ML homework 1 Solutions

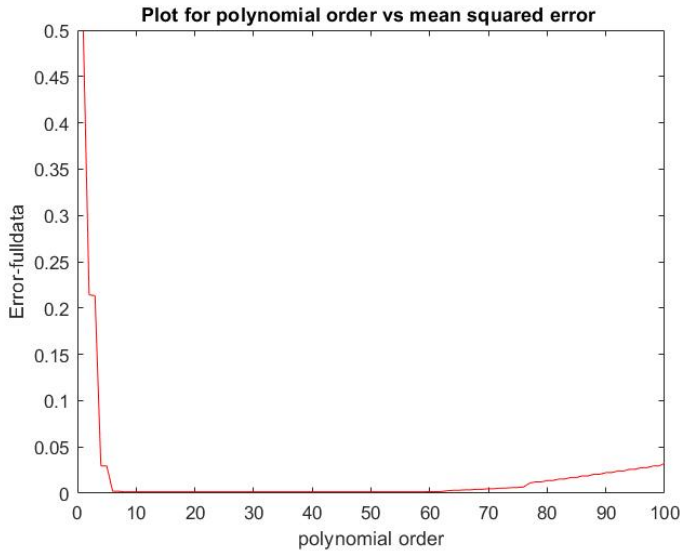
Problem1

The MATLAB code for the first problem is given below the following code uses polyreg function from polyreg.m file.

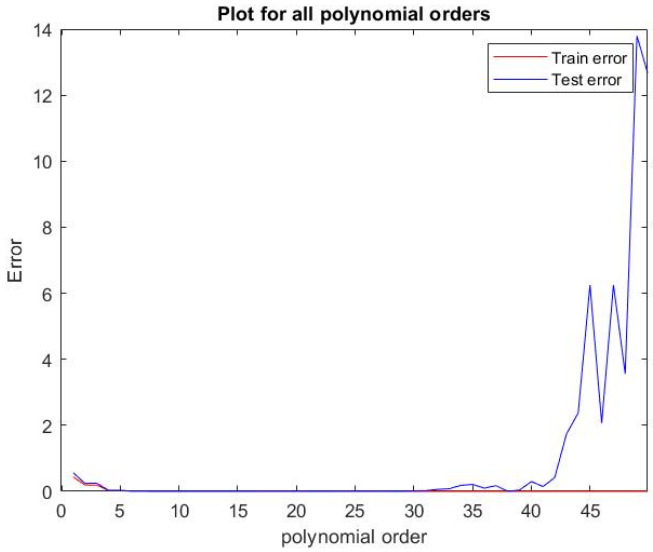
```
clc%clear command prompt
clear%clear workspace
load problem1.mat;
Lx=length(x);
Ly=length(y);
%Shuffle the data
ran=randperm(Lx);
x=x(ran);
y=y(ran);
x=normalize(x);
y=normalize(y);
%The problem asks to fit the data first without train and test split.
num_full=100;
err_full=zeros(num_full);
for m=1:num_full
    [err,model] = polyreg(x,y,m);
    err_full(m)=err;
end
clf
plot(err_full(1:num_full),'r');
xlabel('polynomial order');
ylabel('Error-fulldata');
title("Plot for polynomial order vs mean squared error ");
%Get train and test split
xtr=x(1:fix(Lx/2));
xts=x(fix(Lx/2)+1:end);
ytr=y(1:fix(Lx/2));
yts=y(fix(Lx/2)+1:end);
%len variable contains the maximum order of polynomial used to fit.
len= 50;
xu=1:len;
```

```
testerr=zeros(len);
trainerr=zeros(len);
%fit the model for various polynomial orders
for m=1:len
    [err,model,errT] = polyreg(xtr,ytr,m,xts,yts);
    testerr(m)=errT;
    trainerr(m)=err;
end
%plot test and train error against the polynomial order
figure()
clf
plot(trainerr(1:len),'r');
hold on
plot(testerr(1:len),'b');
title("Plot for all polynomial orders");
legend('Train error','Test error');
xlabel('polynomial order');
ylabel('Error');
%The test error looks small in the 5 to 15 range
figure()
clf
plot(trainerr(5:15),'r');
xticklabels({5:15})
hold on
plot(testerr(5:15),'b');
xticklabels({5:15})
title("Plot to find best polynomial order");
legend('Train error','Test error');
xlabel('polynomial order');
ylabel('Error');
%The order with lowest error and small complexity is 6
%Plot the predicted data against input data
answer=6;
[err,model,errT] = polyreg(xtr,ytr,answer,xts,yts);
qq = zeros(length(x),answer);
for i=1:answer
  qq(:,i) = x.^(answer-i);
end
q  = 1:500 ;
figure()
clf
scatter(x,qq*model)
hold on
scatter(x,y)
legend("predicted data","true data")
title("Plot between predicted and original data");
```

