ML Homework 2: vb2182@nyu.edu

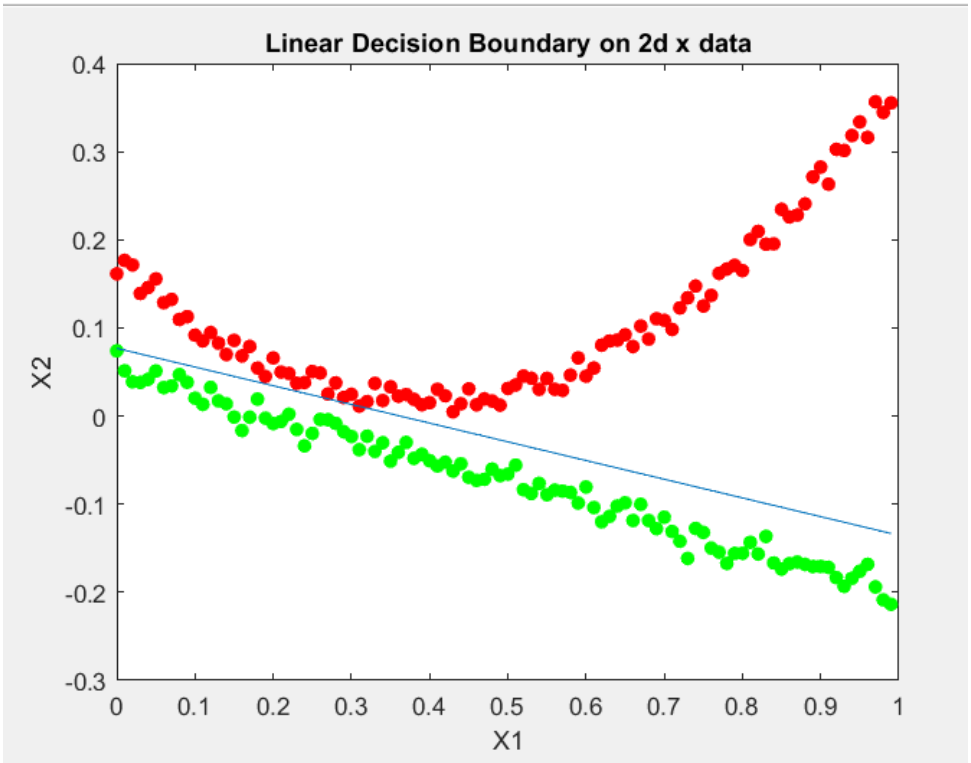
Q1:

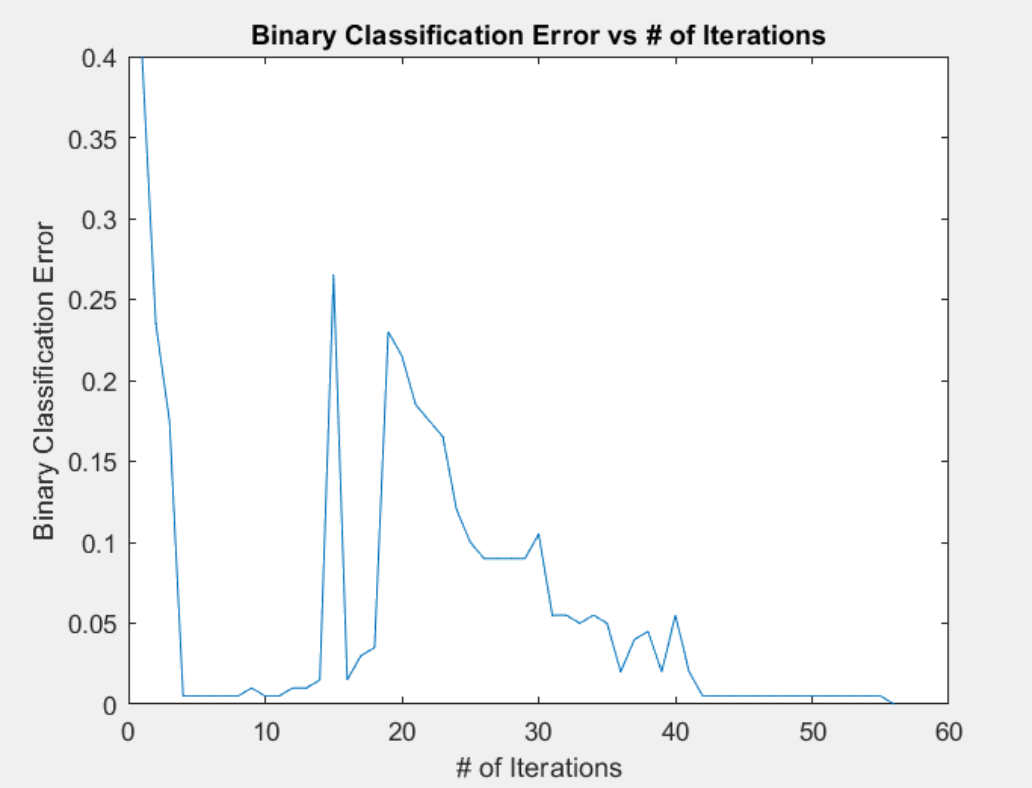
Using Stochastic Gradient Descent (SGD)

