Dimension Reduced Second Order Method for Neural Networks

Li Jingyang

National University of Singapore

li_jingyang@u.nus.edu

August 5, 2022

Overview

Spherical Trust-Region Method

The second-order method[1], at the k-th iterate, update the variable as

$$x^{k+1} = x^k + p^k,$$

where

$$\begin{aligned} p^k = \operatorname*{argmin}_p \quad & (c^k)^T p + \frac{1}{2} p^T Q^k p + \frac{\beta}{3}\alpha^3 \\ \text{s.t.} \quad & \|p\| \le \alpha, \end{aligned}$$

with $c^k = \nabla f(x^k)$ and $Q^k = \nabla^2 f(x^k)$.

[1]From Lecture Note 12.

Dimension Reduced Second Order Method[2]

Let $d^k = x^k - x^{k-1}$, $g^k = \nabla f(x^k)$, we search the best update in the subspace spanned by d^k and g^k

$$x^{k+1} = x^k + p^k, \text{ where } p^k \in \operatorname{span}\{d^k, g^k\}.$$

Then $p^k = -\alpha_1^k g^k + \alpha_2^k d^k$, where

$$\begin{aligned} \alpha^k = \underset{\alpha \in \mathbb{R}^2}{\operatorname{argmin}} \quad & m^k(\alpha) \\ \text{s.t.} \quad & \|\alpha\|_{G^k} \leq \Delta, G^k = \begin{bmatrix} (g^k)^T g^k & -(g^k)^T d^k \\ -(d^k)^T g^k & (d^k)^T d^k \end{bmatrix} \end{aligned}$$

[2] Chuwen Zhang et al. *DRSOM: A Dimension Reduced Second-Order Method and Preliminary Analyses*. 2022. DOI: 10.48550/ARXIV.2208.00208. URL: https://arxiv.org/abs/2208.00208.

Dimension Reduced Second Order Method

Here $m^k(\alpha)$ is the 2-dimensional quadratic model as

$$m^k(\alpha) \coloneqq f(x^k) + (c^k)^T\alpha + \frac{1}{2}\alpha^T Q^k \alpha,$$

where

$$Q^k = \begin{bmatrix} (g^k)^T H^k g^k & -(g^k)^T H^k d^k \\ -(d^k)^T H^k g^k & (d^k)^T H^k d^k \end{bmatrix} \in \mathcal{S}^2, \ c^k = \begin{bmatrix} -\|g^k\|^2 \\ (g^k)^T d^k \end{bmatrix} \in \mathbb{R}^2.$$

If the Hessian $H^k = \nabla^2 f(x^k)$ is not available, we can approximate it using the difference method

$$H^k g^k \sim \frac{\nabla f(x^k + tg^k) - g^k}{t}, \ H^k d^k \sim \frac{\nabla f(x^k + td^k) - g^k}{t}.$$

Feature of Neural Networks

Neural networks have high dimensional variable and complex function form, which makes getting function's value and gradient computational expensive, not to mention the Hessian.

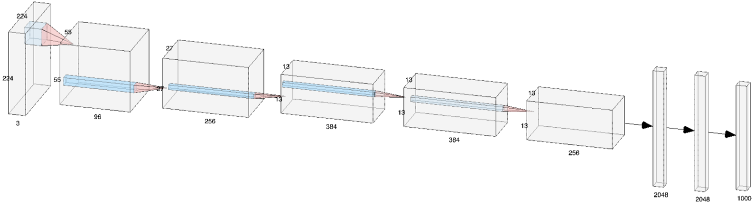


Figure: Architecture of a Convolutional Neural Network.

Feature of Neural Networks

$$Q^k = \begin{bmatrix} (g^k)^T H^k g^k & -(g^k)^T H^k d^k \\ -(d^k)^T H^k g^k & (d^k)^T H^k d^k \end{bmatrix} \in \mathcal{S}^2.$$

Approximate the Hessian using the difference method

$$H^k g^k \sim \frac{\nabla f(x^k + tg^k) - g^k}{t}, \; H^k d^k \sim \frac{\nabla f(x^k + td^k) - g^k}{t}.$$

Not computational efficient for the setting of Neural Networks due to high dimensional variable and complex function formulation.

