Vadam: Fast and Scalable Variational Inference by Perturbing Adam

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

Abstract

Uncertainty computation in deep learning is essential, but variational inference methods are difficult to implement within existing deep-learning code-bases. We propose *Vadam*, a new method for mean-field variational inference that can be implemented within Adam by simply perturbing the network weights during gradient evaluations. The variance of the perturbation represents the weight uncertainty and is automatically obtained from the vector that adapts the learning rate. We derive Vadam using a novel natural-momentum method for mirror descent algorithms. Using Vadam, we are able to "read-off" uncertainty estimates while training with Adam. Vadam requires lower memory, computation, and implementation efforts than existing methods for VI, and our empirical experiments confirm these findings. Our experiments also suggest that the perturbation used in Vadam may be useful for exploration to avoid local sharp minima.

1. Introduction

Deep learning has become widespread in applications such as robotics and medical analysis (LeCun et al., 2015), but these are the applications where one must reason about the uncertainty of a model's prediction. For example, a physician, who relies on a deep-learning diagnostic system to make decisions, must have access to the uncertainty of predictions to make reliable decision. Without any estimate of uncertainty, it is unreasonable to rely on such predictions, because there is always a possibility that the system is just guessing at random.

Exact computation of uncertainty for large deep models can be a daunting task (Gal, 2016). Traditional Bayesian approaches for uncertainty estimation in such models, such as MCMC, converge slowly and can sometimes even be infeasible for large problems. Modern methods such as variational inference (VI), on the other hand, compute approximations to the uncertainty by using optimization algorithms, and show much faster convergence. Such stochastic-gradient (SG) optimization algorithms, with Adam and RMSprop being prominent examples, have been successfully used to optimize the mean-field VI objective in deep learning (Ranganath et al., 2014; Blundell et al., 2015; Salimans et al., 2013; Graves, 2011).

However, an issue with these methods is that their implementation requires major changes to existing deep-learning code bases. For example, implementing the black-box VI (BBVI) method of Ranganath et al. (2014) for Bayesian neural networks requires the network weights to be replaced with random variables. The optimization objective also needs to be changed to the variational lower bound. This increases the memory requirement since each random variable is now represented with two scalar quantities (mean and variance). Computation is also increased due to extra gradient calculations for the variance parameter. These issues make it difficult to incorporate VI methods within the standard deep-learning code-bases, which inhibits their application to real-world problems.

Beside these practical difficulties, there are theoretical concerns as well. Adaptation of step-sizes within an optimizer such as Adam is tricky for VI. The mean and variance are two fundamentally different quantities with different units and their step-sizes might require different types of tuning. Natural-gradient VI methods (Sato, 2001; Honkela et al., 2011; Hoffman et al., 2013; Khan & Lin, 2017) are promising solutions to such theoretical issues but their implementation is also difficult for deep models.

Despite these issues, recent work (Mandt et al., 2017) suggests that it might be possible to "read-off" uncertainty approximations from a standard run of an SG method. With this motivation, we propose a new method, called *Vadam*, that can compute uncertainty approximations by slightly modifying the Adam optimizer. These minor changes are illustrated in Fig. 1. The main difference is that in Vadam the network weights are perturbed during the gradient computation. The variances of perturbations represent the weight uncertainty, and these variances are automatically obtained using the vector that adapts the learning rate in Adam. This

[*]Work done during an internship in RIKEN. [1]RIKEN, Tokyo, Japan [2]SUTD, Singapore [3]University of Oxford, UK [4]University of Edinburgh, UK. Correspondence to: Mohammad Emtiyaz Khan <emtiyaz.khan@riken.jp>.

```
Adam
 1: while not converged do
 2:     $\boldsymbol{\theta} \leftarrow \boldsymbol{\mu}$
 3:     $\mathbf{g} \leftarrow \nabla f(\boldsymbol{\theta})$
 4:     $\mathbf{m} \leftarrow \gamma\, \mathbf{m} + (1-\gamma)\, \mathbf{g}$
 5:     $\mathbf{s} \leftarrow \beta\, \mathbf{s} + (1-\beta)\, (\mathbf{g} \circ \mathbf{g})$
 6:     $\hat{\mathbf{m}} = \mathbf{m}/(1-\gamma^t)$
 7:     $\hat{\mathbf{s}} = \mathbf{s}/(1-\beta^t)$
 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:     $\boldsymbol{\theta} \leftarrow \boldsymbol{\mu} + \boldsymbol{\epsilon}/\sqrt{\mathbf{s}+\lambda}$ where $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}|0, \mathbf{I})$.
 3:     $\mathbf{g} \leftarrow \nabla_\theta f(\boldsymbol{\theta}) + \lambda \boldsymbol{\mu}$
 4:     $\mathbf{m} \leftarrow \gamma\, \mathbf{m} + (1-\gamma)\, \mathbf{g}$
 5:     $\mathbf{s} \leftarrow \beta\, \mathbf{s} + (1-\beta)\, (\mathbf{g} \circ \mathbf{g})$
 6:     $\hat{\mathbf{m}} = \mathbf{m}/(1-\gamma^t)$
 7:     $\hat{\mathbf{s}} = \mathbf{s}/(1-\beta^t)$
 8:     $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \alpha\, \hat{\mathbf{m}}/(\sqrt{\hat{\mathbf{s}}} + \lambda)$
 9:     $t \leftarrow t+1$
10: end while
```

Figure 1. Comparison of Adam (left) with our method Vadam (right). The main differences (highlighted in red) are in line 2, where we perturb the weights when the gradients are evaluated, and in line 3 where a contribution from the prior is added. Without lines 6 and 7 and $\gamma = 0$, Vadam reduces to an Vprop. For $\gamma = 1$, we get Vprop with momentum. Further implementation details are in Appendix G.

enables our implementation to lie entirely within the Adam optimizer, and avoids the need to define new objectives or variables. It also simplifies the tuning of the algorithm since we can utilize the tricks used to make Adam work well. As a result, Vadam requires lower memory, lower computation, and lower implementation efforts compared to the existing methods for VI.

From the theoretical standpoint, Vadam can be derived as an extension of a VI method called "conjugate-computation VI" (CVI) (Khan & Lin, 2017). CVI is a natural-gradient method that works well for moderate-size problems. Unfortunately, it is not suitable for large scale deep-learning problems because it requires the computation of Hessians. We solve this issue by proposing a Hessian approximation method that can be trivially performed within existing deep-learning implementations. This approximation results in an approximated VI method called *Vprop* whose update resembles that of RMSprop. To further improve upon Vprop, we propose *Vadam*, an approximated VI method that incorporates a novel *natural momentum* term to the update. The update of Vadam resembles that of Adam and differs only in one major aspect – the network weights are perturbed during a gradient evaluation in Vadam. These extra steps can be trivially implemented within an existing Adam implementation.

We show that perturbation in Vadam facilitates exploration which is useful during optimization of a complex landscape. An illustration of this is shown in Fig. 6 in Appendix J. With this motivation, we propose an extension of Vadam that can be used to perform optimization of general functions, extending Vadam's application far beyond VI. The resulting algorithm resembles AdaGrad, and we refer to it as *VadaGrad*. VadaGrad enables us to interpolate between optimization and inference, thereby combining the strengths of the two approaches.

Application to real-world data shows that our method converges much faster than existing VI methods, and gives comparable results to the state-of-the-art. We also present several examples demonstrating the application of our methods for exploration in deep reinforcement learning (similar to Fortunato et al. (2017); Plappert et al. (2017)), to avoid local sharp minima (similar to Chaudhari et al. (2016)), and to improve the "marginal value" of adaptive-gradient methods (similar to Wilson et al. (2017)).

2. Background

We consider a probabilistic modeling using N data points $\mathcal{D} = \{\mathcal{D}_i\}_{i=1}^N \overset{\text{i.i.d.}}{\sim} p(\mathcal{D})$. Maximum likelihood estimation (MLE) is a simple approach to model such data. In MLE, a likelihood function $p(\mathcal{D}|\boldsymbol{\theta})$ parameterized by a deep neural network with weights $\boldsymbol{\theta} \in \mathbb{R}^D$ is learned by minimizing $f(\boldsymbol{\theta}) := -\log p(\mathcal{D}|\boldsymbol{\theta})$. This optimization problem can be efficiently solved by applying stochastic-gradient (SG) methods such as RMSprop (Tieleman & Hinton, 2012), AdaGrad (Duchi et al., 2011), and Adam (Kingma & Ba, 2014). For large data and complex models, this procedure is widely popular partly due to the simplicity and efficiency of their implementation, as illustrated by the pseudo-code of Adam for solving MLE in Fig. 1.

Unfortunately, MLE does not provide model uncertainty. In contrast, Bayesian approaches estimate the model's uncertainty by learning the *posterior distribution* $p(\boldsymbol{\theta}|\mathcal{D})$. However, this requires computing the marginal likelihood $p(\mathcal{D})$ which is typically analytically intractable and hard to estimate when N and D is large. Variational inference (VI) methods simplify the uncertainty estimation problem by approximating the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ using a *variational distribution* $q(\boldsymbol{\theta})$ and reformulating the inference problem as an optimization problem.

2.1. Related Works in Gaussian Mean-Field Variational Inference

In this paper, we consider the Gaussian mean-field VI where we assume a Gaussian prior distribution $p(\boldsymbol{\theta}) := \mathcal{N}(\boldsymbol{\theta}|0, \mathbf{I}/\lambda)$ with $\lambda > 0$ and a Gaussian variational distribution $q(\boldsymbol{\theta}) := \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2))$ with mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}^2$. The mean and variance of the Gaussian variational distribution can be obtained by maximizing the lower-bound to the log-marginal likelihood:

$$\max_{\boldsymbol{\mu},\boldsymbol{\sigma}} \; -\mathbb{E}_q[f(\boldsymbol{\theta})] - \mathbb{D}_{KL}[q(\boldsymbol{\theta}) \,\|\, p(\boldsymbol{\theta})] := \mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2). \quad (1)$$

The variational lower bound in (1) can be approximated using K Monte-Carlo (MC) samples $\{\boldsymbol{\theta}_k\}_{k=1}^K \sim q(\boldsymbol{\theta})$. This allows us to compute the gradient of $\mathcal{L}$ and apply SG methods such as Adam to optimize the lower-bound. This approach was pursued by recent VI methods for deep neural networks (Ranganath et al., 2014; Blundell et al., 2015; Graves, 2011). However, implementations of these VI methods differ significantly from Adam's implementation for MLE. For example, the black-box VI (BBVI) method (Ranganath et al., 2014), which is one of the simplest and popular approaches, employs the following SG update:

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + \rho_t \widehat{\nabla}_\mu \mathcal{L}_t, \qquad \boldsymbol{\sigma}_{t+1} = \boldsymbol{\sigma}_t + \rho_t \widehat{\nabla}_\sigma \mathcal{L}_t, \quad (2)$$

where $\rho_t > 0$ is a step-size at iteration t, $\widehat{\nabla}$ denotes an unbiased SG estimate. This update is simple and general, but its implementation differs significantly from the implementation of adaptive-gradient updates for MLE.

