A. Changes in the Camera-Ready Version Compared to the Submitted Version

- Taking reviewer's suggestions into account, we changed the title of our paper. The title of our submitted version was "Vadam: Fast and Scalable Variational Inference by Perturbing Adam".
- In the submitted version, we motivated our approach based on its ease of implementation. In the new version, we changed the motivation to make VI as easy to implement and execute as MLE.
- In the new version, we have added a separate section on related work.
- We improved the discussion of our approximation methods, and added an error analysis.
- Overall conclusions of our paper have also slightly changed in the new version. The new conclusions suggest that there is a trade-off between the ease-of-implementation and quality of uncertainty approximation.
- As per reviewers suggestions, we also made major improvements in our experiment results.
  - We added test log-likelihood in the BNN results. We changed the hyperparameter selection from grid search to Bayesian optimization. We removed two methods from the table, namely PBP and VIG, since they use different splits compared to our setting. We improved the performance of BBVI by using better initialization and learning rates. We corrected a scaling problem with our Vadam method. The new method gives a slightly worse performance than the results present in the submitted version.
  - We added a logistic regression experiment where we evaluate the quality of uncertainty estimates.
  - We added the details of the RL experiments which we forgot to add in the submitted version. We also added a comparison to Adam-based methods in the appendix for the RL experiment.
  - We removed an unclear result about reducing overfitting.

B. Review of Natural-Gradient Variational Inference

Khan & Lin (2017) propose a natural-gradient method for variational inference. In this section, we briefly discuss this method.

Denote the variational objective by $\mathcal{L}(\boldsymbol{\eta})$ for the variational distribution $q_\eta(\boldsymbol{\theta})$ which takes an exponential-family form with natural-parameter $\boldsymbol{\eta}$. The objective is given as follows:

$$\mathcal{L}(\boldsymbol{\eta}) := \sum_{i=1}^{N} \mathbb{E}_q \left[\log p(\mathcal{D}_i|\boldsymbol{\theta})\right] + \mathbb{E}_q \left[\log \frac{p(\boldsymbol{\theta})}{q_\eta(\boldsymbol{\theta})}\right]. \tag{25}$$

We assume that the exponential-family is in minimal representation. Therefore, we can express $\mathcal{L}(\boldsymbol{\eta})$ in terms of the expectation parameter, denoted by $\mathbf{m}$. We denote the new objective by $\mathcal{L}_*(\mathbf{m}) := \mathcal{L}(\boldsymbol{\eta})$. We can also reparameterize q_η in terms of $\mathbf{m}$ and denote it by q_m.

Natural-gradient methods exploit the Riemannian geometry of $q(\boldsymbol{\theta})$ by scaling the gradient by the inverse of the Fisher information matrix. The method of Khan & Lin (2017) simplifies the update by avoiding a direct computation of the FIM. This is made possible due to a relationship between the natural parameter $\boldsymbol{\eta}$ and the expectation parameter $\mathbf{m}$ of an exponential-family distribution. The relationship dictates that the natural gradient with respect to $\boldsymbol{\eta}$ is equal to the gradient with respect to $\mathbf{m}$. This is stated below where FIM is denoted by $\mathbf{F}(\boldsymbol{\eta})$,

$$\mathbf{F}(\boldsymbol{\eta})^{-1} \nabla_\eta \mathcal{L}(\boldsymbol{\eta}) = \nabla_m \mathcal{L}_*(\mathbf{m}), \tag{26}$$

This relationship has been discussed in the earlier work of Hensman et al. (2012) and can also be found in Amari (2016).

The method of Khan & Lin (2017) exploits this result within a mirror descent framework. They propose to use a mirror-descent update in the expectation- parameter space which is equivalent to the natural-gradient update in the natural-parameter space. Therefore, the natural-gradient can be computed by computing the gradient with respect to the expectation parameter. We give a formal statement below.

Theorem 2. *Consider the following mirror-descent step:*

$$\mathbf{m}_{t+1} = \arg\min_{\boldsymbol{m}} \ \langle \mathbf{m}, -\nabla_m \mathcal{L}_*(\mathbf{m}_t)\rangle + \frac{1}{\beta_t}\mathbb{D}_{KL}[q_m(\boldsymbol{\theta}) \,\|\, q_{m_t}(\boldsymbol{\theta})], \tag{27}$$

where $\mathbb{D}_{KL}[\cdot \,\|\, \cdot]$ is the Kullback-Leibler divergence and β_t is the learning rate in iteration t. Each step of this mirror descent update is equivalent to the following natural-gradient descent in the natural-parameter space:

$$\boldsymbol{\eta}_{t+1} = \boldsymbol{\eta}_t + \beta_t \mathbf{F}(\boldsymbol{\eta}_t)^{-1} \nabla_\eta \mathcal{L}(\boldsymbol{\eta}_t) \tag{28}$$

A formal proof of this statement can be found in Raskutti & Mukherjee (2015).

Using (26), the natural-gradient update above can be simply written as the following:

$$\boldsymbol{\eta}_{t+1} = \boldsymbol{\eta}_t + \beta_t \nabla_m \mathcal{L}_*(\mathbf{m}_t) \tag{29}$$

which involves computing the gradient with respect to $\mathbf{m}$ but taking a step in the natural-parameter space.

As we show in the next section, the above relationship enables us to derive a simple natural-gradient update because, for a Gaussian distribution, the gradient with respect to $\mathbf{m}$ has a simple closed-form solution.

C. Derivation of Natural-Gradient Updates for Gaussian Mean-Field Variational Inference

In this section, we derive the natural-gradient update for the Gaussian approximation $q_\eta(\boldsymbol{\theta}) := \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. In the end, we will make the mean-field approximation: $\boldsymbol{\Sigma} = \text{diag}(\boldsymbol{\sigma}^2)$ to get the final update.

We start by defining the natural and expectation parameters of a Gaussian:

$$\boldsymbol{\eta}^{(1)} := \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}, \quad \boldsymbol{\eta}^{(2)} = -\tfrac{1}{2}\boldsymbol{\Sigma}^{-1} \tag{30}$$

$$\mathbf{m}^{(1)} := \mathbb{E}_{q_m}(\boldsymbol{\theta}) = \boldsymbol{\mu}, \quad \mathbf{M}^{(2)} = \mathbb{E}_q(\boldsymbol{\theta}\boldsymbol{\theta}^\top) = \boldsymbol{\mu}\boldsymbol{\mu}^\top + \boldsymbol{\Sigma}. \tag{31}$$

Now we will express the gradient with respect to these expectation parameters in terms of the gradients with respect to $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ using the chain rule (see Appendix B.1 in Khan & Lin (2017) for a derivation):

$$\nabla_{m^{(1)}} \mathcal{L}_* = \nabla_\mu \mathcal{L} - 2\left[\nabla_\Sigma \mathcal{L}\right]\boldsymbol{\mu}, \tag{32}$$

$$\nabla_{M^{(2)}} \mathcal{L}_* = \nabla_\Sigma \mathcal{L}. \tag{33}$$

Next, using the definition of natural parameters, we can rewrite (29) in terms of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$:

$$\boldsymbol{\Sigma}_{t+1}^{-1} = \boldsymbol{\Sigma}_t^{-1} - 2\beta_t\left[\nabla_\Sigma \mathcal{L}_t\right] \tag{34}$$

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\Sigma}_{t+1}\left[\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\mu}_t + \beta_t\left(\nabla_\mu \mathcal{L}_t - 2\left[\nabla_\Sigma \mathcal{L}_t\right]\boldsymbol{\mu}_t\right)\right] \tag{35}$$

$$= \boldsymbol{\Sigma}_{t+1}\left[\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\mu}_t + \beta_t \nabla_\mu \mathcal{L}_t - 2\beta_t\left[\nabla_\Sigma \mathcal{L}_t\right]\boldsymbol{\mu}_t\right] \tag{36}$$

$$= \boldsymbol{\Sigma}_{t+1}\left[\left(\boldsymbol{\Sigma}_t^{-1} - 2\beta_t\left[\nabla_\Sigma \mathcal{L}_t\right]\right)\boldsymbol{\mu}_t + \beta_t \nabla_\mu \mathcal{L}_t\right] \tag{37}$$

$$= \boldsymbol{\Sigma}_{t+1}\left[\boldsymbol{\Sigma}_{t+1}^{-1}\boldsymbol{\mu}_t + \beta_t \nabla_\mu \mathcal{L}_t\right] \tag{38}$$

$$= \boldsymbol{\mu}_t + \beta_t \boldsymbol{\Sigma}_{t+1}\left[\nabla_\mu \mathcal{L}_t\right]. \tag{39}$$

In summary, the natural-gradient update is

$$\boldsymbol{\Sigma}_{t+1}^{-1} = \boldsymbol{\Sigma}_t^{-1} - 2\beta_t\left[\nabla_\Sigma \mathcal{L}_t\right], \tag{40}$$

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + \beta_t \boldsymbol{\Sigma}_{t+1}\left[\nabla_\mu \mathcal{L}_t\right]. \tag{41}$$

By considering a Gaussian mean-field VI with a diagonal covariance: $\boldsymbol{\Sigma} = \text{diag}(\boldsymbol{\sigma}^2)$, we obtain

$$\boldsymbol{\sigma}_{t+1}^{-2} = \boldsymbol{\sigma}_t^{-2} - 2\beta_t\left[\nabla_{\sigma^2} \mathcal{L}_t\right], \tag{42}$$

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + \beta_t \boldsymbol{\sigma}_{t+1}^2\left[\nabla_\mu \mathcal{L}_t\right]. \tag{43}$$

In update (5), we use stochastic gradients instead of exact gradients.

Note that there is an explicit constraint in the above update, i.e., the precision $\boldsymbol{\sigma}^{-2}$ needs to be positive at every step. The learning rate can be adapted to make sure that the constraint is always satisfied. We discuss this method in Appendix D.1. Another option is to make an approximation, such as a Gauss-Newton approximation, to make sure that this constraint is always satisfied. We make this assumption in one of our methods called Variational Online Gauss-Newton Method.

D. Derivation of the Variational Online-Newton method (VON)

In this section, we derive the variational online-Newton (VON) method proposed in Section 3. We will modify the NGVI update in (41).

