Learning Structural Weight Uncertainty for Sequential Decision-Making

Ruiyi Zhang[1] **Chunyuan Li**[1] **Changyou Chen**[2] **Lawrence Carin**[1]
[1]Duke University [2]University at Buffalo
ryzhang@cs.duke.edu, cl319@duke.edu, cchangyou@gmail.com, lcarin@duke.edu

Abstract

Learning probability distributions on the weights of neural networks (NNs) has recently proven beneficial in many applications. Bayesian methods, such as Stein variational gradient descent (SVGD), offer an elegant framework to reason about NN model uncertainty. However, by assuming independent Gaussian priors for the individual NN weights (as often applied), SVGD does not impose prior knowledge that there is often structural information (dependence) among weights. We propose efficient posterior learning of structural weight uncertainty, within an SVGD framework, by employing matrix variate Gaussian priors on NN parameters. We further investigate the learned structural uncertainty in sequential decision-making problems, including contextual bandits and reinforcement learning. Experiments on several synthetic and real datasets indicate the superiority of our model, compared with state-of-the-art methods.

1 Introduction

Deep learning has achieved state-of-the-art performance on a wide range of tasks, including image classification [Krizhevsky et al., 2012], language modeling [Sutskever et al., 2014], and game playing [Silver et al., 2016]. One challenge in training deep neural networks (NNs) is that such models may overfit to the observed data, yielding over-confident decisions in learning tasks. This is partially because most NN learning only seeks a point estimate for the model parameters, failing to quantify parameter uncertainty. A natural way to ameliorate these problems is to adopt a Bayesian neural network (BNN) formulation. By imposing priors on the weights, a BNN utilizes available data to infer an approximate posterior distribution on NN parameters [MacKay, 1992]. When making subsequent predictions (at test time), one performs model averaging over such learned uncertainty, effectively yielding a mixture of NN models [Gal and Ghahramani, 2016, Zhang et al., 2016, Liu and Wang, 2016, Li et al., 2016a, Chen and Zhang, 2017]. BNNs have shown improved performance on modern achitectures, including convolutional and recurrent networks [Li et al., 2016b, Gan et al., 2017, Fortunato et al., 2017].

For computational convenience, traditional BNN learning typically makes two assumptions on the weight distributions: independent isotropic Gaussian distributions as priors, and fully factorized Gaussian proposals as posterior approximation when adopting variational inference [Hernández-Lobato and Adams, 2015, Blundell et al., 2015, Liu and Wang, 2016, Feng et al., 2018, Pu et al., 2017b]. By examining this procedure, we note two limitations: (*i*) the independent Gaussian priors can ignore the anticipated structural information between the weights, and (*ii*) the factorized Gaussian posteriors can lead to unreasonable approximation errors and underestimate model uncertainty (underestimate variances).

Recent attempts have been made to overcome these two issues. For example, [Louizos and Welling, 2016, Sun et al., 2017] introduced *structural priors* with the matrix variate Gaussian (MVG) distribution [Gupta and Nagar, 1999] to impose dependency between weights within each layer of a BNN. Further, nonparametric variational inference methods, *e.g.*, Stein variational gradient descent (SVGD) [Liu and Wang, 2016], iteratively transport a set of particles to approximate the target posterior distribution (without making explicit assumptions about the form of the posterior, avoiding the aforementioned factorization as-

Proceedings of the 21^{st} International Conference on Artificial Intelligence and Statistics (AISTATS) 2018, Lanzarote, Spain. PMLR: Volume 84. Copyright 2018 by the author(s).

sumption). SVGD represents the posterior approximately in terms of a set of particles (samples), and is endowed with guarantees on the approximation accuracy when the number of particles is exactly infinity [Liu, 2017]. However, since the updates within SVGD learning involve kernel computation in the parameter space of interest, the algorithm can be computationally expensive in a high-dimensional space. This becomes even worse in the case of structural priors, where a large amount of additional parameters are introduced, rendering SVGD inefficient when directly applied for posterior inference.

We propose an efficient learning scheme for accurate posterior approximation of NN weights, adopting the MVG structural prior. We provide a new perspective to unify previous structural weight uncertainty methods [Louizos and Welling, 2016, Sun et al., 2017] via the Householder flow [Tomczak and Welling, 2016]. This perspective allows SVGD to approximate a target structural distribution in a lower-dimensional space, and thus is more efficient in inference. We call the proposed algorithm *Structural Stein Variational Gradient Descent* (S^{2}VGD).

We investigate the use of our structural-weight-uncertainty framework for learning policies in sequential decision problems, including contextual bandits and reinforcement learning. In these models, uncertainty is particularly important because greater uncertainty on the weights typically introduces more variability into a decision made by a policy network [Kolter and Ng, 2009], naturally leading the policy to explore. As more data are observed, the uncertainty decreases, allowing the decisions made by a policy network to become more deterministic as the environment is better understood (exploitation when the policy becomes more confident). In all these models, structural weight uncertainty is inferred by our proposed S^{2}VGD.

We conduct several experiments, first demonstrating that S^{2}VGD yields effective performance on classic classification/regression tasks. We then focus our experiments on the motivating applications, sequential decision problems, for which accounting for NN weight uncertainty is believed to be particularly beneficial. In these applications the proposed method demonstrates particular empirical value, while also being computationally practical. The results show that structural weight uncertainty gives better expressive power to describe uncertainty driving better exploration.

2 Preliminaries

2.1 Matrix variate Gaussian distributions

The matrix variate Gaussian (MVG) distribution [Gupta and Nagar, 1999] has three parameters, describing the probability of a random matrix $\mathbf{W} \in \mathbb{R}^{\ell_1 \times \ell_2}$:

$$p(\mathbf{W}) \triangleq \mathcal{MN}(\mathbf{W}; \mathbf{M}, \mathbf{U}, \mathbf{V}) = \frac{\exp\left(\frac{1}{2}\mathrm{tr}[\mathbf{V}^{-1}(\mathbf{W}-\mathbf{M})^\top \mathbf{U}^{-1}(\mathbf{W}-\mathbf{M})]\right)}{(2\pi)^{\ell_1\ell_2/2}|\mathbf{V}|^{\ell_1/2}|\mathbf{U}|^{\ell_2/2}} \quad (1)$$

where $\mathbf{M} \in \mathbb{R}^{\ell_1 \times \ell_2}$ is the mean of the distribution. $\mathbf{U} \in \mathbb{R}^{\ell_1 \times \ell_1}$ and $\mathbf{V} \in \mathbb{R}^{\ell_2 \times \ell_2}$ encode covariance information for the rows and columns of $\mathbf{W}$ respectively. The MVG is closely related to the multivariate Gaussian distribution.

