ENEE439M Machine Learning Project

Shao-Hung, Lee
UID: 11509873

In this project, I have implemented several techniques to process the data and I would like to discuss their performance in the report. In the first section, I will introduce the theorem I use behind my program. After that, I will show the several experiment results conducted by my programs.

1 Algorithm and the Implementation Process

1.1 Main Algorithm: Baye's Classifier and KNN

Baye's Classifier I uses the the formulas of quadratic machine instead of the linear machine to get the better accuracy of the results.

$$W_i = (-1/2)\Sigma^{-1}$$
$$\underline{w_i} = \Sigma^{-1}\underline{ui}$$
$$w_{i0} = (-1/2)\underline{u_i}^T\Sigma^{-1} * \underline{u_i} - (1/2)\ln(\mid \Sigma_i \mid) + \ln(P(W_i))$$

Therefore, I can form the below classify equation for each class. For example, I can get 200 equations form "data.mat" because it is composed of 200 classes.

$$g_i(\underline{x}) = \underline{x}^T W_i \underline{x} + \underline{w_i}^T \underline{x} + w_{i0}$$

KNN KNN stands for Kth Nearest Neighbor and it is a non-parametric classifier. I followed the several steps to achieve my algorithm.

(i) Gather n labeled samples $x_1, x_2, ...x_n$

(ii) For a new sample x, find the set of $x_1, x_2, ...x_n$ which is kth closet to x

(iii) Assign x to the label of nearest set.

When I implement KNN, I sort the distance of the test point and the training points under each class. If there are two classes have the same number of samples included, I will category the test point the class which the nearest point belongs to.

1.2 Techniques to Preprocess Data

MLE Maximum likelihood estimation is applied on the training data before the training data enters the process of Baye's classification. I follow the equation below. For each class w_i:

$$\underline{u_i} = 1/(n_i) \sum_{\underline{x_i} in D_i} \underline{x}$$

$$\Sigma_i = 1/(n_i) \sum_{\underline{x_i} in D_i} (\underline{x} - u_i)(\underline{x} - u_i)^T$$

After obtaining the parameters form the above equations, I can apply them into the formulas of Baye's classifier.

PCA Principal component analysis is applied to transform the data form origin n dimension to the designer's required dimension. i follow the several steps to complete the technique. Note: The eigenvalue problem can be easily resolved by calling the function in Matlab.

(i) Center the Data. $\underline{x_k} = \underline{x_k} - (1/n) \sum_{k=1}^{n} \underline{x_k}$

(ii) Form $\hat{C}$. $\hat{C} = (1/n) * \sum_{k=1}^{n} \underline{x_k} \underline{x_k}^T$

(iii) Solve $\mid \hat{C} - \lambda I \mid = 0$. Can get $\lambda_1, ...\lambda_d$

(iv) For each λ_i, find $\underline{v_i}$ by solving $(\hat{C} - \lambda_i I)\underline{v_i} = 0$ (Eigenvalue Problem)

(v) Finally, choose m eigenvectors $\underline{v_1}...\underline{v_m}$ corresponding to the m highest eigenvalues of $\hat{C}$

LDA Linear discriminant analysis has almost the same motivation of PCA. However, the theorem behind it is quite different form PCA.

	MLEBayes	PCABayes	LDABayes	KNN	PCAKNN	LDAKNN
data.mat	0.64	0.635	0.65	0.595	0.585	0.655
pose.mat	0.529	0.537	0.45	0.235	0.235	0.103
illumination.mat	0.459	0.49	0.39	1	1	1

Table 1: Bayes Classifier and KNN Method

2 General Performance

The above results is input by the three datasets. For "data.mat", I use $face(:,:,3*n-2)$ and $face(:,:,3*n-1)$ to be the training data. Of course, $face(:,:,3*n)$ would become my test data. For "pose.mat", I use the first 11 samples each class to train. For "illumination.mat", I use the first 16 samples each class to train. In addition, the PCA processes to half the original data dimension and the LDA processes to the class number - 1 dimension. I apply K = 1 at KNN method here. (Note: PCABayes and LDABayes are also done with the MLE techniques.)

Because I have implemented the programs and run all the three input data. I just want to show that under the same condition, the difference of accuracy between KNN and Baye's classifier.

Observing form the data, I observe that Baye's classifier performances better in much better when the data set is rather small. When the data set becomes larger, the nearest neighbor gets better performance. In addition, it shows that applying PCA/LDA on the original algorithm, the result can maintain the almost the same. In addition, when I run the program, obviously, I can feel it is much faster if applying the techniques of PCA/LDA.

3 Discussion

3.1 Accuracy of Different Combination

In this section, I will choose different combinations of training data and test data for test. I want to show how the accuracy will change due to the difference between categorization of the training and testing. I choose "Data.mat" to be my input data for this experiment because it only contains three samples for each class. In addition, I uses PCA Baye's classifier to summarize the results.

test data: 3n	test data: 3n-1	test data: 3n-2
0.635	0.665	0.71

3.2 Dimension Changes of PCA

I am just curious about the how the lower dimension change the accuracy. I use the PCAKNN to testify it. I observe that there is little change when the dimension is big enough but still smaller than the original data.

Dimension	1000	1300	1600	1900
pose.mat	0.235	0.235	0.235	0.235
illumination.mat	1	1	1	1

3.3 Accuracy of KNN under K Varies

I will show the change of accuracy with respect to different K. I use the PCAKNN to testify it. I observe form the results that as K increases, the accuracy will decrease.

	K = 1	K = 3	K = 5	K = 7
data.mat	0.585	0.49	0.43	0.4
pose.mat	0.235	0.1912	0.2059	0.2059
illumination.mat	1	0.6176	0.4853	0.4412

3.4 Some Implementation Problems

When I implement the method of Baye's classifier, PCA and LDA, I usually encounter the problem of soling the singular matrix. Therefore, I apply the trick that I add the modified value on the diagonal element of the matrix. This is the actual problem which can not know if you don't implement once by yourself. I think I learned a lot from ding this project.