International Journal of Approximate Reasoning 150 (2022) 139–171

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

www.elsevier.com/locate/ijar

Alpha-divergence minimization for deep Gaussian processes

Carlos Villacampa-Calvo *, Gonzalo Hernández-Muñoz, Daniel Hernández-Lobato

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

ARTICLE INFO

Article history:
Received 4 April 2022
Received in revised form 4 August 2022
Accepted 5 August 2022
Available online 22 August 2022

Keywords:
Deep Gaussian processes
Expectation propagation
α-divergences
Approximate inference
Variational inference

ABSTRACT

This paper proposes the minimization of α-divergences for approximate inference in the context of deep Gaussian processes (DGPs). The proposed method can be considered as a generalization of variational inference (VI) and expectation propagation (EP), two previously used methods for approximate inference in DGPs. Both VI and EP are based on the minimization of the Kullback-Leibler divergence. The proposed method is based on a scalable version of power expectation propagation, a method that introduces an extra parameter α that specifies the targeted α-divergence to be optimized. In particular, such a method can recover the VI solution when $\alpha \to 0$ and the EP solution when $\alpha \to 1$. An exhaustive experimental evaluation shows that the minimization of α-divergences via the proposed method is feasible in DGPs and that choosing intermediate values of the α parameter between 0 and 1 can give better results in some problems. This means that one can improve the results of VI and EP when training DGPs. Importantly, the proposed method allows for stochastic optimization techniques, making it able to address datasets with several millions of instances.

1. Introduction

Neural Networks (NNs) have become very popular recently due to the good results obtained in a wide variety of problems. These models, trained via back-propagation, have significantly improved the state-of-the-art in supervised learning tasks [1]. Moreover, they have been specifically designed to take advantage of underlying structure on the input data. This is the case, *e.g.*, of convolutional NNs or long-short term memory NNs [2,3]. These models can also be trained on GPUs, which significantly reduces the total training time. Nevertheless, in spite of these advantages, NNs have drawbacks, such as over-fitting due to the high number of parameters, or the lack of a confidence estimate on the predicted outputs given the input data [4]. More precisely, standard NNs only generate point-estimate predictions and do not give any information about the certainty of such outcome. Critically, there are applications in which such uncertainty estimates are required including, *e.g.*, computer vision or health related prediction [4].

Gaussian processes (GPs) are, on the other hand, non-parametric machine learning models that have the advantage of outputting a predictive distribution [5]. In other words, they provide an estimation of the uncertainty associated to each prediction. GPs can also be seen as NNs with an infinite number of hidden units [6]. A main limitation of GPs is, however,

* Corresponding author.
E-mail addresses: carlos.villacampa@uam.es (C. Villacampa-Calvo), gonzalo.hernandez@uam.es (G. Hernández-Muñoz), daniel.hernandez@uam.es (D. Hernández-Lobato).

https://doi.org/10.1016/j.ijar.2022.08.003

that they do not scale well with the number of training points. More precisely, they have a cost that is in $\mathcal{O}(N^3)$, where N is the size of the training set. Popular techniques that allow GPs to scale to larger datasets include the use of a set of $M \ll N$ inducing points that summarize the observed data [7,8]. When these techniques are used, the computational cost is reduced to $\mathcal{O}(NM^2)$. Moreover, in this case, GPs can also be trained efficiently using stochastic optimization techniques that allow to address datasets with millions of data-points [9]. Another limitation of GPs is, however, that their expressiveness strongly depends on the kernel or covariance function [5]. Some authors have addressed this limitation by building more complex covariance functions, *e.g.*, by combining several kernels [10,11] or by using a deep neural network to encode the covariance function [12,13].

An alternative to increase the flexibility of GPs is to consider a model given by the concatenation of several GPs, in the same spirit as a deep NN, where the output of one layer is used as the input of the next layer. Such a model is known in the literature as a deep GP [14]. Deep GPs are a generalization of GPs that maintain their main interesting properties such as having good generalization performance and producing accurate predictive distributions that account for output uncertainty [15–18]. These models are more flexible than standard GPs and can, *e.g.*, model functions with different levels of smoothness in the input space, input dependent noise, or provide non-Gaussian predictive distributions [16,19]. Furthermore, there is evidence showing that they provide superior empirical results to those of GPs and to those of Bayesian treatments of NNs to account for prediction uncertainty [16,17]. Nevertheless, in spite of these advantages, inference is more challenging in deep GPs than in GPs. More precisely, in the second and following layers of the deep GP the inputs are random variables specified by the predictive distribution of the previous layer. It is well-known that the output distribution of a GP with random inputs is no-longer Gaussian [16]. Therefore, the consequence is that the output distribution in the second and following layers of the deep GP is no longer Gaussian. To overcome these difficulties several approaches for approximate inference in deep GPs have been proposed in the literature, some of them based on variational inference (VI) [14,15,17] and some based on expectation propagation (EP) [16,19].

VI and EP minimize the Kullback-Leibler (KL) divergence between the posterior distribution of the model's latent variables p and an approximate posterior distribution q [20]. There has been, however, recent work in the literature showing that one can obtain better approximate predictive distributions by optimizing a generalization of the KL-divergence known as the α-divergence [21]. For this, a modification of the power expectation propagation algorithm has been considered [22]. The α-divergence has an extra parameter, $\alpha \in (0, 1)$, that interpolates between the direct KL-divergence used in VI, *i.e.*, $\mathrm{KL}(q||p)$, retrieved when $\alpha \to 0$, and the reversed KL-divergence used in EP, *i.e.*, $\mathrm{KL}(p||q)$, obtained when $\alpha \to 1$. Importantly, several results in the literature show that one can obtain more accurate predictive distributions in the context of standard (swallow) GPs using intermediate values of alpha, *e.g.*, $\alpha = 0.5$ [23,24]. These results motivate the minimization of α-divergences in the context of deep GPs, a natural generalization of GPs. Specifically, the minimization of α-divergences for approximate inference may lead to more accurate predictive distributions, which becomes crucial in some specific applications related, for example, with health-care or computer vision [4]. Accurate predictive distributions can also be useful in the context of active learning, where one chooses the data instance to label and introduce in the training set that is expected to improve the most the model's performance. Those are precisely the instances about whose target value the current model is more uncertain about [24,25].

In this work, we propose to optimize α-divergences in the context of deep GPs, for approximate inference. This is a very challenging task since a deep GP is a complex model composed of several layers of stochastic processes. These processes have an intractable predictive distribution due to the randomness in the inputs of the second and following layers. However, with the aforementioned goal, we describe an efficient algorithm based on power EP that allows for stochastic optimization and hence can handle datasets with millions of training instances. We evaluate such algorithm in several experiments, considering different architectures for the deep GP network. The results obtained show that the optimization of α-divergences is feasible in the context of deep GPs and that varying the value of α can lead sometimes to better results than when using EP of VI, the most common methods for approximate inference with deep GPs. Summing up, the main contributions of our paper are:

1. The first feasible algorithm to train deep GPs by optimizing α-divergences. This method generalizes VI and EP, the two most popular approaches for training these models until now [16,17,19].
2. An exhaustive experimental evaluation that analyzes the influence of the α parameter on the performance of deep GPs. This evaluation considers regression, binary classification and multi-class problems.
3. We give empirical evidence that supports that in some situations one can obtain better results than those of VI or EP when training deep GPs using our proposed method. This is important for the community since deep GPs trained via VI often provide state-of-the-art prediction results [17].

2. Gaussian processes

We define a supervised learning problem as a training set of N d-dimensional data points $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ and their associated output values $\mathbf{y} = (y_1, \ldots, y_N)^\mathrm{T}$. If we let $y_i \in \mathbb{R}$, we deal with regression problems, while for classification problems we consider that y_i can take values from a set of different categories [26]. Let us focus on regression for simplicity and consider that y_i is a function of the inputs plus some additive Gaussian noise, *i.e.*, $y_i = f(\mathbf{x}_i) + \epsilon_i$. That is, $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, with $\mathcal{N}(0, \sigma^2)$ a Gaussian distribution with zero mean and σ^2 variance.

A Gaussian process (GP) model tries to infer $f(\cdot)$ using a Bayesian approach. For that, it defines a GP prior $p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|m(\mathbf{X}), k(\mathbf{X},\mathbf{X}))$, where $\mathcal{N}(\cdot|\boldsymbol{\mu},\boldsymbol{\Sigma})$ denotes the p.d.f. of a multivariate Gaussian distribution with mean $\boldsymbol{\mu}$, and covariance matrix $\boldsymbol{\Sigma}$; $\mathbf{f} = (f(\mathbf{x}_1),\ldots,f(\mathbf{x}_N))^{\mathrm{T}}$ is a vector with the process values at each point in $\mathbf{X}$; $m(\mathbf{X}) \in \mathbb{R}^N$ is the result of applying $m(\cdot)$ to each point in $\mathbf{X}$; $k(\mathbf{X},\mathbf{X}) \in \mathbb{R}^{N\times N}$ is the result of applying $k(\cdot,\cdot)$ to each pair of points in $\mathbf{X}$; finally, m$(\cdot)$ and $k(\cdot,\cdot)$ are the mean and covariance function of the GP, respectively [5]. Since we assume that the observations are contaminated with independent additive Gaussian noise, we set a Gaussian likelihood $p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^N p(y_i|f_i)$ that factorizes over the inputs, where $f_i = f(\mathbf{x}_i)$ and $p(y_i|f_i) = \mathcal{N}(y_i|f_i,\sigma^2)$, with σ^2 the variance of the additive noise ϵ_i associated to each data instance.

We can then apply Bayes rule to obtain a posterior distribution [5]

$$p(\mathbf{f}|\mathbf{y}) = \frac{p(\mathbf{f})p(\mathbf{y}|\mathbf{f})}{p(\mathbf{y})}\,. \tag{1}$$

The quantity in the denominator is a normalization constant called the marginal likelihood that can be maximized in order to find good values for the model hyper-parameters. These include often the variance of the additive noise and the parameters of the covariance function $k(\cdot,\cdot)$ such as the amplitude and the length-scales. See [5] for further details. The predictive distribution for $y_\star$ associated to a new point $\mathbf{x}_\star$ is

$$p(y_\star|\mathbf{y}) = \int_D p(y_\star|f_\star)p(f_\star|\mathbf{f})p(\mathbf{f}|\mathbf{y})d\mathbf{f}df_\star = \mathcal{N}(y_\star|m_\star, v_\star)\,, \tag{2}$$

where $f_\star = f(\mathbf{x}_\star)$, $D = (-\infty,\infty)$ and

$$m_\star = \mathbf{k}_\star^{\mathrm{T}}\left(\mathbf{K}+\sigma^2\mathbf{I}\right)^{-1}\mathbf{y}\,, \qquad v_\star = k_\star - \mathbf{k}_\star^{\mathrm{T}}\left(\mathbf{K}+\sigma^2\mathbf{I}\right)^{-1}\mathbf{k}_\star + \sigma^2\,, \tag{3}$$

with $k_\star = k(\mathbf{x}_\star,\mathbf{x}_\star)$, $\mathbf{k}_\star^{\mathrm{T}} = (k(\mathbf{x}_\star,\mathbf{x}_1),\ldots,k(\mathbf{x}_\star,\mathbf{x}_N))^{\mathrm{T}}$ and $\mathbf{K}$ a matrix with entries $K_{i,j} = k(\mathbf{x}_i,\mathbf{x}_j)$. This is the standard predictive distribution of a GP. See [5] for further details.

In classification problems, however $p(y_i|f_i)$ is non-Gaussian, making exact inference infeasible [5]. In particular, the posterior (1) is no longer Gaussian since the likelihood factors are non-Gaussian. The consequence is that the marginal likelihood $p(\mathbf{y})$ is intractable to compute and hence also (1). Therefore, methods for approximate inference will be needed for classification, such as the Laplace approximation [27], expectation propagation [28] or variational inference [29]. In general, these methods provide a Gaussian approximation for (1), which can then be used in (2). All these methods have a cost that is in $\mathcal{O}(N^3)$, where N is the size of the training set, since they need to invert the covariance matrix $\mathbf{K}$.

2.1. Sparse Gaussian processes

As described before, the cost of GPs is in $\mathcal{O}(N^3)$, which means that in practice GPs can only address problems with a few thousand data instances [5]. In order to reduce this cost, a typical approach is to use the so-called sparse approximations. These approximations rely on a new set $M \ll N$ of inducing points $\mathbf{Z} = \{\mathbf{z}_m\}_{m=1}^M$, located on the same space as the observed data, and their corresponding output process values $\mathbf{u} = (f(\mathbf{z}_1),\ldots,f(\mathbf{z}_M))^{\mathrm{T}}$ that summarize the observed data [8]. The GP prior for $\mathbf{u}$ is $p(\mathbf{u}) = \mathcal{N}(\mathbf{u}|m(\mathbf{Z}),k(\mathbf{Z},\mathbf{Z}))$.

There are two popular approaches for making inference in sparse GPs. The first one assumes that $p(\mathbf{f}|\mathbf{u})$ factorizes over the data instances, *i.e.*, $p(\mathbf{f}|\mathbf{u}) = \prod_{i=1}^N p(f_i|\mathbf{u})$. This can be shown to be equivalent to using an approximate GP prior for $\mathbf{f}$. Namely, $p(\mathbf{f}) = \int p(\mathbf{f}|\mathbf{u})p(\mathbf{u})d\mathbf{u} \approx \int \prod_{i=1}^N p(f_i|\mathbf{u})p(\mathbf{u})d\mathbf{u} = \mathcal{N}(\mathbf{f}|m(\mathbf{X}),\mathbf{Q})$, where $\mathbf{Q}$ has some structure that can be exploited for fast matrix inversion. This approximation is called the Fully Independent Training Conditional (FITC) and results in a training cost in $\mathcal{O}(NM^2)$. See [30] for further details.

The other approach consists in carrying out variational inference in the GP model assuming a constrained approximation for the posterior $p(\mathbf{f},\mathbf{u}|\mathbf{y})$, $q(\mathbf{f},\mathbf{u})$. Specifically, $q(\mathbf{f},\mathbf{u}) = p(\mathbf{f}|\mathbf{u})q(\mathbf{u})$, where $p(\mathbf{f}|\mathbf{u})$ is fixed and given by the GP predictive distribution and $q(\mathbf{u})$ is a tunable multivariate Gaussian whose parameters are adjusted by maximizing a lower bound on the marginal likelihood [7]. In this case, no factorization assumption is made about $p(\mathbf{f}|\mathbf{u})$. Again, the computational cost of this method is in $\mathcal{O}(NM^2)$.

For prediction, in both cases, one carries out approximate inference about $\mathbf{u}$ to obtain an approximate distribution $q(\mathbf{u})$ targeting $p(\mathbf{u}|\mathbf{y})$. Given $q(\mathbf{u}) = \mathcal{N}(\mathbf{u}|\mathbf{m},\mathbf{S})$ the predictive distribution for the value $y_\star$ associated to a new instance $\mathbf{x}_\star$ is given by:

$$p(y_\star|\mathbf{y}) = \int_D p(y_\star|f_\star)p(f_\star|\mathbf{u})q(\mathbf{u})d\mathbf{u}df_\star = \mathcal{N}(y_\star|m_\star, v_\star)\,, \tag{4}$$

where $D = (-\infty,\infty)$ and

$$m_\star = \mathbf{k}_{f_\star,\mathbf{u}}^{\mathrm{T}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{m}\,, \qquad v_\star = k_{f_\star,f_\star} - \mathbf{k}_{f_\star,\mathbf{u}}^{\mathrm{T}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\left(\mathbf{K}_{\mathbf{u},\mathbf{u}}-\mathbf{S}\right)\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{k}_{f_\star,\mathbf{u}} + \sigma^2\,, \tag{5}$$

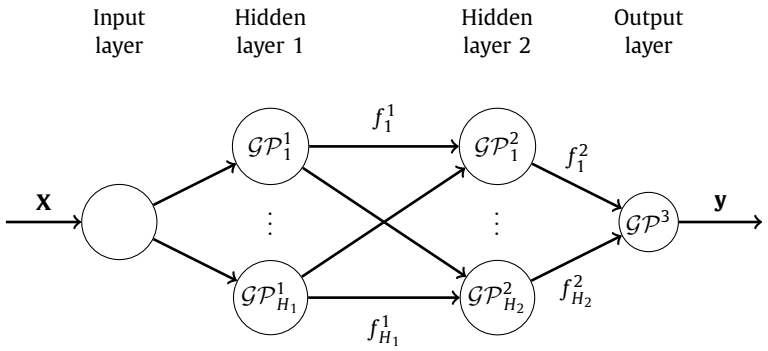

Fig. 1. Deep GP architecture with two hidden fully connected layers and H_l hidden units at each layer. In this case, we assume that the problem is regression or binary classification. Hence, there is a single GP in the last layer.

with $k_{f_\star,f_\star} = k(\mathbf{x}_\star, \mathbf{x}_\star)$, $\mathbf{k}_{f_\star,\mathbf{u}}$ a vector with the covariances between $f_\star$ and $\mathbf{u}$ and $\mathbf{K}_{\mathbf{u},\mathbf{u}}$ a matrix with the covariances among each entry in $\mathbf{u}$. See [7] for further details.

In both approaches the locations of the inducing points $\mathbf{Z}$ are optimized during training by maximizing an estimate of the marginal likelihood. This method works well in practice and only a few inducing points are needed to provide similar results to those of the full GP [7,8]. Furthermore, both methods can be used in the context of classification problems. In this case, expectation propagation is used in combination of the FITC approximation [31]. The variational approach can handle classification problems using one-dimensional quadratures [32].

Finally, even though the variational approach seems better since it does not rely on an independence assumption on $p(\mathbf{f}|\mathbf{u})$, both approaches are found to perform similarly empirically. In particular, the variational objective, although better, is prone to bad local optima and is more difficult to optimize, which means that in practice it gives similar results to those of the FITC approximation [33].

3. Deep Gaussian processes

Deep Gaussian Processes (DGPs) are defined as a composition of GPs, where the output of a GP in one hidden layer is set to be the input of the GPs in the next layer [14]. See Fig. 1 for a DGP architecture example with two hidden layers and three layers in total. Typically, the last layer has dimension equal to one and hence only contains one GP. This is the case, *e.g.*, for binary and regression problems. However, if multi-output regression problems or multi-class classification problems are considered, several GPs can be included in this last layer. Fully connected architectures are often used in a DGP network.

Let L denote the number of layers and H_l the number of hidden units at the l-th layer. If $L = 1$ and $H_1 = 1$, we are back to the single GP model. Independence among GPs of the same layer is often assumed to simplify and speed-up the computations. Furthermore, for scalability reasons, sparse GP approximations are considered for each GP of the DGP network [16,17]. Let $\mathbf{f}_h^l$ be the vector of process values at layer l and unit h associated to the training instances. Therefore, $\mathbf{f}_h^l \in \mathbb{R}^N$. Similarly, let $\mathbf{u}_h^l$ be the process values at layer l and unit h associated to the inducing points of that layer and unit, $\mathbf{Z}_h^l$. In this case, $\mathbf{u}_h^l \in \mathbb{R}^M$. Consider the joint distribution of the targets $\mathbf{y}$, the outputs of the GPs of each layer $\mathbf{f}^l = (\mathbf{f}_1^l, \ldots, \mathbf{f}_{H_l}^l)$, and the inducing outputs associated to each GP of each layer $\mathbf{u}^l = (\mathbf{u}_1^l, \ldots, \mathbf{u}_{H_l}^l)$, for $l = 1, \ldots, L$. This distribution is:

$$p(\mathbf{y}, \{\mathbf{f}^l, \mathbf{u}^l\}_{l=1}^L) = \left[\prod_{l=1}^{L}\prod_{h=1}^{H_l} p(\mathbf{f}_h^l|\mathbf{u}_h^l, \mathbf{f}^{l-1})p(\mathbf{u}_h^l)\right]\prod_{i=1}^{N} p(y_i|\mathbf{f}_i^L)\,, \tag{6}$$

where $\mathbf{f}^0 = \mathbf{X}$ and $\mathbf{f}_i^L$ is the vector of process values at the last layer corresponding to the i-th instance. Therefore, $\mathbf{f}_i^L \in \mathbb{R}^{H_L}$. Note that in (6) we have two types of factors. Namely, the likelihood factor $\prod_{i=1}^N p(y_i|\mathbf{f}_i^L)$ and the DGP prior factor $\prod_{l=1}^L \prod_{h=1}^{H_l} p(\mathbf{f}_h^l|\mathbf{u}_h^l; \mathbf{f}^{l-1})p(\mathbf{u}_h^l)$ [17].

Critically, in order to compute the predictive distribution at layer l, *i.e.*, $p(f_h^l|\mathbf{u}_h^l, \mathbf{f}^{l-1})$ we require the marginalization of the inputs to that layer, $\mathbf{f}^{l-1}$, given by the predictive distribution for $\mathbf{f}^{l-1}$ specified by the previous layer. The predictive distribution of the previous layer is deterministic only for the input data, *i.e.*, $\mathbf{f}^0 = \mathbf{X}$. Therefore, exact inference is infeasible in a DGP model for $L > 1$, even for regression problems. In consequence, approximate inference is needed and, importantly, it is significantly more challenging to carry out than in other models including, *e.g.*, standard GPs with tractable Gaussian predictive distributions.

There are several methods for approximate inference in DGPs that have been proposed in the literature. These include using approximate expectation propagation (EP) [16,19] or variational inference (VI) [14,17]. Both EP and VI find an approximate posterior distribution $q(\{\mathbf{f}^l, \mathbf{u}^l\}_{l=1}^L)$ that looks as similar as possible as the true posterior $p(\{\mathbf{f}^l, \mathbf{u}^l\}_{l=1}^L|\mathbf{y})$, obtained by

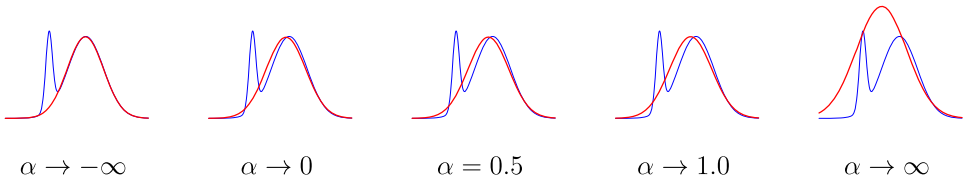

Fig. 2. Illustration of the resulting approximate distribution q shown in red, obtained by minimizing the corresponding α-divergence with respect to a target distribution p that is a mixture of two Gaussians, shown in blue. Both distributions are unnormalized probability distributions. Reproduced from [34]. Best seen in color. (For interpretation of the colors in the figures, the reader is referred to the web version of this article.)

normalizing the joint distribution of the observed and latent variables of the model, (6). VI maximizes a lower bound of the log-marginal likelihood of the model. This is equivalent to minimizing the global Kullback-Leibler (KL) divergence between q and the exact posterior. EP, on the other hand, replaces the likelihood factors of the model by approximate factors that are constrained to be Gaussian [16,19,28,34]. The product of these approximate factors and the prior distribution results in the posterior approximation q. Generally speaking, EP iteratively tunes q by locally minimizing the reversed KL-divergence between each tilted distribution (*i.e.*, the approximate distribution q where we have removed the influence of an approximate likelihood factor and incorporated the corresponding exact likelihood factor) and the approximate distribution q, until convergence.

