homework2

Name:Fang Yuting

Studeng ID:1901212576

Major:19 Fintech

Problem 1

---

1. The code is in file 'closed_form_1.m'
2. **Mathematical formula for the linear model**

$Y = X\beta + \epsilon, \epsilon\ N(0, \sigma^2)$

- **Train data**

$y = -124.5943 + 0.0642x_1 - 0.0065x_2 + 0.0001x_3 - 0.0165x_4 - 0.0066x_5$

$+0.0038x_6 + 0.0931x_7 - 1.5376x_8$

$R^2 = 0.7509$

- **Test data**

$$R^2 = \frac{\operatorname{var}(X\hat{\beta})}{\operatorname{var}(y)} = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

$R^2 = SSR/SST = 0.2252$

3. **Significant variables**

For train model, we find the p-value of variables <0.05.

MEI(x_1),CO2(x_2),CFC-11(x_5),CFC-12(x_6),TSI(x_7),Aerosols(x_8).

4. **Necessary condition for closed form solution :** $(X'X)^{-1}$ exists. X shoule be full rank.

We do the same thing to the climate_change_2.csv.

The rank of (X2train) is 9 but the size of X2train is 10 which means X2train is not in full rank.

Thus, the beta can't be computed because the matrix $(X'X)^{-1}$ is close to singular. The reason is that there is multicollinearity between climate_change_2.csv . There are too many variables and redundant variable. The X is not full rank. As a result, we can't get a $(X'X)^{-1}$.

```
%Problem 1
```

```
%Problem 1.1
ytrclimateData = readtable('climate_change_1.csv');
train_data = climateData{climateData.Year <= 2006,:};
test_data = climateData{climateData.Year >2006,:};

ytrain = train_data(:,[11]);
xtrain = train_data(:,[3:10]);
Xtrain =[ones(length(xtrain),1) xtrain];

beta = closed_form_1(Xtrain,ytrain);

%Problem 1.2 Y=x* beta +epsilon
whichstats = {'beta','rsquare','tstat'};
Reg= regstats(ytrain,xtrain,'linear',whichstats);

ytest = test_data(:,11);
xtest = test_data(:,[3:10]);
Xtest =[ones(length(xtest),1) xtest];

yhat = Xtest * beta;
SSR = sum((yhat - mean(ytest)).^2);
SST = sum((ytest - mean(ytest)).^2);
R2Test = SSR/SST;

%Problem 1.3 the first 1 is for intercept
% 1 2 5 6 7 8
insignificantVariable = find(Reg.tstat.pval < 0.05);

%Problem 1.4 (x' * x)^ (-1) exists
climateData2 = readtable('climate_change_2.csv');
train_data2 = climateData2{climateData2.Year <= 2006,:};
test_data2 = climateData2{climateData2.Year >2006,:};

y2train = train_data2(:,12);
x2train = train_data2(:,1:11);
X2train =[ones(length(x2train),1) x2train];
beta2 = closed_form_1(X2train,y2train);
rank(X2train);
```

```
function beta = closed_form_1(x,y)
    beta = (x'* x)\(x'* y)
end
```

Problem 2

1.**Loss function for linear model with L1 regularization（LASSO regression）** minimize

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}|\beta_j| \;=\mathrm{RSS}+\lambda\sum_{j=1}^{p}|\beta_j|.$$

Loss function for linear model with L2 regularization（Ridge regression）

$$\min_{\theta} J(\theta) \qquad J(\theta)=\frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2+\lambda\sum_{j=1}^{n}\theta_j^2\right]$$

2.The code is in file 'closed_form_1.m'

Problem2:

$y=-2.5925+0.0554x_1+0.0063x_2+0.0001x_3-0.0133x_4-0.0058x_5$

$+0.0036x_6+0.0032x_7-1.3287x_8$

$R^2=0.7148$

3.**Compare the two solutions in Problem 1 and Problem 2. lambda = 0.001**

Problem1:

$y=-124.5943+0.0642x_1-0.0065x_2+0.0001x_3-0.0165x_4-0.0066x_5$

$+0.0038x_6+0.0931x_7-1.5376x_8$

$R^2=0.7509$

Why linear model with L2 regularization is robust.

L2 regularization is a technique to discourage the complexity of the model. It does this by penalizing the loss function.

