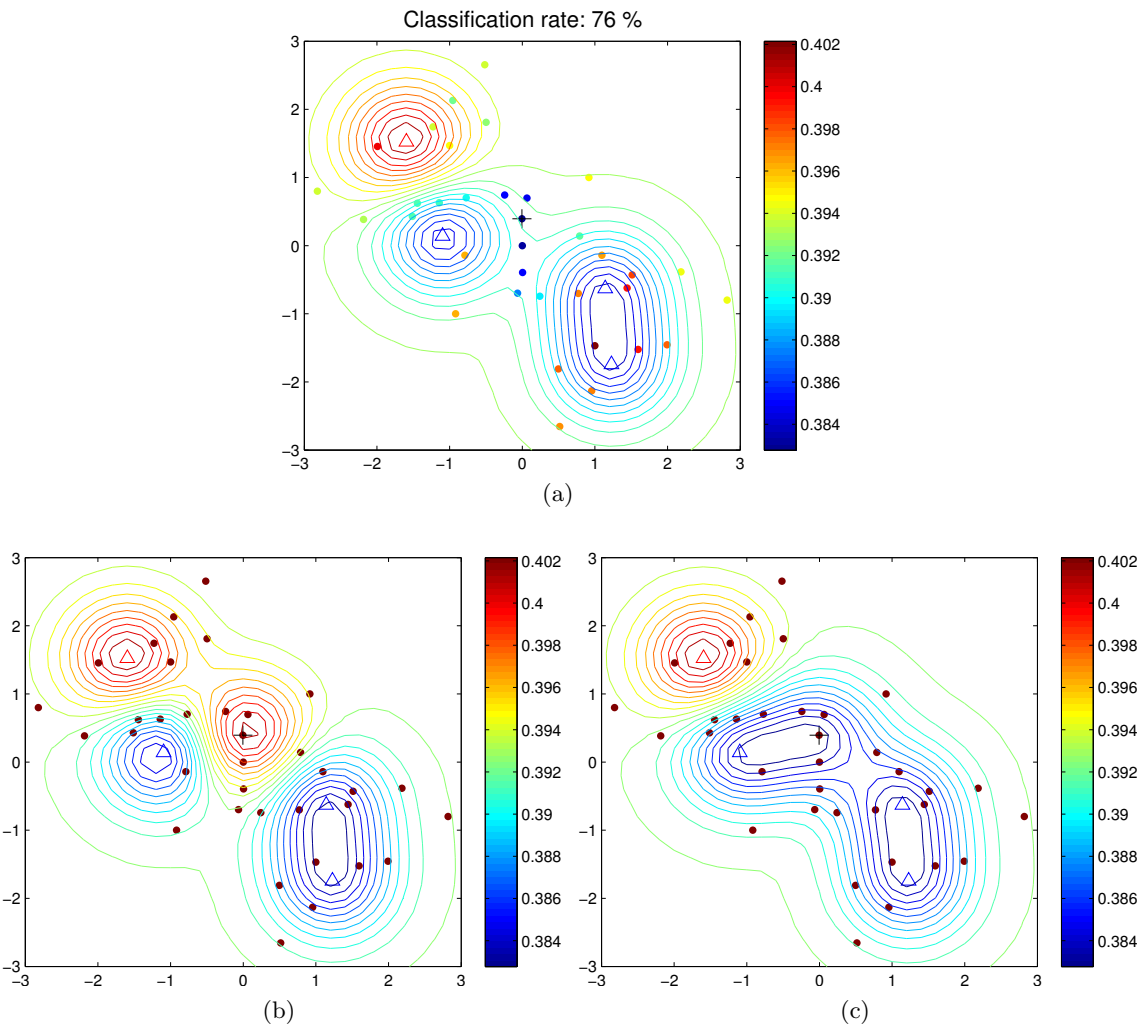


Figure 4.3.: Steps 3: In this step and in step 4 the posterior for a class 1 label (b) and a class -1 label (c) is significantly different. Indicating that the algorithm tends to choose queries which could significantly affect future class predictions on the unlabelled data.

is in class 1 and the (c) panel shows the predicted decision boundary for a class label -1. The title of the (a) panel shows the current classification rate as measured on the unseen test data, before the query is selected.

As we can see in the first step (figure 4.1) the algorithm selects what appears to be a sensible query point. This observation is in a region which is distant from previous observations but is likely to be informative about the class label of other inputs. Step 2 (figure 4.2) again explores the region in class -1. Notice this does not improve the classification rate because the class label which is -1 (blue), does not change the class predictions (this was already a $+1$ region before the query). Step 3 (figure 4.3) now

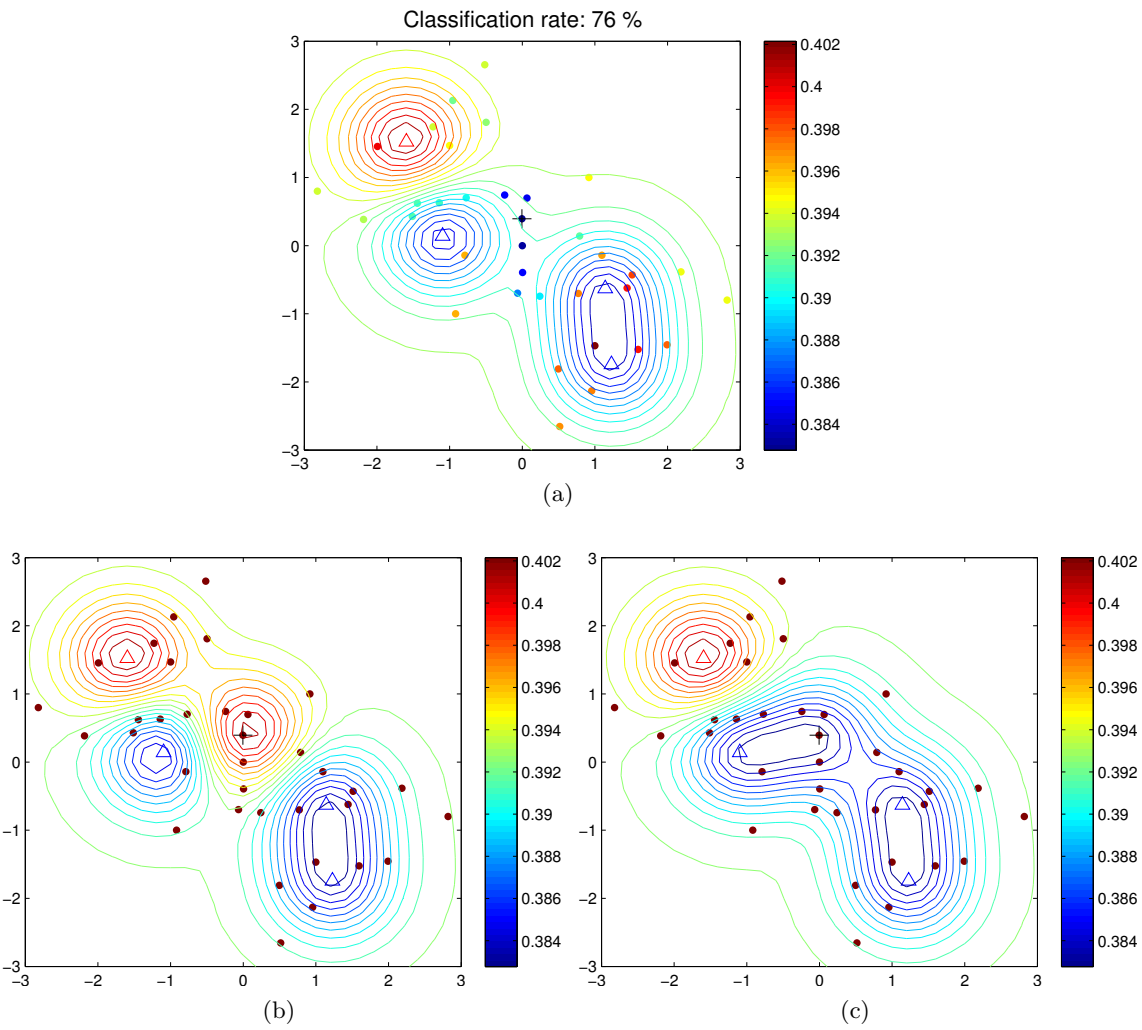


Figure 4.4.: Steps 4: similar to step 3

makes a query which is in the +1 region, and is relatively close to the initial misleading observation. Here the classification rate jumps to 81%. The query in step 4 (figure 4.4) further tests the decision boundary between the two classes and by the 15th query the misleading point has been 'isolated' and the decision boundary is not very different to step 36 (figure 4.6) where most of the training data has been labelled.

The algorithm tends to find a balance between 'exploration' and 'exploitation'. This is because the expected misclassification cost is measured across all unlabelled data, not just a single observation. If we were to sample the unlabelled input about which we were most uncertain for example, step 1 would have queried further away from the initial observations; at either the bottom or right hand side of figure 4.1. Neither of these queries would have been so informative about the remaining unlabelled data. It seems therefore,

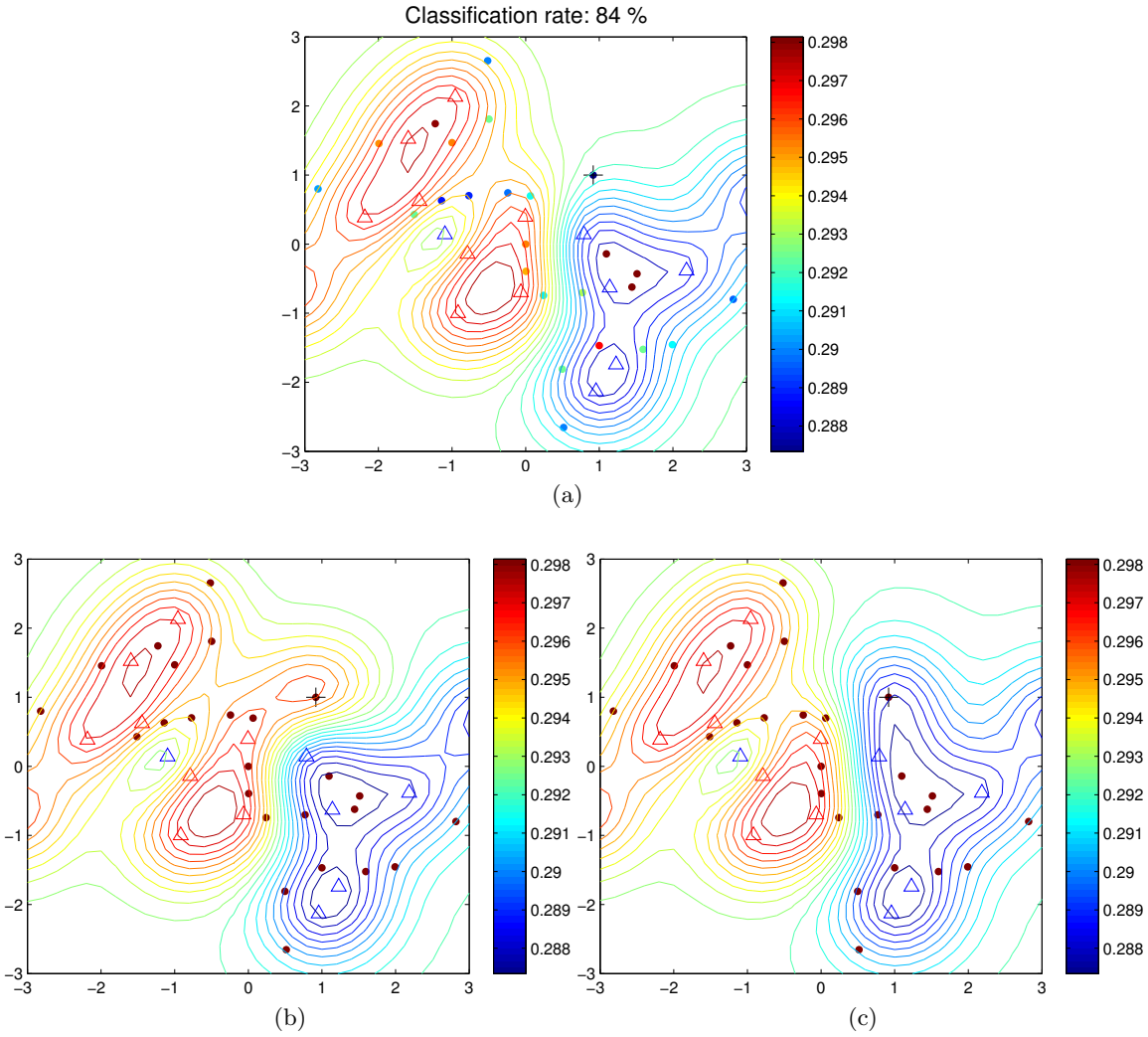


Figure 4.5.: Step 15: We can see that by the 15th query the true class boundary has started to form. Despite the initially misleading observation in class −1.

that the algorithm is querying 'sensibly', given the information it has at its disposal.

4.6.1. Benchmark data-sets

We ran the algorithm on six benchmark classification data-sets, Ripley's synthetic data which was used as a benchmark data-set in Ripley [1996], the *New Ripley* data which is a modification of the original Ripley data [Fieldsend et al., 2006] and the *Ionosphere* which was used as a benchmark data-set by Campbell et al. [2000] and can be found at the UCI repository [Frank and Asuncion, 2010]. The final three data-sets, the *Banana* data, the *Pima Diabetes* data, and the *Breast Cancer* data were used as a benchmark data-set by Tipping [2001] and can also be found [Rätsch, 2011] and [Frank and Asuncion, 2010].

