# COGS 109: Assignment #6

Due on Sunday, December 6, 2015

*Tu, Zhuowen 2pm*

**Kyle Lee**
A01614951

# Problem 1

(a)

```
>> W1

W1 =

    5.9123
   -3.0814
```

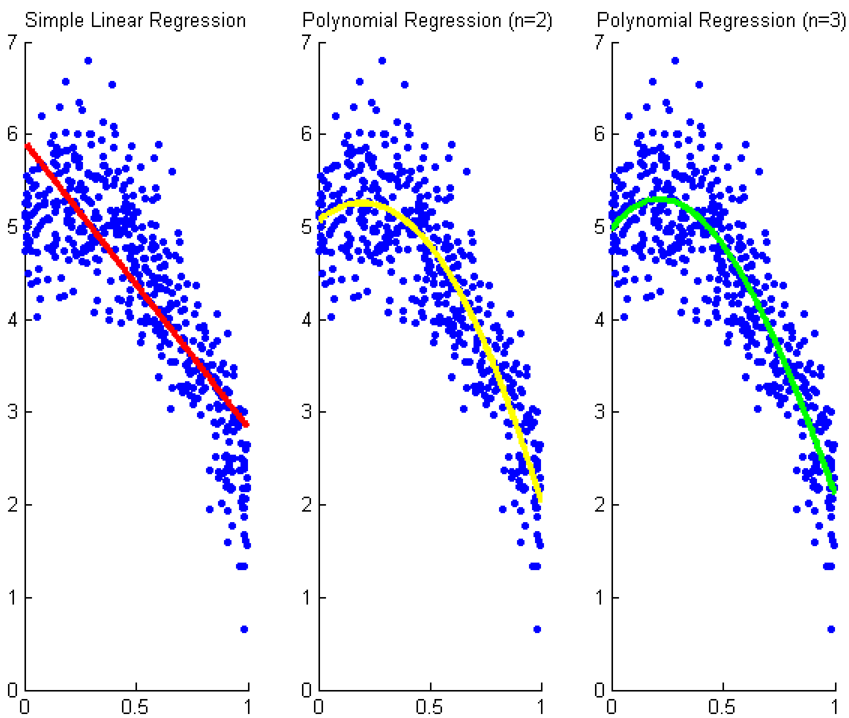
(b)

```
>> W2

W2 =

    5.0619
    1.9758
   -5.0217
```

(c)

```
>> W3

W3 =

    4.9681
    3.1689
   -8.0393
    2.0114
```

(d) Predicted regression lines given derived coefficients



(e) Training error rate

```
Etrain1 =

    0.4005


Etrain2 =

    0.2459


Etrain3 =

    0.2443
```

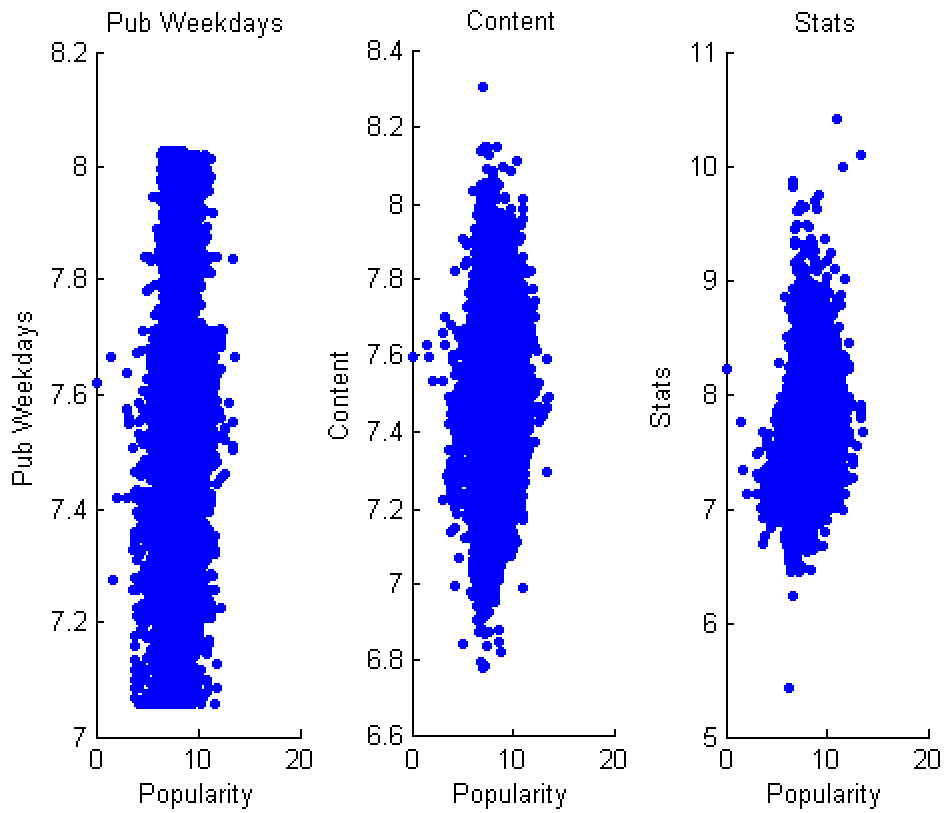
(f) Testing error rate

```
Etest1 =

    0.3793


Etest2 =

    0.7441


Etest3 =

   13.4800
```

