E9 205 – Machine Learning for Signal Processing

Homework # 5
Due date: Nov. 24, 2017

End of the day for coding part and submitting in class on Nov 24 for analytical part.
Coding assignment submitted to e9205mlsp2017 aT gmail doT com

November 15, 2017

1. **Implementing BackPropagation**
   leap.ee.iisc.ac.in/sriram/teaching/MLSP_17/assignments/HW4/Data.tar.gz
   The above dataset has training/test subject faces with happy/sad emotion are provided in the data. Each image is of 100×100 matrix. Perform PCA to reduce the dimension from 10000 to $K = 12$. Implement a 2 hidden layer deep neural network (each layer containing 10 neurons) to classify the happy/sad classes. Use the cross entropy error function with softmax output activations and ReLU hidden layer activations. Perform 20 iterations of back propagation and plot the error on training data as a function of the iteration. What is the test accuracy for this case and how does it change if the number of hiddent neurons is increased to 15. (**Points** 30)

2. **M**NIST data - Download the dataset of hand-written digits
   http : //yann.lecun.com/exdb/mnist/
   containing 10 classes. [Reduce the number of samples in training data if your computing powers are limited as required by random subsampling]. The Keras package is needed for the rest of the question *https : //pypi.python.org/pypi/Keras*

   (a) **I**mplementing DNNs - Use the Keras package to implement a DNN model with 2 and 3 hidden layers. Each hidden layer contains 512 neurons. What is the performance on the test dataset for this classifier

   (b) **I**mplementing CNNs - Use the same package to implement a CNN model with one layer of convolutions (kernel size of 3×3 with a 2-D convolutional layer and having 128 filters) followed by two dense layers of 256 neurons. Compare the performance of the CNN with the DNN.

   (c) Provide your answers with analysis, plots for various choices of hidden layer dimensions and filter sizes in the CNN.

   (**Points** 40)

3. **Neural Networks - Cost Function** - Let us define a NN with softmax output layer and $\{\mathbf{o}_i\}_{i=1}^M$ and $\{\mathbf{y}_i\}_{i=1}^M$ denote the input and targets to the NN. The task is classification with hard targets $\mathbf{y} \in \mathcal{B}^{C\times 1}$, where $\mathcal{B}$ denotes boolean variable (0 *or* 1), and C is the number of classes. Note that every data point $\mathbf{o}_i$ is associated with only one class label

c_i where $c_i \in \{1, 2, .., C\}$ classes. The output of the network is denoted as $\mathbf{v}_i$ where $\mathbf{v}_i = \{v_i^1, v_i^2, .., v_i^C\} \in \mathcal{R}^{C\times 1}$ and $0 < \mathbf{v}_i < 1$. The NN cost function can defined using mean square error (MSE).

$$J_{MSE} = \sum_{i=1}^{M} ||\mathbf{v}_i - \mathbf{y}_i||^2$$

Show that the MSE is bounded in the following manner

$$\sum_{i=1}^{M} \left[1 - v_i^{c_i}\right]^2 \leq J_{MSE} \leq 2 \sum_{i=1}^{M} \left[1 - v_i^{c_i}\right]^2$$

(**Points** 15)

4. **Learning of NN weights** - Consider a quadratic error of the form

$$J = J_0 + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}^*)^T \boldsymbol{H} (\boldsymbol{w} - \boldsymbol{w}^*)$$

where $\boldsymbol{w}^*$ represents the minimum of the function and $\boldsymbol{H}$ represents the Hessian matrix which is positive definite. Let the eigenvalues and eigenvectors of $\boldsymbol{H}$ be denoted by λ_j and $\boldsymbol{u}_j$ respectively for $j = 1, .., n$. The gradient descent based update of $\boldsymbol{w}$ is given by

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^t - \eta \frac{\partial J}{\partial \boldsymbol{w}}$$

where t is the iteration index. If the weights $\boldsymbol{w}$ were initialized to $\boldsymbol{w}^0 = \mathbf{0}$, then show that after Q steps of gradient descent, the weights are given by

$$w_j^Q = ([1 - (1 - \eta\lambda_j)]^Q) w_j^*$$

where $w_j = \boldsymbol{w}^T \boldsymbol{u}_j$. Show that as $Q \to \infty$, $\boldsymbol{w}^Q \to \boldsymbol{w}^*$ if $|1 - \eta\lambda_j| < 1$.

(**Points** 15)