Expert Systems With Applications 98 (2018) 84–92

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Deep learning with *t*-exponential Bayesian kitchen sinks

Harris Partaourides, Sotirios Chatzis*

Department of Electrical Engineering, Computer Engineering, and Informatics, Cyprus University of Technology, Limassol, 3036, Cyprus

ARTICLE INFO

Article history:
Received 23 November 2017
Revised 23 December 2017
Accepted 10 January 2018
Available online 12 January 2018

Keywords:
Random kitchen sinks
Student's-*t* distribution
t-divergence
Variational bayes

ABSTRACT

Bayesian learning has been recently considered as an effective means of accounting for uncertainty in trained deep network parameters. This is of crucial importance when dealing with small or sparse training datasets. On the other hand, shallow models that compute weighted sums of their inputs, after passing them through a bank of arbitrary randomized nonlinearities, have been recently shown to enjoy good test error bounds that depend on the number of nonlinearities. Inspired from these advances, in this paper we examine novel deep network architectures, where each layer comprises a bank of arbitrary nonlinearities, linearly combined using multiple alternative sets of weights. We effect model training by means of approximate inference based on a *t*-divergence measure; this generalizes the Kullback–Leibler divergence in the context of the *t*-exponential family of distributions. We adopt the *t*-exponential family since it can more flexibly accommodate real-world data, that entail outliers and distributions with fat tails, compared to conventional Gaussian model assumptions. We extensively evaluate our approach using several challenging benchmarks, and provide comparative results to related state-of-the-art techniques.

1. Introduction

Deep neural networks (DNNs) have experienced a resurgence of interest; this is due to recent breakthroughs in the field that offer unprecedented empirical results in several challenging real-world tasks, such as image and video understanding (He, Zhang, Ren, & Sun, 2016), natural language understanding and generation (Sutskever, Vinyals, & Le, 2014), and game playing (Silver et al., 2016). Most DNN models are trained by means of some variant of the backpropagation (BP) algorithm. However, despite all these successes, BP suffers from the major shortcoming of being able to obtain only point-estimates of the trained networks. This fact results in the trained networks generating predictions that do not account for uncertainty, e.g. due to the limited or sparse nature of the available training data.

A potential solution towards the amelioration of these issues consists in treating some network components under the Bayesian inference rationale, instead of stochastic optimization (Neal, 1995). Indeed, Bayesian inference principles have been recently met with great success in the context of DNN regularization, e.g. (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014). Specifically, in this paper we are interested in inference of the network *feature functions*. In the literature, this is effected by considering them as stochastic latent variables imposed some mathematically convenient Gaussian process prior (Damianou & Lawrence, 2013). On this basis, one proceeds to infer the corresponding posteriors, based on the available training data. To the latter end, and with the goal of combining accuracy with computational efficiency, expectation-propagation (Bui, Hernández-Lobato, Hernández-Lobato, Li, & Turner, 2016), mean-field (Chatzis & Kosmopoulos, 2015; Damianou & Lawrence, 2013), and probabilistic backpropagation (Bui, Hernández-Lobato, Li, Hernández-Lobato, & Turner, 2015) have been used.

One of the main driving forces behind the unparalleled data modeling and predictive performance of modern DNNs is their capability of effectively learning to extract informative, high-level, hierarchical representations of observed data with latent structure (LeCun, Bengio, & Hinton, 2015). Nevertheless, DNNs are not the only class of models that entail this sort of functionality. Indeed, major advances in machine learning have long been dominated by the development of shallow architectures that compute weighted sums of feature functions; the latter generate nonlinear representations of their input data, which can be determined under a multitude of alternative rationales. For instance, models that belong to the family of support vector machines (SVMs) (Vapnik, 1998) essentially compute weighted sums of positive definite kernels; boosting algorithms, such as AdaBoost (Schapire, 2003), compute weighted sums of weak learners, such as decision trees. In all cases, both the feature functions as well as the associated weights are learned under an empirical risk minimization rationale; for in-

* Corresponding author.
E-mail addresses: c.partaourides@cut.ac.cy (H. Partaourides), sotirios.chatzis@cut.ac.cy, soteri0s@me.com (S. Chatzis).

https://doi.org/10.1016/j.eswa.2018.01.013
0957-4174/

stance, the hinge loss is used in the context of SVMs, while AdaBoost utilizes the exponential loss.

In the same vein, the machine learning community has recently examined a bold, yet quite promising possibility: postulating weighted sums of *random kitchen sinks* (RKS) (Rahimi & Recht, 2009). The main rationale of this family of approaches essentially consists in randomly drawing the employed feature functions (nonlinearities), and limiting model training to the associated (scalar) weights. Specifically, the (entailed parameters of the) postulated nonlinearities are randomly sampled from an appropriate density, which is a priori determined by the practitioners according to some assumptions (Le, Sarlós, & Smola, 2013; Yang, Smola, Song, & Wilson, 2015). As it has been shown, both through theoretical analysis as well as some empirical evidence, such a modeling approach does not yield much inferior performance for a trained classifier compared to the mainstream approach of optimally selecting the employed nonlinearities. In addition, predictive performance is shown to increase with the size of the employed bank of randomly drawn nonlinearities.

Inspired from these advances, this paper introduces a fresh regard towards DNNs: We formulate each DNN layer as a bank of random nonlinearities, which are linearly combined in multiple alternative fashions. This way, the postulated models eventually yield a hierarchical cascade of informative representations of their multivariate observation inputs, that can be used to effectively drive a penultimate regression or classification layer. At each layer, the employed bank of random nonlinearities is sampled from an appropriate postulated density, in a vein similar to RKS. However, in order to alleviate the burden of having to manually design these densities, we elect to infer them in a Bayesian sense. In addition, we elect not to obtain point-estimates of the weights used for combining the drawn nonlinearities. On the contrary, we perform Bayesian inference over them; this way, we allow for strong model regularization, by better accounting for model uncertainty, especially when training data availability is limited or sparse.

As already discussed, Bayesian inference for DNN-type models can be performed under various alternative paradigms. Here, we resort to variational inference ideas, which consist in searching for a proxy in an analytically solvable distribution family that approximates the true underlying posterior distribution. To measure the closeness between the true and the approximate posterior, the relative entropy between these two distributions is used. Specifically, under the typical Gaussian assumption, one can use the Shannon–Boltzmann–Gibbs (SBG) entropy, whereby the relative entropy yields the well known Kullback–Leibler (KL) divergence (Wainwright & Jordan, 2008). However, real-world phenomena tend to entail densities with heavier tails than the simplistic Gaussian assumption.

To account for these facts, in this work we exploit the t-exponential family[1], which was first proposed by Tsallis and coworkers (Sousa & Tsallis, 1994; Tsallis, 1998; Tsallis, Mendes, & Plastino, 1998), and constitutes a special case of the more general ϕ-exponential family (Naudts, 2002; 2004a; 2004b). Of specific practical interest to us is the Students'-t density; this is a bell-shaped distribution with heavier tails and one more parameter (degrees of freedom–DOF) compared to the normal distribution, and tends to a normal distribution for large DOF values (McLachlan & Peel, 2000). This way, it offers a solution to the problem of providing protection against outliers in multidimensional variables (Archambeau & Verleysen, 2007; McLachlan & Peel, 2000; Svensén & Bishop, 2005a), which is a very difficult problem that increases in difficulty with the dimension of the variables (Kosinski, 1999). On top of these merits, the t-exponential family also gives rise to a new t-divergence measure; this can be used for performing approximate inference in a fashion that better accommodates heavy-tailed densities (compared to standard KL-based solutions) (Ding, Vishwanathan, & Qi, 2011).

