Deep Learning Project 3

Group 2
Hong Kong University of Science and Technology

December 15, 2017

Abstract

In this project we have implemented ADMM optimization methods for the convolution layers. For the convolution layers, we considered converting a convoluted matrix into a dense toeplitz matrix. The weights updates of kernels are done by solving a system of equations by sampling equations obtained from Toeplitz matrix. We found that our method worked faster than author's implementation since they only considered the pseudoinverse of the matrix. We used MNIST digit dataset (odd-even classification) and found that our method gives 90.6% over traditional backprop method 79.6%.

1 Introduction

ADMM (Alternating Direction Method of Multipliers) is the divide and conquer approach to solve a convex optimization problem. For this project, we have implemented ADMM for Neural Networks.

2 Weight Update

For each layer l, the optimal solution would minimize $||z_l - W_l a_{l-1}||^2$. The authors approached this problem by taking the pseudoinverse of a_{l-1}, which is a_{l-1}^*. In our opinion, taking pseudoinverse of a very tall matrix (in this case Toeplitz matrix of a_{l-1}) is an overkill and inaccurate. Hence, we approach this problem in a different manner.

Firstly, we write the convolution-kernel operation in the form of equation matrix, let's say $a_{l-1}W_l = z_l$. where W_l is the unravelled kernel and a_{l-1} is the Toeplitz matrix of the convolution layer. Here if $k = rank(W_l)$, then we only choose $k * k$ rows from the augmented $a_{l-1}W_l|z_l$ block and solve them using standard linear equation solvers. We found this to be a lot faster than taking traditional pseudoinverse.

3 Activation Update

Here we tweaked the original formulation which could work well for the ReLU function.

$$a_l = \begin{cases} (\beta_{l+1}W_{l+1}^T W_{l+1} + \gamma_l I)^{-1}(\beta_{l+1}W_{l+1}^T z_{l+1} + \gamma_l z_l), & \text{for } z > 0 \\ (\beta_{l+1}W_{l+1}^T W_{l+1} + \gamma_l I)^{-1}(\beta_{l+1}W_{l+1}^T z_{l+1}), & \text{otherwise} \end{cases}$$

4 Output Update

For this step, we need to minimize,

$$\min_z \gamma_l ||a_l - h_l(z)||^2 + \beta_l ||z - W_l a_{l-1}||^2$$

For the ReLU function, this function would be piecewise linear. Just putting the gradients to zero, we get,

$$\frac{\partial}{\partial z}\left(\gamma_l ||a_l - h_l(z)||^2 + \beta_l ||z - W_l a_{l-1}||^2\right) = 0$$

which gives,

$$z_l = \begin{cases} \frac{\gamma_l a_l + \beta_l w_l a_{l-1}}{\gamma_l + \beta_l}, & \text{for } z > 0 \\ w_l a_{l-1}, & \text{otherwise} \end{cases}$$

5 Experimental Setup

We used MNIST dataset for classification. However, due to time constraints we resorted to binary classification, rather than multiclass classification. Hence, we only classified even against odd numbers classification. Although we found that this was harder than numerical classification, since set of pixels were trying to generalize the image, making it even hard to classify.

References