Name: Xinrun Zhang

ML Homework 2 Report

Instructor: Prof. Kaliappan Gopalan

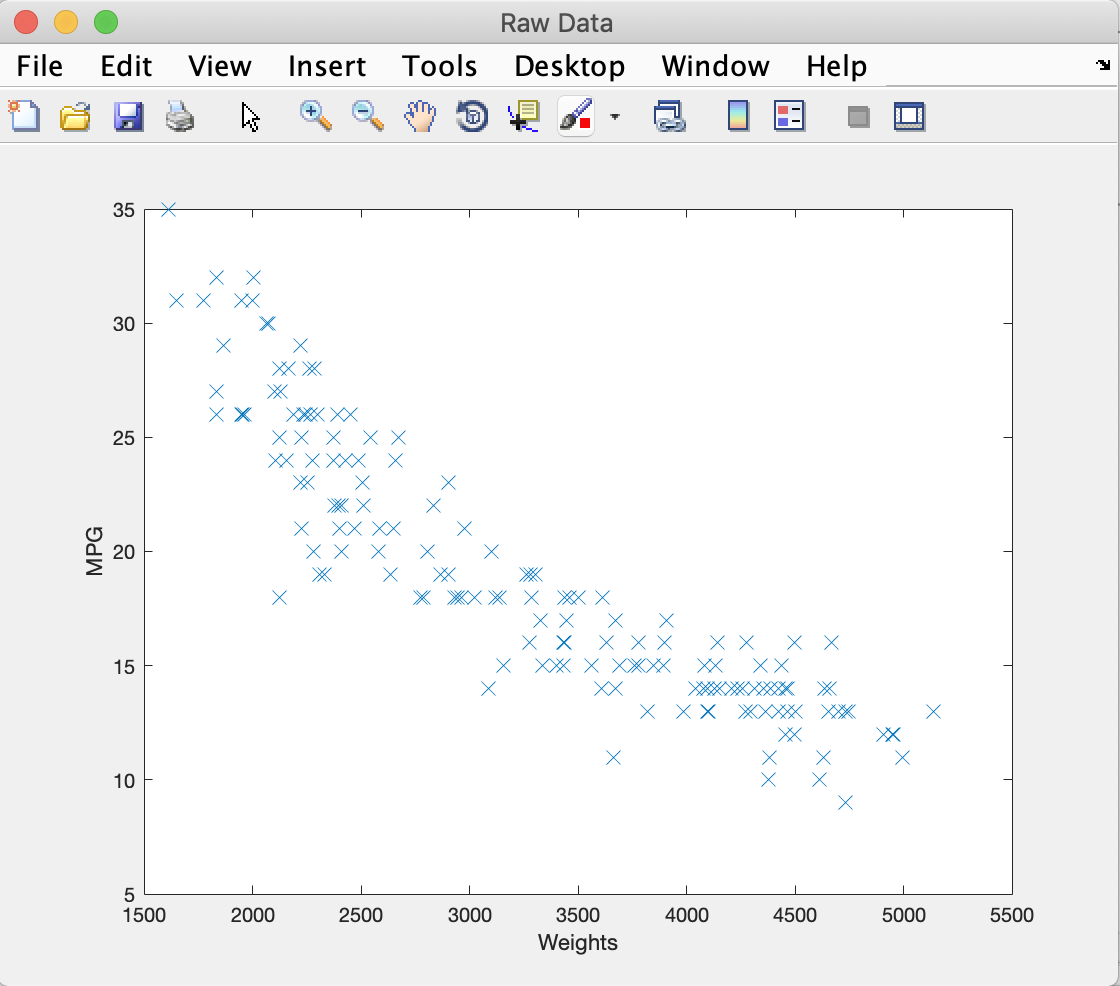
Time: 21/02/2019

# Files

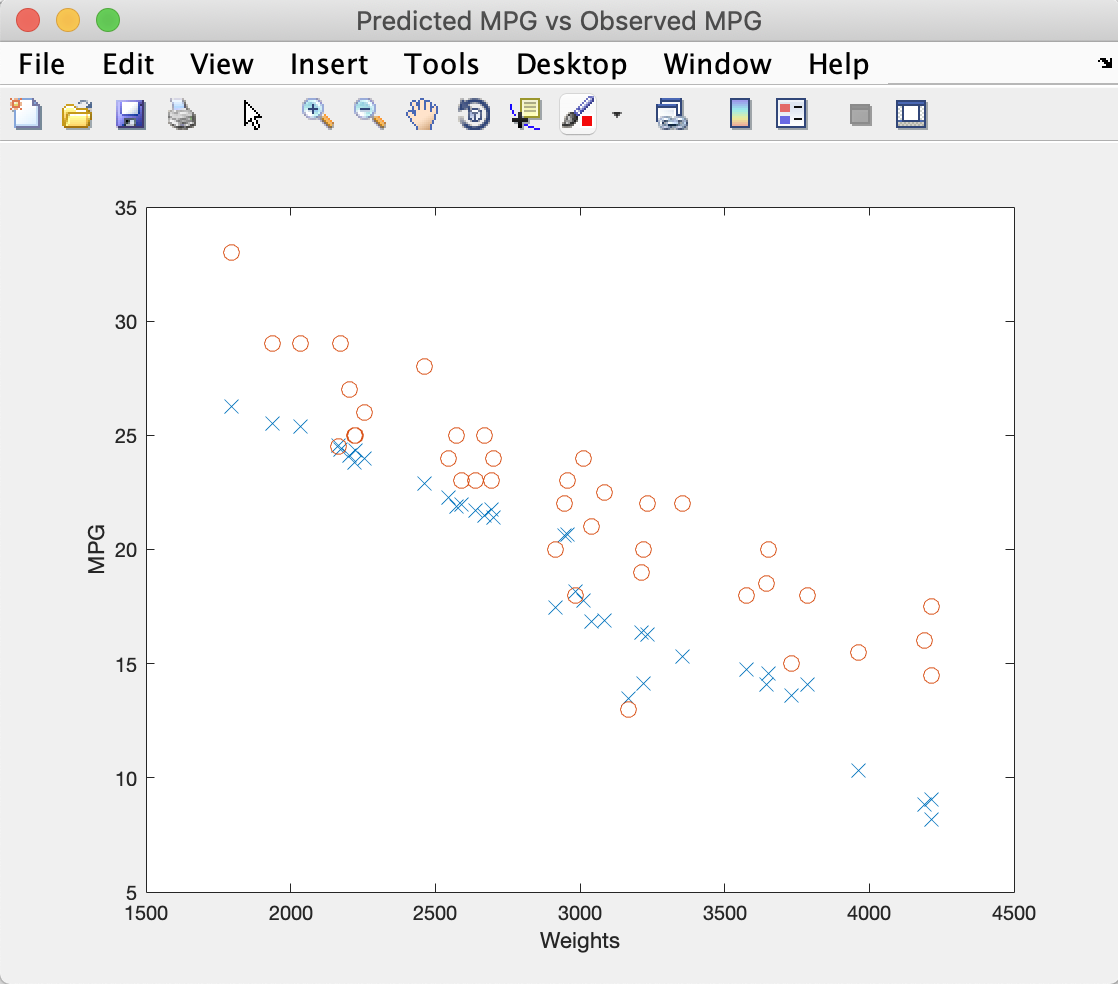
1. main.m – Includes the main program of the project.
2. computeCost.m – This function is used to compute the cost between hypothesis and input y.
3. normalizing.m – This function is used in univariate linear regression to normalize the input x.
4. gradientDescent.m – This function is used in univariate linear regression to execute the gradient descent to find proper parameters.

