Alpha Divergence Minimization in Multi-Class Gaussian Process Classification

Carlos Villacampa-Calvo[a,*], Daniel Hernández-Lobato[a]

[a] *Computer Science Department, Escuela Politécnica Superior, Universidad Autónoma de Madrid, C/ Francisco Tomás y Valiente, 11, Madrid 28049, Spain*

Abstract

This paper analyzes the minimization of α-divergences in the context of multi-class Gaussian process classification. For this task, several methods are explored, including memory and computationally efficient variants of the Power Expectation Propagation algorithm, which allow for efficient training using stochastic gradients and mini-batches. When these methods are used for training, very large datasets (several millions of instances) can be considered. The proposed methods are also very general as they can interpolate between other popular approaches for approximate inference based on Expectation Propagation (EP) ($\alpha \to 1$) and Variational Bayes (VB) ($\alpha \to 0$) simply by varying the α parameter. An exhaustive empirical evaluation analyzes the generalization properties of each of the proposed methods for different values of the α parameter. The results obtained show that one can do better than EP and VB by considering intermediate values of α.

Keywords: Gaussian Processes, Expectation Propagation, α-divergences, Approximate Inference, Variational Inference

*Corresponding author

Email address: `carlos.villacampa@uam.es` (Carlos Villacampa-Calvo)

1. Introduction

Gaussian Processes (GPs) are non-parametric models that can be used to address machine learning problems, including multi-class classification [1]. In these models, the expressiveness grows with the training set size N. Furthermore, they are probabilistic models in which prior knowledge can be easily specified, and they readily provide a predictive distribution which accounts for prediction uncertainty. In spite of these advantages, using Gaussian process in practice is difficult because often the likelihood is not Gaussian. Therefore, exact inference in these models is usually intractable and approximate methods need to be employed. A challenging example is multi-class classification because in this case there is one latent function (GP) per class, and the likelihood factors are more complicated than, for example, in binary classification models. An extra difficulty is that standard approaches for multi-class GP classification require, at least, the inversion of one $N \times N$ matrix per class. This is an expensive operation that limits the applicability of these models to large problems. Notwithstanding, there are several methods that have been proposed for multi-class GP classification [2, 3, 4, 5, 6]. Most of them, however, do not scale well with the size of the training set.

The use of sparse approximations allows to scale-up GPs. These techniques introduce $M \ll N$ inducing points, whose location is learnt during the training process. These points lead to an approximate prior with a low-rank covariance structure [7], reducing the training cost to $\mathcal{O}(NM^2)$. This improved cost has been pushed forward by Hensman et al. [8, 9], which employs a variational Bayes (VB) approximation combined with stochastic optimization techniques that allows to address datasets with millions of instances. Recent work in the literature also combines stochastic optimization techniques with alternative methods for approximate inference based on expectation propagation (EP) [10, 11]. The results obtained indicate that EP and VB have similar training costs, but EP may provide better predictive distributions in terms of the test log-likelihood.

While VB minimizes the Kullback-Leibler (KL) divergence between the approximate and the target distribution, EP minimizes (approximately) the KL-divergence in the reversed way. Recently, Bui et al. [12] suggested a framework for binary GP classification that, by means of the minimization of α-divergences with Power Expectation Propagation (PEP) [13], unifies previous approaches based on VB and EP. The α-divergence generalizes the KL-divergence and different values of the α parameter interpolate ($\alpha \to 0$ and $\alpha = 1$) between the two versions of the KL-divergence described above [14]. Importantly, Bui et al. [12] show that, in the case of binary classification, one can do better than VB and EP by considering an intermediate version of the two KL-divergences.

Here, we extend the minimization of α-divergences for approximate inference of Bui et al. [12] to address multi-class GP classification problems. For this, we describe a multi-class extension of the PEP algorithm for binary GP classification. This extension is not trivial due to the more complicated likelihood factors that appear in the multi-class setting. Furthermore, instead of considering a single latent function, in the multi-class case we have one latent function per class. Besides this, we address here some of the drawbacks of standard PEP, which include the difficulty of using standard optimization techniques and a high memory consumption. More precisely, standard PEP combines gradient-based updates of the model hyper-parameters with closed-form updates to refine the approximate likelihood factors. These approximate factors have to be stored in memory, which is memory expensive. The variants of PEP considered are based on using ideas from approximate EP [15, 16] and from the approximate minimization of α-divergences in the context of Bayesian neural networks [17]. The results obtained in our experiments show that the (approximate) minimization of an intermediate divergence between the ones considered by VB and EP, *i.e.*, setting $\alpha = 0.5$, may work better in terms of the prediction error and the test log-likelihood.

2. Multi-Class Gaussian Processes

Consider a dataset of N labelled examples in the form of a matrix $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^\text{T}$ and a vector of labels $\mathbf{y} = (y_1, \ldots, y_N)^\text{T}$, where $y_i \in \{1 \ldots C\}$ with $C > 2$ the total number of classes. The goal is to predict the label of an unseen instance $\mathbf{x}_*$. In multi-class Gaussian process classification it is usual to use the softmax function. However, it is not the only alternative. Here, we will follow [3] assume that the label y_i of $\mathbf{x}_i$ is generated by the rule $y_i = \arg\max_k f^k(\mathbf{x}_i)$, where each $f^k(\cdot)$ is a latent function associated to a class $k \in \{1 \ldots C\}$. Based on this, the likelihood is a product of N factors such as:

$$p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^{N} p(y_i|\mathbf{f}_i) = \prod_{i=1}^{N} \prod_{k \neq y_i} \Theta\left(f^{y_i}(\mathbf{x}_i) - f^k(\mathbf{x}_i)\right) , \tag{1}$$

where $\Theta(\cdot)$ is the Heaviside function and we have defined $\mathbf{f}^k = (f^k(\mathbf{x}_1), \cdots, f^k(\mathbf{x}_N))^\text{T} \in \mathbb{R}^N$, $\mathbf{f}_i = (f^1(\mathbf{x}_i), \cdots, f^C(\mathbf{x}_i))^\text{T} \in \mathbb{R}^C$ and

$\mathbf{f} = (\mathbf{f}^1, \ldots, \mathbf{f}^C) \in \mathbb{R}^{N \times C}$. The likelihood in (1) can be made more robust to possible labelling errors by adding a parameter ϵ which represents the probability of choosing at random y_i from the set of labels [18]. Then, each factors is:

$$p(y_i|\mathbf{f}_i) = (1 - \epsilon) \prod_{k \neq y_i} \Theta\left(f^{y_i}(\mathbf{x}_i) - f^k(\mathbf{x}_i)\right) + \frac{\epsilon}{C}. \tag{2}$$

We assume a GP prior for each $f^k(\cdot)$ [1]. Particularly, $p(f^k) \sim \mathcal{GP}(0, c(\cdot, \cdot; \xi^k))$ where $c(\cdot, \cdot; \xi^k)$ is a covariance function with hyper-parameters ξ^k. Moreover, we assume these priors to be independent. That is, $p(\mathbf{f}) = \prod_{k=1}^{c} p(\mathbf{f}^k)$, where every $p(\mathbf{f}^k)$ is a multivariate Gaussian distribution. In this model, one can easily include Gaussian additive noise around each latent function. In that case, the labeling rule described is equivalent to the Gumbel-max trick (which leads to a soft-max function), but adding independent Gaussian noise instead of Gumbel noise [19]. The task of interest is to compute a posterior distribution for $\mathbf{f}$ using Bayes rule: $p(\mathbf{f}|\mathbf{y}) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f})/p(\mathbf{y})$. We can then maximize the marginal likelihood $p(\mathbf{y})$ to find good values for the hyper-parameters ξ^k. Nevertheless, as the likelihood factors in (1) and (2) are not Gaussian, we will be unable to compute analytically $p(\mathbf{f}|\mathbf{y})$ and approximate inference will be needed: the Laplace approximation [2], EP [3] or VB [20]. These methods result in a cost of $\mathcal{O}(CN^3)$, where N is the number of instances and C the number of classes, assuming independent GPs per each class (this is the hypothesis made in the rest of paper).

2.1. Sparse Gaussian Processes

To speed up calculations, a typical approach is to use sparse approximations. These approximations rely on introducing a different set of points of size $M \ll N$ called inducing points $\bar{\mathbf{X}}^k = (\bar{\mathbf{x}}_1^k, \ldots, \bar{\mathbf{x}}_M^k)^{\mathrm{T}}$ for each class k, with associated latent values $\bar{\mathbf{f}}^k = (f^k(\bar{\mathbf{x}}_1^k), \ldots, f^k(\bar{\mathbf{x}}_M^k))^{\mathrm{T}}$ [21]. Now, by setting a GP prior on the latent functions associated with the inducing points we can obtain an approximate prior for $\mathbf{f}^k$ as $p(\mathbf{f}^k) = \int p(\mathbf{f}^k|\bar{\mathbf{f}}^k)p(\bar{\mathbf{f}}^k|\bar{\mathbf{X}}^k)d\bar{\mathbf{f}}^k \approx \int [\prod_{i=1}^N p(f^k(\mathbf{x}_i)|\bar{\mathbf{f}}^k)]p(\bar{\mathbf{f}}^k|\bar{\mathbf{X}}^k)d\bar{\mathbf{f}}^k = p_{\mathrm{FITC}}(\mathbf{f}^k|\bar{\mathbf{X}}^k)$, where we have assumed that $p(\bar{\mathbf{f}}) = \prod_{k=1}^C p(\bar{\mathbf{f}}^k|\bar{\mathbf{X}}^k)$ and that the conditional distribution $p(\mathbf{f}^k|\bar{\mathbf{f}}^k)$ factorizes like $\prod_{i=1}^N p(f^k(\mathbf{x}_i)|\bar{\mathbf{f}}^k)$. In other words, marginalizing over the latent values associated with the inducing points $\bar{\mathbf{f}}^k$ will effectively result in an approximate covariance function for the prior on the latent values $\mathbf{f}^k$ [22]. This approximation is known as the Fully Independent Training Conditional (FITC) [7] and gives an approximate inference cost of $\mathcal{O}(NM^2)$.

2.2. Scalable Gaussian Processes: EP

A method for approximate inference in multi-class GP classification is Expectation Propagation (EP) [23]. In EP the latent variables, $\mathbf{f}$, of the process at the training points $\mathbf{X}$ are marginalized out. The task of interest is to approximate the posterior of the process values at the inducing points $\bar{\mathbf{f}} = (\bar{\mathbf{f}}^1, \ldots, \bar{\mathbf{f}}^K)^{\mathrm{T}}$: $p(\bar{\mathbf{f}}|\mathbf{y}) \propto \prod_{i=1}^N \phi_i(\bar{\mathbf{f}})p(\bar{\mathbf{f}})$, where $\phi_i(\bar{\mathbf{f}})$ is a likelihood factor defined as $\phi_i(\bar{\mathbf{f}}) = \int p(y_i|\mathbf{f}_i)p(\mathbf{f}_i|\bar{\mathbf{f}})d\mathbf{f}_i$ and $p(\bar{\mathbf{f}}) = \prod_{k=1}^C p(\bar{\mathbf{f}}^k)$ is the prior distribution over the inducing values. In this last expression $p(\mathbf{f}_i|\bar{\mathbf{f}})$ is a conditional Gaussian distribution that factorizes across classes, *i.e.*, $p(\mathbf{f}_i|\bar{\mathbf{f}}) = \prod_{k=1}^C p(f^k(\mathbf{x}_i)|\bar{\mathbf{f}}^k)$. EP approximates each non-Gaussian factor of the likelihood ϕ_i with a Gaussian factor $\tilde{\phi}_i$ [10]. More precisely, it refines at each iteration a factor $\tilde{\phi}_i$ of the approximate posterior $q(\bar{\mathbf{f}}) \propto \prod_{i=1}^N \tilde{\phi}_i p(\bar{\mathbf{f}})$ by computing the cavity distribution $q^{\setminus i} \propto q/\tilde{\phi}_i$ and then minimizing locally the Kullback-Leibler divergence between $Z_i^{-1}\phi_i q^{\setminus i}$ and q with respect to q, *i.e*, $\text{KL}[Z_i^{-1}\phi_i q^{\setminus i} \parallel q]$ where Z_i is the normalization constant of $\phi_i q^{\setminus i}$. The updated factor is simply $\tilde{\phi}_i^{\text{new}} = Z_i q^{\text{new}}/q^{\setminus i}$. The KL-divergence minimization is done using the derivatives of $\log Z_i$ w.r.t. the parameters of $q^{\setminus i}$ [24] and Z_i can be computed using a one-dimensional quadrature. Note that q is Gaussian because the prior and each $\tilde{\phi}_i$ are Gaussian. The approximation to the marginal likelihood $p(\mathbf{y})$, denoted Z_q, is simply the normalization constant of $\prod_{i=1}^N \tilde{\phi}_i(\bar{\mathbf{f}})p(\bar{\mathbf{f}})$. The gradient of $\log Z_q$ w.r.t. a hyper-parameter ξ_j^k of the k-th covariance function can be easily obtained since the parameters of each $\tilde{\phi}_i$ can be considered fixed after running EP [24].

Recent work in the literature shows that it is possible to scale to large datasets the previous EP approach [11, 10]. One only has to jointly update the approximate factors $\tilde{\phi}_i$ and the model hyper-parameters $\boldsymbol{\xi}^k$. Furthermore, because $\log Z_q$ contains a sum across the data points, stochastic optimization techniques can be used to update the model hyper-parameters. This allows to scale to very large datasets with millions of instances.

A limitation of EP is that the parameters of each ϕ_i have to be stored in memory. A further approximation to EP called Stochastic Expectation Propagation (SEP) [15] assumes that all the approximate factors are tied and only keeps in memory the product of all of them instead of their individual parameters. This reduces the memory cost to $\mathcal{O}(CM^2)$.

Interestingly, the previous derivation of the EP algorithm for approximate inference in multi-class GPC is equivalent to the one that is obtained when one approximates the posterior distribution of $\mathbf{f}$ and $\bar{\mathbf{f}}$, *i.e.*, $p(\mathbf{f}, \bar{\mathbf{f}}|\mathbf{y}) \propto \prod_{i=1}^N p(\mathbf{y}_i|\mathbf{f}_i)p(\mathbf{f}_i|\bar{\mathbf{f}})p(\bar{\mathbf{f}})$, under the constraint that the approximate distribution is $q(\mathbf{f}, \bar{\mathbf{f}}) \propto p(\mathbf{f}|\bar{\mathbf{f}}) \prod_{i=1} \tilde{\phi}_i p(\bar{\mathbf{f}})$, where $p(\mathbf{f}|\bar{\mathbf{f}}) = \prod_{k=1}^C p(\mathbf{f}^k|\bar{\mathbf{f}}^k)$.

That is, each likelihood factor $p(y_i|\mathbf{f}_i)$ has been approximated by the corresponding factor $\tilde{\phi}_i$ which depends on $\overline{\mathbf{f}}$. Specifically, the conditional distribution $p(\mathbf{f}|\overline{\mathbf{f}})$ in q is fixed and we can only update the part of q that depends on the inducing values $\overline{\mathbf{f}}$. In this we case, there is no need to use the FITC approximation. See [12] for further details and the specific equivalence in the regression case.

