Introduction to Machine Learning *Khatwani Mohit, AJ75499*

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.

   The accuracy found out after using 1000 training data and *svmpredict* is 83.13 percent.

4. As a bonus point, please calculate the decision values and accuracy yourself without using *svmpredict* and compare it with the results of *svmpredict*. You should do this by accessing the weight values inside the structured variable *model* in the output of training. The point is to make sure we can interpret the result of training rather than seeing it as a black box.
   This accuracy is calculated by using same 1000 training data and without using inbuilt *svmpredict* function.
   Accuracy reported is the same 83.13 percent.

5. 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.

   The training data dimension is reduced to 50.

Figure 1. shows visualization of first 10 eigen-vectors when dimension is reduced to 50.

Figure 2. shows plot of mean-square-error vs number of dimensions with range [1,500].

6. Project both training and testing data onto the lower dimensional space and train an SVM to see if it improves the performance.

   After projecting training and testing data on lower dimensional space of 50D. Accuracy reported with SVM is 83.37 percent.

7. 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]

   Figure 3. shows Accuracy vs dimension plot using SVM.

8. Matlab has a builtin 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.

   Accuracy reported by using NN provided by sklearn is 89.15 percent.

   Figure 4. shows Accuracy-vs-dimension plot using Neural Network.

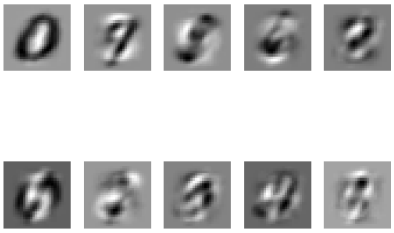

Figure 1: Figure 1. Visualization of top 10 eigen vectors

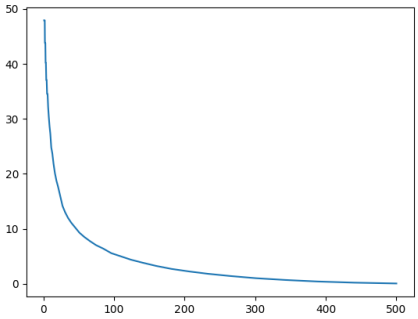

Figure 2: Figure 2. Mean square error vs dimension plot

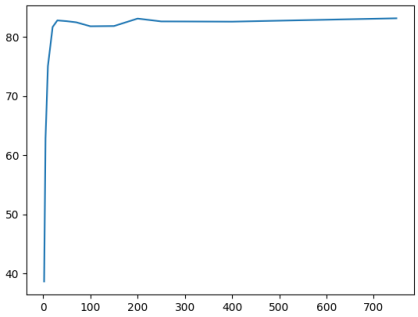

Figure 3: Figure 3. Accuracy vs dimension plot using SVM

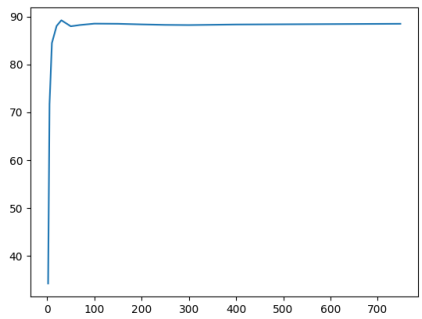

Figure 4: Accuracy vs dimension plot using Neural Network