Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2022) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

26th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2022)

# Variational Bootstrap for Classification

Bogdan Kozyrskiy[a,*], Dimitrios Milios[a], Maurizio Filippone[a]

[a]*EURECOM, 450 Route des Chappes, Biot 06410, France*

**Abstract**

Carrying out Bayesian inference over parameters of statistical models is intractable when the likelihood and the prior are non-conjugate. Variational bootstrap provides a way to obtain samples from the posterior distribution over model parameters, where each sample is the solution of a task where the labels are perturbed. For Bayesian linear regression with a Gaussian likelihood, variational bootstrap yields samples from the exact posterior, whereas for nonlinear models with a Gaussian likelihood some guarantees of approaching the true posterior can be established. In this work, we extend variational bootstrap to the Bernoulli likelihood to tackle classification tasks. We use a transformation of the labels which allows us to turn the classification task into a regression one, and then we apply variational bootstrap to obtain samples from an approximate posterior distribution over the parameters of the model. Variational bootstrap allows us to employ advanced gradient optimization techniques which provide fast convergence. We provide experimental evidence that the proposed approach allows us to achieve classification accuracy and uncertainty estimation comparable with MCMC methods at a fraction of the cost.



*Keywords:* Bayesian classifier; bootstrap; neural networks; Gaussian processes

## 1. Introduction

The Bayesian treatment of statistical models is desirable in applications where quantification of uncertainty is a primary requirement. For many classes of models, this is analytically intractable, and one needs to resort to approximations. Given a statistical model with parameters on which a prior distribution is assumed, such approximations yield an approximation to the posterior distribution over these parameters either in closed form or in the form of samples. Popular approaches include the Laplace Approximation, Variational Inference, and Markov chain Monte Carlo.

In this work, we focus on an alternative approach called *variational bootstrap* [21]. Variational bootstrap works by producing a set of replicas of the data set with perturbed labels. Then, each of these perturbed problems is solved by maximum-a-posteriori-type optimization. Perhaps not surprising, in the Bayesian linear regression case it is possible to introduce perturbations in a way such that the set of solutions to the perturbed problems is distributed exactly

* Corresponding author. Tel.: +33 4 93 00 81 00.
*E-mail address:* Bogdan.Kozyrskiy@eurecom.fr

as the posterior over the parameters. One remarkable property of this approach is that it transforms the problem of characterizing the posterior distribution over model parameters into a set of easily parallelizable optimization problems. Milios et al. [21] provide an extension of this result to the case of regression with deep neural networks featuring ReLU activations, where certain theoretical guarantees are given for the minimization of the KL divergence between the approximate and the true posterior. The main limitation of this approach is that no guarantees are provided for non-Gaussian likelihoods, such as Bernoulli or Multinomial, which are associated with classification problems.

In this work, we broaden the scope of variational bootstrap by extending it to classification problems. We propose to transform Bernoulli/Multinomial distributed labels to a latent representation with a Gaussian noise. We can then apply a model with a Gaussian likelihood to solve a regression problem within this latent space. To carry out this transformation, we follow the method in [20], where classification labels are interpreted as the output of a Dirichlet distribution. We thus propose the combination of variational bootstrap with Dirichlet-based classification; this allows one to obtain reliable uncertainty estimates for the classification model at a lower cost than other Bayesian inference methods, and it enables easy parallelization.

We study the proposed extension to the nonlinear case on deep neural networks for classification with ReLU activations. The transformation of the labels from discrete to continuous allows us to borrow the results obtained in the Gaussian likelihood case. As a result, we obtain a method for which we have guarantees that the optimization of the perturbed problems yields an improvement of the approximation to the posterior over model parameters. In the experiments, we showcase results on various data sets demonstrating that this is a competitive approach to characterize the posterior over model parameters. Crucially, our proposal is extremely easy to parallelize and we view this as a considerable advantage compared to alternatives to obtain the posterior over model parameters.

The proposed extension is also relevant for Bayesian logistic regression applied to large datasets with a large number of features. An interesting application of this approach that we showcase in the paper is the approximation of Gaussian process classifiers with random features [24], which turns the model into a Bayesian logistic regression model. The larger the set of random features, the better the approximation, but this has a negative impact on the computational cost. The proposed approach yields a very practical and effective method to overcome these difficulties.

The paper is organized as follows. Section 2 discusses the related work. In Section 3, we introduce some background concepts, while Section 4 discusses variational bootstrap and the transformation of the labels which we need to frame classification problems as regression. Finally, Section 5 reports the results and Section 6 concludes the paper.

## 2. Related Work

In recent years, neural networks (NN) have gained popularity due to their effectiveness on a broad variety of tasks including image classification [12], natural language processing [5] and many others [2, 10, 23, 22]. One of the most persistent challenges is that NNs tend to make overconfident decisions [11, 14, 16]. In the literature, overconfident decisions are treated by introducing uncertainty to outputs of a NN. In practice, it means that the model produces a predictive distribution of outputs for each input vector. This practice offers ways to quantify uncertainty of predictions, as high predictive variance implies lack of confidence. Methods that provide uncertainty quantification for NNs include deep ensembles [17], Monte Carlo dropout [9] and Bayesian Neural Networks (BNN) [19].

