Neural Control Variates for Monte Carlo Variance Reduction

Ruosi Wan[1], Mingjun Zhong[2], Haoyi Xiong[3], and Zhanxing Zhu[1,4,5]✉

[1] Center for Data Science, Peking University, Beijing, China
{ruoswan,zhanxing.zhu}@pku.edu.cn
[2] School of Computer Science, University of Lincoln, United Kingdom
mzhong@lincoln.ac.uk
[3] Big Data Lab, Baidu Inc., Beijing, China xionghaoyi@baidu.com
[4] School of Mathematical Sciences, Peking University, Beijing, China
[5] Beijing Institute of Big Data Research, Beijing, China

Abstract. In statistics and machine learning, approximation of an intractable integration is often achieved by using the unbiased Monte Carlo estimator, but the variances of the estimation are generally high in many applications. Control variates approaches are well-known to reduce the variance of the estimation. These control variates are typically constructed by employing predefined parametric functions or polynomials, determined by using those samples drawn from the relevant distributions. Instead, we propose to construct those control variates by learning neural networks to handle the cases when test functions are complex. In many applications, obtaining a large number of samples for Monte Carlo estimation is expensive, the adoption of the original loss function may result in severe overfitting when training a neural network. This issue was not reported in those literature on control variates with neural networks. We thus further introduce a constrained control variates with neural networks to alleviate the overfitting issue. We apply the proposed control variates to both toy and real data problems, including a synthetic data problem, Bayesian model evidence evaluation and Bayesian neural networks. Experimental results demonstrate that our method can achieve significant variance reduction compared to other methods.

Keywords: Control variates · Neural networks · Variance reduction · Monte Carlo method.

1 Introduction

Most of modern machine learning and statistical approaches focus on modelling complex data, where manipulating high-dimensional and multi-modal probability distributions is of great importance for model inference and learning. Under this circumstance, evaluating the expectation of certain function $f(\boldsymbol{\theta})$ with respect to a probability distribution $p(\boldsymbol{\theta})$ is ubiquitous,

$$\mu = \mathbb{E}_{\boldsymbol{\theta}\sim p(\boldsymbol{\theta})}[f(\boldsymbol{\theta})] = \int f(\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}, \tag{1}$$

where the random variable of interest $\boldsymbol{\theta} \in \mathbb{R}^D$ is typically high-dimensional.

However, in complex models, the integration is often analytically intractable. This drives the development of sophisticated Monte Carlo methods to facilitate efficient computation [15]. The Monte Carlo method is naturally employed to approximate the expectation, i.e.,

$$\mu \approx \frac{1}{n} \sum_{i=1}^{n} f(\boldsymbol{\theta}_i), \tag{2}$$

where $\{\boldsymbol{\theta}_i\}_{i=1}^n$ are samples drawn from the distribution $p(\boldsymbol{\theta})$. According to the central limit theorem, this estimator converges to μ at the rate $O(1/\sqrt{n})$. For high-dimensional and complex models, *when $p(\boldsymbol{\theta})$ is difficult to sample from* [11] or *the test function f is expensive to evaluate* [5], a "large-n" estimation is computationally prohibited. This directly leads to a high-variance estimator. Therefore, with a limited number of samples, how to reduce the variance of Monte Carlo estimations emerges as an essential issue for its practical use.

Along this line, various variance reduction methods have been introduced in the literature of statistics and numerical analysis. One category aims to develop appropriate samplers for variance reduction, including importance sampling and its variants [2], stratified sampling techniques [16], multi-level Monte Carlo [4] and other sophisticated methods based on Markov chain Monte Carlo (MCMC) [15]. Another category of variance reduction methods is called control variates [1,10,14,13,8,18]. These methods take advantage of random variables with known expectation values, which are negatively correlated with the test function under consideration. Control variates techniques can fully employ the available samples to reduce the variance, which is popular due to its efficiency and effectiveness.

However, existing control variates approaches have several limitations. Firstly, most existing methods use a linear or quadratic form to represent the control function [10,14]. Although these control functions have closed forms, the representation power of them is very limited particularly when the test function $f(\boldsymbol{\theta})$ is complex and non-linear. Control functionals were proposed recently to tackle this problem [13]. However, these estimators may significantly suffer from a curse of dimensionality [12]. Secondly, when the available samples are scarce, optimizing the control variates only based on a small, number of samples might overfit, which means that it is difficult to generalize on the samples obtained later. These restrictions limit their practical performance.

In order to overcome the first issue, some works [8,**?**] employed neural networks to represent the control variates, utilizing the capability of a neural network to represent a complex test function. We name these methods as "Neural Control Variates" (**NCV**). Unfortunately, in the scenario of learning neural networks, applying the commonly used loss function to reduce variance causes severe over-fitting issue, particularly when available training sample size is small. Therefore, we introduce "Constrained Neural Control Variates" (**CNCV**) which makes constraints on the control variates for alleviating the over-fitting issue. Our method is particularly suitable for the cases when the sample space is high-dimensional

or the samples from $p(\boldsymbol{\theta})$ is hard to obtain. We demonstrate the effectiveness of our approach on both synthetic and real machine learning tasks, including 1) expectation of a complex function under the mixture of Gaussian distributions, 2) Bayesian model evidence evaluation and 3) Bayesian neural networks. We show that CNCV achieved the best performance comparing to the state-of-the-art methods in literature.

