

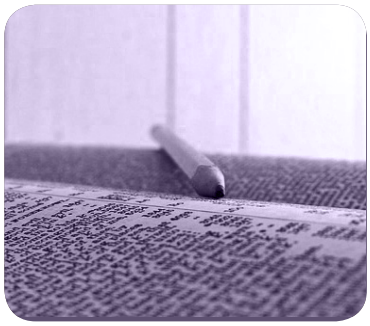
# MÁSTERES de la UAM

Escuela Politécnica Superior / 16-17

Investigación e Innovación en TIC

UNIVERSIDAD AUTONOMA DE MADRID

excelencia Campus Internacional UAM CSIC+

## Procesos gaussianos para problemas de clasificación multiclase en conjuntos de datos grandes

*Carlos Villacampa Calvo*

UAM EDICIONES



UNIVERSIDAD AUTÓNOMA DE MADRID

MASTER'S DEGREE IN ICT RESEARCH AND INNOVATION ($i^2$-ICT)

# Procesos Gaussianos para Problemas de Clasificación Multiclase en Conjuntos de Datos Grandes

MASTER'S THESIS

AUTHOR:
CARLOS VILLACAMPA CALVO

DIRECTOR:
DANIEL HERNÁNDEZ LOBATO

February 4, 2017

UNIVERSIDAD AUTÓNOMA DE MADRID

# *Abstract*

Escuela Politécnica Superior
Computer Science and Engineering Department

Master's Degree in ICT Research and Innovation ($i^2$-ICT)

**Procesos Gaussianos para Problemas de Clasificación Multiclase en Conjuntos de Datos Grandes**

by Carlos VILLACAMPA CALVO

Gaussian processes are non-parametric models that can be used to carry out supervised and unsupervised learning tasks. As they are non-parametric models, their complexity grows with the number of data instances, and as a consequence, they can be used to explain complex phenomena associated with the training dataset. They are also very useful to introduce a priori knowledge in the learning problem, because the characteristics that they can describe are given by a covariance function. Finally, these models are Bayesian models, thus they allow to obtain the uncertainty of the predictions and perform model comparison in an automated way. Despite all these advantages, in practice Gaussian processes have certain limitations. The first one is that the computations needed to train the model are only tractable in regression problems with Gaussian additive noise, and for any other case they need to be approximated. The other problem is their scalability, given that the training cost is cubic with respect to the number of observed data points $N$. In this master thesis, we propose a method for multiclass classification with Gaussian processes that scales well to very large datasets. For that, it uses the Expectation Propagation algorithm, along with the Fully Independent Training Conditional approximation (which introduces $M \ll N$ pseudo-inputs), stochastic gradients and some extra assumptions that reduce the training cost to $O(M^3)$. Experimental results show that this method is competitive with other approaches based on variational inference.

# Contents

# Chapter 1

# Introduction to Machine Learning

In this chapter we will make a brief introduction to machine learning problems and the basic methods for solving them.

We can define machine learning as a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty (Murphy, 2012: p1).

Machine learning is usually divided into two main categories, according to the type of problem that we wish to solve: supervised learning and unsupervised learning. There is a third type of machine learning, called reinforcement learning, but it is less commonly used and it goes beyond the scope of this thesis.

## 1.1 Bayesian machine learning

First of all, we need to understand the Bayesian approach for solving machine learning problems. Let the data be a set of observed points $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$.

### 1.1.1 Likelihood

The *likelihood* function describes the information obtained from the observed data. It is the distribution of the observed data conditional on some parameters $p(\mathbf{X}|\boldsymbol{\theta})$. If we just consider this as a function of $\boldsymbol{\theta}$ and we optimize the parameters $\boldsymbol{\theta}$ by maximizing the likelihood function we obtain what is called the **maximum likelihood estimation** (MLE).

### 1.1.2 Prior

The *prior* distribution introduces the values that we expect to obtain for the parameters before observing the data, or the prior knowledge that we may have about the problem. It is the distribution of the parameter(s) before any data is observed $p(\boldsymbol{\theta})$. This prior knowledge is particularly important in small datasets, where we do not get enough information from the observed data.

### 1.1.3 Posterior

The *posterior* distribution combines the prior and the likelihood by means of the Bayes' theorem in order to capture the values for $\boldsymbol{\theta}$ that are compatible with the observed data.

$$p(\boldsymbol{\theta}|\mathbf{X}) = \frac{p(\mathbf{X}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathbf{X})}\,. \tag{1.1}$$

In general, when we have enough data, the posterior $p(\boldsymbol{\theta}|\mathbf{X})$ becomes peaked on a single concept, namely the **maximum a posteriori** (MAP) estimate (Murphy, 2012: p68).

The denominator in Bayes' theorem $p(\mathbf{X})$ is a normalization constant and is called the **marginal likelihood**. It is useful for comparing several possible models given the observed data (Bishop, 2006: p161).

### 1.1.4 Posterior predictive distribution

The *posterior predictive distribution* is the distribution of a new data point $\mathbf{x}_*$ conditional on the observed data. It is given by:

$$p(\mathbf{x}_*|\mathbf{X}) = \int_{\boldsymbol{\theta}} p(\mathbf{x}_*|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathbf{X})d\boldsymbol{\theta}\,, \tag{1.2}$$

where we have marginalized out the parameters $\boldsymbol{\theta}$ and $p(\mathbf{x}_*|\boldsymbol{\theta})$ is the predictive distribution given $\boldsymbol{\theta}$ for the new data point.

## 1.2 Supervised learning

The goal is to learn a mapping from inputs $\mathbf{x}$ to outputs $y$ given a set of labeled examples $\{(x_i, y_i)\}_{i=1}^N$.

When the output variable $y$ is a real-valued variable the problem is known as **regression**, and when is categorical (it takes a value from a finite set), the problem is known as **classification**.

### 1.2.1 Regression

The simplest method to solve regression problems is called *linear regression* and it consists in fitting a linear combination of the input variables (Bishop, 2006: p138)

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \ldots + w_N x_N\,, \tag{1.3}$$

where $\mathbf{x}$ is the input vector and $\mathbf{w}$ is a weight vector. We will choose the vector $\mathbf{w}$ such that it minimizes an error function (usually the least square error) using the *maximum likelihood* estimator.

To make the model more expressive we can define a new method by considering linear combinations of fixed nonlinear functions of the input variables

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x}) \,, \tag{1.4}$$

where $\phi_j(\mathbf{x})$ are known as *basis functions* and the total number of parameters in this model will be $M$.

It is often convenient to define an additional dummy 'basis function' $\phi_0(\mathbf{x}) = 1$ so that

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}) \,. \tag{1.5}$$

The main issue with this approach is that we can make the model too complex and end up *over-fitting* the data. This happens when we try to model every minor variation in the output, since this is more likely to be noise than true signal (Murphy, 2012: p22). The consequence is that the model will have poor predictive performance, as it overreacts to minor fluctuations in the training data. In order to avoid it, a regularization term can be added to the error function so that we can control the *bias-variance trade-off*, which comes from the decomposition of the expected error in bias error, variance error and irreducible error. The last one cannot be reduced regardless of the algorithm. Biases are simplifying assumptions which tend to generate simpler models, so algorithms with high variance may under-fit the training data. Variance can be seen as the sensitivity of the algorithm to small fluctuations in the training set, so algorithms with high variance tend to over-fit. In general, increasing the variance will decrease the bias and viceversa.

If we use a Bayesian treatment of linear regression we will avoid the over-fitting problem and it will allow us to know the uncertainty of the predictions made by the model and to compare easily the different models.

We assume that the target variable $t$ is given by a deterministic function $y(\mathbf{x}, \mathbf{w})$ with additive Gaussian noise so that

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon \,, \tag{1.6}$$

where $\epsilon$ is a zero mean Gaussian random variable with precision $\beta$. Thus we can write

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1}) \,, \tag{1.7}$$

where $\mathcal{N}(x|\mu, \sigma^2)$ is the probability density function of a Gaussian distribution with mean $\mu$ and variance $\sigma^2$. This is, for the univariate case:

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\sigma^2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \,. \tag{1.8}$$

Making the assumption that the observed data points are drawn independently from (1.7), the *likelihood* function will be

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}) .\beta^{-1}) \tag{1.9}$$

The conjugate *prior* probability distribution over the model parameters $\mathbf{w}$ is given by a Gaussian distribution of the form

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_0, \mathbf{S}_0)\,, \tag{1.10}$$

having mean $\mathbf{m}_0$ and covariance $\mathbf{S}_0$.

The *posterior* distribution $p(\mathbf{w}|\mathbf{t}, \beta)$ is proportional to the product of the likelihood function and the prior and due to the choice of a conjugate Gaussian prior, the posterior will also be Gaussian. This is important because the *predictive distribution* is defined by

$$p(t|\mathbf{t}, \beta) = \int p(t|\mathbf{w}, \beta)p(\mathbf{w}|\mathbf{t}, \beta)d\mathbf{w}\,, \tag{1.11}$$

in which $\mathbf{t}$ is the vector of target variables in the training set. This integral will be analytically tractable because all the terms in it are Gaussian.

### 1.2.2 Classification

In classification, given some observed data with attributes $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ and with associated labels $\mathbf{y} = (y_1, \ldots, y_n)$ where the $y_i$ are categorical variables , the task is to predict the class label of a new instance.

A **discriminant** is a function that receives an input vector $\mathbf{x}$ and assigns it to one of $C$ classes.

The simplest discriminants are linear discriminants, those for which the decision surfaces are hyperplanes. We can then extend to nonlinear problems by using basis functions, as explained in the previous section.

For multiple classes, we can consider the use of $C - 1$ classifiers each of which solves a two-class problem of separating points in a particular class $C_k$ from points not in that class. This is known as a *one-versus-the-rest* classifier. However, it can lead to regions that are ambiguously classified.

Another option is to introduce $C(C - 1)/2$ binary discriminant functions, one for every possible pair of classes. This is known as a *one-versus-one* classifier. It can also lead to ambiguously classified regions.

In order to avoid these difficulties we can use instead a single $C$-class discriminant comprising $C$ linear functions of the form (Bishop, 2006: p183)

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0}\,, \tag{1.12}$$

and then assign an input point $\mathbf{x}$ to class $C_k$ if $y_k(\mathbf{x}) > y_j(\mathbf{x})$ for all $j \neq k$ (Bishop, 2006: p183).

Once we have defined discriminant functions, we can adopt two different approaches:

**Generative models:** will try to model the class-conditional densities $p(\mathbf{x}|C_k))$, as well as the class priors $p(C_k)$, and then use them to compute the posterior probabilities $p(C_k|\mathbf{x})$ through Bayes theorem.

Considering the two class case, the posterior probability for class $C_1$ can be written as

$$\begin{aligned} p(C_1|\mathbf{x}) &= \frac{p(\mathbf{x}|C_1)p(C_1)}{p(\mathbf{x}|C_1)p(C_1) + p(\mathbf{x}|C_2)p(C_2)} \\ &= \frac{1}{1+\exp(-a)} = \sigma(a)\,, \end{aligned} \tag{1.13}$$

where we have defined

$$a = \ln \frac{p(\mathbf{x}|C_1)p(C_1)}{p(\mathbf{x}|C_2)p(C_2)}\,, \tag{1.14}$$

and $\sigma(a)$ is the *logistic sigmoid* function.

Similarly, for the multi-class case, the posterior probability of class $C_k$ is given by

$$\begin{aligned} p(C_k|\mathbf{x}) &= \frac{p(\mathbf{x}|C_k)p(C_k)}{\sum_j p(\mathbf{x}|C_j)p(C_j)} \\ &= \frac{\exp(a_k)}{\sum_j \exp(a_j)} \end{aligned} \tag{1.15}$$

which is known as the *softmax function* and can be viewed as a generalization of the sigmoid. Here the $a_k$ are defined by

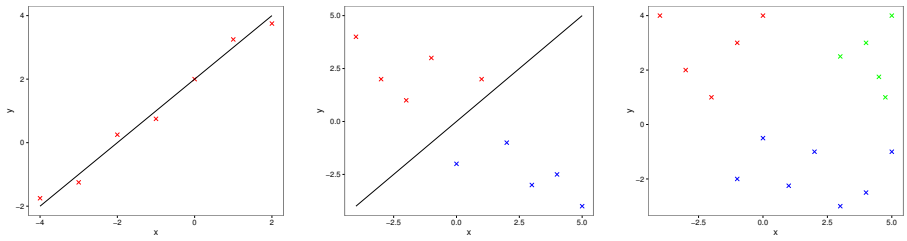
$$a_k = \ln p(\mathbf{x}|C_k)p(C_k) \tag{1.16}$$

One disadvantage of generative models is that they directly estimate densities, which is a very difficult task in high dimensions.

**Discriminative models:** try to model directly the posterior distribution $p(C_k|\mathbf{x})$. One advantage of this approach is that it will have less parameters to be determined. It focuses on estimating classification frontiers, instead of directly estimate densities.

Linear regression models can be generalized to the binary classification setting easily by replacing the Gaussian distribution for $t$ with a Bernoulli distribution and then passing a linear combination of the inputs through the sigmoid function

$$p(t|\mathbf{x}, \mathbf{w}) = Ber(t|\sigma(\mathbf{w}^T\mathbf{x}))\,, \tag{1.17}$$



**(a)** Regression **(b)** Classification **(c)** Clustering

**Figure 1.1:** Examples of machine learning problems

where $Ber(p)$ is a Bernoulli distribution with success probability $p$.This model is called *logistic regression*. However, when using a Bayesian treatment of the logistic regression model, computation of the posterior distribution over the parameters cannot be done exactly, since there is no convenient conjugate prior.

The integral in eq. (1.11) needed to compute the predictive distribution is analytically intractable for logistic regression, so we will have to use an approximated solution as well. In chapter 3 we will see some methods to perform approximate inference, which are very useful to compute approximate posterior and predictive distributions.

## 1.3 Unsupervised learning

Here the goal is to discover some interesting patterns from a set of unlabeled data points. As we are not told what the desired output is for each input, this is a much less well-defined problem.

Some examples of unsupervised learning problems are:

- **Clustering**: It is the most common example of unsupervised learning. The goal is to cluster data into K clusters or groups. k-means is one of the most popular algorithms to perform clustering.
- **Dimensionality reduction**: Sometimes it is convenient to reduce the dimensionality of the data before applying any machine learning algorithm. An example of a method for dimensionality reduction is PCA (Principal Components Analysis).
- **Data completion**: When some of the points in the data are missing we may use unsupervised learning to learn them.

# Chapter 2

# Introduction to Gaussian Processes

In this chapter we will introduce Gaussian processes and how to use them in regression and classification problems.

A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution (Rasmussen and Williams, 2005: p13).

The random variables represent the value of a function $f(\mathbf{x})$ (possibly corrupted by noise) at location $\mathbf{x}$. In our case, $\mathbf{x}$ takes the values of possible inputs.

A Gaussian process is specified by its mean function $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$. We define mean function and the covariance function of a real process $f(\mathbf{x})$ as

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})] \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))] \,. \end{aligned} \tag{2.1}$$

Usually, we will consider the mean function to be equal to zero for simplicity, although it is not strictly necessary.

The covariance function is the crucial ingredient of Gaussian process, it encodes the assumptions about the function we wish to learn. Since a Gaussian process is a collection of random variables, a consistency requirement must be fulfilled. That is, if for example the $\mathcal{GP}$ specifies $(y_1, y_2) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, then it must be true that $y_1 \sim \mathcal{N}(\mu_1, \boldsymbol{\Sigma}_{11})$, where $\mu_1$ is the first element of the mean vector and $\boldsymbol{\Sigma}_{11}$ is a sub-matrix of $\boldsymbol{\Sigma}$. This means that working with a larger set of variables does not change the distribution of the smaller set.

## 2.1 Regression

One can think of a Gaussian process as defining a distribution over functions, and inference taking place directly in the space of functions, this is the function-space view.

In order to perform inference directly in function space, first of all we need to define the covariance function.

**Prediction with Noise-free Observations**

We will first consider the case where the input points are **noise-free**.

All along this thesis we will make use of the RBF (Radial Basis Function) or Gaussian covariance function, which is the most used due to its universality (Micchelli et al., 2006) and to the fact that it is infinitely differentiable, so it is leads to smooth functions (Rasmussen and Williams, 2005: p83). The definition of this function is

$$cov(f(\mathbf{x}_p), f(\mathbf{x}_q)) = k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2 \exp\left(-\sum_{j=1}^{d} \frac{(\mathbf{x}_{pj} - \mathbf{x}_{qj})^2}{2\ell_j^2}\right), \tag{2.2}$$

where the free parameters or *hyperparameters* are:

- $\sigma_f^2$: The signal variance.
- $\ell_j$: The length scale corresponding to the $j$-th dimension.

In Figure 2.1a three functions were generated from a $\mathcal{GP}$ with the RBF covariance function.

It can be shown that this covariance function corresponds to a Bayesian linear regression model with an infinite number of basis functions (Rasmussen and Williams, 2005: p96). The joint distribution of the training output $\mathbf{f}$ and the test output $\mathbf{f}_*$ is

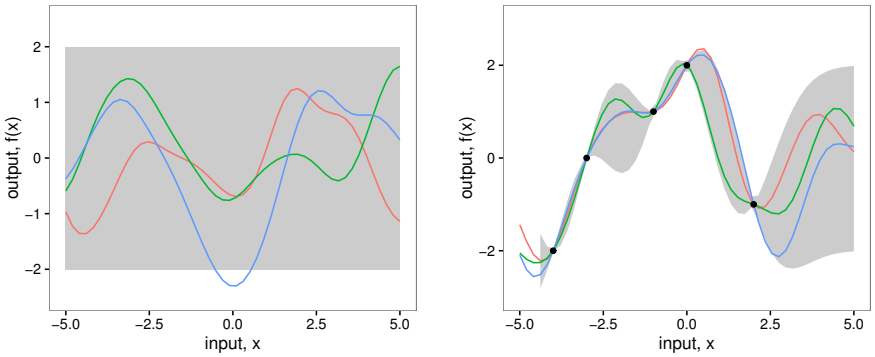
$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(\mathbf{X}, \mathbf{X}) & K(\mathbf{X}, \mathbf{X}_*) \\ K(\mathbf{X}_*, \mathbf{X}) & K(\mathbf{X}_*, \mathbf{X}_*) \end{bmatrix}\right), \tag{2.3}$$

where $\mathbf{X}$ are the training points, $\mathbf{X}_*$ are the test points, $K(\mathbf{A}, \mathbf{B})$ is the covariance matrix between the points in $\mathbf{A}$ and $\mathbf{B}$ and $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the probability density function of a multi-variate Gaussian distribution in $d$ dimensions with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. This is

$$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} \exp(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})). \tag{2.4}$$

What we want then is to get the posterior distribution to draw functions that pass through the training points $\mathbf{X}$. Instead of sampling from this distribution and discard the functions that do not pass through the training points, we are going to condition the joint Gaussian prior on the observations (Roweis, 1999: p2) to give

$$\begin{aligned} \mathbf{f}_* | \mathbf{X}_*, \mathbf{X}, \mathbf{f} \sim \mathcal{N}(&K(\mathbf{X}_*, \mathbf{X}) K(\mathbf{X}, \mathbf{X})^{-1} \mathbf{f}, \\ &K(\mathbf{X}_*, \mathbf{X}_*) - K(\mathbf{X}_*, \mathbf{X}) K(\mathbf{X}, \mathbf{X})^{-1} K(\mathbf{X}, \mathbf{X}_*)). \end{aligned} \tag{2.5}$$




**(a)** $\mathcal{GP}$ prior **(b)** $\mathcal{GP}$ posterior

**Figure 2.1:** Panel (a) shows three functions generated at random from a $\mathcal{GP}$ prior. Panel (b) shows three samples of the posterior distribution given some observed data. The gray area represents the the pointwise mean plus and minus two times the standard deviation for each input value.

We only need to evaluate these expressions for the mean and covariances and sample function values $\mathbf{f}_*$ from the joint distribution.

In Figure 2.1b you can see an example of $\mathcal{GP}$ linear regression where some samples of the posterior distribution are drawn given some input points.

### Prediction with Noisy Observations

In order to have a more realistic model, it is convenient to consider that the inputs that we receive are noisy versions of them ($y = f(\mathbf{x}) + \epsilon$). Assuming $\epsilon$ is additive independent identically distributed Gaussian noise with variance $\sigma_n^2$, the covariance function is now

$$cov(f(\mathbf{x}_p), f(\mathbf{x}_q)) = k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2 \exp\left(-\frac{(\mathbf{x}_p - \mathbf{x}_q)^2}{2\ell^2}\right) + \sigma_n^2 \delta_{pq}\,, \tag{2.6}$$

where $\delta_{pq}$ is a Kronecker delta which is one iff $p = q$ and zero otherwise. The joint distribution is slightly different than the noise-free version of it too

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I} & K(\mathbf{X},\mathbf{X}_*) \\ K(\mathbf{X}_*,\mathbf{X}) & K(\mathbf{X}_*,\mathbf{X}_*) \end{bmatrix}\right). \tag{2.7}$$

With this new definition, the equivalent predictive distribution corresponding to eq. (2.5) is derived, giving

$$\mathbf{f}_*|\mathbf{X},\mathbf{y},\mathbf{X}_* \sim \mathcal{N}(\bar{\mathbf{f}}_*,\ \mathrm{cov}(\mathbf{f}_*))\,, \quad \text{where} \tag{2.8}$$

$$\bar{\mathbf{f}}_* \triangleq \mathbb{E}[\mathbf{f}_*|\mathbf{X},\mathbf{y},\mathbf{X}_*] = K(\mathbf{X}_*,\mathbf{X})[K(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I}]^{-1}\mathbf{y} \tag{2.9}$$

$$\mathrm{cov}(\mathbf{f}_*) = K(\mathbf{X}_*,\mathbf{X}_*) - K(\mathbf{X}_*,\mathbf{X})[K(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I}]^{-1}K(\mathbf{X},\mathbf{X}_*)\,. \tag{2.10}$$

Note that eq. (2.9) is the mean function and eq. (2.10) the covariance function of the posterior process, which is also a Gaussian process.

One of the advantages of the Bayesian treatment of machine learning problems is that model selection can be easily accomplished by means of the marginal likelihood (or evidence) $p(\mathbf{y}|\mathbf{X})$ (Bishop, 2006: p162).

