GO Hessian for Expectation-Based Objectives

Yulai Cong,* Miaoyun Zhao,* Jianqiao Li, Junya Chen, Lawrence Carin
Department of Electrical and Computer Engineering, Duke University

Abstract

An unbiased low-variance gradient estimator, termed GO gradient, was proposed recently for expectation-based objectives $\mathbb{E}_{q_{\boldsymbol{\gamma}}(\boldsymbol{y})}[f(\boldsymbol{y})]$, where the random variable (RV) $\boldsymbol{y}$ may be drawn from a stochastic computation graph (SCG) with continuous (non-reparameterizable) internal nodes and continuous/discrete leaves. Based on the GO gradient, we present for $\mathbb{E}_{q_{\boldsymbol{\gamma}}(\boldsymbol{y})}[f(\boldsymbol{y})]$ an unbiased low-variance Hessian estimator, named GO Hessian, which contains the deterministic Hessian as a special case. Considering practical implementation, we reveal that the GO Hessian *in expectation* obeys the chain rule and is therefore easy-to-use with auto-differentiation and Hessian-vector products, enabling efficient cheap exploitation of curvature information over deep SCGs. As representative examples, we present the GO Hessian for non-reparameterizable gamma and negative binomial RVs/nodes. Leveraging the GO Hessian, we develop a new second-order method for $\mathbb{E}_{q_{\boldsymbol{\gamma}}(\boldsymbol{y})}[f(\boldsymbol{y})]$, with challenging experiments conducted to verify its effectiveness and efficiency.

1 Introduction

Many machine learning problems can be formulated as an optimization problem involving an expectation. A classic such setup (Robbins and Monro 1951) is of the form

$$\text{Framework I: } \min_{\boldsymbol{\vartheta}} \mathcal{J}(\boldsymbol{\vartheta}) \triangleq \mathbb{E}_{q(\boldsymbol{x})}[h(\boldsymbol{x}, \boldsymbol{\vartheta})], \qquad (1)$$

where the random variable (RV) $\boldsymbol{x}$ obeys a distribution $q(\boldsymbol{x})$ unrelated to the parameters $\boldsymbol{\vartheta}$ of interest, and $h(\boldsymbol{x}, \boldsymbol{\vartheta})$ is a continuous function wrt $\boldsymbol{\vartheta}$. General assumptions making $\mathcal{J}(\boldsymbol{\vartheta})$ (and the following $\mathcal{L}(\boldsymbol{\gamma})$) a valid loss function are omitted for simplicity. In practice one often encounters its finite-sum form $\min_{\boldsymbol{\vartheta}} \frac{1}{N}\sum_{i=1}^{N} h(\boldsymbol{x}_i, \boldsymbol{\vartheta})$ with $q(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N} \delta(\boldsymbol{x} - \boldsymbol{x}_i)$, where $\delta(\cdot)$ is the Dirac delta function (this discrete form is typically an approximation, based on N observed samples drawn from the true underlying data distribution). A popular example of Framework I is maximum-likelihood learning with the data distribution $q(\boldsymbol{x})$ and the negative log-likelihood $h(\boldsymbol{x}, \boldsymbol{\vartheta}) = -\log p(\boldsymbol{x}; \boldsymbol{\vartheta})$, where $p(\boldsymbol{x}; \boldsymbol{\vartheta})$ represents the model.

An alternative framework, attracting increasing attention recently, considers the form

$$\text{Framework II: } \min_{\boldsymbol{\gamma}} \mathcal{L}(\boldsymbol{\gamma}) \triangleq \mathbb{E}_{q_{\boldsymbol{\gamma}}(\boldsymbol{y})}[f(\boldsymbol{y})], \qquad (2)$$

where parameters $\boldsymbol{\gamma}$ of interest determine the distribution $q_{\boldsymbol{\gamma}}(\boldsymbol{y})$ that, for example, models a stochastic computation graph (SCG) (Schulman et al. 2015a; Parmas 2018; Weber et al. 2019).[1] Note in general the function $f(\cdot)$ may also be related to $\boldsymbol{\gamma}$; however, as the generalization is straight-forward, we focus on the setup in (2) for simpler derivations. Popular examples of Framework II include the ELBO in variational inference (Bishop 2006; Kingma and Welling 2014), the generator training objective of generative adversarial networks (Goodfellow et al. 2014), and many objectives associated with reinforcement learning (RL) (Schulman et al. 2015b; Finn, Abbeel, and Levine 2017; Foerster et al. 2018).

Many optimization methods have been proposed for Framework I, utilizing the first-order gradient information (Allen-Zhu 2018; Jin et al. 2019) or exploiting the second-order Hessian information (Tripuraneni et al. 2018; Zhou and Gu 2020). Compared with first-order methods, second-order ones are often characterized by convergence in fewer training iterations to a second-order stationary point, requiring less tweaking of meta-parameters (like learning rate), invariance to linear parameter rescaling, and navigating better when facing pathological curvature in deep learning (Martens 2010; Tripuraneni et al. 2018). For computation and memory efficiency in high-dimensions (like for deep neural networks), recent second-order methods often resort to Hessian-free techniques, *i.e.,* Hessian-vector products (HVP), which can be computed as efficiently as gradients (Pearlmutter 1994) and remove the need to construct the full Hessian (Martens 2010; Kohler and Lucchi 2017; Kasai and Mishra 2018).

In contrast to the classic Framework I, few optimization methods have been proposed for Framework II with general SCG $q_{\boldsymbol{\gamma}}(\boldsymbol{y})$, partially because of the significant challenge in even estimating its gradient with low variance without bias in general/non-reparameterizable (subsequently abbreviated as "non-rep") situations (Cong et al. 2019; Weber et al. 2019).

*Equal Contribution. Correspondence to: Miaoyun Zhao, miaoyun9zhao@gmail.com.

[1] When the RV $\boldsymbol{y}$ is reparameterizable (rep) (Salimans, Knowles et al. 2013; Rezende, Mohamed, and Wierstra 2014), Framework II can be rewritten as Framework I for optimization. To distinguish, we focus on the challenging non-rep setup for Framework II by default, despite the proposed GO Hessian also applies to rep RVs.

For second-order optimization of Framework II, most existing works resort to the log-trick,[2] often suffering from high variance and poor sample efficiency, and therefore seeking help from variance reduction control variates with a potential variance-bias trade-off (Heess et al. 2015; Rothfuss et al. 2019). Moreover, to facilitate the implementation via auto-differentiation (AD), cumbersome designs of surrogate losses and control variates are often necessary (Foerster et al. 2018; Mao et al. 2019), which are challenging when derivatives of different orders are used simultaneously (Farquhar, Whiteson, and Foerster 2019), like in meta reinforcement learning (Finn, Abbeel, and Levine 2017). Therefore, an easy-to-use unbiased (gradient and) Hessian estimator for Framework II, with low variance and high sample efficiency, is highly appealing (Liu, Socher, and Xiong 2019).

Different from existing methods that leverage the log-trick, we follow a different research path (Figurnov, Mohamed, and Mnih 2018; Jankowiak and Obermeyer 2018; Cong et al. 2019) that tries to generalize the classic deterministic derivatives/back-propagation to Framework II. Specifically, based on the GO gradient (Cong et al. 2019), we show that its double application delivers an unbiased low-variance Hessian estimator, termed GO Hessian, for Framework II in (2), where $\boldsymbol{y}$ may be drawn from a SCG consisting of *continuous* rep/non-rep internal nodes[3] and *continuous/discrete* leaves, with neural networks as links. Besides proposing the GO Hessian, our other contributions include:

- we reveal that the GO Hessian contains the deterministic Hessian as a special case; they both show intuitively simple patterns obeying the chain rule, except the GO Hessian exhibits slight differences due to SCG randomness;
- we reveal the GO Hessian is easy to use with existing AD software and HVP, enabling computationally and memory efficient exploitation of curvature information over SCGs; specifically, one can simulate one sample per SCG node in the forward pass through that SCG, followed by doubly backward passes to form a one-sample-based Hessian estimate, which is the same as the deterministic Hessian;
- we derive the GO Hessian for non-rep gamma and negative binomial RVs; we also propose a simple yet effective method to make optimization over gamma RVs more friendly to gradient-based methods;
- we empirically show that the GO Hessian often works well with one sample without variance reduction techniques; combining the GO Hessian with an existing method for Framework I (Tripuraneni et al. 2018), we present a novel second-order method for Framework II, theoretically analyze its convergence, and empirically verify its effectiveness and efficiency with challenging experiments.