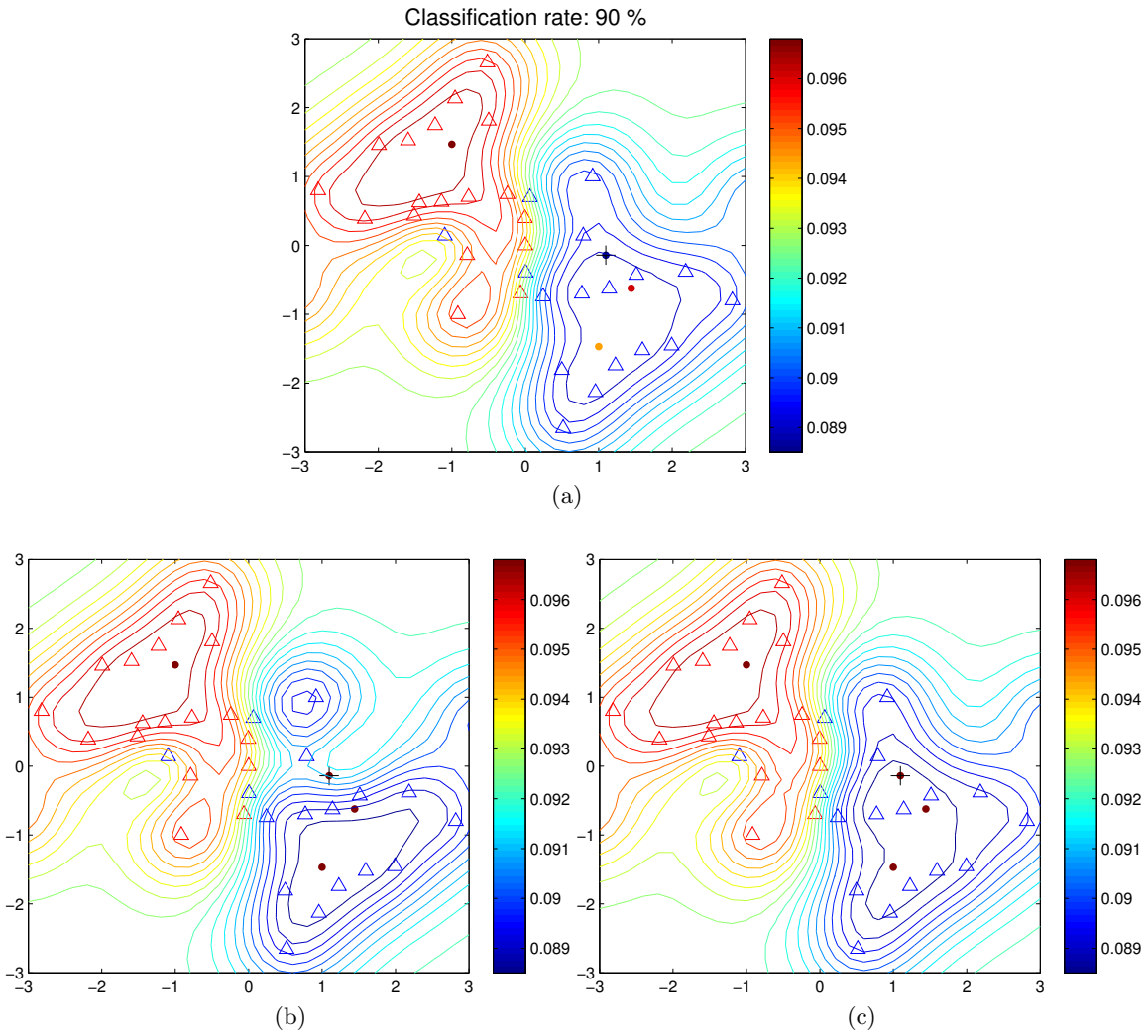


Figure 4.6.: Step 36: Here nearly all the unlabelled points have been queried and we can see the true class labels.

For each benchmark set the algorithm was run 100 times on both active learning and random query, for comparison. In each of the active learning runs, the pool of labelled inputs was initialised with one labelled input from either class, after which the active learner queries one new input in each time-step, for 50 iterations. On each data-set it is assumed that the misclassification cost for either class is the same and therefore the classification threshold $\lambda = 0.5$.

Table 4.1 shows the dimension and size of each of the benchmark data-sets, together with an estimate of the maximum classification rate and AUC. The classification rate measures the performance for a fixed classification threshold λ, while the AUC measures performance across all possible values of λ and is a measure of 'robustness' to variation

Data-set	Dimension	Training	Test	Class Rate	AUC	Comment
Ripley	2	250	1000	91.06%	0.972	Bayes accuracy
New Ripley	2	250	1000	91.68%	0.975	Bayes accuracy
Banana	2	400	4900	88.2%	0.96	empirical
Ionosphere	34	200	151	95.8%	0.98	empirical
Pima Diabetes	8	468	300	76.5%	0.84	empirical
Breast Cancer	9	200	77	73.9%	0.73	empirical

Table 4.1.: The dimension, the number of training, and test inputs, for each of the six benchmark data-sets. The classification rate and the AUC represent the maximum achievable classification rate on each of the data-sets. For the Ripley and New Ripley data the figures given are approximate to the true Bayes maximum classification rate and AUC. The empirical values, given for the other data-sets, represent the classification rates achieved by the GP trained on all of the training data.

in λ. However, in this model it is assumed that the classification threshold λ is known and therefore the classification rate is a better measure of the learner's performance on the objective by which it has learned. If there is uncertainty about the true classification costs, then uncertainty about the decision threshold should be incorporated into the cost function.

In the case of the Ripley data and the new Ripley data, because the true distributions are known mixtures of Gaussians, the Bayesian optimal classification rate can be estimated. For each of the other data-sets the estimate is based on the GPs performance given all of the training data. For Pima diabetes and breast cancer we give the average classification rate over each of the ten folds.

Figures 4.7 and 4.9-4.13 report the classification rates and the AUC for both active learning (the blue lines) and random sampling (the red lines) over 100 runs. The (a) panels indicate the median classification rates over 100 runs, where the dotted lines give the upper and lower inter-quartile range. The (b) panels indicate the mean classification rates over 100 runs, where the dotted lines indicate one standard deviation above and below the mean. For both (a) and (b) the horizontal black line represents the estimated maximum classification rate given in table 4.1.

Note that the standard deviations on the (b) panels should be interpreted with caution, as it is known that the true distribution of rates is not symmetric about the mean. For

example, with both passive learning and active learning there is a small proportion of runs which perform significantly worse than average. This is not surprising as when the learner is unfortunate enough to query a sequence of inputs which are 'misleading' (in the sense that they are outliers of their class and lie on the 'wrong' side of the true decision boundary) then the learner requires more queries in order to 'compensate' and learn an accurate decision boundary. The interquartile range is therefore a better summary of performance, giving a more accurate representation of the spread of performance. The means are, however, worth reporting simply because the predicted cost reduction approach is specifically tailored to minimize the average misclassification cost. They therefore show performance of the algorithm on the specific criterion on which it is optimising.

The (c) panels show the median AUC rates achieved by the algorithm with dotted lines again indicating the inter-quartile range. Here the solid horizontal black lines represents the AUC of a perfect classifier and the dotted horizontal black lines indicates the estimated maximum achievable AUC, given in table 4.1. The horizontal red lines at 0.5 indicates the AUC of a 'lazy' classifier which assigns class labels randomly with probability 0.5. The AUC rates are given to indicate robustness in performance over different decision thresholds λ. However, it should be noted that the cost reduction approach to active learning with GPs is specifically optimising to minimise the classification error with the assumption that the misclassification threshold is known (in this case $\lambda = 0.5$). It is therefore unsurprising that performance under the AUC performance measure is not as consistent as performance on the mean classification rate.

4.6.2. Synthetic data-sets

The three synthetic data-sets, Ripley, New Ripley, and the Banana data are each 2 dimensional. Figure 4.8 shows an example labelled training set for each of the 3 data-sets together with the decision contour for the GP trained with a large amount of data.

Ripley data

The Ripley synthetic data [Ripley, 1996] is a data-set consisting of four spherical Gaussians (each with variance 0.03), two from either class. The two Gaussians in class 1 have centres

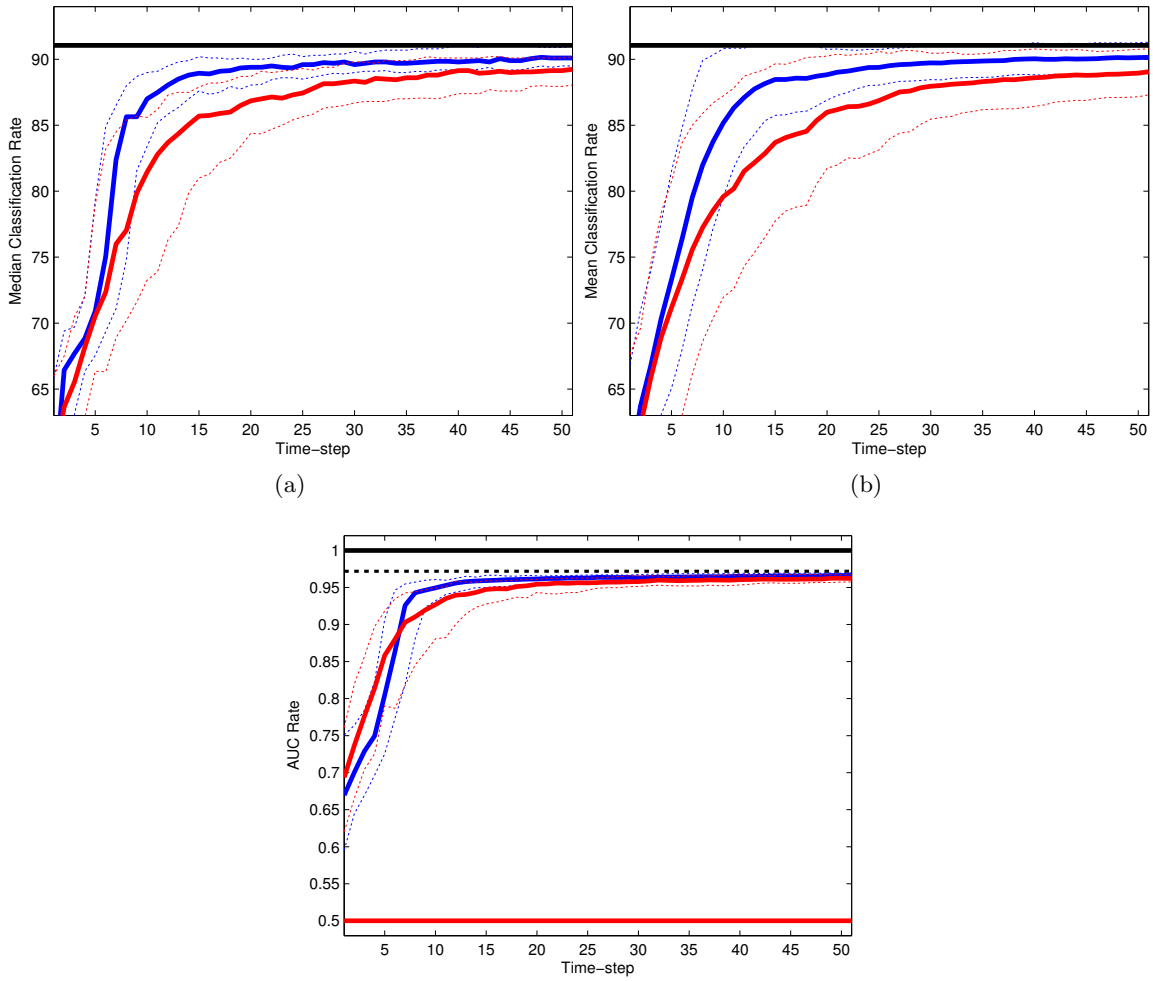


Figure 4.7.: Ripley Data: Classification rates and the AUC for both active learning (blue lines) and random sampling (red lines) over 100 runs. (a) indicates the median classification rate over 100 runs, where the dotted lines give the upper and lower inter-quartile range. (b) indicates the mean classification rates over 100 runs, where the dotted lines indicate one standard deviation above and below the mean. The solid horizontal black line in (a) and (b) represents the maximum achievable classification rate given in table 4.1 (c) shows the median AUC achieved by the algorithm with dotted lines indicating the inter-quartile range. Here the solid horizontal black lines represents the AUC of a perfect classifier and the dotted horizontal black lines indicates the estimated maximum achievable AUC.

$(-0.7, 0.3)$ and $(0.3, 0.3)$ and the centres for the two Gaussians in class -1 are $(-0.3, 0.7)$ and $(0.4, 0.7)$. The relative frequency of inputs from each of the four Gaussians is the same. In each run a new training set of 200 inputs, and test set of 1000 inputs was drawn from the underlying distribution. The decision boundary achieved by the GP on a fully labelled training set is shown by the contour plot in figure 4.8 (a). Inputs in class 1 are

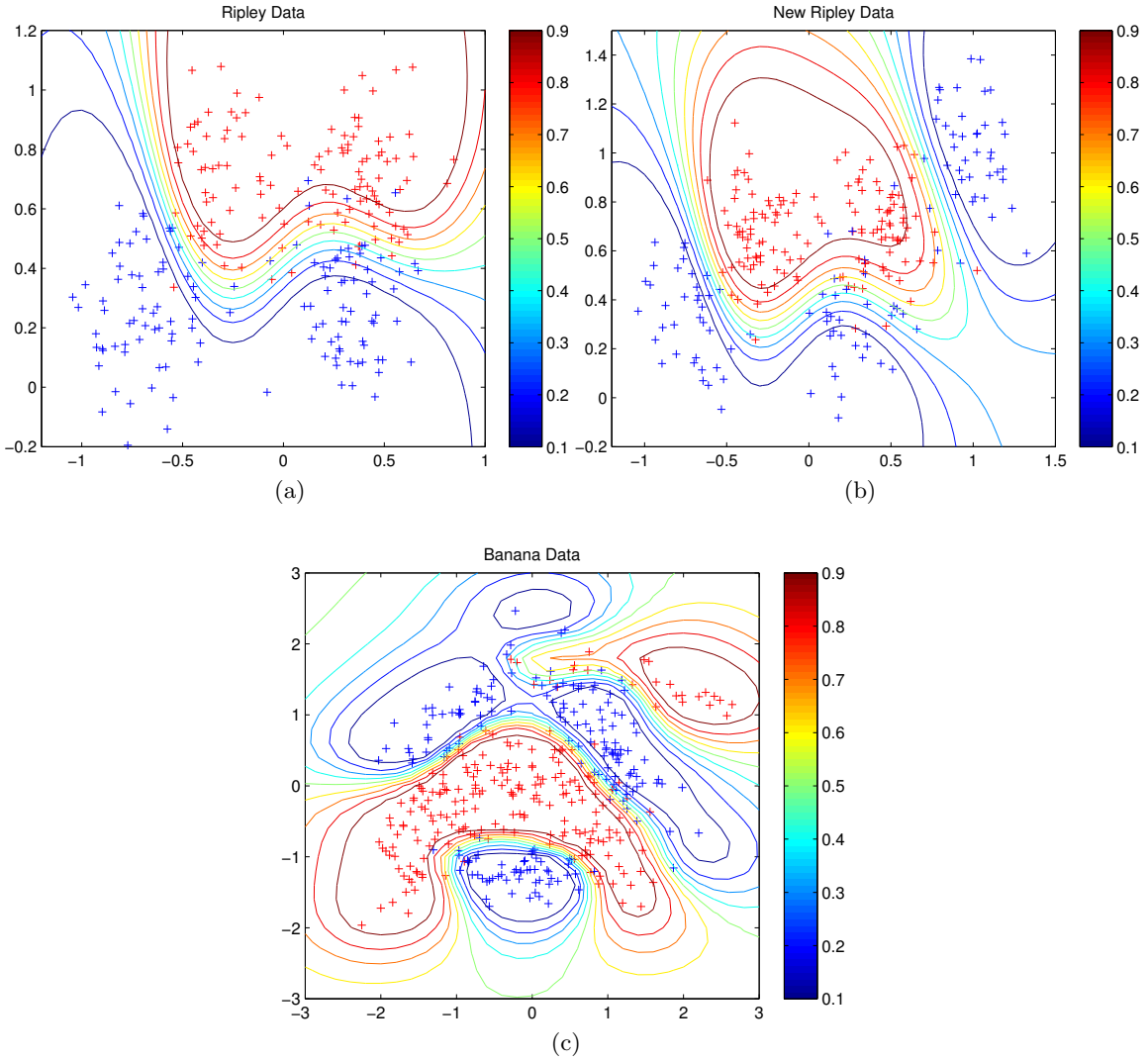


Figure 4.8.: Each panel shows the training data for a different synthetic data training set (a) Ripley data (b) New Ripley data (c) Banana data. Red + indicates a class label +1 and a blue + indicates a class label −1. Contours show the Gaussian process decision boundary given the entire labelled training set.

labelled by red +'s and inputs in class −1 by blue +'s. We can see that this data-set is reasonably well separated by the decision boundary with only a few *outliers* from either class straddling the decision boundary.