(g) Among the training error rates, the training error that corresponds to the cubic regression demonstrates the lowest error at .2443. Among the testing error rates, the testing error that corresponds to simple linear regression demonstrates the lowest error. I would pick the simple linear regression since the absolute error is $|.4005 - .3793| = 0.0212$. We see that the absolute error of quadratic regression is $\approx .5$ while the absolute error of cubic regression is $\approx 13.2$.

## Problem 2

(a) See code for regression models



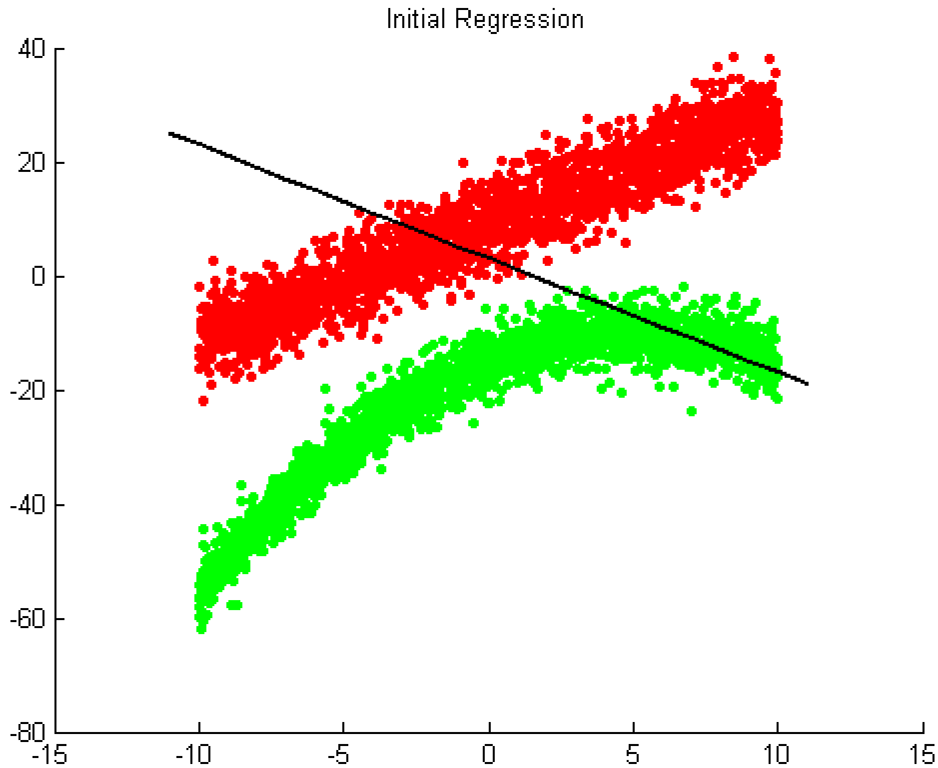

(b)