In this work, our focus is on BNNs. In contrast with the deterministic networks, BNNs consider parameters to be random variables that are associated with some prior distribution. Instead of a training procedure based on optimization, this treatment requires characterizing a posterior distribution over their weights given observations by means of Bayes theorem. The main challenge is that this procedure is intractable for nonlinear models, which has motivated the development of approximations using, for instance, Variational Inference (VI) [1, 8]. These techniques, while being relatively cheap in terms of computational resources, require selecting a family of distributions, that is used for approximation of the posterior distribution. The choice of a family of approximate posteriors has a crucial effect on the performance of the model and on the reliability of uncertainty estimations. Because different models and tasks require different families of approximate posterior distributions, there are no reliable heuristics for this task. As an alternative to approximate inference, it is possible to use Markov Chain Monte Carlo (MCMC) techniques, that allows one to get samples from the true posterior distribution of the weights. The recently proposed Stochastic Gradient Hamiltonian Monte-Carlo (SGHMC) method [4] allows obtaining samples from a true posterior distribution over the weights for

large-scale problems using mini-batching. Despite a significant reduction of the time complexity comparing to other MCMC methods, SGHMC may still be slow to converge.

Another family of Bayesian inference techniques is particle-optimization variational inference (POVI) [18, 6, 21]. Methods from this family use particles that are different instances of the model, which are optimized independently. The resulting set of optimized particles serves as an approximate posterior distribution that is more flexible than VI solutions. In this work we focus on variational bootstrap [21], which provides a theoretical connection between POVI and parametric bootstrap [7] for regression tasks, and our contribution is to extend this method to classification problems.

## 3. Preliminaries

In classification and regression tasks, the objective is to tune the parameters $\boldsymbol{\theta}$ of a function $f(\mathbf{x}, \boldsymbol{\theta})$ so that it best explains a set of given observations. The Bayesian paradigm dictates that we specify a prior distribution $p(\boldsymbol{\theta})$, which captures any prior knowledge about the model. For a dataset $\mathcal{D}\{(\mathbf{x_i}, y_i) \mid i = 1 \dots n\}$ of $n$ input vectors $\mathbf{x_i}$ and labels $y_i$, it is possible to define the posterior distribution over $\boldsymbol{\theta}$ after observing the data $\mathcal{D}$ by means of Bayes' rule:

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}} \tag{1}$$

where $p(\mathcal{D}|\boldsymbol{\theta})$ is known as the *likelihood* model, while the integral term in the denominator above is referred to as *marginal likelihood* or *evidence*.

*Bayesian Neural Networks.* We consider $f$ to be defined as a fully-connected multi-layer perceptron (MLP) with $L$ layers, each of which is given by the following formula:

$$f_l(\mathbf{x}) = \frac{1}{\sqrt{D_{l-1}}}\Big(\mathbf{W}_l\, \varphi(f_{l-1}(\mathbf{x}))\Big) + b_l, \qquad l \in \{1, ..., L+1\}, \tag{2}$$

where $\mathbf{W}_l$ and $b_l$ are the weights and the biases of the $l$-th layer, $\varphi$ is some non-linear function (i.e. ReLU), and $D_l$ denotes the dimension of the input for the corresponding layer. By dividing each layer with $\sqrt{D_l}$, we ensure that the variance of the output does not explode in the limit where the width of the network tends to infinity; this specification is known as NTK parameterization [13]. We shall use $\theta$ to refer to the set of the parameters: $\theta \coloneqq \{\mathbf{W}_l, b_l\}_{l=1}^{L}$, while we define $\boldsymbol{\theta} \coloneqq \mathrm{vec}(\theta) \in \mathbb{R}^m$ to be the corresponding vectorized form.

*Random Fourier Features Approximation of Gaussian Processes.* We shall also examine the case of linear models, as they can serve as scalable approximations to another class of Bayesian models, namely Gaussian processes [25]. Following the random Fourier features (RFF) approximation [24], we consider $\boldsymbol{\phi}(\mathbf{x}_i) \in \mathbb{R}^{D\times 1}$ to be the projection of an input point $\mathbf{x}_i \in \mathbb{R}^{d\times 1}$ onto a feature space of $D$ trigonometric basis functions. Then $\boldsymbol{\Phi} \in \mathbb{R}^{D\times N}$ denotes the design matrix of the entire training set in the feature space. In this case, the model parameters can be directly described as a vector: $\theta \coloneqq \mathbf{w} \in \mathbb{R}^m$. Given a Gaussian likelihood $\mathcal{N}(y; f(\mathbf{x}, \mathbf{w}), \sigma^2)$ and a Gaussian prior over the weights $\mathbf{w} \sim \mathcal{N}(0, \alpha^2\mathbf{I_m})$, then the posterior distribution over $\mathbf{w}$ after observing the dataset $\mathcal{D}$ is known to be Gaussian, yielding the following predictive mean and variance for a test point $\mathbf{x}_*$:

$$\mathrm{E}[f(\mathbf{x}_*)] = \frac{1}{\sigma^2}\boldsymbol{\phi}(\mathbf{x}_*)^\top \mathbf{A}^{-1}\boldsymbol{\Phi}\mathbf{y}, \qquad \mathrm{Var}[f(\mathbf{x}_*)] = \boldsymbol{\phi}(\mathbf{x}_*)^\top \mathbf{A}^{-1}\boldsymbol{\phi}(\mathbf{x}_*), \qquad \text{where } \mathbf{A} = \frac{1}{\sigma^2}\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \frac{1}{\alpha^2}I. \tag{3}$$

We observe that in order to make a prediction, one has to solve two $D \times D$ linear systems: $\mathbf{A}^{-1}\boldsymbol{\Phi}\mathbf{y}$ is solved only once, but $\mathbf{A}^{-1}\boldsymbol{\phi}(\mathbf{x}_*)$ has to be solved for every new test point $\mathbf{x}_*$. So for $n_*$ test points, this translates to $O(n_* \times D^2)$ complexity.