Lemma 1 (Golub and Van Loan [2012])**.** *Assume* $\mathbf{W}$ *follows the MVG distribution in (1), then*

$$\mathsf{vec}(\mathbf{W}) \sim \mathcal{N}(\mathsf{vec}(\mathbf{M}), \mathbf{V} \otimes \mathbf{U}) \quad (2)$$

where $\mathsf{vec}(\mathbf{M})$ *is the vectorization of* $\mathbf{M}$ *by stacking the columns of* $\mathbf{M}$*, and* $\otimes$ *denotes the standard Kronecker product [Golub and Van Loan, 2012].*

Furthermore, a linear transformation of an MVG distribution is still an MVG distribution.

Lemma 2 (Golub and Van Loan [2012])**.** *Assume* $\mathbf{W}$ *follows the MVG distribution in (1),* $\mathbf{A} \in \mathbb{R}^{\ell_2 \times \ell_1}, \mathbf{C} \in \mathbb{R}^{\ell_2 \times \ell_1}$*, then,*

$$\begin{aligned} \mathbf{B} &\triangleq \mathbf{AW} \sim \mathcal{MN}(\mathbf{B}; \mathbf{AM}, \mathbf{AUA}^\top, \mathbf{V}) \\ \mathbf{B} &\triangleq \mathbf{WC} \sim \mathcal{MN}(\mathbf{B}; \mathbf{MC}, \mathbf{U}, \mathbf{C}^\top\mathbf{VC}) \end{aligned} \quad (3)$$

MVG priors for BNNs For classification and regression tasks on data $\mathcal{D} = \{\boldsymbol{d}_1, \cdots, \boldsymbol{d}_N\}$, where $\boldsymbol{d}_i = \{\mathbf{x}_i, \boldsymbol{y}_i\}$, with input $\mathbf{x}_i$ and output $\boldsymbol{y}_i$, an L-layer NN parameterizes the mapping $\{g_\ell\}_{\ell=1}^L$, defining the prediction of $\boldsymbol{y}_i$ for $\mathbf{x}_i$ as:

$$\hat{\boldsymbol{y}}_i = f(\boldsymbol{x}_i) = g_L \circ g_{L-1} \circ \cdots \circ g_0(\mathbf{x}_i), \quad \forall i. \quad (4)$$

where $\circ$ represents function composition, *i.e.*, $\mathcal{A} \circ \mathcal{B}$ means $\mathcal{A}$ is evaluated on the output of $\mathcal{B}$. Each layer g_ℓ represents a nonlinear transformation. For example, with the Rectified Linear Unit (ReLU) activation function [Nair and Hinton, 2010], $g_\ell(\boldsymbol{x}_i) = \mathsf{ReLU}(\mathbf{W}_\ell^\top \boldsymbol{x}_i + \mathbf{b}_\ell)$, where $\mathsf{ReLU}(x) \triangleq \max(0, x)$, $\mathbf{W}_\ell$ is the weight matrix for the ℓth-layer, and $\mathbf{b}_\ell$ the corresponding bias term.

The MVG can be adopted as a prior for the weight matrix in each layer, to impose the prior belief that there are intra-layer weight correlations,

$$\mathbf{W}_\ell \sim \mathcal{MN}(\mathbf{W}; \mathbf{0}, \mathbf{U}_\ell, \mathbf{V}_\ell), \quad (5)$$

where the covariances $\mathbf{U}_\ell, \mathbf{V}_\ell$ have components drawn independently from $\mathsf{Inv\text{-}Gamma}(a_0, b_0)$. The parameters are $\boldsymbol{\theta} \triangleq \{\mathbf{W}_\ell, \log \mathbf{U}_\ell, \log \mathbf{V}_\ell\}$, and the above distributions represent the prior $p(\boldsymbol{\theta})$. Our goal with BNNs is to learn layer-wise structured weight uncertainty, described by the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D}) \propto p(\boldsymbol{\theta})p(\mathcal{D}|\boldsymbol{\theta})$, represented below as p for simplicity. When $\mathbf{U} = \sigma\mathbf{I}$ and $\mathbf{V} = \sigma\mathbf{I}$, we reduce to BNNs with independent isotropic Gaussian priors [Blundell et al., 2015].

2.2 Stein Variational Gradient Descent

SVGD considers a set of particles $\{\boldsymbol{\theta}_i\}_{i=1}^M$ drawn from distribution q, and transforms them to better match the target distribution p, by update:

$$\begin{aligned} &\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i + \epsilon\phi(\boldsymbol{\theta}_i), \\ &\phi = \arg\max_{\phi\in\mathcal{F}} \left\{ \frac{\partial}{\partial\epsilon}\mathsf{KL}(q_{[\epsilon\phi]}||p) \right\}, \end{aligned} \tag{6}$$

where $q_{[\epsilon\phi]}$ is the updated empirical distribution, with ϵ as the step size, and ϕ as a function perturbation direction chosen to minimize the KL divergence between q and p. SVGD considers $\mathcal{F}$ as the unit ball of a vector-valued reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ associated with a kernel $\kappa(\boldsymbol{\theta}, \boldsymbol{\theta}')$. The RBF kernel is usually used as default. It has been shown [Liu and Wang, 2016] that:

$$\begin{aligned} -\frac{\partial}{\partial\epsilon}\mathsf{KL}(q_{[\epsilon\phi]}||p)|_{\epsilon=0} &= \mathbb{E}_{\boldsymbol{\theta}\sim q}[\Gamma_p\phi(\boldsymbol{\theta})], \\ \text{with } \ \Gamma_p\phi(\boldsymbol{\theta}) &\triangleq \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|\mathcal{D})^\top\phi(\boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}}\cdot\phi(\boldsymbol{\theta}), \end{aligned} \tag{7}$$

where $\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta})$ denotes the derivative of the log-density of p; Γ_p is the Stein operator. Assuming that the update function $\phi(\boldsymbol{\theta})$ is in a RKHS with kernel $\kappa(\cdot,\cdot)$, it has been shown in [Liu and Wang, 2016] that (7) is maximized with:

$$\phi(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\theta}\sim q}[\kappa(\boldsymbol{\theta}, \boldsymbol{\theta}')\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|\mathcal{D}) + \nabla_{\boldsymbol{\theta}}\kappa(\boldsymbol{\theta}, \boldsymbol{\theta}')]. \tag{8}$$

The expectation $\mathbb{E}_{\boldsymbol{\theta}\sim q}[\cdot]$ can be approximated by an empirical averaging of particles $\{\boldsymbol{\theta}_i\}_{i=1}^M$, resulting in a practical SVGD procedure as:

$$\begin{aligned} \boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i + \frac{\epsilon}{M}\sum_{j=1}^{M} \Big[&\kappa(\boldsymbol{\theta}_j, \boldsymbol{\theta}_i)\nabla_{\boldsymbol{\theta}_j} \log p(\boldsymbol{\theta}_j|\mathcal{D}) \\ &+\nabla_{\boldsymbol{\theta}_j}\kappa(\boldsymbol{\theta}_j, \boldsymbol{\theta}_i)\Big]. \end{aligned} \tag{9}$$

The first term to the right of the summation in (9) drives the particles $\boldsymbol{\theta}_i$ towards the high probability regions of p, with information sharing across similar particles. The second term repels the particles away from each other, encouraging coverage of the entire distribution. SVGD applies the updates in (9) repeatedly, and the samples move closer to the target distribution p in each iteration. When using state-of-the-art stochastic gradient-based algorithms, *e.g.*, RMSProp [Hinton et al., 2012] or Adam [Kingma and Ba, 2015], SVGD becomes a highly efficient and scalable Bayesian inference method.

