Cmpe 462 - HW2

Şahin Batmaz
2012400117

REPORT

There are 4 files submitted; 0-prepare.m, 1-functions.m, 2_1-gaussians.m, 2_2-knn.m. To run ‘gaussian’ and ‘knn’ scripts, ‘prepare’ and ‘functions’ scripts must be run first.

1. **0-prepare.m**

In this script, points2d.dat file is read. Each class is extracted into separate vectors. From these vectors; training, validation and test datasets are randomly constructed. Then each of these datasets;training,validation and test datasets; are combined 3-dimensional array. With this script, data is prepared to be used.

2. **1-functions.m**

In this script, there are several functions implemented.

EM(k,mydata): It is the expectation maximization algorithm. It takes two arguments. ‘k’ is the number of gaussians that will be fit on the data. ‘mydata’ is the training dataset which contains two dimensional points. To run the function; mean,covariance and size values are initialized. I used built-in ‘kmeans()’ function for the initial mean(s). After that, I calculated covariance estimate for these means. Size values are initialized as ‘1/k’. Then I applied expectation-maximization formulas which are on the lecture pdf. The EM algorithm is put into a for loop of 20 turns; because in the lecture pdf, it is stated that around 15 turns, the parameters are converges to highest likelihood level. In the end, the function returns mean, covariance and size values.

mgLikelihood(xi,mvec,covar): This function calculates the multivariate normal probability density function for the given point xi with the given parameters mvec(mean vectors) and covar(covariance matrix) of the gaussian.

covarianceEstimate(mydata,mvec): This function calculates the covariance matrix estimate for the given mvec(mean vectors) on the mydata(training dataset).

distListOfPointsToPoint(mydata, mypoint): This function calculates the euclidian distance of two dimensional ‘mypoint’ point to each point in ‘mydata’ dataset. In the output, there is a distance value for each point in the ‘mydata’ dataset.

knnVoteResult(cList,k,mydata,mypoint): In this function, mydata is the union of all classes’ training sets. cList is the fixed vector [1,2,3] for the classes. k is the number which states how many nearest neighbours should be checked. Lastly, mypoint is the two dimensional point that will be classified. The function find distance value for each point in the dataset to mypoint point. The function takes the closest k points. Then the function counts the classes of these k points. The class that occurred the most is the estimated class of the mypoint point. In short, this function returns the estimated class of the mypoint point.

3. **2_1-gaussians.m**

In this script, as a first task, all fit cases are calculated. There are 3 classes and for each class there are 3 fitting options; 1,2 or 3 gaussians. Therefore, there are 9 fit cases that are calculated. All of their results are saved into matrices. For calculating the fits, EM function is used.

Then the best model is obtained. In the best model, there are 3 values for 3 classes. Values are for how many gaussians should be fit for that class. Since there are 3 fitting options for each class, there are 27 combinations. For each of them, the error count is calculated for the all validation sets of 3 classes. The configuration that has the lowest error gives the best model.

Then for the best model, results are calculated. The best model is applied with test datasets of classes separately. In each class’s dataset, the estimated classes are counted separately. Which means that, there are 3 test datasets from 3 classes and for each test dataset, there are 3 class estimations. Therefore, a 3x3 matrix is calculated. This is the confusion matrix. From this matrix, the prediction error is calculated since the wrong estimations are accessible.

In the end of script, there is a plot function in comments. If you uncomment it, you can see where the gaussians that are fit in the best model are located.