In this paper we will focus on a variant of EP called Power Expectation Propagation (PEP) that allows for the minimization of a more general family of divergences known as the α-divergences [22,35]. The α-divergence includes a parameter $\alpha \in (0, 1)$ that can be adjusted leading to different divergences between probability distributions. Since the α-divergence includes as particular cases the KL-divergences used by VI and EP, we expect to obtain better results than when these two methods are used. The next section describes how to minimize α-divergences in the context of DGPs, which is a challenging task.

4. Alpha-divergence minimization for DGPs

As we have introduced in the previous section, in this paper we are interested in the minimization of the α-divergence, as described in [35]. This is a family of divergences that generalize the KL divergence [34]. The considered α-divergence between a target probability distributions p and an approximate distribution q of a random variable θ is defined as:

$$D_\alpha[p \parallel q] = \frac{1}{\alpha(1-\alpha)} \left(1 - \int p(\theta)^\alpha q(\theta)^{1-\alpha} d\theta\right), \tag{7}$$

where $\alpha \in \mathbb{R} \backslash \{0, 1\}$. This is the definition given in [35]. Note that there are other definitions of the α-divergence, such as the one introduced in [36]. We can obtain such another definition from (7) simply by replacing α with $(1-\alpha)/2$. We have decided to use the definition given in [35] because it is the one often used in the context of approximate Bayesian inference [21,37,38].

Interestingly, it can be shown that $D_0[p \parallel q] = \lim_{\alpha \to 0} D_\alpha[p \parallel q] = \mathrm{KL}[q \parallel p]$, which is the divergence that we are effectively minimizing in VI, and $D_1[p \parallel q] = \lim_{\alpha \to 1} D_\alpha[p \parallel q] = \mathrm{KL}[p \parallel q]$, which is the divergence minimized in each of the EP updates of the approximate factors. Thus, we can alternate between these two inference schemes by varying the value of α. Another case worth mentioning is when $\alpha = 0.5$, which is the only α-divergence that is symmetric in p and q and it is called the Hellinger distance [21]. In particular, it is possible to show that $D_{0.5}[p \parallel q] = 2 \int \left(\sqrt{p(\theta)} - \sqrt{q(\theta)}\right)^2 d\theta$ [34].

The value of the α parameter in the α-divergence has potentially a strong impact on the resulting approximate distribution q. More precisely, for $\alpha \leq 0$, the a-divergence enforces q to have low density wherever p has low density (hence, it could be considered as zero-forcing). On the other hand, when $\alpha \geq 1$, it can be said that the divergence is inclusive, as indicated in [39]. In this case, the divergence enforces $q > 0$ wherever $p > 0$, hence avoiding zero probability density in regions of the space in which p has high density. See Fig. 2 for an illustration where the distribution need not be normalized.

When α takes values in the interval $(0, 1)$, the resulting q distribution is intermediate between the two extreme possibilities that we have seen so far. In particular, when $\alpha \to 0$, we should expect that the approximate distribution q is more centered in the main mode of p. By contrast, when $a \to 1$, q is expected to cover the target distribution p more and capture more modes. This is illustrated in Fig. 3. This figure shows the results obtained when approximating a target Gaussian distribution p with strong dependencies using a factorizing Gaussian q. We can observe that the results obtained for $\alpha \to 0$ (zero-forcing) and for $\alpha \to 1$ (inclusive behavior) look very different. These results would correspond to VI and EP, respectively. Note that optimizing $D_1[p||q] = \mathrm{KL}[p||q]$ results in q having high density where p has high density (inclusive behavior). By contrast, optimizing $D_0[p||q] = \mathrm{KL}[q||p]$ results in q having low density where p has low density (zero-forcing). Therefore, by adjusting the α parameter we expect to find the best compromise between zero-forcing and more inclusive approximate distributions. Appendix A has the results obtained when minimizing the α-divergence for intermediate values of α. There it is shown that one can interpolate between the zero-forcing and the inclusive behavior simply by choosing α in $(0, 1)$. Importantly, intermediate values of α in $(0, 1)$ may try to capture multiple modes, but will ignore those modes

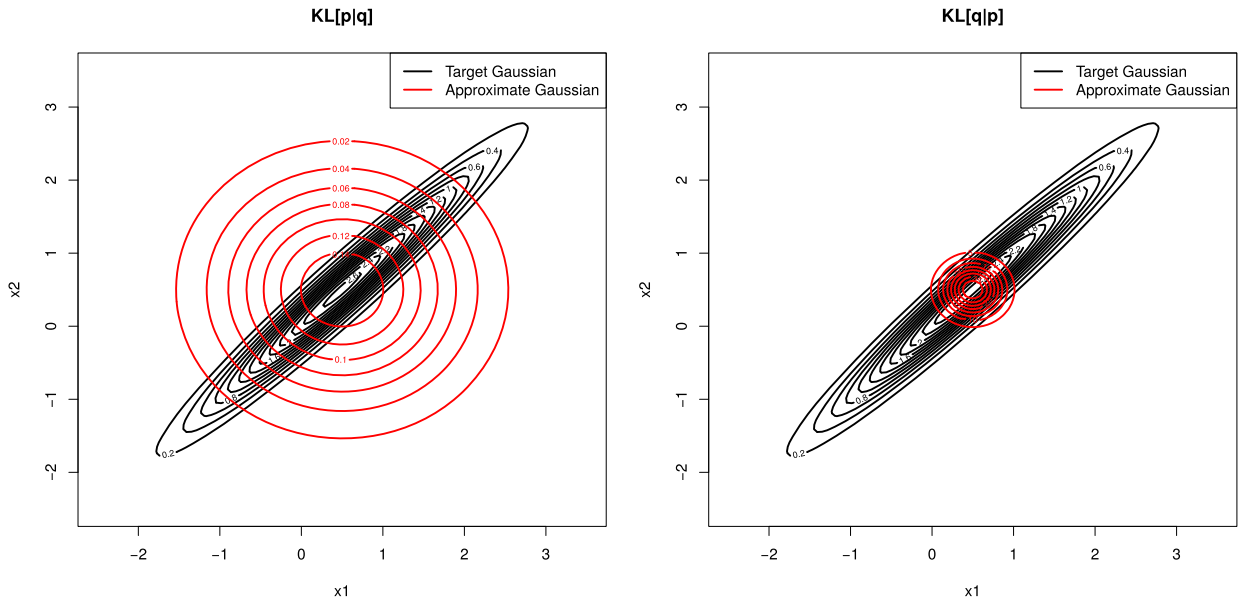

Fig. 3. Level curves of the p.d.f. that results when approximating a Gaussian distribution with strong dependencies using a factorizing Gaussian. (left) Optimization of KL$[p||q]$, obtained when $\alpha \to 1$. (right) Optimization of KL$[q||p]$, obtained when $\alpha \to 0$. We can observe that KL$[p||q]$ results in q having high density where p has high density. By contrast, KL$[q||p]$ results in q having low density where p has low density. Reproduced from [20]. Best seen in color.

that are too far away from the main mass of the target distribution p. How far, will depend on the value of α. Therefore, the minimization of α-divergences brings extra flexibility to the problem of approximate inference and may be useful to better capture the properties of the posterior distribution than EP or VI.

In the following subsection we will introduce the power expectation propagation algorithm (PEP), and how to apply it in the context of DGPs for minimizing α-divergences. Next, we will explain the specific details that are needed for a feasible and efficient implementation of PEP that can address problems with a large number of instances. Importantly, applying PEP in the context of DGPs is not trivial and requires specific modifications for making it feasible. A direct application of PEP will fail due to several intractabilities. We carefully address them to obtain a feasible PEP algorithm in the context of DGPs.

4.1. Power expectation propagation for DGPs

As we mentioned in Section 3, exact inference in the context of DGPs is intractable. Therefore, we have to rely on approximate inference techniques to approximate the posterior distribution that results from normalizing the joint distribution of the observed and latent variables of the model, given in (6). A popular method for this task is power expectation propagation (PEP) [22]. In the context of DGPs, PEP works by replacing each likelihood factor, that may have a complicated form, by a corresponding parametric factor that belongs to the exponential family. Namely, a distribution of the form

$$p(\mathbf{x}|\boldsymbol{\theta}) = \exp(\boldsymbol{\theta}^{\mathrm{T}}\mathbf{s}(\mathbf{x}) - g(\boldsymbol{\theta}))\,, \tag{8}$$

$$g(\boldsymbol{\theta}) = \log \int \exp(\boldsymbol{\theta}^{\mathrm{T}}\mathbf{s}(\mathbf{x}))d\mathbf{x}\,, \tag{9}$$

where $\boldsymbol{\theta}$ are the natural parameters, $\mathbf{s}(\mathbf{x})$ is a vector of sufficient statistics and $g(\boldsymbol{\theta})$ is the log partition function [40]. See [16,19] for further details about similar approximations in the context of expectation propagation (EP) [28]. More precisely, in PEP for DGPs we consider an approximate distribution q targeting the exact posterior, $p(\{\mathbf{f}^l, \mathbf{u}^l\}_{l=1}^L|\mathbf{y})$, that is similar to the one used in [17]. That is,

$$q(\{\mathbf{f}^l, \mathbf{u}^l\}_{l=1}^L) = \prod_{l=1}^{L}\prod_{h=1}^{H_l} p(\mathbf{f}_h^l|\mathbf{u}_h^l, \mathbf{f}^{l-1})q(\mathbf{u}_h^l)\,, \tag{10}$$

where L is the number of layers of the DGP and H_l, the number of units of each layer. Furthermore, $q(\mathbf{u}_h^l) \propto p(\mathbf{u}_h^l)\prod_{i=1}^N \tilde{\phi}_i(\mathbf{u}_h^l)$ with $p(\mathbf{u}_h^l) = \mathcal{N}(\mathbf{u}_h^l|m(\mathbf{Z}), k(\mathbf{Z},\mathbf{Z}))$ the corresponding GP prior for the process values at the inducing points $\mathbf{Z}$ (note that we have eliminated the dependence of $\mathbf{Z}$ on the layer and the unit to simplify the notation). Importantly, the factors $p(\mathbf{f}_h^l|\mathbf{u}_h^l, \mathbf{f}^{l-1})$ in (10) are fixed. Only each $q(\mathbf{u}_h^l)$ will be updated. Therefore, the PEP approximate distribution

q is obtained by replacing each likelihood factor $p(y_i|\mathbf{f}_i^L)$ by a product of approximate factors $\prod_{l=1}^{L}\prod_{h=1}^{H_l}\tilde{\phi}_i(\mathbf{u}_h^l)$, where $\tilde{\phi}_i(\mathbf{u}_h^l) \propto \mathcal{N}(\mathbf{u}_h^l|\boldsymbol{\mu}_h^l, \boldsymbol{\Sigma}_h^l)$, with $\boldsymbol{\mu}_h^l$ and $\boldsymbol{\Sigma}_h^l$ tunable parameters. PEP will iteratively refine the parameters of each of these approximate factors $\tilde{\phi}_i$ so that their influence on $\{\mathbf{u}^l\}_{l=1}^L$ is similar to that of the corresponding exact likelihood factor $p(y_i|\mathbf{f}^L)$. With that goal, first, for each hidden layer l and hidden unit h^l, PEP defines a cavity distribution, $q^{\setminus i}(\mathbf{u}_h^l)$, by removing the i-th approximate factor from $q(\mathbf{u}_h^l)$ to the power of α:

$$q^{\setminus i}(\mathbf{u}_h^l) \propto \frac{q(\mathbf{u}_h^l)}{\tilde{\phi}_i(\mathbf{u}_h^l)^\alpha}\,, \tag{11}$$

where $\propto$ means proportional to and we use the super index $\setminus i$ to denote the cavity distribution. Because all factors are Gaussian, the cavity distribution is also Gaussian. Later it incorporates the true likelihood factor to the power of α to obtain the tilted distribution for the i-th data instance:

$$\hat{p}_i(\{\mathbf{f}^l, \mathbf{u}^l\}_{l=1}^L) = Z_i^{-1} p(y_i|\mathbf{f}_i^L)^\alpha \prod_{l=1}^{L}\prod_{h=1}^{H_l} p(\mathbf{f}_h^l|\mathbf{u}_h^l, \mathbf{f}^{l-1}) q^{\setminus i}(\mathbf{u}_h^l)\,, \tag{12}$$

where Z_i is just a normalization constant. The refined approximate factor to the power of α will be obtained by minimizing the KL-divergence between the tilted distribution and q, *i.e.* $\mathrm{KL}[\hat{p}_i(\{\mathbf{f}^l, \mathbf{u}^l\}_{l=1}^L) \parallel q(\{\mathbf{f}^l, \mathbf{u}^l\}_{l=1}^L)]$. As we assume a particular form for q, where only each $q(\mathbf{u}_h^l)$ can be adjusted, and we are working with distributions in the exponential family, this is done by matching their moments [23,28,40]. Let $q^{\text{new}}(\mathbf{u}_h^l)$ be the updated posterior approximation for each hidden layer l and hidden unit h. The parameters of the corresponding approximate factors are obtained simply as $\prod_{l=1}^{L}\prod_{h=1}^{H_l}\tilde{\phi}_i(\mathbf{u}_h^l)^\alpha = Z_i \prod_{l=1}^{L}\prod_{h=1}^{H^l} q^{\text{new}}(\mathbf{u}_h^l) / \prod_{l=1}^{L}\prod_{h=1}^{H^l} q^{\setminus i}(\mathbf{u}_h^l)$. Hence, because both $q^{\text{new}}(\mathbf{u}_h^l)$ and $q^{\setminus i}(\mathbf{u}_h^l)$ are Gaussian distributions, the approximate factors are un-normalized Gaussian distributions.

Importantly, the PEP update equations effectively minimize locally the α-divergence between the tilted distribution $\hat{p}_i^\star$ defined as

$$\hat{p}_i^\star(\{\mathbf{f}^l, \mathbf{u}^l\}_{l=1}^L) \propto p(y_i|\mathbf{f}_i^L) \prod_{l=1}^{L}\prod_{h=1}^{H_l} p(\mathbf{f}_h^l|\mathbf{u}_h^l, \mathbf{f}^{l-1}) \frac{q(\mathbf{u}_h^l)}{\tilde{\phi}_i(\mathbf{u}_h^l)}\,, \tag{13}$$

and the approximate distribution q, *i.e.*, $D_\alpha[\hat{p}_i^\star||q]$ [34]. To see this, let λ_q be the natural parameters of q. For a distribution q in the exponential family:

$$\begin{aligned}\nabla_{\lambda_q} D_\alpha[\hat{p}_i^\star||q] &= -\frac{1}{\alpha}\int (\hat{p}_i^\star(\boldsymbol{z}))^\alpha q(\boldsymbol{z})^{1-\alpha} \nabla_{\lambda_q} \log q(\boldsymbol{z}) d\boldsymbol{z} \\ &= \frac{Z_{\tilde{p}}}{\alpha}(\mathbb{E}_q[s(\boldsymbol{z})] - \mathbb{E}_{\tilde{p}_i}[s(\boldsymbol{z})]) \propto \nabla_{\lambda_q} \mathrm{KL}[\tilde{p}_i||q]\,,\end{aligned} \tag{14}$$

where $\boldsymbol{z} = \{\mathbf{f}^l, \mathbf{u}^l\}_{l=1}^L$, and where we have defined $\tilde{p}_i \propto (\hat{p}_i^\star)^\alpha q^{1-\alpha}$ with normalization constant $Z_{\tilde{p}}$, $\alpha \neq 0$, and $s(\boldsymbol{z})$ is the vector of sufficient statistics of q. Therefore, $\tilde{p}_i = \hat{p}_i$, where $\hat{p}_i$ is the tilted distribution of PEP, defined in (12). At the minimum both gradients must be equal to zero and the moments of q and $\tilde{p}_i$ must match, which means that minimizing $\mathrm{KL}[\tilde{p}_i||q]$ or equivalently $\mathrm{KL}[\hat{p}_i||q]$ by matching the moments of both distributions is effectively minimizing $D_\alpha[\hat{p}_i^\star||q]$.

In summary, the PEP algorithm consists in applying the steps described below to every approximate factor $\prod_{l=1}^{L}\prod_{h=1}^{H_l}\tilde{\phi}_i(\mathbf{u}_h^l)$ and repeat until all approximate factors have converged. That is,

1. **Remove** an approximate factor to the power of α from the posterior q to compute the cavity distribution $q^{\setminus i}$ as in (11) for each l and h.
2. **Include** the true factor ϕ_i to the power of α to compute the tilted distribution $\hat{p}_i$ as in (12).
3. **Project** onto the approximating family by matching moments.

   $$q^{\text{new}}(\{\mathbf{f}^l, \mathbf{u}^l\}_{l=1}^L) = \arg\min_{q^* \in Q} \mathrm{KL}[\hat{p}_i(\{\mathbf{f}^l, \mathbf{u}^l\}_{l=1}^L) \parallel q^*(\{\mathbf{f}^l, \mathbf{u}^l\}_{l=1}^L)]\,,$$

   with Q the set of $q^*(\{\mathbf{f}^l, \mathbf{u}^l\}_{l=1}^L)$ allowed by (10).
4. **Update** the approximate factor.

   $$\prod_{l=1}^{L}\prod_{h=1}^{H_l}\tilde{\phi}_i(\mathbf{u}_h^l)^\alpha = Z_i \prod_{l=1}^{L}\prod_{h=1}^{H^l} q^{\text{new}}(\mathbf{u}_h^l) / \prod_{l=1}^{L}\prod_{h=1}^{H^l} q^{\setminus i}(\mathbf{u}_h^l)\,,$$

   where Z_i is the normalization constant of $\hat{p}_i$. See (17).
5. **Reconstruct** the approximate distribution q using the updated factor and (10).

Note that in step 3, we assume a particular form for q, described before, where only each $q(\mathbf{u}_h^l)$ can be adjusted. Moreover, we are working with distributions in the exponential family. The consequence is that the minimization in step 3 is done by matching the moments of the two distributions over each $\mathbf{u}_h^l$, where the other variables $\{\mathbf{f}^l\}_{l=1}^L$ are marginalized out [23,28,40].

At convergence, when the approximate factors do not change any more and (14) is equal to zero for each approximate factor, PEP minimizes the α-divergences between the tilted distributions, defined in (13), and q. This minimization is local and in general will be different from minimizing the α-divergence between the target posterior and the approximate distribution q. Such a local minimization is, however, expected to be similar to the global minimization, which is in general intractable [34]. It can be shown, that it becomes a global divergence minimization (between the target posterior and q) only when $\alpha \to 0$ [34]. In this case, the α-divergence tends to the KL-divergence employed in VI. Therefore, by letting $\alpha \to 0$ we retrieve the same optimization problem (and solution) as in VI.

Importantly, PEP gives as well an approximation to the log marginal likelihood. This estimate, $\log Z_q$, is simply obtained by normalizing the approximate joint distribution, in which each likelihood factor has been replaced by the product of the corresponding approximate factors $\tilde{\phi}_i(\mathbf{u}_h^l)$. That is, the un-normalized approximate distribution q. Namely,

$$\log Z_q = g(\boldsymbol{\theta}) - g(\boldsymbol{\theta}_{\text{prior}}) + \frac{1}{\alpha}\sum_{i=1}^{N} \log \tilde{Z}_i \,, \tag{15}$$

$$\log \tilde{Z}_i = \log Z_i + g(\boldsymbol{\theta}^{\setminus i}) - g(\boldsymbol{\theta}) \,, \tag{16}$$

where $\boldsymbol{\theta}$, $\boldsymbol{\theta}^{\setminus i}$ and $\boldsymbol{\theta}_{\text{prior}}$ are the natural parameters of $\prod_{l=1}^L \prod_{h=1}^{H^l} q(\mathbf{u}_h^l)$, the natural parameters of the cavity distribution $\prod_{l=1}^L \prod_{h=1}^{H^l} q^{\setminus i}(\mathbf{u}_h^l)$ and the natural parameters of the prior distribution $\prod_{l=1}^L \prod_{h=1}^{H^l} p(\mathbf{u}_h^l)$, respectively. Furthermore, $g(\boldsymbol{\theta})$ is the log-normalizer of a Gaussian distribution with natural parameters $\boldsymbol{\theta}$, and

$$\log Z_i = \log \int_D p(y_i|\mathbf{f}_i^L)^\alpha \prod_{l=1}^{L}\prod_{h=1}^{H_l} p(f_{h,i}^l|\mathbf{u}_h^l, \mathbf{f}_i^{l-1}) q^{\setminus i}(\mathbf{u}_h^l) d\mathbf{u}_h^l df_{h,i}^l \,, \tag{17}$$

where $D = (-\infty, \infty)$ and $f_{h,i}^l$ is the process value at layer l and unit h corresponding to the i-th data instance. Similarly, $\mathbf{f}_i^{l-1}$ is the vector of process values at layer $l-1$ corresponding to the i-th data instance. Thus, $\mathbf{f}_i^{l-1} \in \mathbb{R}^{H_l}$.

Critically, if PEP converges, it will converge to a stationary point of (15), *i.e.* a point in which the gradient of (15) w.r.t. the parameters of q and each approximate factor is equal to zero. This means that instead of running the PEP updates until convergence, one can aim at optimizing such an objective directly [41]. The problem is that the corresponding optimization problem is a max-min problem, in which one minimizes with respect to the parameters of the approximate factors and maximizes with respect to the parameters of the approximate distribution q. This optimization problem can be solved using a double-loop algorithm, which is computationally very expensive, and most of the times the PEP updates are faster for finding such a stationary point [41]. Nevertheless, in the following section we show how these difficulties and others can be alleviated.

4.2. Making power expectation propagation in DGPs feasible

The algorithm described above cannot be implemented in practice due to several intractabilities. These are addressed here to obtain a feasible PEP algorithm in the context of DGPs. In particular, the integral in (17) cannot be computed in an exact way w.r.t. each $f_{h,i}^l$ when $L > 1$. This prevents the exact computation of the objective (15), which needs to be approximated. The work in [16] proposes to approximate a similar predictive distribution for $\mathbf{f}_h^l$ when $l \leq 2$ using a Gaussian distribution with the same moments as the true distribution. Nevertheless, it is often the case that this distribution does not look similar to a Gaussian and hence this can be a poor choice. The reason for such a deviation from the Gaussianity assumption is the non-linear transformation of the input performed by each GP of the DGP. To overcome this problem, one can follow [19] and approximate the integrals in (17) w.r.t. each $f_{h,i}^l$ using Monte Carlo samples. The resulting noisy estimates and its gradients could be used in combination with stochastic optimization techniques to optimize the target objective (15). For this, each $\mathbf{u}_h^l$ is first marginalized out, which can be done analytically, since their distribution is Gaussian:

$$\begin{aligned} Z_i &= \int_D p(y_i|\mathbf{f}_i^L)^\alpha \prod_{l=1}^{L}\prod_{h=1}^{H_l} q(f_{h,i}^l|\mathbf{f}_i^{l-1}) df_{h,i}^l \\ &= \int_D p(y_i|\mathbf{f}_i^L)^\alpha \prod_{l=1}^{L}\prod_{h=1}^{H_l} \mathcal{N}(f_{h,i}^l|m_{h,i}^l, v_{h,i}^l + {\sigma_h^l}^2) df_{h,i}^l \,, \end{aligned} \tag{18}$$

where $D = (-\infty, \infty)$. In particular, if $\boldsymbol{\mu}_{h,l}^{\setminus i}$ and $\boldsymbol{\Sigma}_{h,l}^{\setminus i}$ are respectively the mean and covariance of $q^{\setminus i}(\mathbf{u}_h^l)$, then $m_{h,i}^l = \mathbf{k}_{\mathbf{f}_{h,i}^l, \mathbf{u}_h^l} \mathbf{K}_{\mathbf{u}_h^l, \mathbf{u}_h^l}^{-1} \boldsymbol{\mu}_{h,l}^{\setminus i}$ and $v_{h,i}^l = \mathbf{k}_{f_{h,i}^l, f_{h,i}^l} + \mathbf{k}_{f_{h,i}^l, \mathbf{u}_h^l} \mathbf{K}_{\mathbf{u}_h^l, \mathbf{u}_h^l}^{-1} (\boldsymbol{\Sigma}_{h,l}^{\setminus i} - \mathbf{K}_{\mathbf{u}_h^l, \mathbf{u}_h^l}) \mathbf{K}_{\mathbf{u}_h^l, \mathbf{u}_h^l}^{-1} \mathbf{k}_{\mathbf{u}_h^l, f_{h,i}^l}$, where $\sigma_h^{l\,2}$ is the variance of the Gaussian noise at the output of layer l and unit h. See [17] for further details.