Computational Issues Applying SVGD with structured distributions for NNs has many challenges. For a weight matrix $\mathbf{W}$ of size $\ell_1 \times \ell_2$, the number of parameters in $\mathbf{U}$ and $\mathbf{V}$ are ℓ_1^2 and ℓ_2^2, respectively. Hence, the total number of parameters $\boldsymbol{\theta}$ needed to describe the distribution is $\ell_1\ell_2 + \ell_1^2 + \ell_2^2$, compared to $\ell_1\ell_2 + 1$ in traditional BNNs that employ isotropic Gaussian priors and factorization.

The increase of parameter dimension by $\ell_1^2 + \ell_2^2 - 1$ leads to significant computational overhead. The problem becomes even more severe in two aspects in calculating the kernels: (*i*) The computation increases quadratically by $M(M-1)(\ell_1^2+\ell_2^2-1)/2$, (*ii*) the approximation to the repelling term in (9) using limited particles can be inaccurate in high dimensions. Therefore, it is desirable to transform the MVG distribution to a lower-dimensional representation.

3 SVGD & Imposition of Structure

3.1 Reparameterization of the MVG

Since the covariance matrices $\mathbf{U}$ and $\mathbf{V}$ are positive definite, we can decompose them as $\mathbf{U} = \mathbf{P}\boldsymbol{\Lambda}_1\boldsymbol{\Lambda}_1\mathbf{P}^\top, \mathbf{V} = \mathbf{Q}\boldsymbol{\Lambda}_2\boldsymbol{\Lambda}_2\mathbf{Q}^\top$, where $\mathbf{P}$ and $\mathbf{Q}$ are the corresponding orthogonal matrices, $\boldsymbol{\Lambda}_1$ and $\boldsymbol{\Lambda}_2$ are diagonal matrices with positive diagonal elements. According to Lemma 2, we show the following reparameterization of MVG:

Proposition 3. *For a random matrix* $\mathbf{C}$ *following an independent Gaussian distribution:*

$$\mathit{vec}(\mathbf{C}) \sim \mathcal{N}(\cdot,\ \mathbf{P}^\top\boldsymbol{\Lambda}_1^{-1}\mathbf{M}\boldsymbol{\Lambda}_2^{-1}\mathbf{Q}, \mathbf{I}), \tag{10}$$

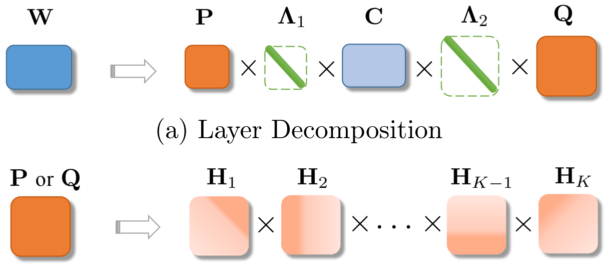
the corresponding full-covariance MVG $\mathbf{W}$ *in* (1) *can be reparameterized as* $\mathbf{W} = \mathbf{P}\boldsymbol{\Lambda}_1\mathbf{C}\boldsymbol{\Lambda}_2\mathbf{Q}^\top$.

The proof is in Section A of the Supplementary Material. Therefore, $\mathbf{W}$ drawn from MVG can be decomposed as the linear product of five random matrices:

- $\mathbf{C}$ as a *standard weight matrix* with an independent Gaussian distribution.
- $\boldsymbol{\Lambda}_1$ and $\boldsymbol{\Lambda}_2$ as the *diagonal matrices*, encoding the structural information within each row and column, respectively.
- $\mathbf{P}$ and $\mathbf{Q}$ as *orthogonal matrices*, which characterize the structural information of weights between rows and columns, respectively.

3.2 MVG as Householder Flows

Based on Proposition 3, we propose a *layer decomposition*: the one-layer weight matrix $\mathbf{W}$ with an MVG prior can be decomposed into a linear product of five matrices, as illustrated in Figure 1(a). Our layer decomposition provides an interesting interpretation for the original MVG layer: it is equivalent to several



(a) Layer Decomposition

(b) Householder Flow

Figure 1: Illustration of the two proposed techniques to reduce parameter size in learning the distribution of $\mathbf{W}$: (a) decomposition of $\mathbf{W}$ as a linear product of five matrices, and (b) approximation of $\mathbf{P}$ or $\mathbf{Q}$ as a linear product of K Householder matrices. Note that each rectangle indicates a matrix, "×" indicates the matrix product, and $\mathbf{H}$ in (b) is constructed using (12).

within-layer transformations. The representations imposed in the standard weight matrix $\mathbf{C}$ are rotated by $\mathbf{P}$ and $\mathbf{Q}$, and re-weighted by $\mathbf{\Lambda}_1$ and $\mathbf{\Lambda}_2$.

Note that the layer decomposition maintains similar computational complexity as the original MVG layer. To reduce the computation bottleneck in the layer decomposition, we further propose to represent $\mathbf{P}$ and $\mathbf{Q}$ using *Householder flows* [Tomczak and Welling, 2016]. Formally, a Householder transformation is a linear transformation that describes a reflection about a hyperplane containing the origin. Householder flow is a series of Householder transformations.