The variational lower-bound in (25) can be re-expressed as

$$\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) := \mathbb{E}_q \left[-N f(\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) - \log q(\boldsymbol{\theta}) \right], \tag{44}$$

where $f(\boldsymbol{\theta}) = -\frac{1}{N} \sum_{i=1}^{N} \log p(\mathcal{D}_i|\boldsymbol{\theta})$. To derive VON, we use the Bonnet's and Price's theorems (Opper & Archambeau, 2009; Rezende et al., 2014) to express the gradients of the expectation of $f(\boldsymbol{\theta})$ with respect to $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ in terms of the gradient and Hessian of $f(\boldsymbol{\theta})$, i.e.,

$$\nabla_\mu \mathbb{E}_q \left[f(\boldsymbol{\theta}) \right] = \mathbb{E}_q \left[\nabla_\theta f(\boldsymbol{\theta}) \right] := \mathbb{E}_q \left[\mathbf{g}(\boldsymbol{\theta}) \right], \tag{45}$$

$$\nabla_\Sigma \mathbb{E}_q \left[f(\boldsymbol{\theta}) \right] = \tfrac{1}{2} \mathbb{E}_q \left[\nabla^2_{\theta\theta} f(\boldsymbol{\theta}) \right] := \tfrac{1}{2} \mathbb{E}_q \left[\mathbf{H}(\boldsymbol{\theta}) \right], \tag{46}$$

where $\mathbf{g}(\boldsymbol{\theta}) := \nabla_\theta f(\boldsymbol{\theta})$ and $\mathbf{H}(\boldsymbol{\theta}) := \nabla^2_{\theta\theta} f(\boldsymbol{\theta})$ denote the gradient and Hessian of $f(\boldsymbol{\theta})$, respectively. Using these, we can rewrite the gradients of $\mathcal{L}$ in the NGVI update in (41) as

$$\begin{aligned}
\nabla_\mu \mathcal{L} &= \nabla_\mu \mathbb{E}_q \left[-N f(\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) - \log q(\boldsymbol{\theta}) \right] && (47)\\
&= - \left(\mathbb{E}_q \left[N \nabla_\theta f(\boldsymbol{\theta}) \right] + \lambda \boldsymbol{\mu} \right) && (48)\\
&= - \left(\mathbb{E}_q \left[N \mathbf{g}(\boldsymbol{\theta}) \right] + \lambda \boldsymbol{\mu} \right), && (49)\\
\nabla_\Sigma \mathcal{L} &= \tfrac{1}{2} \mathbb{E}_q \left[-N \nabla^2_{\theta\theta} f(\boldsymbol{\theta}) \right] - \tfrac{1}{2} \lambda \mathbf{I} + \tfrac{1}{2} \boldsymbol{\Sigma}^{-1} && (50)\\
&= \tfrac{1}{2} \mathbb{E}_q \left[-N \mathbf{H}(\boldsymbol{\theta}) \right] - \tfrac{1}{2} \lambda \mathbf{I} + \tfrac{1}{2} \boldsymbol{\Sigma}^{-1}, && (51)
\end{aligned}$$

By substituting these into the NGVI update of (41) and then approximating the expectation by one Monte-Carlo sample $\boldsymbol{\theta}_t \sim \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$, we get the following updates:

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t - \beta_t \; \boldsymbol{\Sigma}_{t+1} \left[N \mathbf{g}(\boldsymbol{\theta}_t) + \lambda \boldsymbol{\mu}_t \right], \tag{52}$$

$$\boldsymbol{\Sigma}^{-1}_{t+1} = (1 - \beta_t) \boldsymbol{\Sigma}^{-1}_t + \; \beta_t \; \left[N \mathbf{H}(\boldsymbol{\theta}_t) + \lambda \mathbf{I} \right]. \tag{53}$$

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

$$\begin{aligned}
\text{VON (Full-covariance): } \boldsymbol{\mu}_{t+1} &= \boldsymbol{\mu}_t - \beta_t \; \left(\mathbf{S}_{t+1} + \lambda \mathbf{I}/N \right)^{-1} \left(\mathbf{g}(\boldsymbol{\theta}_t) + \lambda \boldsymbol{\mu}_t / N \right), \\
\mathbf{S}_{t+1} &= (1 - \beta_t) \mathbf{S}_t + \beta_t \; \mathbf{H}(\boldsymbol{\theta}_t),
\end{aligned} \tag{54}$$

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

For the mean-field variant, we can use a diagonal Hessian:

$$\begin{aligned}
\text{VON: } \boldsymbol{\mu}_{t+1} &= \boldsymbol{\mu}_t - \beta_t \; \left(\mathbf{g}(\boldsymbol{\theta}_t) + \tilde{\lambda} \boldsymbol{\mu}_t \right) / \left(\mathbf{s}_{t+1} + \tilde{\lambda} \right), \\
\mathbf{s}_{t+1} &= (1 - \beta_t) \mathbf{s}_t + \beta_t \; \text{diag}(\mathbf{H}(\boldsymbol{\theta}_t)),
\end{aligned} \tag{55}$$

where we have defined $\tilde{\lambda} = \lambda/N$, and $\boldsymbol{\theta}_t \sim \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_t, \text{diag}(\boldsymbol{\sigma}^2_t))$ with $\boldsymbol{\sigma}^2_t = 1/[N(\mathbf{s}_t + \tilde{\lambda})]$.

By replacing $\mathbf{g}$ and $\mathbf{H}$ by their stochastic estimates, we obtain the VON update in (8) of the main text.

D.1. Hessian Approximation Using the Reparameterization Trick

In this section we briefly discuss an alternative Hessian approximation approach for mean-field VI beside the generalized Gauss-Newton and gradient magnitude which are discussed in the main paper. This approach is based on the reparameterization trick for expectation of function over a Gaussian distribution. By using the identity in (45) for the mean-field case, we can derive a Hessian approximation using the reparameterization trick as follows:

$$\begin{aligned}
\mathbb{E}_q\left[\nabla^2_{\theta\theta} f(\boldsymbol{\theta})\right] &= 2\nabla_{\sigma^2}\mathbb{E}_q[f(\boldsymbol{\theta})], && (56)\\
&= 2\nabla_{\sigma^2}\mathbb{E}_{\mathcal{N}(\epsilon|0,I)}\left[f(\boldsymbol{\mu}+\boldsymbol{\sigma}\boldsymbol{\epsilon})\right] && (57)\\
&= 2\mathbb{E}_{\mathcal{N}(\epsilon|0,I)}\left[\nabla_{\sigma^2} f(\boldsymbol{\mu}+\boldsymbol{\sigma}\boldsymbol{\epsilon})\right] && (58)\\
&= \mathbb{E}_{\mathcal{N}(\epsilon|0,I)}\left[\nabla_{\theta} f(\boldsymbol{\theta})\boldsymbol{\epsilon}/\boldsymbol{\sigma}\right] && (59)\\
&\approx \hat{\mathbf{g}}(\boldsymbol{\theta})\left(\boldsymbol{\epsilon}/\boldsymbol{\sigma}\right), && (60)
\end{aligned}$$

where $\boldsymbol{\epsilon}\sim\mathcal{N}(\boldsymbol{\epsilon}|0,\mathbf{I})$ and $\boldsymbol{\theta}=\boldsymbol{\mu}+\boldsymbol{\sigma}\boldsymbol{\epsilon}$. By defining $\mathbf{s}_t := \boldsymbol{\sigma}_t^{-2}-\lambda$, we can write the VON update using the reparameterization trick Hessian approximation as

$$\text{VON (Reparam): } \boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t - \alpha_t\ \left(\hat{\mathbf{g}}(\boldsymbol{\theta}_t)+\tilde{\lambda}\boldsymbol{\mu}_t\right)/\left(\mathbf{s}_{t+1}+\tilde{\lambda}\right) \quad (61)$$

$$\mathbf{s}_{t+1} = (1-\beta_t)\mathbf{s}_t + \beta_t\ \left[\hat{\mathbf{g}}(\boldsymbol{\theta}_t)(\boldsymbol{\epsilon}_t/\boldsymbol{\sigma}_t)\right], \quad (62)$$

where $\boldsymbol{\epsilon}_t\sim\mathcal{N}(\boldsymbol{\epsilon}|0,\mathbf{I})$ and $\boldsymbol{\theta}_t=\boldsymbol{\mu}_t+\boldsymbol{\epsilon}_t/\sqrt{N(\mathbf{s}_t+\tilde{\lambda})}$.

One major issue with this approximation is that it might have high variance and $\mathbf{s}_t$ may be negative. To make sure that $\mathbf{s}_t>0$ for all t, 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 $\hat{\mathbf{g}}(\boldsymbol{\theta})\left(\boldsymbol{\epsilon}/\boldsymbol{\sigma}\right)$. 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

$$\beta_t = \min\left\{\beta_0, \delta\min_{d\in\mathcal{I}}\frac{s_d}{|h_d|}\right\}, \quad (63)$$

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). However, this back-tracking method may be computationally expensive and is not trivial to implement within the RMSProp and Adam optimizers.

E. Derivation of Vadam

E.1. Adam as an adaptive heavy-ball method

Consider the following update of Adam (in the pseudocode in main text, we used $\gamma_2=1-\beta$):

$$\begin{aligned}
\text{Adam: } \mathbf{u}_{t+1} &= \gamma_1\mathbf{u}_t + (1-\gamma_1)\hat{\mathbf{g}}(\boldsymbol{\theta}_t)\\
\mathbf{s}_{t+1} &= (1-\beta)\mathbf{s}_t + \beta\ [\hat{\mathbf{g}}(\boldsymbol{\theta}_t)]^2\\
\hat{\mathbf{u}}_{t+1} &= \mathbf{u}_{t+1}/(1-\gamma_1^t)\\
\hat{\mathbf{s}}_{t+1} &= \mathbf{s}_{t+1}/(1-(1-\beta)^t)\\
\boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t - \alpha\ \hat{\mathbf{u}}_{t+1}/(\sqrt{\hat{\mathbf{s}}_{t+1}}+\delta),
\end{aligned} \quad (64)$$

This update can be expressed as the following *adaptive* version of the Polyak's heavy ball[8] method,

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \tilde{\alpha}_t\left[\frac{1}{\sqrt{\hat{\mathbf{s}}_{t+1}}+\delta}\right]\nabla_\theta f(\boldsymbol{\theta}_t) + \tilde{\gamma}_t\left[\frac{\sqrt{\hat{\mathbf{s}}_t}+\delta}{\sqrt{\hat{\mathbf{s}}_{t+1}}+\delta}\right](\boldsymbol{\theta}_t-\boldsymbol{\theta}_{t-1}), \quad (65)$$

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

where $\tilde{\alpha}_t, \tilde{\gamma}_t$ are appropriately defined in terms of γ_1 as shown below:

$$\tilde{\alpha}_t := \alpha \frac{1-\gamma_1}{1-\gamma_1^t}, \quad \tilde{\gamma}_t := \gamma_1 \frac{1-\gamma_1^{t-1}}{1-\gamma_1^t} \tag{66}$$

We will now show that, by using natural gradients in the Polyak's heavy ball, we get an update that is similar to (65). This allows us to implement our approximated NGVI methods by using Adam.