Approximate the Hessian directly

Algorithm 1: Adam Optimizer

Initialize θ_0, $m_0 \leftarrow 0$, $v_0 \leftarrow 0$, $t \leftarrow 0$
While θ_t not converged
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$
$v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
Bias Correction
$\widehat{m_t} \leftarrow \frac{m_t}{1-\beta_1^t}$, $\widehat{v_t} \leftarrow \frac{v_t}{1-\beta_2^t}$
Update
$\theta_t \leftarrow \prod_{\mathcal{F},\sqrt{\widehat{v_t}}} \left(\theta_{t-1} - \boxed{\frac{\alpha \widehat{m_t}}{\sqrt{\widehat{v_t}}+\epsilon}}\right)$

Algorithm 2: AdaBelief Optimizer

Initialize θ_0, $m_0 \leftarrow 0$, $s_0 \leftarrow 0$, $t \leftarrow 0$
While θ_t not converged
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$
$s_t \leftarrow \beta_2 s_{t-1} + (1 - \beta_2) (g_t - m_t)^2 + \epsilon$
Bias Correction
$\widehat{m_t} \leftarrow \frac{m_t}{1-\beta_1^t}$, $\widehat{s_t} \leftarrow \frac{s_t}{1-\beta_2^t}$
Update
$\theta_t \leftarrow \prod_{\mathcal{F},\sqrt{\widehat{s_t}}} \left(\theta_{t-1} - \boxed{\frac{\alpha \widehat{m_t}}{\sqrt{\widehat{s_t}}+\epsilon}}\right)$

Positive diagonal approximation of Hessian using an exponential moving average of the square of gradient and the variance of gradient.

Freedom of Hessian approximation

Approximation of Hessian in Adam and Adabelief are required to be positive diagonal:

$$x^{k+1} = x^k - (H^k)^{-1} g^k.$$

In our DRSOM framework, we do not require the approximation of Hessian to be positive diagonal, even do not require it to be diagonal:

$$Q^k = \begin{bmatrix} (g^k)^T H^k g^k & -(g^k)^T H^k d^k \\ -(d^k)^T H^k g^k & (d^k)^T H^k d^k \end{bmatrix} \in \mathcal{S}^2.$$

Extension of search directions

More directions:

$$p^k \in \text{span}\{g^k, d^k, d^{k-1}, \cdots\},$$

where $d^{k-i} = x^k - x^{k-i-1}$, $i = 0, 1, 2, \cdots$.

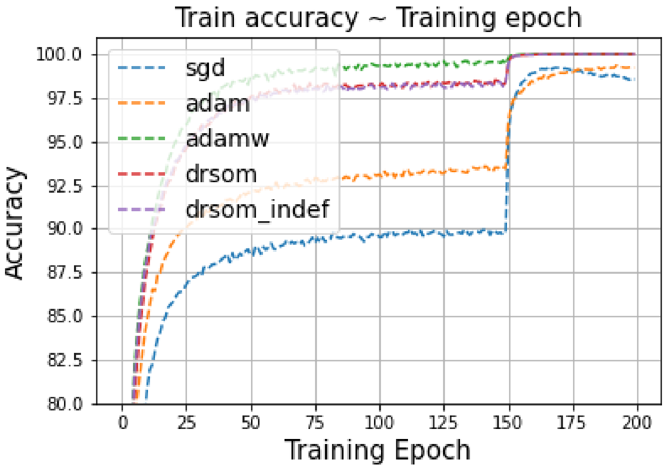
Random directions[3]:

$$p^k \in \text{span}\{\bar{g}^k, u^k\},$$

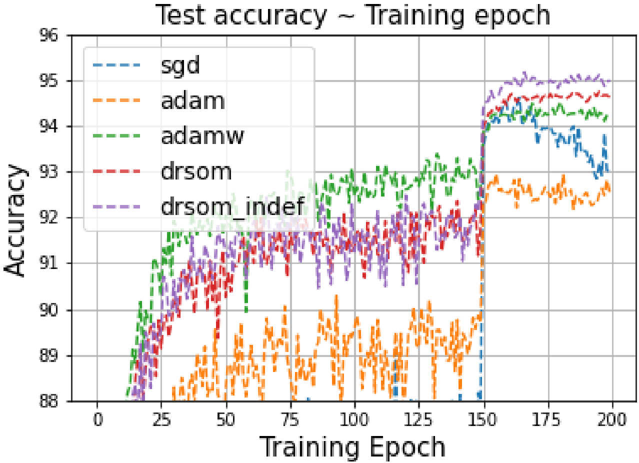
where $\bar{g}^k = g^k / \|g^k\|$, and u^k is a random vector $u^k \in N(0, I - \bar{g}^k(\bar{g}^k)^T)$ such that $E[u^k(u^k)^T + \bar{g}^k(\bar{g}^k)^T] = I$.

[3] From Lecture Note 16.

Train accuracy



Test accuracy



Conclusion

DRSOM is a flexible and robust framework for training neural networks, it has good optimization performance on training dataset and excellent generalization performance on test dataset.

Thank you for your attention!