Lemma 4 ([Sun and Bischof, 1995])**.** *Any orthogonal matrix* $\mathbf{M}$ *of degree* K *can be expressed as a Householder flow,* i.e., *a product of exactly* K *nontrivial Householder matrices., i.e.,*

$$\mathbf{M} = \mathbf{H}_K \mathbf{H}_{K-1} \cdots \mathbf{H}_1. \tag{11}$$

Importantly, each Householder matrix $\mathbf{H}$ is constructed from a Householder vector $\boldsymbol{v}$ (which is orthogonal to the hyperplane):

$$\mathbf{H} = \mathbf{I} - 2\frac{\boldsymbol{v}\boldsymbol{v}^\top}{\|\boldsymbol{v}\|^2}. \tag{12}$$

According to Lemma 4, $\mathbf{P}$ and $\mathbf{Q}$ can be represented as a product of Householder matrices $\{\mathbf{H}_k^{(p)}\}$ and $\{\mathbf{H}_k^{(q)}\}$:

$$\begin{aligned}
\mathbf{P} &= \mathbf{H}_K^{(p)} \mathbf{H}_{K-1}^{(p)} \cdots \mathbf{H}_1^{(p)} \\
\mathbf{Q} &= \mathbf{H}_K^{(q)} \mathbf{H}_{K-1}^{(q)} \cdots \mathbf{H}_1^{(q)} \\
\mathbf{H}_k^{(p)} &= \mathbf{I} - 2\boldsymbol{v}_k^{(p)} \boldsymbol{v}_k^{(p)\top} / \left(\boldsymbol{v}_k^{(p)\top} \boldsymbol{v}_k^{(p)} \right) \\
\mathbf{H}_k^{(q)} &= \mathbf{I} - 2\boldsymbol{v}_k^{(q)} \boldsymbol{v}_k^{(q)\top} / \left(\boldsymbol{v}_k^{(q)\top} \boldsymbol{v}_k^{(q)} \right),
\end{aligned} \tag{13}$$

where $\boldsymbol{v}_k^{(p)}$ and $\boldsymbol{v}_k^{(q)}$ are the kth Householder vector for $\mathbf{P}$ and $\mathbf{Q}$, respectively. This is illustrated in Figure 1 (b). Note that the degree $K \leq \min\{\ell_1, \ell_2\}$, with proof in Section B of Supplementary Material. In practice, K is a trade-off hyperparameter, balancing the approximation accuracy and computation trade-off.

Since Householder flows allow one to represent $\mathbf{P}$ or $\mathbf{Q}$ as K Householder vectors, the parameter sizes reduce from ℓ_1^2 and ℓ_2^2 to $K\ell_1$ and $K\ell_2$. Overall, we can model $\mathbf{W}$ with structured weight priors using only $(K+1)(\ell_1+\ell_2)+\ell_1\ell_2$ parameters. Therefore, we can efficiently capture the structure information with only a slight increase of computational cost (*i.e.*, $(K+1)(\ell_1+\ell_2)$).

Interestingly, our method provides a unifying perspective of previous methods on learning structured weight uncertainty. In terms of prior distributions, when $K=0$ (*i.e.*, $\mathbf{P}=\mathbf{Q}=\mathbf{I}$), our reparameterization reduces to [Louizos and Welling, 2016]. When $K=1$, and $\mathbf{\Lambda}_1=\mathbf{\Lambda}_2=\mathbf{I}$, our reparameterization reduces to [Sun et al., 2017]. In terms of posterior learning methods, when $\mathbf{P}=\mathbf{Q}=\mathbf{\Lambda}_1=\mathbf{\Lambda}_2=\mathbf{I}$ and $M>1$, S^{2}VGD reduces to SVGD; when $M=1$, it reduces to learning an MAP solution.

3.3 Structural BNNs Revisited

We can leverage the layer decomposition and Householder flow above to construct an equivalent BNN by approximating the ℓth MVG layer in (5) with standard Gaussian weight matrices:

$$\begin{aligned}
p(\mathbf{C}|\lambda) &= \mathcal{N}\left(\mathbf{C}_\ell; \mathbf{0}, \lambda\right), \\
p(\boldsymbol{v}_{k\ell}^{(p)}|\phi) &= \mathcal{N}\left(\boldsymbol{v}_{k\ell}^{(p)}; \mathbf{0}, \phi\mathbf{I}\right), \\
p(\boldsymbol{v}_{k\ell}^{(q)}|\phi) &= \mathcal{N}\left(\boldsymbol{v}_{k\ell}^{(q)}; \mathbf{0}, \phi\mathbf{I}\right), \\
p(\mathbf{\Lambda}^{(1)}|\psi) &= \mathcal{N}\left(\mathbf{\Lambda}_\ell^{(1)}; \mathbf{0}, \psi\mathbf{I}\right) \\
p(\mathbf{\Lambda}^{(2)}|\psi) &= \mathcal{N}\left(\mathbf{\Lambda}_\ell^{(2)}; \mathbf{0}, \psi\mathbf{I}\right), \\
\lambda, \phi, \psi &\sim \mathsf{Inv\text{-}Gamma}(\cdot; a_\ell, b_\ell).
\end{aligned} \tag{14}$$

The forms of the likelihood for the last layer are defined according to the specific applications. For regression problems on real-valued response $\boldsymbol{y}$:

$$\begin{aligned}
\mathbf{y}|\mathbf{x}, \mathbf{W}_L &\sim \mathcal{N}(\mathbf{y}; f(\boldsymbol{x}, \mathbf{W}_L), \gamma\mathbf{I}) \\
\gamma &\sim \mathsf{Inv\text{-}Gamma}(\cdot; a_L, b_L),
\end{aligned} \tag{15}$$

with $f(\cdot)$ a neural network defined in (4). For classification problems on discrete labels $\boldsymbol{y}$:

$$\mathbf{y}|\mathbf{x}, \mathbf{W}_L \sim \mathsf{Categorical}(\mathbf{y}; \mathsf{Softmax}(f(\boldsymbol{x}, \mathbf{W}_L))). \tag{16}$$

Note that $\mathbf{W}_L$ can follow the same proposed techniques to reduce parameter size. Therefore, standard SVGD algorithms can be applied to sample from the posterior distribution of each model parameter. Intuitively, in SVGD the kernel function governs the interactions between particles, which employs this information to accelerate convergence and provide better performance. Similarly, the Householder flow, encoding structural information, controls the interactions between weights in each particle.

4 Sequential Decision-Making

A principal motivation for the proposed S^{2}VGD framework is sequential decision-making, including contextual multi-arm bandits (CMABs) and Markov decision processes (MDPs). A challenge in sequential decision-making in the face of uncertainty is the exploration/exploitation trade-off: the trade-off between either taking actions that are most rewarding according to the current knowledge, or taking exploratory actions, which may be less immediately rewarding, but may lead to better-informed decisions in the future. In a Bayesian setting, the exploration/exploration trade-off is naturally addressed by imposing uncertainty into the parameters of a policy model.

4.1 CMABs and Stein Thompson Sampling

CMABs model stochastic, discrete-time and finite action-state space control problems. A CMAB is formally defined as a tuple $\mathcal{C} = \langle \mathcal{S}, \mathcal{A}, P_s, P_r, r \rangle$, where $\mathcal{S}$ is the state/context space, $\mathcal{A}$ the action/arm space, $r \in \mathbb{R}$ is the reward, P_s and P_r are the unknown environment distributions to draw the context and reward, respectively. At each time step t, the agent (*i*) first observes a context $\boldsymbol{s}_t \in \mathcal{S}$, drawn i.i.d. over time from P_s; then (*ii*) chooses an action at $\boldsymbol{a}_t \in \mathcal{A}$ and observes a stochastic reward $r_t(\boldsymbol{a}_t, \boldsymbol{s}_t)$, which is drawn i.i.d. over time from $P_r(\cdot|\boldsymbol{a}_t, \boldsymbol{s}_t)$, conditioned on the current context and action. The agent makes decisions via a policy $\pi(\boldsymbol{a}|\boldsymbol{s})$ that maps each context to a distribution over actions, yielding the probability of choosing action $\boldsymbol{a}$ in state $\boldsymbol{s}$. The goal in CMABs is to learn a policy to maximize the expected total reward in T interactions: $J(\pi) = \mathbb{E}_{P_s,\pi,P_r} \sum_{t=1}^{T} r_t$.

