C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006, ISBN 026218253X. www.GaussianProcess.org/gpml

Chapter 3

Classification

In chapter 2 we have considered *regression* problems, where the targets are real valued. Another important class of problems is *classification*[1] problems, where we wish to assign an input pattern $\mathbf{x}$ to one of C classes, $\mathcal{C}_1, \ldots, \mathcal{C}_C$. Practical examples of classification problems are handwritten digit recognition (where we wish to classify a digitized image of a handwritten digit into one of ten classes 0-9), and the classification of objects detected in astronomical sky surveys into stars or galaxies. (Information on the distribution of galaxies in the universe is important for theories of the early universe.) These examples nicely illustrate that classification problems can either be binary (or two-class, $C = 2$) or multi-class ($C > 2$). binary, multi-class

We will focus attention on *probabilistic classification*, where test predictions take the form of class probabilities; this contrasts with methods which provide only a *guess* at the class label, and this distinction is analogous to the difference between predictive distributions and point predictions in the regression setting. Since generalization to test cases inherently involves some level of uncertainty, it seems natural to attempt to make predictions in a way that reflects these uncertainties. In a practical application one may well seek a class guess, which can be obtained as the solution to a *decision problem*, involving the predictive probabilities as well as a specification of the consequences of making specific predictions (the loss function). probabilistic classification

Both classification and regression can be viewed as *function approximation* problems. Unfortunately, the solution of classification problems using Gaussian processes is rather more demanding than for the regression problems considered in chapter 2. This is because we assumed in the previous chapter that the likelihood function was Gaussian; a Gaussian process prior combined with a Gaussian likelihood gives rise to a posterior Gaussian process over functions, and everything remains analytically tractable. For classification models, where the targets are discrete class labels, the Gaussian likelihood is inappropriate;[2] non-Gaussian likelihood

[1] In the statistics literature classification is often called discrimination.

[2] One may choose to ignore the discreteness of the target values, and use a regression treatment, where all targets happen to be say ± 1 for binary classification. This is known as

in this chapter we treat methods of approximate inference for classification, where exact inference is not feasible.[3]

Section 3.1 provides a general discussion of classification problems, and describes the *generative* and *discriminative* approaches to these problems. In section 2.1 we saw how Gaussian process regression (GPR) can be obtained by generalizing linear regression. In section 3.2 we describe an analogue of linear regression in the classification case, logistic regression. In section 3.3 logistic regression is generalized to yield Gaussian process classification (GPC) using again the ideas behind the generalization of linear regression to GPR. For GPR the combination of a GP prior with a Gaussian likelihood gives rise to a posterior which is again a Gaussian process. In the classification case the likelihood is non-Gaussian but the posterior process can be *approximated* by a GP. The Laplace approximation for GPC is described in section 3.4 (for binary classification) and in section 3.5 (for multi-class classification), and the expectation propagation algorithm (for binary classification) is described in section 3.6. Both of these methods make use of a Gaussian approximation to the posterior. Experimental results for GPC are given in section 3.7, and a discussion of these results is provided in section 3.8.

3.1 Classification Problems

The natural starting point for discussing approaches to classification is the joint probability $p(y, \mathbf{x})$, where y denotes the class label. Using Bayes' theorem this joint probability can be decomposed either as $p(y)p(\mathbf{x}|y)$ or as $p(\mathbf{x})p(y|\mathbf{x})$. This gives rise to two different approaches to classification problems. The first, which we call the *generative* approach, models the class-conditional distributions $p(\mathbf{x}|y)$ for $y = \mathcal{C}_1, \ldots, \mathcal{C}_C$ and also the prior probabilities of each class, and then computes the posterior probability for each class using

generative approach

$$p(y|\mathbf{x}) \;=\; \frac{p(y)p(\mathbf{x}|y)}{\sum_{c=1}^{C} p(\mathcal{C}_c)p(\mathbf{x}|\mathcal{C}_c)}. \tag{3.1}$$

discriminative approach

The alternative approach, which we call the *discriminative* approach, focusses on modelling $p(y|\mathbf{x})$ directly. Dawid [1976] calls the generative and discriminative approaches the sampling and diagnostic paradigms, respectively.

To turn both the generative and discriminative approaches into practical methods we will need to create *models* for either $p(\mathbf{x}|y)$, or $p(y|\mathbf{x})$ respectively.[4] These could either be of parametric form, or non-parametric models such as those based on nearest neighbours. For the generative case a simple, com-

generative model example

least-squares classification, see section 6.5.

[3] Note, that the important distinction is between Gaussian and non-Gaussian likelihoods; regression with a non-Gaussian likelihood requires a similar treatment, but since classification defines an important conceptual and application area, we have chosen to treat it in a separate chapter; for non-Gaussian likelihoods in general, see section 9.3.

[4] For the generative approach inference for $p(y)$ is generally straightforward, being estimation of a binomial probability in the binary case, or a multinomial probability in the multi-class case.

mon choice would be to model the class-conditional densities with Gaussians: $p(\mathbf{x}|\mathcal{C}_c) = \mathcal{N}(\boldsymbol{\mu}_c, \Sigma_c)$. A Bayesian treatment can be obtained by placing appropriate priors on the mean and covariance of each of the Gaussians. However, note that this Gaussian model makes a strong assumption on the form of class-conditional density and if this is inappropriate the model may perform poorly.

discriminative model example

For the binary discriminative case one simple idea is to turn the output of a regression model into a class probability using a *response function* (the inverse of a *link function*), which "squashes" its argument, which can lie in the domain $(-\infty, \infty)$, into the range $[0, 1]$, guaranteeing a valid probabilistic interpretation.

One example is the *linear logistic regression* model

$$p(\mathcal{C}_1|\mathbf{x}) \;=\; \lambda(\mathbf{x}^\top \mathbf{w}), \quad \text{where} \quad \lambda(z) \;=\; \frac{1}{1+\exp(-z)}, \tag{3.2}$$

response function

which combines the linear model with the logistic response function. Another common choice of response function is the cumulative density function of a standard normal distribution $\Phi(z) = \int_{-\infty}^{z} \mathcal{N}(x|0,\, 1)dx$. This approach is known as *probit regression*. Just as we gave a Bayesian approach to linear regression in chapter 2 we can take a parallel approach to logistic regression, as discussed in section 3.2. As in the regression case, this model is an important step towards the Gaussian process classifier.

probit regression

generative or discriminative?

Given that there are the generative and discriminative approaches, which one should we prefer? This is perhaps the biggest question in classification, and we do not believe that there is a right answer, as both ways of writing the joint $p(y, \mathbf{x})$ are correct. However, it is possible to identify some strengths and weaknesses of the two approaches. The discriminative approach is appealing in that it is directly modelling what we want, $p(y|\mathbf{x})$. Also, density estimation for the class-conditional distributions is a hard problem, particularly when $\mathbf{x}$ is high dimensional, so if we are just interested in classification then the generative approach may mean that we are trying to solve a harder problem than we need to. However, to deal with missing input values, outliers and unlabelled data points in a principled fashion it is very helpful to have access to $p(\mathbf{x})$, and this can be obtained from marginalizing out the class label y from the joint as $p(\mathbf{x}) = \sum_y p(y)p(\mathbf{x}|y)$ in the generative approach. A further factor in the choice of a generative or discriminative approach could also be which one is most conducive to the incorporation of any prior information which is available. See Ripley [1996, sec. 2.1] for further discussion of these issues. The Gaussian process classifiers developed in this chapter are discriminative.

missing values

3.1.1 Decision Theory for Classification

The classifiers described above provide predictive probabilities $p(y_*|\mathbf{x}_*)$ for a test input $\mathbf{x}_*$. However, sometimes one actually needs to make a decision and to do this we need to consider decision theory. Decision theory for the regression problem was considered in section 2.4; here we discuss decision theory for classification problems. A comprehensive treatment of decision theory can be found in Berger [1985].

C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006, ISBN 026218253X. www.GaussianProcess.org/gpml

loss, risk

Let $\mathcal{L}(c, c')$ be the loss incurred by making decision c' if the true class is $\mathcal{C}_c$. Usually $\mathcal{L}(c, c) = 0$ for all c. The expected loss[5] (or risk) of taking decision c' given $\mathbf{x}$ is $R_{\mathcal{L}}(c'|\mathbf{x}) = \sum_c \mathcal{L}(c, c')p(\mathcal{C}_c|\mathbf{x})$ and the optimal decision c^* is the one that minimizes $R_{\mathcal{L}}(c'|\mathbf{x})$. One common choice of loss function is the zero-one

zero-one loss

loss, where a penalty of one unit is paid for an incorrect classification, and 0 for a correct one. In this case the optimal decision rule is to choose the class $\mathcal{C}_c$ that maximizes[6] $p(\mathcal{C}_c|\mathbf{x})$, as this minimizes the expected error at $\mathbf{x}$. However,

asymmetric loss

the zero-one loss is not always appropriate. A classic example of this is the difference in loss of failing to spot a disease when carrying out a medical test compared to the cost of a false positive on the test, so that $\mathcal{L}(c, c') \neq \mathcal{L}(c', c)$.

Bayes classifier

The optimal classifier (using zero-one loss) is known as the *Bayes classifier*. By this construction the feature space is divided into *decision regions*

decision regions

$\mathcal{R}_1, \ldots, \mathcal{R}_C$ such that a pattern falling in decision region $\mathcal{R}_c$ is assigned to class $\mathcal{C}_c$. (There can be more than one decision region corresponding to a single class.) The boundaries between the decision regions are known as *decision surfaces* or decision boundaries.

One would expect misclassification errors to occur in regions where the maximum class probability $\max_j p(\mathcal{C}_j|\mathbf{x})$ is relatively low. This could be due to either a region of strong overlap between classes, or lack of training examples

reject option

within this region. Thus one sensible strategy is to add a *reject option* so that if $\max_j p(\mathcal{C}_j|\mathbf{x}) \geq \theta$ for a threshold θ in $(0, 1)$ then we go ahead and classify the pattern, otherwise we reject it and leave the classification task to a more sophisticated system. For multi-class classification we could alternatively require the gap between the most probable and the second most probable class to exceed θ, and otherwise reject. As θ is varied from 0 to 1 one obtains an error-reject curve, plotting the percentage of patterns classified incorrectly against the percentage rejected. Typically the error rate will fall as the rejection rate increases. Hansen et al. [1997] provide an analysis of the error-reject trade-off.

We have focused above on the probabilistic approach to classification, which involves a two-stage approach of first computing a posterior distribution over functions and then combining this with the loss function to produce a decision. However, it is worth noting that some authors argue that if our goal is to eventually make a decision then we should aim to approximate the classification

risk minimization

function that minimizes the risk (expected loss), which is defined as

$$R_{\mathcal{L}}(c) \;=\; \int \mathcal{L}\big(y, c(\mathbf{x})\big) p(y, \mathbf{x})\, dy d\mathbf{x}, \tag{3.3}$$

where $p(y, \mathbf{x})$ is the joint distribution of inputs and targets and $c(\mathbf{x})$ is a classification function that assigns an input pattern $\mathbf{x}$ to one of C classes (see e.g. Vapnik [1995, ch. 1]). As $p(y, \mathbf{x})$ is unknown, in this approach one often then seeks to minimize an objective function which includes the empirical risk $\sum_{i=1}^n \mathcal{L}(y_i, c(\mathbf{x}_i))$ as well as a regularization term. While this is a reasonable

[5] In Economics one usually talks of maximizing expected utility rather than minimizing expected loss; loss is negative utility. This suggests that statisticians are pessimists while economists are optimists.

[6] If more than one class has equal posterior probability then ties can be broken arbitrarily.

C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006, ISBN 026218253X. www.GaussianProcess.org/gpml

method, we note that the probabilistic approach allows the same inference stage to be re-used with different loss functions, it can help us to incorporate prior knowledge on the function and/or noise model, and has the advantage of giving probabilistic predictions which can be helpful e.g. for the reject option.

3.2 Linear Models for Classification

In this section we briefly review linear models for binary classification, which form the foundation of Gaussian process classification models in the next section. We follow the SVM literature and use the labels $y = +1$ and $y = -1$ to distinguish the two classes, although for the multi-class case in section 3.5 we use 0/1 labels. The likelihood is

$$p(y{=}+1|\mathbf{x}, \mathbf{w}) \;=\; \sigma(\mathbf{x}^\top \mathbf{w}), \tag{3.4}$$

given the weight vector $\mathbf{w}$ and $\sigma(z)$ can be any sigmoid[7] function. When using the logistic, $\sigma(z) = \lambda(z)$ from eq. (3.2), the model is usually called simply *logistic regression*, but to emphasize the parallels to linear regression we prefer the term *linear logistic regression*. When using the cumulative Gaussian $\sigma(z) = \Phi(z)$, we call the model *linear probit regression*.

linear logistic regression

linear probit regression

As the probability of the two classes must sum to 1, we have $p(y{=}-1|\mathbf{x}, \mathbf{w}) = 1 - p(y = +1|\mathbf{x}, \mathbf{w})$. Thus for a data point $(\mathbf{x}_i, y_i)$ the likelihood is given by $\sigma(\mathbf{x}_i^\top \mathbf{w})$ if $y_i = +1$, and $1 - \sigma(\mathbf{x}_i^\top \mathbf{w})$ if $y_i = -1$. For symmetric likelihood functions, such as the logistic or probit where $\sigma(-z) = 1 - \sigma(z)$, this can be written more concisely as

concise notation

$$p(y_i|\mathbf{x}_i, \mathbf{w}) \;=\; \sigma(y_i f_i), \tag{3.5}$$

where $f_i \triangleq f(\mathbf{x}_i) = \mathbf{x}_i^\top \mathbf{w}$. Defining the logit transformation as $\text{logit}(\mathbf{x}) = \log\big(p(y{=}+1|\mathbf{x})/p(y{=}-1|\mathbf{x})\big)$ we see that the logistic regression model can be written as $\text{logit}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w}$. The $\text{logit}(\mathbf{x})$ function is also called the *log odds ratio*. Generalized linear modelling [McCullagh and Nelder, 1983] deals with the issue of extending linear models to non-Gaussian data scenarios; the logit transformation is the canonical link function for binary data and this choice simplifies the algebra and algorithms.

logit

log odds ratio

Given a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)|i = 1, \ldots, n\}$, we assume that the labels are generated independently, conditional on $f(\mathbf{x})$. Using the same Gaussian prior $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma_p)$ as for regression in eq. (2.4) we then obtain the un-normalized log posterior

$$\log p(\mathbf{w}|X, \mathbf{y}) \;\stackrel{c}{=}\; -\frac{1}{2}\mathbf{w}^\top \Sigma_p^{-1} \mathbf{w} + \sum_{i=1}^{n} \log \sigma(y_i f_i). \tag{3.6}$$

In the linear regression case with Gaussian noise the posterior was Gaussian with mean and covariance as given in eq. (2.8). For classification the posterior

[7] A sigmoid function is a monotonically increasing function mapping from $\mathbb{R}$ to $[0, 1]$. It derives its name from being shaped like a letter S.

does not have a simple analytic form. However, it is easy to show that for some sigmoid functions, such as the logistic and cumulative Gaussian, the log likelihood is a concave function of $\mathbf{w}$ for fixed $\mathcal{D}$. As the quadratic penalty on $\mathbf{w}$ is also concave then the log posterior is a concave function, which means that it is relatively easy to find its unique maximum. The concavity can also be derived from the fact that the Hessian of $\log p(\mathbf{w}|X, \mathbf{y})$ is negative definite (see section A.9 for further details). The standard algorithm for finding the maximum is Newton's method, which in this context is usually called the *iteratively reweighted least squares* (IRLS) algorithm, as described e.g. in McCullagh and Nelder [1983]. However, note that Minka [2003] provides evidence that other optimization methods (e.g. conjugate gradient ascent) may be faster than IRLS.

concavity

unique maximum

IRLS algorithm

properties of maximum likelihood

Notice that a maximum likelihood treatment (corresponding to an unpenalized version of eq. (3.6)) may result in some undesirable outcomes. If the dataset is linearly separable (i.e. if there exists a hyperplane which separates the positive and negative examples) then maximizing the (unpenalized) likelihood will cause $|\mathbf{w}|$ to tend to infinity, However, this will still give predictions in $[0, 1]$ for $p(y = +1|\mathbf{x}, \mathbf{w})$, although these predictions will be "hard" (i.e. zero or one). If the problem is ill-conditioned, e.g. due to duplicate (or linearly dependent) input dimensions, there will be no unique solution.

As an example, consider linear logistic regression in the case where $\mathbf{x}$-space is two dimensional and there is no bias weight so that $\mathbf{w}$ is also two-dimensional. The prior in weight space is Gaussian and for simplicity we have set $\Sigma_p = I$. Contours of the prior $p(\mathbf{w})$ are illustrated in Figure 3.1(a). If we have a data set $\mathcal{D}$ as shown in Figure 3.1(b) then this induces a posterior distribution in weight space as shown in Figure 3.1(c). Notice that the posterior is non-Gaussian and unimodal, as expected. The dataset is not linearly separable but a weight vector in the direction $(1, 1)^\top$ is clearly a reasonable choice, as the posterior distribution shows. To make predictions based the training set $\mathcal{D}$ for a test point $\mathbf{x}_*$ we have

predictions

$$p(y_* = +1|\mathbf{x}_*, \mathcal{D}) \;=\; \int p(y_* = +1|\mathbf{w}, \mathbf{x}_*) p(\mathbf{w}|\mathcal{D})\, d\mathbf{w}, \tag{3.7}$$

integrating the prediction $p(y_* = +1|\mathbf{w}, \mathbf{x}_*) = \sigma(\mathbf{x}_*^\top \mathbf{w})$ over the posterior distribution of weights. This leads to contours of the predictive distribution as shown in Figure 3.1(d). Notice how the contours are bent, reflecting the integration of many different but plausible $\mathbf{w}$'s.

softmax
multiple logistic

In the multi-class case we use the multiple logistic (or softmax) function

$$p(y = \mathcal{C}_c|\mathbf{x}, W) \;=\; \frac{\exp(\mathbf{x}^\top \mathbf{w}_c)}{\sum_{c'} \exp(\mathbf{x}^\top \mathbf{w}_{c'})}, \tag{3.8}$$

where $\mathbf{w}_c$ is the weight vector for class c, and all weight vectors are collected into the matrix W. The corresponding log likelihood is of the form $\sum_{i=1}^{n} \sum_{c=1}^{C} \delta_{c,y_i} [\mathbf{x}_i^\top \mathbf{w}_c - \log(\sum_{c'} \exp(\mathbf{x}_i^\top \mathbf{w}_{c'}))]$. As in the binary case the log likelihood is a concave function of W.