We compute the marginal likelihood by marginalizing over the function values $\mathbf{f}$ (integrating the likelihood times the prior over $\mathbf{f}$)

$$p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|\mathbf{f}, \mathbf{X}) p(\mathbf{f}|\mathbf{X}) d\mathbf{f} \,. \tag{2.11}$$

Because under the Gaussian process model the prior is Gaussian, $p(\mathbf{f}|\mathbf{X}) \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$ with covariance matrix $\mathbf{K}$, and the likelihood is a factorized Gaussian, $p(\mathbf{y}|\mathbf{f}, \mathbf{X}) \sim \mathcal{N}(\mathbf{f}, \sigma_n^2 \mathbf{I})$, we can assure the product will be another Gaussian (Roweis, 1999: p2). Therefore, we can make use of this identities to perform the integration yielding the log marginal likelihood

$$\log p(\mathbf{y}|\mathbf{X}) = -\frac{1}{2}\mathbf{y}^T(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K} + \sigma_n^2\mathbf{I}| - \frac{n}{2}\log 2\pi \,. \tag{2.12}$$

The log marginal likelihood is usually maximized to find the optimal hyperparameters. For that, we need to compute its gradient with respect to the hyperparameters. Let $\theta_j$ be a hyperparameter of the prior. Then, the gradient of $\log p(\mathbf{y}|\mathbf{X})$ with respect to a hyperparameter $\theta_j$ will be (Rasmussen and Williams, 2005):

$$\begin{aligned} \frac{\partial \log p(\mathbf{y}|\mathbf{X})}{\partial \theta_j} &= \frac{1}{2}\mathbf{y}^T\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \theta_j}\mathbf{K}^{-1}\mathbf{y} - \frac{1}{2}\mathrm{tr}\left(\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \theta_j}\right) \\ &= \frac{1}{2}\mathrm{tr}\left(\left(\boldsymbol{\alpha}\boldsymbol{\alpha}^T - \mathbf{K}^{-1}\right)\frac{\partial \mathbf{K}}{\partial \theta_j}\right) \,. \end{aligned} \tag{2.13}$$

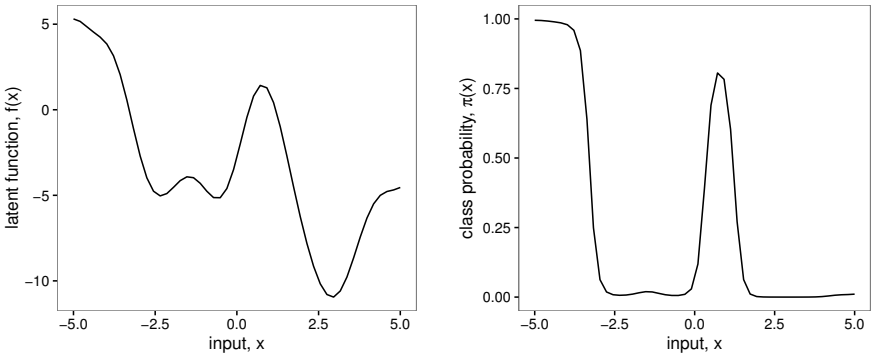
where $\boldsymbol{\alpha} = \mathbf{K}^{-1}\mathbf{y}$ and we have used the identities about the trace and the chain rule for matrix derivatives in (Petersen and Pedersen, 2012).

## 2.2 Binary Classification

In this section we are going to try to find a generalization of the logistic regression model explained in Section 1.2.2.

So as to apply Gaussian processes to the binary classification problem the idea is to place a $\mathcal{GP}$ prior over the latent function $f(\mathbf{x})$ and then pass it through the logistic function to obtain a prior on $\pi(\mathbf{x}) \triangleq p(y = +1|\mathbf{x}) = \sigma(f(\mathbf{x}))$ (Rasmussen and Williams, 2005: p39). In Figures 2.2a and 2.2b an example of the effect of applying a squashing function $\sigma(\cdot)$ on the function $f(x)$ is shown.

Here, we are not particularly interested in the values of the latent function $f$, but in $\pi$, in particular for test cases $\pi(\mathbf{x}_*)$.




**(a)** **(b)**

**Figure 2.2:** Panel (a) shows a sample function $f(x)$ generated from a $\mathcal{GP}$. Panel (b) shows the probability of the class $\pi(x)$ obtained by squashing the function $f(x)$ through the logistic logit function.

First, we need to compute the distribution of the latent variable corresponding to a test case $\mathbf{x}_*$

$$p(f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) = \int p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{f}) p(\mathbf{f}|\mathbf{X}, \mathbf{y}) d\mathbf{f}\,, \tag{2.14}$$

where $p(\mathbf{f}|\mathbf{X}, \mathbf{y}) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{X})/p(\mathbf{y}|\mathbf{X})$ is the posterior distribution over the latent variables of the training data. Given $p(f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*)$ we can use it to make predictions

$$\pi* \triangleq p(y_* = +1|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) = \int \sigma(f_*) p(f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) df_*\,. \tag{2.15}$$

Once again, the problem in classification is that the integral required to compute the marginal likelihood $p(\mathbf{y}|\mathbf{X})$ is analytically intractable due to the non-Gaussian likelihood.

Thus we need to use either analytic approximations of these integrals like the Laplace approximation method (Williams and Barber, 1998) or Monte Carlo methods to make inference in this type of problem. In Chapter 3 we will see some of these methods.

## 2.3 Multi-class Classification

For the multi-class case, each of the $N$ points has $C$ latent functions, one for each class. We define the latent function vector as

$$\begin{aligned}\mathbf{f} &= \left(f^1(\mathbf{x}_1), \ldots, f^1(\mathbf{x}_N), f^2(\mathbf{x}_1), \ldots, f^2(\mathbf{x}_N), \ldots, f^C(\mathbf{x}_1), \ldots, f^C(\mathbf{x}_N)\right) \\ &= (f_1^1, \ldots, f_n^1, f_1^2, \ldots, f_n^2, \ldots, f_1^C, \ldots, f_N^C)\,.\end{aligned} \tag{2.16}$$

The prior over $\mathbf{f}$ has the form $\mathbf{f} \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$ (Rasmussen and Williams, 2005: p48), where the covariance matrix of the prior $\mathbf{K}$ is defined as

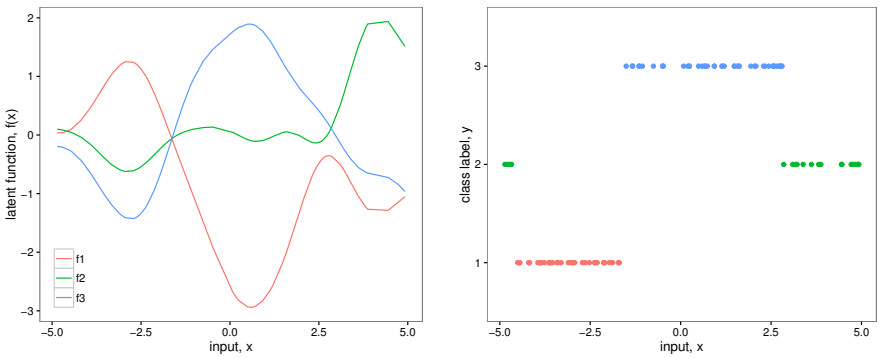
$$\mathbf{K} = \begin{pmatrix} \mathbf{K^1}(\mathbf{X},\mathbf{X}) & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{K^2}(\mathbf{X},\mathbf{X}) & \cdots & \mathbf{0} \\ \vdots & \vdots & \cdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{K^C}(\mathbf{X},\mathbf{X}) \end{pmatrix}, \tag{2.17}$$

where each $\mathbf{K^j}(\mathbf{X},\mathbf{X})$ is the covariance matrix of the latent values related to class $j$ (Kim and Ghahramani, 2006).

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 (Rasmussen and Williams, 2005: p48). Let define $\mathbf{f}_i = (f_i^1,\ldots f_i^C)$ as a vector containing the values of the latent functions for the $i$-th example. The likelihood $p(y_i^c|\mathbf{f}_i)$ at training point $i$ for the class $c$ is defined by using a softmax function as follows

$$p(y_i^c|\mathbf{f}_i) = \pi_i^c = \frac{\exp(f_i^c)}{\sum_{c'}\exp(f_i^{c'})}. \tag{2.18}$$

Then $\boldsymbol{\pi}$ is a vector of the same length as $\mathbf{f}$ with entries $\pi_i^c$ (Rasmussen and Williams, 2005: p48).




**(a)** $\mathcal{GP}$ prior **(b)** Class labels

**Figure 2.3:** One-dimensional classification problem with three classes generated from samples of a $\mathcal{GP}$ prior. Panel (a) shows samples of the $\mathcal{GP}$ prior corresponding to each of the classes. Panel (b) shows the class label for each instance tagged following the rule $y_i = \arg\max_k f_i^k(x)$. Best seen in color.

However, eq. (2.18) is not the only suitable likelihood available for this kind of problems. An alternative approach considers that $y_i = \arg\max_k f_i^k(x)$. This means that the class corresponding to the $i$-th example is the one that takes the maximum latent value for that example. More formally:

$$p(y_i^c|\mathbf{f}_i) = \begin{cases} 1 & \text{if } f_i^c > f_i^k \ \forall k \neq c \\ 0 & \text{otherwise} \end{cases}. \tag{2.19}$$

So the likelihood for the $i$-th input point can be expressed as follows:

$$p(y_i|\mathbf{f}_i) = \prod_{c \neq y_i} \Theta(f_i^{y_i} - f_i^c), \tag{2.20}$$

where $\Theta(\cdot)$ is the Heaviside step function, whose value is $1$ if its argument is positive and $0$ otherwise.

As usual, we can get the posterior of $\mathbf{f}$ by using Bayes' theorem:

$$\begin{aligned} p(\mathbf{f}|\mathbf{X},\mathbf{y}) =& \frac{\left[\prod_i \prod_{c\neq y_i} \Theta(f_i^{y_i} - f_i^c)\right] p(\mathbf{f})}{p(\mathbf{y}|\mathbf{X})} \\ =& \frac{\left[\prod_i \prod_{c\neq y_i} \Theta(f_i^{y_i} - f_i^c)\right] \prod_{k=1}^{C} p(\mathbf{f}^k)}{p(\mathbf{y}|\mathbf{X})}, \end{aligned} \tag{2.21}$$

where $\mathbf{f}^k = (f_1^k, \ldots, f_N^k)$ is a vector containing the values of the latent function associated with the class $k$ for every input point. We will also need some approximation in order to make predictions, because the integral needed to compute the marginal likelihood is not tractable, like in the binary case. In the next chapter, we will see some of these approximate methods to perform inference. The hyperparameter selection will be done by maximizing the approximation to the marginal likelihood $p(\mathbf{y}|\mathbf{X})$.

An issue with the previous likelihood functions is that they do not consider errors in the labels of the data, so over-fitting can become a serious problem when errors far from the decision boundaries are observed (Hernández-Lobato et al., 2011).

Following (Hernández-Lobato et al., 2011) we can define a more robust likelihood function:

$$p(\mathbf{y}|\mathbf{X},\mathbf{z},\mathbf{f}) = \prod_{i=1}^{n} \left( \left[ \prod_{c\neq y_i} \Theta(f_i^{y_i} - f_i^c) \right] (1-\epsilon) + \left[\frac{1}{C}\right] \epsilon \right), \tag{2.22}$$

where $C$ is the number of classes and $\epsilon$ is the probability of a labeling error. It indicates whether the labeling of that instance is wrong and if so, we assume that the label has been randomly selected with uniform probability among the possible classes.

# Chapter 3

# Approximate Inference

In the previous chapter, we have seen that sometimes exact inference is not possible due to the intractability of some of the integrals needed to achieve it. We will present some of the methods and algorithms available to carry out approximate inference.

## 3.1 Deterministic methods

These methods are based on analytical approximations to the posterior distribution, which is replaced with a simpler distribution for which the required expectations can be computed analytically.

### 3.1.1 The Laplace approximation

The idea behind this method is to replace a distribution $p(\mathbf{z})$ over an $M$-dimensional space $\mathbf{z}$ with a Gaussian approximation $q(\mathbf{z})$ (Bishop, 2006: pp213-216). Suppose the distribution $p(\mathbf{z})$ is defined as

$$p(\mathbf{z}) = \frac{1}{Z} f(\mathbf{z}) \,, \tag{3.1}$$

where $Z = \int f(\mathbf{z}) d\mathbf{z}$ is the normalization constant, which is analytically intractable . The goal is to find a Gaussian approximation $q(\mathbf{z})$ centered in the mode of the real distribution $p(\mathbf{z})$.

For that, we need to find its maximum $\mathbf{z}_0$ (where the gradient $\nabla f(\mathbf{z})$ vanishes) and do a second order Taylor expansion of $\log f(\mathbf{z})$ around it

$$\log f(\mathbf{z}) \simeq \log f(\mathbf{z}_0) - \frac{1}{2}(\mathbf{z} - \mathbf{z}_0)^T \mathbf{A} (\mathbf{z} - \mathbf{z}_0) \,, \tag{3.2}$$

where $\mathbf{A} = -\nabla\nabla \log f(\mathbf{z})|_{\mathbf{z}=\mathbf{z}_0}$ is the Hessian evaluated at the mode. Taking the exponential in both sides we get

$$f(\mathbf{z}) \simeq f(\mathbf{z}_0) \exp\left( -\frac{1}{2}(\mathbf{z} - \mathbf{z}_0)^T \mathbf{A} (\mathbf{z} - \mathbf{z}_0) \right) \,, \tag{3.3}$$

which is an unnormalized Gaussian with mean $\mathbf{z}_0$ and covariance $\mathbf{A}^{-1}$. The distribution $q(\mathbf{z})$ is proportional to $f(\mathbf{z})$ and the normalization constant is found using the standard constant for a multivariate Gaussian

$$q(\mathbf{z}) = \frac{|\mathbf{A}|^{1/2}}{2\pi^{M/2}} \exp\left( -\frac{1}{2}(\mathbf{z} - \mathbf{z}_0)^T \mathbf{A} (\mathbf{z} - \mathbf{z}_0) \right) = \mathcal{N}(\mathbf{z}|\mathbf{z}_0, \mathbf{A}^{-1}) \,, \tag{3.4}$$

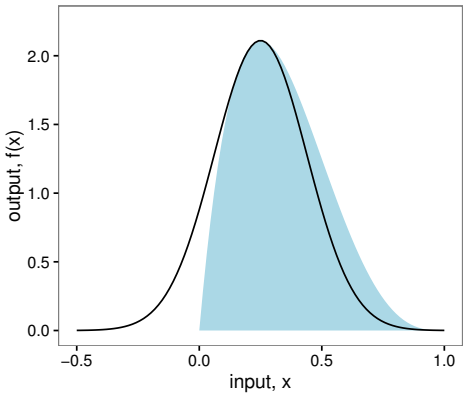



**Figure 3.1:** Laplace approximation to a Beta distribution. The blue shaded curve is the real Beta distribution and the black curve is the Laplace approximation to that distribution.

where $M$ is the number of dimensions.

In Figure 3.1 an example of a Laplace approximation to a Beta distribution is shown.

### 3.1.2 Variational Inference

The Kullback–Leibler divergence is a quantity that measures the similarity between two probability distributions $p$ and $q$ (Kullback and Leibler, 1951). It is always positive and equal to zero if and only if $p = q$.

Let $\mathbf{Z}$ be the set of latent variables and parameters and $\mathbf{X}$ the set of observed data. Our goal is to find an approximation for the posterior distribution $p(\mathbf{Z}|\mathbf{X})$ as well as for the model evidence $p(\mathbf{X})$ (Bishop, 2006: p463). The KL divergence is defined as follows

$$\mathrm{KL}(q \parallel p) = -\int q(\mathbf{Z}) \log \left\{ \frac{p(\mathbf{Z}|\mathbf{X})}{q(\mathbf{Z})} \right\} d\mathbf{Z} \geq 0\,. \tag{3.5}$$

We want to find the latent variables and parameters such that this quantity is minimized. However, this is hard to compute, since $p(\mathbf{Z}|\mathbf{X})$ is assumed to be intractable. To bypass this problem we can decompose the log marginal probability using

$$\log p(\mathbf{X}) = \mathcal{L}(q) + \mathrm{KL}(q \parallel p) \geq \mathcal{L}(q)\,, \tag{3.6}$$

where the lower bound $\mathcal{L}(q)$ is

$$\mathcal{L}(q) = \int q(\mathbf{Z}) \log \left\{ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right\} d\mathbf{Z}\,. \tag{3.7}$$

It is easy to see that minimizing the KL divergence is equivalent to maximizing the lower bound $\mathcal{L}(q)$ because $\log p(\mathbf{X})$ is constant. This is illustrated in Figure 3.2.

In general we will be not able to work directly with the posterior distribution, thus we need to restrict the family of distributions $q(\mathbf{Z})$ in which to

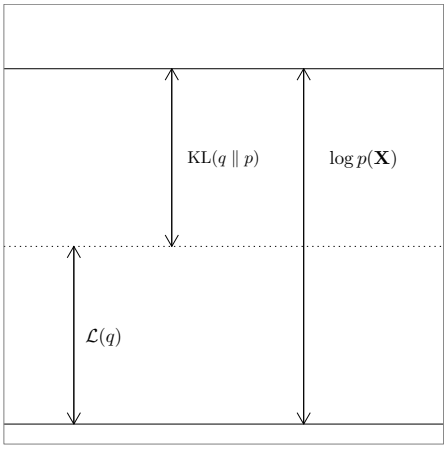



**Figure 3.2:** Decomposition of the log marginal probability

seek. This restriction is imposed to achieve tractability but we must try to use a family of approximations as rich as possible.

In order to restrict the family of distributions, we can use a parametric distribution $p(\mathbf{Z}|\omega)$ with a set of parameters $\omega$, e.g a Gaussian distribution, and then apply some of the standard nonlinear optimization techniques, e.g. gradient ascent, to maximize $\mathcal{L}(q)$, which is equivalent to minimizing $\mathrm{KL}(q \parallel p)$.

Another option is to partition the elements of $\mathbf{Z}$ in $M$ disjoint groups $Z_i$ and then assume that the distribution $q(\mathbf{Z})$ factorizes with respect to these groups, so that

$$q(\mathbf{Z}) = \prod_{i=1}^{M} q_i(\mathbf{Z}_i)\,. \tag{3.8}$$

This approach is called **mean field theory** (Parisi, 1988). Substituting (3.8) in (3.7) we get

$$\begin{aligned}
\mathcal{L}(q) &= \int \prod_{i=1}^{M} q_i(\mathbf{Z}_i) \left\{ \log p(\mathbf{X}, \mathbf{Z}) - \sum_{i=1}^{M} \log q_i(\mathbf{Z}_i) \right\} d\mathbf{Z} \\
&= \int q_j(\mathbf{Z_j}) \left\{ \int \log p(\mathbf{X}, \mathbf{Z}) \prod_{i \neq j} q_i(\mathbf{Z}_i) d\mathbf{Z_i} \right\} d\mathbf{Z_j} \\
&\quad - \int q_j(\mathbf{Z}_j) \log q_j(\mathbf{Z}_j) d\mathbf{Z_j} + \text{const} \\
&= \int q_j(\mathbf{Z_j}) \log \tilde{p}(\mathbf{X}, \mathbf{Z_j}) d\mathbf{Z_j} - \int q_j(\mathbf{Z}_j) \log q_j(\mathbf{Z}_j) d\mathbf{Z_j} + \text{const}\,,
\end{aligned} \tag{3.9}$$

where we have taken out the factor $q_j(\mathbf{Z_j})$ and we have defined the distribution $\log \tilde{p}(\mathbf{X}, \mathbf{Z_j})$ to be

$$\log \tilde{p}(\mathbf{X}, \mathbf{Z_j}) = \mathbb{E}_{i \neq j}[p(\mathbf{X}, \mathbf{Z})] + \text{const}\,. \tag{3.10}$$

The optimal solution for $\log q_j(\mathbf{Z_j})$ is obtained by fixing the values of the other factors and noting that (3.9) is a negative KL between $q_j$ and $\tilde{p}$. This means that the optimal $q_j$ has the following form:

$$\log q_j^*(\mathbf{Z_j}) = \mathbb{E}_{i \neq j}[\log p(\mathbf{X}, \mathbf{Z})] + \text{const}\,. \tag{3.11}$$

Taking the exponential and normalizing we get the solution

$$q_j^*(\mathbf{Z_j}) = \frac{\exp(\mathbb{E}_{i \neq j}[\log p(\mathbf{X}, \mathbf{Z})])}{\int \exp(\mathbb{E}_{i \neq j}[\log p(\mathbf{X}, \mathbf{Z})]) d\mathbf{Z_j}}\,. \tag{3.12}$$

### 3.1.3 Expectation Propagation

The last of the deterministic methods we are going describe is *Expectation Propagation* or EP (Minka, 2001).

This algorithm also minimizes in an approximate way the KL divergence but in the reverse form $\text{KL}(p \parallel q)$, instead of $\text{KL}(q \parallel p)$, which is believed to work better.

In EP, $q(\mathbf{z})$ needs to be a member of the exponential family, which is closed under product and division (Seeger, 2005: p2). This is

$$q(\mathbf{z}) = \exp(\boldsymbol{\eta}^T \mathbf{u}(\mathbf{z}) - g(\boldsymbol{\eta}))\,, \tag{3.13}$$

where $\boldsymbol{\eta}$ is a vector of the natural parameters of $q$, $\mathbf{u}(\mathbf{z})$ are the sufficient statistics and $g(\boldsymbol{\eta})$ is the **log normalizer**, which forces $q$ to be normalized (see Appendix A). This is

$$g(\boldsymbol{\eta}) = \log \int \exp(\boldsymbol{\eta}^T \mathbf{u}(\mathbf{z})) d\mathbf{z}\,. \tag{3.14}$$

If we express the KL divergence in terms of $\boldsymbol{\eta}$ we get

$$\text{KL}(p \parallel q) = g(\boldsymbol{\eta}) - \boldsymbol{\eta}^T \mathbb{E}_{p(\mathbf{z})}[\mathbf{u}(\mathbf{z})] + \text{const}\,. \tag{3.15}$$

We want to minimize this function. For that, we set the gradient with respect to $\boldsymbol{\eta}$ equal to zero, giving

$$\frac{\partial \text{KL}(p \parallel q)}{\partial \boldsymbol{\eta}} = 0 \iff \frac{\partial g(\boldsymbol{\eta})}{\partial \boldsymbol{\eta}} = \mathbb{E}_{p(\mathbf{z})}[\mathbf{u}(\mathbf{z})]\,. \tag{3.16}$$

By computing the gradient of eq. (3.14) with respect to $\boldsymbol{\eta}$ we get

$$\begin{aligned} \frac{\partial g(\boldsymbol{\eta})}{\partial \boldsymbol{\eta}} &= \frac{\partial \log \int \exp(\boldsymbol{\eta}^T \mathbf{u}(\mathbf{z})) d\mathbf{z}}{\partial \boldsymbol{\eta}} \\ &= \frac{1}{\log \int \exp(\boldsymbol{\eta}^T \mathbf{u}(\mathbf{z})) d\mathbf{z}} \frac{\partial \int \exp(\boldsymbol{\eta}^T \mathbf{u}(\mathbf{z})) d\mathbf{z}}{\partial \boldsymbol{\eta}} \\ &= \frac{\int \mathbf{u}(\mathbf{z}) \exp(\boldsymbol{\eta}^T \mathbf{u}(\mathbf{z})) d\mathbf{z}}{\log \int \exp(\boldsymbol{\eta}^T \mathbf{u}(\mathbf{z})) d\mathbf{z}} = \mathbb{E}_{q(\mathbf{z})}[\mathbf{u}(\mathbf{z})]\,. \end{aligned} \tag{3.17}$$

If we combine eq. (3.16) and eq. (3.17) we get:

$$\mathbb{E}_{p(\mathbf{z})}[\mathbf{u}(\mathbf{z})] = \mathbb{E}_{q(\mathbf{z})}[\mathbf{u}(\mathbf{z})]\,, \tag{3.18}$$

and we can conclude that minimizing the KL divergence between two distributions is equal to matching their moments (expected sufficient statistics).

Minimizing $\mathrm{KL}(p \parallel q)$ directly is not tractable analytically. Nevertheless, with EP we can do it in an iterative way.