The first difference is that, compared to MLE, the number of parameters to be optimized in BBVI is doubled ($\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$). If Adam is applied to adapt the step size of BBVI, the memory requirements would be tripled since we need to maintain the scaling and momentum vectors for both $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. Further, step-sizes for $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ typically require different types of tuning since they are two fundamentally different quantities with different units.

Another difference is that BBVI requires gradients with respect to $\boldsymbol{\sigma}$ which often has high variance (Wierstra et al., 2008). This issue was partly solved by Graves (2011) who proposed an approximation based on gradient-magnitude. Blundell et al. (2015) also propose a simple variance reduction technique with the same purpose. These solutions do fix the problem to some extent, but the updates for all these methods still differ significantly from that of adaptive-gradient methods such as Adam.

Lastly, BBVI separately updates $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ and ignores the fact that they are parameters of a distribution and therefore coupled together. It is known that a better approach to optimize parameters of a distribution is to employ natural gradient methods (Amari, 1998). Natural gradient methods has been previously applied to VI by Sato (2001); Honkela et al. (2011); Hoffman et al. (2013); Khan & Lin (2017). However, these methods typically requires high computation effort which make them not directly applicable to deep neural network models.

As discussed above, existing VI methods which optimizes the lower-bound by SG methods requires significantly different implementation than that of SG methods for MLE. In the next section, we show that by performing approximations to a natural gradient update of the lower-bound, we obtain updates which resemble perturbed version of RMSprop and Adam for MLE. This allows our methods to perform VI by simply perturbing RMSprop's and Adam's update for MLE.

3. Vprop: VI using RMSprop

Throughout the paper, all operations between vectors are performed element-wise.

In this section, we derive a method for VI that can be implemented within RMSprop's implementation. We build upon the CVI approach of Khan & Lin (2017). The CVI update exploits the equivalence of the following two updates that correspond to natural-gradients descent and mirror-descent algorithm respectively (Raskutti & Mukherjee, 2015):

$$\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t + \beta_t \left[\mathcal{I}(\boldsymbol{\lambda}_t)\right]^{-1} \nabla_\lambda \mathcal{L}(\boldsymbol{\lambda}_t) \quad (3)$$

$$\mathbf{m}_{t+1} = \arg\min_{\boldsymbol{m}} \; \langle \mathbf{m}, -\nabla_m \mathcal{L}_t \rangle + \frac{1}{\beta_t} \mathbb{D}_{KL}[q \,\|\, q_t], \quad (4)$$

where $\boldsymbol{\lambda}$ and $\mathbf{m}$ are the natural and mean parameter of q respectively, $\mathcal{I}(\boldsymbol{\lambda})$ is the Fisher information matrix (FIM), β_t is the step-size at iteration t, $\mathbb{D}_{KL}[q \,\|\, q_t]$ is the Kullback-Liebler divergence between current distribution q and previous distribution q_t at iteration t. A derivation of the equivalence is given in the Appendix A.

The mirror-descent update is more convenient to implement since the derivative w.r.t. $\mathbf{m}$ are easier to compute than computing FIM. As shown in Appendix B, the mirror descent formulation gives us the following result:

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + \beta_t \; \boldsymbol{\sigma}_{t+1}^2 \left[\nabla_\mu \mathcal{L}_t\right] \quad (5)$$

$$\boldsymbol{\sigma}_{t+1}^{-2} = \boldsymbol{\sigma}_t^{-2} - \; 2\beta_t \; \left[\nabla_\Sigma \mathcal{L}_t\right] \quad (6)$$

This is the update derived by Khan & Lin (2017) for VI. The above update differs from the update of (2) in two aspects. First, here we update the precision $\boldsymbol{\sigma}^{-2}$ while BBVI updates $\boldsymbol{\sigma}$, even though both updates use the gradient with respect to $\boldsymbol{\Sigma}$. Second, the step-size for $\boldsymbol{\mu}_{t+1}$ are *adaptive* here since β_t is scaled by the variance $\boldsymbol{\sigma}_{t+1}^2$. These two properties are useful to build connections to RMSprop.

As we show in the Appendix C, the update (5)-(6) can be expressed as an online Newton-like adapts the scaling vector using the diagonal of the Hessian:

$$\begin{aligned}\text{VON: } \boldsymbol{\mu}_{t+1} &= \boldsymbol{\mu}_t - \alpha_t \ \left(\nabla_\theta f(\boldsymbol{\theta}_t) + \lambda \boldsymbol{\mu}_t\right) / \left(\mathbf{s}_{t+1} + \lambda\right), \\ \mathbf{s}_{t+1} &= (1-\beta_t)\mathbf{s}_t + \beta_t \ \nabla^2_{\theta\theta} f(\boldsymbol{\theta}_t), \end{aligned} \tag{7}$$

where $\boldsymbol{\theta}_t \sim \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t^2)$ with $\boldsymbol{\sigma}_t^2 = 1/(\mathbf{s}_t + \lambda)$. We refer to this update as the variational online-Newton (VON) update.

An issue with this update is that it requires computation of the Hessian $\nabla^2_{\theta\theta} f(\boldsymbol{\theta})$. We propose to approximate the Hessian by the gradient-magnitude which is readily available in the RMSprop implementation (Bottou et al., 2016):

$$\nabla^2_{\theta_d\theta_d} f(\boldsymbol{\theta}) \approx \left(\sum_{i=1}^N \nabla_{\theta_d} f_i(\boldsymbol{\theta})\right)^2 = \hat{g}_d(\boldsymbol{\theta})^2, \tag{8}$$

where $\hat{g}_d(\boldsymbol{\theta}) \approx \nabla_{\theta_d} f(\boldsymbol{\theta})$ is an SG estimate of the gradients. Denoting the vector of $\hat{g}_d(\boldsymbol{\theta})$ by $\hat{\mathbf{g}}(\boldsymbol{\theta})$, we can rewrite the VON update with an approximated Hessian as

$$\begin{aligned}\boldsymbol{\mu}_{t+1} &= \boldsymbol{\mu}_t - \alpha_t \ [\hat{\mathbf{g}}(\boldsymbol{\theta}_t) + \lambda\boldsymbol{\mu}_t]/(\mathbf{s}_{t+1} + \lambda), \\ \mathbf{s}_{t+1} &= (1-\beta_t)\mathbf{s}_t + \beta_t \ [\hat{\mathbf{g}}(\boldsymbol{\theta}_t)]^2, \end{aligned} \tag{9}$$

This update differs from the adaptive-gradient methods in one major aspect – it does not take the square root of the scaling vector $\mathbf{s} + \lambda$. As we show in the Appendix E, using a square-root does not change the fixed point, therefore we can modify the update to get the following update, which we refer to as Vprop due to its resemblance to RMSprop.

$$\begin{aligned}\text{Vprop: } \boldsymbol{\mu}_{t+1} &= \boldsymbol{\mu}_t - \alpha_t \ [\hat{\mathbf{g}}(\boldsymbol{\theta}_t) + \lambda\boldsymbol{\mu}_t]/(\sqrt{\mathbf{s}_{t+1}} + \lambda), \\ \mathbf{s}_{t+1} &= (1-\beta_t)\mathbf{s}_t + \beta_t \ [\hat{\mathbf{g}}(\boldsymbol{\theta}_t)]^2, \end{aligned} \tag{10}$$

This update closely resembles the update of RMSprop:

$$\begin{aligned}\text{RMSprop:} \quad \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t - \alpha_t \ \hat{\mathbf{g}}(\boldsymbol{\theta}_t)/(\sqrt{\mathbf{s}_{t+1}} + \delta), \\ \mathbf{s}_{t+1} &= (1-\beta_t)\mathbf{s}_t + \beta_t \ [\hat{\mathbf{g}}(\boldsymbol{\theta}_t)]^2. \end{aligned} \tag{11}$$

The two updates differ in three ways. First, the gradient in Vprop is evaluated at the perturbed weights $\boldsymbol{\theta}_t \sim \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_t, 1/(\mathbf{s}_t + \lambda))$. Second, the mean update in Vprop-1 has an extra term $\lambda\boldsymbol{\mu}_t$ which arises from the Gaussian prior. Finally, the constant δ in RMSprop is replaced by the prior precision λ.

The interpretation of δ as the regularization parameter is quite useful in practice, since it enables us to derive variational versions of RMSprop that could avoid overfitting. For example, if an MLE estimate found by RMSprop with a particular δ value gives us a good test RMSE, we might want to run Vprop with $\lambda = \delta$. This way we can bring the benefits obtained by a Bayesian approach to the MLE problem. We can then try to increase the value of δ to check if Vprop further reduces the error. Our experiments in Section 6.3 demonstrates this on real data.

We derived Vprop by using gradient-magnitude to approximate the Hessian, but there are other choices of approximation that can be applied as well. For instance, we can apply the reparameterization trick (Kingma & Ba, 2014) as shown in Appendix D to obtain an unbiased approximation:

$$\mathbb{E}_q\left[\nabla^2_{\theta\theta} f(\boldsymbol{\theta})\right] \approx \hat{\mathbf{g}}(\boldsymbol{\theta})\left(\boldsymbol{\epsilon}/\boldsymbol{\sigma}\right). \tag{12}$$

However, this approximation is not suitable for large-scale problems due to its high variance. Another choice of an approximation is the Generalized Gauss-Newton (GGN):

$$\nabla^2_{\theta\theta} f(\boldsymbol{\theta}) \approx \sum_{i=1}^N \left[\nabla_\theta f_i(\boldsymbol{\theta})\right]^2. \tag{13}$$

However, computing a sum of squared-gradient requires reimplementation of gradient computation in standard deep-learning code-bases. As shown in Fig. 2 in our experiments, these alternative approximations may give better approximations than our method, but as shown in Fig. 2 our approximation leads to faster convergence which leads to better overall performance.

4. Vadam: Vprop with Natural Momentum

Adam uses a momentum term and bias-corrections, which typically shows good performance on many deep-learning problems. In this section, we show that we can do the same for Vprop by modifying the mirror-descent update.

Momentum methods generally take the following form expressed as a Polyak's heavy-ball method:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \bar{\alpha}_t \nabla_\theta f(\boldsymbol{\theta}_t) + \bar{\gamma}_t(\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}) \tag{14}$$

where the last term is the momentum term. Adam differs from these methods in the fact that it adapts the last two terms in the above update. Specifically the Adam update shown in Fig. 1 can be expressed as the following *scaled* version of the Polyak's heavy ball[1] method,

$$\begin{aligned}\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t &- \tilde{\alpha}_t \left[\frac{1}{\sqrt{\hat{\mathbf{s}}_{t+1}} + \delta}\right] \nabla_\theta f(\boldsymbol{\theta}_t) \\ &+ \tilde{\gamma}_t \left[\frac{\sqrt{\hat{\mathbf{s}}_t} + \delta}{\sqrt{\hat{\mathbf{s}}_{t+1}} + \delta}\right] (\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}), \end{aligned} \tag{15}$$

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

We will now show that, by replacing the Euclidean distance in the Polyak's heavy ball method by a KL divergence, we can get an Adam like update for Vprop where the step-sizes

[1](Wilson et al., 2017) do not add the constant δ in $\sqrt{\mathbf{s}_t}$ but in Adam a small constant is added.

are adapted. Our derivation proceeds as follows: first, we add a momentum term to the mirror-descent update of (4), then simplify it to bring it to the form (15), and finally define $\tilde{\alpha}_t$ and $\tilde{\gamma}_t$ accordingly to implement the update using the implementation shown in Fig. 1.