We represent the policy using a $\boldsymbol{\theta}$-parameterized neural network $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})$, where MVG priors $p(\boldsymbol{\theta})$ are employed on the weights. At each time t, given the past observations $\mathcal{D}_t \triangleq \{\boldsymbol{d}\}_{j=1}^{t}$, where $\boldsymbol{d}_j = (\boldsymbol{s}_j, \boldsymbol{a}_j, r_j)$, the posterior distribution of $\boldsymbol{\theta}_t$ is updated as $p(\boldsymbol{\theta}_t|\mathcal{D}_t) \propto \prod_{j=1}^{t} p(\boldsymbol{d}_j|\boldsymbol{\theta})p(\boldsymbol{\theta})$.

Thompson sampling [Thompson, 1933] is a popular method to solve CMABs [Li et al., 2011]. It approximates the posterior $p(\boldsymbol{\theta}|\mathcal{D}_t)$ in an online manner. At each step, Thompson sampling (*i*) first draws a set of parameter samples, then (*ii*) picks the action by maximizing the expected reward over current step, *i.e.*, $\boldsymbol{a}_t = \arg\max_{\boldsymbol{a}} \mathbb{E}_{r\sim P_r(\cdot|\boldsymbol{a},\boldsymbol{s}_t;\boldsymbol{\theta}_t)} r_t$, (*iii*) collects data samples after observing the reward r_t, and (*iv*) updates posterior of the policy. We apply the proposed S^{2}VGD for the updates in the final step, and call the new procedure Stein Thompson sampling, summarized in Algorithm 1.

Note our Stein Thompson sampling is a general scheme for exploration/exploitation balance in CMABs. The techniques in [Russo et al., 2017, Kawale et al., 2015] can be adapted in this framework; we leave this for future work.

Algorithm 1 Stein Thompson Sampling

```
Require: $\mathcal{D} = \emptyset$; initialize particles $\Theta_0 = \{\boldsymbol{\theta}_i\}_{i=1}^{M}$;
for $t = 0, 1, 2, \ldots, T$ do
    Receive context $\boldsymbol{s}_t \sim P_s$;
    Draw a particle $\hat{\boldsymbol{\theta}}^t$ from $\Theta_t$;
    Select $\boldsymbol{a}_t = \arg\max_{\boldsymbol{a}} \mathbb{E}_{r\sim P_r(\cdot|\boldsymbol{a},\boldsymbol{s}_t;\hat{\boldsymbol{\theta}}_t)} r_t$;
    Observe reward $r_t \sim P_r$, by performing $\boldsymbol{a}_t$;
    Collect observation: $\mathcal{D}_{t+1} = \mathcal{D}_t \cup (\boldsymbol{s}_t, \boldsymbol{a}_t, r_t)$;
    Update $\Theta_{t+1}$, according to SVGD in (9);
end for
```

4.2 MDPs and Stein Policy Gradient

An MDP is a sequential decision-making procedure in a Markovian dynamical system. It can be seen as an extension of the CMAB, by replacing the context with the notion of a system state, that may dynamically change according to the performed actions and previous state. Formally, an MDP defines a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P_s, P_r, r, \gamma \rangle$, which is similar to a CMAB $\mathcal{C}$ except that (*i*) the next state $\boldsymbol{s}_{t+1}$ is now conditioned on state $\boldsymbol{s}_t$ and action $\boldsymbol{a}_t$, *i.e.*, $\boldsymbol{s}_{t+1} \sim P_s(\cdot|\boldsymbol{s}_t, \boldsymbol{a}_t)$; and (*ii*) a discount factor $0 < \gamma < 1$ for the reward is considered. The goal is to find a policy $\pi(\boldsymbol{a}|\boldsymbol{s})$ to maximize the discounted expected reward: $J(\pi) = \mathbb{E}_{P_s,\pi,P_r} \sum_{t=1}^{T} \gamma^t r_t$.

Policy gradient [Sutton and Barto, 1998] is a family of reinforcement learning methods that solves MDPs by iteratively updating the parameters $\boldsymbol{\theta}$ of the policy to maximize $J(\boldsymbol{\theta}) \triangleq J(\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}))$. Instead of searching for a single policy parameterized by $\boldsymbol{\theta}$, we consider adopting an MVG prior for $p(\boldsymbol{\theta})$, and learning its variational posterior distribution $q(\boldsymbol{\theta})$ using S^{2}VGD. Following [Liu et al., 2017], the objective function is modified as:

$$\max_{q} \{\mathbb{E}_{q(\boldsymbol{\theta})}[J(\boldsymbol{\theta})] - \alpha \mathsf{KL}(q\|p)\}, \tag{17}$$

where $\alpha \in [0, +\infty)$ is the temperature hyper-parameter to balance exploitation and exploration in the policy. The optimal distribution is shown to have a simple closed form [Liu et al., 2017]:

$$q(\boldsymbol{\theta}) \propto \exp\left(\frac{1}{\alpha} J(\boldsymbol{\theta})\right) p(\boldsymbol{\theta}). \tag{18}$$

We iteratively approximate the target distribution as:

$$\triangle\boldsymbol{\theta}_i = \frac{\epsilon}{M}\sum_{j=1}^{M}[\nabla_{\boldsymbol{\theta}_j}\left(\frac{1}{\alpha}J(\boldsymbol{\theta}_j)+\log p(\boldsymbol{\theta}_j)\right)\kappa(\boldsymbol{\theta}_i,\boldsymbol{\theta}_j) + \nabla_{\boldsymbol{\theta}_j}\kappa(\boldsymbol{\theta}_j,\boldsymbol{\theta}_i)], \tag{19}$$

where $J(\boldsymbol{\theta})$ can be approximated with REINFORCE [Williams, 1992] or advantage actor critic [Schulman et al., 2016].

We note two advantages of S^{2}VGD in sequential decision-making: (i) the structural priors can characterize the flexible weight uncertainty, thus providing better exploration-exploitation when learning the policies; (ii) the efficient approximation scheme provides accurate representation of the true posterior while maintaining similar online-processing speed.

5 Experiments

To demonstrate the effectiveness of our S^{2}VGD, we first conduct experiments on the standard regression and classification tasks, with real datasets (two synthetic experiments on classification and regression are given in the Supplementary Material). The superiority of S^{2}VGD is further demonstrated in the experiments on contextual bandits and reinforcement learning.

