# Compact approximations to Bayesian predictive distributions

**Edward Snelson** SNELSON@GATSBY.UCL.AC.UK
Gatsby Computational Neuroscience Unit, University College London, Queen Square, London WC1N 3AR, UK

**Zoubin Ghahramani** ZOUBIN@GATSBY.UCL.AC.UK
Gatsby Computational Neuroscience Unit, University College London, Queen Square, London WC1N 3AR, UK

## Abstract

We provide a general framework for learning precise, compact, and fast representations of the Bayesian predictive distribution for a model. This framework is based on minimizing the KL divergence between the true predictive density and a suitable compact approximation. We consider various methods for doing this, both sampling based approximations, and deterministic approximations such as expectation propagation. These methods are tested on a mixture of Gaussians model for density estimation and on binary linear classification, with both synthetic data sets for visualization and several real data sets. Our results show significant reductions in prediction time and memory footprint.

## 1. Introduction

We consider the problem of representing the predictive distribution of a complex probabilistic model using a low-memory and fast-to-evaluate approximation. This has application to tasks which require a large number of rapid online predictions, for example in tracking or control scenarios. It is also useful for applications which require a compact representation of the model, for example when implementing a pattern recognition system on a mobile phone or PDA.

In most useful models, obtaining the full Bayesian predictive distribution is intractable, and approximations must be made, in either a deterministic manner, or via sampling. Current approximation schemes focus on approximating the posterior distribution over parameters well, and then storing a representation of this distribution to make future predictions. The problem is that these approximations typically lead to large representations which are costly both in time and memory to predict from. In this paper, we present methods that compress these approximations significantly whilst still maintaining accuracy.

In the case of density estimation, we need to be able to evaluate the density under our model at new data points. The full Bayesian predictive distribution takes into account our uncertainty in the model parameters by averaging over them. We want a method that accurately represents this averaging process, whilst still remaining concise for prediction.

In the case of supervised learning, e.g. regression and classification, we typically want to predict the value of an output, given a new set of inputs. If this output is then used in further downstream processing, the full predictive distribution is especially important for propagating uncertainty. Again, we want an approximation to this distribution that has correctly taken into account the averaging over model parameters.

## 2. The approximation framework

Suppose we have a probabilistic model $p(\mathbf{y}|\mathbf{w})$, parameterized by $\mathbf{w}$, on which we have the prior distribution $p(\mathbf{w})$. We have observed a set of data $\mathcal{D}$ consisting of $N$ i.i.d. observations $\{\mathbf{y}_n\}_{n=1}^N$. A standard approach is to approximate the posterior $p(\mathbf{w}|\mathcal{D})$, and use this to make future predictions. If $\mathbf{w}$ is a real-valued vector, then one might find a Gaussian approximation by Laplace's method (Tierney & Kadane, 1986), a variational method (Jaakkola & Jordan, 2000), or expectation propagation (EP) (Minka, 2001). Alternatively a large number of samples from $p(\mathbf{w}|\mathcal{D})$ could be obtained using Markov chain Monte Carlo (MCMC) methods. Predictions can then be made by using this approximation to compute the following average over parameters:

$$p(\mathbf{y}|\mathcal{D}) = \int d\mathbf{w}\, p(\mathbf{y}|\mathbf{w})p(\mathbf{w}|\mathcal{D}) \; . \qquad (1)$$

Appearing in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005. Copyright 2005 by the author(s)/owner(s).

This is called the Bayesian predictive distribution, and it is a Bayesian model average over $\mathbf{w}$. Often instead, only a single value of $\mathbf{w}$ is used to make predictions, for example the posterior mean $\bar{\mathbf{w}} = \int d\mathbf{w}\, \mathbf{w} p(\mathbf{w}|\mathcal{D})$ is used via $p(\mathbf{y}|\bar{\mathbf{w}})$, or other traditional point estimates such as ML or MAP parameters.

However the quantity we are interested in is the predictive distribution $p(\mathbf{y}|\mathcal{D})$, and it is this we will try to approximate with some other parameterized distribution $p(\mathbf{y}|\theta)$. The main motivation for seeking a separate approximation is that we hope to find a compact representation explicitly tailored for prediction. The representations we usually have for the posterior are often not compact, especially in high dimensions. For MCMC methods that sample from $p(\mathbf{w}|\mathcal{D})$, keeping around a large number of samples in memory is costly, both taking up storage and slowing down predictions, and yet only keeping a small number may sacrifice accuracy in predictive probability. The goal of this paper is to find a representation that is both a compact and sufficiently accurate representation of the predictive distribution. Our motivation is not only a concern with memory requirements of the representation, but also with the speed with which the predictive density can be evaluated. By finding an explicit parametric representation, future predictions can be made quickly when compared with evaluating the potentially costly integral (1) on the fly.