Figure 4.7 shows the active learning rates over 100 runs for the Ripley data. For this data-set the active learner consistently outperforms passive learning, and by the 10th observation the median classification rate is almost 10% higher than for passive learning; the passive learner would on average require at least double the number of inputs to achieve this rate. The active learner begins to plateau by about the 18th observation

but maintains a slightly higher classification rate for the full 50 observations. Notice that the interquartile range for the active learning are narrower than for the passive learner indicating less variation than for passive learning.

Figure 4.7 (b) is the equivalent figure showing the mean over 100 runs. Again, the blue lines represent active learning, and the red line represents passive learning. Performance for the mean is similar to that of the median and on average the GP active learning algorithm outperforms random queries, learning rapidly in the first 15 observations. Figure 4.7 (c) shows the AUC for active learning and passive learning in each time-step. Here, the dotted black line represent the Bayesian optimal AUC achievable for the Ripley data. Both active learning and passive learning are performing well by ≈ 20 observations. After 10 observations active learning does appear to have overtaken passive learning and quickly (by about 15 observations) plateaus to near the Bayesian optimal. The main observation here is that the IQR is narrower for active learning than for passive learning indicating a more consistent performance.

New Ripley data

The original Ripley data is a relatively straightforward classification problem, whose decision boundary, as we can see from figure 4.8(a), could be reasonably well approximated by a line. In order to force a more circuitous decision boundary another spherical Gaussian with variance 0.03 was added, with centre at $(1, 1)$ [Fieldsend et al., 2006]. The relative frequency of inputs in the two Gaussians for class -1 remains the same (0.25 for each Gaussian) as for the Ripley data, however the relative frequency in the three Gaussians, with centres $(1, 1)$, $(-0.7, 0.3)$ and $(0.3, 0.3)$ are 0.16, 0.17 and 0.17 respectively. Again the actual Bayes optimal classification rate and AUC can be estimated from the distributions; these are 91.68% and 0.975 respectively, as shown in table 4.3. A training data-set for the new Ripley data is shown in figure 4.8(b), together with the contour plot of the GP decision boundary trained on this data.

The performance of active learning and passive learning is shown in figure 4.9. Here too, active learning performs significantly better than passive learning when compared on either the mean or the median of the test classification rate. The difference between active and

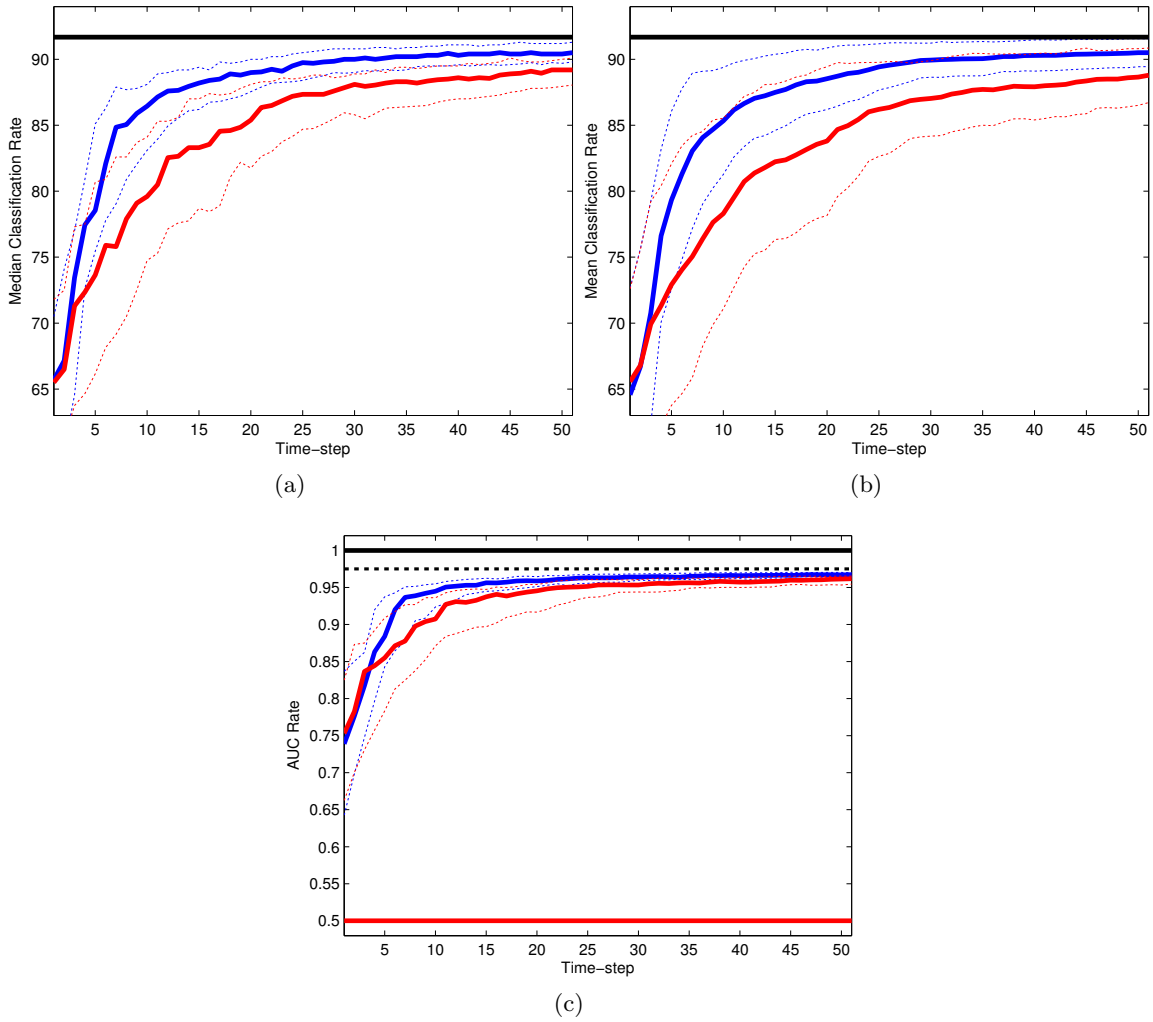


Figure 4.9.: New Ripley Data: Classification rates and the AUC for both active learning (blue lines) and random sampling (red lines) over 100 runs. (a) indicates the median classification rate over 100 runs, where the dotted lines give the upper and lower inter-quartile range. (b) indicates the mean classification rates over 100 runs, where the dotted lines indicate one standard deviation above and below the mean. The solid horizontal black line in (a) and (b) represents the maximum achievable classification rate given in table 4.1 (c) shows the median AUC achieved by the algorithm with dotted lines indicating the inter-quartile range. Here the solid horizontal black lines represents the AUC of a perfect classifier and the dotted horizontal black lines indicates the estimated maximum achievable AUC.

passive is maximum after 7 queries, indicating that the GP algorithm progresses rapidly in the initial phase of observation; again the passive learner requires about three times as many queries to achieve the same classification rate. When comparing the AUC rates active learning appears to outperforming passive learning in the first 20 observations. It is interesting how closely both active learning and passive learning reach to the Bayesian

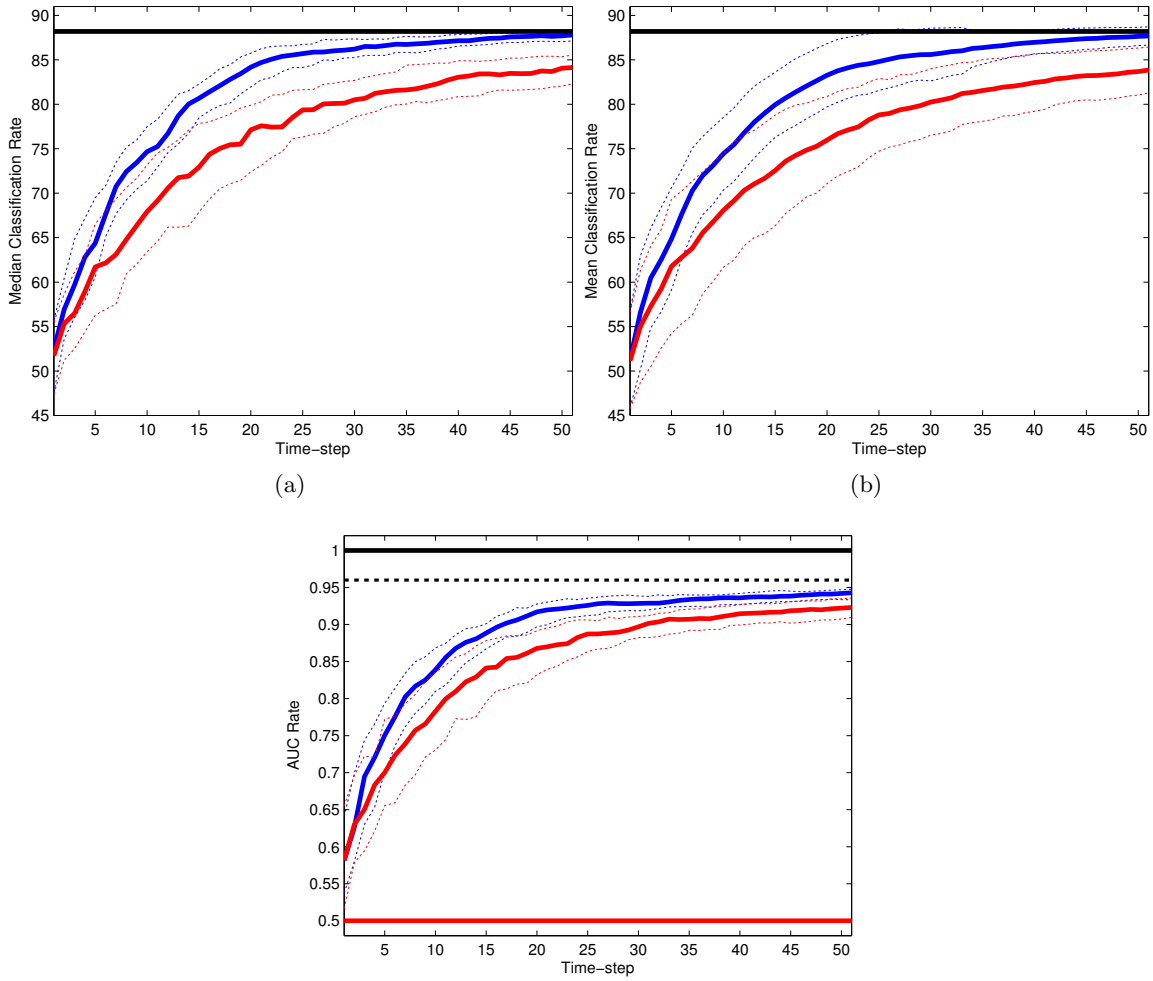


Figure 4.10.: Banana Data: Classification rates and the AUC for both active learning (blue lines) and random sampling (red lines) over 100 runs. (a) indicates the median classification rate over 100 runs, where the dotted lines give the upper and lower inter-quartile range. (b) indicates the mean classification rates over 100 runs, where the dotted lines indicate one standard deviation above and below the mean. The solid horizontal black line in (a) and (b) represents the maximum achievable classification rate given in table 4.1 (c) shows the median AUC achieved by the algorithm with dotted lines indicating the inter-quartile range. Here the solid horizontal black lines represents the AUC of a perfect classifier and the dotted horizontal black lines indicates the estimated maximum achievable AUC.

optimal (dotted black line), indicating that the GP performs particularly well on this data-set.

Banana data

For the Banana data the Bayes optimal classification rate and AUC were not known. Therefore, the horizontal black lines in figure 4.10(a) and (b), and the AUC rate given by the black horizontal dotted line in figure 4.10(c), represent the classification rate and AUC rate (respectively), achieved by the algorithm when given the entire training set of 400 inputs to learn on.

The Banana data shown in figure 4.8(c) appears to represent a more difficult classification problem than either the Ripley data or the New Ripley data. The blue cluster (class -1) in the middle at the bottom of figure 4.8(c), and the red cluster (class 1) in the top right both represent potential hazards for the learner. If the learner has no observations in the blue cluster (class -1) for instance then the decision boundary will likely label all inputs in this cluster as being red (class 1) severely affecting the classification rate. The same is true for points in the red cluster (class 1) in the top right hand corner. An active learning algorithm which only seeks to refine the current decision boundary without reference to how informative that query is to other unlabelled inputs in the pool could easily be led astray here.

It is therefore surprising how well the GP active learning algorithm performs on this data-set. Figure 4.10(a) shows significant separation between the median classification rate achieved by active learning and random sampling. After 11 observations the lower quartile for active learning (blue) is greater than the upper quartile for passive learning. We again see that the interquartile range for active learning is smaller than for passive learning indicating more consistent performance. Even the AUC, which the GP active learning algorithm has not specifically trained to optimise, performs significantly better than passive learning. This may be because the passive learner must wait for queries in the two potentially 'hazardous' clusters while the active learning algorithm will seek to reduce uncertainty about the unlabelled inputs in these regions and therefore queries inputs within those clusters sooner than random sampling.