To summarize, we formulate a hierarchical (multilayer) model, each layer of which comprises a bank of random feature functions (nonlinearities). Each nonlinearity is presented with the layer's input, and generates scalar outputs. These outputs are linearly combined by using multiple alternative sets of mixing weights, to produce the (multivariate) layer's output. At each layer, the postulated nonlinearities are drawn from an appropriate (posterior) density, which is inferred from the data (as opposed to the requirement of conventional RKS that the practitioners manually specify this distribution). Indeed, both the posterior density of the nonlinearities, as well as the posterior over the mixing weights, are inferred in an approximate Bayesian fashion. This renders our model more capable of accommodating limited or sparse training data, while retaining its computational scalability. In addition, in order to allow for our model to account for heavy tails, we postulate that the sought densities belong to the t-exponential family, specifically they constitute (multivariate) Student's-t densities. On this basis, we conduct approximate inference by optimizing a t-divergence functional, that better leverages the advantages of the t-exponential family. We dub our proposed approach the Deep t-Exponential Bayesian Kitchen Sinks (DtBKS) model.

The remainder of this paper is organized as follows: In the following Section, we provide a brief overview of the methodological background of our approach. Subsequently, we introduce our approach, and derive its training and inference algorithms. Then, we perform the experimental evaluation of our approach, and illustrate its merits over the current state-of-the-art. Finally, in the concluding Section of this paper, we summarize our contribution and discuss our results.

2. Methodological background

2.1. Weighted sums of random kitchen sinks

Consider the problem of fitting a function $f : \mathbb{R}^\delta \to \mathcal{Y}$ to a training dataset comprising N input-output pairs $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$, drawn i.i.d. from some unknown distribution $P(\boldsymbol{x}, y)$, with $\boldsymbol{x}_n \in \mathbb{R}^\delta$ and $y_n \in \mathcal{Y}$. In essence, this fitting problem consists in finding a function f that minimizes the empirical risk on the training data

$$R[f] = \frac{1}{N}\sum_{n=1}^{N} c(f(\boldsymbol{x}_n), y_n) \tag{1}$$

where the cost function $c(f(\boldsymbol{x}), y)$ defines the penalty we impose on the deviation between the prediction $f(\boldsymbol{x})$ and the actual value y.

The main underlying idea of data modeling under the weighted sums of RKS rationale consists in postulating functions of the form

$$f(\boldsymbol{x}) = \sum_{s=1}^{S} \alpha_s \xi(\boldsymbol{x}; \boldsymbol{\omega}_s) \tag{2}$$

where the $\{\alpha_s\}_{s=1}^S$ are mixing weights, while the nonlinear feature functions ξ are parameterized by some vector $\boldsymbol{\omega} \in \Omega$, and are bounded s.t.: $|\xi(\boldsymbol{x}; \boldsymbol{\omega}_s)| \leq 1$. Specifically, the vectors $\boldsymbol{\omega}_s$ are samples drawn from an appropriate probability distribution $p(\boldsymbol{\omega})$ with support in Ω, whence we have $\xi : \mathbb{R}^\delta \times \Omega \to [-1, 1]$.

As an outcome of this construction, the fitted function $f(\boldsymbol{x})$ can be essentially viewed as *a weighted sum of a bank of random nonlinearities,* ξ, drawn by appropriately sampling from a selected probability distribution $p(\boldsymbol{\omega})$. On this basis, the fitting procedure reduces to*selecting the weight values*$\{\alpha_s\}_{s=1}^S$, such that we

[1] Also referred to as the q-exponential family or the Tsallis distribution.

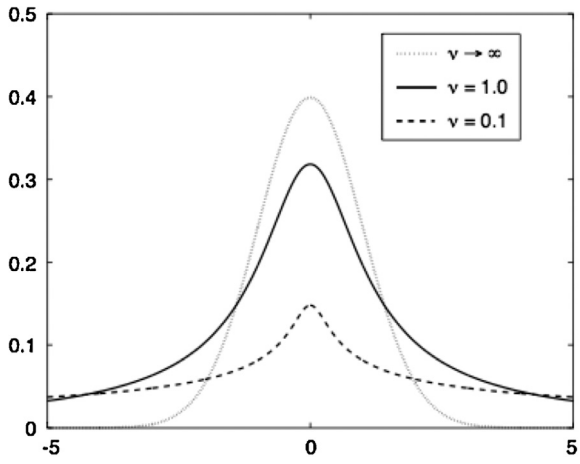

Fig. 1. Univariate student's-t distribution $t(\boldsymbol{y}_t; \boldsymbol{\mu}, \boldsymbol{\Sigma}, \nu)$, with $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ fixed, for various values of ν (Svensén & Bishop, 2005b).

minimize the empirical risk (1); typically, a quadratic loss function is employed to this end. As shown in Rahimi and Recht (2009), for $S \to \infty$, weighted sums of RKS yield predictive models whose true risk is near the lowest true risk attainable by an infinite-dimensional class of functions with optimally selected parameter sets, $\boldsymbol{\omega}$.

Weighted sums of RKS give rise to a modeling paradigm with quite appealing properties: It allows for seamlessly and efficiently employing arbitrarily complex feature functions ξ, since model fitting is limited to the mixing weights; this way, we can easily experiment with feature functions that do not admit simple fitting procedures. On the other hand, RKS require one to (manually) design appropriate distributions $p(\boldsymbol{\omega})$ to draw the employed nonlinearities from; in real-world data modeling tasks, this might prove quite challenging a task.

2.2. The student's-t distribution

The adoption of the multivariate student's-t distribution provides a way to broaden the Gaussian distribution for potential outliers. The probability density function (pdf) of a student's-t distribution with mean vector $\boldsymbol{\mu}$, covariance matrix $\boldsymbol{\Sigma}$, and $\nu > 0$ degrees of freedom is (Liu & Rubin, 1995)

$$t(\boldsymbol{y}_t; \boldsymbol{\mu}, \boldsymbol{\Sigma}, \nu) = \frac{\Gamma\left(\frac{\nu+\delta}{2}\right)|\boldsymbol{\Sigma}|^{-1/2}(\pi\nu)^{-\delta/2}}{\Gamma(\nu/2)\{1 + d(\boldsymbol{y}_t, \boldsymbol{\mu}; \boldsymbol{\Sigma})/\nu\}^{(\nu+\delta)/2}} \tag{3}$$

where δ is the dimensionality of the observations $\boldsymbol{y}_t$, $d(\boldsymbol{y}_t, \boldsymbol{\mu}; \boldsymbol{\Sigma})$ is the squared Mahalanobis distance between $\boldsymbol{y}_t$, $\boldsymbol{\mu}$ with covariance matrix $\boldsymbol{\Sigma}$

$$d(\boldsymbol{y}_t, \boldsymbol{\mu}; \boldsymbol{\Sigma}) = (\boldsymbol{y}_t - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{y}_t - \boldsymbol{\mu}) \tag{4}$$

and $\Gamma(s)$ is the Gamma function, $\Gamma(s) = \int_0^\infty e^{-t} z^{s-1} dz$.

A graphical illustration of the univariate student's-t distribution, with $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ fixed, and for various values of the degrees of freedom ν, is provided in Fig. 1. As we observe, as $\nu \to \infty$, the student's-t distribution tends to a Gaussian with the same $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. On the contrary, as ν tends to zero, the tails of the distribution become longer, thus allowing for a better handling of potential outliers, without affecting the mean or the covariance of the distribution (Chatzis, Kosmopoulos, & Varvarigou, 2009; Chatzis & Kosmopoulos, 2011).

