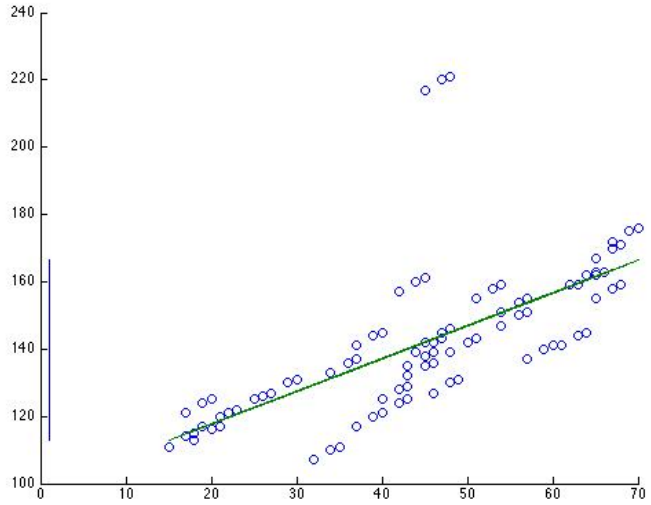
Alex Rosengarten
University of California, San Diego

Feb 17, 2015

HOMEWORK 2 — LaTeX

Problem 1. Problem 1: Regression

(c) Scatter plot of the data with regression line.

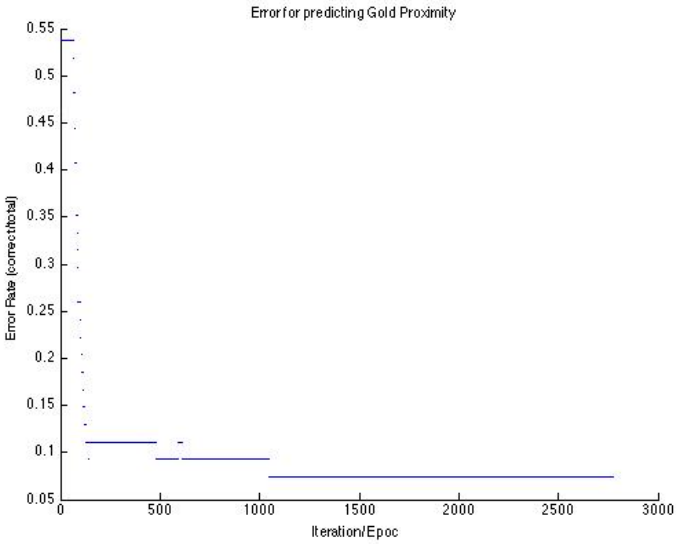


(d) R^2 for first data set: 0.4352. In general, R^2 indicates how well the regression line approximates real data - it is the "goodness of fit" measure. This specific score indicates that the regression line is not that strong of a fit for our data.

(f) 0.7994. This score indicates that our regression line is a good fit for our data.

Problem 2. Problem 2: Classification

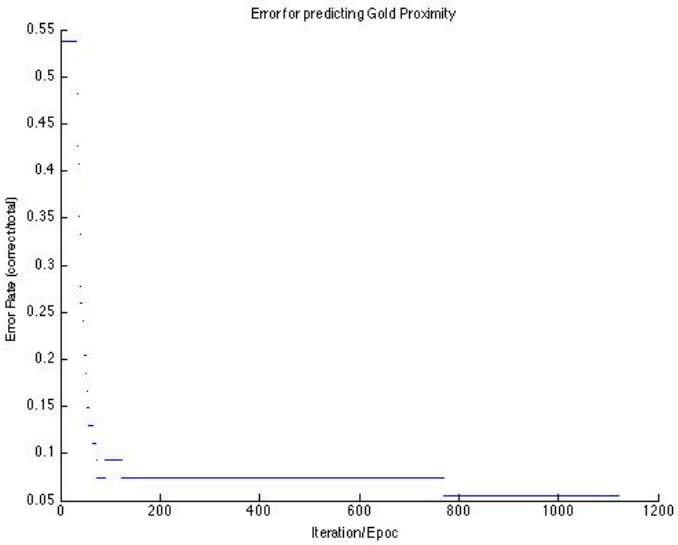
(b) The learning rate I use is 0.001. I repeat the training function until the difference between the old error and new error falls below a threshold value of 0.001 (this is my stopping condition). The graph is as follows:



(c) I used a threshold value of 0.5. That is, if the prediction value was 0.5 or above, it becomes a 1; if it is any value below, it becomes a zero (I used the round function). I chose 0.5 for my threshold value because the sigmoid function will map its input to a value between 0 and 1; if the input to the sigmoid function is zero, then it evaluates to 0.5, so this becomes the decision boundary. My training error was: 0.0741

(d) My test error rate was 0.3.