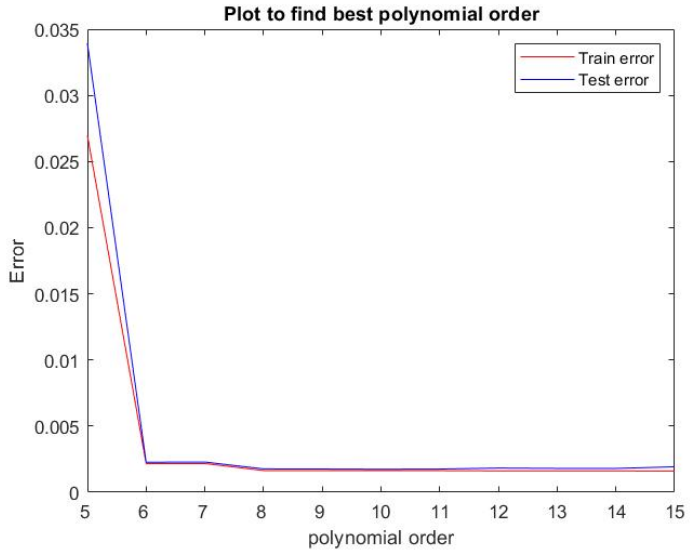
The plot for full data polynomial fit looks like below
From this plot we can see that the range from 6 to 65 looks reasonable since we have the lowest mean squared error in this range



The plots from the above code should like below
The first plot shows how the train and test error change with polynomial order

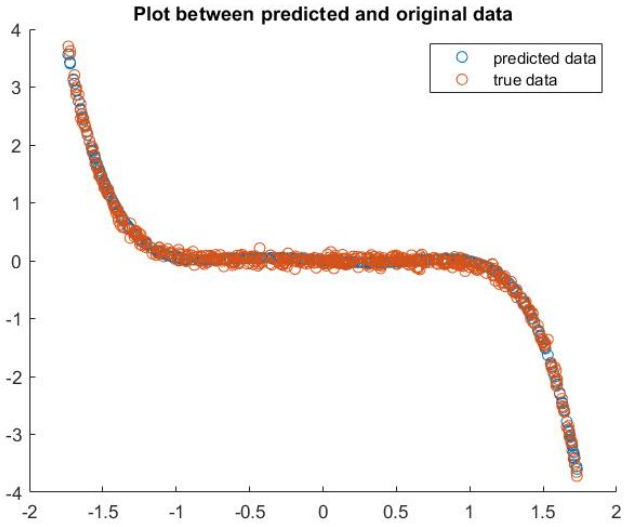


Since the polynomial order is minimum between 5 and 15 so lets look at the error between this range The error becomes stable after order 6 and it is best



to choose the least complex order when we have the same error so lets choose 6.

The plot to compare predicted and input data for the order 6 fit.



Problem 2

The MATLAB code is similar from polyreg.m file but modified as follows to accommodate multi-variate case(regmulti.m).