If the number of test points is large, then it is preferable to directly calculate the decomposition of the matrix $\mathbf{A}$, so that it can be reused to solve the linear systems needed to calculate the predictive distribution for new test points. In many problems however, a large number of features might be required to obtain an accurate approximation of a GP. Later, we show that variational bootstrap can keep the computational cost down, when both $D$ and $n_*$ are large.

## 4. Methods

### *4.1. Variational Bootstrap for Neural Network Regression*

In its original formulation, variational bootstrap was defined as an alternative to Bayesian inference for regression tasks. More specifically, the objective is to approximate the posterior distribution of a nonlinear model with a Gaussian likelihood and a Gaussian prior as follows:

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{i=1}^{n} \mathcal{N}(y_i; f(\mathbf{x}_i, \boldsymbol{\theta}), \sigma^2) \qquad \text{and} \qquad p(\boldsymbol{\theta}) = \mathcal{N}(0, \alpha^2 \mathbf{I}_m). \tag{4}$$

For this model, variational bootstrap yields a set of samples that represents an empirical distribution $q$ and approximates the posterior distribution of the parameters $\boldsymbol{\theta}$. These samples are obtained by optimization of a set of particles. Each particle is defined as the *maximum a posteriori* (MAP) estimate for a regression task on a perturbed version of the joint log-likelihood, with its own set of training labels and mean parameters of the prior distribution. The training labels for each particle are obtained by a parametric bootstrap procedure. For each given label $y_i$, we generate a perturbed label $\tilde{y}_i$ according to the likelihood function, that is Gaussian with variance $\sigma^2$ and mean $y_i$:

$$\tilde{y}_i^{(k)} \sim \mathcal{N}\left(y_i, \sigma^2\right), \; k = 1...K \tag{5}$$

Also, each particle is associated with a unique sample from the prior distribution $\tilde{\boldsymbol{\theta}}^{(k)}$. So the new prior for each model in the ensemble becomes as follows:

$$p(\boldsymbol{\theta}^{(k)}, \tilde{\boldsymbol{\theta}}^{(k)}) \sim \mathcal{N}(\tilde{\boldsymbol{\theta}}^{(k)}, \alpha^2 \mathbf{I}_m), \text{ where } \tilde{\boldsymbol{\theta}}^{(k)} \sim \mathcal{N}(0, \alpha^2 \mathbf{I}_m), \; k = 1...K \tag{6}$$

After resampling $K$ perturbed sets of the labels and parameters of the prior distribution, we can obtain $K$ samples from the approximate posterior, by solving $K$ optimization problems of the form:

$$\text{argmin}_{\boldsymbol{\theta}^{(k)}} \frac{1}{2\sigma_n^2} \sum_{i=1}^{N} (\tilde{y}_i^{(k)} - f(\mathbf{x}_i, \boldsymbol{\theta}^{(k)}))^2 + \frac{1}{2\alpha^2} \|\boldsymbol{\theta}^{(k)} - \tilde{\boldsymbol{\theta}}^{(k)}\|^2, \; k = 1...K \tag{7}$$

The parameters of the particles are updated by a gradient descent algorithm. For the case of NNs under the additional assumption of linear or piecewise linear activation functions, it is shown in [21] that each gradient step optimizes the joint log-likelihood of each model and moves the distribution of parameters $q(\boldsymbol{\theta})$ closer to the true posterior $p(\boldsymbol{\theta}|\mathcal{D})$.

### *4.2. Variational Bootstrap for Random Fourier Features*

We demonstrate here that variational bootstrap can induce computational advantages also for the RFF approximation of Gaussian processes. If the labels are perturbed according to the likelihood so that $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, and the

regularization term is a sample from the prior so that $\tilde{\mathbf{w}} \sim \mathcal{N}(0, \alpha^2 \mathbf{I}_m)$, then the MAP solution is:

$$\hat{\mathbf{w}} = \frac{1}{\sigma^2}\mathbf{A}^{-1}\mathbf{\Phi}(\mathbf{y} + \varepsilon) + \frac{1}{\alpha^2}\mathbf{A}^{-1}\tilde{\mathbf{w}}, \tag{8}$$

The MAP estimate $\hat{\mathbf{w}}$ is a Gaussian random vector, as it is a linear combination of two Gaussian random perturbations: $\varepsilon$ and $\tilde{\mathbf{w}}$. According to [21], we can calculate the expectation and the covariance of $\hat{\mathbf{w}}$ to obtain the true posterior mean and covariance for the weights of the linear model:

$$\mathrm{E}_{\varepsilon,\tilde{\mathbf{w}}}[\hat{\mathbf{w}}] = \frac{1}{\sigma^2}\mathbf{A}^{-1}\mathbf{\Phi}\mathbf{y} =: \bar{\mathbf{w}}, \qquad \mathrm{E}_{\varepsilon,\tilde{\mathbf{w}}}[(\hat{\mathbf{w}} - \bar{\mathbf{w}})(\hat{\mathbf{w}} - \bar{\mathbf{w}})^\top] = \mathbf{A}^{-1} \tag{9}$$

Equation (8) yields samples from the true posterior distribution. From a computational perspective, we observe that it behaves differently from the predictive posterior in (3). Here, exactly two linear systems have to be solved for every sample, as opposed to every test point as in (3). Let $K$ denote the number of samples; then the complexity becomes $O(K \times D^2)$. If the number of test points is larger than the number of posterior samples, this can induce significant computational gains, as we demonstrate in the experimental section.