2 Preliminary

2.1 General and One-sample (GO) Gradient

Containing as special cases the low-variance reparameterization gradient (Salimans, Knowles et al. 2013; Rezende, Mohamed, and Wierstra 2014) and the pathwise derivative (Figurnov, Mohamed, and Mnih 2018; Jankowiak and Obermeyer 2018)[4], the GO gradient (Cong et al. 2019) serves as a general framework of unbiased low-variance gradient estimates for Framework II in (2), where RV $\boldsymbol{y}$ may be drawn from a SCG with continuous rep/non-rep internal nodes and continuous/discrete leaves. With the GO gradient, one can forward pass through the SCG with *one sample* activated per node to estimate the objective, followed by backward-propagating an unbiased low-variance gradient estimate through each node again to the parameters for updating (see Theorem 3 of Cong et al. (2019)). The low-variance and one-sample properties make the GO gradient easy-to-use in practice, *e.g.,* in variational inference with a complicated inference distribution.

To introduce the approach, the simplest setup, *i.e.,* a single-layer RV $\boldsymbol{y}$ satisfying conditional independence $q_{\boldsymbol{\gamma}}(\boldsymbol{y}) = \prod_v q_{\boldsymbol{\gamma}}(y_v)$, is employed to demonstrate the GO gradient, *i.e.,*

$$\nabla_{\boldsymbol{\gamma}}\mathcal{L}(\gamma) = \nabla_{\boldsymbol{\gamma}}\mathbb{E}_{q_{\boldsymbol{\gamma}}(\boldsymbol{y})}[f(\boldsymbol{y})] = \mathbb{E}_{q_{\boldsymbol{\gamma}}(\boldsymbol{y})}\big[\mathbf{G}_{\boldsymbol{\gamma}}^{q_{\boldsymbol{\gamma}}(\boldsymbol{y})}\mathbb{D}_{\boldsymbol{y}}f(\boldsymbol{y})\big], \quad (3)$$

where $\mathbb{D}_{\boldsymbol{y}}f(\boldsymbol{y}) = \big[\cdots, \mathbb{D}_{y_v}f(\boldsymbol{y}), \cdots\big]^T$ with $\mathbb{D}_{y_v}f(\boldsymbol{y}) \triangleq \nabla_{y_v}f(\boldsymbol{y})$ for continuous y_v while $\mathbb{D}_{y_v}f(\boldsymbol{y}) \triangleq f(\boldsymbol{y}^{v+}) - f(\boldsymbol{y})$ for discrete y_v. $\boldsymbol{y}^{v+} \triangleq [\cdots, y_{v-1}, y_v + 1, y_{v+1}, \cdots]^T$. $\mathbf{G}_{\boldsymbol{\gamma}}^{q_{\boldsymbol{\gamma}}(\boldsymbol{y})} = \big[\cdots, g_{\boldsymbol{\gamma}}^{q_{\boldsymbol{\gamma}}(y_v)}, \cdots\big]$ gathers the *variable-nabla* $g_{\boldsymbol{\gamma}}^{q_{\boldsymbol{\gamma}}(y_v)} \triangleq \frac{-1}{q_{\boldsymbol{\gamma}}(y_v)}\nabla_{\boldsymbol{\gamma}}Q_{\boldsymbol{\gamma}}(y_v)$,[5] which has the intuitive meaning of the "derivative" of a RV y_v wrt its parameters $\boldsymbol{\gamma}$. $Q_{\boldsymbol{\gamma}}(y_v)$ is the CDF of $q_{\boldsymbol{\gamma}}(y_v)$.

With the *variable-nabla*, one can informally interpret $\mathbf{G}_{\boldsymbol{\gamma}}^{q_{\boldsymbol{\gamma}}(\boldsymbol{y})}$ as the "gradient" of the RV $\boldsymbol{y}$ wrt the parameters $\boldsymbol{\gamma}$. Similar intuitive patterns hold for deep SCGs with continuous internal nodes. As an informal summarization, the GO gradient *in expectation* obeys the chain rule and acts like its special case of the classic back-propagation algorithm (Rumelhart and Hinton 1986; Cong et al. 2019).

2.2 Hessian-free Techniques

Developed for efficient implementation of second-order optimization in high-dimensions (like for deep neural networks, where the explicit construction of the full Hessian is prohibitive), Hessian-free techniques (Martens 2010; Byrd et al. 2011) exploit HVP for *implicit* usage of the Hessian information. An example technique to calculate HVP leverages

$$[\nabla^2_{\boldsymbol{\vartheta}}\mathcal{J}(\boldsymbol{\vartheta})]\boldsymbol{p} = \nabla_{\boldsymbol{\vartheta}}\big[[\nabla_{\boldsymbol{\vartheta}}\mathcal{J}(\boldsymbol{\vartheta})]^T\boldsymbol{p}\big], \quad (4)$$

where $\boldsymbol{p}$ is a vector uncorrelated with the parameters $\boldsymbol{\vartheta}$ of interest. For better efficiency than the above 2-backward technique, Pearlmutter (1994) proposed a faster HVP calculation that takes about the same amount of computation as a gradient evaluation. The low-cost HVP is essential because common subsolvers used to search for second-order directions merely exploit Hessian information via HVP (Agarwal et al. 2017; Tripuraneni et al. 2018; Zhou and Gu 2020).

[2] Also named the likelihood ratio, score function, or REINFORCE (Williams 1992) estimator. See Section 3.1 for details.

[3] Note discrete internal nodes are not supported.

[4] See Appendix A for the detailed comparisons among these reparameterization-like gradient estimators.

[5] The notations for the variable-nabla and the variable-hess below are optimized to improve the clarity when deep SCGs are of interest; refer to the derivations in Appendix C.3 and Cong et al. (2019).

2.3 Stochastic Cubic Regularization (SCR)

As a second-order method for Framework I in (1), the SCR (Tripuraneni et al. 2018) searches for a second-order stationary point via iteratively minimizing a local third-order Taylor expansion of the objective $\mathcal{J}(\boldsymbol{\vartheta})$, *i.e.,*

$$\boldsymbol{\vartheta}_{t+1} = \text{argmin}_{\boldsymbol{\vartheta}} \begin{bmatrix} \mathcal{J}(\boldsymbol{\vartheta}_t) + \tilde{\boldsymbol{g}}_t^T(\boldsymbol{\vartheta} - \boldsymbol{\vartheta}_t) + \\ \frac{1}{2}(\boldsymbol{\vartheta} - \boldsymbol{\vartheta}_t)^T \tilde{\mathbf{H}}_t (\boldsymbol{\vartheta} - \boldsymbol{\vartheta}_t) + \frac{\rho}{6}\|\boldsymbol{\vartheta} - \boldsymbol{\vartheta}_t\|^3 \end{bmatrix} \quad (5)$$

where $\tilde{\boldsymbol{g}}_t = \tilde{\nabla}_{\boldsymbol{\vartheta}} \mathcal{J}(\boldsymbol{\vartheta}_t)$ and $\tilde{\mathbf{H}}_t = \tilde{\nabla}^2_{\boldsymbol{\vartheta}} \mathcal{J}(\boldsymbol{\vartheta}_t)$ are the *stochastic* gradient and Hessian at $\boldsymbol{\vartheta}_t$, respectively (often estimated with sub-sampled data batches), ρ is the cubic penalty coefficient, and (5) can be solved efficiently with gradient decent (Carmon and Duchi 2016). Since Newton-like methods are much more tolerant to the Hessian estimation error than that of the gradient (Byrd et al. 2011), one can often use significantly less data samples to calculate the stochastic Hessian for better efficiency (Tripuraneni et al. 2018).

3 GO Hessian for Framework II

Targeting efficient second-order optimization of Framework II in (2), we first propose for it an unbiased low-variance Hessian estimator, termed General and One-sample (GO) Hessian, that is based on the GO gradient (Cong et al. 2019) and is easy-to-use in practice. We then combine the proposed GO Hessian with the SCR (Tripuraneni et al. 2018) to propose a novel second-order method for Framework II.

3.1 GO Hessian

A straight-forward way to estimate the Hessian of Framework II lies in exploiting the log-trick $\nabla_{\boldsymbol{\gamma}} q_{\boldsymbol{\gamma}}(\boldsymbol{y}) = q_{\boldsymbol{\gamma}}(\boldsymbol{y}) \nabla_{\boldsymbol{\gamma}} \log q_{\boldsymbol{\gamma}}(\boldsymbol{y})$, that is,

$$\nabla^2_{\boldsymbol{\gamma}} \mathcal{L}(\boldsymbol{\gamma}) = \mathbb{E}_{q_{\boldsymbol{\gamma}}(\boldsymbol{y})} \begin{bmatrix} f(\boldsymbol{y}) [\nabla_{\boldsymbol{\gamma}} \log q_{\boldsymbol{\gamma}}(\boldsymbol{y})][\nabla_{\boldsymbol{\gamma}} \log q_{\boldsymbol{\gamma}}(\boldsymbol{y})]^T \\ + f(\boldsymbol{y}) \nabla^2_{\boldsymbol{\gamma}} \log q_{\boldsymbol{\gamma}}(\boldsymbol{y}) \end{bmatrix}. \quad (6)$$

However, such a log-trick estimation shows high Monte Carlo (MC) variance in both theory and practice (Rezende, Mohamed, and Wierstra 2014; Ruiz, Titsias, and Blei 2016), often seeking help from variance-reduction techniques (Grathwohl et al. 2017; Mao et al. 2019). Besides, for practical implementation with AD, cumbersome designs of surrogate losses and control variates are often necessary (Foerster et al. 2018; Farquhar, Whiteson, and Foerster 2019).