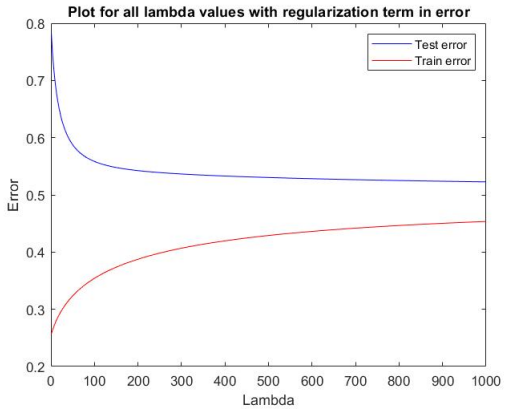
```
function [err,err_unreg,model,errT,errT_unreg] = regmulti(x,y,lambda,xT,yT)
[n,m] = size(x)
model = inv(x'*x + lambda*eye(m))*(x'*y);

err   = (1/(2*length(x)))*sum((y-x*model).^2) + (lambda/(2*length(x))) * (model'*model);
err_unreg= (1/(2*length(x)))*sum((y-x*model).^2) ;

if (nargin==5)

  errT  = (1/(2*length(xT)))*sum((yT-xT*model).^2) + (lambda/(2*length(xT))) *(model'*model)
  errT_unreg  = (1/(2*length(xT)))*sum((yT-xT*model).^2) ;
end
```

Now, lets use this function we have written to perform two-fold cross validation on the dataset to find lambda value.

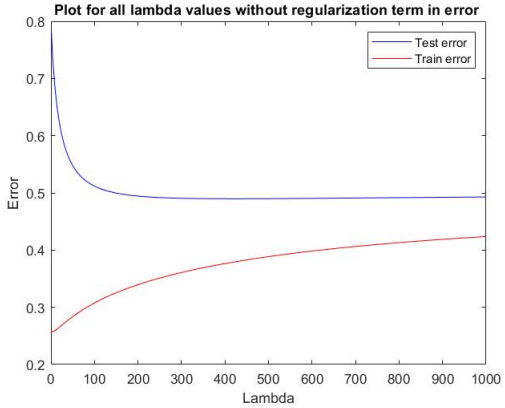
```
clc%clear command prompt
clear%clear workspace
load problem2.mat;
```

```
Lx=length(x);
Ly=length(y);
%Shuffle the data
ran=randperm(Lx);
x=x(ran,:);
y=y(ran,:);
x=normalize(x);
y=normalize(y);
%Get train and test split
xtr=x(1:fix(Lx/2),:);
xts=x(fix(Lx/2)+1:end,:);
ytr=y(1:fix(Lx/2),:);
yts=y(fix(Lx/2)+1:end,:);


testerr=zeros(1000,1);
trainerr=zeros(1000,1);
testerr_unreg=zeros(1000,1);
trainerr_unreg=zeros(1000,1);
%fit the model for various Lambda values
index = 1;
for m=0:1000
    [err,err_unreg,model,errT,errT_unreg] = regmulti(xtr,ytr,m,xts,yts);
    testerr(index)=errT;
    trainerr(index)=err;
    testerr_unreg(index)=errT_unreg;
    trainerr_unreg(index)=err_unreg;
    index = index + 1;
end
%plot test and train error against the Lambda order
clf
plot(testerr(1:1000),'b');

hold on
plot(trainerr(1:1000),'r');
title("Plot for all lambda values with regularization term in error");
legend('Test error','Train error');
xlabel('Lambda');
ylabel('Error');
figure()
clf;
plot(testerr_unreg(1:1000),'b');

hold on
plot(trainerr_unreg(1:1000),'r');
title("Plot for all lambda values without regularization term in error ");
```

```
legend('Test error','Train error');
xlabel('Lambda');
ylabel('Error');
```



Similarly the graph for train and test error without regularization term is as follows



Thus, for lambda value of approximately 700 makes the train and test error stable with approximate error nearing 0.5. We see that as lambda, increases the test error decreases and then becomes stable after a while, whereas, simultaneously the train error increases and becomes stable after a certain limit.