We modify the mirror descent update in (4) by adding a momentum term defined using the KL divergence as follows:

$$\mathbf{m}_{t+1} = \arg\min_{\boldsymbol{m}} \langle \mathbf{m}, -\nabla_m \mathcal{L}_t \rangle + \frac{1}{\beta_t} \mathbb{D}_{KL}[q \,\|\, q_t] - \frac{\alpha_t}{\beta_t} \mathbb{D}_{KL}[q \,\|\, q_{t-1}] \tag{16}$$

Using the KL divergence implies that the momentum term exploits the information geometry defined by q. This is very similar to the idea of natural-gradients, therefore we call this a *natural momentum* method.

The solution of this optimization problem can be obtained similarly to the mirror-descent update of (4), and we provide a detailed derivation in Appendix F. We make approximations similar to the ones used while deriving Vprop. The derivation requires one extra approximation where we assume that the covariance of q_t is close to covariance of q_{t-1}. If the step-size $\tilde{\alpha}_t$ is small, then this approximation is reasonable. After these approximations, we get the following update for the natural-momentum:

$$\begin{aligned} \mathbf{s}_{t+1} &= (1 - \tilde{\alpha}_t)\, \mathbf{s}_t + \tilde{\alpha}_t \, \left[\hat{\mathbf{g}}(\boldsymbol{\theta}_t)\right]^2, \qquad (17)\\ \boldsymbol{\mu}_{t+1} &= \boldsymbol{\mu}_t - \tilde{\alpha}_t \left[\frac{1}{\sqrt{\mathbf{s}_{t+1}} + \lambda}\right] (\nabla_\theta f(\boldsymbol{\theta}_t) + \lambda \boldsymbol{\mu}_t) \\ &\quad + \tilde{\gamma}_t \left[\frac{\sqrt{\mathbf{s}_t} + \lambda}{\sqrt{\mathbf{s}_{t+1}} + \lambda}\right] (\boldsymbol{\mu}_t - \boldsymbol{\mu}_{t-1}), \end{aligned}$$

where $\boldsymbol{\theta}_t \sim \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t^2)$ with $\boldsymbol{\sigma}_t^{-2} = (\mathbf{s}_t + \lambda)$. This is very similar to the update (15) of Adam. We can therefore implement this update by using Adam's update shown in the left pseudo-code in Fig. 1. This will mean that we do not use the same step-size $\tilde{\alpha}_t$ for $\mathbf{s}_t$ and $\boldsymbol{\mu}_t$, rather choose the step-sizes according to the Adam update.

5. VadaGrad for Variational Optimization

Vprop and Vadam perform variational inference, but they can be modified to perform optimization instead of inference. We derive such an algorithm in this section. Our algorithm is a perturbed version of AdaGrad, and we call it VadaGrad. The perturbation property of VadaGrad is useful for optimization problems where the landscape is complex, e.g., when it contains local minima. VadaGrad can interpolate between inference and optimization problems, thereby combining the strengths of the two approaches.

We consider a generic optimization problem: $\min_\theta f(\boldsymbol{\theta})$ where $\boldsymbol{\theta} \in \mathbb{R}^D$. To solve such optimization problems, Staines & Barber (2012) propose Variational Optimization (VO) which optimizes the expectation of f under a Gaussian distribution:

$$\min_{\mu, \sigma^2} \mathbb{E}_{\mathcal{N}(\theta|\mu,\sigma^2)} \left[f(\boldsymbol{\theta})\right] := \mathcal{L}_{\text{VO}}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2). \tag{18}$$

The conditions on f in which VO can be applied are also discussed in Staines & Barber (2012). Under these conditions, VO can be use as a surrogate to the generic optimization problem since $\min_\theta f(\boldsymbol{\theta}) \le \mathbb{E}_{\mathcal{N}(\theta|\mu,\sigma^2)} \left[f(\boldsymbol{\theta})\right]$ and the bound is tight when $\boldsymbol{\sigma}^2 \to 0$, i.e., all mass of $\mathcal{N}(\theta|\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ is at the mode. The main advantage of VO is that $\mathcal{L}_{\text{VO}}$ can be differentiable even when f itself is non-differentiable. This allows us to use stochastic gradient optimizer to solve difficult problems, e.g., those involving non-differentiable problems such as ℓ_1 penalty.

We observe that the VO problem of (18) is similar the VI problem in (1), but the VO problem does not have the KL divergence term. This suggests that VI methods can also be extended to solve VO as well. We extend Vadam for VO by considering the following variational lower-bound with an additional parameter $\tau \in [0, 1]$:

$$\mathcal{L}_{\text{VIVO}}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2) := \mathbb{E}_q \left[f(\boldsymbol{\theta})\right] + \tau \mathbb{D}_{KL}[q(\boldsymbol{\theta}) \,\|\, p(\boldsymbol{\theta})]. \tag{19}$$

The parameter τ allows us to interpolate between inference (VI) and optimization (VO). When $\tau = 1$, the objective corresponds to VI, and when $\tau = 0$, it corresponds to VO. These type of problems are considered in existing works (Blundell et al., 2015; Higgins et al., 2016) where annealing τ has shown to improve convergence. We expect our method to be useful for those problems as well.

To obtain the VadaGrad update, we can apply the derivation of Vprop to the above variational lower-bound. The resulting update is the following:

$$\begin{aligned} \boldsymbol{\mu}_{t+1} &= \boldsymbol{\mu}_t - \alpha_t \, [\hat{\mathbf{g}}(\boldsymbol{\theta}_t) + \tau\lambda\boldsymbol{\mu}_t] / (\sqrt{\mathbf{s}_{t+1}} + \tau\lambda), \\ \mathbf{s}_{t+1} &= (1 - \tau\beta_t)\mathbf{s}_t + \beta_t \, [\hat{\mathbf{g}}(\boldsymbol{\theta}_t)]^2, \end{aligned} \tag{20}$$

By setting $\tau = 0$, we can see obtain VadaGrad:

$$\begin{aligned} \boldsymbol{\mu}_{t+1} &= \boldsymbol{\mu}_t - \alpha_t \, [\hat{\mathbf{g}}(\boldsymbol{\theta}_t)] / \sqrt{\mathbf{s}_{t+1}}, \\ \mathbf{s}_{t+1} &= \mathbf{s}_t + \beta_t \, [\hat{\mathbf{g}}(\boldsymbol{\theta}_t)]^2. \end{aligned} \tag{21}$$

Unlike Vprop and Vadam, the scaling vector $\mathbf{s}_t$ here is monotonically non-decreasing, and it is non-increasing only when the gradient of $f(\boldsymbol{\theta})$ is zero everywhere. This property guarantees that we approach the an optima of f as t increases, therefore in the limit we converge to $\min_\theta f(\boldsymbol{\theta})$. Similarly to AdaGrad, the increase in $\mathbf{s}_t$ might be too aggressive which might lead to very small step-size for the mean update. To speed-up convergence and avoid this issue in practice, we recommend decaying τ from 1 to 0 to perform VI via Vadam in the beginning to encourage exploration. Then, as τ approach 0, we perform VO via VadaGrad where the method eventually puts all mass on the mode of the variational distribution and gives us a point estimate of a local optima of $f(\boldsymbol{\theta})$. We call this approach "Vadam To VadaGrad".

6. Results

6.1. Illustrative Example on Logistic Regression

The left figure in Fig. 2 compares the quality of uncertainty obtained using Vadam on a 2D logistic regression example. We use a toy example given in Murphy (2012). The size of the data is $N = 30$ and the data is generated from a mixture of two Gaussian (details are in the book). The contour in the background shows the exact posterior distribution. We compare to three methods: a method that uses exact Hessian (VI-Hessian), a method that uses the approximations (12) (VI-Reparam), and a method that uses (13) (VI-GGN). We observe in the figure that Vadam's approximation is slightly worse than the other methods. It is unclear whether this will result in worse predictive performance. As we show in the next experiment that our method performs better because it converges more rapidly than the other methods.

6.2. VI for Regression on UCI Datasets

We compare the performance of Vadam for VI on deep regression models. We show that our method converges much faster than BBVI, while giving comparable performance to other baseline. Therefore, despite being easy to implement, our method still performs as well as existing methods.

We repeat the experimental setup[2] in Gal & Ghahramani (2016). Following their work, we use neural networks with one hidden layer and 50 hidden units with ReLU activation functions. The details of experiments and hyperparameter settings are given in Appendix H. For BBVI, we used 200 samples because, for smaller number of samples, BBVI performed worse. For our methods, Vadam-10, Vprop-10 (M), and Vprop-10 use 10 Monto-Carlo (MC) samples, while Vprop uses 1 MC sample.

The right plot in Fig. 2 compares several versions of our methods to BBVI on three datasets. The results on eight UCI datasets are provided in the Appendix H. We observe that all of our methods converge faster than BBVI on all datasets except 'Yatch'. Despite using much fewer samples, our method beat BBVI which is surprising. The MC approximation in BBVI has high variance which is the reason behind its poor performance. Our methods use the gradient-magnitude approximation which has much lower variance. This is perhaps the reason behind their fast convergence. Among our methods, Vadam-10 and Vprop-10 with momentum perform the best. We see that Vprop-10 performs slightly worse because it does not use momentum. Finally, Vprop-1 which uses 1 MC samples is the slowest to converge, but still beats BBVI.

[2]We use the setup publicly available from https://github.com/yaringal/DropoutUncertaintyExps

Table 1. Average test performance in RMSE for UCI datasets where Vadam performs comparably to existing methods.

Dataset	PBP	VIG	Vadam-10	Dropout
Boston	3.01 (0.04)	4.32 (0.06)	2.99 (0.21)	2.97 (0.19)
Concrete	5.67 (0.02)	7.19 (0.03)	5.27 (0.10)	5.23 (0.12)
Energy	1.80 (0.01)	2.65 (0.02)	2.03 (0.08)	1.66 (0.04)
Kin8nm	0.10 (0.00)	0.10 (0.00)	0.08 (0.00)	0.10 (0.00)
Naval	0.01 (0.00)	0.01 (0.00)	0.00 (0.00)	0.01 (0.00)
Power	4.12 (0.01)	4.33 (0.01)	4.16 (0.03)	4.02 (0.04)
Wine	0.64 (0.00)	0.65 (0.00)	0.63 (0.01)	0.62 (0.01)
Yacht	1.02 (0.01)	6.89 (0.15)	3.67 (0.15)	1.11 (0.09)

Table 1 compare the performance with Dropout (Gal & Ghahramani, 2016), Probabilistic Back-Propagation (PBP) (Hernandez-Lobato & Adams, 2015), and the method of (Graves, 2011) (VIG). For PBP and VIG, we use the test RMSE reported in (Gal & Ghahramani, 2016). For all datasets, except Yacht, the overall performance of Vadam is comparable to these methods (although Dropout performs slightly better on some datasets). Therefore, despite being easy to implement, our methods gives comparable performance with the existing methods.

