Fast and Scalable Bayesian Deep Learning by Weight-Perturbation in Adam

Mohammad Emtiyaz Khan [*1] Didrik Nielsen [*1] Voot Tangkaratt [*1] Wu Lin [2] Yarin Gal [3] Akash Srivastava [4]

Abstract

Uncertainty computation in deep learning is essential to design robust and reliable systems. Variational inference (VI) is a promising approach for such computation, but requires more effort to implement and execute compared to maximum-likelihood methods. In this paper, we propose new natural-gradient algorithms to reduce such efforts for Gaussian mean-field VI. Our algorithms can be implemented within the Adam optimizer by perturbing the network weights during gradient evaluations, and uncertainty estimates can be cheaply obtained by using the vector that adapts the learning rate. This requires lower memory, computation, and implementation effort than existing VI methods, while obtaining uncertainty estimates of comparable quality. Our empirical results confirm this and further suggest that the weight-perturbation in our algorithm could be useful for exploration in reinforcement learning and stochastic optimization.

1. Introduction

Deep learning methods have had enormous recent success in fields where prediction accuracy is important, e.g., computer vision and speech recognition. However, for these methods to be useful in fields such as robotics and medical diagnostics, we need to know the uncertainty of our predictions. For example, physicians might need such uncertainty estimates to choose a safe but effective treatment for their patients. Lack of such estimates might result in unreliable decisions which can sometime have disastrous consequences.

One of the goals of Bayesian inference is to provide uncertainty estimates by using the *posterior distribution* obtained using Bayes' rule. Unfortunately, this is infeasible in large models such as Bayesian neural networks. Traditional methods such as Markov Chain Monte Carlo (MCMC) methods converge slowly and might require a large memory (Balan et al., 2015). In contrast, variational inference (VI) methods can scale to large models by using stochastic-gradient (SG) methods, as recent work has shown (Graves, 2011; Blundell et al., 2015; Ranganath et al., 2014; Salimans et al., 2013). These works employ adaptive learning-rate methods, such as RMSprop (Tieleman & Hinton, 2012), Adam (Kingma & Ba, 2015) and AdaGrad (Duchi et al., 2011), for which easy-to-use implementations are available in existing codebases.

Despite their simplicity, these VI methods require more computation, memory, and implementation effort compared to maximum-likelihood estimation (MLE). One reason for this is that the number of parameters in VI is usually much larger than in MLE, which increases the memory and computation costs. Another reason is that existing codebases are designed and optimized for tasks such as MLE, and their application to VI involves significant amount of modifications in the code. We ask the following question: is it possible to avoid these issues and make VI as easy as MLE?

In this paper, we propose to use *natural-gradient methods* to address these issues for Gaussian mean-field VI. By proposing a *natural-momentum method* along with a series of approximations, we obtain algorithms that can be implemented with minimal changes to the existing codebases of adaptive learning-rate methods. The main change involves perturbing the network weights during the gradient computation (see Fig. 1). An uncertainty estimate can be cheaply obtained by using the vector that adapts the learning rate. This requires lower memory, computation, and implementation efforts than existing methods for VI while obtaining uncertainty estimates of comparable quality. Our experimental results confirm this, and suggest that the estimated uncertainty could improve exploration in problems such as reinforcement learning and stochastic optimization.

1.1. Related Work

Bayesian inference in models such as neural networks has a long history in machine learning (MacKay, 2003; Bishop, 2006). Earlier work proposed a variety of algo-

[*]Equal contribution [1]RIKEN Center for Advanced Intelligence project, Tokyo, Japan [2]University of British Columbia, Vancouver, Canada [3]University of Oxford, Oxford, UK [4]University of Edinburgh, Edinburgh, UK. Correspondence to: Mohammad Emtiyaz Khan <emtiyaz.khan@riken.jp>.

Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

```
Adam
 1: while not converged do
 2:    $\boldsymbol{\theta} \leftarrow \boldsymbol{\mu}$
 3:    Randomly sample a data example $\mathcal{D}_i$
 4:    $\mathbf{g} \leftarrow -\nabla \log p(\mathcal{D}_i|\boldsymbol{\theta})$
 5:    $\mathbf{m} \leftarrow \gamma_1 \mathbf{m} + (1-\gamma_1)\, \mathbf{g}$
 6:    $\mathbf{s} \leftarrow \gamma_2 \mathbf{s} + (1-\gamma_2)\, (\mathbf{g} \circ \mathbf{g})$
 7:    $\hat{\mathbf{m}} \leftarrow \mathbf{m}/(1-\gamma_1^t), \quad \hat{\mathbf{s}} \leftarrow \mathbf{s}/(1-\gamma_2^t)$
 8:    $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \alpha\, \hat{\mathbf{m}}/(\sqrt{\hat{\mathbf{s}}} + \delta)$
 9:    $t \leftarrow t+1$
10: end while
```

```
Vadam
 1: while not converged do
 2:    $\boldsymbol{\theta} \leftarrow \boldsymbol{\mu} + \boldsymbol{\sigma} \circ \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$, $\boldsymbol{\sigma} \leftarrow 1/\sqrt{N\mathbf{s} + \lambda}$
 3:    Randomly sample a data example $\mathcal{D}_i$
 4:    $\mathbf{g} \leftarrow -\nabla \log p(\mathcal{D}_i|\boldsymbol{\theta})$
 5:    $\mathbf{m} \leftarrow \gamma_1 \mathbf{m} + (1-\gamma_1)\, (\mathbf{g} + \lambda\boldsymbol{\mu}/N)$
 6:    $\mathbf{s} \leftarrow \gamma_2 \mathbf{s} + (1-\gamma_2)\, (\mathbf{g} \circ \mathbf{g})$
 7:    $\hat{\mathbf{m}} \leftarrow \mathbf{m}/(1-\gamma_1^t), \quad \hat{\mathbf{s}} \leftarrow \mathbf{s}/(1-\gamma_2^t)$
 8:    $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \alpha\, \hat{\mathbf{m}}/(\sqrt{\hat{\mathbf{s}}} + \lambda/N)$
 9:    $t \leftarrow t+1$
10: end while
```

Figure 1. Comparison of Adam (left) and one of our proposed method Vadam (right). Adam performs maximum-likelihood estimation while Vadam performs variational inference, yet the two pseudocodes differ only slightly (differences highlighted in red). A major difference is in line 2 where, in Vadam, weights are perturbed during the gradient evaluations.

rithms such as MCMC methods (Neal, 1995), Laplace's method (Denker & Lecun, 1991), and variational inference (Hinton & Van Camp, 1993; Barber & Bishop, 1998). The mean-field approximation has also been a popular tool from very early on (Saul et al., 1996; Anderson & Peterson, 1987). These previous works lay the foundation of methods now used for Bayesian deep learning (Gal, 2016).