Different from the above method based on the log-trick, our GO Hessian doesn't require additional designs of surrogate losses and it estimates the curvature of Framework II in a pathwise manner, like the classic deterministic Hessian (as detailed below); moreover, it empirically shows low variance, often working well in practice with one-sample-based estimation without variance reduction techniques (see Figure 1 and the experiments).[6] For simplified notations, we first employ the single-layer setup with $q_{\boldsymbol{\gamma}}(\boldsymbol{y}) = \prod_v q_{\boldsymbol{\gamma}}(y_v)$ to demonstrate our main results, which are then generalized to deep SCGs with continuous rep/non-rep internal nodes and continuous/discrete leaves. Proofs are given in Appendix C.

[6]Discussions on the lower variance of the GO Hessian than the log-trick estimation are provided in Appendix E.4.

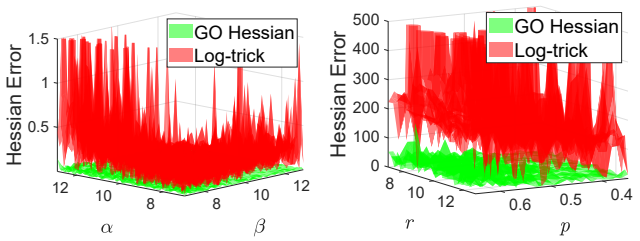


Figure 1: One-sample-based variance comparisons of the log-trick estimation and the GO Hessian on (left) $\nabla^2_{\{\alpha,\beta\}} \mathbb{E}_{\text{Gam}(\alpha,\beta)}[\log \frac{\text{Gam}(\alpha,\beta)}{\text{Gam}(10,10)}]$ and (right) $\nabla^2_{\{r,p\}} \mathbb{E}_{\text{NB}(r,p)}[\log \frac{\text{NB}(r,p)}{\text{NB}(10,0.5)}]$. See Appendix E for details.

Motivated by the GO gradient that leverages the integration-by-parts to form a gradient estimate of Framework II in a pathwise manner, one may try to naively apply the integration-by-parts twice to form a Hessian estimate; however, as shown in Appendix B, such naive derivations lead to complex calculation terms that are difficult to interpret or implement.

Fortunately, we find that (i) the *variable-nabla*, *e.g.,* $g_{\boldsymbol{\gamma}}^{q_{\boldsymbol{\gamma}}(y_v)}$ that constitutes $\mathbf{G}_{\boldsymbol{\gamma}}^{q_{\boldsymbol{\gamma}}(\boldsymbol{y})}$ in (3), is often differentiable with a simple expression (see Table 3 of Cong et al. (2019)); and (ii) the GO gradient has a similar expectation-based expression as the objective $\mathcal{L}(\boldsymbol{\gamma})$. Accordingly, we propose to view the GO gradient of $\mathcal{L}(\boldsymbol{\gamma})$ as another expectation-based vector objective, followed by calculating the GO gradient for that vector objective to form our GO Hessian of $\mathcal{L}(\boldsymbol{\gamma})$. Note this pattern of double applications of a GO gradient resulting in the GO Hessian is the same as that of the deterministic Hessian, which actually is a special case of our GO Hessian as proved below.

Assuming a single-layer continuous setup with $q_{\boldsymbol{\gamma}}(\boldsymbol{y}) = \prod_v q_{\boldsymbol{\gamma}}(y_v)$, the GO Hessian is defined as

$$\begin{aligned} &\nabla^2_{\boldsymbol{\gamma}} \mathcal{L}(\boldsymbol{\gamma}) = \nabla_{\boldsymbol{\gamma}} \mathbb{E}_{q_{\boldsymbol{\gamma}}(\boldsymbol{y})} \left[\mathbf{G}_{\boldsymbol{\gamma}}^{q_{\boldsymbol{\gamma}}(\boldsymbol{y})} \nabla_{\boldsymbol{y}} f(\boldsymbol{y}) \right] \\ &= \mathbb{E}_{q_{\boldsymbol{\gamma}}(\boldsymbol{y})} \left[\mathbf{G}_{\boldsymbol{\gamma}}^{q_{\boldsymbol{\gamma}}(\boldsymbol{y})} [\nabla^2_{\boldsymbol{y}} f(\boldsymbol{y})] \mathbf{G}_{\boldsymbol{\gamma}}^{q_{\boldsymbol{\gamma}}(\boldsymbol{y})T} + \mathbf{H}_{\boldsymbol{\gamma}\boldsymbol{\gamma}}^{q_{\boldsymbol{\gamma}}(\boldsymbol{y})} \nabla_{\boldsymbol{y}} f(\boldsymbol{y}) \right], \end{aligned} \quad (7)$$

where $\mathbf{H}_{\boldsymbol{\gamma}\boldsymbol{\gamma}}^{q_{\boldsymbol{\gamma}}(\boldsymbol{y})}$ is a three-dimensional tensor with its element

$$[\mathbf{H}_{\boldsymbol{\gamma}\boldsymbol{\gamma}}^{q_{\boldsymbol{\gamma}}(\boldsymbol{y})}]_{b,a,v} = g_{\gamma_b}^{q_{\boldsymbol{\gamma}}(y_v)} \nabla_{y_v} g_{\gamma_a}^{q_{\boldsymbol{\gamma}}(y_v)} + \nabla_{\gamma_b} g_{\gamma_a}^{q_{\boldsymbol{\gamma}}(y_v)} \triangleq h_{\gamma_b \gamma_a}^{q_{\boldsymbol{\gamma}}(y_v)} \quad (8)$$

and the tensor-vector product $\mathbf{H}\boldsymbol{a}$ outputs a matrix whose element $[\mathbf{H}\boldsymbol{a}]_{b,a} = \sum_v \mathbf{H}_{b,a,v} \boldsymbol{a}_v$. We name $h_{\gamma_b \gamma_a}^{q_{\boldsymbol{\gamma}}(y_v)}$ the *variable-hess*, because of its intuitive meaning of the second-order "derivative" of a RV y_v wrt parameters $\{\gamma_a, \gamma_b\}$ (see below). Note three components are necessary for employing the GO Hessian, *i.e.,* accessible *variable-nabla* $g_{\gamma_b}^{q_{\boldsymbol{\gamma}}(y_v)}$ and its two gradients $\nabla_{y_v} g_{\gamma_a}^{q_{\boldsymbol{\gamma}}(y_v)}$ and $\nabla_{\gamma_b} g_{\gamma_a}^{q_{\boldsymbol{\gamma}}(y_v)}$, for each node y_v.[7]

For better understanding, we draw parallel comparisons to deterministic optimization with objective $\hat{\mathcal{L}}(\boldsymbol{\gamma}) = f[\hat{\boldsymbol{y}}(\boldsymbol{\gamma})]$, which is a special case of Framework II with $q_{\boldsymbol{\gamma}}(\boldsymbol{y}) = \delta(\boldsymbol{y} - \hat{\boldsymbol{y}}(\boldsymbol{\gamma}))$ and where

$$\begin{aligned} &\nabla^2_{\boldsymbol{\gamma}} \hat{\mathcal{L}}(\boldsymbol{\gamma}) = \nabla_{\boldsymbol{\gamma}} \left[[\nabla_{\boldsymbol{\gamma}} \hat{\boldsymbol{y}}(\boldsymbol{\gamma})][\nabla_{\hat{\boldsymbol{y}}} f(\hat{\boldsymbol{y}})] \right] \\ &= [\nabla_{\boldsymbol{\gamma}} \hat{\boldsymbol{y}}(\boldsymbol{\gamma})][\nabla^2_{\hat{\boldsymbol{y}}} f(\hat{\boldsymbol{y}})][\nabla_{\boldsymbol{\gamma}} \hat{\boldsymbol{y}}(\boldsymbol{\gamma})]^T + [\nabla^2_{\boldsymbol{\gamma}} \hat{\boldsymbol{y}}(\boldsymbol{\gamma})] \nabla_{\hat{\boldsymbol{y}}} f(\hat{\boldsymbol{y}}). \end{aligned} \quad (9)$$

[7]This roughly means analytical PDF/CDF for y_v is accessible.

