CSE 253 Assignment I

Puneeth BommiReddy
A53093725
University of California, San Diego
pbommire@eng.ucsd.edu

1 Problems from Bishop

Please go to the end of PDF for problems 1.1 - 1.4

2 Perceptron

Please go to the end of PDF for parts 1 and 2.

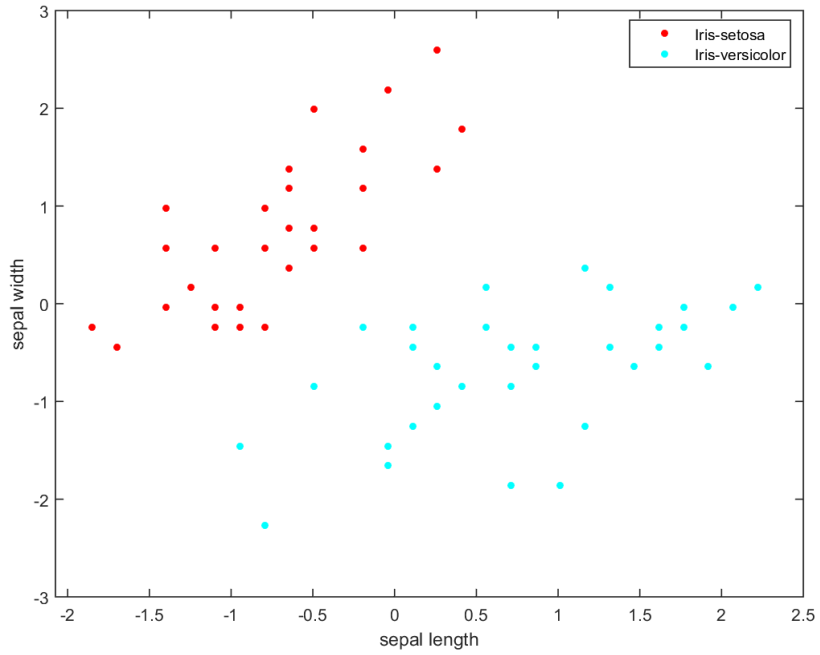
2.2 In my solution, I have used target/teacher = -1,1 (as described in Chapter 4 of PRML by C.M.Bishop) for negative and positive classes respectively. This is contrast with the target/teacher = 0,1 notation used in class. Please note this difference.

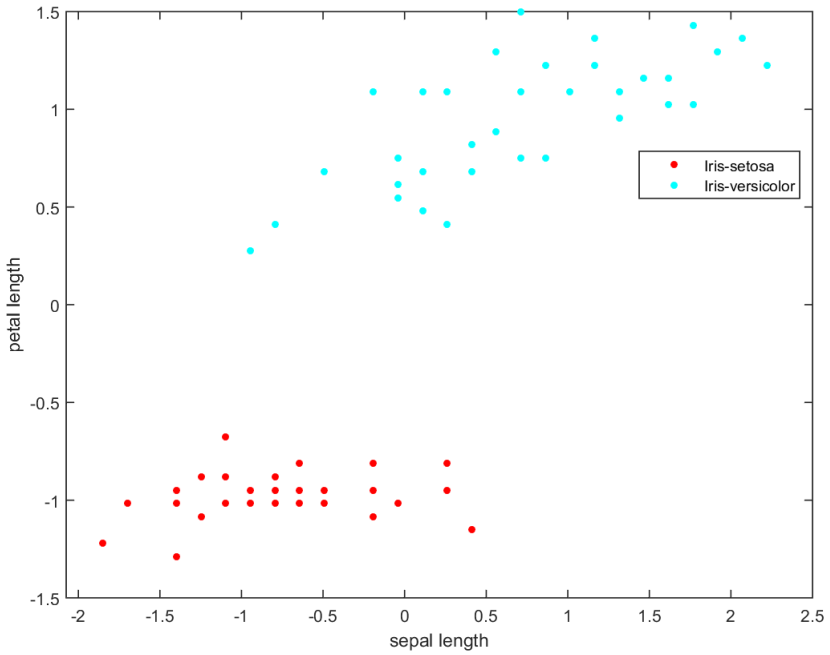
2.2.c No, the solution is not unique. There are an infinite number of lines with slope = -1 and intercept in range (1,2) seperating the point (1,1) from {(0,0), (1,0), (0,1)}. This can be achieved with different learning rates and order of training inputs chosen in the stochastic gradient step. (More details in hand-written part)