Recent approaches (Graves, 2011; Blundell et al., 2015) enable the application of Gaussian mean-field VI methods to large deep-learning problems. They do so by using gradient-based methods. In contrast, we propose to use *natural*-gradient methods which, as we show, lead to algorithms that are simpler to implement and require lower memory and computations than gradient-based methods. Natural gradients are also better suited for VI because they can improve convergence rates by exploiting the information geometry of posterior approximations (Khan et al., 2016). Some of our algorithms inherit these properties too.

A recent independent work on noisy-Adam by Zhang et al. (2018) is algorithmically very similar to our Vadam method, however their derivation lacks a strong motivation for the use of momentum. In our derivation, we incorporate a *natural-momentum* term based on Polyak's heavy-ball method, which provides a theoretical justification for the use of momentum. In addition, we analyze the approximation error introduced in Vadam and discuss ways to reduce it.

Zhang et al. (2018) also propose an interesting extension by using K-FAC, which could find better approximations than the mean-field method. The goal of this approach is similar to other approaches that employ *structured* approximations (Ritter et al., 2018; Louizos & Welling, 2016; Sun et al., 2017). Many other works have explored variety of approximation methods, e.g., Gal & Ghahramani (2016) use dropout for VI, Hernandez-Lobato & Adams (2015); Hasenclever et al. (2017) use expectation propagation, Li et al. (2016); Balan et al. (2015) use stochastic-gradient Langevin dynamics. Such approaches are viable alternatives to the mean-field VI approach we use.

Another related work by Mandt et al. (2017) views SG descent as VI but requires additional effort to obtain posterior approximations, while in our approach the approximation is automatically obtained within an adaptive method.

Our weight-perturbed algorithms are also related to global-optimization methods, e.g., Gaussian-homotopy continuation methods (Mobahi & Fisher III, 2015), smoothed-optimization method (Leordeanu & Hebert, 2008), graduated optimization method (Hazan et al., 2016), and stochastic search methods (Zhou & Hu, 2014). In particular, our algorithm is related to recent approaches in deep learning for exploration to avoid local minima, e.g., natural evolution strategy (Wierstra et al., 2014), entropy-SGD (Chaudhari et al., 2016), and noisy networks for reinforcement learning (Fortunato et al., 2018; Plappert et al., 2018). An earlier version of our work (Khan et al., 2017) focuses exclusively on this problem, and in this paper we modify it to be implemented within an adaptive algorithm like Adam.

2. Gaussian Mean-Field Variational Inference

We consider modeling of a dataset $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_N\}$ by using a deep neural network (DNN). We assume a probabilistic framework where each data example $\mathcal{D}_i$ is sampled independently from a probability distribution $p(\mathcal{D}_i|\boldsymbol{\theta})$ parameterized by a DNN with weights $\boldsymbol{\theta} \in \mathbb{R}^D$, e.g., the distribution could be an exponential-family distribution whose mean parameter is the output of a DNN (Bishop, 2006).

One of the most popular approaches to estimate $\boldsymbol{\theta}$ given $\mathcal{D}$ is maximum-likelihood estimation (MLE), where we maximize the log-likelihood: $\log p(\mathcal{D}|\boldsymbol{\theta})$. This optimization problem can be efficiently solved by applying SG methods such as RMSProp, AdaGrad and Adam. For large problems, these methods are extremely popular, partly due to the simplicity and efficiency of their implementations (see Fig. 1 for Adam's pseudocode).

One of the goals of Bayesian deep learning is to go beyond MLE and estimate the *posterior distribution* of $\boldsymbol{\theta}$ to obtain an uncertainty estimate of the weights. Unfortunately, the computation of the posterior is challenging in deep models. The posterior is obtained by specifying a *prior distribution* $p(\boldsymbol{\theta})$ and then using Bayes' rule: $p(\boldsymbol{\theta}|\mathcal{D}) := p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})/p(\mathcal{D})$. This requires computation of the normalization constant $p(\mathcal{D}) = \int p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}$ which is a very difficult task for DNNs. One source of the difficulty is the size of $\boldsymbol{\theta}$ and $\mathcal{D}$ which are usually very large in deep learning. Another source is the *nonconjugacy* of the likelihood $p(\mathcal{D}_i|\boldsymbol{\theta})$ and the prior $p(\boldsymbol{\theta})$, i.e., the two distributions do not take the same form with respect to $\boldsymbol{\theta}$ (Bishop, 2006). As a result, the product $p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})$ does not take a form with which $p(\mathcal{D})$ can be easily computed. Due to these issues, Bayesian inference in deep learning is computationally challenging.

Variational inference (VI) simplifies the problem by approximating $p(\boldsymbol{\theta}|\mathcal{D})$ with a distribution $q(\boldsymbol{\theta})$ whose normalizing constant is relatively easier to compute. Following previous work (Ranganath et al., 2014; Blundell et al., 2015; Graves, 2011), we choose both $p(\boldsymbol{\theta})$ and $q(\boldsymbol{\theta})$ to be Gaussian distributions with diagonal covariances:

$$p(\boldsymbol{\theta}) := \mathcal{N}(\boldsymbol{\theta}|0, \mathbf{I}/\lambda), \quad q(\boldsymbol{\theta}) := \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2)), \quad (1)$$

where $\lambda \in \mathbb{R}$ is a known precision parameter with $\lambda > 0$, and $\boldsymbol{\mu}, \boldsymbol{\sigma} \in \mathbb{R}^D$ are mean and standard deviation of q. The distribution $q(\boldsymbol{\theta})$ is known as the *Gaussian mean-field variational distribution* and its parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ can be obtained by maximizing the following *variational objective*:

$$\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2) := \sum_{i=1}^{N} \mathbb{E}_q \left[\log p(\mathcal{D}_i|\boldsymbol{\theta})\right] + \mathbb{E}_q \left[\log \frac{p(\boldsymbol{\theta})}{q(\boldsymbol{\theta})}\right]. \quad (2)$$

A straightforward approach used in the previous work (Ranganath et al., 2014; Blundell et al., 2015; Graves, 2011) is to maximize $\mathcal{L}$ by using an SG method, e.g., we can use the following update:

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + \rho_t \widehat{\nabla}_\mu \mathcal{L}_t, \qquad \boldsymbol{\sigma}_{t+1} = \boldsymbol{\sigma}_t + \delta_t \widehat{\nabla}_\sigma \mathcal{L}_t, \quad (3)$$

where t is the iteration number, $\widehat{\nabla}_x \mathcal{L}_t$ denotes an unbiased SG estimate of $\mathcal{L}$ at $\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t^2$ with respect to x, and $\rho_t, \delta_t > 0$ are learning rates which can be adapted using methods such as RMSprop or AdaGrad. These approaches make use of existing codebases for adaptive learning-rate methods to perform VI, which can handle many network architectures and can scale well to large datasets.

Despite this, a direct application of adaptive learning-rate methods for VI may result in algorithms that use more computation and memory than necessary, and also require more implementation effort. Compared to MLE, the memory and computation costs increase because the number of parameters to be optimized is doubled and we now have two vectors $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ to estimate. Using adaptive methods increases this cost further as these methods require storing the scaling vectors that adapt the learning rate for both $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. In addition, using existing codebases require several modifications as they are designed and optimized for MLE. For example, we need to make changes in the computation graph where the objective function is changed to the variational objective and network weights are replaced by random variables. Together, these small issues make VI more difficult to implement and execute than MLE.

