CSL407 Machine Learning
Homework 3

Due on 12/9/2014, 11.55pm

Instructions: Upload to your moodle account one zip file containing the following. Please do not submit hardcopy of your solutions. In case moodle is not accessible email the zip file to the instructor at ckn@iitrpr.ac.in. Late submission is not allowed without prior approval of the instructor. You are expected to follow the honor code of the course while doing this homework.

1. A neatly formatted PDF document with your answers for each of the questions in the homework. You can use latex, MS word or any other software to create the PDF.
2. Include a separate folder named as 'code' containing the scripts for the homework along with the necessary data files. Name the scripts using the problem number.
3. Include a README file explaining how to execute the scripts.
4. Name the ZIP file using the following convention rollnumber_hwnumber.zip

---

In this homework you will be experimenting with multi layer perceptrons. You will also solve few other problems that will help to improve your understanding of multi-layer perceptrons (MLP) and Backpropagation algorithm.

1. (10 points)This is a warm up exercise for problem 4. You will experiment with a simple 2 dimensional 3 class classification problem to visualize decision boundaries learned by a MLP. Our goal is to study the changes to the decision boundary and the training error with respect to parameters such as overlap between the data points of different classes, number of training iterations, number of hidden layer neurons and finally the learning rate. The Matlab script `hw3part1.m` in the `hw3.zip` file provides the necessary skeletal framework for performing these experiments. Before we start the experiments, complete the following functions in the script. The locations in the script where your code needs to be added have been marked as `'TO DO'`.

   (a) `[w v trainerror] = mlptrain(X, Y, H, eta, nEpochs)` that outputs the weights of the connections between input and hidden layer units - `w` and between hidden and output layers units - `v`, given the training data `X`, training labels `Y`, the number of hidden layer units `H`, the learning rate `eta`, and the number of training epochs `nEpochs`. We will assume cross entropy as the error function, sigmoid as the activation function for the hidden layer units and softmax function for the output layer units. The output of the next work will be the probability of classification.

   (b) Compute the training error using the learned weights at the end of each epoch. The error value is stored in the variable `trainerror(epoch)` that is returned by the `mlptrain` function.

   (c) `ydash = mlptest(X, w, v)` that determines the outputs of the output layer units of the MLP, given the data `X`, the learned weights `w` and `v`. This function essentially performs a forward pass on the learned MLP using the data.

   (d) If you have correctly implemented the above functionalities, we are all set to run some experiments. Let us fix the number of epochs to be 1000, the learning rate to be 0.01 and study the relation between the training error and the number of hidden layer units. Vary the number of hidden layer units in powers of 2, starting from 2 and going up till 128. What are your observations? Include in the report a single graph that includes the plots for training error against number of epochs for the different choices of hidden layer units. Also include separate plots for the decision boundaries for three choices of hidden layer units - 2, 4 and 64. What are your observations from these plots?

(e) Let us again fix the number of epochs to be 1000 and also fix number of hidden layer units (a number that you think is reasonable). What happens to the training error curve for three learning rates - 0.001, 0.01 and 0.1. Include in the report a single graph with the training error plots for the three learning rates along with your observations.

(f) Increase the amount of overlap between the classes by changing the value of the standard deviation `sd` of the Gaussian distributions to 0.75. Fix the learning rate to be 0.01 and the number of hidden layer units to be 16. Report the training error at the end of 1000, 5000 and 10000 epochs. What do you observe? Include in the report the decision boundary for these different training epoch values. What changes do you observe to the decision boundary?

2. (3 points) Consider a two layer MLP with two inputs a and b, one hidden unit c, and one output unit d. This network has five weights $w_{ca}, w_{cb}, w_{c0}, v_{dc}, v_{d0}$. Initialize these weights to the values $(0.1, 0.1, 0.1, 0.1, 0.1)$, then give their values after each of the first two training iterations of the Backpropagation algorithm. Assume learning rate $\eta = 0.3$, momentum $\alpha = 0.9$, incremental weight updates and the following training examples

a	b	d
1	0	1
0	1	0

3. (2 points) Design a two input perceptron that implements the boolean function $A \wedge \neg B$.

4. (15 points) In this question you will implement a MLP to classify digits from the MNIST dataset. More specifically, you will compute the gradient using Backpropagation and perform stochastic gradient descent on the data. Let us consider a network with H hidden units, K outputs, and inputs of size D; we can then define the following parameters:

- $\mathbf{w}$ - a $H \times (D+1)$ matrix of weights for the connections between input to hidden layer units. The h^{th} row of this matrix corresponds to the weights of the input connections starting from the bias term to the D^{th} input term coming into the h^{th} hidden layer unit.
- $\mathbf{v}$ - a $K \times (H+1)$ matrix of weights for the connections between the hidden layer to output layer units. Similarly the i^{th} row of this matrix corresponds to the weights of the hidden layer connections starting from the bias term to the H^{th} hidden layer term coming into the i^{th} output layer unit.

We will use tanh as the activation function for the hidden units instead of the sigmoid function. tanh has profile similar to the sigmoid function but has a range [-1 1].

$$tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$$

Now for some input $\mathbf{x}$ we can write the output of the hidden layer units $\mathbf{z}$ as

$$\mathbf{z} = \tanh(\mathbf{x}\mathbf{w}^T)$$

Since the goal is to classify digits, the softmax function will be used in the output layer units and therefore the output of the i^{th} unit will be

$$ydash_i = \frac{exp(\mathbf{z}\mathbf{v}_i^T)}{\sum_{k=1}^{K} exp(\mathbf{z}\mathbf{v}_k^T)}$$

The output $ydash_i$ can be interpreted as the probability that the input is of class i. Given N data samples $\{\mathbf{x}_n, \mathbf{t}_n\}_{n=1}^{N}$, let $ydash_{ni}$ be the i^{th} output evaluated for the n^{th} sample. Then we can write the error on this set as

$$-\sum_{n=1}^{N}\sum_{i=1}^{K} t_{ni} \log(ydash_{ni})$$

(a) Derive the gradient of this error function with respect to the each weight parameter. Note that the derivative of $h(a) = \tanh(a)$ is given by $h'(a) = 1 - h(a)^2$.

(b) Implement a MLP to classify the MNIST digits. The dataset is included in the accompanying zip file (`mnist.mat` and for non-Matlab users `data.txt` and `label.txt`). A skeletal code that loads the dataset and creates the necessary train, validation and test splits is also provided in the file hw3part4.m. You will implement the gradients derived in part(a) of the question. The section where you have to add code has been marked as `'TO DO'`. Set the number of hidden layer units to be 500, learning rate to be 0.01. We will use a modified form of stochastic gradient descent, where instead of updating the weights after every input, the updates are made in batches of input data. Set the batch size to be 25. Set the number of epochs to be 100. Note that the training process will take around 2-3 hours, so make sure that you don't wait till the last minute to start working on the homework.

(c) Include a plot of the training and validation error as a function of epochs, at most 2 instances for each of the 10 digits that are misclassified from the test set.