For many probabilistic models, the joint distribution of data $\mathcal{D}$ and hidden variables (including parameters) $\boldsymbol{\theta}$ comprises a product of factors (Bishop, 2006: p506) in the form

$$p(\mathcal{D}, \boldsymbol{\theta}) = \prod_i f_i(\boldsymbol{\theta}) \,. \tag{3.19}$$

This is the case of independent identically distributed data. Then, by using the Bayes' theorem we know that the posterior distribution is

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{1}{p(\mathcal{D})} \prod_i f_i(\boldsymbol{\theta}) \,. \tag{3.20}$$

Expectation propagation assumes that the approximate posterior factorizes too in a set of approximate factors, each one corresponding to one of the factors in the real posterior

$$q(\boldsymbol{\theta}) = \frac{1}{Z} \prod_i \tilde{f}_i(\boldsymbol{\theta}) \,, \tag{3.21}$$

so that $\tilde{f}_i \simeq f_i$ in regions of high posterior probability and where the normalization constant $Z$ approximates the marginal likelihood $p(\mathcal{D})$. Then, it refines iteratively each approximate factor. For that, in each iteration it removes one of the factors first by defining the unnormalized cavity distribution

$$q^{\setminus j}(\boldsymbol{\theta}) = \frac{q(\boldsymbol{\theta})}{\tilde{f}_j(\boldsymbol{\theta})} \,. \tag{3.22}$$

$q^{\setminus j}$ will have similar form to that of $q$ because of the closure property of the exponential family. For example, if $q$ is Gaussian, $q^{\setminus j}$ will also be Gaussian. After that, it updates the distribution by multiplying by the actual factor and normalizing

$$\frac{1}{Z_j} q^{\setminus j}(\boldsymbol{\theta}) f_j(\boldsymbol{\theta}) = \hat{p}(\boldsymbol{\theta}) \,, \tag{3.23}$$

where the normalization constant is defined as

$$Z_j = \int q^{\setminus j}(\boldsymbol{\theta}) f_j(\boldsymbol{\theta}) d\boldsymbol{\theta} \,. \tag{3.24}$$

Now, $q^{new}(\boldsymbol{\theta})$ is updated by setting its moments equal to those of $\hat{p}(\boldsymbol{\theta})$. This minimizes $\mathrm{KL}[\hat{p} \parallel q]$

The factor $\tilde{f}_j(\boldsymbol{\theta})$ can be found by evaluating the expression

$$\tilde{f}_j(\boldsymbol{\theta}) = Z_j \frac{q^{new}(\boldsymbol{\theta})}{q^{\setminus j}(\boldsymbol{\theta})} \,. \tag{3.25}$$

This guarantees that $q_{new}$ is obtained when $f_j$ is replaced by $\tilde{f}_j$ in (3.23), and that $f_j$ and $\tilde{f}_j$ integrate the same with respect to $q^{\setminus j}$. Namely, $Z_j$.

**Algorithm 1** Expectation Propagation

1: Initialize the approximate factors $\tilde{f}_i(\boldsymbol{\theta})$ to be uniform
2: Initialize the posterior approximation by multiplying all the factors together

$$q(\boldsymbol{\theta}) \propto \prod_i \tilde{f}_i(\boldsymbol{\theta})$$

3: **repeat**
4: Choose a factor $\tilde{f}_j(\boldsymbol{\theta})$ to refine
5: Remove that factor from the approximation

$$q^{\setminus j}(\boldsymbol{\theta}) = \frac{q(\boldsymbol{\theta})}{\tilde{f}_j(\boldsymbol{\theta})}$$

6: Evaluate the new posterior $q^{new}(\boldsymbol{\theta})$ by setting its moments equal to those of $q^{\setminus j}(\boldsymbol{\theta}) f_j(\boldsymbol{\theta})$ and compute the value of the normalization constant

$$Z_j = \int q^{\setminus j}(\boldsymbol{\theta}) f_j(\boldsymbol{\theta}) d\boldsymbol{\theta}$$

7: Find the new factor $\tilde{f}_j(\boldsymbol{\theta})$

$$\tilde{f}_j(\boldsymbol{\theta}) = K \frac{q^{new}(\boldsymbol{\theta})}{q^{\setminus j}(\boldsymbol{\theta})}$$

8: **until** convergence
9: Find the approximation to the model evidence

$$p(\mathcal{D}) = \int \prod_i \tilde{f}_i(\boldsymbol{\theta}) d\boldsymbol{\theta}$$

As we said before, the exponential family is closed under product and division. As a consequence, $\prod_i \tilde{f}_i(\boldsymbol{\theta})$ has a simple form and $\int \prod_i \tilde{f}_i(\boldsymbol{\theta}) d\boldsymbol{\theta}$ is easy to compute.

Once EP has converged, the moments of $p(\mathbf{z})$ and $q(\mathbf{z})$ match, so the factors $\tilde{f}_i(\boldsymbol{\theta})$ can be considered to be fixed (Seeger, 2005). This simplifies the computations needed for the gradient of the marginal likelihood with respect to the hyperparameters.

## 3.2 Monte Carlo techniques

Often, we do not need to know the posterior distribution itself, but to compute expectations with respect to it (Bishop, 2006: p523)

$$\mathbb{E}_{p(\mathbf{z})}[f(\mathbf{z})] = \int f(\mathbf{z}) p(\mathbf{z}) d\mathbf{z}\,. \tag{3.26}$$

These expectations can be approximated by numerical sampling, also known as *Monte Carlo* techniques. These methods do not have bias, but their computational cost is higher.

The idea is to draw samples from the distribution $p(\mathbf{z})$ and then compute the arithmetic mean of the function applied to the samples (Murphy, 2012: p53). Thus

$$\mathbb{E}_{p(\mathbf{z})}[f(\mathbf{z})] = \int f(\mathbf{z})p(\mathbf{z})d\mathbf{z} \approx \frac{1}{N}\sum_{i=1}^{N} f(z_i)\,, \tag{3.27}$$

where the $z_i$ are drawn independently from $p(\mathbf{z})$.

These methods may achieve high accuracy with relatively small number of samples.

However, the main difficulty with this kind of techniques is how to draw samples from the distribution $p(\mathbf{z})$.

### 3.2.1 Sampling from standard distributions

If we want to draw samples from a distribution of standard form, we can use the **inverse probability transform**. Let $F$ be the *cumulative distribution function* or cdf of some distribution we want to sample from, and let $F^{-1}$ be its inverse (Murphy, 2012: p815).

The inverse probability transform states that given a uniform random variable $U \sim U(0,1)$, then $F^{-1}(U) \sim F$.

*Proof.*

$$Pr(F^{-1}(U) \leq x) = Pr(U \leq F(x)) \quad \text{(applying F to both sides)} \tag{3.28}$$
$$= F(x) \quad \text{(because } Pr(U \leq y) = y\text{)}\,, \tag{3.29}$$

where the first line follows since $F$ is a monotonic function, and the second line follows since $U$ is uniform on the unit interval (Murphy, 2012: p816). □

This means that we can draw samples $x$ from a distribution with cumulative distribution function $F$ by generating random samples $u \sim U(0,1)$ and then computing $x = F^{-1}(u) \sim f(x)$, where $f(x)$ is its probability density function.

### 3.2.2 Rejection sampling

An alternative when the inverse probability transform cannot be used is **rejection sampling**.

Suppose that the objective function from which we want to draw samples has the following form:

$$p(\mathbf{z}) = \frac{1}{Z_p}\tilde{p}(\mathbf{z})\,, \tag{3.30}$$

where we can readily draw samples from $\tilde{p}(\mathbf{z})$ and $Z_p$ is the normalization constant.

We define a simpler distribution $q(\mathbf{z})$ called *proposal distribution* and a constant $k$ whose value is chosen such that $kq(\mathbf{z}) \geq \tilde{p}(\mathbf{z})$. The function $kq(\mathbf{z})$ is called a *comparison function* (Bishop, 2006: p529) and provides an upper envelope for $\tilde{p}(\mathbf{z})$ (Murphy, 2012: p817).

We first sample $z_0 \sim q(\mathbf{z})$, which corresponds to picking a random $x$ location, and then we sample $u_0 \sim U[0, kq(z_0)]$, which corresponds to picking a random height. We accept the generated sample $u_0$ if $u_0 \leq \tilde{p}(z_0)$, we reject it otherwise (Bishop, 2006: p529).

The success of this approximation depends on how well the sampling distribution $q(\mathbf{z})$ matches the desired distribution $p(\mathbf{z})$.

### 3.2.3 Importance sampling

Sometimes the purpose of sampling from a certain distribution is to compute expectations of the following form:

$$\mathbb{E}[f] = \int f(\mathbf{z})p(\mathbf{z})d\mathbf{z}\,, \tag{3.31}$$

which can be done through the **importance sampling** framework. The idea is, like in rejection sampling, to use a proposal distribution $q(\mathbf{z})$, from which it is easy to draw samples. Then the expectation 3.31 can be expressed as:

$$\mathbb{E}[f] = \int f(\mathbf{z})\frac{p(\mathbf{z})}{q(\mathbf{z})}q(\mathbf{z})d\mathbf{z}\,. \tag{3.32}$$

We can then approximate the expectation by summing over a finite number of samples $\{\mathbf{z}^{(l)}\}$ drawn from $q(\mathbf{z})$ in the following way (Bishop, 2006: p533):

$$\mathbb{E}[f] \simeq \frac{1}{L}\sum_{l=1}^{L}\frac{p(\mathbf{z}^{(l)})}{q(\mathbf{z}^{(l)})}f(\mathbf{z}^{(l)})\,, \tag{3.33}$$

where the quantities $p(\mathbf{z}^{(l)})/q(\mathbf{z}^{(l)})$ are known as *importance weights*.

It is often the case that we can evaluate the unnormalized target distribution $\tilde{p}(\mathbf{z})$, but not its normalization constant $Z_p$ (Murphy, 2012: p821). We may wish to use a proposal distribution $q(\mathbf{z}) = \tilde{q}(\mathbf{z})/Z_q$, where $Z_q$ could also be unknown, although most times it is known. We then have (Bishop, 2006: p533):

$$\begin{aligned}\mathbb{E}[f] &= \frac{Z_q}{Z_p}\int f(\mathbf{z})\frac{\tilde{p}(\mathbf{z})}{\tilde{q}(\mathbf{z})}q(\mathbf{z})d\mathbf{z} \\ &\simeq \frac{Z_q}{Z_p}\frac{1}{L}\sum_{l=1}^{L}\frac{\tilde{p}(\mathbf{z}^{(l)})}{\tilde{q}(\mathbf{z}^{(l)})}f(\mathbf{z}^{(l)})\,.\end{aligned} \tag{3.34}$$

The ratio $Z_p/Z_q$ can be evaluated as follows:

$$\begin{aligned}\frac{Z_p}{Z_q} &= \frac{1}{Z_q}\int \tilde{p}(\mathbf{z})d\mathbf{z} = \int \frac{\tilde{p}(\mathbf{z})}{\tilde{q}(\mathbf{z})}q(\mathbf{z})d\mathbf{z} \\ &\simeq \frac{1}{L}\sum_{l=1}^{L}\frac{\tilde{p}(\mathbf{z}^{(l)})}{\tilde{q}(\mathbf{z}^{(l)})}\,,\end{aligned} \tag{3.35}$$

and hence we can redefine $\mathbb{E}[f]$ as:

$$\mathbb{E}[f] \simeq \sum_{l=1}^{L}\frac{\tilde{p}(\mathbf{z}^{(l)})/\tilde{q}(\mathbf{z}^{(l)})}{\sum_m \tilde{p}(\mathbf{z}^{(m)})/\tilde{q}(\mathbf{z}^{(m)})}\,. \tag{3.36}$$

As with rejection sampling, the success of the importance sampling approach depends crucially on how well the sampling distribution $q(\mathbf{z})$ matches the desired distribution $p(\mathbf{z})$ (Bishop, 2006: p534).

## 3.3 Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) is a more general and powerful framework for sampling from a wider range of target distributions and which scales well with the dimensionality of the sample space.

Here we also have a proposal distribution, but now we keep the current state $\mathbf{z}^{(\tau)}$ and the proposal distribution $q(\mathbf{z}|\mathbf{z}^{(\tau)})$ depends on this current state (Bishop, 2006: p537). The sequence of the states $\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \ldots$ forms a Markov chain.

### 3.3.1 Metropolis-Hastings algorithm

The *Metropolis-Hastings* algorithms (Hastings, 1970) is a generalization of the more basic Metropolis algorithm (Metropolis et al., 1953), which considers a symmetric proposal distribution, that is $q(\mathbf{z}_A|\mathbf{z}_B) = q(\mathbf{z}_B|\mathbf{z}_A)$ for all values of $\mathbf{z}_A$ and $\mathbf{z}_B$ (Bishop, 2006: p539).

In this case the proposal distribution is no longer symmetric. At step $\tau$, in which the current state is $\mathbf{z}^{\tau}$, we draw a sample $\mathbf{z}^*$ from $q(\mathbf{z}|\mathbf{z}^{(\tau)})$. The sample is accepted with probability $A(\mathbf{z}^*, \mathbf{z}^{(\tau)})$, where:

$$A(\mathbf{z}^*, \mathbf{z}^{(\tau)}) = min\left(1, \frac{\tilde{p}(\mathbf{z}^*)q(\mathbf{z}^{(\tau)}|\mathbf{z}^*)}{\tilde{p}(\mathbf{z}^{(\tau)})q(\mathbf{z}^*|\mathbf{z}^{(\tau)})}\right)\,. \tag{3.37}$$

If the candidate sample is accepted, then $\mathbf{z}^{(\tau+1)} = \mathbf{z}^*$, otherwise $\mathbf{z}^*$ is discarded (Bishop, 2006: p538).

### 3.3.2 Gibbs sampling

Gibbs sampling (Geman and Geman, 1984) can be seen as a special case of the Metropolis-Hastings algorithm.

Consider the distribution $p(\mathbf{z}) = p(z_1, z_2, \ldots, z_m)$. This algorithm replaces at each step one of the variables by a new sample drawn from the distribution of that variable conditioned on the values of all the other variables (Bishop, 2006: p542).

**Algorithm 2** Gibbs Sampling

1: Initialize $\{z_i : i = 1, 2, \ldots, m\}$
2: **for** $\tau = 1, \ldots, T$ **do**
3: Sample $z_1^{\tau+1} \sim p(z_1 | z_2^{(\tau)}, z_3^{(\tau)}, \ldots, z_m^{(\tau)})$
4: Sample $z_2^{\tau+1} \sim p(z_2 | z_1^{(\tau+1)}, z_3^{(\tau)}, \ldots, z_m^{(\tau)})$
5: $\ldots$
6: Sample $z_m^{\tau+1} \sim p(z_m | z_1^{(\tau+1)}, z_2^{(\tau+1)}, \ldots, z_{m-1}^{(\tau+1)})$
7: **end for**

Both Gibbs sampling and Metropolis-Hastings generate Markov chains with a stationary distribution equal to the target distribution $p(\mathbf{z})$, so the samples of this Markov chains approximate $p(\mathbf{z})$. Nevertheless, the generated samples are not independent and we need to generate a lot to obtain just a few independent samples from $p(\mathbf{z})$.

# Chapter 4

# Scalable Gaussian Processes

## 4.1 Introduction

A significant problem with Gaussian process prediction is that it typically scales as $O(N^3)$ where $N$ is the number of instances. In addition, for large datasets, the storage of the Gram matrix and solving the associated linear systems can be prohibitive (Rasmussen and Williams, 2005: p171).

Over the years, there have been a lot of proposals to deal with this problem. In this chapter we present some of these approximations.

### 4.1.1 Reduced-rank Approximations of the Gram Matrix

To solve the $\mathcal{GP}$ regression problem, we need to invert the matrix $\mathbf{K} + \sigma_n^2\mathbf{I}$ (Rasmussen and Williams, 2005: p171). If the matrix $\mathbf{K}$ has rank $q$, we can decompose it like $\mathbf{K} = \mathbf{Q}\mathbf{Q}^T$ and then apply the matrix inversion lemma (Press et al., 1992: p75) to simplify the calculations

$$(\mathbf{Q}\mathbf{Q}^T + \sigma_n^2\mathbf{I}_n)^{-1} = \sigma_n^{-2}\mathbf{I}_n - \sigma_n^{-2}\mathbf{Q}(\sigma_n^2\mathbf{I}_q + \mathbf{Q}^T\mathbf{Q})^{-1}\mathbf{Q}^T\,. \tag{4.1}$$

Now the matrix to be inverted is a $q$ x $q$ matrix.

It is not always the case that $\mathbf{K}$ is low rank. Nevertheless, we can still use a reduced-rank approximation by using its decomposition in eigenvalues and eigenvectors (Golub and Van Loan, 2012). But computing the eigendecomposition is an $O(N^3)$ operation, so we need to find an approximation as well.

For that, we can use the Nyström approximation (Press et al., 1992). Consider selecting a subset $I$ of the data of size $M < N$, called the inducing points. The remaining $(N - M)$ form the subset $R$. Then $\mathbf{K}$ has the form

$$\mathbf{K} = \begin{pmatrix} \mathbf{K}_{MM} & \mathbf{K}_{M(N-M)} \\ \mathbf{K}_{(M-M)M} & \mathbf{K}_{(N-M)(N-M)} \end{pmatrix} \tag{4.2}$$

where $\mathbf{K}_{MM}$ is the covariance matrix for the inducing points and $\mathbf{K}_{M(N-M)}$ is the covariance matrix between the inducing points and the observed data points. $\mathbf{K}_{M(N-M)}$ will be referred to as $\mathbf{K}_{MN}$ (and its transpose as $\mathbf{K}_{NM}$) and the eigenvalues and eigenvectors of $\mathbf{K}_{MM}$ are denoted by $\{\lambda_i^{(m)}\}_{i=1}^M$ and $\{\mathbf{u}_i^{(m)}\}_{i=1}^M$. These are extended to the $N$ points

$$\tilde{\lambda}_i^{(n)} \triangleq \frac{N}{M}\lambda_i^{(m)}, \qquad i = 1, \ldots, M \tag{4.3}$$

$$\mathbf{u}_i^{(n)} \triangleq \sqrt{\frac{M}{N}}\frac{1}{\lambda_i^{(m)}}\mathbf{K}_{NM}\mathbf{u}_i^{(m)}, \qquad i = 1, \ldots, M \tag{4.4}$$

If we take the first $M$ eigenvalues we obtain an approximation to $\mathbf{K}$ as

$$\tilde{\mathbf{K}} = \mathbf{K}_{NM}\mathbf{K}_{MM}^{-1}\mathbf{K}_{MN} \tag{4.5}$$

where $\tilde{\mathbf{K}}$ is the Nyström approximation to $\mathbf{K}$, and its computation takes $O(M^2N)$ (Rasmussen and Williams, 2005: p172).

### 4.1.2 Greedy Approximation

Many of the methods described next in this chapter use an active set $I$ of data points of size $M$ (see previous section), and it is computationally impossible to find the optimal subset of this size (Rasmussen and Williams, 2005: p174).

Instead of selecting the active set randomly, it is more suitable to use a greedy algorithm where we start with an empty active set $I$ and a set $R$ with the indices of all training examples. Then, iteratively add one point that optimizes a criterion $\Delta$. As sometimes computing $\Delta$ for all the points is unfeasible, it can be first chosen a subset $J \subset R$, usually at random from $R$ (Rasmussen and Williams, 2005: p174).

In Algorithm 3 a general framework for greedy approximation is given.

**Algorithm 3** Greedy approximation

1: **input**: m, the subset size
2: Initialize $I = \emptyset$, $R = \{1, \ldots, N\}$
3: **for** $j := 1 \ldots M$ **do**
4: Select working set $J \subset R$
5: Compute $\Delta_j$ for all $j \in J$
6: $i = \text{argmax}_{j \in J}\Delta_j$
7: Include input example $i$ in $I$
8: **end for**
9: **return**: $I$

## 4.2 Approximations for GPR with Fixed Hyperparameters

In this section, we will present some methods to perform Gaussian process regression (GPR) that use the techniques described in Section 4.1 and consider the hyperparameters to be fixed.

### 4.2.1 Subset of Regressors (SR)

As shown in (Silverman, 1985), a $\mathcal{GP}$ regressor can be obtained from a generalized linear regression model

$$f(\mathbf{x}_*) = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_*, \mathbf{x}_i) \tag{4.6}$$

with a prior $\boldsymbol{\alpha} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}^{-1})$.

A simple approximation can be obtained by using only a subset of $M$ regressors

$$f_{SR}(\mathbf{x}_*) = \sum_{i=1}^{M} \alpha_i k(\mathbf{x}_*, \mathbf{x}_i) \tag{4.7}$$

with a prior $\boldsymbol{\alpha}_m \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_{MM}^{-1})$.

This subset can be selected randomly or in a greedy manner. The time complexity to perform the necessary matrix operations is $O(M^2N)$, the prediction of the mean for a new point takes $O(M)$, and the predictive variance takes $O(M^2)$ (Rasmussen and Williams, 2005: p176).

This method was proposed, among others, in (Wahba, 1990).

### 4.2.2 The Nyström Method

This method simply replaces the matrix $\mathbf{K}$ by the Nyström approximation $\tilde{\mathbf{K}}$ in the mean and variance prediction (Williams and Seeger, 2001).

The time complexity to perform the matrix operations needed is $O(M^2N)$, $O(N)$ for the predictive mean of a new point and $O(MN)$ for the predictive variance (Rasmussen and Williams, 2005: p177).

In (Williams et al., 2002) it is suggested that the performance of this approximation is similar to $SR$ for large $M$, but for small $M$ the performance of the Nyström method can be quite poor. However, this method can perform well when the $(M+1)$-th eigenvalue of $\mathbf{K}$ is much smaller than the noise variance $\sigma_n^2$ (Rasmussen and Williams, 2005: p177).

### 4.2.3 Subset of Datapoints (SD)

An alternative to the subset of regressors method is to keep the full regressor and use a subset of $M$ data points.

The problem again is how to select these data points in an efficient way, such that the predictions are sufficiently accurate. Typically this is achieved by greedy algorithms (Rasmussen and Williams, 2005: p177).

A criterion to choose the next data point (or site) to include in the active set $I$ is the *differential entropy score*, as suggested in (Lawrence et al., 2003). It is defined as

$$\Delta_j \triangleq H[p(f_j)] - H[p^{new}(f_j)] \tag{4.8}$$

where $H[p(f_j)]$ is the entropy of the Gaussian at site $j \in J$ and $H[p^{new}(f_j)]$ is the entropy of the Gaussian after the inclusion of the point $j$.

The inversion of the matrix $\mathbf{K}_{MM} + \sigma_n^2\mathbf{I}$ can be done in $O(M^3)$, but if we consider incrementally growing the matrices, the cost is $O(MN)$ per inclusion, giving an overall cost of $O(M^2N)$. This can be done by using use a more numerically efficient strategy, like the Cholesky decomposition, proposed in (Lawrence et al., 2003).

As the number of points selected $M$ grows it makes sense to select a smaller subset in which to evaluate $\Delta_j$. This approach is called *randomized greedy selection method* (Lawrence et al., 2003).

There are other criteria available in order to choose the next site, e.g. the *information gain* criterion (Seeger et al., 2003).

### 4.2.4 Projected Process Approximation (PP)

This approximation selects only $M < N$ latent function values, but projects them up to $N$ dimensions when computing the likelihood (Rasmussen and Williams, 2005: p178).

We are going to denote the $M$ f-values in $I$ as $\mathbf{f}_M$, and the remaining $N - M$ in $R$ as $\mathbf{f}_{N-M}$. These last values will have a conditional distribution $p(\mathbf{f}_{N-M}|\mathbf{f}_M)$, the mean of which is given by $\mathbb{E}[p(\mathbf{f}_{N-M}|\mathbf{f}_M)] = \mathbf{K}_{(n-m)m}\mathbf{K}_{MM}^{-1}\mathbf{f}_M$ (Rasmussen and Williams, 2005: p179). Provided that we replace the true likelihood term for the points in $R$ by $\mathcal{N}(\mathbf{y}_{NM}|\mathbb{E}[\mathbf{f}_{N-M}|\mathbf{f}_M], \sigma_n^2\mathbf{I})$, if we include also the likelihood contribution of the points in $I$ we get

$$q(\mathbf{y}|\mathbf{f}_M) = \mathcal{N}(\mathbf{y}|\mathbf{K}_{NM}\mathbf{K}_{MM}^{-1}\mathbf{f}_M, \sigma_n^2\mathbf{I}) = \mathcal{N}(\mathbf{y}|\mathbb{E}[p(\mathbf{f}|\mathbf{f}_M)], \sigma_n^2\mathbf{I}) \tag{4.9}$$

where the information of the $N$ points is present into the $M$ points of $I$.