6.3. Reducing Overfitting with Vadam

In this section, we show that Vadam can reduce overfitting for learning deep models on small datasets. Our results show that, while Adam and RMSprop, reduce training error to zero, their predictions on the test set gets overconfident as the training progress. To measure this overfitting, we compare test and train log-loss. We expect the overfitting to reduce as the size of the data increases, therefore we compare methods on increasing data sizes.

We compare Vadam to Adam and RMSprop. As discussed in Section 3, the parameter δ in Adam and RMSprop can be interpreted as the prior precision in Vadam. In our experiments, we use the standard setting of δ. The value of δ and other details such as the network architectures, dataset details, and learning rates are given in Table 3 in the Appendix. For Vadam, we use 1 MC sample.

The results for a1a, a3a, a5a and a9a are shown in Fig. 3(a). From left to right the data size increases. Top row shows the train log-loss while the bottom row shows the test log-loss. We report 10 random runs of each method. We clearly see that for small dataset, e.g., a1a, RMSprop and Adam overfit. This overfitting disappears as the data size increases. Surprisingly, the train and test loss for Vadam are always comparable. This behavior does not arise due to Bayesian

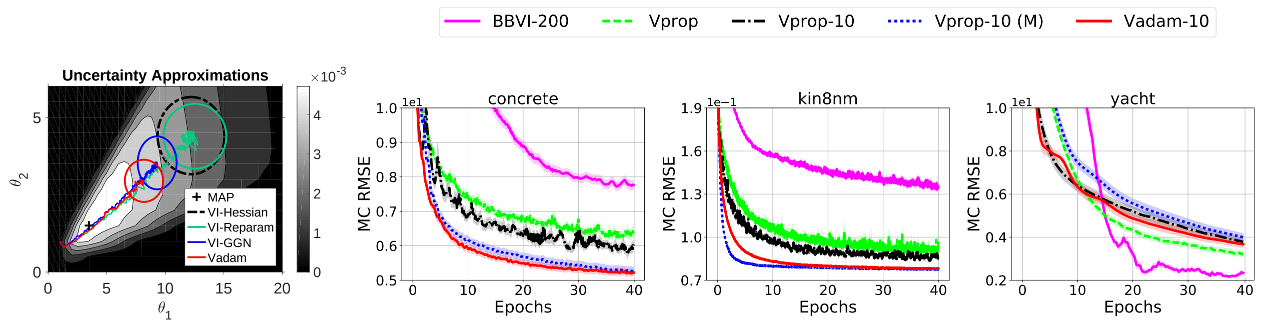
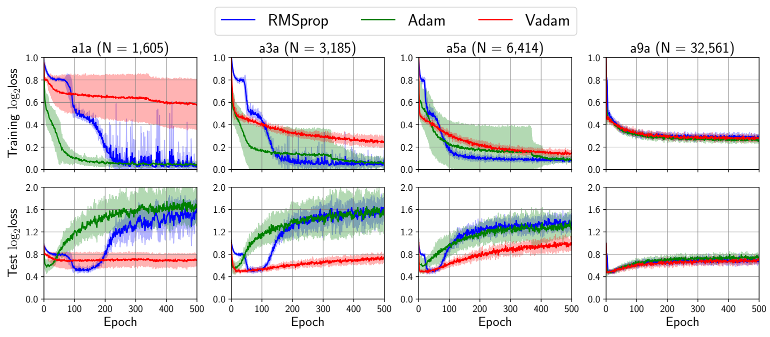
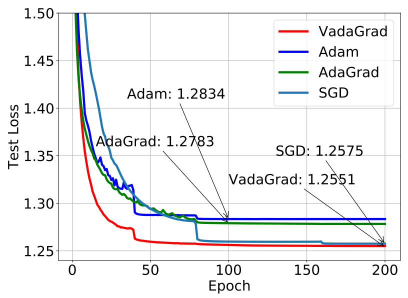


Figure 2. (Left) Illustration of Vadam for 2D logistic regression (Right) Test RMSE for the UCI experiments showing that Vadam-10 converges the fastest while BBVI is the slowest (even after using 200 samples). We also see that methods with momentum (Vprop-10 M and Vadam) work better than methods without them.





(a) Vadam reduces overfitting for smaller datasets. We show train and test log-loss for an increasing data size (from left to right). We see that for smaller datasizes, RMSprop and Adam overfit, while Vadam does not. As the dataset size increases, RMSprop and Adam behave like Vadam.

(b) Results for LSTM on War&Peace dataset. We repeat the results shown in Fig. 2 (b) of Wilson et al. (2017) and show that VadaGrad does not suffer from the issues pointed in that paper, and it performs comparable to SGD.

Figure 3.

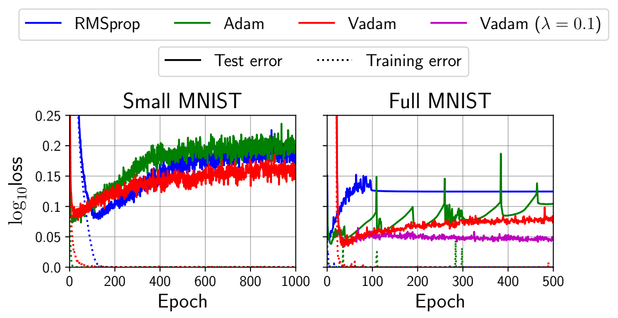


Figure 4. Vadam reduces overfitting on small and full MNIST dataset.

averaging because the prior precision is too low to be of any use. A likely explanation is that, due to perturbation, Vadam is able to avoid minima that overfit.

We repeated these results for MNIST, where we made a small version of the dataset by using the test set as the train set. This gave us a small MNIST dataset with 10,000 examples. Results in Fig. 4 show similar trends, i.e., Vadam overfits less than RMSprop and Adam. For large MNIST, we tried to increase λ to 0.1 and found that the test log-loss stabilizes at a good value. This is shown in purple in Fig. 4.

Finally, Table 2 shows the final test loglosses and test accuracies for all algorithms, where we see that Vadam give lower test log-loss while keeping the test accuracy low.

The results for MNIST can be found in Fig. 4. The results for a1a, a3a, a5a and a9a can be found in Fig. 3(a). Table 2 shows the final test loglosses and test accuracies for the various datasets and algorithms.

From these results we can see that Vadam seems to reduce the tendency to overfit. Vadam gives lower test logloss and higher training logloss compared to the other algorithms even when the prior precision λ is set to be very small. When λ is increased, even better generalization performance was observed. From the experiments on a1a, a3a, a5a and a9a we can see that Vadams tendency to reduce overfitting is even

Table 2. Average test accuracies and test loglosses for the last 50 epochs. For a1a, a3a, a5a and a9a, the $\log_2$loss is presented, while for small and full MNIST, $\log_{10}$loss is presented.

Dataset	Accuracy			Logloss		
	RMSprop	Adam	Vadam	RMSprop	Adam	Vadam
a1a	**0.8098**	0.8071	0.7943	0.4527	0.4999	**0.2091**
a3a	0.8033	0.8027	**0.8140**	0.4688	0.4691	**0.2168**
a5a	0.8081	0.8087	**0.8097**	0.4078	0.3904	**0.2943**
a9a	0.8202	0.8218	**0.8219**	0.2073	0.2229	**0.2058**
Small MNIST	0.9558	**0.9566**	0.9454	0.1858	0.2017	**0.1601**
Full MNIST	0.9821	**0.9831**	0.9821	0.1246	0.1079	**0.0782**

more prominent for the smaller datasets. For the relatively large dataset a9a, the difference between the methods are marginal, while for the smaller datasets, the improved generalization performance can be seen clearly. We conjecture that Vadam might have a tendency to converge to different, flatter minima which can lead to improved generalization performance.

6.4. Experiment for "Marginal Value of Adaptive-Gradient Methods"

Recently, (Wilson et al., 2017) show some examples where adaptive-methods generalize worse than SGD. We repeated their experiments to see whether perturbation in VadaGrad improves the generalization performance of AdaGrad. Fig. 3(b) shows the results of the experiment about the character-level language model (shown in Fig. 2b in (Wilson et al., 2017). We see that VadaGrad achieves the same performance SGD unlike AdaGrad and Adam. We also repeated the CIFAR-10 experiment discussed in the paper, and found that the improvement using VadaGrad was minor. We believe that the training tricks such as batch normalization, batch flip, and dropout play an important role for CIFAR dataset and that is why we did not see an improvement by using VadaGrad.

6.5. Variational Optimization in Reinforcement Learning

In this section, we demonstrate the usefulness of VadaGrad by applying it to perform parameter-based exploration in reinforcement learning (RL). Parameter-based exploration RL can be formulated as a variational optimization problem in (18) where the function f is the negative of the expected cumulative rewards defined as

$$f(\boldsymbol{\theta}) = \mathbb{E}_{p(x)}\left[-Q(\mathbf{x}, \pi(\mathbf{x}; \boldsymbol{\theta}))\right], \qquad (22)$$

where $Q(\mathbf{x}, \mathbf{a})$ is the cumulative reward, $\mathbf{x}$ is a state variable, $\mathbf{a}$ is an action variable, and $\pi(\mathbf{x}; \boldsymbol{\theta})$ is a policy function with parameter $\boldsymbol{\theta}$. For more details of parameter-based exploration RL, we refer to (Rückstieß et al., 2010).

We compare the performance of VadaGrad against that of V-SGD, and plain SGD which directly optimizes (22). We consider the Half-Cheetah task from OpenAI gym (Brockman et al., 2016) where a cheetah robot needs to move forward to obtain high rewards. For all methods, we use deep neural networks with 2 hidden layer where the first and second layer have 400 and 300 hidden units, respectively. Fig. 7 in Appendix K shows the cumulative rewards against training iteration. VadaGrad significantly outperforms both V-SGD and plain SGD. V-SGD is able to find a good policy at first but its performance degenerate over time due to the instability of V-SGD. Plain SGD learns only a sub-optimal policy because it does not perform any exploration.

7. Discussion

We propose a new method called Vadam that enables implementation of VI within Adam. Our method connects inference and optimization algorithms. The connections made in our method are useful in establishing general connections between the two distinct fields. VI methods optimize in the space of variational distribution q and are fundamentally different from continuous optimization methods which optimize in the space of parameter $\boldsymbol{\theta}$. Some other recent works have suggested similar types of results (Gal, 2015; 2016; Mandt et al., 2017). Yet, the results in this paper have clearly shown that there is a close connection between the two fields, and it is possible to design VI methods by only slightly modifying existing optimization methods. Such results are very encouraging and further promote the use of optimization methods as a tool to design scalable algorithms for Bayesian deep learning.

On a practical side, our results show that VI can be implemented entirely within the optimization procedure. This is important since it allows a plug-and-play of deterministic models such as neural networks for uncertainty estimation. By simply running the existing code-base of an optimizer, we surprisingly approximate the optimum of the variational lower bound.