2.3. Scalable Gaussian Processes: VB

Another approach for approximate inference is Variational Bayes (VB) [25, 26, 27, 9]. In this section we will follow the derivation of the lower bound in [9]. VB uses the same likelihood function as EP. The approximate distribution q is the same as the one considered at the end of the previous section. Namely, $q(\mathbf{f},\overline{\mathbf{f}}) = p(\mathbf{f}|\overline{\mathbf{f}})q(\overline{\mathbf{f}})$, where $q(\overline{\mathbf{f}})$ is Gaussian and $p(\mathbf{f}|\overline{\mathbf{f}})$ is fixed. The distribution q is found by minimizing the KL-divergence between q and the exact posterior $p(\mathbf{f},\overline{\mathbf{f}}|\mathbf{y})$. It is possible to show that this minimization is equivalent to the maximization of a lower bound on the log-marginal likelihood $\log p(\mathbf{y})$. This lower bound is obtained by first applying Jensen's inequality to obtain a lower bound to the log conditional $\log p(\mathbf{y}|\overline{\mathbf{f}})$:

$$\log p(\mathbf{y}|\bar{\mathbf{f}}) = \log \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\bar{\mathbf{f}})d\mathbf{f} \geq \mathbb{E}_{p(\mathbf{f}|\bar{\mathbf{f}})}[\log p(\mathbf{y}|\mathbf{f})]\,. \tag{3}$$

Then, a lower bound to the log-marginal likelihood is derived in the same way:

$$\begin{aligned}\log p(\mathbf{y}) &= \log \int q(\bar{\mathbf{f}})p(\mathbf{y}|\bar{\mathbf{f}})p(\bar{\mathbf{f}})/q(\bar{\mathbf{f}})d\bar{\mathbf{f}} \\ &\geq \mathbb{E}_{q(\bar{\mathbf{f}})}[\log p(\mathbf{y}|\bar{\mathbf{f}})] - \mathrm{KL}[q(\bar{\mathbf{f}}) \parallel p(\bar{\mathbf{f}})]\,,\end{aligned} \tag{4}$$

where $q(\bar{\mathbf{f}})$ is approximate Gaussian distribution. By substituting (3) in (4) we obtain the final lower bound:

$$\begin{aligned}
\log p(\mathbf{y}) &\geq \mathbb{E}_{q(\bar{\mathbf{f}})}[\log p(\mathbf{y}|\bar{\mathbf{f}})] - \mathrm{KL}[q(\bar{\mathbf{f}}) \parallel p(\bar{\mathbf{f}})] \\
&\geq \mathbb{E}_{q(\bar{\mathbf{f}})}[\mathbb{E}_{p(\mathbf{f}|\bar{\mathbf{f}})}[\log p(\mathbf{y}|\mathbf{f})]] - \mathrm{KL}[q(\bar{\mathbf{f}}) \parallel p(\bar{\mathbf{f}})] \\
&\geq \mathbb{E}_{q(\mathbf{f})}[\log p(\mathbf{y}|\mathbf{f})] - \mathrm{KL}[q(\bar{\mathbf{f}}) \parallel p(\bar{\mathbf{f}})] \\
&\geq \sum_{i=1}^{N} \mathbb{E}_{q(\mathbf{f}_i)}[\log p(y_i|\mathbf{f}_i)] - \mathrm{KL}[q(\bar{\mathbf{f}}) \parallel p(\bar{\mathbf{f}})]\,, \qquad (5)
\end{aligned}$$

where $q(\mathbf{f}) = \int p(\mathbf{f}|\bar{\mathbf{f}})q(\bar{\mathbf{f}})d\bar{\mathbf{f}}$ and each marginal over $\mathbf{f}_i = (f^1(\mathbf{x}_i), \ldots, f^C(\mathbf{x}_i))^{\mathrm{T}}$ is a product of C Gaussian conditional distributions with mean $\hat{m}_i^k$ and variance $\hat{s}_i^k$, for $k = 1, \ldots, C$. Namely, $q(\mathbf{f}_i) = \prod_{k=1}^{C} \mathcal{N}(f^k(\mathbf{x}_i)|\hat{m}_i^k, \hat{s}_i^k)$.

The lower bound contains a sum over the training examples, so stochastic optimization techniques can be used for its optimization. As in EP, one-dimensional quadratures must be used to approximate the required expectations in (5). Last, this formulation minimizes the global KL-divergence between the approximate distribution q and the posterior, and can be shown to be equivalent to minimizing $\mathrm{KL}[q \parallel Z_i\tilde{\phi}_i q^{\setminus i}]$ (the reversed divergence) in EP [14, 28].

3. Alpha-Divergence Minimization

We introduce the α-divergence [29, 30], a divergence measure that generalizes the KL divergence [14], as well as the different approaches proposed for its minimization in the context of Gaussian processes for multi-class classification. The α-divergence between two probability distributions p and q of a random variable $\boldsymbol{\theta}$ is:

$$D_\alpha[p||q] = \frac{1 - \int p(\boldsymbol{\theta})^\alpha q(\boldsymbol{\theta})^{1-\alpha} d\boldsymbol{\theta}}{\alpha(1-\alpha)}\,, \tag{6}$$

where $\alpha \in \mathbb{R} \setminus \{0, 1\}$.

The case $\alpha = 0.5$ is called the Hellinger distance the only member of the family of α-divergences that is symmetric in p and q [31]. More precisely, $D_{\frac{1}{2}}[p \parallel q] = 2\int_{\boldsymbol{\theta}}(\sqrt{p(\boldsymbol{\theta})} - \sqrt{q(\boldsymbol{\theta})})^2 d\boldsymbol{\theta}$. Furthermore, $D_0[p \parallel q] = \lim_{\alpha \to 0} D_\alpha[p \parallel q] = \mathrm{KL}[q \parallel p]$ is used in VB and $D_1[p \parallel q] = \lim_{\alpha \to 1} D_\alpha[p \parallel q] = \mathrm{KL}[p \parallel q]$ is used in EP, so the α-divergence minimization can easily interpolate between these two methods by changing the value of α.

3.1. Power Expectation Propagation (PEP)

Power Expectation Propagation (PEP) is an extension of EP that instead of minimizing the KL-divergence at each step, minimizes an α-divergence [13]. Importantly, this α-divergence minimization is done by simply minimizing the KL-divergence between some modified distribution and q. Specifically, when computing the cavity distribution $q^{\setminus \alpha i}$, the approximate factor $\tilde{\phi}_i$ to the power of α is removed. That is, $q^{\setminus \alpha i} \propto q/\tilde{\phi}_i^\alpha$. Next, the KL divergence between $Z_i^{-1}\phi_i^\alpha q^{\setminus \alpha i}$ and q, $\mathrm{KL}[Z_i^{-1}\phi_i^\alpha q^{\setminus \alpha i} \parallel q]$, is minimized with respect to q, where Z_i is the normalization constant of $\phi_i^\alpha q^{\setminus \alpha i}$. Note that the factor ϕ_i is raised to the power of α. The updated factor is simply $\tilde{\phi}_i^{\text{new}} = (Z_i q^{\text{new}}/q^{\setminus \alpha i})^{\frac{1}{\alpha}}$, since the exact factor ϕ_i had been raised to the power of α. Importantly, it is possible to show that this minimization is equivalent to minimizing $D_\alpha[Z_i \phi_i q^{\setminus i}||q]$ [14]. More precisely, let λ_q be the parameters of q. For a distribution p and q in the exponential family:

$$\nabla_{\lambda_q} D_\alpha[p||q] = \frac{Z_{\tilde{p}}}{\alpha}(\mathbb{E}_q[s(\boldsymbol{\theta})] - \mathbb{E}_{\tilde{p}}[s(\boldsymbol{\theta})]) \\ \propto \nabla_{\lambda_q}\mathrm{KL}[\tilde{p}||q]\,, \quad (7)$$

where $\tilde{p} \propto p^\alpha q^{1-\alpha}$ and $s(\boldsymbol{\theta})$ is the vector of sufficient statistics of q. At the minimum both gradients must be equal to zero and the moments of q and $\tilde{p}$ must match. If we let $p \propto \phi_i q^{\backslash i}$, as in EP, the corresponding distribution $\tilde{p} \propto \phi_i^\alpha q^{\backslash \alpha i}$, as in PEP. Therefore, at convergence, when the approximate factors do not change any more and (7) is equal to zero for each approximate factor, PEP minimizes the α-divergences between the tilted distributions defined as $\phi_i q^{\backslash i}$, $\forall i$, and q. Importantly, this local divergence minimization becomes a global divergence minimization (between the target posterior and q) only when $\alpha \to 0$ [14]. In all the other cases the global α-divergence minimization is approximate, but accurate as shown in [14].

The PEP algorithm consists in applying the following steps to every approximate factor $\tilde{\phi}_i$ and repeat them until it has converged:

Remove an approximate factor to the power of α from the posterior q to compute the cavity distribution $q^{\backslash \alpha i}$.

$$q^{\backslash \alpha i} = \frac{q}{(\tilde{\phi}_i)^\alpha}$$

Include the true factor ϕ_i to the power of α to compute the tilted distribution $\hat{p}$.

$$\hat{p} = (\phi_i)^\alpha q^{\backslash \alpha i}$$

Project onto the approximating family by matching moments.

$$q^{\mathrm{new}} = \arg\min_{q^*} \mathrm{KL}[\hat{p} \parallel q^*]$$

Update the approximate factor.

$$(\tilde{\phi}_i)^\alpha = \frac{q^{\text{new}}}{q^{\backslash \alpha i}}$$

To apply PEP to the model described in this manuscript, we consider the approximation described at the end of Section 2.2. Namely, $q(\mathbf{f}, \overline{\mathbf{f}}) \propto p(\mathbf{f}|\overline{\mathbf{f}}) \prod_{i=1}^N \tilde{\phi}_i p(\overline{\mathbf{f}})$, where $\tilde{\phi}_i$ are Gaussian factors depending only on $\overline{\mathbf{f}}$. The marginal likelihood approximation of PEP, Z_q, is the normalization constant of the previous expression:

$$\begin{aligned}\log Z_q &= g(\boldsymbol{\theta}_{\text{post}}) - g(\boldsymbol{\theta}_{\text{prior}}) + \frac{1}{\alpha}\sum_{i=1}^N \log \tilde{Z}_i \,, \\ \log \tilde{Z}_i &= \log \mathbb{E}_{q(\mathbf{f}_i)}[(\phi_i / \tilde{\phi}_i)^\alpha]\,,\end{aligned} \tag{8}$$

where each $\phi_i = p(y_i|\mathbf{f}_i)$; $g(\cdot)$ is the log-normalizer of a distribution in the exponential family of q; and $\boldsymbol{\theta}_{\text{post}}$ and $\boldsymbol{\theta}_{\text{prior}}$ are the natural parameters of $q(\overline{\mathbf{f}})$ and the prior, respectively. When $\alpha = 1$, (8) is equivalent to the EP approximation of the log-marginal likelihood. In the limit when $\alpha \to 0$, one can show that (8) tends to the lower bound optimized in VB. See [14] for further details. The expectation in (8) can be computed using one-dimensional quadrature methods. In particular, it is simply related to the probability that one Gaussian random variable is larger than several others (one per each other class label) [18].

As in EP, the gradient of $\log Z_q$ w.r.t. the model hyper-parameters (inducing points locations and parameters of the covariance functions) involves a sum across the data instances. Therefore, mini-batches and stochastic optimization methods can also be used here to maximize $\log Z_q$. This allows to scale-up to very large datasets. Usually, one has to wait until PEP has converged to compute the gradient and update the model hyper-parameters. Nevertheless, it is possible to follow the same approach as in [10] and jointly optimize the approximate factors and the model hyper-parameters.

3.2. Approximate Power EP (APEP)

A limitation of the PEP algorithm described in Section 3.1 is that it needs to keep in memory the parameters of all the approximate factors, which are optimized through PEP updates by moment matching. To overcome this, a first approximation to the PEP method considers stochastic expectation propagation [15], which ties all the approximate factors and only keeps in memory their product, *i.e*, $\tilde{\phi} = \prod_{i=1}^{N} \tilde{\phi}_i$. Note that this only affects the way of computing the cavity distribution $q^{\setminus \alpha i}$. Under this approximation $q^{\setminus \alpha i}$ is computed in an approximate way. Namely, $q^{\setminus \alpha i} \propto q / \tilde{\phi}^{\frac{\alpha}{N}}$, where N is the number of factors (and also data points). Besides this, we optimize the global factor $\tilde{\phi}$ by maximizing $\log Z_q$, the approximation to the log-marginal likelihood, w.r.t. the parameters of $\tilde{\phi}$, instead of using the PEP updates. This is supported by the fact that these updates also find a stationary point of this energy function [14]. This allows the use of standard optimization techniques to find the posterior approximation q, which is defined as $q \propto \tilde{\phi} p(\bar{\mathbf{f}})$.

3.3. Approximate Reparameterized PEP (ARPEP)

As another way to approximately optimize the PEP evidence or energy function, we consider the approach described by Li and Gal [17] for Bayesian neural networks. In that work it is described a reparameterization of the PEP energy function that is compatible with an approximate distribution q that need not belong to the exponential family, although we will also assume a Gaussian form here. In the large data limit, *i.e.*, when $\alpha/N \to 0$, the reparameterized objective is simply approximated by:

$$\log Z_q \approx \frac{1}{\alpha} \sum_{i=1}^{N} \log \mathbb{E}_{q(\mathbf{f}_i)}[p(y_i|\mathbf{f}_i)^\alpha] \\ - \mathrm{KL}[q(\bar{\mathbf{f}}) \parallel p(\bar{\mathbf{f}})] \,. \tag{9}$$

This objective is a combination of the terms appearing in the PEP estimate of the log-marginal likelihood (8) and the lower bound of VB (5). The KL-divergence term in (9) can be understood as a regularization term enforcing q to look similar to the prior. Because this objective also involves a sum across the data points, it can be efficiently optimized both w.r.t. the parameters of q and the model hyper-parameters using stochastic optimization techniques.

3.4. Refined Prior Approximate PEP (RPAPEP)

Some of the solutions obtained by VB can not be retrieved by the methods from Sections 3.2 and 3.1 due to the parameterization resulting in a non positive definite covariance matrix. A last method accounts for this. It is the same method as the one described in Section 3.2, but where we let q be an arbitrary Gaussian distribution and eliminate the assumption that q is proportional to a Gaussian times the prior. Namely, $q \propto \tilde{\phi} p(\bar{\mathbf{f}})$. This is precisely the same hypothesis made by VB or the method described in Section 3.3. For this, we simply let the prior be another extra factor to be refined by PEP. Thus, instead of considering N factors, one per each point, we will have $N+1$ factors, the extra factor corresponding to the prior. Under this setting, the PEP approximate log-marginal likelihood is:

$$\log Z_q = g(\boldsymbol{\theta}_{\text{post}}) + \frac{1}{\alpha} \sum_{i=1}^{N+1} \log \tilde{Z}_i \,. \tag{10}$$

In this method we also retrieve the approximate EP energy objective when $\alpha = 1$ and VB's lower bound as $\alpha \to 0$. Stochastic optimization is also possible.

3.5. Summary of Approximate Inference Methods

In the previous sections we have described four methods that deal with the minimization of α-divergences in Gaussian process models for multi-class classification. In this section we will summarize the characteristics of each of the methods with the aim of giving a better understanding of them. Table 3.5 shows, for each method, a summary of what we think are the most relevant features: the ability to use standard optimization techniques, the use of stochastic EP to make the method memory efficient and whether q has a free Gaussian form.

The first method, PEP, was first introduced in [13] as a generalization of the EP algorithm. In [12] they successfully apply the algorithm to minimize α-divergences with sparse GPs and perform extensive experiments in the regression and binary classification cases. This method is precisely the one in [12], but applied to multi-class classification problems. It follows the general PEP scheme where one has to alternate between updating the approximate factors by moment matching and gradient based optimization of the model hyper-parameters. However, in the original PEP formulation [13], one has to wait until convergence before updating the hyper-parameters and here, we follow [10] and jointly optimize the approximate factors and the hyper-parameters. This method is not memory efficient due to the need of keeping in memory all the approximate factors.

The second method, APEP, was described in [31] as a black-box method that can be applied to general probabilistic models. They propose a simplified objective by tying the approximate factors following [15], and directly optimize the posterior approximation q using the gradients of the simplified objective. This makes the method memory efficient and allows for standard optimization techniques instead of having to rely on the PEP update step to optimize the approximate factors. In this work, we use the APEP method in the specific case of GP models applied to multi-class classification problems.

The third method, ARPEP, was proposed in [17] for the specific case of Bayesian neural networks as an approximate way to minimize α-divergences. We have apply the same idea to the case of multi-class GP classification. This method is also memory efficient, can be used with standard optimization techniques. Also, in this case the posterior approximation q takes the form of a free Gaussian, meaning that it is no longer proportional to a Gaussian times the prior.

The last method, RPAPEP, has been considered because the parameterization used in both PEP and APEP prevents them to reach some of the solutions that can be obtained by VB. To account for this, this method is based on APEP but where we let q be a free Gaussian, instead of a Gaussian times the prior. This method is memory efficient as well and can be used with standard optimization techniques.

