Scalable Gaussian Process Classification via Expectation Propagation

Daniel Hernández-Lobato
Universidad Autónoma de Madrid

José Miguel Hernández-Lobato
Harvard University

Abstract

Variational methods have been recently considered for scaling the training process of Gaussian process classifiers to large datasets. As an alternative, we describe here how to train these classifiers efficiently using expectation propagation (EP). The proposed EP method allows to train Gaussian process classifiers on very large datasets, with millions of instances, that were out of the reach of previous implementations of EP. More precisely, it can be used for (i) training in a distributed fashion where the data instances are sent to different nodes in which the required computations are carried out, and for (ii) maximizing an estimate of the marginal likelihood using a stochastic approximation of the gradient. Several experiments involving large datasets show that the method described is competitive with the variational approach.

1 INTRODUCTION

Gaussian process classification is a framework that can be used to predict the class label associated to a new instance (Rasmussen and Williams, 2006). In the binary case it is considered a non-linear latent function f whose sign at each input location determines the corresponding label. Moreover, a Gaussian process prior is assumed for f. A practical difficulty is, however, that making inference about f is infeasible due to the non-Gaussian likelihood. Nevertheless, very efficient methods can be used to carry out the required computations in an approximate way (Kuss and Rasmussen, 2005; Nickisch and Rasmussen, 2008). The result is a non-parametric classifier that becomes more expressive as the number of training instances increases. Unfortunately, the training cost is $\mathcal{O}(n^3)$, where n is the number of instances. This cost can be improved by using a sparse representation for f (Quiñonero Candela and Rasmussen, 2005). A popular approach in this setting introduces additional data instances in the form of $m \ll n$ inducing points or pseudoinputs, whose location in input space determines the regions where f can change significantly (Snelson and Ghahramani, 2006; Naish-Guzman and Holden, 2008). Sparse representations lead to a training cost that scales like $\mathcal{O}(nm^2)$.

In order for Gaussian process classification to work well, good hyper-parameters for the prior on f (*e.g.*, the level of noise, the length-scales and the amplitude) and good inducing point locations must be learned from the data. This is often done by maximizing an estimate of the marginal likelihood. A limitation of existing methods is, however, that the estimate of the marginal likelihood cannot be expressed as a sum over the data instances (Naish-Guzman and Holden, 2008). This prevents using efficient techniques for maximization, such as stochastic gradient ascent or distributed computations. An exception to this is the recent work by Hensman et al. (2015), which combines ideas from stochastic variational inference (Hoffman et al., 2013) and from variational Gaussian processes (Titsias, 2009) to provide a scalable method for Gaussian process classification that can be applied to datasets with millions of data instances. Nevertheless, a disadvantage of the work of Hensman et al. (2015) is that one-dimensional quadratures are strictly needed since some of the computations do not have a closed form.

In this paper we introduce an alternative to the variational approach described by Hensman et al. (2015) that is based on expectation propagation (EP) (Minka, 2001). The proposed method allows to train Gaussian process classifiers on very large datasets that were out of the reach of previous approaches based on EP. For this, we show that in EP it is possible to update the posterior approximation for the Gaussian process f and the model hyper-parameters, including the inducing points, at the same time. Furthermore, in our EP

Appearing in Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS) 2016, Cadiz, Spain. JMLR: W&CP volume 41. Copyright 2016 by the authors.

formulation the estimate of the marginal likelihood is expressed as a sum over the data. This enables using stochastic gradient ascent to maximize such an estimate to find good model hyper-parameters and inducing points. We also show that the EP updates can also be implemented in a distributed fashion, by splitting the data across several computational nodes. In summary, our new EP formulation for Gaussian process classification has the same advantages as the variational approach of Hensman et al. (2015), with the convenience that all computations are tractable and one-dimensional quadrature methods are not required. Furthermore, our experiments, involving datasets with millions of instances, show that both approaches for Gaussian process classification perform very similar.

The EP algorithm proposed is not restricted to Gaussian process classification. It may be also used for efficient approximate inference in other models. Moreover, to our knowledge, this is the first time that the hyper-parameters and the posterior approximation are updated inside EP at the same time. Previous implementations run EP until convergence with fixed hyper-parameters, to then update the hyper-parameters using a small gradient step. This process is repeated, potentially re-using the last EP solution, until the hyper-parameters no longer change. All major Gaussian process toolboxes follow this approach (Rasmussen and Nickisch, 2010; Vanhatalo et al., 2013; The GPy authors, 2015). We show that such scheme is less efficient since EP may take long time to converge at the beginning, when the hyper-parameters are still very poor. This is also the first work in which stochastic gradients are used to update the hyper-parameters in EP. This enables hyper-parameter learning within EP on massive datasets. These advantages and the good results of our method suggest that, in general, all EP algorithms should be implemented as we propose here.

2 SCALABLE GAUSSIAN PROCESS CLASSIFICATION

We introduce here Gaussian process classification and the model considered. Then, we show that expectation propagation can be used for distributed training and that the model hyper-parameters can be inferred using stochastic gradients. The full details of the proposed EP method are found in the supplementary material.

2.1 Gaussian Process Classification

Assume some data in the form of a matrix of attributes $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^{\mathrm{T}}$ with labels $\mathbf{y} = (y_1, \ldots, y_n)$, where $y_i \in \{-1, 1\}$. The task is to predict the class label of a new instance. For this, we assume the labeling rule $y_i = \mathrm{sign}(f(\mathbf{x}_i) + \epsilon_i)$, where $f(\cdot)$ is a non-linear function and ϵ_i is standard Gaussian noise that accounts for mislabeled instances. Furthermore, we assume a Gaussian process prior over f with zero mean and covariance function $k(\cdot, \cdot)$ (Rasmussen and Williams, 2006). That is, $f \sim \mathcal{GP}(0, k(\cdot, \cdot))$. To make inference about $\mathbf{f} = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n))^{\mathrm{T}}$ given $\mathbf{y}$, Bayes' rule is used. Namely, $p(\mathbf{f}|\mathbf{y}) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f})/p(\mathbf{y})$ where $p(\mathbf{f})$ is a multivariate Gaussian distribution and $p(\mathbf{y})$, the marginal likelihood, can be maximized to find the parameters of the covariance function $k(\cdot, \cdot)$. The likelihood of $\mathbf{f}$ is $p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^n \Phi(y_i f_i)$, where $\Phi(\cdot)$ is the cdf of a standard Gaussian and $f_i = f(\mathbf{x}_i)$. This is a non-Gaussian likelihood which makes the posterior intractable. However, there are several techniques that can be used to get a Gaussian approximation of $p(\mathbf{f}|\mathbf{y})$ (Kuss and Rasmussen, 2005; Nickisch and Rasmussen, 2008). They all result in a non-parametric classifier. Unfortunately, all these methods scale like $\mathcal{O}(n^3)$, where n is the number of training instances.

