Machine Learning (CS696)

Fall 2017

Assignment 2

Submitted by

Ashok Kumar Shrestha
#A25265227

Training Data:

Student	Hours studied - x	Result (0 – fail, 1 – pass) - y
1	0.5	0
2	0.75	0
3	1.00	0
4	1.25	0
5	1.50	0
6	1.75	0
7	1.75	1
8	2.00	0
9	2.25	1
10	2.50	0
11	2.75	1
12	3.00	0
13	3.25	1
14	3.50	0
15	4.00	1
16	4.25	1
17	4.50	1
18	4.75	1
19	5.00	1
20	5.50	1

1. Linear Regression:

Code: Program executed in MATLAB

```
function j_theta = logistic_cost(x,y,m,theta)
   %hypothesis
   h = logistic_hypothesis(x,theta);

   %calculate the cost function
   j_theta = -1/m * sum(y .* log(h) + (1 - y) .* log(1 - h));

end
function [theta, j_list] = linear_gradient_descent(x,y,m,theta,alpha,iterations)
   %preparing j_list which contains list of j_theta cost function values
   j_list = zeros(iterations,1);
```

```
    for i = 1:iterations
        %hypothesis
        h = x*theta;

        theta1 = theta(1) - alpha * (sum((h - y) .* x(:,1))) / m;
        theta2 = theta(2) - alpha * (sum((h - y) .* x(:,2))) / m;
        theta(1) = theta1;
        theta(2) = theta2;

        j_list(i) = linear_cost(x,y,m,theta);
    end
end

function accuracy = linear_prediction(x,y,theta)
    %prediction from linear regression
    h = round(x * theta);

    %checking predicted value with actual value for accuracy
    accuracy = mean(h == y)*100;

end
```

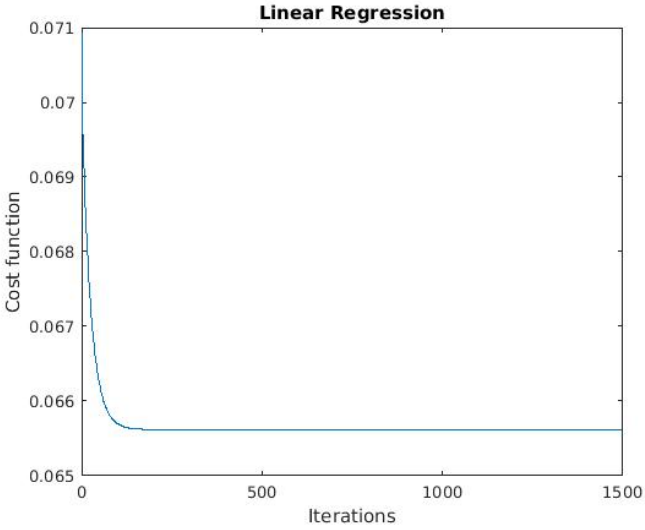
Main program:

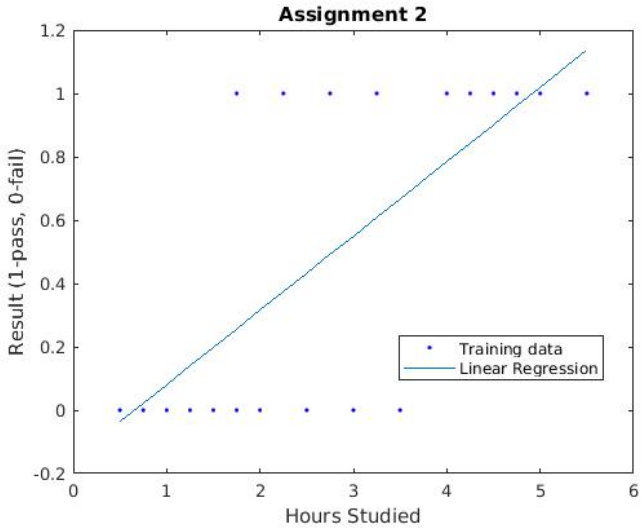
```
clear;
clc;

%read the trainig data
data = load('training_data.txt');

%initializing the variables
x = data(:,1);   %features
y = data(:,2);   %actual result
alpha = 0.1;   %learning rate
m = length(y);   %no. of training data
iterations = 1500;   %no. of iterations for computing gradient descent
theta = zeros(2,1);  %intial parameters

%plot the data
plot(x,y,'b.','MarkerSize',6)
title('Assignment 2')
xlabel('Hours Studied')
ylabel('Result (1-pass, 0-fail)')
%add column ones to X for x0
x = [ones(m,1), x];

%print alpha, intial theta and iterations
fprintf('Alpha: %d\n',alpha)
fprintf('Iterations: %d\n', iterations)
fprintf('Initial Theta-1: %d\n', theta(1))
fprintf('Initial Theta-2: %d\n\n', theta(2))
```

```
%compute linear gradient descent
[linear_theta, linear_j_list] = linear_gradient_descent(x,y,m,theta,alpha,iterations);
fprintf('Linear Regression:\n')
fprintf('Theta-1: %d\nTheta-2: %d\n',linear_theta(1), linear_theta(2))
fprintf('Hypothesis: y = %d + %d * x1\n',linear_theta(1), linear_theta(2))

%Compute Linear regression Accuracy
linear_accuracy = linear_prediction(x,y,linear_theta);
fprintf('Accuracy: %d %%\n',linear_accuracy)

%plotting linear regression line
hold on;
plot(x(:,2),x*linear_theta,'-')

legend('Training data','Linear Regression');
hold off;

%plot cost function vs iterations
plot(1:iterations, linear_j_list, '-')
title('Linear Regression')
xlabel('Iterations')
ylabel('Cost function')
```

