**Matlab Tutorial 2: Exercises for Advanced Data Mining**

**Exercise 1:** Generate *N* = 500 2-dimensional data points that are distributed according to the Gaussian distribution *N(m*,*S)*, with mean *m* = [0, 0]*T* and covariance matrix

*S* =[*σ*11 *σ*12;*σ*21 *σ*22] for the following cases:

1. *σ*11=*σ*22=1, *σ*12 = *σ*21=0
2. *σ*11=*σ*22=0.2, *σ*12 = *σ*21=0
3. *σ*11=*σ*22=2, *σ*12 = *σ*21=0
4. *σ*11=0.2, *σ*22=2, *σ*12 = *σ*21=0
5. *σ*11=2, *σ*22=0.2, *σ*12 = *σ*21=0
6. *σ*11=*σ*22=1, *σ*12 = *σ*21=0.5
7. *σ*11=0.3, *σ*22=2, *σ*12 = *σ*21=0.5
8. *σ*11=0.3, *σ*22=2, *σ*12 = *σ*21=-0.5

Plot each data set and comment on the shape of the clusters formed by the data points.

***Solution: Matlab code for part a)***

mu1 = [0 0]; Sigma1 = [1 0; 0 1];

r1 = mvnrnd(mu1, Sigma1, 500);

figure(1), plot(r1(:,1),r1(:,2),'.');

figure(1), axis equal

figure (1), axis([-7 7 -7 7])

**Example 2:** Assume that a feature follows Gaussian distributions in both classes of a 2-class

classification problem. The respective mean values are *m*1 = 8.75 and *m*2 = 9; their common variance is *σ*2 = 4.

**1.** Generate the vectors *x*1 and *x*2, each containing *N* = 1000 samples from the first and the second distribution, respectively.

**2.** Pretend that the means *m*1 and *m*2, as well as the variance *σ*2, are unknown. Assumed to be known are the vectors *x*1 and *x*2 and the fact that they come from distributions with equal (yet unknown) variance. Use the *t*-test to check whether the mean values of the two distributions differ significantly, using as significance level the value *ρ* = 0.05.

Repeat this procedure for *ρ* = 0.001 and draw conclusions.

***Solution: Matlab code***

%To generate the vectors x1 and x2

randn('seed',0)

m1=8.75;

m2=9;

stdevi=sqrt(4);

N=1000;

x1=m1+stdevi\*randn(1,N);

x2=m2+stdevi\*randn(1,N);

%Apply the t-test using the MATLAB ttest2 function

rho=0.05;

[h] = ttest2(x1,x2,rho)

where

*h* = 0 (corresponding to the null hypothesis *H*0) indicates that there is no evidence, at the *ρ*

significance level, that the mean values are not equal

*h* = 1 (corresponding to the alternative hypothesis *H*1) indicates that the hypothesis that the

means are equal can be rejected, at the *ρ* significance level.

If the latter case is the outcome, the feature is selected; otherwise, it is rejected. In our case, the

result is *h* = 1, which implies that the hypothesis of the equality of the means can be rejected at the 5% significance level. The feature is thus selected.

***Remark***

* The *t*-test assumes that the values of the features are drawn from normal distributions. However, in real applications this is not always the case. Thus, each feature distribution should be tested for *normality* prior to applying the *t*-test. Normality tests may be of the *Lilliefors* or the *Kolmogorov-Smirnov* type, for which MATLAB functions are provided (*lillietest* and *kstest*, respectively).
* If the feature distributions turn out not to be normal, one should choose a nonparametric statistical significance test, such as the *Wilcoxon* rank sum test, using the *ranksum* MATLAB function.