2.3. The t-divergence

The t-divergence was introduced in Ding et al. (2011) as follows:

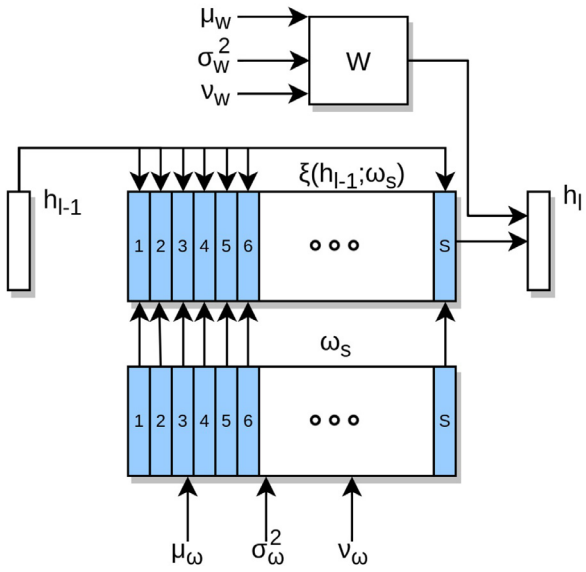

Fig. 2. Graphical illustration of the configuration of one DtBKS model layer.

Definition 1. The t-divergence between two distributions, $q(\boldsymbol{h})$ and $p(\boldsymbol{h})$, is defined as

$$D_t(q||p) = \int \tilde{q}(\boldsymbol{h}) \log_t q(\boldsymbol{h}) d\boldsymbol{h} - \tilde{q}(\boldsymbol{h}) \log_t p(\boldsymbol{h}) d\boldsymbol{h} \tag{5}$$

where $\tilde{q}(\boldsymbol{h})$ is called the escort distribution of $q(\boldsymbol{h})$, and is given by

$$\tilde{q}(\boldsymbol{h}) = \frac{q(\boldsymbol{h})^t}{\int q(\boldsymbol{h})^t d\boldsymbol{h}}, \ t \in \mathbb{R} \tag{6}$$

Importantly, the divergence $D_t(q||p)$ preserves the following two properties:

- $D_t(q||p) \geq 0, \forall q, p$. The equality holds only for $q = p$.
- $D_t(q||p) \neq D_t(p||q)$.

As discussed in Ding et al. (2011), by leveraging the above definition of the t-divergence, $D_t(q||p)$, one can establish an advanced approximate inference framework, much more appropriate for fitting heavy-tailed densities. We exploit these benefits in developing the training algorithm of the proposed DtBKS model, as we shall explain in the following Section.

3. Proposed approach

3.1. Model formulation

Let us consider a DtBKS model with input variables $\boldsymbol{x} \in \mathbb{R}^\delta$ and output (predictable) variables $\boldsymbol{y}$, comprising L layers. In Fig. 2, we provide a graphical illustration of the proposed configuration of one DtBKS model layer. Each layer, l, is presented with an input vector $\boldsymbol{h}^{l-1}$, generated from the preceding layer; at the first layer, this vector is the model input, $\boldsymbol{h}^0 \triangleq \boldsymbol{x}$. This is fed into a bank comprising S randomly drawn feature functions, $\{\xi_s^l(\boldsymbol{h}^{l-1})\}_{s=1}^S$. Specifically, these functions are nonlinearities parameterized by some random vector $\boldsymbol{\omega}$; i.e., $\xi_s^l(\boldsymbol{h}^{l-1}) = \xi(\boldsymbol{h}^{l-1}; \boldsymbol{\omega}_s^l)$. The used samples $\boldsymbol{\omega}_s^l$ are drawn from an appropriate density, which is inferred in a Bayesian sense, as we shall explain next.

This way, we eventually obtain a set of S univariate signals, which we linearly combine to obtain the layer's output. Specifically, we elect to (linearly) combine them in multiple alternative ways, with the goal of obtaining a potent, multidimensional, high-level representation of the original observations. These alternative combinations are computed via a mixing weight matrix,

$\boldsymbol{W}^l \in \mathbb{R}^{\eta \times S}$, where η is the desired dimensionality of the layer's output (i.e., the postulated number of alternative linear combinations).

Eventually, the output vectors at the layers $l \in \{1, \ldots, L-1\}$ yield

$$\boldsymbol{h}^l = \boldsymbol{W}^l [\xi(\boldsymbol{h}^{l-1}; \boldsymbol{\omega}_s^l)]_{s=1}^S \in \mathbb{R}^\eta \tag{7}$$

where $[\chi_s]_{s=1}^S$ denotes the vector-concatenation of the set $\{\chi_s\}_{s=1}^S$. Turning to the penultimate layer of the model, we consider that the output variables are imposed an appropriate conditional likelihood function, the form of which depends on the type of the addressed task. Specifically, in the case of regression tasks, we postulate a multivariate Gaussian of the form

$$p(\boldsymbol{y}|\boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{y} \middle| \boldsymbol{W}^L [\xi(\boldsymbol{h}^{L-1}; \boldsymbol{\omega}_s^L)]_{s=1}^S, \sigma_y^2 \boldsymbol{I}\right) \tag{8}$$

where σ_y^2 is the noise variance. On the other hand, in case of classification tasks, we assume

$$p(\boldsymbol{y}|\boldsymbol{x}) = \text{Softmax}\left(\boldsymbol{y} \middle| \boldsymbol{W}^L [\xi(\boldsymbol{h}^{L-1}; \boldsymbol{\omega}_s^L)]_{s=1}^S\right) \tag{9}$$

This concludes the definition of our model. For brevity, we shall omit the layer indices, l, in the remainder of this paper, wherever applicable. As we observe from Eqs. (7)–(9), at each layer of the proposed model:

(i) We postulate a random feature function $\xi(\cdot)$ that generates scalar outputs. This is effected by taking the inner product of the layer's input with a parameter vector $\boldsymbol{\omega}$.
(ii) We draw many different samples of $\xi(\cdot)$. This is effected by drawing multiple samples from the parameter vector $\boldsymbol{\omega}$, which is essentially considered as a random variable.
(iii) We optimally combine the drawn samples of $\xi(\cdot)$ in a linear fashion, by using the trainable matrix $\boldsymbol{W}$. That is, we compute multiple alternative weighted averages of the drawn samples of $\xi(\cdot)$, whereby the used sets of weights (stored in $\boldsymbol{W}$) are trainable parameters.

Thus, our model constitutes a generalization of random kitchen sinks, whereby the main rationale consists in drawing multiple random samples of a fundamental unit. A key difference between the existing literature, e.g. Yang et al. (2015), Le et al. (2013) and Rahimi and Recht (2009), and our approach consists in the fact that we opt for inferring the distribution we draw samples from, as opposed to using some fixed selection. In addition, we learn how to optimally combine these samples, and do it in multiple alternative ways, so as to generate a multidimensional layer output. Each dimension of the generated output corresponds to a different way of combining the drawn samples of the basic unit $\xi(\cdot)$. This is in contrast to the existing literature on random kitchen sinks, where only shallow architectures with scalar outputs have been considered (Le et al., 2013; Rahimi & Recht, 2009; Yang et al., 2015), whereby the drawn samples are combined in one single way.

3.2. Model training

To perform model training, we opt for a full Bayesian treatment. This is in contrast to most deep learning approaches, which rely on frequentist treatments that yield point-estimates of the model parameters. Indeed, frequentist methods consider model parameters to be fixed, and the training data to be some random sample from an infinite number of existing but unobserved data points. On the contrary, our Bayesian treatment is designed under the assumption of dealing with fixed scarce data and parameters that are random because they are unknowns. This way, by imposing a suitable prior over model parameters and obtaining a corresponding posterior distribution based on the given observed data, our full Bayesian treatment allows for better including uncertainty in learning the model; this is important when dealing with limited or sparse data.