By comparing (7) and (9), interesting patterns arise: (i) the interpretation of $\mathbf{G}_{\boldsymbol{\gamma}}^{q_{\boldsymbol{\gamma}}(\boldsymbol{y})}$ as the "gradient" of RV $\boldsymbol{y}$ wrt parameters $\boldsymbol{\gamma}$ (informally $\mathbf{G}_{\boldsymbol{\gamma}}^{q_{\boldsymbol{\gamma}}(\boldsymbol{y})} \leftrightarrow \nabla_{\boldsymbol{\gamma}}\boldsymbol{y}$) also holds in second-order settings; (ii) the newly-introduced $\mathbf{H}_{\boldsymbol{\gamma}\boldsymbol{\gamma}}^{q_{\boldsymbol{\gamma}}(\boldsymbol{y})}$ can be intuitively interpreted as the "Hessian" of RV $\boldsymbol{y}$ wrt parameters $\boldsymbol{\gamma}$ (informally $\mathbf{H}_{\boldsymbol{\gamma}\boldsymbol{\gamma}}^{q_{\boldsymbol{\gamma}}(\boldsymbol{y})} \leftrightarrow \nabla_{\boldsymbol{\gamma}}^2\boldsymbol{y}$). In fact, the GO Hessian contains the deterministic Hessian as a special case, as detailed in Theorem 1 (see Appendix C.4 for the proof). Note in general the GO Hessian has an additional term due to the RV randomness (*i.e.,* the first item of the *variable-hess* in (8)).

On Discrete RVs. For a single-layer/leaf discrete RV $\boldsymbol{y}$ with PDF $q_{\boldsymbol{\gamma}}(\boldsymbol{y}) = \prod_v q_{\boldsymbol{\gamma}}(y_v)$, the GO Hessian is of the form

$$\begin{aligned}\nabla_{\boldsymbol{\gamma}}^2\mathcal{L}(\boldsymbol{\gamma}) &= \nabla_{\boldsymbol{\gamma}}\mathbb{E}_{q_{\boldsymbol{\gamma}}(\boldsymbol{y})}\big[\mathbf{G}_{\boldsymbol{\gamma}}^{q_{\boldsymbol{\gamma}}(\boldsymbol{y})}\mathbb{D}_{\boldsymbol{y}}f(\boldsymbol{y})\big] \\ &= \mathbb{E}_{q_{\boldsymbol{\gamma}}(\boldsymbol{y})}\big[\mathbf{G}_{\boldsymbol{\gamma}}^{q_{\boldsymbol{\gamma}}(\boldsymbol{y})}[\mathbb{D}_{\boldsymbol{y}}^2 f(\boldsymbol{y})]\mathbf{G}_{\boldsymbol{\gamma}}^{q_{\boldsymbol{\gamma}}(\boldsymbol{y})T} + \overline{\mathbf{H}_{\boldsymbol{\gamma}\boldsymbol{\gamma}}^{q_{\boldsymbol{\gamma}}(\boldsymbol{y})}\mathbb{D}_{\boldsymbol{y}}f(\boldsymbol{y})}\big],\end{aligned} \tag{10}$$

where $\overline{\mathbf{H}_{\boldsymbol{\gamma}\boldsymbol{\gamma}}^{q_{\boldsymbol{\gamma}}(\boldsymbol{y})}\mathbb{D}_{\boldsymbol{y}}f(\boldsymbol{y})}$ represents a matrix with its elements

$$\Big[\overline{\mathbf{H}_{\boldsymbol{\gamma}\boldsymbol{\gamma}}^{q_{\boldsymbol{\gamma}}(\boldsymbol{y})}\mathbb{D}_{\boldsymbol{y}}f(\boldsymbol{y})}\Big]_{b,a} = \sum_v \begin{bmatrix}[g_{\gamma_b}^{q_{\boldsymbol{\gamma}}(y_v)}\mathbb{D}_{y_v}g_{\gamma_a}^{q_{\boldsymbol{\gamma}}(y_v)}]\mathbb{D}_{y_v}f(\boldsymbol{y}^{v+}) \\ +\,[\nabla_{\gamma_b}g_{\gamma_a}^{q_{\boldsymbol{\gamma}}(y_v)}]\mathbb{D}_{y_v}f(\boldsymbol{y})\end{bmatrix}. \tag{11}$$

It is clear that (7) for continuous RVs and (10) for discrete RVs show similar patterns but with slight differences, like the gradient/difference of $f(\boldsymbol{y})$ and the definition of the *variable-hess*. We leave discrete situations as future research and focus mainly on continuous non-rep cases in this paper.

On Deep SCGs. Based on the above derivations for the single-layer continuous/discrete setup, we prove in Appendix C.3 that similar patterns also hold for deep SCGs with *continuous* internal nodes and *continuous/discrete* leaves. The main results are summarized in Theorem 1, with the corresponding complexity analysis given in Appendix D.

Theorem 1 (**GO Hessian**). *For an expectation-based objective $\mathbb{E}_{q_{\boldsymbol{\gamma}}(\boldsymbol{y})}[f(\boldsymbol{y})]$, where $q_{\boldsymbol{\gamma}}(\boldsymbol{y}) = \prod_i q_{\boldsymbol{\gamma}_i}(\boldsymbol{y}_i|pa(\boldsymbol{y}_i))$ models a SCG with continuous internal nodes and continuous/discrete* leaves *(possibly with neural networks as links), $\boldsymbol{y} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots\}$, $\boldsymbol{\gamma} = \{\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \cdots\}$, and each component satisfies conditional independence,* i.e.,

$$q_{\boldsymbol{\gamma}_i}(\boldsymbol{y}_i|pa(\boldsymbol{y}_i)) = \prod\nolimits_v q_{\boldsymbol{\gamma}_i}(y_{i,v}|pa(\boldsymbol{y}_i)) \tag{12}$$

with $y_{i,v}$ being the v-th element of $\boldsymbol{y}_i$, $pa(\boldsymbol{y}_i)$ the parent RVs of node $\boldsymbol{y}_i$, and $q_{\boldsymbol{\gamma}_i}(y_{i,v}|pa(\boldsymbol{y}_i))$ having accessible variable-nabla/variable-hess, double application of the GO gradient (Cong et al. 2019) to that objective delivers an unbiased low-variance Hessian estimator, i.e., *the GO Hessian.*

The GO Hessian in expectation *obeys the chain rule, exhibiting an expression similar to the deterministic Hessian (*e.g., *see* (7) *versus* (9)*), except with the variable-nabla/variable-hess serving as the first-order/second-order "derivative" of a RV. Moreover, when limiting each component $q_{\boldsymbol{\gamma}_i}(\boldsymbol{y}_i|pa(\boldsymbol{y}_i))$ as a Dirac delta function, the GO Hessian degrades into the deterministic Hessian.*

The similarity between the GO Hessian and its special case of the deterministic Hessian is exploited in Section 3.2 to reveal the easy-to-use property of the GO Hessian during practical implementations.

On the Applicability of the GO Hessian. The conditional independence assumption in (12) may be falsely considered limited at first glance. It's worth noting that the GO Hessian also applies to correlated multivariant $q_{\boldsymbol{\gamma}}(\boldsymbol{y})$ if (i) the factorization $q_{\boldsymbol{\gamma}}(\boldsymbol{y}) = q_{\boldsymbol{\gamma}}(y_1)q_{\boldsymbol{\gamma}}(y_2|y_1)\cdots$ is accessible, as $q_{\boldsymbol{\gamma}}(\boldsymbol{y})$ is now a deep SCG; (ii) $q_{\boldsymbol{\gamma}}(\boldsymbol{y})$ can be reparametrized to some extent (such as a MVN or a Dirichlet). Multivariate RVs beyond these two cases are rarely used in practice. From an alternative perspective, our setup generalizes deterministic deep learning with RVs, by leveraging sampling to replace/reinforce traditional nonlinear activation functions. Accordingly, the GO Hessian is deemed widely applicable for most statistical deep learning problems.

On Limited $f(\cdot)$-information. For challenging situations where only zero-order $f(\boldsymbol{y})$-information is available at the current sample $\boldsymbol{y}$ (like in many RL applications (Baxter and Bartlett 2001; Shen et al. 2019)), we reveal that our GO Hessian can be combined with the LAX technique (Grathwohl et al. 2017) to facilitate that issue. Specifically, with a surrogate function $c_{\boldsymbol{\omega}}(\boldsymbol{y})$ (often a neural network) to approximate $f(\boldsymbol{y})$, we unify the zero-order f-evaluation from the log-trick estimation and the low-variance of our GO Hessian via

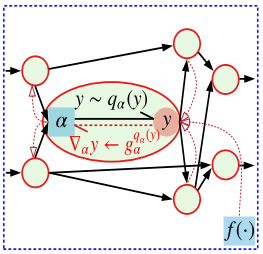
$$\mathbf{H}_{\text{LAX}}[f] = \mathbf{H}_{\text{log-trick}}[f] - \mathbf{H}_{\text{log-trick}}[c_{\boldsymbol{\omega}}] + \mathbf{H}_{\text{GO}}[c_{\boldsymbol{\omega}}],$$

where $\mathbf{H}_{\texttt{method}}[\texttt{func}]$ denotes the Hessian estimator of objective $\mathbb{E}_{q_{\boldsymbol{\gamma}}(\boldsymbol{y})}[\texttt{func}(\boldsymbol{y})]$ based on the `method`. The surrogate parameters $\boldsymbol{\omega}$ can be optimized by minimizing the MC variance of $\mathbf{H}_{\text{LAX}}[f]$ (Grathwohl et al. 2017). Note when $c_{\boldsymbol{\omega}}(\boldsymbol{y}) = f(\boldsymbol{y})$, $\mathbf{H}_{\text{LAX}}[f]$ delivers the same low variance as our GO Hessian.

