Instructions:

1. Prepare a report (including your answers/plots) to be uploaded on Moodle.
2. The report should be typeset (no handwriting allowed except for lengthy derivations, which may be scanned and embedded into the report).
3. Show all steps of your work clearly.
4. Unclear presentation of results will be penalized heavily.
5. No partial credits for unjustified answers.
6. **Use of any toolbox or library for neural networks is prohibited.**
7. Return all Matlab/Python code that you wrote in a single `.m`/`.py` file.
8. Code should be commented,
9. The code file should NOT return an error during runtime.
10. If the code returns an error at any point, the remaining part of your code will not be evaluated (i.e., 0 points).

In this project we will consider the handwritten character recognition by using a Multilayer structure and MNIST dataset. MNIST contains total of 70,000 handwritten characters, each has 28×28 pixel (grayscale) image. 60,000 of these are reserved as training set and the remaining 10,000 are reserved as test set. You may access the dataset through :

http://yann.lecun.com/exdb/mnist/

1 : 28x28=784, and the first thing to do is to convert this image to a vector of length 784. This is called flattening, and simple python, numpy codes etc. can do that. Note that due to grayscale, each pixel has a value between 0-255. To speed up the training, input vectors are usually normalized (simple way is to divide each pixel entry by 255).

2 : For classification we will use a 2 layer structure. Output layer will have 10 neurons, where each output neuron represents a digit. If the network is trained correctly, the output neuron having the maximum value should correspond to the digit presented at the input. There will be a hidden layer with N neurons. Choose 3 different values of N as $N = 300, 500, 1000$.

3 : For the nonlinear activation we will use 2 different cases :

Case 1 : All activations are $tanh(v) = \frac{e^v - e^{-v}}{e^v + e^{-v}}$

Case 2 : Hidden layer activation is $RELU(v) = \max\{0, v\}$, and the output layer activation is $sigmoid(v) = \frac{1}{1+e^{-v}}$.

4 : Preassign the output neurons to digits, e.g. 1st neuron for digit 1, 2nd neuron for digit 2, etc.

For the case 1 nonlinearity, the desired output for the class of digit i is as follows : only the neuron corresponding to the chosen digit class should be 1, all others should be -1.

For the case 2 nonlinearity, the desired output for the class of digit i is as follows : only the neuron corresponding to the chosen digit class should be 1, all others should be 0.

5 : The error to be minimized is classical mean squares error we discussed at the class i.e.

$E(n) = \sum_{r=1}^{10} \mid d_r(n) - o_r(n) \mid^2 \quad , \quad E_{ave} = \frac{1}{N} \sum_{n=1}^{N} E(n)$

6 : The final parameter is the selection of learning coefficient η. Choose 3 different values as $\eta = 0.01, 0.05, 0.09$.

7 : For the initialization of weights, use random initialization, e.g. uniform random distribution in a small and symmetric interval around origin, e.g. in [-0.01,0.01], or random gaussian distribution with zero mean and small σ, e.g. $\sigma = 0.01$. Threshold values are usually initialized as 0.

For each of the cases given in **2**, **3**, **6**, run the back propagation algorithm. Note that we have 3x3x2=18 different runs. (For statistically meaningful results, one should run as many times as possible and take the average, look at the standard deviation etc.) For stopping conditions, you may use an upper bound on the number of epochs, typically 50-100 epochs would suffice. You may also monitor the training set error, and if it changes slowly you may stop the algorithm. After that, record the relevant data, e.g. the training set error, number of epochs (and CPU time, if available), test set error, percentage of correct classification in test set, patterns incorrectly classified, etc.

8 : After **7**, record the best performing case based on test set error/classification accuracy (e.g. N, η and the activation function of the best performing case). **For this best performing configuration only**, run the back propagation again by using mini batch training with mini batch size $N = 10, 50, 100$. Note that this will require 3 extra simulations. Record the relevant data as indicated above. Determine the best performing mini batch size.

9 : Now we will use L_2 weight regularization. The new cost E_{reg} is :

$E_{reg}(n) = E(n) + \frac{\lambda}{2} \sum_{allweights} w_{ij}^2$

For the best performing case in **8** (with the best performing mini batch size), run the back propagation with L_2 regularization cost with $\lambda = 0.001, 0.01$. Record the relevant data as indicated above. Note that this will require 2 extra simulations. Determine the best performing λ. Compare it with the best result of **8** (which is the case $\lambda = 0$).

10 : Write your results in a nice report format, preferably in pdf format.