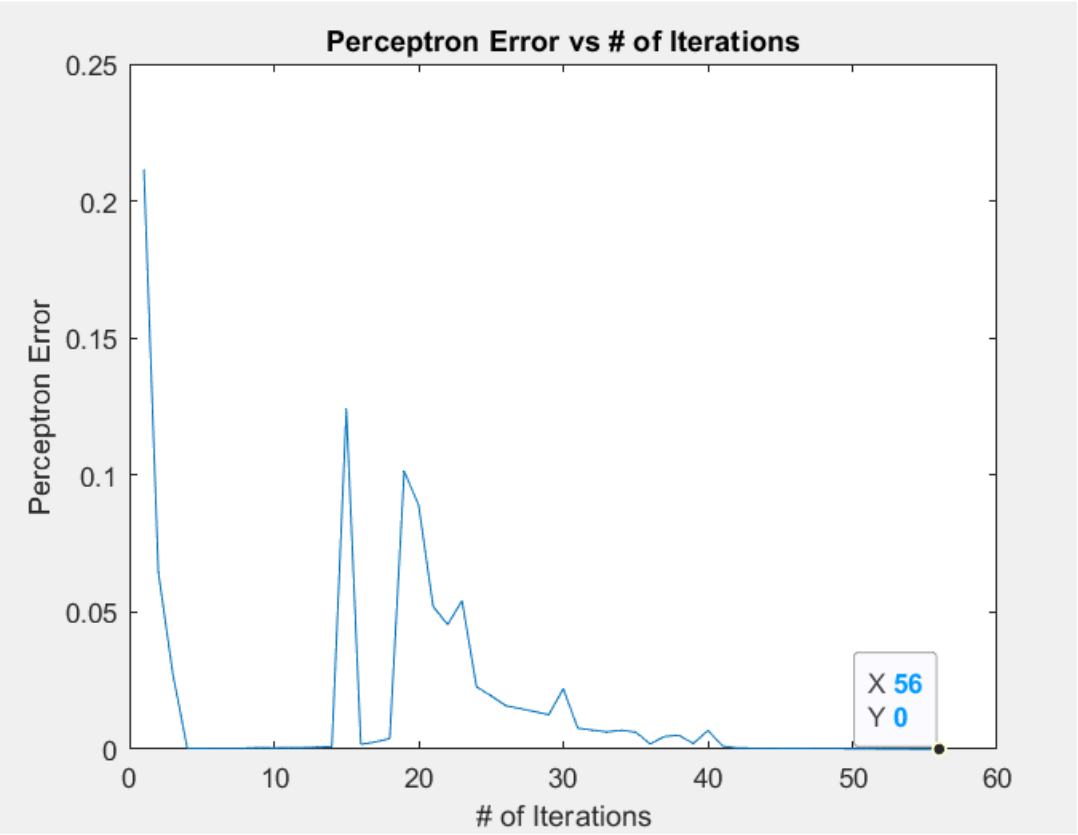
Setting eta (Learning rate or step size) to 1 in the case of SGD, without loss of generality, plotted the following:

As expected, binary classification error converges to zero comparably slowly than perceptron error (convergence to zero at the 56^{th} iteration).