Note that in (18) one can generate samples from the predictive distribution of each layer and unit using the re-parameterization trick, as in [17]. Moreover, if we do that, we have deterministic inputs at each unit and layer $\hat{\mathbf{f}}_{\cdot,i}^{l-1}$ and hence, the conditional distribution of $f_{h,i}^l$ is simply a Gaussian whose parameters can be computed analytically. Namely, $q(f_{h,i}^l | \hat{\mathbf{f}}_{\cdot,i}^{l-1}) = \mathcal{N}(\mathbf{f}_{h,i}^l | m_{h,i}^l, v_{h,i}^l + \sigma_h^{l\,2})$. All the generated samples can then be propagated through the GP layers and the approximation to Z_i is simply:

$$Z_i \approx \hat{Z}_i = \frac{1}{S} \sum_{s=1}^{S} p(y_i | \hat{\mathbf{f}}_{i,s}^L)^\alpha \,, \tag{19}$$

where S is the number of samples and $\hat{\mathbf{f}}_{i,s}^L$ is the s-th sample propagated to the last layer of the DGP. In the case of regression and binary classification, the dimensionality of the last layer is simply $H_L = 1$. Note that as a consequence of using a finite number of samples, the approximation, $\log \hat{Z}_i$, will have some bias. More precisely, $\mathbb{E}[\log \hat{Z}_i] \neq \log Z_i$, as a consequence of the non-linearity of the $\log(\cdot)$ function. In spite of this, however, a similar approximation has shown good empirical results [19], and the bias can be simply reduced by increasing S, the number of samples. According to our experiments a small value for S (*e.g.*, 20 samples) is enough to get good results.

An additional approximation can be introduced in the power expectation propagation algorithm to avoid having to use a double loop optimization algorithm to maximize the objective in (15). This approximation is called stochastic expectation propagation (SEP) [42]. In SEP, all the approximate factors are tied (constrained to have the same parameters) and we only keep in memory their product, *i.e.*, $\tilde{\phi}(\mathbf{u}_h^l) = \prod_{i=1}^N \tilde{\phi}_i(\mathbf{u}_h^l)$. This only affects the way of computing the cavity distribution $q^{\setminus i} \propto q / \tilde{\phi}^{\frac{\alpha}{N}}$, which now is the same for all likelihood factors. The posterior approximation $q(\mathbf{u}_h^l)$ is then defined as $q(\mathbf{u}_h^l) \propto p(\mathbf{u}_h^l) \tilde{\phi}(\mathbf{u}_h^l)$. This approximation has shown to give almost the same results in the expectation propagation algorithm, which is equivalent to PEP for $\alpha \to 1$ [42], and it greatly simplifies the PEP algorithm.

Critically, if all the approximate factors are the same and we have a single global factor, there is a one-to-one relation between the global factor and the posterior approximation q. The consequence is that under this approximation, we can directly find an optimal global factor $\tilde{\phi}$ by maximizing $\log Z_q$ in (15) with standard optimization techniques, and get rid of the pesky PEP updates, and the double loop optimization algorithm described previously. On top of that, we only have to save the global factor, reducing the amount of memory needed to store all the approximate factors (which increases with the number of data points). This is also the approach followed in [16,19], for training DGPs using expectation propagation. The result is a method that can be used for training DGPs that will find the approximate distribution by approximately minimizing α-divergences. Of course the estimate (19) will be noisy and also its gradients. However, these can be used in combination with a stochastic optimization algorithm to approximately optimize the target objective in (15). When $\alpha = 1$, such a method is equivalent to the one described in [19] for training DGPs using approximate EP and Monte Carlo methods. When $\alpha \to 0$ such a method leads to the same objective and solution as the method described in [17] for training DGPs using VI. We expect that intermediate values of α may lead to better results. Note also that the objective in (15) is suitable for mini-batch training since the sum across the training points can be approximated using a mini-batch. This allows to scale to very large datasets. All model hyper-parameters can also be tuned by maximizing the estimate of the marginal likelihood, as in [16,17,19].

Once the model has been trained and q has been estimated, the predictive distribution for a new test point $\mathbf{x}_{\text{new}}$ can be approximated by Monte Carlo sampling, as it is done in the computation of Z_i in (19). We only have to set $\alpha = 1$ and work with the actual posterior approximation q instead of the cavity distribution.

5. Related work

Previous works in the literature have addressed the minimization of α-divergences in the context of GPs [23,24]. In [23] they propose to use PEP as a unifying framework for GP regression and classification as it can recover the objectives that are optimized with EP and VI, which are the most commonly used approximations. They perform an extensive experimental analysis of this method in both regression and binary classification problems, showing that using values of α between 0 and 1 can give better results as an alternative to EP of VI solutions. The work in [24] extends these results to the specific case of multi-class classification with GPs, and compares several approximations to minimize α-divergences. These are also based on PEP. Although the techniques described in Section 4 are similar to the ones employed in [23,24], here we consider a different model. Namely, a DGPs, which can be considered as a generalization of the single layer models employed in [23,24]. Importantly, approximate inference in DGPs is significantly more challenging than in a standard GP since one has to face the problem that computing the predictive distribution is intractable. This requires extra approximations that are not needed in [23,24].

In [21] it is proposed a black-box algorithm to approximately minimize α-divergences in arbitrarily complex probabilistic graphical models. They also use PEP with tied approximate factors, and the objective that they consider is:

$$E[\boldsymbol{\theta}, \boldsymbol{\theta}_{\text{prior}}] = g(\boldsymbol{\theta}_{\text{prior}}) - g(\boldsymbol{\theta}_{\text{post}}) + \frac{1}{\alpha}\sum_{i=1}^{N} \log \mathbb{E}_q \left[\left(\frac{p(y_i|\mathbf{f}_i)}{\tilde{\phi}} \right)^{\alpha} \right], \tag{20}$$

where $\boldsymbol{\theta}_{\text{prior}}$ and $\boldsymbol{\theta}_{\text{post}}$ are the natural parameters of the prior and the approximate posterior q; $g(\boldsymbol{\theta}_{\text{prior}})$ 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 factor $\tilde{\phi}$. The authors evaluate their approach on probit regression and on Bayesian neural networks using a Gaussian approximate distribution that assumes independence. Such a method could be also used in principle in the context of DGPs. However, since it is a black-box method it cannot not use particular properties of the model considered nor the particular approximate distribution q we employ in (10). Moreover, to evaluate the objective it requires generating samples from the approximate distribution q. The approach described in Section 4 uses the fact that the latent variables $\{\mathbf{u}_h^l\}$, for $l = 1, \ldots, L$ and $h = 1, \ldots, H_l$, can be marginalized analytically. By contrast, the method described in [21] would generate samples for these variables to evaluate the objective. This is expected to lead to higher variances in the estimation of the objective and its gradients, resulting in a less efficient optimization process.

In the context of DGP models, the work in [17] uses the same approximate distribution q as in this work (which is the same as the one considered in [19] too). However, q is found using variational inference. For this, the VI lower bound on the marginal likelihood is maximized w.r.t. the parameters of q and the model hyper-parameters:

$$\log p(\mathbf{y}) \geq \mathcal{L} = \sum_{i=1}^{N} \mathbb{E}_q[\log p(y_i|\mathbf{f}_i^L)] - \sum_{l=1}^{L}\sum_{h=1}^{H_l} \text{KL}[q(\mathbf{u}_h^l) \parallel p(\mathbf{u}_h^l)]\,. \tag{21}$$

The required expectations in this objective are intractable, but they can be approximated by Monte Carlo as well by propagating samples as explained in Section 4. In this case, however, the Monte Carlo estimate and its gradients will be unbiased. When $\alpha \to 0$ the objective of our proposed method in (15) converges to this lower bound [22]. Therefore, our proposed method can be seen as a generalization of this approach. A difference between the approach in [17] and this work is that they consider shared inducing points within the same layer, and in our case each GP in a layer has an independent set of inducing points. Another difference is that they absorb the noise between layers in the kernel, and we explicitly parameterize these variables separately from the outputs of the GPs.

In [16] it was first proposed to use EP for approximate inference in the context of DGPs. That work also faces the problem of computing the predictive distribution at each layer of the DGP. They use a Gaussian approximation. In particular, they approximate the output of each GP from the second layer and beyond, using a Gaussian with the same mean and variance as the actual distribution (assuming a Gaussian input random variable to each GP). The approximation considered in Section 4 based on Monte Carlo samples is expected to be more accurate, since it will be able to capture multi-modal patters in the predictive distribution. This is confirmed in [19], where such an approximate predictive distribution is used. The EP objective in [19] is equivalent to the PEP objective we suggested in (15) when $\alpha = 1$. In [19] the authors use shared inducing points within layers and parameterize the noise between layers separating these variables from the outputs of the GPs.

Although not directly related to the minimization of α-divergences, other works in the literature have tried to improve the computational cost and accuracy of GP and DGP models using these units as the main building block. For example, in [43] they introduce SOLVE-GP, an alternative formulation of the VI objective that introduces an extra set of inducing points. This adds more flexibility to each GP model at a smaller computational cost than simply increasing the number of inducing points. Such a method has been evaluated in the context of DGPs. Also related with this approach, the decoupled inducing points methods considered in [44,45] define as well a different inducing points sets to parameterize the mean and covariance of the GP. All these works are orthogonal to ours and can be included in the PEP framework to further increase the flexibility of the model and reduce the associated computational cost.

Importance-weighed VI [46] has been proposed as a more accurate alternative to VI in the context of DGPs. That method receives the noise as an input in the form of latent covariates. These covariates allow to capture complex patterns in the predictive distribution. It requires, however, the inclusion of extra data-dependent latent input attributes in the model and the computation of an approximate posterior distribution for each of them, with different parameters for each data point in the training set. We show in the experiments that our method can also capture complex patterns in the predictive distribution (*e.g.*, several modes or heteroscedastic noise) without the extra cost of including the latent covariates.

In all the methods described, q is assumed to have a parametric form, *e.g.* Gaussian. In [18] they propose to use Hamiltonian Monte Carlo to draw samples from the target distribution allowing more flexible approximations to q. Nevertheless, that method has the difficulty of tuning the model's hyper-parameters (covariance hyper-parameters and inducing points locations) since it does not have an objective function to adjust them. The authors of that paper propose a moving window algorithm with that goal.

Following this idea of having a more flexible approximate distribution q, in [47] it is considered VI in the context of DGPs with an implicit distribution for q, *i.e.*, a distribution from which we can easily generate samples but that lacks a closed-form density expression. Such a distribution can be obtained simply by letting a source of white noise go through a deep neural network. Implicit distributions, however, make difficult evaluating the KL term between q and the prior, in the VI lower bound. To address this problem, the authors of that paper propose to train a discriminator (*i.e.*, a classifier) whose expected output approximates the KL term. Of course, this adds the extra cost of training such a classifier at the same time as the DGP model. The method we propose in Section 4 does not have this problem.

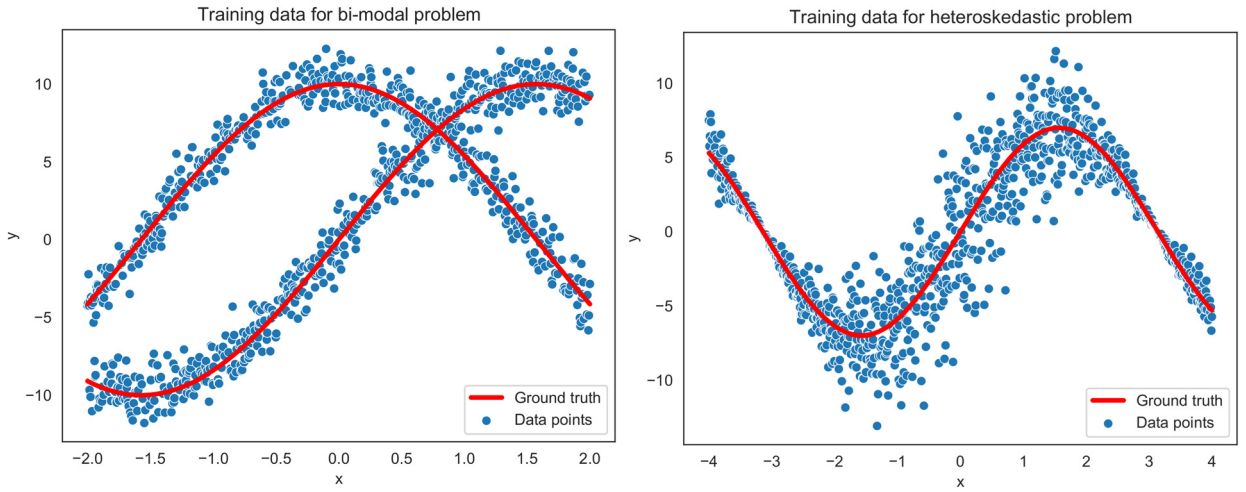

Fig. 4. Training data for each synthetic problem. In red the target function. (left) Bimodal dataset. (right) Dataset with heteroscedastic noise. Best seen in color. (For interpretation of the colors in the figures, the reader is referred to the web version of this article.)

Finally, in the context of Bayesian neural networks, some works have addressed the minimization of α-divergences as well, using non-Gaussian approximate distributions [37,38]. In particular, in [37] it is considered, as the approximate distribution, a mixture between two points of probability mass, one located a zero and another one located at some tunable location. The work in [38] considers also an implicit distribution for q and also relies on training a classifier to approximate the objective. Importantly, while q is not constrained to belong to the exponential family in these two works, the PEP objective described in (15) is further approximated there. The approximation suggested is only expected to be accurate for small values of α or large values of N, *i.e.*, the size of the training set.

The use of probabilistic methods in the context of convolutional neural networks has been recently proposed in [48]. These authors suggest using a GP embedded channel attention (GPCA) module for effective performance improvement in these networks. Specifically, a GP is used to model the correlations among different channels. The GP could be replaced by a deep GPs which could be trained by minimizing α-divergences using the proposed method. This could result in a better predictive distribution and a better over-all performance.

The minimization of α-divergences for approximate inference with deep GPs has potential applications in the context of active learning [49]. Here, one is interested in introducing into the training set the most useful data instances for improving overall prediction performance, after a labeling process. For this, the predictive distribution is critical since it indicates which instances are expected to be most useful for learning. Essentially those are the data instances for which the model is most uncertain about. Therefore, a better predictive distribution should be translated into better active learning results.

Deep GPs also have potential applications in the context of multi-view representation learning [50]. In this setting, a multi-output deep GP is used to learn a non-linear mapping from latent variables to observed variables. The deep GP can exploit similarities in the latent representation of each instance, for each different view. The minimization of α-divergences could be explored in that context. Specifically, a better predictive distribution is expected to lead to a better generative model, and hence to a better latent feature representation.

6. Experiments

In this section we evaluate the performance of the proposed approach for training DGPs using the approximate minimization of α-divergences in several experiments. The method has been implemented in Tensorflow [51].[1] In all the experiments we use ADAM as the stochastic optimizer with the default parameters [52] and a learning rate of 0.001 except when indicated otherwise. We also use the RBF kernel with automatic relevance determination in all GP layers except when indicated otherwise. The initial noise variance in the intermediate noise layers is equal to 10^{-5} and the likelihood noise parameter is initialized to a small value. Namely, $\sigma^2 = 0.01$, except when indicated otherwise.

6.1. Predictive distribution

The first experiment aims to explore the properties of the predictive distribution of the DGPs using PEP for different values of α. For that, we generate synthetic data as in [53] and [19]. In Fig. 4 we plot the training data for the two datasets considered, along with the true function to be predicted. The first problem has a bimodal predictive distribution for y and

[1] Code available at https://github.com/cvillacampa/dgps_pep.

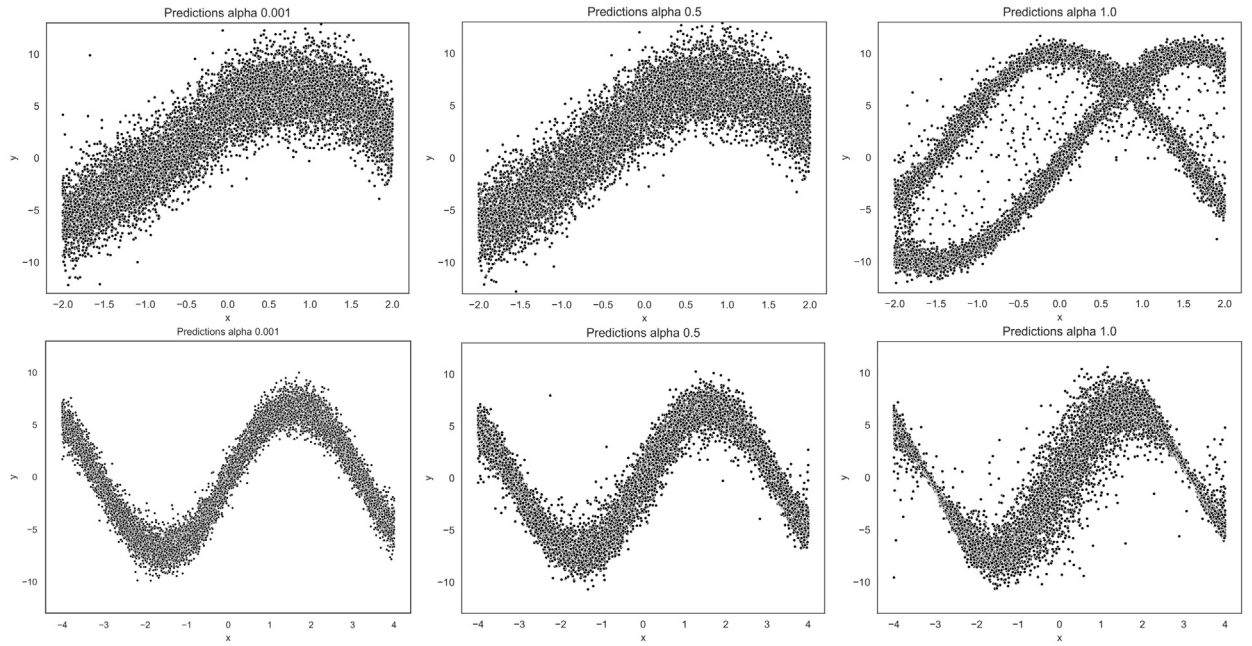

Fig. 5. Samples from the predictive distribution of a DGP trained using different values of α, as indicated on each figure. The top row corresponds to the bimodal problem. The bottom row corresponds to the heteroscedastic problem. The values of alpha considered are 0.001, 0.5 and 1.0. The Appendix contains similar figures for other values of the α parameter.

Table 1
Test RMSE and test log-likelihood for each synthetic datasets.

Problem	α	RMSE	Log-likelihood
Bimodal distribution	10^{-3}	5.203	-3.070
	0.5	5.136	-3.055
	1	**5.110**	**-2.138**
Heteroscedastic noise	10^{-3}	**1.870**	-2.065
	0.5	1.876	-2.016
	1	1.892	**-1.639**

the second one has heteroscedastic noise, *i.e.*, the variance of the noise depends on the input location x. Both datasets have $1,000$ data points for training and 500 for test. We train a DGP model with 4 layers and 3 hidden units. The number of inducing points is set to $M = 10$. We choose a mini-batch size of 50 and the number of samples to propagate through the DGP equal to 50. Similar results are obtained for other numbers of layers and units. In both problems, the DGP is trained for 500 epochs. The learning rate for the first problem has been set to 0.01 and for the second problem to 0.002.

Fig. 5 shows the predictive distribution obtained for different values of α, *i.e.*, $\alpha = \{10^{-3}, 0.5, 1\}$. We observe that choosing a value of $\alpha \to 0$ makes the model unable to capture the complex properties predictive distributions in the datasets, *i.e.*, bimodality and heteroscedastic noise, and as we increase α these properties are better captured. This agrees with the results previously reported in [19], as expected because $\alpha \to 0$ retrieves the VI approach of [17] and $\alpha \to 1$ the EP objective of [19]. Also, it explains why when evaluating the method on the test set, we get similar results in terms of the RMSE but the log-likelihood improves as we increase α, as we can see in Table 1. In Appendix C.1 we have included the results for more values of α between 0 and 1. This indicates that higher values of α seem to produce more accurate predictive distributions. In terms of the root mean squared error (RMSE) all values of α seem to provide similar results.

6.2. Performance on UCI datasets

We evaluate the proposed approach on several datasets from the UCI repository [54]. Namely, 8 regression datasets, 7 binary classification datasets and 8 multi-class classification datasets (see Appendix B for the datasets' details). We use 90% of the data for training and 10% for testing. For the multi-class problems we follow [24,55] and we choose 20% for training and 80% for test for *Satellite*, in *Vowel* we consider only the points belonging to the 6 first classes and finally, for *Waveform* (synthetic) we generate $1,000$ instances and split them in 30% for training and 70% for testing. We consider DGP models with an increasing number of layers, *i.e.*, $L = \{2, 3, 4, 5\}$. The number of hidden units is the minimum between the problem dimension and 30, as in [17,19]. We assume a linear mean function for each GP (except for the last layer) as in [17,19].

All datasets have been trained for 20,000 iterations with a mini-batch size of 100. The number of inducing points is set to $M = 100$. We report averages over 20 different splits of the data into training and testing. The values of α considered range from $\alpha \to 0$ to $\alpha = 1$ with steps of size 0.1. The number of Monte Carlo samples S is 20.

Fig. 6 shows the average rank, across datasets and data splits, of each method in terms of the test RMSE and the average test log-likelihood, for each value of the number of layers L and value of α for the UCI regression datasets. Fig. 7 and 8 show similar results for the binary and the multi-class classification UCI problems, respectively. The rank of each method for a particular data split is computed as follows. The best method according to the considered metric (*e.g.*, RMSE or test log-likelihood) gets rank 1. The second best method gets rank 2, and so on. Therefore, the smallest rank value the better the method performs. We then report average ranks, for a fixed number of GP layers, for each value of α, across datasets and splits. This provides an estimate of how well each method performs over-all for each value of α. Appendix C.2 has extra results with the average error and test log-likelihood for each method on each UCI dataset considered.

We observe that if we care about the performance in terms of the negative test log-likelihood, it is in general better to choose a value of α closer to 1, in regression, binary and multi-class classification problems. This makes sense since the data-dependent term of the objective of the PEP algorithm, defined in (17), becomes closer to the log-likelihood of the training data for $\alpha = 1$. Moreover, this is also compatible with the results of the previous section that showed that the most accurate predictive distribution is obtained when α is close to 1. In terms of the classification error and RMSE, we observe that intermediate values of α, *i.e.*, values of α closer to 0.5, can lead to better results in regression problems. This is particularly the case when the number of layers L is small. This is compatible with the findings of [23] for single-layer GP models. In the case of binary and multi-class classification problems, the differences are not that clear when considering the prediction error. In binary classification it seems that values of α closer to 0.5 seem to give slightly better prediction error results. However, in this case many values of α give similar prediction error. In multi-class classification values of α closer to 1.0 seem to give slightly better prediction error results when $L = 2$. For larger number of layers, α values close to 0.5 also perform well in terms of the prediction error.

