# **Seyed Mahdi Basiri Azad (Erfan):** CS74/174 Homework #1

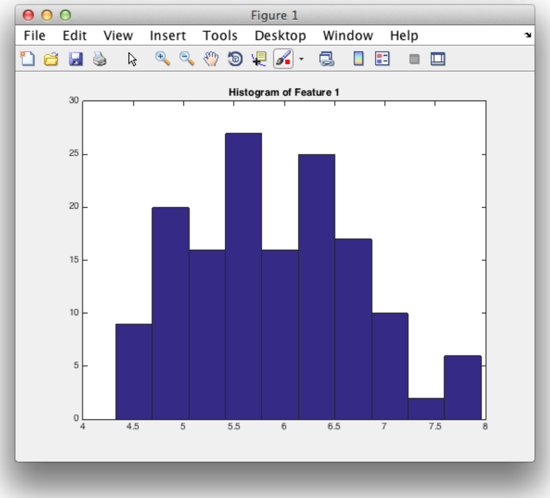
Machine Learning and Statistical Data Analysis: Winter 2016

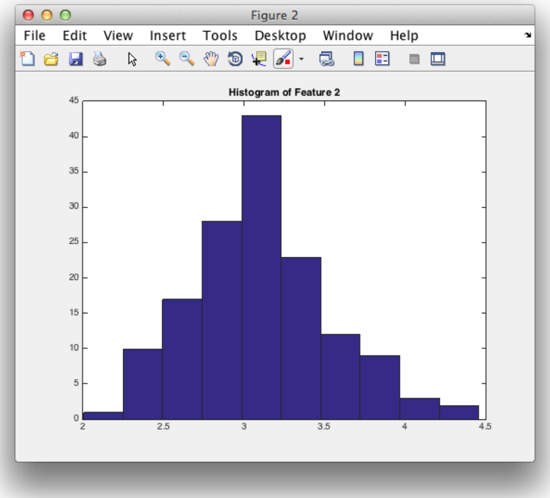
## Problem 1: Matlab & Data Exploration

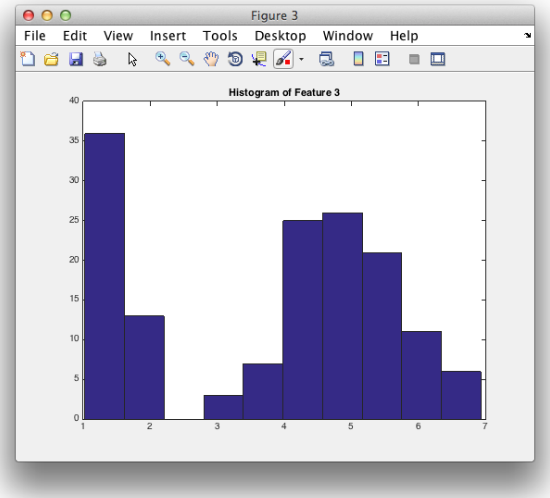
(a) Download and load the "Fisher iris" data set into Matlab (or Octave):

```
iris=load('data/iris.txt'); y=iris(:,end); X=iris(:,1:end-1);

%====1.a -- Plotting the histogram of each feature=====
hist(X(:,1));
title('Histogram of Feature 1');
figure;
hist(X(:,2));
title('Histogram of Feature 2');
figure;
hist(X(:,3));
title('Histogram of Feature 3');
figure;
hist(X(:,4));
title('Histogram of Feature 4');
%=====================================================
```







(b) computing the mean:

```
%=====1.b -- computing the mean of each feature========
means(1) = mean(X(:,1));
means(2) = mean(X(:,2));
means(3) = mean(X(:,3));
means(4) = mean(X(:,4));
%=======================================================
```

