Training Neural Networks with ADMM

Project 3 Report

Team 6

CHAI DI	LU GUANNAN	FENG YAJUN	LIN YUWEI
20458239	20454477	20404460	20446054
Big Data Technology	Big Data Technology	Big Data Technology	Big Data Technology
HKUST	HKUST	HKUST	HKUST
dchai@connect.ust.hk	gluab@connect.ust.hk	yfengaj@connect.ust.hk	ylinbq@connect.ust.hk

1. Problem Description

Nowadays, training neural networks always accompanies with big data volume and large number of iterations, which places a greater demand on efficiency and effectiveness of computation. However, rather than blindly using GPUs and increasing computation speed, we can trace back to the source and figure out some innovative gradient descent methods to solve the optimization problems. Actually, common-used stochastic gradient descent method has its drawbacks: all gradient-based approaches suffer from vanishing gradient problems and easily get stuck at local minima and saddle points. So in the paper "Training Neural Networks without Gradients: A Scalable ADMM Approach", it separates the objective function at each layer of a neural network into two terms that can be updated and solved as a set of minimization sub-problems without gradients, which to some extent helps alleviate the weaknesses met by stochastic gradient descent. This project aims to implement the optimization method in above paper. By training a deep neural network, we compare ADMM (Alternating Direction Methods of Multipliers) and standard back propagation method on image datasets. In this report, section 1 is the introduction of the project. Section 2 introduces the mathematical principles and algorithmic details of ADMM. Section 3 focus on the experiment process and results. Section 4 is the comparison and conclusion.

2. Principles

Given input vector $\boldsymbol{a_{l-1}}$, linear weight vector $\boldsymbol{W_l}$ and a non-linear neural activation function $\boldsymbol{h_l}$, the output is $\boldsymbol{a_l = h_l(W_l a_{l-1})}$. A neural network can be represented as

$$\boldsymbol{f(a_0; W) = W_k h_{k-1}(W_{k-1} h_{k-2}\left(\dots\left(W_2 h_1(W_1 a_0)\right)\right))}$$

where $\boldsymbol{W = \{W_l\}}$ is the weight matrix.

The optimization objective function is

$$\boldsymbol{\min_{W} l(f(a_0; W), y)}$$

To decouple the weights $\boldsymbol{W}$ from nonlinear functions $\boldsymbol{h_l}$, we get

$$\begin{aligned} &\boldsymbol{\min_{\{W_l\},\{a_l\},\{z_l\}} l(z_L, y)} \\ &\quad \boldsymbol{\text{s.t. } z_l = W_l a_{l-1}, \quad l = 1, 2, \dots, L} \\ &\quad\quad \boldsymbol{a_l = h_l(z_l), \quad l = 1, 2, \dots, L} \end{aligned}$$

The Augmented Lagrangian function (adding a $\boldsymbol{l_2}$ penalty) is

$$\begin{aligned} &\boldsymbol{\min_{\{W_l\},\{a_l\},\{z_l\}} L_\lambda(W, a, z)} \\ &= \boldsymbol{\min_{\{W_l\},\{a_l\},\{z_l\}} l(z_L, y) + \langle z_L, \lambda\rangle + \beta_L \|z_L - W_L a_{L-1}\|_2^2} \\ &\quad\quad \boldsymbol{+ \sum_{l=1}^{L-1}\left[\gamma_l \|a_l - h_l(z_l)\|_2^2 + \beta_l \|z_l - W_l a_{l-1}\|_2^2\right]} \end{aligned}$$

Where $\boldsymbol{\lambda}$ is a vector of Lagrange multipliers.

The augmented Lagrangian algorithm employing an alternating direction gives

$$\begin{cases} \boldsymbol{W_l = \operatorname{argmin}_{W_l} L_\lambda(W_l, a_l, z_l)} \\ \boldsymbol{a_l = \operatorname{argmin}_{a_l} L_\lambda(W_l, a_l, z_l)} \\ \boldsymbol{z_l = \operatorname{argmin}_{z_l} L_\lambda(W_l, a_l, z_l)} \\ \boldsymbol{\lambda = \lambda + \beta_L z_L - W_L a_{L-1}} \end{cases}$$