As for the SR model the time complexity of the needed matrix operations is $O(M^2N)$, the computation of the predictive mean for a new point takes $O(M)$ and the predictive variance takes $O(M^2)$ (Rasmussen and Williams, 2005: p180).

The problem of choosing which point to include in the active set $I$ arises again. In (Csató and Opper, 2002) they compute the *novelty* of a new point to decide whether to include it or not. (Seeger et al., 2003) suggest a greedy alternative to the subset selection problem using the information gain criterion, as well as a cheap way of computing it that allows to run the greedy algorithm on all points in $R$ in each iteration.

### 4.2.5 Bayesian Committee Machine (BCM)

The Bayesian committee machine (BCM) as a way of speeding up Gaussian process regression was introduced in (Tresp, 2000). Let $\mathbf{f}_*$ be the vector of function values at the test locations. For this method we split the dataset into $p$ parts $\mathcal{D}_1, \ldots, \mathcal{D}_p$ where $\mathcal{D}_i = (\mathbf{X}_i, \mathbf{y}_i)$ and make the approximation $p(\mathbf{y}_1, \ldots, \mathbf{y}_p|\mathbf{f}_*, \mathbf{X}) \simeq \prod_{i=1}^{p} p(\mathbf{y}_i|\mathbf{f}_*, \mathbf{X}_i)$ (Rasmussen and Williams, 2005: p180). This implies that

$$q(\mathbf{f}_*|\mathcal{D}_1, \ldots \mathcal{D}_p) \propto p(\mathbf{f}_*) \prod_{i=1}^{p} p(\mathbf{y}_i|\mathbf{f}_*, \mathbf{X}_i) = c\frac{\prod_{i=1}^{p} p(\mathbf{f}_*|\mathcal{D}_i)}{p^{p-1}(\mathbf{f}_*)} \tag{4.10}$$

where $c$ is a normalization constant.

There are several ways of partitioning the dataset $\mathcal{D}$. (Tresp, 2000) assigned randomly the points to the partitions, but (Schwaighofer and Tresp, 2002) recommended clustering the data to improve performance.

Schwaighofer and Tresp also recommended making test predictions on blocks of size $M$. In this case, the computational complexity of BCM is $O(pM^3) = O(M^2N)$ for predicting $M$ test points, or $O(MN)$ per test point.

### 4.2.6 Iterative Solution of Linear Systems

In the $GP$ regression problem we need either to invert the matrix $\mathbf{K} + \sigma_n^2\mathbf{I}$ or to solve the linear system $(\mathbf{K} + \sigma_n^2\mathbf{I})\mathbf{v} = \mathbf{y}$.

This linear system can be solved using an iterative method, such as conjugate gradients (GC) (Golub and Van Loan, 2012: sec. 10.2). In $N$ iterations GC will give the exact solution, but in $k < N$ iterations it will give an approximate solution with time complexity $O(kN^2)$.

## 4.3 The Generalized FITC Approximation

As we have seen in previous sections, a usual scheme in order to accelerate training and prediction times is using a sparse approximation where we use an auxiliary set of size $M \ll N$, also known as active set, inducing points or pseudo-inputs, denoted by $\overline{\mathbf{X}} = (\overline{\mathbf{x}}_1, \overline{\mathbf{x}}_2, \ldots, \overline{\mathbf{x}}_M)$. (Quiñonero-Candela and Rasmussen, 2005) demonstrated that many of these schemes are related through different approximations to the joint prior over training and test points.

The "Fully Independent Training Conditional" or FITC approximation appeared originally in (Snelson and Ghahramani, 2006) as the sparse pseudo-input GP (SPGP).

We define a vector $\mathbf{f}$ that contains the latent values associated with the observed points for each class and a vector $\mathbf{u}$ with the latent values associated with the inducing points for each class. Namely:

$$\begin{aligned}\mathbf{f} &= \left(f^1(\mathbf{x}_1),\ldots,f^1(\mathbf{x}_N),f^2(\mathbf{x}_1),\ldots,f^2(\mathbf{x}_N),\ldots,f^C(\mathbf{x}_1),\ldots,f^C(\mathbf{x}_N)\right)\\ &= (f_1^1,\ldots,f_N^1,f_1^2,\ldots,f_N^2,\ldots,f_1^C,\ldots,f_N^C)\end{aligned} \tag{4.11}$$

$$\begin{aligned}\mathbf{u} &= \left(f^1(\overline{\mathbf{x}}_1),\ldots,f^1(\overline{\mathbf{x}}_M),f^2(\overline{\mathbf{x}}_1),\ldots,f^2(\overline{\mathbf{x}}_M),\ldots,f^C(\overline{\mathbf{x}}_1),\ldots,f^C(\overline{\mathbf{x}}_M)\right)\\ &= (u_1^1,\ldots,u_M^1,u_1^2,\ldots,u_M^2,\ldots,u_1^C,\ldots,u_M^C)\,.\end{aligned} \tag{4.12}$$

The relationship between $\mathbf{f}$ and $\mathbf{u}$ is:

$$p(\mathbf{f}) = \int p(\mathbf{f}|\mathbf{u})p(\mathbf{u}|\overline{\mathbf{X}})d\mathbf{u}\,. \tag{4.13}$$

Let place a Gaussian prior on the latent values associated with the inducing inputs $p(\mathbf{u}|\overline{\mathbf{X}}) \sim \mathcal{N}(\mathbf{u}|\mathbf{0},\mathbf{K}_{MM})$, where $\mathbf{K}_{NM}$ is the covariance matrix between the training and the inducing points and $\mathbf{K}_{MM}$ is the covariance matrix of the inducing points. If we set

$$p(\mathbf{f}|\mathbf{u}) \approx \prod_{i=1}^{n} p(f_i|\mathbf{u}) = \mathcal{N}(\mathbf{f}|\mathbf{K}_{NM}\mathbf{K}_{MM}^{-1}\mathbf{u},\mathrm{diag}(\mathbf{K}_{NN}-\mathbf{Q}_{NN}))\,, \tag{4.14}$$

where $\mathbf{Q}_{NN} = \mathbf{K}_{NM}\mathbf{K}_{MM}^{-1}\mathbf{K}_{MN}$, we obtain an equivalent prior of the following form (Quiñonero-Candela and Rasmussen, 2005):

$$p(\mathbf{f}) \approx \mathcal{N}(\mathbf{f}|\mathbf{0},\mathbf{Q}_{NN}-\mathrm{diag}(\mathbf{K}_{NN}-\mathbf{Q}_{NN}))\,, \tag{4.15}$$

where $\mathbf{Q}_{NN}-\mathrm{diag}(\mathbf{K}_{NN}-\mathbf{Q}_{NN})$ is an approximation to the covariance matrix $\mathbf{K}$. The marginal likelihood after observing $y = f(\mathbf{x}) + \epsilon$ is

$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f})d\mathbf{u} \approx \mathcal{N}(\mathbf{y}|\mathbf{0},\mathbf{Q}_{NN}-\mathrm{diag}(\mathbf{K}_{NN}-\mathbf{Q}_{NN})+\mathbf{I}\sigma^2)\,, \tag{4.16}$$

where $\sigma^2$ is the variance of the noise. The computation of the marginal likelihood is exact in the case of regression, but it will need to use approximate inference (see previous chapter) for classification problems. By maximizing the log marginal likelihood we can then optimize the location of the pseudo-inputs and the hyperparameters of the model. The gradient of $\log p(\mathbf{y})$ with respect to a hyperparameter will be similar to the one in (2.13), but using the approximate covariance matrix. This gradient is not expressed as a sum over the data instances, so it will not allow for stochastic optimization.

A main advantage of the FITC approximation versus the other approximations explained in this chapter is that the inducing points are not necessarily a subset of the training points and their location is optimized automatically during the training process by maximizing the marginal likelihood in eq (4.16).

With this approximation, the training time complexity reduces to $O(NM^2)$.

# Chapter 5

# Scalable GP Multi-class Classification via EP

In this chapter, we briefly introduce the proposed method. After that, we show how to use the expectation propagation (EP) algorithm to train this model and how the model hyperparameters can be optimized by using a stochastic approximation of the gradient of the estimate of the marginal likelihood. For full details on the proposed EP method see Appendix B.

## 5.1 Model specification

We want to solve multi-class classification problems. For that, we follow the approach described in Section 2.3 and we define the vector $\mathbf{f}$ containing one latent function for each instance and each class (see eq. (2.16)). We choose as the likelihood function the Heaviside function, like in eq. (2.20).

In order to speed up the training process we are going to use the FITC approximation. As we have seen in the previous chapter, the FITC approximation introduces $M \ll N$ inducing points and makes the assumption described in eq. (4.14), that allows for approximate inference with a cost that depends linearly on $N$. These inducing points will be different for each class and we will have a vector of latent values $\mathbf{u}^k$ associated with the inducing points $\overline{\mathbf{X}}_k$ for each class $k$. Namely:

$$\mathbf{u}^k = (f(\overline{\mathbf{x}}_1^k), f(\overline{\mathbf{x}}_2^k), \ldots, f(\overline{\mathbf{x}}_M^k)) \,, \tag{5.1}$$

where $\overline{\mathbf{x}}_M^k$ is the vector of features for the $M$-th inducing point and for class $k$. By integrating out the latent values $\mathbf{f}$ and applying Baye's rule, we can obtain the posterior on $\mathbf{u}$:

$$p(\mathbf{u}|\mathbf{y}) = \int p(\mathbf{f}, \mathbf{u}|\mathbf{y}) d\mathbf{f} = \frac{\int p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}|\mathbf{u}) p(\mathbf{u}|\overline{\mathcal{X}}) d\mathbf{f}}{p(\mathbf{y}|\overline{\mathcal{X}})} \,, \tag{5.2}$$

where $\mathbf{u} = (\mathbf{u}^1, \ldots, \mathbf{u}^C)$, $\overline{\mathcal{X}} = (\overline{\mathbf{X}}_1, \ldots, \overline{\mathbf{X}}_C)$, $p(\mathbf{u}|\overline{\mathcal{X}}) = \prod_{k=1}^C p(\mathbf{u}^k|\overline{\mathbf{X}}_k)$, $C$ is the number of classes and each $p(\mathbf{u}^k|\overline{\mathbf{X}}_k) \sim \mathcal{N}(\mathbf{u}^k|\mathbf{0}, \mathbf{K}_{MM}^k)$ is the prior corresponding to the $k$-th latent function on the inducing points.

One of the objectives of this method is the use of stochastic gradients, that will allow to significantly accelerate the training process. For that reason, we would like the estimate of the marginal likelihood to be a sum over

data instances.

For that, we use that $p(\mathbf{y}|\mathbf{f})$ already factorizes over the instances, we take advantage of the factorization defined in eq. (4.14) and we do not marginalize $\mathbf{u}^k\,, k = 1, \ldots, C$ unlike in the original FITC formulation. Hence:

$$\begin{aligned} p(\mathbf{u}|\mathbf{y}) &\approx \frac{\prod_{i=1}^{N} \int p(y_i|\mathbf{f}_i) p(\mathbf{f}_i|\mathbf{u}) d\mathbf{f}_i \; p(\mathbf{u}|\overline{\mathcal{X}})}{p(\mathbf{y}|\overline{\mathcal{X}})} \\ &= \frac{\prod_{i=1}^{N} \int p(y_i|\mathbf{f}_i) \prod_{k=1}^{C} p(f_i^k|\mathbf{u}^k) d\mathbf{f}_i \; p(\mathbf{u}|\overline{\mathcal{X}})}{p(\mathbf{y}|\overline{\mathcal{X}})}\,, \end{aligned} \tag{5.3}$$

where $\mathbf{f}_i = (f_i^1, \ldots, f_i^C)^T$. We use the likelihood in eq. (2.20). So:

$$p(\mathbf{u}|\mathbf{y}) = \frac{\prod_{i=1}^{N} \int \left[\prod_{c \neq y_i} \Theta(f_i^{y_i} - f_i^c)\right] \prod_{k=1}^{C} p(f_i^k|\mathbf{u}^k) d\mathbf{f}_i \; p(\mathbf{u}|\overline{\mathcal{X}})}{p(\mathbf{y}|\overline{\mathcal{X}})}\,. \tag{5.4}$$

The problem with this expression is that each factor corresponding to a data instance involves solving an intractable integral. We can also note that the integral is equivalent to the probability that the latent function corresponding to the class of the instance is the one with the higher value:

$$\begin{aligned} &\int \left[\prod_{c \neq y_i} \Theta(f_i^{y_i} - f_i^c)\right] \prod_{k=1}^{C} p(f_i^k|\mathbf{u}^k) d\mathbf{f}_i = \\ &p(f_i^{y_i} > f_i^1, \ldots, f_i^{y_i} > f_i^{y_i-1}, f_i^{y_i} > f_i^{y_i+1} \ldots, f_i^{y_i} > f_i^C) = \\ &p(f_i^{y_i} > f_i^1 | f_i^{y_i} > f_i^2, \ldots, f_i^{y_i} > f_i^{y_i-1}, f_i^{y_i} > f_i^{y_i+1} \ldots, f_i^{y_i} > f_i^C) \\ &p(f_i^{y_i} > f_i^2 | f_i^{y_i} > f_i^3, \ldots, f_i^{y_i} > f_i^{y_i-1}, f_i^{y_i} > f_i^{y_i+1}) \cdots p(f_i^{y_i} > f_i^C) \end{aligned}\,, \tag{5.5}$$

where we have omitted the condition on $\mathbf{u}^k$ for better readability. Then, we can approximate the last expression by:

$$\begin{aligned} &\int \left[\prod_{c \neq y_i} \Theta(f_i^{y_i} - f_i^c)\right] \prod_{k=1}^{C} p(f_i^k|\mathbf{u}^k) d\mathbf{f}_i \simeq \\ &\qquad p(f_i^{y_i} > f_i^1) \cdots p(f_i^{y_i} > f_i^{y_i-1}) p(f_i^{y_i} > f_i^{y_i+1}) \cdots p(f_i^{y_i} > f_i^C)\,, \end{aligned} \tag{5.6}$$

where we have also omitted the condition on $\mathbf{u}^k$.
Each of the factors $p(f_i^{y_i} > f_i^c)$ will have the following form:

$$p(f_i^{y_i} > f_i^c) = p(f_i^{y_i} - f_i^c > 0) = \Phi\left(\frac{\hat{a}_i^{y_i} - \hat{a}_i^c}{\sqrt{\hat{b}_i^{y_i} + \hat{b}_i^c}}\right)\,, \tag{5.7}$$

where $\Phi(\cdot)$ is the cumulative distribution function of the standard Gaussian distribution, $\hat{a}_i^{y_i}$ and $\hat{a}_i^c$ are the expressions for the means of $f_i^{y_i}$ and $f_i^c$ respectively and $\hat{b}_i^{y_i}$ and $\hat{b}_i^c$ are the expressions for their variances. Namely:

$$\hat{a}_i^{y_i} = \mathbf{K}_{i,M}^{y_i}(\mathbf{K}_{MM}^{y_i})^{-1}\mathbf{u}^{y_i} \tag{5.8}$$

$$\hat{a}_i^{c} = \mathbf{K}_{i,M}^{c}(\mathbf{K}_{MM}^{c})^{-1}\mathbf{u}^{c} \tag{5.9}$$

$$\hat{b}_i^{y_i} = \mathbf{K}_{i,i}^{y_i} - \mathbf{K}_{i,M}^{y_i}(\mathbf{K}_{MM}^{y_i})^{-1}\mathbf{K}_{M,i}^{y_i} \tag{5.10}$$

$$\hat{b}_i^{c} = \mathbf{K}_{i,i}^{c} - \mathbf{K}_{i,M}^{c}(\mathbf{K}_{MM}^{c})^{-1}\mathbf{K}_{M,i}^{c}\,, \tag{5.11}$$

where $\mathbf{K}_{i,M}^{c}$ is a row vector with the covariances between the $i$-th input and the inducing points of the covariance matrix corresponding to the class $c$ and $\mathbf{K}_{i,i}^{c}$ is the prior variance of $f_i^c$. For this derivations we have used eq. (2.5). So the approximate posterior distribution on $\mathbf{u}$ will be:

$$p(\mathbf{u}|\mathbf{y}) \simeq \frac{\left[\prod_{i=1}^{N}\prod_{c!=y_i}\Phi\left(\frac{\hat{a}_i^{y_i}-\hat{a}_i^{c}}{\sqrt{\hat{b}_i^{y_i}+\hat{b}_i^{c}}}\right)\right]\prod_{k=1}^{C}p(\mathbf{u}^k|\overline{\mathbf{X}}_k)}{p(\mathbf{y}|\overline{\mathcal{X}})}\,. \tag{5.12}$$

### 5.1.1 Robust Likelihood

We will also consider the likelihood defined in eq. (2.22), which is robust to possible wrong labeled examples. In this case we compute the corresponding one-dimensional integrals by quadrature, so its running time may be higher.

## 5.2 Approximate Inference

The proposed method is based on the expectation propagation algorithm (see Section 3.1.3), which will obtain a Gaussian posterior approximation $q$ that factorizes over the examples.

The posterior defined in eq. (5.12) is not tractable because of the non Gaussian factors $\phi_i^c(\mathbf{u}^{y_i},\mathbf{u}^c) = \Phi(\hat{a}_i^{y_i} - \hat{a}_i^c/\sqrt{\hat{b}_i^{y_i}+\hat{b}_i^c})$. Each of these factors will be approximated by

$$\begin{aligned}\phi_i^c(\mathbf{u}^{y_i},\mathbf{u}^c) \simeq \tilde{\phi}_i^c(\mathbf{u}^{y_i},\mathbf{u}^c) = \tilde{s}_{i,c}\,\tilde{\mathcal{N}}(\mathbf{u}^{y_i}|\tilde{\mathbf{m}}_{y_i},\tilde{\mathbf{V}}_{y_i})\\ \tilde{\mathcal{N}}(\mathbf{u}^c|\tilde{\mathbf{m}}_c,\tilde{\mathbf{V}}_c)\,,\end{aligned} \tag{5.13}$$

where $\tilde{\mathcal{N}}(\mathbf{u}^{y_i}|\tilde{\mathbf{m}}_{y_i},\tilde{\mathbf{V}}_{y_i})$ and $\tilde{\mathcal{N}}(\mathbf{u}^c|\tilde{\mathbf{m}}_c,\tilde{\mathbf{V}}_c)$ are unnormalized Gaussian distributions defined by

$$\tilde{\mathcal{N}}(\mathbf{u}^{y_i}|\tilde{\mathbf{m}}_{y_i},\tilde{\mathbf{V}}_{y_i}) = \exp\left\{-\frac{1}{2}(\mathbf{u}^{y_i})^T\tilde{\mathbf{V}}_{y_i}\mathbf{u}^{y_i} + (\mathbf{u}^{y_i})^T\tilde{\mathbf{m}}_{y_i}\right\} \tag{5.14}$$

$$\tilde{\mathcal{N}}(\mathbf{u}^{c}|\tilde{\mathbf{m}}_{c},\tilde{\mathbf{V}}_{c}) = \exp\left\{-\frac{1}{2}(\mathbf{u}^{c})^T\tilde{\mathbf{V}}_{c}\mathbf{u}^{c} + (\mathbf{u}^{c})^T\tilde{\mathbf{m}}_{c}\right\}\,, \tag{5.15}$$

and $\tilde{s}_{i,c}$ is a constant.

The posterior approximation $q$ is obtained by replacing each factor $\phi_i^c$ by its approximation $\tilde{\phi}_i^c$.

$$q(\mathbf{u}) = \frac{1}{Z_q} \prod_{i=1}^{N} \left[ \prod_{c \neq y_i} \tilde{\phi}_i^c(\mathbf{u}^{y_i}, \mathbf{u}^c) \right] \prod_{k=1}^{C} p(\mathbf{u}^k | \overline{\mathbf{X}}_k) \,, \tag{5.16}$$

where $Z_q$ is an approximation to the marginal likelihood $p(\mathbf{y}|\overline{\mathbf{X}})$. All factors in $q$ are Gaussian, so $q$ will be a product of $C$ multivariate Gaussians in $M$ dimensions. The log of the approximation to the marginal likelihood $p(\mathbf{y}|\overline{\mathbf{X}})$ is:

$$\log Z_q = g(\boldsymbol{\theta}) - g(\boldsymbol{\theta}_{prior}) + \sum_{i=1}^{N} \sum_{c \neq y_i} \log \tilde{s}_{i,c} \tag{5.17}$$

$$\log \tilde{s}_{i,c} = \log Z_{i,c} + g(\boldsymbol{\theta}^{\setminus i,c}) - g(\boldsymbol{\theta}) \,, \tag{5.18}$$

where $\boldsymbol{\theta}$, $\boldsymbol{\theta}^{\setminus i,c}$ and $\boldsymbol{\theta}_{prior}$ are the natural parameters of $q$, $q^{\setminus i,c} \propto q/\tilde{\phi}_i^c$ and $p(\mathbf{u}|\overline{\mathcal{X}})$ respectively and $g(\boldsymbol{\theta}')$ is the log-normalizer of a multivariate Gaussian with natural parameters $\boldsymbol{\theta}'$.

Expectation propagation updates the hyperparameters at convergence, where the gradient of $\log Z_q$ with respect to the parameters of each approximate factor is zero (Seeger, 2005). In particular, the gradient of $\log Z_q$ with respect to a hyperparameter $\xi_j$ is (Seeger, 2005):

$$\frac{\partial \log Z_q}{\partial \xi_j} = \boldsymbol{\eta}^T \frac{\partial \boldsymbol{\theta}_{prior}}{\partial \xi_j} - \boldsymbol{\eta}_{prior}^T \frac{\partial \boldsymbol{\theta}_{prior}}{\partial \xi_j} + \sum_{i=1}^{N} \sum_{c \neq y_i} \frac{\partial \log Z_{i,c}}{\partial \xi_j} \,, \tag{5.19}$$

where $\boldsymbol{\eta}$ and $\boldsymbol{\eta}_{prior}$ are the expected sufficient statistics (see eq. (3.16)) under $q$ and $p(\mathbf{u}|\overline{\mathcal{X}})$ respectively. We can use these gradients to update the hyperparameters by maximizing $\log Z_q$. At this point, the cost of EP is $O(NM^2)$, because of several simplifications when computing the derivatives with respect to the inducing points (Snelson, 2007).

### 5.2.1 Robust likelihood

When using the robust likelihood in eq. (2.22) the exact factors are:

$$\phi_i = \left( \left[ \prod_{c \neq y_i} \Theta(f_i^{y_i} - f_i^c) \right] (1 - \epsilon) + \left[ \frac{1}{C} \right] \epsilon \right) \prod_{k=1}^{C} p(f_i^k | \mathbf{u}^k) df_i^k \,, \tag{5.20}$$

which will be approximated by

$$\phi_i = \prod_{k=1}^{C} \tilde{\mathcal{N}}(\mathbf{u}^k | \tilde{\mathbf{m}}_k, \tilde{\mathbf{V}}_k) \,, \tag{5.21}$$

where $\tilde{\mathcal{N}}(\mathbf{u}^k|\tilde{\mathbf{m}}_k, \tilde{\mathbf{V}}_k)$ is an unnormalized Gaussian distribution with mean vector $\tilde{\mathbf{m}}_k$ and covariance matrix $\tilde{\mathbf{V}}_k$, corresponding to the $k$-th class. The

integral in eq. (5.20) can be evaluated using one-dimensional quadrature techniques.