We will return to the issue of choosing the form for this approximating distribution in the next section 2.1, suffice to say there are a number of possibilities depending on the model involved. A natural way to make the approximation is to try to minimize the KL divergence between the predictive distribution and the approximating distribution:

$$\begin{aligned}\hat{\theta} &= \arg\min_{\theta}\ \mathrm{KL}\big[p(\mathbf{y}|\mathcal{D})\,||\,p(\mathbf{y}|\theta)\big] \\ &= \arg\max_{\theta}\ \int d\mathbf{y}\, p(\mathbf{y}|\mathcal{D}) \log p(\mathbf{y}|\theta)\ .\end{aligned} \tag{2}$$

The KL divergence is zero iff $p(\mathbf{y}|\theta) = p(\mathbf{y}|\mathcal{D})$ and provides a principled information theoretic measure of how close $p(\mathbf{y}|\theta)$ is to $p(\mathbf{y}|\mathcal{D})$.

### 2.1. Choosing the approximating density

One simple possibility is to choose the approximating density to be of exactly the same form as the original model density $p(\mathbf{y}|\mathbf{w})$. Then the problem in (2) becomes one of finding the single best setting of the parameters $\hat{\mathbf{w}}$ of the original model, so that KL is minimized between this and the predictive distribution:

$$\hat{\mathbf{w}} = \arg\max_{\mathbf{w}}\ \int d\mathbf{y}\, p(\mathbf{y}|\mathcal{D}) \log p(\mathbf{y}|\mathbf{w})\ . \tag{3}$$

This 'best setting' is commonly known as the Bayes point, although precise definitions vary (Ruján, 1997; Herbrich et al., 2001). Often the posterior mean $\bar{\mathbf{w}}$ is taken as an approximation to this Bayes point.

Such a single point approximation is often unsuitable as a representation for predictions, because the nature of the averaging process of (1) tends to lead to a predictive density that is of a more complex class of distributions than the original model density. A better alternative is to take an equal weighted mixture of a small number of original model densities:

$$p(\mathbf{y}|\theta) = \frac{1}{M}\sum_{m=1}^{M} p(\mathbf{y}|\mathbf{w}_m)\ , \tag{4}$$

where $\theta = \{\mathbf{w}_m\}_{m=1}^{M}$. One can then search for the $\{\mathbf{w}_m\}$ to minimize KL in (2). This has the nice interpretation of finding a small number of 'best samples' from the posterior to act as the predictive representation, since (4) is exactly the way one would combine samples to approximate the integral of (1). We also know that if we choose $M$ large enough we will be able to approximate the predictive distribution to arbitrary accuracy without overfitting. In practice we will want to choose $M$ to be as small as we can get away with, and adjusting $M$ provides an easy way to make a compactness/accuracy trade-off. Of course we could also consider finding weights for these mixture components, rather than just weighting them equally.

Finally, we do not have to restrict ourselves to working with the same type of approximating distribution as the model distribution at all. We could choose an entirely different class of distribution, preferably something that is of greater complexity than the original model.

## 3. Density Estimation

In the previous section we presented the framework within which one can find an approximating predictive distribution, but we are left with the problem of how to perform the maximization of (2). In this section we propose a sampling based scheme for density estimation. A deterministic alternative discussed in section 6 is a possibility for the future.

### 3.1. A sampling based scheme using 'fake data'

One obvious way to evaluate (2) is to generate samples from $p(\mathbf{y}|\mathcal{D})$, or 'fake data'. First we sample from the posterior $p(\mathbf{w}|\mathcal{D})$ using our favorite sampling method and obtain a large set of samples $\{\mathbf{w}_s\}_{s=1}^{S}$. Given these samples our estimate for the predictive distribution is:

$$p(\mathbf{y}|\mathcal{D}) \approx \frac{1}{S}\sum_{s=1}^{S} p(\mathbf{y}|\mathbf{w}_s) \ . \quad (5)$$

We could just use this as our representation for prediction, but it involves keeping around a large number of samples, and then evaluating the density of a large mixture of the original model, which may be too costly and memory intensive. Instead we can sample again from the mixture (5) to generate 'fake predictive data' $\{\mathbf{y}_i\}_{i=1}^{I}$. The method for finding the approximation (2) is then a maximum likelihood (ML) problem on the generated 'fake data':

$$\hat{\theta} = \arg\max_{\theta} \sum_i \log p(\mathbf{y}_i|\theta) \ . \quad (6)$$

If we choose as our approximation the mixture (4), then we have succeeded in reducing the size of the predictive representation from $S$ samples from the posterior, to a small number $M$ of 'best samples', and our future predictions will be both quick and accurate.