The algorithms developed in this paper solve some of these issues and can be implemented within Adam with minimal changes to the code. We derive our algorithm by approximating a natural-gradient method and then using a natural-momentum method. We now describe our method in detail.

3. Approximate Natural-Gradient VI

In this section, we introduce a natural-gradient method to perform VI and then propose several approximations that enable implementation within Adam.

Natural-gradient VI methods exploit the Riemannian geometry of $q(\boldsymbol{\theta})$ by scaling the gradient with the inverse of its Fisher information matrix (FIM). We build upon the natural-gradient method of Khan & Lin (2017), which simplifies the update by avoiding a direct computation of the FIM. The main idea is to use the *expectation parameters of the exponential-family distribution to compute natural gradients in the natural-parameter space*. We provide a brief description of their method in Appendix B.

For Gaussian mean-field VI, the method of Khan & Lin (2017) gives the following update:

$$\text{NGVI}: \boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + \beta_t \ \boldsymbol{\sigma}_{t+1}^2 \circ \left[\widehat{\nabla}_\mu \mathcal{L}_t\right], \quad (4)$$

$$\boldsymbol{\sigma}_{t+1}^{-2} = \boldsymbol{\sigma}_t^{-2} - \ 2\beta_t \ \left[\widehat{\nabla}_{\sigma^2} \mathcal{L}_t\right], \quad (5)$$

where $\beta_t > 0$ is a scalar learning rate and $\mathbf{a} \circ \mathbf{b}$ denotes the element-wise product between vectors $\mathbf{a}$ and $\mathbf{b}$. We refer to this update as natural-gradient variational inference (NGVI). A detailed derivation is given in Appendix C.

The NGVI update differs from (3) in one major aspect: the learning rate β_t in (4) is adapted by the variance $\boldsymbol{\sigma}_{t+1}^2$. This plays a crucial role in reducing the NGVI update to an Adam-like update, as we show the next section. The update requires a constraint $\boldsymbol{\sigma}^2 > 0$ but, as we show in Section 3.2, we can eliminate this constraint using an approximation.

3.1. Variational Online-Newton (VON)

We start by expressing the NGVI update in terms of the MLE objective, so that we can directly compute gradients on the MLE objective using backpropagation. We start by defining the MLE objective (denoted by f) and minibatch stochastic-gradient estimates (denoted by $\hat{\mathbf{g}}$):

$$f(\boldsymbol{\theta}) := \frac{1}{N}\sum_{i=1}^{N} f_i(\boldsymbol{\theta}), \quad \hat{\mathbf{g}}(\boldsymbol{\theta}) := \frac{1}{M}\sum_{i\in\mathcal{M}} \nabla_\theta f_i(\boldsymbol{\theta}), \quad (6)$$

where $f_i(\boldsymbol{\theta}) := -\log p(\mathcal{D}_i|\boldsymbol{\theta})$ is the negative log-likelihood of i'th data example, and the minibatch $\mathcal{M}$ contains M examples chosen uniformly at random. Similarly, we can obtain a minibatch stochastic-approximation of the Hessian which we denote by $\widehat{\nabla}^2_{\theta\theta} f(\boldsymbol{\theta})$.

As we show in Appendix D, the NGVI update can be written in terms of the stochastic gradients and Hessian of f:

$$\text{VON}: \boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t - \beta_t \left(\hat{\mathbf{g}}(\boldsymbol{\theta}_t) + \tilde{\lambda}\boldsymbol{\mu}_t\right)/(\mathbf{s}_{t+1} + \tilde{\lambda}), \quad (7)$$

$$\mathbf{s}_{t+1} = (1-\beta_t)\mathbf{s}_t + \beta_t \operatorname{diag}[\widehat{\nabla}^2_{\theta\theta} f(\boldsymbol{\theta}_t)], \quad (8)$$

where $\mathbf{a}/\mathbf{b}$ is an element-wise division operation between vectors $\mathbf{a}$ and $\mathbf{b}$, and we have approximated the expectation with respect to q using one Monte-Carlo (MC) sample $\boldsymbol{\theta}_t \sim \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t^2)$ with $\boldsymbol{\sigma}_t^2 := 1/[N(\mathbf{s}_t + \tilde{\lambda})]$ and $\tilde{\lambda} := \lambda/N$. The update can be easily modified when multiple samples are used. This update can leverage backpropagation to perform the gradient and Hessian computation. Since the scaling vector $\mathbf{s}_t$ contains an online estimate of the diagonal of the Hessian, we call this the "variational online-Newton" (VON) method. VON is expected to perform as well as NGVI, but does not require the gradients of the variational objective.

The Hessian can be computed by using methods such as automatic-differentiation or the reparameterization trick. However, since f is a non-convex function, the Hessian can be negative which might make $\boldsymbol{\sigma}^2$ negative, in which case the method will break down. One could use a constrained optimization method to solve this issue, but this might be difficult to implement and execute (we discuss this briefly in Appendix D.1). In the next section, we propose a simple fix this problem by using an approximation.

3.2. Variational Online Gauss-Newton (VOGN)

To avoid negative variances in the VON update, we propose to use the Generalized Gauss-Newton (GGN) approximation (Schraudolph, 2002; Martens, 2014; Graves, 2011):

$$\nabla^2_{\theta_j\theta_j} f(\boldsymbol{\theta}) \approx \frac{1}{M}\sum_{i\in\mathcal{M}} \left[\nabla_{\theta_j} f_i(\theta)\right]^2 := \hat{h}_j(\boldsymbol{\theta}), \quad (9)$$

where θ_j is the j'th element of $\boldsymbol{\theta}$. This approximation will always be nonnegative, therefore if the initial $\boldsymbol{\sigma}^2$ at $t = 1$ is positive, it will remain positive in the subsequent iterations. Using this approximation to update $\mathbf{s}_t$ in (8) and denoting the vector of $\hat{h}_j(\boldsymbol{\theta})$ by $\hat{\mathbf{h}}(\boldsymbol{\theta})$, we get,

$$\text{VOGN}: \ \mathbf{s}_{t+1} = (1-\beta_t)\mathbf{s}_t + \beta_t \ \hat{\mathbf{h}}(\boldsymbol{\theta}_t). \quad (10)$$

Using this update in VON, we get the "variational online Gauss-Newton" (VOGN) algorithm.

The GGN approximation is proposed by Graves (2011) for mean-field Gaussian VI to derive a fast gradient-based method (see Eq. (17) in his paper[1]). This approximation is very useful for our natural-gradient method since it eliminates the constraint on $\boldsymbol{\sigma}^2$, giving VOGN an algorithmic advantage over VON.