(c) Varience and stdv:

```
%=1.c -- computing the Varience and stdv of each feature=
stds(1) = std(X(:,1));
stds(2) = std(X(:,2));
stds(3) = std(X(:,3));
stds(4) = std(X(:,4));

vars(1) = var(X(:,1));
vars(2) = var(X(:,2));
vars(3) = var(X(:,3));
vars(4) = var(X(:,4));
%=======================================================
```

(d) Normalize:

```
%==================1.d -- Normalize====================
norms(:,1) = bsxfun(@rdivide, bsxfun(@minus, X(:,1), mean(X(:,1))), std(X(:,1)));
norms(:,2) = bsxfun(@rdivide, bsxfun(@minus, X(:,2), mean(X(:,2))), std(X(:,2)));
norms(:,3) = bsxfun(@rdivide, bsxfun(@minus, X(:,3), mean(X(:,3))), std(X(:,3)));
norms(:,4) = bsxfun(@rdivide, bsxfun(@minus, X(:,4), mean(X(:,4))), std(X(:,4)));
%=======================================================
```

(e) Plot features against each other:

```
%=========1.e -- Plot features against each other======
scatter(norms(:,1), norms(:,2),50, y, 'fill');
title('Normalized feature1 vs. feature2');
xlabel('feature1');
ylabel('feature2');

figure;
scatter(norms(:,1), norms(:,3),50, y, 'fill');
title('Normalized feature1 vs. feature3');
xlabel('feature1');
ylabel('feature3');

figure;
scatter(norms(:,1), norms(:,4),50, y, 'fill');
title('Normalized feature1 vs. feature4');
xlabel('feature1');
ylabel('feature4');
%====================================================
```

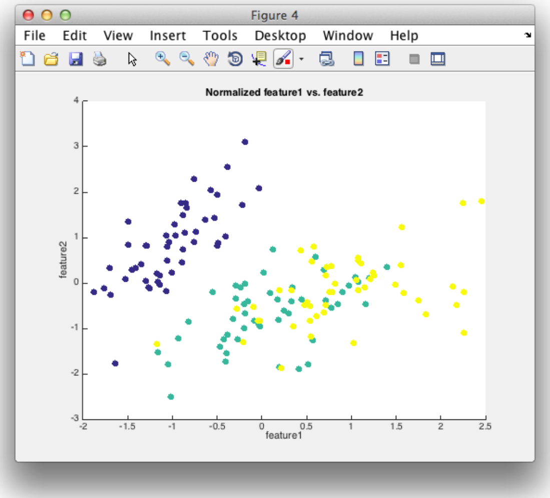


Figure 4
File Edit View Insert Tools Desktop Window Help
Normalized feature1 vs. feature2
feature2
4
3
2
1
0
-1
-2
-3
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
feature1

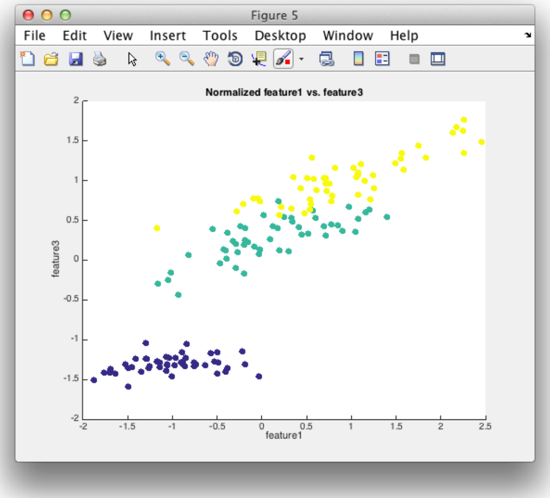
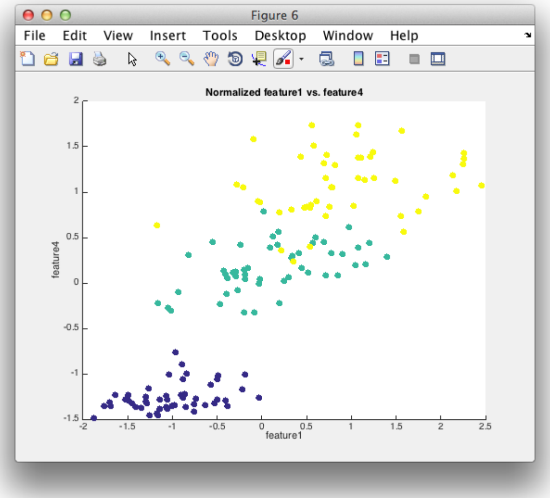


