Michael Hawkins

ECE 174

Fall 2022

Mini Project 2

1) $$\nabla_w f_w(x) = \Big[\overbrace{\emptyset(w_2x_1+w_3x_2+w_4x_3+w_5)}^{w_1},\ \overbrace{\tfrac{\partial\emptyset}{\partial w_2}w_1\emptyset(w_2x_1+w_3x_2+w_4x_3+w_5)}^{w_2},\ \overbrace{\tfrac{\partial\emptyset}{\partial w_3}w_1\emptyset(w_2x_1+w_3x_2+w_4x_3+w_5)}^{w_3},$$

$$\overbrace{\tfrac{\partial\emptyset}{\partial w_4}w_1\emptyset(w_2x_1+w_3x_2+w_4x_3+w_5)}^{w_4},\ \overbrace{\tfrac{\partial\emptyset}{\partial w_5}w_1\emptyset(w_2x_1+w_3x_2+w_4x_3+w_5)}^{w_5},\ \overbrace{\emptyset(w_7x_1+w_8x_2+w_9x_3+w_{10})}^{w_6},$$

$$\overbrace{\tfrac{\partial\emptyset}{\partial w_7}w_6\emptyset(w_7x_1+w_8x_2+w_9x_3+w_{10})}^{w_7},\ \overbrace{\tfrac{\partial\emptyset}{\partial w_8}w_6\emptyset(w_7x_1+w_8x_2+w_9x_3+w_{10})}^{w_8},\ \overbrace{\tfrac{\partial\emptyset}{\partial w_9}w_6\emptyset(w_7x_1+w_8x_2+w_9x_3+w_{10})}^{w_9},$$

$$\overbrace{\tfrac{\partial\emptyset}{\partial w_{10}}w_6\emptyset(w_7x_1+w_8x_2+w_9x_3+w_{10})}^{w_{10}},\ \overbrace{\emptyset(w_{12}x_1+w_{13}x_2+w_{14}x_3+w_{15})}^{w_{11}},\ \overbrace{\tfrac{\partial\emptyset}{\partial w_{12}}w_{11}\emptyset(w_{12}x_1+w_{13}x_2+w_{14}x_3+w_{15})}^{w_{12}},$$

$$\overbrace{\tfrac{\partial\emptyset}{\partial w_{13}}w_{11}\emptyset(w_{12}x_1+w_{13}x_2+w_{14}x_3+w_{15})}^{w_{13}},\ \overbrace{\tfrac{\partial\emptyset}{\partial w_{14}}w_{11}\emptyset(w_{12}x_1+w_{13}x_2+w_{14}x_3+w_{15})}^{w_{14}},\ \overbrace{\tfrac{\partial\emptyset}{\partial w_{15}}w_{11}\emptyset(w_{12}x_1+w_{13}x_2+w_{14}x_3+w_{15})}^{w_{15}},\ \overbrace{1}^{w_{16}}\Big]^T$$

2) $$r(w) = [r_1(w)\ r_2(w)\ \dots\ r_N(w)] \qquad \nabla_w\left(f_w(x^i) - y^i\right)$$