For complete simplicity and concreteness, let us consider the case where the original model is an equal weighted mixture of one dimensional Gaussians with fixed variances: $p(y|\mathbf{w}) = \frac{1}{J}\sum_{j=1}^{J} \mathcal{N}_y(w^{(j)}, v^{(j)})$. Having obtained samples from the posterior $\{\mathbf{w}_s\}$, we generate 'fake data' from the $(J \times S)$ mixture $\frac{1}{JS}\sum_{js} \mathcal{N}_y(w_s^{(j)}, v^{(j)})$. We then perform an ML fit to this generated data with a mixture of $K$ Gaussians, where $K > J$, and where we may well also want to fit variances and mixing proportions too. The outcome of this procedure is that we have reduced the predictive representation obtained from sampling, which is a mixture of $J \times S$ Gaussians, down to a mixture of $K$ Gaussians, where $K$ can be chosen to trade off compactness and accuracy. The results of doing this are shown later in section 5.1.

## 4. Supervised Learning

We extend the framework of the previous section to supervised learning, where we have a conditional model representing the relationship between an input $\mathbf{x}$ and an output $\mathbf{y}$. Here it does not make sense to minimize KL divergence for a particular input $\mathbf{x}$, so instead we consider average KL divergence over an input density $p(\mathbf{x})$:

$$\hat{\theta} = \arg\min_{\theta} \int d\mathbf{x}\, p(\mathbf{x})\, \mathrm{KL}\big[p(\mathbf{y}|\mathbf{x},\mathcal{D}) \,||\, p(\mathbf{y}|\mathbf{x},\theta)\big] \ . \quad (7)$$

This will be optimal in the sense of KL loss, but we could have a more general decision-theoretic loss function $l(y', y)$ – the cost of predicting $y'$ when the true output was $y$. In this case we would want our approximation to minimize expected loss. Different loss functions could lead to interesting different approximations, but we do not explore this further here. See Herbrich et al. (2001) for further discussion of loss functions in binary classification.

Specifying an input density gives us the flexibility to be able to tune the approximation to be best in a particular region of input space where we expect to make predictions. For example we may specify a Gaussian density for $p(\mathbf{x})$, in which case the integral of (7) can be computed by drawing a suitable number of input samples $\{\mathbf{x}_i\}$. Alternatively we may have access to a large amount of unlabeled data, which we can use as a surrogate for samples from an input density. Either way the approximation is found by:

$$\hat{\theta} = \arg\max_{\theta} \sum_{i=1}^{I} \int d\mathbf{y}\, p(\mathbf{y}|\mathbf{x}_i,\mathcal{D}) \log p(\mathbf{y}|\mathbf{x}_i,\theta) \ . \quad (8)$$

### 4.1. Binary linear classifiers

To be concrete let us consider the particular case of a binary linear classifier, where $y \in \{\pm 1\}$ and $\mathbf{x}$ is a real valued $D$-dimensional vector:

$$p(y|\mathbf{x},\mathbf{w}) = \phi(\mathbf{w}^\top \mathbf{x} y) \ . \quad (9)$$

Here $\phi$ is an appropriate link function, for example a logistic function for logistic regression, or a cumulative normal for the probit model. In the case of binary classification, (8) can be rewritten as:

$$\hat{\theta} = \arg\max_{\theta} \sum_{i=1}^{I} \big[a_i \log p(y=+1|\mathbf{x}_i,\theta) + (1-a_i)\log p(y=-1|\mathbf{x}_i,\theta)\big] \ , \quad (10)$$

where $a_i = p(y=+1|\mathbf{x}_i,\mathcal{D})$. This has the interesting interpretation as a weighted maximum likelihood fit to the input sampled data. Of course, we need to be able to calculate the weights $\{a_i\}$ corresponding to the input samples, and this can be done in a number of ways. Once again we consider a sampling based approximation in section 4.1.1, followed by a deterministic approximation in section 4.1.2, and we choose a small mixture representation (4) for the approximation:

$$p(y|\mathbf{x},\theta) = \frac{1}{M}\sum_{m=1}^{M} p(y|\mathbf{x},\mathbf{w}_m) = \frac{1}{M}\sum_{m=1}^{M} \phi(\mathbf{w}_m^\top \mathbf{x} y) \ . \quad (11)$$

### 4.1.1. A sampling scheme

We can obtain a large number of parameter samples from the posterior, and use these samples to make predictions at the $\{\mathbf{x}_i\}$:

$$a_i = p(y = +1|\mathbf{x}_i, \mathcal{D}) \approx \frac{1}{S}\sum_{s=1}^{S} \phi(\mathbf{w}_s^\top \mathbf{x}_i) \; . \qquad (12)$$

Clearly one could just keep around all these samples to make future predictions, rather than try to fit the distribution $p(y|\mathbf{x}, \theta)$, but memory and computational limitations might make this highly impractical. The procedure described in (10) provides a neat way to select a small number of parameter 'samples' from which to make predictions, and it significantly out-performs just randomly picking a few samples to keep, as we show in section 5.2.