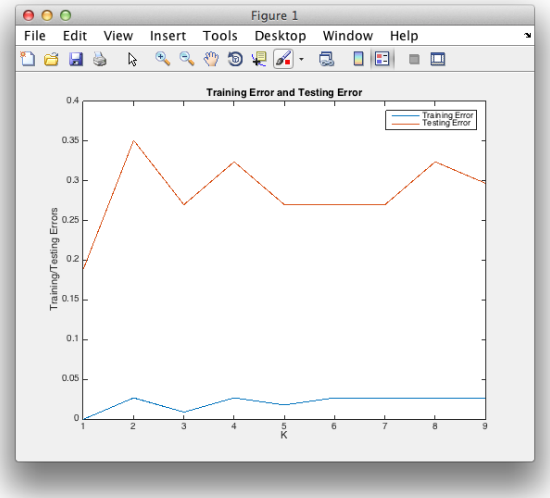
Figure 5
File Edit View Insert Tools Desktop Window Help
Normalized feature1 vs. feature3
feature3
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
feature1



## Problem 2: kNN Predictions

(a) kNN function: Optimal k ==> 6

```
function Ytest_hat = KNN(Xtest, K, Xtrain, Ytrain)
% Implement K-nearest neighborhood classification.
% Input:
%    -- Xtest: feature of test data (a n X d matrix; each row is an instance,
% each column is a feature)
%    -- K: number of nearest neighborhood points
%    -- Xtrain: feature of training data (a n X d matrix)
%    -- Ytrain: true target value of training data (a n X 1 vector)
% Output:
%    -- Ytest_hat: predicted target value on the test data
if K == 1
    dist=pairwise_distance(Xtest, Xtrain);
    [~, maxindex]  = min(dist, [], 2);
    Ytest_hat = Ytrain(maxindex);
else
   dist = pairwise_distance(Xtest, Xtrain);
   %now sort the each row of the dist and return the index of the first k
   %ones
   [~, I] = sort(dist,2); % I is a matrix. Each row is the sorted distance
   %of testing data from the training data
   K_indecies = I(:,1:K); % first K columns of I
   for col = 1:size(K_indecies,2)
       K_labels(:,col) = Ytrain(K_indecies(:,col));
   end
   Ytest_hat = mode(K_labels,2);
end
return;
end
```



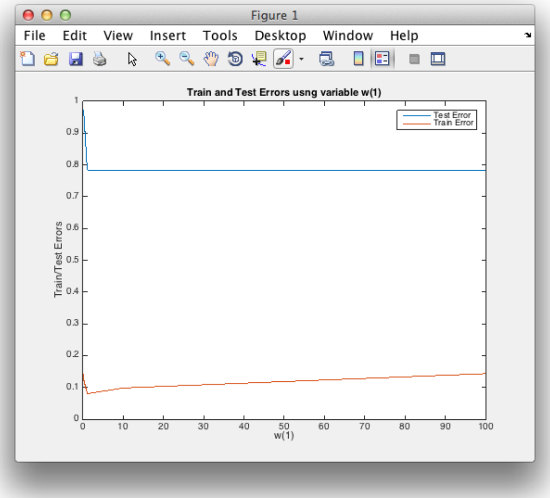
(b) Weighted kNN: Optimal w(1) ==> 1

```
function dist = pairwise_weighted_distance(X1, X2, w)
% Return the pairwise weighted distance between X1 and X2
% X1: n1 X d matrix
% X2: n2 X d matrix
% dist: n1 X n2 matrix, dist(i,j)= dist(X1(i,:), X2(j,:));

n1 = size(X1,1);
n2 = size(X2,1);

var1 = repmat(sum((X1*diag(w)).^2,2), 1, n2); %broken down to three parts for debugging purposes
var2 = repmat(sum((X2*diag(w)).^2,2)', n1, 1);
var3 = ((X1*diag(w))*(X2*diag(w))');

dist=sqrt(var1 + var2 - 2*var3);
end
```