Plots are below.




Binary Classification Error vs # of Iterations
Binary Classification Error
# of Iterations
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
10
20
30
40
50
60

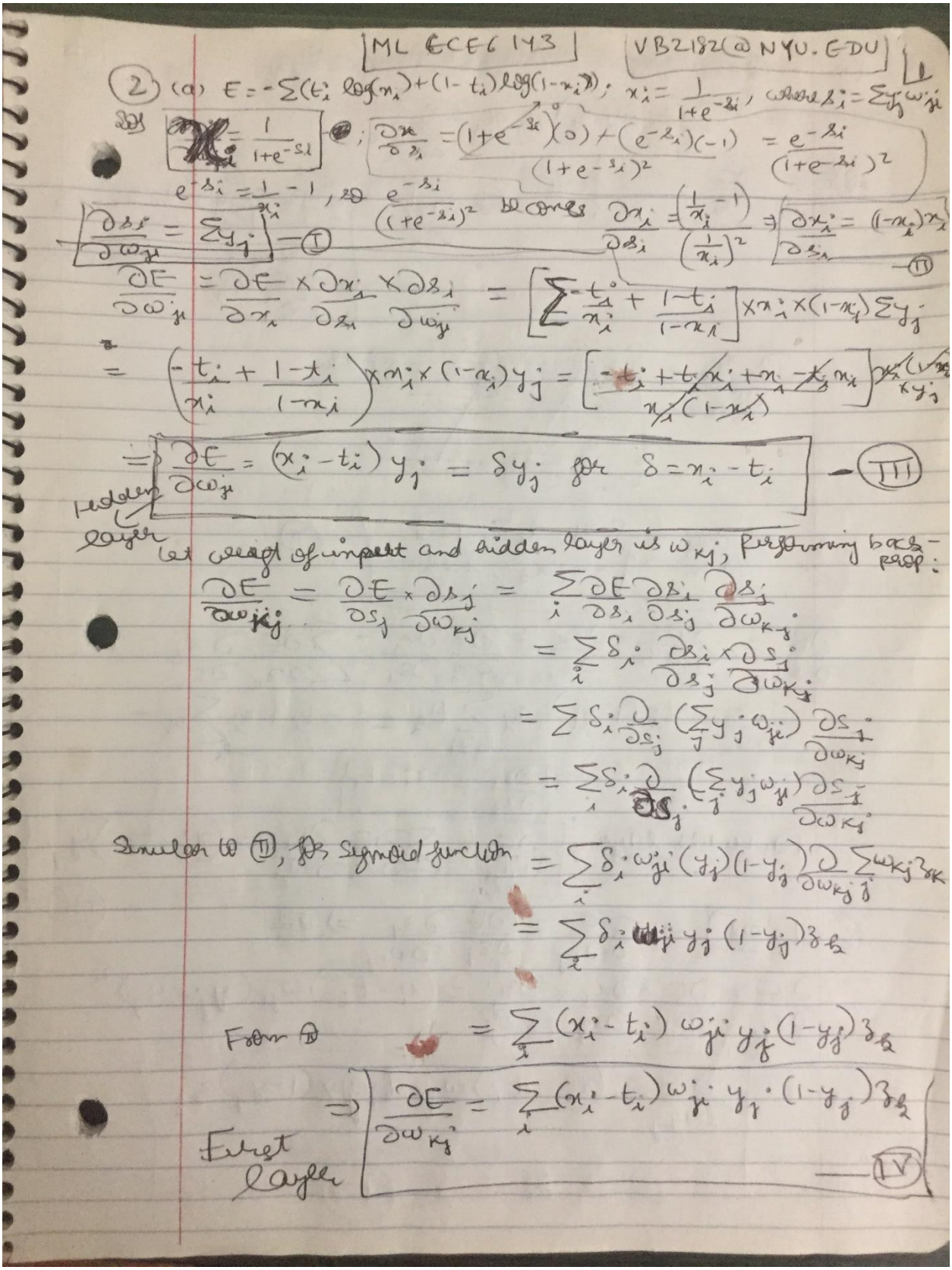

Perceptron Error vs # of Iterations
Perceptron Error
# of Iterations
0.25
0.2
0.15
0.1
0.05
0
0
10
20
30
40
50
60
X 56
Y 0