To respectively solve above update iterations,

(1) $\boldsymbol{W_l}$

$$\boldsymbol{W_l = \operatorname{argmin}_{W_l} \beta_l \|z_l - W_l a_{l-1}\|_2^2}$$

$$\boldsymbol{W_l \leftarrow z_l a_{l-1}^{+}}$$

(2) $\boldsymbol{a_l}$

$$\boldsymbol{a_l} = \underset{\boldsymbol{a_l}}{\mathbf{argmin}}\, \boldsymbol{\gamma_l}\|\boldsymbol{a_l} - \boldsymbol{h_l}(\boldsymbol{z_l})\|_2^2 + \boldsymbol{\beta_{l+1}}\|\boldsymbol{z_{l+1}} - \boldsymbol{W_{l+1}}\boldsymbol{a_l}\|_2^2$$

Taking gradient and setting it to 0, we get

$$(\boldsymbol{\beta_{l+1}}\boldsymbol{W_{l+1}^T}\boldsymbol{W_{l+1}} + \boldsymbol{\gamma_l})\boldsymbol{a_l} = \boldsymbol{\beta_{l+1}}\boldsymbol{W_{l+1}^T}\boldsymbol{z_{l+1}} + \boldsymbol{\gamma_l}\boldsymbol{h_l}(\boldsymbol{z_l}),$$

which leads to

$$\boldsymbol{a_l} \leftarrow (\boldsymbol{\beta_{l+1}}\boldsymbol{W_{l+1}^T}\boldsymbol{W_{l+1}} + \boldsymbol{\gamma_l})^{-1}(\boldsymbol{\beta_{l+1}}\boldsymbol{W_{l+1}^T}\boldsymbol{z_{l+1}} + \boldsymbol{\gamma_l}\boldsymbol{h_l}(\boldsymbol{z_l}))$$

(3) $\boldsymbol{z_l}$

$$\boldsymbol{z_l} \leftarrow \underset{\boldsymbol{z_l}}{\mathbf{argmin}}\, \boldsymbol{\gamma_l}\|\boldsymbol{a_l} - \boldsymbol{h_l}(\boldsymbol{z_l})\|_2^2 + \boldsymbol{\beta_l}\|\boldsymbol{z_l} - \boldsymbol{W_l}\boldsymbol{a_{l-1}}\|_2^2$$

which is a non-convex function but can fortunately be solved in closed form.

(4) $\boldsymbol{\lambda}$

After updating $\{\boldsymbol{W_l}\}, \{\boldsymbol{a_l},\}$, $\{\boldsymbol{z_l}\}$, the Lagrange multiplier update is given by

$$\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \boldsymbol{\beta_L}\boldsymbol{z_L} - \boldsymbol{W_L}\boldsymbol{a_{L-1}}$$

So the algorithm of ADMM is shown below:

Algorithm 1 ADMM for Neural Nets

Input: training features $\{a_0\}$, and labels $\{y\}$,
Initialize: allocate $\{a_l\}_{l=1}^{L=1}$, $\{z_l\}_{l=1}^{L}$, and λ
repeat
for $l = 1, 2, \cdots, L-1$ **do**
$W_l \leftarrow z_l a_{l-1}^{\dagger}$
$a_l \leftarrow (\beta_{l+1} W_{l+1}^T W_{l+1} + \gamma_l I)^{-1}(\beta_{l+1} W_{l+1}^T z_{l+1} + \gamma_l h_l(z_l))$
$z_l \leftarrow \arg\min_z \gamma_l \|a_l - h_l(z)\|^2 + \beta_l \|z_l - W_l a_{l-1}\|^2$
end for
$W_L \leftarrow z_L a_{L-1}^{\dagger}$
$z_L \leftarrow \arg\min_z \ell(z, y) + \langle z_L, \lambda \rangle + \beta_L \|z - W_L a_{l-1}\|^2$
$\lambda \leftarrow \lambda + \beta_L (z_L - W_L a_{L-1})$
until converged

3. Experiments

3.1 Datasets

The dataset used in this project， is from Yann LeCun, Corinna Cortes, Christopher J.C. Burges: “THE MNIST DATABASE”. It is a set of labeled hand-written digits, including a

training set of 60000 datasets in the size of 28*28 pixels, along with a training set labels. And this project utilizes the datasets labeled 0 or 1, which is 12665 out of 60000. Among the 12665 original datasets, 10000 of them are used as training set in this project, and the remaining 2665 sets are for testing.

