Ralph Mouawad – ID 202204667

ENMG 616 – Advanced Optimization Techniques & Algorithms

A copy of the codes that I wrote on MATLAB. For the formal homework assignment, please refer to the other PDF called: 'ENMG616_Assignment3_RalphMouawad'.

Copy of the MATLAB CODES:

```
% Ralph Mouawad - HW 3 - ENMG 616 - ID 202204667
% Logistic Regression on a real Dataset

%% question 1 Importing the datasets -
Xtrain = importdata('Xtrain.mat');
Xtest = importdata('Xtest.mat');
Ytrain = importdata('Ytrain.mat');
Ytest = importdata('Ytest.mat');

%% question 2 Normalizing features of data -
Xtrain_normal = Xtrain;
Xtest_normal = Xtest;
for i = 1:30
    a = max(Xtrain_normal(:,i));
    b = min(Xtrain_normal(:,i));
    for j = 1:length(Xtrain_normal)
        Xtrain_normal(j,i) = (Xtrain_normal(j,i) - b)/ (a - b);
    end
end
for i = 1:30
    a = max(Xtest_normal(:,i));
    b = min(Xtest_normal(:,i));
    for j = 1:length(Xtest_normal)
        Xtest_normal(j,i) = (Xtest_normal(j,i) - b)/ (a - b);
    end
end
min(Xtrain_normal); max(Xtrain_normal); % making sure all my values are between 0 & 1
min(Xtest_normal); max(Xtest_normal);

%% question 3 Classical Gradient Descent with constant step size -
n = 30; % number of features in X
w = zeros(n,1);
b = rand(1,1);
lambda = 0.001;
alpha = 0.001;
nb_iterations = 1000;
nb_rows = length(Xtrain);
nb_rows_test = length(Xtest);
```

```
accuracy_training = zeros(nb_iterations, 1);
accuracy_testing = zeros(nb_iterations,1);
for i = 1:nb_iterations
    dw = 0;
    db = 0;
    for j = 1:nb_rows
        dw = dw + (-Xtrain_normal(j,:) * Ytrain(j))' +
(Xtrain_normal(j,:)*exp(Xtrain_normal(j,:)*w +b)/(1+exp(Xtrain_normal(j,:)*w +b)))';
        db = db + (-Ytrain(j) + (exp(Xtrain_normal(j,:)*w
+b)/(1+exp(Xtrain_normal(j,:)*w +b))));
    end
    % update w and b
    dw = dw + lambda*w;
    w = w - alpha*dw;
    b = b - alpha*db;
    prediction_training = (1 ./( 1 + exp(-(Xtrain_normal*w + b))) > 0.5);
    accuracy_training(i) = sum(prediction_training==Ytrain)/nb_rows; % see how the
accuracy will change at each iteration
    prediction_testing = (1 ./( 1 + exp(-(Xtest_normal*w + b))) > 0.5);
    accuracy_testing(i) = sum(prediction_testing==Ytest)/nb_rows_test;
end
accuracy_training(1000) % check the accuracy at the last iteration
accuracy_testing(1000)
% loss function
Loss = 0;
for i = 1:nb_rows_test
    Loss = Loss - Ytest(i)*(Xtest_normal(i,:)*w + b) + log (1 +
exp(Xtest_normal(i,:)*w + b));
end
Loss = Loss + (lambda/2)*norm(w)^2;
Loss/nb_rows_test;

 % figure
 % plot(accuracy_training, 'red')
 % hold on
 % plot(accuracy_testing,'blue')

%% Question 4 - Stochastic Gradient Descent with constant step size -
n = 30; % number of features in X
w = zeros(n,1);
b = rand(1,1);
lambda = 0.00001;
alpha = 0.1;
nb_iterations = 30000;
nb_rows = length(Xtrain);
nb_rows_test = length(Xtest);
accuracy_training = zeros(nb_iterations, 1);
accuracy_testing = zeros(nb_iterations,1);
for i = 1:nb_iterations
    j = randi(nb_rows); % generate a random index from the 500 rows, j being an
integer not decimal
    dw = (-Xtrain_normal(j,:) * Ytrain(j))' +
(Xtrain_normal(j,:)*exp(Xtrain_normal(j,:)*w +b)/(1+exp(Xtrain_normal(j,:)*w +b)))';
    db = (-Ytrain(j) + (exp(Xtrain_normal(j,:)*w +b)/(1+exp(Xtrain_normal(j,:)*w
+b))));
```

```matlab
    % update w and b
    dw = dw + lambda*w;
    w = w - alpha*dw;
    b = b - alpha*db;
    prediction_training = (1 ./( 1 + exp(-(Xtrain_normal*w + b))) > 0.5);
    accuracy_training(i) = sum(prediction_training==Ytrain)/nb_rows; % see how the
accuracy will change at each iteration
    prediction_testing = (1 ./( 1 + exp(-(Xtest_normal*w + b))) > 0.5);
    accuracy_testing(i) = sum(prediction_testing==Ytest)/nb_rows_test;
end
accuracy_training(30000)% check the accuracy at the last iteration
accuracy_testing(30000)
nb_rows_test = length(Xtest);
% loss function
Loss = 0;
for i = 1:nb_rows_test
    Loss = Loss - Ytest(i)*(Xtest_normal(i,:)*w + b) + log (1 +
exp(Xtest_normal(i,:)*w + b));
end
Loss = Loss + (lambda/2)*norm(w)^2;
Loss/nb_rows_test;

% figure
%  plot(accuracy_training, 'red')
%  hold on
%  plot(accuracy_testing,'blue')

%% question 5 Stochastic Gradient Descent with diminishing step size -

n = 30; % number of features in X
w = zeros(n,1);
b = rand(1,1);
lambda = 0.00001;
alpha = 0.1;
nb_iterations = 30000;
nb_rows = length(Xtrain);
nb_rows_test = length(Xtest);
accuracy_training = zeros(nb_iterations, 1);
accuracy_testing = zeros(nb_iterations,1);
for i = 1:nb_iterations
    alpha = 0.1/sqrt(i);
    j = randi(nb_rows); % generate a random index from the 500 rows, j being an
integer not decimal
    dw = (-Xtrain_normal(j,:) * Ytrain(j))' +
(Xtrain_normal(j,:)*exp(Xtrain_normal(j,:)*w +b)/(1+exp(Xtrain_normal(j,:)*w +b)))';
    db = (-Ytrain(j) + (exp(Xtrain_normal(j,:)*w +b)/(1+exp(Xtrain_normal(j,:)*w
+b))));
    % update w and b
    dw = dw + lambda*w;
    w = w - alpha*dw;
    b = b - alpha*db;
    prediction_training = (1 ./( 1 + exp(-(Xtrain_normal*w + b))) > 0.5);
    accuracy_training(i) = sum(prediction_training==Ytrain)/nb_rows; % see how the
accuracy will change at each iteration
    prediction_testing = (1 ./( 1 + exp(-(Xtest_normal*w + b))) > 0.5);
    accuracy_testing(i) = sum(prediction_testing==Ytest)/nb_rows_test;
end
accuracy_training(30000)% check the accuracy at the last iteration
accuracy_testing(30000)
% loss function
Loss = 0;
for i = 1:nb_rows_test
    Loss = Loss - Ytest(i)*(Xtest_normal(i,:)*w + b) + log (1 +
exp(Xtest_normal(i,:)*w + b));
end
Loss = Loss + (lambda/2)*norm(w)^2;
Loss/nb_rows_test;

% figure
%  plot(accuracy_training, 'red')
%  hold on
%  plot(accuracy_testing,'blue')
```