### 4.1.2. A deterministic scheme

An alternative is to use a deterministic approximation procedure such as expectation propagation (Minka, 2001). EP is an iterative method that approximates a product of factors by a product of simpler factors, for example Gaussians, the integral of which can be computed analytically. We can use EP to find a Gaussian approximation to the posterior, $q(\mathbf{w}) \approx p(\mathbf{w}|\mathcal{D})$, and then use this approximation to calculate the $\{a_i\}$:

$$\begin{aligned} a_i &= p(y = +1|\mathbf{x}_i, \mathcal{D}) \\ &\approx \int \mathrm{d}\mathbf{w}\, p(y = +1|\mathbf{x}_i, \mathbf{w}) q(\mathbf{w}) \\ &= \int \mathrm{d}\mathbf{w}\, \phi(\mathbf{w}^\top \mathbf{x}_i)\, \mathcal{N}_{\mathbf{w}}(\mathbf{m}, V) \; , \end{aligned} \qquad (13)$$

where $\mathcal{N}_{\mathbf{w}}(\mathbf{m}, V)$ is a Gaussian distribution on $\mathbf{w}$ with mean $\mathbf{m}$ and covariance $V$. This integral is only one dimensional, because the linear model projects the integral onto the direction of $\mathbf{x}_i$. Depending on the link function $\phi$, this integral can be computed in a variety of ways. For example, in the probit model, it involves the evaluation of an error function. For the logistic model, the integral can be done by quadrature, and is best done by a pre-computed lookup table for speed.

The standard approach to deterministic approximation would be to keep around the mean and covariance of the Gaussian approximation to the posterior, and use these to make all future predictions. However, depending on the model, the predictions are rather costly to evaluate via (13). E.g. in logistic regression there is a memory/speed trade-off when making this calculation by a lookup table for a large number of predictions. On the other hand, it can be extremely quick to evaluate $p(y|\mathbf{x}, \theta)$ on a large test set, as we show in section 5.3. The other advantage to our method is that in a high dimensional model the predictive representation is much more concise. The covariance matrix of the Gaussian approximation to $p(\mathbf{w}|\mathcal{D})$ grows in size as $D^2$, whereas the mixture representation based on (4) is linear with $D$. This can lead to a significant reduction in memory requirements, and a significant increase in speed in evaluating future predictions.

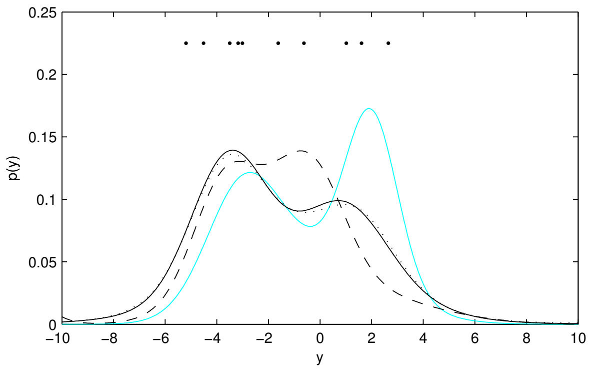

*Figure 1.* A one dimensional mixture of Gaussians model. The cyan/faint line shows the mixture of 3 Gaussians from which the 10 data points shown as black dots were generated. The dotted line shows the predictive distribution obtained by extensive MCMC sampling. The dashed line shows the distribution obtained by keeping just 5 random parameter samples from the MCMC. The solid line shows the ML fit of a mixture of 5 Gaussians to 'fake data' (see section 3.1).

## 5. Results

We have tested the schemes described in the previous sections on a number of examples. We start with very simple examples to illustrate key ideas clearly.

### 5.1. Density estimation

Ten data points were generated i.i.d. from an equal weighted mixture of $J = 3$ 1-d fixed variance Gaussians, as shown in Figure 1. The sampling method of 3.1 was then followed, with a broad spherical Gaussian prior placed on the means. First a large number of samples were drawn from the posterior $p(\mathbf{w}|\mathcal{D})$ using MCMC, 'fake data' was generated from (5), and a mixture of $K = 5$ Gaussians fitted to this data using (6). The variances and mixing proportions were also adjusted in this fit – 15 parameters overall.