E.2. Natural Momentum for Natural Gradient VI

We propose the following update:

$$\boldsymbol{\eta}_{t+1} = \boldsymbol{\eta}_t + \bar{\alpha}_t \widetilde{\nabla}_\eta \mathcal{L}_t + \bar{\gamma}_t(\boldsymbol{\eta}_t - \boldsymbol{\eta}_{t-1}), \tag{67}$$

We can show that (67) can be written as the following mirror descent extension of (27),

$$\mathbf{m}_{t+1} = \arg\min_{\boldsymbol{m}} \ \langle \mathbf{m}, -\nabla_m \mathcal{L}_*(\mathbf{m}_t) \rangle + \frac{1}{\beta_t} \mathbb{D}_{KL}[q_m(\boldsymbol{\theta}) \,\|\, q_{m_t}(\boldsymbol{\theta})] - \frac{\alpha_t}{\beta_t} \mathbb{D}_{KL}[q_m(\boldsymbol{\theta}) \,\|\, q_{m_{t-1}}(\boldsymbol{\theta})], \tag{68}$$

where $\mathcal{L}_*$ refers to the variational lower-bound defined in Appendix B, and α_t and β_t are two learning rates defined in terms of $\bar{\alpha}_t$ and $\bar{\gamma}_t$. The last term here is a *natural momentum* term, which is very similar to the momentum term in the heavy-ball methods. For example, (16) can be written as the following optimization problem:

$$\min_\theta \boldsymbol{\theta}^T \nabla_\theta f_1(\boldsymbol{\theta}_t) + \frac{1}{\beta_t} \|\boldsymbol{\theta} - \boldsymbol{\theta}_t\|^2 - \frac{\alpha_t}{\beta_t} \|\boldsymbol{\theta} - \boldsymbol{\theta}_{t-1}\|^2. \tag{69}$$

In our natural-momentum method, the Euclidean distance is replace by a KL divergence, which explains the name *natural*-momentum.

Equivalence between (68) to (67) can be established by directly taking the derivative, setting it to zero, and simplifying:

$$-\nabla_m \mathcal{L}_*(\mathbf{m}_t) + \frac{1}{\beta_t}(\boldsymbol{\eta}_{t+1} - \boldsymbol{\eta}_t) - \frac{\alpha_t}{\beta_t}(\boldsymbol{\eta}_{t+1} - \boldsymbol{\eta}_{t-1}) = 0 \tag{70}$$

$$\Rightarrow \boldsymbol{\eta}_{t+1} = \frac{1}{1-\alpha_t}\boldsymbol{\eta}_t + \frac{\beta_t}{1-\alpha_t}\nabla_m \mathcal{L}_*(\mathbf{m}_t) - \frac{\alpha_t}{1-\alpha_t}\boldsymbol{\eta}_{t-1}, \tag{71}$$

$$= \boldsymbol{\eta}_t + \frac{\beta_t}{1-\alpha_t}\nabla_m \mathcal{L}_*(\mathbf{m}_t) + \frac{\alpha_t}{1-\alpha_t}(\boldsymbol{\eta}_t - \boldsymbol{\eta}_{t-1}), \tag{72}$$

where we use the fact that gradient of the KL divergence with respect to $\mathbf{m}$ is qual to the difference between the natural parameters of the distributions (Raskutti & Mukherjee, 2015; Khan & Lin, 2017). Noting that the gradient with respect to $\mathbf{m}$ is the natural-gradient with respect to $\boldsymbol{\eta}$. Therefore, defining $\bar{\alpha}_t := \beta_1/(1-\alpha_t)$ and $\bar{\gamma}_t := \alpha_t/(1-\alpha_t)$, we establish that mirror-descent is equivalent to the natural-momentum approach we proposed.

E.3. NGVI with Natural Momentum for Gaussian Approximations

We will now derive the update for a Gaussian approximation $q(\boldsymbol{\theta}) := \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

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 $\mathbf{M}^{(2)} = \boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T$. Similarly to Appendix C, By using the chain rule, we can express the gradient $\nabla_m \mathcal{L}_*$ in terms of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ as

$$\nabla_{m^{(1)}} \mathcal{L}_* = \nabla_\mu \mathcal{L} - 2\left[\nabla_\Sigma \mathcal{L}\right]\boldsymbol{\mu}, \tag{73}$$

$$\nabla_{M^{(2)}} \mathcal{L}_* = \nabla_\Sigma \mathcal{L}. \tag{74}$$

Using the natural parameters of a Gaussian defined as $\boldsymbol{\eta}^{(1)} = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$ and $\boldsymbol{\eta}^{(2)} = -\frac{1}{2}\boldsymbol{\Sigma}^{-1}$, we can rewrite the update (72) in terms of the update for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. First, the update for $\boldsymbol{\Sigma}$ is obtained by plugging $\boldsymbol{\eta}^{(2)} = -\frac{1}{2}\boldsymbol{\Sigma}^{-1}$ into (72):

$$\boldsymbol{\Sigma}_{t+1}^{-1} = \frac{1}{1-\alpha_t}\boldsymbol{\Sigma}_t^{-1} - \frac{\alpha_t}{1-\alpha_t}\boldsymbol{\Sigma}_{t-1}^{-1} - \frac{2\beta_t}{1-\alpha_t}\left[\nabla_\Sigma \mathcal{L}_t\right]. \tag{75}$$

Now, for $\boldsymbol{\mu}$, we first plugging $\boldsymbol{\eta}^{(1)} = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$ into (72) and then rearrange the update to express some of the terms as $\boldsymbol{\Sigma}_{t+1}$:

$$\boldsymbol{\Sigma}_{t+1}^{-1}\boldsymbol{\mu}_{t+1} = \frac{1}{1-\alpha_t}\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\mu}_t - \frac{\alpha_t}{1-\alpha_t}\boldsymbol{\Sigma}_{t-1}^{-1}\boldsymbol{\mu}_{t-1} + \frac{\beta_t}{1-\alpha_t}\left(\nabla_\mu \mathcal{L}_t - 2\left[\nabla_\Sigma \mathcal{L}_t\right]\boldsymbol{\mu}_t\right) \tag{76}$$

$$= \frac{1}{1-\alpha_t}\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\mu}_t - \frac{\alpha_t}{1-\alpha_t}\boldsymbol{\Sigma}_{t-1}^{-1}\boldsymbol{\mu}_{t-1} + \frac{\beta_t}{1-\alpha_t}\left(\nabla_\mu \mathcal{L}_t - 2\left[\nabla_\Sigma \mathcal{L}_t\right]\boldsymbol{\mu}_t\right) + \frac{\alpha_t}{1-\alpha_t}\left(\boldsymbol{\Sigma}_{t-1}^{-1}\boldsymbol{\mu}_t - \boldsymbol{\Sigma}_{t-1}^{-1}\boldsymbol{\mu}_t\right) \tag{77}$$

$$= \left[\frac{1}{1-\alpha_t}\boldsymbol{\Sigma}_t^{-1} - \frac{\alpha_t}{1-\alpha_t}\boldsymbol{\Sigma}_{t-1}^{-1} - \frac{2\beta_t}{1-\alpha_t}\left[\nabla_\Sigma \mathcal{L}_t\right]\right]\boldsymbol{\mu}_t + \frac{\beta_t}{1-\alpha_t}\nabla_\mu \mathcal{L}_t + \frac{\alpha_t}{1-\alpha_t}\boldsymbol{\Sigma}_{t-1}^{-1}(\boldsymbol{\mu}_t - \boldsymbol{\mu}_{t-1}) \tag{78}$$

$$\Rightarrow \boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + \frac{\beta_t}{1-\alpha_t}\boldsymbol{\Sigma}_{t+1}\left[\nabla_\mu \mathcal{L}_t\right] + \frac{\alpha_t}{1-\alpha_t}\boldsymbol{\Sigma}_{t+1}\boldsymbol{\Sigma}_{t-1}^{-1}(\boldsymbol{\mu}_t - \boldsymbol{\mu}_{t-1}), \tag{79}$$

where in the final step, we substitute the definition of $\boldsymbol{\Sigma}_{t+1}^{-1}$ from (75).

To express these updates similar to VON, we make an approximation where we replace the instances of $\boldsymbol{\Sigma}_{t-1}$ by $\boldsymbol{\Sigma}_t$ in both (75) and (79). With this approximation, we get the following:

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + \frac{\beta_t}{1-\alpha_t}\boldsymbol{\Sigma}_{t+1}\left[\nabla_\mu \mathcal{L}_t\right] + \frac{\alpha_t}{1-\alpha_t}\boldsymbol{\Sigma}_{t+1}\boldsymbol{\Sigma}_t^{-1}(\boldsymbol{\mu}_t - \boldsymbol{\mu}_{t-1}), \tag{80}$$

$$\boldsymbol{\Sigma}_{t+1}^{-1} = \boldsymbol{\Sigma}_t^{-1} - \frac{2\beta_t}{1-\alpha_t}\left[\nabla_\Sigma \mathcal{L}_t\right]. \tag{81}$$

We build upon this update to express it as VON update with momentum.