It is interesting to note that in a generative approach where the class-conditional distributions $p(\mathbf{x}|y)$ are Gaussian with the same covariance matrix,

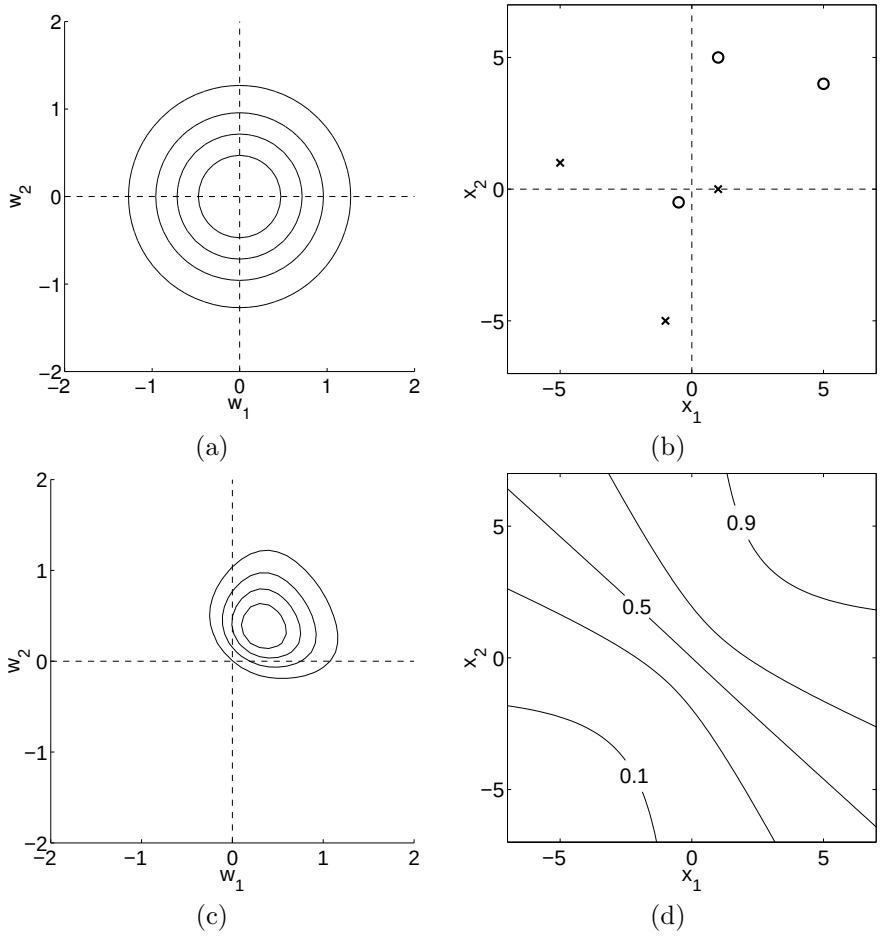


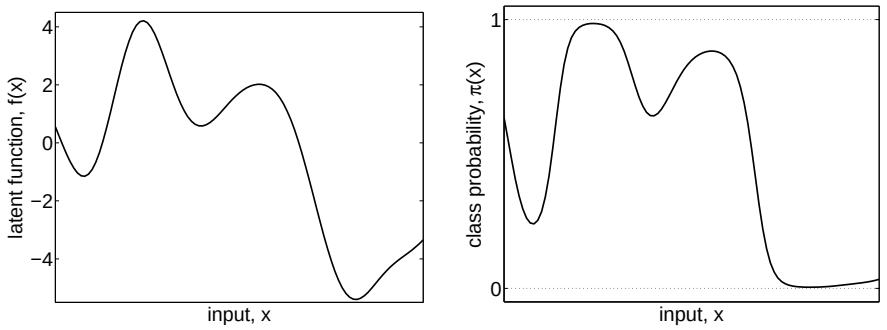
Figure 3.1: Linear logistic regression: Panel (a) shows contours of the prior distribution $p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, I)$. Panel (b) shows the dataset, with circles indicating class $+1$ and crosses denoting class -1. Panel (c) shows contours of the posterior distribution $p(\mathbf{w}|\mathcal{D})$. Panel (d) shows contours of the predictive distribution $p(y_* = +1|\mathbf{x}_*)$.

$p(y|\mathbf{x})$ has the form given by eq. (3.4) and eq. (3.8) for the two- and multi-class cases respectively (when the constant function 1 is included in $\mathbf{x}$).

3.3 Gaussian Process Classification

For binary classification the basic idea behind Gaussian process prediction is very simple—we place a GP prior over the *latent function* $f(\mathbf{x})$ and then "squash" this through the logistic function to obtain a prior on $\pi(\mathbf{x}) \triangleq p(y = +1|\mathbf{x}) = \sigma(f(\mathbf{x}))$. Note that π is a deterministic function of f, and since f is stochastic, so is π. This construction is illustrated in Figure 3.2 for a one-dimensional input space. It is a natural generalization of the linear logistic

latent function



(a) (b)

Figure 3.2: Panel (a) shows a sample latent function $f(x)$ drawn from a Gaussian process as a function of x. Panel (b) shows the result of squashing this sample function through the logistic logit function, $\lambda(z) = (1 + \exp(-z))^{-1}$ to obtain the class probability $\pi(x) = \lambda(f(x))$.

regression model and parallels the development from linear regression to GP regression that we explored in section 2.1. Specifically, we replace the linear $f(\mathbf{x})$ function from the linear logistic model in eq. (3.6) by a Gaussian process, and correspondingly the Gaussian prior on the weights by a GP prior.

nuisance function

The latent function f plays the rôle of a *nuisance function*: we do not observe values of f itself (we observe only the inputs X and the class labels $\mathbf{y}$) and we are not particularly interested in the values of f, but rather in π, in particular for test cases $\pi(\mathbf{x}_*)$. The purpose of f is solely to allow a convenient formulation of the model, and the computational goal pursued in the coming sections will be to remove (integrate out) f.

noise-free latent process

We have tacitly assumed that the latent Gaussian process is *noise-free*, and combined it with smooth likelihood functions, such as the logistic or probit. However, one can equivalently think of adding independent noise to the latent process in combination with a step-function likelihood. In particular, assuming Gaussian noise and a step-function likelihood is exactly equivalent to a noise-free[8] latent process and probit likelihood, see exercise 3.10.1.

Inference is naturally divided into two steps: first computing the distribution of the latent variable corresponding to a test case

$$p(f_*|X, \mathbf{y}, \mathbf{x}_*) \;=\; \int p(f_*|X, \mathbf{x}_*, \mathbf{f}) p(\mathbf{f}|X, \mathbf{y})\, d\mathbf{f}, \tag{3.9}$$

where $p(\mathbf{f}|X, \mathbf{y}) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|X)/p(\mathbf{y}|X)$ is the posterior over the latent variables, and subsequently using this distribution over the latent f_* to produce a probabilistic prediction

$$\bar{\pi}_* \;\triangleq\; p(y_*{=}{+}1|X, \mathbf{y}, \mathbf{x}_*) \;=\; \int \sigma(f_*) p(f_*|X, \mathbf{y}, \mathbf{x}_*)\, df_*. \tag{3.10}$$

[8] This equivalence explains why no numerical problems arise from considering a noise-free process if care is taken with the implementation, see also comment at the end of section 3.4.3.

In the regression case (with Gaussian likelihood) computation of predictions was straightforward as the relevant integrals were Gaussian and could be computed analytically. In classification the non-Gaussian likelihood in eq. (3.9) makes the integral analytically intractable. Similarly, eq. (3.10) can be intractable analytically for certain sigmoid functions, although in the binary case it is only a one-dimensional integral so simple numerical techniques are generally adequate.

Thus we need to use either analytic approximations of integrals, or solutions based on Monte Carlo sampling. In the coming sections, we describe two analytic approximations which both approximate the non-Gaussian joint posterior with a Gaussian one: the first is the straightforward *Laplace approximation* method [Williams and Barber, 1998], and the second is the more sophisticated *expectation propagation* (EP) method due to Minka [2001]. (The cavity TAP approximation of Opper and Winther [2000] is closely related to the EP method.) A number of other approximations have also been suggested, see e.g. Gibbs and MacKay [2000], Jaakkola and Haussler [1999], and Seeger [2000]. Neal [1999] describes the use of Markov chain Monte Carlo (MCMC) approximations. All of these methods will typically scale as $\mathcal{O}(n^3)$; for large datasets there has been much work on further approximations to reduce computation time, as discussed in chapter 8.

The Laplace approximation for the binary case is described in section 3.4, and for the multi-class case in section 3.5. The EP method for binary classification is described in section 3.6. Relationships between Gaussian process classifiers and other techniques such as spline classifiers, support vector machines and least-squares classification are discussed in sections 6.3, 6.4 and 6.5 respectively.

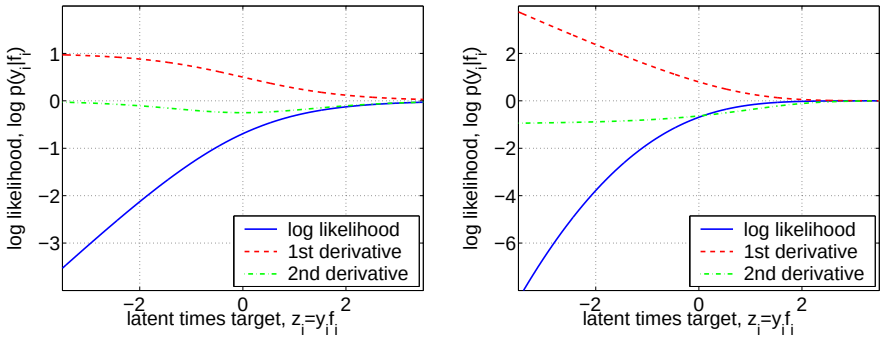
3.4 The Laplace Approximation for the Binary GP Classifier

Laplace's method utilizes a Gaussian approximation $q(\mathbf{f}|X,\mathbf{y})$ to the posterior $p(\mathbf{f}|X,\mathbf{y})$ in the integral (3.9). Doing a second order Taylor expansion of $\log p(\mathbf{f}|X,\mathbf{y})$ around the maximum of the posterior, we obtain a Gaussian approximation

$$q(\mathbf{f}|X,\mathbf{y}) \;=\; \mathcal{N}(\mathbf{f}|\hat{\mathbf{f}},A^{-1}) \;\propto\; \exp\big(-\tfrac{1}{2}(\mathbf{f}-\hat{\mathbf{f}})^\top A(\mathbf{f}-\hat{\mathbf{f}})\big), \tag{3.11}$$

where $\hat{\mathbf{f}} = \mathrm{argmax}_{\mathbf{f}}\, p(\mathbf{f}|X,\mathbf{y})$ and $A = -\nabla\nabla \log p(\mathbf{f}|X,\mathbf{y})|_{\mathbf{f}=\hat{\mathbf{f}}}$ is the Hessian of the negative log posterior at that point.

The structure of the rest of this section is as follows: In section 3.4.1 we describe how to find $\hat{\mathbf{f}}$ and A. Section 3.4.2 explains how to make predictions having obtained $q(\mathbf{f}|\mathbf{y})$, and section 3.4.3 gives more implementation details for the Laplace GP classifier. The Laplace approximation for the marginal likelihood is described in section 3.4.4.



(a), logistic (b), probit

Figure 3.3: The log likelihood and its derivatives for a single case as a function of $z_i = y_i f_i$, for (a) the logistic, and (b) the cumulative Gaussian likelihood. The two likelihood functions are fairly similar, the main qualitative difference being that for large negative arguments the log logistic behaves linearly whereas the log cumulative Gaussian has a quadratic penalty. Both likelihoods are log concave.

3.4.1 Posterior

un-normalized posterior

By Bayes' rule the posterior over the latent variables is given by $p(\mathbf{f}|X,\mathbf{y}) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|X)/p(\mathbf{y}|X)$, but as $p(\mathbf{y}|X)$ is independent of $\mathbf{f}$, we need only consider the un-normalized posterior when maximizing w.r.t. $\mathbf{f}$. Taking the logarithm and introducing expression eq. (2.29) for the GP prior gives

$$\begin{aligned}\Psi(\mathbf{f}) &\triangleq \log p(\mathbf{y}|\mathbf{f}) + \log p(\mathbf{f}|X) \\ &= \log p(\mathbf{y}|\mathbf{f}) - \frac{1}{2}\mathbf{f}^\top K^{-1}\mathbf{f} - \frac{1}{2}\log|K| - \frac{n}{2}\log 2\pi.\end{aligned} \tag{3.12}$$

Differentiating eq. (3.12) w.r.t. $\mathbf{f}$ we obtain

$$\nabla\Psi(\mathbf{f}) = \nabla\log p(\mathbf{y}|\mathbf{f}) - K^{-1}\mathbf{f}, \tag{3.13}$$

$$\nabla\nabla\Psi(\mathbf{f}) = \nabla\nabla\log p(\mathbf{y}|\mathbf{f}) - K^{-1} = -W - K^{-1}, \tag{3.14}$$

where $W \triangleq -\nabla\nabla\log p(\mathbf{y}|\mathbf{f})$ is diagonal, since the likelihood factorizes over cases (the distribution for y_i depends only on f_i, not on $f_{j\neq i}$). Note, that if the likelihood $p(\mathbf{y}|\mathbf{f})$ is log concave, the diagonal elements of W are non-negative, and the Hessian in eq. (3.14) is negative definite, so that $\Psi(\mathbf{f})$ is concave and has a unique maximum (see section A.9 for further details).

log likelihoods and their derivatives

There are many possible functional forms of the likelihood, which gives the target class probability as a function of the latent variable f. Two commonly used likelihood functions are the logistic, and the cumulative Gaussian, see Figure 3.3. The expressions for the log likelihood for these likelihood functions and their first and second derivatives w.r.t. the latent variable are given in the

following table:

$\log p(y_i\|f_i)$	$\frac{\partial}{\partial f_i}\log p(y_i\|f_i)$	$\frac{\partial^2}{\partial f_i^2}\log p(y_i\|f_i)$	
$-\log\big(1+\exp(-y_i f_i)\big)$	$t_i-\pi_i$	$-\pi_i(1-\pi_i)$	(3.15)
$\log\Phi(y_i f_i)$	$\frac{y_i\mathcal{N}(f_i)}{\Phi(y_i f_i)}$	$-\frac{\mathcal{N}(f_i)^2}{\Phi(y_i f_i)^2}-\frac{y_i f_i\mathcal{N}(f_i)}{\Phi(y_i f_i)}$	(3.16)

where we have defined $\pi_i = p(y_i=1|f_i)$ and $\mathbf{t} = (\mathbf{y}+\mathbf{1})/2$. At the maximum of $\Psi(\mathbf{f})$ we have

$$\nabla\Psi \;=\; \mathbf{0} \;\Longrightarrow\; \hat{\mathbf{f}} \;=\; K\big(\nabla\log p(\mathbf{y}|\hat{\mathbf{f}})\big), \tag{3.17}$$

as a self-consistent equation for $\hat{\mathbf{f}}$ (but since $\nabla\log p(\mathbf{y}|\hat{\mathbf{f}})$ is a non-linear function of $\hat{\mathbf{f}}$, eq. (3.17) cannot be solved directly). To find the maximum of Ψ we use Newton's method, with the iteration

Newton's method

$$\begin{aligned}\mathbf{f}^{\text{new}} \;&=\; \mathbf{f}-(\nabla\nabla\Psi)^{-1}\nabla\Psi \;=\; \mathbf{f}+(K^{-1}+W)^{-1}(\nabla\log p(\mathbf{y}|\mathbf{f})-K^{-1}\mathbf{f})\\ &=\; (K^{-1}+W)^{-1}\big(W\mathbf{f}+\nabla\log p(\mathbf{y}|\mathbf{f})\big).\end{aligned} \tag{3.18}$$

To gain more intuition about this update, let us consider what happens to datapoints that are well-explained under $\mathbf{f}$ so that $\partial\log p(y_i|f_i)/\partial f_i$ and W_{ii} are close to zero for these points. As an approximation, break $\mathbf{f}$ into two subvectors, $\mathbf{f}_1$ that corresponds to points that are *not* well-explained, and $\mathbf{f}_2$ to those that are. Then it is easy to show (see exercise 3.10.4) that

$$\begin{aligned}\mathbf{f}_1^{\text{new}} \;&=\; K_{11}(I_{11}+W_{11}K_{11})^{-1}\big(W_{11}\mathbf{f}_1+\nabla\log p(\mathbf{y}_1|\mathbf{f}_1)\big),\\ \mathbf{f}_2^{\text{new}} \;&=\; K_{21}K_{11}^{-1}\mathbf{f}_1^{\text{new}},\end{aligned} \tag{3.19}$$

where K_{21} denotes the $n_2\times n_1$ block of K containing the covariance between the two groups of points, etc. This means that $\mathbf{f}_1^{\text{new}}$ is computed by ignoring entirely the well-explained points, and $\mathbf{f}_2^{\text{new}}$ is predicted from $\mathbf{f}_1^{\text{new}}$ using the usual GP prediction methods (i.e. treating these points like test points). Of course, if the predictions of $\mathbf{f}_2^{\text{new}}$ fail to match the targets correctly they would cease to be well-explained and so be updated on the next iteration.

intuition on influence of well-explained points

Having found the maximum posterior $\hat{\mathbf{f}}$, we can now specify the Laplace approximation to the posterior as a Gaussian with mean $\hat{\mathbf{f}}$ and covariance matrix given by the negative inverse Hessian of Ψ from eq. (3.14)

$$q(\mathbf{f}|X,\mathbf{y}) \;=\; \mathcal{N}\big(\hat{\mathbf{f}},\,(K^{-1}+W)^{-1}\big). \tag{3.20}$$

One problem with the Laplace approximation is that it is essentially uncontrolled, in that the Hessian (evaluated at $\hat{\mathbf{f}}$) may give a poor approximation to the true shape of the posterior. The peak could be much broader or narrower than the Hessian indicates, or it could be a skew peak, while the Laplace approximation assumes it has elliptical contours.

3.4.2 Predictions

latent mean

The posterior mean for f_* under the Laplace approximation can be expressed by combining the GP predictive mean eq. (2.25) with eq. (3.17) into

$$\mathbb{E}_q[f_*|X,\mathbf{y},\mathbf{x}_*] \;=\; \mathbf{k}(\mathbf{x}_*)^\top K^{-1}\hat{\mathbf{f}} \;=\; \mathbf{k}(\mathbf{x}_*)^\top \nabla \log p(\mathbf{y}|\hat{\mathbf{f}}). \tag{3.21}$$

Compare this with the exact mean, given by Opper and Winther [2000] as

$$\begin{aligned}\mathbb{E}_p[f_*|X,\mathbf{y},\mathbf{x}_*] \;&=\; \int \mathbb{E}[f_*|\mathbf{f},X,\mathbf{x}_*]p(\mathbf{f}|X,\mathbf{y})d\mathbf{f} \\ &=\; \int \mathbf{k}(\mathbf{x}_*)^\top K^{-1}\mathbf{f}\; p(\mathbf{f}|X,\mathbf{y})d\mathbf{f} \;=\; \mathbf{k}(\mathbf{x}_*)^\top K^{-1}\mathbb{E}[\mathbf{f}|X,\mathbf{y}],\end{aligned} \tag{3.22}$$

where we have used the fact that for a GP $\mathbb{E}[f_*|\mathbf{f},X,\mathbf{x}_*] = \mathbf{k}(\mathbf{x}_*)^\top K^{-1}\mathbf{f}$ and have let $\mathbb{E}[\mathbf{f}|X,\mathbf{y}]$ denote the posterior mean of $\mathbf{f}$ given X and $\mathbf{y}$. Notice the similarity between the middle expression of eq. (3.21) and eq. (3.22), where the exact (intractable) average $\mathbb{E}[\mathbf{f}|X,\mathbf{y}]$ has been replaced with the modal value $\hat{\mathbf{f}} = \mathbb{E}_q[\mathbf{f}|X,\mathbf{y}]$.

sign of kernel coefficients

A simple observation from eq. (3.21) is that positive training examples will give rise to a positive coefficient for their kernel function (as $\nabla_i \log p(y_i|f_i) > 0$ in this case), while negative examples will give rise to a negative coefficient; this is analogous to the solution to the support vector machine, see eq. (6.34). Also note that training points which have $\nabla_i \log p(y_i|f_i) \simeq 0$ (i.e. that are well-explained under $\hat{\mathbf{f}}$) do not contribute strongly to predictions at novel test points; this is similar to the behaviour of non-support vectors in the support vector machine (see section 6.4).