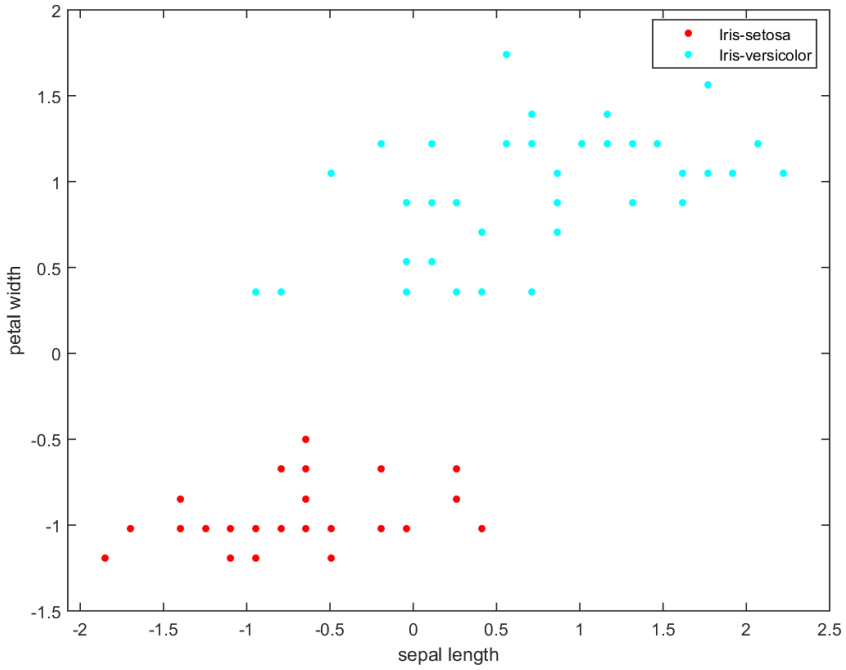
2.3.d

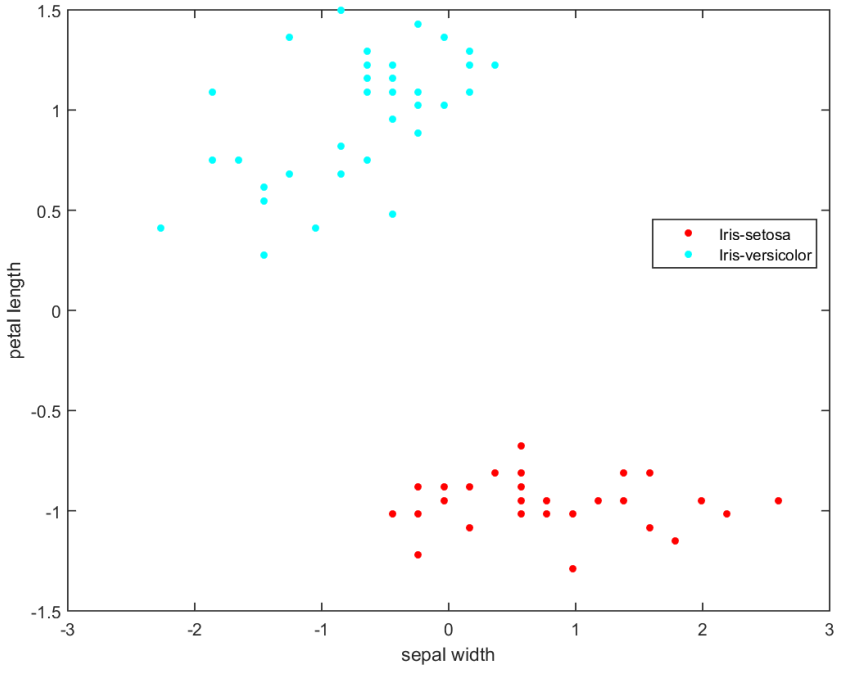
i. Z-scoring restricts the magnitude of the input values. This in turn helps restrict the value of w. That is irrespective of maximum magnitude and range of input (corresponding to zero mean and scaling by standard deviation) w will take on similar ranges of values, instead on blowing up/taking infinitesimal values.