A sparse representation for the Gaussian process f can be used to reduce the training cost. A popular approach introduces a dataset of $m \ll n$ inducing points $\overline{\mathbf{X}} = (\overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_m)^{\mathrm{T}}$ with associated values $\overline{\mathbf{f}} = (f(\overline{\mathbf{x}}_1), \ldots, f(\overline{\mathbf{x}}_m))^{\mathrm{T}}$ (Naish-Guzman and Holden, 2008; Snelson and Ghahramani, 2006). The aim of the inducing points, $\overline{\mathbf{X}}$, is to constrain the form of f and to indicate where it can significantly change. Given $\overline{\mathbf{X}}$, the prior for f is then approximated as $p(\mathbf{f}) = \int p(\mathbf{f}|\overline{\mathbf{f}})p(\overline{\mathbf{f}}|\overline{\mathbf{X}})d\overline{\mathbf{f}} \approx \int \left[\prod_{i=1}^n p(f_i|\overline{\mathbf{f}})\right] p(\overline{\mathbf{f}}|\overline{\mathbf{X}})d\overline{\mathbf{f}} = p_{\mathrm{FITC}}(\mathbf{f}|\overline{\mathbf{X}})$, where the Gaussian conditional $p(\mathbf{f}|\overline{\mathbf{f}})$ is replaced by a factorized distribution $\prod_{i=1}^n p(f_i|\overline{\mathbf{f}})$. This approximation is known as the full independent training conditional (FITC) (Quiñonero Candela and Rasmussen, 2005), and it leads to a Gaussian prior $p_{\mathrm{FITC}}(\mathbf{f}|\overline{\mathbf{X}})$ with a low-rank covariance matrix. This prior allows for approximate inference with cost $\mathcal{O}(nm^2)$. Finally, the inducing points $\overline{\mathbf{X}}$ are regarded as hyper-parameters to be learnt by maximizing the marginal likelihood $p(\mathbf{y})$.

2.2 Model Specification and Expectation Propagation

The original methods based on the FITC approximation do not express the estimate of the marginal likelihood, $p(\mathbf{y})$, as a sum across data instances. This makes infeasible the use of efficient stochastic algorithms for learning the model hyper-parameters. To avoid this, we follow Titsias (2009) and do not marginalize the values $\overline{\mathbf{f}}$ associated to the inducing points. Specifically, the posterior approximation we consider is $p(\mathbf{f}|\mathbf{y}) \approx \int p(\mathbf{f}|\overline{\mathbf{f}})q(\overline{\mathbf{f}})d\overline{\mathbf{f}}$, where q is a Gaussian distribution that approximates $p(\overline{\mathbf{f}}|\mathbf{y})$, *i.e.*, the posterior of the values associated to the inducing points. To obtain q, we use first on the exact posterior the FITC approximation

$p_{\text{FITC}}(\mathbf{f}|\overline{\mathbf{f}})$ introduced in Section 2.1. In particular,

$$\begin{aligned} p(\overline{\mathbf{f}}|\mathbf{y}) &= \frac{\int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\overline{\mathbf{f}})d\mathbf{f}p(\overline{\mathbf{f}}|\overline{\mathbf{X}})}{p(\mathbf{y}|\overline{\mathbf{X}})} \\ &\approx \frac{\int p(\mathbf{y}|\mathbf{f})p_{\text{FITC}}(\mathbf{f}|\overline{\mathbf{f}})d\mathbf{f}p(\overline{\mathbf{f}}|\overline{\mathbf{X}})}{p(\mathbf{y}|\overline{\mathbf{X}})} \\ &= \frac{\prod_{i=1}^n \phi_i(\overline{\mathbf{f}})p(\overline{\mathbf{f}}|\overline{\mathbf{X}})}{p(\mathbf{y}|\overline{\mathbf{X}})}\,, \end{aligned} \quad (1)$$

where $p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^n \Phi(y_i f_i)$, $p_{\text{FITC}}(\mathbf{f}|\overline{\mathbf{f}}) = \prod_{i=1}^n p(f_i|\overline{\mathbf{f}}) = \prod_{i=1}^n \mathcal{N}(f_i|m_i, s_i)$ and $\phi_i(\overline{\mathbf{f}}) = \int \Phi(y_i f_i)\mathcal{N}(f_i|m_i,s_i)df_i = \Phi(y_i m_i/\sqrt{s_i+1})$, with $m_i = \mathbf{K}_{f_i\overline{\mathbf{f}}}\mathbf{K}_{\overline{\mathbf{f}}\overline{\mathbf{f}}}^{-1}\overline{\mathbf{f}}$ and $s_i = \mathbf{K}_{f_if_i} - \mathbf{K}_{f_i\overline{\mathbf{f}}}\mathbf{K}_{\overline{\mathbf{f}}\overline{\mathbf{f}}}^{-1}\mathbf{K}_{\overline{\mathbf{f}}f_i}$. Furthermore, $\mathbf{K}_{\overline{\mathbf{f}}\overline{\mathbf{f}}}$ is a matrix with the prior covariances among the entries in $\overline{\mathbf{f}}$, $\mathbf{K}_{f_i\overline{\mathbf{f}}}$ is a row vector with the prior covariances between f_i and $\overline{\mathbf{f}}$ and $\mathbf{K}_{f_if_i}$ is the prior variance of f_i. Finally, $\mathcal{N}(\cdot|\mathbf{m},\mathbf{\Sigma})$ denotes the p.d.f of a multivariate Gaussian distribution with mean vector equal to $\mathbf{m}$ and covariance matrix equal to $\mathbf{\Sigma}$.

The r.h.s of (1) is an intractable posterior due to the non-Gaussianity of each ϕ_i. We use expectation propagation (EP) to obtain a Gaussian approximation q (Minka, 2001). In EP, each ϕ_i is approximated by an un-normalized Gaussian factor $\tilde{\phi}_i$ which is defined as:

$$\tilde{\phi}_i(\overline{\mathbf{f}}) = \tilde{s}_i \exp\left\{-\tfrac{\tilde{\nu}_i}{2}\overline{\mathbf{f}}^{\text{T}}\boldsymbol{\upsilon}_i\boldsymbol{\upsilon}_i^{\text{T}}\overline{\mathbf{f}} + \tilde{\mu}_i\overline{\mathbf{f}}^{\text{T}}\boldsymbol{\upsilon}_i\right\}\,, \quad (2)$$

where $\boldsymbol{\upsilon}_i = \mathbf{K}_{\overline{\mathbf{f}}\overline{\mathbf{f}}}^{-1}\mathbf{K}_{\overline{\mathbf{f}}f_i}$ is a m dimensional vector, and $\tilde{s}_i$, $\tilde{\nu}_i$ and $\tilde{\mu}_i$ are parameters to be estimated by EP. Importantly, $\tilde{\phi}_i$ has a one-rank precision matrix, which means that in practice only $\mathcal{O}(m)$ parameters need to be stored per each $\tilde{\phi}_i$. This is not an approximation and the optimal Gaussian factor $\tilde{\phi}_i$ approximating ϕ_i has this form (see the supplementary material for more details). The posterior approximation q is obtained by replacing in the r.h.s. of (1) each exact factor ϕ_i with the corresponding approximate factor $\tilde{\phi}_i$. Namely, $q(\overline{\mathbf{f}}) = \prod_{i=1}^n \tilde{\phi}_i(\overline{\mathbf{f}})p(\overline{\mathbf{f}}|\overline{\mathbf{X}})/Z_q$, where Z_q is a normalization constant that approximates the marginal likelihood $p(\mathbf{y}|\overline{\mathbf{X}})$. We note that all factors in q are Gaussian, including the prior. Thus, q is a multivariate Gaussian distribution over m dimensions.