$$D_r(w_t) = \begin{bmatrix} \nabla_w r_1(w_t)^T \\ \nabla_w r_2(w_t)^T \\ \vdots \\ \nabla_w r_N(w_t)^T \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial w_1} f_w(x^i) \\ \frac{\partial f}{\partial w_2} f_w(x^i) \\ \vdots \\ \frac{\partial f}{\partial w} f_w(x^i) \end{bmatrix}\Bigg|_{w=w_t}$$

```matlab
%%%%%%%%%%% INITIAL VALUES AND SETUP %%%%%%%%%%%%

%N number of Data Points
N = 100;

%create N random data points in R^3
x = rand([N 3]);
%create e       %e = 0 for part a-c
e = .1*rand([N 1]);
%create y
y = x(:,1).*x(:,2) + x(:,3) + e(:);
% y = x(:,1).*x(:,3) + 1.5*x(:,1);     Different Function
%lambda and it's matrix
lambda = 10^-5;
lambdaSquaredRootedMatrix = sqrt(lambda) * eye(16);
%initial weights
w = ones([16 1]);
%gamma initialization
gamma = 10^-5;
%tanh equation setup
syms n;
tanh = @(n) ((exp(n)-exp(-n))/(exp(n) + exp(-n)));
%setup given equation for f_w
syms x1 x2 x3 w1 w2 w3 w4 w5 w6 w7 w8 w9 w10 w11 w12 w13 w14 w15 w16;
f = @(x1,x2,x3,w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12,w13,w14,w15,w16) ...
    w1*tanh(w2*x1+w3*x2+w4*x3+w5) + w6*tanh(w7*x1+w8*x2+w9*x3+w10) + ...
    w11*tanh(w12*x1+w13*x2+w14*x3+w15) + w16;

%%%%%%%%%%% END OF INITIAL SETUP %%%%%%%%%%%%%%%%

for i = 1:300   %imax is 300

    initialDerivativeMatrix = derivativeMatrix(x,w,N);

    %residual
    r = zeros([N 1]);
    for k = 1:N
        r(k) = f(x(k,1),x(k,2),x(k,3),w(1),w(2),w(3),w(4),w(5),w(6),...
            w(7),w(8),w(9),w(10),w(11),w(12),w(13),w(14),w(15),w(16)) - y(k,1);
    end


    %b_t
    b = r - initialDerivativeMatrix * w;
    %append zeros
    b = [b;zeros([16 1])];
    %D_h
    Dh = [initialDerivativeMatrix; lambdaSquaredRootedMatrix];



    %final b for iteration
    b = [b; -1*sqrt(gamma)*w];
    %final 'A' [b - (-A)w]
    A = [Dh; (sqrt(gamma) * eye(16))];
    A = -1*A;
    %new w
    new_weights = pinv(transpose(A) * A) * transpose(A) * b;



    %residual of k+1
    r_hat = zeros([N 1]);
    for u = 1:N
    r_hat(u) = f(x(u,1),x(u,2),x(u,3),new_weights(1),new_weights(2),...
        new_weights(3),new_weights(4),new_weights(5),new_weights(6),...
        new_weights(7),new_weights(8),new_weights(9),new_weights(10),...
        new_weights(11),new_weights(12),new_weights(13),new_weights(14),...
        new_weights(15),new_weights(16)) - y(u,1);
    end

```

```
        %loss function for k+1
        h_hat = [r_hat; (lambda*new_weights)];
        loss_hat(i) = transpose(h_hat)*h_hat;

        %loss function for k
        h_bar = [r; (lambda*w)];
        loss_bar(i) = transpose(h_bar)*h_bar;

        %loss function comparisons
        if (loss_hat(i) < loss_bar(i)) %loss function of loss hat is better, so update weights
            w = new_weights;
            gamma = .8*gamma;
        else %increase gamma and redo LS
            gamma = 2*gamma;
        end



        %finishing is when the loss functions are withing 10^-4
        %or less than 10^-3
        if(abs(loss_hat(i) - loss_bar(i)) < 10^-6)
            break;
        end

        %or when loss functions are small
        if loss_hat(i) < 10^-2
            w = new_weights;
            break;
        elseif loss_bar(i) < 10^-2
            break;
        end


    end




    %returns derivative matrix of all N Points (x Points for each row, weights
    %in each column
    function Dr = derivativeMatrix(x,w,N)

        %setup given equation for f_w
        syms x1 x2 x3 w1 w2 w3 w4 w5 w6 w7 w8 w9 w10 w11 w12 w13 w14 w15 w16;
        f = @(x1,x2,x3,w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12,w13,w14,w15,w16) ...
            w1*tanh(w2*x1+w3*x2+w4*x3+w5) + w6*tanh(w7*x1+w8*x2+w9*x3+w10) + ...
            w11*tanh(w12*x1+w13*x2+w14*x3+w15) + w16;

        %differential w/r to each weight
        Dr1(x1,x2,x3,w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12,w13,w14,w15,w16) = diff(f,w1);
        Dr2(x1,x2,x3,w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12,w13,w14,w15,w16) = diff(f,w2);
        Dr3(x1,x2,x3,w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12,w13,w14,w15,w16) = diff(f,w3);
        Dr4(x1,x2,x3,w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12,w13,w14,w15,w16) = diff(f,w4);
        Dr5(x1,x2,x3,w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12,w13,w14,w15,w16) = diff(f,w5);
        Dr6(x1,x2,x3,w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12,w13,w14,w15,w16) = diff(f,w6);
        Dr7(x1,x2,x3,w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12,w13,w14,w15,w16) = diff(f,w7);
        Dr8(x1,x2,x3,w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12,w13,w14,w15,w16) = diff(f,w8);
        Dr9(x1,x2,x3,w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12,w13,w14,w15,w16) = diff(f,w9);
        Dr10(x1,x2,x3,w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12,w13,w14,w15,w16) = diff(f,w10);
        Dr11(x1,x2,x3,w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12,w13,w14,w15,w16) = diff(f,w11);
        Dr12(x1,x2,x3,w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12,w13,w14,w15,w16) = diff(f,w12);
        Dr13(x1,x2,x3,w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12,w13,w14,w15,w16) = diff(f,w13);
        Dr14(x1,x2,x3,w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12,w13,w14,w15,w16) = diff(f,w14);
        Dr15(x1,x2,x3,w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12,w13,w14,w15,w16) = diff(f,w15);
        Dr16(x1,x2,x3,w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12,w13,w14,w15,w16) = diff(f,w16);

        %skeleton derivative matrix for all N points (500)
        Dr = zeros([N 16]);
```