## Problem 3: Maximum Likelihood estimation on discrete variables

(a) To ensure that Pr[X=i] is a valid PMF, the constraint on $\theta = [\theta_1, \theta_2]$ should be such that the sum of all probabilities of X given $\theta$ is less than or equal to 1. Hence: $\theta_1 + 2\theta_1 + \theta_2 = 1 - - > \theta_2 = 1 - 3\theta_1$

(b) The joint probability then is: $L(\Theta) = \prod_i^n P(X_i|\theta) = \theta_1^{n_1} * 2\theta_1^{n_2} * \theta_2^{n3}$

taking the log: $log(L(\Theta)) = n_1 log(\theta_1) + n_2(log(2) + log(\theta_1)) + n_3(log(\theta_2))$

$log(L(\Theta)) = n_1(log(\theta_1)) + n_2(log(2) + log(\theta_1)) + n_3(log(\theta_2))$

$log(L(\Theta)) = (n_1 + n_2)log(\theta_1) + n_3 log(1 - 3\theta_1) + n_2 log(2)$

(c) Taking the derivative WRT $\theta_1$ and setting it to zero will give the MLE:

$\frac{d(log(L(\Theta)))}{d\theta_1} = \frac{n_1+n_2}{\theta_1} + \frac{n_3}{1-3\theta_1} = 0$

$\theta_1 = \frac{n_1+n_2}{3n_1+3n_2-n_3}$

$\theta_2 = 1 - 3(\frac{n_1+n_2}{3n_1+3n_2-n_3})$

## Problem 4: Maximum Likelihood estimation on Gaussian variables

(1) The Code for the PDF is as follows:

```
function px = pdfGaussian(X, mu, Sigma)
% Calculate the likelihood for a multivariate Gaussian distribution
dims = size(X,2); %number of features
var1 = (1/(2*pi*sqrt(det(Sigma))))^dims; %broken down for debugging purpose
var2 = (-1/2)*sum((X-mu)'/(Sigma)*(X-mu));%broken down for debugging purpose
```

```
px = var1*exp(var2);
end
```

(2) To show that the maximum likelihood estimators are empirical mean and covarience, we will follow the procedure below:

   i. Take the log of the likelihood function: $L(\mu, \Sigma : x_1, x_2, ..., x_n)$ because it is easier to take the derivative of its log, however it will preserve the maximum/minimum point.

   ii. Maximize the log of L() (let's call the log(L()) as l()) by taking its partial derivative for $\mu$ and $\Sigma$.

   iii. Extract the empirical mean and covariance from the derivatives.

   $log(L()) = ln(((\frac{1}{2\pi})det(\Sigma)^{-1/2})^n exp(\frac{-1}{2})\Sigma^{-1}\Sigma_j^n(x_j - \mu)^2)$

   $l() = nln(\frac{1}{2\pi}det(\Sigma)^{-1/2}) + (\frac{-1}{2})\Sigma^{-1}\Sigma_j^n(x_j - \mu)^2$ (ln and exp canceled out!)