**Exercise 3:** Draw the ROC curve for a classifier and dataset of your choice.

***Solution:*** *The Matlab code here is illustrated for a forward neural network with ten hidden layers and applied to the Iris data.*

%load the iris data with (t being the class label represented as %a vector with elements equal to zero everywhere except for the %class that belongs to where the value is 1)

[x,t] = iris\_dataset;

%create a neural network with 10 hidden layer

net = patternnet(10);

%apply the neural network to the data

net = train(net,x,t);

view(net)

%predict the class label using the neural network net

y = net(x); %each y is a vector with all zeros except the %element on the position for which the class is 1

%check the performance of the classifier

perf = perform(net,t,y);

%classes = vec2ind(y);

%calculate the true positives, false positives for different %thresholds

[tpr,fpr,th] = roc(t,y)

%plot the ROC curve

plotroc(t,y)

**Exercise 4:** Example of ensemble of classifiers for imbalanced data (based on the Mathworks example at <http://www.mathworks.com/help/stats/ensemble-methods.html#br0g6t1-1>)

This example shows how to classify when one class has many more observations than another. We will experiment with the Random Under Sampling) RUSBoost algorithm, because it is designed to handle this case.

We use the "Cover type" data from the UCI machine learning archive, described in <http://archive.ics.uci.edu/ml/datasets/Covertype> . The data classifies types of forest (ground cover), based on predictors such as elevation, soil type, and distance to water. The data has over 500,000 observations and over 50 predictors, so training and using a classifier is time consuming.

Blackard and Dean [3] describe a neural net classification of this data (<http://www.mathworks.com/help/stats/ensemble-methods.html#br0g6t1-1> ). They quote a 70.6% classification accuracy. RUSBoost obtains over 76% classification accuracy; see steps 6 and 7.

* [Step 1. Obtain the data.](http://www.mathworks.com/help/stats/ensemble-methods.html#zmw57dd0e59665)
* [Step 2. Import the data and prepare it for classification.](http://www.mathworks.com/help/stats/ensemble-methods.html#zmw57dd0e59678)
* [Step 3. Examine the response data.](http://www.mathworks.com/help/stats/ensemble-methods.html#zmw57dd0e59688)
* [Step 4. Partition the data for quality assessment.](http://www.mathworks.com/help/stats/ensemble-methods.html#zmw57dd0e59703)
* [Step 5. Create the ensemble.](http://www.mathworks.com/help/stats/ensemble-methods.html#zmw57dd0e59712)
* [Step 6. Inspect the classification error.](http://www.mathworks.com/help/stats/ensemble-methods.html#zmw57dd0e59730)
* [Step 7. Compact the ensemble.](http://www.mathworks.com/help/stats/ensemble-methods.html#zmw57dd0e59754)

**Step 1. Obtain the data.**

urlwrite('http://archive.ics.uci.edu/ml/machine-learning-databases/covtype/covtype.data.gz','forestcover.gz');

Then, extract the data from the forestcover.gz file. The data is in the covtype.data file.

**Step 2. Import the data and prepare it for classification.**

Import the data into your workspace. Extract the last data column into a variable named Y.

load covtype.data

Y = covtype(:,end);

covtype(:,end) = [];

**Step 3. Examine the response data.**

tabulate(Y)

Value Count Percent

1 211840 36.46%

2 283301 48.76%

3 35754 6.15%

4 2747 0.47%

5 9493 1.63%

6 17367 2.99%

7 20510 3.53%

There are hundreds of thousands of data points. Those of class 4 are less than 0.5% of the total. This imbalance indicates that RUSBoost is an appropriate algorithm.

**Step 4. Partition the data for quality assessment.**

Use half the data to fit a classifier, and half to examine the quality of the resulting classifier.

part = cvpartition(Y,'holdout',0.5);

istrain = training(part); % data for fitting

istest = test(part); % data for quality assessment

tabulate(Y(istrain))

Value Count Percent

1 105920 36.46%

2 141651 48.76%

3 17877 6.15%

4 1374 0.47%

5 4746 1.63%

6 8683 2.99%

7 10255 3.53%

**Step 5. Create the ensemble.**

Use deep trees for higher ensemble accuracy. To do so, set the trees to have minimal leaf size of 5. Set LearnRate to 0.1 in order to achieve higher accuracy as well. The data is large, and, with deep trees, creating the ensemble is time consuming.

t = templateTree('MinLeafSize',5);

tic

rusTree = fitensemble(covtype(istrain,:),Y(istrain),'RUSBoost',1000,t,...