To allow for inferring the distribution that the employed feature functions, ξ, must be drawn from, we first consider that the vectors $\boldsymbol{\omega}$ that parameterize them are student's-t distributed latent variables. We employ the same assumption for the weight matrices, $\boldsymbol{W}$, which we also want to infer in a Bayesian fashion, so as to account for model uncertainty. Specifically, we start by imposing a simple, zero-mean student's-t prior distribution over them, with tied degrees of freedom, at each model layer:

$$p(\boldsymbol{\omega}) = t(\boldsymbol{\omega}|\boldsymbol{0}, \boldsymbol{I}, \nu) \tag{10}$$

$$p(\boldsymbol{W}) = t(\text{vec}(\boldsymbol{W})|\boldsymbol{0}, \boldsymbol{I}, \nu) \tag{11}$$

where $\text{vec}(\cdot)$ is the matrix vectorization operation, and $\nu > 0$ is the degrees of freedom hyperparameter of the imposed priors. On this basis, we seek to devise an efficient means of inferring the corresponding posterior distributions, given the available training data. To this end, we postulate that the sought posteriors approximately take the form of student's-t densities with means, *diagonal* covariance matrices, and degrees of freedom inferred from the data. Hence, we have:

$$q(\boldsymbol{\omega}; \boldsymbol{\phi}) = t(\boldsymbol{\omega}|\boldsymbol{\mu}_{\boldsymbol{\omega}}, \text{diag}(\boldsymbol{\sigma}_{\boldsymbol{\omega}}^2), \nu_{\boldsymbol{\omega}}) \tag{12}$$

$$q(\boldsymbol{W}; \boldsymbol{\phi}) = t(\text{vec}(\boldsymbol{W})|\boldsymbol{\mu}_{\boldsymbol{W}}, \text{diag}(\boldsymbol{\sigma}_{\boldsymbol{W}}^2), \nu_{\boldsymbol{W}}) \tag{13}$$

where $\boldsymbol{\phi} = \{\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i^2, \nu_i\}_{i \in \{\boldsymbol{\omega}, \boldsymbol{W}\}}$, and $\nu_i > 0$, $\forall i$, for all model layers. This selection for the form of the sought parameter posteriors allows for our model to accommodate heavy-tailed underlying densities much better than the usual Gaussian assumption, as we have also discussed in Section 2.2.

Then, to allow for inferring the sought posteriors in a computationally efficient manner, we resort to variational Bayes (Attias, 2000). This consists in derivation of a family of variational posterior distributions $q(.)$, which approximate the true posterior distributions that we need to infer. In essence, this is effected by optimizing an appropriate functional over the variational posterior, which measures model fit to the training data. Hence, variational Bayes casts inference as an approximate optimization problem, yielding a trade-off between accuracy and computational scalability (Attias, 2000).

Specifically, to allow for reaping the most out of the heavy-tailed assumptions of our model, we elect to perform variational Bayes by minimizing the t-divergence between the sought approximate posterior and the postulated joint density over the observed data and the model latent variables (Ding et al., 2011). Thus, the proposed model training objective becomes

$$\begin{aligned} &q(\boldsymbol{\omega}; \boldsymbol{\phi}), q(\boldsymbol{W}; \boldsymbol{\phi}) = \\ &\quad = \arg\min_{q(\cdot)} D_t\left(q(\boldsymbol{\omega}; \boldsymbol{\phi}), q(\boldsymbol{W}; \boldsymbol{\phi}) || p(\boldsymbol{y}; \boldsymbol{\omega}, \boldsymbol{W})\right) \end{aligned} \tag{14}$$

By application of simple algebra, the expression of the t-divergence in (14) yields

$$\begin{aligned} &D_t\left(q(\boldsymbol{\omega}; \boldsymbol{\phi}), q(\boldsymbol{W}; \boldsymbol{\phi}) || p(\boldsymbol{y}; \boldsymbol{\omega}, \boldsymbol{W})\right) \\ &= D_t\left(q(\boldsymbol{\omega}; \boldsymbol{\phi}) || p(\boldsymbol{\omega})\right) + D_t\left(q(\boldsymbol{W}; \boldsymbol{\phi}) || p(\boldsymbol{W})\right) \\ &\qquad - \mathbb{E}_{\tilde{q}(\boldsymbol{\omega}, \boldsymbol{W}; \boldsymbol{\phi})}[\log p(\boldsymbol{y}|\boldsymbol{x})] \end{aligned} \tag{15}$$

where $\tilde{q}(\boldsymbol{\omega}, \boldsymbol{W}; \boldsymbol{\phi})$ is the escort distribution of the sought posterior, and the t-divergence terms pertaining to the parameters $\boldsymbol{\omega}$ and $\boldsymbol{W}$ are summed over all model layers. Then, following Ding et al. (2011), and based on (10)–(13), we obtain that the t-divergence expressions in (15) can be written in the following

form:

$$D_t\big(q(\boldsymbol{\theta};\boldsymbol{\phi})||p(\boldsymbol{\theta})\big) = \sum_i \left\{ \frac{\Psi_{qi}}{1-t}\left(1+\frac{1}{\nu_{\boldsymbol{\theta}}}\right) - \frac{\Psi_p}{1-t}\left(1+\frac{[\boldsymbol{\sigma}_{\boldsymbol{\theta}}^2]_i + [\boldsymbol{\mu}_{\boldsymbol{\theta}}]_i^2}{\nu}\right)\right\} \tag{16}$$

where $\boldsymbol{\theta} \in \{\boldsymbol{\omega}, \text{vec}(\boldsymbol{W})\}$, $[\boldsymbol{\zeta}]_i$ is the ith element of a vector $\boldsymbol{\zeta}$, we denote

$$\Psi_{qi} \triangleq \left(\frac{\Gamma\left(\frac{\nu_{\boldsymbol{\theta}}+1}{2}\right)}{\Gamma\left(\frac{\nu_{\boldsymbol{\theta}}}{2}\right)(\pi\nu_{\boldsymbol{\theta}})^{1/2}[\boldsymbol{\sigma}_{\boldsymbol{\theta}}]_i}\right)^{-\frac{2}{\nu_{\boldsymbol{\theta}}+1}} \tag{17}$$

$$\Psi_p \triangleq \left(\frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)(\pi\nu)^{1/2}}\right)^{-\frac{2}{\nu+1}} \tag{18}$$

and the free hyperparameter t is set as suggested in Ding et al. (2011), yielding:

$$t = \frac{2}{1+\nu_{\boldsymbol{\theta}}} + 1 \tag{19}$$

As we observe from the preceding discussion, the expectation of the conditional log-likelihood of our model, $\mathbb{E}_{\tilde{q}(\boldsymbol{\omega},\boldsymbol{W};\boldsymbol{\phi})}[\log p(\boldsymbol{y}|\boldsymbol{x})]$, is computed with respect to the escort distributions of the sought posteriors. Therefore, *at training time,* the banks of the employed feature functions (i.e., the samples of their parameters, $\{\boldsymbol{\omega}_s\}_{s=1}^S$), must be drawn *from the escort distributions of the derived posteriors.*

Turning to the employed mixing weight matrices, $\boldsymbol{W}$, our consideration of them being latent variables with an inferred posterior perplexes computation of the expressions (7)–(9); indeed, it requires that we compute appropriate posterior expectations of these expressions w.r.t. $\boldsymbol{W}$. To circumvent this problem, we draw multiple samples of the mixing weight matrices, $\{\boldsymbol{W}_s\}_{s=1}^S$, *in an MC fashion,* and *average over the corresponding outcomes to compute the model output. At training time,* Eq. (15) implies that these samples must also be drawn *from the escort distributions of the derived posteriors.*

