Introduction to Machine Learning *Rujuta Palande*

HW4

Please note that only PDF submissions are accepted. We encourage using LaTeX to produce your writeups. You'll need *mydefs.sty* and *notes.sty* which can be downloaded from the course page.

Linear dimensionality reduction:

1. You have used MNIST dataset in the previous homeworks. This time, you will reduce the dimensionality of data to improve the accuracy of digit recognition. Please randomly choose 1000 training data points (you can use the sample code given in HW1).

2. LIBSVM is an off-the-shelf implementation for SVM. It is very powerful and easy to use. It supports various formulations of SVM including kernel SVMs. Please download it, read the README file, and install it with Matlab interface. You can simply train a model by:

   model = svmtrain(training_label_vector, training_instance_matrix [, 'libsvm_options']);

   and test the model by:

   [predicted_label, accuracy, decision_values] = svmpredict(testing_label_vector, testing_instance_matrix, model [, 'libsvm_options']);

   Look at the README file to learn how to use it. Note that adding '-t' to the options in training, uses linear SVM formulation, the one you implemented in the previous homework.

3. Train an SVM on MNIST data using 1000 training data and test it on all testing data and report the accuracy. You should get the predictions from *svmpredict* and calculate the accuracy yourself.
   `Answer : Calculated accuracy is 84.85`

4. Use PCA on training data to reduce the dimensionality of data from 784 to 50. Note that you should subtract the mean of data first. To make it faster, please use the SVD trick that we discussed in the class to avoid calculating 784x784 size covariance matrix. Project all training data into the new 50D feature space, project back to reconstruct the data and compare it with the original data. Change the number of dimensions from 1 to 500, and plot the mean squares error vs. number of dimensions. Visualize the first 10 Eigen vectors by reshaping them and using *imagesc* and *subplot* commands in Matlab. If it is slow on your computer, you can use divide the range of [1,500] logarithmically using *logspace* command to 50 points rather than all 500 points.

5. Project both training and testing data onto the lower dimensional space and train an SVM to see if it improves the performance.
   `Answer : Yes it improves the performance. The accuracy is 92.9`

6. Repeat the above item for different values of dimensions and plot the curve of SVM accuracy vs. number of dimensions. Please use the following number of dimensions: [2 5 10 20 30 50 70 100 150 200 250 300 400 500 748]

7. Matlab has a built-in implementation for neural network. Please take a look at examples for *newff* command and repeat the above two steps using a neural network instead of SVM. The neural network will have 10 outputs, one for each category.

8. `Answer: Calculated the weight vector and bias by using the coef and SVs. Please find the attached code`

Figure 1: Q.4 mean squares error vs. number of dimensions

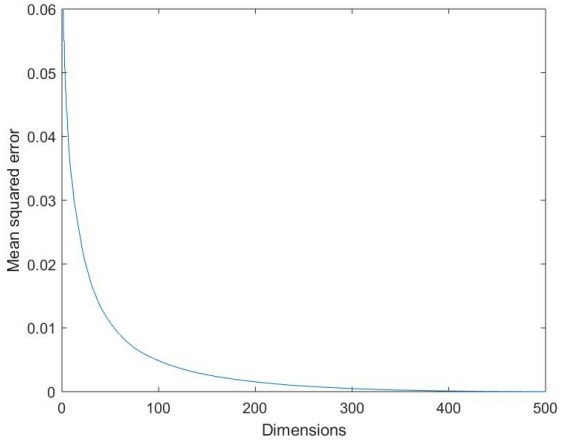

Figure 2: Q.4 visualizing the eigen vectors

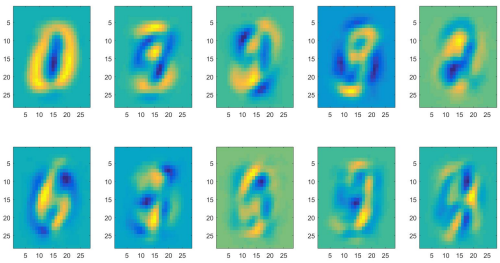

Figure 3: Q.6 SVM accuracy vs. number of dimensions

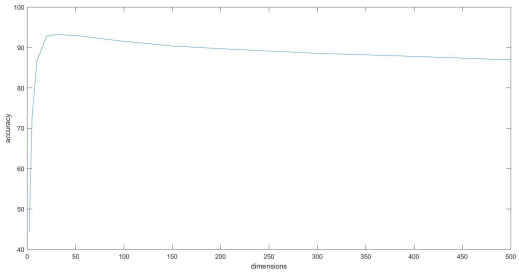

Figure 4: Q.7 SVM accuracy vs. number of dimensions when using neural net and no. of hidden layer 1 and neurons 100