Problem 3

$g(-z) = \frac{1}{1+exp(z)}$
Consider 1- g(z),
$1 - g(z) = 1 - \frac{1}{1+exp(-z)}$
$\implies 1 - g(z) = 1 - \frac{exp(z)}{1+exp(z)}$
$\implies 1 - g(z) = \frac{1}{1+exp(z)} = g(-z)$

Inverse

The inverse: $g^{-1}(y) = x$
$\implies y = g(x)$
$\implies y = \frac{1}{1+exp(-x)}$
$\implies exp(-x) = \frac{1-y}{y}$
$\implies x = ln(\frac{y}{1-y})$
$\therefore g^{-1}(y) = ln(\frac{y}{1-y})$

Problem 4

Calculation of gradient and R_{emp} and its vector form

$$R_{emp} = \frac{1}{N}\sum_{i=1}^{N}[(y_i - 1) * \log(1 - f(x_i;\theta)) - y_i * log(f(x_i;\theta))] \tag{1}$$

$$R_{emp} = -\frac{1}{N}\sum_{i=1}^{N}[(1 - y_i) * \log(1 - f(x_i;\theta)) + y_i * log(f(x_i;\theta))] \tag{2}$$

$$\frac{\partial R_{emp}}{\partial \theta_j} = -\frac{1}{N}\sum_{i=1}^{N}[-(1-y_i)\frac{1}{1-f(x_i;\theta)}f(x_i;\theta)(1-f(x_i;\theta))+y_i\frac{1}{f(x_i;\theta)}(1-f(x_i;\theta))f(x_i;\theta)]\frac{\partial}{\partial \theta_j}(\theta^T x_i) \tag{3}$$

$$\frac{\partial R_{emp}}{\partial \theta_j} = \frac{1}{N}\sum_{i=1}^{N}[f(\theta;x_i) - y_i] * x_{ij} \tag{4}$$

$$\theta := \theta - \frac{\eta}{N} X^T (f(X;\theta) - Y) \tag{5}$$

Note:

$$\frac{\partial}{\partial \theta} f(x_i;\theta) = f(x_i;\theta)(1 - f(x_i;\theta)) \tag{6}$$

Binary Logistic Regression (Main Function)

```
clc;
clear;
load dataset4.mat;
    shapeX = size(X);
    theta = ones(shapeX(2),1);  %Declaration of parameter vector
    theta_prev = zeros(shapeX(2),1);
    iter = 1;  %number of iterations
    alpha = 0.1; %Learning rate
    max_iter =10000;
    costs = zeros(max_iter,1); % R emperical vector
    accuracy = zeros(max_iter,1); %Matrix to contain the accuracy of prediction over test da
    err=zeros(max_iter,1);
    tolerance = 0.001;
    while (norm(theta-theta_prev)>tolerance) && (iter<max_iter)
        [cost,grad,f] = Remp(X,Y,theta);
        theta_prev=theta;
        theta = theta-alpha*grad;
        costs(iter) = cost;
        [acc,err_temp] = Prediction(X,Y,theta);
        accuracy(iter)=acc;
        err(iter)=err_temp;
        iter=iter+1;

    end
    disp("Number of iterations");
    disp(iter-1);
    subplot(3,1,1)
    plot(1:iter-1,costs(1:iter-1))
    title("Empirical risk")
    subplot(3,1,2)
    plot(1:iter-1,accuracy(1:iter-1))
    title("accuracy")
    subplot(3,1,3)
    plot(1:iter-1,err(1:iter-1))
    title("binary classification error")
    figure()
    mask1=Y==0;
    mask2=Y==1;
```

```
X_out=X(mask1,:,:);
disp(X(1,1))
X_out1=X(mask2,:,:);
XX = (-theta(3)-X(:,1)*theta(1))/theta(2);
%gscatter(X(:,1),X(:,2),Y,'br','xo') requires machine learning toolbox
scatter(X_out(:,1),X_out(:,2),'bs')
hold on
scatter(X_out1(:,1),X_out1(:,2),'ro')
hold on
plot(X(:,1),XX,'k')
title("Decision Boundary for the dataset");
legend('0','1',"Decision Boundary")
```

