Fast Second-Order Stochastic Backpropagation for Variational Inference

{ Kai Fan[1], Ziteng Wang[2], Jeffery Beck[1], James Kwok[2] and Katherine Heller[1] }
[1]Duke University, [2]Hong Kong University of Science and Technology

Abstract

We propose a second-order (Hessian or Hessian-free) based optimization method for variational inference inspired by Gaussian backpropagation, and argue that quasi-Newton optimization can be developed as well. This is accomplished by generalizing the gradient computation in stochastic backpropagation via a reparameterization trick with lower complexity. As an illustrative example, we apply this approach to the problems of Bayesian logistic regression and variational auto-encoder (VAE). Additionally, we compute bounds on the estimator variance of intractable expectations for the family of Lipschitz continuous function. Our method is practical, scalable and model free. We demonstrate our method on several real-world datasets and provide comparisons with other stochastic gradient methods to show substantial enhancement in convergence rates.

Reparametererization

Consider how to optimize an expectation of the form $\mathbb{E}_{q_{\boldsymbol{\theta}}}[f(\mathbf{z}|\mathbf{x})]$, where $\mathbf{z}$ and $\mathbf{x}$ refer to latent variables and observed variables respectively, and expectation is taken w.r.t distribution $q_{\boldsymbol{\theta}}$ and f is some smooth loss function. Here we particularly consider d_z dimensional Gaussian $q = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}, \mathbf{C})$, it suffers high cost on 2nd order derivative, e.g.

$$\nabla^2_{\mu_i,C_{k,l}}\mathbb{E}_q[f(\mathbf{z})] = 0.5 * \mathbb{E}_q\left[\nabla^3_{z_i,z_k,z_l} f(\mathbf{z})\right]$$
$$\nabla^2_{C_{i,j},C_{k,l}}\mathbb{E}_q[f(\mathbf{z})] = 0.25 * \mathbb{E}_q[\nabla^4_{z_i,z_j,z_k,z_l} f(\mathbf{z})]$$

Reparametererization Let $q = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}, \mathbf{R}\mathbf{R}^\top)$, then $\mathbf{z} = \boldsymbol{\mu} + \mathbf{R}\boldsymbol{\epsilon}$. We have low cost for derivative computation (up to 2nd order).

$$\nabla^2_{\boldsymbol{\mu},\mathbf{R}}\mathbb{E}_{\mathcal{N}(\boldsymbol{\mu},\mathbf{C})}[f(\mathbf{z})] = \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I}_{d_z})}[\boldsymbol{\epsilon}^\top \otimes \mathbf{H}]$$
$$\nabla^2_{\mathbf{R}}\mathbb{E}_{\mathcal{N}(\boldsymbol{\mu},\mathbf{C})}[f(\mathbf{z})] = \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I}_{d_z})}[(\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T) \otimes \mathbf{H}]$$

where gradient $\mathbf{g}$, Hessian $\mathbf{H}$ are evaluated at $\boldsymbol{\mu} + \mathbf{R}\boldsymbol{\epsilon}$ in terms of $f(\mathbf{z})$.
These two results w.r.t $\boldsymbol{\mu}, \mathbf{R}$ make parallelization possible, and reduce computational cost of the Hessian-vector multiplication due to the fact that $(A^\top \otimes B)vec(V) = vec(AVB)$.

Parameter Embedding

Standard multivariate Gaussian distribution has limited flexibility as an approximate distribution, we consider a parameter embedding trick with $\boldsymbol{\mu}(\boldsymbol{\theta}, \mathbf{x}), \mathbf{R}(\boldsymbol{\theta}, \mathbf{x})$ where $\boldsymbol{\theta} = (\theta_l)_{l=1}^d$ is the implicit but interested parameter.
We have similar fast Hessian derivation formula.