### 5.2.2 Scalable Expectation Propagation

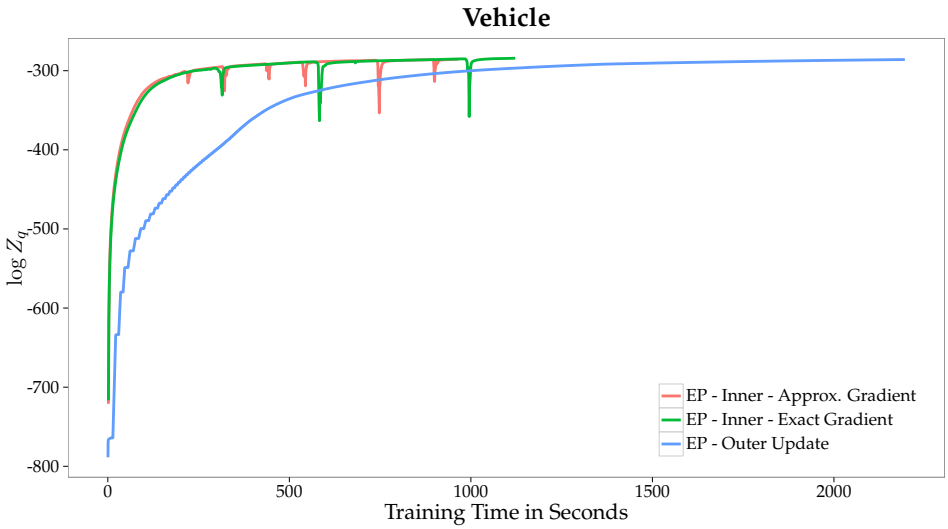



**Figure 5.1:** Value of $\log Z_q$ obtained when the updates of the hyperparameters are performed after EP has converged (outer) and after each update of the approximate factors $\phi_i^c$ (inner), with the exact gradient, and the approximation.

Expectation propagation needs to wait until convergence in order to update the hyperparameters, which is when eq. (5.19) is valid. This will cause the algorithm to be very inefficient at the beginning, since EP will take several iterations to converge. We follow (Hernández-Lobato and Hernández-Lobato, 2015) and update the approximate factors $\tilde{\phi}_i^c$ and the hyperparameters at the same time. This implies adding some extra terms to the gradient in eq. (5.19) because EP has not converged and the moments of $Z_{i,c}^{-1}\phi_i^c q^{\setminus i,c}$ and $q$ do not match, but in practice the extra terms are very small and can be ignored (Hernández-Lobato and Hernández-Lobato, 2015). Figure 5.1 shows the evolution of $\log Z_q$ on the Vehicle dataset of the UCI repository (Lichman, 2013) when updating the hyperparameters after EP has converged and when updating the hyperparameters and the approximate factors in parallel. It can be seen that doing the inner updates is significantly faster because it does not need to wait until convergence, giving similar results when the approximate gradient is used instead of the exact one.

#### Training using minibatches

Stochastic optimization can be used with the described method. For that, the training data is split in minibatches $\mathcal{M}_k$ of size at most the number of inducing points $M$. For each minibatch $\mathcal{M}_k$, each factor $\tilde{\phi}_i^c$ is refined and $q$ is updated afterwards. The gradient used in this case to update the hyperparameters is a stochastic approximation of (5.19):

$$\frac{\partial \log Z_q}{\partial \xi_j} \approx \boldsymbol{\eta}^T \frac{\partial \boldsymbol{\theta}_{prior}}{\partial \xi_j} - \boldsymbol{\eta}_{prior}^T \frac{\partial \boldsymbol{\theta}_{prior}}{\partial \xi_j} + \frac{N}{|\mathcal{M}_k|} \sum_{l \in \mathcal{M}_k} \sum_{c \neq y_i} \frac{\partial \log Z_{l,c}}{\partial \xi_j} . \quad (5.22)$$

This training scheme updates more often the hyperparameters, since there is no need to go through all the data. When using minibatches, the training cost scales like $O(M^3)$.

### Stochastic Expectation Propagation

An important limitation of EP is that it needs to keep in memory the individual approximate factors $\tilde{\phi}_i^c$ to be able to compute the cavity distribution $q^{\setminus i,c} \propto q/\tilde{\phi}_i^c$. The number of approximate factors increases with the number of data points $N$, so when $N$ is big the memory needed can be prohibitively large. In order to improve the memory usage, what can be done is to only keep the product of the approximate factors $\tilde{\Phi} = \prod \tilde{\phi}_i^c$ and assume that all the approximate factors are the same so that when EP computes the cavity distribution it approximates them by obtaining the corresponding fraction of $\tilde{\Phi}$, i.e. $\tilde{\phi}_i^c \simeq \tilde{\Phi}^{\frac{1}{\tilde{N}}}$, where $\tilde{N}$ is the number of factors. This approach, called stochastic expectation propagation, was first introduced in (Li et al., 2015) and it can reduce the memory consumption by a factor of $N$ performing almost as well as full EP. Figure 5.2 illustrates the differences between this normal EP and the stochastic EP (SEP) when approximating a target distribution $p(\boldsymbol{\theta})$.

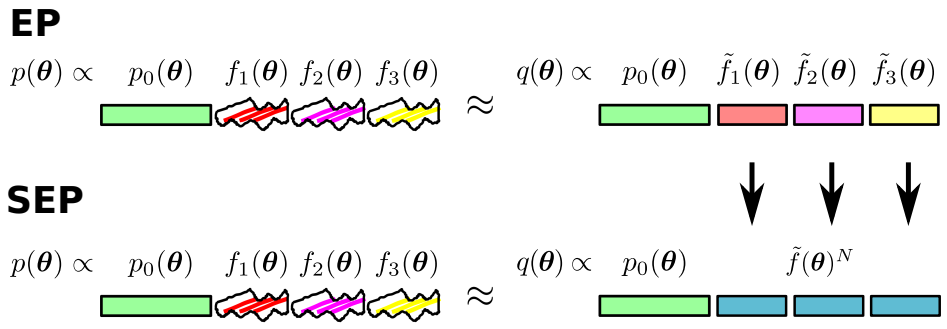



**Figure 5.2:** Differences between full EP and stochastic EP. In the stochastic version instead of saving the parameters of each approximate factor we consider that all the factor are the same and we save only the parameters of one factor. Source of the figure [1].

## 5.3 Related methods

The proposed method is based on the previous work in (Hernández-Lobato and Hernández-Lobato, 2015), which is a method for binary classification using Gaussian processes with EP for large datasets. The proposed method is a generalization that can work with multi-class problems.

[1] https://jmhldotorg.files.wordpress.com/2015/12/poster_sep_nips2015.pdf

A related method uses a scalable variational approach instead of expectation propagation to solve multi-class classification problems with Gaussian processes (Hensman et al., 2015b). For that, it maximizes a lower bound on the log marginal likelihood of the form:

$$\log p(\mathbf{y}) \geq \sum_{i=1}^{N} \mathbb{E}_{q(f_i)}[\log p(y_i|f_i)] - \mathrm{KL}[q(\mathbf{u}) \parallel p(\mathbf{u})] = \mathcal{L}(q) \geq 0\,, \qquad (5.23)$$

which is obtained by using Jensen's inequality. This method uses the robust likelihood in (Hernández-Lobato, 2010) and quadrature to approximate the required expectations.

The other related method is the one in (Kim and Ghahramani, 2006), which uses the same likelihood as the proposed approach. This method can be combined with the original FITC approximation (GFITC) (Naish-Guzman and Holden, 2007) generalized to address classification problems. However, unlike the proposed approach, the original FITC approximation marginalizes $\mathbf{u}$ (the values associated to the inducing points) and the marginal likelihood approximation cannot be expressed as a sum across the data instances. This prevents using stochastic gradients.

# Chapter 6

# Experiments

We are going to compare the proposed method for Multi-class Gaussian process classification based on scalable Expectation Propagation (EP) with (i) the original generalized FITC approximation (GFITC) applied on the model of (Kim and Ghahramani, 2006), (ii) a version of the method that updates the hyperparameters once the EP algorithm has converged (EP outer), (iii) the proposed method that uses the robust likelihood proposed in (Hernández-Lobato et al., 2011) (REP), (iv) the same method that uses expectation propagation (SEP) to save memory (Li et al., 2015) and (v) the scalable variational inference (VI) method from (Hensman et al., 2015b). In the robust methods a value of $\epsilon = 0.001$ has been used. This value is the one employed in (Hensman et al., 2015b). All methods are implemented in R (R Core Team, 2016).

## 6.1 Performance on datasets from the UCI repository

The first experiment evaluates the performance (test error and negative test log likelihood) of all the methods on 8 datasets from the UCI repository (Lichman, 2013). The training process is done using batch optimization methods (we go through all the data to compute the gradients). We use $90\%$ of the data from training and the remaining $10\%$ for testing, expect for the Satellite dataset, where due to the big size of the dataset we have chosen $20\%$ for training and $80\%$ for testing. For the Waveform dataset, which is synthetic, we have generated $10^3$ instances and split them in $30\%$ for training and $70\%$ for testing. Finally, for the Vowel dataset we have selected only the points that belongs to the six first classes, in order to reduce the size of the problem (From 990 instances to 540). See Table 6.1 for the details of the datasets used in this experiment.

We report averages over $20$ repetitions of the experiments. Both the training/test splits and the initial location of the inducing points have been chosen at random from the datasets, and they are the same for all the methods, as well as the initial hyperparameters. The experiments have been executed using different values for the number of inducing points $M$, in particular $5\%$, $10\%$ and $20\%$. All methods are trained using batch optimization algorithms for $250$ iterations or until convergence.

In Tables 6.3 and 6.2 we report the average negative test log likelihood, the average test error and the average training time for each method. The values printed in bold are the best results for each dataset. We can see that the proposed methods (EP, REP and SEP) have similar performance to that

**Table 6.1:** Characteristics of the UCI datasets used in the experiments

| Dataset | #Instances | #Attributes | #Classes |
|---|---|---|---|
| glass | 214 | 9 | 6 |
| new-thyroid | 215 | 5 | 3 |
| satellite | 6435 | 36 | 6 |
| svmguide2 | 391 | 20 | 3 |
| vehicle | 846 | 18 | 4 |
| vowel | 540 | 10 | 6 |
| waveform | 1000 | 21 | 3 |
| wine | 178 | 13 | 3 |

of GFITC and VI, and sometimes they obtain the best results. The fastest method is EP because it does not use quadrature to compute the integrals in the likelihood. EP outer is slower than EP because it waits until EP has converged to do an update of the hyperparameters. VI is slightly faster than REP and SEP even though they all use quadrature to approximate intractable integrals in the likelihood. This is because the lower bound of VI and its gradients are easier to compute than the corresponding calculations required in REP and SEP. The negative test log likelihood is in some datasets much higher for VI than for the other methods, due to the fact that VI does not optimize $\log \mathbb{E}[p(\mathbf{y}|\mathbf{f})]$ (the log likelihood) directly, but instead it optimizes $\mathbb{E}[\log p(\mathbf{y}|\mathbf{f})]$, which is a lower bound.

## 6.2 Learning the location of the inducing points

In this experiment we use a synthetic two dimensional dataset with three classes, generated by sampling the latent functions from the $\mathcal{GP}$ prior and applying the rule $y_i = \arg\max_k f_i^k(x)$. We want to analyze the behavior of the different methods (EP, REP, SEP, VI and GFITC) with several values for the number of inducing points (from $M = 1$ to $M = 256$). The initial location of the inducing points is selected at random and it is the same for all the methods. As the focus of the experiment is set on the location of the pseudo-inputs after training, the hyperparameters are fixed to their true value and they are not updated during the training process. All methods but VI are trained using batch methods during $2000$ iterations. VI is trained using stochastic gradients for $2000$ epochs because the batch version often gets stuck in local optima otherwise.

The obtained results are shown in Figure 6.1. Each column shows a different number of inducing points. Blue, red and green points represent the training data, black lines are decision boundaries and black border points are pseudo-inputs. We can see that as we increase the number of inducing points GFITC tends to overlap them, which can be understood as a pruning mechanism (Bauer et al., 2016). This behavior is explained in detail in (Bauer et al., 2016). The proposed methods EP and REP give similar results. SEP does not overlap the resulting inducing points as $M$ grows. VI seems to give poorer results with a low value of $M$. This is due to a worst

| | Problem | GFITC | EP | EP outer | REP | SEP | VI |
|---|---|---|---|---|---|---|---|
| M = 5% | glass | **0.24 ± 0.09** | 0.31 ± 0.09 | 0.31 ± 0.08 | 0.33 ± 0.1 | 0.32 ± 0.08 | 0.35 ± 0.1 |
| | new-thyroid | **0.02 ± 0.02** | 0.04 ± 0.06 | 0.02 ± 0.02 | 0.06 ± 0.17 | 0.03 ± 0.03 | 0.03 ± 0.03 |
| | satellite | 0.12 ± 0 | **0.11 ± 0** | 0.11 ± 0 | 0.11 ± 0 | 0.12 ± 0 | 0.12 ± 0 |
| | svmguide2 | 0.2 ± 0.08 | 0.2 ± 0.07 | 0.2 ± 0.07 | 0.2 ± 0.07 | 0.21 ± 0.07 | **0.19 ± 0.05** |
| | vehicle | 0.17 ± 0.05 | 0.17 ± 0.04 | 0.17 ± 0.04 | 0.18 ± 0.05 | 0.18 ± 0.04 | **0.16 ± 0.05** |
| | vowel | **0.05 ± 0.03** | 0.09 ± 0.04 | 0.07 ± 0.03 | 0.09 ± 0.04 | 0.09 ± 0.03 | 0.06 ± 0.03 |
| | waveform | 0.16 ± 0.01 | **0.15 ± 0.01** | 0.15 ± 0.01 | 0.15 ± 0.01 | 0.16 ± 0.01 | 0.17 ± 0.02 |
| | wine | 0.03 ± 0.03 | 0.03 ± 0.04 | 0.03 ± 0.05 | **0.02 ± 0.03** | 0.03 ± 0.03 | 0.04 ± 0.04 |
| | **Avg. Time** | 194.81 ± 337.06 | 71.05 ± 118.6 | 166.99 ± 294.13 | 281.04 ± 298.68 | 270.76 ± 287.06 | 164.7 ± 186.75 |
| M = 10% | glass | **0.21 ± 0.07** | 0.29 ± 0.09 | 0.29 ± 0.08 | 0.31 ± 0.09 | 0.32 ± 0.08 | 0.35 ± 0.08 |
| | new-thyroid | 0.03 ± 0.02 | **0.02 ± 0.02** | 0.03 ± 0.02 | 0.02 ± 0.03 | 0.02 ± 0.03 | 0.03 ± 0.03 |
| | satellite | **0.11 ± 0** | 0.11 ± 0 | 0.11 ± 0 | 0.11 ± 0 | 0.11 ± 0 | 0.12 ± 0 |
| | svmguide2 | 0.19 ± 0.07 | 0.2 ± 0.07 | 0.2 ± 0.06 | 0.2 ± 0.07 | 0.19 ± 0.08 | **0.17 ± 0.06** |
| | vehicle | 0.17 ± 0.04 | 0.16 ± 0.03 | 0.17 ± 0.03 | 0.18 ± 0.04 | 0.16 ± 0.04 | **0.15 ± 0.04** |
| | vowel | **0.03 ± 0.03** | 0.05 ± 0.03 | 0.05 ± 0.02 | 0.06 ± 0.03 | 0.07 ± 0.03 | 0.05 ± 0.02 |
| | waveform | 0.17 ± 0.01 | 0.16 ± 0.01 | **0.15 ± 0.01** | 0.16 ± 0.01 | 0.16 ± 0.01 | 0.18 ± 0.01 |
| | wine | 0.04 ± 0.04 | 0.02 ± 0.03 | 0.02 ± 0.03 | **0.01 ± 0.02** | 0.02 ± 0.03 | 0.03 ± 0.03 |
| | **Avg. Time** | 509.73 ± 963.62 | 160.78 ± 288.09 | 433.75 ± 850.67 | 347.05 ± 430.57 | 329.81 ± 399.77 | 213.73 ± 281.28 |
| M = 20% | glass | **0.2 ± 0.07** | 0.28 ± 0.09 | 0.27 ± 0.09 | 0.31 ± 0.08 | 0.32 ± 0.08 | 0.36 ± 0.08 |
| | new-thyroid | 0.03 ± 0.02 | 0.02 ± 0.03 | 0.03 ± 0.02 | 0.02 ± 0.03 | **0.02 ± 0.02** | 0.03 ± 0.03 |
| | satellite | **0.11 ± 0** | 0.11 ± 0 | 0.11 ± 0 | 0.11 ± 0 | 0.12 ± 0 | 0.11 ± 0.01 |
| | svmguide2 | 0.2 ± 0.07 | 0.19 ± 0.07 | 0.19 ± 0.07 | **0.19 ± 0.06** | 0.2 ± 0.07 | 0.2 ± 0.07 |
| | vehicle | 0.17 ± 0.04 | 0.16 ± 0.04 | 0.17 ± 0.03 | 0.17 ± 0.05 | 0.16 ± 0.04 | **0.15 ± 0.04** |
| | vowel | 0.03 ± 0.03 | 0.03 ± 0.02 | **0.02 ± 0.02** | 0.04 ± 0.02 | 0.05 ± 0.02 | 0.03 ± 0.02 |
| | waveform | 0.17 ± 0.01 | **0.16 ± 0.01** | 0.16 ± 0.01 | 0.16 ± 0.01 | 0.16 ± 0.01 | 0.18 ± 0.01 |
| | wine | 0.04 ± 0.04 | **0.01 ± 0.03** | 0.02 ± 0.03 | 0.02 ± 0.03 | 0.02 ± 0.03 | 0.03 ± 0.04 |
| | **Avg. Time** | 1612.28 ± 3300.87 | 452.09 ± 874.83 | 1510.91 ± 3243.11 | 579.89 ± 908.76 | 546.88 ± 834.9 | 398.39 ± 653.96 |

**Table 6.2:** Average test error for each method and average training time in seconds.

| | Problem | GFITC | EP | EP outer | REP | SEP | VI |
|---|---|---|---|---|---|---|---|
| M = 5% | glass | **0.6 ± 0.22** | 0.78 ± 0.27 | 0.78 ± 0.2 | 0.78 ± 0.16 | 0.78 ± 0.19 | 2.45 ± 0.66 |
| | new-thyroid | **0.06 ± 0.05** | 0.11 ± 0.14 | 0.07 ± 0.03 | 0.24 ± 0.71 | 0.07 ± 0.04 | 0.09 ± 0.07 |
| | satellite | 0.33 ± 0.02 | 0.31 ± 0.01 | **0.3 ± 0.01** | 0.3 ± 0.01 | 0.31 ± 0.01 | 0.62 ± 0.05 |
| | svmguide2 | 0.63 ± 0.25 | 0.63 ± 0.25 | **0.6 ± 0.2** | 0.6 ± 0.2 | 0.62 ± 0.22 | 1.03 ± 0.36 |
| | vehicle | **0.32 ± 0.07** | 0.34 ± 0.07 | 0.33 ± 0.06 | 0.35 ± 0.06 | 0.35 ± 0.07 | 0.76 ± 0.24 |
| | vowel | **0.16 ± 0.06** | 0.25 ± 0.05 | 0.24 ± 0.05 | 0.27 ± 0.05 | 0.27 ± 0.05 | 0.41 ± 0.2 |
| | waveform | 0.42 ± 0.05 | 0.36 ± 0.02 | **0.33 ± 0.01** | 0.36 ± 0.02 | 0.38 ± 0.02 | 0.89 ± 0.07 |
| | wine | 0.08 ± 0.07 | 0.07 ± 0.06 | 0.1 ± 0.05 | **0.07 ± 0.05** | 0.07 ± 0.05 | 0.08 ± 0.1 |
| | **Avg. Time** | 194.81 ± 337.06 | 71.05 ± 118.6 | 166.99 ± 294.13 | 281.04 ± 298.68 | 270.76 ± 287.06 | 164.7 ± 186.75 |
| M = 10% | glass | **0.58 ± 0.22** | 0.74 ± 0.26 | 0.74 ± 0.19 | 0.76 ± 0.15 | 0.77 ± 0.23 | 2.17 ± 0.64 |
| | new-thyroid | 0.07 ± 0.05 | 0.06 ± 0.04 | 0.07 ± 0.03 | 0.07 ± 0.05 | 0.06 ± 0.04 | **0.05 ± 0.05** |
| | satellite | 0.34 ± 0.02 | 0.3 ± 0.01 | **0.29 ± 0.01** | 0.29 ± 0.01 | 0.32 ± 0.01 | 0.58 ± 0.04 |
| | svmguide2 | 0.67 ± 0.25 | 0.67 ± 0.28 | 0.64 ± 0.24 | **0.61 ± 0.21** | 0.64 ± 0.23 | 0.9 ± 0.44 |
| | vehicle | **0.33 ± 0.07** | 0.33 ± 0.08 | 0.33 ± 0.07 | 0.35 ± 0.06 | 0.35 ± 0.08 | 0.71 ± 0.19 |
| | vowel | **0.14 ± 0.05** | 0.19 ± 0.05 | 0.19 ± 0.04 | 0.21 ± 0.04 | 0.22 ± 0.05 | 0.3 ± 0.2 |
| | waveform | 0.42 ± 0.04 | 0.36 ± 0.03 | **0.34 ± 0.02** | 0.36 ± 0.02 | 0.4 ± 0.03 | 0.85 ± 0.06 |
| | wine | 0.07 ± 0.06 | **0.06 ± 0.05** | 0.09 ± 0.04 | 0.06 ± 0.05 | 0.06 ± 0.05 | 0.07 ± 0.07 |
| | **Avg. Time** | 509.73 ± 963.62 | 160.78 ± 288.09 | 433.75 ± 850.67 | 347.05 ± 430.57 | 329.81 ± 399.77 | 213.73 ± 281.28 |
| M = 20% | glass | **0.6 ± 0.3** | 0.75 ± 0.25 | 0.74 ± 0.19 | 0.74 ± 0.15 | 0.75 ± 0.22 | 2.31 ± 0.68 |
| | new-thyroid | 0.07 ± 0.05 | 0.06 ± 0.05 | 0.07 ± 0.03 | 0.07 ± 0.08 | **0.05 ± 0.04** | 0.06 ± 0.05 |
| | satellite | 0.34 ± 0.02 | 0.3 ± 0.01 | **0.29 ± 0.01** | 0.29 ± 0.01 | 0.33 ± 0.02 | 0.53 ± 0.04 |
| | svmguide2 | 0.67 ± 0.24 | 0.65 ± 0.26 | 0.63 ± 0.24 | **0.62 ± 0.21** | 0.67 ± 0.24 | 0.94 ± 0.38 |
| | vehicle | **0.33 ± 0.07** | 0.33 ± 0.08 | 0.34 ± 0.07 | 0.34 ± 0.06 | 0.34 ± 0.08 | 0.63 ± 0.18 |
| | vowel | **0.12 ± 0.04** | 0.16 ± 0.05 | 0.16 ± 0.04 | 0.18 ± 0.04 | 0.2 ± 0.04 | 0.14 ± 0.13 |
| | waveform | 0.43 ± 0.06 | 0.37 ± 0.03 | **0.35 ± 0.02** | 0.37 ± 0.03 | 0.43 ± 0.03 | 0.8 ± 0.07 |
| | wine | 0.07 ± 0.07 | **0.05 ± 0.05** | 0.09 ± 0.04 | 0.06 ± 0.06 | 0.06 ± 0.05 | 0.06 ± 0.08 |
| | **Avg. Time** | 1612.28 ± 3300.87 | 452.09 ± 874.83 | 1510.91 ± 3243.11 | 579.89 ± 908.76 | 546.88 ± 834.9 | 398.39 ± 653.96 |

**Table 6.3:** Average negative test log likelihood for each method and average training time in seconds.

initial estimation of the approximation to the posterior distribution $q$ because it is initialized to the prior and updated using gradient steps, which requires several iterations to get a good estimate of this distribution. By contrast, EP methods do not use gradient steps to update $q$ and, therefore, do not need a learning rate. They are more efficient at finding a better estimation of $q$ at the beginning of the learning process. VI also places the inducing points near the decision boundaries, which is consistent with the results reported in (Hernández-Lobato and Hernández-Lobato, 2015)(Hensman et al., 2015a) for the binary case.