### *4.3. Dirichlet Label Transformation*

The goal of Bayesian classification is to estimate the distribution of class probabilities for an input data point. The Gaussian likelihood model we discussed in the previous section is not appropriate for a classification task; it is more reasonable to use a Multinomial likelihood instead. For a $C$-class classification problem, the class label $\mathbf{y}$ for the input point $\mathbf{x}$ is a sample from the Multinomial distribution $\mathbf{y} \sim \mathrm{Cat}(\boldsymbol{\pi})$. The authors of [20] propose to use a $C$-dimensional Dirichlet distribution to model the distribution of class probabilities $\boldsymbol{\pi} \sim \mathrm{Dir}(\boldsymbol{\alpha})$, with parameters $\boldsymbol{\alpha} = [\alpha_1, ..., \alpha_C]^T$. Any label observations are treated as Dirichlet distributions: if an input point $\mathbf{x}$ belongs to a class $k$, then it corresponds to the following Dirichlet parameters:

$$\alpha_i = \begin{cases} 1 + \alpha_\epsilon, & \text{if } i = k. \\ \alpha_\epsilon, & \text{if } i \neq k. \end{cases} \tag{10}$$

The term $\alpha_\epsilon > 0$ represents a small quantity added to the Dirichlet parameters in order to guarantee a valid Dirichlet distribution. Then it is possible to represent samples from the $C$-dimensional Dirichlet distribution as samples from $C$ Gamma distributions: $\pi_i = \frac{z_i}{\sum_{c=1}^{C} z_c}$, where $z_i \sim \mathrm{Gamma}(\alpha_i, 1)$, for $i \in \{1, ..., C\}$.

The original paper proposes to approximate the Gamma distribution by a Lognormal$(\tilde{y}_i, \sigma_i^2)$ distribution whose parameters are determined by moment matching. Because the logarithm of the log-normally distributed random variable has a Gaussian distribution $\mathcal{N}(\tilde{y}_i, \sigma_i^2)$, it becomes possible to use a Gaussian likelihood and thus transform the classification problem into a regression problem: the transformed labels $\tilde{y}_i$ become the new targets for the inputs $\mathbf{x}_i$ and $\sigma_i^2$ becomes the variance parameter of the Gaussian likelihood.

$$\sigma_i^2 = \log(1/\alpha_i + 1) \qquad \hat{y}_i = \log \alpha_i - \sigma_i^2/2 \tag{11}$$

It is worth mentioning, that the expression (11) produces different noise parameters $\boldsymbol{\sigma}^2$ for each observation $\mathbf{y}$. This means that after the transformation, we are dealing with heteroskedastic linear regression.

In order to train the whole classification model, it is required to solve $C$ regression problems, one for each dimension of the Dirichlet distribution. To make predictions, one has to apply a softmax transformation to the outputs of the $C$ regression models $\mathbf{f} = [f_1, ..., f_C]^T$, as follows:

$$\mathbb{E}[\pi_i|\mathbf{x}] = \int \frac{\exp(f_i(\mathbf{x}))}{\sum_{c=1}^{C} \exp(f_c(\mathbf{x})} p(f_i(\mathbf{x})|X) d\mathbf{f}(\mathbf{x}) \tag{12}$$

### *4.4. Classification with Variational Bootstrap*

One of the key components of variational bootstrap for regression is the data resampling via parametric bootstrap. In the case of classification, it is not straightforward to adjust this strategy to produce perturbed versions of the class labels in a way that reflects the nature of the Bernoulli (or the Multinomial) likelihood. For example, a class label $y$ can take values in $\{0, 1\}$; if we locally fit a distribution $Bern(p)$ to each $y$, the maximum-likelihood parameter will be the one-sample mean, i.e., $p = 0$ or $p = 1$. If we use this fitted model (i.e. Bernoulli with parameter 0 or 1) to resample new labels, this will deterministically produce either 0 or 1, depending on the original label.

Therefore, we have adopted a strategy that combines variational bootstrap with the Dirichlet labels transformation. Consider a dataset $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$ of input vectors $\mathbf{x}_i$ and labels $y_i$ for a $C$-class classification problem. As a first step, we transform each label $y_i$ into a pair $\{\hat{\mathbf{y}}_i, \boldsymbol{\sigma}_i\}$, $\hat{\mathbf{y}}_i \in \mathbb{R}^C$, $\boldsymbol{\sigma}_i \in \mathbb{R}^C$ by the means of the Dirichlet label transformation. The $C$-class classification problem is then transformed into a $C$-dimensional heteroskedastic regression problem with labels $\hat{\mathbf{y}}_i$ and observation noise variances $\boldsymbol{\sigma}_i^2$; each training sample and each dimension of the output has observation noise with its own variance.