How good is this approximation? For an MLE problem, the approximation error of the GGN in (9) decreases as the model-fit improves during training (Martens, 2014). For VI, we expect the same however, since $\boldsymbol{\theta}$ are sampled from q, the expectation of the error is unlikely to be zero. Therefore, the solutions found by VOGN will typically differ from those found by VON, but their performances are expected to be similar.

An issue with VOGN is that its implementation is not easy within existing deep-learning codebases. This is because these codebases are optimized to directly compute the sum of the gradients over minibatches, and do not support computation of individual gradients as required in (9). A solution for such computations is discussed by Goodfellow (2015), but this requires additional implementation effort. Instead, we address this issue by using another approximation in the next section.

3.3. Variational RMSprop (Vprop)

To simplify the implementation of VOGN, we propose to approximate the Hessian by the *gradient magnitude* (GM) (Bottou et al., 2016):

$$\nabla^2_{\theta_j\theta_j} f(\boldsymbol{\theta}) \approx \left[\frac{1}{M}\sum_{i\in\mathcal{M}} \nabla_{\theta_j} f_i(\theta)\right]^2 = \left[\hat{g}_j(\boldsymbol{\theta})\right]^2. \quad (11)$$

Compared to the GGN which computes the sum of squared-gradients, this approximation instead computes the square of the sum. This approximation is also used in RMSprop which uses the following update given weights $\boldsymbol{\theta}_t$:

$$\text{RMSprop}: \boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha_t \ \hat{\mathbf{g}}(\boldsymbol{\theta}_t)/(\sqrt{\bar{\mathbf{s}}_{t+1}} + \delta), \quad (12)$$

$$\bar{\mathbf{s}}_{t+1} = (1-\beta_t)\bar{\mathbf{s}}_t + \beta_t \left[\hat{\mathbf{g}}(\boldsymbol{\theta}_t) \circ \hat{\mathbf{g}}(\boldsymbol{\theta}_t)\right], \quad (13)$$

where $\bar{\mathbf{s}}_t$ is the vector that adapts the learning rate and δ is a small positive scalar added to avoid dividing by zero.

[1]There is a discrepancy between Eq. (17) and (12) in Graves (2011), however the text below Eq. (12) mentions relationship to FIM, from which it is clear that the GGN approximation is used.

The update of $\bar{\mathrm{s}}_t$ uses the GM approximation to the Hessian (Bottou et al., 2016). Adam and AdaGrad also use this approximation.

Using the GM approximation and an additional modification in the VON update, we can make the VON update very similar to RMSprop. Our modification involves taking the square-root over s_{t+1} in (7) and then using the GM approximation for the Hessian. We also use different learning rates α_t and β_t to update $\boldsymbol{\mu}$ and s, respectively. The resulting update is very similar to the RMSprop update:

$$\text{Vprop:}\quad \boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t - \alpha_t\, (\hat{\mathrm{g}}(\boldsymbol{\theta}_t) + \tilde{\lambda}\boldsymbol{\mu}_t)/(\sqrt{\mathrm{s}_{t+1}} + \tilde{\lambda}), \\ \mathrm{s}_{t+1} = (1-\beta_t)\mathrm{s}_t + \beta_t\, [\hat{\mathrm{g}}(\boldsymbol{\theta}_t) \circ \hat{\mathrm{g}}(\boldsymbol{\theta}_t)], \quad (14)$$

where $\boldsymbol{\theta}_t \sim \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t^2)$ with $\boldsymbol{\sigma}_t^2 := 1/[N(\mathrm{s}_t + \tilde{\lambda})]$. We call this update “Variational RMSprop” or simply “Vprop”.

The Vprop update resembles RMSprop but with three differences (highlighted in red). First, the gradient in Vprop is evaluated at the weights $\boldsymbol{\theta}_t$ sampled from $\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t^2)$. This is a *weight-perturbation* where the variance $\boldsymbol{\sigma}_t^2$ of the perturbation is obtained from the vector s_t that adapts the learning rate. The variance is also the uncertainty estimates. Therefore, VI can be performed simply by using an RMSprop update with a few simple changes. The second difference between Vprop and RMSprop is that Vprop has an extra term $\tilde{\lambda}\boldsymbol{\mu}_t$ in the update of $\boldsymbol{\mu}_t$ which is due to the Gaussian prior. Finally, the third difference is that the constant δ in RMSprop is replaced by $\tilde{\lambda}$.

3.4. Analysis of the GM approximation

It is clear that the GM approximation might not be the best approximation of the Hessian. Taking square of a sum leads to a sum with M^2 terms which, depending on the correlations between the individual gradients, would either shrink or expand the estimate. The following theorem formalizes this intuition. It states that, given a minibatch of size M, the expectation of the GM approximation is somewhere between the GGN and square of the full-batch gradient.

Theorem 1. *Denote the full-batch gradient with respect to θ_j by $g_j(\boldsymbol{\theta})$ and the corresponding full-batch GGN approximation by $h_j(\boldsymbol{\theta})$. Suppose minibatches $\mathcal{M}$ are sampled from the uniform distribution $p(\mathcal{M})$ over all $\binom{N}{M}$ minibatches, and denote a minibatch gradient by $\hat{g}_j(\boldsymbol{\theta};\mathcal{M})$, then the expected value of the GM approximation is the following,*

$$\mathbb{E}_{p(\mathcal{M})}\left[\hat{g}_j(\boldsymbol{\theta};\mathcal{M})^2\right] = w h_j(\boldsymbol{\theta}) + (1-w)[g_j(\boldsymbol{\theta})]^2, \quad (15)$$

where $w = \frac{1}{M}(N-M)/(N-1)$.

A proof is given in Appendix G. This result clearly shows the bias introduced in the GM approximation and also that the bias increases with the minibatch size. For a minibatch of size $M = 1$, we have $w = 1$ and the GM is an unbiased estimator of the GGN, but when $M = N$ it is purely the magnitude of the gradient and does not contain any second-order information.

Therefore, if our focus is to obtain uncertainty estimates with good accuracy, VOGN with $M = 1$ might be a good choice since it is as easy as Vprop to implement. However, this might require a small learning-rate and converge slowly. Vprop with $M > 1$ will converge fast and is much easier to implement than VOGN with $M > 1$, but might result in slightly worse estimates. Using Vprop with $M = 1$ may not be as good because of the square-root[2] over s_t.

4. Variational Adam (Vadam)

We now propose a *natural-momentum* method which will enable an Adam-like update.

Momentum methods generally take the following form that uses Polyak’s heavy-ball method:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \bar{\alpha}_t \nabla_\theta f_1(\boldsymbol{\theta}_t) + \bar{\gamma}_t(\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}), \quad (16)$$

where f_1 is the function we want to maximize and the last term is the momentum term. We propose a *natural-momentum* version of this algorithm which employs natural-gradients instead of the gradients. We assume q to be an exponential-family distribution with natural-parameter $\boldsymbol{\eta}$. We propose the following natural-momentum method in the natural-parameter space:

$$\boldsymbol{\eta}_{t+1} = \boldsymbol{\eta}_t + \bar{\alpha}_t \tilde{\nabla}_\eta \mathcal{L}_t + \bar{\gamma}_t(\boldsymbol{\eta}_t - \boldsymbol{\eta}_{t-1}), \quad (17)$$

where $\tilde{\nabla}$ denotes the natural-gradients in the natural-parameter space, i.e., the gradient scaled by the Fisher information matrix of $q(\boldsymbol{\theta})$.