# Multivariate Linear Regression with Gradient Descent

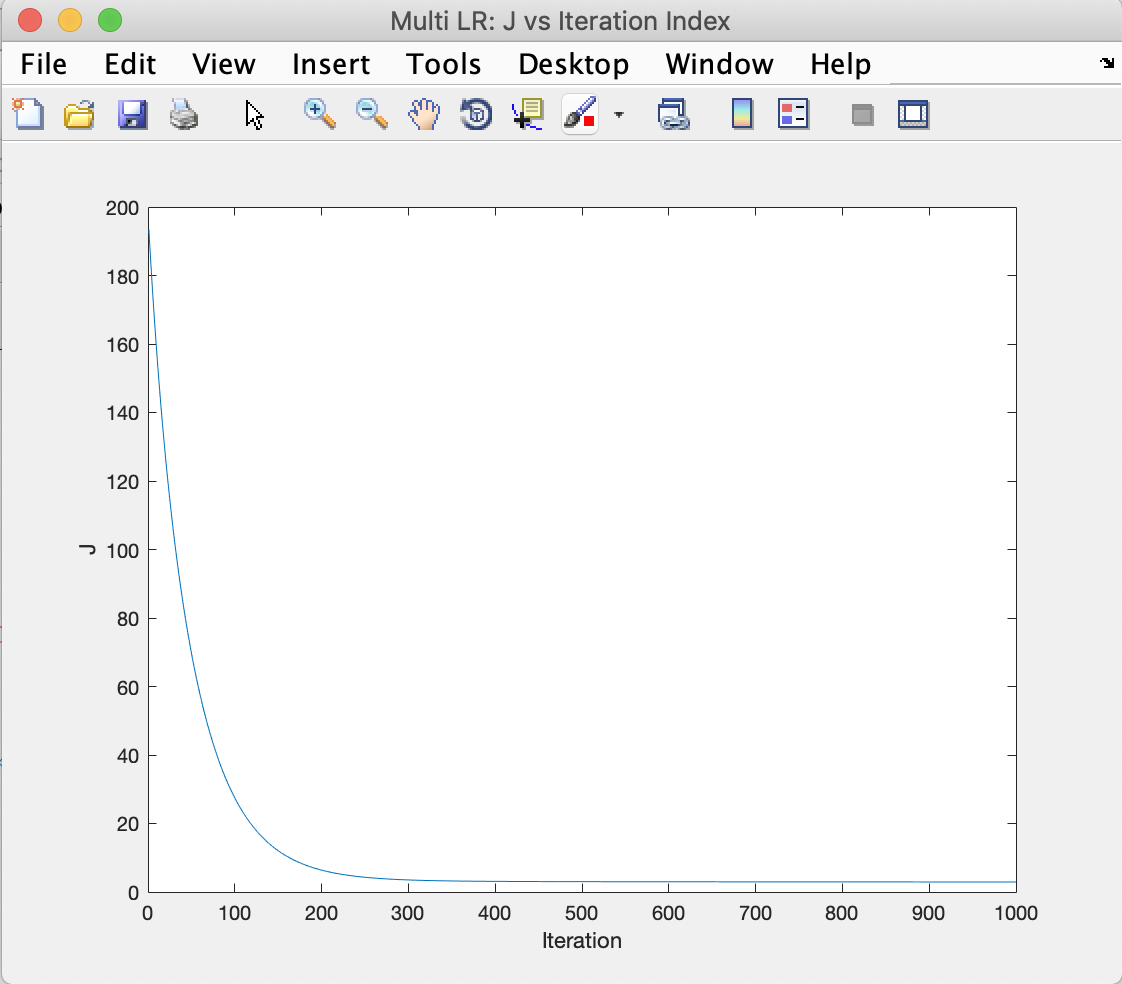
1. Scatter plot of *y* vs. *x*, where *x* is still the weight (as in HW 1)



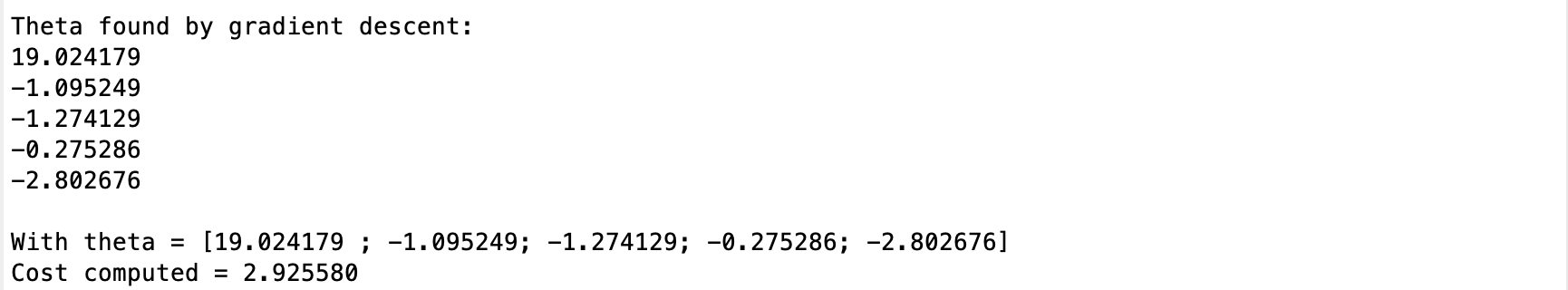
1. Plot of predicted MPG against observed MPG for the validation data