Code in Matlab:

```
clc; %clear work command window

clear variables;
clf;
path="C:\Users\vxbha\Desktop\VB2182_ML_HW2\";
load("data3.mat");

%Initialization of Data
N=length(data);
X=[data(:,1:2),ones(N,1)]; %need to add array of all ones to accomodate bias

Y=data(:,3);
theta=randn(3,1); %3-by-1 vector of random numbers

missed_values=1;

current_iteration=1;
binary_classification_error=[];
perceptron_error=[];
%Employing SGD
while(missed_values~=0)
    for i=1:N
        Loss=Y(i)*(X(i,:)*theta);
        if Loss<=0
            %Update
            theta = theta+Y(i)*X(i,:)';
            y1=classification_func(theta,X,Y);
            y2 = perceptron_func(theta,X,Y);
            binary_classification_error(current_iteration)=mean(y1);
            perceptron_error(current_iteration)=mean(y2);
        end
    end
    missed_values=mean(y2);

    current_iteration=current_iteration+1;
end

figure(1)
for i=1:N
    if Y(i)==1
        plot(X(i,1),X(i,2),'.','MarkerSize',20,'MarkerEdgeColor','g');
    elseif Y(i)==-1
        plot(X(i,1),X(i,2),'.','MarkerSize',20,'MarkerEdgeColor','r');
    end
    hold on
end
xlabel('X1');
ylabel('X2');
x = [min(X(:,1)):1/200:max(X(:,1))];
x2 = (-theta(3)*ones(N,1)-theta(1)*x)/theta(2);
plot(x,x2);
title("Linear Decision Boundary on 2d x data");
plotoutput=("Linear Decision Boundary on 2d x data.png");
print(path+plotoutput,"-dpng");
```

```
figure(2)
plot(binary_classification_error);
xlabel('# of Iterations');
ylabel('Binary Classification Error');
title("Binary Classification Error vs # of Iterations");
plotoutput=("Binary Classification Error vs # of Iterations.png");
print(path+plotoutput,"-dpng");

figure(3)
plot(perceptron_error);
xlabel('# of Iterations');
ylabel('Perceptron Error');
title("Perceptron Error vs # of Iterations");
plotoutput=("Perceptron Error vs # of Iterations.png");
print(path+plotoutput,"-dpng");

function f = classification_func(theta,x,Y)

    y = x*theta;
    f = ones(length(x),1);
    for i=1:length(x)
        if y(i)*Y(i)<0
         f(i)=1;
        elseif y(i)*Y(i)>=0
            f(i)=0; %instead of -1, I have equated it to zero so the classification
error converges to 0.
        end
    end
end



function f = perceptron_func(theta,x,Y)
    y=x*theta;
    f=zeros(length(x),1);
    for i=1:length(x)
        if Y(i)*y(i)<0
            f(i)=-Y(i)*y(i);
        end
    end
end
```

Q2 (a)



ML ECE6143 VB2182@NYU.EDU

(2) (a) $E = -\sum (t_i \log(x_i) + (1-t_i)\log(1-x_i))$; $x_i = \frac{1}{1+e^{-z_i}}$, where $z_i = \sum_j y_j \omega_{ji}$

$x_i = \frac{1}{1+e^{-z_i}}$; $\frac{\partial x_i}{\partial z_i} = \frac{(1+e^{-z_i})(0) + (e^{-z_i})(-1)}{(1+e^{-z_i})^2} = \frac{e^{-z_i}}{(1+e^{-z_i})^2}$

$e^{-z_i} = \frac{1}{x_i} - 1$, so $\frac{e^{-z_i}}{(1+e^{-z_i})^2}$ becomes $\frac{\partial x_i}{\partial z_i} = \frac{\left(\frac{1}{x_i} - 1\right)}{\left(\frac{1}{x_i}\right)^2} \Rightarrow \frac{\partial x_i}{\partial z_i} = (1-x_i)x_i$ — (II)

$\frac{\partial z_i}{\partial \omega_{ji}} = y_j$ — (I)