Importantly, we do not know a priori which is the optimal value of α, and in the small UCI datasets considered it could be the case that the DGP model is already enough flexible to see any notable differences in performance, specially if the number of layers L is high. However, a smart choice would be to start exploring the solutions around $\alpha = 0.5$ and lower the value of α slightly if we care about the RMSE and prediction error, in regression and binary classification, respectively. We should choose α closer to 1 if we care about the prediction error in multi-class problems, and if we care about the quality of the predictive distribution in general, as we have also observed in the experiments of Section 6.1.

A similar dependence of the model's performance with respect to the different values of α has already been reported in the literature in previous works for other models [19,23,24,55,56]. Again, the behavior observed can be explained by noting that the objective that we are optimizing gets more similar to the log-likelihood as we choose values of α closer to 1. That is, a term similar to $\log \mathbb{E}_q[p(y_i|\mathbf{f}_i)^L]$, the data log-likelihood, appears in (17), but where q is replaced by the cavity distribution. These terms contribute to the PEP objective in (15), which is maximized.

Finally, in terms of the number of layers, the results obtained do not vary much. A similar dependence of the results on α is observed, although the differences are less clear in regression problems. Only in multi-class classification problems, reported in Fig. 8, we observe that larger values of α clearly favor better prediction errors for $L = 2$. When the number of layers L grows, values of α close to 0.5 also seem to give good prediction errors in these problems.

6.3. Performance on bigger datasets

In the last experiments, we compare the performance of the DGP methods, trained for different values of α, as a function of the training time. Again, we consider different number of GP layers L in each model. The experimental setup considered is the same as in the UCI experiments. All models have trained for a maximum training time of 5 days. This includes also the time used for computing the performance on the test set.

In these experiments, we choose bigger datasets to show that the method is scalable. Table B.5 in the appendix shows the characteristics of the datasets considered. These datasets are very big with up to several million data instances. Therefore, we report the performance in only one split of the data into training and test. Appendix B has the details of the splits considered. Figs. 9, 10 and 11 show the results for the Year, HIGGS and MNIST datasets, respectively. These figures show the test log-likelihood and the prediction error (the RMSE in the case of the Year dataset since it is a regression problem) as a function of the computational time. We show results for each number of layers L considered.

The Year dataset is a regression dataset, and we observe in Fig. 9 that for all values of the number of layers the values of α that converge to a better solution in terms of the negative log-likelihood are values closer to 1. In terms of the test RMSE, intermediate values of α that are closer to 0 seem to lead to better solutions, but the differences are small. These results are compatible with the ones obtained in the previous section. The behavior described is similar for each value of the number of layers L considered.

The HIGGS dataset is a binary classification dataset. Fig. 10 shows that the differences in performance both in terms of the test error and negative test log-likelihood are not very big. However, higher values of α, *i.e.*, that are closer to 1, lead to better solutions in terms of both metrics. This is particularly the case when the number of layers L is small. When the number of layer is large, the differences are less clear and intermediate values of α seem to perform slightly better. Nevertheless, the differences are very small.

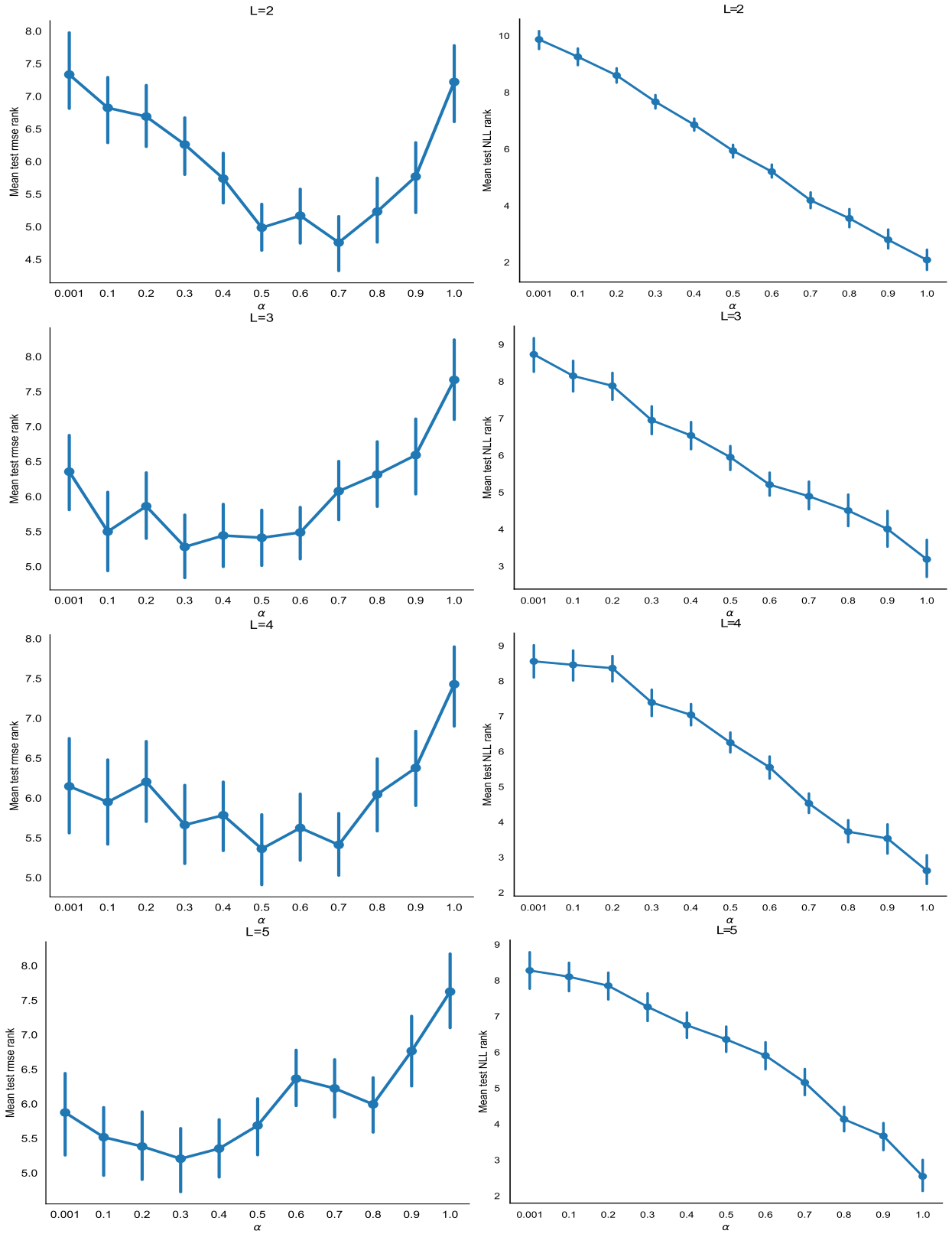

Fig. 6. Average rank for each number of layers and value of α, across datasets and data splits, in terms of the test RMSE (left-column) and the negative test log-likelihood (right-column). Results are shown for the UCI regression datasets.

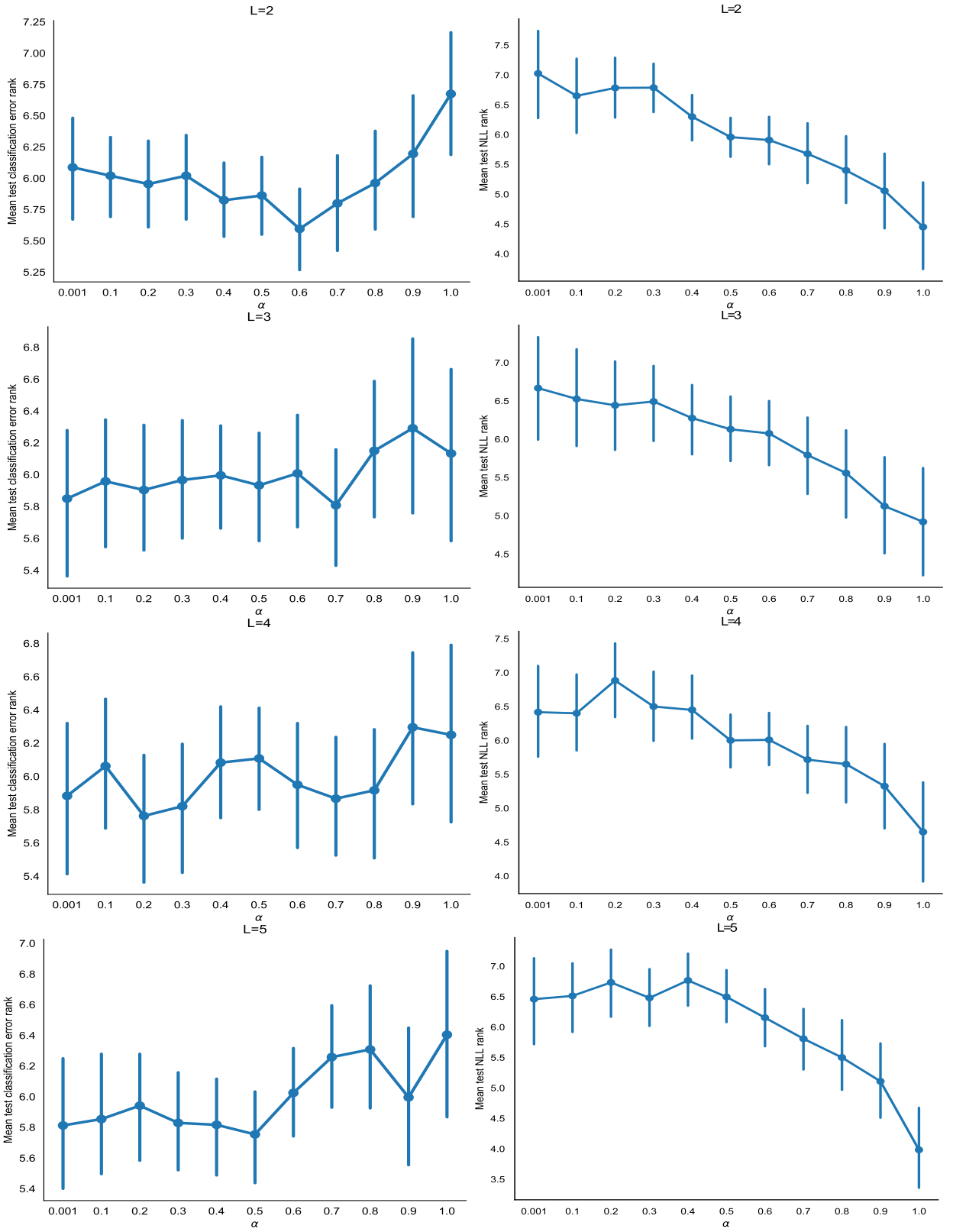

Fig. 7. Average rank for each number of layers and value of α, across datasets and data splits, in terms of the test error (left-column) and the negative test log-likelihood (right-column). Results are shown for the UCI binary classification datasets.

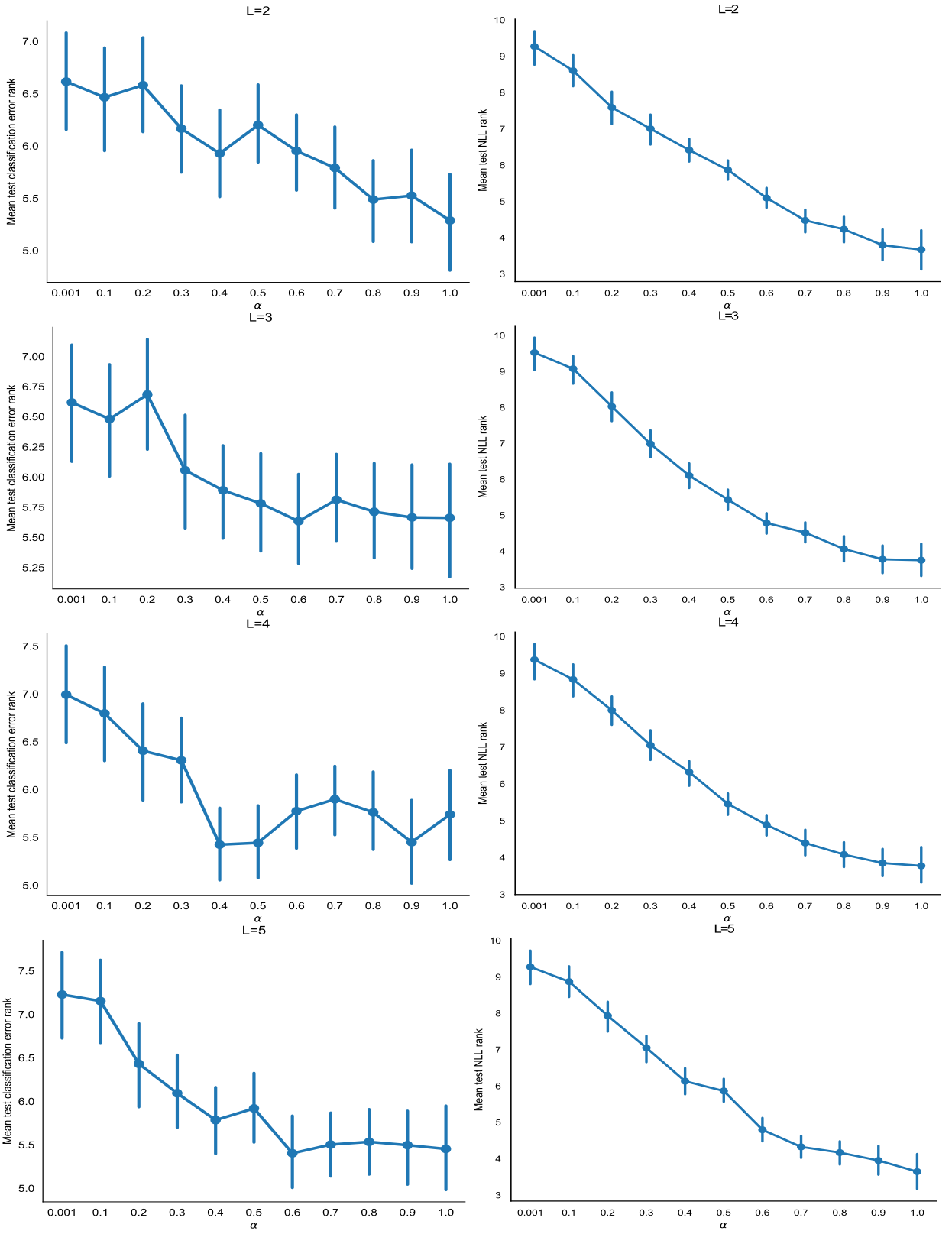

Fig. 8. Average rank for each number of layers and value of α, across datasets and data splits, in terms of the test error (left-column) and the negative test log-likelihood (right-column). Results are shown for the UCI multi-class classification datasets.

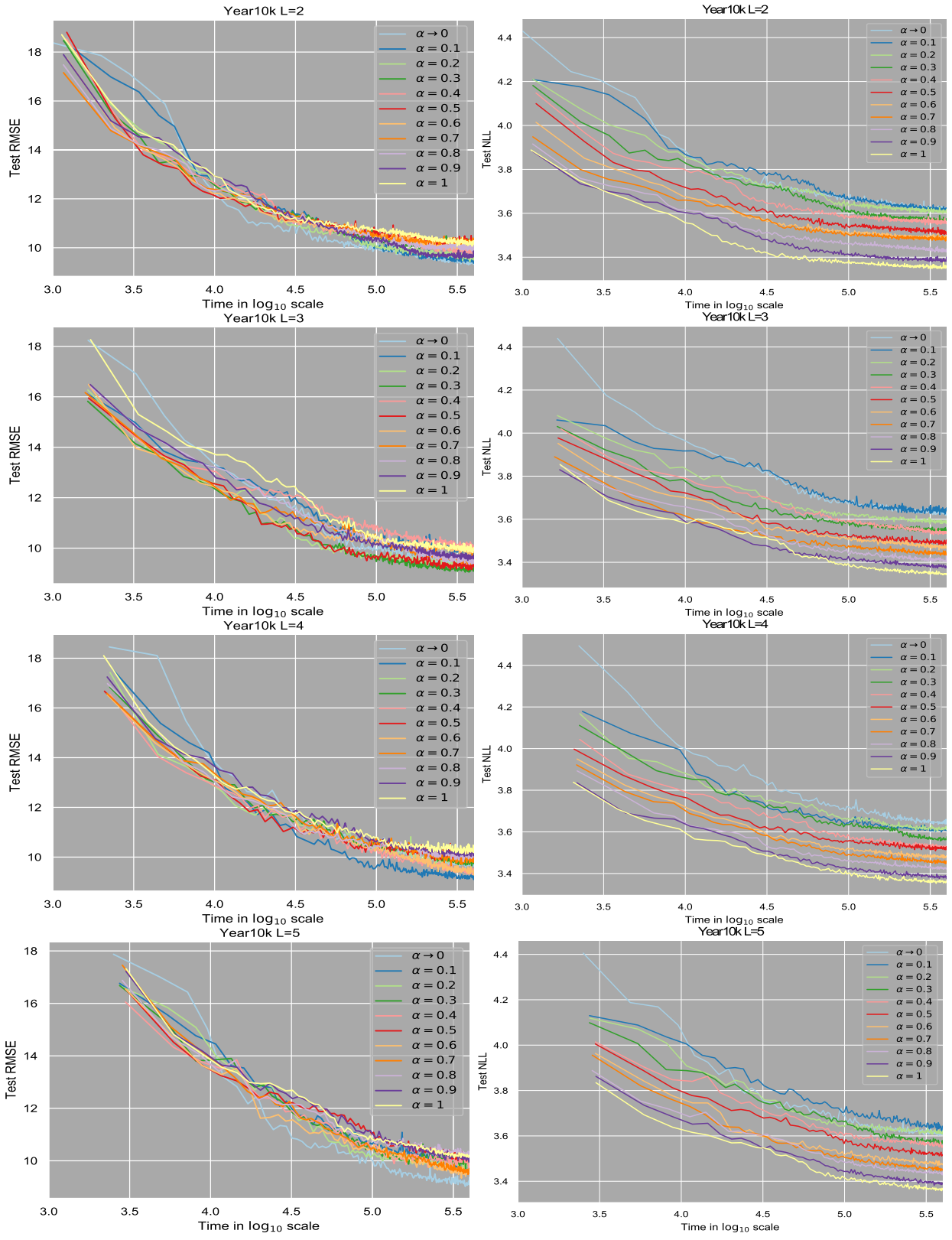

Fig. 9. Test RMSE (left) and negative test log-likelihood (right) as a function of training time in seconds on the Year dataset. Best seen in color. (For interpretation of the colors in the figures, the reader is referred to the web version of this article.)

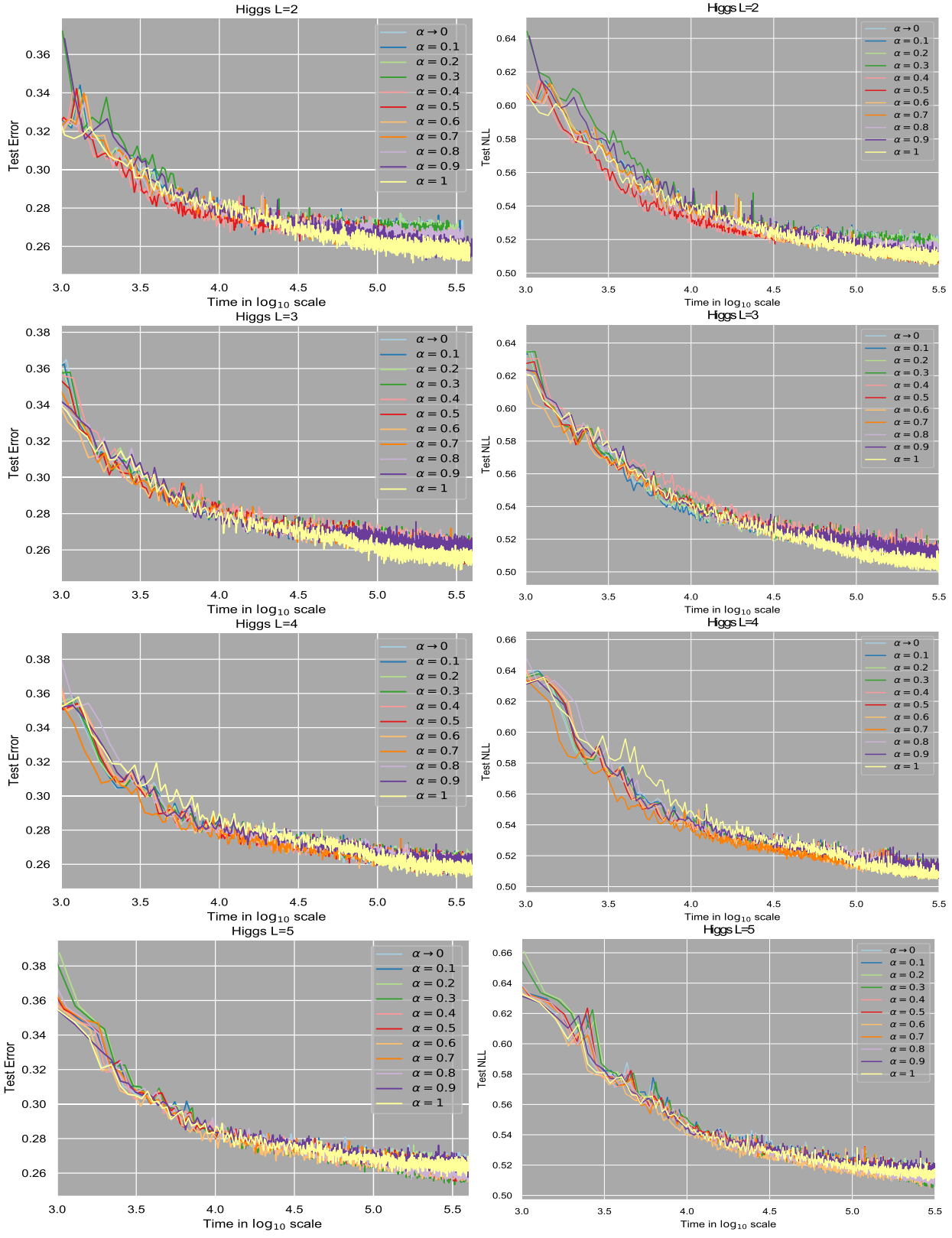

Fig. 10. Test error (left) and negative test log-likelihood (right) as a function of training time in seconds on the HIGGS dataset. Best seen in color. (For interpretation of the colors in the figures, the reader is referred to the web version of this article.)