1. Plot of J vs. Iteration index for the training data



1. Minimum J and the hypothesis parameters from gradient descent code

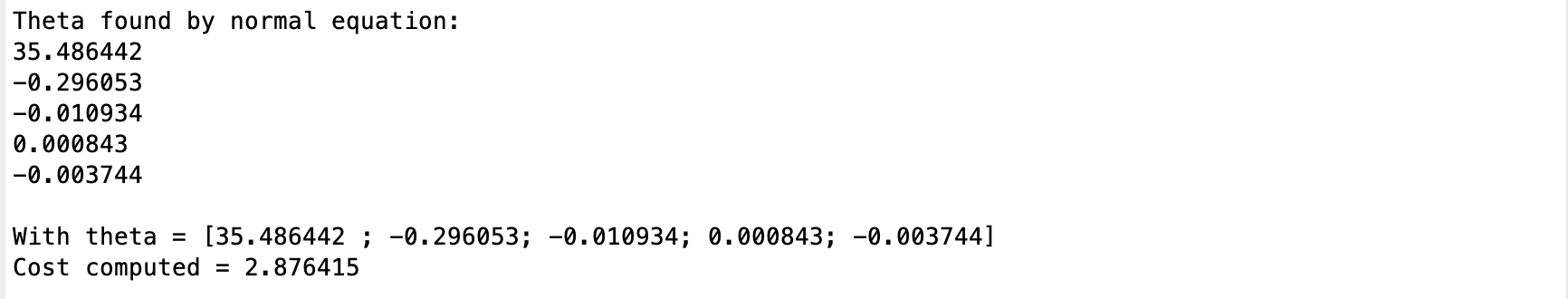


1. Mean squared error err\_cv

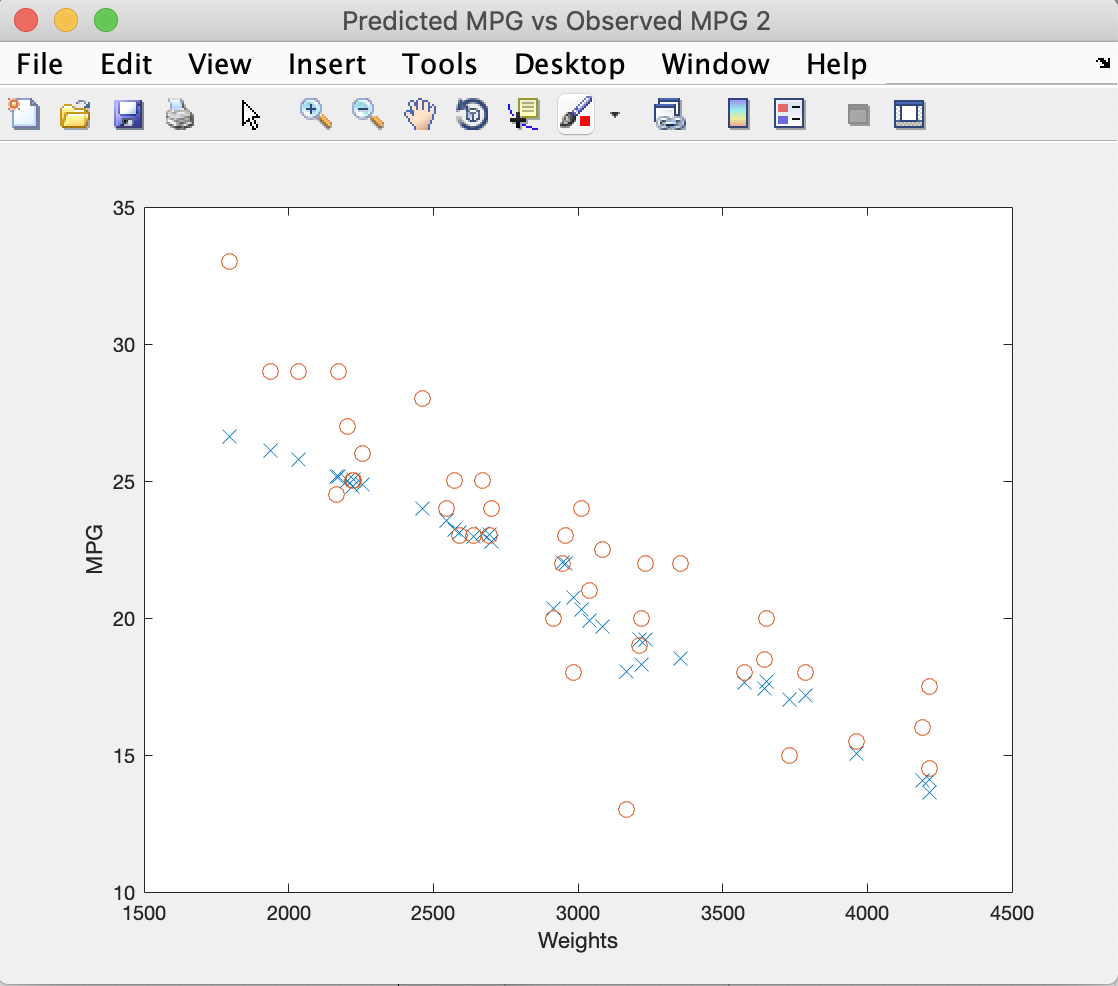


# Multivariate Linear Regression with Normal Equation

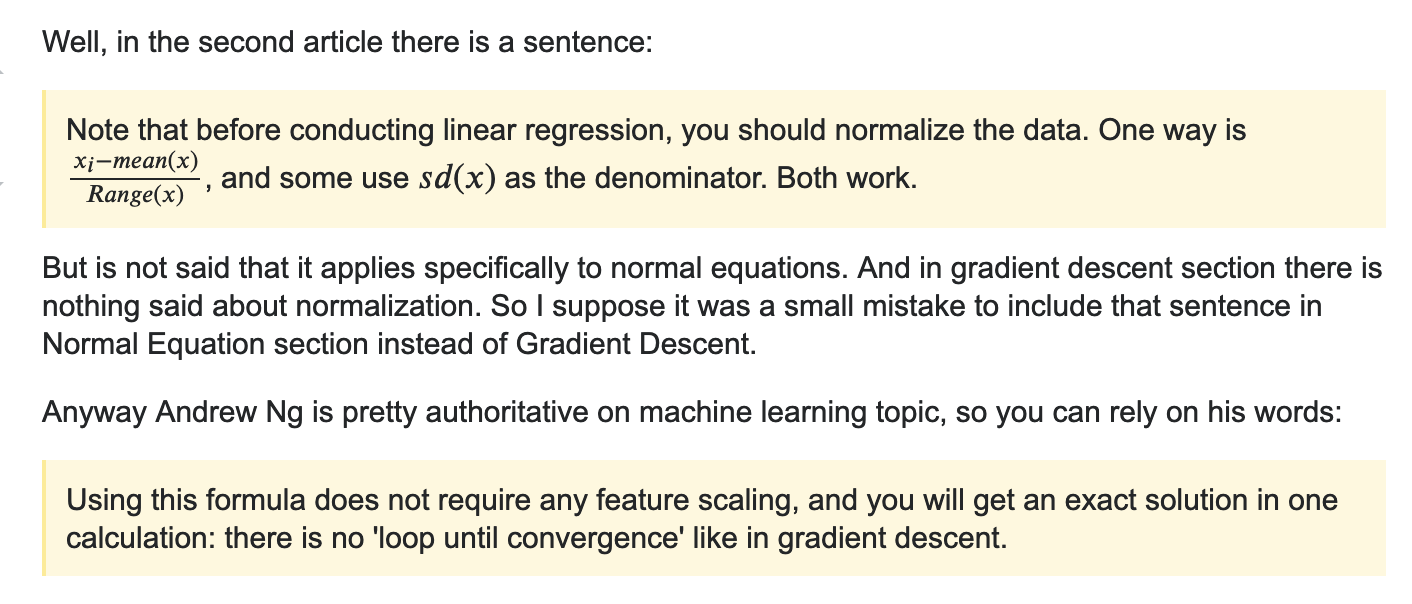