Workspace

Name	Value
A	132x16 double
b	132x1 double
Dh	116x16 double
f	@(x1,x2,x3,w1,w2,w3,w4,w5,w6,w7,w8,...
gamma	0.0104
h_bar	116x1 double
h_hat	116x1 double
i	51
initialDerivativeM...	100x16 double
k	100
lambda	1.0000e-05
lambdaSquaredRo...	16x16 double
loss_bar	1x231 double
loss_hat	1x231 double
n	1x1 sym
N	100
new_weights	16x1 double
r	100x1 double
r_hat	100x1 double
tanh	@(n)((exp(n)-exp(-n))/(exp(n)+exp(-n)))
u	100
w	16x1 double
w1	1x1 sym
w10	1x1 sym
w11	1x1 sym
w12	1x1 sym
w13	1x1 sym
w14	1x1 sym
w15	1x1 sym
w16	1x1 sym
w2	1x1 sym
w3	1x1 sym
w4	1x1 sym
w5	1x1 sym
w6	1x1 sym
w7	1x1 sym
w8	1x1 sym
w9	1x1 sym
x	100x3 double
x1	1x1 sym
x2	1x1 sym
x3	1x1 sym
y	100x1 double

λ = 10^-5

new_weights

16x1 double

	1
1	3.9466
2	0.3504
3	0.4896
4	-0.0019
5	-1.0956
6	7.4884
7	-0.1613
8	0.0489
9	0.1336
10	0.0054
11	3.0629
12	0.3950
13	-0.5555
14	0.0038
15	0.7428
16	1.1751