We compare S^{2}VGD with related Bayesian learning algorithms, including VMG [Louizos and Welling, 2016], PBP_MV [Sun et al., 2017], and SVGD [Liu and Wang, 2016]. The RMSprop optimizer is employed if there is no specific declaration. For SVGD-based methods, we use a RBF kernel $\kappa(\boldsymbol{\theta},\boldsymbol{\theta}') = \exp(-\|\boldsymbol{\theta}-\boldsymbol{\theta}'\|_2^2/h)$, with the bandwidth set to $h = \texttt{med}^2/\log M$. [Oates et al., 2016, Gorham and Mackey, 2017] Here `med` is the median of the pairwise distance between particles. The hyper-parameters $a_\ell = 1, b_\ell = 0.1$. The experimental codes of this paper are available at: `https://github.com/zhangry868/S2VGD`.

We first study the role of hyperparameters in S^{2}VGD: the number of Householder transformations K and the number of particles M. This is investigated by a classification task from [Liu and Wang, 2016] on the Covertype dataset with 581,012 data points and 54 features.

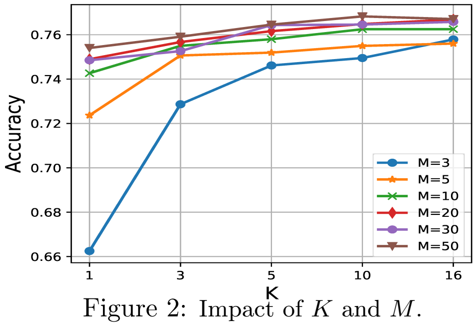


Figure 2: Impact of K and M.

We perform 5 runs for each setting and report the mean of testing accuracy in Figure 2. As expected, increasing M or K improves the performance, as they lead to a more accurate approximation. Interestingly, when M is small, increasing K gives significant improvement. Furthermore, when M is large, the change of K yields similar performance. Therefore, we set $K = 1$ and $M = 20$ unless otherwise specified.

5.1 Regression

We use a single-layer BNN for regression tasks. Following [Li et al., 2015], 10 UCI public datasets are considered: 100 hidden units for 2 large datasets (Protein and YearPredict), and 50 hidden units for the other 8 small datasets. We repeat the experiments 20 times for all datasets except for Protein and YearPredict, which we repeat 5 times and once, respectively, for computation considerations [Sun et al., 2017]. The batch size for the two large datasets is set to 1000, while it is 100 for the small datasets. The datasets are randomly split into 90% training and 10% testing. We adopt the root mean squared error (RMSE) and test log-likelihood as the evaluation criteria.

The experimental results are shown in Table 1, from which we observe that *i*) weight structure information is useful (SVGD is the only method without structure, and it yields inferior performance); *ii*) algorithms with non-parametric assumptions, *i.e.,* the Stein-based methods, perform better; and *iii*) when combined with structure information, our method achieves state-of-the-art results.

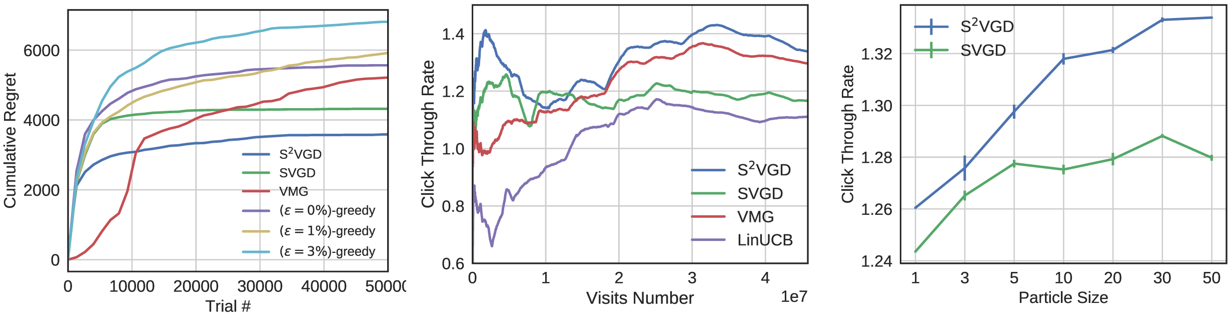
5.2 Classification

We perform the classification tasks on the standard MNIST dataset, which consists of handwritten digits of size 28×28, with 50,000 images for training and 10,000 for testing. A two-layer model 784-X-X-10 with ReLU activation function is used, and X is the number of hidden units for each layer. The training epoch is set to 100. The test errors for network (X-X) sizes 400-400 and 800-800 are reported in Table 2. We observe that the Bayesian methods generally perform better than their optimization counterparts. The proposed S^{2}VGD improves SVGD by a significant margin. Increasing K also improves performance, demonstrating the advantages of incorporating structured weight uncertainty into the model. See [Li et al., 2016a, Louizos and Welling, 2016, Blundell et al., 2015] for details on the other methods with which we compare.

We wish to verify that the performance gain of S^{2}VGD is due to the special structural design of the network architecture, rather than the increasing number of model parameters. This is demonstrated by training a NN with 415-415 hidden units using SVGD, which yields test error 1.49%. It has slightly more parameters than our 400-400 network trained by S^{2}VGD (K=10), but worse performance.

Table 1: Averaged predictions with standard deviations in terms of RMSE and log-likelihood on test sets.

	Test RMSE				Test Log likelihood			
Dataset	VMG	PBP_MV	SVGD	S^{2}VGD	VMG	PBP_MV	SVGD	S^{2}VGD
Boston	2.70 ± 0.13	2.76 ± 0.43	2.96±0.10	**2.56 ± 0.33**	-2.46 ± 0.09	-3.01 ± 0.26	-2.50±0.03	**−2.43 ± 0.10**
Energy	0.54 ± 0.02	0.48 ± 0.04	1.37±0.05	**0.38 ± 0.02**	-1.06 ± 0.03	-2.37 ± 0.03	-1.77±0.02	**−0.55 ± 0.04**
Concrete	4.89 ± 0.12	4.66 ± 0.44	5.32±0.10	**4.25 ± 0.37**	-3.01 ± 0.03	-3.22 ± 0.05	-3.08±0.02	**−2.90 ± 0.07**
Kin8nm	0.08 ± 0.00	0.08 ± 0.00	0.09±0.00	**0.07 ± 0.00**	1.10 ± 0.01	0.78 ± 0.02	0.98±0.01	**1.15 ± 0.01**
Naval	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	2.46 ± 0.00	4.37 ± 0.17	4.09±0.01	**4.79 ± 0.05**
CCPP	4.04 ± 0.04	3.91 ± 0.09	4.03±0.03	**3.84 ± 0.08**	-2.82 ± 0.01	-2.81 ± 0.02	-2.82±0.01	**−2.77 ± 0.02**
Winequality	0.61 ± 0.04	0.61 ± 0.02	0.61±0.01	**0.59 ± 0.02**	-0.95 ± 0.01	-0.99 ± 0.07	-0.93±0.01	**−0.90 ± 0.03**
Yacht	0.48 ± 0.18	0.53 ± 0.14	0.86±0.05	**0.47 ± 0.11**	-1.30 ± 0.02	-1.67 ± 0.24	-1.23±0.04	**−0.81 ± 0.14**
Protein	**4.13 ± 0.02**	4.38 ± 0.01	4.61±0.01	4.15 ± 0.04	**−2.84 ± 0.00**	-2.91 ± 0.03	-2.95±0.00	-2.84 ± 0.01
YearPredict	8.78 ± NA	8.84 ± NA	**8.68 ± NA**	8.73 ± NA	-3.59 ± NA	-3.58±NA	-3.62 ± NA	**−3.57 ± NA**