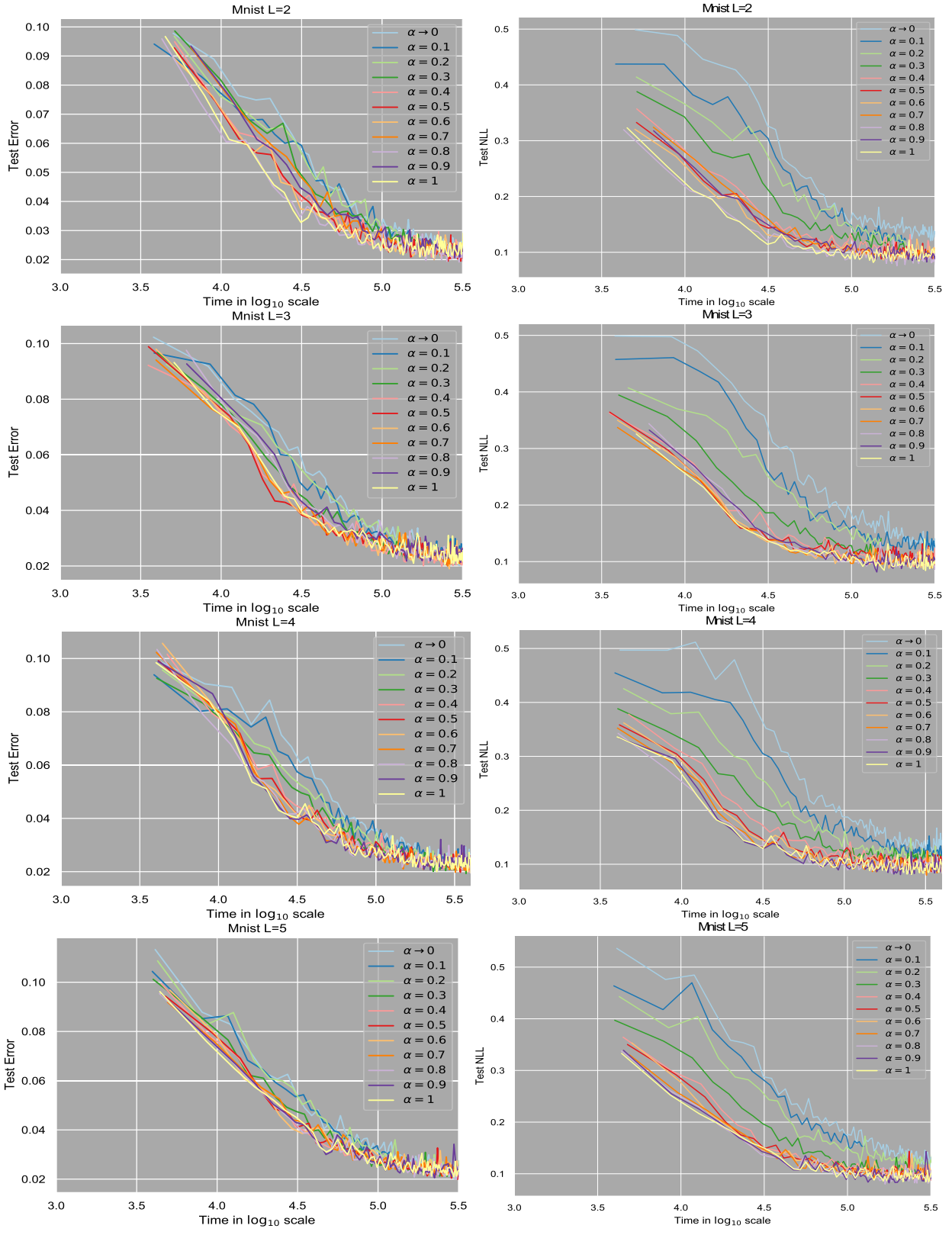

Fig. 11. Test error (left) and negative test log-likelihood (right) as a function of training time in seconds on the MNIST dataset. Best seen in color. (For interpretation of the colors in the figures, the reader is referred to the web version of this article.)

Finally, the MNIST is a multi-class classification dataset. In the MNIST experiments we use in the first layer a 9-degree polynomial kernel with automatic relevance determination as there is previous evidence that it works better with this dataset [25,57]. Fig. 11 shows the results obtained. We observe that values of α closer to 1 lead to better results in terms of the test log-likelihood. In terms of the prediction error all values of α give similar results. The results obtained are similar for each number of layers L considered.

We have extra results on large problems in Appendix C.3. There, we also include a table showing the final performance for each method on each large-scale dataset considered.

7. Conclusions

The optimization of α-divergences allows to interpolate between approximate inference methods that are closer to VI when $\alpha \to 0$ or EP as $\alpha \to 1$. Previous works in the literature had already considered the optimization of these divergences for approximate inference in the context of Gaussian processes [23,24] or in the context of Bayesian neural networks [21,37]. In this work, we have focused on the minimization of α-divergences in the context of deep Gaussian processes (DPGs) [14]. This is a very challenging task due to the complexity of these models. In particular, the predictive distribution of a DGP is intractable. The reason is that in the second and following layers the inputs are random variables coming from the predictions of the previous layer. We have proposed here a novel method for training DGPs by minimizing α-divergences that is based on the power expectation propagation (PEP) algorithm [22]. Critically, a naive implementation of PEP in the context of DGPs fails due to several intractabilities. We have carefully addressed them providing a fully functional algorithm. The proposed method is also memory efficient and suitable for stochastic optimization that allows to tackle big datasets with up to several millions of data instances. Importantly, instead of carrying out the PEP updates of the approximate factors, as in the original PEP formulation, we directly optimize the approximation to the marginal likelihood to find a good approximate distribution q.

We have shown that optimizing α-divergences in the context of DGPs is an alternative that is worth of being considered as it can provide better results by varying the parameter α of the algorithm in some situations. We have carried out several experiments in different problems including regression, binary and multi-class classification for different values of α to assess this. The results of the experiments suggest that in regression, binary and multi-class problems, in general, the performance in terms of the test log-likelihood gets better as we increase the value of α to values close to 1. In regression problems, however, the performance in terms of the RMSE is better for intermediate values of α between (0 and 0.5). This means that one can obtain better results in the context of DGPs, in terms of this metric, by optimizing a different divergence than the one considered in VI or EP.

In binary classification problems, the differences are not that clear. In any case, values of α between (0 and 0.5) seem to provide slightly better results in terms of the prediction error, on average. In the case of multi-class problems, higher values of α, *i.e.*, those that are close to 1 seem to provide better results in terms of the prediction error.

The findings obtained agree with previous works on GPs and the minimization of α-divergences [23,24]. Also, as we increase the flexibility of the DGP model by increasing the number of layers we have observed that the differences between values of α reduce.

In general, finding the optimal value of α for a specific model and learning problem is a difficult task. Our suggestion is to do a grid-search combined with cross validation to estimate the associated performance metric to see what value of α performs best.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors gratefully acknowledge the use of the facilities of Centro de Computación Científica (CCC) at Universidad Autónoma de Madrid. The authors also acknowledge financial support from Spanish Plan Nacional I+D+i, Ministerio de Ciencia e Innovación, grant PID2019-106827GB-I00 / AEI / 10.13039/501100011033.

Appendix A. Level curves for the approximate Gaussian distribution q using different α-divergences

In Fig. A.12 of this section, we show the resulting level curves of the p.d.f. that approximates a factorizing Gaussian obtained by minimizing the α-divergence for different values of α. We observe that the results for $\alpha \to 0$ and $\alpha \to 1$ coincide with those of Fig. 3, as expected. We also observe that an intermediate value of α provides an approximate solution that balances zero-forcing ($\alpha \to 0$) and inclusive behaviors ($\alpha \to 1$).

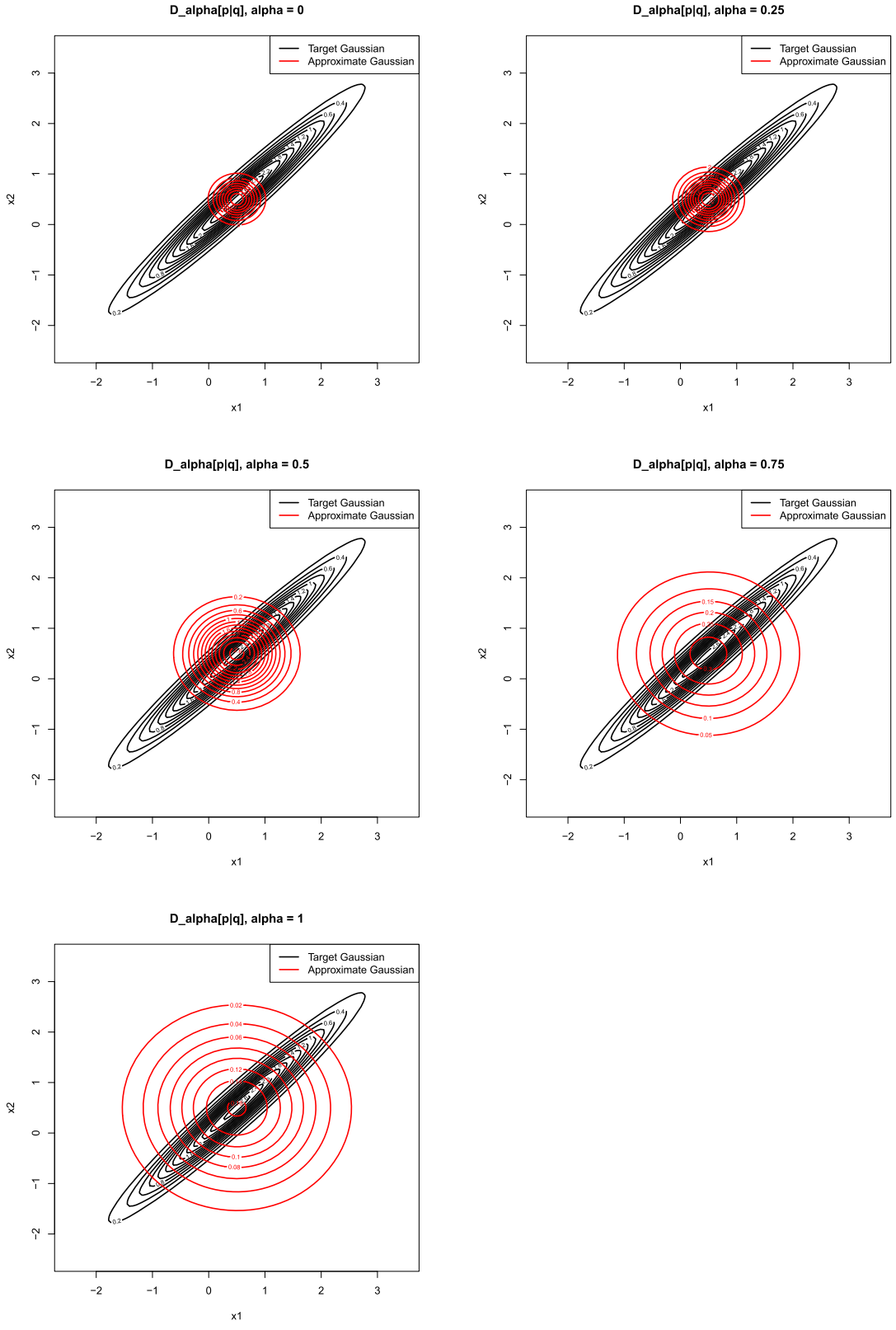

Fig. A.12. Level curves of the p.d.f. that results when approximating a Gaussian distribution p with strong dependencies using a factorizing Gaussian for different values of α. Best seen in color.

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

Dataset	#Instances	#Attributes
Boston	506	13
Concrete	1,030	8
Energy	768	8
Kin8	8,191	8
Naval	11,934	16
Power	9,568	4
Protein	45,730	9
Wine	1,599	11

Table B.3
Characteristics of the binary classification datasets from the UCI Repository.

Dataset	#Instances	#Attributes
Australian	690	14
Breast	683	10
Crabs	200	7
Ionosphere	351	34
Pima	768	8
Sonar	208	60
Banknote	1,372	4

Table B.4
Characteristics of the multi-class classification 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

Table B.5
Characteristics of the big datasets.

Dataset	#Instances	#Attributes	#Classes
Year	515,345	90	–
HIGGS	11,000,000	28	2
MNIST	70,000	784	10

Appendix B. Details of the datasets

Tables B.2, B.3 and B.4 show the characteristics of the regression, binary and multi-class classification datasets considered from the UCI repository in the main document. These tables show, for each problem, the number of samples, the number of attributes and the number of class labels (for the multi-class problems).

Table B.5 summarizes the characteristics of the big datasets of Section 6.3. The number of instances is the sum of the training and test instances. In the experiments we choose 10,000 instances for testing and the rest for training, except in MNIST, which comes already with a training-test partition with $60,000$ instances for training and $10,000$ for testing.

Appendix C. Extra experimental results

C.1. Performance of the predictive distribution

See Figs. C.13, C.14 and Table C.6.

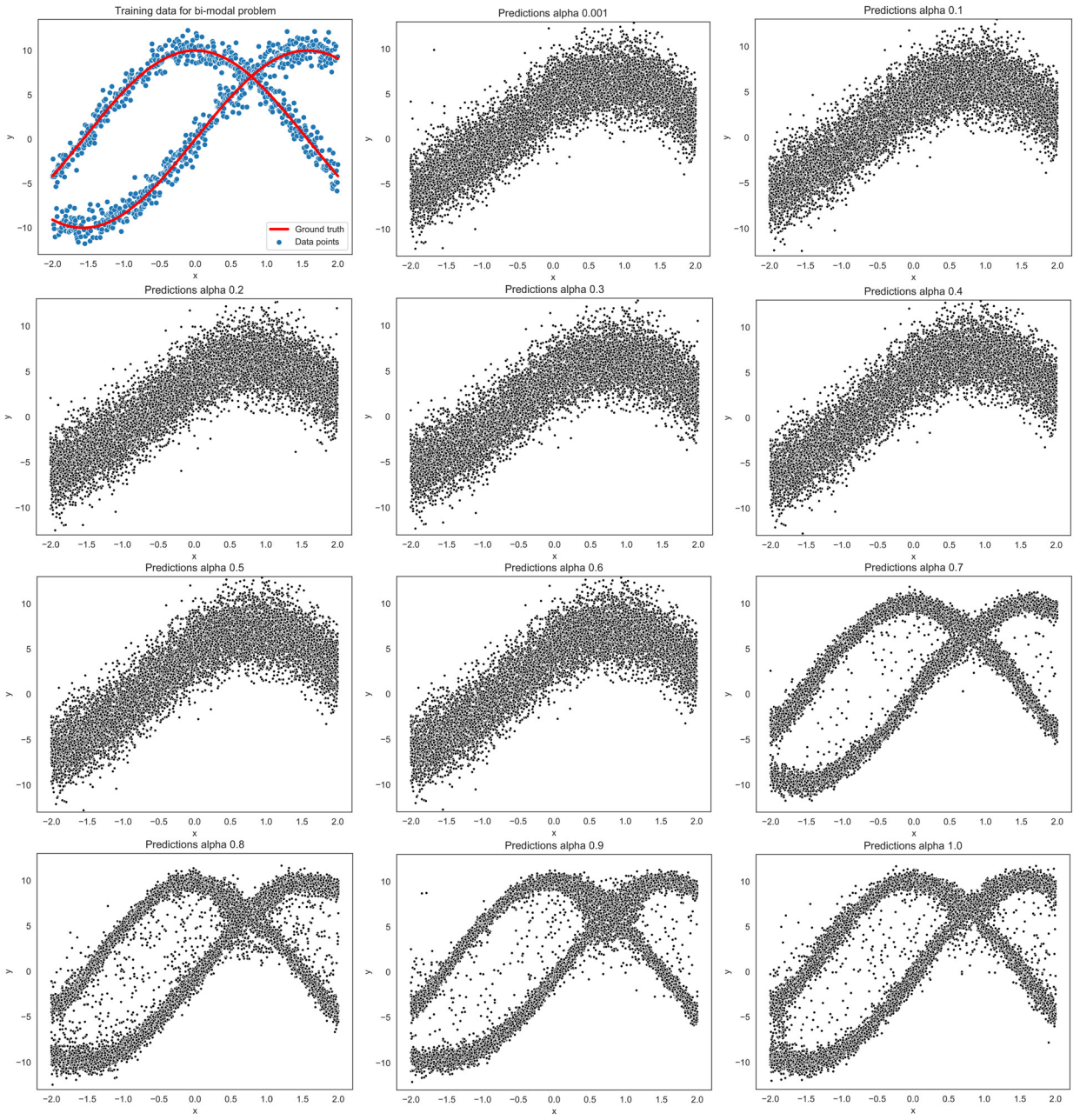

Fig. C.13. Samples from the bimodal predictive distribution using different values of α.

C.2. Performance on the UCI datasets

See Tables C.7–C.12.

C.3. Performance on bigger datasets

See Tables C.13 and C.14.

References

[1] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–444.
[2] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, vol. 25, 2012, pp. 1097–1105.

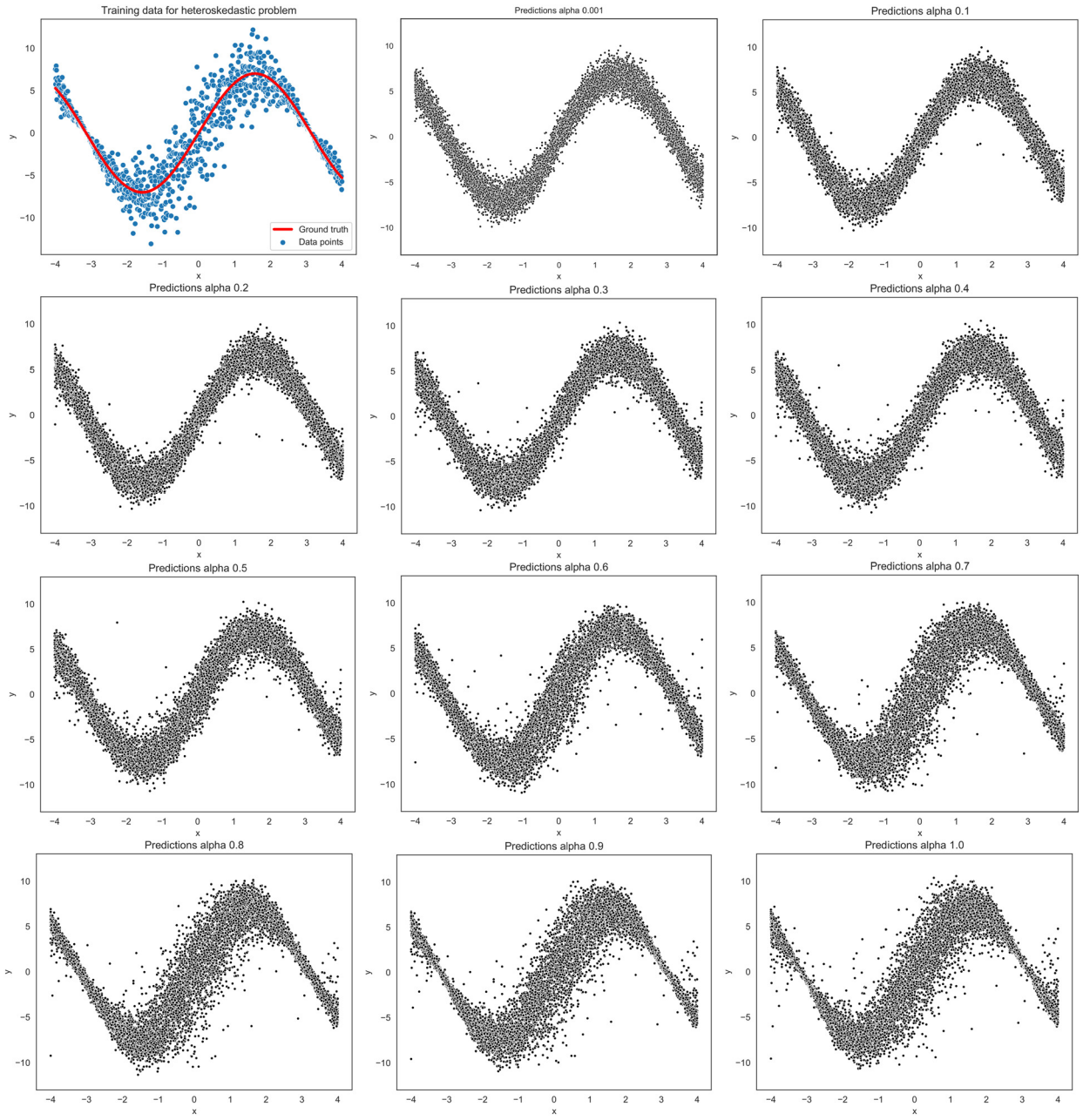

Fig. C.14. Samples from the heteroscedastic predictive distribution using different values of α.

[3] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. (8) (1997) 1735–1780.
[4] Y. Gal, Uncertainty in deep learning, PhD thesis, University of Cambridge, 2016.
[5] C.E. Rasmussen, C.K.I. Williams, Gaussian Processes for Machine Learning, Adaptive Computation and Machine Learning, The MIT Press, 2006.
[6] R.M. Neal, Bayesian Learning for Neural Networks, vol. 118, Springer Science & Business Media, 2012.
[7] M.K. Titsias, Variational learning of inducing variables in sparse Gaussian processes, in: International Conference on Artificial Intelligence and Statistics, 2009, pp. 567–574.
[8] E. Snelson, Z. Ghahramani, Sparse Gaussian processes using pseudo-inputs, in: Advances in Neural Information Processing Systems, 2006, pp. 1257–1264.
[9] J. Hensman, N. Fusi, N.D. Lawrence, Gaussian processes for big data, in: Uncertainty in Artificial Intelligence, 2013, pp. 282–290.
[10] D. Duvenaud, J. Lloyd, R. Grosse, J. Tenenbaum, Z. Ghahramani, Structure discovery in nonparametric regression through compositional kernel search, in: Proceedings of the 30th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, PMLR, vol. 28, 17–19 Jun 2013, pp. 1166–1174.
[11] A. Wilson, R. Adams, Gaussian process kernels for pattern discovery and extrapolation, in: International Conference on Machine Learning, in: PMLR, 2013, pp. 1067–1075.

Table C.6
Test RMSE and test log-likelihood for the synthetic datasets.