There are two limitations within our VI methods. First, our methods only perform Gaussian mean-field VI. However, we expect that our methods can be further extended to non-Gaussian VI as well since the CVI method can perform VI with an exponential family. Second, our methods use gradient-magnitude to approximate the Hessian because it can be trivially implement based on the existing optimizers. However, better approximation methods for the Hessian may also be implemented based on existing optimizers as well. Such approximation methods will be investigated in our future work.

References

Amari, Shun-Ichi. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

Bhatnagar, Shalabh. Adaptive Newton-based multivariate smoothed functional algorithms for simulation optimization. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 18(1):2, 2007.

Blundell, Charles, Cornebise, Julien, Kavukcuoglu, Koray, and Wierstra, Daan. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

Bottou, Léon, Curtis, Frank E, and Nocedal, Jorge. Optimization methods for large-scale machine learning. *arXiv preprint arXiv:1606.04838*, 2016.

Brockman, Greg, Cheung, Vicki, Pettersson, Ludwig, Schneider, Jonas, Schulman, John, Tang, Jie, and Zaremba, Wojciech. Openai gym, 2016.

Chaudhari, Pratik, Choromanska, Anna, Soatto, Stefano, and LeCun, Yann. Entropy-sgd: Biasing gradient descent into wide valleys. *arXiv preprint arXiv:1611.01838*, 2016.

Duchi, John, Hazan, Elad, and Singer, Yoram. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

Fortunato, Meire, Azar, Mohammad Gheshlaghi, Piot, Bilal, Menick, Jacob, Osband, Ian, Graves, Alex, Mnih, Vlad, Munos, Remi, Hassabis, Demis, Pietquin, Olivier, et al. Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*, 2017.

Gal, Yarin. Rapid prototyping of probabilistic models: Emerging challenges in variational inference. In *Advances in Approximate Bayesian Inference workshop, NIPS*, 2015.

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

Gal, Yarin and Ghahramani, Zoubin. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Balcan, Maria Florina and Weinberger, Kilian Q. (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR. URL http://proceedings.mlr.press/v48/gal16.html.

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

Hernandez-Lobato, Jose Miguel and Adams, Ryan. Probabilistic backpropagation for scalable learning of bayesian neural networks. In Bach, Francis and Blei, David (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1861–1869, Lille, France, 07–09 Jul 2015. PMLR. URL http://proceedings.mlr.press/v37/hernandez-lobatoc15.html.

Higgins, Irina, Matthey, Loic, Pal, Arka, Burgess, Christopher, Glorot, Xavier, Botvinick, Matthew, Mohamed, Shakir, and Lerchner, Alexander. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

Hoffman, Matthew D, Blei, David M, Wang, Chong, and Paisley, John. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

Honkela, A., Raiko, T., Kuusela, M., Tornio, M., and Karhunen, J. Approximate Riemannian conjugate gradient learning for fixed-form variational Bayes. *Journal of Machine Learning Research*, 11:3235–3268, 2011.

Khan, Mohammad Emtiyaz and Lin, Wu. Conjugate-computation variational inference: Converting variational inference in non-conjugate models to inferences in conjugate models. *arXiv preprint arXiv:1703.04265*, 2017.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

LeCun, Yann, Bengio, Yoshua, and Hinton, Geoffrey. Deep learning. *nature*, 521(7553):436, 2015.

Mandt, Stephan, Hoffman, Matthew D, and Blei, David M. Stochastic gradient descent as approximate bayesian inference. *arXiv preprint arXiv:1704.04289*, 2017.

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

Opper, M. and Archambeau, C. The Variational Gaussian Approximation Revisited. *Neural Computation*, 21(3): 786–792, 2009.

Plappert, Matthias, Houthooft, Rein, Dhariwal, Prafulla, Sidor, Szymon, Chen, Richard Y, Chen, Xi, Asfour, Tamim, Abbeel, Pieter, and Andrychowicz, Marcin. Parameter space noise for exploration. *arXiv preprint arXiv:1706.01905*, 2017.

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

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

Rezende, Danilo Jimenez, Mohamed, Shakir, and Wierstra, Daan. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

Rückstieß, Thomas, Sehnke, Frank, Schaul, Tom, Wierstra, Daan, Sun, Yi, and Schmidhuber, Jürgen. Exploring parameter space in reinforcement learning. *Paladyn*, 1(1):14–24, 2010. doi: 10.2478/s13230-010-0002-4. URL https://doi.org/10.2478/s13230-010-0002-4.

Salimans, T., Ho, J., Chen, X., Sidor, S., and Sutskever, I. Evolution Strategies as a Scalable Alternative to Reinforcement Learning. *ArXiv e-prints*, March 2017.

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

Sato, Masa-Aki. Online model selection based on the variational Bayes. *Neural Computation*, 13(7):1649–1681, 2001.

Spall, James C. Adaptive stochastic approximation by the simultaneous perturbation method. *IEEE transactions on automatic control*, 45(10):1839–1853, 2000.

Staines, J. and Barber, D. Variational Optimization. *ArXiv e-prints*, December 2012.

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

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

Wierstra, Daan, Schaul, Tom, Peters, Jan, and Schmidhuber, Juergen. Natural evolution strategies. In *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, pp. 3381–3387. IEEE, 2008.

Wilson, Ashia C, Roelofs, Rebecca, Stern, Mitchell, Srebro, Nati, and Recht, Benjamin. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems*, pp. 4151–4161, 2017.

A. Equivalent of Natural-Gradient descent and Mirror Descent

The method of (Khan & Lin, 2017) derives the natural gradient updates using the mean parameterization of the Gaussian distribution $\mathbf{m} := \{\boldsymbol{\mu}, \boldsymbol{\mu}\boldsymbol{\mu}^T + \boldsymbol{\Sigma}\}$ (see Section 3.2 and 3.4.1 of (Wainwright & Jordan, 2008) on the basics of exponential family and mean-parameterization respectively). Using the result of (Raskutti & Mukherjee, 2015), they show that the following mirror-descent update is equivalent to the natural-gradient update of (3):

$$\mathbf{m}_{t+1} = \arg\min_{\boldsymbol{m}} \; \langle \mathbf{m}, \nabla_m \mathcal{L}_t \rangle + \frac{1}{\beta_t} \mathbb{D}_{KL}[q \,\|\, q_t], \tag{23}$$

where $q := q(\boldsymbol{\theta}|\mathbf{m})$, $q_t := q(\boldsymbol{\theta}|\mathbf{m}_t)$, and $\mathbb{D}_{KL}[q \,\|\, q_t] = \mathbb{E}_q[\log(q/q_t)]$ denotes the KL divergence. Here, $\mathcal{L}$ refers to the negative lower bound in Eq. 1. The equivalence is easy to establish by taking a derivative of the above objective:

$$\nabla_m \mathcal{L}_t + \frac{1}{\beta_t}(\boldsymbol{\lambda}_{t+1} - \boldsymbol{\lambda}_t) = 0 \tag{24}$$

where we used the fact that the derivative of KL divergence w.r.t. $\mathbf{m}$ is equal to the difference of natural-parameters. Noting that the gradient $\nabla_m \mathcal{L}_t$ is in fact equal to $[\mathcal{I}(\boldsymbol{\lambda}_t)]^{-1} \nabla_\lambda \mathcal{L}(\boldsymbol{\lambda}_t)$, we have the equivalence.

B. Mirror-Descent Update

We derive these updates for a full covariance matrix case. We denote the covariance matrix by $\boldsymbol{\Sigma}$. As shown in the main text, the derivative of the mirror-descent objective is the following,

$$\nabla_m \mathcal{L}_t + \frac{1}{\beta_t}(\boldsymbol{\lambda}_{t+1} - \boldsymbol{\lambda}_t) = 0 \tag{25}$$

This gives us the following update,

$$\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t - \beta_t \nabla_m \mathcal{L}_t. \tag{26}$$

Recalling that the mean parameters of a Gaussian $q(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ are $\mathbf{m}^{(1)} = \boldsymbol{\mu}$ and $\mathbf{M}^{(2)} = \boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T$ and using the chain rule, we can express the gradient $\nabla_m \mathcal{L}_t$ in terms of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$,

$$\nabla_{m^{(1)}} \mathcal{L} = \nabla_\mu \mathcal{L} - 2\left[\nabla_\Sigma \mathcal{L}\right] \boldsymbol{\mu} \tag{27}$$

$$\nabla_{M^{(2)}} \mathcal{L} = \nabla_\Sigma \mathcal{L}. \tag{28}$$

Finally, recalling that the natural parameters of a Gaussian $q(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ are $\boldsymbol{\lambda}^{(1)} = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$ and $\boldsymbol{\lambda}^{(2)} = -\frac{1}{2}\boldsymbol{\Sigma}^{-1}$, we can rewrite the updates in terms of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$,

$$\boldsymbol{\Sigma}_{t+1}^{-1} = \boldsymbol{\Sigma}_t^{-1} + 2\beta_t \left[\nabla_\Sigma \mathcal{L}_t\right] \tag{29}$$

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\Sigma}_{t+1}\left[\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\mu}_t - \beta_t\left(\nabla_\mu \mathcal{L}_t - 2\left[\nabla_\Sigma \mathcal{L}_t\right]\boldsymbol{\mu}_t\right)\right] \tag{30}$$

$$= \boldsymbol{\Sigma}_{t+1}\left(\boldsymbol{\Sigma}_t^{-1} + 2\beta_t\left[\nabla_\Sigma \mathcal{L}_t\right]\right)\boldsymbol{\mu}_t - \beta_t \boldsymbol{\Sigma}_{t+1}\left[\nabla_\mu \mathcal{L}_t\right] \tag{31}$$

$$= \boldsymbol{\mu}_t - \beta_t \boldsymbol{\Sigma}_{t+1}\left[\nabla_\mu \mathcal{L}_t\right]. \tag{32}$$

Using $\boldsymbol{\Sigma} = \text{diag}(\boldsymbol{\sigma}^2)$ we get the desired update.

C. Variational Online-Newton method (VON)

We now show that the update (5)-(6) correspond to an online Newton method. For our derivation we use a full covariance matrix denoted by $\boldsymbol{\Sigma}$. The mean-field version can be obtain by using a diagonal matrix instead of $\boldsymbol{\Sigma}$.