We can also compute $\mathbb{V}_q[f_*|X,\mathbf{y}]$, the variance of $f_*|X,\mathbf{y}$ under the Gaussian approximation. This comprises of two terms, i.e.

$$\begin{aligned}\mathbb{V}_q[f_*|X,\mathbf{y},\mathbf{x}_*] \;&=\; \mathbb{E}_{p(f_*|X,\mathbf{x}_*,\mathbf{f})}[(f_* - \mathbb{E}[f_*|X,\mathbf{x}_*,\mathbf{f}])^2] \\ &+\; \mathbb{E}_{q(\mathbf{f}|X,\mathbf{y})}[(\mathbb{E}[f_*|X,\mathbf{x}_*,\mathbf{f}] - \mathbb{E}[f_*|X,\mathbf{y},\mathbf{x}_*])^2].\end{aligned} \tag{3.23}$$

The first term is due to the variance of f_* if we condition on a particular value of $\mathbf{f}$, and is given by $k(\mathbf{x}_*,\mathbf{x}_*) - \mathbf{k}(\mathbf{x}_*)^\top K^{-1}\mathbf{k}(\mathbf{x}_*)$, cf. eq. (2.19). The second term in eq. (3.23) is due to the fact that $\mathbb{E}[f_*|X,\mathbf{x}_*,\mathbf{f}] = \mathbf{k}(\mathbf{x}_*)^\top K^{-1}\mathbf{f}$ depends on $\mathbf{f}$ and thus there is an additional term of $\mathbf{k}(\mathbf{x}_*)^\top K^{-1}\,\mathrm{cov}(\mathbf{f}|X,\mathbf{y})K^{-1}\mathbf{k}(\mathbf{x}_*)$.

latent variance

Under the Gaussian approximation $\mathrm{cov}(\mathbf{f}|X,\mathbf{y}) = (K^{-1}+W)^{-1}$, and thus

$$\begin{aligned}\mathbb{V}_q[f_*|X,\mathbf{y},\mathbf{x}_*] \;&=\; k(\mathbf{x}_*,\mathbf{x}_*) - \mathbf{k}_*^\top K^{-1}\mathbf{k}_* + \mathbf{k}_*^\top K^{-1}(K^{-1}+W)^{-1}K^{-1}\mathbf{k}_* \\ &=\; k(\mathbf{x}_*,\mathbf{x}_*) - \mathbf{k}_*^\top (K+W^{-1})^{-1}\mathbf{k}_*,\end{aligned} \tag{3.24}$$

where the last line is obtained using the matrix inversion lemma eq. (A.9).

averaged predictive probability

Given the mean and variance of f_*, we make predictions by computing

$$\bar{\pi}_* \;\simeq\; \mathbb{E}_q[\pi_*|X,\mathbf{y},\mathbf{x}_*] \;=\; \int \sigma(f_*)q(f_*|X,\mathbf{y},\mathbf{x}_*)\,df_*, \tag{3.25}$$

C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006, ISBN 026218253X. www.GaussianProcess.org/gpml

where $q(f_*|X, \mathbf{y}, \mathbf{x}_*)$ is Gaussian with mean and variance given by equations 3.21 and 3.24 respectively. Notice that because of the non-linear form of the sigmoid the predictive probability from eq. (3.25) is different from the sigmoid of the expectation of $\mathbf{f}$: $\hat{\pi}_* = \sigma(\mathbb{E}_q[f_*|\mathbf{y}])$. We will call the latter the *MAP prediction* to distinguish it from the *averaged predictions* from eq. (3.25).

MAP prediction

In fact, as shown in Bishop [1995, sec. 10.3], the predicted test labels given by choosing the class of highest probability obtained by averaged and MAP predictions are identical for *binary*[9] classification. To see this, note that the decision boundary using the the MAP value $\mathbb{E}_q[f_*|X, \mathbf{y}, \mathbf{x}_*]$ corresponds to $\sigma(\mathbb{E}_q[f_*|X, \mathbf{y}, \mathbf{x}_*]) = 1/2$ or $\mathbb{E}_q[f_*|X, \mathbf{y}, \mathbf{x}_*] = 0$. The decision boundary of the averaged prediction, $\mathbb{E}_q[\pi_*|X, \mathbf{y}, \mathbf{x}_*] = 1/2$, also corresponds to $\mathbb{E}_q[f_*|X, \mathbf{y}, \mathbf{x}_*] = 0$. This follows from the fact that $\sigma(f_*) - 1/2$ is antisymmetric while $q(f_*|X, \mathbf{y}, \mathbf{x}_*)$ is symmetric.

identical binary decisions

Thus if we are concerned only about the most probable classification, it is not necessary to compute predictions using eq. (3.25). However, as soon as we also need a confidence in the prediction (e.g. if we are concerned about a reject option) we need $\mathbb{E}_q[\pi_*|X, \mathbf{y}, \mathbf{x}_*]$. If $\sigma(z)$ is the cumulative Gaussian function then eq. (3.25) can be computed analytically, as shown in section 3.9. On the other hand if σ is the logistic function then we need to resort to sampling methods or analytical approximations to compute this one-dimensional integral. One attractive method is to note that the logistic function $\lambda(z)$ is the c.d.f. (cumulative density function) corresponding to the p.d.f. (probability density function) $p(z) = \mathrm{sech}^2(z/2)/4$; this is known as the logistic or sech-squared distribution, see Johnson et al. [1995, ch. 23]. Then by approximating $p(z)$ as a mixture of Gaussians, one can approximate $\lambda(z)$ by a linear combination of error functions. This approximation was used by Williams and Barber [1998, app. A] and Wood and Kohn [1998]. Another approximation suggested in MacKay [1992d] is $\bar{\pi}_* \simeq \lambda(\kappa(f_*|\mathbf{y})\bar{f}_*)$, where $\kappa^2(f_*|\mathbf{y}) = (1 + \pi\mathbb{V}_q[f_*|X, \mathbf{y}, \mathbf{x}_*]/8)^{-1}$. The effect of the latent predictive variance is, as the approximation suggests, to "soften" the prediction that would be obtained using the MAP prediction $\hat{\pi}_* = \lambda(\bar{f}_*)$, i.e. to move it towards 1/2.

3.4.3 Implementation

We give implementations for finding the Laplace approximation in Algorithm 3.1 and for making predictions in Algorithm 3.2. Care is taken to avoid numerically unstable computations while minimizing the computational effort; both can be achieved simultaneously. It turns out that several of the desired terms can be expressed in terms of the symmetric positive definite matrix

$$B = I + W^{\frac{1}{2}} K W^{\frac{1}{2}}, \tag{3.26}$$

computation of which costs only $\mathcal{O}(n^2)$, since W is diagonal. The B matrix has eigenvalues bounded below by 1 and bounded above by $1 + n \max_{ij}(K_{ij})/4$, so for many covariance functions B is guaranteed to be well-conditioned, and it is

[9] For multi-class predictions discussed in section 3.5 the situation is more complicated.

input: K (covariance matrix), $\mathbf{y}$ (± 1 targets), $p(\mathbf{y}|\mathbf{f})$ (likelihood function)
2: $\mathbf{f} := \mathbf{0}$ initialization
repeat Newton iteration
4: $W := -\nabla\nabla \log p(\mathbf{y}|\mathbf{f})$ eval. W e.g. using eq. (3.15) or (3.16)
$L := \text{cholesky}(I + W^{\frac{1}{2}} K W^{\frac{1}{2}})$ $B = I + W^{\frac{1}{2}} K W^{\frac{1}{2}}$
6: $\mathbf{b} := W\mathbf{f} + \nabla \log p(\mathbf{y}|\mathbf{f})$
$\mathbf{a} := \mathbf{b} - W^{\frac{1}{2}} L^\top \backslash (L \backslash (W^{\frac{1}{2}} K \mathbf{b}))$ } eq. (3.18) using eq. (3.27)
8: $\mathbf{f} := K\mathbf{a}$
until convergence objective: $-\frac{1}{2}\mathbf{a}^\top \mathbf{f} + \log p(\mathbf{y}|\mathbf{f})$
10: $\log q(\mathbf{y}|X, \theta) := -\frac{1}{2}\mathbf{a}^\top \mathbf{f} + \log p(\mathbf{y}|\mathbf{f}) - \sum_i \log L_{ii}$ eq. (3.32)
return: $\hat{\mathbf{f}} := \mathbf{f}$ (post. mode), $\log q(\mathbf{y}|X, \theta)$ (approx. log marg. likelihood)

Algorithm 3.1: Mode-finding for binary Laplace GPC. Commonly used convergence criteria depend on the difference in successive values of the objective function $\Psi(\mathbf{f})$ from eq. (3.12), the magnitude of the gradient vector $\nabla\Psi(\mathbf{f})$ from eq. (3.13) and/or the magnitude of the difference in successive values of $\mathbf{f}$. In a practical implementation one needs to secure against divergence by checking that each iteration leads to an increase in the objective (and trying a smaller step size if not). The computational complexity is dominated by the Cholesky decomposition in line 5 which takes $n^3/6$ operations (times the number of Newton iterations), all other operations are at most quadratic in n.

thus numerically safe to compute its Cholesky decomposition $LL^\top = B$, which is useful in computing terms involving B^{-1} and $|B|$.

The mode-finding procedure uses the Newton iteration given in eq. (3.18), involving the matrix $(K^{-1}+W)^{-1}$. Using the matrix inversion lemma eq. (A.9) we get

$$(K^{-1} + W)^{-1} \;=\; K - KW^{\frac{1}{2}} B^{-1} W^{\frac{1}{2}} K, \tag{3.27}$$

where B is given in eq. (3.26). The advantage is that whereas K may have eigenvalues arbitrarily close to zero (and thus be numerically unstable to invert), we can safely work with B. In addition, Algorithm 3.1 keeps the vector $\mathbf{a} = K^{-1}\mathbf{f}$ in addition to $\mathbf{f}$, as this allows evaluation of the part of the objective $\Psi(\mathbf{f})$ in eq. (3.12) which depends on $\mathbf{f}$ without explicit reference to K^{-1} (again to avoid possible numerical problems).

Similarly, for the computation of the predictive variance $\mathbb{V}_q[f_*|\mathbf{y}]$ from eq. (3.24) we need to evaluate a quadratic form involving the matrix $(K + W^{-1})^{-1}$. Rewriting this as

$$(K + W^{-1})^{-1} \;=\; W^{\frac{1}{2}} W^{-\frac{1}{2}} (K + W^{-1})^{-1} W^{-\frac{1}{2}} W^{\frac{1}{2}} \;=\; W^{\frac{1}{2}} B^{-1} W^{\frac{1}{2}} \tag{3.28}$$

achieves numerical stability (as opposed to inverting W itself, which may have arbitrarily small eigenvalues). Thus the predictive variance from eq. (3.24) can be computed as

$$\begin{aligned} \mathbb{V}_q[f_*|\mathbf{y}] \;&= k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}(\mathbf{x}_*)^\top W^{\frac{1}{2}} (LL^\top)^{-1} W^{\frac{1}{2}} \mathbf{k}(\mathbf{x}_*) \\ &= k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{v}^\top \mathbf{v}, \quad \text{where} \quad \mathbf{v} \;=\; L \backslash (W^{\frac{1}{2}} \mathbf{k}(\mathbf{x}_*)), \end{aligned} \tag{3.29}$$

which was also used by Seeger [2003, p. 27].

input: $\hat{\mathbf{f}}$ (mode), X (inputs), $\mathbf{y}$ (± 1 targets), k (covariance function), $p(\mathbf{y}|\mathbf{f})$ (likelihood function), $\mathbf{x}_*$ test input

2: $W := -\nabla\nabla \log p(\mathbf{y}|\hat{\mathbf{f}})$

$L := \text{cholesky}(I + W^{\frac{1}{2}} K W^{\frac{1}{2}})$ $\quad B = I + W^{\frac{1}{2}} K W^{\frac{1}{2}}$

4: $\bar{f}_* := \mathbf{k}(\mathbf{x}_*)^\top \nabla \log p(\mathbf{y}|\hat{\mathbf{f}})$ $\quad$ eq. (3.21)

$\mathbf{v} := L \backslash \big(W^{\frac{1}{2}} \mathbf{k}(\mathbf{x}_*)\big)$

6: $\mathbb{V}[f_*] := k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{v}^\top \mathbf{v}$ $\quad \Big\}$ eq. (3.24) using eq. (3.29)

$\bar{\pi}_* := \int \sigma(z) \mathcal{N}(z|\bar{f}_*, \mathbb{V}[f_*]) dz$ $\quad$ eq. (3.25)

8: **return**: $\bar{\pi}_*$ (predictive class probability (for class 1))

Algorithm 3.2: Predictions for binary Laplace GPC. The posterior mode $\hat{\mathbf{f}}$ (which can be computed using Algorithm 3.1) is input. For multiple test inputs lines $4-7$ are applied to each test input. Computational complexity is $n^3/6$ operations once (line 3) plus n^2 operations per test case (line 5). The one-dimensional integral in line 7 can be done analytically for cumulative Gaussian likelihood, otherwise it is computed using an approximation or numerical quadrature.

In practice we compute the Cholesky decomposition $LL^\top = B$ during the Newton steps in Algorithm 3.1, which can be re-used to compute the predictive variance by doing backsubstitution with L as discussed above. In addition, L may again be re-used to compute $|I_n + W^{\frac{1}{2}} K W^{\frac{1}{2}}| = |B|$ (needed for the computation of the marginal likelihood eq. (3.32)) as $\log |B| = 2 \sum \log L_{ii}$. To save computation, one could use an incomplete Cholesky factorization in the Newton steps, as suggested by Fine and Scheinberg [2002]. incomplete Cholesky factorization

Sometimes it is suggested that it can be useful to replace K by $K + \epsilon I$ where ϵ is a small constant, to improve the numerical conditioning[10] of K. However, by taking care with the implementation details as above this should not be necessary.

3.4.4 Marginal Likelihood

It will also be useful (particularly for chapter 5) to compute the Laplace approximation of the marginal likelihood $p(\mathbf{y}|X)$. (For the regression case with Gaussian noise the marginal likelihood can again be calculated analytically, see eq. (2.30).) We have

$$p(\mathbf{y}|X) \;=\; \int p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}|X)\, d\mathbf{f} \;=\; \int \exp\big(\Psi(\mathbf{f})\big)\, d\mathbf{f}. \qquad (3.30)$$

Using a Taylor expansion of $\Psi(\mathbf{f})$ locally around $\hat{\mathbf{f}}$ we obtain $\Psi(\mathbf{f}) \simeq \Psi(\hat{\mathbf{f}}) - \frac{1}{2}(\mathbf{f} - \hat{\mathbf{f}})^\top A (\mathbf{f} - \hat{\mathbf{f}})$ and thus an approximation $q(\mathbf{y}|X)$ to the marginal likelihood as

$$p(\mathbf{y}|X) \;\simeq\; q(\mathbf{y}|X) \;=\; \exp\big(\Psi(\hat{\mathbf{f}})\big) \int \exp\big(-\tfrac{1}{2}(\mathbf{f} - \hat{\mathbf{f}})^\top A (\mathbf{f} - \hat{\mathbf{f}})\big)\, d\mathbf{f}. \qquad (3.31)$$

[10] Neal [1999] refers to this as adding "jitter" in the context of Markov chain Monte Carlo (MCMC) based inference; in his work the latent variables $\mathbf{f}$ are explicitly represented in the Markov chain which makes addition of jitter difficult to avoid. Within the analytical approximations of the *distribution* of $\mathbf{f}$ considered here, jitter is unnecessary.

This Gaussian integral can be evaluated analytically to obtain an approximation to the log marginal likelihood

$$\log q(\mathbf{y}|X,\boldsymbol{\theta}) \;=\; -\tfrac{1}{2}\hat{\mathbf{f}}^\top K^{-1}\hat{\mathbf{f}} + \log p(\mathbf{y}|\hat{\mathbf{f}}) - \tfrac{1}{2}\log|B|, \tag{3.32}$$

where $|B| = |K|\cdot|K^{-1}+W| = |I_n + W^{\frac{1}{2}}KW^{\frac{1}{2}}|$, and $\boldsymbol{\theta}$ is a vector of hyperparameters of the covariance function (which have previously been suppressed from the notation for brevity).

∗ 3.5 Multi-class Laplace Approximation

Our presentation follows Williams and Barber [1998]. We first introduce the vector of latent function values at all n training points and for all C classes

$$\mathbf{f} \;=\; \big(f_1^1,\ldots,f_n^1,f_1^2,\ldots,f_n^2,\ldots,f_1^C,\ldots,f_n^C\big)^\top. \tag{3.33}$$

Thus $\mathbf{f}$ has length Cn. In the following we will generally refer to quantities pertaining to a particular class with superscript c, and a particular case by subscript i (as usual); thus e.g. the vector of C latents for a particular case is $\mathbf{f}_i$. However, as an exception, vectors or matrices formed from the covariance function for class c will have a subscript c. The prior over $\mathbf{f}$ has the form $\mathbf{f}\sim\mathcal{N}(\mathbf{0},K)$. As we have assumed that the C latent processes are uncorrelated, the covariance matrix K is block diagonal in the matrices $K_1,\ldots,K_C$. Each individual matrix K_c expresses the correlations of the latent function values within the class c. Note that the covariance functions pertaining to the different classes can be different. Let $\mathbf{y}$ be a vector of the same length as $\mathbf{f}$ which for each $i=1,\ldots,n$ has an entry of 1 for the class which is the label for example i and 0 for the other $C-1$ entries.

softmax

Let π_i^c denote output of the softmax at training point i, i.e.

$$p(y_i^c|\mathbf{f}_i) \;=\; \pi_i^c \;=\; \frac{\exp(f_i^c)}{\sum_{c'}\exp(f_i^{c'})}. \tag{3.34}$$

un-normalized posterior

Then $\boldsymbol{\pi}$ is a vector of the same length as $\mathbf{f}$ with entries π_i^c. The multi-class analogue of eq. (3.12) is the log of the un-normalized posterior

$$\Psi(\mathbf{f}) \;\triangleq\; -\tfrac{1}{2}\mathbf{f}^\top K^{-1}\mathbf{f} + \mathbf{y}^\top\mathbf{f} - \sum_{i=1}^{n}\log\Big(\sum_{c=1}^{C}\exp f_i^c\Big) - \tfrac{1}{2}\log|K| - \tfrac{Cn}{2}\log 2\pi. \tag{3.35}$$

As in the binary case we seek the MAP value $\hat{\mathbf{f}}$ of $p(\mathbf{f}|X,\mathbf{y})$. By differentiating eq. (3.35) w.r.t. $\mathbf{f}$ we obtain

$$\nabla\Psi \;=\; -K^{-1}\mathbf{f} + \mathbf{y} - \boldsymbol{\pi}. \tag{3.36}$$

Thus at the maximum we have $\hat{\mathbf{f}} = K(\mathbf{y}-\hat{\boldsymbol{\pi}})$. Differentiating again, and using

$$-\frac{\partial^2}{\partial f_i^c\partial f_i^{c'}}\log\sum_j\exp(f_i^j) \;=\; \pi_i^c\delta_{cc'} + \pi_i^c\pi_i^{c'}, \tag{3.37}$$

we obtain[11]

$$\nabla\nabla\Psi \;=\; -K^{-1} - W, \;\; \text{where} \;\; W \;\triangleq\; \operatorname{diag}(\boldsymbol{\pi}) - \Pi\Pi^\top, \tag{3.38}$$

where Π is a $Cn \times n$ matrix obtained by stacking vertically the diagonal matrices $\operatorname{diag}(\boldsymbol{\pi}^c)$, and $\boldsymbol{\pi}^c$ is the subvector of $\boldsymbol{\pi}$ pertaining to class c. As in the binary case notice that $-\nabla\nabla\Psi$ is positive definite, thus $\Psi(\mathbf{f})$ is concave and the maximum is unique (see also exercise 3.10.2).

As in the binary case we use Newton's method to search for the mode of Ψ, giving

$$\mathbf{f}^{\text{new}} \;=\; (K^{-1} + W)^{-1}(W\mathbf{f} + \mathbf{y} - \boldsymbol{\pi}). \tag{3.39}$$

This update if coded naïvely would take $\mathcal{O}(C^3n^3)$ as matrices of size Cn have to be inverted. However, as described in section 3.5.1, we can utilize the structure of W to bring down the computational load to $\mathcal{O}(Cn^3)$.