The second step involves an application of the variational bootstrap method, where we generate $K$ independent sets of regression labels, as well as $K$ sets of prior parameters. Perturbations of each transformed label $\hat{\mathbf{y}}_i$ from the training set are sampled from the corresponding distribution with variance $\boldsymbol{\sigma}_i^2$ as follows:

$$\tilde{\mathbf{y}}_i \sim \mathcal{N}(\hat{\mathbf{y}}_i, \text{Diag}(\boldsymbol{\sigma}_i^2)) \qquad \text{where} \qquad \text{Diag}(\boldsymbol{\sigma}_i^2) = \mathbf{I} \odot (\boldsymbol{\sigma}_i^2 \mathbf{1}^\top) \tag{13}$$

**Algorithm 1** Variational bootstrap for MLP

**Input:** $\mathbf{X}$, $\mathbf{y}$, $\alpha$, $\alpha_\epsilon$, $h$
**Output:** $\mathbf{w} \sim q(\mathbf{w})$
**for** $i \leftarrow 1$ **to** $N$ **do**
    $\hat{\mathbf{y}}_i, \boldsymbol{\sigma}_i \leftarrow \text{DirichletTransform}(y_i, \alpha_\epsilon)$ ▷ Eq. (11)
**end for**
**for** $k \leftarrow 1$ **to** $K$ **do**
    $\tilde{\mathbf{y}}_1^{(k)}, \ldots \tilde{\mathbf{y}}_N^{(k)} \sim \mathcal{N}(\hat{\mathbf{y}}_1, \boldsymbol{\sigma}_1^2), \ldots, \mathcal{N}(\hat{\mathbf{y}}_N, \boldsymbol{\sigma}_N^2)$
    Draw sample $\tilde{\mathbf{w}}$ from $\mathcal{N}(0, \alpha^2\mathbf{I})$
    Initialize $\mathbf{w}^{(k)} \leftarrow \tilde{\mathbf{w}}$
    $\mathbf{w}^{(k)} \leftarrow$ for each output dimension **optimize** (7)
**end for**

**Algorithm 2** Variational bootstrap for RFF

**Input:** $\boldsymbol{\Phi}$, $\mathbf{y}$, $\alpha$, $\alpha_\epsilon$
**Output:** $\{\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(K)}\} \sim p(\mathbf{w}|\boldsymbol{\Phi}, \mathbf{y})$
**for** $i \leftarrow 1$ **to** $N$ **do**
    $\hat{\mathbf{y}}_i, \boldsymbol{\sigma}_i \leftarrow \text{DirichletTransform}(y_i, \alpha_\epsilon)$
**end for**
**for** $k \leftarrow 1$ **to** $K$ **do**
    $\tilde{\mathbf{y}}_1^{(k)}, \ldots \tilde{\mathbf{y}}_N^{(k)} \sim \mathcal{N}(\hat{\mathbf{y}}_1, \boldsymbol{\sigma}_1^2), \ldots, \mathcal{N}(\hat{\mathbf{y}}_N, \boldsymbol{\sigma}_N^2)$
    Draw sample $\tilde{\mathbf{w}}$ from $\mathcal{N}(0, \alpha^2\mathbf{I})$
    $\mathbf{w}^{(k)} \leftarrow$ for each output dimension **solve** (8)
**end for**

The procedure for applying variational bootstrap to classification is summarized in Algorithm 1 for MLP models, and in Algorithm 2 for the RFF Gaussian process approximation. In both cases, parameters of the prior distribution for each particle are sampled according to (6), while the labels are perturbed as in (13). Then in Algorithm 1 we optimize $K$ models independently. For Algorithm 2, the main difference is that we obtain the solution of regression

problem (7) analytically instead of using gradient optimization. After optimization of the $K$ particles, it is possible to perform predictions using a Monte Carlo approximation of Equation (12). As a final remark, our choice to employ the Dirichlet label transformation and to treat the classification problem as regression implies that our methods enjoys any convergence guarantees that can be proven for the regression case.

## 5. Results

### 5.1. Toy dataset

We first demonstrate the application of variational bootstrap on a synthetic one-dimensional binary classification problem. We considered a Gaussian process classifier based on a radial basis function (RBF) kernel, and we approximated it with a Bayesian logistic regression model on a set of random features. We used Random Fourier Features approximation of the RBF kernel proposed in [24]. For our experiment we considered 5,000 random features. On the left panel of Fig. 1, we show the distribution over functions corresponding to the approximate posterior distribution over the parameters obtained with variational bootstrap with the Dirichlet label transformation; optimization was performed via the L-BFGS algorithm. In the middle panel of the figure, we report the distribution over functions obtained by the same approximation of the model, but where inference is carried out by Markov chain Monte Carlo (MCMC). On the right panel of the figure, we show the distribution of functions obtained with the Laplace approximation. For the toy dataset, we used the Metropolis-Hastings algorithm with 100 chains. For the prediction, we took the last sample from each chain. R-hat convergence diagnostic [3] showed that it takes around $10^6$ steps before convergence. The comparison shows a remarkable property of the proposed approach to accurately approximate the posterior over model parameters without the need for expensive or excessively long computations. In fact, the L-BFGS algorithm converged for variational bootstrap after 16 iterations only.