124 iterations, 500 Points

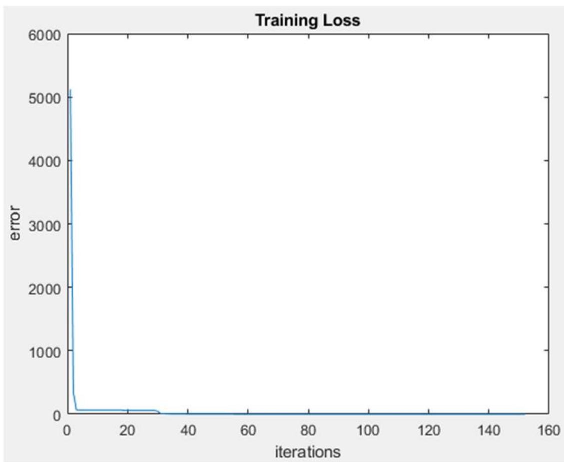
Y = X1 * X2 + X3

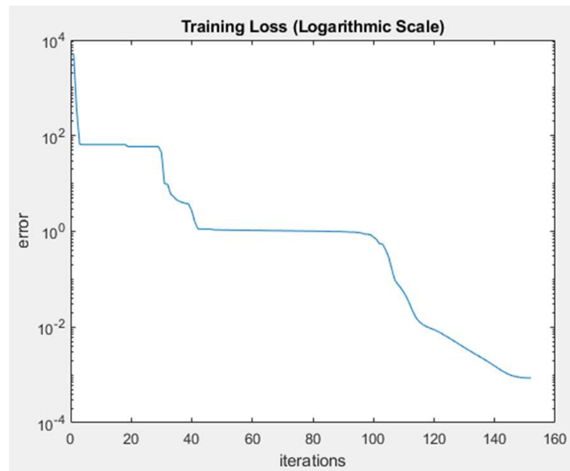
Loss function = 8.48 * 10^-4

RMS = .8709

100 Test Points

RMS = .8427





---

new_weights

16x1 double

	1
1	3.9197
2	0.3499
3	0.4895
4	-0.0018
5	-1.1114
6	7.9243
7	-0.1484
8	0.0485
9	0.1265
10	-0.0022
11	3.1218
12	0.3933
13	-0.5515
14	0.0025
15	0.7634
16	1.1548

152 iterations, 500 Points

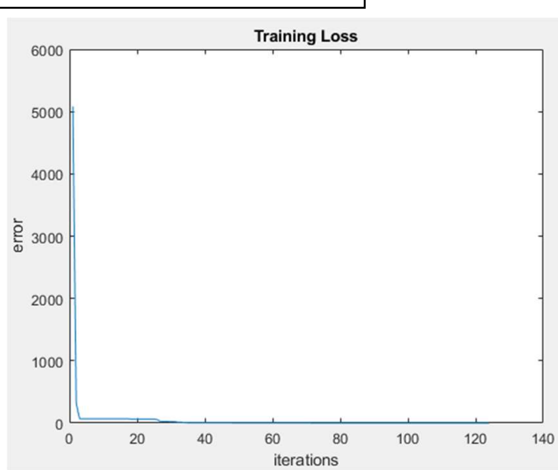
Y = X1 * X2 + X3

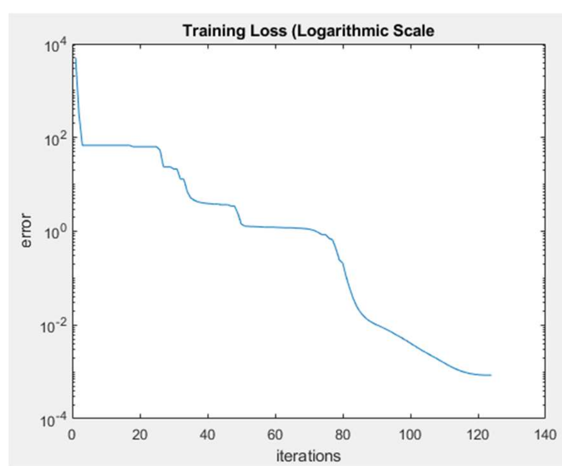
Loss function = 8.59 * 10^-4

RMS = .8693

100 Test Points

RMS = .8839





new_weights 16x1 double	1
1	7.7327
2	0.2665
3	-9.9044e-04
4	0.3276
5	-1.0635
6	10.7734
7	0.2613
8	0.0163
9	-0.3507
10	1.7575
11	10.7756
12	0.2866
13	-0.0112
14	-0.3342
15	1.6622
16	-14.1006