(a) Simulation results on Mushroom (b) New Article Recommendation (c) Particle Size on Yahoo!Today

Figure 3: Experimental results of Contextual Bandits

Table 2: Classification error of FNN on MNIST.

Method	Test Error	
	400-400	800-800
S^{2}VGD (K=10)	**1.36%**	**1.30%**
S^{2}VGD (K=1)	1.43%	1.39%
SVGD	1.53%	1.47%
SGLD	1.64%	1.41%
RMSprop	1.59%	1.43%
RMSspectral	1.65%	1.56%
SGD	1.72%	1.47%
VMG, variational dropout	1.15%	-
BPB, Gaussian	1.82%	1.99%
BPB, scale mixture	**1.32%**	1.34%
SGD, dropout	1.51%	1.33%

5.3 Contextual Bandits

Simulation We first simulate a contextual-bandit problem with the UCI mushrooms dataset. Following [Blundell et al., 2015], the provided features of each mushroom are regarded as the context. A reward of 5 is given when an agent eats an edible mushroom. Otherwise, if a mushroom is poisonous and the agent eats it, a reward of -10 or 5 will be received, both with probability 0.5; if the agent decides not to eat the mushroom, it receives a reward of 0. We use a two-layer BNN with ReLU and 50 hidden units to represent the policy. We compared our method with a standard baseline, ε-greedy policy with $\varepsilon = 0\%$ (pure greedy), $1\%, 3\%$, respectively [Sutton and Barto, 1998].

We evaluate the performance of different algorithm by cumulative regret [Sutton and Barto, 1998], a measure of the loss caused by playing suboptimal bandit arms. The results are plotted in Figure 3(a). Thompson sampling with 3 different strategies to update policy are considered: S^{2}VGD, SVGD and VMG. S^{2}VGD shows lower regret at the beginning of learning than SVGD, and the lowest final cumulative regret among all methods. We hypothesize that our method captures the internal weight correlation, and the structural uncertainty can effectively help the agent learn to make less mistakes in exploration with less observations.

News Article Recommendation We consider personalized news article recommendation on Yahoo! [Li et al., 2010], where each time a user visits the portal, a news article from a dynamic pool of candidates is recommended based on the user's profile (context). The dataset contains 45,811,883 user visits to the Today Module in a 10-day period in May 2009. For each visit, both the user and each of the 20 candidate articles are associated with a feature vector of 6 dimensions [Li et al., 2010].

The goal is to recommend an article to a user based on its behavior, or, formally, maximize the total number of clicks on the recommended articles. The procedure is regraded as a CMAB problem, where articles are treated as arms. The reward is defined to be 1 if the article is clicked on and 0 otherwise. A one-layer NN with ReLU and 50 hidden units is used as the policy network. The classic LinUCB [Li et al., 2010] is also used as the baseline. The performance is evaluated by an unbiased offline evaluation protocol: the average normalized accumulated click-through-rate (CTR) in every 20000 observations [Li et al., 2010, 2011]. The normalized CTRs are plotted in Figure 3(b). It is clear that S^{2}VGD consistently outperforms other methods. The fact that S^{2}VGD and VMG perform better than SVGD and the baseline LinUCB indicates that structural information helps algorithms to better balance exploration and exploitation.

To further verify the influence of the particle size M on the sequential decision-making, we vary the M from 1 to 50 on the-first-day data. All algorithms are repeated 10 times and their mean performances are plotted in Figure 3(c). We observe that the CTR keeps improving when M becomes larger. S^{2}VGD dominates the performance of SVGD with much higher CTRs, the gap becomes larger as M increases. Since larger M typically leads to more accurate posterior estimation of the policy, indicating again that the accurately learned structural uncertainty are beneficial for CMABs.

5.4 Reinforcement Learning

We apply our S^{2}VGD to policy gradient learning. All experiments are conducted with the OpenAI `rllab` toolkit [Duan et al., 2016]. Three classical continuous control tasks are considered: Cartpole Swing-Up, Double Pendulum, and Cartpole. Following the settings in [Liu et al., 2017, Houthooft et al., 2016], the policy is parameterized as a two-layer (25-10 hidden units) neural network with `tanh` as the activation function. The maximal length of horizon is set to 500. SVGD and S^{2}VGD use a sample size of 10000 for policy gradient estimation, and $M = 16$. For the easy task, Cartpole, all agents are trained for 100 episodes. For the two complex tasks, Cartpole Swing-Up and Double Pendulum, all agents are trained up to 1000 episodes. We consider two different methods to estimate the gradients: REINFORCE [Williams, 1992] and advantage actor critic (A2C) [Schulman et al., 2016]

Figure 4 plots the mean (dark curves) and standard derivation (light areas) of discounted rewards over 5 runs. In all tasks and value-estimation setups, S^{2}VGD converges faster than SVGD and finally converges to higher average rewards. The results are even comparable to [Houthooft et al., 2016], in which a subtle reward mechanism is incorporated to encourage exploration. It demonstrates that simply adding structural information on policy networks using S^{2}VGD improves the agent's exploration ability. We also add a baseline method called SVGD* that applies SVGD to train a network of similar size (25-16 hidden units) with the one reparameterized by S^{2}VGD (K=4). The fact that S^{2}VGD (K=4) converges better than SVGD* demonstrates that our structural uncertainty is key to excellent performance.