#### 5.1.1. UCI Datasets

Table 1. UCI datasets used for evaluation.

| Dataset | Classes | Training instances | Test instances | Dimensionality |
|---|---|---|---|---|
| Magic | 2 | 14020 | 5000 | 10 |
| HTRU2 | 2 | 12898 | 5000 | 8 |
| MiniBoo | 2 | 120064 | 10000 | 50 |
| Drive | 11 | 48509 | 10000 | 54 |
| Letter | 26 | 15000 | 5000 | 16 |
| Mocap | 5 | 68095 | 10000 | 37 |

We evaluated our method on several UCI classification problems outlined in Table 1. We applied variational bootstrap with the Dirichlet label transformation on two different models. First, we considered a two-hidden layer MLP model with a ReLU activation function and 512 neurons in each hidden layer. We used Adam [15] to optimize the parameters of the model. This model is referred to as VBoot-MLP. The second model we considered is the RFF

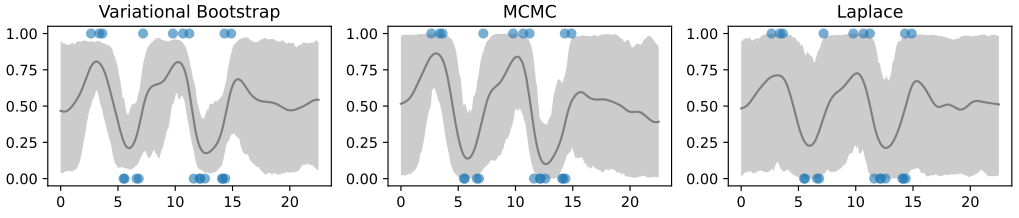


Fig. 1. Comparison between the predictions with the regression weights, obtained by the variational bootstrap (left), MCMC (middle) and Laplace approximation (right)

approximation [24] of a GP, where 5,000 random features were used to approximate an RBF kernel with fixed hyperparameters. In this case, we generated 50 samples as prescribed in Equation (8); this required solving 50 linear systems of order $O(D^2)$, instead of thousands as required by Equation (3). This model is referred to as VBoot-RFF.

The performance of variational bootstrap is compared against a regular BNN, whose posterior has been approximated by means of MCMC sampling, and in particular SGHMC [4]. The baseline BNN uses the same prior distribution over the parameters and a Bernoulli likelihood, while its output is given by a softmax activation function. VBoot-RFF is compared against the Dirichlet transformation with a Sparse GP approach proposed in [20]. The hyperparameters of this model were optimized as in the original paper. We evaluate performance using several metrics, including classification error and mean negative log-likelihood (MNLL). The results are outlined in Tables 2 and 3, respectively. The models considered are referred to as VBoot-MLP, VBoot-RFF, Sparse GP, and MCMC-MLP respectively.

Considering the MLP models, our experiments show that the proposed approach is competitive when compared against principled framework such as MCMC. Most importantly, our approach has significantly different behavior in terms of convergence speed, as it relies on optimization rather than sampling. We demonstrate this property empirically by monitoring the progression of validation error for variational bootstrap and MCMC. In Fig. 2, we report classification error on held-out data over the first 200 training epochs. These results indicate that variational bootstrap technique provides much faster convergence compared to SGHMC. On the other hand, only in some cases SGHMC eventually settles to significantly lower validation error. We used Wilcoxon test [27] to compare classification errors and MNLLs for VBoot-MLP and MCMC-MLP methods. The test did not show any statistically significant difference regarding the performance of these two methods within 0.05 significance level. The lack of statistical significance appears partly due to the small number of used data splits. Nevertheless, variational bootstrap is shown to achieve a good trade-off between accuracy and efficiency. This trade-off can be further exploited in practice, as variational bootstrap is trivially parallelizable.

Regarding the GP-based models, we also see that VBoot-RFF is highly competitive against traditional sparse GPs that rely on inducing points. In fact, according to the one-sided Wilcoxon signed-rank test, the VBoot-RFF results are slightly better than Sparse GP in a statistically significant way for some of the datasets (marked with "*" in Tables 2 and 3). Our variational bootstrap framework allowed us to use a large number of random features (i.e., 5,000), which would not be possible for many of the datasets considered. This can be seen in Fig. 3, which shows the computation time for VBoot-RFF and Sparse GP models as a function of the number of random features and inducing points correspondingly. Of course, using more random features/inducing points results in a better approximation of the full GP model, but it also increases computational complexity. We see that VBoot-RFF has better scalability properties compared to sparse GPs. Regarding the computation times reported for Sparse GPs, we note that we have excluded the initial K-Means step needed to initialize the inducing locations, so in practice, sparse GPs are more expensive than what we report here. Interestingly, sparse GPs could not scale beyond 1,000 or 2,000 inducing points for some datasets, due to memory errors.

Table 2. Classification error for variational bootstrap with the Dirichlet transformation and Markov-chain Monte Carlo approaches.

| Dataset | VBoot-MLP | MCMC-MLP | VBoot-RFF | Sparse GP |
|---|---|---|---|---|
| Magic | 0.12±0.01 | 0.12±0.01 | 0.13±0.01 | 0.13±0.01 |
| HTRU2 | 0.02±0.01 | 0.02±0.00 | 0.02±0.00 | 0.02±0.01 |
| MiniBoo | 0.11±0.01 | 0.09±0.01 | 0.08±0.01 | 0.08±0.01 |
| Drive | 0.01±0.00 | 0.001±0.000 | 0.01±0.00* | 0.02±0.01 |
| Letter | 0.05±0.01 | 0.03±0.01 | 0.05±0.01* | 0.08±0.01 |
| Mocap | 0.005±0.000 | 0.007±0.000 | 0.02±0.00* | 0.03±0.01 |