In L2 regularization, the coeffficient beta shinks and this helps to solve the overfitting problem. It makes the coefficient of terms smaller than OLS and helps simplify the model, thus the optimal solution won't be influnced much by data fluction.

4. **Change the regularization parameter lambda**

The R^2 table are shown below.

lambda	Training set	Testing set
10	0.6746	0.9409
1	0.6795	0.8468
0.1	0.6945	0.6733
0.01	0.7117	0.5853
0.001	**0.7148**	0.5625

In order to select the best parameter to fit model, we do 10-fold cross validation.

lambda	$MeanSquareError$ **in validation set(10-fold cross validation)**
10	0.0109
1	0.0106
0.1	0.0099
0.01	0.0098
0.001	**0.0097**

Based on the result of 10-fold cross validation, we decide to choose **lambda = 0.001** as the parameter.

```
% Problem 2
% Problem 2.1 see in the homework.markdown

% Problem 2.2
climateData = readtable('climate_change_1.csv');
train_data = climateData{climateData.Year <= 2006,:};
test_data = climateData{climateData.Year >2006,:};

y = train_data(:,[11]);
x = train_data(:,[3:10]);
X =[ones(length(x),1) x];
beta_ridge = closed_form_2(X,y,10)

% Problem 2.3
beta_OLS = closed_form_1(X,y)
beta_ridge = closed_form_2(X,y,10)

% Problem 2.4
lambdaPara = [0.001,0.01,0.1,1,10];
y = train_data(:,[11]);
x = train_data(:,[3:10]);
```

```matlab
X =[ones(length(x),1) x];

%Testing data R2
    yy = test_data(:,11);
    xx = test_data(:,[3:10]);
    XX =[ones(length(xx),1) xx];

for i =1:5
    beta_ridge = closed_form_2(X,y,lambdaPara(i));
    yhat = X * beta_ridge;
    SSR_train = sum((yhat - mean(y)).^2);
    SST_train = sum((y - mean(y)).^2);
    R2Test_train(i) = SSR_train/SST_train;

    yhat = XX * beta_ridge;
    SSR_test = sum((yhat - mean(yy)).^2);
    SST_test = sum((yy - mean(yy)).^2);
    R2Test_test(i) = SSR_test/SST_test;
end
disp(R2Test_train)
disp(R2Test_test)

% Problem 2.4 CV-validation in trainset
lambdaPara = [0.001,0.01,0.1,1,10];
[m,n] = size(train_data);
indices = crossvalind('Kfold',train_data(1:m,n),10);

for i = 1:5
    for k = 1:10
        test = (indices ==k);
        train = ~test;
        train_data_new = train_data(train,:);
        test_data_new = train_data(test,:);
        y = train_data_new(:,11);
        x = train_data_new(:,3:10);
        X =[ones(length(x),1) x];
        beta_ridge = closed_form_2(X,y,lambdaPara(i));

        yy = test_data_new(:,11);
        xx = test_data_new(:,3:10);
        XX =[ones(length(xx),1) xx];

        yhat = XX * beta_ridge;
        ybar = mean(yy);
        MSE(k) = mean((yy -  yhat).^2);
    end
    meanMSE(i) = mean(MSE);
end
disp(meanMSE)
```

```
function beta = closed_form_2(x,y,lambda)
    [m,n] = size(x);
    I = eye(n);
    temp = x' * x + eye(n) *lambda;
    if det(temp) ==0
        disp('This matrix is singular, cannot do inverse');
    end
    beta = (temp)\(x'* y);
end
```

Problem 3

1. **Use less variables to train model.**

First we select the variable that is significant in Problem1(p value < 0.05).

MEI(x_1),CO2(x_2),CFC-11(x_5),CFC-12(x_6),TSI(x_7),Aerosols(x_8).

We hava a correlation matrix of the six variables. As shown below, we find the some variables are highly related.

1.0000 -0.0411 -0.0508 0.0690 0.0083 -0.1545

-0.0411 1.0000 **0.9767** 0.5141 **0.8527** 0.1774

-0.0508 **0.9767** 1.0000 0.5225 **0.8679** 0.1998

0.0690 0.5141 0.5225 1.0000 **0.8690** 0.2720

0.0083 **0.8527 0.8679 0.8690** 1.0000 0.2553

-0.1545 0.1774 0.1998 0.2720 0.2553 1.0000

We drop some on the variable and get a matrix that is not highly correlated. We remain the variables MEI(x_1),CO2(x_2),CFC-11(x_5),TSI(x_7). The corelation matrix are shown below.