Figure 1 shows that the 'true' predictive distribution, as calculated by using all the samples, is almost exactly approximated by using a mixture of only 5 Gaussians. Thus this is an extremely concise representation for making future predictions. The figure also shows that

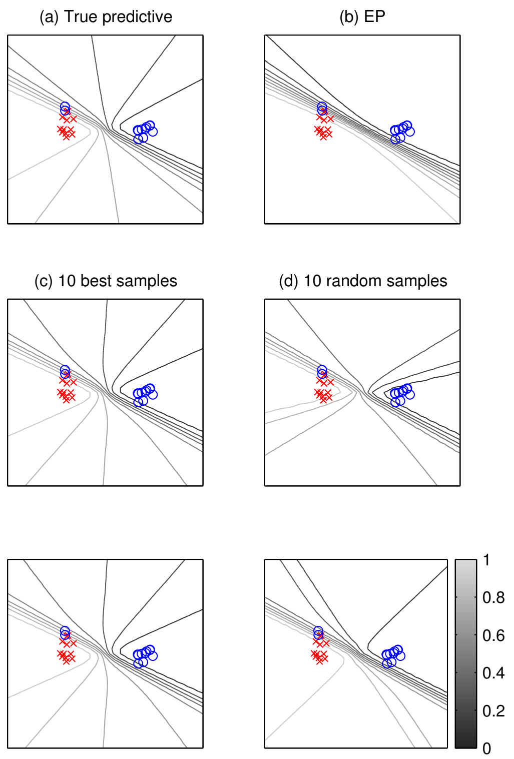

*Figure 2.* Contour plots showing approximations to the predictive distribution $p(y = +1|\mathbf{x}, \mathcal{D})$. The gray lines are lines of constant predictive probability of value shown by the side-bar. The blue circles ($-1$) and the red crosses ($+1$) are the original data. (a) Predictive distribution from extensive sampling using MCMC. (b) Direct predictions using Gaussian obtained from EP. (c) Mixture of 10 'best samples'. (d) 10 random samples from the MCMC procedure. The third row contains another 10 'best samples', relearned with a random initialization, and another 10 different random samples from the MCMC.

fitting the mixture of Gaussians is a lot better than just picking 5 random samples from the posterior (an equivalent number of parameters because this corresponds to a mixture of $J \times 5 = 15$ Gaussians).

## 5.2. Supervised learning – sampling based

The second example we consider is logistic regression, where for visualization purposes we consider a 2D case first. 24 data points were generated, half labeled $+1$, the half other $-1$; these points are shown as circles and crosses in Figure 2. A large number of samples from the posterior $p(\mathbf{w}|\mathcal{D})$ were generated, using a spherical Gaussian prior, and a fraction of these are shown in Figure 3. Figure 2(a) shows a contour plot of the predictive distribution obtained from using all the samples, in a region of space surrounding the training data. This plot can be regarded as ground truth, to which

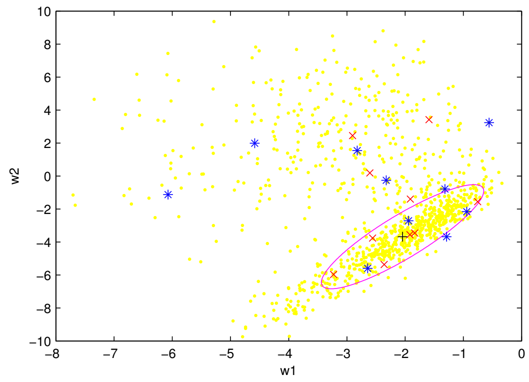

*Figure 3.* The posterior for the binary logistic classifier with data as shown in Figure 2. The axes are the two weights $(w_1, w_2) = \mathbf{w}$ of the logistic classifier. The dots represent samples from the posterior $p(\mathbf{w}|\mathcal{D})$. The solid line is the two standard deviation ellipse for the EP approximation to the posterior. The stars are 10 random samples from the posterior (see Fig. 2(d)). The crosses are 10 'best samples' as found by using the approximation of section 4.1.1 (see Fig. 2(c)).

our approximation methods are aiming to match.

Then a Gaussian approximation to the posterior was found using EP. The details of the derivation of the EP procedure for logistic regression are lengthy and are not shown here, but they follow Minka (2001) closely. Figure 2(b) shows the predictive distribution obtained directly from this approximation using the same formula as (13). One can see that in this case the EP approximation is not good; this can also be understood by looking at the EP two standard deviation ellipse plotted in Figure 3, where we see that the EP Gaussian approximation misses a large extended region of the posterior density. For such an example where EP does not provide a good approximation, then a sampling based approximation is the way to go. For this example, we therefore show how the method of 4.1.1 leads to a concise and accurate representation of the 'truth' 2(a).