EP updates each $\tilde{\phi}_i$ iteratively until-convergence as follows. First, $\tilde{\phi}_i$ is removed from q by computing $q^{\setminus i} \propto q/\tilde{\phi}_i$. Then, we minimize the Kullback-Leibler divergence between $Z_i^{-1}\phi_i q^{\setminus i}$, and q, *i.e.*, $\text{KL}[Z_i^{-1}\phi_i q^{\setminus i}||q]$, with respect to q, where Z_i is the normalization constant of $\phi_i q^{\setminus i}$. This involves matching the mean and the covariances of $Z_i^{-1}\phi_i q^{\setminus i}$, which can be obtained from the derivatives of $\log Z_i$ with respect to the (natural) parameters of $q^{\setminus i}$ (Seeger, 2006). Given a new distribution q, the approximate factor is $\tilde{\phi}_i = Z_i q/q^{\setminus i}$. This enforces that $\tilde{\phi}_i$ is similar to ϕ_i in regions of high posterior probability as estimated by $q^{\setminus i}$. These updates are done in parallel for efficiency reasons, *i.e.*, we compute $q^{\setminus i}$ and the new q, for $i = 1\,\ldots, n$, at the same time, and then update $\tilde{\phi}_i$ as before (Van Gerven et al., 2009; Hernández-Lobato et al., 2011). The new approximation q is obtained by multiplying all the $\tilde{\phi}_i$ and $p(\overline{\mathbf{f}}|\overline{\mathbf{X}})$. The normalization constant of q, Z_q, is the EP approximation of the marginal likelihood $p(\mathbf{y}|\overline{\mathbf{X}})$. The log of this constant is (Seeger, 2006):

$$\log Z_q = g(\boldsymbol{\theta}) - g(\boldsymbol{\theta}_{\text{prior}}) + \textstyle\sum_{i=1}^n \log \tilde{Z}_i\,, \quad (3)$$

where $\log \tilde{Z}_i = \log Z_i + g(\boldsymbol{\theta}^{\setminus i}) - g(\boldsymbol{\theta})$, with $\boldsymbol{\theta}$, $\boldsymbol{\theta}^{\setminus i}$ and $\boldsymbol{\theta}_{\text{prior}}$ being the natural parameters of q, $q^{\setminus i}$ and $p(\overline{\mathbf{f}}|\overline{\mathbf{X}})$, respectively. Furthermore, Z_i is the normalization constant of $\phi_i q^{\setminus i}$, and $g(\boldsymbol{\theta})$ is the log-normalizer of a multivariate Gaussian with natural parameters $\boldsymbol{\theta}$.

If EP converges (the $\tilde{\phi}_i$ do not change), the gradient of $\log Z_q$ with respect to the parameters of each $\tilde{\phi}_i$ is zero (Seeger, 2006). Thus, the gradient of $\log Z_q$ with respect to a hyper-parameter ξ_j (*i.e.*, a parameter of the covariance function $k(\cdot,\cdot)$ or a component of $\overline{\mathbf{X}}$) is:

$$\frac{\partial \log Z_q}{\partial \xi_j} = \left(\boldsymbol{\eta}^{\text{T}} - \boldsymbol{\eta}^{\text{T}}_{\text{prior}}\right)\frac{\partial \boldsymbol{\theta}_{\text{prior}}}{\partial \xi_j} + \sum_{i=1}^n \frac{\partial \log Z_i}{\partial \xi_j}\,, \quad (4)$$

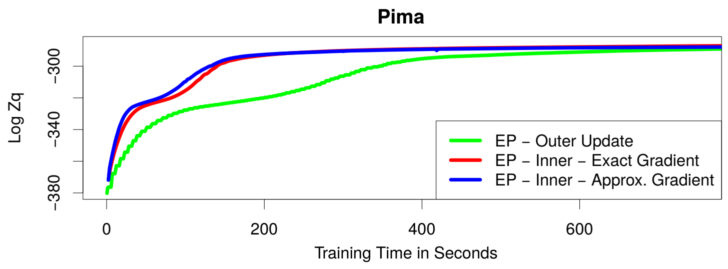
where $\boldsymbol{\eta}$ and $\boldsymbol{\eta}_{\text{prior}}$ are the expected sufficient statistics under q and the prior $p(\overline{\mathbf{f}}|\overline{\mathbf{X}})$, respectively. See the supplementary material. Using (4) the model hyper-parameters can be estimated by maximizing $\log Z_q$ via gradient ascent. Typically, one alternates between running EP until convergence for fixed hyper-parameters, and updating the hyper-parameters using a gradient step. After training, q gives a predictive distribution for the label $y_\star$ of a new instance $\mathbf{x}_\star$:

$$p(y_\star|\mathbf{y},\overline{\mathbf{X}}) \approx \int p(y_\star|f_\star)p(f_\star|\overline{\mathbf{f}})q(\overline{\mathbf{f}})d\overline{\mathbf{f}}df_\star\,. \quad (5)$$

Because several simplifications occur when computing the derivatives with respect to the inducing points (Snelson, 2007), the total training time is $\mathcal{O}(nm^2)$.