We show in Appendix E that, for Gaussian $q(\boldsymbol{\theta})$, we can express the above update as a VON update with momentum[3]:

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t - \tilde{\alpha}_t \left[\frac{1}{\mathrm{s}_{t+1} + \tilde{\lambda}}\right] \circ \left(\nabla_\theta f(\boldsymbol{\theta}_t) + \tilde{\lambda}\boldsymbol{\mu}_t\right) \\ + \tilde{\gamma}_t \left[\frac{\mathrm{s}_t + \tilde{\lambda}}{\mathrm{s}_{t+1} + \tilde{\lambda}}\right] \circ (\boldsymbol{\mu}_t - \boldsymbol{\mu}_{t-1}), \quad (18)$$

$$\mathrm{s}_{t+1} = (1-\tilde{\alpha}_t)\,\mathrm{s}_t + \tilde{\alpha}_t\, \nabla^2_{\theta\theta} f(\boldsymbol{\theta}_t), \quad (19)$$

where $\boldsymbol{\theta}_t \sim \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t^2)$ with $\boldsymbol{\sigma}_t^2 = 1/[N(\mathrm{s}_t + \tilde{\lambda})]$. This update is similar to (17), but here the learning rates are adapted. An attractive feature of this update is that it is very similar to Adam. Specifically the Adam update shown in

[2] Note that the square-root does not affect a fixed point (see Appendix H) but it might still affect the steps taken by the algorithm.

[3] This is not an exact update for (17). We make one approximation in Appendix E to derive it.

Fig. 1 can be expressed as the following adaptive version of (16) as shown in Wilson et al. (2017)[4],

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \tilde{\alpha}_t \left[\frac{1}{\sqrt{\hat{\mathbf{s}}_{t+1}} + \delta} \right] \circ \nabla_\theta f(\boldsymbol{\theta}_t)$$

$$+ \tilde{\gamma}_t \left[\frac{\sqrt{\hat{\mathbf{s}}_t} + \delta}{\sqrt{\hat{\mathbf{s}}_{t+1}} + \delta} \right] \circ (\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}), \quad (20)$$

$$\hat{\mathbf{s}}_{t+1} = (1 - \beta_t)\, \hat{\mathbf{s}}_t + \beta_t \, [\hat{\mathbf{g}}(\boldsymbol{\theta}_t)]^2, \quad (21)$$

where $\tilde{\alpha}_t, \tilde{\gamma}_t$ are appropriately defined in terms of the Adam's learning rate α and γ_1: $\tilde{\alpha}_t := \alpha\,(1-\gamma_1)\,/\,(1-\gamma_1^t)$ and $\tilde{\gamma}_t := \gamma_1 \left(1 - \gamma_1^{t-1}\right)(1-\gamma_1^t)$.

Using a similar procedure as the derivation of Vprop, we can express the update as an Adam-like update, which we call "variational Adam" or simply "Vadam". A pseudocode is given in Fig. 1, where we use learning rate $\gamma_2 := 1 - \beta$. A derivation is in Appendix E.4.

5. Variational AdaGrad (VadaGrad)

Vprop and Vadam perform variational inference, but they can be modified to perform optimization instead of inference. We now derive such an algorithm which turns out to be a variational version of AdaGrad.

We follow Staines & Barber (2013) who consider minimization of black-box functions $F(\boldsymbol{\theta})$ via the *variational optimization*[5] (VO) framework. In this framework, instead of directly minimizing $F(\boldsymbol{\theta})$, we minimize its expectation $\mathbb{E}_q\,[F(\boldsymbol{\theta})]$ under a distribution $q(\boldsymbol{\theta}) := \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ with respect to $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$. The main idea behind VO is that the expectation can be used as a surrogate to the original optimization problem since $\min_\theta F(\boldsymbol{\theta}) \leq \mathbb{E}_q\,[F(\boldsymbol{\theta})]$. The equality is attained when $\boldsymbol{\sigma}^2 \to 0$, i.e., all mass of $\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ is at the mode. The main advantage of VO is that $\mathbb{E}_q\,[F(\boldsymbol{\theta})]$ is differentiable even when F itself is non-differentiable. This way we can use SG optimizers to solve such problems.

Similarly to Vprop, we can derive an algorithm for VO by noting that VO can be seen special case of the VI problem (2) where the KL term is absent and $F(\boldsymbol{\theta})$ is the negative log-likelihood. With this in mind, we define the following variational objective with an additional parameter $\tau \in [0, 1]$:

$$\mathcal{L}_F(\boldsymbol{\mu}, \boldsymbol{\sigma}^2) := -\mathbb{E}_q\,[F(\boldsymbol{\theta})] + \tau \mathbb{E}_q \left[\log \frac{p(\boldsymbol{\theta})}{q(\boldsymbol{\theta})} \right]. \quad (22)$$

The parameter τ allows us to interpolate between inference and optimization. When $\tau = 1$, the objective corresponds to VI with a negative log-likelihood $F(\boldsymbol{\theta})$, and when $\tau = 0$, it corresponds to VO. Similar objectives have been proposed in existing works (Blundell et al., 2015; Higgins et al., 2016) where τ is used to improve convergence.

For twice-differentiable F, we can follow a similar derivation as Section 3, and obtain the following algorithm,

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t - \alpha_t \left(\widehat{\nabla}_\theta F(\boldsymbol{\theta}) + \tau\lambda\boldsymbol{\mu}_t\right)/(\mathbf{s}_{t+1} + \tau\lambda), \quad (23)$$

$$\mathbf{s}_{t+1} = (1 - \tau\beta_t)\mathbf{s}_t + \beta_t \, \widehat{\nabla}^2_{\theta\theta} F(\boldsymbol{\theta}), \quad (24)$$

where $\boldsymbol{\theta}_t \sim \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t^2)$ with $\boldsymbol{\sigma}_t^2 := 1/(\mathbf{s}_t + \tau\lambda)$. This algorithm is identical to the VON algorithm when $\tau = 1$, but when $\tau = 0$, we perform VO with an algorithm which is a diagonal version of the Variational Adaptive-Newton (VAN) algorithm proposed in Khan et al. (2017). By setting value of τ between 0 and 1, we can interpolate between VO and VI. When the function is not differentiable, we can still compute the derivative of $\mathbb{E}_q[F(\boldsymbol{\theta})]$ by using methods such as REINFORCE (Williams, 1992).