(e) I used the same learning rate and stopping condition as before. My graph of the error rate is as follows:



I used the same threshold of 0.5 on the output. My training error was: 0.0556

My testing error was: 0.3.

The cross-entropy function performed a lot faster than the SSE function: As you can see in the graphs, the cross-entropy cost function converged in about half the epocs as the SSE function. The reason for this probably has to do with the fact that we don't need to calculate the prime of the sigmoid function, making the learning rate a lot faster.

Problem 3. Appendix: Source code

Listing 1: My main HW2 script

```
% hw2_main

% % PROBLEM 1: Regression
% (a) Read in the data from data_pre.txt
% <index, one, age, systolic blood pressure>
% Col titled one is constant feature with value 1 for each data sample
blood_data_simple = dlmread('./data_pa_2/data_pre.txt', ',');
X = blood_data_simple(:,2:3);
Y = blood_data_simple(:,4);

% (b) Approximate the function f using linear regression. Specifically, use
% the closed form solution to obtain the parameters W that minimize the
% squred error loss L(W) = 1/2 || Y - f(X)||^2
% Hints: You will need to use the constant ones feature to train the bias,
%        Do not use gradient descent.
%        Read problem Q1(e) so you do not need to "generalize" your code
%        later. Keep it general the first time!

B_hat = (X'*X)\(X'*Y);

% (c) Obtain a scatter plot of the data. Plot a line Y = f(x) using the
% parameters you calculated.

figure; hold on;
scatter(X(:,2),Y(:));
plot(X,X*B_hat);
hold off;

% (d) Coffeficient of determination: R^2
%  R^2 = Sum_i(Y_ihat - Y_ibar)^2/Sum_i(Y_i - Y_bar)^2

Y_bar = mean(Y);
SS_tot = sum((Y-Y_bar).^2);
SS_reg = sum((X*B_hat-Y_bar).^2);
R_squared = SS_reg/SS_tot;
disp(R_squared);
```

```
37
38 % (e) Generalize your code to run regression on a higher dimensional data
39 % set viz. file data_16D.txt.
40 blood_data_16 = dlmread('./data_pa_2/data_16D.txt');
41 X16 = blood_data_16(:,2:end-1);
42 Y16 = blood_data_16(:,end);
43
44 B16_hat = (X16'*X16)\(X16'*Y16);
45
46 % (f) Report the R^2 score from the regression in part e. What does this
47 % score indicate?
48
49 Y16_bar = mean(Y16);
50 SS16_tot = sum((Y16-Y16_bar).^2);
51 SS16_reg = sum((X16*B16_hat-Y16_bar).^2);
52 R16_squared = SS16_reg/SS16_tot;
53 disp(R16_squared);
54
55 % PROBLEM 2: Classification
56
57 % (a) Read in data_LR.txt [and data_LR_test.txt]
58 data_LR = dlmread('./data_pa_2/data_LR.txt');
59 data_LR_test = dlmread('./data_pa_2/data_LR_test.txt');
60
61
62 X = data_LR(:,1:4);
63 X(:,4) = 1; % Bias
64 Y = data_LR(:,4);
65
66
67 X_test = data_LR_test(:,1:4);
68 X_test(:,4) = 1; % Bias
69 Y_test = data_LR_test(:,4);
70 Y1_test = Y_test(:,1);
71
72
73
74 % (b) Using a single neuron with a logistic activation function, train your
75 % weights using gradient descent to minimize the SSE: 1/2||T-f(net)||^2.
76 % Report (i) your learning rate, (ii) a graph of your error rare with
77 % each epoch, and (iii) your stopping criteria.
78
79 learning_rate = 0.001; % Learning rate (i)
80 stop_criteria = 0.0001; % Stop Criteria (iii)
81
82
```

```
[Y1_hat, W1] = sigtrainSSE( X, Y, learning_rate, stop_criteria, @calcSSE );
[Y2_hat, W2] = sigtrainCE( X, Y, learning_rate, stop_criteria, @calcCE );


% (c) Using the trained network and a simple step threshold t on the
% output, attept to classify all the training samples and report the
% training error. What theshold value did you use? Why?

Y1_class = round(Y1_hat); % Round sets x >= .5 to 1 and x < .5 to 0
Y2_class = round(Y2_hat);

trainErr1 = sum(Y1_class ~= Y)/numel(Y)
trainErr2 = sum(Y2_class ~= Y)/numel(Y)

% (d) Finally, use your trained logistic regression network with the output
% thresholded by t to classify the test samples given in the test file
% data_LR_tests.txt. Report your test error rate.
Test1 = sigmoid(X_test*W1);
Test2 = sigmoid(X_test*W2);

testErr1 = sum(round(Test1) ~= Y_test')/numel(Y_test)
testErr2 = sum(round(Test2) ~= Y_test')/numel(Y_test)



% (e) Given our 2 classes C_0 and C_1 we can model the output to be a
% random var y^(n) = f(W^t*x^(n)) ~ as i.i.d. Bernousli trials. y^(n)
% represents the prediction on the nth input pattern, i.e. x^(n).
% Maximinizng the likelihood of data[x^(n)] given the parameters[y^(n)] and
% the labels [t^(n)] with respect to the weights W leads to the cross
% entrpy loss function:
% L_y(W) = -Sum_n(t^(n) log(y^(n)) + (1-t^(n))log(1-y^(n)))

% Repeat steps 2(b), 2(c), and 2(d) using the cross-entropy loss function.
% Compare the performance of the SSE loss and cross-entropy loss in terms
% of speed of convergence.
```