Problem	α	RMSE	Log-likelihood
Bimodal distribution	10^{-3}	5.203	-3.070
	0.1	5.201	-3.069
	0.2	5.211	-3.071
	0.3	5.192	-3.067
	0.4	5.135	-3.056
	0.5	5.136	-3.055
	0.6	5.138	-3.051
	0.7	5.206	-2.245
	0.8	5.222	-2.190
	0.9	5.179	**-2.127**
	1	**5.110**	-2.138
Heteroscedastic noise	10^{-3}	1.870	-2.065
	0.1	1.880	-2.063
	0.2	1.882	-2.063
	0.3	1.910	-2.082
	0.4	1.905	-2.068
	0.5	**1.876**	-2.016
	0.6	1.921	-1.955
	0.7	1.893	-1.863
	0.8	1.896	-1.780
	0.9	1.897	-1.701
	1	1.892	**-1.639**

[12] R. Calandra, J. Peters, C.E. Rasmussen, M.P. Deisenroth, Manifold Gaussian processes for regression, in: International Joint Conference on Neural Networks, 2016, pp. 3338–3345.
[13] A.G. Wilson, Z. Hu, R. Salakhutdinov, E.P. Xing, Deep kernel learning, in: International Conference on Artificial Intelligence and Statistics, 2016, pp. 370–378.
[14] A. Damianou, N. Lawrence, Deep Gaussian processes, in: Proceedings of the Sixteenth International Workshop on Artificial Intelligence and Statistics (AISTATS), 2013, pp. 207–215.
[15] K. Cutajar, E.V. Bonilla, P. Michiardi, M. Filippone, Random feature expansions for deep Gaussian processes, in: International Conference on Machine Learning, 2017, pp. 884–893.
[16] T.D. Bui, D. Hernández-Lobato, J.M. Hernández-Lobato, Y. Li, R.E. Turner, Deep Gaussian processes for regression using approximate expectation propagation, in: International Conference on Machine Learning, 2016, pp. 1472–1481.
[17] H. Salimbeni, M. Deisenroth, Doubly stochastic variational inference for deep Gaussian processes, in: Advances in Neural Information Processing Systems, vol. 30, 2017, pp. 4589–4600.
[18] M. Havasi, J.M. Hernández-Lobato, J.J. Murillo-Fuentes, Inference in deep Gaussian processes using stochastic gradient Hamiltonian Monte Carlo, in: Advances in Neural Information Processing Systems, 2018, pp. 7517–7527.
[19] G. Hernández-Muñoz, C. Villacampa-Calvo, D. Hernández-Lobato, Deep Gaussian processes using expectation propagation and Monte Carlo methods, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2020, pp. 479–494.
[20] C.M. Bishop, Pattern Recognition and Machine Learning, Information Science and Statistics, Springer-Verlag New York, Inc., ISBN 0-387-31073-8, 2006.
[21] J.M. Hernández-Lobato, Y. Li, M. Rowland, T.D. Bui, D. Hernández-Lobato, R.E. Turner, Black-box alpha divergence minimization, in: International Conference on Machine Learning, 2016, pp. 1511–1520.
[22] T. Minka, Power EP, Technical report, Microsoft Research, 2004.
[23] T.D. Bui, J. Yan, R.E. Turner, A unifying framework for Gaussian process pseudo-point approximations using power expectation propagation, J. Mach. Learn. Res. 18 (104) (2017) 1–72.
[24] C. Villacampa-Calvo, D. Hernández-Lobato, Alpha divergence minimization in multi-class Gaussian process classification, Neurocomputing 378 (2020) 210–227.
[25] C. Villacampa-Calvo, B. Zaldívar, E.C. Garrido-Merchán, D. Hernández-Lobato, Multi-class Gaussian process classification with noisy inputs, J. Mach. Learn. Res. 22 (36) (2021) 1–52.
[26] K.P. Murphy, Machine Learning: A Probabilistic Perspective, The MIT Press, ISBN 978-0-262-01802-9, 2012.
[27] C.K.I. Williams, D. Barber, Bayesian classification with Gaussian processes, IEEE Trans. Pattern Anal. Mach. Intell. 20 (1998) 1342–1351.
[28] T. Minka, Expectation propagation for approximate Bayesian inference, in: Proceedings of the 17th Annual Conference on Uncertainty in Artificial Intelligence, 2001, pp. 362–369.
[29] M.N. Gibbs, D.J.C. MacKay, Variational Gaussian process classifiers, IEEE Trans. Neural Netw. 11 (2000) 1458–1464.
[30] J. Quiñonero-Candela, C.E. Rasmussen, A unifying view of sparse approximate Gaussian process regression, J. Mach. Learn. Res. 6 (2005) 1939–1959.
[31] A. Naish-Guzman, S. Holden, The generalized FITC approximation, in: Advances in Neural Information Processing Systems, 2008, pp. 1057–1064.
[32] J. Hensman, A. Matthews, Z. Ghahramani, Scalable variational Gaussian process classification, in: Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, 2015.
[33] M. Bauer, M. van der Wilk, C.E. Rasmussen, Understanding probabilistic sparse Gaussian process approximations, in: Advances in Neural Information Processing Systems, vol. 29, 2016, pp. 1533–1541.
[34] T. Minka, Divergence Measures and Message Passing, Technical report, Microsoft Research, 2005.
[35] S. Amari, Differential-Geometrical Methods in Statistics, vol. 28, Springer Science & Business Media, 1985.
[36] Y. Matsuyama, The α-em algorithm: surrogate likelihood maximization using α-logarithmic information measures, IEEE Trans. Inf. Theory 49 (3) (2003) 692–706, https://doi.org/10.1109/TIT.2002.808105.
[37] Y. Li, Y. Gal, Dropout inference in Bayesian neural networks with alpha-divergences, in: International Conference on Machine Learning, 2017, pp. 2052–2061.
[38] S. Rodríguez-Santana, D. Hernández-Lobato, Adversarial α-divergence minimization for Bayesian approximate inference, Neurocomputing (ISSN 0925-2312) 471 (2022) 260–274, https://doi.org/10.1016/j.neucom.2020.09.076.

Table C.7
Regression test log-likelihood (the higher the better) results for UCI datasets. Averages over 20 splits and standard errors.

	Problem	$\alpha \to 0$	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$	$\alpha = 0.6$	$\alpha = 0.7$	$\alpha = 0.8$	$\alpha = 0.9$	$\alpha = 1$
$L = 2$	BOSTON	-2.4 ± 0.05	-2.38 ± 0.05	-2.38 ± 0.05	-2.34 ± 0.05	-2.33 ± 0.05	-2.3 ± 0.05	-2.29 ± 0.04	-2.27 ± 0.05	**-2.25 ± 0.04**	-2.26 ± 0.04	-2.29 ± 0.04
	POWER	-2.83 ± 0.01	-2.83 ± 0.01	-2.83 ± 0.01	-2.82 ± 0.01	-2.82 ± 0.01	-2.82 ± 0.01	-2.82 ± 0.01	-2.81 ± 0.01	-2.81 ± 0.01	-2.8 ± 0.01	**-2.76 ± 0.01**
	CONCRETE	-3.02 ± 0.01	-3.01 ± 0.02	-3.01 ± 0.02	-3.0 ± 0.02	-3.0 ± 0.02	-2.98 ± 0.02	-2.97 ± 0.02	-2.96 ± 0.03	-2.93 ± 0.03	-2.91 ± 0.03	**-2.89 ± 0.03**
	ENERGY	-0.94 ± 0.01	-0.9 ± 0.01	-0.88 ± 0.01	-0.86 ± 0.01	-0.84 ± 0.01	-0.82 ± 0.01	-0.8 ± 0.01	-0.78 ± 0.01	-0.74 ± 0.02	-0.71 ± 0.02	**-0.68 ± 0.02**
	KIN8NM	1.23 ± 0	1.25 ± 0	1.26 ± 0	1.27 ± 0	1.28 ± 0	1.29 ± 0	1.3 ± 0	1.31 ± 0	1.33 ± 0	1.34 ± 0	**1.36 ± 0.01**
	NAVAL	6.37 ± 0.01	6.39 ± 0.02	6.45 ± 0.02	6.49 ± 0.01	6.53 ± 0.02	6.58 ± 0.01	6.6 ± 0.01	6.64 ± 0.02	6.66 ± 0.02	**6.69 ± 0.01**	6.63 ± 0.03
	WINE RED	-0.96 ± 0.01	-0.96 ± 0.01	-0.96 ± 0.02	-0.95 ± 0.01	-0.95 ± 0.01	-0.95 ± 0.01	-0.95 ± 0.01	-0.95 ± 0.02	-0.94 ± 0.01	-0.94 ± 0.02	**0.19 ± 0.09**
	YACHT	-1.04 ± 0.01	-0.93 ± 0.02	-0.81 ± 0.02	-0.73 ± 0.02	-0.66 ± 0.02	-0.59 ± 0.02	-0.52 ± 0.02	-0.46 ± 0.02	-0.41 ± 0.02	-0.36 ± 0.02	**-0.31 ± 0.02**
$L = 3$	BOSTON	-2.45 ± 0.09	-2.4 ± 0.05	-2.45 ± 0.08	-2.41 ± 0.07	-2.33 ± 0.06	-2.33 ± 0.07	-2.34 ± 0.05	-2.33 ± 0.07	-2.32 ± 0.05	**-2.3 ± 0.05**	**-2.3 ± 0.05**
	POWER	-2.82 ± 0.01	-2.82 ± 0.01	-2.82 ± 0.01	-2.81 ± 0.01	-2.81 ± 0.01	-2.81 ± 0.01	-2.81 ± 0.01	-2.8 ± 0.01	-2.79 ± 0.01	-2.77 ± 0.01	**-2.73 ± 0.01**
	CONCRETE	-2.98 ± 0.03	-2.97 ± 0.03	-2.97 ± 0.03	-2.96 ± 0.03	-2.97 ± 0.03	-2.96 ± 0.04	**-2.93 ± 0.04**	-2.94 ± 0.04	-2.97 ± 0.04	-3.01 ± 0.06	-3.08 ± 0.08
	ENERGY	-0.83 ± 0.02	-0.81 ± 0.01	-0.81 ± 0.02	-0.79 ± 0.02	-0.77 ± 0.02	-0.75 ± 0.02	-0.72 ± 0.02	-0.7 ± 0.02	-0.68 ± 0.03	-0.65 ± 0.03	**-0.62 ± 0.04**
	KIN8NM	1.29 ± 0	1.3 ± 0	1.31 ± 0	1.31 ± 0	1.31 ± 0	1.32 ± 0	1.32 ± 0	1.32 ± 0.01	1.33 ± 0.01	1.34 ± 0.01	**1.35 ± 0.01**
	NAVAL	6.35 ± 0.02	6.31 ± 0.07	6.37 ± 0.03	6.46 ± 0.02	6.47 ± 0.05	6.48 ± 0.06	6.48 ± 0.05	6.47 ± 0.05	**6.52 ± 0.04**	6.44 ± 0.07	6.48 ± 0.05
	WINE RED	-0.98 ± 0.02	-0.98 ± 0.02	-0.98 ± 0.02	-0.97 ± 0.02	-0.98 ± 0.02	-0.98 ± 0.02	-0.97 ± 0.02	-0.96 ± 0.01	-0.96 ± 0.02	-0.96 ± 0.02	**1.04 ± 0.08**
	YACHT	-0.76 ± 0.02	-0.64 ± 0.02	-0.53 ± 0.02	-0.47 ± 0.02	-0.41 ± 0.03	-0.36 ± 0.03	-0.33 ± 0.04	-0.3 ± 0.03	-0.26 ± 0.03	-0.22 ± 0.04	**-0.2 ± 0.03**
$L = 4$	BOSTON	-2.37 ± 0.07	-2.38 ± 0.06	-2.41 ± 0.05	-2.42 ± 0.06	-2.38 ± 0.04	-2.34 ± 0.04	-2.32 ± 0.05	-2.3 ± 0.04	**-2.29 ± 0.04**	**-2.29 ± 0.05**	-2.31 ± 0.06
	POWER	-2.82 ± 0.01	-2.82 ± 0.01	-2.82 ± 0.01	-2.81 ± 0.01	-2.81 ± 0.01	-2.81 ± 0.01	-2.8 ± 0.01	-2.79 ± 0.01	-2.78 ± 0.01	-2.77 ± 0.01	**-2.74 ± 0.01**
	CONCRETE	-3.0 ± 0.03	-3.02 ± 0.02	-3.01 ± 0.03	-3.0 ± 0.03	-3.01 ± 0.03	-2.99 ± 0.03	-3.0 ± 0.04	**-2.97 ± 0.04**	-2.99 ± 0.05	-3.05 ± 0.05	-2.98 ± 0.06
	ENERGY	-0.82 ± 0.01	-0.82 ± 0.01	-0.81 ± 0.02	-0.79 ± 0.02	-0.74 ± 0.02	-0.74 ± 0.02	-0.73 ± 0.02	-0.71 ± 0.02	-0.67 ± 0.02	-0.65 ± 0.02	**-0.62 ± 0.02**
	KIN8NM	1.3 ± 0.01	1.31 ± 0	1.31 ± 0.01	1.32 ± 0.01	1.32 ± 0	1.32 ± 0.01	1.32 ± 0.01	1.33 ± 0.01	1.33 ± 0.01	1.34 ± 0	**1.35 ± 0.01**
	NAVAL	6.18 ± 0.05	6.15 ± 0.08	6.25 ± 0.04	6.29 ± 0.06	6.26 ± 0.06	6.26 ± 0.06	6.2 ± 0.09	6.35 ± 0.08	6.4 ± 0.08	6.34 ± 0.09	**6.49 ± 0.07**
	WINE RED	-0.98 ± 0.02	-0.97 ± 0.02	-0.98 ± 0.02	-0.96 ± 0.01	-0.96 ± 0.01	-0.95 ± 0.01	-0.95 ± 0.01	-0.95 ± 0.01	-0.94 ± 0.02	-0.94 ± 0.01	**0.92 ± 0.05**
	YACHT	-0.78 ± 0.02	-0.66 ± 0.02	-0.6 ± 0.03	-0.51 ± 0.02	-0.45 ± 0.02	-0.39 ± 0.02	-0.34 ± 0.03	-0.28 ± 0.03	-0.25 ± 0.02	-0.24 ± 0.03	**-0.23 ± 0.03**
$L = 5$	BOSTON	-2.34 ± 0.06	-2.68 ± 0.06	-2.74 ± 0.07	-2.77 ± 0.09	-2.79 ± 0.09	-2.77 ± 0.08	-2.68 ± 0.08	-2.59 ± 0.06	-2.46 ± 0.04	-2.37 ± 0.04	**-2.31 ± 0.05**
	POWER	-2.83 ± 0.01	-2.82 ± 0.01	-2.82 ± 0.01	-2.81 ± 0.01	-2.81 ± 0.01	-2.81 ± 0.01	-2.8 ± 0.01	-2.79 ± 0.01	-2.79 ± 0.01	-2.77 ± 0.01	**-2.74 ± 0.01**
	CONCRETE	-3.01 ± 0.04	-3.02 ± 0.03	-3.02 ± 0.03	-3.02 ± 0.03	-2.98 ± 0.03	-3.0 ± 0.03	-2.97 ± 0.04	-2.96 ± 0.04	-2.96 ± 0.03	**-2.95 ± 0.04**	-2.97 ± 0.04
	ENERGY	-0.83 ± 0.02	-0.84 ± 0.01	-0.82 ± 0.01	-0.81 ± 0.02	-0.79 ± 0.02	-0.79 ± 0.02	-0.79 ± 0.01	-0.79 ± 0.02	-0.77 ± 0.02	-0.76 ± 0.02	**-0.72 ± 0.02**
	KIN8NM	1.31 ± 0.01	1.32 ± 0.01	1.32 ± 0	1.32 ± 0.01	1.32 ± 0.01	1.32 ± 0.01	1.33 ± 0	1.33 ± 0.01	1.34 ± 0.01	**1.35 ± 0.01**	**1.35 ± 0.01**
	NAVAL	6.08 ± 0.06	6.25 ± 0.03	6.29 ± 0.02	6.25 ± 0.04	6.28 ± 0.03	6.28 ± 0.03	6.21 ± 0.07	6.27 ± 0.06	6.37 ± 0.03	6.24 ± 0.08	**6.4 ± 0.05**
	WINE RED	-1.0 ± 0.02	-0.97 ± 0.01	-0.97 ± 0.01	-0.95 ± 0.01	-0.96 ± 0.01	-0.96 ± 0.01	-0.95 ± 0.01	-0.95 ± 0.01	-0.94 ± 0.01	-0.94 ± 0.01	**1.12 ± 0.03**
	YACHT	-0.86 ± 0.04	-0.71 ± 0.02	-0.62 ± 0.02	-0.53 ± 0.02	-0.46 ± 0.02	-0.4 ± 0.02	-0.36 ± 0.02	-0.33 ± 0.03	-0.3 ± 0.02	-0.29 ± 0.02	**-0.28 ± 0.03**

Table C.8
Regression RMSE (the lower the better) results for UCI datasets. Averages over 20 splits and standard errors.

	Problem	$\alpha \to 0$	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$	$\alpha = 0.6$	$\alpha = 0.7$	$\alpha = 0.8$	$\alpha = 0.9$	$\alpha = 1$
$L = 2$	BOSTON	2.81 ± 0.2	2.81 ± 0.2	2.85 ± 0.2	2.77 ± 0.2	2.8 ± 0.2	2.71 ± 0.1	2.7 ± 0.1	**2.64 ± 0.2**	2.72 ± 0.1	2.69 ± 0.2	2.78 ± 0.2
	POWER	4.07 ± 0.04	4.06 ± 0.04	4.06 ± 0.04	4.06 ± 0.04	4.05 ± 0.04	4.04 ± 0.04	4.04 ± 0.03	**4.03 ± 0.04**	**4.03 ± 0.03**	**4.03 ± 0.04**	**4.03 ± 0.03**
	CONCRETE	4.98 ± 0.1	4.98 ± 0.1	4.99 ± 0.09	4.94 ± 0.1	4.96 ± 0.1	4.94 ± 0.1	4.93 ± 0.1	**4.91 ± 0.1**	**4.91 ± 0.1**	4.92 ± 0.1	4.98 ± 0.1
	ENERGY	0.51 ± 0.02	0.51 ± 0.01	0.5 ± 0.01	0.51 ± 0.01	0.5 ± 0.01	0.5 ± 0.01	0.5 ± 0.01	0.5 ± 0.01	**0.49 ± 0.01**	**0.49 ± 0.01**	**0.49 ± 0.01**
	KIN8NM	**0.07 ± 0**	**0.07 ± 0**	**0.07 ± 0**	**0.07 ± 0**	**0.07 ± 0**	**0.07 ± 0**	**0.07 ± 0**	**0.07 ± 0**	**0.07 ± 0**	**0.07 ± 0**	**0.07 ± 0**
	NAVAL	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**
	WINE RED	**0.63 ± 0.01**	**0.63 ± 0.01**	**0.63 ± 0.01**	**0.63 ± 0.01**	**0.63 ± 0.01**	**0.63 ± 0.01**	**0.63 ± 0.01**	**0.63 ± 0.01**	**0.63 ± 0.01**	**0.63 ± 0.01**	0.67 ± 0.01
	YACHT	0.52 ± 0.02	**0.49 ± 0.03**	0.52 ± 0.03	0.53 ± 0.03	0.53 ± 0.04	0.53 ± 0.03	0.55 ± 0.04	0.55 ± 0.03	0.56 ± 0.04	0.56 ± 0.04	0.57 ± 0.04
$L = 3$	BOSTON	2.81 ± 0.2	2.92 ± 0.2	2.86 ± 0.2	2.84 ± 0.2	2.78 ± 0.2	**2.71 ± 0.1**	2.72 ± 0.1	2.92 ± 0.2	2.9 ± 0.2	2.91 ± 0.2	2.86 ± 0.2
	POWER	4.06 ± 0.04	4.04 ± 0.03	4.04 ± 0.03	4.04 ± 0.04	4.03 ± 0.04	4.03 ± 0.04	4.03 ± 0.03	4.03 ± 0.04	4.03 ± 0.03	**4.02 ± 0.03**	4.03 ± 0.04
	CONCRETE	4.87 ± 0.2	**4.74 ± 0.2**	4.81 ± 0.2	4.79 ± 0.1	4.87 ± 0.1	4.91 ± 0.1	4.91 ± 0.1	5.04 ± 0.1	5.13 ± 0.1	5.15 ± 0.1	5.26 ± 0.2
	ENERGY	0.51 ± 0.02	0.5 ± 0.01	0.5 ± 0.01	0.49 ± 0.02	0.49 ± 0.02	0.49 ± 0.02	**0.48 ± 0.02**	**0.48 ± 0.02**	**0.48 ± 0.02**	**0.48 ± 0.02**	0.49 ± 0.02
	KIN8NM	**0.07 ± 0**	**0.07 ± 0**	**0.07 ± 0**	**0.07 ± 0**	**0.07 ± 0**	**0.07 ± 0**	**0.07 ± 0**	**0.07 ± 0**	**0.07 ± 0**	**0.07 ± 0**	**0.07 ± 0**
	NAVAL	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**
	WINE RED	**0.64 ± 0.01**	**0.64 ± 0.01**	**0.64 ± 0.01**	**0.64 ± 0.01**	**0.64 ± 0.01**	0.65 ± 0.01	**0.64 ± 0.01**	**0.64 ± 0.01**	**0.64 ± 0.01**	**0.64 ± 0.01**	0.68 ± 0.01
	YACHT	0.48 ± 0.03	0.46 ± 0.04	0.46 ± 0.03	0.46 ± 0.04	0.46 ± 0.04	**0.45 ± 0.04**	0.49 ± 0.04	0.54 ± 0.05	0.58 ± 0.05	0.6 ± 0.05	0.64 ± 0.05
$L = 4$	BOSTON	**2.67 ± 0.1**	2.74 ± 0.1	2.76 ± 0.1	3.04 ± 0.3	2.99 ± 0.2	2.94 ± 0.2	2.93 ± 0.2	2.92 ± 0.2	2.94 ± 0.2	2.95 ± 0.2	2.99 ± 0.2
	POWER	4.07 ± 0.04	4.06 ± 0.04	4.05 ± 0.04	4.05 ± 0.03	4.04 ± 0.04	**4.03 ± 0.03**	**4.03 ± 0.04**	**4.03 ± 0.03**	**4.03 ± 0.03**	**4.03 ± 0.04**	4.04 ± 0.03
	CONCRETE	5.04 ± 0.1	4.96 ± 0.1	4.98 ± 0.1	**4.91 ± 0.1**	5.02 ± 0.1	4.97 ± 0.1	5.11 ± 0.1	5.09 ± 0.1	5.16 ± 0.1	5.2 ± 0.2	5.32 ± 0.1
	ENERGY	0.5 ± 0.02	0.5 ± 0.01	0.51 ± 0.02	0.5 ± 0.01	**0.48 ± 0.01**	0.49 ± 0.01	0.49 ± 0.01	0.49 ± 0.02	**0.48 ± 0.01**	0.49 ± 0.01	0.49 ± 0.01
	KIN8NM	**0.07 ± 0**	**0.07 ± 0**	**0.07 ± 0**	**0.07 ± 0**	**0.07 ± 0**	**0.07 ± 0**	**0.07 ± 0**	**0.07 ± 0**	**0.07 ± 0**	**0.07 ± 0**	**0.07 ± 0**
	NAVAL	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**
	WINE RED	0.64 ± 0.01	0.64 ± 0.01	0.64 ± 0.01	**0.63 ± 0.01**	**0.63 ± 0.01**	**0.63 ± 0.01**	**0.63 ± 0.01**	**0.63 ± 0.01**	**0.63 ± 0.01**	**0.63 ± 0.01**	0.68 ± 0.01
	YACHT	**0.47 ± 0.03**	0.48 ± 0.03	0.48 ± 0.02	0.48 ± 0.03	0.48 ± 0.03	0.49 ± 0.03	0.49 ± 0.04	0.5 ± 0.04	0.56 ± 0.04	0.6 ± 0.06	0.62 ± 0.06
$L = 5$	BOSTON	**2.71 ± 0.1**	3.67 ± 0.3	3.98 ± 0.3	4.04 ± 0.4	4.29 ± 0.4	4.23 ± 0.4	3.98 ± 0.3	3.71 ± 0.3	3.37 ± 0.2	3.25 ± 0.2	3.04 ± 0.2
	POWER	4.09 ± 0.04	4.05 ± 0.04	4.05 ± 0.04	4.05 ± 0.04	4.05 ± 0.03	**4.04 ± 0.04**	4.06 ± 0.04	4.05 ± 0.03	**4.04 ± 0.04**	4.05 ± 0.04	4.06 ± 0.04
	CONCRETE	5.01 ± 0.1	**4.92 ± 0.1**	5.01 ± 0.2	5.04 ± 0.2	4.97 ± 0.1	5.07 ± 0.1	5.02 ± 0.1	5.12 ± 0.1	5.15 ± 0.1	5.17 ± 0.1	5.32 ± 0.1
	ENERGY	0.51 ± 0.02	**0.5 ± 0.01**	**0.5 ± 0.01**	0.51 ± 0.01	**0.5 ± 0.01**	0.51 ± 0.02	0.53 ± 0.01	0.55 ± 0.01	0.53 ± 0.02	0.56 ± 0.02	0.57 ± 0.02
	KIN8NM	0.07 ± 0	0.07 ± 0	0.07 ± 0	**0.06 ± 0**	0.07 ± 0	0.07 ± 0	0.07 ± 0	0.07 ± 0	0.07 ± 0	0.07 ± 0	0.07 ± 0
	NAVAL	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**
	WINE RED	0.65 ± 0.01	0.64 ± 0.01	0.64 ± 0.01	**0.63 ± 0.01**	**0.63 ± 0.01**	**0.63 ± 0.01**	**0.63 ± 0.01**	**0.63 ± 0.01**	**0.63 ± 0.01**	**0.63 ± 0.01**	0.68 ± 0.01
	YACHT	0.56 ± 0.07	0.48 ± 0.03	0.47 ± 0.03	**0.46 ± 0.03**	**0.46 ± 0.03**	**0.46 ± 0.04**	0.48 ± 0.03	0.5 ± 0.04	0.55 ± 0.04	0.59 ± 0.04	0.59 ± 0.04

Table C.9
Binary classification test log-likelihood (the higher the better) results for UCI datasets. Averages over 20 splits and standard errors.