2 Control Variates

The generic control variates aims to estimate the expectation $\mu = \mathbb{E}_{p(\boldsymbol{\theta})}[f(\boldsymbol{\theta})]$ with reduced variance. The principle behind the control variates relies on constructing an auxiliary function $\tilde{f}(\boldsymbol{\theta}) = f(\boldsymbol{\theta}) + g(\boldsymbol{\theta})$ such that

$$\mathbb{E}_{p(\boldsymbol{\theta})}[g(\boldsymbol{\theta})] = 0. \tag{3}$$

Thus the desired expectation can be replaced by that of the auxiliary function

$$\mu = \mathbb{E}_{p(\boldsymbol{\theta})}[f(\boldsymbol{\theta})] = \mathbb{E}_{p(\boldsymbol{\theta})}[\tilde{f}(\boldsymbol{\theta})]. \tag{4}$$

It is possible to obtain a variance-reduced Monte Carlo estimator by *selecting or optimizing* $g(\boldsymbol{\theta})$ so that the variance $\mathbb{V}_{p(\boldsymbol{\theta})}[\tilde{f}(\boldsymbol{\theta})] < \mathbb{V}_{p(\boldsymbol{\theta})}[f(\boldsymbol{\theta})]$. Intuitively, variance reduction can be achieved when $g(\boldsymbol{\theta})$ is negatively correlated with $f(\boldsymbol{\theta})$ under $p(\boldsymbol{\theta})$, since much of the randomness "cancels out" in the auxiliary function $\tilde{f}(\boldsymbol{\theta})$.

The selection of an appropriate form of control function $g(\boldsymbol{\theta})$ is crucial for the performance of variance reduction. A tractable class of so called zero-variance control variates was proposed in [1,10]. Those control variates are expressed as a function of the gradient of the log-density, $\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta})$, i.e. the score function $\boldsymbol{s}(\boldsymbol{\theta})$. Concretely, it has the following form

$$g(\boldsymbol{\theta}) = \Delta_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}) \cdot \nabla_{\boldsymbol{\theta}} \log(p(\boldsymbol{\theta})), \tag{5}$$

where the gradient operator $\nabla_{\boldsymbol{\theta}} = [\partial/\partial\theta_1, \ldots, \partial/\partial\theta_D]^T$, the Laplace operator $\Delta_{\boldsymbol{\theta}} = \sum_{i=1}^{D} \partial^2/\partial\theta_i^2$, and "$\cdot$" denotes the inner product. The function $Q(\boldsymbol{\theta})$ is often referred to as the trial function. The target is now to find a trial function so that $g(\boldsymbol{\theta})$ and $f(\boldsymbol{\theta})$ are negatively correlated. This could thus reduce the variance of the Monte Carlo estimation. As the trial function could be arbitrary under those mild conditions given in [10], a parametric function could be used for $Q(\boldsymbol{\theta})$. For example, when $Q(\boldsymbol{\theta}) = \boldsymbol{a}^T\boldsymbol{\theta}$, which is a first degree polynomial function, the auxiliary function becomes

$$\tilde{f}(\boldsymbol{\theta}) = f(\boldsymbol{\theta}) + \boldsymbol{a}^T \boldsymbol{s}(\boldsymbol{\theta}) \tag{6}$$

as was proposed in [10]. The optimal choice of the parameter $\boldsymbol{a}$ that minimizes the variance of $\tilde{f}(\boldsymbol{\theta})$ is $\boldsymbol{a} = -\boldsymbol{\Sigma}_{\boldsymbol{ss}}^{-1}\boldsymbol{\sigma}(\boldsymbol{s}, f)$, where $\boldsymbol{\Sigma}_{\boldsymbol{ss}} = \mathbb{E}[\boldsymbol{s}\boldsymbol{s}^T]$, $\boldsymbol{\sigma}(\boldsymbol{s}, f) = \mathbb{E}[\boldsymbol{s}f]$. Obviously, the representation power of these polynomials is limited, and therefore control functionals have been proposed recently where the trial function is

stochastic. For example, the trial function could be a kernel function [13]. In order for using these control variates, we firstly estimate these required parameters in the trial function by using some training samples $\{\boldsymbol{\theta}_i\}_{i=1}^n$. Then the learned control variates can be used for test samples.

However, there are several drawbacks of the current zero-variance techniques:

- *Dilemma between effectiveness and efficiency.* Although increasing the order of polynomial could potentially increase the representation power and reduce more variance, the number the parameters needs to be learned would grow exponentially. As pointed out by [10], when quadratic polynomials are used, $Q(\boldsymbol{\theta}) = \boldsymbol{a}^T\boldsymbol{\theta} + \boldsymbol{\theta}^T\boldsymbol{B}\boldsymbol{x}/2$, the number of parameters will be $D(D+3)/2$. Thus, finding the optimal coefficients requires dealing with $\boldsymbol{\Sigma_{ss}}$ which is a matrix of dimension of order D^2. Similar issue occurs when employing the control functionals. This makes the use of high order polynomials computationally expensive when faced with high-dimensional sampling spaces.
- *Poor generalization with small sample size.* With small sizes of training samples and complex $p(\boldsymbol{\theta})$, the learned control variates could potentially overfit the training samples, i.e. generalize poorly over new samples. This is because a small size of training samples might be insufficient for representing the full distributional information of $p(\boldsymbol{\theta})$.

These limitations motivate the development of neural control variates and a novel loss function to alleviate overfitting issue when learning the neural control variates, which will be elaborated below.

3 Neural Control Variates

Firstly, we focus on alleviating the dilemma between effectiveness and efficiency on designing control variates in high-dimensional sample space. To this end, the trail function is designed as a neural network [8,18], we name this strategy as neural control variates (NCV). Equipped with neural network, their excellent capability of representing complex functions and overcoming the curse of dimensionality can be fully employed in high-dimensional scenarios [6].

Instead of relying on the control variates (5) introduced in [10], we use the following Stein control variates based on Stein identity [17,13],

$$g(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \cdot \Phi(\boldsymbol{\theta}) + \Phi(\boldsymbol{\theta}) \cdot \nabla_{\boldsymbol{\theta}} \log(p(\boldsymbol{\theta})), \tag{7}$$

where $\Phi(\boldsymbol{\theta})$ is the trial function. Note that in order for $\mathbb{E}[g(\boldsymbol{\theta})] = 0$, we assume mild zero boundary conditions on Φ, such that $p(\theta)\Phi(\theta) = 0$ at the boundary or $\lim_{\|x\|\to\infty} p(\theta)\Phi(\theta) = 0$ [10,9,13]. Compared with Eq (5), Stein control variates is preferred due to its computational advantages since evaluating the second order derivatives of the trial function is avoided. Note that when the trial function $Q(\boldsymbol{\theta})$ is a linear or quadratic polynomial, the Stein trial function $\Phi(\boldsymbol{\theta})$ is constant or linear, respectively.