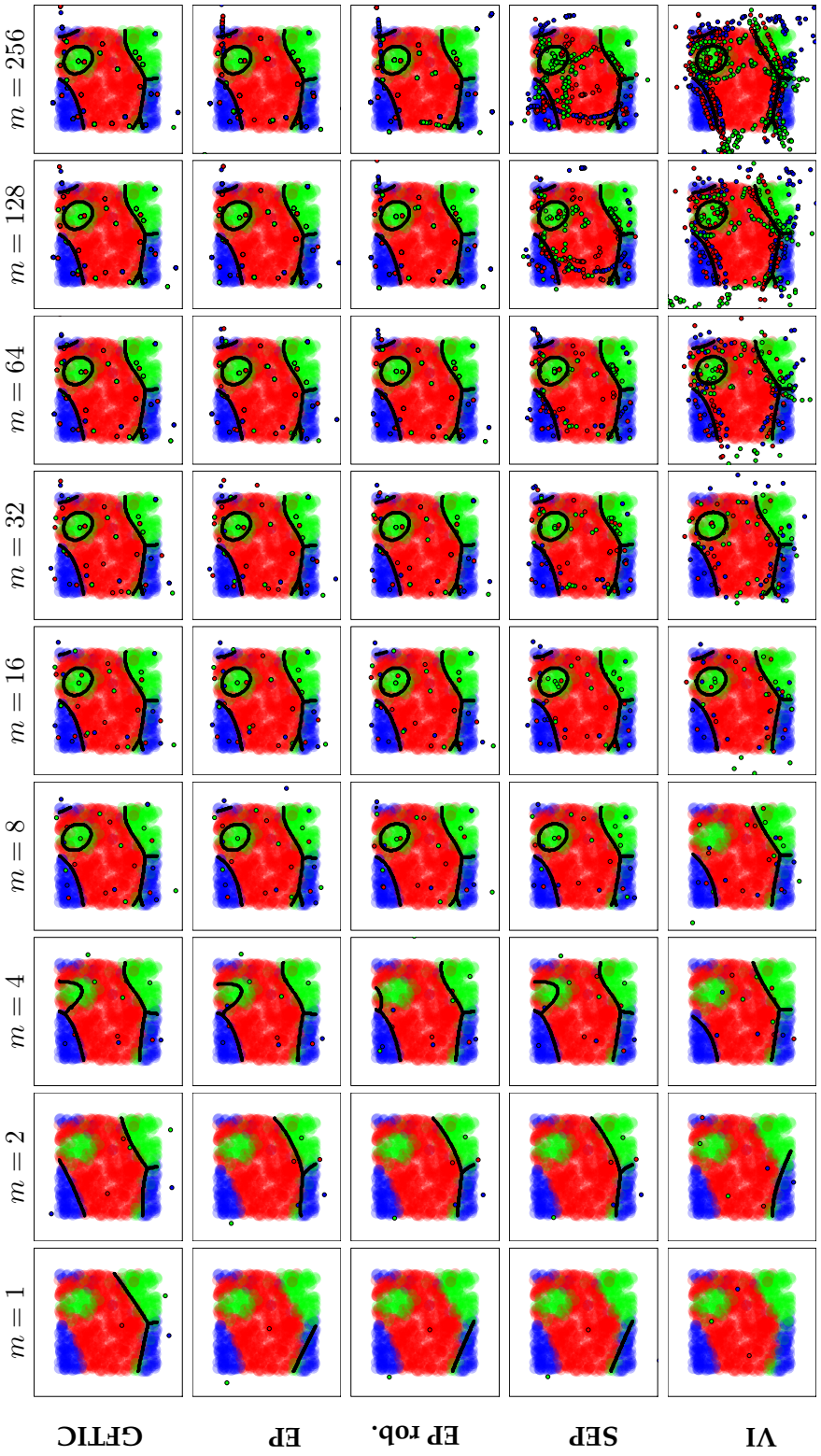



**Figure 6.1:** Effect of increasing the inducing points for EP, REP, SEP, VI and MGFITC (best seen in color).

## 6.3 Performance as a function of time

We measure the performance of the different methods as a function of the training time for the Satellite dataset. For this experiment, only GFITC, EP and VI are considered for better readability. We use again $90\%$ for training and $10\%$ for testing. Each of the methods is tested for a different number of inducing points $M = 4, 50, 200$. The results are averages over $100$ realizations of the experiments. We use batch training.

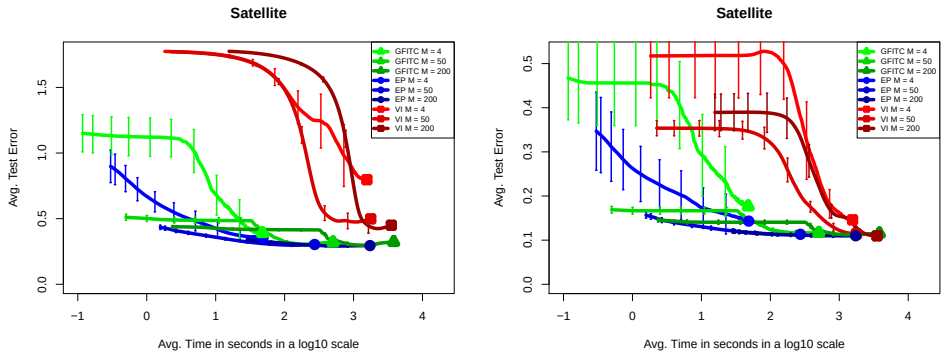



**Figure 6.2:** Prediction performance (negative test log likelihood on the left and test error on the right) of each method on the Satellite dataset as a function of the training time measured in seconds (in a log 10 scale). Different numbers of inducing points are considered, i.e., M = 4, 50, 200.

The results are shown in Figure 6.2. We observe that EP is the method that provides the best performance at the lowest training time. Again, it is faster than GFITC because it optimizes the posterior approximation $q$ and the hyperparameters at the same time, while GFITC waits until EP has converged to update the hyperparameters. VI is not very efficient in comparison, especially for small values of $M$, because it requires the computation of the lower bound by quadrature, which is expensive. It also starts with a worst estimation of the approximation $q$ as explained in Section 6.2, because it updates $q$ by gradient descent, which requires several iterations to get a good estimate of this distribution.

## 6.4 Training using minibatches and stochastic gradients

We evaluate the performance of the proposed methods on the MNIST dataset using minibatches of $200$ data points to update the posterior approximation $q$ and to compute an stochastic approximation of the gradient of the hyperparameters. The learning rate for all the methods is computed using the Adam algorithm with parameters $\alpha = 10^{-3}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$ (Kingma and Ba, 2014). Note that GFITC does not allow for this type of stochastic optimization, so it is not considered for comparison. We use $60,000$ instances for training and $10,000$ for testing. The number of inducing points is set to the same size of the minibatches $M = 200$. In Figure

6.3 (top) we report the performance (negative test log likelihood and test error) as a function of the training time in seconds (in $\log_{10}$ scale). We can see that in this larger dataset all the methods obtain similar results, but the proposed methods (EP, REP and SEP) converge in a lower training time than VI.

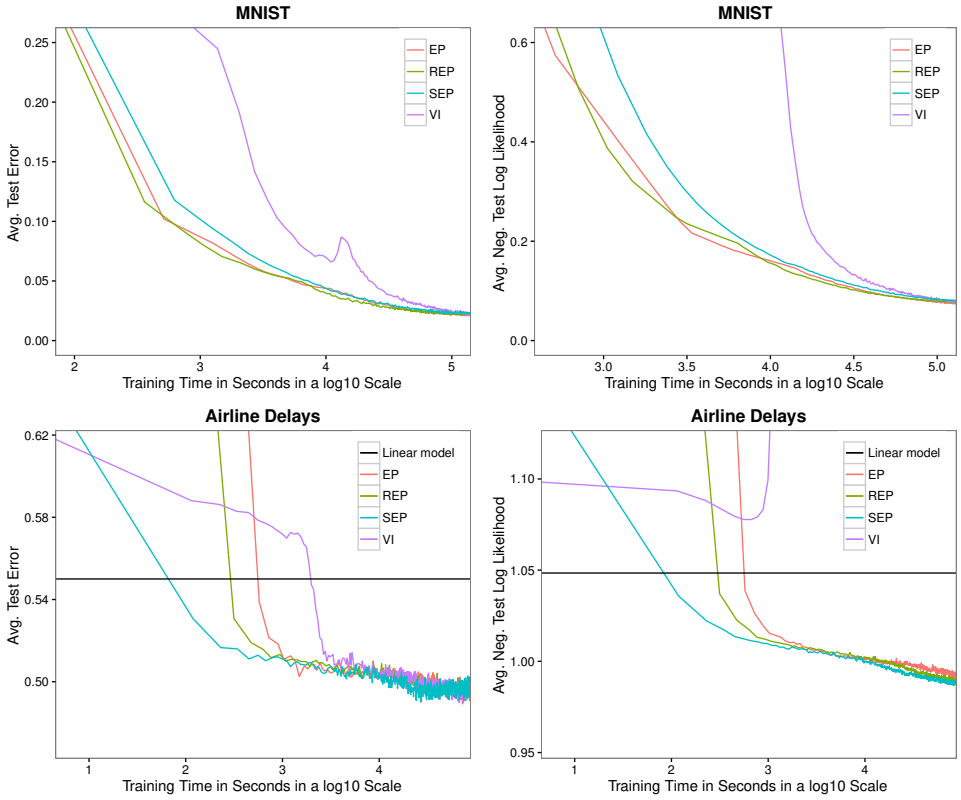



**Figure 6.3:** Performance on the MNIST dataset as a function of time (in $\log_{10}$ scale) using stochastic gradients with a minibatch size of 200. (bottom) Same results for the Airline delays dataset. The performance of a linear classifier is also included for comparison.

The last experiment is carried out on a dataset that considers all commercial flights within the USA between January 2008 and April 2008 (available at http://stat-computing.org/dataexpo/2009). The task is to classify the flights according to their delay time in three classes: on time, more than 5 minutes delay or more than 5 minutes before scheduled time. After removing the instances with missing data we are left with $2,127,068$ instances, from which $10,000$ are used for testing and the rest for training. Like for MNIST we have used Adam algorithm with the same parameters and a minibatch of 200 data points. We also use $M = 200$ inducing points. The results obtained are shown in Figure 6.3 (bottom). It is also shown the performance of a linear classifier for comparison. In this case, VI starts giving a better approximation but the methods based on EP converge faster than VI. Also, the negative log likelihood of VI starts increasing at some point (the range of the axis have been reduced for readability), which is due to the difference between the objective that optimizes VI $\mathbb{E}[\log p(\mathbf{y}|\mathbf{f})]$ (see eq. (5.23)) and the log likelihood $\log \mathbb{E}[p(\mathbf{y}|\mathbf{f})]$. In Figure 6.4 it can be

seen that when the objective of VI is optimized, the log likelihood decreases, giving rise to this behavior.

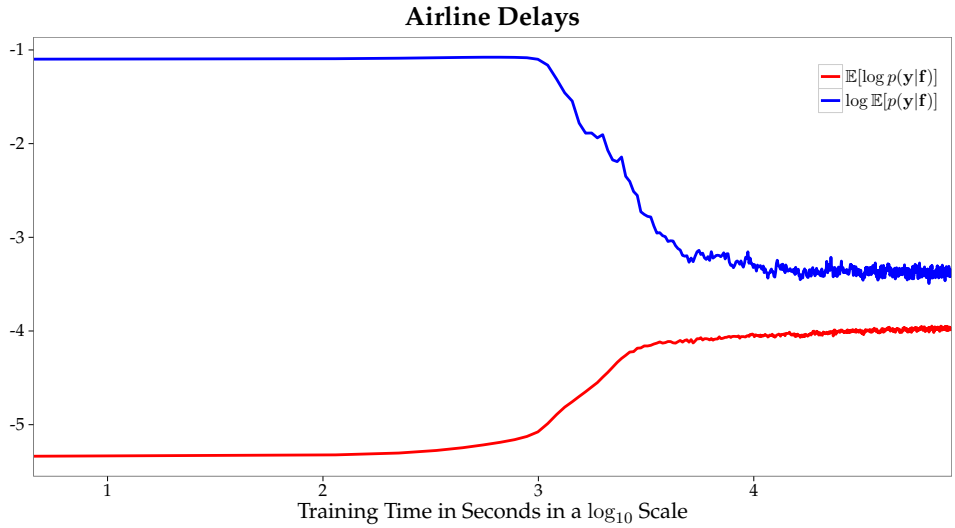



**Figure 6.4:** Comparison between the objective optimized by VI ($\mathbb{E}[\log p(\mathbf{y}|\mathbf{f})]$) and the log likelihood ($\log \mathbb{E}[p(\mathbf{y}|\mathbf{f})]$), evaluated in the test set.

# Chapter 7

# Conclusions and Future Work

We have proposed a method for multi-class classification with Gaussian processes that scales well to very large datasets. For that, we have used the EP algorithm, as well as the FITC approximation by introducing $M \ll N$ inducing points. Our method allows for stochastic optimization, since the estimate of the marginal likelihood is expressed as a sum across the data instances. We have also considered the use of a robust likelihood (REP) (Hernández-Lobato et al., 2011) and a stochastic version of EP to reduce the memory usage (SEP) (Li et al., 2015). The overall training cost for the method is $O(M^3)$.

The proposed methods (EP, REP and SEP) have been compared with other approaches from the literature, such as a scalable variational method (VI) (Hensman et al., 2015b), or the model considered by (Kim and Ghahramani, 2006) combined with the original FITC approximation (GFITC) (Naish-Guzman and Holden, 2007). Our experiments show that EP and REP tend to overlap the inducing points like GFITC. This can be seen as a pruning technique. The proposed methods outperform GFITC in large datasets because the last one does not allow for stochastic gradients. As opposed to EP, the other methods (REP, SEP and VI) use a likelihood that requires the use of quadrature to solve the integrals in the likelihood. For that reason, EP is the faster is small datasets. Our methods are more efficient than VI at finding a better estimation of $q$ at the beginning, because VI uses gradient steps to update $q$ and the EP updates are free of any learning rate. Finally, the performance of the proposed methods is better than VI in terms of the test log likelihood due to the difference between the objective that optimizes VI $\mathbb{E}[\log p(\mathbf{y}|\mathbf{f})]$ (see eq. (5.23)) and the log likelihood $\log \mathbb{E}[p(\mathbf{y}|\mathbf{f})]$.

## 7.1 Future work

A possible improvement to the methods that use the robust likelihood is to optimize the parameter $\epsilon$ during the training process by maximizing the marginal likelihood. It could also be interesting to explore other divergence measures to find the parameters of the approximate factors and $q$ that are different from $\mathrm{KL}[p \parallel q]$ (EP methods) and $\mathrm{KL}[q \parallel p]$ (VI). An example is the minimization of $\alpha$-divergences (Hernández-Lobato et al., 2015). Another potential improvement is to introduce the use of MCMC sampling to approximate the posterior of the model hyperparameters (Hensman et al., 2015b). Finally, we could use other sparse approximations instead of FITC,

i.e. the one in (Lázaro-Gredilla et al., 2010), which places the inducing points in the frequency domain of the covariance function.

# Appendix A

# Gaussian distribution

In this Appendix we are going to describe some interesting properties about the Gaussian distribution. It is based on the Appendix A.3 of (Hernández-Lobato, 2010).

The Gaussian distribution is a continuous probability distribution of a $d$-dimensional vector $\mathbf{x} \in \mathbb{R}^d$. It receives as parameters a $d$-dimensional vector of means $\boldsymbol{\mu}$ and a $d$x$d$ covariance matrix $\boldsymbol{\Sigma}$. The probability density function of $\mathbf{x}$ is usually denoted as $\mathcal{N}(\mathbf{x})$ and is given by

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\} . \tag{A.1}$$

The Gaussian distribution belongs to the exponential family as its probability density function (A.1) can be written as

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \exp\left\{\boldsymbol{\eta}^T \mathbf{u}(\mathbf{x}) - g(\boldsymbol{\eta})\right\} , \tag{A.2}$$

where

$$\boldsymbol{\eta} = (\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}, -\frac{1}{2}\boldsymbol{\Sigma}^{-1}) \tag{A.3}$$

$$\mathbf{u}(\mathbf{x}) = (\mathbf{x}, \mathbf{x}\mathbf{x}^T) , \tag{A.4}$$

and $g(\boldsymbol{\eta}) = \frac{d}{2}\log 2\pi + \frac{1}{2}\log|\boldsymbol{\Sigma}| + \frac{1}{2}\boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$ is the log-normalizer. The natural moments are obtained by computing the expectation of $\mathbf{u}(\mathbf{x})$ under eq. (A.1)

$$\mathbb{E}[\mathbf{x}] = \boldsymbol{\mu} , \tag{A.5}$$

$$\mathbb{E}[\mathbf{x}\mathbf{x}^T] = \boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T . \tag{A.6}$$

Let define two $d$-dimensional vectors $\mathbf{x} \sim \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $\mathbf{x}' \sim \mathcal{N}(\mathbf{x}'|\boldsymbol{\mu}', \boldsymbol{\Sigma}')$. The Kullback-Leibler divergence between the two distributions is

$$\mathrm{KL}[\mathbf{x} \parallel \mathbf{x}'] = \frac{1}{2}\left(\frac{|\boldsymbol{\Sigma}'|}{|\boldsymbol{\Sigma}|}\right) + \mathrm{tr}(\boldsymbol{\Sigma}'^{-1}\boldsymbol{\Sigma}\mathbf{I}) + (\boldsymbol{\mu}' - \boldsymbol{\mu})^T\boldsymbol{\Sigma}'^{-1}(\boldsymbol{\mu}' - \boldsymbol{\mu}) . \tag{A.7}$$

Let $t(\mathbf{x})$ be a function of $\mathbf{x}$ and let

$$\tilde{p}(\mathbf{x}) = \frac{1}{Z} t(\mathbf{x}) \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \,, \tag{A.8}$$

$$Z = \int t(\mathbf{x}) \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) d\mathbf{x} \,. \tag{A.9}$$

Then, we have that

$$\mathbb{E}_{\tilde{p}}[\mathbf{x}] = \boldsymbol{\mu} + \boldsymbol{\Sigma} \frac{\partial \log Z}{\partial \boldsymbol{\mu}} \,, \tag{A.10}$$

$$\mathbb{E}_{\tilde{p}}[\mathbf{x}\mathbf{x}^T] - \mathbb{E}_{\tilde{p}}[\mathbf{x}]\mathbb{E}_{\tilde{p}}[\mathbf{x}]^T = \boldsymbol{\Sigma} - \boldsymbol{\Sigma} \left( \frac{\partial \log Z}{\partial \boldsymbol{\mu}} \left( \frac{\partial \log Z}{\partial \boldsymbol{\mu}} \right)^T - 2 \frac{\partial \log Z}{\partial \boldsymbol{\Sigma}} \right) \boldsymbol{\Sigma} \,. \tag{A.11}$$

The exponential family of distributions is closed under product and division. The product of two Gaussian distribution is also Gaussian, but not normalized. Namely:

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) \propto \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \,, \tag{A.12}$$

where

$$\boldsymbol{\Sigma} = (\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1})^{-1} \,, \tag{A.13}$$

$$\boldsymbol{\mu} = \boldsymbol{\Sigma} \left( \boldsymbol{\Sigma}_1^{-1} \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_2^{-1} \boldsymbol{\mu}_2 \right) \,, \tag{A.14}$$

and the normalization constant $z$ of the product is given by

$$z = \sqrt{\frac{|\boldsymbol{\Sigma}|}{(2\pi)^d |\boldsymbol{\Sigma}_1||\boldsymbol{\Sigma}_2|}} \exp \left\{ -\frac{1}{2} (\boldsymbol{\mu}_1^T \boldsymbol{\Sigma}_1^{-1} \boldsymbol{\mu}_1 + \boldsymbol{\mu}_2^T \boldsymbol{\Sigma}_2^{-1} \boldsymbol{\mu}_2 - \boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}) \right\} \,. \tag{A.15}$$

Similarly, the division of two Gaussian distribution is also Gaussian. In particular,

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) / \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) \propto \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \,, \tag{A.16}$$

where

$$\boldsymbol{\Sigma} = (\boldsymbol{\Sigma}_1^{-1} - \boldsymbol{\Sigma}_2^{-1})^{-1} \,, \tag{A.17}$$

$$\boldsymbol{\mu} = \boldsymbol{\Sigma} \left( \boldsymbol{\Sigma}_1^{-1} \boldsymbol{\mu}_1 - \boldsymbol{\Sigma}_2^{-1} \boldsymbol{\mu}_2 \right) \,, \tag{A.18}$$

and the corresponding normalization constant $z$ is given by

$$z = \sqrt{\frac{(2\pi)^d |\boldsymbol{\Sigma}||\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|}} \exp \left\{ -\frac{1}{2} (\boldsymbol{\mu}_1^T \boldsymbol{\Sigma}_1^{-1} \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2^T \boldsymbol{\Sigma}_2^{-1} \boldsymbol{\mu}_2 - \boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}) \right\} \,. \tag{A.19}$$

# Appendix B

# Calculations

In this Appendix we give all the details to implement the EP algorithm for the proposed method described in Chapter 5. In particular, we describe how to reconstruct the posterior approximation from the approximate factors and how to refine these factors. We also detail the computation of the EP approximation to the marginal likelihood and its gradients.