2.3 Scalable Expectation Propagation

Current EP implementations re-run EP (re-using the previous solution) after each single gradient ascent update of the hyper-parameters since (4) is only valid if EP has converged. Such a scheme is very inefficient at the beginning, when the estimate of the model hyper-parameters is often very poor, and EP may require several iterations to converge. To circumvent this, we propose to update the approximate factors $\tilde{\phi}_i$ and the model hyper-parameters ξ_j at the same time. That is, after a parallel update of all the approximate factors, we update the hyper-parameters using gradient



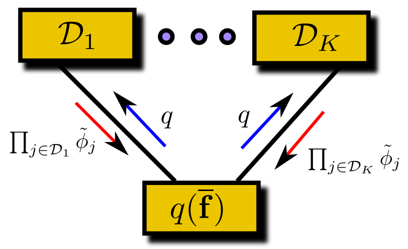

$\mathcal{D}_1$
$\mathcal{D}_K$
q
q
$\prod_{j\in\mathcal{D}_1}\tilde{\phi}_j$
$\prod_{j\in\mathcal{D}_K}\tilde{\phi}_j$
$q(\bar{\mathbf{f}})$

Figure 1: (left) Value of $\log Z_q$ obtained when updating the hyper-parameters after EP has converged (outer) and just after each update of the approximate factors $\tilde{\phi}_i$ (inner), when using the exact gradient, and the approximation (4), which assumes matched moments between $Z_i^{-1}\phi_i q^{\setminus i}$ and q. $m = 300$. (right) Distribution of the EP updates across K computational nodes storing a subset $\mathcal{D}_1, \ldots, \mathcal{D}_K$ of the data. Best seen in color.

ascent assuming that each $\tilde{\phi}_i$ is fixed. However, because EP has not converged, the moments of $Z_i^{-1}\phi_i q^{\setminus i}$ and q need not match. Thus, extra terms must be added to (4) to get the exact gradient. In spite of this, our experiments show that the extra terms are very small and can be ignored. In practice, we use (4) for the inner update of the hyper-parameters. Figure 1 (left) shows that the approach described successfully maximizes $\log Z_q$ on the *Pima* dataset from the UCI repository (A. Asuncion, 2007). Because we do not wait for convergence in EP, the method is significantly faster. The idea of why the approach described works in practice is as follows. The EP update of each $\tilde{\phi}_i$ can be seen as a (natural) gradient ascent step on $\log Z_q$ when all $\tilde{\phi}_j$, with $j \neq i$, remain fixed (Heskes and Zoeter, 2002). Furthermore, those updates are very effective for finding a stationary point of $\log Z_q$ with respect to the parameters of each $\tilde{\phi}_i$ since EP typically converges. Thus, it is natural that an inner update of the hyper-parameters when all $\tilde{\phi}_i$ remain fixed is an effective method for finding a maximum of $\log Z_q$.

Distributed training: The method described is suitable for distributed computation using the ideas in (Gelman et al., 2014; Xu et al., 2014). In particular, the training data can be split in K subsets $\mathcal{D}_1, \ldots, \mathcal{D}_K$ which are sent to K computational nodes. A master node stores the posterior approximation q, which is sent to each computational node. Then, node k updates each $\tilde{\phi}_j$ with $j \in \mathcal{D}_k$ and returns $\prod_{j\in\mathcal{D}_k}\tilde{\phi}_j$ to the master node. After each node has done this, the master node updates q using $p(\bar{\mathbf{f}}|\bar{\mathbf{X}})$ and the messages received. Because the gradient of the hyper-parameters (4) involves a sum over the data instances, its computation can also be distributed among the K computational nodes. Thus, the total training cost of the EP method can be reduced by a factor of K to $\mathcal{O}(nm^2/K)$. Figure (1) (right) illustrates the scheme described.

Training using minibatches: The method described is also suitable for stochastic optimization. For this, the data are split in minibatches $\mathcal{M}_k$ of size $s \ll n$, with n the total number of instances. For each minibatch $\mathcal{M}_k$, each $\tilde{\phi}_j$ with $j \in \mathcal{M}_k$ is refined, and q is updated afterwards. Next, the model hyper-parameters are updated via gradient ascent using a stochastic approximation of (4). Namely,

$$\frac{\partial Z_q}{\partial \xi_j} \approx \left(\boldsymbol{\eta}^{\mathrm{T}} - \boldsymbol{\eta}_{\mathrm{prior}}^{\mathrm{T}}\right)\frac{\partial \boldsymbol{\theta}_{\mathrm{prior}}}{\partial \xi_j} + C\sum_{l\in\mathcal{M}_k}\frac{\partial \log Z_l}{\partial \xi_j}\,, \quad (6)$$

where $C = n/|\mathcal{M}_k|$. After the update, q is reconstructed. With this training scheme we update the model hyper-parameters more frequently, and the training cost scales like $\mathcal{O}(m^3)$. The memory resources scale, however, like $\mathcal{O}(nm)$, since we store $m+1$ parameters per each approximate factor $\tilde{\phi}_i$.

3 RELATED WORK

A related method for binary classification with GPs uses scalable variational inference (SVI) (Hensman et al., 2015). Since $p(\mathbf{y}|\bar{\mathbf{f}}) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\bar{\mathbf{f}})d\mathbf{f}$, we obtain the bound $\log p(\mathbf{y}|\bar{\mathbf{f}}) \geq \mathbb{E}_{p(\mathbf{f}|\bar{\mathbf{f}})}[\log p(\mathbf{y}|\mathbf{f})]$ by taking the logarithm and using Jensen's inequality. Let $q(\bar{\mathbf{f}})$ be a Gaussian approximation of $p(\bar{\mathbf{f}}|\mathbf{y})$. Then,

$$\begin{aligned}\log p(\mathbf{y}) &= \log \int q(\bar{\mathbf{f}})p(\mathbf{y}|\bar{\mathbf{f}})p(\bar{\mathbf{f}}|\bar{\mathbf{X}})/q(\bar{\mathbf{f}})d\bar{\mathbf{f}} \\ &\geq \mathbb{E}_{q(\bar{\mathbf{f}})}[\log p(\mathbf{y}|\bar{\mathbf{f}})] - \mathrm{KL}[q(\bar{\mathbf{f}})||p(\bar{\mathbf{f}}|\bar{\mathbf{X}})]\,, \quad (7)\end{aligned}$$

by Jensen's inequality, with $\mathrm{KL}[\cdot||\cdot]$ the Kullback Leibler divergence. Using the first bound in (7) gives

$$\begin{aligned}\log p(\mathbf{y}) &\geq \mathbb{E}_{q(\bar{\mathbf{f}})}[\mathbb{E}_{p(\mathbf{f}|\bar{\mathbf{f}})}[\log p(\mathbf{y}|\mathbf{f})]] - \mathrm{KL}[q(\bar{\mathbf{f}})||p(\bar{\mathbf{f}}|\bar{\mathbf{X}})] \\ &\geq \mathbb{E}_{q(\mathbf{f})}[\log p(\mathbf{y}|\mathbf{f})] - \mathrm{KL}[q(\bar{\mathbf{f}})||p(\bar{\mathbf{f}}|\bar{\mathbf{X}})] \\ &\geq \textstyle\sum_{i=1}^{n}\mathbb{E}_{q(f_i)}[\log p(y_i|f_i)] - \\ &\quad \mathrm{KL}[q(\bar{\mathbf{f}})||p(\bar{\mathbf{f}}|\bar{\mathbf{X}})]\,, \quad (8)\end{aligned}$$