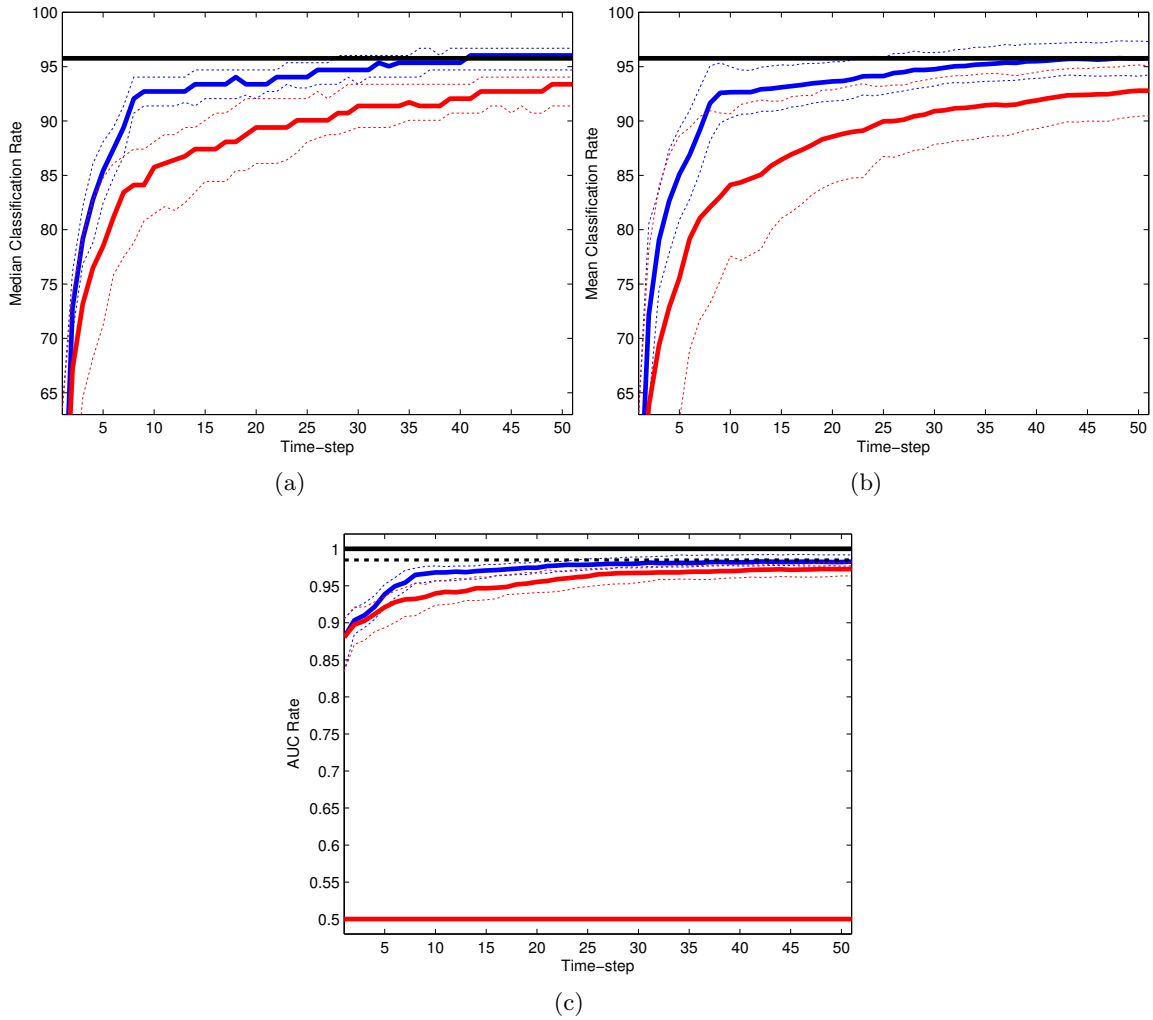


Figure 4.11.: Ionosphere Data: Classification rates and the AUC for both active learning (blue lines) and random sampling (red lines) over 100 runs. (a) indicates the median classification rate over 100 runs, where the dotted lines give the upper and lower inter-quartile range. (b) indicates the mean classification rates over 100 runs, where the dotted lines indicate one standard deviation above and below the mean. The solid horizontal black line in (a) and (b) represents the maximum achievable classification rate given in table 4.1 (c) shows the median AUC achieved by the algorithm with dotted lines indicating the inter-quartile range. Here the solid horizontal black lines represents the AUC of a perfect classifier and the dotted horizontal black lines indicates the estimated maximum achievable AUC.

4.6.3. Non-synthetic data-sets

The following three data-sets, Ionosphere, Pima diabetes, and Breast Cancer [Frank and Asuncion, 2010] are not synthetic and have been extensively used to test the performance of machine learning algorithms. The Ionosphere data is 34 dimensional, while the Pima diabetes and Breast Cancer data are 8 and 9 dimensional respectively. Therefore, the

example decision boundaries shown in figure 4.8 for the synthetic data-sets, cannot be replicated for the non-synthetic data. While the Ionosphere data is the highest dimensional data-set (of the six benchmark data-sets) it is however a considerably easier classification problem than either the Pima diabetes or the Breast Cancer data. This is apparent from the maximum achievable classification rate by the GP on the Ionosphere data of 95.8%, the highest among all six data-sets. In contrast the Pima diabetes data and the Breast Cancer data represent the hardest of the six classification problems with best classification rates of 76.5% and 73.9% respectively.

Ionosphere data

Figure 4.11 shows the performance of the GP active learner and random sampling on the Ionosphere data. It can be seen from figure 4.11(a) and (b) that by 9 queries the active learning rate consistently reaches 93% compared to passive learning which is still lingering at around 83% and takes on average 45 observations to reach the same classification rate. The inter-quartile range is particularly narrow for active learning on this data and the classification rate for active learning remains consistently higher than for passive learning right up to 50 observations. The upper and lower IQR for active learning is small at 50 observations and is comparable to the maximum classification rate (based on 200 training inputs) indicating that active learning relatively consistently finds a solution comparable to the top-line classification with only a quarter of the observations. Figure 4.11(c) shows that the AUC for active learning is again higher than for passive learning, with the lower quartile of active learning matching the upper quartile for passive learning at around 7 observations and active learning achieving close to the top-line with relative consistency by about 25 observations.

These consistently good results for the GP active learner are probably the result of relatively easy data-set. The achievable AUC for this problem which is 0.98 (close to perfect classification) indicates that the data is almost completely separable.

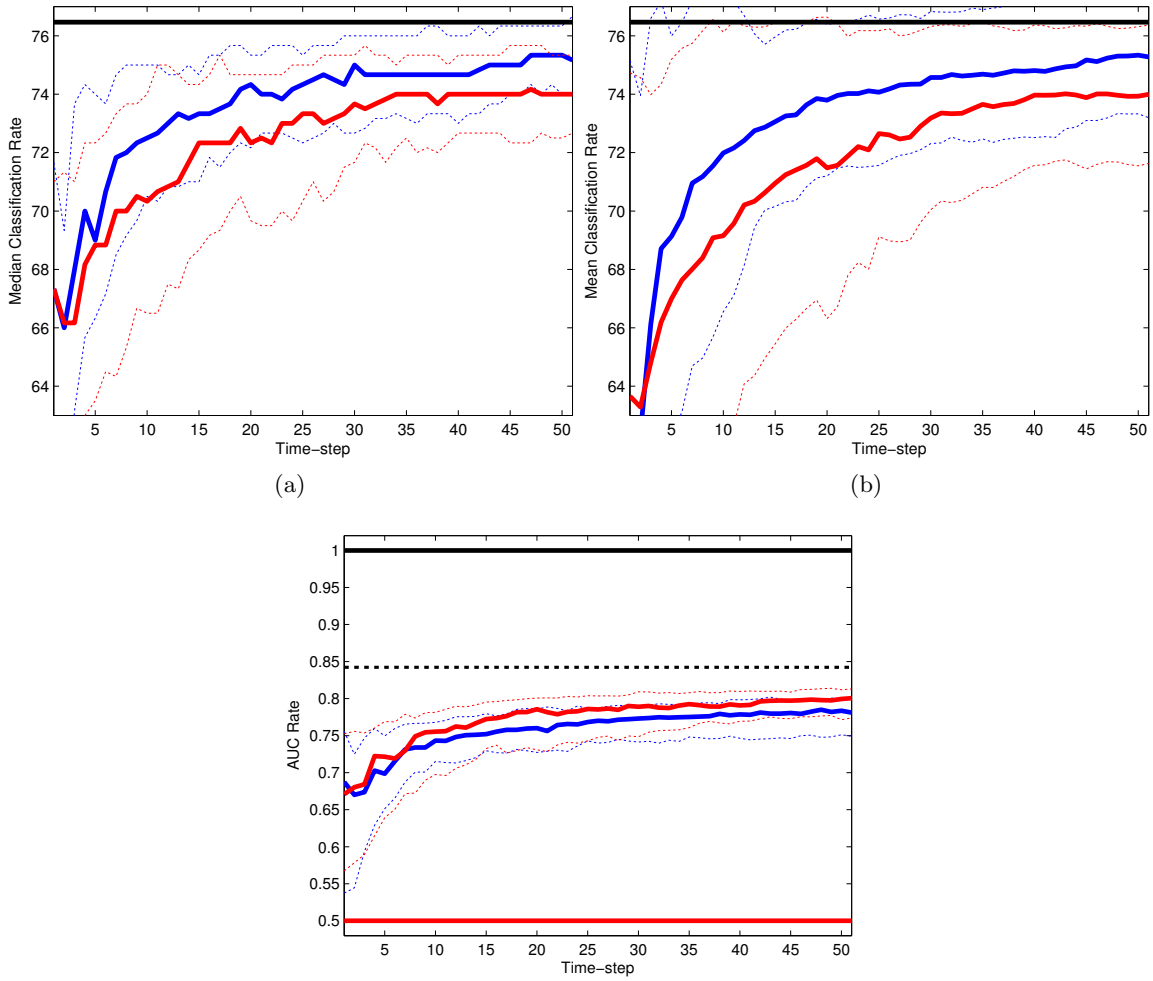


Figure 4.12.: Diabetes Data: Classification rates and the AUC for both active learning (blue lines) and random sampling (red lines) over 100 runs. (a) indicates the median classification rate over 100 runs, where the dotted lines give the upper and lower inter-quartile range. (b) indicates the mean classification rates over 100 runs, where the dotted lines indicate one standard deviation above and below the mean. The solid horizontal black line in (a) and (b) represents the maximum achievable classification rate given in table 4.1 (c) shows the median AUC achieved by the algorithm with dotted lines indicating the inter-quartile range. Here the solid horizontal black lines represents the AUC of a perfect classifier and the dotted horizontal black lines indicates the estimated maximum achievable AUC.

Pima diabetes data

In contrast, the Pima diabetes data we can see that the classification problem is considerably less separable with respect to the GP. The maximum classification rate achieved by the GP given all the 468 training inputs is 76.5%. Greater overlap between classes implies a higher frequency of 'misleading' observations which is reflected in figure 4.12(a) and

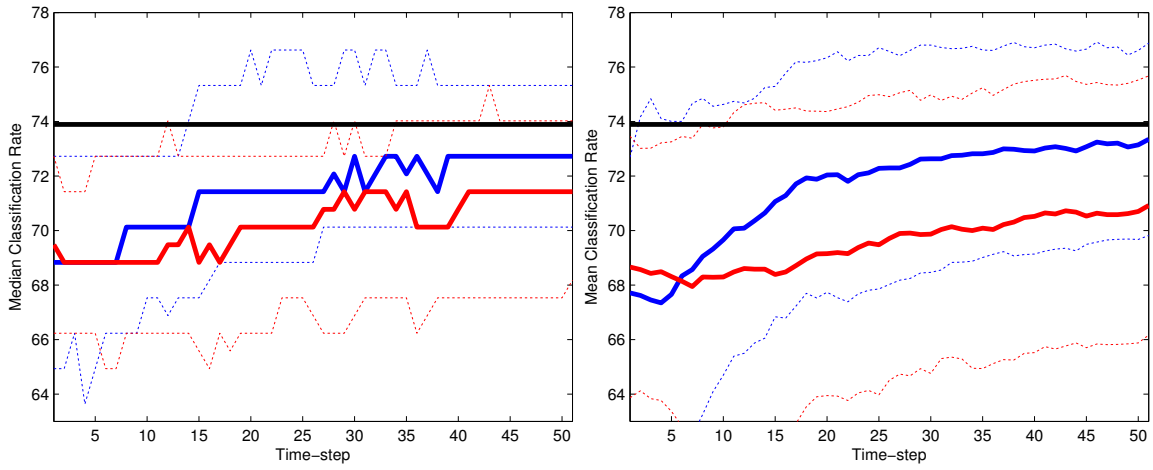
(b) which show considerable variation between runs for both active learning and passive learning. This is reflected in the high IQR and standard deviation for both active learning and passive learning. It is also demonstrated by the difference between the median and the mean in the early queries of the run. The mean is lower than the median indicating the presence of a proportion of runs where initially misleading observations considerably affect the performance in the initial $\approx 10 - 15$ queries and skew the mean downwards. Notice that queries later in the run do compensate for this and by 30 observations there is little difference between means and medians.

While the median and mean classification rates are higher for active learning than for passive learning for all time-steps after the first 2 observations the median for active learning sits within the IQR of passive learning. However, as shown in figure 4.12(a) the IQR for active learning is narrower than for passive learning indicating that performance (in terms of classification rate) is still slightly more consistent for active learning than for random sampling. In particular the lower quartile for active learning is close to the median for passive learning showing that only approximately a quarter of the time will active learning perform worse than the median for passive learning.

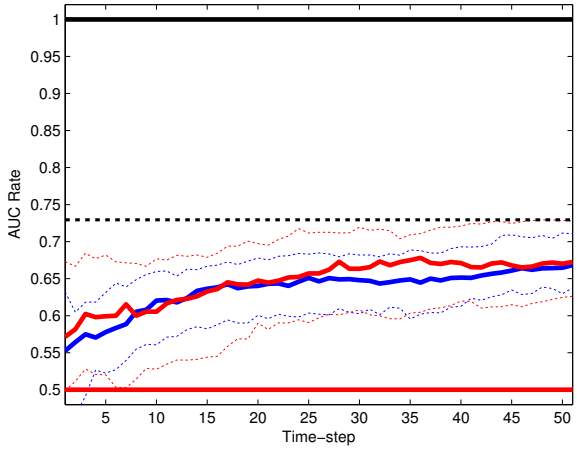
In contrast figure 4.12(c) shows that the AUC for random sampling marginally outperforms that of active learning, neither achieving particularly close to the maximum achievable for this data-set which is 0.84. This indicates that the Pima diabetes data is sensitive to changes in the classification threshold.

Breast Cancer data

For the Breast cancer data there are a total of 277 labelled instances in the benchmark data-set [Rätsch, 2011]. These are divided into standard 'folds' of the data. Each fold contains a training set of 200 observations and a test set of just 77 observations. Figures 4.13 (a)(b) and (c) show the mean, median classification rates and AUC averaged over 100 runs (10 runs on each of the first 10 folds). Unfortunately, because of the small number of examples in the test sets, some discretisation of the median and interquartile ranges is apparent in 4.13(a) caused by the relatively small number of possible classification rates available. The horizontal black line in figure 4.13 (a) and (b) represent the average



(a) (b)



(c)

Figure 4.13.: Breast Cancer Data: Classification rates and the AUC for both active learning (blue lines) and random sampling (red lines) over 100 runs. (a) indicates the median classification rate over 100 runs, where the dotted lines give the upper and lower inter-quartile range. (b) indicates the mean classification rates over 100 runs, where the dotted lines indicate one standard deviation above and below the mean. The solid horizontal black line in (a) and (b) represents the maximum achievable classification rate given in table 4.1 (c) shows the median AUC achieved by the algorithm with dotted lines indicating the inter-quartile range. Here the solid horizontal black lines represents the AUC of a perfect classifier and the dotted horizontal black lines indicates the estimated maximum achievable AUC.

classification rate attained over each of the 10 folds. The small test sets also results in a larger range of classification rates between folds (with a standard deviation of 3.6%) which explains why the upper quartile range in figure 4.13(a) ventures above 73.9% the rate achieved with all the training inputs.

Here, both active learning and passive learning lie well within the interquartile range of each other, however, as with the Pima diabetes data, the lower quartile range for the active learner is approximate to the median for passive learning, indicating that only approximately a quarter of the time will an active learning run fall below the median for passive learning. The difference in performance is more apparent in figure 4.13(b) where the mean rate is higher for the active learner for all time-steps after the 6th observation. However, it is difficult to say whether this is significant, given the large spread of both the interquartile range and the standard deviation. Looking at figure 4.13(c) it can again be seen that there is little difference between the active learning and passive learning when it comes to the AUC. Neither have achieved near the maximum achievable rate of 0.73 by 51 observations.

4.7. Performance measures for benchmark data-sets

The *deficiency* suggested by Baram et al. [2004] and the AUC performance measure from Guyon et al. [2010], which was used to compare active learning algorithms in the IJCNN 2010 active learning challenge, were described in section 2.2.6 and illustrated in figure 2.2. The deficiency is based on the mean classification rates achieved which are shown in panel (b) of figures 4.7-4.13. This is defined when the mean performance of the active learner is better than that of the passive learner. Let A be the area between the active learning rate (blue line) and the maximum achievable rate (black line). Let B be the area between the active learning rate (blue line) and the passive learning rate (red line), then the deficiency is given by $\frac{A}{A+B}$. The deficiency gives a score in the range $[0, 1)$ where 1 constitutes no improvement over passive learning and scores closer to 0 represent more 'efficient' active learning.