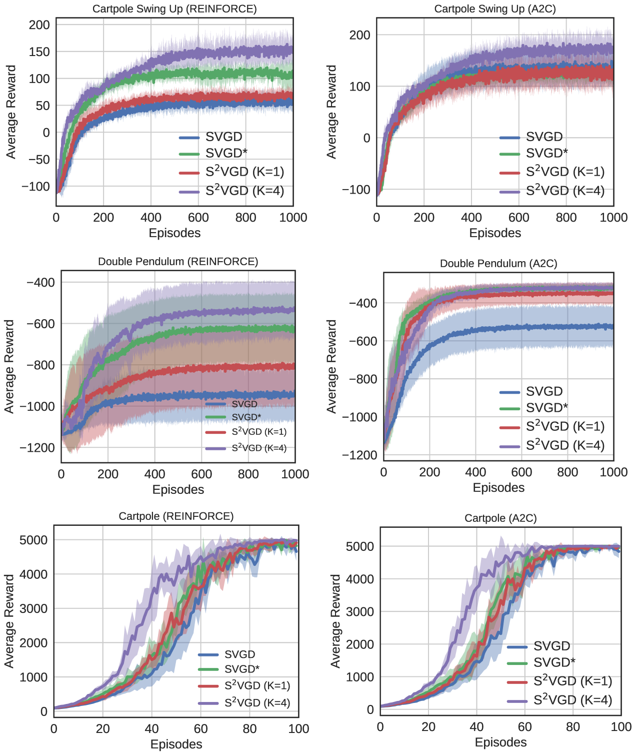


Figure 4: Learning curves by S^{2}VGD and SVGD with REINFORCE (left) and A2C (right).

6 Conclusions

We have proposed S^{2}VGD, an efficient Bayesian posterior learning scheme for the weights of BNNs with structural MVG priors. To achieve this, we derive a new reparametrization for the MVG to unify previous structural priors, and adopt the SVGD algorithm for accurate posterior learning. By transforming the MVG into a lower-dimensional representation, S^{2}VGD avoids computation of related kernel matrices in high-dimensional space. The effectiveness of our framework is validated on several real-world tasks, including regression, classification, contextual bandits and reinforcement learning. Extensive experimental results demonstrate its superiority relative to related algorithms.

Our empirical results on sequential decision-making suggest the benefits of including inter-weight structure within the model, when computing policy uncertainty for online decision-making in an uncertain environments. More sophisticated methods for leveraging uncertainty for exploration/exploration balance may be a promising direction for future work. For example, explicitly encouraging exploration using learned structural uncertainty [Houthooft et al., 2016].

Acknowledgements We acknowledge Qiang Liu and Yang Liu for making their code public. We thank Chenyang Tao for proofreading. This research was supported in part by ARO, DARPA, DOE, NGA, ONR and NSF.

References

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *ICML*, 2015.

Changyou Chen and Ruiyi Zhang. Particle optimization in stochastic gradient mcmc. *arXiv:1711.10927*, 2017.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *ICML*, 2016.

Yihao Feng, Dilin Wang, and Qiang Liu. Learning to draw samples with amortized stein variational gradient descent. 2018.

Meire Fortunato, Charles Blundell, and Oriol Vinyals. Bayesian recurrent neural networks. *arXiv preprint arXiv:1704.02798*, 2017.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, 2016.

Zhe Gan, Chunyuan Li, Changyou Chen, Yunchen Pu, Qinliang Su, and Lawrence Carin. Scalable bayesian learning of recurrent neural networks for language modeling. *ACL*, 2017.

Soumya Ghosh, Francesco Maria Delle Fave, and Jonathan Yedidia. Assumed density filtering methods for learning bayesian neural networks. In *AAAI*, 2016.

Gene H Golub and Charles F Van Loan. *Matrix Computations.* 2012.

Jackson Gorham and Lester Mackey. Measuring sample quality with kernels. *arXiv:1703.01717*, 2017.

Arjun K Gupta and Daya K Nagar. *Matrix Variate Distributions.* 1999.

José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *ICML*, 2015.

Geoffrey E Hinton, Nitish Srivastava, and Kevin Swersky. Rmsprop: Divide the gradient by a running average of its recent magnitude. *Neural Networks for Machine Learning, Coursera*, 2012.

Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *NIPS*, 2016.

Jaya Kawale, Hung H Bui, Branislav Kveton, Long Tran-Thanh, and Sanjay Chawla. Efficient thompson sampling for onlineï£ij matrix-factorization recommendation. In *NIPS*, 2015.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

J Zico Kolter and Andrew Y Ng. Near-bayesian exploration in polynomial time. In *ICML*, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

Chunyuan Li, Changyou Chen, David E Carlson, and Lawrence Carin. Preconditioned stochastic gradient langevin dynamics for deep neural networks. In *AAAI*, 2016a.

Chunyuan Li, Andrew Stevens, Changyou Chen, Yunchen Pu, Zhe Gan, and Lawrence Carin. Learning weight uncertainty with stochastic gradient mcmc for shape classification. In *CVPR*, 2016b.

Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *WWW*, 2010.

Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *WSDM*, 2011.

Yingzhen Li, José Miguel Hernández-Lobato, and Richard E Turner. Stochastic expectation propagation. In *NIPS*, 2015.

Qiang Liu. Stein variational gradient descent as gradient flow. In *NIPS*, 2017.

Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *NIPS*, 2016.

Yang Liu, Prajit Ramachandran, Qiang Liu, and Jian Peng. Stein variational policy gradient. In *UAI*, 2017.

Christos Louizos and Max Welling. Structured and efficient variational deep learning with matrix gaussian posteriors. In *NIPS*, 2016.

David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural Computation*, 1992.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

Chris J Oates, Jon Cockayne, François-Xavier Briol, and Mark Girolami. Convergence rates for a class of estimators based on stein's identity. *arXiv:1603.03220*, 2016.

Yunchen Pu, Liqun Chen, Shuyang Dai, Weiyao Wang, Chunyuan Li, and Lawrence Carin. Symmetric variational autoencoder and connections to adversarial learning. 2017a.

Yunchen Pu, Zhe Gan, Ricardo Henao, Chunyuan Li, Shaobo Han, and Lawrence Carin. Stein variational autoencoder. 2017b.

Daniel Russo, David Tse, and Benjamin Van Roy. Time-sensitive bandit learning and satisficing thompson sampling. *arXiv:1704.09028*, 2017.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *ICLR*, 2016.

David Silver, Aja Huang, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 2016.

Shengyang Sun, Changyou Chen, and Lawrence Carin. Learning structured weight uncertainty in bayesian neural networks. In *AISTATS*, 2017.

Xiaobai Sun and Christian Bischof. A basis-kernel representation of orthogonal matrices. *SIAM Journal on Matrix Analysis and Applications*, 1995.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. 1998.

William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 1933.

Jakub M Tomczak and Max Welling. Improving variational auto-encoders using householder flow. *arXiv:1611.09630*, 2016.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 1992.

Yizhe Zhang, Xiangyu Wang, Changyou Chen, Ricardo Henao, Kai Fan, and Lawrence Carin. Towards unifying hamiltonian monte carlo and slice sampling. In *NIPS*, 2016.

Yizhe Zhang, Changyou Chen, Zhe Gan, Ricardo Henao, and Lawrence Carin. Stochastic gradient monomial gamma sampler. 2017.