E.4. Variational Online Newton with Natural Momentum

Now, we derive VON with natural momentum. To do so, we follow the same procedure used to derive the VON update in Section D. 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 $\boldsymbol{\theta}_t \sim \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$. Finally, we redefine the matrix $\mathbf{S}_t := (\boldsymbol{\Sigma}_t^{-1} - \lambda\mathbf{I})/N$. With this we get the following update which is a momentum version of VON:

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t - \frac{\beta_t}{1-\alpha_t}\left(\mathbf{S}_{t+1} + \tilde{\lambda}\mathbf{I}\right)^{-1}\left(\mathbf{g}(\boldsymbol{\theta}_t) + \tilde{\lambda}\boldsymbol{\mu}_t\right) + \frac{\alpha_t}{1-\alpha_t}(\mathbf{S}_{t+1} + \tilde{\lambda}\mathbf{I})^{-1}(\mathbf{S}_t + \tilde{\lambda}\mathbf{I})(\boldsymbol{\mu}_t - \boldsymbol{\mu}_{t-1}), \tag{82}$$

$$\mathbf{S}_{t+1} = \left(1 - \frac{\beta_t}{1-\alpha_t}\right)\mathbf{S}_t + \frac{\beta_t}{1-\alpha_t}\,\mathbf{H}(\boldsymbol{\theta}_t), \tag{83}$$

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

To get a momentum version of Vprop, we follow a similar method to Section 3. That is, we first employ a mean-field approximation, and then replace the Hessian by the gradient-magnitude approximation. Doing so gives us

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t - \frac{\beta_t}{1-\alpha_t}\left[\frac{1}{\mathbf{s}_{t+1} + \tilde{\lambda}}\right]\left(\mathbf{g}(\boldsymbol{\theta}_t) + \tilde{\lambda}\boldsymbol{\mu}_t\right) + \frac{\alpha_t}{1-\alpha_t}\left[\frac{\mathbf{s}_t + \tilde{\lambda}}{\mathbf{s}_{t+1} + \tilde{\lambda}}\right](\boldsymbol{\mu}_t - \boldsymbol{\mu}_{t-1}), \tag{84}$$

$$\mathbf{s}_{t+1} = \left(1 - \frac{\beta_t}{1-\alpha_t}\right)\mathbf{s}_t + \frac{\beta_t}{1-\alpha_t}\,\left[\mathbf{g}(\boldsymbol{\theta}_t)\right]^2, \tag{85}$$

where $\boldsymbol{\theta}_t \sim \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t^2)$ with $\boldsymbol{\sigma}_t^2 = 1/[N(\mathbf{s}_t + \tilde{\lambda})]$. Finally, we use an unbiased gradient estimate $\hat{\mathbf{g}}(\boldsymbol{\theta})$, introduce the square-root for the scaling vector in the mean update, and define step-sizes $\tilde{\alpha}_t := \beta_t/(1-\alpha_t)$ and $\tilde{\gamma}_t := \alpha_t/(1-\alpha_t)$. The result is a *Vprop with momentum* update:

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t - \tilde{\alpha}_t\left[\frac{1}{\sqrt{\mathbf{s}_{t+1}} + \tilde{\lambda}}\right]\left(\hat{\mathbf{g}}(\boldsymbol{\theta}_t) + \tilde{\lambda}\boldsymbol{\mu}_t\right) + \tilde{\gamma}_t\left[\frac{\sqrt{\mathbf{s}_t} + \tilde{\lambda}}{\sqrt{\mathbf{s}_{t+1}} + \tilde{\lambda}}\right](\boldsymbol{\mu}_t - \boldsymbol{\mu}_{t-1}), \tag{86}$$

$$\mathbf{s}_{t+1} = (1-\tilde{\alpha}_t)\,\mathbf{s}_t + \tilde{\alpha}_t\,\left[\hat{\mathbf{g}}(\boldsymbol{\theta}_t)\right]^2, \tag{87}$$

where $\boldsymbol{\theta}_t \sim \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t^2)$ with $\boldsymbol{\sigma}_t^2 = 1/[N(\mathbf{s}_t + \tilde{\lambda})]$. This is very similar to the update (65) of Adam expressed in the momentum form. By introducing the bias correction term for $\mathbf{m}$ and $\mathbf{s}$, we can implement this update by using Adam's update shown in Fig. 1. The final update of Vadam is shown below, where we highlight the differences from Adam in red.

$$\boldsymbol{\theta}_t \sim \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_t, 1/[N(\mathbf{s}_t + \tilde{\lambda})]), \tag{88}$$

$$\mathbf{u}_{t+1} = \gamma_1 \mathbf{u}_t + (1-\gamma_1)\left(\hat{\mathbf{g}}(\boldsymbol{\theta}_t) + \tilde{\lambda}\boldsymbol{\mu}_t\right) \tag{89}$$

$$\mathbf{s}_{t+1} = (1-\beta)\mathbf{s}_t + \beta\ [\hat{\mathbf{g}}(\boldsymbol{\theta}_t)]^2 \tag{90}$$

$$\hat{\mathbf{u}}_{t+1} = \mathbf{u}_{t+1}/(1-\gamma_1^t) \tag{91}$$

$$\hat{\mathbf{s}}_{t+1} = \mathbf{s}_{t+1}/(1-(1-\beta)^t) \tag{92}$$

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t - \alpha\ \hat{\mathbf{u}}_{t+1}/(\sqrt{\hat{\mathbf{s}}_{t+1}} + \tilde{\lambda}). \tag{93}$$

Note that we do not use the same step-size $\tilde{\alpha}_t$ for $\mathbf{s}_t$ and $\boldsymbol{\mu}_t$, but rather choose the step-sizes according to the Adam update. In the pseudocode, we define $\gamma_2 = 1-\beta$.

F. The VadaGrad Update

By setting $\tau = 0$ in (23), we get the following update:

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t - \alpha_t\ \left[\widehat{\nabla}_\theta F(\boldsymbol{\theta})/\mathbf{s}_{t+1}\right], \tag{94}$$

$$\mathbf{s}_{t+1} = \mathbf{s}_t + \beta_t\ \widehat{\nabla}^2_{\theta\theta} F(\boldsymbol{\theta}), \tag{95}$$

where $\boldsymbol{\theta}_t \sim \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t^2)$ with $\boldsymbol{\sigma}_t^2 := 1/\mathbf{s}_t$. By replacing the Hessian by a GM approximation, and taking the square-root as in Vprop, we get the following update we call VadaGrad:

$$\begin{aligned}\boldsymbol{\mu}_{t+1} &= \boldsymbol{\mu}_t - \alpha_t\ \left[\widehat{\nabla}_\theta F(\boldsymbol{\theta})/\sqrt{\mathbf{s}_{t+1}}\right], \\ \mathbf{s}_{t+1} &= \mathbf{s}_t + \beta_t\ \left[\widehat{\nabla}_\theta F(\boldsymbol{\theta}) \circ \widehat{\nabla}_\theta F(\boldsymbol{\theta})\right],\end{aligned} \tag{96}$$

where $\boldsymbol{\theta}_t \sim \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t^2)$ with $\boldsymbol{\sigma}_t^2 := 1/\mathbf{s}_t$.

G. Proof of Theorem 1

Let $\mathbf{g}_i := \nabla_\theta f_i(\boldsymbol{\theta})$ denote the gradient for an individual data point, $\mathbf{g}_\mathcal{M} := \frac{1}{M}\sum_{i\in\mathcal{M}} \nabla_\theta f_i(\boldsymbol{\theta})$ denote the average gradient over a minibatch of size M and $\mathbf{g} = \frac{1}{N}\sum_{i=1}^N \nabla_\theta f_i(\boldsymbol{\theta})$ denote the average full-batch gradient. Let $p(i)$ denote a uniform distribution over the data samples $\{1, 2, ..., N\}$ and $p(\mathcal{M})$ a uniform distribution over the $\binom{N}{M}$ possible minibatches of size M. Let further $\mathbf{G}$ denote the average GGN matrix,

$$\mathbf{G} = \frac{1}{N}\sum_{i=1}^N \mathbf{g}_i\mathbf{g}_i^T = \mathbb{E}_{p(i)}[\mathbf{g}_i\mathbf{g}_i^T]. \tag{97}$$

Using the following two results,

$$\mathrm{Cov}_{p(i)}[\mathbf{g}_i] = \mathbb{E}_{p(i)}[\mathbf{g}_i\mathbf{g}_i^T] - \mathbb{E}_{p(i)}[\mathbf{g}_i]\mathbb{E}_{p(i)}[\mathbf{g}_i]^T = \mathbf{G} - \mathbf{g}\mathbf{g}^T, \tag{98}$$

$$\mathrm{Cov}_{p(\mathcal{M})}[\mathbf{g}_\mathcal{M}] = \mathbb{E}_{p(\mathcal{M})}[\mathbf{g}_\mathcal{M}\mathbf{g}_\mathcal{M}^T] - \mathbb{E}_{p(\mathcal{M})}[\mathbf{g}_\mathcal{M}]\mathbb{E}_{p(\mathcal{M})}[\mathbf{g}_\mathcal{M}]^T = \mathbb{E}_{p(\mathcal{M})}[\mathbf{g}_\mathcal{M}\mathbf{g}_\mathcal{M}^T] - \mathbf{g}\mathbf{g}^T, \tag{99}$$

along with Theorem 2.2 of Cochran (1977) which states that

$$\mathrm{Cov}_{p(\mathcal{M})}[\mathbf{g}_\mathcal{M}] = \frac{1-\frac{M}{N}}{M}\frac{N}{N-1}\mathrm{Cov}_{p(i)}[\mathbf{g}_i], \tag{100}$$

we get the following:

$$\mathbb{E}_{p(\mathcal{M})}[\mathbf{g}_\mathcal{M}\mathbf{g}_\mathcal{M}^T] = w\mathbf{G} + (1-w)\mathbf{g}\mathbf{g}^T, \tag{101}$$

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

Denoting dimension j of the full-batch gradient by $\mathbf{g}_j(\boldsymbol{\theta})$, dimension j of the average gradient over a minibatch by $\hat{\mathbf{g}}_j(\boldsymbol{\theta}; \mathcal{M})$ and dimension j of the diagonal of the average GGN, we get the stated result.

H. Proof to Show That Fixed-Point of Vprop Do Not Change with Square-root

We now show that the fixed-points do not change when we take the square root of $\mathbf{s}_{t+1}$. Denote the variational distribution at iteration t by $q_t := \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t^2)$. Assume no stochasticity, i.e., we compute the full-batch gradients and also can exactly compute the expectation with respect to q.

A fixed point $q_*(\boldsymbol{\theta}) := \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_*, \boldsymbol{\sigma}_*^2)$ of the variational objective satisfies the following:

$$N\mathbb{E}_{q_*}\left[\nabla_\theta f(\boldsymbol{\theta})\right] + \lambda\boldsymbol{\mu}_* = 0, \quad N\mathbb{E}_{q_*}\left[\text{diag}\left(\nabla^2_{\theta\theta} f(\boldsymbol{\theta})\right)\right] + \lambda - \boldsymbol{\sigma}_*^2 = 0, \tag{102}$$

If we replace the Hessian by the GM approximation, we get the following fixed-point:

$$N\mathbb{E}_{q_*}\left[\nabla_\theta f(\boldsymbol{\theta})\right] + \lambda\boldsymbol{\mu}_* = 0, \quad N\mathbb{E}_{q_*}\left[\left(\nabla_\theta f(\boldsymbol{\theta})\right)^2\right] + \lambda - \boldsymbol{\sigma}_*^2 = 0, \tag{103}$$

This fixed-point does not depend on the fact whether we scale by using the square-root or not. However, the iterations do depend on it and the scaling is expected to affect the convergence and also the path that we take to approach the solution.