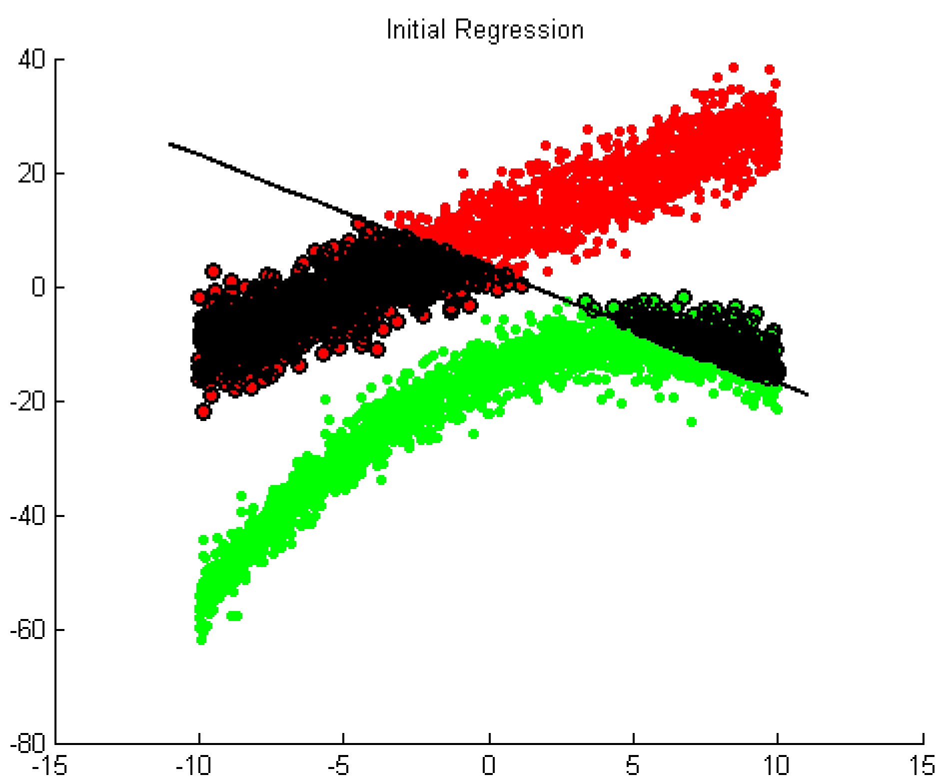
(c) I would argue that the category Stats is the best among the three to predict the online news since there is not as much variance and outlier data compared to Content and Pub Weekdays. This relates to training errors because now we can develop a better model that reduces training error during learning. As a result, our testing error is reduced when we try to apply new data onto trained data. If our data is more concentrated in a particular region for some big $n$, then our training error decreases. We observe the Content which shows that the outliers will cause our training errors to be higher compared to Stats. For Pub Weekdays, there is too much variance which makes it the worst category to predict the online news.

## Problem 3

(a) Initialize weights (see code)

(b) Scatter plot with decision boundary with initial weights (see next page)