The Laplace approximation gives us a Gaussian approximation $q(\mathbf{f}|X,\mathbf{y})$ to the posterior $p(\mathbf{f}|X,\mathbf{y})$. To make predictions at a test point $\mathbf{x}_*$ we need to compute the posterior distribution $q(\mathbf{f}_*|X,\mathbf{y},\mathbf{x}_*)$ where $\mathbf{f}(\mathbf{x}_*) \triangleq \mathbf{f}_* = (f_*^1,\ldots,f_*^C)^\top$. In general we have

predictive distribution for f_*

$$q(\mathbf{f}_*|X,\mathbf{y},\mathbf{x}_*) \;=\; \int p(\mathbf{f}_*|X,\mathbf{x}_*,\mathbf{f})q(\mathbf{f}|X,\mathbf{y})\,d\mathbf{f}. \tag{3.40}$$

As $p(\mathbf{f}_*|X,\mathbf{x}_*,\mathbf{f})$ and $q(\mathbf{f}|X,\mathbf{y})$ are both Gaussian, $q(\mathbf{f}_*|X,\mathbf{y},\mathbf{x}_*)$ will also be Gaussian and we need only compute its mean and covariance. The predictive mean for class c is given by

$$\mathbb{E}_q[f^c(\mathbf{x}_*)|X,\mathbf{y},\mathbf{x}_*] \;=\; \mathbf{k}_c(\mathbf{x}_*)^\top K_c^{-1}\hat{\mathbf{f}}^c \;=\; \mathbf{k}_c(\mathbf{x}_*)^\top(\mathbf{y}^c - \hat{\boldsymbol{\pi}}^c), \tag{3.41}$$

where $\mathbf{k}_c(\mathbf{x}_*)$ is the vector of covariances between the test point and each of the training points for the cth covariance function, and $\hat{\mathbf{f}}^c$ is the subvector of $\hat{\mathbf{f}}$ pertaining to class c. The last equality comes from using eq. (3.36) at the maximum $\hat{\mathbf{f}}$. Note the close correspondence to eq. (3.21). This can be put into a vector form $\mathbb{E}_q[\mathbf{f}_*|\mathbf{y}] = Q_*^\top(\mathbf{y} - \hat{\boldsymbol{\pi}})$ by defining the $Cn \times C$ matrix

$$Q_* \;=\; \begin{pmatrix} \mathbf{k}_1(\mathbf{x}_*) & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \mathbf{k}_2(\mathbf{x}_*) & \ldots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \ldots & \mathbf{k}_C(\mathbf{x}_*) \end{pmatrix}. \tag{3.42}$$

Using a similar argument to eq. (3.23) we obtain

$$\begin{aligned} \operatorname{cov}_q(\mathbf{f}_*|X,\mathbf{y},\mathbf{x}_*) \;&= \Sigma + Q_*^\top K^{-1}(K^{-1}+W)^{-1}K^{-1}Q_* \\ &= \operatorname{diag}(\mathbf{k}(\mathbf{x}_*,\mathbf{x}_*)) - Q_*^\top(K + W^{-1})^{-1}Q_*, \end{aligned} \tag{3.43}$$

where Σ is a diagonal $C \times C$ matrix with $\Sigma_{cc} = k_c(\mathbf{x}_*,\mathbf{x}_*) - \mathbf{k}_c^\top(\mathbf{x}_*)K_c^{-1}\mathbf{k}_c(\mathbf{x}_*)$, and $\mathbf{k}(\mathbf{x}_*,\mathbf{x}_*)$ is a vector of covariances, whose c'th element is $k_c(\mathbf{x}_*,\mathbf{x}_*)$.

[11] There is a sign error in equation 23 of Williams and Barber [1998] but not in their implementation.

input: K (covariance matrix), $\mathbf{y}$ (0/1 targets)
2: $\mathbf{f} := \mathbf{0}$ initialization
repeat Newton iteration
4: compute $\boldsymbol{\pi}$ and Π from $\mathbf{f}$ with eq. (3.34) and defn. of Π under eq. (3.38)
for $c := 1 \ldots C$ **do**
6: $L := \text{cholesky}(I_n + D_c^{\frac{1}{2}} K_c D_c^{\frac{1}{2}})$
$E_c := D_c^{\frac{1}{2}} L^\top \backslash (L \backslash D_c^{\frac{1}{2}})$ E is block diag. $D^{\frac{1}{2}}(I_{Cn} + D^{\frac{1}{2}} K D^{\frac{1}{2}})^{-1} D^{\frac{1}{2}}$
8: $z_c := \sum_i \log L_{ii}$ compute $\frac{1}{2}$ log determinant
end for
10: $M := \text{cholesky}(\sum_c E_c)$
$\mathbf{b} := (D - \Pi\Pi^\top)\mathbf{f} + \mathbf{y} - \boldsymbol{\pi}$ $\mathbf{b} = W\mathbf{f} + \mathbf{y} - \boldsymbol{\pi}$ from eq. (3.39)
12: $\mathbf{c} := EK\mathbf{b}$
$\mathbf{a} := \mathbf{b} - \mathbf{c} + ERM^\top \backslash (M \backslash (R^\top \mathbf{c}))$ } eq. (3.39) using eq. (3.45) and (3.47)
14: $\mathbf{f} := K\mathbf{a}$
until convergence objective: $-\frac{1}{2}\mathbf{a}^\top \mathbf{f} + \mathbf{y}^\top \mathbf{f} + \sum_i \log\big(\sum_c \exp(f_c^i)\big)$
16: $\log q(\mathbf{y}|X,\theta) := -\frac{1}{2}\mathbf{a}^\top \mathbf{f} + \mathbf{y}^\top \mathbf{f} + \sum_i \log\big(\sum_c \exp(f_c^i)\big) - \sum_c z_c$ eq. (3.44)
return: $\hat{\mathbf{f}} := \mathbf{f}$ (post. mode), $\log q(\mathbf{y}|X,\theta)$ (approx. log marg. likelihood)

Algorithm 3.3: Mode-finding for multi-class Laplace GPC, where $D = \text{diag}(\boldsymbol{\pi})$, R is a matrix of stacked identity matrices and a subscript c on a block diagonal matrix indicates the $n \times n$ submatrix pertaining to class c. The computational complexity is dominated by the Cholesky decomposition in lines 6 and 10 and the forward and backward substitutions in line 7 with total complexity $\mathcal{O}((C+1)n^3)$ (times the number of Newton iterations), all other operations are at most $\mathcal{O}(Cn^2)$ when exploiting diagonal and block diagonal structures. The memory requirement is $\mathcal{O}(Cn^2)$. For comments on convergence criteria for line 15 and avoiding divergence, refer to the caption of Algorithm 3.1 on page 46.

We now need to consider the predictive distribution $q(\boldsymbol{\pi}_*|\mathbf{y})$ which is obtained by softmaxing the Gaussian $q(\mathbf{f}_*|\mathbf{y})$. In the binary case we saw that the predicted classification could be obtained by thresholding the mean value of the Gaussian. In the multi-class case one *does* need to take the variability around the mean into account as it can affect the overall classification (see exercise 3.10.3). One simple way (which will be used in Algorithm 3.4) to estimate the mean prediction $\mathbb{E}_q[\boldsymbol{\pi}_*|\mathbf{y}]$ is to draw samples from the Gaussian $q(\mathbf{f}_*|\mathbf{y})$, softmax them and then average.

marginal likelihood

The Laplace approximation to the marginal likelihood can be obtained in the same way as for the binary case, yielding

$$\log p(\mathbf{y}|X,\boldsymbol{\theta}) \simeq \log q(\mathbf{y}|X,\boldsymbol{\theta}) \tag{3.44}$$
$$= -\tfrac{1}{2}\hat{\mathbf{f}}^\top K^{-1}\hat{\mathbf{f}} + \mathbf{y}^\top \hat{\mathbf{f}} - \sum_{i=1}^{n} \log\Big(\sum_{c=1}^{C} \exp \hat{f}_i^c\Big) - \tfrac{1}{2}\log|I_{Cn} + W^{\frac{1}{2}} K W^{\frac{1}{2}}|.$$

As for the inversion of $K^{-1} + W$, the determinant term can be computed efficiently by exploiting the structure of W, see section 3.5.1.

In this section we have described the Laplace approximation for multi-class classification. However, there has also been some work on EP-type methods for the multi-class case, see Seeger and Jordan [2004].

input: K (covariance matrix), $\hat{\mathbf{f}}$ (posterior mode), $\mathbf{x}_*$ (test input)
compute $\boldsymbol{\pi}$ and Π from $\hat{\mathbf{f}}$ using eq. (3.34) and defn. of Π under eq. (3.38)
for $c := 1 \ldots C$ **do**
  $L := \text{cholesky}(I_n + D_c^{\frac{1}{2}} K_c D_c^{\frac{1}{2}})$
  $E_c := D_c^{\frac{1}{2}} L^\top \backslash (L \backslash D_c^{\frac{1}{2}})$ E is block diag. $D^{\frac{1}{2}}(I_{Cn} + D^{\frac{1}{2}} K D^{\frac{1}{2}})^{-1} D^{\frac{1}{2}}$
end for
$M := \text{cholesky}(\sum_c E_c)$
for $c := 1 \ldots C$ **do**
  $\boldsymbol{\mu}_*^c := (\mathbf{y}^c - \boldsymbol{\pi}^c)^\top \mathbf{k}_*^c$ latent test mean from eq. (3.41)
  $\mathbf{b} := E_c \mathbf{k}_*^c$
  $\mathbf{c} := E_c(R(M^\top \backslash (M \backslash (R^\top \mathbf{b}))))$
  for $c' := 1 \ldots C$ **do**
    $\Sigma_{cc'} := \mathbf{c}^\top \mathbf{k}_*^{c'}$
  end for
  $\Sigma_{cc} := \Sigma_{cc} + k_c(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{b}^\top \mathbf{k}_*^c$ } latent test covariance from eq. (3.43)
end for
$\boldsymbol{\pi}_* := \mathbf{0}$ initialize Monte Carlo loop to estimate
for $i := 1 : S$ **do** predictive class probabilities using S samples
  $\mathbf{f}_* \sim \mathcal{N}(\boldsymbol{\mu}_*, \Sigma)$ sample latent values from joint Gaussian posterior
  $\boldsymbol{\pi}_* := \boldsymbol{\pi}_* + \exp(f_*^c) / \sum_{c'} \exp(f_*^{c'})$ accumulate probability eq. (3.34)
end for
$\bar{\boldsymbol{\pi}}_* := \boldsymbol{\pi}_* / S$ normalize MC estimate of prediction vector
return: $\mathbb{E}_{q(\mathbf{f})}[\boldsymbol{\pi}(\mathbf{f}(\mathbf{x}_*)) | \mathbf{x}_*, X, \mathbf{y}] := \bar{\boldsymbol{\pi}}_*$ (predicted class probability vector)

Algorithm 3.4: Predictions for multi-class Laplace GPC, where $D = \text{diag}(\boldsymbol{\pi})$, R is a matrix of stacked identity matrices and a subscript c on a block diagonal matrix indicates the $n \times n$ submatrix pertaining to class c. The computational complexity is dominated by the Cholesky decomposition in lines 4 and 7 with a total complexity $\mathcal{O}((C+1)n^3)$, the memory requirement is $\mathcal{O}(Cn^2)$. For multiple test cases repeat from line 8 for each test case (in practice, for multiple test cases one may reorder the computations in lines 8-16 to avoid referring to all E_c matrices repeatedly).

3.5.1 Implementation

The implementation follows closely the implementation for the binary case detailed in section 3.4.3, with the slight complications that K is now a block diagonal matrix of size $Cn \times Cn$ and the W matrix is no longer diagonal, see eq. (3.38). Care has to be taken to exploit the structure of these matrices to reduce the computational burden.

The Newton iteration from eq. (3.39) requires the inversion of $K^{-1} + W$, which we first re-write as

$$(K^{-1} + W)^{-1} \;=\; K - K(K + W^{-1})^{-1} K, \tag{3.45}$$

using the matrix inversion lemma, eq. (A.9). In the following the inversion of the above matrix $K + W^{-1}$ is our main concern. First, however, we apply the

matrix inversion lemma, eq. (A.9) to the W matrix:[12]

$$\begin{aligned} W^{-1} \;=\; (D - \Pi\Pi^\top)^{-1} \;&=\; D^{-1} - R(I - R^\top D R)^{-1} R^\top \\ &=\; D^{-1} - R O^{-1} R^\top, \end{aligned} \tag{3.46}$$

where $D = \mathrm{diag}(\boldsymbol{\pi})$, $R = D^{-1}\Pi$ is a $Cn \times n$ matrix of stacked I_n unit matrices, we use the fact that $\boldsymbol{\pi}$ normalizes over classes: $R^\top D R = \sum_c D_c = I_n$ and O is the zero matrix. Introducing the above in $K + W^{-1}$ and applying the matrix inversion lemma, eq. (A.9) again we have

$$\begin{aligned} (K + W^{-1})^{-1} \;&=\; (K + D^{-1} - R O^{-1} R^\top)^{-1} \\ &=\; E - ER(O + R^\top E R)^{-1} R^\top E \;=\; E - ER(\textstyle\sum_c E_c)^{-1} R^\top E. \end{aligned} \tag{3.47}$$

where $E = (K + D^{-1})^{-1} = D^{\frac{1}{2}}(I + D^{\frac{1}{2}} K D^{\frac{1}{2}})^{-1} D^{\frac{1}{2}}$ is a block diagonal matrix and $R^\top E R = \sum_c E_c$. The Newton iterations can now be computed by inserting eq. (3.47) and (3.45) in eq. (3.39), as detailed in Algorithm 3.3. The predictions use an equivalent route to compute the Gaussian posterior, and the final step of deriving predictive class probabilities is done by Monte Carlo, as shown in Algorithm 3.4.

3.6 Expectation Propagation

The expectation propagation (EP) algorithm [Minka, 2001] is a general approximation tool with a wide range of applications. In this section we present only its application to the specific case of a GP model for binary classification. We note that Opper and Winther [2000] presented a similar method for binary GPC based on the fixed-point equations of the Thouless-Anderson-Palmer (TAP) type of mean-field approximation from statistical physics. The fixed points for the two methods are the same, although the precise details of the two algorithms are different. The EP algorithm naturally lends itself to sparse approximations, which will not be discussed in detail here, but touched upon in section 8.4.

The object of central importance is the posterior distribution over the latent variables, $p(\mathbf{f}|X, \mathbf{y})$. In the following notation we suppress the explicit dependence on hyperparameters, see section 3.6.2 for their treatment. The posterior is given by Bayes' rule, as the product of a normalization term, the prior and the likelihood

$$p(\mathbf{f}|X, \mathbf{y}) \;=\; \frac{1}{Z} p(\mathbf{f}|X) \prod_{i=1}^{n} p(y_i|f_i), \tag{3.48}$$

where the prior $p(\mathbf{f}|X)$ is Gaussian and we have utilized the fact that the likelihood factorizes over the training cases. The normalization term is the marginal likelihood

$$Z \;=\; p(\mathbf{y}|X) \;=\; \int p(\mathbf{f}|X) \prod_{i=1}^{n} p(y_i|f_i)\, d\mathbf{f}. \tag{3.49}$$

[12] Readers who are disturbed by our sloppy treatment of the inverse of singular matrices are invited to insert the matrix $(1 - \varepsilon)I_n$ between Π and $\Pi^\top$ in eq. (3.46) and verify that eq. (3.47) coincides with the limit $\varepsilon \to 0$.

C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006, ISBN 026218253X. www.GaussianProcess.org/gpml

So far, everything is exactly as in the regression case discussed in chapter 2. However, in the case of classification the likelihood $p(y_i|f_i)$ is not Gaussian, a property that was used heavily in arriving at analytical solutions for the regression framework. In this section we use the probit likelihood (see page 35) for binary classification

$$p(y_i|f_i) \;=\; \Phi(f_i y_i), \tag{3.50}$$

and this makes the posterior in eq. (3.48) analytically intractable. To overcome this hurdle in the EP framework we approximate the likelihood by a *local likelihood approximation*[13] in the form of an un-normalized Gaussian function in the latent variable f_i

$$p(y_i|f_i) \;\simeq\; t_i(f_i|\tilde{Z}_i, \tilde{\mu}_i, \tilde{\sigma}_i^2) \;\triangleq\; \tilde{Z}_i \mathcal{N}(f_i|\tilde{\mu}_i, \tilde{\sigma}_i^2), \tag{3.51}$$

which defines the *site parameters* $\tilde{Z}_i$, $\tilde{\mu}_i$ and $\tilde{\sigma}_i^2$. Remember that the notation $\mathcal{N}$ is used for a normalized Gaussian distribution. Notice that we are approximating the likelihood, i.e. a probability distribution which normalizes over the *targets* y_i, by an un-normalized Gaussian distribution over the *latent* variables f_i. This is reasonable, because we are interested in how the likelihood behaves as a function of the latent f_i. In the regression setting we utilized the Gaussian shape of the likelihood, but more to the point, the Gaussian distribution for the outputs y_i also implied a Gaussian shape as a function of the latent variable f_i. In order to compute the posterior we are of course primarily interested in how the likelihood behaves as a function of f_i.[14] The property that the likelihood should normalize over y_i (for any value of f_i) is not simultaneously achievable with the desideratum of Gaussian dependence on f_i; in the EP approximation we abandon exact normalization for tractability. The product of the (independent) local likelihoods t_i is

site parameters

$$\prod_{i=1}^{n} t_i(f_i|\tilde{Z}_i, \tilde{\mu}_i, \tilde{\sigma}_i^2) \;=\; \mathcal{N}(\tilde{\boldsymbol{\mu}}, \tilde{\Sigma}) \prod_i \tilde{Z}_i, \tag{3.52}$$

where $\tilde{\boldsymbol{\mu}}$ is the vector of $\tilde{\mu}_i$ and $\tilde{\Sigma}$ is diagonal with $\tilde{\Sigma}_{ii} = \tilde{\sigma}_i^2$. We approximate the posterior $p(\mathbf{f}|X, \mathbf{y})$ by $q(\mathbf{f}|X, \mathbf{y})$

$$\begin{aligned} q(\mathbf{f}|X, \mathbf{y}) \;&\triangleq\; \frac{1}{Z_{\text{EP}}} p(\mathbf{f}|X) \prod_{i=1}^{n} t_i(f_i|\tilde{Z}_i, \tilde{\mu}_i, \tilde{\sigma}_i^2) \;=\; \mathcal{N}(\boldsymbol{\mu}, \Sigma), \\ \text{with } \; \boldsymbol{\mu} \;&=\; \Sigma \tilde{\Sigma}^{-1} \tilde{\boldsymbol{\mu}}, \;\; \text{and } \; \Sigma \;=\; (K^{-1} + \tilde{\Sigma}^{-1})^{-1}, \end{aligned} \tag{3.53}$$

where we have used eq. (A.7) to compute the product (and by definition, we know that the distribution must normalize correctly over $\mathbf{f}$). Notice, that we use the tilde-parameters $\tilde{\boldsymbol{\mu}}$ and $\tilde{\Sigma}$ (and $\tilde{Z}$) for the *local likelihood approximations*,

[13] Note, that although each likelihood approximation is *local*, the posterior approximation produced by the EP algorithm is *global* because the latent variables are coupled through the prior.

[14] However, for computing the *marginal likelihood* normalization becomes crucial, see section 3.6.2.

and plain $\boldsymbol{\mu}$ and Σ for the parameters of the *approximate posterior*. The normalizing term of eq. (3.53), $Z_{\text{EP}} = q(\mathbf{y}|X)$, is the EP algorithm's approximation to the normalizing term Z from eq. (3.48) and eq. (3.49).

KL divergence

How do we choose the parameters of the local approximating distributions t_i? One of the most obvious ideas would be to minimize the Kullback-Leibler (KL) divergence (see section A.5) between the posterior and its approximation: $\text{KL}\big(p(\mathbf{f}|X,\mathbf{y})||q(\mathbf{f}|X,\mathbf{y})\big)$. Direct minimization of this KL divergence for the joint distribution on $\mathbf{f}$ turns out to be intractable. (One can alternatively choose to minimize the reversed KL divergence $\text{KL}\big(q(\mathbf{f}|X,\mathbf{y})||p(\mathbf{f}|X,\mathbf{y})\big)$ with respect to the distribution $q(\mathbf{f}|X,\mathbf{y})$; this has been used to carry out variational inference for GPC, see, e.g. Seeger [2000].)

Instead, the key idea in the EP algorithm is to update the individual t_i approximations sequentially. Conceptually this is done by iterating the following four steps: we start from some current approximate posterior, from which we leave out the current t_i, giving rise to a marginal *cavity distribution*. Secondly, we combine the cavity distribution with the exact likelihood $p(y_i|f_i)$ to get the desired (non-Gaussian) marginal. Thirdly, we choose a Gaussian approximation to the non-Gaussian marginal, and in the final step we compute the t_i which makes the posterior have the desired marginal from step three. These four steps are iterated until convergence.