## 6. Conclusions

In this paper we proposed a novel way to carry out Bayesian inference for classification models based on Neural Networks and Gaussian processes. For NNs, this is important because, while they achieve state-of-the-art performance in many tasks, they lack a principled way to characterize uncertainty in predictions, so they represent a class of models

Table 3. MNLL for variational bootstrap with the Dirichlet transformation and Markov-chain Monte Carlo approaches.

| Dataset | VBoot-MLP | MCMC-MLP | VBoot-RFF | Sparse GP |
|---|---|---|---|---|
| Magic | 0.31±0.00 | 0.29±0.00 | 0.33±0.01* | 0.35±0.01 |
| HTRU2 | 0.08±0.00 | 0.07±0.00 | 0.07±0.01 | 0.07±0.01 |
| MiniBoo | 0.25±0.01 | 0.22±0.00 | 0.20±0.01* | 0.21±0.01 |
| Drive | 0.06±0.00 | 0.01±0.00 | 0.08±0.03 | 0.08±0.01 |
| Letter | 0.31±0.00 | 0.24±0.00 | 0.28±0.01 | 0.25±0.01* |
| Mocap | 0.02±0.00 | 0.03±0.00 | 0.09±0.01* | 0.13±0.01 |

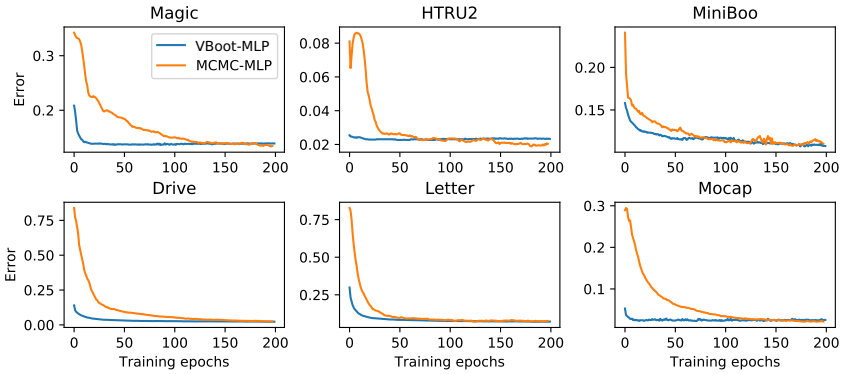


Fig. 2. Convergence of the classification error on validation data for variational bootstrap and SGHMC methods

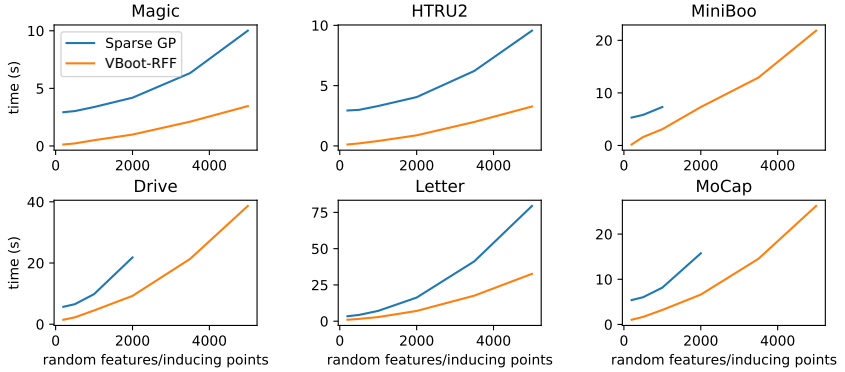


Fig. 3. Computation time complexity for VBoot-RFF and Sparse GP as function of the number of random features and inducing points correspondingly.

for which a Bayesian treatment is highly desirable but mathematically and computationally challenging. For Gaussian processes, instead, while the Bayesian treatment is at the core of its formulation, classification tasks are difficult to handle because they require expensive approximations. Our work provides a practical and easily parallelizable way to tackle all these limitations.

We are currently investigating parallel implementations of the proposed approach to considerably accelerate inference of large-scale problems by operating on clusters of computing machines. In addition, we are exploring the application of our approach to problems involving optical-based computing hardware, also known as Optical Processing Units [26]. OPUs offer a fast and low-power way to approximate Gaussian processes through random features,

and because they are capable of generating millions of these at the speed of light, we believe that our approach could be the key to exploit these computations effectively.

Another possible direction is the adaptation of our framework towards a more efficient marginal likelihood maximization for GPs, which is a standard practice to tune kernel hyperparameters [25]. In this work, we have treated GPs by means of fixed feature map approximations, which correspond to fixed hyperparameter values. Estimating marginal likelihoods through samples is an open research question, and it can be the subject of future work.

## References


[1] Blundell, C., Cornebise, J., Kavukcuoglu, K., Wierstra, D., 2015. Weight Uncertainty in Neural Network, in: Proceedings of the 32nd International Conference on Machine Learning, PMLR. pp. 1613–1622.

[2] Bourilkov, D., 2019. Machine and Deep Learning Applications in Particle Physics. International Journal of Modern Physics A 34, 1930019.

[3] Brooks, S.P., Gelman, A., 1998. General Methods for Monitoring Convergence of Iterative Simulations. Journal of computational and graphical statistics 7, 434–455.

[4] Chen, T., Fox, E., Guestrin, C., 2014. Stochastic Gradient Hamiltonian Monte Carlo, in: Proceedings of the 31st International Conference on Machine Learning, PMLR. pp. 1683–1691.