Based on the previous results, and following Ding et al. (2011), we can easily obtain the expressions of these escort distributions of the derived posteriors, that we need to sample from at training time. Specifically, it is easy to show that these escort distributions yield a factorized form, that reads:

$$\tilde{q}(\boldsymbol{\theta};\boldsymbol{\phi}) = t\left(\boldsymbol{\theta}|\boldsymbol{\mu}_{\boldsymbol{\theta}}, \frac{\nu_{\boldsymbol{\theta}}}{\nu_{\boldsymbol{\theta}}+2}\text{diag}(\boldsymbol{\sigma}_{\boldsymbol{\theta}}^2), \nu_{\boldsymbol{\theta}}+2\right) \quad \forall\boldsymbol{\theta} \in \{\boldsymbol{\omega}, \text{vec}(\boldsymbol{W})\} \tag{20}$$

Notably, our inference algorithm yields an MC estimator of the proposed DtBKS model. Unfortunately, MC estimators are notorious for their vulnerability to unacceptably high variance. In this work, we resolve these issues by adopting the reparameterization trick of Kingma and Welling (2014), adapted to the t-exponential family. This trick consists in a smart reparameterization of the MC samples, $\{\boldsymbol{\theta}_s\}_{s=1}^S$, drawn from a distribution $\boldsymbol{\theta} \sim q(\boldsymbol{\theta};\boldsymbol{\phi})$; this is obtained via a differentiable transformation $\boldsymbol{g}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon})$ of an (auxiliary) random variable $\boldsymbol{\epsilon}$ with low variance:

$$\boldsymbol{\theta} = \boldsymbol{g}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}) \quad \text{with} \quad \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}) \tag{21}$$

In our case, the smart reparameterization of the MC samples drawn from the student's-t escort densities (20) yields the expression:

$$\boldsymbol{\theta}_s = \boldsymbol{\theta}(\boldsymbol{\epsilon}_s) = \boldsymbol{\mu}_{\boldsymbol{\theta}} + \left(\frac{\nu_{\boldsymbol{\theta}}}{\nu_{\boldsymbol{\theta}}+2}\right)^{1/2} \boldsymbol{\sigma}_{\boldsymbol{\theta}}\boldsymbol{\epsilon}_s \tag{22}$$

where $\boldsymbol{\epsilon}_s$ is random student's-t noise with unitary variance:

$$\boldsymbol{\epsilon}_s \sim t(\mathbf{0}, \boldsymbol{I}, \nu_{\boldsymbol{\theta}}+2) \tag{23}$$

On this basis,*at training time*, we replace the samples of both the weight matrices, $\boldsymbol{W}$, as well as the feature function parameters, $\boldsymbol{\omega}$, with the expression (22), where sampling is performed w.r.t. the low-variance random variable $\boldsymbol{\epsilon}$. Then, the resulting (reparameterized) t-divergence objective (15) can be minimized by means of any off-the-shelf stochastic optimization algorithm, yielding low variance estimators. To this end, in this work we utilize AdaM (Kingma & Ba, 2015). We initialize the sought posterior hyperparameters by setting them equal to the hyperparameters of the imposed priors.

3.3. Inference algorithm

Having obtained a training algorithm for our proposed DtBKS model, we can now proceed to elaborate on how inference is performed using our method. To this end, it is needed that we compute the *posterior expectation* of the model's output, as usual when performing Bayesian inference. Thus, *at inference time*, we need to *draw samples from the derived posteriors*, (12) and (13), in an MC fashion. This entails drawing from the posteriors, at each layer, of a set comprising S samples of: (i) the vectors $\boldsymbol{\omega}$ that parameterize the employed feature functions; and (ii) the mixing weight matrices, $\boldsymbol{W}$, used to combine the outputs of the drawn banks of feature functions. Note that this is in contrast to the training algorithm of DtBKS, where the use of the t-divergence objective (15) gives rise to the requirement of drawing from the associated *escort distributions* (20), while the need of training reliable estimators requires utilization of the reparameterization trick.

4. Experimental evaluation

In this section, we perform a thorough experimental evaluation of our proposed DtBKS model. We provide a quantitative assessment of the efficacy, the effectiveness, and the computational efficiency of our approach, combined with deep qualitative insights into few of its key performance characteristics. To this end, we consider several benchmarks from the UCI machine learning repository (UCI-Rep) (Asuncion & Newman, 2007) that pertain to both regression and classification tasks, as well as the well-known InfiMNIST classification benchmark (Loosli, Canu, & Bottou, 2007). The considered datasets, as well as their main characteristics (i.e., their number of training examples, N, and input dimensionality, δ) are summarized in Table 1.

With the exception of the ISOLET dataset from UCI-Rep, as well as InfiMNIST, the rest of the considered benchmarks do not provide a split into training and test sets. In these cases, we account for this lack by running our experiments 20 times, with different random data splits into training and test sets, and compute performance means and standard deviations; we use a randomly selected 90% of the data for model training, and the rest for evaluation purposes. In the case of regression tasks, we use the root mean square error (RMSE) as our performance metric; we employ the misclassification rate for the considered classification benchmarks.

To obtain some comparative results, we also evaluate an existing alternative approach for Bayesian inference of deep network nonlinearities, namely deep Gaussian processes (DGPs) (Damianou & Lawrence, 2013). Indeed, the DGP is the existing type of Bayesian deep learning approaches that is most closely related to our approach; this is the case, since DGP does also allow for inferring the employed nonlinearities, but under a completely different rationale. Specifically, we evaluate the most recent variant of DGPs, presented in Cutajar, Bonilla, Michiardi, and Filippone (2017), which allows for the model to efficiently scale to large datasets. Finally, we also provide comparisons to a state-of-the-art Deep Learning approach, namely a Dropout network (Srivastava et al., 2014), as

Table 1
Characteristics of the considered datasets.

Dataset	N	δ	Task	Performance metric
Boston housing	506	13	Regression	RMSE
Concrete	1030	8	Regression	RMSE
Energy	768	8	Regression	RMSE
Power plant	9568	4	Regression	RMSE
Protein	45,730	9	Regression	RMSE
Wine (White)	4898	11	Regression	RMSE
Wine (Red)	1588	11	Regression	RMSE
Breast cancer diagnostic (wdbc)	569	30	Classification	Misclassification rate
ISOLET	7797	617	Classification	Misclassification rate
Gas sensor	13,910	128	Classification	Misclassification rate
Parkinson's (Oxford Parkinson's disease detection dataset)	197	22	Classification	Misclassification rate
Spam	4601	56	Classification	Misclassification rate
LSVT voice rehabilitation	126	310	Classification	Misclassification rate
InfiMNIST	8+ Million	784	Classification	Misclassification rate

well as a baseline approach, namely SVMs using both linear and RBF kernels.

In all cases, our specification of the priors imposed on D*t*BKS considers a low value for the degrees of freedom hyperparameter, $\nu = 2.1$. In the case of regression tasks, we employ a noise variance equal to $\sigma_y^2 = \exp(-2)$. Turning to the selection of the form of the drawn feature functions, ξ, we consider a simple *trigonometric* formulation, which is inspired from the theory of random Fourier projections of RBF kernels (Rudin, 2011). Specifically, we postulate $\xi(\boldsymbol{x}; \boldsymbol{\omega}) = \frac{1}{2}\cos(\boldsymbol{\omega}^T\boldsymbol{x}) + \frac{1}{2}\sin(\boldsymbol{\omega}^T\boldsymbol{x})$. Note that under this selection for the form of $\xi(\boldsymbol{x}; \boldsymbol{\omega})$, and setting $\nu_{\boldsymbol{\omega}} \to \infty$ and $\nu_{\boldsymbol{W}} \to \infty$, D*t*BKS reduces to the DGP variant of (Cutajar et al., 2017) with a postulated RBF kernel. For computational efficiency, we limit the number of drawn samples to 100, during both D*t*BKS training and inference on the test data. AdaM is run with the default hyperparameter values.