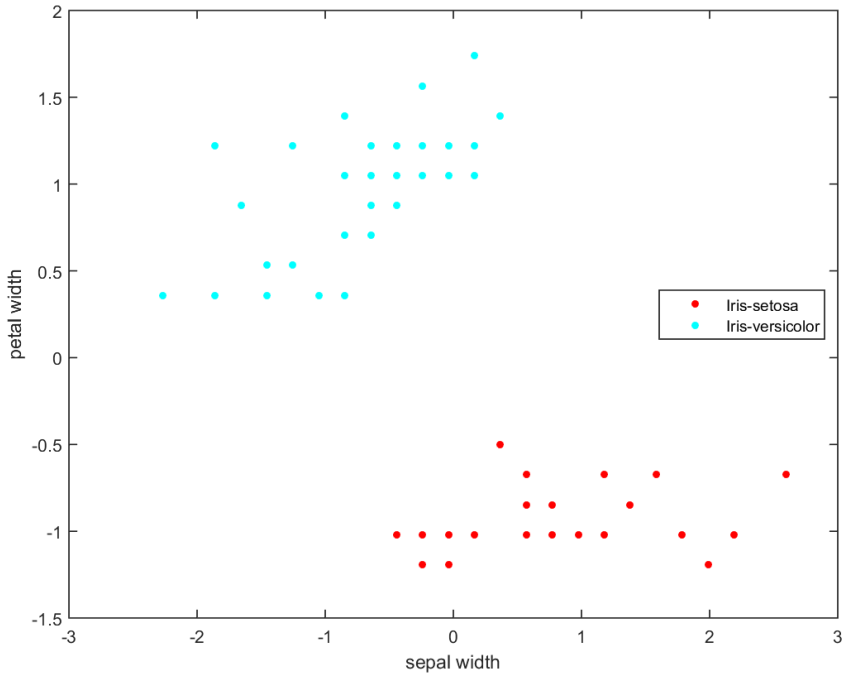
ii.

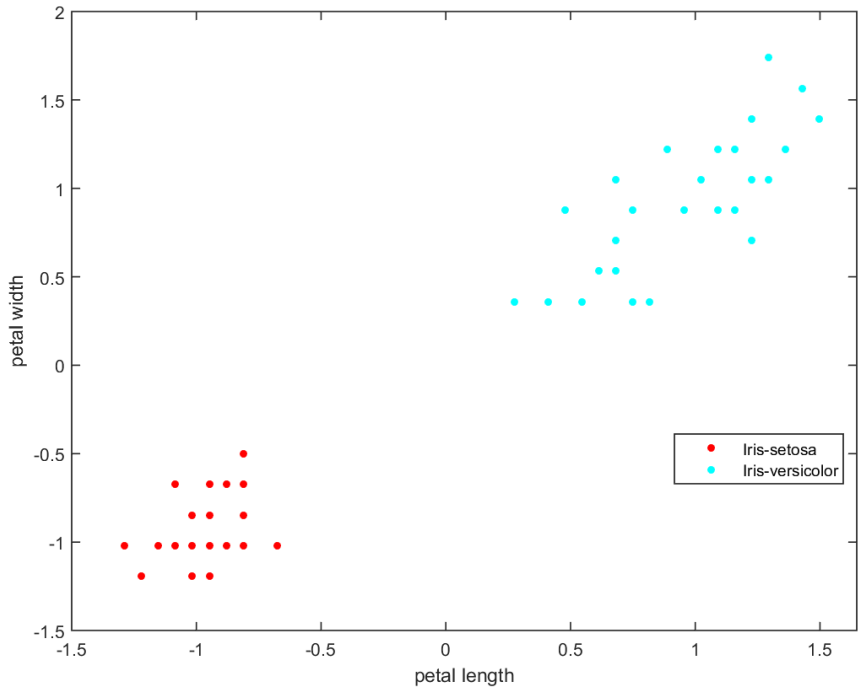

Iris-setosa
Iris-versicolor
sepal width
sepal length
3
2
1
0
-1
-2
-3
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5


Iris-setosa
Iris-versicolor
petal length
sepal length
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5


Iris-setosa
Iris-versicolor
petal width
sepal length
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5


Iris-setosa
Iris-versicolor
petal length
sepal width
1.5
1
0.5
0
-0.5
-1
-1.5
-3
-2
-1
0
1
2
3





Yes, the classes are linearly seperable in all feature spaces. Because the data is like that?

iv. The test error rate is 0.

v. My learning rate was 1. Increasing the rate scales w by the corresponding amount.

rate = 1 w1 = [-2.0000 -1.2746 1.2902 -1.5005 -1.4082]

rate = 0.25 w2 = [-0.5000 -0.3186 0.3226 -0.3751 -0.3520] = 0.25*w1

rate = 2 w3 = [-4.0000 -2.5492 2.5805 -3.0010 -2.8163] = 2*w1

3 Logistic and Softmax Regression

First level headings are lower case (except for first word and proper nouns), flush left, bold and in point size 12. One line space before the first level heading and ½ line space after the first level heading.

3.3 Read in Data. A script from Stanford's Ufldl tutorial was used to import the MNIST images into MATLAB from http://ufldl.stanford.edu/wiki/index.php/Using_the_MNIST_Dataset.