[5] Devlin, J., Chang, M.W., Lee, K., Toutanova, K., 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Association for Computational Linguistics. pp. 4171–4186.

[6] D' Angelo, F., Fortuin, V., 2021. Repulsive Deep Ensembles are Bayesian, in: Advances in Neural Information Processing Systems.

[7] Efron, B., 2012. Bayesian Inference and the Parametric Bootstrap. Ann. Appl. Stat. 6, 1971–1997.

[8] Gal, Y., Ghahramani, Z., 2016a. Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference, in: 4th International Conference on Learning Representations (ICLR) workshop track.

[9] Gal, Y., Ghahramani, Z., 2016b. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning, in: Proceedings of The 33rd International Conference on Machine Learning, PMLR. pp. 1050–1059.

[10] Gawehn, E., Hiss, J.A., Schneider, G., 2016. Deep learning in drug discovery. Molecular informatics 35, 3–14.

[11] Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q., 2017. On Calibration of Modern Neural Networks, in: Proceedings of the 34th International Conference on Machine Learning, PMLR. pp. 1321–1330.

[12] He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep Residual Learning for Image Recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778.

[13] Jacot, A., Gabriel, F., Hongler, C., 2018. Neural Tangent Kernel: Convergence and Generalization in Neural Networks, in: Advances in Neural Information Processing Systems.

[14] Kendall, A., Gal, Y., 2017. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?, in: Advances in Neural Information Processing Systems.

[15] Kingma, D.P., Ba, J., 2015. Adam: A Method for Stochastic Optimization, in: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.

[16] Kurakin, A., Goodfellow, I.J., Bengio, S., 2018. Adversarial examples in the physical world, in: Artificial intelligence safety and security. Chapman and Hall/CRC, pp. 99–112.

[17] Lakshminarayanan, B., Pritzel, A., Blundell, C., 2017. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles, in: Advances in Neural Information Processing Systems.

[18] Liu, Q., Wang, D., 2016. Stein Variational Gradient Descent: A General Purpose Bayesian Inference Algorithm, in: Advances in Neural Information Processing Systems.

[19] MacKay, D.J., 1995. Bayesian neural networks and density networks. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 354, 73–80.

[20] Milios, D., Camoriano, R., Michiardi, P., Rosasco, L., Filippone, M., 2018. Dirichlet-based Gaussian Processes for Large-scale Calibrated Classification, in: Advances in Neural Information Processing Systems.

[21] Milios, D., Michiardi, P., Filippone, M., 2021. A Variational View on Bootstrap Ensembles as Bayesian Inference, in: AABI 2021, 3rd Symposium on Advances in Approximate Bayesian Inference, January-February 2021.

[22] Nguyen, H., Kieu, L.M., Wen, T., Cai, C., 2018. Deep learning methods in transportation domain: a review. IET Intelligent Transport Systems 12, 998–1004.

[23] Purwins, H., Li, B., Virtanen, T., Schlüter, J., Chang, S.Y., Sainath, T., 2019. Deep Learning for Audio Signal Processing. IEEE Journal of Selected Topics in Signal Processing 13, 206–219.

[24] Rahimi, A., Recht, B., 2007. Random Features for Large-Scale Kernel Machines, in: Advances in Neural Information Processing Systems.

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

[26] Saade, A., Caltagirone, F., Carron, I., Daudet, L., Drémeau, A., Gigan, S., Krzakala, F., 2016. Random Projections through multiple optical scattering: Approximating kernels at the speed of light, in: 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE. pp. 6215–6219.

[27] Wilcoxon, F., 1945. Individual Comparisons by Ranking Methods. Biometrics Bulletin 1, 80–83.

**List of changes from the original submission**

The following changes have been made according to the comments from the reviewers.

*General changes*

We replaced the abbreviation VB, that states for variational bootstrap with VBoot, to avoid confusion with Variational Bayes. We fixed several typos and incoherences in mathematical notation.

*Changes in the **Related work** section*

We have emphasized the key differences between our method and parametric Variational inference techniques in terms of flexibility of approximated posterior distribution and over MCMC techniques in terms of computational speed. Regarding particle-based methods, which are more closely related to ours, we emphasized that our method extends to classification problems, which was not explored in the context of variational bootstrap.

*Changes in the **Methods** section*

We added pseudocode algorithms that describe the application of our method to multilinear perceptrons and Random Fourier Feature linear models.

*Changes in the **Results** section*

We added Sparse Gaussian process (GP) baseline, and we have revised the discussion of the experimental results by focusing on the qualitative differences between our method (VBoot-MLP & VBoot-RFF) and the baselines.

We used the one-sided Wilcoxon signed-rank test to verify that our method is comparable with the baselines in terms of classification error and MNLL. We used a Wilcoxon pairwise statistical test because it allows to compare the performance of two methods within different splits of each dataset. The performances of two methods on a single data split represented by a pair of two values. The statistical analysis of the whole experiment boils down to a comparison of the performance metrics within pairs for all data splits. That is why we needed a pairwise test. And given the fact, that we do not have any assumptions about the distribution of the analyzed values, Wilcoxon test is an adequate choice.

We also performed an additional set of experiments for Random Fourier Feature and Sparse GP models. In these experiments, we showed that our method demonstrates better scalability properties than regular Sparse GPs.