197 iterations, 500 Points

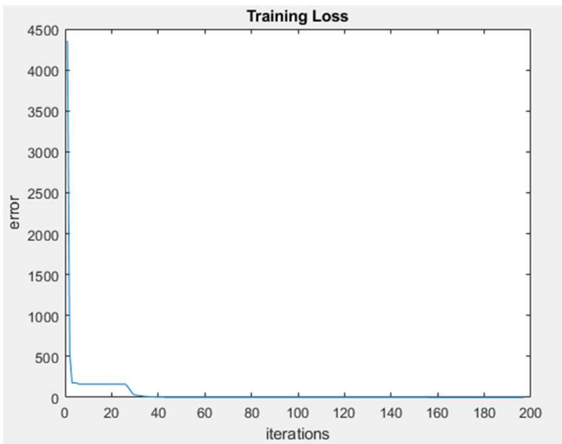
Y = X1 * X3 + 1.5*X2

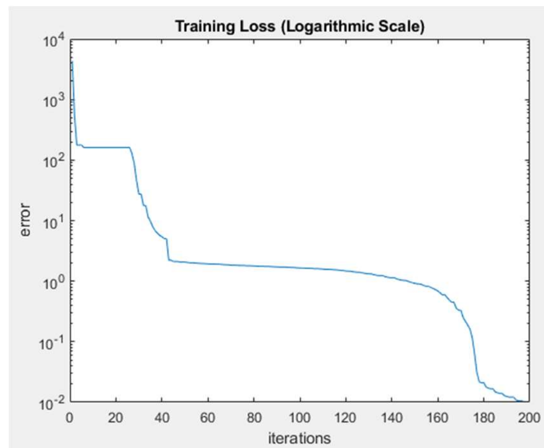
Loss function = .0105

RMS = 1.0195

100 Test Points

RMS = 1.0118





new_weights 16x1 double	1
1	10.1790
2	0.2287
3	0.0042
4	0.3121
5	-1.1354
6	14.9447
7	0.3464
8	0.0427
9	-0.2705
10	2.1172
11	14.9488
12	0.2044
13	-0.0208
14	-0.3568
15	1.6153
16	-20.0694

339 iterations, 500 Points

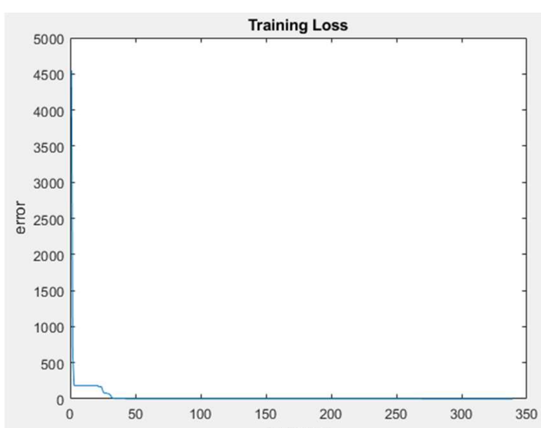
Y = X1 * X3 + 1.5*X2

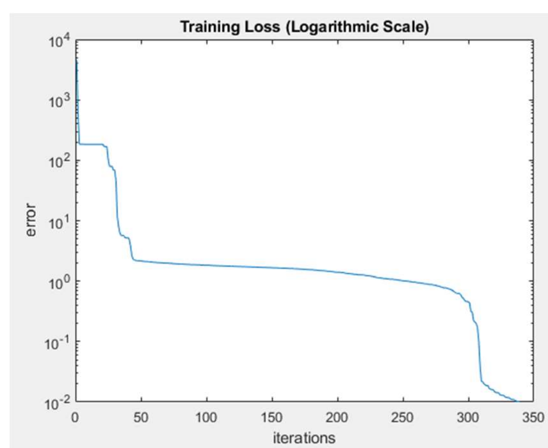
Loss function = .0102

RMS = .9868

100 Test Points

RMS = 1.0046





new_weights

16x1 double

	1
1	4.8478
2	-0.6488
3	0.2477
4	-0.0505
5	0.8598
6	2.4382
7	0.9310
8	0.2978
9	0.0529
10	-1.4384
11	4.9974
12	0.2760
13	-0.1496
14	0.2221
15	-0.2398
16	0.0245