1.0000 -0.0411 0.0690 -0.1545

-0.0411 1.0000 0.5141 0.1774

0.0690 0.5141 1.0000 0.2720

-0.1545 0.1774 0.2720 1.0000

workflow to select feature

- If we have experience, we can split some insignificant feature or we can import the whole data in the workspace.
- Then, split the data into training set and testing set.
- Using CV , split training data set into sub-training set and validation set and build a better model using LASSO regression to do feature selection.
- Use mean of MSE on validation set as the optimal measure, the least MSE will be the best model.
- Using the result of train model to test set.

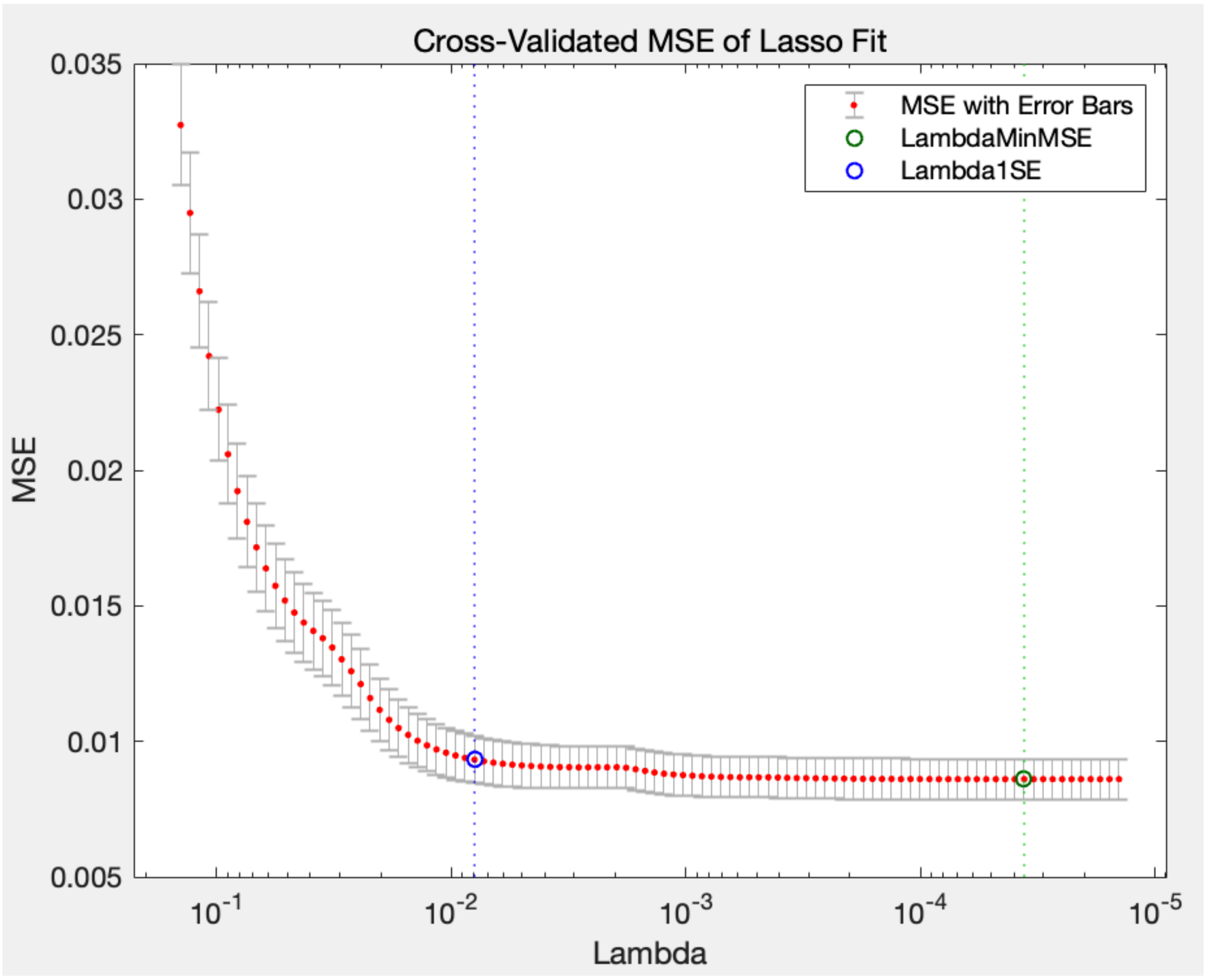
2. **Train a better model than the model in Problem 2.**

In LASSO regression, we have 10-fold Cross Validation.