The connection to the online-Newton can be shown by using the Bonnet's and Price's theorems (Opper & Archambeau, 2009; Rezende et al., 2014). We use these theorems to express the gradients of the expectation with respect to $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ in terms of the gradient and Hessian of the function, as shown below:

$$\nabla_\mu \mathbb{E}_q\left[f(\boldsymbol{\theta})\right] = \mathbb{E}_q\left[\nabla_\theta f(\boldsymbol{\theta})\right], \tag{33}$$

$$\nabla_\Sigma \mathbb{E}_q\left[f(\boldsymbol{\theta})\right] = \tfrac{1}{2}\mathbb{E}_q\left[\nabla_{\theta\theta}^2 f(\boldsymbol{\theta})\right]. \tag{34}$$

Using these, we can rewrite the gradients as follows:

$$\nabla_\mu \mathcal{L} = -\nabla_\mu \mathbb{E}_q\left[-f(\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) - \log q(\boldsymbol{\theta})\right] \tag{35}$$

$$= \mathbb{E}_q\left[\nabla_\theta f(\boldsymbol{\theta})\right] + \lambda\boldsymbol{\mu}, \tag{36}$$

$$\nabla_\Sigma \mathcal{L} = \tfrac{1}{2}\mathbb{E}_q\left[\nabla^2_{\theta\theta} f(\boldsymbol{\theta})\right] + \tfrac{1}{2}\lambda\mathbf{I} - \tfrac{1}{2}\boldsymbol{\Sigma}^{-1}, \tag{37}$$

By substituting these into (5) and (6) and replacing the expectations with a sample $\boldsymbol{\theta}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$, we get the following updates:

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t - \beta_t\ \boldsymbol{\Sigma}_{t+1}\left[\nabla_\theta f(\boldsymbol{\theta}_t) + \lambda\boldsymbol{\mu}_t\right] \tag{38}$$

$$\boldsymbol{\Sigma}^{-1}_{t+1} = (1-\beta_t)\boldsymbol{\Sigma}^{-1}_t +\ \beta_t\ \left[\nabla^2_{\theta\theta} f(\boldsymbol{\theta}_t) + \lambda\mathbf{I}\right] \tag{39}$$

By defining a matrix $\mathbf{S}_t := \boldsymbol{\Sigma}^{-1}_t - \lambda\mathbf{I}$, we get the following:

$$\text{VON: } \boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t - \beta_t\ \left(\mathbf{S}_{t+1} + \lambda\mathbf{I}\right)^{-1}\left(\nabla_\theta f(\boldsymbol{\theta}_t) + \lambda\boldsymbol{\mu}_t\right),$$

$$\mathbf{S}_{t+1} = (1-\beta_t)\mathbf{S}_t + \beta_t\ \nabla^2_{\theta\theta} f(\boldsymbol{\theta}_t), \tag{40}$$

where $\boldsymbol{\theta}_t \sim \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$ with $\boldsymbol{\Sigma}_t = (\mathbf{S}_t + \lambda\mathbf{I})^{-1}$. We refer to this update as the Variational Online-Newton (VON) method because it resembles a regularized version of online Newton's method where the scaling matrix is estimated online using the Hessians.

In the above update, the step-size for the update of mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ are the same. In practice, it is better to use two separate step-size. We denote the step-size for $\boldsymbol{\mu}$ by α_t and use β_t for $\boldsymbol{\Sigma}$.

D. Hessian Approximation Using the Reparameterization Trick

We can derive a Hessian approximation as follows:

$$\mathbb{E}_q\left[\nabla^2_{\theta\theta} f(\boldsymbol{\theta})\right] = 2\nabla_{\sigma^2}\mathbb{E}_q[f(\boldsymbol{\theta})], \tag{41}$$

$$= 2\nabla_{\sigma^2}\mathbb{E}_{\mathcal{N}(\epsilon|0,I)}\left[f(\boldsymbol{\mu} + \boldsymbol{\sigma}\boldsymbol{\epsilon})\right], \tag{42}$$

$$= 2\mathbb{E}_{\mathcal{N}(\epsilon|0,1)}\left[\nabla_{\sigma^2} f(\boldsymbol{\mu} + \boldsymbol{\sigma}\boldsymbol{\epsilon})\right], \tag{43}$$

$$= \mathbb{E}_{\mathcal{N}(\epsilon|0,1)}\left[\nabla_\theta f(\boldsymbol{\theta})\boldsymbol{\epsilon}/\boldsymbol{\sigma}\right], \tag{44}$$

$$\approx \hat{\mathbf{g}}(\boldsymbol{\theta})\,(\boldsymbol{\epsilon}/\boldsymbol{\sigma}), \tag{45}$$

The first step is obtained using (33). In general, we can use the stochastic approximation by simultaneous perturbation (SPSS) method (Bhatnagar, 2007; Spall, 2000; Salimans et al., 2017) to compute derivatives.

By defining $\mathbf{s}_t := \boldsymbol{\sigma}_t^{-2}$, we can write the update as follows:

$$\text{VON using Reparam: } \boldsymbol{\theta}_t = \boldsymbol{\mu}_t + \boldsymbol{\epsilon}_t/\sqrt{\mathbf{s}_t},\ \text{ where } \epsilon_t \sim \mathcal{N}(0, \mathbf{I})$$

$$\mathbf{s}_{t+1} = (1 - \tau\beta_t)\mathbf{s}_t + \beta_t\left[\hat{\mathbf{g}}(\boldsymbol{\theta}_t)\boldsymbol{\epsilon}_t\sqrt{\mathbf{s}_t}\right] \tag{46}$$

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t - \beta_t\ \left[\hat{\mathbf{g}}(\boldsymbol{\theta}_t)/\mathbf{s}_{t+1}\right]$$

This update resembles the RMSprop updates shown in (11). It differs from it in ways. First, in the update of $\boldsymbol{\mu}$, there is no square-root over the scaling vector $\mathbf{s}_{t+1}$ and also no δ is added. Second, the $\hat{\mathbf{g}}(\boldsymbol{\theta}_t)$ is replace by $\epsilon_t/\boldsymbol{\sigma}_t$. These changes can easily be made in the existing implementation of RMSprop.

One major issue with this estimate is that it might have high variance and $\mathbf{s}_t$ may not always be positive-definite. To make sure that each step $\mathbf{s} > 0$, we can use a simple back-tracking method described below. Denote element d of $\mathbf{s}$ as s_d and simplify notation by denoting h_d to be d'th element of $\hat{\mathbf{g}}(\boldsymbol{\theta})\,(\boldsymbol{\epsilon}/\boldsymbol{\sigma})$. For $\mathbf{s}$ to remain positive, we need $s_d + \beta_t h_d > 0, \forall d$. As h_d can become negative, a too large value for β_t will move $\mathbf{s}$ out of the feasible set. We thus have to find the largest value we can set β_t to such that $\mathbf{s}$ is still in the feasible set. Let $\mathcal{I}$ denote the indices d for which $s_d + \beta_t h_d \le 0$. We can ensure that $\mathbf{s}$ stays in the feasible set by setting

$$\beta_t = \min\left\{\beta_0, \delta \min_{d\in\mathcal{I}} \frac{s_d}{|h_d|}\right\}, \tag{47}$$

where β_0 is the maximum learning rate and $0 < \delta < 1$ is a constant to keep $\mathbf{s}$ strictly within the feasible set (away from the borders).

E. Proof to Show That Fixed-Point Do Not Change

We now show that the fixed-points of updates in (9) and (10) are equivalent. Denote the variational distribution at iteration t by $q_t := \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t^2)$. Assume that at $t \to \infty$, q_t converges to $q_* := \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_*, \boldsymbol{\sigma}_*^2)$. Denote the scale vectors corresponding to q_t and q_* by $\mathbf{s}_t$ and $\mathbf{s}_*$ respectively. At a fixed point, we have $\boldsymbol{\mu}_t = \boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_*$ and $\mathbf{s}_t = \mathbf{s}_{t+1} = \mathbf{s}_*$. Plugging this into the update (9) and replacing the $\hat{\mathbf{g}}(\boldsymbol{\theta})$ and $[\hat{\mathbf{g}}(\boldsymbol{\theta})]^2$ with their expectations with respect to q_*, we get the following condition that a fixed-point q_* must satisfy:

$$\mathbb{E}_{q_*}\left[\hat{g}(\boldsymbol{\theta})\right] + \lambda\boldsymbol{\mu}_* = 0, \quad \mathbb{E}_{q_*}\left[\{\hat{g}(\boldsymbol{\theta})\}^2\right] - \mathbf{s}_* = 0, \tag{48}$$

If we repeat this for the update (10) we recover the same condition, implying that the scaling vector is irrelevant for fixed points. The scaling vectors do however affect the rate of convergence. The square-root version in RMSprop and other adaptive gradient methods is known to work well in practice, which is why we use the Vprop-1 updates where the we take the square-root of the scaling vector.

F. Derivation of Vadam

Consider the following update of Adam:

$$\begin{aligned} \text{Adam: } \mathbf{m}_{t+1} &= \gamma\mathbf{m}_t + (1-\gamma)\hat{\mathbf{g}}(\boldsymbol{\theta}_t) \\ \mathbf{s}_{t+1} &= \beta\mathbf{s}_t + (1-\beta)\ [\hat{\mathbf{g}}(\boldsymbol{\theta}_t)]^2 \\ \hat{\mathbf{m}}_{t+1} &= \mathbf{m}_{t+1}/(1-\gamma^{t+1}) \\ \hat{\mathbf{s}}_{t+1} &= \mathbf{s}_{t+1}/(1-\beta^{t+1}) \\ \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t - \alpha\ \hat{\mathbf{m}}_{t+1}/(\sqrt{\hat{\mathbf{s}}_{t+1}} + \delta), \end{aligned} \tag{49}$$

This update can be expressed as the following *scaled* version of the Polyak's heavy ball[3] method,

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \tilde{\alpha}_t\left[\frac{1}{\sqrt{\hat{\mathbf{s}}_{t+1}}+\delta}\right]\nabla_\theta f(\boldsymbol{\theta}_t) + \tilde{\gamma}_t\left[\frac{\sqrt{\hat{\mathbf{s}}_t}+\delta}{\sqrt{\hat{\mathbf{s}}_{t+1}}+\delta}\right](\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}), \tag{50}$$

where $\tilde{\alpha}_t, \tilde{\gamma}_t$ are appropriately defined in terms of γ as shown below:

$$\tilde{\alpha}_t := \alpha\frac{1-\gamma}{1-\gamma^t}, \quad \tilde{\gamma}_t := \gamma\frac{1-\gamma^{t-1}}{1-\gamma^t} \tag{51}$$

We will now show that, by replacing the Euclidean distance in the Polyak's heavy ball method by a KL divergence, we can get an Adam like update for Vprop where the step-sizes are adapted.

We propose the following modification to our mirror descent update shown in (4),

$$\mathbf{m}_{t+1} = \arg\min_{\boldsymbol{m}} \langle \mathbf{m}, \nabla_m\mathcal{L}_t\rangle + \frac{1}{\beta_t}\mathbb{D}_{KL}[q\,\|\,q_t] - \frac{\alpha_t}{\beta_t}\mathbb{D}_{KL}[q\,\|\,q_{t-1}], \tag{52}$$

where $\mathcal{L}$ refers to the negative of the ELBO defined in Eq. 1. The last term here is a momentum term, which is very similar to the momentum term in (14), but is defined using the KL divergence instead of the Euclidean distance.

We first assume that the covariance matrix is full $\boldsymbol{\Sigma}$ and then make the mean-field approximation.

