Assignment 2 Machine Learning Course

Farhad Mohammad Kazemi

1.a)

Actually I implemented Batch Gradient Descent and Stochastic Gradient Descent for comparison them using matlab.

```
%%%%%%%%%%%%          Farhad Mohammad Kazemi       %%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%           Assignment 2.1         %%%%%%%%%%%%%%%%%%
close all
clc
clear all;
load('A2T1.mat');
x=A2T1(:,1:2);
y=A2T1(:,3);

m = length(y); % store the number of training examples
n = size(x,2); % number of features
theta_batch_vec = [0 0]';
theta_stoch_vec = [0 0]';
alpha = 0.002;
err = [0 0]';
%theta_batch_vec_v = zeros(10000,2);
theta_batch_vec_v = zeros(5000,2);
theta_stoch_vec_v = zeros(500*5000,2);
for kk = 1:5000
    % batch gradient descent - loop over all training set
    h_theta_batch = (x*theta_batch_vec);
    h_theta_batch_v = h_theta_batch*ones(1,n);
    y_v = y*ones(1,n);
    theta_batch_vec = theta_batch_vec - alpha*1/m*sum((h_theta_batch_v - 
y_v).*x).';
    theta_batch_vec_v(kk,:) = theta_batch_vec;
    j_theta_batch(kk) = 1/(2*m)*sum((h_theta_batch - y).^2);

    % stochastic gradient descent - loop over one training set at a time
    for (jj = 1:500)
        h_theta_stoch = (x(jj,:)*theta_stoch_vec);
        h_theta_stoch_v = h_theta_stoch*ones(1,n);
        y_v = y(jj,:)*ones(1,n);
        theta_stoch_vec = theta_stoch_vec - alpha*1/m*((h_theta_stoch_v - 
y_v).*x(jj,:)).';
        j_theta_stoch(kk) = 1/(2*m)*sum((h_theta_stoch - y).^2);
        theta_stoch_vec_v(500*(kk-1)+jj,:) = theta_stoch_vec;
    end
end
figure;
j_theta_stoch10epoch=[];
for (f=1:10:5000)
```

```
j_theta_stoch10epoch=[j_theta_stoch10epoch,j_theta_stoch(1,f)];
end
plot(1:10:5000,j_theta_stoch10epoch);
xlabel('epochs');
ylabel('J(theta)');
title(sprintf('Stochastic Gradient Descent'));


figure;
plot(j_theta_batch);
xlabel('epochs');
ylabel('J(theta)');
title(sprintf('Batch Gradient Descent'));
```

1.b)

1) Starts with some initial *θ*
***2)* Repeatedly changes *θ* to make *J(θ)* smaller until (hopefully) converges into a value of *θ* that**

minimizes *J(θ)*

Actually,

in the **Batch Gradient Descent**, the parameter vector θ is updated as,

$$\theta_i := \theta_i - \alpha \sum_{j=1}^{m} \left[h_\theta(x^j) - y^j \right] x_i^j.$$

(loop over all elements of training set in one iteration)

For **Stochastic Gradient Descent**, the vector gets updated as, at each iteration the algorithm goes over only one among j^{th} training set, i.e.

$for\ j\ = 1\ to\ m:$

$$\theta_i := \theta_i - \alpha \left[h_\theta(x^j) - y^j \right] x_i^j.$$

When the training set is large, Stochastic Gradient Descent can be useful (as we need not go over the full data to get the first set of the parameter vector θ)

For this dataset, we can see that both batch and stochastic gradient descent converged to reasonably close values.