107 iterations, 500 Points

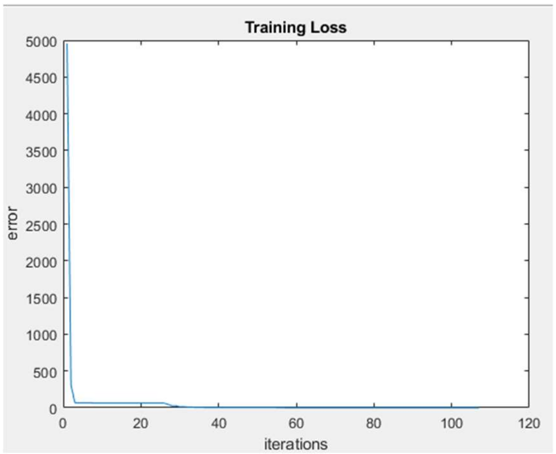
Y = X1 * X2 + X3 + e

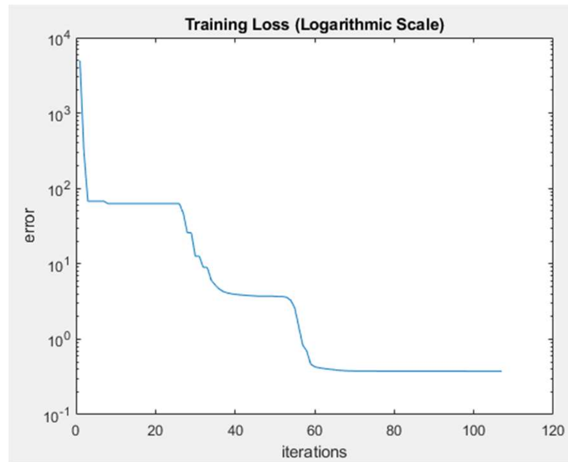
Loss function = .3756

RMS = .8962

100 Test Points

RMS = .9160





VARIABLE

new_weights

16x1 double

	1
1	4.5909
2	-0.4018
3	0.4893
4	-0.0284
5	1.0974
6	2.5655
7	0.4817
8	0.5607
9	-0.0055
10	-1.3028
11	9.2585
12	0.0482
13	-0.1240
14	0.1168
15	-0.1705
16	0.1360

190 iterations, 500 Points

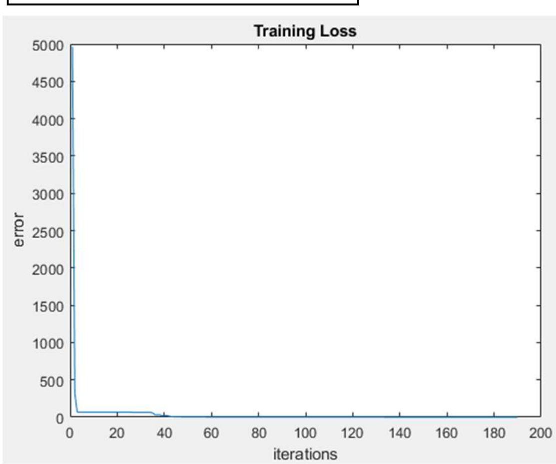
Y = X1 * X2 + X3 + e

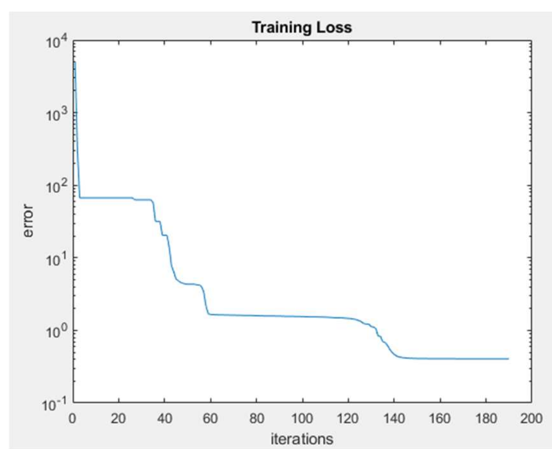
Loss function = .4058

RMS = .8949

100 Test Points

RMS = .8930





To compare the test data, the RMS of it was computed to determine how close f_w is to y. When the algorithm was completed successfully and the RMS was calculated, the test data RMS was very close to the training data RMS with a max difference in RMS of .03 so we can say that our findings were consistent.

With my own function, I changed x1,x2,x3 around and a coefficient in one of the terms. The results for this function were not as good as the given function (RMS slightly above 1), however the difference in f_w and y are still very small when comparing RMS.

With the noisy data, consistent results were achieved. This may be due to the value of the noise (noise was at most about 10% of the data point) however it seems as though the noise didn't affect the function very much.

Overall, most parts of the project were successful with the only inconsistency in my own function for y.