Lecture 7

```
>> rng('default') % For reproducibility
X = mvnrnd([0;0],[1 .9;.9 1],1000);

>> mean(X)

ans =

  -0.0326   -0.0133

>> cov(X)

ans =

   0.9979    0.8941
   0.8941    0.9905

>> Y = [1 1;1 -1;-1 1;-1 -1];
>> Y

Y =

    1     1
    1    -1
   -1     1
   -1    -1

>> d2_mahal = mahal(Y,X)

d2_mahal =

   1.1095
  20.3632
  19.5939
   1.0137

>> Z = [0,0];
>> mahal(Z,X)

ans =

   0.0024

>> smean = mean(X);
>> scov = cov(X);
```

```
>> pdist2(Z, smean, 'mah', scov)

ans =

   0.0491

>> 0.0491^2

ans =

   0.0024
```

http://www.ece.uah.edu/~dwpan/course/ee610/code/KNN/knn_demo.m

```
>> N = 10;     % Number of samples per class
% X is the data array, C is the class label vector
[X, C] = dataset_generate(N);
% Plot the data samples of the two classes
figure;
gscatter(X(:,1),X(:,2), C, 'rb','ox');
axis square
axis equal

K = 5;  % Look at the K nearest neighbors
>>
>> L = length(X);

Xin = mean(X)      % An example input query vector

Xin =

   1.2486    1.7158

>> % Calculate the Euclidean distances
Dist = zeros(1, L);
for i = 1:L
   Dist(i) = norm(X(i,:) - Xin);
end
>> % Sort the distances
[D_sorted, Index] = sort(Dist);
>> doc sort
>> D_sorted(1:5)

ans =

   1.6981    2.4970    3.0014    3.6606    3.7609

>> Index(1:5)

ans =

    9   10   17    2   15

>> % Look at the first K classes
Classes = C(Index(1:K));
>> whos C
  Name      Size            Bytes  Class
Attributes

  C        20x1              160  double

>> Classes

Classes =

    1
    1
    2
    1
    2
```

```
>> % Using Matlab toolbox
Model = fitcknn(X, C, 'NumNeighbors',K);
>> Model.Prior

ans =

   0.5000    0.5000
```

Data Standardization (Normalization)

```
>> Z = random('exp',2,1,1000000);
>> whos Z
  Name      Size              Bytes  Class     Attributes

  Z         1x1000000         8000000  double

>> figure; histogram(Z)
>> mean(Z)

ans =

   1.9970

>> std(Z)

ans =

   1.9960

>> ZN = (Z - mean(Z))/std(Z);
>> mean(ZN)

ans =

  -7.2163e-17

>> std(ZN)

ans =

     1

>> figure; histogram(ZN)
```

Normalized the training dataset, as well as the input data point

```
>> % Alternatively, use the function
X_std = normalize(X);

>> X_std_1 = zeros(L, 2);
X_std = zeros(L, 2);

Mu = zeros(1,2);
Std = zeros(1,2);
>>
>> for i = 1:2
   Mu(i) = mean(X(:,i));
   Std(i) = std(X(:,i));
   X_std_1(:,i) = (X(:,i) - Mu(i))/Std(i);
end
>> % Normalize the Xin
for i = 1:2
   Xin_std(:,i) = (Xin(:,i) - Mu(i))/Std(i);
end
```

```
>> % Repeat the KNN algorithm

% Calculate the Euclidean distances
Dist_std = zeros(1, L);
for i = 1:L
   Dist_std(i) = norm(X_std(i,:) - Xin_std);
end

% Sort the distances
[D_sorted_std, Index_std] = sort(Dist_std);
% First 5 values of the index vector are different
from above
%Index_std(1:5)
%ans =
%    9   10   15    2   17

% Look at the first K classes
Classes_std = C(Index_std(1:K));

% The predicted label for the input vector Xin
(majority vote)
label_std = mode(Classes_std)

label_std =

    1
```

```
>> % Using Matlab toolbox
Model_std = fitcknn(X, C, 'NumNeighbors',K, 'Standardize', 1);
>> Mu

Mu =

   1.2486    1.7158

>> Model_std.Mu

ans =

   1.2486    1.7158

>> Std

Std =

   6.1113    3.5793

>> Model_std.Sigma

ans =

   6.1113    3.5793

>> [label,score,cost] = predict(Model_std,Xin)

label =

     1


score =

   0.6000    0.4000


cost =

   0.4000    0.6000
```

```
import numpy as np
# Need to change the path name for the infile below
infile = "mvnrnd.csv"
dataset = np.loadtxt(infile, delimiter=',')
X = dataset[:, 0:2]
y = dataset[:,2]

from sklearn import preprocessing
scaler = preprocessing.StandardScaler().fit(X)

# Mean
scaler.mean_
# Standard deviation
scaler.scale_
X.std(axis = 0)

X_scaled = scaler.transform(X)
X_scaled.mean(axis = 0)
X_scaled.var(axis = 0)

from sklearn.neighbors import KNeighborsClassifier
neigh = KNeighborsClassifier(n_neighbors=5)
neigh.fit(X_scaled, y)

# Apply the same scaling on the test data Xin
Xin = X.mean(axis = 0)
Xin_scaled = scaler.transform([Xin])

neigh.predict(Xin_scaled)
neigh.predict_proba(Xin_scaled)
```

```
scaler.mean_
Out[21]: array([1.24856439, 1.71580852])

scaler.scale_
Out[23]: array([5.95652587, 3.48863232])

X.std(axis = 0)
Out[24]: array([5.95652587, 3.48863232])

X_scaled = scaler.transform(X)

X_scaled.mean(axis = 0)
Out[26]: array([ 1.11022302e-17, -1.11022302e-17])

X_scaled.var(axis = 0)
Out[27]: array([1., 1.])


Xin = X.mean(axis = 0)

Xin_scaled = scaler.transform([Xin])

neigh.predict(Xin_scaled)
Out[34]: array([1.])

neigh.predict_proba(Xin_scaled)
Out[35]: array([[0.6, 0.4]])
```

Pipeline in sklearn

```
# Alternatively, use pipeline

from sklearn.pipeline import make_pipeline
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler


pipe = make_pipeline(StandardScaler(), KNeighborsClassifier(n_neighbors=5))

pipe.fit(X, y)
pipe.predict([Xin])
pipe.predict_proba([Xin])
```

```
pipe.fit(X, y)
Out[40]:
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('kneighborsclassifier', KNeighborsClassifier())])

pipe.predict([Xin])
Out[41]: array([1.])

pipe.predict_proba([Xin])
Out[42]: array([[0.6, 0.4]])
```