$\frac{\partial E}{\partial \omega_{ji}} = \frac{\partial E}{\partial x_i} \times \frac{\partial x_i}{\partial z_i} \times \frac{\partial z_i}{\partial \omega_{ji}} = \left[\sum -\frac{t_i}{x_i} + \frac{1-t_i}{1-x_i}\right] \times x_i \times (1-x_i) \sum y_j$

$= \left(-\frac{t_i}{x_i} + \frac{1-t_i}{1-x_i}\right) \times x_i \times (1-x_i) y_j = \left[\frac{-t_i + t_i x_i + x_i - t_i x_i}{x_i(1-x_i)}\right] x_i (1-x_i) y_j$

$\Rightarrow \frac{\partial E}{\partial \omega_{ji}} = (x_i - t_i) y_j = \delta y_j$ for $\delta = x_i - t_i$ — (III)

Hidden layer: Let weight of input and hidden layer is ω_{kj}, Performing back-prop:

$\frac{\partial E}{\partial \omega_{kj}} = \frac{\partial E}{\partial s_j} \times \frac{\partial s_j}{\partial \omega_{kj}} = \sum_i \frac{\partial E}{\partial s_i} \frac{\partial s_i}{\partial s_j} \frac{\partial s_j}{\partial \omega_{kj}}$

$= \sum_i \delta_i \frac{\partial s_i}{\partial s_j} \times \frac{\partial s_j}{\partial \omega_{kj}}$

$= \sum \delta_i \frac{\partial}{\partial s_j} \left(\sum_j y_j \omega_{ji}\right) \frac{\partial s_j}{\partial \omega_{kj}}$

$= \sum_i \delta_i \frac{\partial}{\partial s_j} \left(\sum_j y_j \omega_{ji}\right) \frac{\partial s_j}{\partial \omega_{kj}}$

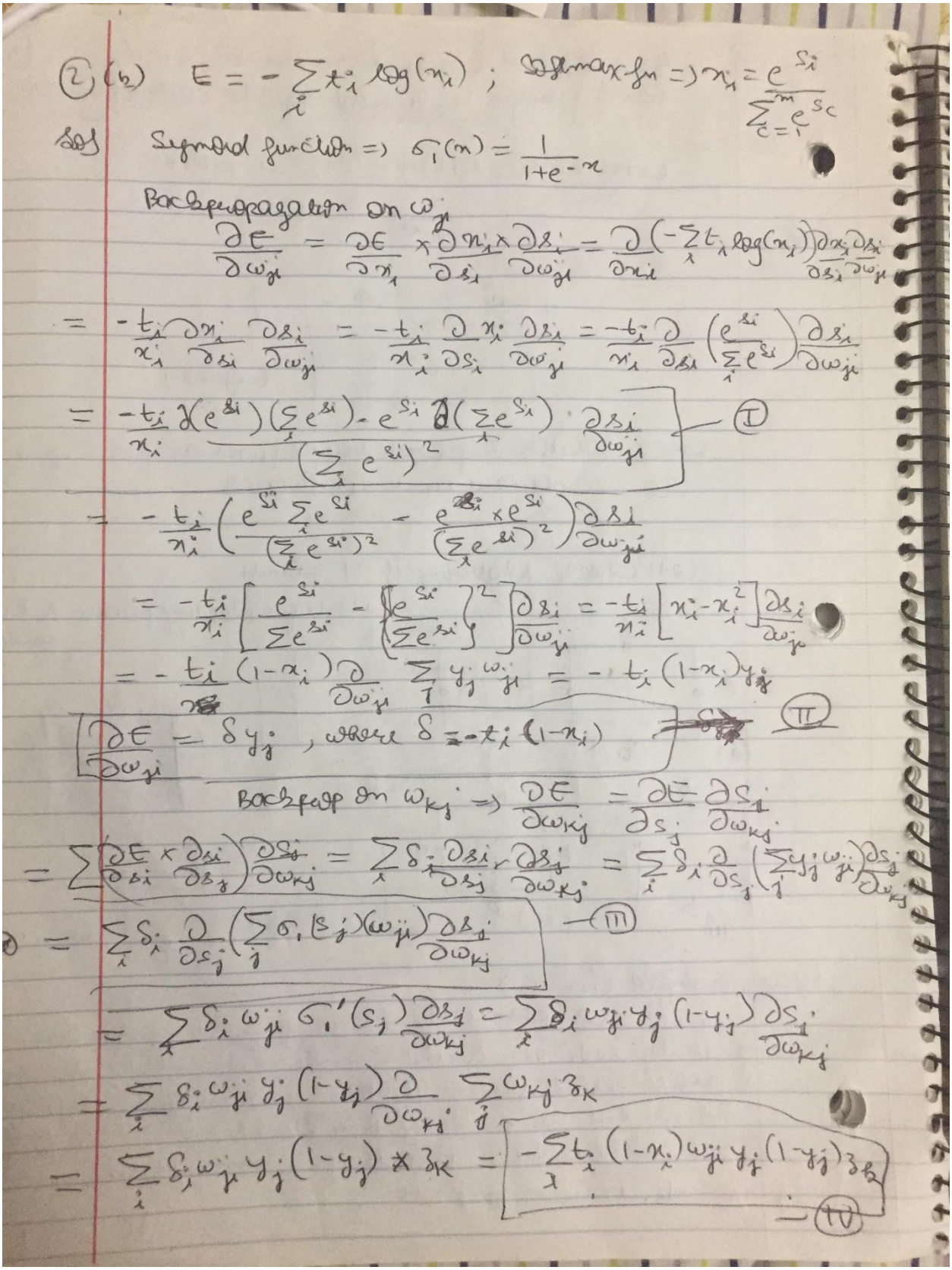
Similar to (II), for Sigmoid function $= \sum_i \delta_i \omega_{ji} (y_j)(1-y_j) \frac{\partial}{\partial \omega_{kj}} \sum_j \omega_{kj} z_k$