For the AUC performance measure the global score G is given by:

$$G = \sum_i \frac{ALC_i - \textsc{Arand}}{\textsc{Amax} - \textsc{Arand}} \tag{4.24}$$

Here ALC_i is the AUC active learning rate in the ith time-step shown by the blue line in panel (c) of figures 4.7-4.13. The solid horizontal black line and horizontal red line

represent AMAX (which is 1 for all time-steps) and ARAND (which is 0.5 for all time-steps) respectively. The AUC performance measure is therefore the proportion of area which the AUC line for active learning takes within the upper and lower bounds of the horizontal black (AMAX) and horizontal red lines (ARAND). In contrast to the deficiency, the best global score possible is 1 and the worst score is 0.

These measures aim to give an overall score of the performance of an active learning algorithm over all time-steps of the run. While these values are reported, the merit of these performance measures has to be considered with some caution. The specific criterion being optimised by the cost reduction approach as it is applied here is the expected misclassification cost for a given threshold λ. It is therefore unsurprising that the AUC rate for the active learner tends to perform worse than the mean classification rate which is specifically optimised by the algorithm. It is the assumption of the model that the true classification threshold λ is known. As a performance measure the ROC is only relevant when there is ambiguity about λ.

In contrast the deficiency does measure the algorithm according to a known threshold. In this sense it is more relevant when considering the performance of the GP active learning algorithm. However, there is some ambiguity as to what constitutes the maximum achievable classification rate. For the Ripley and New Ripley data good estimates of the Bayes optimal rates are used. This is the preferred situation when comparing between active learning classifiers, however for most data-sets the Bayes optimal rate is not known. For the remaining four data-sets the estimated maximum is based on the classification rate achieved by the GP when given the entire labelled training set (as opposed to one observation at a time for active learning). However for the Breast Cancer data in particular the standard deviation of the classification rate achieved by the GP between folds is particularly high 3.6%. Using the mean over the 10 folds (as we have done) is equivalent to measuring the deficiency for each of the 10 folds separately and in each case using the classification rate attained on the full training set as the top-line estimate. Adding to this the high standard deviation of the mean classification rates for both the Breast Cancer data and the Pima Diabetes data implies that the deficiency score should be treated with some caution for these data-sets.

data-set	IJCNN	Deficiency	CLR Act	CLR Pass	AUC Act	AUC Pass
Ripley	0.873	0.677	86.6	84.5	0.937	0.934
New Ripley	0.891	0.579	87.5	84.4	0.946	0.932
Banana	0.775	0.558	81.3	75.9	0.887	0.852
Ionosphere	0.940	0.435	92.3	87.8	0.970	0.954
Pima Diabetes		*0.708*	73.5	72.3	0.758	0.773
Breast Cancer		*0.687*	71.4	70.2	0.635	0.646

Table 4.2.: Columns 2 and 3 show the IJCNN performance measure which is based on AUC for ROC and the *Deficiency* which is based on the classification rates. CLR Act and CLR Pass show the area under the mean classification rate learning curve divided by the number of time-steps, for active and passive learning respectively. AUC Act and AUC Pass show the area under the AUC learning rate divided by the number of time-steps. The IJCNN performance measure for Pima diabetes and Breast Cancer are not given because the AUC for passive learning is higher than for the GP active learning.

It can be seen from Table 4.3 that by both performance measures the GP algorithm performs best on the Ionosphere data which is consistent with figure 4.11 where we can see that active learning performs considerably better than passive learning for both classification rate and AUC rates. The next best deficiency score among the six data-sets is 0.56 for the Banana data which is clear from figure 4.10(b) however the AUC score is lower than for either of the Ripley data-sets. This is because the deficiency measurement is based on the maximum achievable rate for the data-sets, while the AUC performance measure is not. Therefore the AUC score for the Banana data is affected by the lower achievable rates and shallower learning curves, when compared to either of the Ripley data-sets.

It is interesting to note that passive learning performs almost identically for the New Ripley data and the Ripley data both in terms of classification rate and AUC rate, indicating that the presence of one extra cluster in the new Ripley data makes little difference for passive learning. However, we can see that the deficiency for the new Ripley data is less than for the Ripley data. In this case it seems that active learning has gained some relative advantage over passive learning because of the cluster. This is because the GP active learning is more likely to query an unlabelled cluster where a class label is most informative about nearby unlabelled inputs.

4.8. Comparrison with other algorithms

In this section a comparison is made between the active Bayes, GP algorithm described in this chapter, which we call RASM, and three other algorithms. The first of the algorithms chosen for comparison is the SVM active learning algorithm SIMPLE [Schohn and Cohn, 2000, Campbell et al., 2000, Tong and Koller, 2000] described in section 2.2.2. At each step this algorithm selects to query the unlabelled observation which lies closest to the current decision boundary. This was implemented using the SVM-light package of [Schwaighofer, 2008].

The second algorithm is an SVM version of SELF-CONF [Roy and McCallum, 2001] which is described in section 2.2.4. This is an expected cost reduction approach to active learning which estimates the posterior entropy of class predictions for each potential query in the pool. This was applied using the active learning Matlab implementation of [Begleiter, 2005].

The third algorithm, which we call UNCT, is an uncertainty sampling approach which uses the same measure of uncertainty as Kapoor et al. [2007]'s paper on object categorisation (this paper was described in section 4.5), namely the absolute value of the mean divided by the square root of the total predicted variance. The GP implementation of UNCT uses the same EP algorithm, and the same covariance function (the sqared exponential covariance), as RASM. Unlike RASM there is no 'lookahead' and the query is selected which has the highest current level of uncertainty given the data.

4.8.1. Hyperparameters

The GP for both RASM and UNCT was implemented using the Rasmussen and Nickisch [2006] software. For all data-sets the squared exponential covariance function was used:

$$\mathbf{k}(\mathbf{z}_p, \mathbf{z}_q) = \nu \exp(-\frac{1}{2}(\mathbf{z}_p - \mathbf{z}_q)'\mathbf{L}^{-1}(\mathbf{z}_p - \mathbf{z}_q)). \tag{4.25}$$

where $\mathbf{L}$ is a diagonal matrix with diagonal elements $\{l_1, ..., l_D\}$ corresponding to length-scalse for each input dimension. Here ν represents the signal variance. The hyperparameters for the GP are the length-scales and the signal variance. On each data-set the

Data-set	c	γ	ClassRate
Breast Cancer	5	0.03	74%
Pima Diabetes	5	0.01	71%
Ionosphere	5.75	0.16	95.4%
Waveform	5.8	0.088	88%
USPS	5.25	0.03	95%

Table 4.3.: Hyperparameters for SVM on the five data-sets tested together with the corresponding classification rate when trained on the whole training set. c is the trade-off parameter between margin width and the number of misclassified observations. γ is the kernel width for the RBF kernel used.

hyperparameters were optimised on the whole training set using automatic relevance determination (ARD).

SIMPLE and SELF-CONF were implemented using an SVM. For all data-sets a radial basis function (RBF) kernel was applied. Here the hyperparameters are γ which defines the kernel width for the RBF and the error penalization parameter c which defines a trade-off between margin width and the number of misclassified observations. For each data-set hyperparameters were selected which maximised the classification rate of the SVM on the whole training set, these hyperparameters are shown in table 4.3.

4.8.2. Data-sets

Performance is compared on 5 data-sets, Ionosphere, Breast Cancer, Pima Diabetes, Waveform, and USPS. The first three (Ionosphere, Breast Cancer, Pima Diabetes) are previously described in section 4.6.1. The other two benchmark data sets Waveform and USPS are now briefly described.

Waveform data

The Waveform data set is a standard benchmark data set from the IDA Benchmark Repository [Rätsch, 2011]. This data set has 21 input dimensions with a training set of 400 labelled inputs and a test set of 4600 labelled inputs. For ease of comparison the data-set is given with 100 standard folds of the data. This data set was used by Tipping [2001] who reports classification rates of 10.3% for SVMs and 10.9% for RVMs.

USPS data

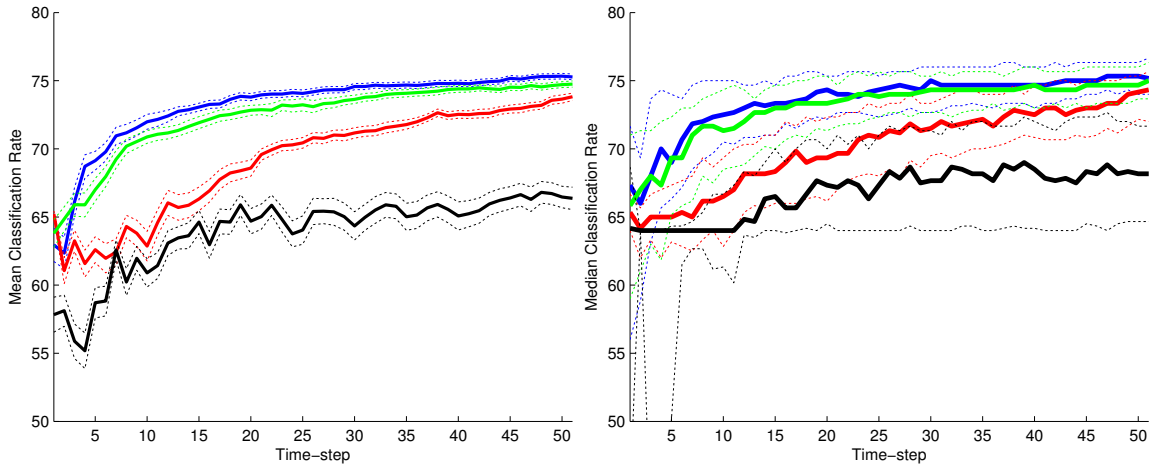
The original USPS data from the UCI machine learning repository [Frank and Asuncion, 2010] is multi-class, consisting of 16×16 greyscale images of handwritten characters for each of the digits from 0 to 9. Here we simply consider the binary classification task of distinguishing 3's from 5's. As described in chapter 3 of Rasmussen and Williams [2006] the original USPS data suffers from an imbalance whereby the test set and the training set were selected differently. Hence, the distribution of the original 7291 training cases and the distribution of the 2007 test cases are different. Following the Rasmussen and Williams [2006] partitioning of the data for the 3's v 5's classification task, a training set of 767 instances and test set of 773 are randomly drawn from the original UCI training and test sets.

4.8.3. Results

Figures 4.14 - 4.18 show comparison results over 100 runs of the data for each of the four algorithms, RASM (blue), UNCT (green), SIMPLE (red) and SELF-CONF (black). In each figure panel (a) shows the mean classification rate over all runs and panel (b) shows the median classification rate over all runs. In panel (a) the dotted lines represent one standard deviation for the error on the estimate of the mean. In panel (b) the dotted lines represent the upper and lower quartile range of the 100 runs. The results for the test-stets are:

Pima Diabetes

The Pima diabetes data as defined in section 4.6.1. The results for RASM (blue), UNCT (green), SIMPLE (red) and SELF-CONF (black) are shown in figure 4.14 and are as follows. For this data-set there is some difference between the classification rate achieved by the Gaussian process, when given all of the training data (76.5%) and the classification rate achieved by the SVM on all of the training data (71%). This in part accounts for the difference in performance between the GP algorithms (RASM and UNCT) compared to the SVM algorithms (SIMPLE and SELF-CONF) which is particularly noticeable in the early stages of learning, the first 15 to 20 observations. Here RASM which performs the full Bayesian lookahead performs only marginally better than UNCT, (the uncertainty



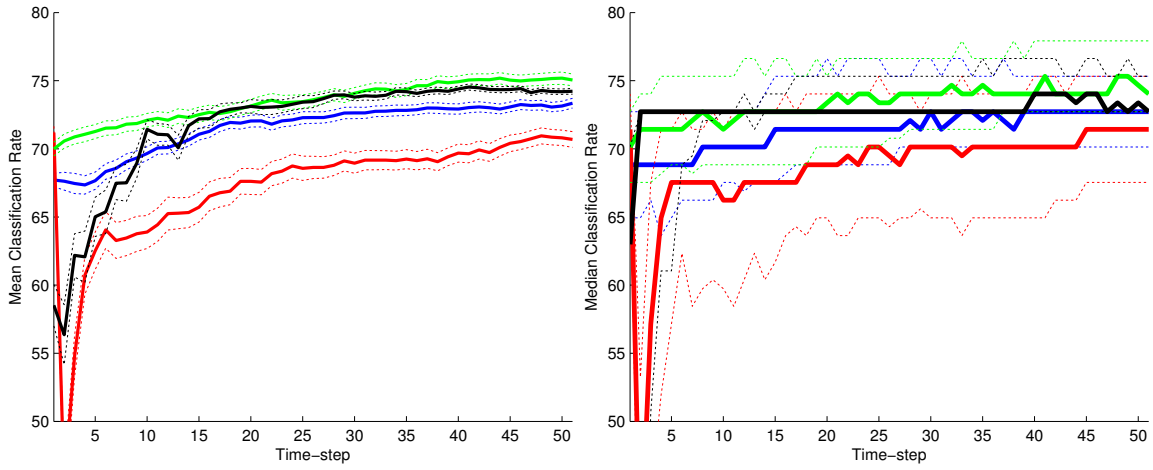
(a) (b)

Figure 4.14.: Diabetes data: Panel (a): mean classification rate over 100 runs of the data. Dotted lines represent one standard deviation of the estimate of the mean. Panel (b): median classification rate over 100 runs of the data. Here dotted lines represent the inter quartile range. The four algorithms compared are RASM (blue), UNCT (green), SIMPLE (red) and SELF-CONF (black)

sampling approach) and there is considerable overlap between the IQR of RASM and of UNCT. In contrast the SVM algorithms do not perform as well on this data set. SIMPLE makes steady progress and by 50 observations has attained a comparable performance to the GP algorithms. SELF-CONF performs badly on this data set reaching a plateux of about 65% classification rate, after about 20 observations.