## B.1 Reconstruction of the posterior approximation

In this section we show how to obtain the posterior distribution by multiplying the approximate factors $\tilde{\phi}_i^c(\mathbf{u}^{y_i}, \mathbf{u}^c)$ and the prior $p(\mathbf{u}|\overline{\mathcal{X}})$. From the main manuscript we know that these elements have the following form:

$$\tilde{\phi}_i^c(\mathbf{u}^{y_i}, \mathbf{u}^c) = \tilde{s}_{i,c}\, \tilde{\mathcal{N}}(\mathbf{u}^{y_i}|\tilde{\mathbf{m}}_{y_i}, \tilde{\mathbf{V}}_{y_i})\tilde{\mathcal{N}}(\mathbf{u}^c|\tilde{\mathbf{m}}_c, \tilde{\mathbf{V}}_c) \tag{B.1}$$

$$p(\mathbf{u}|\overline{\mathcal{X}}) = \prod_{k=1}^{C} p(\mathbf{u}^k|\overline{\mathbf{X}}_k) = \prod_{k=1}^{C} \mathcal{N}(\mathbf{u}^k|\mathbf{0}, \mathbf{K}_{MM}^k)\,, \tag{B.2}$$

where $\tilde{\mathcal{N}}(\mathbf{u}^c|\tilde{\mathbf{m}}_c, \tilde{\mathbf{V}}_c)$ is an unnormalized Gaussian with natural parameters $\tilde{\mathbf{m}}_c$ and $\tilde{\mathbf{V}}_c$ and $\mathbf{K}_{MM}^k$ is a covariance matrix of size $M$x$M$ with the prior covariance among the values of the inducing points $\overline{\mathbf{X}}_k$. $\tilde{\mathbf{V}}_{y_i}$, $\tilde{\mathbf{V}}_c$, $\tilde{\mathbf{m}}_{y_i}$ and $\tilde{\mathbf{m}}_c$ have the following especial form (see Section B.3 for the detailed derivation):

$$\tilde{\mathbf{V}}_{y_i} = C_{i,c}^{1,y_i}\mathbf{w}_i^{y_i}(\mathbf{w}_i^{y_i})^T \tag{B.3}$$

$$\tilde{\mathbf{V}}_c = C_{i,c}^{1}\mathbf{w}_i^{c}(\mathbf{w}_i^{c})^T \tag{B.4}$$

$$\tilde{\mathbf{m}}_{y_i} = C_{i,c}^{2,y_i}\mathbf{w}_i^{y_i} \tag{B.5}$$

$$\tilde{\mathbf{m}}_c = C_{i,c}^{2}\mathbf{w}_i^{c}\,. \tag{B.6}$$

where $\mathbf{w}_i^{y_i} = \mathbf{K}_{i,M}^{y_i}(\mathbf{K}_{MM}^{y_i})^{-1}$, $\mathbf{w}_i^{c} = \mathbf{K}_{i,M}^{c}(\mathbf{K}_{MM}^{c})^{-1}$, and $C_{i,c}^{1,y_i}$, $C_{i,c}^{1}$, $C_{i,c}^{2,y_i}$ and $C_{i,c}^{2}$ are:

$$C_{i,c}^{1,y_i} = \left( \left[ \frac{\alpha_{i,c}^2 + \alpha_{i,c}\beta_{i,c}}{b_i^{y_i} + b_i^c} \right]^{-1} - (\mathbf{w}_i^{y_i})^T \mathbf{V}_{y_i}^{\setminus i,c} \mathbf{w}_i^{y_i} \right)^{-1} \tag{B.7}$$

$$C_{i,c}^{1} = \left( \left[ \frac{\alpha_{i,c}^2 + \alpha_{i,c}\beta_{i,c}}{b_i^{y_i} + b_i^c} \right]^{-1} - (\mathbf{w}_i^{c})^T \mathbf{V}_{c}^{\setminus i,c} \mathbf{w}_i^{c} \right)^{-1} \tag{B.8}$$

$$C_{i,c}^{2,y_i} = \left[ \frac{\alpha_{i,c}}{\sqrt{b_i^{y_i} + b_i^c}} + C_{i,c}^{1,y_i} (\mathbf{w}_i^{y_i})^T \mathbf{m}_{y_i}^{\setminus i,c} + \frac{\alpha_{i,c}}{\sqrt{b_i^{y_i} + b_i^c}} C_{i,c}^{1,y_i} (\mathbf{w}_i^{y_i})^T \mathbf{V}_{y_i}^{\setminus i,c} \mathbf{w}_i^{y_i} \right] \tag{B.9}$$

$$C_{i,c}^{2} = \left[ \frac{\alpha_{i,c}}{\sqrt{b_i^{y_i} + b_i^c}} + C_{i,c}^{1} (\mathbf{w}_i^{c})^T \mathbf{m}_{c}^{\setminus i,c} + \frac{\alpha_{i,c}}{\sqrt{b_i^{y_i} + b_i^c}} C_{i,c}^{1} (\mathbf{w}_i^{c})^T \mathbf{V}_{c}^{\setminus i,c} \mathbf{w}_i^{c} \right] . \tag{B.10}$$

We also know from the main manuscript that the posterior approximation will have the following form

$$q(\mathbf{u}) = \frac{1}{Z_q} \prod_{i=1}^{N} \left[ \prod_{c \neq y_i} \tilde{\phi}_i^c(\mathbf{u}^{y_i}, \mathbf{u}^c) \right] \prod_{K=1}^{C} p(\mathbf{u}^k | \overline{\mathbf{X}}_k) \, . \tag{B.11}$$

Given that all the factors are Gaussian, a distribution that is closed under product and division, $q(\mathbf{u})$ is also Gaussian. In particular, the posterior approximation $q(\mathbf{u}) = \prod_{k=1}^{C} \mathcal{N}(\mathbf{u}|, \mathbf{m}_k, \mathbf{V}_k)$. The parameters of this distribution can be obtained by using the formulas given in the Appendix A, leading to

$$\mathbf{V}_k = [(\mathbf{K}_{MM}^k)^{-1} + \mathbf{W}_k \Delta_k \mathbf{W}_k^T]^{-1} \tag{B.12}$$

$$\mathbf{m}_k = \mathbf{V}_k \mathbf{W}_k \tilde{\boldsymbol{\mu}}_k \, , \tag{B.13}$$

where $\mathbf{W}_k = (\mathbf{w}_1^k, \ldots, \mathbf{w}_N^k)$ is a $M$x$N$ matrix, $\Delta_k$ is a diagonal $N$x$N$ matrix where each component of the diagonal has the following form

$$\Delta_{i,i}^k = C_{i,k}^1 \mathbb{1}(k \neq y_i) + \left( \sum_{k' \neq y_i} C_{i,k'}^{1,y_i} \right) \mathbb{1}(k = y_i) \, , \tag{B.14}$$

where $\mathbb{1}(\cdot)$ is an indicator function, which will be 1 if the condition holds and zero otherwise, and $\tilde{\boldsymbol{\mu}}_k$ is a vector where each component is defined by

$$\tilde{\mu}_i^k = C_{i,k}^2 \mathbb{1}(k \neq y_i) + \left( \sum_{k' \neq y_i} C_{i,k'}^{2,y_i} \right) \mathbb{1}(k = y_i) \, . \tag{B.15}$$

## B.2 Computation of the cavity distribution

Here we will obtain the expressions for the parameters of the cavity distribution $q^{\setminus i,c}$. This distribution is computed by dividing the posterior approximation by the corresponding approximate factor:

$$q(\mathbf{u})^{\backslash i,c} \propto \frac{q(\mathbf{u})}{\tilde{\phi}_i^c(\mathbf{u}^{y_i}, \mathbf{u}^c)}\,. \tag{B.16}$$

Given that all factors are Gaussian, the resulting distribution will also be Gaussian. The parameters can be obtain by using again the formulas in the Appendix A:

$$\begin{aligned}
\mathbf{V}_{y_i}^{\backslash i,c} &= (\mathbf{V}_{y_i}^{-1} - \tilde{\mathbf{V}}_{y_i})^{-1} \\
&= (\mathbf{V}_{y_i}^{-1} - C_{i,c}^{1,y_i}\mathbf{w}_i^{y_i}(\mathbf{w}_i^{y_i})^T)^{-1} \\
&= \mathbf{V}_{y_i} + \mathbf{V}_{y_i}\mathbf{w}_i^{y_i}[(C_{i,c}^{1,y_i})^{-1} - \mathbf{w}_i^{y_i}\mathbf{V}_{y_i}(\mathbf{w}_i^{y_i})^T]^{-1}(\mathbf{w}_i^{y_i})^T\mathbf{V}_{y_i}
\end{aligned} \tag{B.17}$$

$$\mathbf{V}_c^{\backslash i,c} = \mathbf{V}_c + \mathbf{V}_c\mathbf{w}_i^c[(C_{i,c}^1)^{-1} - \mathbf{w}_i^c\mathbf{V}_c(\mathbf{w}_i^c)^T]^{-1}(\mathbf{w}_i^c)^T\mathbf{V}_c \tag{B.18}$$

$$\begin{aligned}
\mathbf{m}_{y_i}^{\backslash i,c} &= \mathbf{V}_{y_i}^{\backslash i,c}(\mathbf{V}_{y_i}^{-1}\mathbf{m}_{y_i} - \tilde{\mathbf{m}}_{y_i}) \\
&= \mathbf{V}_{y_i}^{\backslash i,c}(\mathbf{V}_{y_i}^{-1}\mathbf{m}_{y_i} - C_{i,c}^{2,y_i}\mathbf{w}_i^{y_i}) \\
&= \mathbf{V}_{y_i}^{\backslash i,c}\mathbf{V}_{y_i}^{-1}\mathbf{m}_{y_i} - C_{i,c}^{2,y_i}\mathbf{w}_i^{y_i}\mathbf{V}_{y_i}^{\backslash i,c} \\
&= \mathbf{V}_{y_i}\mathbf{V}_{y_i}^{-1}\mathbf{m}_{y_i} + \mathbf{V}_{y_i}\mathbf{w}_i^{y_i}[(C_{i,c}^{1,y_i})^{-1} - \mathbf{w}_i^{y_i}\mathbf{V}_{y_i}(\mathbf{w}_i^{y_i})^T]^{-1}(\mathbf{w}_i^{y_i})^T\mathbf{V}_{y_i}\mathbf{V}_{y_i}^{-1}\mathbf{m}_{y_i} \\
&\quad - C_{i,c}^{2,y_i}\mathbf{w}_i^{y_i}\mathbf{V}_{y_i}^{\backslash i,c} \\
&= \mathbf{m}_{y_i} + \mathbf{V}_{y_i}\mathbf{w}_i^{y_i}[(C_{i,c}^{1,y_i})^{-1} - \mathbf{w}_i^{y_i}\mathbf{V}_{y_i}(\mathbf{w}_i^{y_i})^T]^{-1}(\mathbf{w}_i^{y_i})^T\mathbf{m}_{y_i} - \mathbf{V}_{y_i}\mathbf{w}^T C_{i,c}^{2,y_i} \\
&\quad - \mathbf{V}_{y_i}\mathbf{w}_i^{y_i}[(C_{i,c}^{1,y_i})^{-1} - \mathbf{w}_i^{y_i}\mathbf{V}_{y_i}(\mathbf{w}_i^{y_i})^T]^{-1}(\mathbf{w}_i^{y_i})^T\mathbf{V}_{y_i}\mathbf{w}_i^{y_i}C_{i,c}^{2,y_i}
\end{aligned} \tag{B.19}$$

$$\begin{aligned}
\mathbf{m}_c^{\backslash i,c} = \mathbf{m}_c &+ \mathbf{V}_c\mathbf{w}_i^c[(C_{i,c}^1)^{-1} - \mathbf{w}_i^c\mathbf{V}_c(\mathbf{w}_i^c)^T]^{-1}(\mathbf{w}_i^c)^T\mathbf{m}_c - \mathbf{V}_c\mathbf{w}^T C_{i,c}^2 \\
&- \mathbf{V}_c\mathbf{w}_i^c[(C_{i,c}^1)^{-1} - \mathbf{w}_i^c\mathbf{V}_c(\mathbf{w}_i^c)^T]^{-1}(\mathbf{w}_i^c)^T\mathbf{V}_c\mathbf{w}_i^c C_{i,c}^2\,,
\end{aligned} \tag{B.20}$$

where we have used the Woodbury matrix identity and set $\mathbf{w}_i^{y_i} = \mathbf{K}_{i,M}^{y_i}(\mathbf{K}_{MM}^{y_i})^{-1}$, $\mathbf{w}_i^c = \mathbf{K}_{i,M}^c(\mathbf{K}_{MM}^c)^{-1}$ and the constants $C_{i,c}^{1,y_i}$, $C_{i,c}^1$, $C_{i,c}^{2,y_i}$ and $C_{i,c}^2$ are the same that we mentioned in Section B.1.

## B.3 Update of the approximate factors

In this section we show how to find the approximate factors $\tilde{\phi}_i^c$ once the cavity distribution $q^{\backslash i,c}$ has already been computed. We know that the exact factor is:

$$\phi_i^c = \Phi\left(\frac{\hat{a}_i^{y_i} - \hat{a}_i^c}{\sqrt{\hat{b}_i^{y_i} + \hat{b}_i^c}}\right)\,, \tag{B.21}$$

where $\hat{a}_i^{y_i} = \mathbf{K}_{i,M}^{y_i}(\mathbf{K}_{MM}^{y_i})^{-1}\mathbf{u}^{y_i}$, $\hat{a}_i^c = \mathbf{K}_{i,M}^c(\mathbf{K}_{MM}^c)^{-1}\mathbf{u}^c$, $\hat{b}_i^{y_i} = \mathbf{K}_{i,i}^{y_i} - \mathbf{K}_{i,M}^{y_i}(\mathbf{K}_{MM}^{y_i})^{-1}\mathbf{K}_{M,i}^{y_i}$ and $\hat{b}_i^c = \mathbf{K}_{i,i}^c - \mathbf{K}_{i,M}^c(\mathbf{K}_{MM}^c)^{-1}\mathbf{K}_{M,i}^c$. The normalization constant of $\phi_i^c q^{\backslash i,c}$ has the following form:

$$
\begin{aligned}
Z_{i,c} &= \int \Phi\left(\frac{\hat{a}_i^{y_i} - \hat{a}_i^c}{\sqrt{\hat{b}_i^{y_i} + \hat{b}_i^c}}\right) \mathcal{N}(\mathbf{u}^{y_i}|\mathbf{m}_{y_i}^{\setminus i,c}, \mathbf{V}_{y_i}^{\setminus i,c}) \mathcal{N}(\mathbf{u}^c|\mathbf{m}_c^{\setminus i,c}, \mathbf{V}_c^{\setminus i,c}) d\mathbf{u}^{y_i} d\mathbf{u}^c \\
&= \Phi\left(\frac{a_i^{y_i} - a_i^c}{\sqrt{b_i^{y_i} + b_i^c}}\right),
\end{aligned}
\tag{B.22}
$$

where:

$$a_i^{y_i} = \mathbf{K}_{i,M}^{y_i}(\mathbf{K}_{MM}^{y_i})^{-1}\mathbf{m}_{y_i}^{\setminus i,c} = \mathbf{w}_i^{y_i}\mathbf{m}_{y_i}^{\setminus i,c} \tag{B.23}$$

$$a_i^c = \mathbf{K}_{i,M}^c(\mathbf{K}_{MM}^c)^{-1}\mathbf{m}_c^{\setminus i,c} = \mathbf{w}_i^c\mathbf{m}_c^{\setminus i,c} \tag{B.24}$$

$$b_i^{y_i} = \mathbf{K}_{i,i}^{y_i} - \mathbf{K}_{i,M}^{y_i}(\mathbf{K}_{MM}^{y_i})^{-1}\mathbf{K}_{M,i}^{y_i} + \mathbf{K}_{i,M}^{y_i}(\mathbf{K}_{MM}^{y_i})^{-1}\mathbf{V}_{y_i}^{\setminus i,c}(\mathbf{K}_{MM}^{y_i})^{-1}\mathbf{K}_{M,i}^{y_i} \tag{B.25}$$

$$b_i^c = \mathbf{K}_{i,i}^c - \mathbf{K}_{i,M}^c(\mathbf{K}_{MM}^c)^{-1}\mathbf{K}_{M,i}^c + \mathbf{K}_{i,M}^c(\mathbf{K}_{MM}^c)^{-1}\mathbf{V}_c^{\setminus i,c}(\mathbf{K}_{MM}^c)^{-1}\mathbf{K}_{M,i}^c\,, \tag{B.26}$$

where $\mathbf{w}_i^{y_i} = \mathbf{K}_{i,M}^{y_i}(\mathbf{K}_{MM}^{y_i})^{-1}$ and $\mathbf{w}_i^c = \mathbf{K}_{i,M}^c(\mathbf{K}_{MM}^c)^{-1}$. We compute the derivatives of $\log Z_{i,c}$ with respect to the parameters of $q^{\setminus i,c}$:

$$\frac{\partial \log Z_{i,c}}{\partial \mathbf{m}_{y_i}^{\setminus i,c}} = \frac{\mathcal{N}(\beta_{i,c}|0,1)}{\Phi(\beta_{i,c})}\frac{1}{\sqrt{b_i^{y_i} + b_i^c}}\mathbf{w}_i^{y_i} = \frac{\alpha_{i,c}}{\sqrt{b_i^{y_i} + b_i^c}}\mathbf{w}_i^{y_i} \tag{B.27}$$

$$\frac{\partial \log Z_{i,c}}{\partial \mathbf{m}_c^{\setminus i,c}} = \frac{-\alpha_{i,c}}{\sqrt{b_i^{y_i} + b_i^c}}\mathbf{w}_i^c \tag{B.28}$$

$$\frac{\partial \log Z_{i,c}}{\partial \mathbf{V}_{y_i}^{\setminus i,c}} = -\frac{1}{2}\frac{\mathcal{N}(\beta_{i,c}|0,1)}{\Phi(\beta_{i,c})}\frac{a_i^{y_i} - a_i^c}{\sqrt{b_i^{y_i} + b_i^c}}\frac{1}{b_i^{y_i} + b_i^c} = -\frac{1}{2}\alpha_{i,c}\beta_{i,c}\frac{1}{b_i^{y_i} + b_i^c}\mathbf{w}_i^{y_i}(\mathbf{w}_i^{y_i})^T \tag{B.29}$$

$$\frac{\partial \log Z_{i,c}}{\partial \mathbf{V}_c^{\setminus i,c}} = -\frac{1}{2}\frac{N(\beta_{i,c}|0,1)}{\Phi(\beta_{i,c})}\frac{a_i^{y_i} - a_i^c}{\sqrt{b_i^{y_i} + b_i^c}}\frac{1}{b_i^{y_i} + b_i^c} = -\frac{1}{2}\alpha_{i,c}\beta_{i,c}\frac{1}{b_i^{y_i} + b_i^c}\mathbf{w}_i^c(\mathbf{w}_i^c)^T\,, \tag{B.30}$$

where

$$\alpha_{i,c} = \frac{N(\beta_{i,c}|0,1)}{\Phi(\beta_{i,c})} \tag{B.31}$$

$$\beta_{i,c} = \frac{a_i^{y_i} - a_i^c}{\sqrt{b_i^{y_i} + b_i^c}}\,. \tag{B.32}$$

By following the Appendix A we can obtain the moments of $\phi_i^c q^{\setminus i,c}$ (means $\hat{\mathbf{m}}_{y_i}$, $\hat{\mathbf{m}}_c$ and covariances $\hat{\mathbf{V}}_{y_i}$, $\hat{\mathbf{V}}_c$) from the derivatives of $\log Z_{i,c}$ with respect to the parameters of $q^{\setminus i,c}$. Namely:

$$\hat{\mathbf{m}}_{y_i} = \mathbf{m}_{y_i}^{\setminus i,c} + \mathbf{V}_{y_i}^{\setminus i,c} \frac{\partial \log Z_{i,c}}{\partial \mathbf{m}_{y_i}^{\setminus i,c}} = \mathbf{m}_{y_i}^{\setminus i,c} + \mathbf{V}_{y_i}^{\setminus i,c} \frac{\alpha_{i,c}}{\sqrt{b_i^{y_i} + b_i^c}} \mathbf{w}_i^{y_i} \tag{B.33}$$

$$\hat{\mathbf{m}}_c = \mathbf{m}_c^{\setminus i,c} - \mathbf{V}_c^{\setminus i,c} \frac{\partial \log Z_{i,c}}{\partial \mathbf{m}_c^{\setminus i,c}} = \mathbf{m}_c^{\setminus i,c} - \mathbf{V}_c^{\setminus i,c} \frac{\alpha_{i,c}}{\sqrt{b_i^{y_i} + b_i^c}} \mathbf{w}_i^c \tag{B.34}$$

$$\begin{aligned}
\hat{\mathbf{V}}_{y_i} &= \mathbf{V}_{y_i}^{\setminus i,c} - \mathbf{V}_{y_i}^{\setminus i,c} \left( \left( \frac{\partial \log Z_{i,c}}{\partial \mathbf{m}_{y_i}^{\setminus i,c}} \right) \left( \frac{\partial \log Z_{i,c}}{\partial \mathbf{m}_{y_i}^{\setminus i,c}} \right)^T - 2 \frac{\partial \log Z_{i,c}}{\partial \mathbf{V}_{y_i}^{\setminus i,c}} \right) \mathbf{V}_{y_i}^{\setminus i,c} \\
&= \mathbf{V}_{y_i}^{\setminus i,c} - \mathbf{V}_{y_i}^{\setminus i,c} \left[ \frac{\alpha_{i,c}^2}{b_i^{y_i} + b_i^c} \mathbf{w}_i^{y_i} (\mathbf{w}_i^{y_i})^T + \frac{\alpha_{i,c}\beta_{i,c}}{b_i^{y_i} + b_i^c} \mathbf{w}_i^{y_i} (\mathbf{w}_i^{y_i})^T \right] \mathbf{V}_{y_i}^{\setminus i,c} \\
&= \mathbf{V}_{y_i}^{\setminus i,c} - \mathbf{V}_{y_i}^{\setminus i,c} \left[ \frac{\alpha_{i,c}^2 + \alpha_{i,c}\beta_{i,c}}{b_i^{y_i} + b_i^c} \mathbf{w}_i^{y_i} (\mathbf{w}_i^{y_i})^T \right] \mathbf{V}_{y_i}^{\setminus i,c}
\end{aligned} \tag{B.35}$$

$$\begin{aligned}
\hat{\mathbf{V}}_c &= \mathbf{V}_c^{\setminus i,c} - \mathbf{V}_c^{\setminus i,c} \left( \left( \frac{\partial \log Z_{i,c}}{\partial \mathbf{m}_c^{\setminus i,c}} \right) \left( \frac{\partial \log Z_{i,c}}{\partial \mathbf{m}_c^{\setminus i,c}} \right)^T - 2 \frac{\partial \log Z_{i,c}}{\partial \mathbf{V}_c^{\setminus i,c}} \right) \mathbf{V}_c^{\setminus i,c} \\
&= \mathbf{V}_c^{\setminus i,c} - \mathbf{V}_c^{\setminus i,c} \left[ \frac{\alpha_{i,c}^2 + \alpha_{i,c}\beta_{i,c}}{b_i^{y_i} + b_i^c} \mathbf{w}_i^c (\mathbf{w}_i^c)^T \right] \mathbf{V}_c^{\setminus i,c} .
\end{aligned} \tag{B.36}$$

Now we can find the parameters of the approximate factor $\tilde{\phi}_i^c$, which is obtained as $\tilde{\phi}_i^c = Z_{i,c} q^{new} / q^{\setminus i,c}$, where $q^{new}$ is a Gaussian distribution with the parameters of $\phi_i^c q^{\setminus i,c}$, that we just computed. By following the equations given in the Appendix A we obtain the precision matrices of the approximate factor:

$$\begin{aligned}
\tilde{\mathbf{V}}_{y_i} &= \hat{\mathbf{V}}_{y_i}^{-1} - (\mathbf{V}_{y_i}^{\setminus i,c})^{-1} \\
&= \left( \mathbf{V}_{y_i}^{\setminus i,c} - \mathbf{V}_{y_i}^{\setminus i,c} \mathbf{w}_i^{y_i} \left[ \frac{\alpha_{i,c}^2 + \alpha_{i,c}\beta_{i,c}}{b_i^{y_i} + b_i^c} \right] (\mathbf{w}_i^{y_i})^T \mathbf{V}_{y_i}^{\setminus i,c} \right)^{-1} - (\mathbf{V}_{y_i}^{\setminus i,c})^{-1} \\
&= (\mathbf{V}_{y_i}^{\setminus i,c})^{-1} + (\mathbf{V}_{y_i}^{\setminus i,c})^{-1} \mathbf{V}_{y_i}^{\setminus i,c} \mathbf{w}_i^{y_i} \left( \left[ \frac{\alpha_{i,c}^2 + \alpha_{i,c}\beta_{i,c}}{b_i^{y_i} + b_i^c} \right]^{-1} - (\mathbf{w}_i^{y_i})^T \mathbf{V}_{y_i}^{\setminus i,c} (\mathbf{V}_{y_i}^{\setminus i,c})^{-1} \mathbf{V}_{y_i}^{\setminus i,c} \mathbf{w}_i^{y_i} \right)^{-1} \\
&\qquad (\mathbf{w}_i^{y_i})^T \mathbf{V}_{y_i}^{\setminus i,c} (\mathbf{V}_{y_i}^{\setminus i,c})^{-1} - (\mathbf{V}_{y_i}^{\setminus i,c})^{-1} \\
&= \mathbf{w}_i^{y_i} \left( \left[ \frac{\alpha_{i,c}^2 + \alpha_{i,c}\beta_{i,c}}{b_i^{y_i} + b_i^c} \right]^{-1} - (\mathbf{w}_i^{y_i})^T \mathbf{V}_{y_i}^{\setminus i,c} \mathbf{w}_i^{y_i} \right)^{-1} (\mathbf{w}_i^{y_i})^T ,
\end{aligned} \tag{B.37}$$

where we have used the Woodbury matrix identity to invert $\hat{\mathbf{V}}_{y_i}^{-1}$. Let define $C_{i,c}^{1,y_i}$ and $C_{i,c}^1$ as

$$C_{i,c}^{1,y_i} = \left( \left[ \frac{\alpha_{i,c}^2 + \alpha_{i,c}\beta_{i,c}}{b_i^{y_i} + b_i^c} \right]^{-1} - (\mathbf{w}_i^{y_i})^T \mathbf{V}_{y_i}^{\setminus i,c} \mathbf{w}_i^{y_i} \right)^{-1} \tag{B.38}$$

$$C_{i,c}^{1} = \left( \left[ \frac{\alpha_{i,c}^2 + \alpha_{i,c}\beta_{i,c}}{b_i^{y_i} + b_i^c} \right]^{-1} - (\mathbf{w}_i^{c})^T \mathbf{V}_{c}^{\setminus i,c} \mathbf{w}_i^{c} \right)^{-1} . \tag{B.39}$$