Using the results of Appendix A and taking derivative and setting to zero, we get the following:

$$\nabla_m\mathcal{L}_t + \frac{1}{\beta_t}(\boldsymbol{\lambda}_{t+1} - \boldsymbol{\lambda}_t) - \frac{\alpha_t}{\beta_t}(\boldsymbol{\lambda}_{t+1} - \boldsymbol{\lambda}_t) = 0 \tag{53}$$

$$\Rightarrow \boldsymbol{\lambda}_{t+1} = \frac{1}{1-\alpha_t}\boldsymbol{\lambda}_t - \frac{\alpha_t}{1-\alpha_t}\boldsymbol{\lambda}_{t-1} - \frac{\beta_t}{1-\alpha_t}\nabla_m\mathcal{L}_t. \tag{54}$$

[3](Wilson et al., 2017) do not add the constant δ in $\sqrt{\mathbf{s}_t}$ but in Adam a small constant is added.

Recalling that the mean parameters of a Gaussian $q(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ are $\mathbf{m}^{(1)} = \boldsymbol{\mu}$ and $\mathbf{M}^{(2)} = \boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T$ and using the chain rule, we can express the gradient $\nabla_m \mathcal{L}_t$ in terms of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$,

$$\nabla_{m^{(1)}} \mathcal{L} = \nabla_\mu \mathcal{L} - 2\left[\nabla_\Sigma \mathcal{L}\right] \boldsymbol{\mu} \tag{55}$$
$$\nabla_{M^{(2)}} \mathcal{L} = \nabla_\Sigma \mathcal{L}. \tag{56}$$

Using the natural parameters of a Gaussian defined as $\boldsymbol{\lambda}^{(1)} = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$ and $\boldsymbol{\lambda}^{(2)} = -\frac{1}{2}\boldsymbol{\Sigma}^{-1}$, we can rewrite the update (54) the update for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. First, the update for $\boldsymbol{\Sigma}$ is the following,

$$\boldsymbol{\Sigma}_{t+1}^{-1} = \frac{1}{1-\alpha_t}\boldsymbol{\Sigma}_t^{-1} - \frac{\alpha_t}{1-\alpha_t}\boldsymbol{\Sigma}_{t-1}^{-1} + \frac{2\beta_t}{1-\alpha_t}\left[\nabla_\Sigma \mathcal{L}_t\right] \tag{57}$$

Now, for $\boldsymbol{\mu}$, we can rearrange the update to express some of the terms as $\boldsymbol{\Sigma}_{t+1}$:

$$\begin{aligned}
\boldsymbol{\Sigma}_{t+1}^{-1}\boldsymbol{\mu}_{t+1} &= \frac{1}{1-\alpha_t}\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\mu}_t - \frac{\alpha_t}{1-\alpha_t}\boldsymbol{\Sigma}_{t-1}^{-1}\boldsymbol{\mu}_{t-1} - \frac{\beta_t}{1-\alpha_t}\left(\nabla_\mu \mathcal{L}_t - 2\left[\nabla_\Sigma \mathcal{L}_t\right]\boldsymbol{\mu}_t\right) && (58)\\
&= \left[\frac{1}{1-\alpha_t}\boldsymbol{\Sigma}_t^{-1} - \frac{\alpha_t}{1-\alpha_t}\boldsymbol{\Sigma}_{t-1}^{-1} + \frac{2\beta_t}{1-\alpha_t}\left[\nabla_\Sigma \mathcal{L}_t\right]\right]\boldsymbol{\mu}_t - \frac{\beta_t}{1-\alpha_t}\nabla_\mu \mathcal{L}_t + \frac{\alpha_t}{1-\alpha_t}\boldsymbol{\Sigma}_{t-1}^{-1}(\boldsymbol{\mu}_t - \boldsymbol{\mu}_{t-1}) && (59)\\
&= \boldsymbol{\mu}_t - \frac{\beta_t}{1-\alpha_t}\boldsymbol{\Sigma}_{t+1}\left[\nabla_\mu \mathcal{L}_t\right] + \frac{\alpha_t}{1-\alpha_t}\boldsymbol{\Sigma}_{t+1}\boldsymbol{\Sigma}_{t-1}^{-1}(\boldsymbol{\mu}_t - \boldsymbol{\mu}_{t-1}). && (60)
\end{aligned}$$

where the first step is obtained by simply plugging the definition of $\boldsymbol{\lambda}^{(1)}$ in (54), the second step is obtained by adding and subtracting the term $\frac{\alpha_t}{1-\alpha_t}\boldsymbol{\Sigma}_{t-1}^{-1}\boldsymbol{\mu}_t$ and rearranging, and the third step is obtained by substituting the definition of $\boldsymbol{\Sigma}_{t+1}$ from (57).

To express these updates similar to Adam, we need to make an approximation where replace the instances of $\boldsymbol{\Sigma}_{t-1}$ by $\boldsymbol{\Sigma}_t$ in both (57) and (60). With this approximation, we get the following:

$$\boldsymbol{\Sigma}_{t+1}^{-1} = \boldsymbol{\Sigma}_t^{-1} + \frac{2\beta_t}{1-\alpha_t}\left[\nabla_\Sigma \mathcal{L}_t\right] \tag{61}$$
$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t - \frac{\beta_t}{1-\alpha_t}\boldsymbol{\Sigma}_{t+1}\left[\nabla_\mu \mathcal{L}_t\right] + \frac{\alpha_t}{1-\alpha_t}\boldsymbol{\Sigma}_{t+1}\boldsymbol{\Sigma}_t^{-1}(\boldsymbol{\mu}_t - \boldsymbol{\mu}_{t-1}). \tag{62}$$

From here, we follow the same procedure used to derived the VON update in Section C. That is, we first use Bonnet's and Price's theorem to express the gradients with respect to $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ in terms of the expectations of gradients and Hessian of $f(\boldsymbol{\theta})$. Then, we substitute the expectation with a sample $\boldsymbol{\theta}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$. Finally, we redefine the matrix $\mathbf{S}_t = \boldsymbol{\Sigma}_t^{-1} - \lambda\mathbf{I}$. With this we get the following update which a momentum version of VON:

$$\text{VON with Natural Momentum: } \mathbf{S}_{t+1} = \left(1 - \frac{\beta_t}{1-\alpha_t}\right)\mathbf{S}_t + \frac{\beta_t}{1-\alpha_t}\ \nabla_{\theta\theta}^2 f(\boldsymbol{\theta}_t), \tag{63}$$
$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t - \frac{\beta_t}{1-\alpha_t}\ \left(\mathbf{S}_{t+1} + \lambda\mathbf{I}\right)^{-1}\left(\nabla_\theta f(\boldsymbol{\theta}_t) + \lambda\boldsymbol{\mu}_t\right) + \frac{\alpha_t}{1-\alpha_t}(\mathbf{S}_{t+1} + \lambda\mathbf{I})^{-1}(\mathbf{S}_t + \lambda\mathbf{I})(\boldsymbol{\mu}_t - \boldsymbol{\mu}_{t-1}),$$

where $\boldsymbol{\theta}_t \sim \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$ with $\boldsymbol{\Sigma}_t = (\mathbf{S}_t + \lambda\mathbf{I})^{-1}$.

To get a momentum version of Vprop, we follow similar to Section 3. That is, we first employ mean-field approximation, then replace the Hessian by the gradient-magnitude approximation, and finally use a square-root of the scaling vectors. Defining $\tilde{\alpha}_t := \beta_t/(1-\alpha_t)$ and $\tilde{\gamma}_t := \alpha_t/(1-\alpha_t)$, we can write the update for Vprop with momentum,

$$\text{Vprop with Natural Momentum: } \mathbf{s}_{t+1} = (1-\tilde{\alpha}_t)\,\mathbf{s}_t + \tilde{\alpha}_t\ \left[\hat{\mathbf{g}}(\boldsymbol{\theta}_t)\right]^2, \tag{64}$$
$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t - \tilde{\alpha}_t\ \left[\frac{1}{\sqrt{\mathbf{s}_{t+1}} + \lambda}\right]\left(\nabla_\theta f(\boldsymbol{\theta}_t) + \lambda\boldsymbol{\mu}_t\right) + \tilde{\gamma}_t\left[\frac{\sqrt{\mathbf{s}_t} + \lambda}{\sqrt{\mathbf{s}_{t+1}} + \lambda}\right](\boldsymbol{\mu}_t - \boldsymbol{\mu}_{t-1}),$$

where $\boldsymbol{\theta}_t \sim \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t^2)$ with $\boldsymbol{\sigma}_t^{-2} = (\mathbf{s}_t + \lambda)$. This is very similar to the update (50) of Adam expressed in the momentum form. We can implement this update by using Adam's update shown in Fig. 1. This will mean that we do not use the same step-size $\tilde{\alpha}_t$ for $\mathbf{s}_t$ and $\boldsymbol{\mu}_t$, rather choose the step-sizes according to the Adam update.

The final update is shown below, where we highlight the differences from Adam in red.

$$
\begin{aligned}
\text{Vadam: } & \boldsymbol{\theta}_t \sim \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t^2), \text{ where } \boldsymbol{\sigma}_t^2 = 1/(\mathbf{s}_t + \lambda) \\
\mathbf{m}_{t+1} &= \gamma \mathbf{m}_t + (1-\gamma)\hat{\mathbf{g}}(\boldsymbol{\theta}_t) \\
\mathbf{s}_{t+1} &= \beta \mathbf{s}_t + (1-\beta)\ [\hat{\mathbf{g}}(\boldsymbol{\theta}_t)]^2 \\
\hat{\mathbf{m}}_{t+1} &= \mathbf{m}_{t+1}/(1-\gamma^{t+1}) \\
\hat{\mathbf{s}}_{t+1} &= \mathbf{s}_{t+1}/(1-\beta^{t+1}) \\
\boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t - \alpha\ \hat{\mathbf{m}}_{t+1}/(\sqrt{\hat{\mathbf{s}}_{t+1}} + \lambda),
\end{aligned}
\tag{65}
$$

G. Implementation Details

Monte-Carlo Sampling: Typically, we use 1 samples, however in some problems this converges slowly. For such problems using 5-10 samples was sufficient. We used mirror sampling where we draw one ϵ and added a negative sample $-\epsilon$ corresponding to it. This helped us improve the stability of our method.

Step-size selection: In most of our experiments we use default values of RMSprop and Adam.

Mini-batches: When using mini-batches to estimate the gradients, there are multiple options for our gradient-magnitude approximation. For example, when using a mini-batch of size M, we could weight the gradients by N/M and then square these gradients to obtain the Hessian approximation. Alternatively we could square the gradients first and then weight the resulting Hessian approximation by N/M. We found that the two options usually give very similar results on practical problems. In our experiments, we used the first alternative which shrinks the variance a bit more, but gives more stable behavior.

Details of our Tensorflow implementation: We provide a reference implementation of Vprop and Vadam in Tensorflow for deep learning models that currently operates by explicitly perturbing the weights before computing the gradients. As such, it maintains a copy of the unperturbed weights at each iteration and uses it to compute the final parameter updates. We also clip the gradients and the precision parameter to ensure stability during training. In the future we plan to incorporate our methods in the Optimisation API.