   $l() = -nln(\sqrt{2\pi}) - nln(\Sigma) - (\frac{-1}{2})\Sigma^{-1}\Sigma_j^n(x_j - \mu)^2)$

   first for $\mu$:

   $\frac{dl}{d\mu} = \frac{-1}{2\Sigma}\Sigma_i^n(x_i - \mu)2 = 0$

   $= \frac{-1}{\Sigma}\Sigma_i^n x_i - n\mu = 0$

   for it to be equal to zero the summation part has to be equal to zero:

   $\Sigma_i^n x_i - n\mu = 0$

   $\mu = \frac{1}{n}\Sigma_i^n x_i$

   Now for $\Sigma$:

   $\frac{dl}{d\Sigma} = 0 = \frac{-n}{2\Sigma} + (-\Sigma_i^n(x_i - \mu)^2)\frac{-1}{2\Sigma^2})$

   $\frac{1}{2\Sigma}(-n + \frac{1}{\Sigma}\Sigma_i^n(x_i - \mu)^2) = 0$

   Which is equal to zero if the inside part is equal to zero which will leave us with the coverience formula:

   $\Sigma = \frac{1}{n}\Sigma_i^n(x_i - \mu)^2$

## Problem 5: Bayes Classifier

(a) $p(y = -1) = \frac{6}{10}$
$p(y = 1) = \frac{4}{10}$

$p(x_1|y = -1) = \frac{3}{6}$
$p(x_2|y = -1) = \frac{5}{6}$
$p(x_3|y = -1) = \frac{4}{6}$
$p(x_4|y = -1) = \frac{5}{6}$

$p(x_5|y=-1)=\frac{2}{6}$

$p(x_1|y=1)=\frac{3}{4}$
$p(x_2|y=1)=\frac{0}{4}$
$p(x_3|y=1)=\frac{3}{4}$
$p(x_4|y=1)=\frac{2}{4}$
$p(x_5|y=1)=\frac{1}{4}$

$p(x_1=1)=\frac{6}{10}$
$p(x_2=1)=\frac{5}{10}$
$p(x_3=1)=\frac{7}{10}$
$p(x_4=1)=\frac{7}{10}$
$p(x_5=1)=\frac{3}{10}$

(b) we calculate $p(y|x_i)=\frac{p(x|y)p(y)}{p(x)}$ for all the features. where $p(x|y)$ for multiple features is simple the multiplication of individual features p(x|y). Given that the sum of that probabilities is 1 then if the result is >=0.5 we will classify as y=1 (Read) and if <0.5 the will classify as y=-1 (Don't Read).

case of x=(0,0,0,0,0)

$p(y=1|0,0,0,0,0)=\frac{p(x_1=0|y=1)*p(x_2=0|y=1)*...*p(x_5=0|y=1)*p(y=1)}{p(x_1=0|y=1)*p(x_2=0|y=1)*...*p(x_5=0|y=1)*p(y=1)+p(x_1=0|y=0)*p(x_2=0|y=0)*...*p(x_5=0|y=0)*p(y=0)}$

if the result is >= 0.5 classify as (y=1 –> Read) otherwise dont read

case of x=(1,1,0,1,0) Same as above:

$p(y=1|1,1,0,1,0)=\frac{p(x_1=1|y=1)*p(x_1=0|y=1)*...*p(x_5=0|y=1)*p(y=1)}{p(x_1=1|y=1)*p(x_2=1|y=1)*...*p(x_5=0|y=1)*p(y=1)+p(x_1=1|y=0)*p(x_2=1|y=0)*...*p(x_5=0|y=0)*p(y=0)}$

if the result is >= 0.5 classify as (y=1 –> Read) otherwise dont read

(c) The posterior probability of y=1 is the sum of the probabilities calculated above divided by their number:

$p(y|(1,1,0,1,0))=\frac{1}{n}\Sigma_i^n p(y|x_i)=0.301$

(d) We should use the Naive Bayes because the features are not related with each other so we can consider them to be independent features which produces the Naive Bayes classifier. Also give the number of features (5) we will have $2^5$ points which as it increases it will produce over-fitting problem.

## Problem 6: Gaussian Bayes Classifier