In more detail, we optimize the t_i approximations sequentially, using the approximation so far for all the other variables. In particular the approximate posterior for f_i contains three kinds of terms:

1. the prior $p(\mathbf{f}|X)$
2. the local approximate likelihoods t_j for all cases $j \neq i$
3. the exact likelihood for case i, $p(y_i|f_i) = \Phi(y_i f_i)$

Our goal is to combine these sources of information and choose parameters of t_i such that the marginal posterior is as accurate as possible. We will first combine the prior and the local likelihood approximations into the *cavity distribution*

$$q_{-i}(f_i) \;\propto\; \int p(\mathbf{f}|X) \prod_{j \neq i} t_j(f_j|\tilde{Z}_j, \tilde{\mu}_j, \tilde{\sigma}_j^2) df_j, \tag{3.54}$$

and subsequently combine this with the exact likelihood for case i. Conceptually, one can think of the combination of prior and the $n-1$ approximate likelihoods in eq. (3.54) in two ways, either by explicitly multiplying out the terms, or (equivalently) by removing approximate likelihood i from the approximate posterior in eq. (3.53). Here we will follow the latter approach. The marginal for f_i from $q(\mathbf{f}|X,\mathbf{y})$ is obtained by using eq. (A.6) in eq. (3.53) to give

$$q(f_i|X,\mathbf{y}) \;=\; \mathcal{N}(f_i|\mu_i, \sigma_i^2), \tag{3.55}$$

where $\sigma_i^2 = \Sigma_{ii}$. This marginal eq. (3.55) contains one approximate term

(namely t_i) "too many", so we need to divide it by t_i to get the cavity distribution

cavity distribution

$$q_{-i}(f_i) \triangleq \mathcal{N}(f_i|\mu_{-i},\sigma^2_{-i}), \qquad (3.56)$$
$$\text{where } \mu_{-i} = \sigma^2_{-i}(\sigma_i^{-2}\mu_i - \tilde{\sigma}_i^{-2}\tilde{\mu}_i), \text{ and } \sigma^2_{-i} = (\sigma_i^{-2} - \tilde{\sigma}_i^{-2})^{-1}.$$

Note that the cavity distribution and its parameters carry the subscript $-i$, indicating that they include all cases except number i. The easiest way to verify eq. (3.56) is to multiply the cavity distribution by the local likelihood approximation t_i from eq. (3.51) using eq. (A.7) to recover the marginal in eq. (3.55). Notice that despite the appearance of eq. (3.56), the cavity mean and variance are (of course) not dependent on $\tilde{\mu}_i$ and $\tilde{\sigma}_i^2$, see exercise 3.10.5.

To proceed, we need to find the new (un-normalized) Gaussian marginal which best approximates the product of the cavity distribution and the exact likelihood

$$\hat{q}(f_i) \triangleq \hat{Z}_i\mathcal{N}(\hat{\mu}_i,\hat{\sigma}_i^2) \simeq q_{-i}(f_i)p(y_i|f_i). \qquad (3.57)$$

It is well known that when $q(x)$ is Gaussian, the distribution $q(x)$ which minimizes $\mathrm{KL}\big(p(x)||q(x)\big)$ is the one whose first and second moments match that of $p(x)$, see eq. (A.24). As $\hat{q}(f_i)$ is un-normalized we choose additionally to impose the condition that the zero-th moments (normalizing constants) should match when choosing the parameters of $\hat{q}(f_i)$ to match the right hand side of eq. (3.57). This process is illustrated in Figure 3.4.

The derivation of the moments is somewhat lengthy, so we have moved the details to section 3.9. The desired posterior marginal moments are

$$\hat{Z}_i = \Phi(z_i), \qquad \hat{\mu}_i = \mu_{-i} + \frac{y_i\sigma^2_{-i}\mathcal{N}(z_i)}{\Phi(z_i)\sqrt{1+\sigma^2_{-i}}}, \qquad (3.58)$$
$$\hat{\sigma}_i^2 = \sigma^2_{-i} - \frac{\sigma^4_{-i}\mathcal{N}(z_i)}{(1+\sigma^2_{-i})\Phi(z_i)}\Big(z_i + \frac{\mathcal{N}(z_i)}{\Phi(z_i)}\Big), \quad \text{where} \quad z_i = \frac{y_i\mu_{-i}}{\sqrt{1+\sigma^2_{-i}}}.$$

The final step is to compute the parameters of the approximation t_i which achieves a match with the desired moments. In particular, the product of the cavity distribution and the local approximation must have the desired moments, leading to

$$\begin{aligned} \tilde{\mu}_i &= \tilde{\sigma}_i^2(\hat{\sigma}_i^{-2}\hat{\mu}_i - \sigma_{-i}^{-2}\mu_{-i}), \qquad \tilde{\sigma}_i^2 = (\hat{\sigma}_i^{-2} - \sigma_{-i}^{-2})^{-1}, \\ \tilde{Z}_i &= \hat{Z}_i\sqrt{2\pi}\sqrt{\sigma^2_{-i}+\tilde{\sigma}_i^2}\exp\big(\tfrac{1}{2}(\mu_{-i}-\tilde{\mu}_i)^2/(\sigma^2_{-i}+\tilde{\sigma}_i^2)\big), \end{aligned} \qquad (3.59)$$

which is easily verified by multiplying the cavity distribution by the local approximation using eq. (A.7) to obtain eq. (3.58). Note that the desired marginal posterior variance $\hat{\sigma}_i^2$ given by eq. (3.58) is guaranteed to be smaller than the cavity variance, such that $\tilde{\sigma}_i^2 > 0$ is always satisfied.[15]

This completes the update for a local likelihood approximation t_i. We then have to update the approximate posterior using eq. (3.53), but since only a

[15] In cases where the likelihood is log concave, one can show that $\tilde{\sigma}_i^2 > 0$, but for a general likelihood there may be no such guarantee.

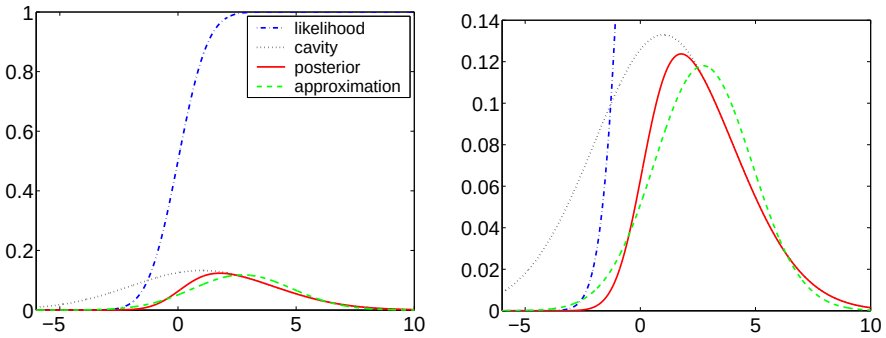


Figure 3.4: Approximating a single likelihood term by a Gaussian. Panel (a) dash-dotted: the exact likelihood, $\Phi(f_i)$ (the corresponding target being $y_i = 1$) as a function of the latent f_i, dotted: Gaussian cavity distribution $\mathcal{N}(f_i|\mu_{-i}=1,\ \sigma^2_{-i}=9)$, solid: posterior, dashed: posterior approximation. Panel (b) shows an enlargement of panel (a).

single site has changed one can do this with a computationally efficient rank-one update, see section 3.6.3. The EP algorithm is used iteratively, updating each local approximation in turn. It is clear that several passes over the data are required, since an update of one local approximation potentially influences all of the approximate marginal posteriors.

3.6.1 Predictions

The procedure for making predictions in the EP framework closely resembles the algorithm for the Laplace approximation in section 3.4.2. EP gives a Gaussian approximation to the posterior distribution, eq. (3.53). The approximate predictive mean for the latent variable f_* becomes

$$\begin{aligned}\mathbb{E}_q[f_*|X,\mathbf{y},\mathbf{x}_*] \;&=\; \mathbf{k}_*^\top K^{-1}\boldsymbol{\mu} \;=\; \mathbf{k}_*^\top K^{-1}(K^{-1}+\tilde{\Sigma}^{-1})^{-1}\tilde{\Sigma}^{-1}\tilde{\boldsymbol{\mu}} \\ &=\; \mathbf{k}_*^\top (K+\tilde{\Sigma})^{-1}\tilde{\boldsymbol{\mu}}.\end{aligned} \tag{3.60}$$

The approximate latent predictive variance is analogous to the derivation from eq. (3.23) and eq. (3.24), with $\tilde{\Sigma}$ playing the rôle of W

$$\mathbb{V}_q[f_*|X,\mathbf{y},\mathbf{x}_*] \;=\; k(\mathbf{x}_*,\mathbf{x}_*) - \mathbf{k}_*^\top (K+\tilde{\Sigma})^{-1}\mathbf{k}_*. \tag{3.61}$$

The approximate predictive distribution for the binary target becomes

$$q(y_*=1|X,\mathbf{y},\mathbf{x}_*) \;=\; \mathbb{E}_q[\pi_*|X,\mathbf{y},\mathbf{x}_*] \;=\; \int \Phi(f_*)q(f_*|X,\mathbf{y},\mathbf{x}_*)\,df_*, \tag{3.62}$$

where $q(f_*|X,\mathbf{y},\mathbf{x}_*)$ is the approximate latent predictive Gaussian with mean and variance given by eq. (3.60) and eq. (3.61). This integral is readily evaluated using eq. (3.80), giving the predictive probability

$$q(y_*=1|X,\mathbf{y},\mathbf{x}_*) \;=\; \Phi\Big(\frac{\mathbf{k}_*^\top (K+\tilde{\Sigma})^{-1}\tilde{\boldsymbol{\mu}}}{\sqrt{1+k(\mathbf{x}_*,\mathbf{x}_*)-\mathbf{k}_*^\top (K+\tilde{\Sigma})^{-1}\mathbf{k}_*}}\Big). \tag{3.63}$$

3.6.2 Marginal Likelihood

The EP approximation to the marginal likelihood can be found from the normalization of eq. (3.53)

$$Z_{\text{EP}} \;=\; q(\mathbf{y}|X) \;=\; \int p(\mathbf{f}|X)\prod_{i=1}^{n} t_i(f_i|\tilde{Z}_i,\tilde{\mu}_i,\tilde{\sigma}_i^2)\,d\mathbf{f}. \tag{3.64}$$

Using eq. (A.7) and eq. (A.8) in an analogous way to the treatment of the regression setting in equations (2.28) and (2.30) we arrive at

$$\begin{aligned}\log(Z_{\text{EP}}|\boldsymbol{\theta}) \;=\; &-\frac{1}{2}\log|K+\tilde{\Sigma}| - \frac{1}{2}\tilde{\boldsymbol{\mu}}^\top(K+\tilde{\Sigma})^{-1}\tilde{\boldsymbol{\mu}} \\ &+\sum_{i=1}^{n}\log\Phi\big(\frac{y_i\mu_{-i}}{\sqrt{1+\sigma_{-i}^2}}\big) + \frac{1}{2}\sum_{i=1}^{n}\log(\sigma_{-i}^2+\tilde{\sigma}_i^2) + \sum_{i=1}^{n}\frac{(\mu_{-i}-\tilde{\mu}_i)^2}{2(\sigma_{-i}^2+\tilde{\sigma}_i^2)},\end{aligned} \tag{3.65}$$

where $\boldsymbol{\theta}$ denotes the hyperparameters of the covariance function. This expression has a nice intuitive interpretation: the first two terms are the marginal likelihood for a regression model for $\tilde{\boldsymbol{\mu}}$, each component of which has independent Gaussian noise of variance $\tilde{\Sigma}_{ii}$ (as $\tilde{\Sigma}$ is diagonal), cf. eq. (2.30). The remaining three terms come from the normalization constants $\tilde{Z}_i$. The first of these penalizes the cavity (or leave-one-out) distributions for not agreeing with the classification labels, see eq. (3.82). In other words, we can see that the marginal likelihood combines two desiderata, (1) the means of the local likelihood approximations should be well predicted by a GP, and (2) the corresponding latent function, when ignoring a particular training example, should be able to predict the corresponding classification label well.

3.6.3 Implementation

The implementation for the EP algorithm follows the derivation in the previous section closely, except that care has to be taken to achieve numerical stability, in similar ways to the considerations for Laplace's method in section 3.4.3. In addition, we wish to be able to specifically handle the case were some site variances $\tilde{\sigma}_i^2$ may tend to infinity; this corresponds to ignoring the corresponding likelihood terms, and can form the basis of *sparse* approximations, touched upon in section 8.4. In this limit, everything remains well-defined, although this is not obvious e.g. from looking at eq. (3.65). It turns out to be slightly more convenient to use natural parameters $\tilde{\tau}_i,\ \tilde{\nu}_i$ and $\tau_{-i},\ \nu_{-i}$ for the site and cavity parameters

natural parameters

$$\tilde{\tau}_i \;=\; \tilde{\sigma}_i^{-2}, \quad \tilde{S} \;=\; \text{diag}(\tilde{\boldsymbol{\tau}}), \quad \tilde{\boldsymbol{\nu}} \;=\; \tilde{S}\tilde{\boldsymbol{\mu}}, \quad \tau_{-i} \;=\; \sigma_{-i}^{-2}, \quad \nu_{-i} \;=\; \tau_{-i}\mu_{-i} \tag{3.66}$$

rather than $\tilde{\sigma}_i^2,\ \tilde{\mu}_i$ and $\sigma_{-i}^2,\ \mu_{-i}$ themselves. The symmetric matrix of central importance is

$$B \;=\; I + \tilde{S}^{\frac{1}{2}}K\tilde{S}^{\frac{1}{2}}, \tag{3.67}$$

which plays a rôle equivalent to eq. (3.26). Expressions involving the inverse of B are computed via Cholesky factorization, which is numerically stable since

input: K (covariance matrix), $\mathbf{y}$ (± 1 targets)
2: $\tilde{\boldsymbol{\nu}} := \mathbf{0}$, $\tilde{\boldsymbol{\tau}} := \mathbf{0}$, $\Sigma := K$, $\boldsymbol{\mu} := \mathbf{0}$ initialization and eq. (3.53)
repeat
4: **for** $i := 1$ to n **do**
$\tau_{-i} := \sigma_i^{-2} - \tilde{\tau}_i$ } compute approximate cavity para-
6: $\nu_{-i} := \sigma_i^{-2}\mu_i - \tilde{\nu}_i$ } meters ν_{-i} and τ_{-i} using eq. (3.56)
compute the marginal moments $\hat{\mu}_i$ and $\hat{\sigma}_i^2$ using eq. (3.58)
8: $\Delta\tilde{\tau} := \hat{\sigma}_i^{-2} - \tau_{-i} - \tilde{\tau}_i$ and $\tilde{\tau}_i := \tilde{\tau}_i + \Delta\tilde{\tau}$ } update site parameters
$\tilde{\nu}_i := \hat{\sigma}_i^{-2}\hat{\mu}_i - \nu_{-i}$ } $\tilde{\tau}_i$ and $\tilde{\nu}_i$ using eq. (3.59)
10: $\Sigma := \Sigma - \left((\Delta\tilde{\tau})^{-1} + \Sigma_{ii}\right)^{-1}\mathbf{s}_i\,\mathbf{s}_i^\top$ } update Σ and $\boldsymbol{\mu}$ by eq. (3.70) and
$\boldsymbol{\mu} := \Sigma\tilde{\boldsymbol{\nu}}$ } eq. (3.53). $\mathbf{s}_i$ is column i of Σ
12: **end for**
$L := \text{cholesky}(I_n + \tilde{S}^{\frac{1}{2}}K\tilde{S}^{\frac{1}{2}})$ } re-compute the approximate
14: $V := L^\top \backslash \tilde{S}^{\frac{1}{2}}K$ } posterior parameters Σ and $\boldsymbol{\mu}$
$\Sigma := K - V^\top V$ and $\boldsymbol{\mu} := \Sigma\tilde{\boldsymbol{\nu}}$ } using eq. (3.53) and eq. (3.68)
16: **until** convergence
compute $\log Z_{\text{EP}}$ using eq. (3.65), (3.73) and (3.74) and the existing L
18: **return**: $\tilde{\boldsymbol{\nu}}$, $\tilde{\boldsymbol{\tau}}$ (natural site param.), $\log Z_{\text{EP}}$ (approx. log marg. likelihood)

Algorithm 3.5: Expectation Propagation for binary classification. The targets $\mathbf{y}$ are used only in line 7. In lines 13-15 the parameters of the approximate posterior are re-computed (although they already exist); this is done because of the large number of rank-one updates in line 10 which would eventually cause loss of numerical precision in Σ. The computational complexity is dominated by the rank-one updates in line 10, which takes $\mathcal{O}(n^2)$ per variable, i.e. $\mathcal{O}(n^3)$ for an entire sweep over all variables. Similarly re-computing Σ in lines 13-15 is $\mathcal{O}(n^3)$.

the eigenvalues of B are bounded below by one. The parameters of the Gaussian approximate posterior from eq. (3.53) are computed as

$$\Sigma \;=\; (K^{-1} + \tilde{S})^{-1} \;=\; K - K(K + \tilde{S}^{-1})^{-1}K \;=\; K - K\tilde{S}^{\frac{1}{2}}B^{-1}\tilde{S}^{\frac{1}{2}}K. \quad (3.68)$$

After updating the parameters of a site, we need to update the approximate posterior eq. (3.53) taking the new site parameters into account. For the inverse covariance matrix of the approximate posterior we have from eq. (3.53)

$$\Sigma^{-1} \;=\; K^{-1} + \tilde{S}, \quad \text{and thus} \quad \Sigma_{\text{new}}^{-1} \;=\; K^{-1} + \tilde{S}_{\text{old}} + (\tilde{\tau}_i^{\text{new}} - \tilde{\tau}_i^{\text{old}})\mathbf{e}_i\mathbf{e}_i^\top, \quad (3.69)$$

where $\mathbf{e}_i$ is a unit vector in direction i, and we have used that $\tilde{S} = \text{diag}(\tilde{\boldsymbol{\tau}})$. Using the matrix inversion lemma eq. (A.9), on eq. (3.69) we obtain the new Σ

$$\Sigma^{\text{new}} \;=\; \Sigma^{\text{old}} - \frac{\tilde{\tau}_i^{\text{new}} - \tilde{\tau}_i^{\text{old}}}{1 + (\tilde{\tau}_i^{\text{new}} - \tilde{\tau}_i^{\text{old}})\Sigma_{ii}^{\text{old}}}\mathbf{s}_i\mathbf{s}_i^\top, \quad (3.70)$$

in time $\mathcal{O}(n^2)$, where $\mathbf{s}_i$ is the i'th column of Σ^{old}. The posterior mean is then calculated from eq. (3.53).

In the EP algorithm each site is updated in turn, and several passes over all sites are required. Pseudocode for the EP-GPC algorithm is given in Algorithm

input: $\tilde{\boldsymbol{\nu}}$, $\tilde{\boldsymbol{\tau}}$ (natural site param.), X (inputs), $\mathbf{y}$ (± 1 targets), k (covariance function), $\mathbf{x}_*$ test input

2: $L := \text{cholesky}(I_n + \tilde{S}^{\frac{1}{2}} K \tilde{S}^{\frac{1}{2}})$ $\quad B = I_n + \tilde{S}^{\frac{1}{2}} K \tilde{S}^{\frac{1}{2}}$

$\mathbf{z} := \tilde{S}^{\frac{1}{2}} L^\top \backslash (L \backslash (\tilde{S}^{\frac{1}{2}} K \tilde{\boldsymbol{\nu}}))$

4: $\bar{f}_* := \mathbf{k}(\mathbf{x}_*)^\top (\tilde{\boldsymbol{\nu}} - \mathbf{z})$ } eq. (3.60) using eq. (3.71)

$\mathbf{v} := L \backslash \big(\tilde{S}^{\frac{1}{2}} \mathbf{k}(\mathbf{x}_*)\big)$

6: $\mathbb{V}[f_*] := k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{v}^\top \mathbf{v}$ } eq. (3.61) using eq. (3.72)

$\bar{\pi}_* := \Phi(\bar{f}_* / \sqrt{1 + \mathbb{V}[f_*]})$ eq. (3.63)

8: **return**: $\bar{\pi}_*$ (predictive class probability (for class 1))

Algorithm 3.6: Predictions for expectation propagation. The natural site parameters $\tilde{\boldsymbol{\nu}}$ and $\tilde{\boldsymbol{\tau}}$ of the posterior (which can be computed using algorithm 3.5) are input. For multiple test inputs lines 4-7 are applied to each test input. Computational complexity is $n^3/6 + n^2$ operations once (line 2 and 3) plus n^2 operations per test case (line 5), although the Cholesky decomposition in line 2 could be avoided by storing it in Algorithm 3.5. Note the close similarity to Algorithm 3.2 on page 47.