where $q(\mathbf{f}) = \int p(\mathbf{f}|\bar{\mathbf{f}})q(\bar{\mathbf{f}})d\bar{\mathbf{f}}$ and $q(f_i)$ is the i-th marginal of $q(\mathbf{f})$. Let $q(\bar{\mathbf{f}}) = \mathcal{N}(\bar{\mathbf{u}}|\mathbf{m},\mathbf{S})$, with $\mathbf{m}$ and $\mathbf{S}$ variational parameters. Because $p(\mathbf{f}|\bar{\mathbf{f}}) = \mathcal{N}(\mathbf{f}|\mathbf{A}\bar{\mathbf{f}}, \mathbf{K}_{\mathbf{ff}} - \mathbf{A}\mathbf{K}_{\mathbf{f}\bar{\mathbf{f}}}^{\mathrm{T}})$, where $\mathbf{A} = \mathbf{K}_{\mathbf{f}\bar{\mathbf{f}}}\mathbf{K}_{\bar{\mathbf{f}}\bar{\mathbf{f}}}^{-1}$, and $\mathbf{K}_{\mathbf{f}\bar{\mathbf{f}}}$ is a matrix with the covariances between pairs of observed inputs and inducing points, $q(\mathbf{f}) =$

$\mathcal{N}(\mathbf{f}|\mathbf{Am}, \mathbf{K_{ff}} + \mathbf{A}(\mathbf{S} - \mathbf{K_{\overline{f}\overline{f}}})\mathbf{A}^{\mathrm{T}})$. $\mathbf{S}$ is encoded in practice as $\mathbf{LL}^{\mathrm{T}}$ and the lower bound (8) is maximized with respect to $\mathbf{m}$, $\mathbf{L}$, the inducing points $\overline{\mathbf{X}}$ and any hyper-parameter using either batch, stochastic or distributed optimization techniques. In the stochastic case, small minibatches are considered and the gradient of $\sum_{i=1}^{n} \mathbb{E}_{q(f_i)}[\log p(y_i|f_i)]$ in (8) is subsampled and scaled accordingly. In the distributed case, the gradient of the sum is computed in parallel. The computational cost of this method is $\mathcal{O}(nm^2)$, when trained in a batch setting, and $\mathcal{O}(m^3)$, when using minibatches and stochastic gradients. A practical disadvantage is, however, that $\mathbb{E}_{q(f_i)}[\log p(y_i|f_i)]$ has no analytic solution. These expectations and their gradients must be approximated using one-dimensional quadratures. By contrast, in the approach described in Section 2.2 all the required computations have a closed form solution.

Another related method is the generalized FITC approximation (GFITC) of Naish-Guzman and Holden (2008), in which the values $\overline{\mathbf{f}}$ associated to the inducing points are marginalized, as indicated in Section 2.1. This generates the FITC prior $p_{\mathrm{FITC}}(\mathbf{f}|\overline{\mathbf{X}})$ which leads to a computational cost that is $\mathcal{O}(nm^2)$. Expectation propagation (EP) is also used in such a model to approximate $p(\mathbf{f}|\mathbf{y}, \overline{\mathbf{X}})$. In particular, EP replaces with a Gaussian factor each likelihood factor of the form $p(y_i|f_i) = \Phi(y_i f_i)$. A limitation is, however, that the estimate of the marginal likelihood $p(\mathbf{y}|\overline{\mathbf{X}})$ provided by EP in this model does not contain a sum over the data instances. Thus, GFITC does not allow for stochastic nor distributed optimization of the model hyper-parameters. This is not the case of the estimate described in (3) for the proposed approach.

A similar model to the one described in Section 2.2 has been proposed by Qi et al. (2010). These authors also use EP for approximate inference. However, the moments of the process are matched at $\mathbf{f}$, instead of at $\overline{\mathbf{f}}$. This leads to equivalent, but more complicated EP updates. Furthermore, the inducing points $\overline{\mathbf{X}}$ (which are considered to be noisy) are not learned from the observed data, but kept fixed. This is a serious limitation since finding good locations for the inducing points is strictly required to get good prediction results. The main advantage with respect to GFITC is that training can be done in an online fashion. This is also the case of the approach proposed in Section 2.2.

Other methods have been proposed in the literature (Henao and Winther, 2012), but they do not allow for stochastic optimization of the hyper-parameters.

4 EXPERIMENTS

We compare the proposed method for Gaussian process classification based on a scalable EP algorithm (SEP) with (i) the generalized FITC approximation (GFITC) of Naish-Guzman and Holden (2008) and (ii) the scalable variational inference (SVI) method of Hensman et al. (2015). SEP and GFITC use fast parallel EP updates that require only matrix multiplications and avoid loops over the data. All methods are implemented in R. The code is found in the supplementary material. Finally, to guarantee a fair comparison, all hyper-parameters (including the inducing points) are set to the same initial values in each method.

4.1 Datasets from the UCI Repository

A first set of experiments evaluates the predictive performance of SEP, GFITC and SVI on 7 datasets extracted from the UCI repository (A. Asuncion, 2007). We use 90% of the data for training and 10% for testing and report averages over 20 repetitions of the experiments. All methods are trained using batch algorithms for 250 iterations. Both GFITC and SVI use L-BFGS-B. In SEP we use gradient ascent with an adaptive learning rate (see the supplementary material). We report for each method the average negative test log-likelihood. A squared exponential covariance function with automatic relevance determination, an amplitude parameter and an additive noise parameter is employed. The initial inducing points are chosen at random from the training data. All other hyper-parameters are initialized to the same values. A different number of inducing points m are considered. Namely, 15%, 25% and 50% of the total number of instances. The results of these experiments are displayed in Table 1. The best method is shown in bold face. The proposed approach, *i.e.*, SEP, obtains similar results to GFITC and SVI and sometimes is the best method. Table 1 also reports the average training time in seconds. The fastest method is SEP followed by SVI. GFITC is the slowest method since it runs EP until convergence to evaluate then the gradient of the approximate marginal likelihood. By contrast, SEP updates at the same time the approximate factors and the model hyper-parameters, which is more efficient.

4.2 Analysis of Inducing Point Learning

Using the setting of the previous section, we focus on the two dimensional *Banana* dataset and analyze the location of the inducing points inferred by each method. We initialize the inducing points at random from the training set and progressively increase their number m from 4 to 256. Figure 2 shows the results obtained. For small values of m, *i.e.*, $m = 4$ or $m = 8$, SEP and SVI provide very similar locations for the inducing points. The estimates provided by GFITC for $m = 4$ are different as a consequence of arriving to a sub-optimal local maximum of the estimate of

Table 1: Average negative test log likelihood for each method and average training time in seconds.