Function for gradient and R_{emp} Calculation

```
function [cost,grad,f] = Remp(X,Y,theta)
    m = length(Y);
    grad = zeros(size(theta));
    f = sigmoid(theta'*X')';
    cost = (-1/m)*sum(Y.*log(f)+(1-Y).*log(1-f));
    for j = 1:size(grad)
        grad(j) = (1/m)*sum((f-Y).*X(:,j));
    end
end
```

Function for $f(x;\theta)$ calculation

```
function Y = sigmoid(X)
    Y = 1./(1+exp(-X));
end
```

Function for Prediction using the computed θ^*

```
function [accuracy,error] = Prediction(X,Y,theta)
    f = sigmoid(theta'*X')';
    error = 0;
    for idx = 1:size(X)
        if(f(idx)>=0.5)
            f(idx) = 1;
        end
        if(f(idx)<0.5)
            f(idx) = 0;
        end
    end
        err = Y - f;
        for idx = 1:size(X)
            if(err(idx)~=0)
                error=error+1;
```

```
            end
        end
        accuracy = 100*(size(X)-error)/size(X);
end
```

Number of iterations for the given learning rate and tolerance is 6500

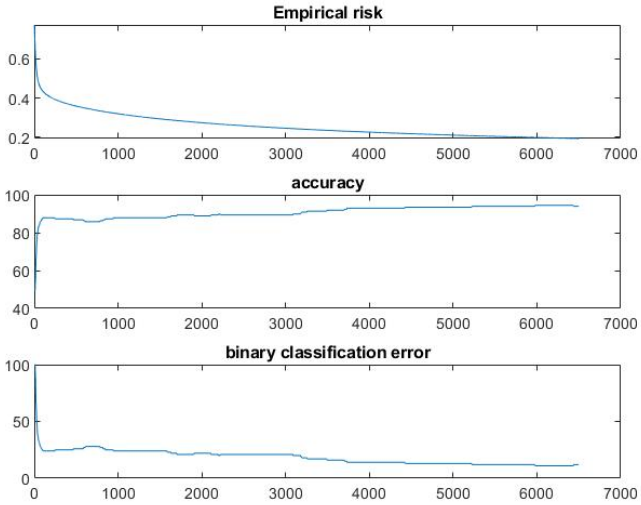


Figure 1: Empirical Risk, Accuracy and Binary Classification error plot against iteration

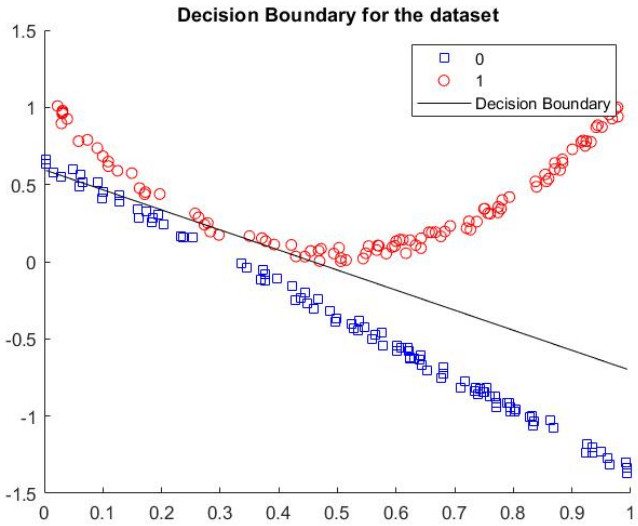


Figure 2: Decision Boundary for the data-set