I. Details for the Logistic Regression Experiments

I.1. Toy Example

We used the toy example given in Murphy (2012) (see Fig. 8.6 in the book). The data is generated from a mixture of two Gaussians (details are given in the book). We used the generating mechansim desribed in the book to generate $N = 60$ examples. For all methods, a prior precision of $\lambda = 0.01$ and 1 MC sample is used. The initial settings of all methods are $\alpha_0 = 0.1$ and $\beta_0 = 0.9$. For every iteration t, the learning rates are decayed as

$$\alpha_t = \frac{\alpha_0}{1 + t^{0.55}}, \qquad \beta_t = 1 - \frac{1 - \beta_0}{1 + t^{0.55}}. \tag{104}$$

Vadam and VOGN are run for 83,333 epochs using a minibatch size of $M = 10$ (corresponding to 500,000 iterations). VOGN-1 is run for 8000 epochs with a minibatch size of $M = 1$ (also corresponding to 500,000 iterations).

I.2. Real-Data Experiments

Datasets for logistic regression are available at `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html`. For the Breast Cancer dataset, we use the hyper-parameters found by Khan & Lin (2017). For USPS, we used the procedure of Khan & Lin (2017) to find the hyperparameter. All details are given in Table 2. For all datasets we use 20 random splits.

Table 2. Datasets for logistic regression. N_{Train} is the number of training data.

Dataset	N	D	N_{Train}	Hyperparameters	M
USPS3vs5	1,781	256	884	$\lambda = 25$	64
Breast-cancer-scale	683	10	341	$\lambda = 1.0$	32

Performance comparison of MF-Exact, VOGN-1, Vadam: We used 20 random 50-50 splits of the USPS 3vs5 dataset. For all methods, a prior precision of $\lambda = 25$ is used. MF-Exact and Vadam are run for 10000 epochs with a minibatch size of $M = 64$. The learning rates for both methods are decayed according to (104) with initial settings $\alpha_0 = 0.01$ and $\beta_0 = 0.99$. For Vadam, 1 MC sample is used. VOGN-1, on the other hand, is run for 200 epochs with a minibatch size of $M = 1$, using 1 MC sample and learning rates $\alpha = 0.0005$ and $\beta = 0.9995$.

Minibatch experiment comparing VOGN-1 and Vadam: We use the Breast-Cancer dataset with 20 random initializations. For both VOGN-1 and Vadam, a prior precision of $\lambda = 1$ is used. For VOGN-1, the learning rates are set to $\alpha = 0.0005$ and $\beta = 0.9995$. It is run for 2000 epochs using a minibatch size of $M = 1$ and 1 MC sample. For Vadam, the learning rates are decayed according to (104) with initial settings $\alpha_0 = 0.01$ and $\beta_0 = 0.99$. The method is run with 1 MC sample for various minibatch sizes $M \in \{1, 8, 16, 32, 64\}$.

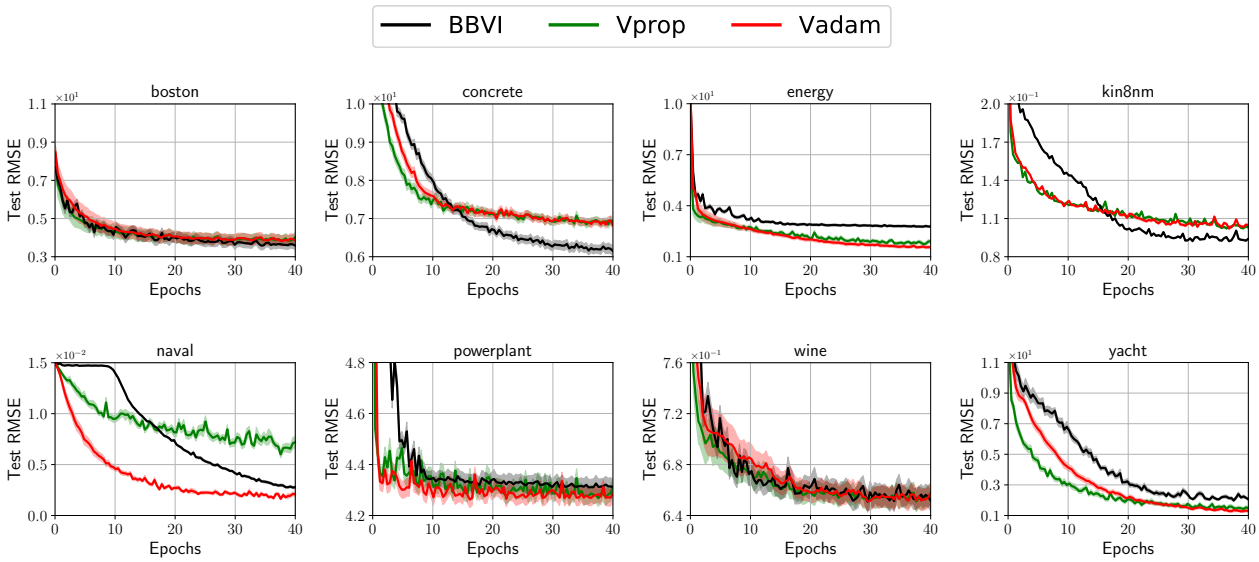

Figure 4. The mean plus-minus one standard error of the Test RMSE (using 100 Monte Carlo samples) on the test sets of UCI experiments. The mean and standard errors are computed over the 20 data splits.

J. Details for the Bayesian Neural Network Experiment

The 8 datasets together with their sizes N and number of features D are listed in Table 1. For each of the datasets, we use the 20 random train-test splits provided by Gal & Ghahramani (2016). Following earlier work, we use 30 iterations of Bayesian Optimization (BO) to tune the prior precision λ and the noise precision τ. For each iteration of BO, 5-fold cross-validation is used to evaluate the considered hyperparameter setting. This is repeated for each of the 20 train-test splits for each dataset. The final values reported in the table for each dataset are the mean and standard error from these 20 runs. The final runs for the 8 datasets are shown in Figure 4.

Following earlier work, we use neural networks with one hidden layer and 50 hidden units with ReLU activation functions. All networks were trained for 40 epochs. For the 4 smallest datasets, we use a minibatch size of 32, 10 MC samples for Vadam and 20 MC samples for BBVI. For the 4 larger datasets, we use a minibatch size of 128, 5 MC samples for Vadam and 10 MC samples for BBVI. For evaluation, 100 MC samples were used in all cases.

For BBVI, we optimize the variational objective using the Adam optimizer. For both BBVI and Vadam we use a learning rate of $\alpha = 0.01$ and set $\beta = 0.99$ and $\gamma = 0.9$ to encourage convergence within 40 epochs. For both BBVI and Vadam, the initial precision of the variational distribution q was set to 10.

K. Details for the Exploration for Deep Reinforcement Learning Experiment

Reinforcement learning (RL) aims to solve the sequential decision making problem where at each discrete time step t an agent observes a state $\mathbf{s}_t$ and selects an action $\mathbf{a}_t$ using a policy π, i.e., $\mathbf{a}_t \sim \pi(\mathbf{a}|\mathbf{s}_t)$. The agent then receives an immediate reward $r_t = r(\mathbf{s}_t, \mathbf{a}_t)$ and observes a next state $\mathbf{s}_t \sim p(\mathbf{s}'|\mathbf{s}_t, \mathbf{a}_t)$. The goal in RL is to learn the optimal policy π^* which maximizes the expected return $\mathbb{E}\left[\sum_t^\infty \gamma^{t-1} r_t\right]$ where γ is the discounted factor and the expectation is taken over a sequence of densities $\pi(\mathbf{a}|\mathbf{s}_t)$ and $p(\mathbf{s}'|\mathbf{s}_t, \mathbf{a}_t)$.

A central component of RL algorithms is the Q-function, $Q^\pi(\mathbf{s}, \mathbf{a})$, which denotes the expected return after executing an action $\mathbf{a}$ in a state $\mathbf{s}$ and following the policy π afterwards. Formally, the Q-function is defined as $Q^\pi(\mathbf{s}, \mathbf{a}) = \mathbb{E}\left[\sum_{t=1}^\infty \gamma^{t-1} r_t | \mathbf{s}_1 = \mathbf{s}, \mathbf{a}_1 = \mathbf{a}\right]$. The Q-function also satisfies a recursive relation also known as the Bellman equation: $Q^\pi(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{p(\mathbf{s}'|\mathbf{s},\mathbf{a})\pi(\mathbf{a}'|\mathbf{s}')}\left[Q^\pi(\mathbf{s}', \mathbf{a}')\right]$. Using the Q-function and a parameterized policy π_θ, the goal of

reinforcement learning can be simply stated as finding a policy parameter $\boldsymbol{\theta}$ which maximizes the expected Q-function[9]:

$$\max_{\boldsymbol{\theta}} \mathbb{E}_{p(\mathbf{s})\pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{s})} \left[Q^{\pi}(\mathbf{s}, \mathbf{a}) \right]. \tag{105}$$

In practice, the Q-function is unknown and is commonly approximated by a parameterized function $\widehat{Q}_{\omega}(\mathbf{s}, \mathbf{a})$ with parameter $\boldsymbol{\omega}$ learned such that it satisfies the Bellman equation on average: $\min_{\boldsymbol{\omega}} \mathbb{E}_{p(\mathbf{s})\beta(\mathbf{a}|\mathbf{s})}[(r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\pi_{\boldsymbol{\theta}}(\mathbf{a}'|\mathbf{s}')}[\widehat{Q}_{\tilde{\boldsymbol{\omega}}}(\mathbf{s}', \mathbf{a}')] - \widehat{Q}_{\boldsymbol{\omega}}(\mathbf{s}, \mathbf{a}))^2]$, where $\beta(\mathbf{a}|\mathbf{s})$ is a behavior policy used to collect samples and $\tilde{\boldsymbol{\omega}}$ is either a copy or a slowly updated value of $\boldsymbol{\omega}$ whose $\nabla_{\boldsymbol{\omega}} \widehat{Q}_{\tilde{\boldsymbol{\omega}}}(\mathbf{s}', \mathbf{a}') = 0$. By using an approximated Q-function, the goal of RL is to find a policy parameter maximizing the expected value of $\widehat{Q}_{\boldsymbol{\omega}}(\mathbf{s}, \mathbf{a})$:

$$\max_{\boldsymbol{\theta}} \mathbb{E}_{p(\mathbf{s})\pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{s})} \left[\widehat{Q}_{\boldsymbol{\omega}}(\mathbf{s}, \mathbf{a}) \right] := \min_{\boldsymbol{\theta}} -\mathbb{E}_{p(\mathbf{s})\pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{s})} \left[\widehat{Q}_{\boldsymbol{\omega}}(\mathbf{s}, \mathbf{a}) \right] := \min_{\boldsymbol{\theta}} F(\boldsymbol{\theta}). \tag{106}$$

In the remainder, we consider the minimization problem $\min_{\boldsymbol{\theta}} F(\boldsymbol{\theta})$ to be consistent with the variational optimization problem setting in the main text.