$$\nabla_{\theta_l}\mathbb{E}_{\mathcal{N}(\boldsymbol{\mu},\mathbf{C})}[f(\mathbf{z})] = \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})}\left[\mathbf{g}^\top \frac{\partial(\boldsymbol{\mu} + \mathbf{R}\boldsymbol{\epsilon})}{\partial\theta_l}\right]$$
$$\nabla^2_{\theta_{l_1}\theta_{l_2}}\mathbb{E}_{\mathcal{N}(\boldsymbol{\mu},\mathbf{C})}[f(\mathbf{z})] = \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})}\left[\mathbf{g}^\top \frac{\partial^2(\boldsymbol{\mu} + \mathbf{R}\boldsymbol{\epsilon})}{\partial\theta_{l_1}\partial_{l_2}} + \frac{\partial(\boldsymbol{\mu} + \mathbf{R}\boldsymbol{\epsilon})}{\partial\theta_{l_1}}^\top \mathbf{H} \frac{\partial(\boldsymbol{\mu} + \mathbf{R}\boldsymbol{\epsilon})}{\partial\theta_{l_2}}\right]$$

Note that the 1st and 2nd order gradient computations only involve matrix-vector or vector-vector multiplication, thus leading to an algorithmic complexity $\mathcal{O}(d_z^2)$ for each θ_l.

Hessian-vector Operation

If d is large, computation of $\mathbf{G}_{\boldsymbol{\theta}}$ and $\mathbf{H}_{\boldsymbol{\theta}}$ (differ from $\mathbf{g}$, $\mathbf{H}$ above) will be linear and quadratic w.r.t d, which may be unacceptable. Therefore, we reduce the computational complexity w.r.t d.
Let $F = \mathbb{E}_{\mathcal{N}(\boldsymbol{\mu},\mathbf{C})}[f(\mathbf{z})]$ and reparameterize $\mathbf{z}$ again, then we have

$$\mathbf{H}_{\boldsymbol{\theta}}\mathbf{v} = \frac{\partial}{\partial\gamma}\nabla F(\boldsymbol{\theta} + \gamma\mathbf{v})\Big|_{\gamma=0}$$
$$= \frac{\partial}{\partial\gamma}\mathbb{E}_{\mathcal{N}(0,\mathbf{I})}\left[\mathbf{g}^\top \frac{\partial(\boldsymbol{\mu}(\boldsymbol{\theta}) + \mathbf{R}(\boldsymbol{\theta})\boldsymbol{\epsilon})}{\partial\boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}\leftarrow\boldsymbol{\theta}+\gamma\mathbf{v}}\right]\Big|_{\gamma=0}$$
$$= \mathbb{E}_{\mathcal{N}(0,\mathbf{I})}\left[\frac{\partial}{\partial\gamma}\left(\mathbf{g}^\top \frac{\partial(\boldsymbol{\mu}(\boldsymbol{\theta}) + \mathbf{R}(\boldsymbol{\theta})\boldsymbol{\epsilon})}{\partial\boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}\leftarrow\boldsymbol{\theta}+\gamma\mathbf{v}}\right)\right]_{\gamma=0}$$

This trick is actually $\mathcal{R}$-operator and can be implemented by backpropagation.
Since all the gradients are evaluated in expectation involving intractable integral, we need Monte Carlo estimation. If f is an L-Lipschitz differentiable function and $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}_{d_z})$, we have **Variance Bound** $\mathbb{E}[(f(\boldsymbol{\epsilon}) - \mathbb{E}[f(\boldsymbol{\epsilon})])^2] \leq \frac{L^2\pi^2}{4}$ and **Bias Bound** $\mathbb{P}\left(\left|\frac{1}{M}\sum_{m=1}^M f(\boldsymbol{\epsilon}_m) - \mathbb{E}[f(\boldsymbol{\epsilon})]\right| \geq t\right) \leq 2e^{-\frac{2Mt^2}{\pi^2L^2}}$.

Variational Auto-Encoder

We apply these facts to stochastic variational inference. The first model we consider is variational auto-encoder (VAE), shown in Fig. 1. Basically, VAE describes an embedding process from the perspective of a Gaussian latent variable model. Each data point $\mathbf{x}$ follows a generative model $p_{\boldsymbol{\psi}}(\mathbf{x}|\mathbf{z})$ constructed by a non-linear transformation with unknown parameters $\boldsymbol{\psi}$ and a prior distribution $p_{\boldsymbol{\psi}}(\mathbf{z})$. The recognition model $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is used to approximate the true posterior $p_{\boldsymbol{\psi}}(\mathbf{z}|\mathbf{x})$, where $\boldsymbol{\phi}$ is similar to the parameters of a variational distribution.