Finally, we build a less variable model.

$$y = 0.0460x_1 + 0.0077x_2 + 0.0069x_4 + 0.00538x_7 - 1.1426x_8$$

In the testing set ,the $R^2 = 0.7487$, higher than Problem 2.



```
%Problem 3
climateData = readtable('climate_change_1.csv');
train_data = climateData{climateData.Year <= 2006,:};
test_data = climateData{climateData.Year > 2006,:};

y = train_data(:,11);
x = train_data(:,3:10);
yy = test_data(:,11);
xx = test_data(:,3:10);

%X =[ones(length(x),1) x];
[B,FitInfo] = lasso(x,y,'CV',10);
fig = figure;
lassoPlot(B,FitInfo,'PlotType','CV')
legend('show');
```

```
idxLambda1SE = FitInfo.Index1SE;
coef = B(:,idxLambda1SE);
coef0 = FitInfo.Intercept(idxLambda1SE);

yhat_test = xx * coef + coef0;
%yhat_test = xx * coef;
SSR_test = sum((yhat_test - mean(yy)).^2);
SST_test = sum((yy - mean(yy)).^2);
R2Test = SSR_test / SST_test
```

Problem4

Write down the gradiend descent algorithm iterative expression of updating the solution of linear model.

Hypothesis: $h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

Parameters: $\theta_0, \theta_1, \ldots, \theta_n$

Cost function:

$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Gradient descent:

Repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \ldots, \theta_n)$$

} (simultaneously update for every $j = 0, \ldots, n$)

Implement it in *gradientDescend* function

The code is in file 'gradientDescend.m'.

I also write a funtion in file 'SGD.m'.

If subset = m, then the SGD become GD.

If subset = 1, SGD is the real SGD.

if 1<subset <m, SGD is mini-batch GD.

```
function Problem4_gradientDescend(X,y)
% Example sample to use this function:
% X is a m * n matrix , we have m data point , the feature is n.
% y is the true value of the data point.
% X = [1 4 3;2 5 6 ;5 1 2 ;4 2 2];
% y = [19;26;19;20];
```

```
[m,n] = size(X); % m is the number of dataset
alpha = 0.002;   %learning rate
num_iters = 1000;
theta = zeros(n, 1);

% cost function
J_history = zeros(num_iters, 1);

for iter = 1:num_iters
    h= zeros(m,1);
    h = X*theta;
    J_history(iter) = (1/(2*m))*sum((h-y).^2);

    tmp1 = zeros(size(X,2),1);
    for i=1:m
        tmp1= tmp1+(h(i)-y(i)).*X(i,:)';
    end

    theta = theta - (alpha/m)*tmp1;            %Every time to update the theta
    disp(J_history(iter));
    disp(theta);
end
    plot(J_history)
    title('the cost function of 1000 iters')
end
```

```
function Problem4_SGD(X,y)
% Example sample to use this function:
% X is a m * n matrix , we have m data point , the feature is n.
% y is the true value of the data point.
% X = [1 4 3;2 5 6 ;5 1 2 ;4 2 2];
% y = [19;26;19;20];
% subset must smaller than m , subset = 2;

[m,n] = size(X); % m is the number of dataset
alpha = 0.002;   %learning rate
num_iters = 200;
theta = zeros(n, 1);
subset = 1 % 1 is for SGD , b is for mini-batch, m is for GD

% cost function
J_history = zeros(num_iters, 1);

for iter = 1:num_iters
    index = randsample(length(X),subset,'false');
    subX = X(index', :);
    suby = y(index',:);
    h= zeros(subset,1);
    h = subX*theta;
    J_history(iter) = (1/(2*subset))*sum((h-suby).^2);

    tmp1 = zeros(size(subX,2),1);
    for i=1:subset
        tmp1= tmp1+(h(i)-suby(i)).*subX(i,:)';
    end

    theta = theta - (alpha/m)*tmp1;            %Every time to update the theta
    disp(J_history(iter));
    disp(theta);
end
end
```