K.1. Stochastic Policy Gradient and Deterministic Policy Gradient

The RL objective in (106) is often minimized by gradient descent. The gradient computation depends on stochasticity of $\pi_{\boldsymbol{\theta}}$. For a stochastic policy, $\pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{s})$, *policy gradient* or REINFORCE can be computed using the likelihood ratio trick:

$$F(\boldsymbol{\theta}) = -\mathbb{E}_{p(\mathbf{s})\pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{s})} \left[\widehat{Q}_{\boldsymbol{\omega}}(\mathbf{s}, \mathbf{a}) \right], \quad \nabla_{\boldsymbol{\theta}} F(\boldsymbol{\theta}) = -\mathbb{E}_{p(\mathbf{s})\pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{s})} \left[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{s}) \widehat{Q}_{\boldsymbol{\omega}}(\mathbf{s}, \mathbf{a}) \right]. \tag{107}$$

For a deterministic policy $\pi_{\boldsymbol{\theta}}(\mathbf{s})$, *deterministic policy gradient* (DPG) (Silver et al., 2014) can be computed using the chain-rule:

$$F(\boldsymbol{\theta}) = -\mathbb{E}_{p(\mathbf{s})} \left[\widehat{Q}_{\boldsymbol{\omega}}(\mathbf{s}, \pi_{\boldsymbol{\theta}}(\mathbf{s})) \right], \quad \nabla_{\boldsymbol{\theta}} F(\boldsymbol{\theta}) = -\mathbb{E}_{p(\mathbf{s})} \left[\nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(\mathbf{s}) \nabla_{\mathbf{a}} \widehat{Q}_{\boldsymbol{\omega}}(\mathbf{s}, \pi_{\boldsymbol{\theta}}(\mathbf{s})) \right]. \tag{108}$$

As discussed by Silver et al. (2014), the deterministic policy gradient is more advantageous than the stochastic counter part due to its lower variance. However, the issue of a deterministic policy is that it does not perform *exploration* by itself. In practice, exploration is done by injecting a noise to the policy output, i.e., $\mathbf{a} = \pi_{\boldsymbol{\theta}}(\mathbf{s}) + \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon}$ is a noise from some random process such as Gaussian noise. However, action-space noise may be insufficient in some problems (Rückstieß et al., 2010). Next, we discussed parameter-based exploration approach where exploration is done in the *parameter space*. Then, we show that such exploration can be achieved by simply applying VadaGrad and Vadam to policy gradient methods.

K.2. Parameter-based Exploration Policy Gradient

Parameter-based exploration policy gradient (Rückstieß et al., 2010) relaxes the RL objective in (106) by assuming that the parameter $\boldsymbol{\theta}$ is sampled from a Gaussian distribution $q(\boldsymbol{\theta}) := \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ with a diagonal covariance. Formally, it solves an optimization problem

$$\min_{\boldsymbol{\mu}, \boldsymbol{\sigma}^2} \ \mathbb{E}_{\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\sigma}^2)}[F(\boldsymbol{\theta})], \tag{109}$$

where $F(\boldsymbol{\theta})$ is either the objective function for the stochastic policy in (107) or the deterministic policy in (108). In each time step, the agent samples a policy parameter $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ and uses it to determine an action[10]. This exploration strategy is advantageous since the stochasticity of $\boldsymbol{\theta}$ allows the agent to exhibit much more richer explorative behaviors when compared with exploration by action noise injection.

Notice that (109) is exactly the variational optimization problem discussed in the main text. As explained in the main text, this problem can be solved by our methods. In the next section, we apply VadaGrad and Vadam to the deep deterministic policy gradient and show that a parameter-exploration strategy induced by our methods allows the agent to achieve a better performance when compared to existing methods.

[9] In this section, we omit a case where the state distribution $p(\mathbf{s})$ depends on the policy. In practice many policy gradient methods (especially actor-critic type methods) also often ignore the dependency between the state distribution and the policy.

[10] The original work of (Rückstieß et al., 2010) considers an episode-based method where the policy parameter is sampled only at the start of an episode. However, we consider DPG which is a step-based method. Therefore, we sample the policy parameter in every time step. Note that we may only sample the policy parameter at the start of the episode as well.

K.3. Parameter-based Exploration Deep Deterministic Policy Gradient via VadaGrad and Vadam

While parameter-based exploration strategy can be applied to both stochastic and deterministic policies, it is commonly applied to a deterministic policy. In our experiment, we adopt a variant of deterministic policy gradient called *deep deterministic policy gradient* (DDPG) (Lillicrap et al., 2015). In DDPG, the policy $\pi_{\boldsymbol{\theta}}(\mathbf{s})$ and the Q-function $\widehat{Q}_{\boldsymbol{\omega}}(\mathbf{s},\mathbf{a})$ are represented by deep neural networks. To improve stability, DDPG introduces target networks, $\pi_{\tilde{\boldsymbol{\theta}}}(\mathbf{s})$ and $\widehat{Q}_{\tilde{\boldsymbol{\omega}}}(\mathbf{s},\mathbf{a})$, whose weight parameters are updated by $\tilde{\boldsymbol{\theta}} \leftarrow (1-\tau)\tilde{\boldsymbol{\theta}} + \tau\boldsymbol{\theta}$ and $\tilde{\boldsymbol{\omega}} \leftarrow (1-\tau)\tilde{\boldsymbol{\omega}} + \tau\boldsymbol{\omega}$ for $0 < \tau < 1$. The target network are used to update the Q-function by solving

$$\min_{\boldsymbol{\omega}} \mathbb{E}_{p(\mathbf{s})\beta(\mathbf{a}|\mathbf{s})}\left[\left(r(\mathbf{s},\mathbf{a}) + \gamma\widehat{Q}_{\tilde{\boldsymbol{\omega}}}(\mathbf{s}',\pi_{\tilde{\boldsymbol{\theta}}}(\mathbf{s}')) - \widehat{Q}_{\boldsymbol{\omega}}(\mathbf{s},\mathbf{a})\right)^2\right]. \tag{110}$$

Gradient descent on (110) yields an update: $\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} + \kappa\mathbb{E}_{p(\mathbf{s})\beta(\mathbf{a}|\mathbf{s})}\left[\left(r(\mathbf{s},\mathbf{a}) + \gamma\widehat{Q}_{\tilde{\boldsymbol{\omega}}}(\mathbf{s}',\pi_{\tilde{\boldsymbol{\theta}}}(\mathbf{s}')) - \widehat{Q}_{\boldsymbol{\omega}}(\mathbf{s},\mathbf{a})\right)\nabla_{\boldsymbol{\omega}}\widehat{Q}_{\boldsymbol{\omega}}(\mathbf{s},\mathbf{a})\right]$, where $\kappa > 0$ is the step-size. DDPG also uses a *replay buffer* which is a first-in-first-out queue that store past collected samples. In each update iteration, DDPG uniformly draws M minibatch training samples from the replay buffer to approximate the expectations.

To apply VadaGrad to solve (109), in each update iteration we sample $\boldsymbol{\theta}_t \sim \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t^2)$ and then updates the mean and variance using the VadaGrad update in (96). The deterministic policy gradient $\nabla_{\boldsymbol{\theta}} F(\boldsymbol{\theta})$ can be computed using the chain-rule as shown in (108). The computational complexity of DDPG with VadaGrad is almost identical to DDPG with Adagrad, except that we require gradient computation at sampled weight $\boldsymbol{\theta}_t$. Algorithm 1 below outlines our *parameter-based exploration DPG via VadaGrad* where the only difference from DDPG is the sampling procedure in line 3. Note that we use $N = 1$ in this case. Also note that the target policy network $\pi_{\tilde{\boldsymbol{\mu}}}$ does not performed parameter-exploration and it is updated by the mean $\boldsymbol{\mu}$ instead of a sampled weight $\boldsymbol{\theta}$.

In VadaGrad, the precision matrix always increases overtime and it guarantees that the policy eventually becomes deterministic. This is beneficial since it is known that there always exists a deterministic optimal policy for MDP. However, this behavior may not be desirable in practice since the policy may become deterministic too fast which leads to premature convergence. Moreover, for VadaGrad the effective gradient step-size, $\frac{\alpha}{\sqrt{\mathbf{s}_{t+1}}}$, will be close to zero for a nearly deterministic variational distribution q which leads to no policy improvement. This issue can be avoided by applying Vadam instead of VadaGrad. As will be shown in the experiment, Vadam allows the agent to keep explore and avoids the premature convergence issue. Note that parameter-based exploration RL is not a VI problem and we require some modification to the Vadam update, as shown in line 3 of Algorithm 2 below.

We perform experiment using the Half-Cheetah task from the OpenAI gym platform. We compare DDPG with **VadaGrad** and **Vadam** against four baseline methods.

- **SGD-Plain:** the original DDPG without any noise injection optimized by SGD,
- **Adam-Plain:** the original DDPG without any noise injection optimized by Adam,
- **SGD-Explore:** a naive parameter exploration DDPG based on VO optimized by SGD, and
- **Adam-Explore:** a naive parameter exploration DDPG based on VO optimized by Adam.

In SGD-Explore and Adam-Explore, we separately optimizes the mean and variance of the Gaussian distribution:

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t - \alpha\nabla_{\boldsymbol{\mu}}\mathbb{E}_{\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_t,\boldsymbol{\sigma}_t^2)}[F(\boldsymbol{\theta})], \qquad \boldsymbol{\sigma}_{t+1}^2 = \boldsymbol{\sigma}_t^2 - \alpha^{(\sigma)}\nabla_{\boldsymbol{\sigma}^2}\mathbb{E}_{\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_t,\boldsymbol{\sigma}_t^2)}[F(\boldsymbol{\theta})], \tag{111}$$

where $\alpha > 0$ is the mean step-size and $\alpha^{(\sigma)} > 0$ is the variance step-size. The gradients of $\mathbb{E}_{\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_t,\boldsymbol{\sigma}_t^2)}[F(\boldsymbol{\theta})]$ are computed by chain-rule and automatic-differentiation. For Adam-Explore, two Adam optimizers with different scaling vectors are used to independently update the mean and variance.