DGP is evaluated considering multiple selections of the number of Gaussian processes per layer, as well as the number of layers, using RBF kernels and arc-cosine kernels (Cutajar et al., 2017); in each experimental case, we report results pertaining to the best-performing DGP configuration. Similar is the case for Dropout networks, which are evaluated considering multiple alternatives for the number of layers and the output size of each hidden layer; we employ ReLU nonlinearities (Nair & Hinton, 2010).

Our deep learning source code has been developed in Python, using the Tensorflow library (Abadi et al., 2015); it can be found on https://github.com/Partaourides/DtBKS.We have also made use of a DGP implementation provided by M. Filippone[2]. The evaluation of SVM-type algorithms was performed by utilizing Python's scikit-learn toolbox (Pedregosa et al., 2011). We run our experiments on an Intel Xeon 2.5GHz server with 64GB RAM and an NVIDIA Tesla K40 GPU.

4.1. Comparative results

We begin our exposition by providing the best empirical performance of our method, and showing how it compares to the alternatives. These outcomes have been obtained by experimenting with different selections for the number of layers, L, and the output size of each hidden layer, η (i.e., for $l \in \{1, \ldots, L-1\}$). Our results are outlined in Table 2; in all cases, we provide therein (in parentheses) the model configurations that obtained the reported (best empirical) performance[3]

We observe that our approach outperforms DGP in the considered regression benchmarks; in all cases, these empirical performance differences are found to be statistically significant, by running the paired student's-*t* test. On the other hand, D*t*BKS outperforms DGP in only three out of the seven considered classification benchmarks, with statistically significant differences (according to the paired student's-*t* test), while yielding comparable outcomes in the rest. In addition, D*t*BKS outperforms Dropout in all the considered classification benchmarks; the paired student's-*t* test shows that these empirical performance differences are statistically significant. The only exception to this finding is InfiMNIST, where the results are essentially comparable. On the other hand, D*t*BKS significantly outperforms Dropout in the Protein regression benchmark, while yielding comparable performance in the rest considered regression tasks (according to the outcomes of the paired Student's-*t* test). Finally, both baseline SVM model configurations are completely outperformed by D*t*BKS, in all cases.

4.2. Further investigation

Further, it is interesting to provide a feeling of how D*t*BKS model performance changes with the selection of the number of layers, L, and the dimensionality of each hidden layer, η (i.e., for $l \in \{1, \ldots, L-1\}$). To examine these aspects, in Fig. 3(a)–(c) we plot model performance fluctuation with η, setting the number of layers equal to $L = 2, 3,$ and 4, respectively, for few characteristic experimental cases comprising limited training data. As we observe, D*t*BKS performance is significantly affected by both these selections. Note also that the associated performance fluctuation patterns of D*t*BKS are quite different among the illustrated examples. These findings are congruent with the behavior of all existing state-of-the-art deep learning approaches. It is also important to mention the high standard deviation of the observed performances in some cases where we set $L = 4$; we attribute this unstable behavior to overfitting due to insufficient training data.

4.3. Are t-exponential Bayesian kitchen sinks more potent than random kitchen sinks?

Finally, it is extremely interesting to examine how beneficial it is for D*t*BKS to infer a posterior distribution over the (random variables that parameterize the) employed feature functions, instead of using a simple, manually selected density. To examine this aspect, we repeat our experiments by drawing the vectors $\boldsymbol{\omega}$, that parameterize the feature functions, ξ, from the postulated simple priors, $p(\boldsymbol{\omega})$. Hence, we adopt an RKS-type rationale in drawing the feature functions, ξ, as opposed to utilizing the inferred posteri-

[2] https://github.com/mauriziofilippone/deep_gp_random_features.

[3] This selection was performed by means of leave-one-out cross-validation, considering $L \in \{2, 3, 4, 5\}$ and $\eta \in \{\lceil \delta/4 \rceil, \lceil \delta/2 \rceil, \lceil 3\delta/4 \rceil, \delta\}$.

Table 2
Obtained performance for best model configuration (RMSE for regression tasks, misclassification rate for classification tasks; the lower the better).

Dataset	DtBKS	DGP	Dropout	Linear SVM	RBF-kernel SVM
Boston housing	0.2939 ± 0.04 ($L = 2, \eta = 3$)	0.3897 ± 0.1 ($L = 2, \eta = 3$)	0.2516 ± 0.06 ($L = 2, \eta = 123$)	0.5027 ± 0.1	0.3471 ± 0.1
Concrete	0.3213 ± 0.02 ($L = 2, \eta = 5$)	0.4501 ± 0.03 ($L = 2, \eta = 5$)	0.3228 ± 0.03 ($L = 2, \eta = 163$)	0.64 ± 0.04	0.3957 ± 0.03
Energy	0.1285 ± 0.01 ($L = 2, \eta = 6$)	0.1636 ± 0.02 ($L = 2, \eta = 6$)	0.1322 ± 0.01 ($L = 2, \eta = 175$)	0.3183 ± 0.03	0.25 ± 0.03
Power plant	0.2366 ± 0.01 ($L = 3, \eta = 4$)	0.2401 ± 0.01 ($L = 3, \eta = 4$)	0.2236 ± 0.01 ($L = 3, \eta = 200$)	0.2671 ± 0.01	0.2399 ± 0.01
Protein	0.6113 ± 0.01 ($L = 2, \eta = 9$)	0.6734 ± 0.01 ($L = 2, \eta = 9$)	0.7453 ± 0.01 ($L = 2, \eta = 200$)	0.8673 ± 0.01	0.7648 ± 0.01
Wine (White)	0.7684 ± 0.02 ($L = 2, \eta = 11$)	0.8072 ± 0.02 ($L = 2, \eta = 11$)	0.7609 ± 0.02 ($L = 2, \eta = 200$)	0.8551 ± 0.02	0.7751 ± 0.02
Wine (Red)	0.7564 ± 0.04 ($L = 2, \eta = 5$)	0.7791 ± 0.04 ($L = 2, \eta = 5$)	0.7570 ± 0.05 ($L = 2, \eta = 145$)	0.8064 ± 0.05	0.7693 ± 0.05
Breast cancer diagnostic (wdbc)	0.0116 ± 0.01 ($L = 2, \eta = 21$)	0.0116 ± 0.01 ($L = 2, \eta = 21$)	0.0710 ± 0.06 ($L = 2, \eta = 170$)	0.0304 ± 0.02	0.025 ± 0.02
ISOLET	$0.055 \pm NA$ ($L = 2, \eta = 205$)	$0.0654 \pm NA$ ($L = 2, \eta = 205$)	$0.1256 \pm NA$ ($L = 2, \eta = 133$)	$0.055 \pm NA$	$0.063 \pm NA$
Gas sensor	0.0136 ± 0.002 ($L = 3, \eta = 106$)	0.0094 ± 0.002 ($L = 3, \eta = 106$)	0.0688 ± 0.07 ($L = 3, \eta = 183$)	0.0159 ± 0.001	0.0168 ± 0.003
Parkinson's	0.0658 ± 0.05 ($L = 2, \eta = 15$)	0.0842 ± 0.05 ($L = 2, \eta = 15$)	0.0976 ± 0.09 ($L = 2, \eta = 168$)	0.1342 ± 0.07	0.1079 ± 0.04
Spam	0.0543 ± 0.01 ($L = 2, \eta = 46$)	0.0517 ± 0.01 ($L = 2, \eta = 46$)	0.1629 ± 0.03 ($L = 2, \eta = 182$)	0.0777 ± 0.01	0.0666 ± 0.01
LSVT voice rehabilitation	0.1375 ± 0.07 ($L = 2, \eta = 51$)	0.3250 ± 0.11 ($L = 2, \eta = 51$)	0.3782 ± 0.17 ($L = 2, \eta = 116$)	0.2583 ± 0.09	0.1667 ± 0.08
InfiMNIST	$0.0093 \pm NA$ ($L = 4, \eta = 100$)	$0.0096 \pm NA$ ($L = 4, \eta = 100$)	$0.0096 \pm NA$ ($L = 4, \eta = 113$)	$0.25 \pm NA$	$0.25 \pm NA$