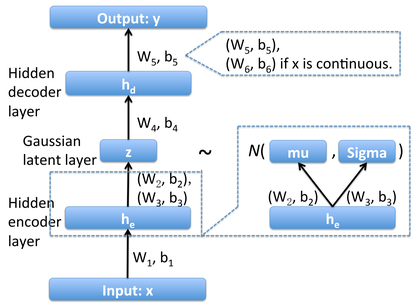

Figure 1: VAE represented by Deep Neural Nets

$$\log p_{\boldsymbol{\psi}}(\mathbf{x}^{(i)}) \geq \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})}[\log p_{\boldsymbol{\psi}}(\mathbf{x}^{(i)}|\mathbf{z})] - D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\psi}}(\mathbf{z})) = \mathcal{L}(\mathbf{x}^{(i)})$$

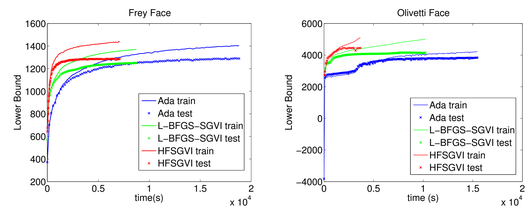

Figure 2: Convergence

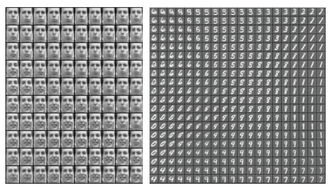

Figure 3: Manifold Learning

Logistic Regression

Given a dataset $\{\mathbf{x}_i, y_i\}_{i=1}^N$, where $y_i \in \{-1, 1\}$ is the binary label, the Bayesian logistic regression models the probability of outputs conditional on features and the coefficients $\boldsymbol{\beta}$ with an imposed prior. The likelihood and the prior usually take the form as $\prod_{i=1}^N g(y_i\mathbf{x}_i^\top\boldsymbol{\beta})$ and $\mathcal{N}(0, \boldsymbol{\Lambda})$ respectively, where g is sigmoid function and $\boldsymbol{\Lambda}$ is a diagonal covariance matrix for simplicity. We can propose a variational Gaussian distribution $q(\boldsymbol{\beta}|\boldsymbol{\mu}, \mathbf{C})$ to approximate the posterior of regression parameter. If we further assume a diagonal $\mathbf{C}$, a factorized form $\prod_{j=1}^D q(\beta_j|\mu_j, \sigma_j)$ is both efficient and practical for inference.

$$\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\sigma}) = \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(0,\mathbf{I})}[\log l(\boldsymbol{\mu} + \boldsymbol{\sigma} \odot \mathbf{z})] + \frac{1}{2}\sum_{i=1}^d \log\frac{\sigma_i^2}{\sigma_i^2 + \mu_i^2},$$

where l is the likelihood function.

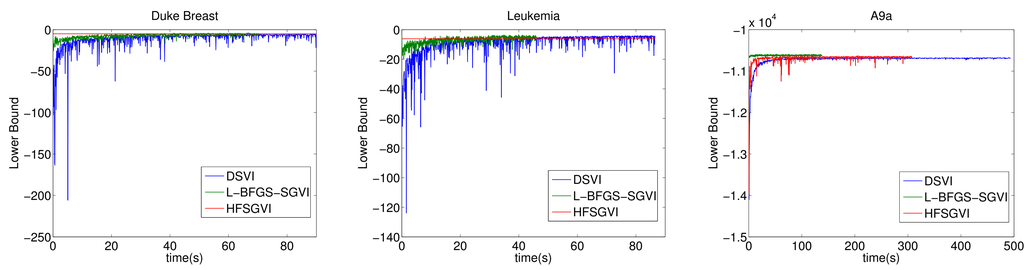

Figure 4: Convergence