README for Assignment 2 CS698U

abhinav garg

January 2017

Contents

Table 1: Comparision of time taken by fc and conv

Implemetation Method	Time taken by Conv (in ms)	Time taken by fc (in ms)
Tensorflow	0.3586	0.167
MatConvNet	0.5239	0.2480

1 Code Flow

The code contains following files. Their details are explained below:-

1. **MLP_Configuration_Driver.m** :- This is the main function which contains all the confuguration parameters and drives the program
2. **train_MLP.m** :- This is the main function which learns and trains the weight vector
3. **Activation.m** :- This file contains activation function and how to apply it.
4. **drev_Activation.m** :- Derivative of activation function.
5. **testMLP.m** :- This function is used to validate on test data and plot the graphs.
6. **evaluateMLP.m** :- Used to evaluate MLP after training it.
7. **Forward.m** :- Forward pass of the conv network
8. **conv.m** :- Backward pass of the conv layers.

Below is description of each file:-

1.1 MLP_Configuration_Driver.m

This function first loads MNIST data Then sets appropriate parameters.

1.1.1 Parameters that can be set/Degrees of freedom

I have implemented many more features than were asked in the assignment and hence there are a lot of variables that can be tuned to obtain optimum performance. The code is highly modular and commented.

1. learningRate :- learningRate for various methods
2. momentum :- momentum values for GD with momentum
3. methodToUse :- which trainig algorithm to use. 1 $\implies$ GD with Momentum , 3 $\implies$ Adagrad
4. a :- this parameters is to be set in activation and drev_Avtivation. a=1 $\implies$ Relu , a = 0 $\implies$ tanh.
5. batchSize : This can be provided as a list of all the batch sizes on which to run the code
6. epochs :- number of iterations of batchSize.

1.2 train_MLP.m

In this file the code for MLP is same as before we have just added conv code also this is summarized in 2 weights matrices Weights_conv and Weights_bias. And 2 functions Forward and conv for forward and backward propogation resp.

We also then plot training error.

We also plot validation error

1.3 testMLP.m and EvaluateMLP.m

These methods do testing and evaluation similar to forward pass.

2 Comparision of MLP v/s Matlab v/s Tensorflow

Earlir with MLP we were able to obtain accuracies of 97%. With conv layers added we are able to go upto 99%. This is due to the fact that MLP doesn't capture the locality features in an image. Whereas a conv net have filters to cature locality features.

Due to a better optimizer and huristics like decaying learning rate. Tensorflow converged faster than matlab implementation.

The graphs can be found in plots folder that suggests the above observations.

3 Time taken by conv vs fc

Table1 shows the findings. As we can see that time taken by conv layers are double as compared to that by fc. This is due to the fact that we have a large number of FLOPS in conv layers. And hence, They are more time consuming.

Also, Due to matlab's poor memory managment timings in matlab are in general higher than python

4 Number of params

The number of parameters can be calculated layer wise adding th weights dimension and biases

1. **Conv1:-** 5x5x6 Weights + 6 biases = 156
2. **Conv2:-** 5x5x6x16 Weights + 16 biases = 2416
3. **TOTAL Conv:- 2572**
4. **fc1:-** 400x120 Weights + 120 biases = 48120
5. **fc2:-** 120x84 Weights + 84 biases = 10164
6. **fc3:-** 84x10 Weights + 10 biases = 850
7. **TOTAL fc:- 59134**

We can clearly see that number of params of fc layers are much more than those of conv layer. This is due to the fact that conv layers share weights whereas fc layers don't.

5 Visualization and t-sne

The model was ran for over **10 days** to collect datas of various variables and obtain many types of graphs. All these graphs can be found in plots folder.

5.1 Visualizations

We can clearly see that after the first conv layer activations are that of boundaries and linear faetures. A notable activation is the last one which captures the negative of the image i.e. it captures the background

In maxpool we see the dimensions getting reduced and images going blurry.

In second conv more curvy features starts to appear. In digits that have curves like 0 and 8 we see neurons for upper curve, side curves etc.

5.2 t-Sne

The remarkable sucess of our model can be very clerly seen from the t-sne plot of our model. Initial t-sne plot is present in /plot/t-sne/t-sne.jpg which clearly tells how mingled these digit classes are. After processing through our network all the 10 classes seperates out clearly and hence we see such great results.

6 Batch Size

Plots present in validation and error folder. We see that as bacth size increases we need to increase learning rate so as to maintain same accuracy. This is due to the fact that more batchSize implies less updates. And also sharper gradients. And hence, we can afford a higher learning rate in that direction. The time taken by the model to train reduces as we do less frequent updates. Also as batch size increases our graphs get less noisy and we get smoother convergence. Again owing to the fact that we now have sharper and averaged gradients

A lot of plots can be found in these folders these indicate the progress through the project. the ones in the new folder are the final. Outside are the without bias graph which clearly shows overfitting.

7 Weights matrix

I have saved 4 epochs trained matrices to /matlab/weights folder for easy loading and testing.

8 References

1. tensorflow without a Phd. Most of the tensorflow code(vizualizations especially) is taken from this site and modified for lenet5
2. Numeric Gradient
3. GD methods
4. Adam
5. git I have consulted some repositories whenever stuck
6. cs231 course
7. Backprop of conv layers