	$m = 15\%$			$m = 25\%$			$m = 50\%$		
Problem	**GFITC**	**SEP**	**SVI**	**GFITC**	**SEP**	**SVI**	**GFITC**	**SEP**	**SVI**
Australian	.68 ± .06	.69 ± .07	**.63 ± .05**	.68 ± .08	.67 ± .07	**.63 ± .05**	.67 ± .09	.64 ± .05	**.63 ± .05**
Breast	**.10 ± .05**	.11 ± .05	.10 ± .05	.11 ± .06	.11 ± .05	**.10 ± .05**	.11 ± .05	.11 ± .05	**.10 ± .05**
Crabs	.07 ± .07	**.06 ± .06**	.07 ± .06	**.06 ± .07**	.06 ± .06	.07 ± .07	**.06 ± .07**	.06 ± .06	.09 ± .06
Heart	.43 ± .12	.40 ± .13	**.39 ± .11**	.42 ± .12	.41 ± .12	**.40 ± .11**	.42 ± .13	.41 ± .11	**.40 ± .10**
Ionosphere	.30 ± .22	.26 ± .19	**.26 ± .14**	.29 ± .23	**.27 ± .20**	.27 ± .18	.30 ± .24	.27 ± .19	**.26 ± .16**
Pima	.54 ± .08	.52 ± .07	**.49 ± .05**	.53 ± .07	.51 ± .06	**.50 ± .05**	.53 ± .07	.50 ± .05	**.49 ± .05**
Sonar	.35 ± .13	**.33 ± .10**	.40 ± .17	.35 ± .12	**.32 ± .10**	.40 ± .19	.35 ± .13	**.29 ± .09**	.35 ± .16
Avg. Time	59 ± 4	17 ± 1	40 ± 2	133 ± 6	37 ± 2	65 ± 3	494 ± 29	130 ± 5	195 ± 10

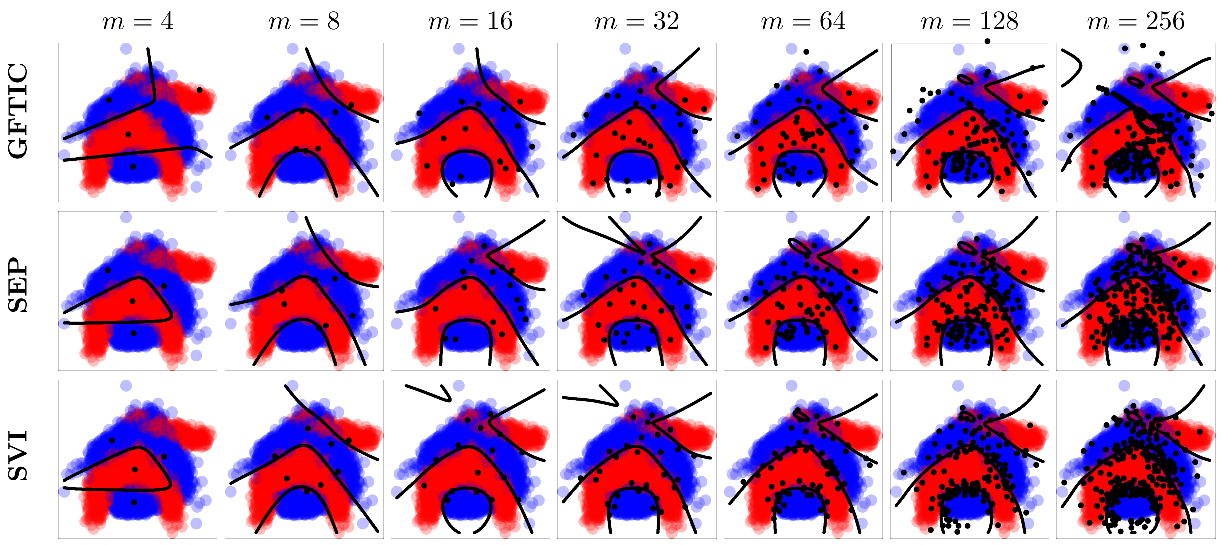


Figure 2: Effect of increasing the inducing points for SEP, SVI and GFITC. Each column shows a different number of inducing points from $m = 4$ to $m = 256$. Blue and red points represent training data from the banana dataset. Inducing points are black dots and decision boundaries are black lines. Best seen in color.

the marginal likelihood. If the initial inducing points are chosen differently, GFITC gives the same solution as SEP and SVI. We also observe that SEP and SVI quickly provide (*i.e.*, for $m = 16$) estimates of the decision boundaries that look similar to the ones obtained with larger values of m (*i.e.*, $m = 256$). These results confirm that SEP is able to find good locations for the inducing points. Finally, we note that SVI seems to prefer placing the inducing points near the decision boundaries. This is not the case of GFITC nor SEP.

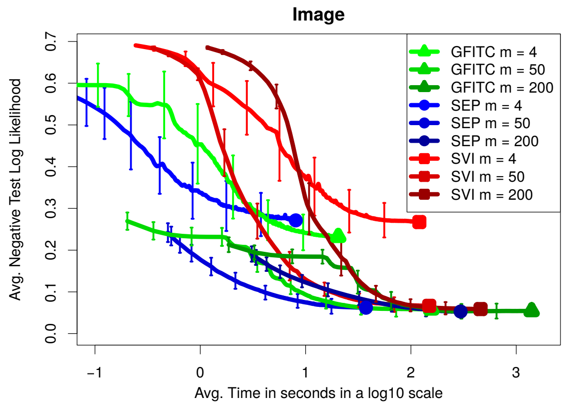
4.3 Performance as a Function of Time

We profile each method to show the prediction performance on the *Image* dataset a function of the training time, for different numbers of inducing points m, *i.e.*, 4, 50 and 200. Training is done as in Section 4.1. We report averages over 100 realizations of the experiments. The results are displayed in Figure 3 (left). We observe that the proposed method SEP provides the best performance at the lowest computational time. It is faster than GFITC because in SEP we update the posterior approximation q and the hyper-parameters at the same time. By contrast, GFITC waits until EP has converged to update the hyper-parameters. SVI also takes more time than SEP to obtain a similar level of performance. This method requires a few extra matrix multiplications with cost $\mathcal{O}(nm^2)$ to evaluate the gradient of the hyper-parameters. Furthermore, the initial performance of SVI is significantly worse than the one of GFITC and SEP, for each value of m. More precisely, after one iteration, both GFITC and SEP have updated each approximate factor, leading to a good estimate of q, the posterior approximation, which is then used for hyper-parameter estimation. By contrast, SVI updates q using gradient ascent which requires several iterations to get a good estimate of this distribution. Thus, at the beginning, SVI is most probably updating the model hyper-parameters when q is still a very bad approximation of the posterior.

4.4 Training in a Distributed Fashion

We illustrate the utility of SEP and SVI to carry out distributed training[1]. We consider the MNIST dataset and use 60,000 instances for training and 10,000 instances for testing. The number of inducing points m is set equal to 200 in both SEP and SVI. The task is

[1]GFITC does not allow for this type of training.