3.4 Logistic Regression

1st Method: Initially I trained each 2-way classifier for 1000 iterations. Upon increasing the iterations beyond that point the test-set classification performance did not change significantly (it would oscillate sometimes very slighty about the same point).

2nd Method: After that I changed the training scheme to include learning rate annealing. That is I reduced the learning rate in half everytime the diffrence in validation error (test-set classification error) did not change by atleast 1%. However the classification performance did not improve significantly (improved by 0.1%) copared to the 1st Method.

3.4.a Individual Classifier Test Accuracy

The error rates for the 2-way classifiers in order from 0 to 9 are:

[0.0080 0.0070 0.0265 0.0265 0.0240 0.0260 0.0210 0.0300 0.0470 0.0385]

Where an error rate of 0.0265 corresponds to 2.65% misclassification and so on.

The corresponding accuracy rates = 1 - error rates are :

[0.9920 0.9930 0.9735 0.9735 0.9760 0.9740 0.9790 0.9700 0.9530 0.9615]

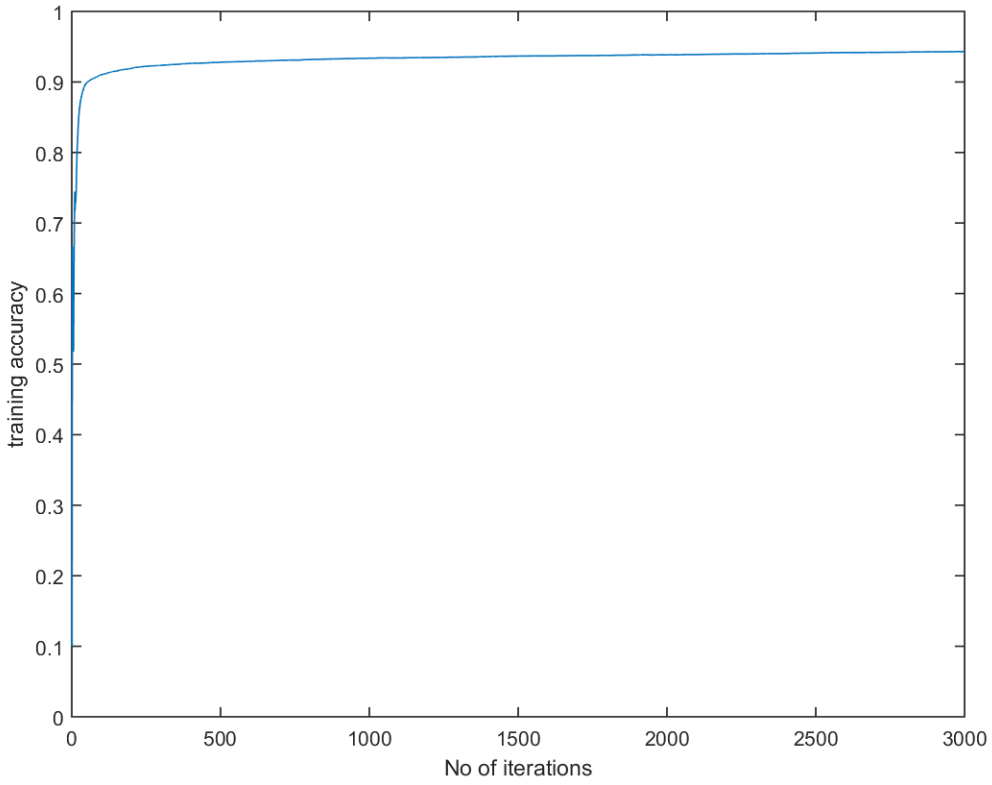
3.4.b Overall Test Accuracy

The overall test accuracy was 0.8845 (88.45%) = 1 - test_err_LR. test_err_LR is the corresponding variable in the code that measures the test error for Logistic Regression.

3.5 Softmax Regression

The gradient descent was run for 3000 iterations at which point the classifier pretty much converged.

3.5.a Training Error vs No of Gradient Descent Iterations



3.5.b Softmax Test Accuracy

The overall test accuracy was 0.8915 (89.15%) = 1 - test_err_SR. test_err_SR is the corresponding variable in the corresponding code that measures the test error for Softmax Regression.

3.5.c Comparison with Logistic Regression

Softmax Regression is marginally better than Logistic Regression (0.7%).