The precision matrices of the approximate factors will be then:

$$\begin{aligned} \tilde{\mathbf{V}}_{y_i} &= C_{i,c}^{1,y_i} \mathbf{w}_i^{y_i} (\mathbf{w}_i^{y_i})^T \\ \tilde{\mathbf{V}}_{c} &= C_{i,c}^{1} \mathbf{w}_i^{c} (\mathbf{w}_i^{c})^T . \end{aligned} \tag{B.40}$$

For the first natural parameter we proceed in a similar way

$$\begin{aligned} \tilde{\mathbf{m}}_{y_i} &= \hat{\mathbf{V}}_{y_i}^{-1} \hat{\mathbf{m}}_{y_i} - (\mathbf{V}_{y_i}^{\setminus i,c})^{-1} \mathbf{m}_{y_i}^{\setminus i,c} \\ &= ((\mathbf{V}_{y_i}^{\setminus i,c})^{-1} + \tilde{\mathbf{V}}_{y_i}) \hat{\mathbf{m}}_{y_i} - (\mathbf{V}_{y_i}^{\setminus i,c})^{-1} \mathbf{m}_{y_i}^{\setminus i,c} \\ &= (\mathbf{V}_{y_i}^{\setminus i,c})^{-1} \hat{\mathbf{m}}_{y_i} + \tilde{\mathbf{V}}_{y_i} \hat{\mathbf{m}}_{y_i} - (\mathbf{V}_{y_i}^{\setminus i,c})^{-1} \mathbf{m}_{y_i}^{\setminus i,c} \\ &= (\mathbf{V}_{y_i}^{\setminus i,c})^{-1} \left[ \mathbf{m}_{y_i}^{\setminus i,c} + \mathbf{V}_{y_i}^{\setminus i,c} \frac{\alpha_{i,c}}{\sqrt{b_i^{y_i} + b_i^c}} \mathbf{w}_i^{y_i} \right] \\ &\qquad + \tilde{\mathbf{V}}_{y_i} \left[ \mathbf{m}_{y_i}^{\setminus i,c} + \mathbf{V}_{y_i}^{\setminus i,c} \frac{\alpha_{i,c}}{\sqrt{b_i^{y_i} + b_i^c}} \mathbf{w}_i^{y_i} \right] - (\mathbf{V}_{y_i}^{\setminus i,c})^{-1} \mathbf{m}_{y_i}^{\setminus i,c} \\ &= (\mathbf{V}_{y_i}^{\setminus i,c})^{-1} \mathbf{m}_{y_i}^{\setminus i,c} + (\mathbf{V}_{y_i}^{\setminus i,c})^{-1} \mathbf{V}_{y_i}^{\setminus i,c} \frac{\alpha_{i,c}}{\sqrt{b_i^{y_i} + b_i^c}} \mathbf{w}_i^{y_i} \\ &\qquad + \tilde{\mathbf{V}}_{y_i} \left[ \mathbf{m}_{y_i}^{\setminus i,c} + \mathbf{V}_{y_i}^{\setminus i,c} \frac{\alpha_{i,c}}{\sqrt{b_i^{y_i} + b_i^c}} \mathbf{w}_i^{y_i} \right] - (\mathbf{V}_{y_i}^{\setminus i,c})^{-1} \mathbf{m}_{y_i}^{\setminus i,c} \\ &= \frac{\alpha_{i,c}}{\sqrt{b_i^{y_i} + b_i^c}} \mathbf{w}_i^{y_i} + \tilde{\mathbf{V}}_{y_i} \mathbf{m}_{y_i}^{\setminus i,c} + \tilde{\mathbf{V}}_{y_i} \mathbf{V}_{y_i}^{\setminus i,c} \frac{\alpha_{i,c}}{\sqrt{b_i^{y_i} + b_i^c}} \mathbf{w}_i^{y_i} \\ &= \frac{\alpha_{i,c}}{\sqrt{b_i^{y_i} + b_i^c}} \mathbf{w}_i^{y_i} + C_{i,c}^{1,y_i} \mathbf{w}_i^{y_i} (\mathbf{w}_i^{y_i})^T \mathbf{m}_{y_i}^{\setminus i,c} + \frac{\alpha_{i,c}}{\sqrt{b_i^{y_i} + b_i^c}} C_{i,c}^{1,y_i} \mathbf{w}_i^{y_i} (\mathbf{w}_i^{y_i})^T \mathbf{V}_{y_i}^{\setminus i,c} \mathbf{w}_i^{y_i} \\ &= \left[ \frac{\alpha_{i,c}}{\sqrt{b_i^{y_i} + b_i^c}} + C_{i,c}^{1,y_i} (\mathbf{w}_i^{y_i})^T \mathbf{m}_{y_i}^{\setminus i,c} + \frac{\alpha_{i,c}}{\sqrt{b_i^{y_i} + b_i^c}} C_{i,c}^{1,y_i} (\mathbf{w}_i^{y_i})^T \mathbf{V}_{y_i}^{\setminus i,c} \mathbf{w}_i^{y_i} \right] \mathbf{w}_i^{y_i} , \end{aligned} \tag{B.41}$$

where we have used that $(V_{y_i}^{new})^{-1} = V_{y_i}^{-1} + \tilde{V}_{y_i}$. If we define $C_{i,c}^{2,y_i}$ and $C_{i,c}^{2}$ as

$$C_{i,c}^{2,y_i} = \left[ \frac{\alpha_{i,c}}{\sqrt{b_i^{y_i} + b_i^c}} + C_{i,c}^{1,y_i} (\mathbf{w}_i^{y_i})^T \mathbf{m}_{y_i}^{\setminus i,c} + \frac{\alpha_{i,c}}{\sqrt{b_i^{y_i} + b_i^c}} C_{i,c}^{1,y_i} (\mathbf{w}_i^{y_i})^T \mathbf{V}_{y_i}^{\setminus i,c} \mathbf{w}_i^{y_i} \right] \tag{B.42}$$

$$C_{i,c}^{2} = \left[ \frac{\alpha_{i,c}}{\sqrt{b_i^{y_i} + b_i^c}} + C_{i,c}^{1} (\mathbf{w}_i^{c})^T \mathbf{m}_{c}^{\setminus i,c} + \frac{\alpha_{i,c}}{\sqrt{b_i^{y_i} + b_i^c}} C_{i,c}^{1} (\mathbf{w}_i^{c})^T \mathbf{V}_{c}^{\setminus i,c} \mathbf{w}_i^{c} \right] , \tag{B.43}$$

we obtain the following expressions for the first natural parameters:

$$\tilde{\mathbf{m}}_{y_i} = C_{i,c}^{2,y_i} \mathbf{w}_i^{y_i} \tag{B.44}$$

$$\tilde{\mathbf{m}}_c = C_{i,c}^{2} \mathbf{w}_i^{c} \,. \tag{B.45}$$

Once we have these parameters we can compute the value of the normalization constant $\tilde{s}_{i,c}$, which guarantees that the approximate factor integrates the same as the exact factor with respect to $q^{\setminus i,c}$. Let $\boldsymbol{\theta}$ be the natural parameters of $q$ after the update and $\boldsymbol{\theta}^{\setminus i,c}$ the natural parameters of the cavity distribution $q^{\setminus i,c}$. Then,

$$\tilde{s}_{i,c} = \log Z_{i,c} + g(\boldsymbol{\theta}^{\setminus i,c}) - g(\boldsymbol{\theta}) \,, \tag{B.46}$$

where $g(\boldsymbol{\theta})$ is the log-normalizer of a multivariate Gaussian with natural parameters $\boldsymbol{\theta}$.

## B.4 Estimate of the marginal likelihood

As we have seen in Chapter 5, the estimate of the log marginal likelihood is

$$\log Z_q = g(\boldsymbol{\theta}) - g(\boldsymbol{\theta}_{prior}) + \sum_{i=1}^{N} \sum_{c \neq y_i} \log \tilde{s}_{i,c} \tag{B.47}$$

$$\log \tilde{s}_{i,c} = \log Z_{i,c} + g(\boldsymbol{\theta}^{\setminus i,c}) - g(\boldsymbol{\theta}) \,, \tag{B.48}$$

where $\boldsymbol{\theta}$, $\boldsymbol{\theta}^{\setminus i,c}$ and $\boldsymbol{\theta}_{prior}$ are the natural parameters of $q$, $q^{\setminus i,c}$ and $p(\mathbf{u}|\overline{\mathcal{X}})$ respectively and $g(\boldsymbol{\theta}')$ is the log-normalizer of a multivariate Gaussian with natural parameters $\boldsymbol{\theta}'$. If $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are the natural parameters of that Gaussian distribution over $m$ dimensions, then

$$g(\boldsymbol{\theta}') = \frac{m}{2} \log 2\pi + \frac{1}{2} \log |\boldsymbol{\Sigma}| + \frac{1}{2} \boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} \,, \tag{B.49}$$

which leads to

$$\log Z_q = \sum_{k=1}^{C} \frac{1}{2} \log |\mathbf{V}_k| + \frac{1}{2} \mathbf{m}_k^T \mathbf{V}_k^{-1} \mathbf{m}_k - \frac{1}{2} |\mathbf{K}_{MM}^k| + \sum_{i=1}^{N} \sum_{c \neq y_i} \log \tilde{s}_{i,c} \,, \tag{B.50}$$

with

$$\begin{aligned} \log \tilde{s}_{i,c} = & -\frac{1}{2} \log |1 - C_1 \mathbf{w}_k^T \mathbf{V}_k \mathbf{w}_k| + \frac{1}{2} \mathbf{m}_k^T \mathbf{w}_k ((C_1^k)^{-1} - \mathbf{w}_k^T \mathbf{V}_k \mathbf{w}_k)^{-} 1 \mathbf{w}_k^T \mathbf{m}_k \\ & + \frac{1}{2} (C_2^k)^2 \mathbf{w}_k^T \mathbf{V}_k \mathbf{w}_k + \frac{1}{2} (C_2^k)^2 \mathbf{w}_k^T \mathbf{V}_k \mathbf{w}_k ((C_1^k)^{-1} - \mathbf{w}_k^T \mathbf{V}_k \mathbf{w}_k)^{-1} \mathbf{w}_k^T \mathbf{V}_k \mathbf{w}_k \\ & - C_2^k \mathbf{w}_k^T \mathbf{m}_k - C_2^k \mathbf{w}_k^T \mathbf{V}_k \mathbf{w}_k ((C_1^k)^{-1} - \mathbf{w}_k^T \mathbf{V}_k \mathbf{w}_k)^{-1} \mathbf{w}_k^T \mathbf{m}_k \,, \end{aligned} \tag{B.51}$$

where we have used the Woodbury matrix identity, the matrix determinant lemma, that $(\mathbf{V}^{\setminus i,c})^{-1} = \mathbf{V}_k^{-1} - \tilde{\mathbf{V}}_k$, that $\mathbf{m}_k^{\setminus i,c} = \mathbf{V}^{\setminus i,c}(\mathbf{V}^{-1}\mathbf{m} - \tilde{\mathbf{m}})$ and using the vectors $\mathbf{w}_i^{y_i}$, $\mathbf{w}_i^c$ and the constants $C_{i,c}^{1,y_i}$, $C_{i,c}^1$, $C_{i,c}^{2,y_i}$ and $C_{i,c}^2$ already defined in Section B.1 and Section B.3.

## B.5 Gradient of $\log Z_q$ after convergence

Here we obtain the expression for the gradient of $\log Z_q$ after EP has converged. Let denote $\xi_j$ to one hyperparameter of the model (parameter of the covariance function or a component of the inducing points) and $\boldsymbol{\theta}$ and $\boldsymbol{\theta}_{prior}$ to the natural parameters of $q$ and $p(\mathbf{u}|\overline{\mathcal{X}})$ respectively. When EP has converged, the approximate factors can be considered to be fixed (it does not change with the model hyperparameters) (Seeger, 2005). In this case, it is only necessary to consider the direct dependency of $\log Z_{i,c}$ on $\xi_j$ (Seeger, 2005). Then, the gradient is given by:

$$\begin{aligned}\frac{\partial \log Z_q}{\partial \xi_j} &= \left(\frac{\partial g(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right)^T \frac{\partial \boldsymbol{\theta}}{\partial \xi_j} - \left(\frac{\partial g(\boldsymbol{\theta}_{prior})}{\partial \boldsymbol{\theta}_{prior}}\right)^T \frac{\partial \boldsymbol{\theta}_{prior}}{\partial \xi_j} + \sum_{i=1}^{N}\sum_{c\neq y_i} \frac{\partial \log Z_{i,c}}{\partial \xi_j} \\ &= \boldsymbol{\eta}^T \frac{\partial \boldsymbol{\theta}}{\partial \xi_j} - \left(\boldsymbol{\eta}_{prior}\right)^T \frac{\partial \boldsymbol{\theta}_{prior}}{\partial \xi_j} + \sum_{i=1}^{N}\sum_{c\neq y_i} \frac{\partial \log Z_{i,c}}{\partial \xi_j} \\ &= \boldsymbol{\eta}^T \frac{\partial \boldsymbol{\theta}_{prior}}{\partial \xi_j} - \left(\boldsymbol{\eta}_{prior}\right)^T \frac{\partial \boldsymbol{\theta}_{prior}}{\partial \xi_j} + \sum_{i=1}^{N}\sum_{c\neq y_i} \frac{\partial \log Z_{i,c}}{\partial \xi_j}\,,\end{aligned} \tag{B.52}$$

where we have used the chain rule of matrix derivatives (Petersen and Pedersen, 2012), the especial form of the derivatives when using inducing points (Snelson, 2007) and that $\boldsymbol{\theta} = \boldsymbol{\theta}_{prior} + \sum_{i=1}^{N}\sum_{c\neq y_i} \boldsymbol{\theta}_i^c$, with $\boldsymbol{\theta}_i^c$ the natural parameters of the approximate factor $\tilde{\phi}_i^c$. This gradient coincides with the one in eq. (5.19).

## B.6 Predictive distribution

Once the training has completed, we can use the posterior approximation to make predictions for new instances. For that, we first compute an approximate posterior evaluated at the location of the new instance $f(\mathbf{x}^*)$, denoted by $f^*$:

$$\begin{aligned}p(f^*|\mathbf{y},\overline{\mathcal{X}}) &= \int p(f^*|\mathbf{u})p(\mathbf{u}|\mathbf{y},\overline{\mathcal{X}})d\mathbf{u} \approx \int p(f^*|\mathbf{u})q(\mathbf{u})d\mathbf{u} \\ &\approx \prod_{k=1}^{C} \mathcal{N}(f^*|\mathbf{m}_k^*, \mathbf{V}_k^*)\,,\end{aligned} \tag{B.53}$$

where $\mathbf{m}_k^* = \mathbf{K}_{f^*,M}^k(\mathbf{K}_{MM}^k)^{-1}\mathbf{m}_k$ and $\mathbf{V}_k^* = \mathbf{K}_{f^*,f^*} - \mathbf{K}_{f^*,M}^k(\mathbf{K}_{MM}^k)^{-1}\mathbf{K}_{M,f^*}^k + \mathbf{K}_{f^*,M}^k(\mathbf{K}_{MM}^k)^{-1}\mathbf{V}_k(\mathbf{K}_{MM}^k)^{-1}\mathbf{K}_{M,f^*}^k$. $\mathbf{K}_{f^*,f^*}$ is the prior variance of the test point and $\mathbf{K}_{f^*,M}$ is the covariance matrix between the test point and the inducing points.

We can use this approximate posterior to obtain an approximate predictive distribution for the class label $y^*$:

$$
\begin{aligned}
p(y^*|\mathbf{x}^*, \mathbf{y}, \overline{\mathbf{X}}) &= \int p(y^*|\mathbf{x}^*, f^*) p(f^*|\mathbf{y}, \overline{\mathcal{X}}) df^* \\
&= \int p(y^*|\mathbf{x}^*, f^*) \prod_{k=1}^{C} \mathcal{N}(f^*|\mathbf{m}_k^*, \mathbf{V}_k^*) df^* \\
&= \int \left[ \prod_{c \neq y_i} \Theta(f_{y_i}^* - f_c^*) \right] \prod_{k=1}^{C} \mathcal{N}(f^*|\mathbf{m}_k^*, \mathbf{V}_k^*) df^* \\
&= \int \left[ \prod_{c \neq y_i} \Theta(f_{y_i}^* - f_c^*) \right] \prod_{c \neq y_i} \mathcal{N}(f^*|\mathbf{m}_c^*, \mathbf{V}_c^*) df^* \mathcal{N}(f_{y_i}^*|\mathbf{m}_{y_i}^*, \mathbf{V}_{y_i}^*) \\
&= \int \left[ \prod_{c \neq y_i} \Theta(f_{y_i}^* - f_c^*) \mathcal{N}(f^*|\mathbf{m}_c^*, \mathbf{V}_c^*) df^* \right] \mathcal{N}(f_{y_i}^*|\mathbf{m}_{y_i}^*, \mathbf{V}_{y_i}^*) \\
&= \int \prod_{c \neq y_i} \Phi\left( \frac{f_{y_i}^* - \mathbf{m}_c^*}{\sqrt{\mathbf{V}_c^*}} \right) \mathcal{N}(f_{y_i}^*|\mathbf{m}_{y_i}^*, \mathbf{V}_{y_i}^*) df^* ,
\end{aligned}
\tag{B.54}
$$

where $\Phi(\cdot)$ is the cumulative distribution function of a Gaussian distribution. This is an integral in one dimension and can easily be solved by quadrature.

# Bibliography

Bauer, M. S., van der Wilk, M., and Rasmussen, C. E. (2016). Understanding Probabilistic Sparse Gaussian Process Approximations. *arXiv:1606.04820 [stat]*. arXiv: 1606.04820.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Csató, L. and Opper, M. (2002). Sparse On-Line Gaussian Processes. *Neural Computation*, 14(3):641–668.

Geman, S. and Geman, D. (1984). Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741.

Golub, G. H. and Van Loan, C. F. (2012). *Matrix computations*, volume 3. JHU Press.

Hastings, W. K. (1970). Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika*, 57(1):97–109.

Hensman, J., Matthews, A., and Ghahramani, Z. (2015a). Scalable Variational Gaussian Process Classification. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*.

Hensman, J., Matthews, A. G., Filippone, M., and Ghahramani, Z. (2015b). MCMC for variationally sparse Gaussian processes. In *Advances in Neural Information Processing Systems*, pages 1648–1656.

Hernández-Lobato, D. (2010). *Prediction Based on Averages over Automatically Induced Learners: Ensemble Methods and Bayesian Techniques*. PhD thesis, Universidad Autónoma de Madrid.

Hernández-Lobato, D. and Hernández-Lobato, J. M. (2015). Scalable Gaussian Process Classification via Expectation Propagation. *ArXiv e-prints*.

Hernández-Lobato, D., Hernández-lobato, J. M., and Dupont, P. (2011). Robust Multi-Class Gaussian Process Classification. In Shawe-taylor, J., Zemel, R. s., Bartlett, P., Pereira, F. c. n., and Weinberger, K. q., editors, *Advances in Neural Information Processing Systems 24*, pages 280–288.

Hernández-Lobato, J. M., Li, Y., Rowland, M., Hernández-Lobato, D., Bui, T., and Turner, R. E. (2015). Black-box $\alpha$-divergence Minimization. *arXiv:1511.03243 [stat]*. arXiv: 1511.03243.

Kim, H.-C. and Ghahramani, Z. (2006). Bayesian Gaussian process classification with the EM-EP algorithm. *IEEE transactions on pattern analysis and machine intelligence*, 28(12):1948–1959.

Kingma, D. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*. arXiv: 1412.6980.

Kullback, S. and Leibler, R. A. (1951). On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86.

Lawrence, N., Seeger, M., and Herbrich, R. (2003). Fast Sparse Gaussian Process Methods: The Informative Vector Machine. *Proceedings of the 16th Annual Conference on Neural Information Processing Systems*, pages 609–616.

Li, Y., Hernandez-Lobato, J. M., and Turner, R. E. (2015). Stochastic Expectation Propagation. *arXiv:1506.04132 [cs, stat]*. arXiv: 1506.04132.

Lichman, M. (2013). *UCI Machine Learning Repository*. University of California, Irvine, School of Information and Computer Sciences.

Lázaro-Gredilla, M., Quiñonero-Candela, J., Rasmussen, C. E., and Figueiras-Vidal, A. R. (2010). Sparse Spectrum Gaussian Process Regression. *Journal of Machine Learning Research*, 11(Jun):1865–1881.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of State Calculations by Fast Computing Machines. *Journal of Chemical Physics*, 21:1087–1092.

Micchelli, C. A., Xu, Y., and Zhang, H. (2006). Universal Kernels. *Journal of Machine Learning Research*, 7(Dec):2651–2667.

Minka, T. P. (2001). Expectation Propagation for Approximate Bayesian Inference. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, UAI'01, pages 362–369, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press.

Naish-Guzman, A. and Holden, S. (2007). The generalized FITC approximation. In *Advances in Neural Information Processing Systems*, pages 1057–1064.

Petersen, K. B. and Pedersen, M. S. (2012). *The Matrix Cookbook*. Technical University of Denmark.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992). *Numerical Recipes in C (2Nd Ed.): The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA.

Quiñonero-Candela, J. and Rasmussen, C. E. (2005). A Unifying View of Sparse Approximate Gaussian Process Regression. *J. Mach. Learn. Res.*, 6:1939–1959.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rasmussen, C. E. and Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Roweis, S. (1999). Gaussian Identities.

Schwaighofer, A. and Tresp, V. (2002). Transductive and inductive methods for approximate Gaussian process regression. In *Advances in Neural Information Processing Systems*, pages 953–960.

Seeger, M. (2005). Expectation propagation for exponential families. Technical report, Department of EECS, University of California, Berkeley.

Seeger, M., Williams, C. K. I., and Lawrence, N. D. (2003). Fast Forward Selection to Speed Up Sparse Gaussian Process Regression. In *IN WORKSHOP ON AI AND STATISTICS 9*.

Silverman, B. W. (1985). Some Aspects of the Spline Smoothing Approach to Non-Parametric Regression Curve Fitting. *Journal of the Royal Statistical Society. Series B (Methodological)*, 47(1):1–52.

Snelson, E. and Ghahramani, Z. (2006). Sparse Gaussian Processes using Pseudo-inputs. In *Advances in Neural Information Processing Systems*, pages 1257–1264. MIT press.

Snelson, E. L. (2007). *Flexible and efficient Gaussian process models for machine learning*. PhD thesis, Citeseer.

Tresp, V. (2000). A Bayesian Committee Machine. *Neural Computation*, 12(11):2719–2741.

Wahba, G. (1990). *Spline Models for Observational Data*. CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics.

Williams, C. K. I. and Barber, D. (1998). Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351.

Williams, C. K. I., Rasmussen, C., Scwaighofer, A., and Tresp, V. (2002). Observations on the Nyström Method for Gaussian Process Prediction. Technical report.

Williams, C. K. I. and Seeger, M. (2001). Using the Nyström Method to Speed Up Kernel Machines. In Leen, T. K., Dietterich, T. G., and Tresp, V., editors, *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press.