To obtain the 10 mixture fit of 2(c), 500 input samples from a broad Gaussian input density were used, and by the procedure of 4.1.1, 10 'best posterior samples' were learned so as to approximate the true predictive distribution 2(a). As we can see, the predictive distribution in 2(c) looks very similar to that in 2(a), and is significantly better than either the EP approximation, or the distribution predicted by choosing to keep 10 random samples as our representation (2(d)). The second row of Figure 2 also shows that our method is very robust to different random initializations of the gradient algorithm. This is in contrast to choosing different batches of 10 random samples, in which case

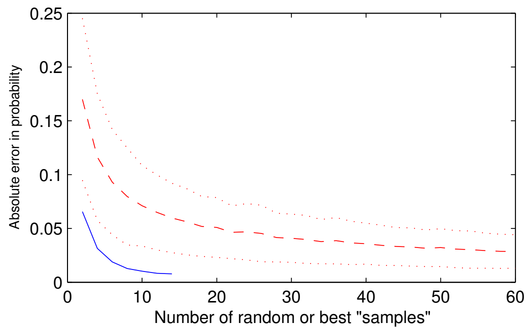

*Figure 4.* The accuracy of the predictive representation is plotted as a function of the number of random samples (red dashed) and 'best samples' (solid blue). The dotted lines are 1 standard deviation lines for the random sample method. The variability of the 'best sample' method is so small that it would not show up on this plot.

there is high variability to the predictive contours.

The difference between carefully tuning our 'samples' by KL minimization, to just picking some random samples, is further highlighted by Figure 4. This figure was generated by taking the absolute difference in probability[1] between the distributions of Figure 2(c) or 2(d) and the ground truth 2(a), spatially averaged over a fine grid of points. This was done 1000 times, with different random initializations for the 'best samples' gradient algorithm, and with different random batches of samples for the random samples method. Finally the experiment was repeated varying the number of samples (random or best) over a large range.

We have already demonstrated that tuning 'samples' using (10) leads to greater accuracy than using the *same* number of random samples. However one might think that we could just use a few more random samples, and achieve a comparable accuracy, without all the expense of choosing them by KL minimization. Figure 4 shows that, at least for this example, this is not the case. The gain in accuracy with increasing number of random samples is a very slow process. In fact one would need about 500 random samples to achieve the same mean accuracy as just 10 'best samples'. Even then, the variance of the error for the random sample method is very high, whereas the variance is essentially non-existent for the 'best sample' method – it finds the same quality of solution every time.

### 5.2.1. Experiments on real data sets

We applied the sampling based method of the previous section to several real datasets. Two data sets (Pima, Spambase) were taken from the UCI repository (Blake & Merz, 1998) and two (SA heart disease, USPS digits) from Hastie et al. (2001). Our method is particularly useful in the case where one has a small amount of data, and consequently uncertainty is high. For this reason, from each of the data sets we took just 50 data points for training, and left the rest (usually a large number) for testing. As in the previous section, we placed a Gaussian prior on the weights for a logistic classifier, and sampled extensively from the posterior distribution of the weights to obtain the 'ground truth' predictive distribution. To find a compact representation for prediction, as before we tuned a small number of 'best samples' to this distribution. As a comparison we also chose the same number of samples randomly from the posterior distribution. To measure performance on the test sets, we thresholded these predictive distributions to obtain a measure of classification error.

The results are shown in Figure 5. The actual value of test error is not important, as firstly we only trained on a very small number of points, and secondly we are using a very simple logistic classifier. The important comparison is between the 'ground truth' error rate (obtained by extensive sampling and marked by the straight lines) and the different compact approximations. For each of the data sets we see a very similar effect to that of the synthetic 2D data in Figure 4. We find that choosing random samples instead of tuning them degrades mean performance, and leads to very high variance. In contrast, spending a little extra time in training by searching for 'best samples' gives an error rate very close to 'ground truth', with very high reliability. Suppose we initially took 5000 samples from the posterior, and we choose to represent the predictive distribution with 50 tuned 'samples', then we have reduced by a factor of 100 both the time for future predictions and the memory requirement.

Although the difference in error rates between tuned and random sampling shown in the plots is not large, we expect these differences to increase if one was to use a more sophisticated classification model than logistic regression. This is because a linear classifier tends to produce a relatively simple posterior distribution in parameter space. A more complex classifier, with many different types of parameters, is likely to give a complicated posterior distribution, and here random selection of samples is expected to perform even worse than shown. It is worth remembering that we can use our method to tune 'samples' on *any* Bayesian model where derivatives can be taken with respect to the parameters.

[1]E.g. a difference in predictive probability of 0.006 might mean that the true probability at input $\mathbf{x}$ was $p(y = +1|\mathbf{x}, \mathcal{D}) = 0.4$ whereas our method gave 0.394.