When Hessian is difficult to compute, we can employ a GM approximation and take the square-root as we did in Vprop. For $\tau = 0$, the updates turn out to be similar to AdaGrad, which we call "variational AdaGrad" or simply "VadaGrad". The exact updates are given in Appendix F. Unlike Vprop and Vadam, the scaling vector $\mathbf{s}_t$ in VadaGrad is a weighted sum of the past gradient-magnitudes. Therefore, the entries in $\mathbf{s}_t$ never decrease, and the variance estimate of VadaGrad never expands. This implies that it is highly likely that $q(\boldsymbol{\theta})$ will converge to a Dirac delta and therefore arrive at a minimum of F.

6. Results

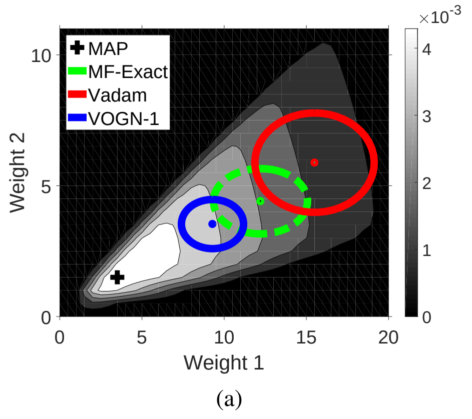
In this section, our goal is to show that the quality of the uncertainty approximations obtained using our algorithms are comparable to existing methods, and computation of uncertainty is scalable. We present results on Bayesian logistic regression for classification, Bayesian neural networks for regression, and deep reinforcement learning. An additional result illustrating avoidance of local-minima using Vadam is in Appendix L. Another result showing benefits of weight-perturbation in Vadam is in Appendix M. The code to reproduce our results is available at **`https://github.com/emtiyaz/vadam`**.

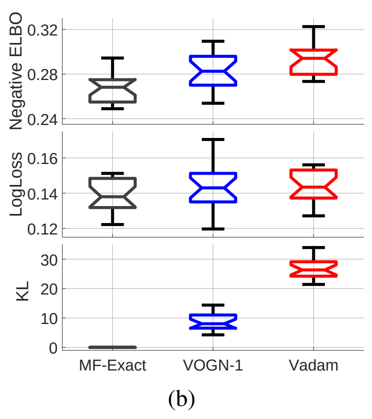
6.1. Uncertainty Estimation in Logistic Regression

In this experiment, we compare the posterior approximations found with our algorithms to the optimal variational approximation that minimizes the variational objective. For Bayesian logistic regression we can compute the optimal mean-field Gaussian approximations using the method described in Marlin et al. (2011) (refer to as 'MF-Exact'), and compare it to the following methods: VOGN with minibatch size $M = 1$ and a momentum term (referred to as 'VOGN-

[4]Wilson et al. (2017) do not use the constant δ, but in Adam a small constant is added for numerical stability.

[5]The exact conditions on F under which VO can be applied are also discussed by Staines & Barber (2013).





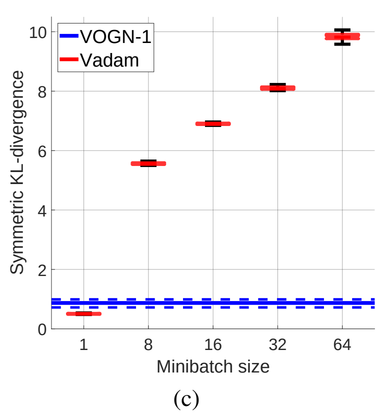


Figure 2. Experiments on Bayesian logistic regression showing (a) posterior approximations on a toy example, (b) performance on 'USPS-3v5' measuring negative ELBO, log-loss, and the symmetric KL divergence of the posterior approximation to MF-Exact, (c) symmetric KL divergence of Vadam for various minibatch sizes on 'Breast-Cancer' compared to VOGN with a minibatch of size 1.

1'), and Vadam with $M \geq 1$ (referred to as 'Vadam'). Since our goal is to compare the accuracy of posterior approximations and not the speed of convergence, we run both the methods for many iterations with a small learning rate to make sure that they converge.

We use three datasets: a toy dataset ($N = 60, D = 2$), USPS-3vs5 ($N = 1781, D = 256$) and Breast-Cancer ($N = 683, D = 10$). Details are in Appendix I.

Fig. 2(a) visualizes the approximations on a two-dimensional toy example from Murphy (2012). The true posterior distribution is shown with the contour in the background. Both, Vadam and VOGN-1 find approximations that are different from MF-Exact, which is clearly due to differences in the type of Hessian approximations they use.

For real datasets, we compare performances using three metrics. First, the negative of the variational objective on the training data (the evidence lower-bound or ELBO), log-loss on the test data, and the symmetric KL distance between MF-Exact and the approximation found by a method. Fig. 2(b) shows the results averaged over 20 random splits of the USPS-3vs5 dataset. ELBO and log-loss are comparable for all methods, although Vadam does slightly worse on ELBO and VOGN-1 has slightly higher variance for log-loss. However, performance on the KL distance clearly shows the difference in the quality of posterior approximations. VOGN-1 performs quite well since it uses an unbiased approximation of the GNN. Vadam does worse due to the bias introduced in the GM approximation with miniatch $M > 1$, as indicated by Theorem 1.

Fig. 2(c) further shows the effect of M where, for each M, we plot results for 20 random initializations on one split of the Breast-Cancer dataset. As we decrease M, Vadam's performance gets better, as expected. For $M = 1$, it closely matches VOGN-1. The results are still different because Vadam does not reduce to VOGN-1, even when $M = 1$ due to the use of the square-root over $\mathbf{s}_t$.

6.2. Uncertainty Estimation in Neural Network

We show results on the standard UCI benchmark. We repeat the experimental setup used in Gal & Ghahramani (2016). Following their work, we use a neural network with one hidden layer, 50 hidden units, and ReLU activation functions. We use the 20 splits of the data provided by Gal & Ghahramani (2016) for training and testing[6]. We use Bayesian optimization to select the prior precision λ and noise precision of the Gaussian likelihood. Further details of the experiments are given in Appendix J.

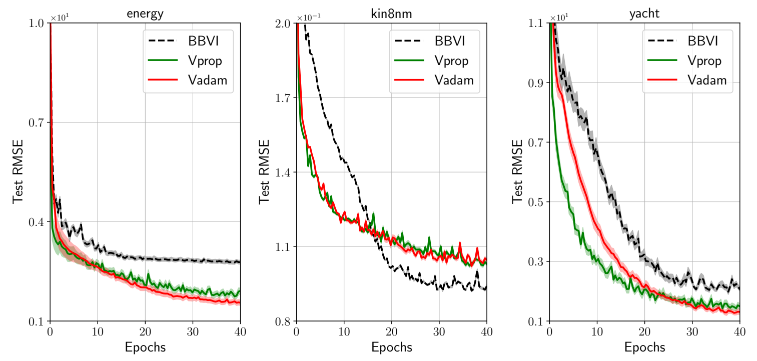
We compare Vadam to MC-Dropout (Gal & Ghahramani, 2016) using the results reported in Gal & Ghahramani (2016). We also compare to an SG method using the reparameterization trick and the Adam optimizer (referred to as 'BBVI'). For a fair comparison, the Adam optimizer is run with the same learning rates as Vadam, although these can be tuned further to get better performance.