'LearnRate',0.1,'nprint',100);

toc

Training RUSBoost...

Grown weak learners: 100

Grown weak learners: 200

Grown weak learners: 300

Grown weak learners: 400

Grown weak learners: 500

Grown weak learners: 600

Grown weak learners: 700

Grown weak learners: 800

Grown weak learners: 900

Grown weak learners: 1000

Elapsed time is 918.258401 seconds.

**Step 6. Inspect the classification error.**

Plot the classification error against the number of members in the ensemble.

figure;

tic

plot(loss(rusTree,covtype(istest,:),Y(istest),'mode','cumulative'));

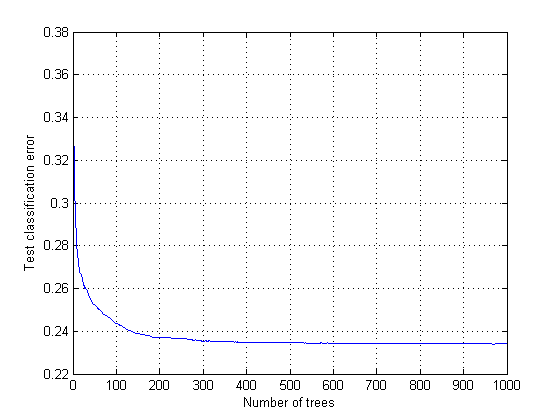
toc

grid on;

xlabel('Number of trees');

ylabel('Test classification error');

Elapsed time is 775.646935 seconds.



The ensemble achieves a classification error of under 24% using 150 or more trees. It achieves the lowest error for 400 or more trees.

Examine the confusion matrix for each class as a percentage of the true class.

tic

Yfit = predict(rusTree,covtype(istest,:));

toc

tab = tabulate(Y(istest));

bsxfun(@rdivide,confusionmat(Y(istest),Yfit),tab(:,2))\*100

Elapsed time is 427.293168 seconds.

ans =

Columns 1 through 6

83.3771 7.4056 0.0736 0 1.7051 0.2681

18.3156 66.4652 2.1193 0.0162 9.3435 2.8239

0 0.0839 90.8038 2.3885 0.6545 6.0693

0 0 2.4763 95.8485 0 1.6752

0 0.2739 0.6530 0 98.6518 0.4213

0 0.1036 3.8346 1.1400 0.4030 94.5187

0.2340 0 0 0 0.0195 0

Column 7

7.1705

0.9163

0

0

0

0

99.7465

All classes except class 2 have over 80% classification accuracy, and classes 3 through 7 have over 90% accuracy. But class 2 makes up close to half the data, so the overall accuracy is not that high.

**Step 7. Compact the ensemble.**

The ensemble is large. Remove the data using the [compact](http://www.mathworks.com/help/stats/classificationensemble.compact.html) method.

cmpctRus = compact(rusTree);

sz(1) = whos('rusTree');

sz(2) = whos('cmpctRus');

[sz(1).bytes sz(2).bytes]

ans =

1.0e+09 \*

1.6947 0.9790

The compacted ensemble is about half the size of the original.

Remove half the trees from cmpctRus. This action is likely to have minimal effect on the predictive performance, based on the observation that 400 out of 1000 trees give nearly optimal accuracy.

cmpctRus = removeLearners(cmpctRus,[500:1000]);

sz(3) = whos('cmpctRus');

sz(3).bytes

ans =

475495669

The reduced compact ensemble takes about a quarter the memory of the full ensemble. Its overall loss rate is under 24%:

L = loss(cmpctRus,covtype(istest,:),Y(istest))

L =

0.2326