3.5. There is no formal guarantee of convergence, but several authors have reported that EP for Gaussian process models works relatively well.[16]

For the predictive distribution, we get the mean from eq. (3.60) which is evaluated using

$$\begin{aligned} \mathbb{E}_q[f_*|X,\mathbf{y},\mathbf{x}_*] &= \mathbf{k}_*^\top (K + \tilde{S}^{-1})^{-1} \tilde{S}^{-1} \tilde{\boldsymbol{\nu}} = \mathbf{k}_*^\top \big(I - (K + \tilde{S}^{-1})^{-1} K\big) \tilde{\boldsymbol{\nu}} \\ &= \mathbf{k}_*^\top (I - \tilde{S}^{\frac{1}{2}} B^{-1} \tilde{S}^{\frac{1}{2}} K) \tilde{\boldsymbol{\nu}}, \end{aligned} \tag{3.71}$$

and the predictive variance from eq. (3.61) similarly by

$$\begin{aligned} \mathbb{V}_q[f_*|X,\mathbf{y},\mathbf{x}_*] &= k(\mathbf{x}_*,\mathbf{x}_*) - \mathbf{k}_*^\top (K + \tilde{S}^{-1})^{-1} \mathbf{k}_* \\ &= k(\mathbf{x}_*,\mathbf{x}_*) - \mathbf{k}_*^\top \tilde{S}^{\frac{1}{2}} B^{-1} \tilde{S}^{\frac{1}{2}} \mathbf{k}_*. \end{aligned} \tag{3.72}$$

Pseudocode for making predictions using EP is given in Algorithm 3.6.

Finally, we need to evaluate the approximate log marginal likelihood from eq. (3.65). There are several terms which need careful consideration, principally due to the fact the $\tilde{\tau}_i$ values may be arbitrarily small (and cannot safely be inverted). We start with the fourth and first terms of eq. (3.65)

$$\begin{aligned} \tfrac{1}{2} \log |T^{-1} + \tilde{S}^{-1}| - \tfrac{1}{2} \log |K + \tilde{\Sigma}| &= \tfrac{1}{2} \log |\tilde{S}^{-1}(I + \tilde{S} T^{-1})| - \tfrac{1}{2} \log |\tilde{S}^{-1} B| \\ &= \tfrac{1}{2} \sum_i \log(1 + \tilde{\tau}_i \tau_{-i}^{-1}) - \sum_i \log L_{ii}, \end{aligned} \tag{3.73}$$

where T is a diagonal matrix of cavity precisions $T_{ii} = \tau_{-i} = \sigma_{-i}^{-2}$ and L is the Cholesky factorization of B. In eq. (3.73) we have factored out the matrix $\tilde{S}^{-1}$ from both determinants, and the terms cancel. Continuing with the part of the

[16] It has been conjectured (but not proven) by L. Csató (personal communication) that EP is guaranteed to converge if the likelihood is log concave.

fifth term from eq. (3.65) which is quadratic in $\tilde{\boldsymbol{\mu}}$ together with the second term

$$\begin{aligned} &\tfrac{1}{2}\tilde{\boldsymbol{\mu}}^\top (T^{-1}+\tilde{S}^{-1})^{-1}\tilde{\boldsymbol{\mu}} - \tfrac{1}{2}\tilde{\boldsymbol{\mu}}^\top (K+\tilde{\Sigma})^{-1}\tilde{\boldsymbol{\mu}} \\ &\qquad = \tfrac{1}{2}\tilde{\boldsymbol{\nu}}^\top \tilde{S}^{-1}\big((T^{-1}+\tilde{S}^{-1})^{-1} - (K+\tilde{S}^{-1})^{-1}\big)\tilde{S}^{-1}\tilde{\boldsymbol{\nu}} \qquad (3.74) \\ &\qquad = \tfrac{1}{2}\tilde{\boldsymbol{\nu}}^\top \big((K^{-1}+\tilde{S})^{-1} - (T+\tilde{S})^{-1}\big)\tilde{\boldsymbol{\nu}} \\ &\qquad = \tfrac{1}{2}\tilde{\boldsymbol{\nu}}^\top \big(K - K\tilde{S}^{\frac{1}{2}}B^{-1}\tilde{S}^{\frac{1}{2}}K - (T+\tilde{S})^{-1}\big)\tilde{\boldsymbol{\nu}}, \end{aligned}$$

where in eq. (3.74) we apply the matrix inversion lemma eq. (A.9) to both parenthesis to be inverted. The remainder of the fifth term in eq. (3.65) is evaluated using the identity

$$\tfrac{1}{2}\boldsymbol{\mu}_{-i}^\top (T^{-1}+\tilde{S}^{-1})^{-1}(\boldsymbol{\mu}_{-i} - 2\tilde{\boldsymbol{\mu}}) \;=\; \tfrac{1}{2}\boldsymbol{\mu}_{-i}^\top T(\tilde{S}+T)^{-1}(\tilde{S}\boldsymbol{\mu}_{-i} - 2\tilde{\boldsymbol{\nu}}), \qquad (3.75)$$

where $\boldsymbol{\mu}_{-i}$ is the vector of cavity means μ_{-i}. The third term in eq. (3.65) requires in no special treatment and can be evaluated as written.

3.7 Experiments

In this section we present the results of applying the algorithms for GP classification discussed in the previous sections to several data sets. The purpose is firstly to illustrate the behaviour of the methods and secondly to gain some insights into how good the performance is compared to some other commonly-used machine learning methods for classification.

Section 3.7.1 illustrates the action of a GP classifier on a toy binary prediction problem with a 2-d input space, and shows the effect of varying the length-scale ℓ in the SE covariance function. In section 3.7.2 we illustrate and compare the behaviour of the two approximate GP methods on a simple one-dimensional binary task. In section 3.7.3 we present results for a binary GP classifier on a handwritten digit classification task, and study the effect of varying the kernel parameters. In section 3.7.4 we carry out a similar study using a multi-class GP classifier to classify digits from all ten classes 0-9. In section 3.8 we discuss the methods from both experimental and theoretical viewpoints.

3.7.1 A Toy Problem

Figure 3.5 illustrates the operation of a Gaussian process classifier on a binary problem using the squared exponential kernel with a variable length-scale and the logistic response function. The Laplace approximation was used to make the plots. The data points lie within the square $[0,1]^2$, as shown in panel (a). Notice in particular the lone white point amongst the black points in the NE corner, and the lone black point amongst the white points in the SW corner.

In panel (b) the length-scale is $\ell = 0.1$, a relatively short value. In this case the latent function is free to vary relatively quickly and so the classifications

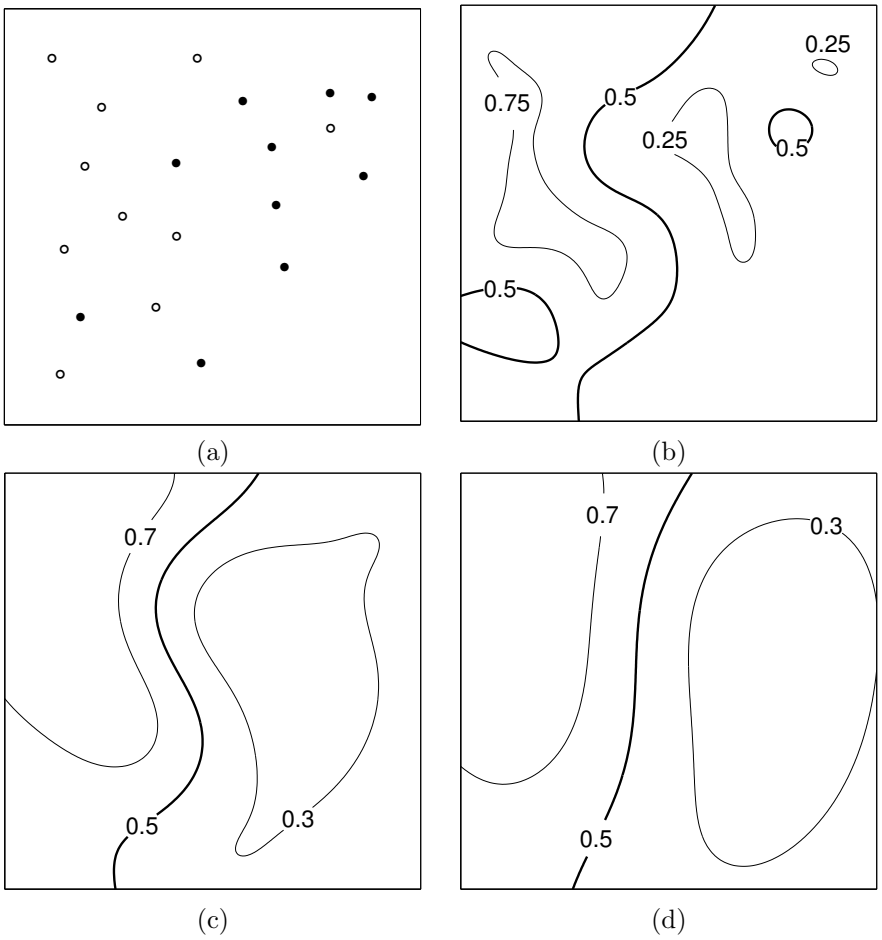
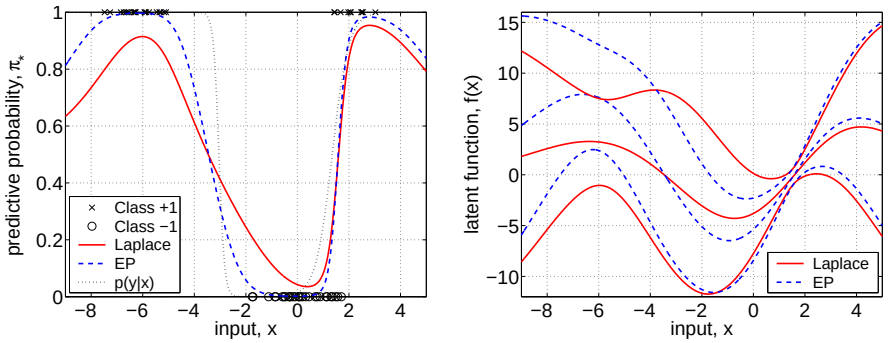


Figure 3.5: Panel (a) shows the location of the data points in the two-dimensional space $[0,1]^2$. The two classes are labelled as open circles (+1) and closed circles (-1). Panels (b)-(d) show contour plots of the predictive probability $\mathbb{E}_q[\pi(\mathbf{x}_*)|\mathbf{y}]$ for signal variance $\sigma_f^2 = 9$ and length-scales ℓ of 0.1, 0.2 and 0.3 respectively. The decision boundaries between the two classes are shown by the thicker black lines. The maximum value attained is 0.84, and the minimum is 0.19.

provided by thresholding the predictive probability $\mathbb{E}_q[\pi(\mathbf{x}_*)|\mathbf{y}]$ at 0.5 agrees with the training labels at all data points. In contrast, in panel (d) the length-scale is set to $\ell = 0.3$. Now the latent function must vary more smoothly, and so the two lone points are misclassified. Panel (c) was obtained with $\ell = 0.2$. As would be expected, the decision boundaries are more complex for shorter length-scales. Methods for setting the hyperparameters based on the data are discussed in chapter 5.



(a) (b)

Figure 3.6: One-dimensional toy classification dataset: Panel (a) shows the dataset, where points from class +1 have been plotted at $\pi = 1$ and class −1 at $\pi = 0$, together with the predictive probability for Laplace's method and the EP approximation. Also shown is the probability $p(y=+1|x)$ of the data generating process. Panel (b) shows the corresponding distribution of the latent function $f(x)$, showing curves for the mean, and ±2 standard deviations, corresponding to 95% confidence regions.

3.7.2 One-dimensional Example

Although Laplace's method and the EP approximation often give similar results, we here present a simple one-dimensional problem which highlights some of the differences between the methods. The data, shown in Figure 3.6(a), consists of 60 data points in three groups, generated from a mixture of three Gaussians, centered on −6 (20 points), 0 (30 points) and 2 (10 points), where the middle component has label −1 and the two other components label +1; all components have standard deviation 0.8; thus the two left-most components are well separated, whereas the two right-most components overlap.

Both approximation methods are shown with the same value of the hyperparameters, $\ell = 2.6$ and $\sigma_f = 7.0$, chosen to maximize the approximate marginal likelihood for Laplace's method. Notice in Figure 3.6 that there is a considerable difference in the value of the predictive probability for negative inputs. The Laplace approximation seems overly cautious, given the very clear separation of the data. This effect can be explained as a consequence of the intuition that the influence of "well-explained data points" is effectively reduced, see the discussion around eq. (3.19). Because the points in the left hand cluster are relatively well-explained by the model, they don't contribute as strongly to the posterior, and thus the predictive probability never gets very close to 1. Notice in Figure 3.6(b) the 95% confidence region for the latent function for Laplace's method actually includes functions that are negative at $x = -6$, which does not seem appropriate. For the positive examples centered around $x = 2$ on the right-hand side of Figure 3.6(b), this effect is not visible, because the points around the transition between the classes at $x = 1$ are not so "well-explained"; this is because the points near the boundary are competing against the points from the other class, attempting to pull the latent function in opposite directions. Consequently, the datapoints in this region all contribute strongly.

Another sign of this effect is that the uncertainty in the latent function, which is closely related to the "effective" local density of the data, is very small in the region around $x = 1$; the small uncertainty reveals a high effective density, which is caused by all data points in the region contributing with full weight. It should be emphasized that the example was artificially constructed specifically to highlight this effect.

Finally, Figure 3.6 also shows clearly the effects of uncertainty in the latent function on $\mathbb{E}_q[\pi_*|\mathbf{y}]$. In the region between $x = 2$ to $x = 4$, the latent mean in panel (b) increases slightly, but the predictive probability *decreases* in this region in panel (a). This is caused by the increase in uncertainty for the latent function; when the widely varying functions are squashed through the non-linearity it is possible for both classes to get high probability, and the average prediction becomes less extreme.

3.7.3 Binary Handwritten Digit Classification Example

Handwritten digit and character recognition are popular real-world tasks for testing and benchmarking classifiers, with obvious application e.g. in postal services. In this section we consider the discrimination of images of the digit 3 from images of the digit 5 as an example of binary classification; the specific choice was guided by the experience that this is probably one of the most difficult binary subtasks. 10-class classification of the digits 0-9 is described in the following section.

USPS dataset

We use the US Postal Service (USPS) database of handwritten digits which consists of 9298 segmented 16×16 greyscale images normalized so that the intensity of the pixels lies in $[-1, 1]$. The data was originally split into a training set of 7291 cases and a testset of the remaining 2007 cases, and has often been used in this configuration. Unfortunately, the data in the two partitions was collected in slightly different ways, such that the data in the two sets did not stem from the same distribution.[17] Since the basic underlying assumption for most machine learning algorithms is that the distribution of the training and test data should be identical, the original data partitions are not really suitable as a test bed for learning algorithms, the interpretation of the results being hampered by the change in distribution. Secondly, the original test set was rather small, sometimes making it difficult to differentiate the performance of different algorithms. To overcome these two problems, we decided to pool the two partitions and randomly split the data into two identically sized partitions of 4649 cases each. A side-effect is that it is not trivial to compare to results obtained using the original partitions. All experiments reported here use the repartitioned data. The binary 3s vs. 5s data has 767 training cases, divided 406/361 on 3s vs. 5s, while the test set has 773 cases split 418/355.

USPS repartitioned

squared exponential covariance function

We present results of both Laplace's method and EP using identical experimental setups. The squared exponential covariance function $k(\mathbf{x}, \mathbf{x}') =$

[17] It is well known e.g. that the original test partition had more difficult cases than the training set.

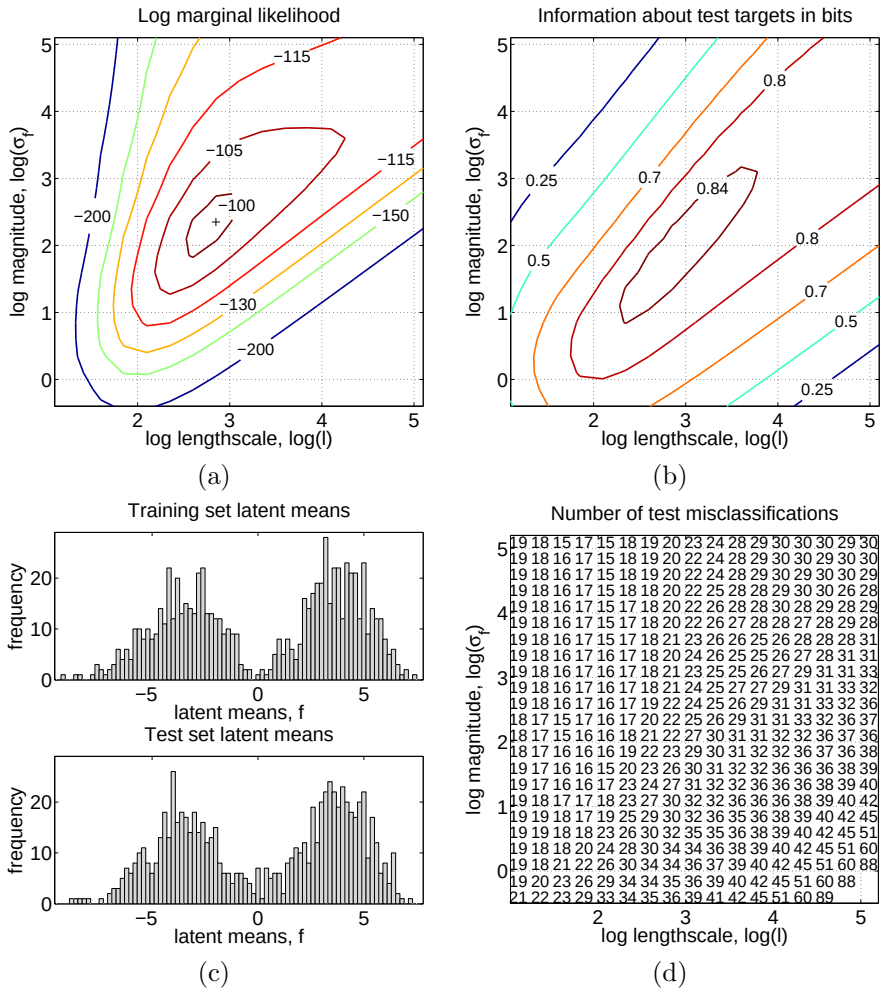


Figure 3.7: Binary Laplace approximation: 3s vs. 5s discrimination using the USPS data. Panel (a) shows a contour plot of the log marginal likelihood as a function of $\log(\ell)$ and $\log(\sigma_f)$. The marginal likelihood has an optimum at $\log(\ell) = 2.85$ and $\log(\sigma_f) = 2.35$, with an optimum value of $\log p(\mathbf{y}|X, \boldsymbol{\theta}) = -99$. Panel (b) shows a contour plot of the amount of information (in excess of a simple base-line model, see text) about the test cases in bits as a function of the same variables. The statistical uncertainty (because of the finite number of test cases) is about ± 0.03 bits (95% confidence interval). Panel (c) shows a histogram of the latent means for the training and test sets respectively at the values of the hyperparameters with optimal marginal likelihood (from panel (a)). Panel (d) shows the number of test errors (out of 773) when predicting using the sign of the latent mean.

hyperparameters

$\sigma_f^2 \exp(-|\mathbf{x} - \mathbf{x}'|^2/2\ell^2)$ was used, so there are two free parameters, namely σ_f (the process standard deviation, which controls its vertical scaling), and the length-scale ℓ (which controls the input length-scale). Let $\boldsymbol{\theta} = (\log(\ell), \log(\sigma_f))$ denote the vector of hyperparameters. We first present the results of Laplace's method in Figure 3.7 and discuss these at some length. We then briefly compare these with the results of the EP method in Figure 3.8.