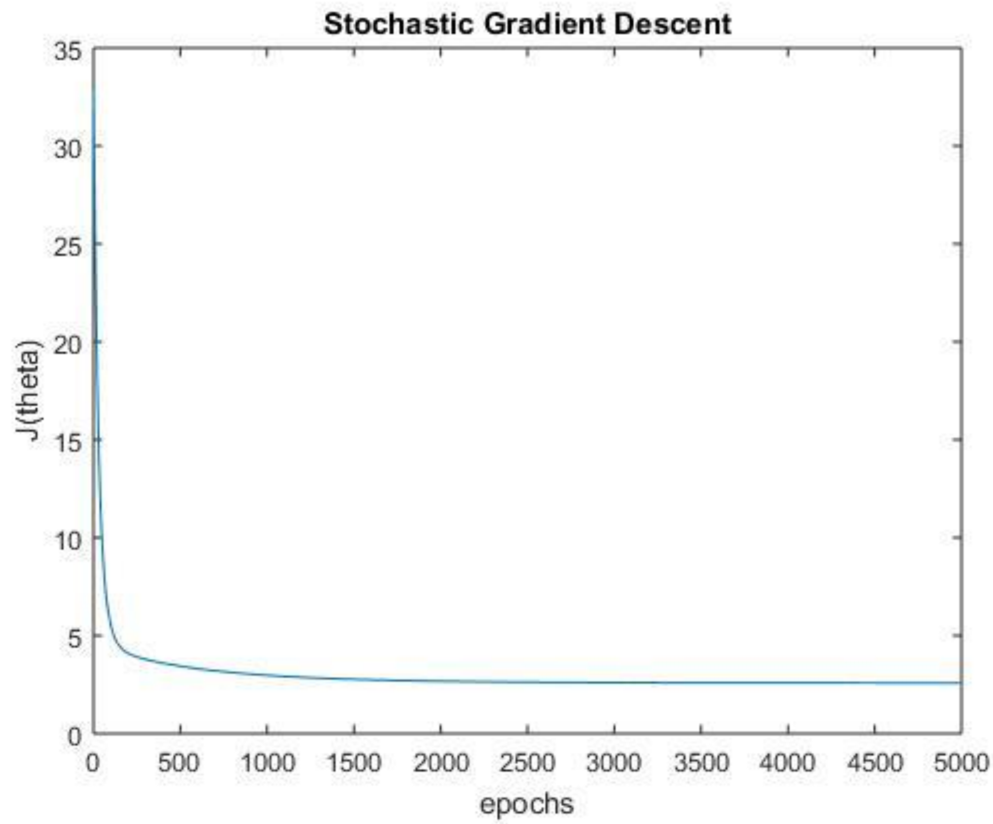
Source Code

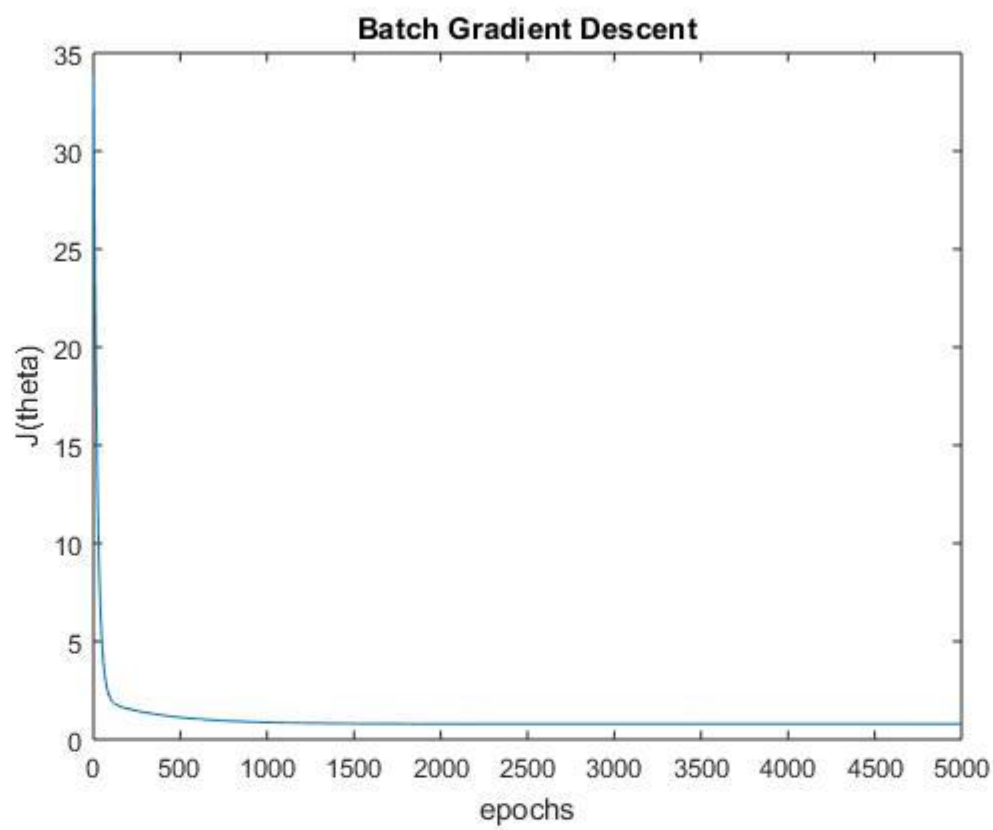
```
alpha = 0.002;
%theta_batch_vec_v = zeros(10000,2);
theta_batch_vec_v = zeros(5000,2);
theta_stoch_vec_v = zeros(500*5000,2);
for kk = 1:5000
    % stochastic gradient descent - loop over one training set at a time
    for (jj = 1:500)
        h_theta_stoch = (x(jj,:)*theta_stoch_vec);
        h_theta_stoch_v = h_theta_stoch*ones(1,n);
        y_v = y(jj,:)*ones(1,n);
        theta_stoch_vec = theta_stoch_vec - alpha*1/m*((h_theta_stoch_v -
y_v).*x(jj,:)).';
        j_theta_stoch(kk) = 1/(2*m)*sum((h_theta_stoch - y).^2);
        theta_stoch_vec_v(500*(kk-1)+jj,:) = theta_stoch_vec;
    end
end
```

1.c)

Plotting the variation of $J(\theta)$ for different values of θ

As you can see, Batch Gradient Descent converged earlier in comparison to Stochastic Gradient Descent for 5000 epoch.


Stochastic Gradient Descent
J(theta)
epochs
35
30
25
20
15
10
5
0
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000


Batch Gradient Descent
J(theta)
epochs
35
30
25
20
15
10
5
0
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000

1.d) The final values I found for the theta after 5000 epoch.

As you see, the quantities of theta that we reached for **Stochastic Gradient Descent**

theta_stoch_vec (in the source code)

theta0=1.66441078748969

theta1=-2.51585976441818

And for **Batch Gradient Descent**

theta_batch_vec (in the source code)

theta0=1.66451097780882

theta1=-2.51591109111926