3.2 GO Hessian Is Easy-to-use

In practice, to explicitly construct/store the GO Hessian may be prohibitively expensive, especially for SCGs with neural-network-based links. Fortunately, we find the GO Hessian is easy to use in practice with AD and HVP. The key observations include (i) the Framework II objective, its GO gradient, and its GO Hessian are all expectations over the same $q_{\boldsymbol{\gamma}}(\boldsymbol{y})$; and (ii) when estimated with one shared sample, they all act the same as their counterpart/special-case in deterministic optimization, except with the *variable-nabla/variable-hess* serving as the first-order/second-order "derivative" at each RV node. Therefore, based on one shared sample, one can simply manipulate well-developed AD software (like PyTorch (Paszke et al. 2017) or TensorFlow (Abadi et al. 2015)) to guarantee correct *variable-nabla/variable-hess* for each node, which in turn delivers an easy-to-use exploitation of the one-sample-estimated GO Hessian via existing AD and HVP techniques. Multi-sample-based estimation can be implemented with parallel computations.

Figure 2(a) shows a demonstrative example, where, thanks to conditional independence, we zoom in on a scalar RV node y of the SCG for illustration, $\boldsymbol{\alpha}$ denotes the distribution parameters of that node (*e.g.,* the shape and rate of a gamma RV), and one sample is *stochastically activated* for subsequent forward pass. To exploit the one-sample-estimated GO Hessian over the SCG with AD/HVP, one may employ the general approach in Figure 2(b) to guarantee correct *variable-nabla/variable-hess* for each node, which then delivers seam-



(a) Forward/backward pass

```
class GOActivation(Function):
  def forward(ctx, α):
    y ~ q_α(y)
    ctx.save_for_backward(α, y)
    return y
  def backward(ctx, grad_y):
    α, y = ctx.saved_tensors
    grad_α = grad_y × [ 𝕊[g_α^{q_α(y)}]
      +y𝕊[∇_y g_α^{q_α(y)}]−𝕊[y∇_y g_α^{q_α(y)}]
      +α𝕊[∇_α g_α^{q_α(y)}]−𝕊[α∇_α g_α^{q_α(y)}]]
    return grad_α
```

(b) PyTorch-like pseudo code

Figure 2: Illustration of the simplicity in jointly implementing the GO gradient and Hessian. A red circle is a RV node. Black solid arrows indicate both the forward pass and parameters $\boldsymbol{\gamma}$ of the SCG. Red dashed arrows show the backward pass through the current node. $\mathbb{S}[\cdot]$ is the stop-gradient operator.

less (double) back-propagation through that SCG (as the rest computations are deterministic and well-defined in existing AD software). Note simpler approaches than the one in Figure 2(b) potentially exist, *e.g.,* one associated with the method in Pearlmutter (1994) or one designed for a specific RV.

3.3 Second-order Optimization of Framework II

Benefiting from the low-variance and easy-to-use properties of the GO Hessian, one can readily combine it with existing second-order methods for Framework I to develop novel variants for Framework II in (2). Considering practical applications like variational inference, we employ a more common objective here for presentation, *i.e.,* $\min_{\boldsymbol{\gamma}} \mathcal{L}(\boldsymbol{\gamma}) \triangleq \mathbb{E}_{q(\boldsymbol{x})q_{\boldsymbol{\gamma}}(\boldsymbol{y}|\boldsymbol{x})}[f(\boldsymbol{x},\boldsymbol{y})]$, where $q(\boldsymbol{x})$ is the data distribution.

We employ the stochastic cubic regularization (SCR) (Tripuraneni et al. 2018)[8] that exploits stochastic gradient/Hessian information within its subroutine (see (5)). By leveraging the GO gradient and our GO Hessian in place of the classic gradient/Hessian, we present in Algorithm 1 a new second-order method for Framework II, termed SCR-GO. The Cubic-Subsolver and Cubic-Finalsolver (given in Appendix F) minimize the local third-order Taylor approximation of Framework II, mimicking (5). The detailed convergence analysis is provided in Appendix G, where a gamma-related special case is discussed.

4 GO Hessian for Common RVs

Based on the *variable-nabla*s summarized in Table 3 of Cong et al. (2019) and the definitions in (8) and (11), one can derive the *variable-hess* (or its components) for many kinds of RVs,[9] which are essential for easy-to-use curvature exploitation over SCGs that are flexibly constructed by those RVs. Although the derivations for most RVs are straight-forward, we highlight two challenging but interesting RVs for demonstration following Cong et al. (2019), *i.e.,* continuous non-rep gamma and discrete negative binomial RVs.

[8] One may consider the SRVRC$_{\text{free}}$ from Zhou and Gu (2020).

[9] Note for each RV type, one need only derive the variable-hess/components for once, whose expressions are then reusable.

Algorithm 1 SCR-GO for $\min_{\boldsymbol{\gamma}} \mathbb{E}_{q(\boldsymbol{x})q_{\boldsymbol{\gamma}}(\boldsymbol{y}|\boldsymbol{x})}[f(\boldsymbol{x},\boldsymbol{y})]$

Input: Batch sizes N_g, N_H, initialization $\boldsymbol{\gamma}_0$, total iterations T, and final tolerance ϵ.
Output: ϵ-second-order stationary point $\boldsymbol{\gamma}^*$ or $\boldsymbol{\gamma}_{T+1}$.
for $t = 0, 1, \cdots, T$ **do**
  Sample $\{\boldsymbol{x}_i\}_{i=1}^{N_g}$ and $\{\boldsymbol{x}'_j\}_{j=1}^{N_H}$ from $q(\boldsymbol{x})$
  Sample $\boldsymbol{y}_i \sim q_{\boldsymbol{\gamma}_t}(\boldsymbol{y}|\boldsymbol{x}_i)$ and $\boldsymbol{y}'_j \sim q_{\boldsymbol{\gamma}_t}(\boldsymbol{y}|\boldsymbol{x}'_j)$
  Estimate GO gradient $\tilde{\boldsymbol{g}}_t$ with $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{N_g}$
  Estimate GO Hessian $\tilde{\mathbf{H}}_t[\cdot]$ with $\{\boldsymbol{x}'_j, \boldsymbol{y}'_j\}_{j=1}^{N_H}$
  $\boldsymbol{\Delta}, \Delta \leftarrow$ Cubic-Subsolver$(\tilde{\boldsymbol{g}}_t, \tilde{\mathbf{H}}_t[\cdot], \epsilon)$
  $\boldsymbol{\gamma}_{t+1} = \boldsymbol{\gamma}_t + \boldsymbol{\Delta}$
  if $\Delta > -\sqrt{\epsilon^3/\rho}/100$ **then**
    $\boldsymbol{\Delta} \leftarrow$ Cubic-Finalsolver$(\tilde{\boldsymbol{g}}_t, \tilde{\mathbf{H}}_t[\cdot], \epsilon)$
    $\boldsymbol{\gamma}^* = \boldsymbol{\gamma}_t + \boldsymbol{\Delta}$ and **break**

4.1 GO Hessian for Non-rep Gamma RVs

To demonstrate the effectiveness of the proposed techniques, we focus on situations with non-rep gamma RVs in our experiments. Such a concentration is motivated by their broad practical applications (Boland 2007; Mendoza-Parra et al. 2013; Al-Ahmadi 2014; Wright et al. 2014; Belikov 2017) and by their fundamental utility in statistics and machine learning. For example, many popular distributions can be reparameterized as gamma (Leemis and McQueston 2008), such as exponential, chi-squared, inverse-gamma, log-gamma, beta, and Dirichlet; other ones can be mixed via gamma (Zhou et al. 2012; Zhou and Carin 2015), like gamma-normal-mixed student-t and gamma-Poisson-mixed negative binomial. Accordingly, the presented techniques for gamma can be readily extended to those gamma-related cases of Framework II.