We now represent the trial function $\Phi(\boldsymbol{\theta})$ by a neural network $\Phi(\boldsymbol{\theta}; \boldsymbol{w})$ parameterized by the weights $\boldsymbol{w}$. The control function becomes $g(\boldsymbol{\theta}; \boldsymbol{w}) =$

$\nabla_{\boldsymbol{\theta}} \cdot \Phi(\boldsymbol{\theta}; \boldsymbol{w}) + \Phi(\boldsymbol{\theta}; \boldsymbol{w}) \cdot \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta})$. In order for variance reduction, we solve the following optimization problem

$$\min_{\boldsymbol{w}} \mathbb{V}_{p(\boldsymbol{\theta})}[f(\boldsymbol{\theta}) + g(\boldsymbol{\theta}; \boldsymbol{w})], \tag{8}$$

which does not have a closed-form in general. Typically, it is assumed that the variance could be approximated by using independent Monte Carlo samples and so the optimization problem is given by

$$\min_{\boldsymbol{w}} \frac{1}{n} \sum_{i=1}^{n} [f(\boldsymbol{\theta}_i) + g(\boldsymbol{\theta}_i; \boldsymbol{w})]^2 - (\mu_0 + \mu_g)^2, \tag{9}$$

where $\mu_0 = E(f(\theta))$ and $\mu_g = E(g(\boldsymbol{\theta}; \boldsymbol{w})) = 0$. Instead, the following optimization problem will be solved

$$\min_{\boldsymbol{w}} \frac{1}{n} \sum_{i=1}^{n} [f(\boldsymbol{\theta}_i) + g(\boldsymbol{\theta}_i; \boldsymbol{w})]^2 \tag{10}$$

where $\{\boldsymbol{\theta}_i\}_{i=1}^n$ are samples drawn from $p(\boldsymbol{\theta})$. Standard back-propagation techniques and stochastic gradient descent (SGD) can then be adopted to obtain the optimal weights of the neural networks.

Unfortunately, when the distribution $p(\boldsymbol{\theta})$ is high-dimensional and multi-modal, such as Bayesian neural networks, it would be very expensive to draw many samples for training control variates. Given a limited computational budget, it typically produces a rather small number of samples that are not sufficient for learning the control variates. Consequently, the learned parameters for control variates can easily overfit over the training samples, and thus could not generalize well on new samples drawn from $p(\boldsymbol{\theta})$. This overfitting phenomenon was not noticed in [8,18], since the considered applications in their scenarios only involve either a simple probability distribution $p(\boldsymbol{\theta})$ or simple target function $f(\boldsymbol{\theta})$.

Therefore, in the following, we propose a new objective function for learning the neural control variates to alleviate the overfitting; and demonstrate its benefits in various applications.

4 Constrained Neural Control Variates

In this section, we propose constrained neural control variates (CNCV) for alleviating overfitting. Now we take a closer look at why the original objective function of NCV tends to bring a poor control variates if one optimizes the Eq.(10) in the scenario that only a small number of samples from $p(\boldsymbol{\theta})$ are available.

Firstly we note that the objective functions in Eq. (9) and Eq. (10) are not the same although μ_0 is a constant, because the variance must be non-negative. For example, if we have a small number of samples, the learned neural network for g could overfit the data so that the objective function in Eq. (10) could hit the global minimum 0 due to the powerful capacity in approximation of the

neural networks. Therefore, we have to optimize Eq. (10) with a constraint such that $\frac{1}{n}\sum_{i=1}^{n}[f(\boldsymbol{\theta}_i)+g(\boldsymbol{\theta}_i;\boldsymbol{w})]^2 \geq \mu_0^2$. With this constraint, the solution would be $g(\boldsymbol{\theta}_i;\boldsymbol{w}) = -f(\boldsymbol{\theta}_i)+\mu_0$ when using a small number of samples. Without this constraint, we can easily observe that with a small n and a large-capacity neural network for representing $\Phi(\boldsymbol{\theta};\boldsymbol{w})$, optimizing Eq. (10) can easily result in "point-wise" fitting, $g(\boldsymbol{\theta}_i;\boldsymbol{w}) = -f(\boldsymbol{\theta}_i)$, for each sample $\boldsymbol{\theta}_i$, thus achieving the minimal value of the objective. So it violates the constraint that the population mean of $g(\boldsymbol{\theta};\boldsymbol{w})$ is zero. Therefore, directly minimizing Eq. (10) can cause severe overfitting. We thus propose two strategies for dealing with this issue.

1. **Centering control variates**. Based on our analysis on optimizing Eq. (10), it introduces bias for the true $g(\boldsymbol{\theta};\boldsymbol{w})$. To compensate this bias, we center the function $g(\boldsymbol{\theta};\boldsymbol{w})$ and set $g(\boldsymbol{\theta};\boldsymbol{w}) = \widetilde{g}(\boldsymbol{\theta};\boldsymbol{w}) - \mu$ where μ should be close to μ_0. The parameter μ could also be learned during the training. Now if we substitute g in Eq. (10), the optimal function would be $\widetilde{g}(\boldsymbol{\theta}_i;\boldsymbol{w}) = -f(\boldsymbol{\theta}_i)+\mu$ which would assure the required constraint $\frac{1}{n}\sum_{i=1}^{n}[f(\boldsymbol{\theta}_i)+\widetilde{g}(\boldsymbol{\theta}_i;\boldsymbol{w})]^2 \geq \mu_0^2$. Note that in the following we assume g is a centered function, and so denote $\widetilde{g}$ by g for simplicity.
2. **Regularization**. We prefer a minimized variance of the function g. Thus the other strategy is to control the variance of the function g, $\mathbb{E}[g^2]$, to regularize the complexity of the neural networks.