4. Related work

Other works in the literature have addressed the approximate minimization of α-divergences. In particular, [31] also approximate the Power EP objective with a simplified energy function by tying the approximate factors. More

	PEP	APEP	ARPEP	RPAPEP
Standard optimization		✓	✓	✓
Memory efficient		✓	✓	✓
Free Gaussian			✓	✓

Table 1: Summary of the proposed methods

precisely, the objective considered by these authors is:

$$\begin{aligned} E[\boldsymbol{\theta}_{\text{prior}}, \boldsymbol{\theta}] =& g(\boldsymbol{\theta}_{\text{prior}}) - g(\boldsymbol{\theta}_{\text{post}}) \\ & - \frac{1}{\alpha} \textstyle\sum_{i=1}^{N} \log \mathbb{E}_q \left[\left(\frac{p(\mathbf{y}_i|\mathbf{f}_i)}{\tilde{\phi}_i} \right)^{\alpha} \right] , \end{aligned} \qquad (11)$$

where $\boldsymbol{\theta}_{\text{prior}}$ and $\boldsymbol{\theta}_{\text{post}}$ are the natural parameters of the prior and the approximate posterior q respectively; $g(\boldsymbol{\theta}_{\text{prior}})$ and $g(\boldsymbol{\theta}_{\text{post}})$ are their log-normalizers; $\boldsymbol{\theta} = (\boldsymbol{\theta}_{\text{post}} - \boldsymbol{\theta}_{\text{prior}})/N$ are the parameters of the global approximate factor $\tilde{\phi}$; and $p(\mathbf{y}_i|\mathbf{f}_i)$ is the true likelihood factor. This method's objective will be equivalent to the one obtained by the approximation in Section 3.2, but critically, the expectations in (11), which may involve multiple random variables and may lack an analytic expression, are approximated via Monte Carlo. The result is black-box algorithm that can be used for approximate inference in arbitrary complicated models. In principle this method could also be used for approximate inference in the context of multi-class Gaussian process classification. Notwithstanding, in this particular case, the required expectations can be evaluated using one-dimensional quadrature methods, which is believed to be significantly more efficient than using a Monte Carlo estimate of the same quantity. Therefore, the approaches considered in the the present work are expected to be more efficient for approximately optimizing the PEP objective. Moreover, a Monte Carlo estimate of (11) will lead to a biased objective due to the non-linearity of the logarithm function.

The minimization of α-divergences for binary Gaussian process classification has also been explored by Bui et al. [12]. These authors also use power EP as a unifying framework to work with GPs and α-divergences. Moreover, Bui [32] also compares in the binary classification case PEP with APEP, finding that they give similar results. However, despite the extensive experimental results in [12, 32], the multi-class classification case was not specifically considered nor analyzed. Importantly, the extension from binary to multi-class problems is more challenging. Instead of having one single latent function, in the multi-class case there is one latent function per each class in the problem. Furthermore, the likelihood factors are also more complicated, and even lack an analytical expression. Our work complements that of Bui et al. [12] by providing a careful and exhaustive analysis of the multi-class case, which has been systematically overlooked by the literature on Gaussian processes. Besides this, we also consider alternative methods for approximately optimizing the PEP objective. These methods are memory efficient (the memory cost is independent of N) and can make use of standard optimization techniques (*i.e.*, they do not use PEP updates of the approximate factors).

Other methods for approximate inference in GPs with arbitrary likelihoods can also target the VB objective in Section 2.3 [33, 34, 35]. Instead of quadratures, they rely on a Monte Carlo approximation, which is expected to lead to higher variance in the gradients estimation and to affect negatively the optimization process.

5. Experiments

In this section, we intend to compare the performance of the different values of α when using α-divergences in the multi-class setting. As we can retrieve VI solution by making $\alpha \to 0$ and the EP solution with $\alpha = 1$ we are comparing the proposed methods to the state-of-the-art methods for scalable approximate inference with Gaussian processes. We compare the methods described in Section 3.1 to Section 3.4 (PEP, APEP, ARPEP and RPAPEP) in several experiments. The R code of each method is found in the supplementary material. All methods start with the same hyper-parameters (including the location of the inducing points), which are optimized by maximizing the estimate of the log-marginal likelihood. We use an ARD Gaussian kernel for each latent function [1], and optimize the amplitude and additive noise parameter. An implementation in R of all the compared methods is available at `http://arantxa.ii.uam.es/%7edhernan/alpha-mgpc/R-code_alpha_MGPC.zip`. In the Appendix C there is a comparison between the PEP method and some baseline methods, including label regression, Laplace approximation and MCMC based method, showing that the predictive distribution of PEP is good.

5.1. Performance on UCI Datasets

We compare the four methods, for different values of α, over 8 UCI-repository [36] multi-class problems. These problems are fairly small (see Appendix B for the datasets' details), but will show how each method performs on standard problems. Later on, we will consider larger datasets. Because the datasets are small we use here batch optimization. We use 90% of the data for training and 10% for testing, except for *Satellite* which is bigger (we choose 20% for training and 80% for test). In *Vowel* we consider only the points belonging to the 6 first classes. Finally, in *Waveform* (synthetic) we generate 1000 instances and split them in 30% for training and 70% for testing. All methods are trained for 500 iterations using l-BFGS, except for PEP, which uses gradient ascent with an adaptive learning rate (described in Appendix A.7). We consider three values for the number of inducing points M. Namely, 5%, 10% and 20% of the number of training data N. The values of α considered range from $\alpha \to 0$ to $\alpha = 1$ with steps of size 0.1. We report averages over 20 repetitions.

Figure 1 shows, for each method, the average rank for each value of α, in terms of the test log-likelihood. Average ranks are computed, for a fixed method, for each value of α, across dataset and splits and values of the number of inducing points M. Besides this, we analyze which method is better, given a particular value of α. For that, we compute the average rank of each method across datasets, splits and values of M, this time fixing α and varying the method instead of the other way around. This rank is shown using a color pattern with *red* meaning a higher average rank and *blue* a lower average rank. We observe that for PEP, the Hellinger value $\alpha = 0.5$, seems to give better performance in terms of the negative test log-likelihood. For APEP and ARPEP, a value of α between 0.6 and 0.8 gives better results. Finally, RPAPEP gives in general worst results than the other methods for almost all values of α. Results for ranks computed in terms of the test error are shown in Appendix D. They do not differ significantly from ones shown here.

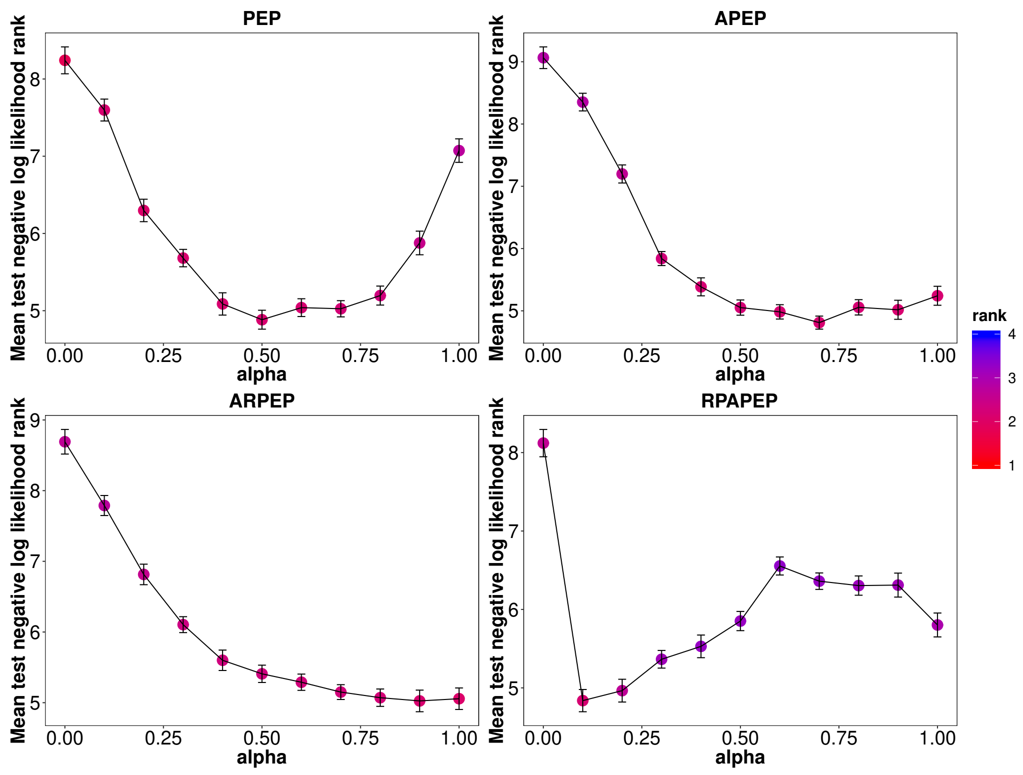

Figure 1: Avg. test neg. log-likelihood rank for each method and each value of α. The color of the points indicates the average rank a method compared with the others. Best seen in color.

5.2. Analysis of Inducing Points Location

We also analyze, for each method, the location of the inducing points for different values of α. We use a synthetic two-dimensional problem with three classes reproduced from [10]. We consider $1,000$ training points and a fixed number of inducing points $M = 128$. The initial location of the inducing points is chosen at random and it is the same for all the methods. In these experiments we we keep fixed the other hyper-parameters to their true value

[10]. PEP and APEP are trained using batch methods and ARPEP and RPAPEP are trained using stochastic methods to avoid sub-optimal solutions. In batch training we consider $2,000$ iterations and when using mini-batches we consider $2,000$ epochs. Additionally, we use ADAM for training (default settings) and 100 as the mini-batch size [37].

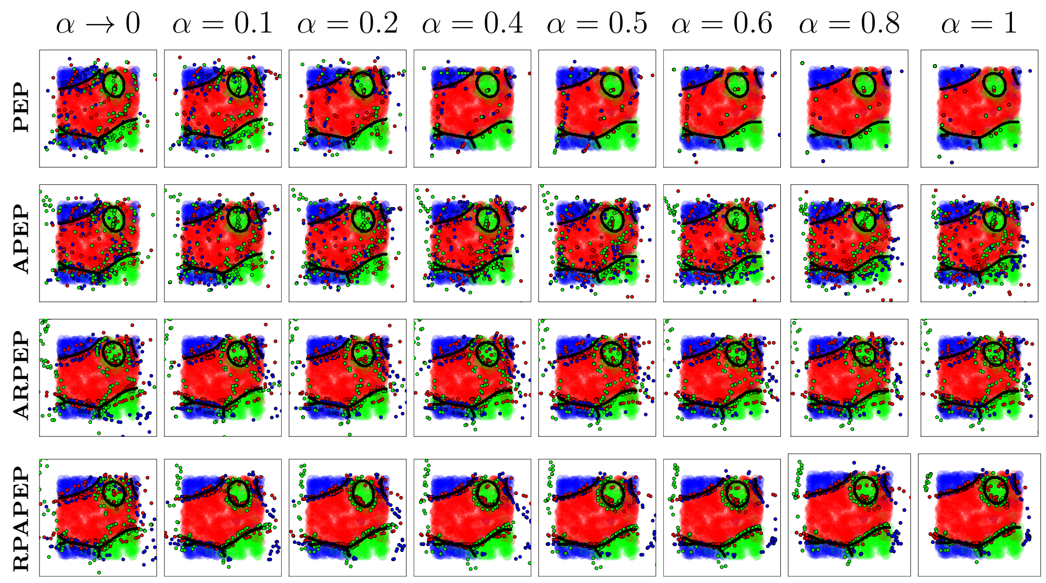

Figure 2: Decision boundaries and inducing points location for different values of α ($M = 128$).

Figure 2 shows the final location of the inducing points. Blue, red and green points are training data, black lines are the decision boundaries and black border points are the inducing points. We expect that for values of α near zero, the inducing points would tend to place near the decision boundaries, since this is the behavior observed in [8, 10, 12]. Indeed, this is the case of ARPEP and RPAPEP, probably because they are the two formulations in which q has a free Gaussian form. By contrast, in PEP and APEP this behavior is not observed. As we increase α, in PEP the inducing points overlap, which can be seen as an inducing point pruning technique, previously reported in [38, 10]. This behavior is not observed for the other methods, probably as a consequence of using either a different parameterization for q, or due to the approximation employed in APEP. Interestingly, for RPAPEP the inducing points tend to be even closer to the decision boundary as we approach $\alpha = 1$. Finally, in APEP and ARPEP α does not have a strong influence in the location of the inducing points.

5.3. Performance in Terms of Training Time

We compare the performance of each method as a function of the training time on the Satellite dataset. Training is done as in Section 5.1. We consider $M = 4$, 20 and 100. We also set $\alpha \to 0$ and $\alpha = 0.3, 0.5, 0.8, 1$. We report averages over 100 repetitions of the experiments. Figure 3 shows the average test negative log-likelihood for each method and each value of α. Similar plots for the test error are included in Appendix D. In general, when $\alpha \to 0$, and in the case of the method VB, we obtain the worst results. For PEP the best performance is obtained for $\alpha = 0.3$ and $\alpha = 0.5$. For the rest of the methods it seems that values between $\alpha = 0.8$ and $\alpha = 1$ tend to give good over-all results. RPAPEP seems to slightly over-fit the training data, which may explain the worse results of this method in the UCI datasets. ARPEP, RPAPEP give almost the same results as VB for $\alpha \to 0$, which is the expected behavior. In PEP and APEP this is not the case, probably because of the different parameterization of q, which is proportional to a Gaussian times the prior. Finally, PEP gives better results earlier, probably as a consequence of using PEP-updates to refine q, instead of gradient-based updates.

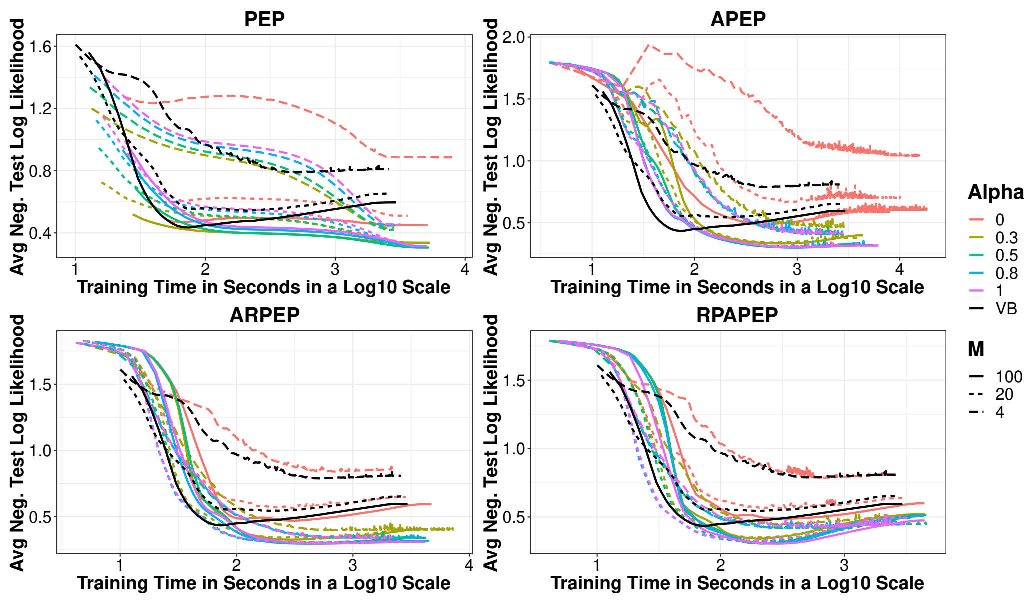

Figure 3: Neg. test log-likelihood on the Satellite dataset for different values of α and M. Best seen in color.

5.4. Performance on MNIST

When addressing very large datasets one can no longer rely on batch training, and mini-batches and stochastic gradients are required. A large problem is MNIST [39], with 60,000 instances for training and 10,000 for testing. We train the proposed methods on this dataset setting $M = 200$ and using a mini-batch size of 200. We consider several values for α from $\alpha \to 0$ to $\alpha = 1$ with a step-size of 0.1. Each method is trained using ADAM (except PEP which uses EP-updates to refine q) with the default parameters [37]. We include the results for Variational Bayes (VB) for reference. Figure 4 shows the test negative log-likelihood for each method and each value of α. The same plots but for the test error are included in Appendix D. All the methods seem to give similar results, but APEP reaches the optimal solution first. Importantly, in each method the lower the value of α the faster the convergence.

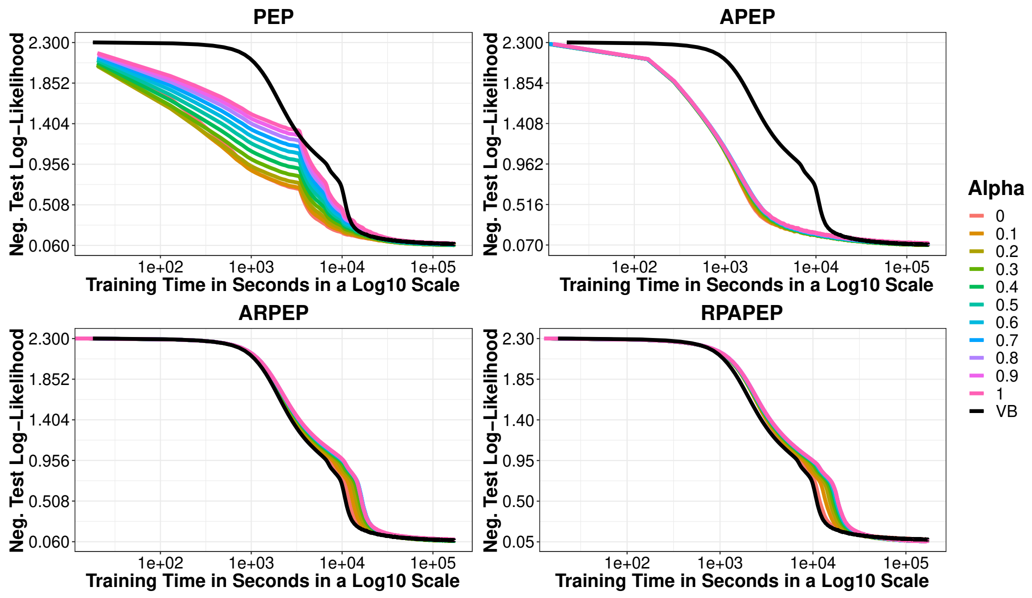

Figure 4: Negative test log-likelihood for each method on the MNIST dataset for each α. Best seen in color.