Output:

```
Alpha: 1.000000e-01
Iterations: 1500
Initial Theta-1: 0
Initial Theta-2: 0

Linear Regression:
Theta-1: -1.539353e-01
Theta-2: 2.345956e-01
Hypothesis: y = -1.539353e-01 + 2.345956e-01 * x1
Accuracy: 75 %
```





2. Logistic Regression:

Code: Program executed in MATLAB

```
function h = logistic_hypothesis(x, theta)
   z = x * theta;
   h = 1./(1 + exp(-z));
end

function j_theta = logistic_cost(x,y,m,theta)
   %hypothesis
   h = logistic_hypothesis(x,theta);

   %calculate the cost function
   j_theta = -1/m * sum(y .* log(h) + (1 - y) .* log(1 - h));
end

function [theta, j_list] = logistic_gradient_descent(x,y,m,theta,alpha,iterations)
   %preparing j_list which contains list of j_theta cost function values
   j_list = zeros(iterations,1);

   for i = 1:iterations
      %hypothesis
      h = logistic_hypothesis(x,theta);

      theta1 = theta(1) - alpha * sum((h - y) .* x(:,1)) / m;
      theta2 = theta(2) - alpha * sum((h - y) .* x(:,2)) / m;

      theta(1) = theta1;
      theta(2) = theta2;

      j_list(i) = logistic_cost(x,y,m,theta);
    end
end

function accuracy = logistic_prediction(x,y,theta)
   %prediction from linear regression
   h = round(logistic_hypothesis(x, theta));

   %checking predicted value with actual value for accuracy
   accuracy = mean(h == y)*100;
end
```

Main program:

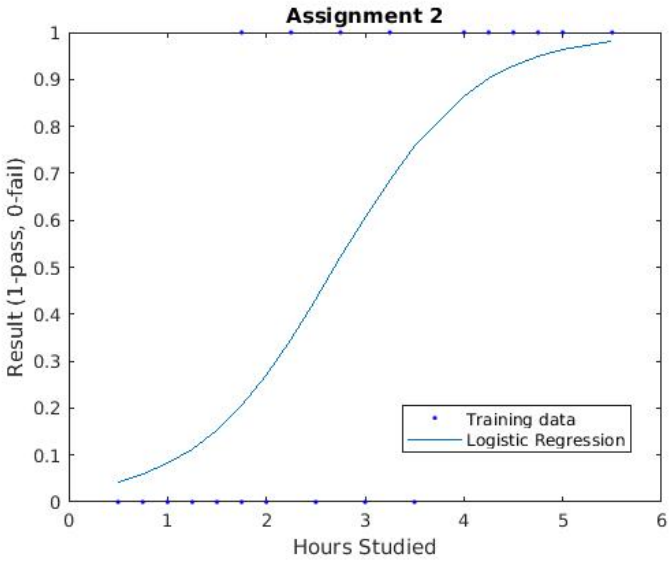
```
clear;
clc;

%read the trainig data
data = load('training_data.txt');
```