APPENDIX

2.3 PERCEPTRON CODE:

```
n_train = size(iristrain,1); %Used MATLAB's interface to import csv
n_test = size(iristest,1);
iristrain_mat = cell2mat(iristrain(:,1:4));
iristest_mat = cell2mat(iristest(:,1:4));
ztrain = zscore(iristrain_mat);
%ztest = zscore(iristest_mat);
ztest = iristest_mat - repmat(mean(iristrain_mat),[n_test 1]); %Using
training mean to zscore
std_train = std(iristrain_mat); %Using training std to zscore
for i = 1:4
    ztest(:,i) = ztest(:,i)/std_train(i);
end

train_target = ones(n_train,1);
test_target = ones(n_test,1);
for i = 1:size(iristrain_mat,1)
    if strcmp(iristrain{i,5},'Iris-versicolor')
```

```
        train_target(i) = 0;
    end
end
for i = 1:size(iristest_mat,1)
    if strcmp(iristest{i,5},'Iris-versicolor')
        test_target(i) = 0;
    end
end
labels = {'sepal length','sepal width','petal length','petal width'};
for i = 1:4
    for j = i+1:4

gscatter(ztrain(:,i),ztrain(:,j),iristrain(:,5));xlabel(labels{i});ylabe
l(labels{j});
        print(sprintf('fig%i%i',i,j),'-dpng');
    end
end
w = zeros(5,1);
n_err = 10;
eta = 1;
while n_err > 0 %Since data is linearly seperable, training till zero
error
    for i = 1:n_train %Iterating over each datapoint instead of
"randomly choosing"
        out = [1,ztrain(i,:)]*w >= 0;
        if out ~= train_target(i)
            if train_target(i) == 1
                w = w + eta*[1,ztrain(i,:)]';
            else
                w = w - eta*[1,ztrain(i,:)]';
            end
        end
    end
    output = [ones(n_train,1),ztrain]*w >= 0;
    n_err = sum(abs(output - train_target));
end

test_output = [ones(n_test,1),ztest]*w >= 0;
test_err = sum(abs(test_output - test_target));
```

3.3 LOADING IMAGES:

```
%% Loading Images
images = loadMNISTImages('train-images.idx3-ubyte');
labels = loadMNISTLabels('train-labels.idx1-ubyte');
train_images = [ones(1,20000);images(:,1:20000)];
train_labels = labels(1:20000,:);
images = loadMNISTImages('t10k-images.idx3-ubyte');
labels = loadMNISTLabels('t10k-labels.idx1-ubyte');
test_images = [ones(1,2000);images(:,1:2000)];
test_labels = labels(1:2000,:);
```

3.4 LOGISTIC REGRESSION:

```
n_train = 20000;n_test = 2000;eta = 1/n_train;
n_iter = 1000;
W = zeros(10,785); % Weight Matrix [No of classifiers X Input Size]
%Logistic Regression
```

```
for class_lbl = 0:9 %Computing weight for each class
    class_lbl
    for i = 1:n_iter
        y = (1./(1 + exp(-W(class_lbl+1,:)*train_images)))'; %Softmax
output given current weights
        t = double((train_labels == class_lbl)); % 0-1 vector of targets
for each image. 1 if image belongs to class 'class_lbl'
        err_sum = train_images*(y-t); %Sum of error over all training
images
        W(class_lbl+1,:) = W(class_lbl+1,:) - eta*err_sum'; %Gradient
descent based weight update
    end
end
%Calculating Individual Classifier Error
dec_mat = W*test_images;
classifier_err = zeros(10,1);
for classifier = 0:9
    dec = (1./(1 + exp(-dec_mat(classifier+1,:)))) > 0.5;
    trgt = (test_labels == classifier);
    classifier_err(classifier+1) = sum(abs(dec'-trgt))/n_test;
end
%Calculating Test Error
[~,I] = max(dec_mat,[],1);
test_output = (I - 1)';
test_err_LR = sum(1-(test_output == test_labels))/n_test;
```

3.5 SOFTMAX REGRESSION:

```
n_train = 20000;n_test = 2000;eta = 1/n_train;
n_iter = 3000;
W = zeros(10,785); % Weight Matrix [No of classes X Input Size]
T = zeros(10,n_train); %Target matrix.
for sample = 1:n_train
    T(train_labels(sample)+1,sample) = 1;% Each column has a '1' in the
location of the true class of that image
end
train_error = zeros(n_iter,1); %Training error over iterations
% Softmax Regression
for iter = 1:n_iter
    Y = exp(W*train_images); %Computing class activations 'a'
    for sample = 1:n_train
        Y(:,sample) = Y(:,sample)/sum(Y(:,sample)); %Normalizing
activations
    end
    W = W - eta*(Y - T)*train_images'; %Gradient descent based weight
update
    %Training error over iterations
    [~,I] = max(Y,[],1);
    train_output = (I - 1)';
    train_error(iter) = sum(1-(train_output == train_labels))/n_train;
end
plot(1 - train_error); ylabel('training accuracy'); xlabel('No of
iterations'); print('figlast','-dpng');
%Calculating training error
[~,I] = max(W*test_images,[],1); test_output = (I-1)';
test_error_SR = sum(1-(test_output == test_labels))/n_test;
```