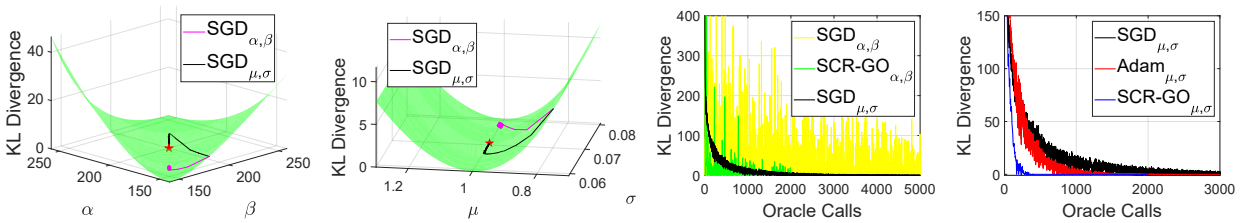
As discussed following (8) and shown in Figure 2(b), three components are crucial in constructing a GO Hessian for a continuous RV y with PDF $q_{\boldsymbol{\alpha}}(y)$, that is,

$$g_{\boldsymbol{\alpha}}^{q_{\boldsymbol{\alpha}}(y)},\ \nabla_y g_{\boldsymbol{\alpha}}^{q_{\boldsymbol{\alpha}}(y)},\ \text{and}\ \nabla_{\boldsymbol{\alpha}} g_{\boldsymbol{\alpha}}^{q_{\boldsymbol{\alpha}}(y)}. \tag{13}$$

For a gamma RV $\hat{y} \sim \mathrm{Gam}(\alpha, \beta)$, the distribution parameters $\boldsymbol{\alpha}$ in general contain both the shape α and the rate β. However, we notice the reparameterization of $\hat{y} = y/\beta, y \sim \mathrm{Gam}(\alpha, 1)$, with which one can leave the derivatives wrt β to AD for simplicity and focus solely on the non-rep part associated with α. Accordingly, we only need to calculate the three components in (13) for $q_\alpha(y) = \mathrm{Gam}(y; \alpha, 1)$. Moving detailed derivations to Appendix H for clarity, we yield

$$\begin{aligned}
g_\alpha^{q_\alpha(y)} =& -[\log y - \psi(\alpha)]\gamma(\alpha, y)y^{1-\alpha}e^y \\
&+ ye^y\alpha^{-2}{}_2F_2(\alpha,\alpha;\alpha+1,\alpha+1;-y) \\
\nabla_y g_\alpha^{q_\alpha(y)} =& [\psi(\alpha) - \log y] + g_\alpha^{q_\alpha(y)}(y - \alpha + 1)/y \\
\nabla_\alpha g_\alpha^{q_\alpha(y)} =& \psi^{(1)}(\alpha)\gamma(\alpha, y)y^{1-\alpha}e^y \\
&+[\log y - \psi(\alpha)]ye^y\alpha^{-2}{}_2F_2(\alpha,\alpha;\alpha+1,\alpha+1;-y) \\
&-2ye^y\alpha^{-3}{}_3F_3(\alpha,\alpha,\alpha;\alpha+1,\alpha+1,\alpha+1;-y)
\end{aligned} \tag{14}$$

where $\psi(\alpha)$ is the digamma function, $\psi^{(m)}(x)$ the polygamma function of order m, $\gamma(\alpha, y)$ the lower incomplete gamma function, and ${}_pF_q(a_1, \cdots, a_p; b_1, \cdots, b_q; x)$ is



(a) α-β space (b) μ-σ space (c) (d)

Figure 3: (a-b) Demonstration of reverse KL divergences and SGD trajectories in α-β and μ-σ parameter spaces for Section 4.1. The subscript indicates the parameter space. SGD$_{\alpha,\beta}$ leverages a learning rate of 10, while the more efficient SGD$_{\mu,\sigma}$ only uses 10^{-3}. The red star denotes the optimum. 2,000 iterations are used to generate both trajectories, which are adapted to the other side for clear comparisons. (c-d) Training objectives versus the number of oracle calls for Section 5.1. The same curve of SGD$_{\mu,\sigma}$ is shown in both plots for clear comparisons.

the generalized hypergeometric function. Reparameterizing the rate β first, followed by substituting the components in (14) into the approach in Figure 2(b), one enables easy-to-use exploitation of GO Hessian with AD over a non-rep gamma node. Figure 1 illustrates the low variance of our GO Hessian.

A Gradient-friendly Reparameterization. To model a gamma node within a SCG, a naive method would parameterize shape $\alpha = \text{softplus}(\gamma_\alpha)$ and rate $\beta = \text{softplus}(\gamma_\beta)$, where without loss of generality $\boldsymbol{\gamma} = \{\gamma_\alpha, \gamma_\beta\}$ is considered as the parameters of interest. However, we find empirically that such a naive modeling may not be friendly to gradient-based methods, especially when target shape and/or rate are large. Figure 3(a) shows an example with the reverse KL objective $\text{KL}[\text{Gam}(y;\alpha,\beta)||\text{Gam}(y;200,200)]$; with that modeling, SGD (labeled as SGD$_{\alpha,\beta}$) bounces between two slopes at the bottom of the "valley" and advances slowly. Alternatively, noticing that the valley bottom is approximately located in a line where $\text{Gam}(y;\alpha,\beta)$ shares the same mean as the target, we propose to reparameterize via mean μ and standard deviation σ, *i.e.,* $q_{\boldsymbol{\gamma}}(y) = \text{Gam}(y; \frac{\mu^2}{\sigma^2}, \frac{\mu}{\sigma^2})$ with $\mu = \text{softplus}(\gamma_\mu)$ and $\sigma = \text{softplus}(\gamma_\sigma)$. With this reparameterization, we obtain an approximately decorrelated objective surface (see Figure 3(b)) that is more friendly to gradient-based methods; it's apparent SGD in the μ-σ space, termed SGD$_{\mu,\sigma}$, converges to the optimum much faster.

4.2 GO Hessian for Discrete NB RVs

For a NB RV $y \sim q_{\boldsymbol{\alpha}}(y) = \text{NB}(r,p)$, the distribution parameters $\boldsymbol{\alpha} = \{r,p\}$ contain both the number of failures r and the success probability p. From the definition in (11), three components are necessary to calculate the GO Hessian, *i.e.,* $g_{\boldsymbol{\alpha}}^{q_{\boldsymbol{\alpha}}(y)}$, $\mathbb{D}_y g_{\boldsymbol{\alpha}}^{q_{\boldsymbol{\alpha}}(y)}$, and $\nabla_{\boldsymbol{\alpha}} g_{\boldsymbol{\alpha}}^{q_{\boldsymbol{\alpha}}(y)}$. Due to space constraints, analytic expressions and detailed derivations are given in Appendix I. The low variance of the GO Hessian is demonstrated in Figure 1.

5 Experiments

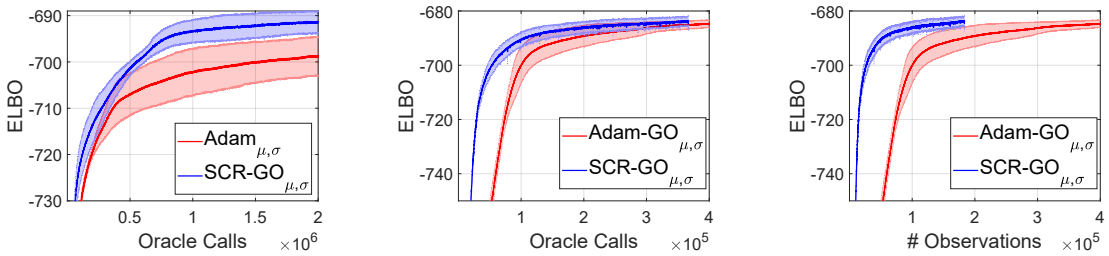
The proposed techniques are verified with challenging experiments where non-rep gamma RVs are of interest. Generalizing Section 4.1, we first test our SCR-GO on minimizing the reverse KL divergence between two gamma RVs. Next, we consider mean-field variational inference for Poisson factor analysis (PFA; which is closely related to the LDA (Blei, Ng, and Jordan 2003)) (Zhou and Carin 2015; Zhou, Cong, and Chen 2015). Finally concerning deep neural networks, the SCR-GO is tested on training variational encoders, mimicking the VAE (Kingma and Welling 2014), for PFA and its deep generalization of the Poisson gamma belief network (PGBN) (Zhou, Cong, and Chen 2016; Cong et al. 2017).