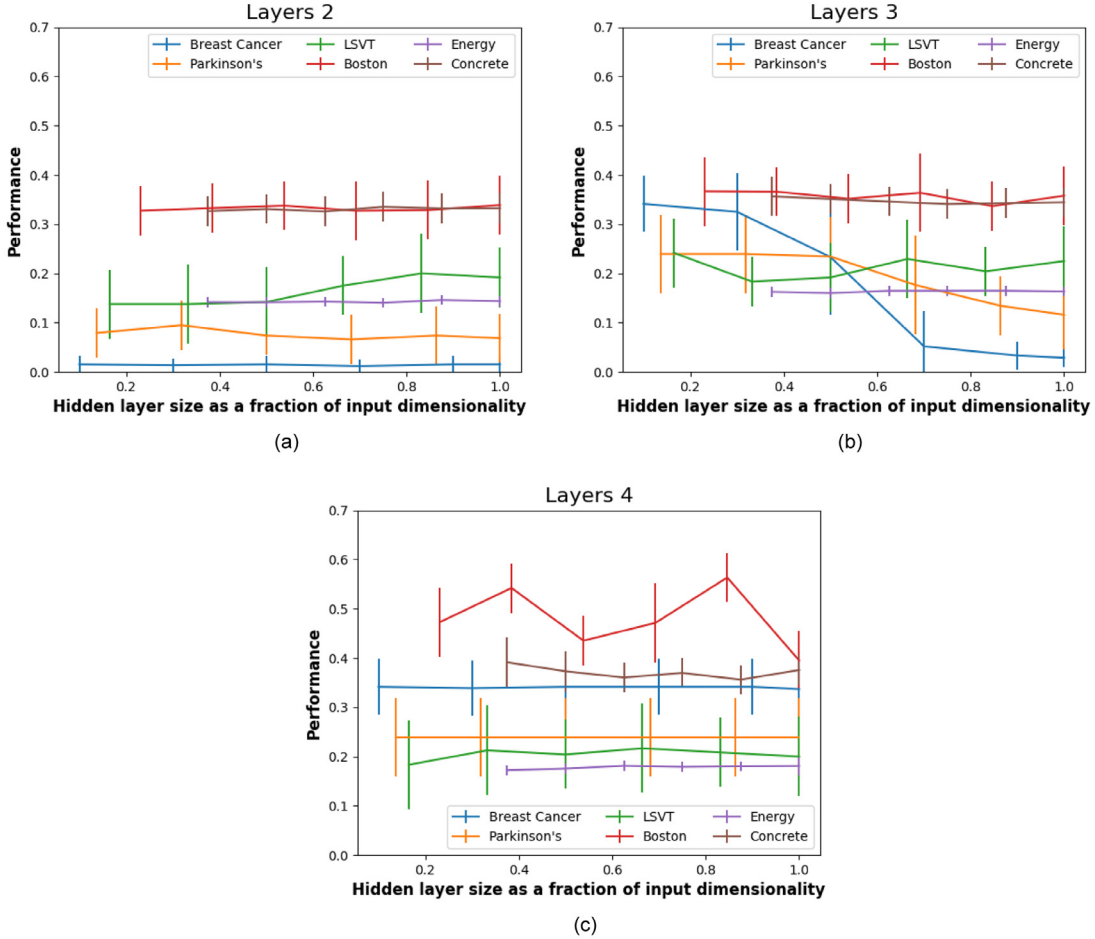

Fig. 3. DtBKS performance fluctuation with the number of layers, L, and the output size of each hidden layer, η (as a fraction of input dimensionality, δ): (a) $L = 2$; (b) $L = 3$; (c) $L = 4$. Performance metrics are the RMSE for regression tasks, and the misclassification rate for classification tasks (the lower the better).

Table 3
D*t*BKS performance when replacing *t*-exponential Bayesian kitchen sinks with random kitchen sinks. Performance metrics are the RMSE for regression tasks, and the misclassification rate for classification tasks (the lower the better).

Dataset	Performance	Comparison to full D*t*BKS model (from Table 2)
Boston housing	0.3199 ± 0.04	0.2939 ± 0.04
Concrete	0.3586 ± 0.04	0.3213 ± 0.02
Energy	0.1494 ± 0.01	0.1285 ± 0.01
Power plant	0.2301 ± 0.01	0.2366 ± 0.01
Protein	0.7110 ± 0.01	0.6113 ± 0.01
Wine (White)	0.7787 ± 0.02	0.7684 ± 0.02
Wine (Red)	0.7720 ± 0.04	0.7564 ± 0.04
Breast cancer diagnostic (wdbc)	0.0188 ± 0.02	0.0116 ± 0.01
ISOLET	$0.2245 \pm NA$	$0.0552 \pm NA$
Gas Sensor	0.0175 ± 0.003	0.0136 ± 0.002
Parkinson's	0.0895 ± 0.06	0.0658 ± 0.05
Spam	0.0748 ± 0.01	0.0543 ± 0.01
LSVT voice rehabilitation	0.1208 ± 0.04	0.1375 ± 0.07
InfiMNIST	$0.0603 \pm NA$	$0.0093 \pm NA$

ors, $q(\boldsymbol{\omega})$ [or their corresponding escort distributions, $\tilde{q}(\boldsymbol{\omega})$, during training].

Our findings are provided in Table 3; these results correspond to selections of the number of layers, L, and the output size, η, similar to the values reported in Table 2. Our empirical evidence is quite conspicuous: (i) merely drawing the postulated nonlinearities from a simple prior, yet inferring a student's-*t* posterior over the mixing weights, $\boldsymbol{W}$, as discussed previously, yields notably competitive performance; (ii) inferring posteriors over the nonlinearities, under the discussed D*t*BKS rationale, gives a statistically significant boost to the obtained modeling performance, except for *Power Plant* and *LSVT,* where we reckon that overfitting is induced (due to insufficient training data availability).

4.4. Computational complexity

Another significant aspect that affects the efficacy of a machine learning technique is its computational complexity. To investigate this aspect, we scrutinize the derived D*t*BKS algorithm, both regarding its asymptotic behavior, as well as in terms of its total computational costs. Our observations can be summarized as follows: For the model configurations yielding the performance statistics of Table 1, D*t*BKS takes on average 4 times longer than Dropout *per algorithm iteration*, probably due to the entailed $\Gamma(\cdot)$ functions in (16), and their derivatives; DGP takes on average 2 times longer than Dropout. On the other hand, D*t*BKS training converges much faster than all the considered competitors. These differences are so immense that, as an outcome, the *total time* required *by all the evaluated methods* is of the same order of magnitude. Indeed, we typically observe that D*t*BKS takes much less time than DGP, and usually not much longer than Dropout; these outcomes are summarized in Table 4. Hence, we deduce that D*t*BKS yields the observed predictive performance improvement without undermining computational efficiency and scalability.

Table 4
Wall-clock times of the evaluated deep learning approaches (in minutes).

Dataset	D*t*BKS	RKS	DGP	Dropout
Boston housing	8	4	11	12
Concrete	20	7	48	14
Energy	23	8	166	22
Power plant	94	43	91	49
Protein	74	48	35	43
Wine (White)	20	14	13	10
Wine (Red)	8	7	10	6
Breast cancer diagnostic (wdbc)	14	5	4	15
ISOLET	195	31	109	20
Gas sensor	97	35	39	48
Parkinson's	9	6	7	20
Spam	42	22	23	13
LSVT voice rehabilitation	11	5	1	3
InfiMNIST	650	1165	489	238

5. Conclusions

In this paper, we introduced a fresh view towards deep learning, which consists in postulating banks of randomly drawn nonlinearities at each model layer. To alleviate the burden of having to manually specify the distribution these nonlinear feature functions are drawn from, we elected to infer them in a Bayesian sense. This also renders our model more robust to scenarios dealing with limited or sparse training data availability.