Table 1 shows the performance in terms of the test RMSE and the test log-likelihood. The better method out of BBVI and Vadam is shown in boldface found using a paired t-test with p-value > 0.01. Both methods perform comparably, which supports our conclusion, however, MC-Dropout outperforms both the methods. Fig. 3(a) compares the speed of convergence, showing that Vadam and Vprop both give comparable convergence to BBVI. More results on UCI datasets are in Appendix J.

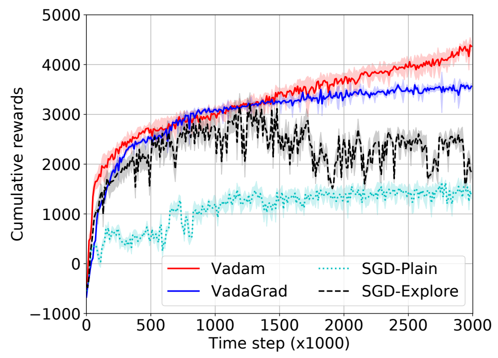
[6]The splits are publicly available from `https://github.com/yaringal/DropoutUncertaintyExps`

Table 1. Performance comparisons for BNN regression. The better method out of BBVI and Vadam is shown in boldface according to a paired t-test with p-value> 0.01. Both methods perform comparably but MC-Dropout outperforms them.

			Test RMSE			Test log-likelihood		
Dataset	**N**	**D**	**MC-Dropout**	**BBVI**	**Vadam**	**MC-Dropout**	**BBVI**	**Vadam**
Boston	506	13	2.97 ± 0.19	**3.58 ± 0.21**	3.93 ± 0.26	-2.46 ± 0.06	**-2.73 ± 0.05**	-2.85 ± 0.07
Concrete	1030	8	5.23 ± 0.12	**6.14 ± 0.13**	6.85 ± 0.09	-3.04 ± 0.02	**-3.24 ± 0.02**	-3.39 ± 0.02
Energy	768	8	1.66 ± 0.04	2.79 ± 0.06	**1.55 ± 0.08**	-1.99 ± 0.02	-2.47 ± 0.02	**-2.15 ± 0.07**
Kin8nm	8192	8	0.10 ± 0.00	**0.09 ± 0.00**	0.10 ± 0.00	0.95 ± 0.01	**0.95 ± 0.01**	0.76 ± 0.00
Naval	11934	16	0.01 ± 0.00	**0.00 ± 0.00**	**0.00 ± 0.00**	3.80 ± 0.01	**4.46 ± 0.03**	**4.72 ± 0.22**
Power	9568	4	4.02 ± 0.04	4.31 ± 0.03	**4.28 ± 0.03**	-2.80 ± 0.01	**-2.88 ± 0.01**	**-2.88 ± 0.01**
Wine	1599	11	0.62 ± 0.01	**0.65 ± 0.01**	0.66 ± 0.01	-0.93 ± 0.01	**-1.00 ± 0.01**	-1.01 ± 0.01
Yacht	308	6	1.11 ± 0.09	2.05 ± 0.06	**1.32 ± 0.10**	-1.55 ± 0.03	-2.41 ± 0.02	**-1.70 ± 0.03**



(a) Vadam and Vprop show comparable convergence and performance to BBVI on Bayesian neural networks.



(b) Vadam and VadaGrad outperform SGD-based methods to perform exploration in deep RL.

Figure 3. Results of regression on UCI and deep reinforcement learning experiments.

6.3. Exploration in Deep Reinforcement Learning

A good exploration strategy is crucial in reinforcement learning (RL) since the data is sequentially collected. We show that weight-perturbation in Vadam improves exploration in RL. Due to space constraints, we only provide a brief summary of our results, and give details in Appendix K.

We consider the deep deterministic policy gradient (DDPG) method for the Half-Cheetah task using a two-layer neural networks with 400 and 300 ReLU hidden units (Lillicrap et al., 2015). We compare Vadam and VadaGrad to two SGD methods, one of which does exploration (referred to as 'SGD-Explore'), and the other does not (referred to as 'SGD-plain'). Figure 3(b) shows the cumulative rewards (higher is better) of each method against training iterations. VadaGrad and Vadam clearly learn faster than both SGD-Plain and SGD-Explore. We also compare the performances against Adam variants of SGD-Plain and SGD-Explore. Their results, given in the Appendix K, show that Vadam and VadaGrad still learn faster, but only in the beginning and Adam based methods can catch up quickly. This suggests that the exploration strategy has a high impact on the early learning performance in the Half-Cheetah task, and the effect of good exploration decreases over time as the agent collect more informative training samples.

7. Discussion

In this paper, we present new VI algorithms which are as simple to implement and execute as algorithms for MLE. We obtain them by using a series of approximations and a natural momentum method for a natural-gradient VI method. The resulting algorithms can be implemented within Adam with minimal changes. Our empirical findings confirm that our proposed algorithms obtain comparable uncertainty estimates to existing VI methods, but require less computational and implementation effort[7].

An interesting direction we hope to pursue in the future is to generalize our natural-gradient approach to other types of approximation, e.g., exponetial-family distributions and their mixtures. We would also like to further explore the application to areas such as RL and stochastic optimization.

[7]We made many new changes in this camera-ready version of the paper. A list of the changes is given in Appendix A.

Acknowledgements

We thank the anonymous reviewers for their feedback. We greatly appreciate many insightful discussions with Aaron Mishkin (UBC) and Frederik Kunstner (EPFL), and also thank them for their help on carrying out experiments and reviewing the manuscript. We would also like to thank Roger Grosse and David Duvenaud from the University of Toronto for useful discussions. We would like to thank Zuozhu Liu (SUTD, Singapore) for his help with the experiment on deep RL and logistic regression. Finally, we are thankful for the RAIDEN computing system at the RIKEN Center for AI Project, which we extensively used for our experiments.

References

Amari, S. *Information geometry and its applications*. Springer, 2016.

Anderson, J. R. and Peterson, C. A mean field theory learning algorithm for neural networks. *Complex Systems*, 1:995–1019, 1987.

Balan, A. K., Rathod, V., Murphy, K. P., and Welling, M. Bayesian dark knowledge. In *Advances in Neural Information Processing Systems*, pp. 3438–3446, 2015.

Barber, D. and Bishop, C. M. Ensemble learning in Bayesian neural networks. *Generalization in Neural Networks and Machine Learning*, 168:215–238, 1998.

Bishop, C. M. *Pattern Recognition and Machine Learning*. 2006.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural networks. In *International Conference on Machine Learning*, pp. 1613–1622, 2015.

Bottou, L., Curtis, F. E., and Nocedal, J. Optimization methods for large-scale machine learning. *arXiv preprint arXiv:1606.04838*, 2016.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. OpenAI Gym, 2016.

