EE6132 : Advanced Topics in Signal Processing

Programming Assignment 1: MNIST Classification using Multilayer Neural Networks

Due Date: Sep 01, 2019

Instructions

1. You may do the assignment in MATLAB, Python.
2. For any questions, please schedule a time with TAs before deadline according to their convenience. Please use moodle discussion threads for posting your doubts and also check it before mailing to TAs, if the same question has been asked earlier.
3. Submit a single zip file in the moodle named as *PA1_Rollno.zip* containing report and folders containing corresponding codes.
4. Read the problem fully to understand the whole procedure.
5. Do not copy any part from any source including your friends, seniors or the internet.
6. Late submissions will be evaluated for reduced marks and for each day after the deadline we will reduce the weightage by 10%.

The aim of this assignment is to experiment with Multilayer Feedforward Neural Network (MLP) with Backpropagation (BP) learning. Please write your own code for BP algorithm and MLP training and testing. Do not use any libraries or copy directly any code from any resource coding BP gives you greater insight into the practical issues.

Dataset : The aim is to code a complete handwritten digit recognizer and test it on the MNIST digit dataset. (Download the dataset and its description from: http://yann.lecun.com/exdb/mnist/). The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

MLP Architecture and Training (baseline model): Input to the network is a digit image (28×28) flattened as 784 dimensional vector x_i. Input is followed by layers $h_1(500)$, $h_2(250)$, $h_3(100)$. 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 $\hat{y}_i$. Now use cross entropy loss between $\hat{y}_i$ and y_i, where y_i is the one hot representation of ground truth label.

1. Implement forward and backward propagation of the MLP without using any of the deep learning frameworks.

2. Train the MLP using cross entropy loss with a minibatch size of 64. Use *Gradient Descent* for optimizing the model parameters. Run the training algorithm for 15 epochs.

   Initialize the biases to 0 and sample the model weights from uniform distribution (Glorot Initialization): $w_{i,j}^{l} \sim \mathcal{U}\left[-d_l, d_l\right]$ where $d_l = \sqrt{\frac{6}{fan\_in + fan\_out}}$ where fan_in is the number of input units in the weight tensor and fan_out is the number of output units in the weight tensor.

   Note: You need to submit your training progress as a plot showing train loss and test loss convergence 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. Report the confusion matrix that shows the kind of errors that your classifier makes. In this problem, your confusion matrix is a 10×10 matrix, where the rows represent the true label of a test sample and the columns represent the predicted labels of the classifier. Report the average error rate as well as standard deviation of the error rate along with other metrics such as Precision, Recall/Sensitivity, and F1 score.

Activation Function: In this sub-question, we experiment with different activation functions.

1. On the baseline model, change the activation function to (a) *ReLU* (b) *Tanh*. Compare the convergence with the *sigmoid* activation. Submit a training plot along with other necessary metrics. Comment on the results.

2. In each of the above cases, compute the percentage of inactive neurons (i.e., neurons with gradient magnitude less than 10^{-5}). Submit a plot for percentage of inactive neurons vs iterations for each of the hidden layer. Comment on the results

Regularization: In this part of the assignment, we study the effect of various regularization algorithms on the performance of the MLP.

1. Try adding a little bit of gaussian noise to h in the forward pass. How does it effect results? Try adding a little bit of gaussian noise to h when doing the backward pass. How does it effect results? Which is better? Comment on the results.

2. Try one variation of data-augmentation using *de-skewing*, *adding noise*, etc. and also try one variation with other regularization terms such as L_1-*Regularization*, L_2-*regularization/weight-decay*, *tangentprop*, etc. and compare your results with the baseline model without regularization. Comment on the results.

Hand Crafted Features: Here, we study the performance of MLP, K-NN and SVM techniques on MNIST dataset with off-the-shelf features as input.

1. Instead of using the flattened image as input, use any of the off-the-shelf feature extractor[1] to get the feature representation for images. Due credit will be given for choosing non-trivial feature extraction.

   Choose your own architecture (number of hidden layers, number of nodes in each layer, activations, loss function, regularization).

   Your recognizer needs to read the image data, extract features from it and use a multilayer feedforward neural network classifier to recognize any test image.

   Submit a training plot along with other necessary metrics. Comment on the results.

2. Compare the results from above with classification using *k-NN* (Euclidean distance based) and *SVM*[2] classifier. Present an analysis and discussion of your results.

–end–

[1] SIFT, SURF, HOG, MSER etc. Open-source implementations are available online for all of these. For example, If you are using MATLAB, some of these are availble as in-built functions or you can use VLFeat library. If you are using Python, you can use OpenCV.

[2] Python - use LIBSVM or scikit; MATLAB - use VLFeat