3.2 Process

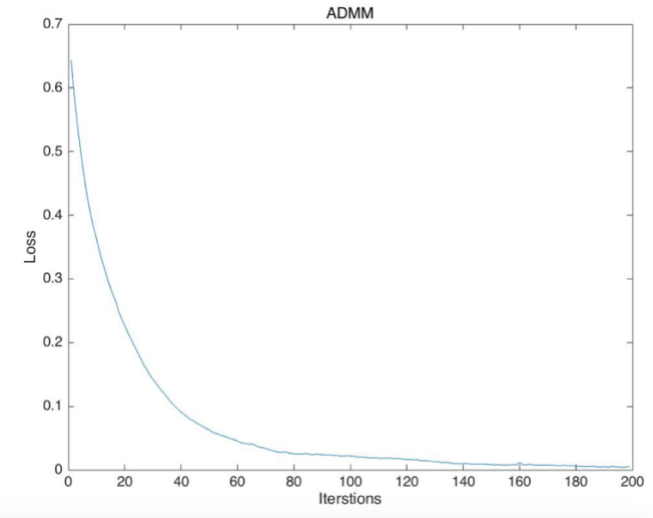
Firstly, a simple preprocessing is conducted. The original dataset is 60000 28*28 images, after filtering labels with 0 or 1, 12665 28*28 images remain, then the 12665 matrix are reshaped to a vector with the length of 784.

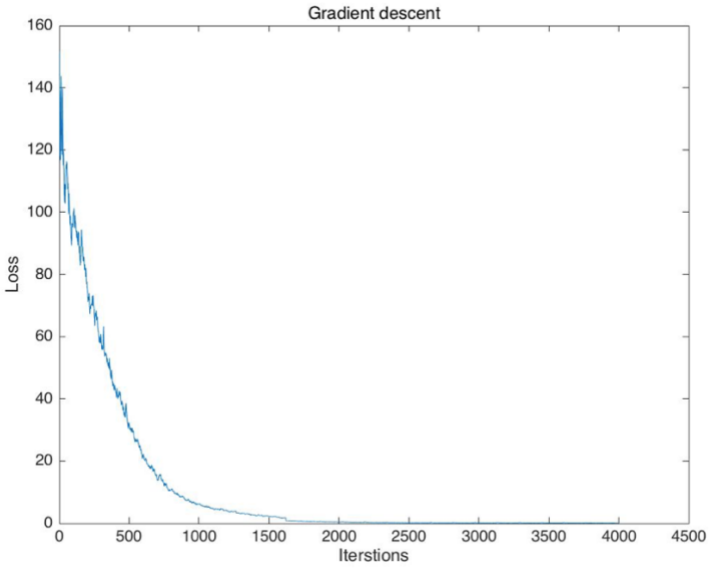
Afterwards, a fully connected neural network is built up, with size 100, 50, 10 and 1 sequentially, along with the mean square error function to calculate the loss. In the network, different parameter update functions, which are ADMM and Gradient descent, are implemented. To compare the performance, the evaluation criteria accuracy and error rate are calculated in each iterations.

Finally, the result in the accuracy aspect is 0.99061913696 for ADMM compared to 0.994746716689 for Gradient descent.

4. Comparison

The charts below show a comparison of performance between ADMM and Gradient descent in the form of loss.





1. The charts show that the ADMM is much smoother compared with Gradient descent. In the whole experiment process, for Gradient descent, the accuracy is quite different in each run, meanwhile there are occasions when the loss is decreasing while the accuracy remains.
2. Compared with Gradient, the loss with ADMM decreases much faster. Considering loss, ADMM reaches below 0.5 after about 60 iterations, while Gradient descent needs about 1500 iterations to get the same result.

5. Reference

5.1 Yann LeCun, Corinna Cortes, Christopher J.C. Burges: "THE MNIST DATABASE". http://yann.lecun.com/exdb/mnist/

5.2 Gavin Taylor, Ryan Burmeister, Zheng Xu, Bharat Singh, Ankit Patel, Tom Goldstein: Proceedings of The 33rd International Conference on Machine Learning, PMLR 48:2722-2731, 2016.