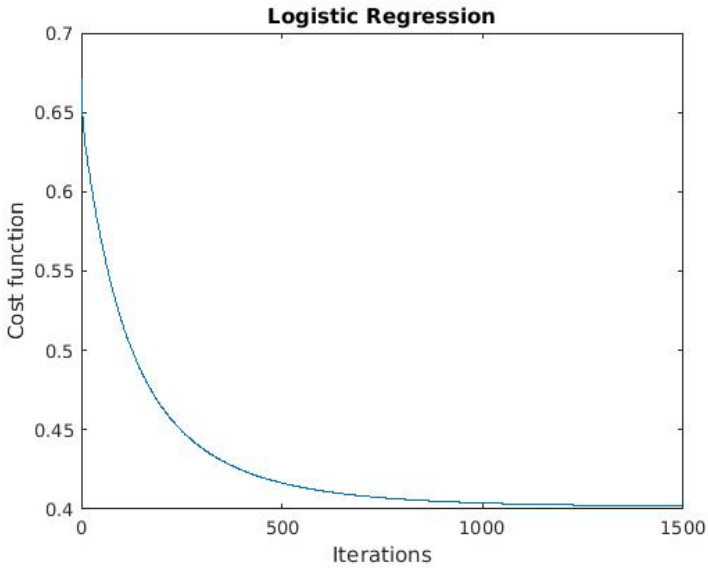
```
%initializing the variables
x = data(:,1);   %features
y = data(:,2);   %actual result
alpha = 0.1;   %learning rate
m = length(y);   %no. of training data
iterations = 1500;   %no. of iterations for computing gradient descent
theta = zeros(2,1);  %intial parameters

%plot the data
plot(x,y,'b.','MarkerSize',6)
title('Assignment 2')
xlabel('Hours Studied')
ylabel('Result (1-pass, 0-fail)')

%add column ones to X for x0
x = [ones(m,1), x];

%print alpha, intial theta and iterations
fprintf('Alpha: %d\n',alpha)
fprintf('Iterations: %d\n', iterations)
fprintf('Initial Theta-1: %d\n', theta(1))
fprintf('Initial Theta-2: %d\n\n', theta(2))

%compute logistic gradient descent
[logistic_theta, logistic_j_list] = logistic_gradient_descent(x,y,m,theta,alpha,iterations);
fprintf('\nLogistic Regression:\n')
fprintf('Theta-1: %d\nTheta-2: %d\n',logistic_theta(1), logistic_theta(2))
fprintf('Hypothesis: y = 1/(1 + exp(%d + %d * x1))\n',logistic_theta(1), logistic_theta(2))

%Compute Logistic regression accuracy
logistic_accuracy = logistic_prediction(x,y,logistic_theta);
fprintf('Accuracy: %d %%\n',logistic_accuracy)

%plotting logistic regression line
hold on;
plot(x(:,2),logistic_hypothesis(x,logistic_theta),'-');
legend('Training data','Logistic Regression');
hold off;

%plot cost function vs iterations
plot(1:iterations, logistic_j_list, '-')
title('Logistic Regression')
xlabel('Iterations')
ylabel('Cost function')
```

Output:

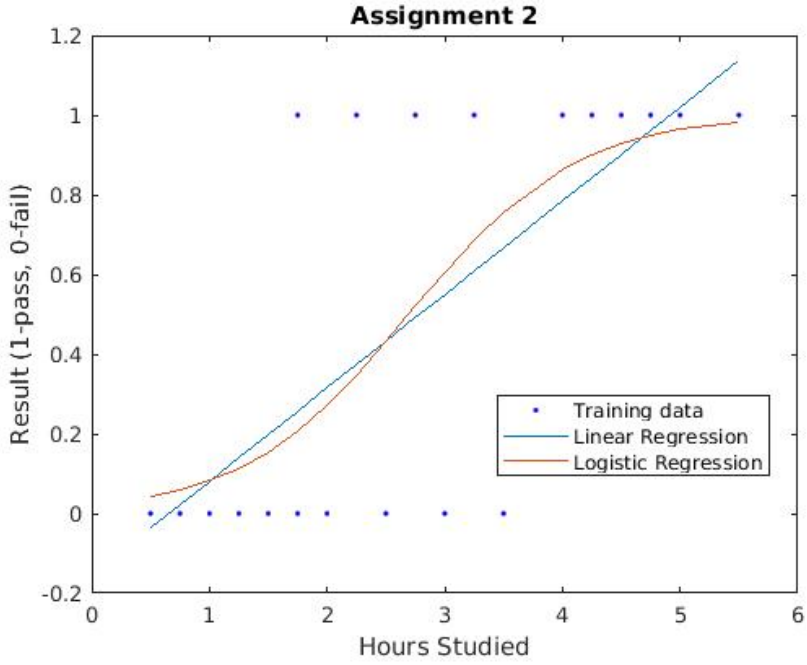
```
Alpha: 1.000000e-01
Iterations: 1500
Initial Theta-1: 0
Initial Theta-2: 0
```

Logistic Regression:
Theta-1: -3.839488e+00
Theta-2: 1.424188e+00
Hypothesis: y = 1/(1 + exp(-3.839488e+00 + 1.424188e+00 * x1))
Accuracy: 80 %





3. Plotting both hypothesis function $0 < x < 6$.



Observation:

Linear regression and the logistic regression are both supervised learning. In linear regression, the outcome or predicted value is continues and can have infinite possible values (– ∞ to + ∞). Linear regression can be used to predict the values/number such as to predict the percent score of the student for particular course. Whereas in logistic regression, the outcome is limited possible values and gives the probability of the value. For example, we can use logistic regression to predict if the student pass or fail based on the no. of hours he/she studied. Though linear regression is best for calculating the value based on given training data sets, it can also be used for classification problem but it is not as efficient as logistic regression. In linear regression we can classify the predicted value based on the predefined threshold value/parameters. Such as, if the predicted value is >0.5 we classify the student to pass else fail. Though this method may work for certain training data sets, we can not guarantee its success if the training data sets are not continues. Even for the single training data which is way off the grid in data sets can set major error setback in linear regression. For example, in given problem if one data was (10,1) that is one student studies 10 hours and passes, then our linear regression graph/plot will have less slope which in-turn reduces its classification accuracy. In this scenario, we can define our threshold value to be >0.2 for improving accuracy but this kind of procedure is not favored for classification in machine learning. However, if same data was given for logistic regression, it would not much affect the accuracy.

4. Algorithm:

For y = 1:
$j\_theta = 100*\Sigma_{i=1}^{m} (y_i*(1 – h(\theta)_i))$
Cost function goes from 100 to 0 linearly as hypothesis function goes from 0 to 1

for y = 0
$j\_theta = 100*\Sigma_{i=1}^{m} ((1 – y_i) * h(\theta)_i)$
Cost function goes from 0 to 100 linearly as hypothesis function goes from 0 to 1

So, total cost function is:
$J(\theta) = 100*\Sigma_{i=1}^{m} (y_i*(1 – h(\theta)_i) + (1 – y_i) * h(\theta)_i) / m$

Similarly, gradient descent is obtained from partial derivation of cost function as below:

$\theta_j = \theta_j – 100 * \alpha * \Sigma_{i=1}^{m} ((1 - 2*y_i)*x_{ij}) / m$

Here, we use sigmoid function for our hypothesis.

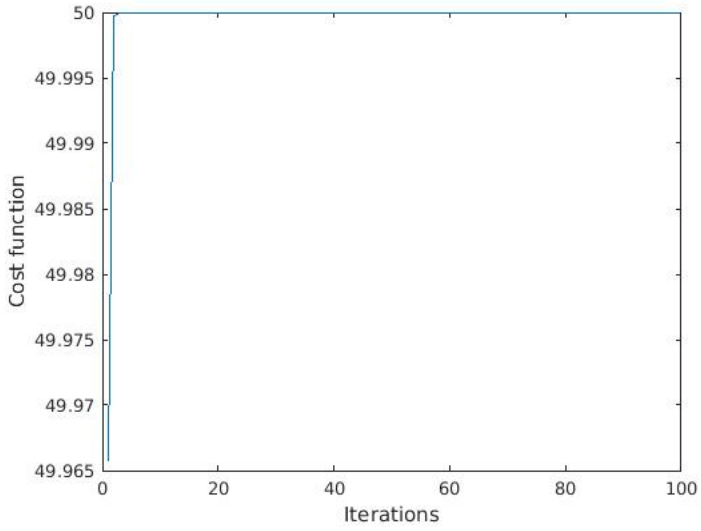
$h_\theta (x) = g(\theta^t X)$, where and $g(z) = 1/(1 + e^{-z})$

Code:

```
function h = logistic_hypothesis(x, theta)
   z = x * theta;
   h = 1./(1 + exp(-z));
end

function j_theta = llogistic_cost(x,y,m,theta)
   %hypothesis
```

```
    h = logistic_hypothesis(x,theta);

    %calculate the cost function
    j_theta = 100./(m) * sum((1-y) .* (h) + y .* (1 - h));
end

function [theta, j_list] = llogistic_gradient_descent(x,y,m,theta,alpha,iterations)
    %preparing j_list which contains list of j_theta cost function values
    j_list = zeros(iterations,1);

    for i = 1:iterations
        %hypothesis
        h = logistic_hypothesis(x,theta);

        theta1 = theta(1) - 100 * alpha * sum((1 - 2*y) .* (x(:,1))) / m;
        theta2 = theta(2) - 100 * alpha * sum((1 - 2*y) .* (x(:,2))) / m;

        theta(1) = theta1;
        theta(2) = theta2;

        j_list(i) = llogistic_cost(x,y,m,theta);
     end
end
```

Output:



Observation:

The standard logistic algorithm produces sigmoid function as hypothesis and the cost function is curve with one optimal minimum point where convergence occurs. In above algorithm, the cost function varies linearly. The maximum value of cost function is 100 and minimum is 0. Here convergence occurs at less no. of iterations(6) with alpha 0.01. The cost function value initially decreases linearly till convergence point is reached and then increases afterwards and remain constant after certain no. of iterations. As observed this algorithm is not as efficient as logistic algorithm.