Experimental Settings. We follow (Xu, Roosta-Khorasani, and Mahoney 2017; Tripuraneni et al. 2018; Yu, Xu, and Gu 2018; Roosta et al. 2018; Kasai and Mishra 2018) to show training objectives versus the number of oracle calls (calculations of gradients and/or HVPs); this is deemed a fair metric because it's independent of implementation-details/system-configurations and ideally an HVP can "take about the same amount of computation as a gradient" (Pearlmutter 1994).[10] We compare SCR-GO to standard SGD and the popular Adam (Kingma and Ba 2014) that leverage the GO gradient. For both SGD and Adam, learning rates from $\{0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1\}$ are tested with the best-tuned results shown. Other settings are given in Appendix J. Code will be available at github.com/YulaiCong/GOHessian.

5.1 Minimizing the Reverse KL Divergence Between Gamma RVs

To demonstrate the effectiveness of the μ-σ reparameterization introduced in Section 4.1 and the efficiency achieved from exploiting the curvature information via the GO Hessian, we first consider a simplified example, with the objective $\text{KL}[\text{Gam}(y;\alpha,\beta)||\text{Gam}(y;200,1)]$, for better introduction. SGD, Adam, and our SCR-GO are compared within both α-β and μ-σ parameter spaces.

[10] This is the ideal/theoretical situation for evaluation. However, it may not hold for our current implementation, which uses the less efficient 2-backward technique in (4) (due to the missing forward-mode AD) and calculates special functions with a surrogate lib (see Appendix H). The implementation also makes it impossible for fair comparisons wrt wall-clock time. With our implementation/computer, a GO-HVP in gamma-related experiments is about 3 times more expensive than a GO gradient.



(a) (b) (c)

Figure 4: Training curves of mean-field variational inference (a) and variational encoder (b-c) for PFA on MNIST. Variances are estimated based on 5 random seeds. See Appendix J.2 and J.3 for more results.

The training curves of the compared methods are given in Figures 3(c)-3(d). By comparing $\text{SGD}_{\alpha,\beta}$ with $\text{SGD}_{\mu,\sigma}$ in Figure 3(c), it's clear that the μ-σ reparameterization method leads to a much faster convergence with smoother training curves, similar to those from deterministic optimization. By contrast, $\text{SGD}_{\alpha,\beta}$ visits both high and low KL values frequently (bouncing within a valley bottom as shown in Figure 3(a)), with a much slower convergence to the optimum. Thanks to the exploited curvature information, our $\text{SCR-GO}_{\alpha,\beta}$ shows a clearly improved convergence relative to $\text{SGD}_{\alpha,\beta}$. When moving to the μ-σ space (see Figure 3(d)), our $\text{SCR-GO}_{\mu,\sigma}$ delivers an even faster and more stabilized convergence than its counterpart $\text{SCR-GO}_{\alpha,\beta}$ and also $\text{SGD}_{\mu,\sigma}$ and $\text{Adam}_{\mu,\sigma}$, demonstrating the effectiveness of both the μ-σ reparameterization and the curvature exploitation via the GO Hessian.

5.2 Mean-field Variational Inference for PFA

For practical applications, we leverage the proposed techniques to develop efficient mean-field variational inference for the PFA, whose generative process is

$$p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{z}) : \boldsymbol{x} \sim \text{Pois}(\boldsymbol{x}|\mathbf{W}\boldsymbol{z}), \boldsymbol{z} \sim \text{Gam}(\boldsymbol{z}|\boldsymbol{\alpha}_0,\boldsymbol{\beta}_0),$$

where $\boldsymbol{x}$ is the count data variable, $\mathbf{W}$ the topic matrix with each column/topic $\boldsymbol{w}_k$ located in the simplex, *i.e.,* $w_{vk} > 0, \sum_v w_{vk} = 1$, $\boldsymbol{z}$ the latent code, and $\boldsymbol{\theta} = \{\mathbf{W}, \boldsymbol{\alpha}_0, \boldsymbol{\beta}_0\}$. For mean-field variational inference, we assume variational approximation distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}) = \text{Gam}(\boldsymbol{z}; \frac{\boldsymbol{\mu}^2}{\boldsymbol{\sigma}^2}, \frac{\boldsymbol{\mu}}{\boldsymbol{\sigma}^2})$ with $\boldsymbol{\phi} = \{\boldsymbol{\mu}, \boldsymbol{\sigma}\}$. Accordingly given training dataset $\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$, the objective is to maximize

$$\text{ELBO}(\boldsymbol{\theta}, \{\boldsymbol{\phi}_i\}) = \frac{1}{N}\sum\nolimits_{i=1}^{N} \mathbb{E}_{q_{\boldsymbol{\phi}_i}(\boldsymbol{z}_i)}\Big[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_i,\boldsymbol{z}_i)}{q_{\boldsymbol{\phi}_i}(\boldsymbol{z}_i)}\Big].$$

The training curves from a well-tuned Adam optimizer and our SCR-GO are shown in Figure 4(a) (see Appendix J.2 for more results). It's clear that with the additional curvature information exploited via GO-HVP, our SCR-GO exhibits a faster convergence to a better local optimum, with a lower variance than the well-tuned Adam optimizer.

5.3 Variational Encoders for PFA and PGBN

To test the effectiveness of the presented techniques when combined with deep neural networks, we consider developing a variational encoder for PFA mimicking the VAE, that is,

$$q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \text{Gam}(\boldsymbol{z}; \tfrac{\boldsymbol{\mu}^2}{\boldsymbol{\sigma}^2}, \tfrac{\boldsymbol{\mu}}{\boldsymbol{\sigma}^2}), \boldsymbol{\mu} = \text{NN}_{\boldsymbol{\mu}}(\boldsymbol{x}), \boldsymbol{\sigma} = \text{NN}_{\boldsymbol{\sigma}}(\boldsymbol{x}), \quad (15)$$

where $\text{NN}(\cdot)$ denotes a neural network and $\boldsymbol{\phi}$ contains all the parameters of $\text{NN}_{\boldsymbol{\mu}}(\cdot)$ and $\text{NN}_{\boldsymbol{\sigma}}(\cdot)$. The objective is to maximize $\text{ELBO}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{z}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\big]$.

Figures 4(b)-4(c) show the training objectives versus the number of oracle calls and processed observations. It's clear that the proposed SCR-GO performs better than a well-tuned Adam optimizer in terms of oracle calls and data efficiency, when applied to a model with deep neural networks. The better performance of SCR-GO is attributed to its exploitation of the curvature information via the GO Hessian, which takes into consideration the correlation among parameters within $p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{z})$ and $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ and utilizes an (implicit) adaptive learning rate mimicking the classical Newton's method.

For further testing under more challenging settings with a hierarchically-structured $q_{\boldsymbol{\gamma}}(\boldsymbol{y})$ for Framework II in (2), we consider developing a variational encoder for a 2-layer PGBN. Specifically, with $\boldsymbol{z} = \{\boldsymbol{z}_1, \boldsymbol{z}_2\}$,

$$p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{z}) : \begin{cases} \boldsymbol{x} \sim \text{Pois}(\boldsymbol{x}|\mathbf{W}_1\boldsymbol{z}_1), \boldsymbol{z}_1 \sim \text{Gam}(\boldsymbol{z}_1|\mathbf{W}_2\boldsymbol{z}_2, \boldsymbol{c}_2), \\ \boldsymbol{z}_2 \sim \text{Gam}(\boldsymbol{z}_2|\boldsymbol{\alpha}_0, \boldsymbol{\beta}_0) \end{cases}$$

$$q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) : \boldsymbol{z}_2 \sim q_{\boldsymbol{\phi}_2}(\boldsymbol{z}_2|\boldsymbol{z}_1), \boldsymbol{z}_1 \sim q_{\boldsymbol{\phi}_1}(\boldsymbol{z}_1|\boldsymbol{x}),$$

where $\boldsymbol{\theta} = \{\mathbf{W}_1, \mathbf{W}_2, \boldsymbol{c}_2, \boldsymbol{\alpha}_0, \boldsymbol{\beta}_0\}$, $\boldsymbol{\phi} = \{\boldsymbol{\phi}_1, \boldsymbol{\phi}_2\}$, and both $q_{\boldsymbol{\phi}_2}(\boldsymbol{z}_2|\boldsymbol{z}_1)$ and $q_{\boldsymbol{\phi}_1}(\boldsymbol{z}_1|\boldsymbol{x})$ are constructed as in (15). Due to space constraints, the experimental details and results are moved to Appendix J.3, where one observes similar plots as those in Figure 4, confirming the effectiveness and efficiency of the presented techniques.

6 Conclusions

An unbiased low-variance Hessian estimator, termed GO Hessian, is proposed to efficiently exploit curvature information for an expectation-based objective over a SCG, with continuous rep/non-rep internal nodes and continuous/discrete leaves. Containing the deterministic Hessian as a special case, the GO Hessian is easy-to-use with AD and HVP, enabling a low cost second-order optimization over deep SCGs. Based on the proposed GO Hessian, a new second-order optimization method is proposed for the expectation-based objective, which empirically performs better than a well-tuned Adam optimizer in challenging situations with non-rep gamma RVs.