	Problem	$\alpha \to 0$	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$	$\alpha = 0.6$	$\alpha = 0.7$	$\alpha = 0.8$	$\alpha = 0.9$	$\alpha = 1$
$L = 2$	AUSTRALIAN	**-0.62 ± 0.01**	**-0.62 ± 0.01**	**-0.62 ± 0.01**	**-0.62 ± 0.01**	**-0.62 ± 0.01**	-0.63 ± 0.01	-0.63 ± 0.01	-0.63 ± 0.01	-0.64 ± 0.01	-0.64 ± 0.01	-0.65 ± 0.01
	IONO	**-0.19 ± 0.02**	-0.2 ± 0.02	-0.2 ± 0.02	-0.2 ± 0.02	-0.2 ± 0.02	-0.2 ± 0.02	-0.2 ± 0.02	-0.2 ± 0.02	**-0.19 ± 0.02**	**-0.19 ± 0.02**	**-0.19 ± 0.02**
	PIMA	**-0.46 ± 0.01**	**-0.46 ± 0.01**	**-0.46 ± 0.01**	**-0.46 ± 0.01**	**-0.46 ± 0.01**	**-0.46 ± 0.01**	**-0.46 ± 0.01**	**-0.46 ± 0.01**	**-0.46 ± 0.01**	-0.47 ± 0.01	-0.47 ± 0.01
	SONAR	-0.48 ± 0.05	-0.47 ± 0.04	**-0.46 ± 0.05**	-0.48 ± 0.04	-0.49 ± 0.04	-0.5 ± 0.05	-0.5 ± 0.05	-0.5 ± 0.05	-0.49 ± 0.05	-0.49 ± 0.04	-0.49 ± 0.05
	BREAST	**-0.1 ± 0.01**	**-0.1 ± 0.01**	**-0.1 ± 0.01**	**-0.1 ± 0.01**	**-0.1 ± 0.01**	**-0.1 ± 0.01**	**-0.1 ± 0.01**	**-0.1 ± 0.01**	**-0.1 ± 0.01**	**-0.1 ± 0.01**	**-0.1 ± 0.01**
	CRABS	**-0.0 ± 0**	**-0.0 ± 0**	**-0.0 ± 0**	**-0.0 ± 0**	**-0.0 ± 0**	**-0.0 ± 0**	-0.01 ± 0	-0.01 ± 0	-0.01 ± 0	-0.01 ± 0	-0.01 ± 0
$L = 3$	AUSTRALIAN	**-0.62 ± 0.01**	**-0.62 ± 0.01**	**-0.62 ± 0.01**	**-0.62 ± 0.01**	**-0.62 ± 0.01**	**-0.62 ± 0.01**	**-0.62 ± 0.01**	-0.63 ± 0.01	-0.63 ± 0.01	-0.64 ± 0.01	-0.65 ± 0.01
	IONO	-0.18 ± 0.02	-0.18 ± 0.01	-0.18 ± 0.01	-0.18 ± 0.02	-0.18 ± 0.01	-0.18 ± 0.01	**-0.17 ± 0.01**	**-0.17 ± 0.01**	**-0.17 ± 0.02**	**-0.17 ± 0.01**	**-0.17 ± 0.02**
	PIMA	-0.48 ± 0.01	-0.48 ± 0.01	-0.48 ± 0.01	**-0.47 ± 0.01**	-0.48 ± 0.01	-0.48 ± 0.01	-0.48 ± 0.01	-0.48 ± 0.01	-0.48 ± 0.01	-0.49 ± 0.02	-0.49 ± 0.01
	SONAR	**-0.49 ± 0.04**	-0.51 ± 0.03	**-0.49 ± 0.03**	-0.51 ± 0.05	-0.5 ± 0.04	**-0.49 ± 0.04**	-0.51 ± 0.04	-0.51 ± 0.04	-0.51 ± 0.04	-0.5 ± 0.04	**-0.49 ± 0.04**
	BREAST	**-0.1 ± 0.01**	**-0.1 ± 0.01**	**-0.1 ± 0.01**	**-0.1 ± 0.01**	**-0.1 ± 0.01**	**-0.1 ± 0.01**	**-0.1 ± 0.01**	**-0.1 ± 0.01**	-0.11 ± 0.01	-0.11 ± 0.01	-0.11 ± 0.01
	CRABS	**-0.01 ± 0**	**-0.01 ± 0**	**-0.01 ± 0**	**-0.01 ± 0**	**-0.01 ± 0**	**-0.01 ± 0**	**-0.01 ± 0**	**-0.01 ± 0**	**-0.01 ± 0**	**-0.01 ± 0**	**-0.01 ± 0**
$L = 4$	AUSTRALIAN	**-0.62 ± 0.01**	**-0.62 ± 0.01**	**-0.62 ± 0.01**	**-0.62 ± 0.01**	**-0.62 ± 0.01**	**-0.62 ± 0.01**	**-0.62 ± 0.01**	**-0.62 ± 0.01**	**-0.62 ± 0.01**	-0.63 ± 0.01	-0.64 ± 0.01
	IONO	**-0.2 ± 0.02**	**-0.2 ± 0.02**	**-0.2 ± 0.02**	**-0.2 ± 0.02**	**-0.2 ± 0.02**	**-0.2 ± 0.02**	-0.21 ± 0.02	-0.22 ± 0.02	-0.22 ± 0.02	-0.21 ± 0.02	-0.21 ± 0.02
	PIMA	-0.51 ± 0.02	-0.53 ± 0.02	-0.52 ± 0.02	-0.54 ± 0.02	-0.51 ± 0.02	-0.56 ± 0.06	-0.49 ± 0.02	-0.49 ± 0.02	-0.53 ± 0.05	**-0.48 ± 0.02**	**-0.48 ± 0.01**
	SONAR	-0.44 ± 0.05	-0.43 ± 0.06	-0.43 ± 0.04	-0.43 ± 0.05	-0.43 ± 0.05	-0.43 ± 0.05	-0.42 ± 0.05	-0.42 ± 0.04	-0.42 ± 0.05	-0.42 ± 0.04	**-0.41 ± 0.04**
	BREAST	**-0.1 ± 0.01**	**-0.1 ± 0.01**	**-0.1 ± 0.01**	**-0.1 ± 0.01**	**-0.1 ± 0.01**	**-0.1 ± 0.01**	**-0.1 ± 0.01**	**-0.1 ± 0.01**	**-0.1 ± 0.01**	**-0.1 ± 0.01**	**-0.1 ± 0.01**
	CRABS	**-0.01 ± 0**	**-0.01 ± 0**	**-0.01 ± 0**	**-0.01 ± 0**	**-0.01 ± 0**	**-0.01 ± 0**	**-0.01 ± 0**	**-0.01 ± 0**	**-0.01 ± 0**	**-0.01 ± 0**	**-0.01 ± 0**
$L = 5$	AUSTRALIAN	**-0.62 ± 0.01**	**-0.62 ± 0.01**	**-0.62 ± 0.01**	**-0.62 ± 0.01**	**-0.62 ± 0.01**	**-0.62 ± 0.01**	**-0.62 ± 0.01**	**-0.62 ± 0.01**	**-0.62 ± 0.01**	**-0.62 ± 0.01**	-0.63 ± 0.01
	IONO	**-0.19 ± 0.01**	**-0.19 ± 0.02**	-0.2 ± 0.02	-0.21 ± 0.02	-0.2 ± 0.02	-0.2 ± 0.02	-0.2 ± 0.02	-0.21 ± 0.02	-0.22 ± 0.03	-0.2 ± 0.02	-0.21 ± 0.02
	PIMA	-0.5 ± 0.03	-0.5 ± 0.03	-0.5 ± 0.03	-0.5 ± 0.04	**-0.47 ± 0.02**	**-0.47 ± 0.02**	**-0.47 ± 0.02**	-0.48 ± 0.02	-0.48 ± 0.02	-0.49 ± 0.02	-0.51 ± 0.02
	SONAR	-0.46 ± 0.06	-0.45 ± 0.06	-0.45 ± 0.05	-0.43 ± 0.05	-0.41 ± 0.04	**-0.4 ± 0.05**	-0.41 ± 0.04	-0.41 ± 0.05	-0.41 ± 0.04	-0.42 ± 0.04	-0.41 ± 0.04
	BREAST	**-0.09 ± 0.01**	**-0.09 ± 0.01**	**-0.09 ± 0.01**	**-0.09 ± 0.01**	-0.1 ± 0.01	-0.1 ± 0.01	-0.1 ± 0.01	-0.1 ± 0.01	-0.1 ± 0.01	-0.1 ± 0.01	-0.1 ± 0.01
	CRABS	**-0.01 ± 0**	**-0.01 ± 0**	**-0.01 ± 0**	**-0.01 ± 0**	**-0.01 ± 0**	**-0.01 ± 0**	**-0.01 ± 0**	**-0.01 ± 0**	**-0.01 ± 0**	**-0.01 ± 0**	**-0.01 ± 0**

Table C.10
Binary classification test error (the lower the better) results for UCI datasets. Averages over 20 splits and standard errors.

	Problem	$\alpha \to 0$	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$	$\alpha = 0.6$	$\alpha = 0.7$	$\alpha = 0.8$	$\alpha = 0.9$	$\alpha = 1$
$L = 2$	AUSTRALIAN	0.34 ± 0.01	0.34 ± 0.01	0.34 ± 0.01	0.34 ± 0.01	**0.33 ± 0.01**	0.34 ± 0.01	0.34 ± 0.01	0.34 ± 0.01	0.35 ± 0.01	0.36 ± 0.01	0.37 ± 0.01
	IONO	**0.06 ± 0.01**	0.07 ± 0.01	**0.06 ± 0.01**	0.07 ± 0.01	0.07 ± 0.01	0.07 ± 0.01	**0.06 ± 0.01**	**0.06 ± 0.01**	**0.06 ± 0.01**	**0.06 ± 0.01**	**0.06 ± 0.01**
	PIMA	**0.23 ± 0.01**	**0.23 ± 0.01**	**0.23 ± 0.01**	**0.23 ± 0.01**	**0.23 ± 0.01**	**0.23 ± 0.01**	**0.23 ± 0.01**	**0.23 ± 0.01**	**0.23 ± 0.01**	**0.23 ± 0.01**	**0.23 ± 0.01**
	SONAR	0.19 ± 0.02	0.19 ± 0.02	0.19 ± 0.02	0.19 ± 0.02	**0.18 ± 0.02**	**0.18 ± 0.02**	**0.18 ± 0.02**	0.19 ± 0.02	0.19 ± 0.02	0.2 ± 0.02	0.21 ± 0.02
	BREAST	**0.03 ± 0**	**0.03 ± 0**	**0.03 ± 0**	**0.03 ± 0**	**0.03 ± 0**	**0.03 ± 0**	**0.03 ± 0**	**0.03 ± 0**	**0.03 ± 0**	**0.03 ± 0**	**0.03 ± 0**
	CRABS	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**
$L = 3$	AUSTRALIAN	**0.33 ± 0.01**	0.34 ± 0.01	0.34 ± 0.01	0.34 ± 0.01	0.34 ± 0.01	0.34 ± 0.01	0.34 ± 0.01	0.34 ± 0.01	0.35 ± 0.01	0.35 ± 0.01	0.36 ± 0.01
	IONO	**0.06 ± 0.01**	**0.06 ± 0.01**	**0.06 ± 0.01**	**0.06 ± 0.01**	0.07 ± 0.01	0.07 ± 0.01	0.07 ± 0.01	**0.06 ± 0.01**	**0.06 ± 0.01**	0.07 ± 0.01	0.07 ± 0.01
	PIMA	0.23 ± 0.01	0.23 ± 0.01	0.23 ± 0.01	0.23 ± 0.01	0.24 ± 0.01	0.24 ± 0.01	0.24 ± 0.01	0.24 ± 0.01	0.24 ± 0.01	0.24 ± 0.01	**0.22 ± 0.01**
	SONAR	0.22 ± 0.02	0.23 ± 0.02	0.23 ± 0.02	0.23 ± 0.02	0.22 ± 0.02	0.21 ± 0.02	0.21 ± 0.02	0.21 ± 0.02	**0.2 ± 0.01**	0.21 ± 0.01	**0.2 ± 0.01**
	BREAST	0.04 ± 0	**0.03 ± 0**	**0.03 ± 0**	**0.03 ± 0**	**0.03 ± 0**	**0.03 ± 0**	**0.03 ± 0**	**0.03 ± 0**	**0.03 ± 0**	**0.03 ± 0**	**0.03 ± 0**
	CRABS	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**
$L = 4$	AUSTRALIAN	**0.33 ± 0.01**	**0.33 ± 0.02**	**0.33 ± 0.01**	0.34 ± 0.01	0.34 ± 0.01	**0.33 ± 0.01**	0.34 ± 0.01	0.34 ± 0.01	**0.33 ± 0.02**	0.34 ± 0.01	0.36 ± 0.01
	IONO	0.07 ± 0.01	0.07 ± 0.01	0.07 ± 0.01	0.07 ± 0.01	0.07 ± 0.01	**0.06 ± 0.01**	**0.06 ± 0.01**	**0.06 ± 0.01**	**0.06 ± 0.01**	0.07 ± 0.01	0.07 ± 0.01
	PIMA	0.24 ± 0.01	0.25 ± 0.01	**0.23 ± 0.01**	**0.23 ± 0.01**	**0.23 ± 0.01**	0.24 ± 0.01	0.24 ± 0.01	0.24 ± 0.01	0.24 ± 0.01	0.24 ± 0.01	**0.23 ± 0.01**
	SONAR	**0.15 ± 0.02**	**0.15 ± 0.02**	**0.15 ± 0.02**	0.16 ± 0.02	0.17 ± 0.02	0.17 ± 0.02	0.17 ± 0.01	0.17 ± 0.02	0.17 ± 0.02	0.17 ± 0.02	0.17 ± 0.02
	BREAST	**0.03 ± 0**	0.04 ± 0	**0.03 ± 0**	**0.03 ± 0**	**0.03 ± 0**	**0.03 ± 0**	**0.03 ± 0**	**0.03 ± 0**	0.04 ± 0	0.04 ± 0	0.04 ± 0
	CRABS	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**
$L = 5$	AUSTRALIAN	**0.32 ± 0.01**	**0.32 ± 0.01**	**0.32 ± 0.01**	**0.32 ± 0.01**	**0.32 ± 0.01**	0.33 ± 0.01	0.34 ± 0.01	0.34 ± 0.01	0.34 ± 0.01	0.33 ± 0.01	0.35 ± 0.01
	IONO	**0.07 ± 0.01**	**0.07 ± 0.01**	**0.07 ± 0.01**	0.08 ± 0.01	0.08 ± 0.01	**0.07 ± 0.01**	0.08 ± 0.01	**0.07 ± 0.01**	0.08 ± 0.01	0.08 ± 0.01	**0.07 ± 0.01**
	PIMA	**0.23 ± 0.01**	**0.23 ± 0.01**	**0.23 ± 0.01**	**0.23 ± 0.01**	**0.23 ± 0.01**	**0.23 ± 0.01**	**0.23 ± 0.01**	0.24 ± 0.01	0.24 ± 0.01	**0.23 ± 0.01**	0.24 ± 0.01
	SONAR	0.16 ± 0.02	0.16 ± 0.01	0.16 ± 0.01	**0.15 ± 0.01**	**0.15 ± 0.02**	0.16 ± 0.02	0.16 ± 0.02	0.17 ± 0.01	0.17 ± 0.01	0.17 ± 0.02	0.18 ± 0.01
	BREAST	**0.03 ± 0**	**0.03 ± 0**	**0.03 ± 0**	**0.03 ± 0**	**0.03 ± 0**	**0.03 ± 0**	**0.03 ± 0**	**0.03 ± 0**	**0.03 ± 0**	**0.03 ± 0**	**0.03 ± 0**
	CRABS	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**	**0.0 ± 0**

Table C.11
Multi-class classification test log-likelihood (the higher the better) results for UCI datasets. Averages over 20 splits and standard errors.

	Problem	$\alpha \to 0$	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$	$\alpha = 0.6$	$\alpha = 0.7$	$\alpha = 0.8$	$\alpha = 0.9$	$\alpha = 1$
$L = 2$	GLASS	-2.31 ± 0.2	-2.11 ± 0.2	-1.62 ± 0.1	-1.63 ± 0.1	-1.58 ± 0.1	-1.47 ± 0.1	-1.31 ± 0.1	-1.2 ± 0.1	-1.12 ± 0.08	-1.08 ± 0.07	**-1.04 ± 0.06**
	NEW-THYROID	-0.66 ± 0.1	-0.64 ± 0.1	-0.6 ± 0.2	-0.63 ± 0.2	-0.54 ± 0.1	-0.48 ± 0.1	-0.43 ± 0.1	-0.38 ± 0.08	-0.33 ± 0.07	-0.32 ± 0.07	**-0.3 ± 0.07**
	SVMGUIDE2	-1.02 ± 0.09	-0.97 ± 0.08	-0.87 ± 0.08	-0.8 ± 0.07	-0.76 ± 0.07	-0.75 ± 0.06	-0.69 ± 0.06	-0.67 ± 0.05	-0.67 ± 0.05	-0.66 ± 0.05	**-0.65 ± 0.05**
	SATELLITE	-0.69 ± 0.02	-0.6 ± 0.01	-0.51 ± 0.01	-0.44 ± 0.01	-0.38 ± 0.01	-0.36 ± 0.00	-0.35 ± 0.00	**-0.34 ± 0.00**	**-0.34 ± 0.00**	**-0.34 ± 0.00**	**-0.34 ± 0.00**
	VEHICLE	-0.83 ± 0.06	-0.69 ± 0.05	-0.56 ± 0.04	-0.5 ± 0.03	-0.44 ± 0.03	-0.42 ± 0.03	-0.41 ± 0.03	-0.4 ± 0.02	**-0.38 ± 0.02**	**-0.38 ± 0.02**	**-0.38 ± 0.02**
	VOWEL	-0.39 ± 0.05	-0.37 ± 0.04	-0.3 ± 0.04	-0.27 ± 0.03	-0.28 ± 0.03	**-0.26 ± 0.03**	**-0.26 ± 0.03**	-0.28 ± 0.03	-0.29 ± 0.02	-0.31 ± 0.03	-0.32 ± 0.03
	WAVEFORM	-0.86 ± 0.01	-0.79 ± 0.02	-0.71 ± 0.02	-0.6 ± 0.02	-0.53 ± 0.02	-0.48 ± 0.02	-0.45 ± 0.01	-0.43 ± 0.01	-0.41 ± 0.01	-0.4 ± 0.01	**-0.39 ± 0.01**
	WINE	-0.32 ± 0.07	-0.31 ± 0.07	-0.31 ± 0.07	-0.29 ± 0.07	-0.28 ± 0.05	-0.28 ± 0.06	-0.27 ± 0.06	-0.26 ± 0.06	-0.25 ± 0.05	-0.24 ± 0.05	**-0.23 ± 0.05**
$L = 3$	GLASS	-2.36 ± 0.2	-2.37 ± 0.2	-1.73 ± 0.2	-1.61 ± 0.1	-1.49 ± 0.1	-1.29 ± 0.1	-1.19 ± 0.1	-1.16 ± 0.09	-1.1 ± 0.09	-1.06 ± 0.07	**-1.04 ± 0.07**
	NEW-THYROID	-0.6 ± 0.1	-0.65 ± 0.1	-0.75 ± 0.2	-0.61 ± 0.1	-0.46 ± 0.1	-0.4 ± 0.1	-0.36 ± 0.09	-0.34 ± 0.08	-0.32 ± 0.06	-0.31 ± 0.07	**-0.29 ± 0.06**
	SVMGUIDE2	-1.01 ± 0.1	-0.98 ± 0.07	-0.88 ± 0.07	-0.78 ± 0.07	-0.78 ± 0.07	-0.75 ± 0.07	-0.71 ± 0.06	-0.7 ± 0.06	-0.68 ± 0.05	**-0.66 ± 0.06**	-0.67 ± 0.05
	SATELLITE	-0.72 ± 0.03	-0.59 ± 0.01	-0.51 ± 0.01	-0.42 ± 0.01	-0.38 ± 0.01	-0.36 ± 0.01	-0.35 ± 0.00	**-0.34 ± 0.00**	**-0.34 ± 0.00**	**-0.34 ± 0.00**	**-0.34 ± 0.00**
	VEHICLE	-0.81 ± 0.05	-0.71 ± 0.05	-0.57 ± 0.04	-0.49 ± 0.03	-0.46 ± 0.03	-0.44 ± 0.03	-0.41 ± 0.02	-0.39 ± 0.03	**-0.38 ± 0.02**	**-0.38 ± 0.02**	**-0.38 ± 0.02**
	VOWEL	-0.42 ± 0.04	-0.37 ± 0.05	-0.32 ± 0.04	-0.29 ± 0.04	-0.28 ± 0.03	**-0.26 ± 0.03**	-0.27 ± 0.03	-0.3 ± 0.03	-0.31 ± 0.03	-0.32 ± 0.03	-0.32 ± 0.03
	WAVEFORM	-0.88 ± 0.01	-0.8 ± 0.01	-0.7 ± 0.02	-0.59 ± 0.02	-0.52 ± 0.02	-0.47 ± 0.01	-0.45 ± 0.01	-0.43 ± 0.01	-0.41 ± 0.01	-0.4 ± 0.01	**-0.39 ± 0.01**
	WINE	-0.33 ± 0.08	-0.32 ± 0.08	-0.31 ± 0.07	-0.29 ± 0.07	-0.28 ± 0.06	-0.27 ± 0.07	-0.26 ± 0.05	-0.25 ± 0.06	-0.24 ± 0.06	-0.24 ± 0.05	**-0.23 ± 0.05**
$L = 4$	GLASS	-2.59 ± 0.2	-2.38 ± 0.2	-1.62 ± 0.1	-1.83 ± 0.2	-1.5 ± 0.1	-1.28 ± 0.1	-1.2 ± 0.1	-1.14 ± 0.09	-1.09 ± 0.07	-1.05 ± 0.08	**-1.03 ± 0.07**
	NEW-THYROID	-0.56 ± 0.1	-0.62 ± 0.2	-0.64 ± 0.1	-0.65 ± 0.2	-0.48 ± 0.1	-0.42 ± 0.1	-0.4 ± 0.09	-0.34 ± 0.07	-0.31 ± 0.08	-0.3 ± 0.07	**-0.29 ± 0.06**
	SVMGUIDE2	-1.04 ± 0.09	-0.97 ± 0.09	-0.89 ± 0.07	-0.81 ± 0.07	-0.78 ± 0.08	-0.75 ± 0.07	-0.71 ± 0.06	-0.69 ± 0.06	-0.68 ± 0.05	-0.68 ± 0.05	**-0.67 ± 0.06**
	SATELLITE	-0.72 ± 0.03	-0.57 ± 0.01	-0.51 ± 0.01	-0.43 ± 0.01	-0.38 ± 0.00	-0.36 ± 0.00	-0.35 ± 0.01	**-0.34 ± 0.00**	**-0.34 ± 0.00**	**-0.34 ± 0.00**	**-0.34 ± 0.00**
	VEHICLE	-0.93 ± 0.06	-0.83 ± 0.06	-0.62 ± 0.04	-0.53 ± 0.04	-0.49 ± 0.04	-0.44 ± 0.03	-0.4 ± 0.02	-0.38 ± 0.02	**-0.37 ± 0.02**	**-0.37 ± 0.02**	**-0.37 ± 0.02**
	VOWEL	-0.43 ± 0.05	-0.34 ± 0.04	-0.31 ± 0.04	-0.28 ± 0.04	**-0.26 ± 0.03**	-0.27 ± 0.03	-0.29 ± 0.03	-0.29 ± 0.02	-0.3 ± 0.02	-0.31 ± 0.02	-0.32 ± 0.03
	WAVEFORM	-0.9 ± 0.02	-0.82 ± 0.02	-0.7 ± 0.01	-0.59 ± 0.02	-0.52 ± 0.02	-0.47 ± 0.01	-0.44 ± 0.01	-0.43 ± 0.01	-0.41 ± 0.01	-0.4 ± 0.01	**-0.39 ± 0.01**
	WINE	-0.31 ± 0.07	-0.31 ± 0.06	-0.31 ± 0.08	-0.29 ± 0.07	-0.27 ± 0.06	-0.27 ± 0.06	-0.26 ± 0.06	-0.25 ± 0.06	-0.24 ± 0.05	**-0.23 ± 0.05**	**-0.23 ± 0.04**
$L = 5$	GLASS	-2.75 ± 0.2	-2.83 ± 0.2	-1.92 ± 0.2	-1.58 ± 0.1	-1.36 ± 0.1	-1.25 ± 0.09	-1.21 ± 0.09	-1.13 ± 0.08	-1.1 ± 0.07	-1.07 ± 0.07	**-1.04 ± 0.07**
	NEW-THYROID	-0.69 ± 0.1	-0.64 ± 0.1	-0.61 ± 0.1	-0.62 ± 0.1	-0.49 ± 0.1	-0.58 ± 0.1	-0.54 ± 0.1	-0.45 ± 0.09	-0.43 ± 0.08	-0.41 ± 0.07	**-0.31 ± 0.06**
	SVMGUIDE2	-1.1 ± 0.08	-1.03 ± 0.1	-0.91 ± 0.08	-0.82 ± 0.07	-0.79 ± 0.06	-0.75 ± 0.07	-0.71 ± 0.06	-0.7 ± 0.06	-0.7 ± 0.06	**-0.69 ± 0.06**	**-0.69 ± 0.06**
	SATELLITE	-0.79 ± 0.03	-0.6 ± 0.01	-0.51 ± 0.01	-0.44 ± 0.01	-0.38 ± 0.01	-0.36 ± 0.00	-0.35 ± 0.00	**-0.34 ± 0.00**	**-0.34 ± 0.00**	**-0.34 ± 0.00**	**-0.34 ± 0.00**
	VEHICLE	-0.96 ± 0.06	-0.93 ± 0.07	-0.86 ± 0.07	-0.75 ± 0.06	-0.59 ± 0.05	-0.54 ± 0.04	-0.39 ± 0.03	-0.4 ± 0.03	**-0.38 ± 0.02**	-0.39 ± 0.02	**-0.38 ± 0.02**
	VOWEL	-0.47 ± 0.07	-0.38 ± 0.05	-0.33 ± 0.05	-0.28 ± 0.04	**-0.27 ± 0.03**	-0.29 ± 0.03	-0.28 ± 0.03	-0.28 ± 0.02	-0.3 ± 0.03	-0.3 ± 0.02	-0.31 ± 0.02
	WAVEFORM	-0.94 ± 0.02	-0.82 ± 0.02	-0.72 ± 0.02	-0.59 ± 0.02	-0.51 ± 0.02	-0.47 ± 0.02	-0.44 ± 0.01	-0.42 ± 0.01	-0.41 ± 0.01	-0.4 ± 0.01	**-0.39 ± 0.01**
	WINE	-0.31 ± 0.07	-0.31 ± 0.08	-0.3 ± 0.06	-0.29 ± 0.07	-0.27 ± 0.06	-0.27 ± 0.06	-0.26 ± 0.06	-0.25 ± 0.06	-0.24 ± 0.05	**-0.23 ± 0.05**	**-0.23 ± 0.04**