Date

1. $\int_0^\infty e^{-r^2} r^{d-1}\, dr$ Let $r^2 = u \Rightarrow 2r\,dr = du \Rightarrow dr = \frac{1}{2r}du = \frac{1}{2\sqrt{u}}du$ $[u^{1/2} = r]$

$= \frac{1}{2}\int_0^\infty e^{-u} u^{\frac{d-1}{2}} u^{-1/2}\, du$

$= \frac{1}{2}\int_0^\infty e^{-u} u^{\frac{d}{2}-1}\, du = \Gamma(d/2)$ from 1.43

$\prod_{i=1}^{d} \int_{-\infty}^{\infty} e^{-x_i^2} dx_i = \prod_{i=1}^{d} (2\pi)^{1/2} = (2\pi)^{d/2}$

[$d=2 \Rightarrow$ $S_2 = \frac{2\pi}{\Gamma(1)} = 2\pi$ // since $r=1$]

$\therefore S_d = \frac{\prod_{i=1}^{d} \int_{-\infty}^{\infty} e^{-x_i^2} dx_i}{\int_0^\infty e^{-r^2} r^{d-1} dr} = \frac{(2\pi)^{d/2}}{\Gamma(d/2)}$ [$d=3 \Rightarrow$ $S_3 = \frac{(2\pi)^{3/2}}{\sqrt{\pi}/2} = 4\pi = 4\pi r^2$]

2. (i) From 1.43 surface area (SA) ~~volume~~ of a hypersphere of radius a is $S_d a^{d-1}$ * [when $a=1$ $SA = S_d$, can also be checked for circle 2sphere & 3sphere].

Integrating infinitesimal hypersphere surface areas to get volume:

$$V_d = \int_0^a S_d r^{d-1}\, dr = S_d \left[\frac{r^d}{d}\right]_0^a = \frac{S_d a^d}{d}$$

[* NOTE:
Although it may look like a huge leap to say SA of radius a is $S_d a^{d-1}$, it is not. We know when $a=1$ $SA = S_d$ and that in a d-dimensional space surface area (SA) $\propto a^{d-1}$.

$\therefore K S_d a^{d-1} = SA\ (\text{radius} = a)$ (K constant)

Evaluating at $a=1$; $K S_d a^{d-1}\big|_{a=1} = S_d$ (from 1.43)

$\Rightarrow K = 1$

$\therefore SA(\text{radius} = a) = S_d a^{d-1}$]

(ii) $\frac{\text{Volume of sphere}}{\text{Volume of cube}} = \frac{S_d\, a^d}{d \cdot (2a)^d} = \frac{S_d}{2^d\, d} = \frac{\pi^{d/2}}{d\, 2^{d-1}\, \Gamma(d/2)}$

$d/2 = x+1$

$x = d/2 - 1$

$\propto \frac{\pi^{d/2}}{d\, 2^{d-1} (2\pi)^{1/2} e^{-[d/2 - 1]} \left(\frac{d}{2} - 1\right)^{(d/2 - 1/2)}}$

[Stirling's approximation]

$= K \cdot \frac{\pi^{d/2} \cdot e^{d/2}}{2d} \cdot \frac{1}{\left(\frac{d}{2} - 1\right)^{(d/2 - 1/2)}}$ ← constant independent of d.

$= K \cdot \left(\frac{\sqrt{\pi e}}{2}\right)^d \cdot \frac{1}{\left(\frac{d}{2} - 1\right)^{(d/2 - 1/2)}}$ ← > 1

$= K \frac{\alpha^d}{\left(\frac{d}{2} - 1\right)^{(d/2 - 1/2)}}$ $[\alpha > 1]$

$= K' \frac{\alpha^d}{\left(\sqrt{\frac{d}{2} - 1}\right)^d}$ $\left[K' = K\left(\sqrt{\frac{d}{2} - 1}\right)\right]$

$= K' \left[\frac{\alpha}{\sqrt{\frac{d}{2} - 1}}\right]^d$

but $\lim_{d \to \infty} \frac{\alpha}{\sqrt{\frac{d}{2} - 1}} = 0$ [α is a constant].