$= \sum_i \delta_i \omega_{ji} y_j (1-y_j) z_k$

From (I) $= \sum_i (x_i - t_i) \omega_{ji} y_j (1-y_j) z_k$

Hidden layer $\Rightarrow \frac{\partial E}{\partial \omega_{kj}} = \sum_i (x_i - t_i) \omega_{ji} y_j (1-y_j) z_k$ — (IV)

2 (b)



② (b) $E = -\sum_i t_i \log(x_i)$; Softmax fn $\Rightarrow x_i = \frac{e^{s_i}}{\sum_{c=1}^{m} e^{s_c}}$

Sol Sigmoid function $\Rightarrow \sigma_1(x) = \frac{1}{1+e^{-x}}$

Backpropagation on ω_{ji}

$$\frac{\partial E}{\partial \omega_{ji}} = \frac{\partial E}{\partial x_i} \times \frac{\partial x_i}{\partial s_i} \times \frac{\partial s_i}{\partial \omega_{ji}} = \frac{\partial}{\partial x_i}\left(-\sum_i t_i \log(x_i)\right) \frac{\partial x_i}{\partial s_i} \frac{\partial s_i}{\partial \omega_{ji}}$$

$$= -\frac{t_i}{x_i} \frac{\partial x_i}{\partial s_i} \frac{\partial s_i}{\partial \omega_{ji}} = -\frac{t_i}{x_i} \frac{\partial}{\partial s_i} x_i \frac{\partial s_i}{\partial \omega_{ji}} = -\frac{t_i}{x_i} \frac{\partial}{\partial s_i}\left(\frac{e^{s_i}}{\sum_i e^{s_i}}\right) \frac{\partial s_i}{\partial \omega_{ji}}$$

$$= \frac{-t_i}{x_i} \underbrace{\frac{(e^{s_i})(\sum_i e^{s_i}) - e^{s_i}\,\partial(\sum_i e^{s_i})}{(\sum_i e^{s_i})^2} \frac{\partial s_i}{\partial \omega_{ji}}} \quad \text{— (I)}$$

$$= -\frac{t_i}{x_i}\left(\frac{e^{s_i}\sum_i e^{s_i}}{(\sum_i e^{s_i})^2} - \frac{e^{s_i} \times e^{s_i}}{(\sum_i e^{s_i})^2}\right) \frac{\partial s_i}{\partial \omega_{ji}}$$