1. Hypothesis parameters (theta) for the closed form



1. Plot of predicted MPG against observed MPG for the validation data



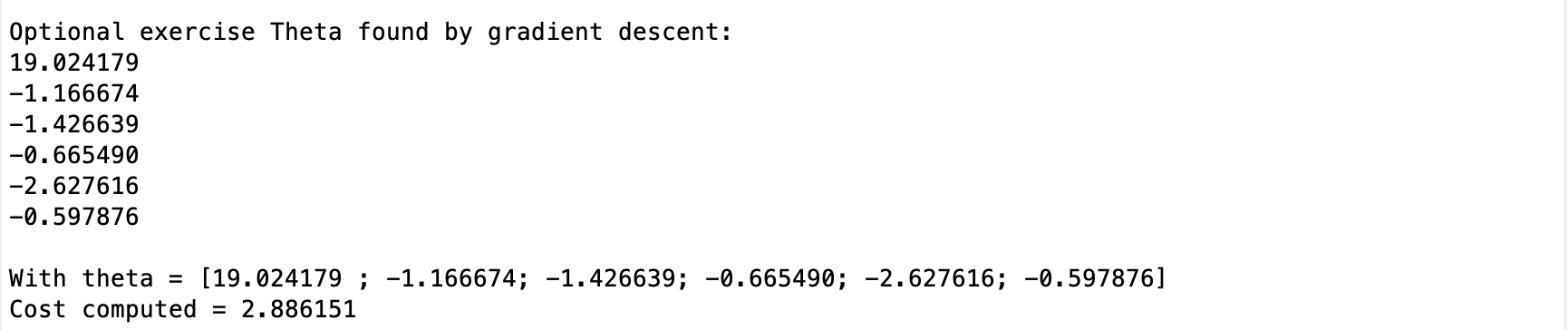
1. I didn’t apply feature scaling (normalization) because when I searched this question on Google I found:



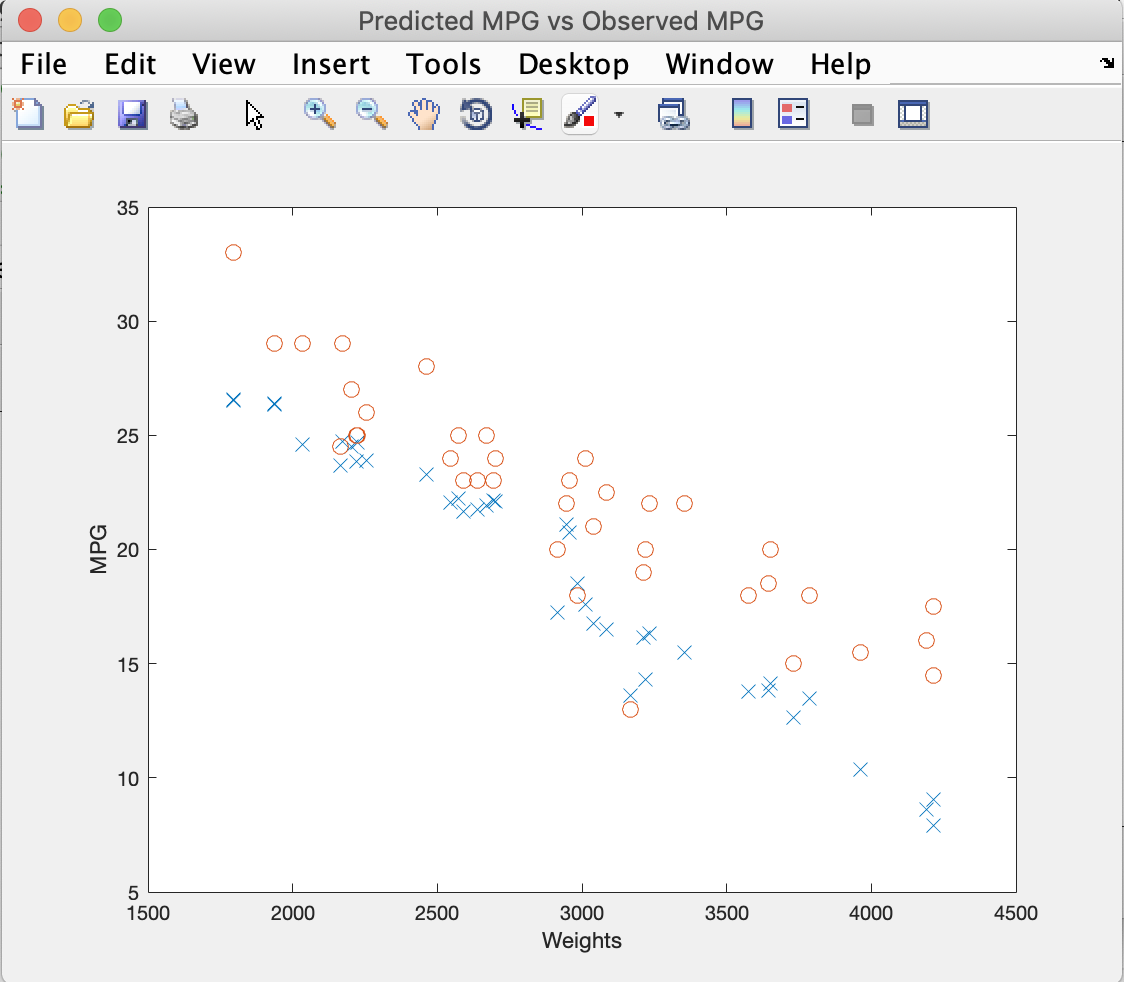
The result of normal equation is not as good as I expected, there are maybe some method to get a better result.

