Data / Information Fusion

1399-1400-2

Assignment #5

Due date: Khordad 28, 1400

In this take-home exercise, you will design and implement a multi-classifier classification system that employs five classifiers as follows:

- Decision tree with your selected parameters
- SVM with your selected kernel and parameters
- KNN, with 2 selected K values of your choice.
- Naïve Bayes

a) Generate two data sets *X* and *Y*, each consisting of 500 data points in the 100-dimensional feature space. Classes are equiprobable, which follow Gaussian distributions with arbitrary means m_1 and m_2 such that the Euclidean distance between them (*d*), can be variable. The covariance matrices are $S_X = 0.2I$, and $S_Y = 0.4I$ where *I* is the 100 × 100 identity matrix.

b) Select 70% of data in X_1 and Y_1 as training set and remaining in X_2 and Y_2 as test set.

c) Investigate the *accuracy* and *precision* of each individual classifier on test data with relevance to the *distance* between class means (*d*). Adjust the classifiers parameters to get the best performance you can, and draw the *accuracy* vs. *d* and *precision* vs. *d* curves for each individual classifier.

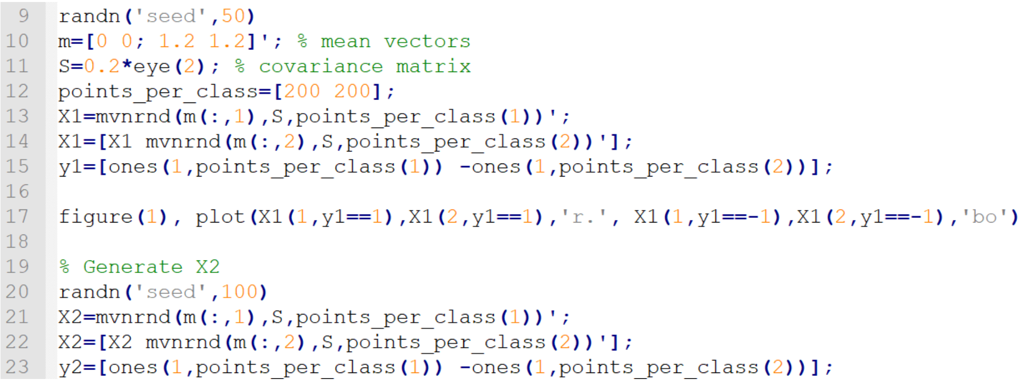
d) Repeat parts **b** and **c** 100 times, each time randomly split data into train and test datasets and show the average accuracy curve and its standard deviation.

e) Now, try to integrate classifier outputs in a five-classifier fusion system. Repeat parts **b** and **c,** and **d**, but this time, instead of single classifiers, use simple majority voting, decision template, and OWA, to combine the results and obtain the output of fusion system. Discuss the results and the effect of fusion system on the classification performance in comparison with individual classifiers.

f) Investigate the role of feature space dimensionality in the fusion system performance. Change the feature space dimensionality from 1 to 100. For each round, fix *d*, such that the best test accuracy for individual classifiers be 70%. Then draw the *accuracy* vs. *dimensionality* plot for the multi-classifier system and discuss about it.

Comments:

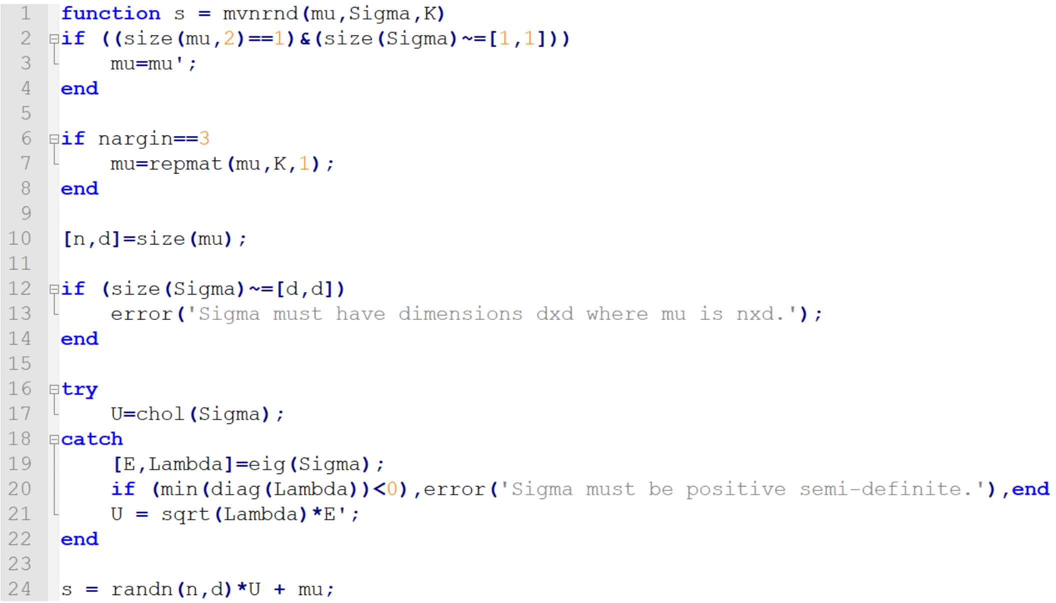
- You can implement your code in Matlab, Python, R, or any other programming language.
- You may use scikit-learn package as well.
- In Matlab, suppose in a 2-dimensional space, we are given two equiprobable classes, which follow Gaussian distributions with means $m1 = [0, 0]T$ and $m2 = [1.2, 1.2]T$ and covariance matrices $S1 = S2 = 0.2I$, where I is the 2 × 2 identity matrix. To generate 200 datapoints, we can use the following code:



```
randn('seed',50)
m=[0 0; 1.2 1.2]'; % mean vectors
S=0.2*eye(2); % covariance matrix
points_per_class=[200 200];
X1=mvnrnd(m(:,1),S,points_per_class(1))';
X1=[X1 mvnrnd(m(:,2),S,points_per_class(2))'];
y1=[ones(1,points_per_class(1)) -ones(1,points_per_class(2))];

figure(1), plot(X1(1,y1==1),X1(2,y1==1),'r.', X1(1,y1==-1),X1(2,y1==-1),'bo')

% Generate X2
randn('seed',100)
X2=mvnrnd(m(:,1),S,points_per_class(1))';
X2=[X2 mvnrnd(m(:,2),S,points_per_class(2))'];
y2=[ones(1,points_per_class(1)) -ones(1,points_per_class(2))];
```



```
function s = mvnrnd(mu,Sigma,K)
if ((size(mu,2)==1)&(size(Sigma)~=[1,1]))
    mu=mu';
end

if nargin==3
    mu=repmat(mu,K,1);
end

[n,d]=size(mu);

if (size(Sigma)~=[d,d])
    error('Sigma must have dimensions dxd where mu is nxd.');
end

try
    U=chol(Sigma);
catch
    [E,Lambda]=eig(Sigma);
    if (min(diag(Lambda))<0),error('Sigma must be positive semi-definite.'),end
    U = sqrt(Lambda)*E';
end

s = randn(n,d)*U + mu;
```

Good luck!