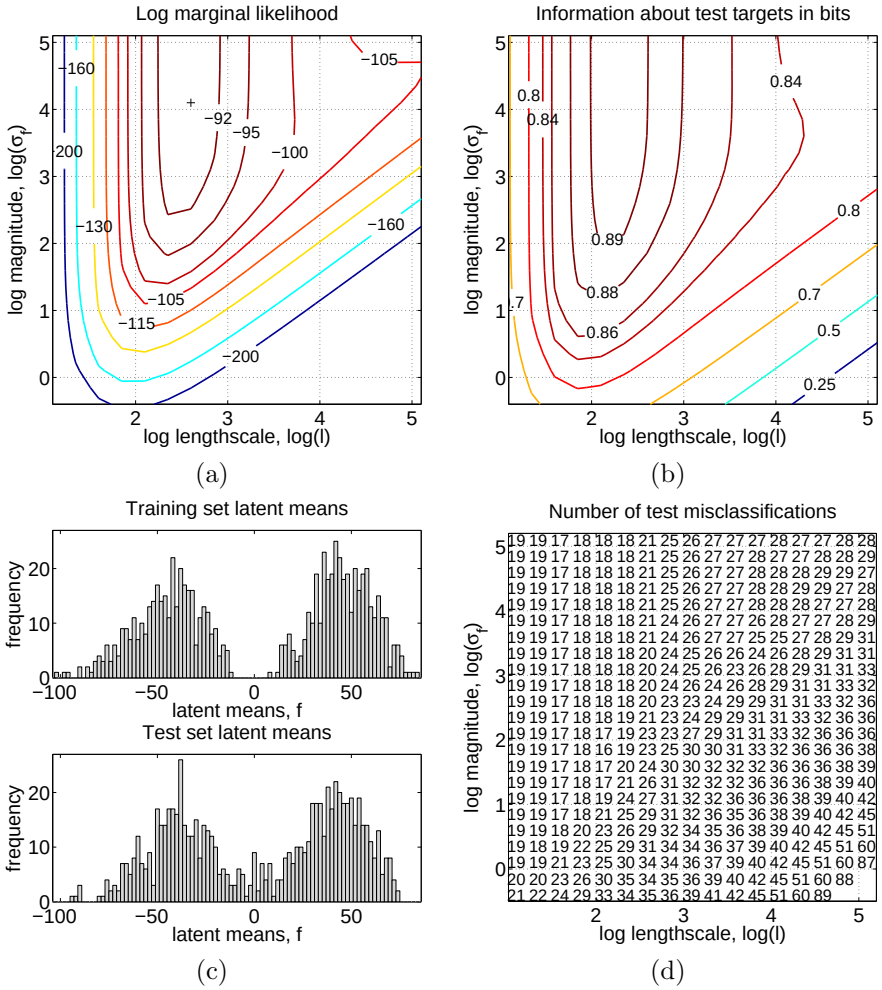


Figure 3.8: The EP algorithm on 3s vs. 5s digit discrimination task from the USPS data. Panel (a) shows a contour plot of the log marginal likelihood as a function of the hyperparameters $\log(\ell)$ and $\log(\sigma_f)$. The marginal likelihood has an optimum at $\log(\ell) = 2.6$ at the maximum value of $\log(\sigma_f)$, but the log marginal likelihood is essentially flat as a function of $\log(\sigma_f)$ in this region, so a good point is at $\log(\sigma_f) = 4.1$, where the log marginal likelihood has a value of -90. Panel (b) shows a contour plot of the amount of information (in excess of the baseline model) about the test cases in bits as a function of the same variables. Zero bits corresponds to no information and one bit to perfect binary generalization. The 773 test cases allows the information to be determined within ± 0.035 bits. Panel (c) shows a histogram of the latent means for the training and test sets respectively at the values of the hyperparameters with optimal marginal likelihood (from panel a). Panel (d) shows the number of test errors (out of 773) when predicting using the sign of the latent mean.

Laplace results

In Figure 3.7(a) we show a contour plot of the approximate log marginal likelihood (LML) $\log q(\mathbf{y}|X, \boldsymbol{\theta})$ as a function of $\log(\ell)$ and $\log(\sigma_f)$, obtained from runs on a grid of 17 evenly-spaced values of $\log(\ell)$ and 23 evenly-spaced values of $\log(\sigma_f)$. Notice that there is a maximum of the marginal likelihood

near $\log(\ell) = 2.85$ and $\log(\sigma_f) = 2.35$. As will be explained in chapter 5, we would expect that hyperparameters that yield a high marginal likelihood would give rise to good predictions. Notice that an increase of 1 unit on the log scale means that the probability is 2.7 times larger, so the marginal likelihood in Figure 3.7(a) is fairly well peaked.

test log predictive probability

There are at least two ways we can measure the quality of predictions at the test points. The first is the test log predictive probability $\log_2 p(y_*|\mathbf{x}_*, \mathcal{D}, \boldsymbol{\theta})$. In Figure 3.7(b) we plot the average over the test set of the test log predictive probability for the same range of hyperparameters. We express this as the amount of information in bits about the targets, by using log to the base 2. Further, we off-set the value by subtracting the amount of information that a simple base-line method would achieve. As a base-line model we use the best possible model which does not use the inputs; in this case, this model would just produce a predictive distribution reflecting the frequency of the two classes in the training set, i.e.

base-line method

$$-418/773 \log_2(406/767) - 355/773 \log_2(361/767) \;=\; 0.9956 \text{ bits}, \qquad (3.76)$$

essentially 1 bit. (If the classes had been perfectly balanced, and the training and test partitions also exactly balanced, we would arrive at exactly 1 bit.) Thus, our scaled information score used in Figure 3.7(b) would be zero for a method that did random guessing and 1 bit for a method which did perfect classification (with complete confidence). The information score measures how much information the model was able to extract from the inputs about the identity of the output. Note that this is *not* the mutual information between the model output and the test targets, but rather the Kullback-Leibler (KL) divergence between them. Figure 3.7 shows that there is a good qualitative agreement between the marginal likelihood and the test information, compare panels (a) and (b).

interpretation of information score

error rate

The second (and perhaps most commonly used) method for measuring the quality of the predictions is to compute the number of test errors made when using the predictions. This is done by computing $\mathbb{E}_q[\pi_*|\mathbf{y}]$ (see eq. (3.25)) for each test point, thresholding at 1/2 to get "hard" predictions and counting the number of errors. Figure 3.7(d) shows the number of errors produced for each entry in the 17×23 grid of values for the hyperparameters. The general trend in this table is that the number of errors is lowest in the top left-hand corner and increases as one moves right and downwards. The number of errors rises dramatically in the far bottom righthand corner. However, note in general that the number of errors is quite small (there are 773 cases in the test set).

The qualitative differences between the two evaluation criteria depicted in Figure 3.7 panels (b) and (d) may at first sight seem alarming. And although panels (a) and (b) show similar trends, one may worry about using (a) to select the hyperparameters, if one is interested in minimizing the test misclassification rate. Indeed a full understanding of all aspects of these plots is quite involved, but as the following discussion suggests, we can explain the major trends.

First, bear in mind that the effect of increasing ℓ is to make the kernel function broader, so we might expect to observe effects like those in Figure 3.5

where large widths give rise to a lack of flexibility. Keeping ℓ constant, the effect of increasing σ_f is to increase the magnitude of the values obtained for $\hat{\mathbf{f}}$. By itself this would lead to "harder" predictions (i.e. predictive probabilities closer to 0 or 1), but we have to bear in mind that the variances associated will also increase and this increased uncertainty for the latent variables tends to "soften" the predictive probabilities, i.e. move them closer to 1/2.

The most marked difference between Figure 3.7(b) and (d) is the behaviour in the the top left corner, where classification error rate remains small, but the test information and marginal likelihood are both poor. In the left hand side of the plots, the length scale ℓ is very short. This causes most points to be deemed "far away" from most other points. In this regime the prediction is dominated by the class-label of the nearest neighbours, and for the task at hand, this happens to give a low misclassification rate. In this parameter region the test latent variables $\mathbf{f}_*$ are very close to zero, corresponding to probabilities very close to 1/2. Consequently, the predictive probabilities carry almost no information about the targets. In the top left corner, the predictive probabilities for all 773 test cases lie in the interval $[0.48, 0.53]$. Notice that a large amount of information implies a high degree of correct classification, but not vice versa. At the optimal marginal likelihood values of the hyperparameters, there are 21 misclassifications, which is slightly higher that the minimum number attained which is 15 errors.

In exercise 3.10.6 readers are encouraged to investigate further the behaviour of $\hat{\mathbf{f}}$ and the predictive probabilities etc. as functions of $\log(\ell)$ and $\log(\sigma_f)$ for themselves.

EP results

In Figure 3.8 we show the results on the same experiment, using the EP method. The findings are qualitatively similar, but there are significant differences. In panel (a) the approximate log marginal likelihood has a different shape than for Laplace's method, and the maximum of the log marginal likelihood is about 9 units on a natural log scale larger (i.e. the marginal probability is $\exp(9) \simeq 8000$ times higher). Also note that the marginal likelihood has a ridge (for $\log \ell = 2.6$) that extends into large values of $\log \sigma_f$. For these very large latent amplitudes (see also panel (c)) the probit likelihood function is well approximated by a step function (since it transitions from low to high values in the domain $[-3, 3]$). Once we are in this regime, it is of course irrelevant exactly how large the magnitude is, thus the ridge. Notice, however, that this does not imply that the prediction will always be "hard", since the variance of the latent function also grows.

Figure 3.8 shows a good qualitative agreement between the approximate log marginal likelihood and the test information, compare panels (a) and (b). The best value of the test information is significantly higher for EP than for Laplace's method. The classification error rates in panel (d) show a fairly similar behaviour to that of Laplace's method. In Figure 3.8(c) we show the latent means for training and test cases. These show a clear separation on the training set, and much larger magnitudes than for Laplace's method. The absolute values of the entries in $\mathbf{f}_*$ are quite large, often well in excess of 50, which may suggest very "hard" predictions (probabilities close to zero or one),

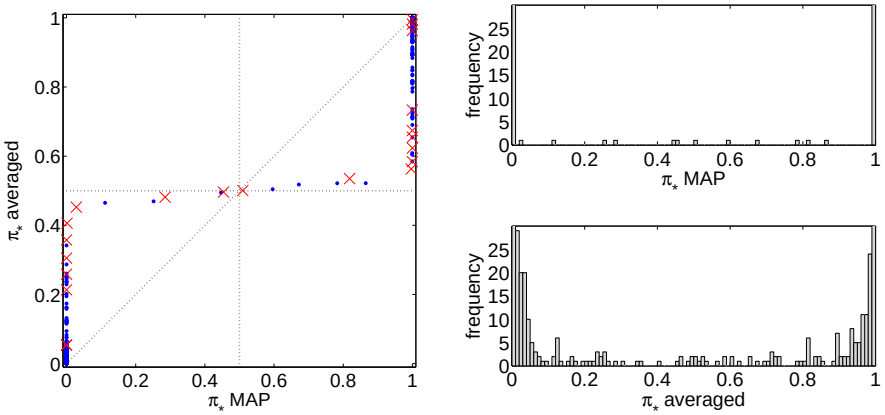


Figure 3.9: MAP vs. averaged predictions for the EP algorithm for the 3's vs. 5's digit discrimination using the USPS data. The optimal values of the hyperparameters from Figure 3.8(a) $\log(\ell) = 2.6$ and $\log(\sigma_f) = 4.1$ are used. The MAP predictions $\sigma(\mathbb{E}_q[f_*|\mathbf{y}])$ are "hard", mostly being very close to zero or one. On the other hand, the averaged predictions $\mathbb{E}_q[\pi_*|\mathbf{y}]$ from eq. (3.25) are a lot less extreme. In panel (a) the 21 cases that were misclassified are indicated by crosses (correctly classified cases are shown by points). Note that only 4 of the 21 misclassified points have confident predictions (i.e. outside $[0.1, 0.9]$). Notice that all points fall in the triangles below and above the horizontal line, confirming that averaging does not change the "most probable" class, and that it always makes the probabilities less extreme (i.e. closer to 1/2). Panel (b) shows histograms of averaged and MAP predictions, where we have truncated values over 30.

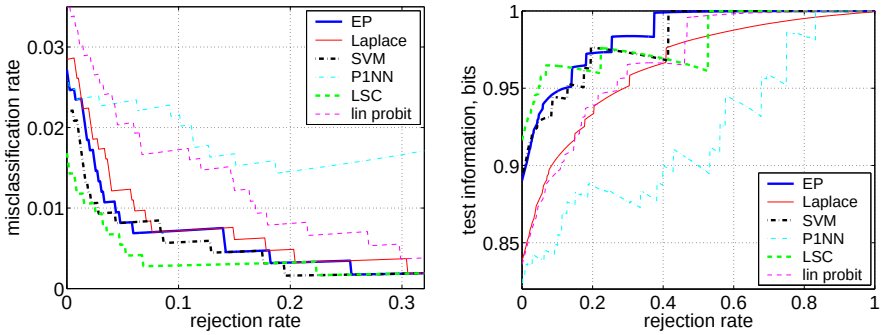
since the sigmoid saturates for smaller arguments. However, when taking the uncertainties in the latent variables into account, and computing the predictions using averaging as in eq. (3.25) the predictive probabilities are "softened". In Figure 3.9 we can verify that the averaged predictive probabilities are much less extreme than the MAP predictions.

In order to evaluate the performance of the two approximate methods for GP classification, we compared to a linear probit model, a support vector machine, a least-squares classifier and a nearest neighbour approach, all of which are commonly used in the machine learning community. In Figure 3.10 we show error-reject curves for both misclassification rate and the test information measure. The error-reject curve shows how the performance develops as a function of the fraction of test cases that is being rejected. To compute these, we first modify the methods that do not naturally produce probabilistic predictions to do so, as described below. Based on the predictive probabilities, we reject test cases for which the maximum predictive probability is smaller than a threshold. Varying the threshold produces the error-reject curve.

error-reject curve

The GP classifiers applied in Figure 3.10 used the hyperparameters which optimized the approximate marginal likelihood for each of the two methods. For the GP classifiers there were two free parameters σ_f and ℓ. The linear probit model (linear logistic models are probably more common, but we chose the probit here, since the other likelihood based methods all used probit) can be

linear probit model



(a) (b)

Figure 3.10: Panel (a) shows the error-reject curve and panel (b) the amount of information about the test cases as a function of the rejection rate. The probabilistic one nearest neighbour (P1NN) method has much worse performance than the other methods. Gaussian processes with EP behaves similarly to SVM's although the classification rate for SVM for low rejection rates seems to be a little better. Laplace's method is worse than EP and SVM. The GP least squares classifier (LSC) described in section 6.5 performs the best.

implemented as GP model using Laplace's method, which is equivalent to (although not computationally as efficient as) iteratively reweighted least squares (IRLS). The covariance function $k(\mathbf{x}, \mathbf{x}') = \theta^2 \mathbf{x}^\top \mathbf{x}'$ has a single hyperparameter, θ, which was set by maximizing the log marginal likelihood. This gives $\log p(\mathbf{y}|X, \theta) = -105$, at $\theta = 2.0$, thus the marginal likelihood for the linear covariance function is about 6 units on a natural log scale lower than the maximum log marginal likelihood for the Laplace approximation using the squared exponential covariance function.

support vector machine

The support vector machine (SVM) classifier (see section 6.4 for further details on the SVM) used the same SE kernel as the GP classifiers. For the SVM the rôle of ℓ is identical, and the trade-off parameter C in the SVM formulation (see eq. (6.37)) plays a similar rôle to σ_f^2. We carried out 5-fold cross validation on a grid in parameter space to identify the best combination of parameters w.r.t. the error rate; this turned out to be at $C = 1$, $\ell = 10$. Our experiments were conducted using the SVMTorch software [Collobert and Bengio, 2001]. In order to compute probabilistic predictions, we squashed the test-activities through a cumulative Gaussian, using the methods proposed by Platt [2000]: we made a parameterized linear transformation of the test-activities and fed this through the cumulative Gaussian.[18] The parameters of the linear transformation were chosen to maximize the log predictive probability, evaluated on the hold-out sets of the 5-fold cross validation.

probabilistic one nearest neighbour

The probabilistic one nearest neighbour (P1NN) method is a simple natural extension to the classical one nearest neighbour method which provides probabilistic predictions. It computes the leave-one-out (LOO) one nearest neighbour prediction on the training set, and records the fraction of cases π where the LOO predictions were correct. On test cases, the method then pre-

[18] Platt [2000] used a logistic whereas we use a cumulative Gaussian.

dicts the one nearest neighbour class with probability π, and the other class with probability $1-\pi$. Rejections are based on thresholding on the distance to the nearest neighbour.

The least-squares classifier (LSC) is described in section 6.5. In order to produce probabilistic predictions, the method of Platt [2000] was used (as described above for the SVM) using the predictive means only (the predictive variances were ignored[19]), except that instead of the 5-fold cross validation, leave-one-out cross-validation (LOO-CV) was used, and the kernel parameters were also set using LOO-CV.

Figure 3.10 shows that the three best methods are the EP approximation for GPC, the SVM and the least-squares classifier (LSC). Presenting both the error rates and the test information helps to highlight differences which may not be apparent from a single plot alone. For example, Laplace's method and EP seem very similar on error rates, but quite different in test information. Notice also, that the error-reject curve itself reveals interesting differences, e.g. notice that although the P1NN method has an error rate comparable to other methods at zero rejections, things don't improve very much when rejections are allowed. Refer to section 3.8 for more discussion of the results.

3.7.4 10-class Handwritten Digit Classification Example

We apply the multi-class Laplace approximation developed in section 3.5 to the 10-class handwritten digit classification problem from the (repartitioned) USPS dataset, having $n = 4649$ training cases and $n_* = 4649$ cases for testing, see page 63. We used a squared exponential covariance function with two hyperparameters: a single signal amplitude σ_f, common to all 10 latent functions, and a single length-scale parameter ℓ, common to all 10 latent functions and common to all 256 input dimensions.

The behaviour of the method was investigated on a grid of values for the hyperparameters, see Figure 3.11. Note that the correspondence between the log marginal likelihood and the test information is not as close as for Laplace's method for binary classification in Figure 3.7 on page 64. The maximum value of the log marginal likelihood attained is -1018, and for the hyperparameters corresponding to this point the error rate is 3.1% and the test information 2.67 bits. As with the binary classification problem, the test information is standardized by subtracting off the negative entropy (information) of the targets which is -3.27 bits. The classification error rate in Figure 3.11(c) shows a clear minimum, and this is also attained at a shorter length-scale than where the marginal likelihood and test information have their maxima. This effect was also seen in the experiments on binary classification.

To gain some insight into the level of performance we compared these results with those obtained with the probabilistic one nearest neighbour method P1NN, a multiple logistic regression model and a SVM. The P1NN first uses an

[19] Of course, one could also have tried a variant where the full latent predictive distribution was averaged over, but we did not do that here.

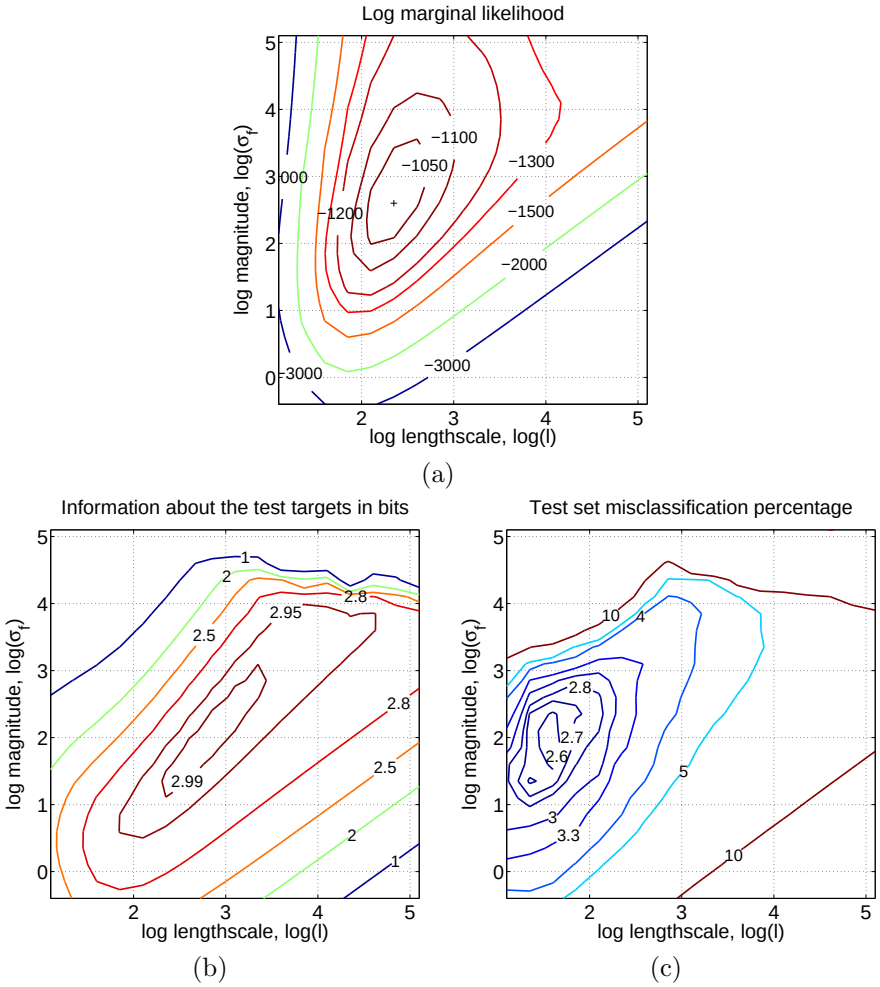


Figure 3.11: 10-way digit classification using the Laplace approximation. Panel (a) shows the approximate log marginal likelihood, reaching a maximum value of $\log p(\mathbf{y}|X,\boldsymbol{\theta}) = -1018$ at $\log \ell = 2.35$ and $\log \sigma_f = 2.6$. In panel (b) information about the test cases is shown. The maximum possible amount of information about the test targets, corresponding to perfect classification, would be 3.27 bits (the entropy of the targets). At the point of maximum marginal likelihood, the test information is 2.67 bits. In panel (c) the test set misclassification rate is shown in percent. At the point of maximum marginal likelihood the test error rate is 3.1%.

internal leave-one-out assessment on the training set to estimate its probability of being correct, π. For the test set it then predicts the nearest neighbour with probability π and all other classes with equal probability $(1-\pi)/9$. We obtained $\pi = 0.967$, a test information of 2.98 bits and a test set classification error rate of 3.0%.