All methods use the DDPG network architectures as described by Lillicrap et al. (2015); two-layer neural networks with 400 and 300 ReLU hidden units. The output of the policy network is scaled by a hyperbolic tangent to bound the actions. The minibatch size is $M = 64$. All methods optimize the Q-network by Adam with step-size $\kappa = 10^{-3}$. The target Q-network and target policy network use a moving average step-size $\tau = 10^{-3}$. The expectation over the variational distribution of all

Method		α	γ_2	γ_1	$\alpha^{(\sigma)}$	$\gamma_2^{(\sigma)}$	$\gamma_1^{(\sigma)}$
VO/VI	VadaGrad	10^{-2}	0.99	-	-	-	-
	Vadam	10^{-4}	0.999	0.9	-	-	-
Plain	SGD-Plain	10^{-4}	-	-	-	-	-
	Adam-Plain	10^{-4}	0.999	0.9	-	-	-
Explore	SGD-Explore	10^{-4}	-	-	10^{-2}	-	-
	Adam-Explore	10^{-4}	0.999	0.9	10^{-2}	0.999	0.9

Table 3. Hyper-parameter setting for the deep reinforcement learning experiment. We refer to Algorithm 1 and 2 for the meaning of each hyper-parameter.

methods are approximated using one MC sample. For optimizing the policy network, we use the step-sizes given in Table 3. For Adam we use $\delta = 10^{-8}$. We also use the same value of $\lambda = 10^{-8}$ for Vadam. The initial precision for SGD-Explore, Adam-Explore, and VadaGrad is $\boldsymbol{\sigma}_{t=1}^{-2} = 10000$.

For Vadam we use $\mathbf{s}_{t=1} = 0$ for the initial second-order moment estimate and add a constant value of $c = 10000$ to the precision matrix $\boldsymbol{\sigma}_t^{-2} = \mathbf{s}_t + \lambda + c$ for sampling (see line 3 of Algorithm 2). We do this for two reasons. First, we set $\mathbf{s}_{t=1} = 0$ so that the initial conditions and hyper-parameters of Vadam and Adam-Plain are exactly the same. Second, we add a constant c to prevent an ill-conditioned precision matrix during sampling. Without this constant, the initial precision value of $\boldsymbol{\sigma}_{t=1}^{-2} = \lambda = 10^{-8}$ is highly ill-conditioned and sampled weights do not contain any information. This is highly problematic in RL since we do not have training samples at first and the agent needs to collect training samples from scratch. Training samples collected initially using $\boldsymbol{\sigma}_{t=1}^{-2} = 10^{-8}$ is highly uninformative (e.g., all actions are either the maximum or minimum action values) and the agent cannot correctly estimate uncertainty. We emphasize that the constant is only used for sampling and not used for gradient update. We expect that this numerical trick is not required for a large enough value of λ, but setting an appropriate value of λ is not trivial in deep learning and is not in the scope of this paper. As such, we leave finding a more appropriate approach to deal with this issue as a future work.

We perform experiment using the Half-Cheetah task from the OpenAI gym platform (Brockman et al., 2016). We measure the performance of each method by computing cumulative rewards along 20 test episodes without exploration. The early learning performance of Vadam, Adam-Plain, and Adam-Explore in Figure 5 shows that Vadam learns faster than the other methods. We conjecture that exploration through a variational distribution allows Vadam agent to collect more information training samples when compare to Adam-Plain. While Adam-Explore also performs exploration through a variational distribution, its performance is quite unstable with high fluctuations. This fluctuation in Vadam-Explore is likely because the mean and variance of the variational distribution are optimized independently. In contrast, Vadam uses natural-gradient to optimizes the two quantities in a strongly correlated manner, yielding a more stable performance.

Figure 6 shows the learning performance for a longer period of training for all methods. We can see that VadaGrad learns faster than SGD and Adam-based methods initially, but it suffers from a premature convergence and are outperformed by Adam-based methods. In contrast, Vadam does not suffer from the premature convergence. We can also observe that while Adam-based method learn slower than Vadam initially, they eventually catch up with Vadam and obtain a comparable performance at the 3 million time-steps. We conjecture that this is because exploration strategy is very important in the early stage of learning where the agent does not have sufficient amount of informative training samples. As learning progress and there is a sufficient amount of informative samples, exploration would not help much. Nonetheless, we can still see that Vadam and Adam-Explore give slightly better performance than Adam-Plain, showing that parameter-based exploration is still beneficial for DDPG.

L. Toy Example on Local-Minima Avoidance using Vadam

Fig. 7 shows an illustration of variational optimization on a two-dimensional objective function. The objective function $h(x, y) = \exp\{-(x\sin(20y) + y\sin(20x))^2 - (x\cos(10y) - y\sin(10x))^2\}$ is taken from Fig. 5.2 in Robert & Casella (2005). Variational optimization is performed by gradually turning off the KL-term for Vadam, thus annealing Vadam towards VadaGrad. This is referred to as "Vadam to VadaGrad". We show results for 4 multiple runs of each method started with a different initial values. The figure shows that variational optimization can better navigate the landscape to reach the flat (and global) minimum than gradient descent.

Algorithm 1 Parameter-based exploration DDPG via VadaGrad

Initialize: Variational distribution $\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_1, \mathbf{s}_1^{-1})$ with random initial mean and initial precision $\mathbf{s}_1 = 10000$.
for Time step $t = 1, ..., \infty$ **do**
  Sample policy parameter $\boldsymbol{\theta}_t \sim \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_t, \mathbf{s}_t^{-1})$.
  Observe $\mathbf{s}_t$, execute $\mathbf{a}_t = \pi_{\boldsymbol{\theta}_t}(\mathbf{s}_t)$ observe r_t and transit to $\mathbf{s}'_t$. Then add $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}'_t)$ to a replay buffer $\mathcal{D}$.
  Drawn M minibatch samples $\{(\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i)\}_{i=1}^M$ from $\mathcal{D}$.
  Update the Q-network weight $\boldsymbol{\omega}$ by stochastic gradient descent or Adam:

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t + \kappa \sum_{i=1}^M \left(r_i + \gamma \widehat{Q}_{\tilde{\boldsymbol{\omega}}_t}(\mathbf{s}'_i, \pi_{\tilde{\boldsymbol{\mu}}_t}(\mathbf{s}'_i)) - \widehat{Q}_{\boldsymbol{\omega}_t}(\mathbf{s}_i, \mathbf{a}_i) \right) \nabla_{\boldsymbol{\omega}} \widehat{Q}_{\boldsymbol{\omega}_t}(\mathbf{s}_i, \mathbf{a}_i)/M.$$

  Compute deterministic policy gradient using the sampled policy parameter:

$$\hat{\nabla}_{\boldsymbol{\theta}} F(\boldsymbol{\theta}_t) = -\sum_{i=1}^M \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}_t}(\mathbf{s}_i) \nabla_{\mathbf{a}} Q_{\boldsymbol{\omega}_{t+1}}(\mathbf{s}_i, \pi_{\boldsymbol{\theta}_t}(\mathbf{s}_i))/M.$$

  Update the mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}^2$ by VadaGrad:

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t - \alpha \hat{\nabla}_{\boldsymbol{\theta}} F(\boldsymbol{\theta}_t)/\sqrt{\mathbf{s}_{t+1}}, \quad \mathbf{s}_{t+1} = \mathbf{s}_t + (1-\gamma_2)[\hat{\nabla}_{\boldsymbol{\theta}} F(\boldsymbol{\theta}_t)]^2.$$

  Update target network parameters $\tilde{\boldsymbol{\omega}}_{t+1}$ and $\tilde{\boldsymbol{\mu}}_{t+1}$ by moving average:

$$\tilde{\boldsymbol{\omega}}_{t+1} = (1-\tau)\tilde{\boldsymbol{\omega}}_t + \tau \boldsymbol{\omega}_{t+1}, \quad \tilde{\boldsymbol{\mu}}_{t+1} = (1-\tau)\tilde{\boldsymbol{\mu}}_t + \tau \boldsymbol{\mu}_{t+1}.$$

end for

Algorithm 2 Parameter-based exploration DDPG via Vadam

Initialize: Initial mean $\boldsymbol{\mu}_1$, 1st-order moment $\mathbf{m}_1 = 0$, 2nd-order moment $\mathbf{s}_1 = 0$, prior $\lambda = 10^{-8}$, constant $c = 10000$.
for Time step $t = 1, ..., \infty$ **do**
  Sample policy parameter $\boldsymbol{\theta}_t \sim \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_t, (\mathbf{s}_t + \lambda + c)^{-1})$.
  Observe $\mathbf{s}_t$, execute $\mathbf{a}_t = \pi_{\boldsymbol{\theta}_t}(\mathbf{s}_t)$ observe r_t and transit to $\mathbf{s}'_t$. Then add $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}'_t)$ to a replay buffer $\mathcal{D}$.
  Drawn M minibatch samples $\{(\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i)\}_{i=1}^M$ from $\mathcal{D}$.
  Update the Q-network weight $\boldsymbol{\omega}$ by stochastic gradient descent or Adam:

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t + \kappa \sum_{i=1}^M \left(r_i + \gamma \widehat{Q}_{\tilde{\boldsymbol{\omega}}_t}(\mathbf{s}'_i, \pi_{\tilde{\boldsymbol{\mu}}_t}(\mathbf{s}'_i)) - \widehat{Q}_{\boldsymbol{\omega}_t}(\mathbf{s}_i, \mathbf{a}_i) \right) \nabla_{\boldsymbol{\omega}} \widehat{Q}_{\boldsymbol{\omega}_t}(\mathbf{s}_i, \mathbf{a}_i)/M.$$

  Compute deterministic policy gradient using the sampled policy parameter:

$$\hat{\nabla}_{\boldsymbol{\theta}} F(\boldsymbol{\theta}_t) = -\sum_{i=1}^M \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}_t}(\mathbf{s}_i) \nabla_{\mathbf{a}} Q_{\boldsymbol{\omega}_{t+1}}(\mathbf{s}_i, \pi_{\boldsymbol{\theta}_t}(\mathbf{s}_i))/M.$$