Listing 2: SSE sigmoid training function

```
function [ Y_hat, W ] = sigtrainSSE( X, Y, learning_rate, stop_criteria, errFunc)
%UNTITLED5 Summary of this function goes here
%   Detailed explanation goes here

N = size(X, 1);
d = size(X, 2);

W = rand(d,1);  % Weights: num inputs x 1
```

```
9
10
11 Y_hat = zeros(N,1); % Predictions
12
13
14 figure; hold on;
15 err_old = Inf;
16 err = -Inf;
17 errRate = sum(Y ~= round(Y_hat))/numel(Y);
18 plot(0,errRate);
19 xlabel('Iteration/Epoc');
20 ylabel('Error Rate (correct/total)');
21 title('Error for predicting Gold Proximity');
22 i = 0;
23 while ( abs(err_old - err) > stop_criteria)
24     for P = 1:N
25
26         Y_hat(P) = sigmoid(X(P,:)*W);
27
28         W = W + learning_rate * (Y(P) - Y_hat(P)) ...
29               * (sigmoid(X(P,:)*W) * (1-sigmoid(X(P,:)*W)')).* X(P,:)';
30     end
31     err_old = err;
32     err = errFunc(Y, Y_hat);
33     i = i + 1;
34     errRate = sum(Y ~= round(Y_hat))/numel(Y);
35     plot(i, errRate);
36 end
37 hold off;
38 end
```

Listing 3: Cross Entropy sigmoid training function

```
1 function [ Y_hat, W ] = sigtrainCE( X, Y, learning_rate, stop_criteria, errFunc)
2 %UNTITLED5 Summary of this function goes here
3 %   Detailed explanation goes here
4
5 N = size(X, 1);
6 d = size(X, 2);
7
8 W = rand(d,1);  % Weights: num inputs x 1
9
10
11 Y_hat = zeros(N,1); % Predictions
12
13
14 figure; hold on;
```

```
15 err_old = Inf;
16 err = -Inf;
17 errRate = sum(Y ~= round(Y_hat))/numel(Y);
18 plot(0,errRate);
19 xlabel('Iteration/Epoc');
20 ylabel('Error Rate (correct/total)');
21 title('Error for predicting Gold Proximity');
22 i = 0;
23 while ( abs(err_old - err) > stop_criteria )
24     for P = 1:N
25
26         Y_hat(P) = sigmoid(X(P,:)*W);
27
28         W = W + learning_rate * (X(P,:)'*(Y(P)-Y_hat(P)));
29     end
30     err_old = err;
31     err = errFunc(Y, Y_hat);
32     i = i + 1;
33     errRate = sum(Y ~= round(Y_hat))/numel(Y);
34     plot(i, errRate);
35 end
36 hold off;
37 end
```

Listing 4: Sigmoid function

```
1 function [ out ] = sigmoid( u )
2 %UNTITLED4 Summary of this function goes here
3 %   Detailed explanation goes here
4 out = 1/(1+exp(-u));
5
6 end
```

Listing 5: Sum Squared Error

```
1 function [ errorRate ] = calcSSE( predictions, labels )
2 %calcError Calculates and returns the error rate of the perceptron
3 %   Input: predictins, labels
4 %   Output: errorRate = average error rate
5
6 errorRate = sum((labels - predictions).^2, 1)/2;
7
8
9
10 end
```

Listing 6: Cross Entropy Error

```
function [ out ] = calcCE( Y,Y_hat )
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here
out = 0;
for i = 1:numel(Y)
    out = out + -1*(Y(i)*log(Y_hat(i)) + (1-Y(i))*log(1-Y_hat(i)));
end

end
```

Submitted by Alex Rosengarten on Feb 17, 2015.