H. Details of Experiments on UCI dataset

The details of data size are given in (Gal & Ghahramani, 2016). For all datasets, we used mini-batches of size 128 and train the neural networks for 40 epochs. For Vprop, Vprop with momentum, and Vadam, we use step-sizes $\alpha = 0.01$, $\beta = 0.999$, and $\gamma = 0.9$. For BBVI, we optimize an approximated variational lower-bound using Adam with step-sizes $\alpha = 0.1$, $\beta = 0.999$, and $\gamma = 0.9$. For our methods and BBVI, we choose the prior precision λ and the noise precision τ by grid-search and choose the candidates pair (λ, τ) that give the minimum RMSE for each dataset. For BBVI, we approximate the variational lower bound using 200 MC samples and optimize it using Adam with step-sizes $\alpha = 0.1$, $\beta = 0.999$, and $\gamma = 0.9$. For test RMSE, we report results over 20 different train-test splits provided by the authors of (Gal & Ghahramani, 2016).

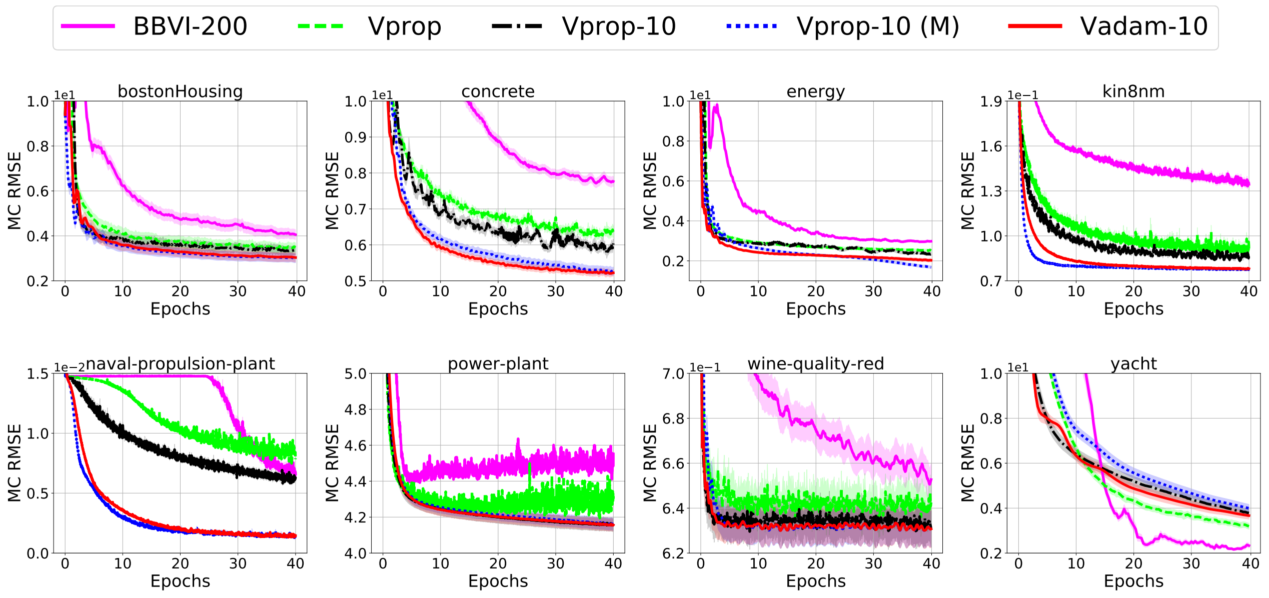


Figure 5. The mean and standard error of the Monte-Carlo RMSE (MC RMSE) on the test sets of UCI experiments. The mean and standard error are computed over 20 data splits. Vprop with 1 MC sample is denoted by Vprop-1. Vprop-10 denotes Vprop with 10 MC samples. Both Vprop with momentum and Vadam use 10 MC samples.

I. Details of LIBSVM and MNIST experiments

Table 3. A list of datasets, models and hyperparameters used for experiments on MNIST and LIBSVM datasets.

Dataset	**N**	**Hidden Units**	**Act.**	α	β	γ	δ/λ
a1a	1605	(64,32,16,8,4)	tanh	0.01	0.99	0.9	10^{-8}
a3a	3185	(64,32,16,8,4)	tanh	0.01	0.99	0.9	10^{-8}
a5a	6414	(64,32,16,8,4)	tanh	0.01	0.99	0.9	10^{-8}
a9a	32561	(64,32,16,8,4)	tanh	0.01	0.99	0.9	10^{-8}
Small MNIST	10000	(400,100,50)	tanh	0.001	0.999	0.9	10^{-8}
Full MNIST	60000	(400, 400)	ReLU	0.001	0.999	0.9	10^{-6}

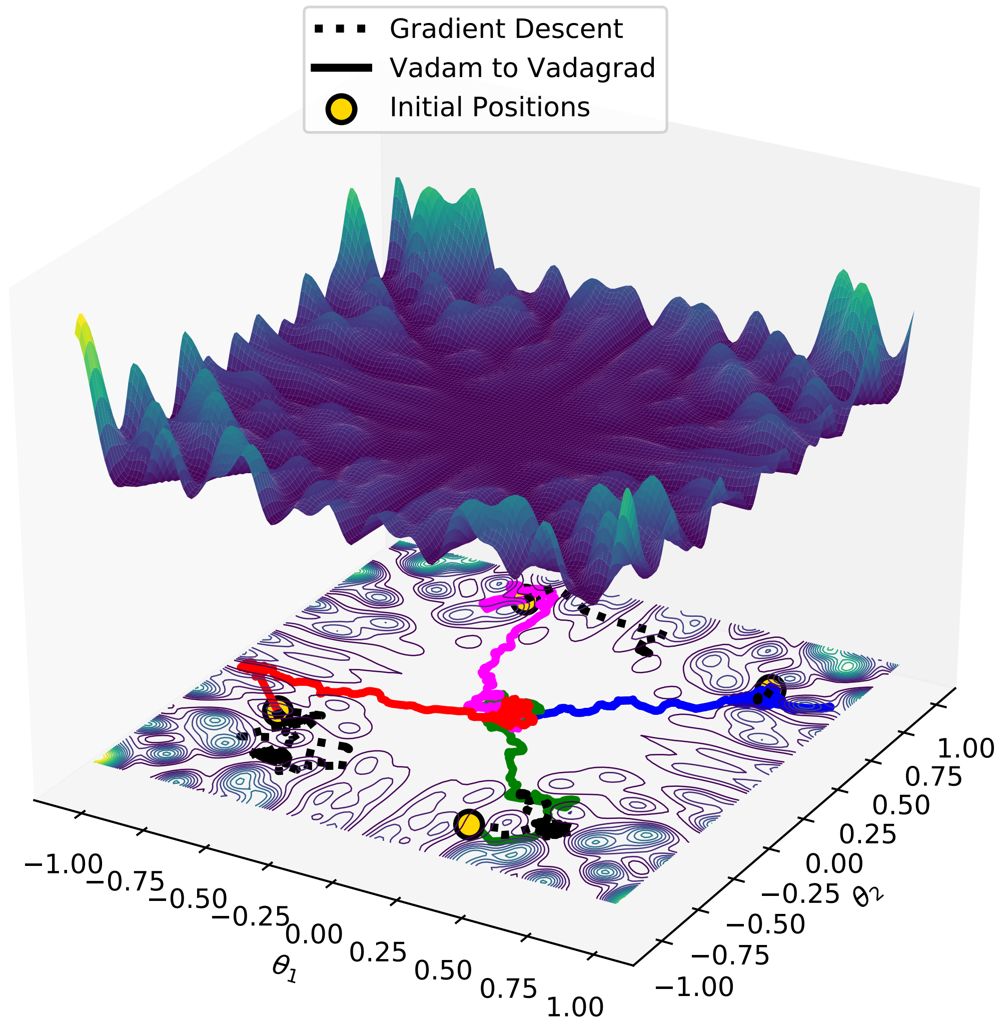


Figure 6. Illustration of variational optimization for a complex two-dimensional objective function. Variational optimization were performed by gradually turning off the KL-term for Vadam, thus annealing Vadam towards VadaGrad. This procedure was run from four different initial positions. The four runs are shown in solid lines in different colors. Gradient descent (shown in dotted, black lines) were also initialized at the same locations. Vadam-VadaGrad shows ability to navigate the landscape to reach the flat (and global) minimum, while gradient descent gets stuck in various locations.

J. Illustration of Variational Optimization

K. Experimental Results for On Variational Optimization for Reinforcement Learning

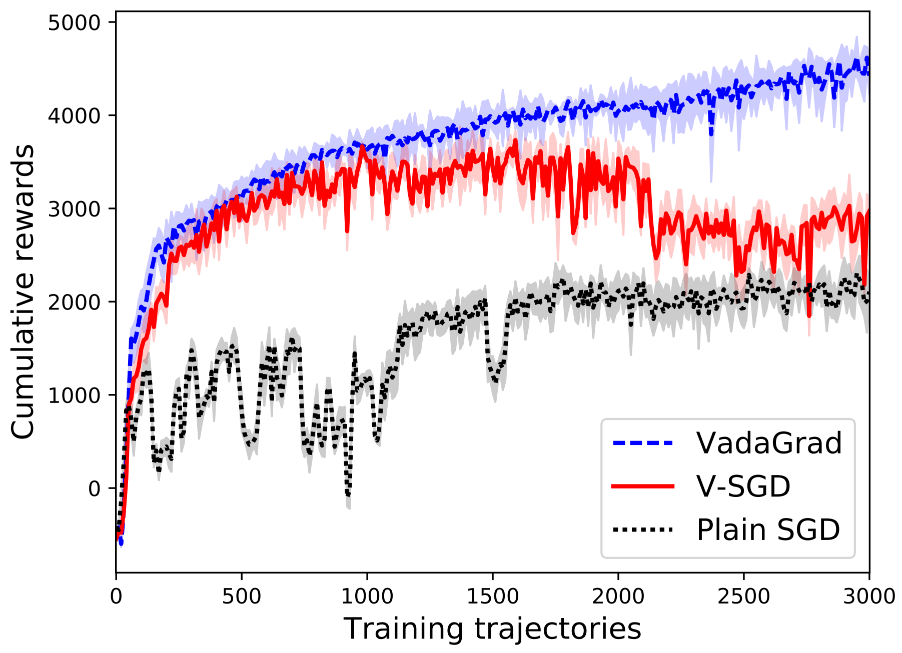


Figure 7. The cumulative rewards obtained by VadaGrad, V-SGD and plain SGD on the half-cheetah task. The mean and standard error are computed over 5 trials.

L. Experimental details for the "Marginal Value" Experiment

We use the fixed-decay scheme mentioned by the authors. We decay α ($\hat{\alpha}$) at every K epochs by dividing α ($\hat{\alpha}$) by 10.

Method	$\hat{\alpha}$	$\hat{\beta}$	K	s_0
VAdaGrad	0.0075	0.5	40	750
VAG	0.001	0.5	100	750
VSGD	2.0	0.5	40	10000

For the baseline methods, we tune the step size α and K. The best values can be found as below.

Method	α	K	β
SGD	0.5	80	NA
AdaGrad	0.025	80	0.5
Adam	0.0012	40	NA