Combining the two strategies, the novel objective function can be formulated as the following,

$$\min_{\boldsymbol{w},\mu} \frac{1}{n}\sum_{i=1}^{n}\Big[[f(\boldsymbol{\theta}_i)+g(\boldsymbol{\theta}_i;\boldsymbol{w})-\mu]^2 + \lambda g(\boldsymbol{\theta}_i;\boldsymbol{w})^2\Big], \tag{11}$$

where λ is the regularization parameter, and the population variance $\mathbb{V}[g]$ is estimated by its empirical samples as regularization term.

The random initialization of μ can slow down the training process and cause overfitting. To obtain a better performance, two optional initializing strategies could be used:

1. Simply using $\frac{1}{n}\sum_{i=1}^{n} f(\boldsymbol{\theta}_i)$ as the initializing value of μ;
2. Pre-train the model with larger λ till converged, then retain the value of μ, randomly initializing other variables and re-train the model with smaller λ.

When the number of samples n is big enough, strategy 1 is recommended; when n is small or $\mathbb{V}(f(\boldsymbol{\theta}))$ is relatively large compared with $Ef(\boldsymbol{\theta})$, strategy 2 is recommended.

5 Experiments

To evaluate our proposed method, we apply CNCV to a synthetic problem and two real scenarios, which are thermodynamic integration for Bayesian model

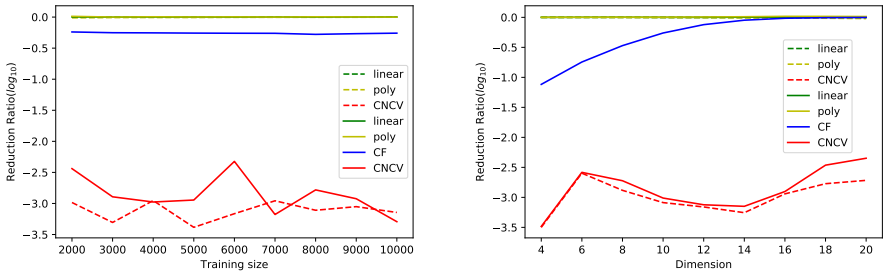


Fig. 1. Synthetic data. (**Left**) Variance reduction ratio *v.s.* number of training samples with $D = 10$; (**Right**) Variance reduction ratio *v.s.* dimension with training sample size $n = 5000$.

evidence evaluation, and Bayesian neural networks. For comparison purposes, control functionals (CF) [13] and polynomial control variates [10,14] are also applied to these problems. The performance of the trained control variates are measured by the variance reduction ratio on the test data set, i.e.,

$$\frac{\mathbb{V}_{p(\boldsymbol{\theta})}[f(\boldsymbol{\theta}) + g(\boldsymbol{\theta})]}{\mathbb{V}_{p(\boldsymbol{\theta})}(f(\boldsymbol{\theta}))]}.$$

We used fully connected neural networks to represent the trial function in all the experiments. We found that for the experiments presented in the following, a medium-sized network is empirically sufficient to achieve good performance. More details on network architectures are provided in Appendix.

5.1 Synthetic Data

To illustrate the advantage of NCV on dealing with high-dimensional problems over other methods, we consider to approximate the expectation of $f(\boldsymbol{\theta}) = \sin(\pi/D \sum_{i=1}^{D} \theta_i)$ where $\boldsymbol{\theta} \in \mathbb{R}^D$ which is a mixture of Gaussians, i.e., $p(\boldsymbol{\theta}) = 0.5\mathcal{N}(-1, \boldsymbol{I}) + 0.5\mathcal{N}(1, \boldsymbol{I})$.

Figure 1 shows the variance reduction ratio on test data ($N = 500$) with respect to varying the number of training samples and the dimensions. In both cases, we can observe that CNCV outperforms linear, quadratic control variates and control functional. Particularly, when increasing the dimensions of $\boldsymbol{\theta}$, CNCV can still achieve much lower variance reduction ratio compared with control functional.

Furthermore, we evaluated the two constraints made on the control variates in the Section 4. To highlight the comparison, we consider the modified function $f(\boldsymbol{\theta}) = 10 \sin(\pi/D \sum_{i=1}^{D} \boldsymbol{\theta}_i) + \mu_0$ where $p(\boldsymbol{\theta}) = 0.5\mathcal{N}(-1, \boldsymbol{I}) + 0.5\mathcal{N}(1, \boldsymbol{I})$, $\boldsymbol{\theta} \in \mathbb{R}^{10}$. Here $\mu_0 \in [0, 9]$ represents the mean of $f(\theta)$, and $\sqrt{var(f(\theta))} \approx 7.5$. To evaluate our methods, we generated 1000 samples, where 500 samples were used for

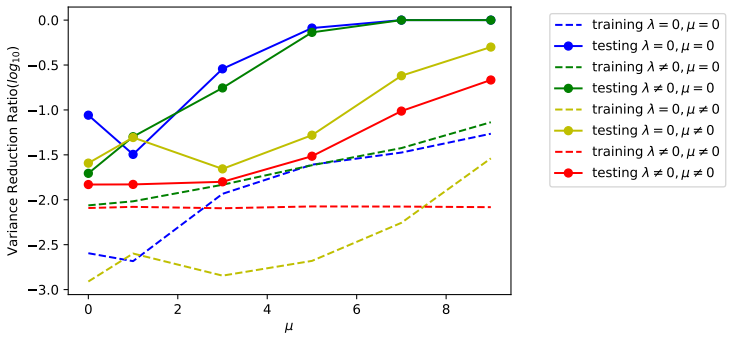


Fig. 2. Variance reduction ratio of four types of NCV versus the oracle mean μ_0. The μ in the control variates was initialized to 0. Dashed and solid lines plot the results on training and test data respectively.