5.5. *Performance on Airline Delays*

We consider a dataset with information about the flights within the USA between 01/2008 and 04/2008 [1] . It has three classes: Flight on time, with more than 5 minutes of delay, or arrived 5 minutes before scheduled time. We consider 8 attributes: age of the aircraft, distance covered, airtime, departure time, arrival time, day of the week, day of the month and month. After

[1]`http://stat-computing.org/dataexpo/2009`

removing the data with missing values, $2,127,068$ instances remain, from which $10,000$ are used for testing and the rest for training. We evaluate each method using the same setting as on the MNIST dataset. We also include the results obtained by VB for reference.

Figure 5 shows the negative test log-likelihood of each method as a function of time. The results are similar in terms of the test error (see Appendix D). Regarding the negative test log-likelihood, as α approaches 0, worse results are obtained. This had previously been observed in [10], and is believed to be a consequence of the particular objective that is optimized by VB. As α increases, the approximation to the log-marginal likelihood of PEP resembles more the EP objective, which is closer to the test-log likelihood $\log \mathbb{E}_{q(\mathbf{f}_i)}[p(y_i|\mathbf{f}_i)]$. This explains the better results of $\alpha = 1$. Here, $\alpha = 0.5$ also provides good results.

5.6. Active Learning: Waveform

As a way of measuring the quality of the predictive distribution we have conducted a last experiment on the waveform dataset. We consider an active learning approach where we will iteratively add a new data point to the training set. For that, we will need an initial training set, a test set to evaluate the performance and a validation set from which to select the new data points. To choose which point to select next from the validation set, we will use the predictive distribution of the proposed methods, by selecting the point in which the entropy is highest and hence adopting an explorative approach. We compare this selection mechanism versus selecting the next point at random from the validation set. We start with 100 points for training, 500 for test and 400 for validation, and we will add 100 new points to the

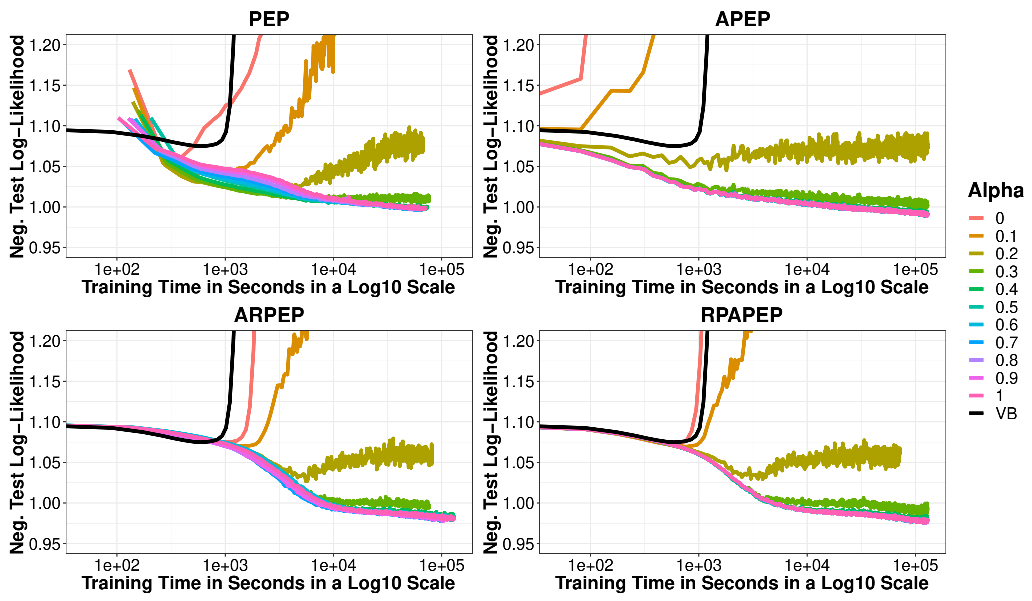

Figure 5: Neg. test log-likelihoods on the Airline Delays dataset for each α. Best seen in color.

training set. All methods are trained using l-BFGS the first time for 250 iterations and then re-trained each time we add a new point for 25 more iterations, reusing the solution obtained so far. For the PEP algorithm, when adding a new point to the training set one must also add a new approximate factor. This made the retraining process start in a bad solution leading to bad results. In order to overcome this problem, we have combined both the PEP updates of the approximate factors and l-BFGS, by alternating between updating the approximate factors and optimizing the model hyper-parameters with l-BFGS. In this case, as we are updating the factors several times at each l-BFGS iteration in an internal loop the training process is more costly, therefore we have reduced the initial training from 250 to 50 iterations and the

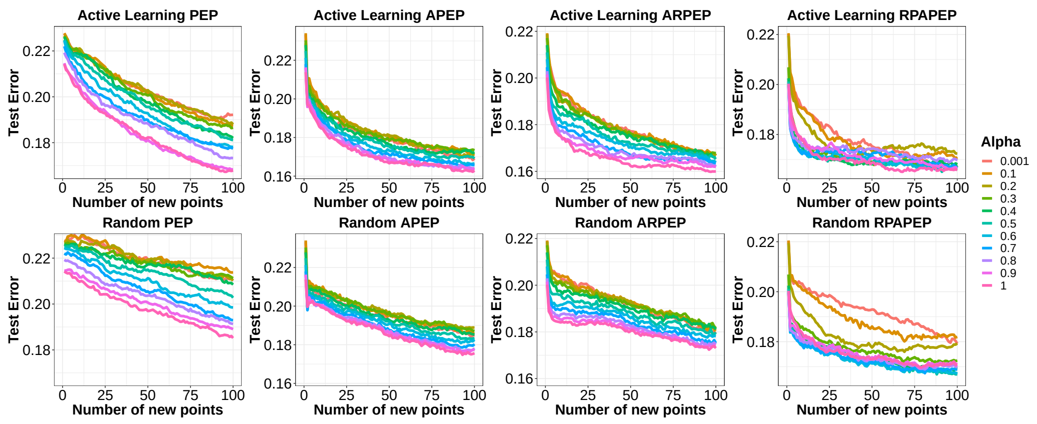

Figure 6: Test error on the Waveform dataset as a function of the number of added points to the training set, selected using an active learning approach (top) and selected at random (botton). Best seen in color.

retraining from 25 to 5 iterations. We report averages over 100 repetitions.

Figure 6 shows, for each value of α, the classification error in the test set as a function of the number of new added points. In the top row, each new point has been added by means of the active learning approach, selecting the point in which the entropy is highest. In the bottom row, each new point has been selected at random. We observe that the error is lower for values close to $\alpha = 1$, both for the active learning approach and random selection. Also, the test error is always lower for the active learning approach than for random selection for all the methods, showing the utility of the predictive distribution for this type of problems.

In Figure 7 it is shown, for each value of α, the reduction in the test error w.r.t. the initial error. For PEP, the reduction in the error is higher when we choose higher values of α, but for the other methods values of $\alpha \to 0$ give

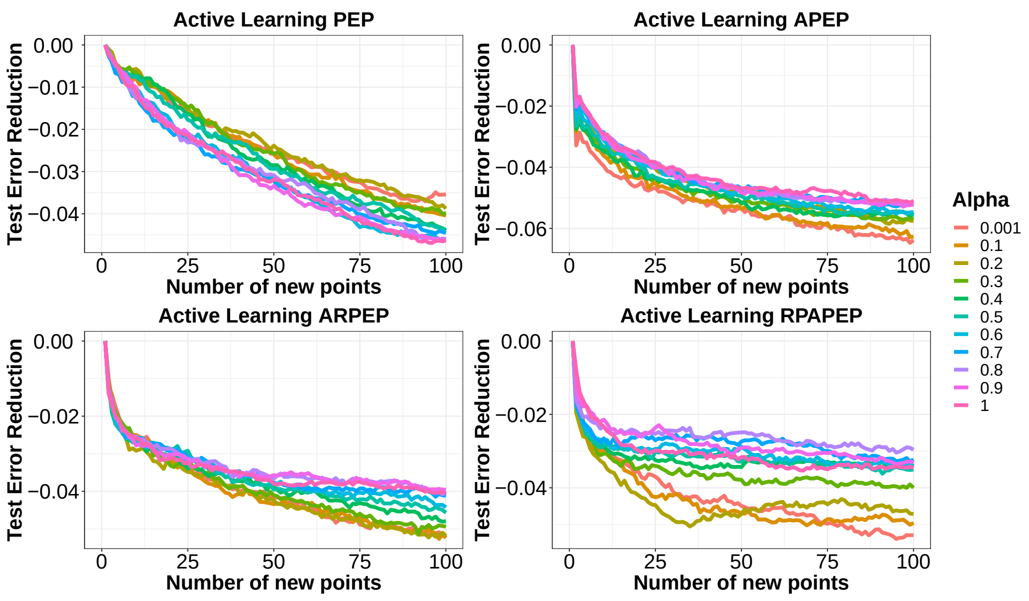

Figure 7: Test error reduction on the Waveform dataset as a function of the number of added points to the training set, selected using an active learning approach. Best seen in color.

better test error reduction. This is because the initial test error as $\alpha \to 0$ is worse for all the methods but, at the end of the training process, all the values of α give similar values for the test error, although slightly better for values near $\alpha = 1$. In the case of PEP, the difference in the test error between lower and higher values of α is bigger at the end, resulting in a better reduction for higher values of α.

6. Conclusions

The optimization of α-divergences allows to interpolate between approximate inference methods that are closer to VB when $\alpha \to 0$ or EP as $\alpha \to 1$.

Previous work in the literature had already considered the optimization of these divergences for approximate inference [12, 31, 17]. In this work, we have analyzed its specific minimization in the case of multi-class classification using GPs. We have compared four approximate methods for this: PEP, APEP, ARPEP and RPAPEP. These approximations are memory efficient (except PEP) and can be combined with batch training methods, as well as with stochastic training methods. When using mini-batches and stochastic techniques, the training cost is $\mathcal{O}(CM^3)$. We have done several experiments comparing the proposed approximations for different values of α.

While none of the proposed methods seems to be superior to the others (except RPAPEP, which performs slightly worse), there are some points that one should keep in mind when using them in practice. First of all, PEP is not memory efficient as it needs to keep in memory all the approximate factors. This clearly a drawback with respect to the other methods, especially when working with big datasets. Also, these factors are optimized through PEP updates, which makes the implementation slightly more complicated, as we cannot rely on standard optimization techniques like in the other methods. APEP and ARPEP give very similar results and can be used indistinctly. A difference between those two methods is that the ARPEP objective is more similar to the one optimized in VB, including the fact that the posterior q takes the form of a free Gaussian, which is why it exhibits some of the properties previously reported for VB (*e.g.*, the inducing points tend to place near the decision boundaries).

The results obtained show that intermediate values of α, *e.g.*, $\alpha = 0.5$, can provide in general better results than standard approximate inference methods based on VB or EP in some of the problems investigated. This agrees with previous results for the case of regression or binary classification problems, as indicated in [12].

Acknowledgements

We acknowledge the use of the facilities of Centro de Computación Científica (CCC) at Universidad Autónoma de Madrid and support from the Spanish *Ministerio de Economía, Industria y Competitividad*, grants TIN2016-76406-P and TEC2016-81900-REDT.

Appendix A. Details for implementing PEP

Appendix A.1. Introduction

In this document we detail all the steps needed to implement the PEP algorithm described in the main manuscript. 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 PEP approximation to the marginal likelihood and its gradients, as well as those of the proposed approximations in the main manuscript. Finally, we include some additional experimental results.

Appendix A.2. Reconstruction of the posterior approximation

In this section we show how to obtain the posterior distribution by multiplying the approximate factors $\tilde{\phi}_i(\overline{\mathbf{f}})$ and the prior $p(\overline{\mathbf{f}})$. Each factor ϕ_i will be replaced by PEP for an approximate Gaussian factor $\tilde{\phi}_i$ of the form:

$$\tilde{\phi}_i(\overline{\mathbf{f}}) = \tilde{Z}_i \prod_{k=1}^{C} \exp\left\{ -\frac{1}{2}(\overline{\mathbf{f}}^k)^{\mathrm{T}} \tilde{\mathbf{V}}_{i,k} \overline{\mathbf{f}}^k + (\overline{\mathbf{f}}^k)^{\mathrm{T}} \tilde{\mathbf{m}}_{i,k} \right\} , \tag{A.1}$$

where $\tilde{\mathbf{V}}_i, k$ and $\tilde{\mathbf{m}}_{i,k}$ have the following especial form (see Appendix A.4 for the detailed derivation):

$$\tilde{\mathbf{V}}_{i,k} = C^1_{i,k} \boldsymbol{\upsilon}^k_i (\boldsymbol{\upsilon}^k_i)^{\mathrm{T}} , \tag{A.2}$$

$$\tilde{\mathbf{m}}_{i,k} = C^2_{i,k} \boldsymbol{\upsilon}^k_i , \tag{A.3}$$

where we have defined $\boldsymbol{\upsilon}^k_i = (\mathbf{k}^k_{\mathbf{x}_i \overline{\mathbf{X}}^k})^{\mathrm{T}} (\mathbf{K}^k_{\overline{\mathbf{X}}^k \overline{\mathbf{X}}^k})^{-1}$ and $\mathbf{K}^k_{\overline{\mathbf{X}}^k \overline{\mathbf{X}}^k}$ is a $M \times M$ matrix with the cross covariances between $\overline{\mathbf{f}}^k$ and $C^1_{i,k}$ and $C^2_{i,k}$ are parameters found by PEP. We also know from the main manuscript that the prior has the following form:

$$p(\overline{\mathbf{f}}) = \prod_{k=1}^{C} p(\overline{\mathbf{f}}^k | \overline{\mathbf{X}}^k) = \prod_{k=1}^{C} \mathcal{N}(\overline{\mathbf{f}}^k | \mathbf{0}, \mathbf{K}^k_{\overline{\mathbf{X}}^k \overline{\mathbf{X}}^k}) , \tag{A.4}$$

So the posterior approximation will have the following form

$$q(\overline{\mathbf{f}}) = \frac{1}{Z_q} \left[\prod_{i=1}^{N} \tilde{\phi}_i \right] \prod_{k=1}^{C} p(\overline{\mathbf{f}}^k | \overline{\mathbf{X}}^k) . \tag{A.5}$$

Given that all the factors are Gaussian, a distribution that is closed under product and division, $q(\overline{\mathbf{f}})$ is also Gaussian. In particular, the posterior approximation is defined as $q(\overline{\mathbf{f}}) = \prod_{k=1}^{C} \mathcal{N}(\overline{\mathbf{f}}|, \mathbf{m}_k, \mathbf{V}_k)$. The parameters of this distribution can be obtained by using the formulas given in the Appendix of [40] for the product of two Gaussians, leading to

$$\begin{aligned} \mathbf{V}_k &= \left[(\mathbf{K}^k_{\overline{\mathbf{X}}^k \overline{\mathbf{X}}^k})^{-1} + \boldsymbol{\Upsilon}_k \boldsymbol{\Delta}_k \boldsymbol{\Upsilon}_k^{\mathrm{T}} \right]^{-1} , \\ \mathbf{m}_k &= \mathbf{V}_k \boldsymbol{\Upsilon}_k \tilde{\boldsymbol{\mu}}_k , \end{aligned} \tag{A.6}$$

where $\boldsymbol{\Upsilon}_k = (\boldsymbol{\upsilon}_1^k, \ldots, \boldsymbol{\upsilon}_N^k)$ is a $M \times N$ matrix, $\boldsymbol{\Delta}_k$ is a diagonal $N \times N$ matrix with diagonal entries equal to $C_{i,k}^1$ and $\tilde{\boldsymbol{\mu}}_k$ is a vector where each component is equal to $C_{i,k}^2$.