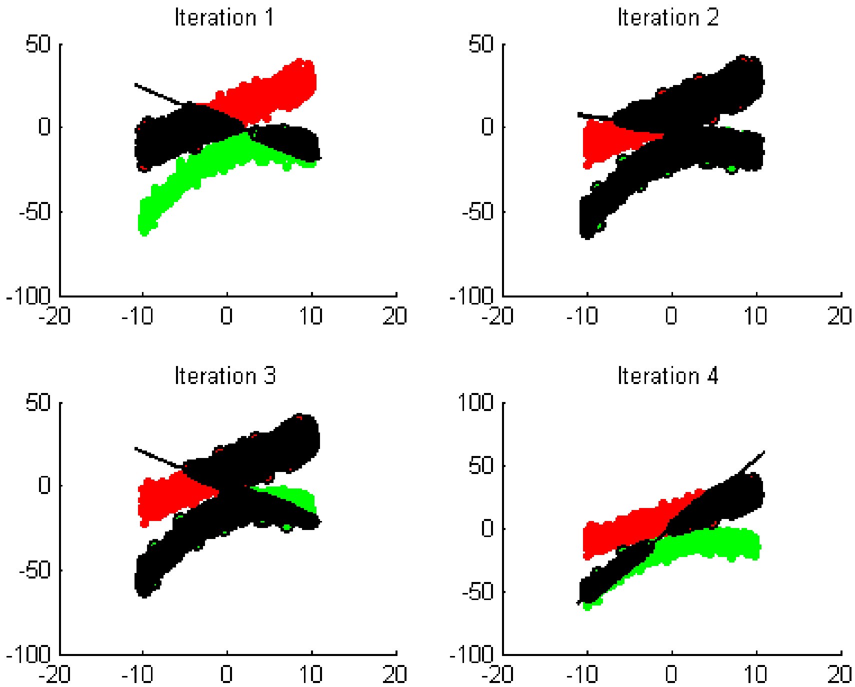

Initial Regression
40
20
0
-20
-40
-60
-80
-15
-10
-5
0
5
10
15


Initial Regression
40
20
0
-20
-40
-60
-80
-15
-10
-5
0
5
10
15

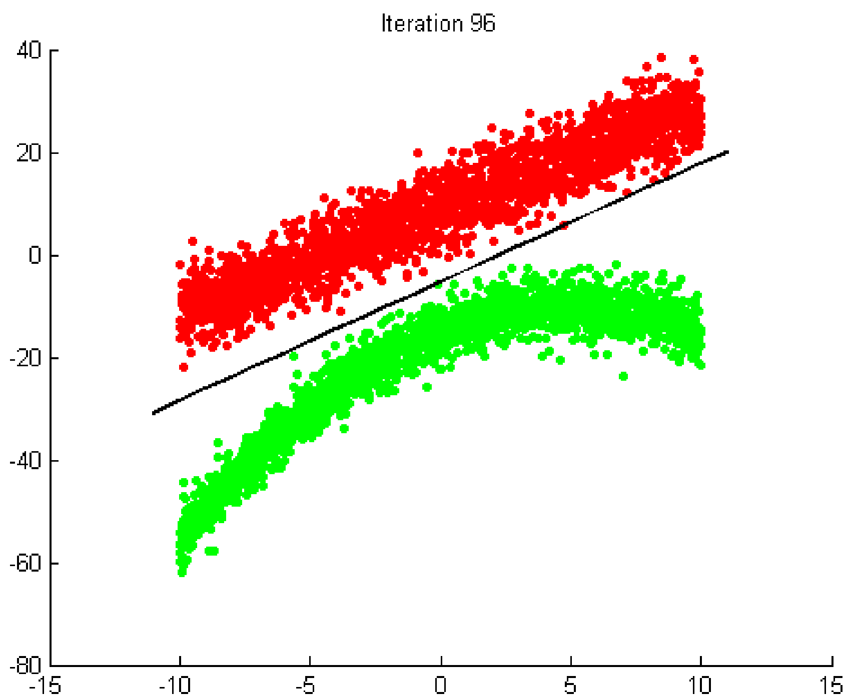
(c)

(d) Update weights (see code)