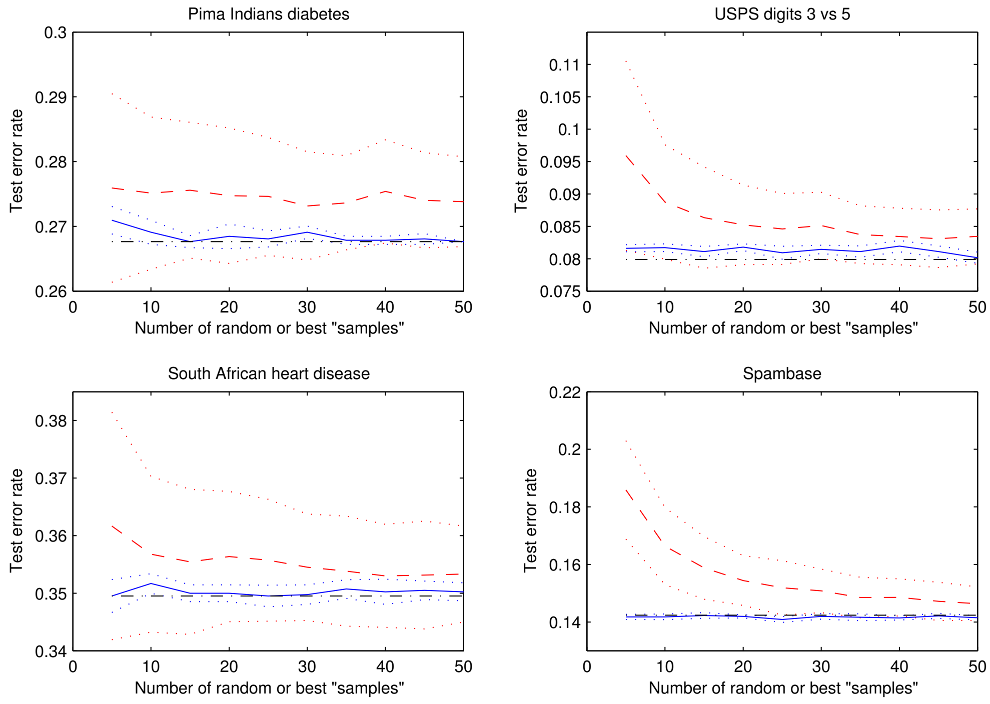

*Figure 5.* The classification test error rate is shown for 4 different real data sets as a function of the number of random samples (red dashed/dotted), or the number of best 'samples' (blue solid/dotted). The dotted lines are 1 standard deviation lines from multiple trials. The error rate from extensive sampling is shown as the straight black dot-dashed line.

## 5.3. Supervised learning – deterministic

Another regime in which the mixture representation really helps is when we know we are going to have to make a large number of predictions in a high dimensional problem, and where we believe that a deterministic approximation such as EP achieves high enough accuracy. As discussed in 4.1.2, the size of the EP representation scales as $D^2$, as opposed to the linear scaling of the mixture representation, and the integral (13) proves costly over a large test set. To simulate this regime, a similar training set to the previous example was generated, but with a large number of dimensions $D$, and a large test set of size 100,000 was also generated. Then computation times (in Matlab) for prediction were measured, including the extra training time taken by our method to find the 'best samples', whilst checking that the accuracy of predictions of the two methods remained equivalent. The results are summarized in Table 1.

Just 10 mixture components were used in the fitted model, and yet the accuracy remains extremely good, as shown by the final column which measures how much the mixture approximation deviates from the 'truth' (MCMC) over the test set[1]. So to summarize, we buy ourselves three things in high dimensions. Firstly, as the table shows, the actual prediction times

| $D$ | 100 | 150 | 200 |
|---|---|---|---|
| EP prediction time /s | 8.92 | 58.3 | 147 |
| extra training time for mixture /s | 2.78 | 3.77 | 9.32 |
| mixture prediction time /s | 0.825 | 1.15 | 1.56 |
| mean difference in predictive probability[1] | 0.006 | 0.007 | 0.008 |

*Table 1.* Comparison of training and prediction times on a high dimensional problem

on the large test set are much faster when using the mixture representation (by about a factor of 100 for $D = 200$, and scaling linearly rather than quadratically in $D$). Secondly the storage requirement of this representation is much smaller. Thirdly, in the case of logistic regression, we do not have to store a huge lookup table for the integral of (13) in order to make the predictions (though we do still need it for training).

## 6. Extensions and related work

One of the limitations with the approaches presented in this paper so far is that the final predictive approximation can only be as accurate as the intermediate approximation of the posterior, never more so. Of course we do gain greatly in compactness and speed in all the ways described in the previous section. However, it may be possible to find a more accurate direct deterministic approximation to the predictive distribution that bypasses the approximation of the posterior. We demonstrate this idea for the case of density estimation by considering directly approximating (2). Expanding:

$$\begin{aligned}\mathcal{L}(\theta) &= \int \mathrm{d}\mathbf{y}\, p(\mathbf{y}|\mathcal{D}) \log p(\mathbf{y}|\theta) \\ &\propto \int \mathrm{d}\mathbf{y} \left[ \int \mathrm{d}\mathbf{w}\, p(\mathbf{y}|\mathbf{w}) \prod_{n=1}^{N} p(\mathbf{y}_n|\mathbf{w}) p(\mathbf{w}) \right] \log p(\mathbf{y}|\theta) \\ &= \int \mathrm{d}\mathbf{w} \prod_{n=1}^{N} p(\mathbf{y}_n|\mathbf{w}) p(\mathbf{w}) \underbrace{\left[ \int \mathrm{d}\mathbf{y}\, p(\mathbf{y}|\mathbf{w}) \log p(\mathbf{y}|\theta) \right]}_{t_{N+1}(\mathbf{w})} \\ &= \int \mathrm{d}\mathbf{w} \prod_{n=0}^{N+1} t_n(\mathbf{w}) \,, \end{aligned} \tag{14}$$

where we have substituted (1) and assumed i.i.d. data. Here $t_0(\mathbf{w}) \triangleq p(\mathbf{w})$ and $t_n(\mathbf{w}) \triangleq p(\mathbf{y}_n|\mathbf{w})$, $n = 1 \ldots N$. Since (14) ends up being an integral of a product of factors, we can use EP to approximate this integral. The way EP is employed here is similar to the usual calculation of the evidence, but with one extra term $t_{N+1}$. When it comes to matching moments in EP, we need to be able to compute quantities such as:

$$\begin{aligned} Z_{N+1} &= \int \mathrm{d}\mathbf{w}\, q(\mathbf{w}) t_{N+1}(\mathbf{w}) \\ &= \int \mathrm{d}\mathbf{y} \left[ \int \mathrm{d}\mathbf{w}\, q(\mathbf{w}) p(\mathbf{y}|\mathbf{w}) \right] \log p(\mathbf{y}|\theta) \,, \end{aligned} \tag{15}$$

where $q(\mathbf{w})$ is a Gaussian approximation. Suppose $\mathbf{w}$ represents the means in the simple fixed variance mixture of Gaussians model of section 3.1. Then the inner bracketed integral can be done analytically, and is another mixture of Gaussians itself. We are then left with the outer integral over $\mathbf{y}$. This can be done numerically, for example quadrature for low-dimensions, or sampling from the bracketed mixture of Gaussians for high dimensions.

To maximize $\mathcal{L}(\theta)$, we also want gradients with respect to $\theta$. These gradients can be approximated in the same fashion where the extra term in the product becomes:

$$\nabla_\theta \mathbf{t}_{N+1}(\mathbf{w}) = \int \mathrm{d}\mathbf{y}\, p(\mathbf{y}|\mathbf{w}) \frac{\nabla_\theta\, p(\mathbf{y}|\theta)}{p(\mathbf{y}|\theta)} \,. \tag{16}$$

The problem with this method is that it is not possible to approximate this extra derivative term with a Gaussian, as it is possibly a complicated function of $\mathbf{w}$ with positive, negative and zero regions. Although not currently viable with tools in the EP framework, this approach may become possible with some modifications, or with other approximate integration techniques.

The idea of focusing approximations on predictive performance has been explored in the context of Bayesian network (BN) model selection (Heckerman & Chickering, 2000). The authors compare the structure of the most probable BN with the BN which best matches the predictive distribution. They found that the latter was generally more complex, which agrees with our arguments in section 2.1. A natural extension of their work, in line with our approach, would be to fit a mixture of BNs as the approximating model.

## 7. Conclusions

We have described a general framework for making approximations to predictive distributions that can be applied to a wide variety of models. We have demonstrated significant advantages over more traditional approaches – we can find a compact representation for fast prediction, whilst still retaining the desired properties of parameter averaging. For logistic regression we have shown that tuning a few 'best samples' using a large number of true samples, reduces storage requirements for prediction by a factor of 50 over choosing a random subset, and massively reduces variability in the predictive distribution (Figure 4). We see the same effect when we test the method on real data sets, except that this time the effect is measured through actual test classification error. Moreover prediction time can also be reduced by about two orders of magnitude for a high dimensional problem (Table 1).

## References

Blake, C., & Merz, C. (1998). UCI repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The Elements of Statistical Learning*. Springer-Verlag. `http://www-stat-class.stanford.edu/~tibs/ElemStatLearn/`.

Heckerman, D., & Chickering, D. (2000). A comparison of scientific and engineering criteria for Bayesian model selection. *Statistics and Computing*, *10*, 55–62.

Herbrich, R., Graepel, T., & Campbell, C. (2001). Bayes point machines. *JMLR*, *1*, 245–279.

Jaakkola, T., & Jordan, M. (2000). Bayesian parameter estimation via variational methods. *Statistics and Computing*, *10*, 25–37.

Minka, T. P. (2001). *A family of algorithms for approximate Bayesian inference*. Doctoral dissertation, M.I.T.

Ruján, P. (1997). Playing billiards in version space. *Neural Computation*, *9*, 99–122.

Tierney, L., & Kadane, J. B. (1986). Accurate approximations for posterior moments and marginal densities. *Journal of the American Statistical Association*, *81*.