Breast Cancer

The Breast cancer data as defined in section 4.6.1. The results for RASM (blue), UNCT (green), SIMPLE (red) and SELF-CONF (black) are shown in figure 4.15 and are as follows. For this data-set the classification rate achieved by the GP and the SVM on the whole training set were similar (both around 74%). SELF-CONF performs considerably better on the Breast Cancer data than on the other 4 benchmark data-sets. Starting from a lower initial classification rate that the Gaussian process it learns rapidly in the first 10 time-steps. From time-step 15 performance is comparable to UNCT. SIMPLE does not perform as well, apparently suffering from misleading observations in the first 3-4 observations. Here RASM performs slightly worse than both UNCT throughout and is overtaken by SELF-CONF after about 13 time-steps. There is considerable overlap between the interquartile range of UNCT, RASM and SELF-CONF with a dip in the



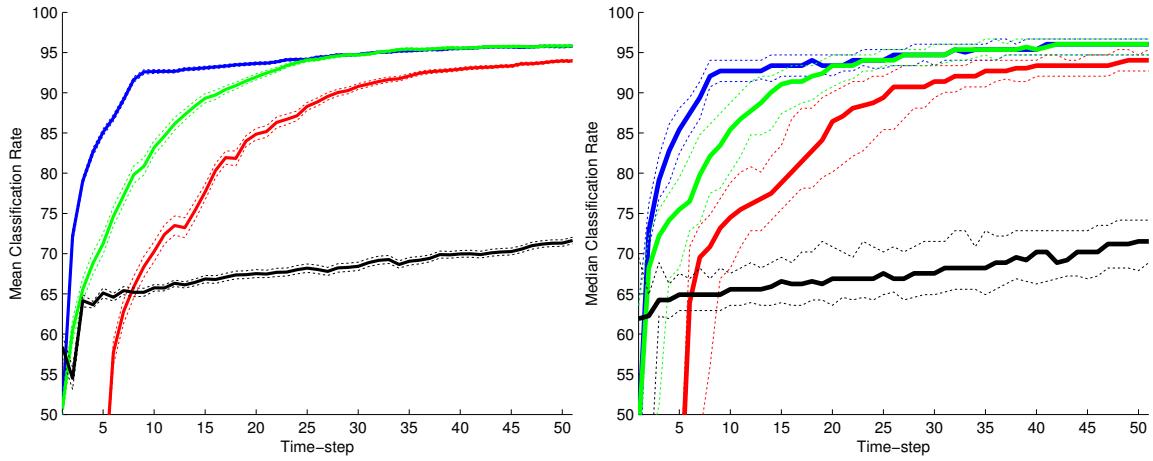
(a) (b)

Figure 4.15.: Breast Cancer data: Panel (a): mean classification rate over 100 runs of the data. Dotted lines represent one standard deviation of the estimate of the mean. Panel (b): median classification rate over 100 runs of the data. Here dotted lines represent the inter quartile range. The four algorithms compared are RASM (blue), UNCT (green), SIMPLE (red) and SELF-CONF (black)

lower quartile for SIMPLE. The upper quartile for SIMPLE is in the same range as the other three algorithms indicating that the lower average performance is due to a higher percentage of poorly performing runs.

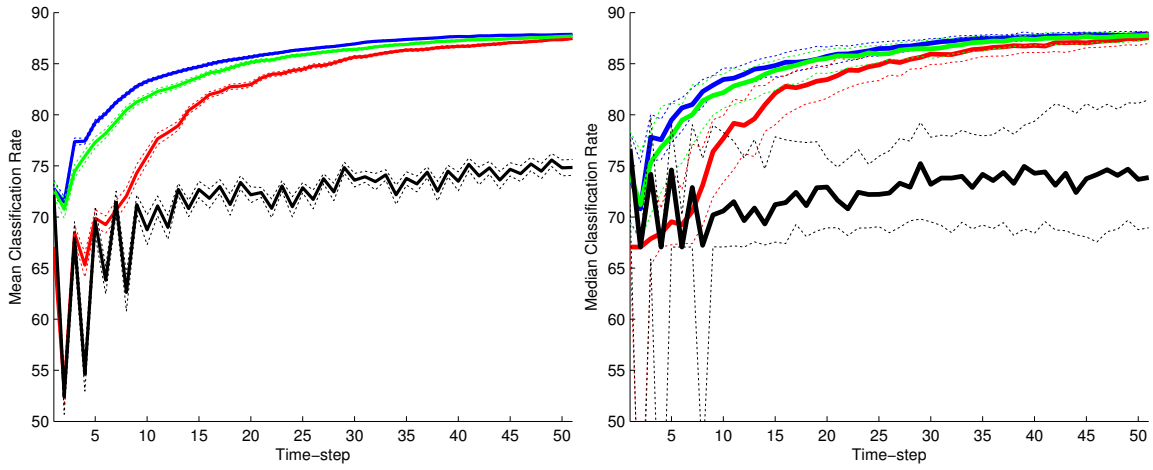
Ionosphere

The Ionosphere data as defined in section 4.6.1. The results for RASM (blue), UNCT (green), SIMPLE (red) and SELF-CONF (black) are shown in figure 4.16 and are as follows. As with the Breast cancer data the classification rate for the GP and the SVM on the whole training set are the same (for the Ionosphere data this is $\approx 95\%$). For this data-set RASM clearly out-performs the other 3 algorithms in the early stages of learning. It can be seen from panel (b) that the interquartile range for RASM is very narrow indicating a high level of consistency in performance between runs. By the 7 time-step RASM has consistently reached a classification rate above 90% and it is not until time-step 23 that UNCT eventually catches up with RASM. The interquartile range for UNCT and SIMPLE are less narrow than for RASM indicating a lower level of consistency between runs. After 6 time-steps SIMPLE exhibits a sharp increase median classification rate, however by time-step 50 it has still not caught up with either GP algorithm. SELF-CONF performs poorly on this data-set never reaching a classification rate above 70%.



(a) (b)

Figure 4.16.: Ionosphere data: Panel (a): mean classification rate over 100 runs of the data. Dotted lines represent one standard deviation of the estimate of the mean. Panel (b): median classification rate over 100 runs of the data. Here dotted lines represent the inter quartile range. The four algorithms compared are RASM (blue), UNCT (green), SIMPLE (red) and SELF-CONF (black)

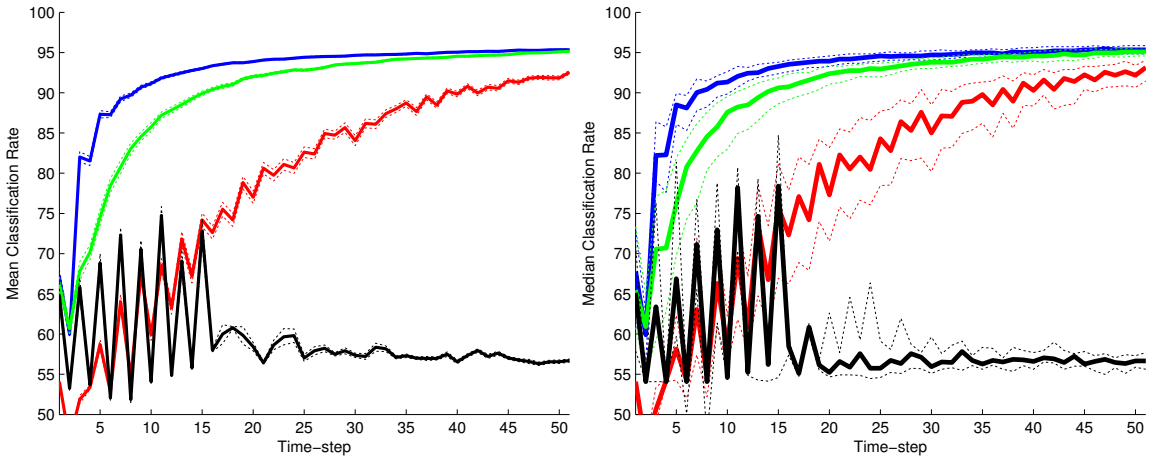


(a) (b)

Figure 4.17.: Waveform data: Panel (a): mean classification rate over 100 runs of the data. Dotted lines represent one standard deviation of the estimate of the mean. Panel (b): median classification rate over 100 runs of the data. Here dotted lines represent the inter quartile range. The four algorithms compared are RASM (blue), UNCT (green), SIMPLE (red) and SELF-CONF (black)

Waveform

The Waveform data as defined in section 4.8.2. The results for RASM (blue), UNCT (green), SIMPLE (red) and SELF-CONF (black) are shown in figure 4.17 and are as follows. Again the classification rate on the whole training set is comparable between



(a) (b)

Figure 4.18.: USPS data: Panel (a): mean classification rate over 100 runs of the data. Dotted lines represent one standard deviation of the estimate of the mean. Panel (b): median classification rate over 100 runs of the data. Here dotted lines represent the inter quartile range. The four algorithms compared are RASM (blue), UNCT (green), SIMPLE (red) and SELF-CONF (black)

the SVM and the GP, both achieving a classification rate of $\approx 80\%$. RASM and UNCT clearly outperform SIMPLE on this data-set and again SELF-CONF performs badly. The interquartile range (panel (b)) and the standard deviation on the estimate of the mean (panel (a)) are particularly narrow for RASM and UNCT indicating a high level of consistency between runs. UNCT performs comparably to RASM on this data set with a marginally higher mean classification rate for RASM in the early stages. SIMPLE starts off worse but gradually catches up with the GP algorithms by time-step 50. Again SELF-CONF performs badly exhibiting a fairly serious plateaux which never reaches above a 75% classification rate. For this data-set SELF-CONF exhibits a fairly serious osciliatory effect, particularly in the first 10 time-steps. This implies that the margin of the SVM is over compensating for each new observation causing the decision boundary to leap back with each new observations.

USPS

The USPS data as defined in section 4.8.2. The results for RASM (blue), UNCT (green), SIMPLE (red) and SELF-CONF (black) are shown in figure 4.18 and are as follows. Both the SVM and GP achieve $\approx 95\%$ classification rate on the whole training set. Here the GP active learning algorithms significantly outperform SIMPLE and SELF-CONF. RASM

has a narrow interquartile range and there is some separation between RASM and UNCT in the first 20 time-steps, with UNCT lying outside the interquartile range of RASM. SIMPLE does not perform as well as either of the GP algorithms appearing to learn at a slower rate in the early stages. SELF-CONF performs particularly poorly on this data-set oscillating wildly in the first 16 time-steps before settling on a very low classification rate ($\approx 57\%$). SIMPLE also exhibits the same oscillatory effect but to a lesser extent, with classification rate gradually improving over time.

Conclusion of comparison

In all 5 benchmark sets used for comparison RAMS and UNCT were seen to outperform SIMPLE, with RASM significantly outperforming UNCT in the early stages for the Ionosphere data and the USPS data. SELF-CONF performed in general worse than the other three algorithms except for the Breast cancer data-set where it outperforms both SIMPLE and RASM. On the Waveform and Ionosphere data-sets SELF-CONF exhibits serious oscillation between time-steps, to a lesser extend this oscillation effect is also apparent for SIMPLE on the USPS data.

RASM therefore demonstrates a generally consistent level of performance when compared to UNCT, SIMPLE and SELF-CONF. Even for the Breast cancer data, the only data set where it did not perform equal or better to the other algorithms RASM still does not perform considerably worse. However, compared to all three other algorithms, RASM takes considerably more time to run. For example our implementation on Matlab took 18 hours and 33 minutes to perform a 100 runs on the Waveform data set, this equates to approximately 12 minutes for a single run, compared to approximately 6 seconds for a single run of UNCT. SIMPLE which runs on SVM-light and calls mex files runs even faster taking approx 1.8 seconds for a single run of the Waveform data.

The conclusion of this comparison therefore is that, there is in general some improvement in performance from using RASM, and this improvement was significant on both the Ionoshpere data and the USPS data, however this performance is to be balanced with a considerably higher computation cost. For many applications UNCT appears to be a reasonable compromise between computation speed and performance, however when the

labelling cost is high compared to computation cost RASM should be used.

4.9. Summary

The objective of this chapter was to simplify the query density framework presented in chapter 3 to the pool-based setting for active learning. In section 4.2 we started by examining what happens to the model when the query density collapses to a Dirac delta function and single point queries are made. This was further simplified to the pool-based setting in section 4.3. The expected cost reduction approach to active learning in the pool-based setting was then presented in section 4.4. This scheme allows for arbitrary classifiers with a probabilistic output. We then implemented this approach using GP classification with the expectation propagation approximation. The GP and the expectation propagation algorithm are described in section 4.5. This is a non-parametric Bayesian approach to classification. Finally, in section 4.6 the GP classifier is demonstrated for six benchmark data-sets. The results demonstrate the Bayesian expected cost reduction approach in the pool-based setting for active learning with improvement over passive learning on all six data-sets and show that this approach is feasible for real data.

5. Discussion

The focus of this thesis has been the cost reduction approach to active learning. This has been advanced in two directions. In chapter 3 we presented a Bayesian cost reduction framework which is able to sample new input queries from arbitrary query densities. The Bayesian update for this model incorporates both the query and how the query was sampled. This is a very flexible model which, by adjusting the type of query density used, can encompass both passive learning and different forms of active learning.

In chapter 4 it was shown how active learning in the pool-based setting can be viewed as a simplification of the query density framework. The cost reduction approach to active learning was then applied to Gaussian processes using expectation propagation to approximate the posterior distributions. This implementation is more computationally feasible than the full query density framework and is applicable to data sets of a reasonable size. Our results in both chapters indicate active learning rates which outperform passive learning.

5.1. Query density framework

As stated in the discussion of chapter 3, the query density framework has the potential to inform the debate on when active learning or passive learning should be preferred. We have shown for example, that there exist problems where the optimal query for a learner may be to sample broadly within some region of input space, rather than at a single input, or across the entire input space.

Currently, the query density model is principally theoretical in nature. The search space across potential query densities is large, even compared to the search space for

potential input queries and the integrations required to implement the framework are currently computationally too expensive for all but the simplest of problems. The focus of this thesis has been to clarify the structure which such an approach would take and to demonstrate its implementation on a simple problem.

We now examine several different directions in which the framework could potentially be made more feasible.

The first approach to improving the feasibility of the framework is to reduce the number of potential queries which need to be evaluated for the learner to select a query. The implementation of the framework on the probabilistic high-low game performs a brute force search across the entire space of potential query densities. This is required in order to plot figures of the expected cost predictions for each potential query, but may not be necessary to find the query with minimum expected cost. In general any optimisation procedure could be used to search for the query with minimum expected cost while minimising the number of potential queries on which the expected cost must be evaluated.

Secondly, it is possible to simplify the framework by using a point estimate such as maximum likelihood or MAP, in parameter space before performing the lookahead. The current model draws samples from parameter space before drawing hypothetical observations from the model. This approach averages over parameter uncertainty when performing the lookahead. An estimate of the expected cost could however be attained by taking the current MAP estimate of the parameters, drawing hypothetical observations only from the most likely current model.

Thirdly, a more practical implementation of the framework would require faster ways to perform the Bayesian update. The natural simplification would be to estimate the posterior distribution by some known parametrised distribution, or combination of distributions. Unfortunately because of the presence of the query density it is unlikely that a variational approach would work. However, it is potentially feasible to apply expectation maximisation to optimise the parameters of the posterior distribution.

5.2. Pool based setting

In chapter 5 the cost reduction approach to active learning was presented in the pool-based setting. There has been previous work in this direction: Roy and McCallum [2001], Guo and Greiner [2007] and Zhu et al. [2003] have implemented the expected cost reduction approach with various classifiers, however Zhu et al.'s implementation is the only Bayesian approach which does not make an IID assumption on how the data is selected. The work of Zhu et al. is couched in the language of Gaussian random fields but is equivalent to performing least squares classification with Gaussian processes. It is therefore similar to our implementation. The main difference between expectation propagation and least squares classification is the way in which the data is interpreted. Least squares classification assumes the target labels $y \in \{-1, +1\}$ are continuous observations with Gaussian noise. For least squares classification probability predictions are simply rescaled regression estimates for y. The advantage of the *expectation propagation* approach which we implement using the Gaussian processes software of Rasmussen and Nickisch [2006] is that it models y as a binary class label, approximating the actual probability of class membership.