# Optional Exercise

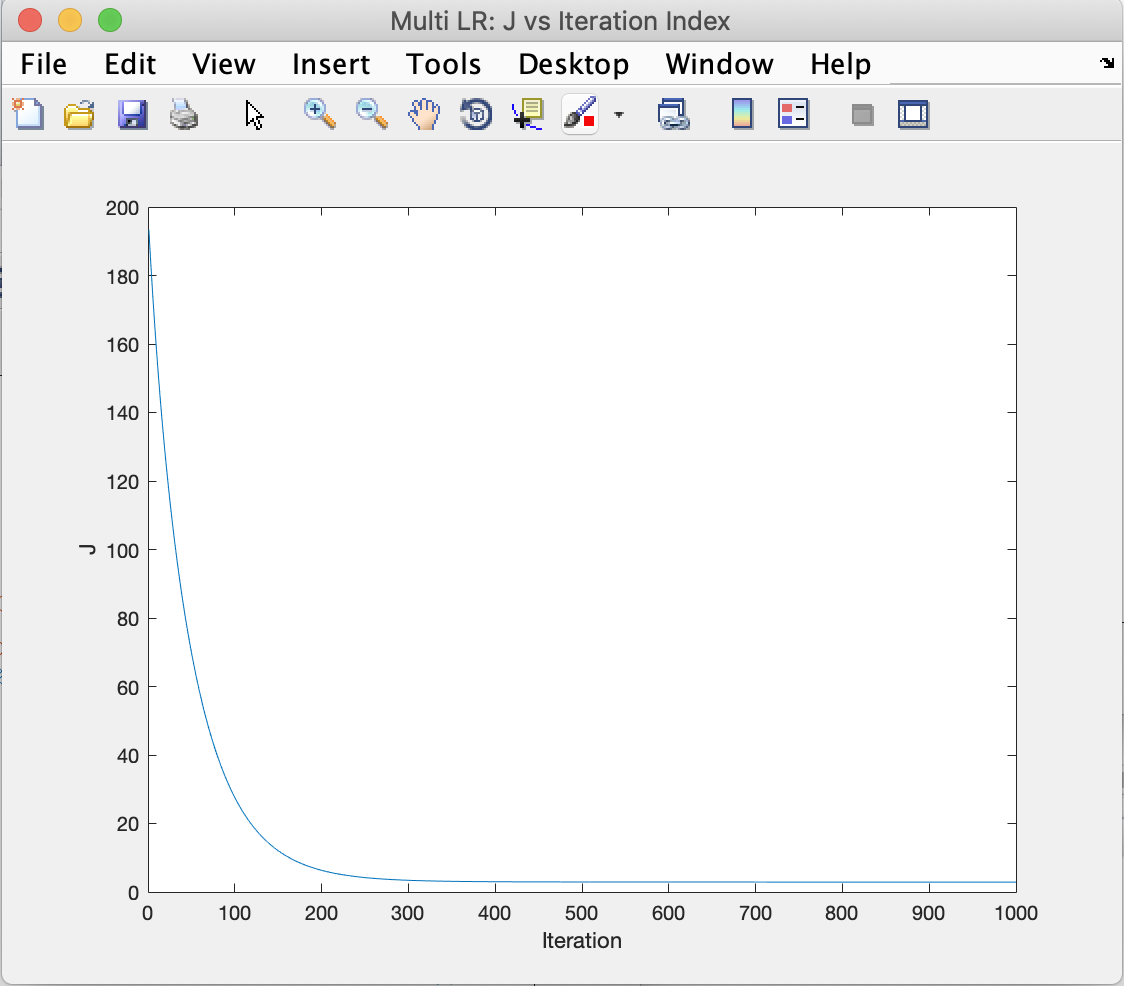
1. Minimum J and the hypothesis parameters from gradient descent code



1. Plot of predicted MPG against observed MPG for the validation data



1. Plot of J vs. Iteration index for the training data



1. The result of 5 features LR is pretty closed to the situation that with 4 features LR. I believe that we can improve the result by adding more data to the training set instead of adding new features.

# Codes

1. main.m

%% Machine Learning Homework 2

% Author: Xinrun Zhang

% Time: 02/20/2019 14?39

% =====================================================================

%% Initialization

clear ; close all; clc

% Import the data;

fprintf('Initializing...\n')

fprintf('Reading the data...\n');

A = xlsread('AutoData\_HW2.xlsx');

% Extract the data, generate training data and validation data;

x = A(1:160,2:5);

y = A(1:160,1);

x\_val = A(161:200,2:5);

y\_val = A(161:200,1);

% Plot the data;

fprintf('Visualizing the data...\n\n')

figure('Name','Raw Data','NumberTitle','off');

plot(x(:,4),y,'x','MarkerSize',8);

ylabel('MPG');

xlabel('Weights');

% Initialize the theta vector;

theta = zeros(5, 1);

% Initialize the gradient descent parameters

iteration = 1000;

alpha = 0.01;

% =====================================================================

%% Data processing

fprintf('Data processing...\n')

% Call the normalizing function;

x\_nor = normalizing(x);

x\_val\_nor = normalizing(x\_val);

% Add a column of ones to xnor;

X = [ones(160, 1), x\_nor(:,1:4)];

X\_val = [ones(40, 1), x\_val\_nor(:,1:4)];

% =====================================================================

%% Multivariate linear regression

J = computeCost(X, y, theta); %compute the cost

fprintf('\nWith theta = [0 ; 0]\nCost computed = %f\n', J);

% Running gradient descent

theta = gradientDescent(X, y, theta, iteration, alpha);

% Print the output, including new theta and J;

fprintf('\nTheta found by gradient descent:\n');

fprintf('%f\n', theta);

J = computeCost(X, y, theta);

fprintf('\nWith theta = [%f ; %f; %f; %f; %f]\nCost computed = %f\n', theta(1),theta(2),theta(3),theta(4),theta(5), J);