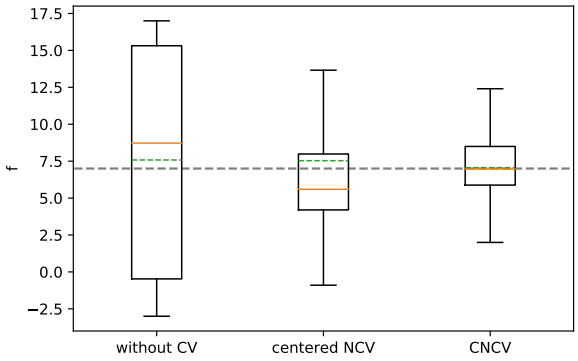


Fig. 3. Boxplot of the samples for the function f on test data. The orange solid line represents the median, the green dashed represents the sample mean, and the grey dashed line represents the oracle mean $\mu_0 = 7$.

training and the rest were used for testing. Four neural control variates schemes with and without the constraints were applied to these samples. These schemes are: 1) not regularized, and not centered ($\lambda = 0, \mu = 0$); 2) regularized, and not centered($\lambda \neq 0, \mu = 0$); 3) not regularized, and centered($\lambda = 0, \mu \neq 0$); 4) regularized, and centered($\lambda \neq 0, \mu \neq 0$).

Figure 2 reports the variance reduction ratio values for training and test data when varying μ_0. It can be shown that NCV without constraints can easily be over-fitted with the training data. As μ_0 increases, NCV with $\lambda = 0, \mu = 0$ and NCV with $\lambda \neq 0, \mu = 0$ were not able to reduce the variance for the test data. This shows that when μ_0 is too large compared to the standard deviation, the

control variates without constraints tends to fit $-f(\boldsymbol{\theta})$ rather than $-f(\boldsymbol{\theta}) + \mu_0$ on the training data, which results in over-fitting.

Figure 3 suggests that CNCV ($\lambda \neq 0, \mu \neq 0$) outperforms all the other methods. The NCV schemes with centered control variates ($\mu \neq 0$) were always better than the ones without centered control variates ($\mu = 0$). We can also see that the regularized control variates ($\lambda \neq 0$) can improve the performance.

The μ was initialized to 0 in all experiments shown in Figure 2. To better understand the effect of the constraints on NCV, we reported the distribution of the samples $f(\boldsymbol{\theta})$ from test sets in Figure 3. It shows that although NCV, which was not regularized but centered ($\lambda = 0, \mu \neq 0$), reduced the variance, the method does introduced bias so that the sample mean was away from the true mean μ_0. The CNCV reduced the variance without introducing bias.

In the following, we apply our proposed CNCV to two difficult problems with small number of samples. In these two cases, original NCV approach tends to severely overfit the training samples, leading to extremely poor generalization performance. Thus, we will not report the results of NCV.

5.2 Thermodynamic Integral for Bayesian Model Evidence Evaluation

In Bayesian analysis, data $\boldsymbol{y}$ is assumed to have been generated under a collection of putative models, $\{\mathcal{M}_i\}$. To compare these candidate models, the Bayesian model evidence is constructed as $p(\boldsymbol{y}|\mathcal{M}_i) = \int p(\boldsymbol{y}|\boldsymbol{\theta}, \mathcal{M}_i)p(\boldsymbol{\theta}|\mathcal{M}_i)d\boldsymbol{\theta}$ where $\boldsymbol{\theta}$ are the parameters associated with model $\mathcal{M}_i$. Unfortunately, for most of the models of interest, this integral is unavailable in closed form. Thus many techniques were proposed to approximate the model evidence. Thermodynamic integration (TI) [3] is among the most promising approach to estimate the evidence. This approach is derived from the standard thermodynamic identity,

$$\log p(\boldsymbol{y}) = \int_0^1 \mathbb{E}_{p(\boldsymbol{\theta}|\boldsymbol{y},t)}[\log p(\boldsymbol{y}|\boldsymbol{\theta})]dt, \tag{12}$$

where $p(\boldsymbol{\theta}|\boldsymbol{y}, t) \propto p(\boldsymbol{y}|\boldsymbol{\theta})^t p(\boldsymbol{\theta})$ ($t \in [0, 1]$) is called power posterior. Note that we have dropped the model indicator $\mathcal{M}_i$ for simplicity. Here t is known as an inverse temperature parameter. In many cases, the posterior expectation $\mathbb{E}_{p(\boldsymbol{\theta}|\boldsymbol{y},t)} \log(p(\boldsymbol{y}|\boldsymbol{\theta}))$ can not be analytically computed, thus the Monte Carlo integration is applied. However, Monte Carlo integration often suffer large variance when sample size is not large enough.

In [14], the zero-variance control variates (5) were used to reduce the variance for TI, so that the posterior expectation is approximated by

$$\frac{1}{N}\sum_{i=1}^{N} \log p(\boldsymbol{y}|\boldsymbol{\theta}_i^t) + \Delta Q_t(\boldsymbol{\theta}_i^t) + \nabla Q_t(\boldsymbol{\theta}_i^t) \cdot \nabla \log(p(\boldsymbol{\theta}_i^t|\boldsymbol{y}, t) \tag{13}$$

where $\{\boldsymbol{\theta}_i^t\}_{i=1}^N$ are drawn from the posterior $p(\boldsymbol{\theta}|\boldsymbol{y}, t)$. In [14], the trial function $Q_t(\boldsymbol{\theta})$ was assumed as a linear or quadratic function, which corresponds to a

constant or linear function for $\Phi(\boldsymbol{\theta})$ in Stein control variates [6]. These methods achieved excellent performance for simple models [14]. However, they are struggling in some scenarios, for example, a negative example which is Goodwin Oscillator given in [14]. Note that Goodwin Oscillator is a nonlinear dynamical system,

$$\frac{d\boldsymbol{x}}{ds} = f(\boldsymbol{x}, s; \boldsymbol{\theta}), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0, \tag{14}$$

where the form of $f(\cdot)$ is provided in Appendix. Assuming within only a subset of time points $\{s_i\}_{i=0}^N$, the solution of (14), i.e. $\boldsymbol{x}(s_i, \boldsymbol{\theta})$, is observed under Gaussian noise $\boldsymbol{\varepsilon}(s) \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})$, where σ^2 denotes the variance of the noise. That means the observation $\boldsymbol{y}(s_i) = \boldsymbol{x}(s_i) + \boldsymbol{\varepsilon}(s_i)$. Then we have the likelihood

$$p(\boldsymbol{y}|\boldsymbol{\theta}, \boldsymbol{x}_0, \sigma) = \prod_{i=1}^{N} \mathcal{N}(\boldsymbol{y}(s_i)|\boldsymbol{x}(s_i; \boldsymbol{\theta}; \boldsymbol{x}_0), \sigma^2 \boldsymbol{I}). \tag{15}$$