The expected cost reduction approach to active learning is one of the most computationally intensive approaches to active learning because it evaluates the expected risk associated with each potential query before the learner makes each query. Using expectation propagation scales as $\mathcal{O}(m^3)$, where m is the total number of inputs in the pool compared to $\mathcal{O}(m^2)$ for least squares classification. Depending on the dimension of the inputs, the current Matlab implementation is therefore limited to a pool size of around 5000 inputs. This limitation is somewhat moderated by the observation that each input in the pool is evaluated independently and therefore the implementation can easily be parallelised. However, there are many practical problems of a reasonable size to which this algorithm could be applied.

The time taken to construct the necessary covariance matrices scales linearly with the dimension of the inputs. However the covariances between inputs only needs to be evaluated once and does not affect the time required to perform the lookahead. It is therefore feasible to implement the algorithm with relatively high dimensional inputs. If the dimen-

sion of the inputs is large then it would possible to perform preprocessing on the data. For example principal component analysis (PCA) [e.g. Jolliffe, 2002] might be applied to reduce the dimensionality of the inputs to a manageable dimension which is however large enough to preserve the decision boundary for classification.

When the pool size is larger than 5000, in its current form, the expectation propagation algorithm implementation of expected cost reduction active learning would require simplification, such as sampling. For example when using a radial basis kernel, observations tend to have a local affect on the decision boundary affecting the probability of class membership for inputs close to the observation more than for inputs which are distant to the observation. The lookahead could therefore be implemented more efficiently by only considering training data and test data local to the potential query. A precise scheme for this implementation would require some decision on which inputs are considered to be close enough to the potential query to be included. It seems reasonable for this decision to be based on the values in the covariance matrix for the given kernel.

5.3. Closing remarks

In conclusion we have presented a query density framework for the expected cost reduction approach to active learning and have also applied the cost reduction approach to active learning in the pool-based setting. These approaches do not require any separability assumption on the data and are flexible enough to incorporate any efficient probabilistic classifier. While this approach is 'optimal' in the sense that it seeks to optimise the objective function on which the learner is evaluated, the expected cost reduction approach is also one of the most computationally intensive approaches to active learning. As has been discussed here, it is an area of future work to find computationally efficient algorithms for the expected cost reduction approach, without compromising on the probabilistic validity of the model.

A. Misclassification costs for the Pie Example

Here we find the expected misclassification cost for the pie example, where there are N slices and the relative density of $p(x)$ is M times higher in the only class 1 slice than in the class 0 slices. λ_{01}, λ_{10} are known and the decision threshold is defined by $\lambda = \lambda_{01}/(\lambda_{01} + \lambda_{10})$ and $(\mathbf{x}_q, y_q)$ are the input and label pair returned by the query.

We evaluate the misclassification cost for two cases, depending on the relative density M. This is because the relative density will effect the output of the decision function in the case where $y_q = 0$. Let $\tau = \frac{\lambda}{(1-\lambda)}(N-2)$ be the threshold at which the decision function changes output. For simplicity we first evaluate the case where $M > \tau$ and the decision function returns a class 1 label for unlabelled slices, before considering the case when the decision function returns a class 0 label and $M \leq \tau$.

For both cases $M > \tau$ and $M \leq \tau$ the result is consistent with the argument given in section 3.5.2 where a broad query is best for any $M > 1$ and any λ not equal to 0 or 1.

A.0.1. Single point query

We start by observing that for any single point query $\mathbf{x}_q$, which by definition will lie in at most one slice, it is immediately apparent that $p(y_q = 1) = \frac{1}{N}$ and $P(y_q = 0) = \frac{N-1}{N}$. The aim is to evaluate the expected risk $R(\mathbf{x}_q)$. Notice that this risk depends on how $\mathbf{x}_q$ was sampled and also on whether $M > \tau$ or $M \leq \tau$. We initially follow the argument with these dependencies suppressed (for notational simplicity) in order to find the expected misclassification cost for a point query when $M > \tau$. Only in equation (A.7) do we restore the dependency and write $R(\mathbf{x}_q \,|\, \mathbf{x}_q \sim \delta_{\mathbf{x}_q}, M > \tau)$.

We start by evaluating $R(\mathbf{x} \,|\, y_q = 1, \mathbf{x}_q)$ and $R(\mathbf{x} \,|\, y_q = 0, \mathbf{x}_q)$, the expected risk associated with an arbitrary input $\mathbf{x}$ given class labels 1 and 0 respectively for y_q. We then find the expectation over $\mathbf{x}$ to find the overall risks associated with a query-label pair $(\mathbf{x}_q, y_q)$; these are $R(y_q = 1, \mathbf{x}_q)$ and $R(y_q = 0, \mathbf{x}_q)$ when $y_q = 1$ and $y_q = 0$ respectively.

When $y_q = 1$ the only class 1 slice has been identified and all other class labels can be inferred. Because the problem is separable the misclassification cost is therefore zero $R(\mathbf{x} \,|\, y_q = 1, \mathbf{x}_q) = 0$ for all $\mathbf{x}$, which implies that $R(y_q = 1, \mathbf{x}_q) = 0$. However when $y_q = 0$ no information about the remaining $N - 1$ slices can be inferred, and:

$$p(\mathbf{y} = 1 \,|\, \mathbf{x}, y_q = 0, \mathbf{x}_q) = \begin{cases} 0 & \text{if } \mathbf{x} \in S_{\mathbf{x}_q} \\ \frac{M}{M+N-2} & \text{otherwise.} \end{cases} \tag{A.1}$$

Here, $S_{\mathbf{x}_q}$ refers to the slice which was queried by $\mathbf{x}_q$. The M appears in (A.1) because of the difference in relative density between the two classes. Having found the posterior target predictions the algorithm assigns class label predictions so as to minimise the expected misclassification cost. It makes these assignments via the decision function which is defined as:

$$\mathcal{A}(\mathbf{x} \,|\, \mathbf{x}_q) = \begin{cases} 1 & \text{if } p(\mathbf{y} = 1 \,|\, \mathbf{x}, y_q = 0, \mathbf{x}_q) \geq \lambda \\ 0 & \text{otherwise.} \end{cases} \tag{A.2}$$

As described above we evaluate the case where $M > \frac{\lambda}{(1-\lambda)}(N-2)$ and $\mathcal{A}(\mathbf{x} \notin S_{\mathbf{x}_q} \,|\, \mathbf{x}_q) = 1$. In the slice which is already labelled as being in class 0 it follows that $\mathcal{A}(\mathbf{x} \in S_{\mathbf{x}_q} \,|\, \mathbf{x}_q) = 0$.

To evaluate $R(y_q = 0, \mathbf{x}_q)$ we consider the case when $\mathbf{x} \in S_{\mathbf{x}_q}$ and $\mathbf{x} \notin S_{\mathbf{x}_q}$ separately, using the definition:

$$\begin{aligned} R(y_q = 0, \mathbf{x}_q) = {} & p(\mathbf{x} \in S_{\mathbf{x}_q} \,|\, y_q = 0, \mathbf{x}_q) R(\mathbf{x} \in S_{\mathbf{x}_q} \,|\, y_q = 0, \mathbf{x}_q) \\ & + p(\mathbf{x} \notin S_{\mathbf{x}_q} \,|\, y_q = 0, \mathbf{x}_q) R(\mathbf{x} \notin S_{\mathbf{x}_q} \,|\, y_q = 0, \mathbf{x}_q) \end{aligned} \tag{A.3}$$

to combine the results. In the queried slice the class label is known and $R(\mathbf{x} \in S_{\mathbf{x}_q} \,|\, y_q = 0, \mathbf{x}_q) = 0$. In the unlabelled slices the decision function returns a class

label of 1 and therefore:

$$R(\mathbf{x} \notin S_{\mathbf{x}_q} \,|\, y_q = 0, \mathbf{x}_q) = p(y = 0 \,|\, \mathbf{x} \notin S_{\mathbf{x}_q}, y_q = 0, \mathbf{x}_q)\lambda_{01} = \frac{\lambda_{01}(N-2)}{M+N-2} \tag{A.4}$$

Where $p(y = 0 \,|\, \mathbf{x} \notin S_{\mathbf{x}_q}, y_q = 0, \mathbf{x}_q)$ is found from equation (A.1). Observe that the probability that $\mathbf{x}$ is in the labelled slice is $p(\mathbf{x} \in S_{\mathbf{x}_q} \,|\, y_q = 0, \mathbf{x}_q) = \frac{1}{M+N-1}$, and for the unlabelled slice is $p(\mathbf{x} \notin S_{\mathbf{x}_q} \,|\, y_q = 0, \mathbf{x}_q) = \frac{M+N-2}{M+N-1}$. Substituting these in equation (A.3) gives:

$$R(y_q = 0, \mathbf{x}_q) = \frac{\lambda_{01}(N-2)(M+N-2)}{(M+N-1)(M+N-2)} = \frac{\lambda_{01}(N-2)}{(M+N-1)}. \tag{A.5}$$

This is combined with the observation above that $R(y_q = 1, \mathbf{x}_q) = 0$ by using:

$$R(\mathbf{x}_q) = p(y_q = 0 \,|\, \mathbf{x}_q)R(y_q = 0, \mathbf{x}_q) + p(y_q = 1 \,|\, \mathbf{x}_q)R(y_q = 0, \mathbf{x}_q) \tag{A.6}$$

and noting that $p(y_q = 0 \,|\, \mathbf{x}_q)$ in the case of a point query is simply $\frac{N-1}{N}$. Putting these together we have:

$$R(\mathbf{x}_q \,|\, \mathbf{x}_q \sim \delta_{\mathbf{x}_q}, M > \tau) = \frac{\lambda_{01}(N-1)(N-2)}{N(M+N-1)} \tag{A.7}$$

The overall expected misclassification cost for a single point query when $M > \frac{\lambda}{(1-\lambda)}(N-2)$.

A.0.2. Random query

The only difference in the case of a random query where $\mathbf{x}_q \sim p(\mathbf{x})$ is that,
$p(y_q = 0 \,|\, \mathbf{x}_q) = \frac{N-1}{M+N-1}$ which gives:

$$R(\mathbf{x}_q \,|\, \mathbf{x}_q \sim p(\mathbf{x}), M > \tau) = \frac{\lambda_{01}(N-1)(N-2)}{(M+N-1)^2} \tag{A.8}$$

This is because whether $\mathbf{x}_q \sim p(\mathbf{x})$ or is a single point query will not effect the learners posterior knowledge about the remaining unlabelled slices. Note however, for example that for a broad query which is uniform over $N-3$ slices, there would be a different information set for unlabelled slices which formed part of the query and those that did not.

We now find the expected misclassification cost when $M \leq \frac{\lambda}{(1-\lambda)}(N-2)$, following the same approach. In this case equation (A.2) gives $\mathcal{A}(\mathbf{x} \notin S_{\mathbf{x}_q} \,|\, \mathbf{x}_q) = 0$ and $\mathcal{A}(\mathbf{x} \in S_{\mathbf{x}_q} \,|\, \mathbf{x}_q) = 1$. From equation (A.3) we have:

$$R(\mathbf{x} \notin S_{\mathbf{x}_q} \,|\, y_q = 0, \mathbf{x}_q) = p(y = 1 \,|\, \mathbf{x} \notin S_{\mathbf{x}_q}, y_q = 0, \mathbf{x}_q)\lambda_{10} = \frac{M\lambda_{10}}{(M+N-2)} \tag{A.9}$$

as above the misclassification risk in the labelled slice is $R(\mathbf{x} \in S_{\mathbf{x}_q} \,|\, y_q = 0, \mathbf{x}_q) = 0$ and $p(\mathbf{x} \notin S_{\mathbf{x}_q} \,|\, y_q = 0, \mathbf{x}_q) = \frac{M+N-2}{M+N-1}$. These are combined using equation (A.3) to give:

$$R(y_q = 0, \mathbf{x}_q) = \left(\frac{M\lambda_{10}}{(M+N-2)}\right)\left(\frac{M+N-2}{M+N-1}\right) = \frac{\lambda_{10}M}{M+N-1} \tag{A.10}$$

Equation (A.6) is applied in a similar manner to above (because $p(y_q \,|\, \mathbf{x}_q)$) remains the same) and we have:

$$R(\mathbf{x}_q \,|\, \mathbf{x}_q \sim \delta_{\mathbf{x}_q}, M \leq \tau) = \left(\frac{\lambda_{10}M}{M+N-1}\right)\left(\frac{N-1}{N}\right) \tag{A.11}$$

For a point query and:

$$R(\mathbf{x}_q \,|\, \mathbf{x}_q \sim p(\mathbf{x}), M \leq \tau) = \left(\frac{\lambda_{10}M}{M+N-1}\right)\left(\frac{N-1}{M+N-1}\right) \tag{A.12}$$

for a broad sample $\mathbf{x}_q \sim p(\mathbf{x})$. As we can see in both cases a broad query is better when $M > 1$.

B. Parameter posterior $p(\alpha, \beta \mid x_1, \phi)$ for a uniform prior in the probabilistic high-low game

For the probabilistic high-low game as defined in section 3.3 we evaluate the posterior parameter distribution $p(\alpha, \beta \mid x_1)$ for a single *unlabelled* input x_1 and a uniform prior $p(\alpha, \beta)$, with the constraint that $\beta \leq \alpha$. As with equation 3.17 the likelihood of x for particular parameters α, β is given by:

$$p(x|\alpha, \beta) = \frac{w_0}{\alpha}\mathcal{I}_{[0,\alpha)}(x) + \frac{w_1}{1-\beta}\mathcal{I}_{[\beta,1)}(x), \tag{B.1}$$

where $\mathcal{I}_S(x)$ is the indicator function equal to 1 when $x \in S$ and 0 otherwise. When the query density is uniform on the interval $[q_-, q_+)$, and is given by $q(x; \phi) = \frac{\mathcal{I}_{(q_-,q_+)}(x)}{|q_+ - q_-|}$ (for query parameters $\phi := \{q_-, q_+\}$), then the denominator, $|q_+ - q_-|$, cancels and the sample oracle becomes:

$$Q(x|\alpha, \beta, \phi) = \frac{\mathcal{I}_{(q_-,q_+)}(x)\, p(x|\alpha, \beta)}{\int_0^1 \mathcal{I}_{(q_-,q_+)}(x) p(x|\alpha, \beta)\, dx}. \tag{B.2}$$

The posterior distribution for the parameters given the observation of one unlabelled input x_1 is:

$$p(\alpha, \beta|x_0, q) = \frac{Q(x|\alpha, \beta, \phi)\, p(\alpha, \beta)}{\int_0^1 \int_0^1 Q(x|\alpha, \beta, \phi) p(\alpha, \beta)\, d\alpha d\beta} \; \forall \alpha,\beta \text{ s.t. } \beta \leq \alpha, \tag{B.3}$$

and for a uniform prior $p(\alpha, \beta)$, with the constraint that $\beta \leq \alpha$:

$$p(\alpha, \beta) = \begin{cases} 2 \text{ if } \alpha \leq \beta \\ 0 \text{ if } \alpha > \beta \end{cases} \tag{B.4}$$

equation B.3 reduces to:

$$p(\alpha, \beta | x_1, \phi) = \frac{Q(x|\alpha, \beta, \phi)}{2 \int_0^1 \int_0^\alpha Q(x|\alpha, \beta, \phi) d\alpha\beta}. \tag{B.5}$$

