# CSCI 5523 Project 3 Report

In this project, three java programs are established to apply two classify models on a specific dataset. The three classify models are the k-nearest neighbor, the ridge regression and the non-negative ridge regression. The whole dataset is divided into three subsets which different operations performed on the data. The first subset which is stored in the file rep1, performs no operations on the original data. The format of lines for this file is just object labels with corresponding pix values. The second subset which is stored in rep2 performed a SVD operation on the raw data. The whole data now is stored as a 30-dimensional SVD representation. for the rep3, which is the third part of the whole dataset, contains the 495-dimensional second order representation for the above SVD.

The mechanism of k-nearest neighbor is that we use a train set and apply the set that we are trying to classify to the train set and find the k nearest neighbors in the train set. Then the next step is to assign the point to class which is the counters for the majority of the k neighbors of that point. The purpose for training the data is that we have no idea about which k value will generate the best answer. By performing the training on the validation set and evaluation the accuracies, we can find the k value with the highest accuracies so that we can apply this specific k value to perform k-nearest neighbor on the test set.

For the k-nearest neighbor, the running results of my program is stored in three text files. Specifically, the command lines for my program is *Path_For_TrainSet Path_For_ValidationSet Path_For_TestSet output_file.txt* with spaces as the separators. In my trails of this program, I use the names output_file1.txt, output_file2.txt and output_file3.txt as the output file names for the corresponding inputs from rep1, rep2 and rep3. The details of the execution results of my program can be checked in these three files contained in the directory. The general output results is summarized as follows:

| | K value for highest accuracy | Value for accuracy |
|---|---|---|
| Rep1 | 3 | 0.9613077384523095 |
| Rep2 | 5 | 0.9619076184763048 |
| Rep3 | 3 | 0.9625074985002999 |

As can be shown above, the highest value of K for each subset is shown above. The k values are very similar, they are about 3. The accuracy results for these three representations are still very similar and very high. The execution time for these three results are very fast compared with the execution times using the ridge regression.

The mechanism of the ridge regression is that we apply a regression model to assign wight to each dimension so that the generated coefficient weight vectors can assign the object to the class with the minimum error. In order to prevent overfitting, the formula of the objective function is

$$\underset{w}{\text{minimize}}\ f(w) = \underset{w}{\text{minimize}}\ (||\mathbf{X}w - y||^2 + \lambda||w||^2),$$

The lambda is a user supplied variable which serves as the offset or penalty factor for the overfitting. Vector w is initialize by randomly assign numbers to it. The for each iteration of the program, the ith entry of the vector w will be updated by taking partial derivative and assign 0 to the partial derivative. Then after one iteration which updates the whole values of the vector w, objective functions recalculated to see if the new result generated by the previous iteration can or cannot

satisfy certain threshold so that the program can be terminated and the ideal vector w can be generated. The detail formula of set derivative to zero can be check in the instruction file and the final result for setting partial derivative to 0 is shown as follows:

$$w_i^{new} = \frac{\mathbf{X}_i^T(y - \mathbf{X}_{-i}w_{-i})}{(\mathbf{X}_i^T\mathbf{X}_i + \lambda)}$$

The regression java program will use this formula to update a certain entry of vector w in each iteration.

The program will first establish lists of vector w corresponding to various classifiers given various lambda values based on the train set. then the validation set will be used to find the lambda value so that the whole classifier model can achieve the highest accuracy. Finally, with the lambda value we just found, we will use the set combined by train set and validation set to establish a classifier model and use this model to classify the test set. The accuracy value for each stage of my program will be show in the standard output. The detailed classification results can also be checked in three output text files which are output_file1.txt, output_file2.txt and output_file3. The weight vector w of each subset can also be checked in three wight files named as weight_file1.txt, weight_file2.txt and weight_file3.txt. The command line for this program therefore will be *Path_For_TrainSet Path_For_ValidationSet Path_For_TestSet output_file.txt weight_file.txt.*

For more specific implementations of the program, notice that the only changing variable in the above formula is the vector with the ith entry of w be excluded. For the runtime of my program, I first calculates Xi^T * y and store the results in an array so that the program will not perform unnecessary computations. Similarly, the computational results of (Xi^T * Xi + lambda) and Xi^T * X-i are also stored in proper data structures so that no more extra computation will take place in each loop.

The details of the execution results are stored in the output files of my trails. I attach the general results of the program for three subsets in the following table:

| | value of lambda | Accuracy for test set | Accuracy for validation set |
|---|---|---|---|
| Rep1 | 5.0 | 0.828134373125375 | 0.8079807980798079 |
| Rep2 | 0.01 | 0.8215356928614277 | 0.8223822382238224 |
| Rep3 | 0.01 | 0.9553089382123575 | 0.9513951395139514 |

As shown above, the lambda value for the first represent which is the one without any SVD operations has relatively high lambda value and the lambda value for the two subsets with SVD operations are relatively small. the accuracy value for the first two subsets are relative low compared with the third subset and are around 80%. However, the performance for the last subsets is the best with accuracy around 95%.

The last part of this project is the non-negative. The general idea for this project is very similar to the ridge regression. But the difference is that the class label this time is either 1 or 0. Therefore in this project, there will be no negative value. We will generate the vector w in the same manner as the one generated by the ridge regression. The a python program is applied to visualize the images. The details of the images are listed in the project directory.  But it is easy to detect from the images generated from the python program, that there exits certain areas which have very distinct colors than the rest of the image.