Appendix A.3. Computation of the cavity distribution

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

$$q(\overline{\mathbf{f}})^{\setminus \alpha i} \propto \frac{q(\overline{\mathbf{f}})}{\tilde{\phi}_i(\overline{\mathbf{f}})^{\alpha}}\,. \tag{A.7}$$

Given that all factors are Gaussian, the resulting distribution will also be Gaussian. The parameters can be obtained by using again the formulas in the Appendix of [40]. However, because $\tilde{\phi}_i$ only depends on $\overline{\mathbf{f}}^k$, only these components of $q(\overline{\mathbf{f}})$ will change. The corresponding parameters of $q(\overline{\mathbf{f}})^{\setminus \alpha i}$ are:

$$\begin{aligned}
\mathbf{V}_k^{\setminus \alpha i} &= (\mathbf{V}_k^{-1} - \alpha \tilde{\mathbf{V}}_{i,k})^{-1} \\
&= (\mathbf{V}_k^{-1} - \alpha C_{i,k}^{1,k} \boldsymbol{\upsilon}_i^k (\boldsymbol{\upsilon}_i^k)^{\mathrm{T}})^{-1} \\
&= \mathbf{V}_k + \mathbf{V}_k \boldsymbol{\upsilon}_i^k [(\alpha C_{i,k}^{1,k})^{-1} - \boldsymbol{\upsilon}_i^k \mathbf{V}_k (\boldsymbol{\upsilon}_i^k)^{\mathrm{T}}]^{-1} (\boldsymbol{\upsilon}_i^k)^{\mathrm{T}} \mathbf{V}_k\,,
\end{aligned} \tag{A.8}$$

$$\begin{aligned}
\mathbf{m}_k^{\setminus \alpha i} &= \mathbf{V}_k^{\setminus \alpha i} (\mathbf{V}_k^{-1} \mathbf{m}_k - \alpha \tilde{\mathbf{m}}_{i,k}^k) \\
&= \mathbf{V}_k^{\setminus \alpha i} (\mathbf{V}_k^{-1} \mathbf{m}_k - C_{i,k}^{2,k} \boldsymbol{\upsilon}_i^k) \\
&= \mathbf{V}_k^{\setminus \alpha i} \mathbf{V}_k^{-1} \mathbf{m}_k - \alpha C_{i,k}^{2,k} \boldsymbol{\upsilon}_i^k \mathbf{V}_k^{\setminus \alpha i} \\
&= \mathbf{V}_k \mathbf{V}_k^{-1} \mathbf{m}_k + \mathbf{V}_k \boldsymbol{\upsilon}_i^k [(\alpha C_{i,k}^{1,k})^{-1} - \boldsymbol{\upsilon}_i^k \mathbf{V}_k (\boldsymbol{\upsilon}_i^k)^{\mathrm{T}}]^{-1} (\boldsymbol{\upsilon}_i^k)^{\mathrm{T}} \mathbf{V}_k \mathbf{V}_k^{-1} \mathbf{m}_k \\
&\quad - \alpha C_{i,k}^{2,k} \boldsymbol{\upsilon}_i^k \mathbf{V}_k^{\setminus \alpha i} \\
&= \mathbf{m}_k + \mathbf{V}_k \boldsymbol{\upsilon}_i^k [(\alpha C_{i,k}^{1,k})^{-1} - \boldsymbol{\upsilon}_i^k \mathbf{V}_k (\boldsymbol{\upsilon}_i^k)^{\mathrm{T}}]^{-1} (\boldsymbol{\upsilon}_i^k)^{\mathrm{T}} \mathbf{m}_k - \mathbf{V}_k \boldsymbol{\upsilon}^{\mathrm{T}} \alpha C_{i,k}^{2,k} \\
&\quad - \mathbf{V}_k \boldsymbol{\upsilon}_i^k [(\alpha C_{i,k}^{1,k})^{-1} - \boldsymbol{\upsilon}_i^k \mathbf{V}_k (\boldsymbol{\upsilon}_i^k)^{\mathrm{T}}]^{-1} (\boldsymbol{\upsilon}_i^k)^{\mathrm{T}} \mathbf{V}_k \boldsymbol{\upsilon}_i^k \alpha C_{i,k}^{2,k}\,,
\end{aligned} \tag{A.9}$$

where we have used the Woodbury matrix identity and $\boldsymbol{\upsilon}_i^k$, $C_{i,k}^1$ and $C_{i,k}^2$ are the parameters specified in Appendix A.2.

550 *Appendix A.4. Update of the approximate factors*

In this section we show how to find the approximate factors $\tilde{\phi}_i$ once the cavity distribution $q^{\backslash \alpha i}$ has already been computed. We can compute the moments of $\phi_i q^{\backslash \alpha i}$ by getting the derivatives of $\log Z_i$ with respect to the parameters of $q^{\backslash \alpha i}$, as indicated in the Appendix of [40]. For that, note that:

$$\mathbf{m}_k = \boldsymbol{\upsilon}_i^k \mathbf{m}^{\backslash \alpha i} \tag{A.10}$$

$$\mathbf{V}_k = \kappa^k_{\mathbf{x}_i \mathbf{x}_i} - (\boldsymbol{\upsilon}_i^k)^{\mathrm{T}} (\mathbf{K}^k_{\overline{\mathbf{X}}^k \overline{\mathbf{X}}^k})^{-1} \boldsymbol{\upsilon}_i^k + (\boldsymbol{\upsilon}_i^k)^{\mathrm{T}} \mathbf{V}^{\backslash \alpha i} \boldsymbol{\upsilon}_i^k \tag{A.11}$$

The derivatives are:

$$\frac{\partial \log Z_i}{\partial \mathbf{m}_k^{\backslash \alpha i}} = \frac{\partial \log Z_i}{\partial \mathbf{m}_k} \frac{\partial \mathbf{m}_k}{\partial \mathbf{m}_k^{\backslash \alpha i}} = \frac{\partial \log Z_i}{\partial \mathbf{m}_k} \boldsymbol{\upsilon}_i^k \,, \tag{A.12}$$

$$\frac{\partial \log Z_i}{\partial \mathbf{V}_k^{\backslash \alpha i}} = \frac{\partial \log Z_i}{\partial \mathbf{V}_k} \frac{\partial \mathbf{V}_k}{\partial \mathbf{V}_k^{\backslash \alpha i}} = \frac{\partial \log Z_i}{\partial \mathbf{V}_k} \boldsymbol{\upsilon}_i^k (\boldsymbol{\upsilon}_i^k)^{\mathrm{T}} \,, \tag{A.13}$$

where $\boldsymbol{\upsilon}_i^k$ is the parameter specified in Appendix A.2. By following the Appendix of [40] we can obtain the moments of $\phi_i q^{\backslash \alpha i}$ (mean $\hat{\mathbf{m}}_c$ and covariance $\hat{\mathbf{V}}_c$) from the derivatives of $\log Z_i$ with respect to the parameters of $q^{\backslash \alpha i}$. Namely:

$$\hat{\mathbf{m}}_{i,k} = \mathbf{m}_k^{\backslash \alpha i} + \mathbf{V}_k^{\backslash \alpha i} \frac{\partial \log Z_i}{\partial \mathbf{m}_k^{\backslash \alpha i}} = \mathbf{m}_k^{\backslash \alpha i} + \mathbf{V}_k^{\backslash \alpha i} \frac{\partial \log Z_i}{\partial \mathbf{m}_k} \boldsymbol{\upsilon}_i^k \tag{A.14}$$

$$\begin{aligned}
\hat{\mathbf{V}}_{i,k} &= \mathbf{V}_k^{\backslash \alpha i} - \mathbf{V}_k^{\backslash \alpha i} \left(\left(\frac{\partial \log Z_i}{\partial \mathbf{m}_k^{\backslash \alpha i}} \right) \left(\frac{\partial \log Z_i}{\partial \mathbf{m}_k^{\backslash \alpha i}} \right)^{\mathrm{T}} - 2 \frac{\partial \log Z_i}{\partial \mathbf{V}_k^{\backslash \alpha i}} \right) \mathbf{V}_k^{\backslash \alpha i} \\
&= \mathbf{V}_k^{\backslash \alpha i} - \mathbf{V}_k^{\backslash \alpha i} \left(\left(\frac{\partial \log Z_i}{\partial \mathbf{m}_k} \right) \left(\frac{\partial \log Z_i}{\partial \mathbf{m}_k} \right)^{\mathrm{T}} \boldsymbol{\upsilon}_i^k (\boldsymbol{\upsilon}_i^k)^{\mathrm{T}} - 2 \frac{\partial \log Z_i}{\partial \mathbf{V}_k} \boldsymbol{\upsilon}_i^k (\boldsymbol{\upsilon}_i^k)^{\mathrm{T}} \right) \mathbf{V}_k^{\backslash \alpha i} \\
&= \mathbf{V}_k^{\backslash \alpha i} - \mathbf{V}_k^{\backslash \alpha i} \left(\left(\frac{\partial \log Z_i}{\partial \mathbf{m}_k} \right) \left(\frac{\partial \log Z_i}{\partial \mathbf{m}_k} \right)^{\mathrm{T}} - 2 \frac{\partial \log Z_i}{\partial \mathbf{V}_k} \right) \boldsymbol{\upsilon}_i^k (\boldsymbol{\upsilon}_i^k)^{\mathrm{T}} \mathbf{V}_k^{\backslash \alpha i} \,.
\end{aligned} \tag{A.15}$$

Now we can find the parameters of the approximate factor $\tilde{\phi}_i$, which is obtained as $\tilde{\phi}_i = Z_i q^{\mathrm{new}} / q^{\backslash \alpha i}$, where q^{new} is a Gaussian distribution with the parameters of $\phi_i q^{\backslash \alpha i}$ just computed. By following the equations given in the Appendix of [40] we obtain the precision matrices of the approximate factor:

$$\begin{aligned}
\tilde{\mathbf{V}}_{i,k} &= (\hat{\mathbf{V}}_{i,k})^{-1} - (\mathbf{V}_k^{\backslash \alpha i})^{-1} \\
&= \left(\mathbf{V}_k^{\backslash \alpha i} - \mathbf{V}_k^{\backslash \alpha i}\boldsymbol{v}_i^k \left[\left(\frac{\partial \log Z_i}{\partial \mathbf{m}_k}\right)\left(\frac{\partial \log Z_i}{\partial \mathbf{m}_k}\right)^{\mathrm{T}} - 2\frac{\partial \log Z_i}{\partial \mathbf{V}_k}\right](\boldsymbol{v}_i^k)^{\mathrm{T}}\mathbf{V}_k^{\backslash \alpha i}\right)^{-1} - (\mathbf{V}_k^{\backslash \alpha i})^{-1} \\
&= (\mathbf{V}_k^{\backslash \alpha i})^{-1} + (\mathbf{V}_k^{\backslash \alpha i})^{-1}\mathbf{V}_k^{\backslash \alpha i}\boldsymbol{v}_i^k \left(\left[\left(\frac{\partial \log Z_i}{\partial \mathbf{m}_k}\right)\left(\frac{\partial \log Z_i}{\partial \mathbf{m}_k}\right)^{\mathrm{T}} - 2\frac{\partial \log Z_i}{\partial \mathbf{V}_k}\right]^{-1}\right. \\
&\quad \left. - (\boldsymbol{v}_i^k)^{\mathrm{T}}\mathbf{V}_k^{\backslash \alpha i}(\mathbf{V}_k^{\backslash \alpha i})^{-1}\mathbf{V}_k^{\backslash \alpha i}\boldsymbol{v}_i^k\right)^{-1}(\boldsymbol{v}_i^k)^{\mathrm{T}}\mathbf{V}_k^{\backslash \alpha i}(\mathbf{V}_k^{\backslash \alpha i})^{-1} - (\mathbf{V}_k^{\backslash \alpha i})^{-1} \\
&= \boldsymbol{v}_i^k \left(\left[\left(\frac{\partial \log Z_i}{\partial \mathbf{m}_k}\right)\left(\frac{\partial \log Z_i}{\partial \mathbf{m}_k}\right)^{\mathrm{T}} - 2\frac{\partial \log Z_i}{\partial \mathbf{V}_k}\right]^{-1} - (\boldsymbol{v}_i^k)^{\mathrm{T}}\mathbf{V}_k^{\backslash \alpha i}\boldsymbol{v}_i^k\right)^{-1}(\boldsymbol{v}_i^k)^{\mathrm{T}}\,, \qquad \text{(A.16)}
\end{aligned}$$

where we have used the Woodbury matrix identity to compute $(\hat{\mathbf{V}}_{i,k})^{-1}$. Let us define $C_{i,k}^1$ as:

$$C_{i,k}^1 = \frac{1}{\alpha}\left(\left[\left(\frac{\partial \log Z_i}{\partial \mathbf{m}_k}\right)\left(\frac{\partial \log Z_i}{\partial \mathbf{m}_k}\right)^{\mathrm{T}} - 2\frac{\partial \log Z_i}{\partial \mathbf{V}_k}\right]^{-1} - (\boldsymbol{v}_i^k)^{\mathrm{T}}\mathbf{V}_k^{\backslash \alpha i}\boldsymbol{v}_i^k\right)^{-1}\,, \qquad \text{(A.17)}$$

where we divide by α because we retrieve α times the approximate factor from PEP. The precision matrix of the approximate factors will be then:

$$\tilde{\mathbf{V}}_{i,k} = C_{i,k}^1 \boldsymbol{v}_i^k (\boldsymbol{v}_i^k)^{\mathrm{T}}\,. \qquad \text{(A.18)}$$

For the first natural parameter we proceed in a similar way

$$
\begin{aligned}
\tilde{\mathbf{m}}_{i,k}^{y_i} &= (\hat{\mathbf{V}}_{i,k}^{y_i})^{-1}\hat{\mathbf{m}}_{i,k}^{y_i} - (\mathbf{V}_{y_i}^{\setminus\alpha i})^{-1}\mathbf{m}_{y_i}^{\setminus\alpha i} \\
&= ((\mathbf{V}_{y_i}^{\setminus\alpha i})^{-1} + \tilde{\mathbf{V}}_{i,k}^{y_i})\hat{\mathbf{m}}_{i,k}^{y_i} - (\mathbf{V}_{y_i}^{\setminus\alpha i})^{-1}\mathbf{m}_{y_i}^{\setminus\alpha i} \\
&= (\mathbf{V}_{y_i}^{\setminus\alpha i})^{-1}\hat{\mathbf{m}}_{i,k}^{y_i} + \tilde{\mathbf{V}}_{i,k}^{y_i}\hat{\mathbf{m}}_{i,k}^{y_i} - (\mathbf{V}_{y_i}^{\setminus\alpha i})^{-1}\mathbf{m}_{y_i}^{\setminus\alpha i} \\
&= (\mathbf{V}_{y_i}^{\setminus\alpha i})^{-1}\left[\mathbf{m}_k^{\setminus\alpha i} + \mathbf{V}_k^{\setminus\alpha i}\frac{\partial\log Z_i}{\partial\mathbf{m}_k}\boldsymbol{\upsilon}_i^k\right] \\
&\qquad + \tilde{\mathbf{V}}_{i,k}^{y_i}\left[\mathbf{m}_k^{\setminus\alpha i} + \mathbf{V}_k^{\setminus\alpha i}\frac{\partial\log Z_i}{\partial\mathbf{m}_k}\boldsymbol{\upsilon}_i^k\right] - (\mathbf{V}_{y_i}^{\setminus\alpha i})^{-1}\mathbf{m}_{y_i}^{\setminus\alpha i} \\
&= (\mathbf{V}_{y_i}^{\setminus\alpha i})^{-1}\mathbf{m}_{y_i}^{\setminus\alpha i} + (\mathbf{V}_{y_i}^{\setminus\alpha i})^{-1}\mathbf{V}_{y_i}^{\setminus\alpha i}\frac{\partial\log Z_i}{\partial\mathbf{m}_k}\boldsymbol{\upsilon}_i^{y_i} \\
&\qquad + \tilde{\mathbf{V}}_{i,k}^{y_i}\left[\mathbf{m}_{y_i}^{\setminus\alpha i} + \mathbf{V}_{y_i}^{\setminus\alpha i}\frac{\partial\log Z_i}{\partial\mathbf{m}_k}\boldsymbol{\upsilon}_i^{y_i}\right] - (\mathbf{V}_{y_i}^{\setminus\alpha i})^{-1}\mathbf{m}_{y_i}^{\setminus\alpha i} \\
&= \frac{\partial\log Z_i}{\partial\mathbf{m}_k}\boldsymbol{\upsilon}_i^{y_i} + \tilde{\mathbf{V}}_{i,k}^{y_i}\mathbf{m}_{y_i}^{\setminus\alpha i} + \tilde{\mathbf{V}}_{i,k}^{y_i}\mathbf{V}_{y_i}^{\setminus\alpha i}\frac{\partial\log Z_i}{\partial\mathbf{m}_k}\boldsymbol{\upsilon}_i^{y_i} \\
&= \frac{\partial\log Z_i}{\partial\mathbf{m}_k}\boldsymbol{\upsilon}_i^{y_i} + C_{i,k}^{1,y_i}\boldsymbol{\upsilon}_i^{y_i}(\boldsymbol{\upsilon}_i^{y_i})^{\mathrm{T}}\mathbf{m}_{y_i}^{\setminus\alpha i} + \frac{\partial\log Z_i}{\partial\mathbf{m}_k}C_{i,k}^{1,y_i}\boldsymbol{\upsilon}_i^{y_i}(\boldsymbol{\upsilon}_i^{y_i})^{\mathrm{T}}\mathbf{V}_{y_i}^{\setminus\alpha i}\boldsymbol{\upsilon}_i^{y_i} \\
&= \left[\frac{\partial\log Z_i}{\partial\mathbf{m}_k} + C_{i,k}^{1,y_i}(\boldsymbol{\upsilon}_i^{y_i})^{\mathrm{T}}\mathbf{m}_{y_i}^{\setminus\alpha i} + \frac{\partial\log Z_i}{\partial\mathbf{m}_k}C_{i,k}^{1,y_i}(\boldsymbol{\upsilon}_i^{y_i})^{\mathrm{T}}\mathbf{V}_{y_i}^{\setminus\alpha i}\boldsymbol{\upsilon}_i^{y_i}\right]\boldsymbol{\upsilon}_i^{y_i}\,,
\end{aligned}
\tag{A.19}
$$