Therefore, the posterior parameter distribution $p(\alpha, \beta | x_1, q) \propto Q(x|\alpha, \beta, \phi)$ and it is enough to evaluate equation (B.2) in order to have an un-normalised analytic form for the posterior distribution. Given equation (B.1) the denominator in equation (B.2) is:

$$\int_0^1 \mathcal{I}_{[q_-, q_+)} \Big\{ \frac{w_0}{\alpha} \mathcal{I}_{[0,\alpha)}(x) + \frac{w_1}{1-\beta} \mathcal{I}_{[\beta,1)}(x) \Big\} dx. \tag{B.6}$$

We integrate the first term of this by noting that $\mathcal{I}_{[q_-,q_+)} \mathcal{I}_{[0,\alpha)} = \mathcal{I}_{[q_-,q_+) \cap [0,\alpha)} = \mathcal{I}_{[q_-, min(q_+,\alpha))}$ so that when $\alpha \leq q_-$ this equals zero, and when $\alpha > q_-$ the first term of equation (B.6) becomes:

$$\int_{q_-}^{min(q_+,\alpha)} \frac{w_0}{\alpha} dx = \frac{w_0}{\alpha} (min(q_+, \alpha) - q_-). \tag{B.7}$$

Similarly we integrate the second term by noting that $\mathcal{I}_{[q_-,q_+)} \mathcal{I}_{[\beta,1)} = \mathcal{I}_{[max(q_-,\beta),q_+)}$ so when $\beta \geq q_+$ this equals zero and when $\beta < q_+$ the second term of equation (B.6) becomes:

$$\int_{max(q_-,\beta)}^{q_+} \frac{w_1}{1-\beta} dx = \frac{w_1}{1-\beta} (q_+ - max(q_-, \beta)). \tag{B.8}$$

Putting these together in equation (B.6) is equal to:

$$\mathcal{I}_{[q_-,1)}(\alpha) \frac{w_0 (min(q_+, \alpha) - q_-)}{\alpha} + \mathcal{I}_{[0,q_+)}(\beta) \frac{w_1 (q_+ - max(q_-, \beta))}{1-\beta} \tag{B.9}$$

We can substitute this back into equation (B.5) and when the prior $p(\alpha, \beta)$ is uniform,

	$0<\alpha<q_-$ $\mathcal{I}_{[q_-,1)}(\alpha)=0$	$q_-<\alpha<x_1$ $\mathcal{I}_{[q_-,1)}(\alpha)=1$	$x_1<\alpha<q_+$ $\mathcal{I}_{[q_-,1)}(\alpha)=1$	$q_+<\alpha<1$ $\mathcal{I}_{[q_-,1)}(\alpha)=1$
$0<\beta<q_-$ $\mathcal{I}_{[q_-,1)}(\beta)=1$	$\frac{1}{(q_+-q_-)}$	$\frac{\frac{w_1}{(1-\beta)}}{\frac{w_0(\alpha-q_-)}{(\alpha)}+\frac{w_1(q_+-q_-)}{(1-\beta)}}$	$\frac{\frac{w_0}{(\alpha)}+\frac{w_1}{(1-\beta)}}{\frac{w_0(\alpha-q_-)}{\alpha}+\frac{w_1(q_+-q_-)}{(1-q_+)}}$	$\frac{\frac{w_0}{\alpha}+\frac{w_1}{(1-\beta)}}{\frac{w_0(q_+-q_-)}{\alpha}+\frac{w_1(q_+-q_-)}{(1-q_+)}}$
$q_-<\beta<x_1$ $\mathcal{I}_{[q_-,1)}(\beta)=1$	0	$\frac{\frac{w_1}{(1-\beta)}}{\frac{w_0(\alpha-q_-)}{\alpha}+\frac{w_1(q_+-\beta)}{(1-\beta)}}$	$\frac{\frac{w_0}{\alpha}+\frac{w_1}{(1-\beta)}}{\frac{w_0(\alpha-q_-)}{\alpha}+\frac{w_1(q_+-\beta)}{(1-\beta)}}$	$\frac{\frac{w_0}{\alpha}+\frac{w_1}{(1-\beta)}}{\frac{w_0(q_+-q_-)}{\alpha}+\frac{w_1(q_+-\beta)}{(1-\beta)}}$
$x_1<\beta<q_+$ $\mathcal{I}_{[q_-,1)}(\beta)=1$	0	0	$\frac{\frac{w_0}{\alpha}}{\frac{w_0(\alpha-q_-)}{\alpha}+\frac{w_1(q_+-\beta)}{(1-\beta)}}$	$\frac{\frac{w_0}{\alpha}}{\frac{w_0(q_+-q_-)}{\alpha}+\frac{w_1(q_+-\beta)}{(1-\beta)}}$
$q_+<\beta<1$ $\mathcal{I}_{[q_-,1)}(\beta)=0$	0	0	0	$\frac{1}{(q_+-q_-)}$

Table B.1.: Values of $Q(x_1|\alpha,\beta,\phi) \propto p(\alpha,\beta\,|\,x_1,\phi)$ for different values of α,β when $\beta \leq \alpha$.

we find an expression of the form $p(\alpha,\beta|x_1,\phi) = \frac{G}{2\int_0^1\int_0^\alpha G d\alpha d\beta}$, where G is given by:

$$G = \frac{\mathcal{I}_{[q_-,q_+)}\left\{\frac{w_0}{\alpha}\mathcal{I}_{[0,\alpha)}(x) + \frac{w_1}{1-\beta}\mathcal{I}_{[\beta,1)}(x)\right\}}{\mathcal{I}_{[q_-,1)}(\alpha)\frac{w_0(min(q_+,\alpha)-q_-)}{\alpha} + \mathcal{I}_{[0,q_+)}(\beta)\frac{w_1(q_+-max(q_-,\beta))}{1-\beta}} \tag{B.10}$$

which can be evaluated for different ranges of α and β to give table B.1.

Bibliography

S. Argamon-Engelson and I. Dagan. Committee-based sample selection for probabilistic classifiers. *Journal of Artificial Intelligence Research*, 11:335–360, 1999.

H. Attias. A variational Bayesian framework for graphical models. In *Advances in Neural Information Processing Systems 12*, pages 209–215. MIT Press, 2000.

M.F. Balcan, A. Beygelzimer, and J. Langford. Agnostic active learning. *Proceedings of the 23rd International Conference on Machine Learning*, pages 65–72, 2006.

M.F. Balcan, S. Hanneke, and J. Wortman. The true sample complexity of active learning. *Machine Learning*, 80(2-3):111–139, 2008.

Y. Baram, R. El-Yaniv, and K. Luz. Online choice of active learning algorithms. *Journal of Machine Learning Research*, 5:255–291, 2004.

R. Begleiter. Active learning algorithms for matlab, 2005. URL `http://tinyurl.com/65hpfbf`.

J.N. Bernardo and A.F.M. Smith. *Bayesian Theory*. Wiley, 1994.

A. Beygelzimer, S. Dasgupta, and J. Langford. Importance weighted active learning. *Proceedings of the International Conference on Machine Learning (ICML)*, pages 49–56, 2009.

C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

C. Campbell and Y. Ying. In R.J. Brachman and T.G. Dietterich, editors, *Learning with Support Vector Machines*, number 10 in Synthesis Lectures on Artificial intelligence and machine learning. Morgan and Claypool, 2011.

C. Campbell, N. Cristianini, and A. Smola. Query learning with large margin classifiers. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 111–118, 2000.

Y. Chen and S. Mani. Study of active learning in the challenge. *International Conference on Machine Learning*, pages pp 3840–3846, 2010.

D.A. Cohn, Z. Ghahramani, and M.I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:pp 129–145, 1996.

T.M. Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers EC-14*, pages 326–334, 1965.

T.M. Cover and J.A. Thomas. *Elements of Information Theory*. Wiley, 1991.

M.K. Cowles and B.P Carlin. Markov chain monte carlo convergence diagnostics: A comparative review. *Journal of the American Statistical Association*, 91(434):883–904, 1996.

S. Dasgupta, A.T. Kalai, and C. Monteleoni. Analysis of perceptron-based active learning. In *Conference on Learning Theory*, pages 249–263, 2005.

R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. Wiley-Interscience Publication, 2000.

S. Eguchi and J. Copas. Interpreting Kullback-Leibler divergence with the Neyman-Pearson lemma. *Journal of Multivariate Analysis*, 97:2034–2040, October 2006.

J.E. Fieldsend, T.C. Bailey, R.M. Everson, D. Partridge, W.J. Krzanowski, A. Hernandez, and V. Schetinin. A Bayesian methodology for estimating uncertainty of decisions in safety-critical systems. In *Proceeding of the 2006 conference on Integrated Intelligent Systems for Engineering Design*, pages 82–96. IOS Press, 2006.

D. Fink. A compendium of conjugate priors. Technical Report 59717, Montana State University, 1997.

A. Frank and A. Asuncion. UCI machine learning repository, 2010. URL `http://archive.ics.uci.edu/ml`.

Y. Freund, H.S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. In *Machine Learning*, volume 28, pages 133–168, 1997.

D Gamerman. In C. Chatfield, J. Lindsey, M. Tanner, and J. Zidek, editors, *Markov Chain Monte Carlo stochastic simulation for Bayesian inference*, Texts in Statistical Science Series. Chapman and Hall/CRC, 2002.

D. Gamerman and H.F. Lopes. *Markov chain Monte Carlo: stochastic simulation for Bayesian inference*. Taylor and Francis, 2006.

W.R. Gilks, S. Richardson, and D.J. Spiegelhalter. *Markov chain Monte Carlo in practice*. Chapman and Hall, 1996.

Y. Guo and R. Greiner. Optimistic active learning using mutual information. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, pages 823–829, 2007.

I. Guyon, G. Cawley, G. Dror, and V. Lemaire. Design and analysis of the WCCI 2010 active learning challenge. *International Joint Conference on Neural Networks*, pages 1531–1538, 2010.

W.K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:97–109, 1970.

G. Hinton and T.J. Sejnowski. *(eds.) Unsupervised Learning, Foundations of Neural Computation*. The MIT press, 1999.

S.C.H. Hoi, R. Jin, J. Zhu, and M.R. Lyu. Batch mode active learning and its application to medical image classification. *Proc. 23rd International Conference on Machine Learning*, pages 417–424, 2006.

M. Jordan, T. Ghahrami, and L. Saul. An introduction to variational methods for graphical models. In *Machine Learning*, volume 37, pages 183–233. MIT Press, November 1999.

L.P. Kaelbling, M.L. Littman, and A.W. Moore. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell. Active learning with Gaussian processes for object categorization. *IEEE International Conference on Computer Vision*, 0:1–8, 2007.

S.B. Kotsiantis. Supervised machine learning: A review of classification techniques. *Informatica*, 31:249–268, 2007.

A. Krause and C. Guestrin. Nonmyopic active learning of Gaussian processes: An exploration-exploitation approach. In *International Conference on Machine Learning*, pages 449–456, 2007.

S. Kullback. *Information Theory and Statistics*. Wiley, 1959.

S. Kullback and R. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22:79–86, March 1951.

N.D. Lawrence, M. Seeger, and R. Herbrich. Fast sparse Gaussian process methods: The informative vector machine. In *Advances in Neural Information Processing Systems 15*, pages 609–616. MIT Press, 2003.

D.D. Lewis and W.A. Gale. A sequential algorithm for training text classifiers. In *17th annual international ACM SIGIR conference*, volume 29, pages 3–12, 1994.

S.Z. Li and A.K. Jain. *(eds) Handbook of Face Recognition*. Springer, 2005.

C.X. Ling and J. Du. Active learning with direct query construction. In *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 480–487, 2008.

D. MacKay. *Bayesian Methods for Adaptive Models*. PhD thesis, California Institute of Technology, 1991.

D. MacKay. Ensemble learning and evidence maximization. Technical report, Cavendish Laboratory, University of Cambridge, 1995.

D.J.C. MacKay. Introduction to Monte Carlo methods. In M.I. Jordan, editor, *Learning in Graphical Models*, NATO Science Series, pages 175–204. Kluwer, 1998.

R. Martinez-Cantin, Nando de Freitas, A. Doucet, and J.A. Castellano. Active policy learning for robot planning and exploration under uncertainty. In *Robotics: Science and Systems III*, June 2007.

N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21: 1087–92, 1953.

T.P. Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, MIT, 2001.

D. Nguyen-tuong and J. Peters. Local Gaussian process regression for real-time model-based robot control. In *International Converence on Intelligent Robots and Systems*, pages 380–385, 2008.

S. Park and S. Choi. Gaussian process regression for voice activity detection and speech enhancement. In *International Joint Conference on Neural Networks*, pages 2879–2882, 2008.

J. Peters, S. Schaal, and B. Schlkopf. Towards machine learning of motor skills. *Autonome Mobile Systeme*, 4:138–144, 2007.

J.C. Platt. A fast algorithm for training support vector machines using sequential minimal optimization. In *Advances in kernel methods*, pages 185–205. MIT Press, 1999.

L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, pages 257–286, 1989.

C.E. Rasmussen and H Nickisch. Gaussian process software, 2006. URL `http://www.gaussianprocess.org/gpml/`.

C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

G Rätsch. Ida benchmark repository, 2011. URL http://www.fml.tuebingen.mpg.de/Members/raetsch/benchmark.

B.D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996. URL http://www.stats.ox.ac.uk/pub/PRNN/.

N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proc. 18th International Conference on Machine Learning*, pages 441–448. Morgan Kaufmann, 2001.

B. Schlkopf and A.J. Smola. *Learning with Kernels, Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT press, 2002.

G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In *Proc. 17th International Conf. on Machine Learning*, pages 839–846. Morgan Kaufmann, San Francisco, CA, 2000.

A. Schwaighofer. SVM-light matlab interface, 2008. URL http://svmlight.joachims.org/.

F. Sebastiani and C.N.D. Ricerche. Machine learning in automated text categorization. *ACM Computing Surveys*, 34:1–47, 2002.

S. Seo, M. Wallat, T. Graepel, and K. Obermayer. Gaussian process regression: active data selection and test point rejection. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*, volume 3, pages 241–246, 2000.

B. Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2010.

B. Settles and M. Craven. An analysis of active learning strategies for sequence labeling tasks. pages 1070–1079. Conference on Empirical Methods on Natural Language Processing, 2008.

B. Settles, M. Craven, and S. Ray. Multiple-instance active learning. In *Advances in Neural Information Processing Systems NIPS*, pages 1289–1296. MIT Press, 2008.

H.S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the Fifth Workshop on Computational Learning Theory*, pages 287–294, 1992.

R.S. Sutton and A.G. Barto. *Reinforcement Learning, An Introduction*. The MIT press, 1999.

M.E. Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, pages 211–244, June 2001.

S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Proc. 17th International Conference on Machine Learning*, pages 999–1006, 2000.

S. Tong and D. Koller. Active learning for parameter estimation in Bayesian networks. In *Neural Information Processing Systems*, pages 647–653, 2001.

V. Vapnik. *Estimation of Dependences Based on Empirical Data. Springer Series in Statistics*. Springer, 1982.

V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

V. Vapnik and C. Cortes. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

H. Xu, X. Wang, Y. Liao, and C. Zheng. An uncertainty sampling-based active learning approach for support vector machines. *International Conference on Artificial Intelligence and Computational Intelligence*, pages 208–213, 2009.

X. Zhu, J. Lafferty, and Z. Ghahramani. Combining active learning and semi-supervised learning using Gaussian fields and harmonic functions. In *ICML 2003 workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, pages 58–65, 2003.