$\therefore \lim_{d \to \infty} K' \left[\frac{\alpha}{\sqrt{\frac{d}{2} - 1}}\right]^d = 0$

(iii) $d_{\text{centre-corner}} = \sqrt{d\left(\frac{2a}{2}\right)^2} = a\sqrt{d}$

$d_{\perp \text{ to edges}} = \frac{2a}{2} = a$

$\therefore \frac{d_{\text{centre-corner}}}{d_{\perp}} = \frac{a\sqrt{d}}{a} = \sqrt{d}$

3. (i) $f = \dfrac{\frac{S_d\, a^d}{d} - \frac{S_d\,(a-\epsilon)^d}{d}\left[\text{Vol}(a) - \text{Vol}(a-\epsilon)\right]}{\frac{S_d\, a^d}{d}}$

$= 1 - \left(\frac{a-\epsilon}{a}\right)^d$

$= 1 - \left[1 - \frac{\epsilon}{a}\right]^d$

$1 - \frac{\epsilon}{a} < 1 \quad \therefore \left(1 - \frac{\epsilon}{a}\right)^d$ is a product of fractions < 1 $[0<]$

$\therefore \lim_{d\to\infty} \left(1 - \frac{\epsilon}{a}\right)^d = 0$

$\therefore f \to 1$ as $d \to \infty$

(ii) $d=2 \quad 1 - (0.99)^2 = 1 - 0.9801 = 0.0199$

$d=10 \quad 1 - (0.99)^{10} \simeq 1 - 0.90438 \simeq 0.0956$

$d=1000 \quad 1 - (0.99)^{1000} \simeq 1 - 4.317 \times 10^{-5} \simeq 0.999956\,8$

(iii) $d=2 \quad (1/2)^2 = 1/4 = 0.25$

$d=10 \quad (1/2)^{10} = 0.000976562\,5$

$d=1000 \quad (1/2)^{1000} \simeq 9.33 \times 10^{-302}$

4 $p(\|x\|) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left[\frac{-\|x\|^2}{2\sigma^2}\right]$ but $\|x\|^2 = r^2$

$\Rightarrow p(r) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left[\frac{-r^2}{2\sigma^2}\right]$

infinitesimal probability mass = $p(r)\,dV$

$= p(r)\, S_r\, dr$

$= \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left[\frac{-r^2}{2\sigma^2}\right] S_d\, r^{d-1} \epsilon$ [given $dr = \epsilon$]

$\therefore = \frac{S_d\, r^{d-1}}{(2\pi\sigma^2)^{1/2}} \exp\left[\frac{-r^2}{2\sigma^2}\right] \epsilon$

$\uparrow$

$p(r)$

$\frac{dp(r)}{dr} = 0 \Rightarrow (d-1) r^{d-2} \exp\left[\frac{-r^2}{2\sigma^2}\right] = r^{d-1} \exp\left[\frac{-r^2}{2\sigma^2}\right] + \frac{1}{2\sigma^2} 2r$

$\Rightarrow \frac{d-1}{r} = \frac{2r}{2\sigma^2}$

$\Rightarrow r^2 = \sigma^2 (d-1) \Rightarrow r = \sigma\sqrt{d-1}$

when d is large $r \simeq \sigma\sqrt{d}$

$p(\hat{r}+\epsilon) = \frac{S_d\, (\hat{r}+\epsilon)^{d-1}}{(2\pi\sigma^2)^{1/2}} \exp\left[-\frac{(\hat{r}+\epsilon)^2}{2\sigma^2}\right]$

$= \frac{S_d}{(2\pi\sigma^2)^{1/2}} (\hat{r}+\epsilon)^{d-1} \exp\left[\frac{-(\hat{r}+\epsilon)^2}{2\sigma^2}\right]$

$\frac{p(\hat{r}+\epsilon)}{p(\hat{r})} = \left(\frac{\hat{r}+\epsilon}{\hat{r}}\right)^{d-1} \exp\left[\frac{-1}{2\sigma^2}\left[\hat{r}^2 + \epsilon^2 + 2\hat{r}\epsilon - \hat{r}^2\right]\right]$

$= \left(1 + \frac{\epsilon}{\hat{r}}\right)^{d-1} \exp\left[\frac{-1}{2\sigma^2}\left[\epsilon^2 + 2\hat{r}\epsilon\right]\right]$

Perceptron.

1. The decision boundary is:

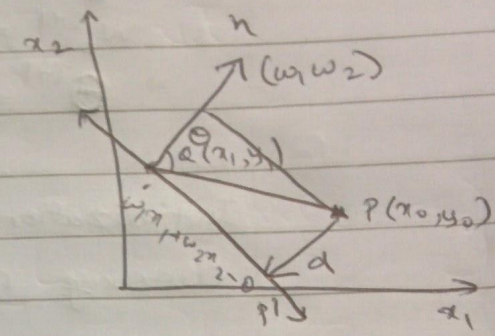
$$y(x) = w^T x - \theta = 0$$

$$\Rightarrow w_1 x_1 + w_2 x_2 - \theta = 0$$

(Proof from Wikipedia).