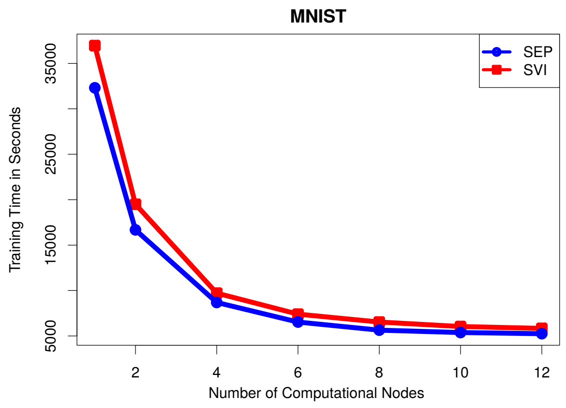


Figure 3: (left) Prediction performance of each method on the *Image* dataset as a function of the training time measured in seconds (in a $\log_{10}$ scale). Different numbers of inducing points are considered, *i.e.*, $m = 4, 50, 200$. Best seen in color. (right) Average training time in seconds for SEP and SVI on the MNIST dataset as a function of the number of computational nodes employed in the process of distributed training. Best seen in color.

to discriminate odd from even digits, which is a highly non-linear problem. We distribute the data across an increasing number of nodes from 1 to 12 using a machine with 12 CPUs. The process of distributed training is simulated via the R package *doMC*, which allows to execute for loops in parallel with a few lines of code. In SVI we parallelize the computation of the terms (corresponding either to both the lower bound or the gradient) that depend on the training instances. In SEP we parallelize the updates of the approximate factors and the computation of the estimate of the gradient of hyper-parameters. Figure 3 (right) shows the training time in seconds of each method as a function of the number of nodes (CPUs) considered. We observe that using more than 1 nodes significantly reduces the training time of SEP and SVI, until 6 nodes are reached. After this, no improvements are observed, probably because process synchronization becomes a bottle-neck. The test error and the avg. neg. test log likelihood of SVI is 2.2% and 0.0655, respectively, while for SEP they are 2.7% and 0.0694. These values are the same independently of the number of nodes considered. The R code to reproduce these experiments is found in the supplementary material.

4.5 Training using Stochastic Gradients

We evaluate the performance of SEP and SVI on the MNIST dataset when the training process is implemented using minibatches of 200 instances. Each minibatch is used to update the posterior approximation q and to compute a stochastic approximation of the gradient of the hyper-parameters. Note that GFITC does not allow for this type of stochastic optimization. The learning rate employed for updating the hyper-parameters is computed using the Adadelta method in both SEP and SVI with $\rho = 0.9$ and $\epsilon = 10^{-5}$ (Zeiler, 2012). The number of inducing points is set equal to the minibatch size, *i.e.*, 200. We report the performance on the test set (prediction error and average negative test log likelihood) as a function of the training time. We compare the results of these methods (stochastic) with the variants of SEP and SVI that use all data instances for the estimation of the gradient (batch). Figure 4 (top) shows the results obtained. We observe that stochastic methods (either SEP or SVI) obtain good results even before batch methods have completed a single hyper-parameter update. Furthermore, the performance of the stochastic variants of SEP and SVI in terms of the test error or the avg. neg. log likelihood is very similar. These are 1.8% and 0.0528 for SEP, and 2.0% and 0.0654 for SVI, respectively. They are better than the results reported by Hensman et al. (2015). The R code to reproduce these experiments is found in the supplementary material.

Our last experiments consider information about all commercial flights in the USA from January 2008 to April 2008. The task is the same as in (Hensman et al., 2015). Namely, to predict whether a flight was delayed or not based on 8 attributes: age of the aircraft, distance that needs to be covered, airtime, departure time, arrival time, day of the week, day of the month and month. After removing instances with missing values 2,127,068 instances remain. From these, 10,000 are used for testing and the rest are used for training the stochastic variants of SVI and SEP (batch methods are infeasible in this dataset). We use a minibatch of size 200 and set $m = 200$ and compare results with a logistic regression classifier. The results obtained are displayed in Figure 4 (bottom). We observe that both SEP and SVI outperform the linear model, which shows that the problem is non-linear. Eventually SEP and SVI provide similar performance results, probably because in this large dataset the posterior distribution is very close to be Gaussian. However, SEP improves

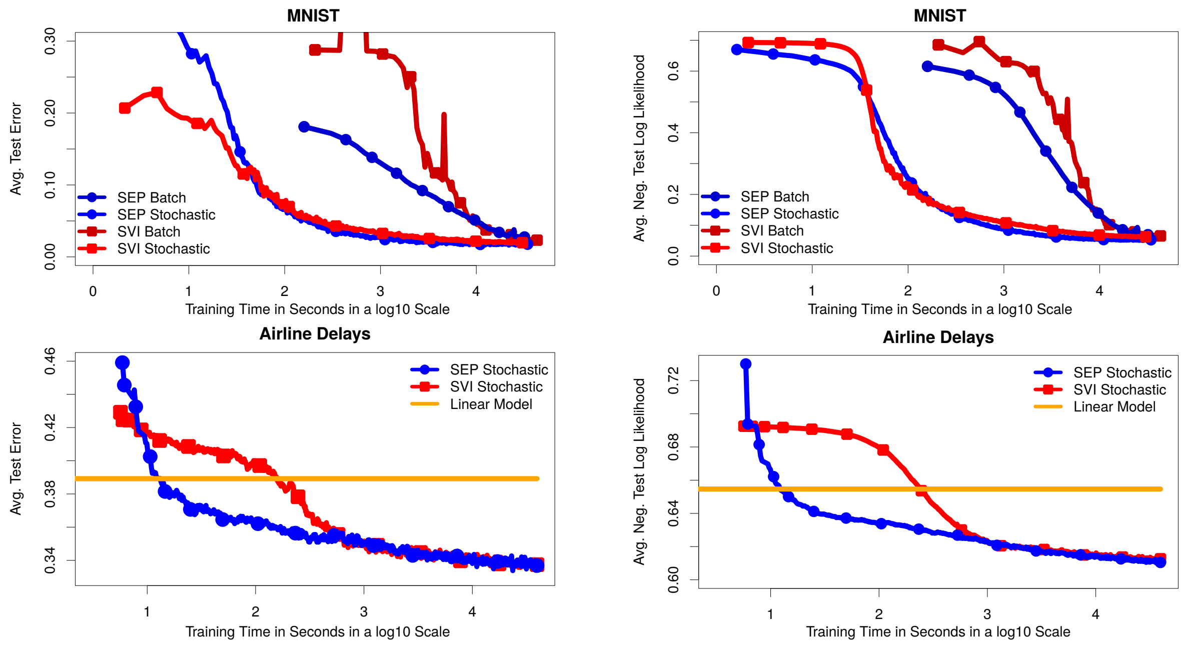


Figure 4: (top) Average test error and and average negative test log likelihood for SEP and SVI as a function of training time on the MNIST dataset. Stochastic variants use a minibatch size equal to 200 to approximate the gradients. Batch variants use all data instances for the evaluation of the gradient. (bottom) Results for the Airline delays dataset where batch methods are not feasible. The performance of a linear logistic regression classifier is also shown. Best seen in color.

results more quickly. This supports that, at the beginning, the EP updates of SEP are more effective for estimating q than the gradient updates of SVI. We believe that SVI is most likely updating the hyper-parameters using a poor estimate of q, at the beginning.