% =====================================================================

%% Validation

% Generate predict data

y\_prd = X\_val\*theta;

% Calculate the mean squared error err\_cv

err\_cv = immse(y\_prd,y\_val);

fprintf('The mean suqare error err\_cv is: %f\n',err\_cv);

% Plot of predicted MPG against observed MPG for the validation data

figure('Name','Predicted MPG vs Observed MPG','NumberTitle','off');

plot(x\_val(:,4),y\_prd,'x','MarkerSize',8);

ylabel('MPG');

xlabel('Weights');

hold on;

plot(x\_val(:,4),y\_val,'o','MarkerSize',8);

% =====================================================================

%% Normal equation

% Data\_processing

X\_ne = [ones(160,1), x(:,1:4)];

X\_ne\_val = [ones(40,1), x\_val(:,1:4)];

% Normal equation

theta\_ne = pinv(X\_ne)\* y;

%theta\_ne = (X'\*X)^(-1)\* X' \* y;

y\_prd\_ne = X\_ne\_val\*theta\_ne;

% Print the output, including new theta and J;

fprintf('\nTheta found by normal equation:\n');

fprintf('%f\n', theta\_ne);

J = computeCost(X\_ne, y, theta\_ne);

fprintf('\nWith theta = [%f ; %f; %f; %f; %f]\nCost computed = %f\n', theta\_ne(1),theta\_ne(2),theta\_ne(3),theta\_ne(4),theta\_ne(5), J);

% Plot of predicted MPG against observed MPG for the validation data

figure('Name','Predicted MPG vs Observed MPG 2','NumberTitle','off');

plot(x\_val(:,4),y\_prd\_ne,'x','MarkerSize',8);

ylabel('MPG');

xlabel('Weights');

hold on;

plot(x\_val(:,4),y\_val,'o','MarkerSize',8);

% =====================================================================

%% Optional exercise

% Extract the data, generate training data and validation data;

x\_oe = A(1:160,2:6);

y\_oe = A(1:160,1);

x\_oe\_val = A(161:200,2:6);

y\_oe\_val = A(161:200,1);

% Initialize the theta vector;

theta\_oe = zeros(6, 1);

% Initialize the gradient descent parameters

iteration = 1000;

alpha = 0.01;

% =====================================================================

% Data processing

x\_oe\_nor = normalizing(x\_oe);

X\_oe = [ones(160, 1), x\_oe\_nor(:,1:5)];

x\_oe\_val\_nor = normalizing(x\_oe\_val);

X\_oe\_val = [ones(40, 1), x\_oe\_val\_nor(:,1:5)];

% Gradient Descent

theta\_oe = gradientDescent(X\_oe, y, theta\_oe, iteration, alpha);

% Print the output, including new theta and J;

fprintf('\nOptional exercise Theta found by gradient descent:\n');

fprintf('%f\n', theta\_oe);

J = computeCost(X\_oe, y\_oe, theta\_oe);

fprintf('\nWith theta = [%f ; %f; %f; %f; %f; %f]\nCost computed = %f\n', theta\_oe(1),theta\_oe(2),theta\_oe(3),theta\_oe(4),theta\_oe(5), theta\_oe(6), J);

% =====================================================================

% Validation

% Generate predict data

y\_prd = X\_oe\_val\*theta\_oe;

% Plot of predicted MPG against observed MPG for the validation data

figure('Name','Predicted MPG vs Observed MPG','NumberTitle','off');

plot(x\_oe\_val(:,4),y\_prd,'x','MarkerSize',8);

ylabel('MPG');

xlabel('Weights');

hold on;

plot(x\_oe\_val(:,4),y\_val,'o','MarkerSize',8);

% =====================================================================

1. computeCost.m

function J = computeCost(X, y, theta)

m = length(y);

J = 0;

hypothesis = X \* theta;

J = (1/ (2\*m)) \* (hypothesis - y)' \* (hypothesis - y);

end

1. gradientDescent.m

function [theta, J\_history ] = gradientDescent(X, y, theta, iteration, alpha)

m = length(y);

J\_history = zeros(iteration, 1);

for i = 1:iteration

    hypothesis = X\*theta;

    theta = theta - (alpha / m) \* (X' \* (hypothesis - y));

    J\_history(i) = computeCost(X, y, theta);

end

figure('Name','Multi LR: J vs Iteration Index','NumberTitle','off');

plot(J\_history); % plot J vs iteration index

ylabel('J');

xlabel('Iteration');

end

1. normalizing.m

function x\_nor = normalizing(x)

x\_nor = x;

%delta = x\_max - x\_min;

x\_mean = mean(x);

x\_nor = (x - x\_mean)./ (std(x));

end