In this context, we postulated that the sought posteriors constitute multivariate student's-*t* densities. This assumption allows for our model to better cope with heavy-tailed underlying densities; these are quite common in real-world data modeling scenarios, yet they cannot be captured sufficiently enough by the usual Gaussian assumptions. Then, to allow for reaping the most out of the heavy tails of student's-*t* densities, we performed approximate Bayesian inference for our model under a novel objective function construction. This was based on a *t*-divergence functional, which better accommodates heavy-tailed densities.

We exhaustively evaluated our approach using challenging benchmark datasets; we offered thorough insights into its key performance characteristics. This way, we illustrated that our proposed approach outperforms the existing alternatives in terms of predictive accuracy, without undermining the overall computational scalability, both in terms of training time and of prediction generation time. We also showed that data-driven inference of a posterior distribution from which we can draw the employed banks of nonlinearities yields better results than drawing from a simple prior.

One direction for further research concerns postulating nonelliptical latent variable densities, which can account for skewness in a fashion similar, e.g., to Partaourides and Chatzis (2017) and Chatzis (2010). Introduction of a solid means of capturing conditional heteroscedasticity in modeled sequential data, in a fashion similar, e.g., to Platanios and Chatzis (2014), is also a challenge of immense interest. On a different vein, we must emphasize that our approach is not capable of modeling spatial dynamics and dependencies the way, e.g., convolutional networks do. This is similar to related approaches, such as the DGP model and Dropout networks, which are also not designed with such tasks in mind. However, enabling such capabilities in the context of our D*t*BKS framework would be extremely auspicious for the model performance in the

context of real-world applications, dealing with challenging image data. Hence, addressing these challenges and examining the associated opportunities remains to be explored in our future work.

References

Abadi, M. et al. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org http://tensorflow.org/.

Archambeau, C., & Verleysen, M. (2007). Robust Bayesian clustering. *Neural Networks, 20*, 129–138.

Asuncion, A., Newman, D. (2007). UCI machine learning repository. http://www.ics.uci.edu/~mlearn/MLRepository.html.

Attias, H. (2000). A variational Bayesian framework for graphical models. In *Proceedings of NIPS'00*.

Bui, T., Hernández-Lobato, D., Hernández-Lobato, J., Li, Y., & Turner, R. (2016). Deep gaussian processes for regression using approximate expectation propagation. In *Proceedings of ICML*.

Bui, T. D., Hernández-Lobato, J. M., Li, Y., Hernández-Lobato, D., & Turner, R. E. (2015). Training Deep Gaussian Processes using Stochastic Expectation Propagation and Probabilistic Backpropagation. In *Proceedings of NIPS*.

Chatzis, S. (2010). Hidden Markov models with nonelliptically contoured state densities. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 32*(12), 2297–2304.

Chatzis, S., Kosmopoulos, D., & Varvarigou, T. (2009). Robust sequential data modeling using an outlier tolerant hidden Markov model. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 31*(9), 1657–1669.

Chatzis, S. P., & Kosmopoulos, D. (2011). A variational Bayesian methodology for hidden Markov models utilizing Student's-t mixtures. *Pattern Recognition, 44*(2), 295–306.

Chatzis, S. P., & Kosmopoulos, D. (2015). A latent manifold Markovian dynamics Gaussian process. *IEEE Transactions on Neural Networks and Learning Systems, 25*(1), 70–83.

Cutajar, K., Bonilla, E. V., Michiardi, P., & Filippone, M. (2017). Random feature expansions for deep Gaussian processes. In *Proceedings of ICML*.

Damianou, A., & Lawrence, N. (2013). Deep Gaussian processes. In *Proceedings of AISTATS*.

Ding, N., Vishwanathan, S. N., & Qi, Y. (2011). *t*-divergence based approximate inference. In *Proceedings of NIPS*.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of CVPR*.

Kingma, D., & Ba, J. (2015). Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Kingma, D., & Welling, M. (2014). Auto-encoding variational Bayes. In *Proceedings of ICLR'14*.

Kosinski, A. (1999). A procedure for the detection of multivariate outliers. *Computational Statistics and Data Analysis, 29*, 145–161.

Le, Q., Sarlós, T., & Smola, A. (2013). Fastfood — approximating kernel expansions in loglinear time. *Proceeedings of ICML*.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 512*, 436–444.

Liu, C., & Rubin, D. (1995). ML estimation of the *t* distribution using EM and its extensions, ECM and ECME. *Statistica Sinica, 5*(1), 19–39.

Loosli, G., Canu, S., & Bottou, L. (2007). Training invariant support vector machines using selective sampling. In L. Bottou, O. Chapelle, D. DeCoste, & J. Weston (Eds.), *Large scale kernel machines* (pp. 301–320). Cambridge, MA: MIT Press.

McLachlan, G., & Peel, D. (2000). *Finite mixture models*. New York: Wiley Series in Probability and Statistics.

Nair, V., & Hinton, G. (2010). Rectified linear units improve restricted Boltzmann machines. In *Proceedings of ICML*.

Naudts, J. (2002). Deformed exponentials and logarithms in generalized thermostatistics. *Physica A, 316*, 323–334.

Naudts, J. (2004a). Estimators, escort proabilities, and ϕ-exponential families in statistical physics. *Journal of Inequalities in Pure and Applied Mathematics, 5*(4).

Naudts, J. (2004b). Generalized thermostatistics and mean-field theory. *Physica A, 332*, 279–300.

Neal, R. M. (1995). *Bayesian learning for neural networks*. University of Toronto Ph.D. thesis.

Partaourides, H., & Chatzis, S. P. (2017). Asymmetric deep generative models. *Neurocomputing, 241*, 90–96.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830.

Platanios, E. A., & Chatzis, S. P. (2014). Gaussian process-mixture conditional heteroscedasticity. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 36*(5), 888–900.

Rahimi, A., & Recht, B. (2009). Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Proceedings of NIPS*.

Rudin, W. (2011). *Fourier analysis on groups*. John Wiley & Sons.

Schapire, R. E. (2003). The boosting approach to machine learning: An overview. *Nonlinear Estimation and Classification*. Springer.

Silver, D., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature, 529*, 484–489.

Sousa, A., & Tsallis, C. (1994). Student'S *t*- and *r*-distributions: Unified derivation from an entropic variational principle. *Physica A, 236*, 52–57.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research, 15*, 1929–1958.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Proceedings of NIPS*.

Svensén, M., & Bishop, C. M. (2005a). Robust Bayesian mixture modelling. *Neurocomputing, 64*, 235–252.

Svensén, M., & Bishop, C. M. (2005b). Robust Bayesian mixture modelling. *Neurocomputing, 64*, 235–252.

Tsallis, C. (1998). Possible generalization of Boltzmann-Gibbs statistics. *Journal of Statistical Physics, 52*, 479–487.

Tsallis, C., Mendes, R. S., & Plastino, A. R. (1998). The role of constraints within generalized nonextensive statistics.. *Physica A, 261*, 534–554.

Vapnik, V. N. (1998). *Statistical learning theory*. New York: Wiley.

Wainwright, M. J., & Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning, 1*(1–2), 1–305.

Yang, Z., Smola, A. J., Song, L., & Wilson, A. G. (2015). Á la carte — Learning fast kernels. In *Proceedings of AISTATS*.