The expectation of log likelihood under the power posterior, i.e., $\mathbb{E}_{p(\boldsymbol{\theta}|\boldsymbol{y},t)} \log p(\boldsymbol{y}|\boldsymbol{\theta})$, needs to be evaluated. In [14], the authors demonstrated the failure of polynomial-type of control function since the log-likelihood surface is highly multi-modal and there is much weaker canonical correlation between the scores and the log posterior.

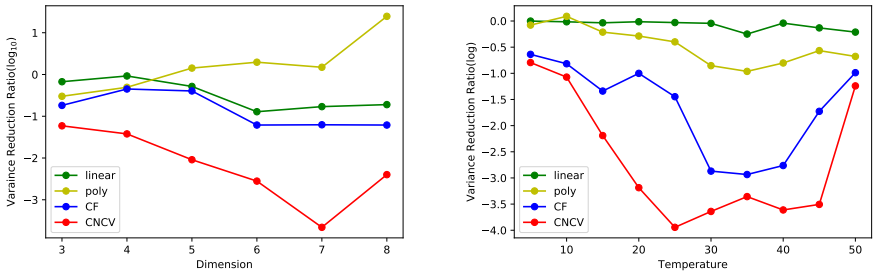
In practice, sampling from Goodwin Oscillator is difficult and computationally expensive since simulating the underlying ordinary differential equation is extremely time-consuming. This directly leads to the situation that the available training samples for control variates are not sufficient. We show in the following that the proposed CNCV can be employed to deal with this issue. To illustrate the benefits of CNCV, we compared it to other methods with various sizes of training samples and temperatures. For comparison purposes we evaluated the variance reduction ratios on both training and test sets (500 samples for test). The experiment settings are the same as those in [14].

Figure 4 shows the experimental results when applying different types of control variates. It can be easily observed that the linear and quadratic methods could hardly reduce the variance of the Goodwin Oscillator model on testing set, while CNCV obtained the lowest variance reduction ratio comparing to all other methods. Control functional can significantly reduce the variance when dimension is low, but CNCV still can get the lowest variance reduction ratio, inspite of the problem dimensions or temperatures.

5.3 Uncertainty Quantification in Bayesian Neural Network

Standard neural network training via optimization is equivalent to maximum likelihood estimation (MLE for short). Given the training samples, $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$,

[6] We will call the trial function $Q(\boldsymbol{\theta})$ as the constant or linear type trial functions $\Phi(\boldsymbol{\theta})$ in the following.



(a) (b)

Fig. 4. Variance reduction ratio on test set of four different types of control variates (linear, quadratic, CF and CNCV). 3000 samples were used for training and the other 3000 samples were used for testing. (a) The average variance reduction ratio on test data versus the problem dimension; (b) The average variance reduction ratio on the test data for different temperatures.

and denote $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^N$ and $\boldsymbol{Y} = \{\boldsymbol{y}_i\}_{i=1}^N$, the weight parameter $\boldsymbol{\theta}$ of the neural networks is estimated by

$$\hat{\boldsymbol{\theta}} = \text{argmax}_{\boldsymbol{\theta}} \sum_{i=1}^{N} \log p(\boldsymbol{y}_i|\boldsymbol{x}_i;\boldsymbol{\theta}) \tag{16}$$

However, the solution of MLE lacks theoretical justification from the probabilistic perspective to deal with the parameter uncertainty as well as structure uncertainty. Moreover, standard neural networks are often susceptible to producing over-confident predictions. Bayesian natural network [11] was introduced for implementing uncertainty quantification. Firstly, one provides a prior distribution over the weights, $p_0(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{0}, \sigma_0^2\boldsymbol{I})$, where σ_0^2 is the variance magnitude. Assuming the likelihood function has the form, $p(\boldsymbol{y}_i|\boldsymbol{x}_i;\boldsymbol{\theta}) = \mathcal{N}\left(\boldsymbol{y}_i|NN(\boldsymbol{x}_i;\boldsymbol{\theta}), \sigma^2\boldsymbol{I}\right)$, then the posterior of the weights $\boldsymbol{\theta}$,

$$p(\boldsymbol{\theta}|\boldsymbol{X},\boldsymbol{Y}) \propto \prod_{i=1}^{N} p(\boldsymbol{y}_i|\boldsymbol{x}_i,\boldsymbol{\theta})p_0(\boldsymbol{\theta}).$$

The uncertainty of the model, typically formulated as the expectation of a specific statistics $f(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y})$ could be computed based on the posterior distribution of the weights,

$$\mu_f = \mathbb{E}[f(\boldsymbol{\theta};\boldsymbol{x},\boldsymbol{y})] = \int f(\boldsymbol{\theta};\boldsymbol{x},\boldsymbol{y})p(\boldsymbol{\theta}|\boldsymbol{X},\boldsymbol{Y})d\boldsymbol{\theta} \tag{17}$$

Due to the analytic intractability of the integral, the expectation of $f(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y})$ is estimated using Monte Carlo integration

$$\mu_f \approx \frac{1}{M}\sum_{i=1}^{M} f(\boldsymbol{\theta}_i, \boldsymbol{x}, \boldsymbol{y}) \tag{18}$$

where $\{\boldsymbol{\theta}_i\}_{i=1}^M$ is drawn from the posterior $p(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{Y})$.