$$= -\frac{t_i}{x_i}\left[\frac{e^{s_i}}{\sum e^{s_i}} - \left\{\frac{e^{s_i}}{\sum e^{s_i}}\right\}^2\right] \frac{\partial s_i}{\partial \omega_{ji}} = -\frac{t_i}{x_i}\left[x_i - x_i^2\right] \frac{\partial s_i}{\partial \omega_{ji}}$$

$$= -\frac{t_i}{x_i}(1-x_i) \frac{\partial}{\partial \omega_{ji}} \sum_j y_j \omega_{ji} = -t_i(1-x_i) y_j$$

$$\boxed{\frac{\partial E}{\partial \omega_{ji}} = \delta y_j \text{, where } \delta = -t_i(1-x_i)} \quad \text{— (II)}$$

Backprop on $\omega_{kj} \Rightarrow \frac{\partial E}{\partial \omega_{kj}} = \frac{\partial E}{\partial s_j} \frac{\partial s_j}{\partial \omega_{kj}}$

$$= \sum_i \left(\frac{\partial E}{\partial s_i} \times \frac{\partial s_i}{\partial s_j}\right) \frac{\partial s_j}{\partial \omega_{kj}} = \sum_i \delta_i \frac{\partial s_i}{\partial s_j} \times \frac{\partial s_j}{\partial \omega_{kj}} = \sum_i \delta_i \frac{\partial}{\partial s_j}\left(\sum_j y_j \omega_{ji}\right) \frac{\partial s_j}{\partial \omega_{kj}}$$

$$= \sum_i \delta_i \frac{\partial}{\partial s_j}\left(\sum_j \sigma_1(s_j)(\omega_{ji})\right) \frac{\partial s_j}{\partial \omega_{kj}} \quad \text{— (III)}$$

$$= \sum_i \delta_i \omega_{ji} \sigma_1'(s_j) \frac{\partial s_j}{\partial \omega_{kj}} = \sum_i \delta_i \omega_{ji} y_j (1-y_j) \frac{\partial s_j}{\partial \omega_{kj}}$$

$$= \sum_i \delta_i \omega_{ji} y_j (1-y_j) \frac{\partial}{\partial \omega_{kj}} \sum_j \omega_{kj} z_k$$

$$= \sum_i \delta_i \omega_{ji} y_j (1-y_j) * z_k = \boxed{-\sum_i t_i (1-x_i) \omega_{ji} y_j (1-y_j) z_k} \quad \text{— (IV)}$$

Q3

(3) For the distribution $\{P_K \mid k = 1, 2, \ldots, N\}$ [3]

Entropy $(H) = -\sum_{k=1}^{N} P_K \log P_K$ (To maximize)

Soln. Constraint $\sum_{k=1}^{N} P_k = 1$ (as per the given distribution) — (I)

~~So minimize $F(P_1, P_2, \ldots, \lambda_0) = -\sum P_K \log P_K$~~

To maximize H with a ~~negative~~ sign, minimize the following instead

$$F(P_1, P_2, \ldots P_N, \lambda_0) = \sum_{k=1}^{N} P_K \log P_K - \lambda_0 (\Sigma P_K - 1) \quad \text{— (II)}$$

$$\frac{dF}{dP_1} = \log P_1 + \frac{P_1}{P_1} + 0 - \lambda_0 = 0 \quad \text{(To minimize)}$$

$$\Rightarrow \log P_1 - \lambda_0 + 1 = 0 \quad \text{— (i)}$$

$\frac{\partial F}{\partial P_2} =$

~~$\frac{dF}{dP_2} = \log P_2 - \lambda_0 + 1 = 0$ — (ii)~~

$$\log P_1 = \lambda_0 - 1 \quad \text{— (i)}$$

$$\frac{dF}{dP_2} = \log P_2 - \lambda_0 + 1 = 0$$

$$\Rightarrow \log P_2 = \lambda_0 - 1 \quad \text{— (ii)}$$

Similarly $\log P_3 = \lambda_0 - 1$ — (iii)

$\log P_K = \lambda_0 - 1$ — (k)

$\log P_N = \lambda_0 - 1$ — (n)

~~Add (i) to (n)~~

$\Rightarrow$ ~~$\log P_1 + \ldots + \log P_N = N(\ldots)$~~

$$\log P_1 = \log P_2 \cdots = \log P_N = \lambda_0 - 1 \quad \text{— (III)}$$

So $P_1 = P_2 = \cdots = P_N$ — (IV)

