Programming Assignment-1: MNIST Classification using Multilayer Perceptron

Anil Kumar, ee15s055@ee.iitm.ac.in

Due date: Aug 29th, Tuesday, 11:55pm

Note:

1. For any questions, please schedule a time with TAs before deadline according to their convenience. Please use moodle dicussion threads for posting your doubts and also check it before mailing to TAs, if the same question has been asked earlier.
2. Submit a single zip file in the moodle named as PA1_Rollno.zip containing report and folders containing corresponding codes.
3. Read the problem fully to understand the whole procedure.
4. Late submissions will be evaluated for reduced marks and for each day after the deadline we will reduce the weightage by 10%.

In this assignment, we will do classification on MNIST digits dataset using multilayer perceptrons. We will use a three hidden layer network for this. Input to the network is a digit image (28×28) flattened as 784 dimensional vector $\mathbf{x_i}$. Input is followed by $\mathbf{h_1}(1000)$, $\mathbf{h_2}(500)$, $\mathbf{h}_3(250)$. The number of units are mentioned beside the corresponding hidden layer. Use sigmoid activation for all these hidden layers. The output unit consists of 10 units for each digit. Let the output activation be linear. Since you want to do classification use softmax to get probabilities of digit belonging to 10 classes. This will be a 10 dimensional vector $\hat{\mathbf{y}}_\mathbf{i}$. Now use cross entropy loss between $\hat{\mathbf{y}}_\mathbf{i}$ and $\mathbf{y_i}$, where $\mathbf{y_i}$ is the one hot representation of ground truth label.

Note: You are not allowed to use any toolbox for back propagation algorithm implementation.

Numerical stability: Softmax$\left(\frac{e^{a_i}}{\sum_j e^{a_j}}\right)$ and cross entropy loss involves evaluating intermediate term e^{a_i} and $\sum_j e^{a_j}$. These term can be very large depending on a_i and their division might lead to numerical instability. To avoid it follow the hack mentioned in this link [1], under section *Practical issues: Numeric stability*.

Also, implement the cross entropy loss after softmax as follows:

$$L = -\sum_k p_k \log\left(\frac{e^{a_k}}{\sum_j e^{a_j}}\right) \quad (1)$$

$$= -\sum_k p_k \left(a_k - \log\sum_j e^{a_j}\right) \quad (2)$$

[1] http://cs231n.github.io/linear-classify/

1 Data preparation

Download the MNIST data from Lecun's page[2]. Follow instructions at this link [3] to load them MNIST data in matlab. It has standard train(60,000) and test samples(10,000). You have to use them accordingly for training and testing.

2 Training

Use a minibatch size of 64 while training. Use l_2 regularization with small value of regularization, λ around 0.005. Run the training for 8,000 iterations. For parameter initialization using random gaussian with standard deviation close to zero (say 0.08). Use SGD for training with momentum based acceleration.

3 Submissions

1. Submit the back-propagation equations for all the layers in terms of matrix notation followed in the class. Include them in the report.

2. You need to submitt your training progress as a plot showing train loss and test loss converengence over iterations. Test loss need not be calculated for every iteration, instead you can calculate it for every 200 iterations or so. Observe the convergence with varying learning rates.

3. Learning rate scheduling is an important aspect for training. Try a simple strategy of decaying learning rate by a factor, say $\gamma = 0.85$ every 250 iterations. Compare the training loss convergence plots with and with out the scheduling. Comment on the plots.

4. Now change the activation function to relu instead of sigmoid and submit the training plots and the final test accuracy. Compare this convergence with the sigmoid activation.

5. In both the cases after training, choose a fixed set of 20 images from test set and show the network's top 3 guesses for each of them.

[2] http://yann.lecun.com/exdb/mnist/

[3] http://ufldl.stanford.edu/wiki/index.php/Using_the_MNIST_Dataset