However, the large number of parameters and complex structure of networks make the sampling from the posterior extremely hard. Typically, only a small number of samples could be obtained. Consequently, small sample size and the complex structure of the posterior distribution will lead to a high variance of the estimator (18). Thus, we consider reducing the variance of Monte Carlo estimator by NCVA.

Uncertainty quantification on predictions with out-of-distribution inputs. The neural networks learned with the MLE principal could achieve a high-accuracy performance, when the training data and test data come from the same data distribution. But when an out-of-bag (OOB) sample, i.e., a sample whose label is not included in the training set, is fed into the models, the MLE model is very likely to identify the OOB sample as a certain in-bag class with very high confidence, i.e. the prediction score is close to 1. We hope to construct a robust classifier which won't misclassify the OOB samples with very high confidence. The Bayesian neural network is considered to be effective to handle this situation. However, evaluating the expected prediction score under the posterior of $\boldsymbol{w}$ still suffers large variance issues. Hence we considered to reduce the variance of BNN prediction score and handle the over-confident issues of OOB samples.

We implemented a simple image classification task to evaluate the effectiveness of CNCV. We selected all the images with label "6" and "9" from the MNIST dataset and constructed a convolutional neural network for Bayesian classifier with the output $\{f(\boldsymbol{x}, \boldsymbol{\theta}_i)\}_{i=1}^M$ as the probability of class assignment, where $\boldsymbol{\theta}_i \sim p(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{Y})$ on the two categorizes and select the images with label "8" as the out-of-distribution samples $\boldsymbol{x}_{\text{out}}$ for test. We constructed the control variates to reduce the variance of the estimator $\hat{P}(y = \text{“6”}|\boldsymbol{x}_{\text{out}})$ using NCV,

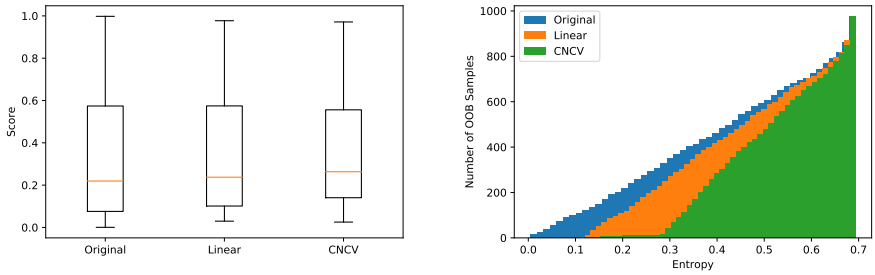
$$\hat{P}(y = \text{“6”}|\boldsymbol{x}_{\text{out}}) = \frac{1}{M}\sum_{i=1}^{M} \hat{P}(y = \text{“6”}|\boldsymbol{x}_{\text{out}}, \boldsymbol{\theta}_i) + g(\boldsymbol{\theta}_i, \boldsymbol{x}_{\text{out}}) \tag{19}$$

The parameters $\{\boldsymbol{\theta}_i\}$ are sampled based on the training set, and hyperparameters are tuned using the validation set. Both the training and validation data are composed of the images with labels ”6” or ”9”. We evaluated the control variates methods on the test set, consisting of the images with the label “8”. We evaluate the control variates by computing the following variance ratio

$$\frac{1}{N}\sum_{i=1}^{N} \frac{\mathbb{V}_{p(\boldsymbol{\theta}|\boldsymbol{X},\boldsymbol{Y})}[p(y_i = \text{"6"}|\boldsymbol{x}_i) + g(\boldsymbol{x}_i, \boldsymbol{\theta})]}{\mathbb{V}_{p(\boldsymbol{\theta}|\boldsymbol{X},\boldsymbol{Y})}[p(y_i = \text{"6"}|\boldsymbol{x}_i)]} \tag{20}$$

Due to the high dimensionality of this problem, quadratic control variates and control functional failed to obtain satisfying variance reduction, and we thus do not report their results.

Figure 5(a) shows that the overall distribution of BNN ensemble prediction does not change significantly, where CNCV produces slightly better results. This is expected since the classifier has not seen the OOB samples during training,



(a) (b)

Fig. 5. Performance of CNCV on BNN with OOB samples (a) boxplot of the prediction score of OOB samples defined in Equation (19). The prediction scores were computed based on BNN ensemble classifier for those variance reduction methods. (b) The accumulated empirical distribution of the entropy computed by prediction scores using Equation (21).

which makes it impossible to yield a stable prediction probability. On the other hand, Figure 5(b) depicts the the entropy of these OOB samples computed using prediction score via BNN:

$$Entropy(x_{OOB}) = -\hat{p}\log(\hat{p}) - (1-\hat{p})\log(1-\hat{p}), \tag{21}$$

where $\hat{p}$ is the BNN prediction score evaluated from Eq. (19). $Entropy(x_{OOB})$ is in the range $[0, \log 2]$. Samples with low entropy close to 0 means they will be classified as 6 or 9 with very high confidence. It could be seen from Figure 5(b) that BNN prediction with control variates has less over-confident scores over OOB samples. That means that BNN prediction with CNCV yields the least over-confident scores compared with vanilla BNN and that with linear control variates.

6 Conclusion

We have proposed neural control variates for variance reduction. We have shown that the neural control variates could have the over-fitting problem when using a small number of samples. To alleviate this over-fitting problem, we proposed constrained neural control variates, where the control variates is centered and regularized. We demonstrated the effectiveness of the proposed methods on synthetic data and two challenging Monte Carlo integration tasks. However, the theoretical justification of the proposed method will be investigated in our future research.