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

$$
C_{i,k}^2 = \frac{1}{\alpha}\left[\frac{\partial\log Z_i}{\partial\mathbf{m}_k} + C_{i,k}^1(\boldsymbol{\upsilon}_i^k)^{\mathrm{T}}\mathbf{m}_k^{\setminus\alpha i} + \frac{\partial\log Z_i}{\partial\mathbf{m}_k}C_{i,k}^1(\boldsymbol{\upsilon}_i^k)^{\mathrm{T}}\mathbf{V}_k^{\setminus\alpha i}\boldsymbol{\upsilon}_i^k\right]\,, \tag{A.20}
$$

we obtain the following expressions for the first natural parameter:

$$
\tilde{\mathbf{m}}_{i,k} = C_{i,k}^2\boldsymbol{\upsilon}_i^k\,. \tag{A.21}
$$

Once we have these parameters we can compute the value of the normalization constant $\tilde{Z}_i$, which guarantees that the approximate factor integrates the same as the exact factor with respect to $q^{\setminus\alpha i}$. The log of this constant is:

$$
\log\tilde{Z}_i = \log\mathbb{E}_{q(\mathbf{f}_i)}\left[\left(\frac{\phi_i}{\tilde{\phi}_i}\right)^{\alpha}\right] = \log\mathbb{E}_{q(\mathbf{f}_i)}\left[\left(\frac{p(y_i|\mathbf{f}_i)}{\tilde{\phi}_i}\right)^{\alpha}\right] + g(\boldsymbol{\theta}^{\setminus\boldsymbol{\alpha i}}) - g(\boldsymbol{\theta})\,. \tag{A.22}
$$

As we are using the robust likelihood $p(y_i|\mathbf{f}_i) = (1-\epsilon)\prod_{k\neq y_i}\Theta\left(f^{y_i}(\mathbf{x}_i) - f^k(\mathbf{x}_i)\right) + \frac{\epsilon}{C}$ we will need to use one-dimensional quadrature techniques to compute the expectation $\mathbb{E}_{q(\mathbf{f}_i)}\left[\left(p(y_i|\mathbf{f}_i)/\tilde{\phi}_i\right)^\alpha\right]$. This expectation is simply related to the probability that a Gaussian random variable is larger than several others (one per each other class label) [18].

Appendix A.5. Parallel EP updates and damping

We update all approximate factors in parallel. This means that we compute all the quantities required for updating each of the approximate factors at once (in particular the quantities derived for the cavity distribution $q^{\backslash \alpha i}$). Parallel updates are faster than sequential EP updates because there is no need to introduce a loop over the data. All computations can be carried out in terms of matrix vector multiplications that are often more efficient. A disadvantage of parallel updates is, however, that they may lead to unstable PEP updates. To prevent unstable PEP updates we used damped PEP updates. These simply replace the PEP updates of each approximate factor with a linear combination of old and new parameters. For example, we set $\tilde{C}^1_{i,k} = (\tilde{C}^1_{i,k})^{\text{new}}\rho + (\tilde{C}^1_{i,k})^{\text{old}}(1-\rho)$ in the case of the $\tilde{C}_{i,k}$ parameter of the approximate factor (we do this with all the parameters). In the previous expression $\rho \in [0,1]$ a value that specifies the amount of damping. If $\rho = 0$ no update happens. If $\rho = 1$ we obtain the original EP update. Importantly, damping does not change the EP convergence points so it does not affect to the quality of the solution.

Appendix A.6. Estimate of the marginal likelihood

As we have seen in the main manuscript, the estimate of the log marginal likelihood is:

$$\log Z_q = g(\boldsymbol{\theta}) - g(\boldsymbol{\theta}_{\text{prior}}) + \frac{1}{\alpha}\sum_{i=1}^{N}\log \tilde{Z}_i \tag{A.23}$$

$$\log \tilde{Z}_i = \log \mathbb{E}_{q(\mathbf{f}_i)}\left[\left(\frac{p(y_i|\mathbf{f}_i)}{\tilde{\phi}_i}\right)^{\alpha}\right] + g(\boldsymbol{\theta}^{\setminus \boldsymbol{\alpha i}}) - g(\boldsymbol{\theta}) \tag{A.24}$$

where $\boldsymbol{\theta}$, $\theta^{\setminus \alpha i}$ and $\boldsymbol{\theta}_{\text{prior}}$ are the natural parameters of q, $q^{\setminus \alpha i}$ and $p(\overline{\mathbf{f}})$ 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 mean and covariance matrix of that Gaussian distribution over D dimensions, then

$$g(\boldsymbol{\theta}') = \frac{D}{2}\log 2\pi + \frac{1}{2}\log|\boldsymbol{\Sigma}| + \frac{1}{2}\boldsymbol{\mu}^{\text{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\,, \tag{A.25}$$

which leads to

$$\log Z_q = \sum_{k=1}^{C}\frac{1}{2}\log|\mathbf{V}_k| + \frac{1}{2}\mathbf{m}_k^{\text{T}}\mathbf{V}_k^{-1}\mathbf{m}_k - \frac{1}{2}|\mathbf{K}^k_{\overline{\mathbf{X}}^k\overline{\mathbf{X}}^k}| + \frac{1}{\alpha}\sum_{i=1}^{N}\log\tilde{Z}_i\,, \tag{A.26}$$

with

$$\begin{aligned}\log \tilde{Z}_i = \log \mathbb{E}_{q(\mathbf{f}_i)}\left[\left(\frac{p(y_i|\mathbf{f}_i)}{\tilde{\phi}_i}\right)^{\alpha}\right] + \frac{1}{2}\log\left|\mathbf{V}_{y_i}^{\setminus\alpha i}\right| \\ + \frac{1}{2}\left(\mathbf{m}_{y_i}^{\setminus\alpha i}\right)^{\text{T}}\left(\mathbf{V}_{y_i}^{\setminus\alpha i}\right)^{-1}\mathbf{m}_{y_i}^{\setminus\alpha i} \\ - \frac{1}{2}\log\left|\mathbf{V}_{y_i}^{\setminus\alpha i}\right| - \frac{1}{2}\left(\mathbf{m}_{y_i}^{\setminus\alpha i}\right)^{\text{T}}\left(\mathbf{V}_{y_i}^{\setminus\alpha i}\right)^{-1}\mathbf{m}_{y_i}^{\setminus\alpha i}\,.\end{aligned} \tag{A.27}$$

This expression can be evaluated very efficiently using the Woodbury matrix identity; the matrix determinant lemma; that $(\mathbf{V}_k^{\setminus\alpha i})^{-1} = \mathbf{V}_k^{-1} - \tilde{\mathbf{V}}_{i,k}$; that $\mathbf{m}_k^{\setminus\alpha i} = \mathbf{V}_k^{\setminus\alpha i}(\mathbf{V}_k^{-1}\mathbf{m}_k - \tilde{\mathbf{m}}_{i,k})$; and the special form of the parameters of the approximate factors $\tilde{\mathbf{V}}_{i,k}$ and $\tilde{\mathbf{m}}_{i,k}$.

Appendix A.7. Gradient of log Zq after convergence and learning rate

We derive the expression for the gradient of $\log Z_q$ after PEP has converged. Let ξ_j^k be one hyper-parameter of the model (*i.e.*, a parameter of one of the covariance functions or a component of the inducing points) and $\boldsymbol{\theta}$ and $\boldsymbol{\theta}_{\text{prior}}$ to the natural parameters of q and $p(\overline{\mathbf{f}})$ respectively. When PEP has converged, the approximate factors can be considered to be fixed (it does not change with the model hyper-parameters) [24]. In this case, it is only necessary to consider the direct dependency of $\log Z_i$ on ξ_j^k [24]. But in our case, we update the hyper-parameters at each PEP iteration, so we will need to consider the indirect dependency too. The gradient is given by:

$$\begin{aligned}
\frac{\partial \log Z_q}{\partial \xi_j^k} &= \left(\frac{\partial g(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right)^{\text{T}} \frac{\partial \boldsymbol{\theta}}{\partial \xi_j^k} - \left(\frac{\partial g(\boldsymbol{\theta}_{\text{prior}})}{\partial \boldsymbol{\theta}_{\text{prior}}}\right)^{\text{T}} \frac{\partial \boldsymbol{\theta}_{\text{prior}}}{\partial \xi_j^k} + \frac{1}{\alpha}\sum_{i=1}^N \frac{\partial \log \mathbb{E}_{q(\mathbf{f}_i)}\left[\left(\frac{p(y_i|\mathbf{f}_i)}{\tilde{\phi}_i}\right)^{\alpha}\right]}{\partial \xi_j^k} \\
&\quad + \left(\frac{\partial g(\boldsymbol{\theta}^{\setminus \alpha i})}{\partial \boldsymbol{\theta}^{\setminus \alpha i}}\right)^{\text{T}} \frac{\partial \boldsymbol{\theta}^{\setminus \alpha i}}{\partial \xi_j^k} - \left(\frac{\partial g(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right)^{\text{T}} \frac{\partial \boldsymbol{\theta}}{\partial \xi_j^k} \\
&= \boldsymbol{\eta}^{\text{T}} \frac{\partial \boldsymbol{\theta}}{\partial \xi_j^k} - (\boldsymbol{\eta}_{\text{prior}})^{\text{T}} \frac{\partial \boldsymbol{\theta}_{\text{prior}}}{\partial \xi_j^k} + \frac{1}{\alpha}\sum_{i=1}^N \frac{\partial \log \mathbb{E}_{q(\mathbf{f}_i)}\left[\left(\frac{p(y_i|\mathbf{f}_i)}{\tilde{\phi}_i}\right)^{\alpha}\right]}{\partial \xi_j^k} \\
&\quad + (\boldsymbol{\eta}^{\setminus \alpha i})^{\text{T}} \frac{\partial \boldsymbol{\theta}^{\setminus \alpha i}}{\partial \xi_j^k} - \boldsymbol{\eta}^{\text{T}} \frac{\partial \boldsymbol{\theta}}{\partial \xi_j^k} \\
&= \boldsymbol{\eta}^{\text{T}} \frac{\partial \boldsymbol{\theta}_{\text{prior}}}{\partial \xi_j^k} - (\boldsymbol{\eta}_{\text{prior}})^{\text{T}} \frac{\partial \boldsymbol{\theta}_{\text{prior}}}{\partial \xi_j^k} + \frac{1}{\alpha}\sum_{i=1}^N \frac{\partial \log \mathbb{E}_{q(\mathbf{f}_i)}\left[\left(\frac{p(y_i|\mathbf{f}_i)}{\tilde{\phi}_i}\right)^{\alpha}\right]}{\partial \xi_j^k} \\
&\quad + (\boldsymbol{\eta}^{\setminus \alpha i})^{\text{T}} \frac{\partial \boldsymbol{\theta}_{\text{prior}}}{\partial \xi_j^k} - \boldsymbol{\eta}^{\text{T}} \frac{\partial \boldsymbol{\theta}_{\text{prior}}}{\partial \xi_j^k} \\
&= (\boldsymbol{\eta}^{\text{T}} - \boldsymbol{\eta}_{\text{prior}}^{\text{T}}) \frac{\partial \boldsymbol{\theta}_{\text{prior}}}{\partial \xi_j^k} + \frac{1}{\alpha}\sum_{i=1}^N \frac{\partial \log \mathbb{E}_{q(\mathbf{f}_i)}\left[\left(\frac{p(y_i|\mathbf{f}_i)}{\tilde{\phi}_i}\right)^{\alpha}\right]}{\partial \xi_j^k} \\
&\quad + (\boldsymbol{\eta}^{\setminus \alpha i} - \boldsymbol{\eta})^{\text{T}} \frac{\partial \boldsymbol{\theta}_{\text{prior}}}{\partial \xi_j^k}\,,
\end{aligned} \tag{A.28}$$

where we have used the chain rule of matrix derivatives [41], the especial form of the derivatives when using inducing points [42] and that $\boldsymbol{\theta} = \boldsymbol{\theta}_{\text{prior}} + \sum_{i=1}^N \boldsymbol{\theta}_i$, with $\boldsymbol{\theta}_i$ the natural parameters of the approximate factor $\tilde{\phi}_i$. Furthermore, $\boldsymbol{\eta}$

and $\boldsymbol{\eta}_{\text{prior}}$ are expected sufficient statistics under the posterior approximation q and the prior, respectively. This gradient coincides with the one in the main manuscript.

It is important to note that one has to use the chain rule of matrix derivatives when trying to use the previous expression to compute the gradient. In particular, natural parameters and expected sufficient statistics are expressed in the form of matrices. Thus, one has to use in practice the chain rule of matrix derivatives, as indicated in [41]. For example:

$$(\boldsymbol{\eta} - \boldsymbol{\eta}_{\text{prior}})^{\text{T}} \frac{\partial \boldsymbol{\theta}_{\text{prior}}}{\partial \xi_j^k} = -\frac{1}{2} \text{trace} \left(\mathbf{M}_k^{\text{T}} \frac{\partial \mathbf{K}^k_{\overline{\mathbf{X}}^k \overline{\mathbf{X}}^k}}{\partial \xi_j^k} \right) , \tag{A.29}$$

where

$$\mathbf{M}_k = \left(\mathbf{K}^k_{\overline{\mathbf{X}}^k\overline{\mathbf{X}}^k}\right)^{-1} - \left(\mathbf{K}^k_{\overline{\mathbf{X}}^k\overline{\mathbf{X}}^k}\right)^{-1} \mathbf{V}_k \left(\mathbf{K}^k_{\overline{\mathbf{X}}^k\overline{\mathbf{X}}^k}\right)^{-1} - \left(\mathbf{K}^k_{\overline{\mathbf{X}}^k\overline{\mathbf{X}}^k}\right)^{-1} \mathbf{m}_k \mathbf{m}_k^{\text{T}} \left(\mathbf{K}^k_{\overline{\mathbf{X}}^k\overline{\mathbf{X}}^k}\right)^{-1} , \tag{A.30}$$

where $\mathbf{V}_k$ and $\mathbf{m}_k$ are the covariance matrix and mean vector of the k-th component of q. Furthermore, several standard properties of the trace can be employed to simplify the computations. In particular, the trace is invariant to cyclic rotations. Namely, trace($\mathbf{ABCD}$) = trace($\mathbf{DABC}$). The derivatives with respect to each $\log \mathbb{E}_{q(\mathbf{f}_i)} \left[\left(p(y_i|\mathbf{f}_i)/\tilde{\phi}_i \right)^{\alpha} \right]$ can be computed using quadrature techniques.

In our experiments we use an adaptive learning rate for the batch PEP methods. This learning rate is different for each hyper-parameter. The rule that we use is to increase the learning rate by 2% if the sign of the estimate of the gradient for that hyper-parameter does not change between two consecutive iterations. If a change is observed, we multiply the learning rate by 1/2. When applying stochastic optimization methods, we use the ADAM method with the default settings to estimate the learning rate [37].