4. **2_2-knn.m**

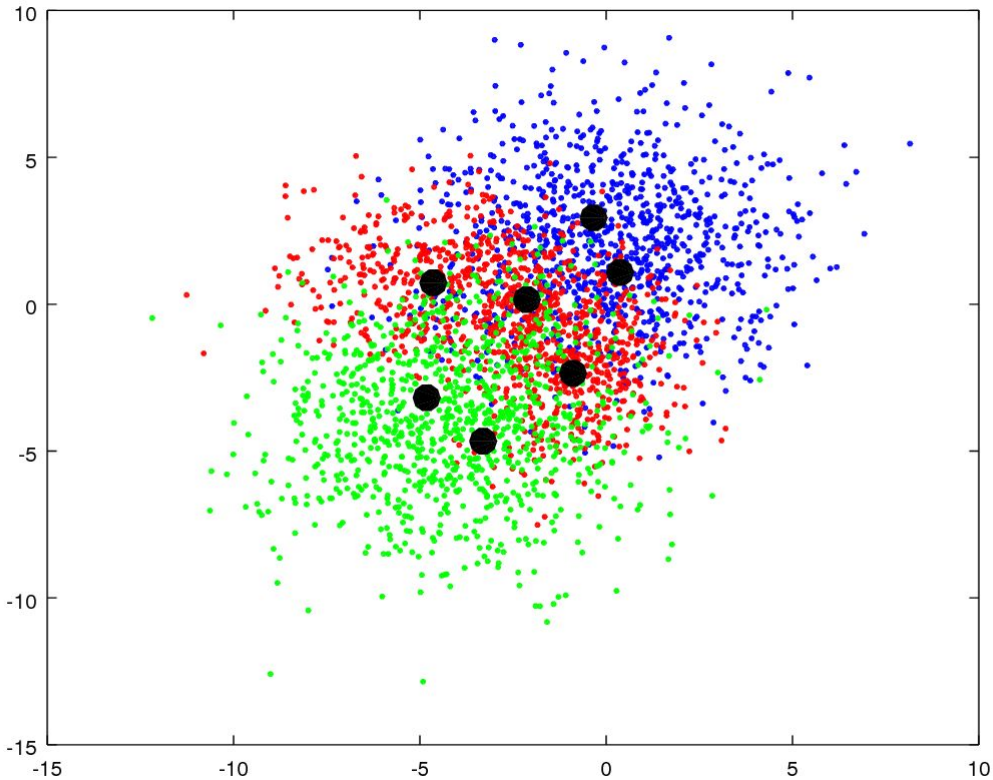
In this script, knn technique is applied. For each k option, the system is tested. For each point in the validation datasets of all classes, a classification is made by applying the knn method on the points of training datasets of all classes. For each k option, the errors are counted. The one with the lowest error count becomes the best model.

Then, for each k option, test datasets are applied separately as in the gaussians script. For each test dataset, the estimated classes are counted. Therefore, a 3x3 matrix is obtained for each k option. This step is the same procedure as it is done in gaussians script. Each matrix is the confusion matrix for the k option they are coming from. From these matrices, wrong estimations are obtained and for each k option, prediction error is calculated.

RESULTS OF MIXTURE OF GAUSSIANS

Best model

Number of gaussians for class 0 => 2
Number of gaussians for class 1 => 3
Number of gaussians for class 2 => 2



Best model prediction error

24.6667 %

Best model confusion matrix

	Predicted class no --> 0	1	2
Class 0 test set :	367	124	9
Class 1 test set :	55	387	58
Class 2 test set :	7	117	376

RESULTS OF KNN

Best model

---

k Nearest Neighbour size = 40

Prediction errors for each k option; 1,10,40

---

k Nearest Neighbour size = 1 --> 38.4667 %
k Nearest Neighbour size = 10 --> 26.8667 %
k Nearest Neighbour size = 40 --> 25.0667 %

Confusion matrix for each k option; 1,10,40

---

k Nearest Neighbour size = 1

	Predicted class no --> 0	1	2
Class 0 test set :	314	134	52
Class 1 test set :	111	279	110
Class 2 test set :	51	119	330

---

k Nearest Neighbour size = 10

	Predicted class no --> 0	1	2
Class 0 test set :	373	113	14
Class 1 test set :	76	358	66
Class 2 test set :	14	120	366

---

k Nearest Neighbour size = 40

	Predicted class no --> 0	1	2
Class 0 test set :	369	122	9
Class 1 test set :	59	383	58
Class 2 test set :	6	122	372