  Update and correct the bias of the 1st-order moment $\mathbf{m}$ and the 2nd-order moment $\mathbf{s}$ by Vadam:

$$\mathbf{m}_{t+1} = \gamma_1 \mathbf{m}_t + (1-\gamma_1)(\hat{\nabla}_{\boldsymbol{\theta}} F(\boldsymbol{\theta}_t) + \lambda \boldsymbol{\mu}_t), \qquad \mathbf{s}_{t+1} = \gamma_2 \mathbf{s}_t + (1-\gamma_2)[\hat{\nabla}_{\boldsymbol{\theta}} F(\boldsymbol{\theta}_t)]^2,$$
$$\hat{\mathbf{m}}_{t+1} = \mathbf{m}_t/(1-\gamma_1^t), \qquad \hat{\mathbf{s}}_{t+1} = \mathbf{s}_{t+1}/(1-\gamma_2^t).$$

  Update the mean $\boldsymbol{\mu}$ using the moment estimates by Vadam:

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t - \alpha \hat{\mathbf{m}}_{t+1}/(\sqrt{\hat{\mathbf{s}}_t} + \lambda).$$

  Update target network parameters $\tilde{\boldsymbol{\omega}}_{t+1}$ and $\tilde{\boldsymbol{\mu}}_{t+1}$ by moving average:

$$\tilde{\boldsymbol{\omega}}_{t+1} = (1-\tau)\tilde{\boldsymbol{\omega}}_t + \tau \boldsymbol{\omega}_{t+1}, \quad \tilde{\boldsymbol{\mu}}_{t+1} = (1-\tau)\tilde{\boldsymbol{\mu}}_t + \tau \boldsymbol{\mu}_{t+1}.$$

end for

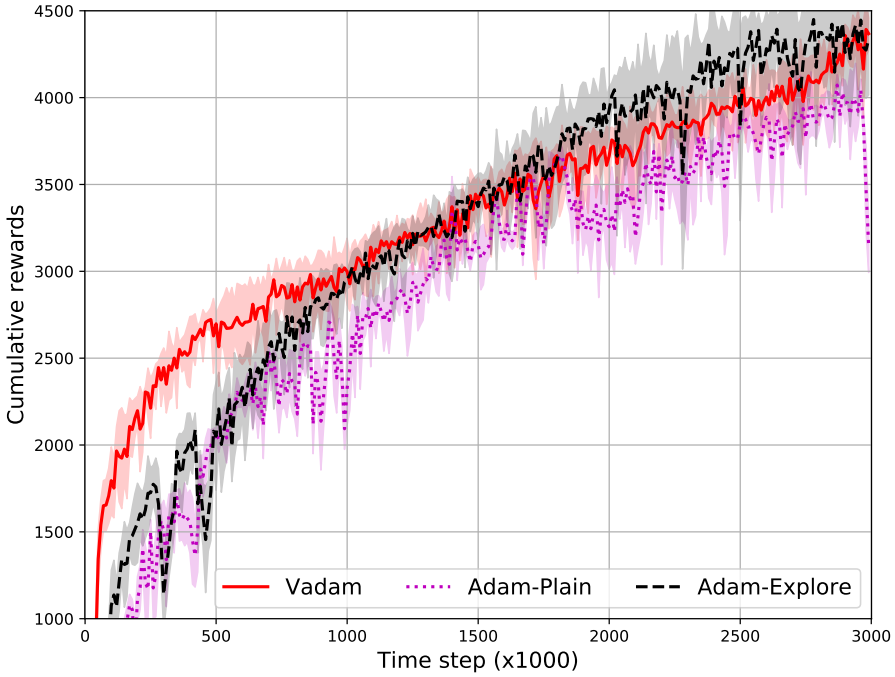

Figure 5. The early learning performance of Vadam, Adam-Plain and Adam-Explore, on the half-cheetah task in the reinforcement learning experiment. Vadam shows faster learning in this early stage of learning. The mean and standard error are computed over 5 trials.

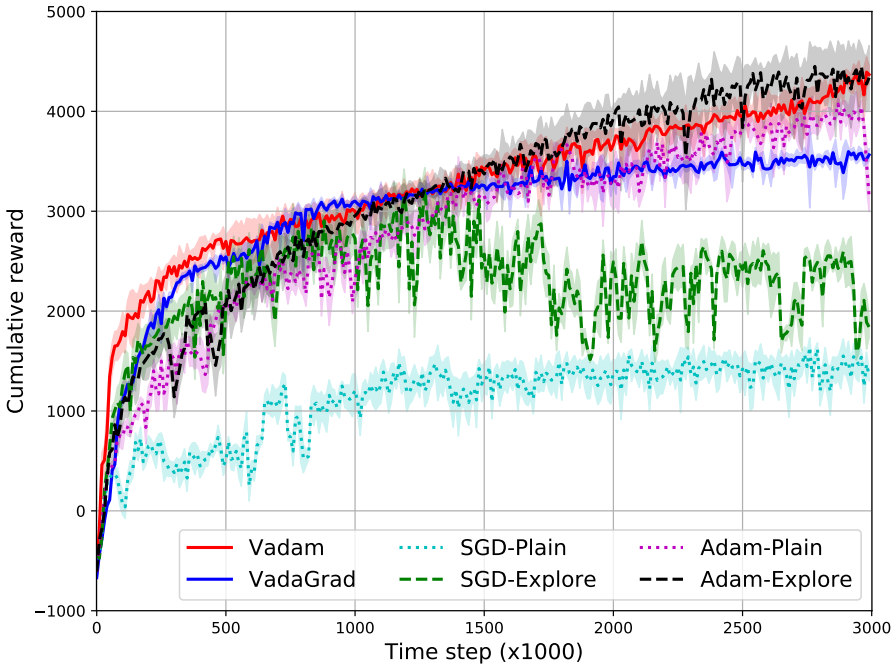

Figure 6. The performance of all evaluated methods on the half-cheetah task in the reinforcement learning experiment. Vadam and Adam-based methods perform well overall and give comparable final performance. VadaGrad also learns well but shows sign of premature convergence. SGD-based method do not learn well throughout. The mean and standard error are computed over 5 trials.

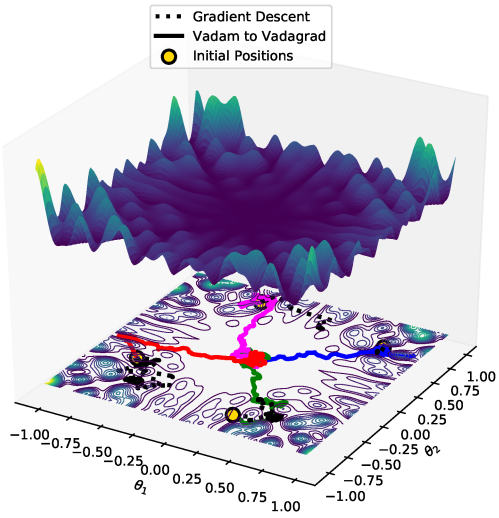

Figure 7. Illustration of variational optimization on a complex 2D objective function. Variational optimization is performed from four different initial positions. The four runs are shown in solid lines in different colors. Gradient descent (shown in dotted, black lines) is also initialized at the same locations. 'Vadam to VadaGrad' shows ability to navigate the landscape to reach the flat (and global) minimum, while gradient descent gets stuck in various locations.

M. Experiment on Improving "Marginal Value of Adaptive-Gradient Methods"

Recently, Wilson et al. (2017) show some examples where adaptive gradient methods, namely Adam and AdaGrad, generalize worse than SGD. We repeated their experiments to see whether weight-perturbation in VadaGrad improves the generalization performance of AdaGrad. We firstly consider an experiment on the character-level language modeling on the novel War and Peace dataset (shown in Fig. 2b in Wilson et al. (2017)). Figure 8 shows the test error of SGD, AdaGrad, Adam and VadaGrad. We can see that VadaGrad generalizes well and achieves the same performance as 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 regularization techniques such as batch normalization, batch flip, and dropout, play an important role for the CIFAR-10 dataset and that is why we did not see an improvement by using VadaGrad. Further investigation will be done in future works.

We use the following hyper-parameter setting in this experiment. For all methods, we divide the step-size α by 10 once every K epochs, as described by Wilson et al. (2017). For VadaGrad, AdaGrad, and Adam, we fixed the value of scaling vector step-size β and do not decay. For AdaGrad and Adam, the initial value of scaling vector is 0. For VadaGrad, we use the initial value $\mathbf{s}_1 = 10^{-4}$ to avoid numerical issue of sampling from Gaussian with 0 precision. (Recall that VadaGrad does not have a prior λ). We perform grid search to find the best value of initial α, K, and β for all methods except Adam which use the default value of $\beta = 0.999$ and $\gamma = 0.9$. The best hyper-parameter values which give the minimum test loss are given in Table 4.

Method	K	α	β
VadaGrad	40	0.0075	0.5
SGD	80	0.5	-
AdaGrad	80	0.025	0.5
Adam	40	0.0012	0.999

Table 4. Hyper-parameter setting for the "Marginal Value of Adaptive-Gradient Methods" experiment.

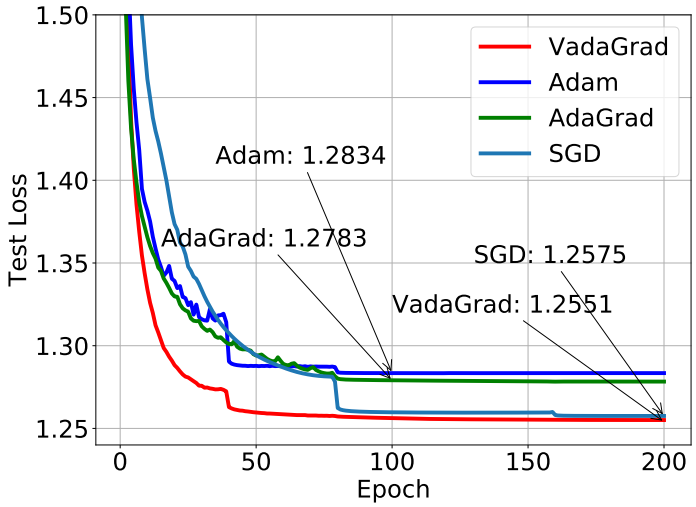

Figure 8. Results for character-level language modeling using the War and Peace dataset for the "Marginal Value of Adaptive-Gradient Methods" experiment. We repeat the results shown in Fig. 2 (b) of Wilson et al. (2017) and show that VadaGrad does not suffer from the issues pointed in that paper, and it performs comparable to SGD.