Appendix A.8. 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 $\mathbf{x}^\star$, denoted by $\mathbf{f}^\star = (f^1(\mathbf{x}^\star), \ldots, f^C(\mathbf{x}^\star))^\mathrm{T}$:

$$p(\mathbf{f}^\star|\mathbf{y}) = \int p(\mathbf{f}^\star|\overline{\mathbf{f}})p(\overline{\mathbf{f}}|\mathbf{y})d\overline{\mathbf{f}} = \int p(\mathbf{f}^\star|\overline{\mathbf{f}})q(\overline{\mathbf{f}})d\overline{\mathbf{f}} \approx \prod_{k=1}^{C} \mathcal{N}(f^k(\mathbf{x}^\star)|m_k^\star, v_k^\star)\,, \quad \text{(A.31)}$$

where:

$$m_k^\star = (\mathbf{k}^k_{\mathbf{x}^\star,\overline{\mathbf{X}}^k})^\mathrm{T}(\mathbf{K}^k_{\overline{\mathbf{X}}^k\overline{\mathbf{X}}^k})^{-1}\mathbf{m}_k \quad \text{(A.32)}$$

$$v_k^\star = \kappa^k_{\mathbf{x}^\star,\mathbf{x}^\star} - (\mathbf{k}^k_{\mathbf{x}^\star,\overline{\mathbf{X}}^k})^\mathrm{T}(\mathbf{K}^k_{\overline{\mathbf{X}}^k\overline{\mathbf{X}}^k})^{-1}\mathbf{k}^k_{\mathbf{x}^\star,\overline{\mathbf{X}}^k} + (\mathbf{k}^k_{\mathbf{x}^\star,\overline{\mathbf{X}}^k})^\mathrm{T}(\mathbf{K}^k_{\overline{\mathbf{X}}^k\overline{\mathbf{X}}^k})^{-1}\mathbf{V}_k(\mathbf{K}^k_{\overline{\mathbf{X}}^k\overline{\mathbf{X}}^k})^{-1}\mathbf{k}^k_{\mathbf{x}^\star,\overline{\mathbf{X}}^k}\,. \quad \text{(A.33)}$$

This approximate posterior can be used to obtain an approximate predictive distribution for the class label $y^\star$:

$$\begin{aligned}
p(y^\star|\mathbf{x}^\star,\mathbf{y}) &= \int p(y^\star|\mathbf{x}^\star,\mathbf{f}^\star)p(\mathbf{f}^\star|\mathbf{y})d\mathbf{f}^\star \\
&= \int p(y^\star|\mathbf{x}^\star,\mathbf{f}^\star)\prod_{k=1}^{C}\mathcal{N}(f^k(\mathbf{x}^\star)|\mathbf{m}_k^\star,\mathbf{V}_k^\star)d\mathbf{f}^\star \\
&= \int \left[(1-\epsilon)\prod_{k\neq y^\star}\Theta\left(f^{y^\star}(\mathbf{x}^\star) - f^c(\mathbf{x}^\star)\right) + \frac{\epsilon}{C}\right] \\
&\qquad \prod_{k=1}^{C}\mathcal{N}(f^k(\mathbf{x}^\star)|m_k^\star,v_k^\star)d\mathbf{f}^\star \\
&= \int \left[(1-\epsilon)\prod_{k\neq y^\star}\Theta\left(f^{y^\star}(\mathbf{x}^\star) - f^k(\mathbf{x}^\star)\right) + \frac{\epsilon}{C}\right] \\
&\qquad \prod_{k\neq y^\star}\mathcal{N}(f^k(\mathbf{x}^\star)|\mathbf{m}_k^\star,\mathbf{V}_k^\star)d\mathbf{f}^\star\mathcal{N}(f^{y^\star}(\mathbf{x}^\star)|\mathbf{m}_{y^\star}^\star,\mathbf{V}_{y^\star}^\star) \\
&= \int \left[(1-\epsilon)\prod_{k\neq y^\star}\Phi\left(\frac{f^{y^\star}(\mathbf{x}^\star) - \mathbf{m}_k^\star}{\sqrt{v_k^\star}}\right) + \frac{\epsilon}{C}\right] \\
&\qquad \mathcal{N}(f^{y^\star}(\mathbf{x}^\star)|m_{y^\star}^\star,v_{y^\star}^\star)df^{y^\star}(\mathbf{x}^\star)\,,
\end{aligned} \quad \text{(A.34)}$$

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

Appendix B. Details of the UCI Datasets

Table B.2 shows the characteristics of the datasets considered from the UCI repository in the main document. This table shows, for each problem, the number of samples, the number of attributes and the number of class labels.

Table B.2: Characteristics of the datasets from the UCI Repository.

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

Appendix C. Comparison to Baseline Methods

In this section we compare the PEP algorithm ($\alpha \to 0$, $\alpha = 0.5$ and $\alpha = 1$) with three baseline methods: label regression, Laplace approximation and a MCMC method that uses Gibbs sampling. We have performed these experiments on the 8 UCI repository datasets that are summarized in Table B.2.

Label regression implementation uses the inducing point approximation and EP algorithm. Note that in the regression case, EP results in exact inference, as the likelihood factors are Gaussian [1]. The approximate factors are updated via regular EP updates, and the model hyper-parameters are optimized by gradient ascent with an adaptive learning rate (described in Appendix A.7). We consider three values for the number of inducing points M. Namely, 5%, 10% and 20% of the number of training data N. We report averages over 100 repetitions.

In the case of Laplace and MCMC, we first obtain the model hyper-parameters using EP (PEP with $\alpha = 1$) and then we train the methods to optimize the approximation. This is done because learning the hyper-parameters with these two methods is not scalable.

For Laplace, the gradients of the approximation to the marginal likelihood w.r.t. to the hyper-parameters cannot be computed efficiently using sparse approximations, since they have an explicit dependence on the hyper-parameters and an indirect dependence through the mode [43]. This is precisely why there are no works in the literature that use Laplace approximation with sparse GPs. The Laplace approximation uses the softmax likelihood function.

The MCMC method that we have considered used Gibbs sampling. Gibbs sampling generates samples from the joint target distribution by replacing the value of one of the variables by a value drawn from the distribution of that variable conditioned on the values of the remaining variables [43]. This method is asymptotically unbiased.

In Table C.3 it is shown the test error and in Table C.4 we report the test log-likelihood. By looking at the results, we observe that performance of PEP is similar to the one of MCMC, so the predictive distribution of PEP is fairly good.

In conclusion, PEP gives good predictive distributions with the chosen likelihood function, and better than using the softmax. The softmax likelihood can be more robust than considering Gaussian noise like this work, but it makes inference more complicated, and the lack of robustness can be partially compensated by using the robust-max likelihood, where we introduce some noise in the labels by considering possible labelling errors with probability ϵ, even if ϵ is small.

Appendix D. Additional Experimental Results

In this section we add some extra experimental results that did not fit in the main manuscript. In Figure D.8 we show the mean test error rank for each of the proposed methods and several values of α. We report averages over 8 datasets from the UCI repository and 20 splits. Results for PEP and APEP are similar to the ones in the main manuscript in terms of the negative test log likelihood. However, ARPEP seems to give better results in terms of the test error with $\alpha = 0.8$ or $\alpha = 0.9$.

	Problem	**MCMC**	**Laplace**	**Label Regression**	**PEP** ($\alpha \to 0$)	**PEP** ($\alpha = 0.5$)	**PEP** ($\alpha = 1$)
$\mathbf{M = 5\%}$	glass	0.32 ± 0.01	**0.26 ± 0.01**	0.44 ± 0.01	0.36 ± 0.01	0.33 ± 0.01	0.36 ± 0.01
	new-thyroid	0.05 ± 0	0.05 ± 0	0.15 ± 0.01	**0.03 ± 0**	0.04 ± 0	0.12 ± 0.01
	satellite	0.11 ± 0	0.12 ± 0	0.26 ± 0	0.11 ± 0	**0.11 ± 0**	0.11 ± 0
	svmguide2	0.18 ± 0.01	0.18 ± 0.01	0.21 ± 0.01	0.19 ± 0.01	**0.17 ± 0.01**	0.24 ± 0.01
	vehicle	0.19 ± 0	0.22 ± 0.01	0.25 ± 0	0.18 ± 0.01	**0.18 ± 0**	0.19 ± 0
	vowel	0.08 ± 0	0.15 ± 0.01	0.29 ± 0.01	0.06 ± 0	**0.05 ± 0**	0.09 ± 0.01
	waveform	**0.16 ± 0**	0.16 ± 0	0.26 ± 0	0.18 ± 0	0.16 ± 0	0.22 ± 0
	wine	0.03 ± 0	0.04 ± 0	0.03 ± 0	0.03 ± 0	**0.03 ± 0**	0.05 ± 0.01
	Avg. Time	5.02 ± 0.18	62.64 ± 3.77	131.61 ± 9.78	1684.83 ± 56.14	1625.81 ± 59.68	1495.37 ± 71.49
$\mathbf{M = 10\%}$	glass	0.32 ± 0.01	**0.25 ± 0.01**	0.43 ± 0.01	0.36 ± 0.01	0.31 ± 0.01	0.33 ± 0.01
	new-thyroid	0.04 ± 0	0.04 ± 0	0.12 ± 0.01	**0.03 ± 0**	0.03 ± 0	0.08 ± 0.01
	satellite	**0.11 ± 0**	0.12 ± 0	0.24 ± 0	0.11 ± 0	0.11 ± 0	0.11 ± 0
	svmguide2	0.18 ± 0.01	**0.18 ± 0.01**	0.22 ± 0.01	0.2 ± 0.01	0.18 ± 0.01	0.19 ± 0.01
	vehicle	0.19 ± 0	0.22 ± 0.01	0.23 ± 0	**0.17 ± 0**	0.17 ± 0	0.18 ± 0
	vowel	0.05 ± 0	0.19 ± 0.01	0.21 ± 0.01	0.04 ± 0	**0.03 ± 0**	0.05 ± 0
	waveform	**0.16 ± 0**	0.16 ± 0	0.26 ± 0	0.18 ± 0	0.17 ± 0	0.19 ± 0
	wine	**0.02 ± 0**	0.03 ± 0	0.03 ± 0	0.03 ± 0	0.02 ± 0	0.03 ± 0
	Avg. Time	9.87 ± 0.36	202.17 ± 14.77	153.83 ± 9.79	1865.4 ± 68.1	1814.39 ± 78.13	1724.34 ± 82.62
$\mathbf{M = 20\%}$	glass	0.32 ± 0.01	**0.24 ± 0.01**	0.39 ± 0.01	0.36 ± 0.01	0.31 ± 0.01	0.32 ± 0.01
	new-thyroid	0.04 ± 0	0.04 ± 0	0.1 ± 0.01	0.04 ± 0.01	**0.03 ± 0**	0.06 ± 0.01
	satellite	**0.11 ± 0**	0.12 ± 0	0.23 ± 0	0.11 ± 0	0.11 ± 0	0.11 ± 0
	svmguide2	0.18 ± 0.01	0.18 ± 0.01	0.23 ± 0.01	0.19 ± 0.01	0.18 ± 0.01	**0.18 ± 0.01**
	vehicle	0.17 ± 0	0.22 ± 0	0.22 ± 0	0.17 ± 0.01	**0.16 ± 0**	0.17 ± 0
	vowel	0.03 ± 0	0.23 ± 0.01	0.11 ± 0	0.03 ± 0	**0.02 ± 0**	0.03 ± 0
	waveform	0.16 ± 0	0.16 ± 0	0.26 ± 0	0.18 ± 0	0.17 ± 0	**0.16 ± 0**
	wine	0.03 ± 0	0.03 ± 0	0.04 ± 0	0.02 ± 0	**0.02 ± 0**	0.02 ± 0
	Avg. Time	22.89 ± 0.71	766.85 ± 58.39	224.41 ± 11.01	2169.13 ± 93.34	2138.39 ± 82.94	2073.18 ± 108.69

Table C.3: Average test error for each method and average training time in seconds.

	Problem	**MCMC**	**Laplace**	**Label Regression**	**PEP** ($\alpha \to 0$)	**PEP** ($\alpha = 0.5$)	**PEP** ($\alpha = 1$)
M = 5%	glass	0.8 ± 0.02	**0.8 ± 0.01**	1.1 ± 0.03	2.05 ± 0.07	0.81 ± 0.02	0.9 ± 0.02
	new-thyroid	0.12 ± 0.01	0.26 ± 0.01	0.38 ± 0.02	0.11 ± 0.01	**0.09 ± 0.01**	0.35 ± 0.01
	satellite	**0.3 ± 0**	0.5 ± 0.01	0.74 ± 0	0.51 ± 0	0.31 ± 0	0.3 ± 0
	svmguide2	**0.53 ± 0.02**	0.54 ± 0.01	0.7 ± 0.02	0.9 ± 0.06	0.57 ± 0.02	0.65 ± 0.02
	vehicle	0.37 ± 0.01	0.6 ± 0.01	0.58 ± 0.01	0.57 ± 0.04	**0.36 ± 0.01**	0.37 ± 0.01
	vowel	0.27 ± 0.01	0.62 ± 0.01	0.7 ± 0.01	0.38 ± 0.02	**0.17 ± 0.01**	0.29 ± 0.01
	waveform	**0.37 ± 0**	0.39 ± 0	0.59 ± 0	0.67 ± 0.01	0.4 ± 0	0.69 ± 0.01
	wine	**0.08 ± 0.01**	0.14 ± 0.01	0.09 ± 0	0.1 ± 0.01	0.08 ± 0.01	0.49 ± 0.01
	Avg. Time	5.02 ± 0.19	62.64 ± 3.87	131.61 ± 10.43	1684.83 ± 49.43	1625.81 ± 58.99	1495.37 ± 67.77
M = 10%	glass	0.79 ± 0.02	**0.78 ± 0.01**	1.05 ± 0.03	1.97 ± 0.07	0.8 ± 0.02	0.8 ± 0.02
	new-thyroid	0.09 ± 0.01	0.15 ± 0	0.33 ± 0.02	0.1 ± 0.01	**0.08 ± 0.01**	0.31 ± 0.01
	satellite	0.3 ± 0	0.36 ± 0	0.71 ± 0	0.5 ± 0	0.32 ± 0	**0.29 ± 0**
	svmguide2	0.54 ± 0.02	**0.52 ± 0.01**	0.73 ± 0.03	0.88 ± 0.05	0.6 ± 0.03	0.56 ± 0.02
	vehicle	**0.36 ± 0**	0.59 ± 0.01	0.55 ± 0.01	0.54 ± 0.02	0.36 ± 0.01	0.36 ± 0.01
	vowel	0.21 ± 0	0.8 ± 0.02	0.55 ± 0.01	0.26 ± 0.02	**0.14 ± 0**	0.22 ± 0.01
	waveform	**0.37 ± 0**	0.38 ± 0	0.59 ± 0	0.69 ± 0.01	0.43 ± 0	0.62 ± 0.01
	wine	0.07 ± 0.01	0.13 ± 0.01	0.1 ± 0.01	0.08 ± 0.01	**0.07 ± 0.01**	0.39 ± 0.01
	Avg. Time	9.87 ± 0.36	202.17 ± 12.79	153.83 ± 11.07	1865.4 ± 73.57	1814.39 ± 70.72	1724.34 ± 89.06
M = 20%	glass	0.78 ± 0.02	**0.77 ± 0.01**	1 ± 0.02	1.94 ± 0.07	0.8 ± 0.02	0.79 ± 0.02
	new-thyroid	**0.09 ± 0.01**	0.15 ± 0.01	0.28 ± 0.03	0.16 ± 0.04	0.1 ± 0.01	0.27 ± 0.01
	satellite	0.29 ± 0	0.4 ± 0.01	0.69 ± 0	0.48 ± 0	0.32 ± 0	**0.29 ± 0**
	svmguide2	0.55 ± 0.02	**0.53 ± 0.01**	0.77 ± 0.03	0.78 ± 0.04	0.59 ± 0.03	0.55 ± 0.02
	vehicle	**0.35 ± 0.01**	0.6 ± 0.01	0.53 ± 0.01	0.53 ± 0.02	0.36 ± 0.01	0.35 ± 0.01
	vowel	0.2 ± 0	1.09 ± 0.02	0.37 ± 0.01	0.16 ± 0.02	**0.13 ± 0**	0.19 ± 0
	waveform	0.38 ± 0	**0.38 ± 0**	0.61 ± 0	0.7 ± 0.01	0.46 ± 0.01	0.53 ± 0.01
	wine	**0.07 ± 0.01**	0.18 ± 0.01	0.1 ± 0.01	0.08 ± 0.01	0.07 ± 0.01	0.32 ± 0.01
	Avg. Time	22.89 ± 0.86	766.85 ± 49.9	224.41 ± 12.44	2169.13 ± 82.92	2138.39 ± 90.18	2073.18 ± 106.53

Table C.4: Average negative test log likelihood for each method and average training time in seconds.