~~So $N \times \log P_1 =$~~

$$\log P_1 = \lambda_0 - 1$$

$$P_1 = e^{\lambda_0 - 1} \quad \text{— (V)}$$

$P_1 + P_2 + P_3 + \cdots + P_N$ or $\sum_{K=1}^{N} P_K = 1$ as per (I)

~~$I = N$~~

$$\sum_{K=1}^{N} P_K = N\left(e^{\lambda_0 - 1}\right)$$

$$1 = N\left(e^{\lambda_0 - 1}\right)$$

$$\Rightarrow \frac{1}{N} = e^{\lambda_0 - 1}$$

$$\lambda_0 = \ln\left(\frac{1}{N}\right) + 1 \quad \text{— (VI)}$$

~~Or~~

Substitute (VI) in

$$\log P_K = \lambda_0 - 1$$

$$\log P_K = \ln\left(\frac{1}{N}\right)$$

$$\Rightarrow P_K = \frac{1}{N} \quad \text{for } K = 1, 2, \cdots, N$$

So when $P_1 = P_2 = \cdots = P_K = \frac{1}{N}$

As the entropy is maximized

So, distribution for entropy maximization is the Normal distribution

Q4

(4) Axis aligned square means a square with shape aligned to the coordinate axes. [5]

Let's check shattering of 3 points

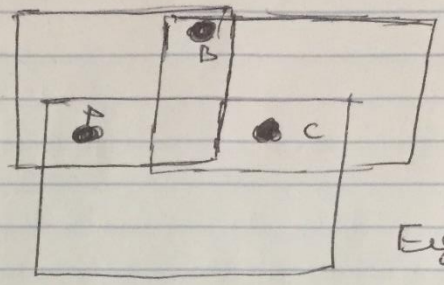
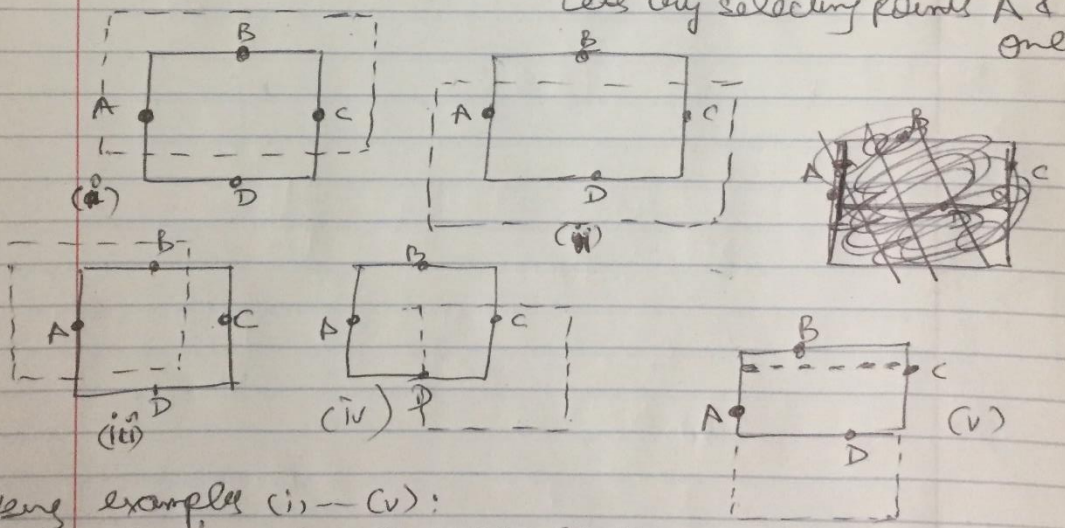


Figure 1

We can shatter 3 points using square as per Figure 1
So VC dimension is at least 3

Let's check shattering of 4 points

Let's try selecting points A & C only



Using examples (i) – (v):
In (i) - (iv) points are on the edges of a square.
If we try selecting A and C only, we either miss the other while trying to form axis-aligned square.
Or we end up selecting B or D also (which should be different class)
(v) is on the edges of a rectangle. Same issue

So VC dimension can't be 4, it's 3