(e) $2 \times 2$ subplots with 4 iterations




(f) It takes 96 iterations (including the initial weights)



```
%%%% Homework #6 %%%%
%% QUESTION 1 %%
% PART A %
% simple linear regression
A1 = cat(2,ones(length(Xtrain),1),Xtrain);
W1 = A1\Ytrain;

% PART B %
% linear regression with quadratic term
A2 = cat(2,ones(length(Xtrain),1),Xtrain,Xtrain.^2);
W2 = A2\Ytrain;

% PART C %
% linear regression with cubic term
A3 = cat(2,ones(length(Xtrain),1),Xtrain,Xtrain.^2,Xtrain.^3);
W3 = A3\Ytrain;


% PART D %
figure

% Generate predicted regression lines given the derived coefficients.
Xpred = transpose(linspace(min(Xtrain),max(Xtrain),500));  % create an X series to draw lines

% concatenate with simple linear regression
A1pred = cat(2, ones(length(Xpred),1),Xpred);
Ypred1 = A1pred*W1;

% plot simple linear regression
subplot(1,3,1);
scatter(Xtrain,Ytrain,20,'filled');
hold on;
plot(Xpred,Ypred1,'r','LineWidth',3);
title('Simple Linear Regression')

% concatenate with linear regression with quadratic term
A2pred = cat(2, ones(length(Xpred),1),Xpred, Xpred.^2);
Ypred2 = A2pred*W2;

% plot linear regression with quadratic term
subplot(1,3,2);
scatter(Xtrain,Ytrain,20,'filled');
hold on;
plot(Xpred,Ypred2,'y','LineWidth',3);
title('Polynomial Regression (n=2)')

% concatenate with linear regression with cubic term
A3pred = cat(2, ones(length(Xpred),1),Xpred, Xpred.^2, Xpred.^3);
Ypred3 = A3pred*W3;

% plot linear regression with cubic term
subplot(1,3,3);
scatter(Xtrain,Ytrain,20,'filled');
```

```
hold on;
plot(Xpred,Ypred3,'g','LineWidth',3);
title('Polynomial Regression (n=3)')

% PART E %
% calculate training error rate
Etrain1 = mean((Ytrain - A1*W1).^2);
Etrain2 = mean((Ytrain - A2*W2).^2);
Etrain3 = mean((Ytrain - A3*W3).^2);

% PART F %
% calculate testing error erates
A1test = cat(2,ones(length(Xtest),1),Xtest);
Etest1 = mean((Ytest - A1test*W1).^2);

A2test = cat(2,ones(length(Xtest),2),Xtest);
Etest2 = mean((Ytest - A2test*W2).^2);

A3test = cat(2,ones(length(Xtest),3),Xtest);
Etest3 = mean((Ytest - A3test*W3).^2);

% PART G %
% Compare error rates
Etrain1
Etest1

Etrain2
Etest2

Etrain3
Etest3


%% QUESTION 2 %%
% PART A %
% Build multiple regression models to predict the
% popularity of online news, one for each category of features.

figure
A1Weekday = cat(2,ones(length(Pub_Weekdays),1),Pub_Weekdays);
W1Weekday = A1Weekday\Popularity;

A2Content = cat(2,ones(length(Content),1),Content);
W2Content = A2Content\Popularity;

A3Stats = cat(2,ones(length(Stats),1),Stats);
W3Stats = A3Stats\Popularity;

% PART B %
% Plot popularity versus Pub Weekday and label
subplot(1,3,1);
scatter(Popularity,A1Weekday*W1Weekday,20,'filled');
title('Pub Weekdays')
```

```
xlabel('Popularity')
ylabel('Pub Weekdays')

% Plot popularity versus Content and label
subplot(1,3,2);
scatter(Popularity,A2Content*W2Content,20,'filled');
title('Content')
xlabel('Popularity')
ylabel('Content')

% Plot popularity versus Stats and label
subplot(1,3,3);
scatter(Popularity,A3Stats*W3Stats,20,'filled');
title('Stats')
xlabel('Popularity')
ylabel('Stats')


%% QUESTION 3 %%
% PART A %
figure
% Initialize weights
w1 = 2;
w2 = 1;
b = -3;

% PART B %
% draw scatter plots of data points with labels from target
scatter(x1(target==-1),x2(target==-1),10,'g','filled');
hold on

% overlay scatter plot with decision boundary with initial weights
scatter(x1(target==1),x2(target==1),10,'r','filled');
x_test = -11:11; %define an arbitrary x sequence for drawing the line
y_test = (-w1*x_test-b)/w2;
plot(x_test,y_test,'k','linewidth',2);
title('Initial Regression');

% PART C %
figure
% draw scatter plots of data points with labels from target
scatter(x1(target==-1),x2(target==-1),10,'g','filled');
hold on

% overlay scatter plot with decision boundary with initial weights
scatter(x1(target==1),x2(target==1),10,'r','filled');
x_test = -11:11; %define an arbitrary x sequence for drawing the line
y_test = (-w1*x_test-b)/w2;
plot(x_test,y_test,'k','linewidth',2);
title('Initial Regression');

err_id=[];
N = length(x1)
```

```
for i = 1:N %loop through all points
     net=w1*x1(i)+w2*x2(i)+b;