Table C.12
Multi-class classification test error (the lower the better) results for UCI datasets. Averages over 20 splits and standard errors.

	Problem	$\alpha \to 0$	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$	$\alpha = 0.6$	$\alpha = 0.7$	$\alpha = 0.8$	$\alpha = 0.9$	$\alpha = 1$
$L = 2$	GLASS	0.43 ± 0.03	0.41 ± 0.03	0.48 ± 0.04	0.45 ± 0.02	0.46 ± 0.03	0.45 ± 0.03	0.47 ± 0.04	0.45 ± 0.03	0.43 ± 0.03	**0.4 ± 0.03**	**0.4 ± 0.03**
	NEW-THYROID	0.11 ± 0.02	0.12 ± 0.02	0.13 ± 0.03	0.14 ± 0.03	0.13 ± 0.03	0.13 ± 0.03	0.13 ± 0.03	0.12 ± 0.03	**0.09 ± 0.02**	**0.09 ± 0.02**	**0.09 ± 0.02**
	SVMGUIDE2	**0.19 ± 0.02**	0.2 ± 0.01	**0.19 ± 0.02**	0.2 ± 0.01	0.2 ± 0.01	0.21 ± 0.01	0.21 ± 0.01	0.2 ± 0.01	0.2 ± 0.01	0.2 ± 0.01	**0.19 ± 0.01**
	SATELLITE	0.13 ± 0.00	**0.12 ± 0.00**	**0.12 ± 0.00**	**0.12 ± 0.00**	**0.12 ± 0.00**	**0.12 ± 0.00**	**0.12 ± 0.00**	**0.12 ± 0.00**	**0.12 ± 0.00**	**0.12 ± 0.00**	**0.12 ± 0.00**
	VEHICLE	0.18 ± 0.01	0.18 ± 0.01	0.18 ± 0.01	**0.17 ± 0.01**	**0.17 ± 0.01**	0.18 ± 0.01	0.18 ± 0.01	0.18 ± 0.01	0.18 ± 0.01	0.18 ± 0.01	0.18 ± 0.01
	VOWEL	0.1 ± 0.01	0.1 ± 0.01	0.1 ± 0.01	**0.09 ± 0.01**	**0.09 ± 0.01**	**0.09 ± 0.01**	**0.09 ± 0.01**	0.1 ± 0.01	**0.09 ± 0.01**	0.11 ± 0.01	0.11 ± 0.01
	WAVEFORM	0.19 ± 0.00	0.19 ± 0.00	0.19 ± 0.00	0.18 ± 0.00	0.18 ± 0.00	0.17 ± 0.00	0.17 ± 0.00	0.17 ± 0.00	**0.16 ± 0.00**	**0.16 ± 0.00**	**0.16 ± 0.00**
	WINE	0.07 ± 0.01	0.07 ± 0.01	0.07 ± 0.01	0.07 ± 0.01	0.07 ± 0.01	0.07 ± 0.01	0.07 ± 0.01	**0.06 ± 0.01**	0.07 ± 0.01	0.07 ± 0.01	**0.06 ± 0.01**
$L = 3$	GLASS	**0.39 ± 0.02**	0.46 ± 0.03	0.48 ± 0.03	0.46 ± 0.02	0.46 ± 0.03	0.49 ± 0.04	0.45 ± 0.04	0.45 ± 0.04	0.43 ± 0.03	0.4 ± 0.03	0.4 ± 0.03
	NEW-THYROID	0.13 ± 0.02	0.13 ± 0.02	0.15 ± 0.04	0.14 ± 0.03	0.11 ± 0.02	0.11 ± 0.03	0.1 ± 0.03	0.1 ± 0.02	**0.09 ± 0.02**	**0.09 ± 0.02**	**0.09 ± 0.02**
	SVMGUIDE2	0.2 ± 0.02	0.2 ± 0.01	0.2 ± 0.01	**0.19 ± 0.01**	0.2 ± 0.01	0.2 ± 0.02	0.21 ± 0.01	0.2 ± 0.01	0.2 ± 0.01	0.2 ± 0.01	0.2 ± 0.01
	SATELLITE	0.13 ± 0.00	**0.12 ± 0.00**	**0.12 ± 0.00**	**0.12 ± 0.00**	**0.12 ± 0.00**	**0.12 ± 0.00**	**0.12 ± 0.00**	**0.12 ± 0.00**	**0.12 ± 0.00**	**0.12 ± 0.00**	0.13 ± 0.00
	VEHICLE	0.19 ± 0.01	**0.18 ± 0.01**	**0.18 ± 0.01**	**0.18 ± 0.01**	**0.18 ± 0.01**	**0.18 ± 0.01**	**0.18 ± 0.01**	**0.18 ± 0.01**	**0.18 ± 0.01**	**0.18 ± 0.01**	**0.18 ± 0.01**
	VOWEL	0.11 ± 0.01	**0.09 ± 0.01**	0.1 ± 0.01	**0.09 ± 0.01**	**0.09 ± 0.02**	**0.09 ± 0.01**	**0.09 ± 0.01**	0.1 ± 0.01	0.11 ± 0.01	0.11 ± 0.01	0.12 ± 0.01
	WAVEFORM	0.19 ± 0.00	0.19 ± 0.00	0.18 ± 0.00	0.18 ± 0.00	0.18 ± 0.00	0.17 ± 0.00	0.17 ± 0.00	0.17 ± 0.00	**0.16 ± 0.00**	**0.16 ± 0.00**	**0.16 ± 0.00**
	WINE	0.07 ± 0.01	0.07 ± 0.01	0.07 ± 0.01	0.07 ± 0.01	0.07 ± 0.01	0.07 ± 0.01	0.07 ± 0.01	**0.06 ± 0.01**	0.07 ± 0.01	**0.06 ± 0.01**	**0.06 ± 0.01**
$L = 4$	GLASS	0.44 ± 0.02	0.47 ± 0.03	0.45 ± 0.02	0.49 ± 0.03	0.46 ± 0.02	0.45 ± 0.03	0.47 ± 0.03	0.44 ± 0.03	0.41 ± 0.03	**0.4 ± 0.03**	0.41 ± 0.03
	NEW-THYROID	0.13 ± 0.02	0.14 ± 0.03	0.13 ± 0.02	0.15 ± 0.03	0.12 ± 0.02	0.11 ± 0.02	0.12 ± 0.03	0.1 ± 0.02	0.1 ± 0.02	**0.09 ± 0.02**	**0.09 ± 0.02**
	SVMGUIDE2	0.2 ± 0.01	0.21 ± 0.02	0.21 ± 0.01	0.2 ± 0.02	**0.19 ± 0.01**	0.2 ± 0.01	0.2 ± 0.01	0.2 ± 0.01	0.21 ± 0.01	0.2 ± 0.02	0.2 ± 0.01
	SATELLITE	0.13 ± 0.00	**0.12 ± 0.00**	**0.12 ± 0.00**	**0.12 ± 0.00**	**0.12 ± 0.00**	**0.12 ± 0.00**	**0.12 ± 0.00**	**0.12 ± 0.00**	**0.12 ± 0.00**	**0.12 ± 0.00**	0.13 ± 0.00
	VEHICLE	0.19 ± 0.01	0.19 ± 0.01	0.18 ± 0.01	0.18 ± 0.01	**0.17 ± 0.01**	0.18 ± 0.01	0.18 ± 0.01	0.18 ± 0.01	0.18 ± 0.01	0.18 ± 0.01	0.18 ± 0.01
	VOWEL	0.11 ± 0.01	**0.09 ± 0.01**	0.1 ± 0.01	**0.09 ± 0.01**	**0.09 ± 0.01**	**0.09 ± 0.01**	0.1 ± 0.01	0.1 ± 0.01	0.1 ± 0.01	0.11 ± 0.01	0.12 ± 0.01
	WAVEFORM	0.19 ± 0.00	0.19 ± 0.00	0.19 ± 0.00	0.18 ± 0.00	0.18 ± 0.00	0.17 ± 0.00	0.17 ± 0.00	0.17 ± 0.00	**0.16 ± 0.00**	**0.16 ± 0.00**	**0.16 ± 0.00**
	WINE	**0.06 ± 0.01**	0.07 ± 0.01	0.07 ± 0.01	0.07 ± 0.01	0.07 ± 0.01	0.07 ± 0.01	**0.06 ± 0.01**	**0.06 ± 0.02**	0.07 ± 0.01	**0.06 ± 0.01**	**0.06 ± 0.01**
$L = 5$	GLASS	0.46 ± 0.03	0.51 ± 0.03	0.46 ± 0.02	0.44 ± 0.02	0.46 ± 0.03	0.48 ± 0.04	0.45 ± 0.03	0.43 ± 0.04	0.42 ± 0.03	0.42 ± 0.04	**0.41 ± 0.03**
	NEW-THYROID	0.15 ± 0.03	0.15 ± 0.03	0.16 ± 0.04	0.16 ± 0.04	0.14 ± 0.03	0.16 ± 0.03	0.15 ± 0.03	0.12 ± 0.03	0.14 ± 0.03	0.13 ± 0.03	**0.09 ± 0.02**
	SVMGUIDE2	0.21 ± 0.01	0.21 ± 0.01	0.2 ± 0.02	0.2 ± 0.01	**0.19 ± 0.01**	0.2 ± 0.01	0.2 ± 0.01	0.21 ± 0.01	0.21 ± 0.01	0.21 ± 0.01	0.2 ± 0.01
	SATELLITE	0.14 ± 0.00	**0.12 ± 0.00**	**0.12 ± 0.00**	**0.12 ± 0.00**	**0.12 ± 0.00**	**0.12 ± 0.00**	**0.12 ± 0.00**	**0.12 ± 0.00**	**0.12 ± 0.00**	**0.12 ± 0.00**	0.13 ± 0.00
	VEHICLE	0.19 ± 0.01	0.2 ± 0.02	0.19 ± 0.01	0.2 ± 0.01	0.19 ± 0.01	0.19 ± 0.01	**0.18 ± 0.01**	0.19 ± 0.01	**0.18 ± 0.01**	**0.18 ± 0.01**	**0.18 ± 0.01**
	VOWEL	0.11 ± 0.01	0.11 ± 0.01	0.1 ± 0.01	**0.09 ± 0.01**	**0.09 ± 0.01**	0.1 ± 0.01	**0.09 ± 0.01**	0.1 ± 0.01	0.1 ± 0.01	0.1 ± 0.01	0.11 ± 0.01
	WAVEFORM	0.19 ± 0.00	0.19 ± 0.00	0.19 ± 0.00	0.18 ± 0.00	0.17 ± 0.00	0.17 ± 0.00	0.17 ± 0.00	**0.16 ± 0.00**	**0.16 ± 0.00**	**0.16 ± 0.00**	**0.16 ± 0.00**
	WINE	**0.06 ± 0.01**	0.07 ± 0.01	0.07 ± 0.01	0.07 ± 0.01	0.07 ± 0.01	0.07 ± 0.01	**0.06 ± 0.01**	**0.06 ± 0.01**	0.07 ± 0.01	**0.06 ± 0.01**	**0.06 ± 0.01**

Table C.13
Negative test log-likelihood (the lower the better) results for the big datasets.

	Problem	$\alpha \to 0$	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$	$\alpha = 0.6$	$\alpha = 0.7$	$\alpha = 0.8$	$\alpha = 0.9$	$\alpha = 1$
$L = 2$	YEAR	3.62	3.62	3.61	3.57	3.56	3.51	3.49	3.49	3.43	3.39	**3.35**
	HIGGS	0.52	0.52	0.52	0.52	**0.51**	**0.51**	**0.51**	**0.51**	**0.51**	**0.51**	**0.51**
	MNIST	0.14	0.12	0.11	0.11	0.1	0.09	0.09	0.09	**0.08**	0.09	0.1
$L = 3$	YEAR	3.63	3.65	3.58	3.56	3.54	3.49	3.46	3.44	3.40	3.37	**3.35**
	HIGGS	0.51	0.51	0.51	0.51	0.51	**0.5**	0.51	0.51	0.51	0.51	**0.5**
	MNIST	0.13	0.12	0.1	0.1	0.11	0.1	0.1	0.1	**0.09**	**0.09**	**0.09**
$L = 4$	YEAR	3.64	3.61	3.61	3.56	3.52	3.52	3.47	3.45	3.42	3.38	**3.37**
	HIGGS	**0.51**	**0.51**	**0.51**	**0.51**	**0.51**	**0.51**	**0.51**	**0.51**	**0.51**	**0.51**	**0.51**
	MNIST	0.13	0.13	0.12	0.11	0.11	0.12	0.11	0.11	0.09	**0.08**	0.1
$L = 5$	YEAR	3.61	3.62	3.60	3.57	3.56	3.51	3.47	3.45	3.43	3.39	**3.36**
	HIGGS	**0.51**	**0.51**	**0.51**	**0.51**	**0.51**	0.52	**0.51**	0.52	0.52	0.52	**0.51**
	MNIST	0.11	0.12	0.12	0.1	0.1	**0.09**	0.12	0.1	**0.09**	**0.09**	**0.09**

Table C.14
Test error (the lower the better) results for the big datasets. RMSE is reported for Year and classification error for the rest.

	Problem	$\alpha \to 0$	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$	$\alpha = 0.6$	$\alpha = 0.7$	$\alpha = 0.8$	$\alpha = 0.9$	$\alpha = 1$
$L = 2$	YEAR	**9.42**	9.47	9.49	10.08	9.76	10.12	9.97	10.14	9.87	9.67	10.15
	HIGGS	0.269	0.269	0.266	0.267	0.262	0.262	0.261	0.262	0.264	**0.256**	**0.256**
	MNIST	0.025	0.022	0.024	0.022	0.021	0.021	0.022	0.023	**0.02**	**0.02**	0.023
$L = 3$	YEAR	9.29	9.94	9.13	**9.11**	10.10	9.24	9.63	9.52	9.55	9.56	9.78
	HIGGS	0.257	0.256	0.258	0.258	0.262	0.255	0.261	0.257	0.257	0.256	**0.251**
	MNIST	0.023	0.023	0.02	**0.019**	0.02	0.024	0.023	0.023	0.02	0.022	0.023
$L = 4$	YEAR	9.59	**9.19**	9.97	9.71	9.50	9.92	9.36	9.69	9.96	10.06	10.14
	HIGGS	0.262	**0.258**	0.264	0.261	**0.258**	0.261	0.264	**0.258**	**0.258**	0.262	0.259
	MNIST	0.022	0.023	0.023	0.022	**0.021**	0.027	0.028	0.025	**0.021**	**0.021**	0.024
$L = 5$	YEAR	**9.13**	9.69	9.68	9.83	10.00	10.02	9.40	9.55	10.15	9.88	10.16
	HIGGS	0.261	0.26	0.263	**0.257**	0.262	0.268	0.26	0.266	0.264	0.265	0.258
	MNIST	**0.02**	0.023	0.024	0.021	0.023	0.022	0.028	0.024	0.022	0.023	0.021

[39] B.J. Frey, R. Patrascu, T.S. Jaakkola, J. Moran, Sequentially fitting "inclusive" trees for inference in noisy-OR networks, in: Neural Information Processing Systems, 2000, pp. 472–478.
[40] M. Seeger, Expectation propagation for exponential families, Technical report, Department of EECS, University of California, Berkeley, 2006.
[41] T. Heskes, O. Zoeter, Expectation propagation for approximate inference in dynamic Bayesian networks, in: Uncertainty in Artificial Intelligence, 2002, pp. 216–223.
[42] Y. Li, J.M. Hernández-Lobato, R.E. Turner, Stochastic expectation propagation, in: Advances in Neural Information Processing Systems, vol. 28, 2015, pp. 2323–2331.
[43] J. Shi, M.K. Titsias, A. Mnih, Sparse orthogonal variational inference for Gaussian processes, in: 23rd International Conference on Artificial Intelligence and Statistics (AISTATS), 2020.
[44] M. Havasi, J.M. Hernández-Lobato, J.J. Murillo-Fuentes, Deep Gaussian processes with decoupled inducing inputs, arXiv:1801.02939 [stat], 2018.
[45] H. Salimbeni, C.-A. Cheng, B. Boots, M. Deisenroth, Orthogonally decoupled variational Gaussian processes, in: 32nd Conference on Neural Information Processing Systems, 2018.
[46] H. Salimbeni, V. Dutordoir, J. Hensman, M. Deisenroth, Deep Gaussian processes with importance-weighted variational inference, in: Kamalika Chaudhuri, Ruslan Salakhutdinov (Eds.), Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, USA, in: Proceedings of Machine Learning Research, PMLR, vol. 97, 09–15 Jun 2019, pp. 5589–5598.
[47] H. Yu, Y. Chen, B.K.H. Low, P. Jaillet, Z. Dai, Implicit posterior variational inference for deep Gaussian processes, in: Advances in Neural Information Processing Systems, vol. 32, 2019, pp. 14502–14513.
[48] Jiyang Xie, Zhanyu Ma, Dongliang Chang, Guoqiang Zhang, Jun Guo, Gpca: a probabilistic framework for Gaussian process embedded channel attention, IEEE Trans. Pattern Anal. Mach. Intell. (2021).
[49] Jie Yang, Diego Klabjan, Bayesian active learning for choice models with deep Gaussian processes, IEEE Trans. Intell. Transp. Syst. 22 (2020) 1080–1092.
[50] Shiliang Sun, Wenbo Dong, Qiuyang Liu, Multi-view representation learning with deep Gaussian processes, IEEE Trans. Pattern Anal. Mach. Intell. 43 (2020) 4453–4468.
[51] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: large-scale machine learning on heterogeneous systems, http://tensorflow.org/, 2015, software available from tensorflow.org.
[52] D.P. Kingma, J. Ba, ADAM: a method for stochastic optimization, in: International Conference on Learning Representations, 2015, pp. 1–15.
[53] S. Depeweg, J.M. Hernández-Lobato, F. Doshi-Velez, S. Udluft, Learning and policy search in stochastic dynamical systems with Bayesian neural networks, in: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 2017.
[54] D. Dua, C. Graff, UCI Machine Learning Repository, University of California, Irvine, School of Information and Computer Sciences, 2017, http://archive.ics.uci.edu/ml.

[55] 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.
[56] T. Bui, Efficient Deterministic Approximate Bayesian Inference for Gaussian Process Models, PhD thesis, University of Cambridge, 2017.
[57] R. Henao, O. Winther, Predictive active set selection methods for Gaussian processes, Neurocomputing 80 (2012) 10–18.