Chaudhari, P., Choromanska, A., Soatto, S., LeCun, Y., Baldassi, C., Borgs, C., Chayes, J. T., Sagun, L., and Zecchina, R. Entropy-sgd: Biasing gradient descent into wide valleys. In *International Conference on Learning Representations*, 2016.

Cochran, W. G. *Sampling Techniques, 3rd Edition.* John Wiley, 1977. ISBN 0-471-16240-X.

Denker, J. S. and Lecun, Y. Transforming neural-net output levels to probability distributions. In *Advances in Neural Information Processing Systems*, pp. 853–859, 1991.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

Fortunato, M., Azar, M. G., Piot, B., Menick, J., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., et al. Noisy networks for exploration. In *International Conference on Learning Representations*, 2018.

Gal, Y. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

Gal, Y. and Ghahramani, Z. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pp. 1050–1059, 2016.

Goodfellow, I. Efficient Per-Example Gradient Computations. *ArXiv e-prints*, October 2015.

Graves, A. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, pp. 2348–2356, 2011.

Hasenclever, L., Webb, S., Lienart, T., Vollmer, S., Lakshminarayanan, B., Blundell, C., and Teh, Y. W. Distributed Bayesian learning with stochastic natural gradient expectation propagation and the posterior server. *Journal of Machine Learning Research*, 18:1–37, 2017.

Hazan, E., Levy, K. Y., and Shalev-Shwartz, S. On graduated optimization for stochastic non-convex problems. In *International Conference on Machine Learning*, pp. 1833–1841, 2016.

Hensman, J., Rattray, M., and Lawrence, N. D. Fast variational inference in the conjugate exponential family. In *Advances in Neural Information Processing Systems*, pp. 2888–2896, 2012.

Hernandez-Lobato, J. M. and Adams, R. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *International Conference on Machine Learning*, pp. 1861–1869, 2015.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2016.

Hinton, G. E. and Van Camp, D. Keeping the neural networks simple by minimizing the description length of the weights. In *Annual Conference on Computational Learning Theory*, pp. 5–13, 1993.

Khan, M. E. and Lin, W. Conjugate-computation variational inference: converting variational inference in non-conjugate models to inferences in conjugate models. In *International Conference on Artificial Intelligence and Statistics*, pp. 878–887, 2017.

Khan, M. E., Babanezhad, R., Lin, W., Schmidt, M., and Sugiyama, M. Faster stochastic variational inference using proximal-gradient methods with general divergence functions. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2016.

Khan, M. E., Lin, W., Tangkaratt, V., Liu, Z., and Nielsen, D. Variational Adaptive-Newton Method for Explorative Learning. *ArXiv e-prints*, November 2017.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Leordeanu, M. and Hebert, M. Smoothing-based optimization. In *Computer Vision and Pattern Recognition*, pp. 1–8, 2008.

Li, C., Chen, C., Carlson, D. E., and Carin, L. Preconditioned stochastic gradient langevin dynamics for deep neural networks. In *AAAI Conference on Artificial Intelligence*, pp. 4–10, 2016.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015.

Louizos, C. and Welling, M. Structured and efficient variational deep learning with matrix gaussian posteriors. In *International Conference on Machine Learning*, pp. 1708–1716, 2016.

MacKay, D. J. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

Mandt, S., Hoffman, M. D., and Blei, D. M. Stochastic gradient descent as approximate Bayesian inference. *Journal of Machine Learning Research*, 18:1–35, 2017.

Marlin, B., Khan, M., and Murphy, K. Piecewise bounds for estimating Bernoulli-logistic latent Gaussian models. In *International Conference on Machine Learning*, 2011.

Martens, J. New insights and perspectives on the natural gradient method. *arXiv preprint arXiv:1412.1193*, 2014.

Mobahi, H. and Fisher III, J. W. A theoretical analysis of optimization by Gaussian continuation. In *AAAI Conference on Artificial Intelligence*, pp. 1205–1211, 2015.

Murphy, K. P. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN 0262018020, 9780262018029.

Neal, R. M. *Bayesian learning for neural networks*. PhD thesis, University of Toronto, 1995.

Opper, M. and Archambeau, C. The variational Gaussian approximation revisited. *Neural Computation*, 21(3):786–792, 2009.

Plappert, M., Houthooft, R., Dhariwal, P., Sidor, S., Chen, R. Y., Chen, X., Asfour, T., Abbeel, P., and Andrychowicz, M. Parameter space noise for exploration. In *International Conference on Learning Representations*, 2018.

Ranganath, R., Gerrish, S., and Blei, D. M. Black box variational inference. In *International conference on Artificial Intelligence and Statistics*, pp. 814–822, 2014.

Raskutti, G. and Mukherjee, S. The information geometry of mirror descent. *IEEE Transactions on Information Theory*, 61 (3):1451–1457, 2015.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pp. 1278–1286, 2014.

Ritter, H., Botev, A., and Barber, D. A scalable laplace approximation for neural networks. In *International Conference on Learning Representations*, 2018.

Robert, C. P. and Casella, G. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005. ISBN 0387212396.

Rückstieß, T., Sehnke, F., Schaul, T., Wierstra, D., Sun, Y., and Schmidhuber, J. Exploring parameter space in reinforcement learning. *Paladyn*, 1(1):14–24, 2010. doi: 10.2478/s13230-010-0002-4.

Salimans, T., Knowles, D. A., et al. Fixed-form variational posterior approximation through stochastic linear regression. *Bayesian Analysis*, 8(4):837–882, 2013.

Saul, L. K., Jaakkola, T., and Jordan, M. I. Mean field theory for sigmoid belief networks. *Journal of Artificial Intelligence Research*, 4:61–76, 1996.

Schraudolph, N. N. Fast curvature matrix-vector products for second-order gradient descent. *Neural computation*, 14(7):1723–1738, 2002.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. A. Deterministic policy gradient algorithms. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pp. 387–395, 2014.

Staines, J. and Barber, D. Optimization by variational bounding. In *European Symposium on Artificial Neural Networks*, 2013.

Sun, S., Chen, C., and Carin, L. Learning structured weight uncertainty in Bayesian neural networks. In *International Conference on Artificial Intelligence and Statistics*, pp. 1283–1292, 2017.

Tieleman, T. and Hinton, G. Lecture 6.5-RMSprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning 4*, 2012.

Wierstra, D., Schaul, T., Glasmachers, T., Sun, Y., Peters, J., and Schmidhuber, J. Natural evolution strategies. *Journal of Machine Learning Research*, 15(1):949–980, 2014.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8 (3-4):229–256, 1992.

Wilson, A. C., Roelofs, R., Stern, M., Srebro, N., and Recht, B. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems*, pp. 4151–4161, 2017.

Zhang, G., Sun, S., Duvenaud, D. K., and Grosse, R. B. Noisy natural gradient as variational inference. *arXiv preprint arXiv:1712.02390*, 2018.

Zhou, E. and Hu, J. Gradient-based adaptive stochastic search for non-differentiable optimization. *IEEE Transactions on Automatic Control*, 59(7):1818–1832, 2014.