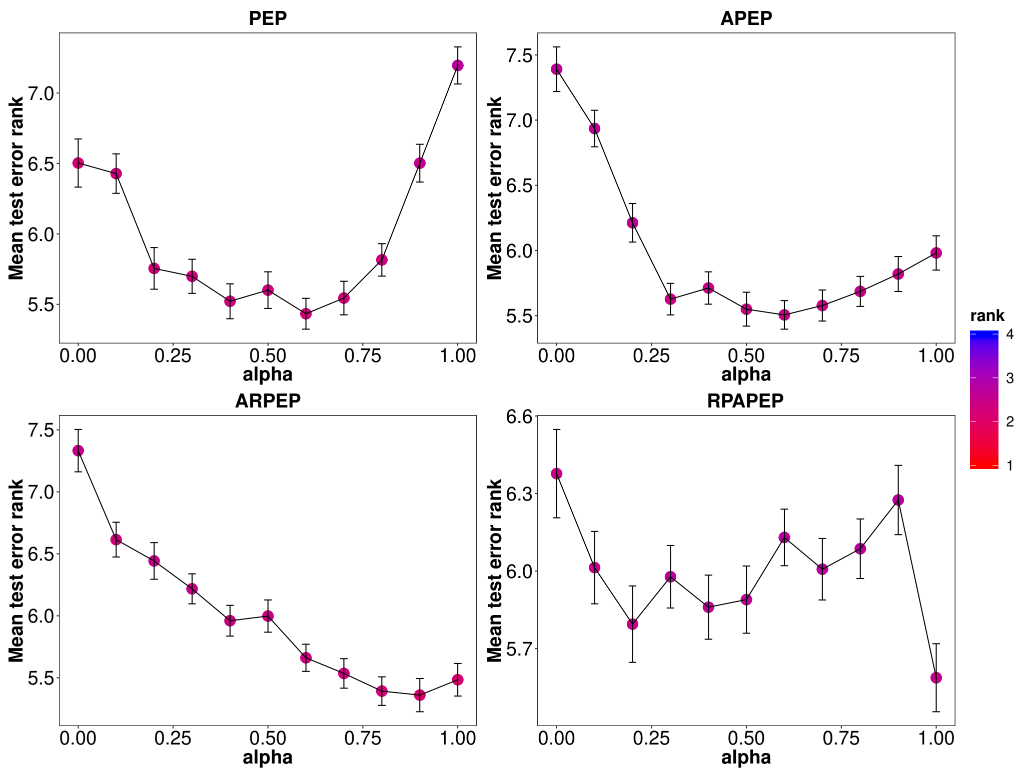

Figure D.8: Mean test error rank for different methods and different values of α. The color of the points indicates the average rank of the method compared with the others. Best seen in color.

The next result is from the Satellite dataset of the UCI repository. We show the performance as a function of the time in terms of the test error. It gives similar results as for the negative test log likelihood (shown in the main manuscript).

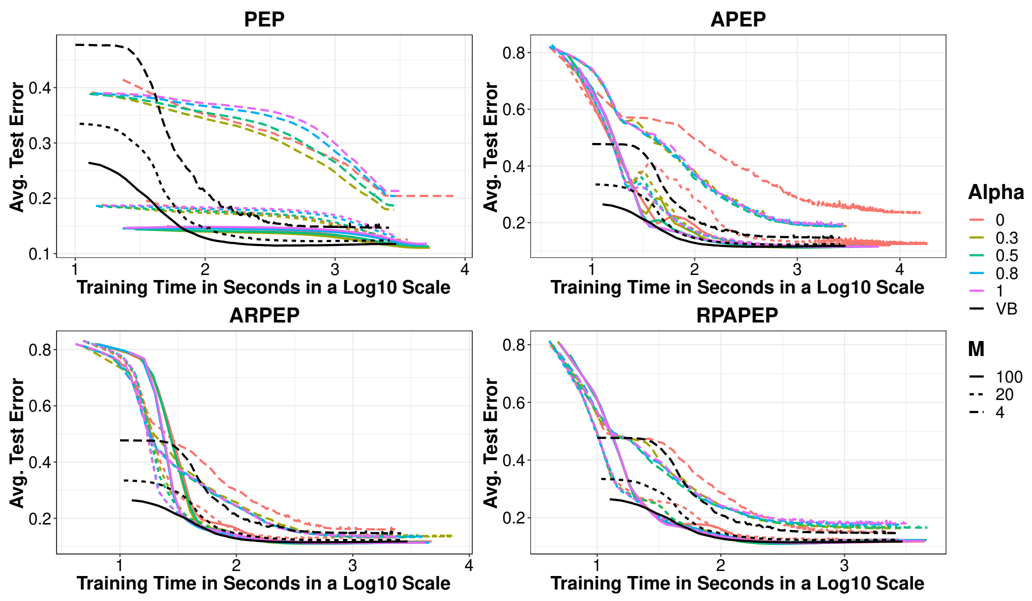

Figure D.9: Mean test error rank for different methods and different values of α. Best seen in color.

In Figure D.10 we show the results for the MNIST dataset in terms of the test error. Here we see that for the test error, values near $\alpha \to 0$ do not converge first, but instead intermediate values such as $\alpha = 0.5$ tend to arrive faster to the good solution.

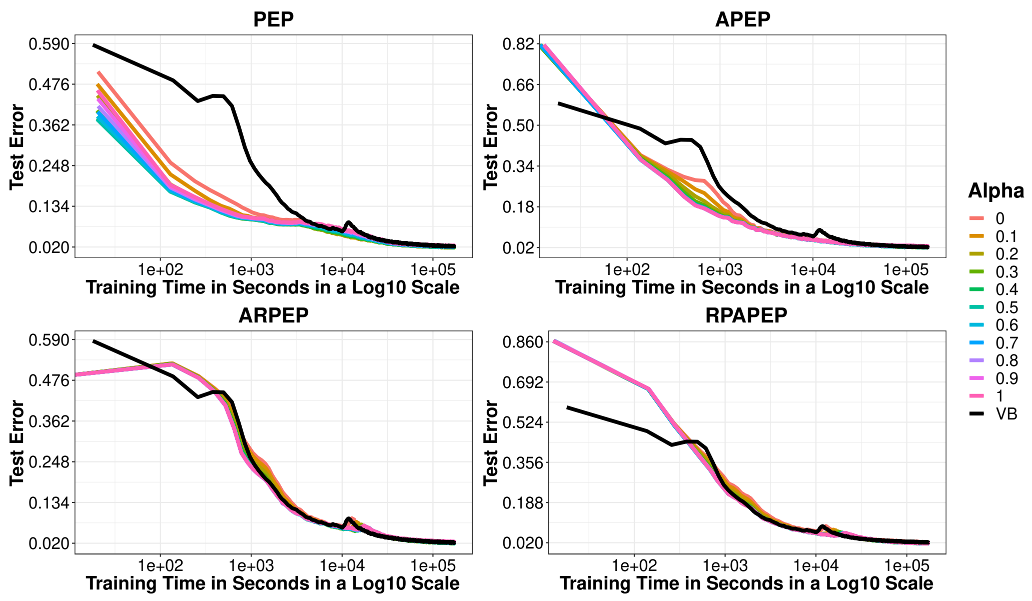

Figure D.10: Mean test error rank for different methods and different values of α for MNIST dataset. Best seen in color.

686 Regarding the Airline Delays dataset, we observe similar results when
687 talking of the test error as in MNIST. The results are shown in Figure D.11

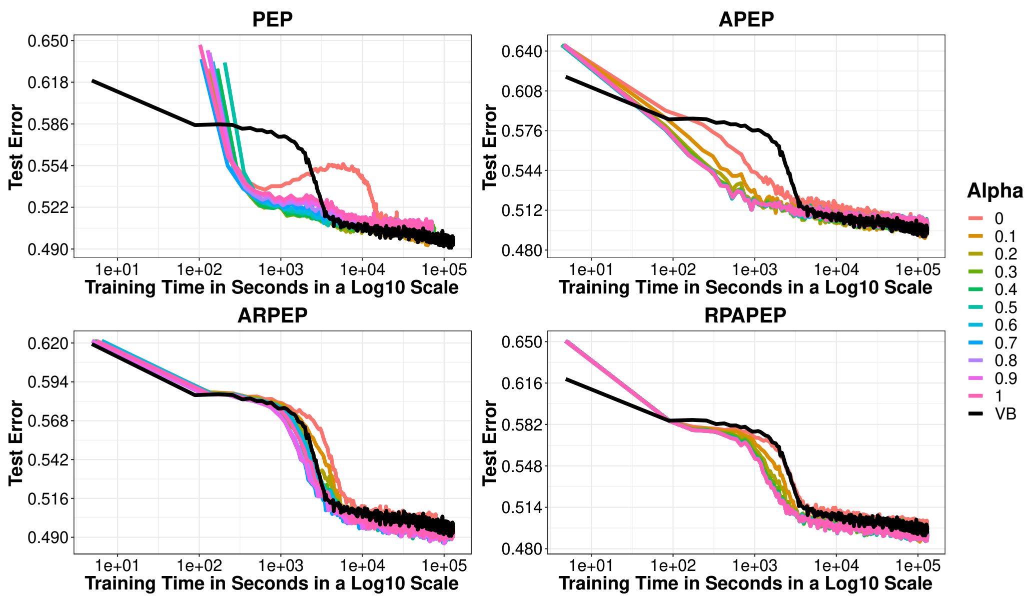

Figure D.11: Mean test error rank for different methods and different values of α for Airline Delays dataset. Best seen in color.

688 References

689 References

690 [1] C. E. Rasmussen, C. K. I. Williams, Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning), The MIT Press, 2006.

693 [2] C. K. I. Williams, D. Barber, Bayesian classification with Gaussian processes, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (1998) 1342–1351.

696 [3] H.-C. Kim, Z. Ghahramani, Bayesian Gaussian process classification

with the EM-EP algorithm, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (2006) 1948–1959.

[4] M. Girolami, S. Rogers, Variational Bayesian multinomial probit regression with Gaussian process priors, Neural Computation 18 (2006) 1790–1817.

[5] K. M. A. Chai, Variational multinomial logit gaussian process, Journal of Machine Learning Research 13 (2012) 1745–1808.

[6] J. Riihimäki, P. Jylänki, A. Vehtari, Nested expectation propagation for Gaussian process classification with a multinomial probit likelihood, Journal of Machine Learning Research 14 (2013) 75–109.

[7] J. Quiñonero-Candela, C. E. Rasmussen, A unifying view of sparse approximate Gaussian process regression, Journal of Machine Learning Research 6 (2005) 1939–1959.

[8] J. Hensman, A. Matthews, Z. Ghahramani, Scalable variational Gaussian process classification, in: Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, 2015.

[9] J. Hensman, A. G. Matthews, M. Filippone, Z. Ghahramani, MCMC for variationally sparse Gaussian processes, in: Advances in Neural Information Processing Systems 28, 2015, pp. 1648–1656.

[10] C. Villacampa-Calvo, D. Hernández-Lobato, Scalable Multi-Class Gaussian Process Classification using Expectation Propagation, in: International Conference on Machine Learning, 2017, pp. 3550–3559.

[11] D. Hernández-Lobato, J. M. Hernández-Lobato, Scalable Gaussian process classification via expectation propagation, in: Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, 2016, p. 168–176.

[12] T. D. Bui, J. Yan, R. E. Turner, A Unifying Framework for Gaussian Process Pseudo-Point Approximations using Power Expectation Propagation, Journal of Machine Learning Research 18 (2017) 1–72.

[13] T. Minka, Power EP, Technical Report, Microsoft Research, 2004.

[14] T. Minka, Divergence Measures and Message Passing, Technical Report, Microsoft Research, 2005.

[15] Y. Li, J. M. Hernández-Lobato, R. E. Turner, Stochastic expectation propagation, in: Advances in Neural Information Processing Systems 28, 2015, pp. 2323–2331.

[16] T. Bui, D. Hernández-Lobato, J. M. Hernández-Lobato, Y. Li, R. Turner, Deep Gaussian processes for regression using approximate expectation propagation, in: International Conference on Machine Learning, 2016, pp. 1472–1481.

[17] Y. Li, Y. Gal, Dropout Inference in Bayesian Neural Networks with Alpha-divergences, in: International Conference on Machine Learning, 2017, pp. 2052–2061.

[18] D. Hernández-Lobato, J. Hernández-Lobato, P. Dupont, Robust multi-class Gaussian process classification, in: Advances in Neural Information Processing Systems 24, 2011, pp. 280–288.

[19] C. J. Maddison, D. Tarlow, T. Minka, A* sampling, in: Advances in Neural Information Processing Systems, 2014, pp. 3086–3094.

[20] M. N. Gibbs, D. J. C. Mackay, Variational Gaussian process classifiers, IEEE Transactions on Neural Networks 11 (2000) 1458–1464.

[21] E. Snelson, Z. Ghahramani, Sparse Gaussian processes using pseudo-inputs, in: Advances in Neural Information Processing Systems, 2006, pp. 1257–1264.

[22] A. Naish-Guzman, S. Holden, The generalized FITC approximation, in: Advances in Neural Information Processing Systems, 2008, pp. 1057–1064.

[23] T. Minka, Expectation propagation for approximate Bayesian inference, in: Proceedings of the 17th Annual Conference on Uncertainty in Artificial Intelligence, 2001, pp. 362–36.

[24] M. Seeger, Expectation Propagation for Exponential Families, Technical Report, Department of EECS, University of California, Berkeley, 2006.

[25] M. Opper, C. Archambeau, The variational gaussian approximation revisited, Neural Computation 21 (2009) 786–792.

[26] M. Titsias, Variational learning of inducing variables in sparse Gaussian processes, in: International Conference on Artificial Intelligence and Statistics, 2009, pp. 567–574.

[27] E. Khan, S. Mohamed, K. P. Murphy, Fast bayesian inference for

non-conjugate gaussian process regression, in: Advances in Neural Information Processing Systems 25, 2012, pp. 3140–3148.

[28] A. G. d. G. Matthews, J. Hensman, R. Turner, Z. Ghahramani, On sparse variational methods and the kullback-leibler divergence between stochastic processes, in: Artificial Intelligence and Statistics, 2016, pp. 231–239.

[29] S. Amari, Differential-geometrical methods in statistics, volume 28, Springer Science & Business Media, 1985.

[30] H. Zhu, R. Rohwer, Information Geometric Measurements of Generalisation, Technical Report, Aston University, 1995.

[31] J. Hernández-Lobato, Y. Li, M. Rowland, T. Bui, D. Hernández-Lobato, R. Turner, Black-Box Alpha Divergence Minimization, in: International Conference on Machine Learning, 2016, pp. 1511–1520.

[32] T. Bui, Efficient Deterministic Approximate Bayesian Inference for Gaussian Process Models, Ph.D. thesis, 2017.

[33] A. Dezfouli, E. V. Bonilla, Scalable inference for Gaussian process models with black-box likelihoods, in: Advances in Neural Information Processing Systems, 2015, pp. 1414–1422.

[34] K. Krauth, E. V. Bonilla, K. Cutajar, M. Filippone, Autogp: Exploring the capabilities and limitations of Gaussian process models, in: Uncertainty in Artificial Intelligence, 2017.

[35] R. Sheth, Y. Wang, R. Khardon, Sparse variational inference for generalized GP models, in: International Conference on Machine Learning, 2015, pp. 1302–1311.

[36] M. Lichman, UCI machine learning repository, 2013. URL: `http://archive.ics.uci.edu/ml`.

[37] D. P. Kingma, J. Ba, ADAM: a method for stochastic optimization, in: Intrernational Conference on Learning Representations, 2015, pp. 1–15.

[38] M. Bauer, M. van der Wilk, C. E. Rasmussen, Understanding probabilistic sparse Gaussian process approximations, in: Advances in Neural Information Processing Systems 29, 2016, pp. 1533–1541.

[39] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (1998) 2278–2324.

[40] D. Hernández-Lobato, Prediction Based on Averages over Automatically Induced Learners: Ensemble Methods and Bayesian Techniques, Ph.D. thesis, 2010.

[41] K. B. Petersen, M. S. Pedersen, The Matrix Cookbook, Technical University of Denmark, 2012.

[42] E. Snelson, Flexible and efficient Gaussian process models for machine learning, Ph.D. thesis, 2007.

[43] C. M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), Springer-Verlag New York, Inc., 2006.