We also compare to multiple linear logistic regression. One way to implement this method is to view it as a Gaussian process with a linear covariance

function, although it is equivalent and computationally more efficient to do the Laplace approximation over the "weights" of the linear model. In our case there are 10×257 weights (256 inputs and one bias), whereas there are 10×4696 latent function values in the GP. The linear covariance function $k(\mathbf{x},\mathbf{x}') = \theta^2\mathbf{x}^\top\mathbf{x}'$ has a single hyperparameter θ (used for all 10 latent functions). Optimizing the log marginal likelihood w.r.t. θ gives $\log p(\mathbf{y}|X,\theta) = -1339$ at $\theta = 1.45$. Using this value for the hyperparameter, the test information is 2.95 bits and the test set error rate is 5.7%.

Finally, a support vector machine (SVM) classifier was trained using the same SE kernel as the Gaussian process classifiers. (See section 6.4 for further details on the SVM.) As in the binary SVM case there were two free parameters ℓ (the length-scale of the kernel), and the trade-off parameter C (see eq. (6.37)), which plays a similar rôle to σ_f^2. We carried out 5-fold cross-validation on a grid in parameter space to identify the best combination of parameters w.r.t. the error rate; this turned out to be at $C = 1$, $\ell = 5$. Our experiments were conducted using the SVMTorch software [Collobert and Bengio, 2001], which implements multi-class SVM classification using the one-versus-rest method described in section 6.5. The test set error rate for the SVM is 2.2%; we did not attempt to evaluate the test information for the multi-class SVM.

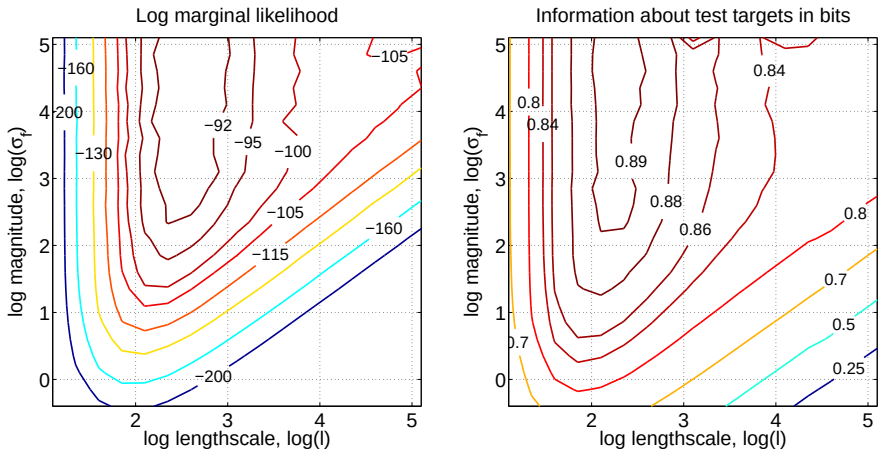
3.8 Discussion

In the previous section we presented several sets of experiments comparing the two approximate methods for inference in GPC models, and comparing them to other commonly-used supervised learning methods. In this section we discuss the results and attempt to relate them to the properties of the models.

For the binary examples from Figures 3.7 and 3.8, we saw that the two approximations showed quite different qualitative behaviour of the approximated log marginal likelihood, although the exact marginal likelihood is of course identical. The EP approximation gave a higher maximum value of the log marginal likelihood (by about 9 units on the log scale) and the test information was somewhat better than for Laplace's method, although the test set error rates were comparable. However, although this experiment seems to favour the EP approximation, it is interesting to know how close these approximations are to the exact (analytically intractable) solutions. In Figure 3.12 we show the results of running a sophisticated Markov chain Monte Carlo method called Annealed Importance Sampling [Neal, 2001] carried out by Kuss and Rasmussen [2005]. The USPS dataset for these experiments was identical to the one used in Figures 3.7 and 3.8, so the results are directly comparable. It is seen that the MCMC results indicate that the EP method achieves a very high level of accuracy, i.e. that the difference between EP and Laplace's method is caused almost exclusively by approximation errors in Laplace's method.

Monte Carlo results

The main reason for the inaccuracy of Laplace's method is that the high dimensional posterior is skew, and that the symmetric approximation centered on the mode is not characterizing the posterior volume very well. The posterior



(a) (b)

Figure 3.12: The log marginal likelihood, panel (a), and test information, panel (b), for the USPS 3's vs. 5's binary classification task computed using Markov chain Monte Carlo (MCMC). Comparing this to the Laplace approximation Figure 3.7 and Figure 3.8 shows that the EP approximation is surprisingly accurate. The slight wiggliness of the contour lines are caused by finite sample effects in the MCMC runs.

is a combination of the (correlated) Gaussian prior centered on the origin and the likelihood terms which (softly) cut off half-spaces which do not agree with the training set labels. Therefore the posterior looks like a correlated Gaussian restricted to the orthant which agrees with the labels. Its mode will be located close to the origin in that orthant, and it will decrease rapidly in the direction towards the origin due to conflicts from the likelihood terms, and decrease only slowly in the opposite direction (because of the prior). Seen in this light it is not surprising that the Laplace approximation is somewhat inaccurate. This explanation is corroborated further by Kuss and Rasmussen [2005].

suitablility of the covariance function

It should be noted that all the methods compared on the binary digits classification task except for the linear probit model are using the squared distance between the digitized digit images measured directly in the image space as the sole input to the algorithm. This distance measure is not very well suited for the digit discrimination task—for example, two similar images that are slight translations of each other may have a huge squared distance, although of course identical labels. One of the strengths of the GP formalism is that one can use prior distributions over (latent, in this case) functions, and do inference based on these. If however, the prior over functions depends only on one particular aspect of the data (the squared distance in image space) which is not so well suited for discrimination, then the prior used is also not very appropriate. It would be more interesting to design covariance functions (parameterized by hyperparameters) which are more appropriate for the digit discrimination task, e.g. reflecting on the known invariances in the images, such as the "tangent-distance" ideas from Simard et al. [1992]; see also Schölkopf and Smola [2002, ch. 11] and section 9.10. The results shown here follow the common approach of using a generic

covariance function with a minimum of hyperparameters, but this doesn't allow us to incorporate much prior information about the problem. For an example in the GP framework for doing inference about multiple hyperparameters with more complex covariance functions which provide clearly interpretable information about the data, see the carbon dioxide modelling problem discussed on page 118.

* 3.9 Appendix: Moment Derivations

Consider the integral of a cumulative Gaussian, Φ, with respect to a Gaussian

$$Z = \int_{-\infty}^{\infty} \Phi\big(\frac{x-m}{v}\big)\mathcal{N}(x|\mu,\sigma^2)\,dx, \quad \text{where} \quad \Phi(x) = \int_{-\infty}^{x} \mathcal{N}(y)\,dy, \tag{3.77}$$

initially for the special case $v>0$. Writing out in full, substituting $z = y - x + \mu - m$ and $w = x - \mu$ and interchanging the order of the integrals

$$\begin{aligned} Z_{v>0} &= \frac{1}{2\pi\sigma v}\int_{-\infty}^{\infty}\int_{-\infty}^{x} \exp\big(-\frac{(y-m)^2}{2v^2} - \frac{(x-\mu)^2}{2\sigma^2}\big)\,dy\,dx \\ &= \frac{1}{2\pi\sigma v}\int_{-\infty}^{\mu-m}\int_{-\infty}^{\infty} \exp\big(-\frac{(z+w)^2}{2v^2} - \frac{w^2}{2\sigma^2}\big)\,dw\,dz, \end{aligned} \tag{3.78}$$

or in matrix notation

$$\begin{aligned} Z_{v>0} &= \frac{1}{2\pi\sigma v}\int_{-\infty}^{\mu-m}\int_{-\infty}^{\infty} \exp\big(-\frac{1}{2}\begin{bmatrix} w \\ z \end{bmatrix}^\top \begin{bmatrix} \frac{1}{v^2}+\frac{1}{\sigma^2} & \frac{1}{v^2} \\ \frac{1}{v^2} & \frac{1}{v^2} \end{bmatrix}\begin{bmatrix} w \\ z \end{bmatrix}\big)\,dw\,dz \\ &= \int_{-\infty}^{\mu-m}\int_{-\infty}^{\infty} \mathcal{N}\Big(\begin{bmatrix} w \\ z \end{bmatrix}\Big|\,\mathbf{0},\, \begin{bmatrix} \sigma^2 & -\sigma^2 \\ -\sigma^2 & v^2+\sigma^2 \end{bmatrix}\Big)\,dw\,dz, \end{aligned} \tag{3.79}$$

i.e. an (incomplete) integral over a joint Gaussian. The inner integral corresponds to marginalizing over w (see eq. (A.6)), yielding

$$Z_{v>0} = \frac{1}{\sqrt{2\pi(v^2+\sigma^2)}}\int_{-\infty}^{\mu-m} \exp\Big(-\frac{z^2}{2(v^2+\sigma^2)}\Big)dz = \Phi\big(\frac{\mu-m}{\sqrt{v^2+\sigma^2}}\big), \tag{3.80}$$

which assumed $v > 0$. If v is negative, we can substitute the symmetry $\Phi(-z) = 1 - \Phi(z)$ into eq. (3.77) to get

$$Z_{v<0} = 1 - \Phi\big(\frac{\mu-m}{\sqrt{v^2+\sigma^2}}\big) = \Phi\big(-\frac{\mu-m}{\sqrt{v^2+\sigma^2}}\big). \tag{3.81}$$

Collecting the two cases, eq. (3.80) and eq. (3.81) we arrive at

$$Z = \int\Phi\big(\frac{x-m}{v}\big)\mathcal{N}(x|\mu,\sigma^2)\,dx = \Phi(z), \quad \text{where} \quad z = \frac{\mu-m}{v\sqrt{1+\sigma^2/v^2}}, \tag{3.82}$$

for general $v \neq 0$. We wish to compute the moments of

$$q(x) = Z^{-1}\Phi\big(\frac{x-m}{v}\big)\mathcal{N}(x|\mu,\sigma^2), \tag{3.83}$$

where Z is given in eq. (3.82). Perhaps the easiest way to do this is to differentiate w.r.t. μ on both sides of eq. (3.82)

$$\begin{aligned} \frac{\partial Z}{\partial \mu} &= \int \frac{x-\mu}{\sigma^2}\Phi\big(\frac{x-m}{v}\big)\mathcal{N}(x|\mu,\sigma^2)\,dx = \frac{\partial}{\partial \mu}\Phi(z) \iff \\ \frac{1}{\sigma^2}\int x\Phi\big(\frac{x-m}{v}\big)\mathcal{N}(x|\mu,\sigma^2)\,dx - \frac{\mu Z}{\sigma^2} &= \frac{\mathcal{N}(z)}{v\sqrt{1+\sigma^2/v^2}}, \end{aligned} \tag{3.84}$$

where we have used $\partial\Phi(z)/\partial\mu = \mathcal{N}(z)\partial z/\partial\mu$. We recognize the first term in the integral in the top line of eq. (3.84) as Z/σ^2 times the first moment of q which we are seeking. Multiplying through by σ^2/Z and rearranging we obtain first moment

$$\mathbb{E}_q[x] = \mu + \frac{\sigma^2\mathcal{N}(z)}{\Phi(z)v\sqrt{1+\sigma^2/v^2}}. \tag{3.85}$$

Similarly, the second moment can be obtained by differentiating eq. (3.82) twice

$$\begin{aligned} \frac{\partial^2 Z}{\partial \mu^2} &= \int\Big[\frac{x^2}{\sigma^4} - \frac{2\mu x}{\sigma^4} + \frac{\mu^2}{\sigma^4} - \frac{1}{\sigma^2}\Big]\Phi\big(\frac{x-m}{v}\big)\mathcal{N}(x|\mu,\sigma^2)\,dx = -\frac{z\mathcal{N}(z)}{v^2+\sigma^2} \\ &\iff \mathbb{E}_q[x^2] = 2\mu\mathbb{E}_q[x] - \mu^2 + \sigma^2 - \frac{\sigma^4 z\mathcal{N}(z)}{\Phi(z)(v^2+\sigma^2)}, \end{aligned} \tag{3.86}$$

where the first and second terms of the integral in the top line of eq. (3.86) are multiples of the first and second moments. The second central moment after reintroducing eq. (3.85) into eq. (3.86) and simplifying is given by second moment

$$\mathbb{E}_q\big[(x-\mathbb{E}_q[x])^2\big] = \mathbb{E}_q[x^2] - \mathbb{E}_q[x]^2 = \sigma^2 - \frac{\sigma^4\mathcal{N}(z)}{(v^2+\sigma^2)\Phi(z)}\Big(z + \frac{\mathcal{N}(z)}{\Phi(z)}\Big). \tag{3.87}$$

3.10 Exercises

1. For binary GPC, show the equivalence of using a noise-free latent process combined with a probit likelihood and a latent process with Gaussian noise combined with a step-function likelihood. Hint: introduce explicitly additional noisy latent variables $\tilde{f}_i$, which differ from f_i by Gaussian noise. Write down the step function likelihood for a single case as a function of $\tilde{f}_i$, integrate out the noisy variable, to arrive at the probit likelihood as a function of the noise-free process.

2. Consider a multinomial random variable $\mathbf{y}$ having C states, with $y_c = 1$ if the variable is in state c, and 0 otherwise. State c occurs with probability π_c. Show that $\mathrm{cov}(\mathbf{y}) = \mathbb{E}[(\mathbf{y}-\boldsymbol{\pi})(\mathbf{y}-\boldsymbol{\pi})^\top] = \mathrm{diag}(\boldsymbol{\pi}) - \boldsymbol{\pi}\boldsymbol{\pi}^\top$. Observe that $\mathrm{cov}(\mathbf{y})$, being a covariance matrix, must necessarily be positive semidefinite. Using this fact show that the matrix $W = \mathrm{diag}(\boldsymbol{\pi}) - \Pi\Pi^\top$ from eq. (3.38) is positive semidefinite. By showing that the vector of all ones is an eigenvector of $\mathrm{cov}(\mathbf{y})$ with eigenvalue zero, verify that the matrix is indeed positive semidefinite, and not positive definite. (See section 4.1 for definitions of positive semidefinite and positive definite matrices.)

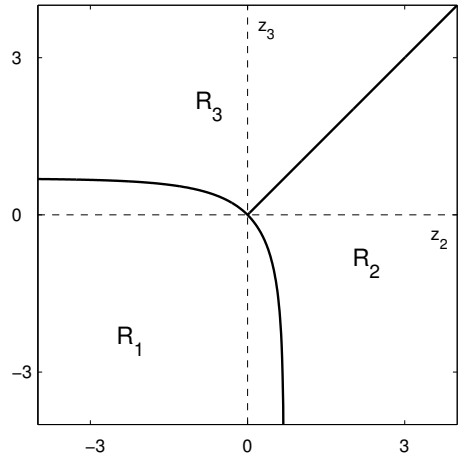


Figure 3.13: The decision regions for the three-class softmax function in z_2-z_3 space.

3. Consider the 3-class softmax function

$$p(\mathcal{C}_c) \;=\; \frac{\exp(f_c)}{\exp(f_1)+\exp(f_2)+\exp(f_3)},$$

where $c = 1,\ 2,\ 3$ and f_1, f_2, f_3 are the corresponding activations. To more easily visualize the decision boundaries, let $z_2 = f_2 - f_1$ and $z_3 = f_3 - f_1$. Thus

$$p(\mathcal{C}_1) \;=\; \frac{1}{1+\exp(z_2)+\exp(z_3)}, \tag{3.88}$$

and similarly for the other classes. The decision boundary relating to $p(\mathcal{C}_1) > 1/3$ is the curve $\exp(z_2) + \exp(z_3) = 2$. The decision regions for the three classes are illustrated in Figure 3.13. Let $\mathbf{f} = (f_1, f_2, f_3)^\top$ have a Gaussian distribution centered on the origin, and let $\boldsymbol{\pi}(\mathbf{f}) = \text{softmax}(\mathbf{f})$. We now consider the effect of this distribution on $\bar{\boldsymbol{\pi}} = \int \boldsymbol{\pi}(\mathbf{f}) p(\mathbf{f})\, d\mathbf{f}$. For a Gaussian with given covariance structure this integral is easily approximated by drawing samples from $p(\mathbf{f})$. Show that the classification can be made to fall into any of the three categories depending on the covariance matrix. Thus, by considering displacements of the mean of the Gaussian by ϵ from the origin into each of the three regions we have shown that overall classification depends not only on the mean of the Gaussian but also on its covariance. Show that this conclusion is still valid when it is recalled that $\mathbf{z}$ is derived from $\mathbf{f}$ as $\mathbf{z} = T\mathbf{f}$ where

$$T = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \end{pmatrix},$$

so that $\text{cov}(\mathbf{z}) = T\text{cov}(\mathbf{f})T^\top$.

4. Consider the update equation for $\mathbf{f}^{\text{new}}$ given by eq. (3.18) when some of the training points are well-explained under $\mathbf{f}$ so that $t_i \simeq \pi_i$ and $W_{ii} \simeq 0$

for these points. Break $\mathbf{f}$ into two subvectors, $\mathbf{f}_1$ that corresponds to points that are *not* well-explained, and $\mathbf{f}_2$ to those that are. Re-write $(K^{-1}+W)^{-1}$ from eq. (3.18) as $K(I+WK)^{-1}$ and let K be partitioned as $K_{11},\ K_{12},\ K_{21},\ K_{22}$ and similarly for the other matrices. Using the partitioned matrix inverse equations (see section A.3) show that

$$\begin{aligned} \mathbf{f}_1^{\text{new}} &= K_{11}(I_{11}+W_{11}K_{11})^{-1}\big(W_{11}\mathbf{f}_1+\nabla\log p(\mathbf{y}_1|\mathbf{f}_1)\big), \\ \mathbf{f}_2^{\text{new}} &= K_{21}K_{11}^{-1}\mathbf{f}_1^{\text{new}}. \end{aligned} \tag{3.89}$$

See section 3.4.1 for the consequences of this result.

5. Show that the expressions in eq. (3.56) for the cavity mean μ_{-i} and variance σ_{-i}^2 do not depend on the approximate likelihood terms $\tilde{\mu}_i$ and $\tilde{\sigma}_i^2$ for the corresponding case, despite the appearance of eq. (3.56).

6. Consider the USPS 3s vs. 5s prediction problem discussed in section 3.7.3. Use the implementation of the Laplace binary GPC provided to investigate how $\hat{\mathbf{f}}$ and the predictive probabilities etc. vary as functions of $\log(\ell)$ and $\log(\sigma_f)$.

C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006, ISBN 026218253X. www.GaussianProcess.org/gpml