5 CONCLUSIONS

We have shown that expectation propagation (EP) can be used for Gaussian process classification in large scale problems. This scenario was previously considered infeasible for this approximate inference method. The scalable variant of EP proposed in this paper (SEP) allows for (i) training in a distributed fashion in which the data are sent to different computational nodes and (ii) for updating the posterior approximation and the model hyper-parameters at the same time using minibatches and a stochastic approximation of the gradient of the estimate of the marginal likelihood.

The proposed method, SEP, has been compared with other approaches from the literature such as the generalized FITC approximation (GFITC) and a scalable variational inference (SVI) method. The results obtained show that SEP outperforms GFITC in large datasets in which that method is infeasible. Furthermore, SEP is competitive with SVI in large datasets and provides similar and sometimes even better performance at an equivalent computational cost. If small minibatches are used for training, the cost of SEP is $\mathcal{O}(m^3)$, where m is the number of inducing points.

A disadvantage of SEP is that the memory requirements are $\mathcal{O}(nm)$, where n is the number of instances. Nevertheless, we believe this cost may be reduced to $\mathcal{O}(m^2)$ by using the approach of Li et al. (2015), in which the likelihood of the model is approximated using a single Gaussian factor. In our experiments SEP seems to provide better results than SVI at the early iterations. An explanation for this is a better estimation of the posterior approximation q when using the EP updates, which are free of any learning rate, than when using the gradient steps employed by SVI.

Finally, the proposed EP algorithm is not restricted to Gaussian process classification. It may be used for efficient approximate inference in other models. The good results obtained also show that EP algorithms should always be implemented as we suggest. That is, q and all the model hyper-parameters should be updated at the same time. Currently, all major Gaussian process toolboxes do not follow this approach (they run EP until convergence after each single update of the hyper-parameters), and they could benefit from it.

Acknowledgments

DHL acknowledges the use of the facilities of Centro de Computacin Científica (CCC) at Universidad Autónoma de Madrid. JMHL is supported by the Rafael del Pino Foundation. DHL is supported by Plan Nacional I+D+i, Grants TIN2013-42351-P and TIN2015-70308-REDT, and by CAM, Grant S2013/ICE-2845 CASI-CAM-CM.

References

A. Asuncion, D. N. (2007). UCI machine learning repository. Online available at: http://www.ics.uci.edu/∼mlearn/MLRepository.html.

Gelman, A., Vehtari, A., Jylänki, P., Robert, C., Chopin, N., and Cunningham, J. (2014). Expectation propagation as a way of life. *ArXiv e-prints*. arXiv:1412.4869.

Henao, R. and Winther, O. (2012). Predictive active set selection methods for Gaussian processes. *Neurocomputing*, 80:10–18.

Hensman, J., Matthews, A., and Ghahramani, Z. (2015). Scalable variational Gaussian process classification. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*.

Hernández-Lobato, D., Hernández-Lobato, J. M., and Dupont, P. (2011). Robust multi-class Gaussian process classification. In *Advances in Neural Information Processing Systems 24*.

Heskes, T. and Zoeter, O. (2002). Expectation propagation for approximate inference in dynamic Bayesian networks. In *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence*.

Hoffman, M., Blei, D., Wang, C., and Paisley, J. (2013). Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347.

Kuss, M. and Rasmussen, C. (2005). Assessing approximate inference for binary Gaussian process classification. *Journal of Machine Learning Research*, 6:1679–1704.

Li, Y., Hernández-Lobato, J. M., and Turner, R. E. (2015). Stochastic expectation propagation. In *Advances in Neural Information Processing Systems 28*, pages 2323–2331.

Minka, T. (2001). Expectation propagation for approximate Bayesian inference. In *Annual Conference on Uncertainty in Artificial Intelligence*, pages 362–36.

Naish-Guzman, A. and Holden, S. (2008). The generalized FITC approximation. In *Advances in Neural Information Processing Systems 20*.

Nickisch, H. and Rasmussen, C. (2008). Approximations for binary Gaussian process classification. *Journal of Machine Learning Research*, 9:2035–2078.

Qi, Y., Abdel-Gawad, A., and Minka, T. (2010). Sparse-posterior Gaussian processes for general likelihoods. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*.

Quiñonero Candela, J. and Rasmussen, C. (2005). A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, pages 1935–1959.

Rasmussen, C. and Williams, C. (2006). *Gaussian Processes for Machine Learning.* The MIT Press.

Rasmussen, C. E. and Nickisch, H. (2010). Gaussian processes for machine learning (GPML) toolbox. *Journal of Machine Learning Research*, 11:3011–3015.

Seeger, M. (2006). Expectation propagation for exponential families. Technical report, Department of EECS, University of California, Berkeley.

Snelson, E. (2007). *Flexible and efficient Gaussian process models for machine learning.* PhD thesis, Gatsby Computational Neuroscience Unit, University College London.

Snelson, E. and Ghahramani, Z. (2006). Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems 18*.

The GPy authors (2015). GPy: A Gaussian process framework in python. `http://github.com/SheffieldML/GPy`.

Titsias, M. (2009). Variational Learning of Inducing Variables in Sparse Gaussian Processes. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Van Gerven, M., Cseke, B., Oostenveld, R., and Heskes, T. (2009). Bayesian source localization with the multivariate Laplace prior. In *Advances in Neural Information Processing Systems 22*, pages 1901–1909.

Vanhatalo, J., Riihimäki, J., Hartikainen, J., Jylänki, P., Tolvanen, V., and Vehtari, A. (2013). GPstuff: Bayesian modeling with Gaussian processes. *Journal of Machine Learning Research*, 14:1175–1179.

Xu, M., Lakshminarayanan, B., Teh, Y. W., Zhu, J., and Zhang, B. (2014). Distributed bayesian posterior sampling via moment sharing. In *Advances in Neural Information Processing Systems 27*, pages 3356–3364. Curran Associates, Inc.

Zeiler, M. (2012). ADADELTA: An adaptive learning rate method. *ArXiv e-prints.* arXiv:1212.5701.