     if net>=0 %set output to 1 if net >=0
        output(i) = 1;
     else %set output to -1 if net <0
       output(i) = -1;
    end

    % Determine error indices, if any
     if output(i)==target(i)
        incorrect(i) = 0;
    else
        incorrect(i) = 1;
        err_id=[err_id i]; %add index of index of incorrect output to err_id
     end
end

% if there are errors, classify them
if err_id >0
    scatter(x1(target==-1),x2(target==-1),10,'g','filled');
    hold on
     scatter(x1(target==1),x2(target==1),10,'r','filled');
    x_test = -11:11; %define an arbitrary x sequence for drawing the line
    y_test = (-w1*x_test-b)/w2;
    plot(x_test,y_test,'k','linewidth',2);
    scatter(x1(err_id),x2(err_id) ,50,'k','linewidth',2);
end

% PART D %
% update the weights for decision boundary given error
w1=w1+(target(err_id(1))-output(err_id(1)))*x1(err_id(1));
w2=w2+(target(err_id(1))-output(err_id(1)))*x2(err_id(1));
b = b+(target(err_id(1))-output(err_id(1)));

% PART E %
% Go through 4 iterations and label iteration
figure
w1 = 2;
w2 = 1;
b = -3;

for j=1:4
    % subplot on j^th entry
    subplot(2,2,j)

    scatter(x1(target==-1),x2(target==-1),10,'g','filled');
     hold on
     scatter(x1(target==1),x2(target==1),10,'r','filled');
    x_test = -11:11; %define an arbitrary x sequence for drawing the line
    y_test = (-w1*x_test-b)/w2;
    plot(x_test,y_test,'k','linewidth',2);
```

```
    err_id=[];
    N = length(x1);
    for i = 1:N %loop through all points
        net=w1*x1(i)+w2*x2(i)+b;
        if net>=0 %set output to 1 if net >=0
            output(i) = 1;
        else %set output to -1 if net <0
           output(i) = -1;
        end

        if output(i)==target(i)
            incorrect(i) = 0;
        else
            incorrect(i) = 1;
            err_id=[err_id i]; %add index of index of incorrect output to err_id
        end
    end

    % if there are errors, show errors on the graph
    if err_id >0
        scatter(x1(target==-1),x2(target==-1),10,'g','filled');
        hold on
        scatter(x1(target==1),x2(target==1),10,'r','filled');
        x_test = -11:11; %define an arbitrary x sequence for drawing the line
        y_test = (-w1*x_test-b)/w2;
        plot(x_test,y_test,'k','linewidth',2);
        scatter(x1(err_id),x2(err_id) ,50,'k','linewidth',2);
    end

    % update the weights for decision boundary given the error
    w1=w1+(target(err_id(1))-output(err_id(1)))*x1(err_id(1));
    w2=w2+(target(err_id(1))-output(err_id(1)))*x2(err_id(1));
    b = b+(target(err_id(1))-output(err_id(1)));

    % print iteration
    title(sprintf('Iteration %d', j));
end

%% Part F %%
% Complete perceptron algorithm and print out iteration
figure
w1 = 2;
w2 = 1;
b = -3;
iter = 1;

% Loop until break
while 0<1
    % initialize empty array and count x1
    err_id=[];
    N = length(x1);

    for i = 1:N %loop through all points
```

```
        net=w1*x1(i)+w2*x2(i)+b;
        if net>=0 %set output to 1 if net >=0
            output(i) = 1;
        else %set output to -1 if net <0
           output(i) = -1;
        end
        if output(i)==target(i)
            incorrect(i) = 0;
        else
            incorrect(i) = 1;
            err_id=[err_id i]; %add index of index of incorrect output to err_id
        end
    end

    % if there are no errors, then print out line and current iteration
    if incorrect == 0
        scatter(x1(target==-1),x2(target==-1),10,'g','filled');
        hold on
        scatter(x1(target==1),x2(target==1),10,'r','filled');
        x_test = -11:11; %define an arbitrary x sequence for drawing the line
        y_test = (-w1*x_test-b)/w2;
        plot(x_test,y_test,'k','linewidth',2);

        % print out iteration count and break out of while loop
        title(sprintf('Iteration %d', iter))
        break
    end

    % update the weights for decision boundary given error
    w1=w1+(target(err_id(1))-output(err_id(1)))*x1(err_id(1));
    w2=w2+(target(err_id(1))-output(err_id(1)))*x2(err_id(1));
    b = b+(target(err_id(1))-output(err_id(1)));

    % update iteration
    iter = iter + 1;
end
% print iteration count
iter
```