References

1. Assaraf, Roland, and Michel Caffarel. "Zero-variance principle for Monte Carlo algorithms." Physical review letters 83(23):4682 (1999)
2. Jean Cornuet, JEAN-MICHEL MARIN, Antonietta Mira, and Christian PRobert. Adaptive multiple importance sampling.Scandinavian Journal ofStatistics, 39(4):798–812 (2012).
3. Daan Frenkel and Berend Smit.Understanding molecular simulation: fromalgorithms to applications, volume 1. Elsevier (2001).
4. Michael B Giles. Multilevel monte carlo methods. InMonte Carlo andQuasi-Monte Carlo Methods 2012, pages 83–103. Springer (2013)
5. Dave Higdon, Jordan D McDonnell, Nicolas Schunck, Jason Sarich, andStefan M Wild. A bayesian approach for parameter estimation and predictionusing a computationally intensive model.Journal of Physics G: Nuclear andParticle Physics, 42(3):034009 (2015).
6. Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning.nature,521(7553):436, (2015).
7. Chunyuan Li, Changyou Chen, David E Carlson, and Lawrence Carin. Preconditioned stochastic gradient langevin dynamics for deep neural networks.InAAAI, volume 2, page 4 (2016)
8. Hao Liu, Yihao Feng, Yi Mao, Dengyong Zhou, Jian Peng, and Qiang Liu.Action-dependent control variates for policy optimization via stein identity.InICLR, (2018).
9. Qiang Liu and Dilin Wang. Stein variational gradient descent: A generalpurpose bayesian inference algorithm. In:Advances In Neural InformationProcessing Systems, pages 2378–2386 (2016).
10. Antonietta Mira, Reza Solgi, and Daniele Imparato. Zero variance markovchain monte carlo for bayesian estimators. Statistics and Computing,23(5):653–662 (2013).
11. Radford M Neal. Bayesian learning for neural networks, volume 118. Springer Science & Business Media (2012).
12. Chris J Oates, Jon Cockayne, François-Xavier Briol, and Mark Girolami. Convergence rates for a class of estimators based on stein's method. *arXivpreprint arXiv:1603.03220* (2016).
13. Chris J Oates, Mark Girolami, and Nicolas Chopin. Control functionals for monte carlo integration. Journal of the Royal Statistical Society: Series B(Statistical Methodology), 79(3):695–718 (2017).
14. Chris J Oates, Theodore Papamarkou, and Mark Girolami. The controlled thermodynamic integral for bayesian model evidence evaluation. Journal of the American Statistical Association, 111(514):634–645 (2016).
15. Christian P Robert. Monte carlo methods. Wiley Online Library (2004).
16. Reuven Y Rubinstein and Dirk P Kroese. Simulation and the Monte Carlo method, volume 10. John Wiley & Sons (2016).
17. Charles Stein et al. A bound for the error in the normal approximation to the distribution of a sum of dependent random variables. In:Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 2: Probability Theory. The Regents of the University of California (1972).
18. George Tucker, Andriy Mnih, Chris J Maddison, John Lawson, and Jascha Sohl-Dickstein. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. In:Advances in Neural Information Processing Systems, pages 2624–2633 (2017).

A Formulas for Goodwin Oscillator

The nonlinear dynamic system of the Goodwin Oscillator used in [14] is given by:

$$\begin{aligned}
\frac{dx_1}{ds} &= \frac{a_1}{1 + a_2 x_g^{\rho}} - \alpha x_1 \\
\frac{dx_2}{ds} &= k_1 x_1 - \alpha x_2 \\
&\vdots \\
\frac{dx_g}{ds} &= k_{g-1} x_{g-1} - \alpha x_g.
\end{aligned} \tag{22}$$

The solution $\boldsymbol{x}(s; \theta, \boldsymbol{x}_0)$ of this dynamical system depends on the uncertain parameters $\alpha, a_1, a_2, k_1, \ldots, k_{g-1}$. Similar to the settings in [14], we assume $\boldsymbol{x}_0 = [0, \ldots, 0]$ and $\sigma = 0.1$ are both known and take sampling times to be $s = 41, \ldots, 80$. Parameters were assigned independent $\Gamma(2, 1)$ prior distributions. We generated data using $a_1 = 1, a_2 = 3, k_1 = 2, k_2, \ldots, k_{g-1} = 1, \alpha = 0.5$. We generated the posterior samples of the weights using MCMC with parallel tempering. In each dimension cases ($g \in \{3, 4, \cdots, 8\}$) the Markorv Chain runs 100, 000 iterations to ensure converge, 6000 samples randomly drawn from the last 50, 000 iterations were used in the final experiments.

The trial function ϕ used in Goodwin Oscillator is a two layers fully connected neural network, where each layer has 40 neurons. The activation function is the Sigmoid function.

B Uncertainty Quantification in Bayesian Neural Network: Out-of-Bag Sample Detection

The basic model consists of two convolutional layers, two max-pooling layers and a fully connected layers, with kernel size $(5 \times 5 \times 2)$, $(2 \times 3 \times 3 \times 3)$, (147×2) respectively. The prior distribution of the weight was set to standard normal distribution $\mathcal{N}(0, 1)$. The samples of the weights were generated using preconditioned Stochastic Gradient Langevin Dynamic [7]. 1000 samples were generated to construct the Bayesian neural network prediction. The trial function $\phi(\boldsymbol{\theta}, x) : \Theta \times \mathbb{X} \longrightarrow \mathbb{R}$ was defined as:

$$\phi(\boldsymbol{\theta}, x) = \alpha^T h(W_0 \boldsymbol{\theta} + \psi(x)) \tag{23}$$

where $\psi(x)$ consists of two convolutional layers with kernel size $(5 \times 5 \times 2)$, $(2 \times 3 \times 3 \times 3)$, two max-pooling layers and relu activation. $W_0 \in \mathbb{R}^{147 \times 407}$, h is the sigmoid function. and $\alpha \in \mathbb{R}^{147}$. Thus the neural control varaites of the BNN prediction is:

$$g(\boldsymbol{\theta}, x) = \nabla_{\boldsymbol{\theta}} \cdot \phi(\boldsymbol{\theta}, x) + \phi(\boldsymbol{\theta}, x) \nabla_{\boldsymbol{\theta}} \cdot \log p(\boldsymbol{\theta} | \boldsymbol{X}) \tag{24}$$

where $\nabla \cdot f = \sum_i \frac{\partial f}{\partial x_i}$.