Acknowledgments

We thank the anonymous reviewers for their constructive comments that helped improve the paper. The research was supported in part by DARPA, DOE, NIH, NSF and ONR. The Titan Xp GPU used was donated by the NVIDIA Corporation.

References

Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Goodfellow, I.; Harp, A.; Irving, G.; Isard, M.; Jia, Y.; Jozefowicz, R.; Kaiser, L.; Kudlur, M.; Levenberg, J.; Mané, D.; Monga, R.; Moore, S.; Murray, D.; Olah, C.; Schuster, M.; Shlens, J.; Steiner, B.; Sutskever, I.; Talwar, K.; Tucker, P.; Vanhoucke, V.; Vasudevan, V.; Viégas, F.; Vinyals, O.; Warden, P.; Wattenberg, M.; Wicke, M.; Yu, Y.; and Zheng, X. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. URL https://www.tensorflow.org/. Software available from tensorflow.org.

Agarwal, N.; Allen-Zhu, Z.; Bullins, B.; Hazan, E.; and Ma, T. 2017. Finding approximate local minima faster than gradient descent. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, 1195–1199. ACM.

Al-Ahmadi, S. 2014. The gamma-gamma signal fading model: A survey [wireless corner]. *IEEE Antennas and Propagation Magazine* 56(5): 245–260.

Allen-Zhu, Z. 2018. Natasha 2: Faster non-convex optimization than SGD. In *NeurIPS*, 2675–2686.

Baxter, J.; and Bartlett, P. 2001. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research* 15: 319–350.

Belikov, A. 2017. The number of key carcinogenic events can be predicted from cancer incidence. *Scientific reports* 7(1): 1–8.

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

Blei, D. M.; Ng, A. Y.; and Jordan, M. I. 2003. Latent Dirichlet allocation. *JMLR* 3: 993–1022.

Boland, P. 2007. *Statistical and probabilistic methods in actuarial science*. Chapman and Hall/CRC.

Byrd, R.; Chin, G.; Neveitt, W.; and Nocedal, J. 2011. On the use of stochastic Hessian information in optimization methods for machine learning. *SIAM Journal on Optimization* 21(3): 977–995.

Carmon, Y.; and Duchi, J. 2016. Gradient descent efficiently finds the cubic-regularized non-convex newton step. *arXiv preprint arXiv:1612.00547* .

Cong, Y.; Chen, B.; Liu, H.; and Zhou, M. 2017. Deep latent Dirichlet allocation with topic-layer-adaptive stochastic gradient Riemannian MCMC. In *ICML*.

Cong, Y.; Zhao, M.; Bai, K.; and Carin, L. 2019. GO Gradient for Expectation-Based Objectives. In *ICLR*. URL https://openreview.net/forum?id=ryf6Fs09YX.

Farquhar, G.; Whiteson, S.; and Foerster, J. 2019. Loaded DiCE: Trading off Bias and Variance in Any-Order Score Function Gradient Estimators for Reinforcement Learning. In *NeurIPS*, 8149–8160.

Figurnov, M.; Mohamed, S.; and Mnih, A. 2018. Implicit Reparameterization Gradients. In *NeurIPS*.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 1126–1135. JMLR. org.

Foerster, J.; Farquhar, G.; Al-Shedivat, M.; Rocktäschel, T.; Xing, E.; and Whiteson, S. 2018. DiCE: The infinitely differentiable monte-carlo estimator. In *ICML*, 10204–10214.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *NIPS*, 2672–2680.

Grathwohl, W.; Choi, D.; Wu, Y.; Roeder, G.; and Duvenaud, D. 2017. Backpropagation through the Void: Optimizing control variates for black-box gradient estimation. *arXiv:1711.00123* .

Heess, N.; Wayne, G.; Silver, D.; Lillicrap, T.; Erez, T.; and Tassa, Y. 2015. Learning continuous control policies by stochastic value gradients. In *NeurIPS*, 2944–2952.

Jankowiak, M.; and Obermeyer, F. 2018. Pathwise Derivatives Beyond the Reparameterization Trick. In *ICML*.

Jin, C.; Netrapalli, P.; Ge, R.; Kakade, S.; and Jordan, M. 2019. Stochastic Gradient Descent Escapes Saddle Points Efficiently. *arXiv preprint arXiv:1902.04811* .

Kasai, H.; and Mishra, B. 2018. Inexact trust-region algorithms on Riemannian manifolds. In *NeurIPS*, 4249–4260.

Kingma, D.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Kingma, D. P.; and Welling, M. 2014. Auto-encoding variational Bayes. In *ICLR*.

Kohler, J.; and Lucchi, A. 2017. Sub-sampled cubic regularization for non-convex optimization. In *ICML*, 1895–1904. JMLR. org.

Leemis, L.; and McQueston, J. 2008. Univariate distribution relationships. *The American Statistician* 62(1): 45–53.

Liu, H.; Socher, R.; and Xiong, C. 2019. Taming MAML: Efficient unbiased meta-reinforcement learning. In *ICML*, 4061–4071.

Mao, J.; Foerster, J.; Rocktaschel, T.; Al-Shedivat, M.; Farquhar, G.; and Whiteson, S. 2019. A baseline for any order gradient estimation in stochastic computation graphs .

Martens, J. 2010. Deep learning via Hessian-free optimization. In *ICML*, volume 27, 735–742.

Mendoza-Parra, M.; Nowicka, M.; Van Gool, W.; and Gronemeyer, H. 2013. Characterising ChIP-seq binding patterns by model-based peak shape deconvolution. *BMC genomics* 14(1): 834.

Parmas, P. 2018. Total stochastic gradient algorithms and applications in reinforcement learning. In *NeurIPS*, 10204–10214.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in PyTorch .

Pearlmutter, B. 1994. Fast exact multiplication by the Hessian. *Neural computation* 6(1): 147–160.

Rezende, D. J.; Mohamed, S.; and Wierstra, D. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*.

Robbins, H.; and Monro, S. 1951. A stochastic approximation method. *The annals of mathematical statistics* 400–407.

Roosta, F.; Liu, Y.; Xu, P.; and Mahoney, M. 2018. Newton-MR: Newton's Method Without Smoothness or Convexity. *arXiv preprint arXiv:1810.00303* .

Rothfuss, J.; Lee, D.; Clavera, I.; Asfour, T.; and Abbeel, P. 2019. ProMP: Proximal meta-policy search. In *ICLR*.

Ruiz, F. J. R.; Titsias, M. K.; and Blei, D. 2016. The generalized reparameterization gradient. In *NIPS*, 460–468.

Rumelhart, D.; and Hinton, G. 1986. Learning representations by back-propagating errors. *Nature* 323(9).

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

Schulman, J.; Heess, N.; Weber, T.; and Abbeel, P. 2015a. Gradient estimation using stochastic computation graphs. In *NIPS*, 3528–3536.

Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; and Abbeel, P. 2015b. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438* .

Shen, Z.; Ribeiro, A.; Hassani, H.; Qian, H.; and Mi, C. 2019. Hessian aided policy gradient. In *ICML*, 5729–5738.

Tripuraneni, N.; Stern, M.; Jin, C.; Regier, J.; and Jordan, M. 2018. Stochastic cubic regularization for fast nonconvex optimization. In *NeuIPS*, 2899–2908.

Weber, T.; Heess, N.; Buesing, L.; and Silver, D. 2019. Credit assignment techniques in stochastic computation graphs. *arXiv preprint arXiv:1901.01761* .

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

Wright, M.; Winter, I.; Forster, J.; and Bleeck, S. 2014. Response to best-frequency tone bursts in the ventral cochlear nucleus is governed by ordered inter-spike interval statistics. *Hearing research* 317: 23–32.

Xu, P.; Roosta-Khorasani, F.; and Mahoney, M. 2017. Second-order optimization for non-convex machine learning: An empirical study. *arXiv preprint arXiv:1708.07827* .

Yu, Y.; Xu, P.; and Gu, Q. 2018. Third-order smoothness helps: Faster stochastic optimization algorithms for finding local minima. In *NeurIPS*, 4525–4535.

Zhou, D.; and Gu, Q. 2020. Stochastic recursive variance-reduced cubic regularization methods. In *AISTATS*, 3980–3990.

Zhou, M.; and Carin, L. 2015. Negative binomial process count and mixture modeling. *TPAMI* 37(2): 307–320.

Zhou, M.; Cong, Y.; and Chen, B. 2015. The Poisson gamma belief network. In *NIPS*, 3025–3033.

Zhou, M.; Cong, Y.; and Chen, B. 2016. Augmentable gamma belief networks. *JMLR* 17(1): 5656–5699.

Zhou, M.; Hannah, L.; Dunson, D. B.; and Carin, L. 2012. Beta-Negative Binomial Process and Poisson Factor Analysis. In *AISTATS*, 1462–1471.