The normal is $(w_1, w_2) = n$

Let Q be any point (x_1, y_1) & P (x_0, y_0). & d perpendicular distance.



$$\vec{QP} \cdot \hat{n} = \|\vec{QP}\| \|n\| \cos\theta \qquad \text{but } \vec{QP} \cos\theta = d$$

$$\therefore d = \frac{\vec{QP} \cdot \hat{n}}{\|n\|}$$

$$= \frac{(x_0 - x_1, y_0 - y_1) \cdot (w_1, w_2)}{\sqrt{w_1^2 + w_2^2}}$$

$$= \frac{w_1 x_0 + w_2 y_0 - w_1 x_1 - w_2 y_1}{\sqrt{w_1^2 + w_2^2}}$$

but $w_1 x_1 + w_2 y_1 - \theta = 0$ [x_1, y_1 lie on the line]

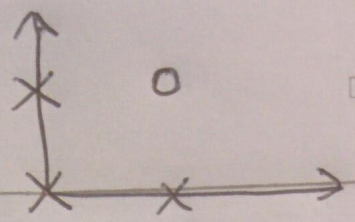
$$\therefore d = \frac{w_1 x_0 + w_2 y_0 - \theta}{\|W\|} \qquad \|W\| = \sqrt{w_1^2 + w_2^2}$$

substituting $x_0, y_0 = 0, 0$

$$d = \frac{-\theta}{\|W\|} = \frac{w_0}{\|W\|}$$

[NOTE: distance can be defined to be negative for point (0,0), it is just a matter of convention]

or $\frac{-w_0}{\|W\|}$ [according to the convention]



Date

2. (i) Perceptron Learning Rule:

$$w^{(\tau+1)} = w^{(\tau)} + \eta \phi_n t_n$$

[Assuming linear features]

$$\Rightarrow \begin{aligned} w_1^{\tau+1} &= w_1^{\tau} + \eta x_1^n t^n \\ w_2^{\tau+1} &= w_2^{\tau} + \eta x_2^n t^n \\ w_0^{\tau+1} &= w_0^{\tau} + \eta x_0^n t^n \end{aligned}$$

[for misclassified patterns only]

$$\Rightarrow \theta^{\tau+1} = \theta^{\tau} - \eta t^n \quad [w_0 = -\theta,\ x_0 = 1\ [\text{bias}]]$$

Initialization: $w_1^0 = 0,\ w_2^0 = 0 \quad \theta^0 = 0$

choosing $\eta = 1$ arbitrarily. $C_1 \equiv \overset{\text{Teach}}{1} \quad C_0 \equiv \overset{\text{Teach}}{-1}$

x_1	x_2	out	Teach	w_1	w_2	θ
0	0	0	-1	0	0	1
0	1	-1	1	0	1	0
0	0	0	-1	0	1	1
1	0	-1	1	1	1	0
0	0	0	-1	1	1	1

It is easy to check $w_1 = 1, w_2 = 1\ \ \theta = 1$ has converged.

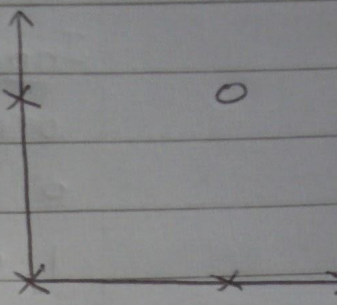
Input	$w_1 x_1 + w_2 x_1 - \theta$	Class	output.
0 0	-1	0	

Assuming $w_1x_1 + w_2x_2 - \theta \geq 0 \Rightarrow C_1$ [output = 1]

or $w_1x_1 + w_2x_2 - \theta < 0 \Rightarrow C_0$ [output = 0]

2(a) Perceptron learning Rule:

$\vec{w}^{\tau+1} = \vec{w}^{\tau} + \eta \vec{x} t_n$ [Assuming linear features].



Let us choose target/output = 1 = 1

target/output = 0 = −1

to be consistent with perceptron learning.

& $\eta = 1$ as given in the question.

$\therefore \quad w_0^{\tau+1} = w_0^{\tau} + x_0^n t^n$

$\Rightarrow \quad \theta_0^{\tau+1} = \theta_0^{\tau} - t^n \quad \ldots (1)$ [$w_0 = -\theta$, $x_0 = 1$ [bias]]

$w_1^{\tau+1} = w_1^{\tau} + x_1^n t^n$

$w_2^{\tau+1} = w_2^{\tau} + x_2^n t^n$

[$\because$ teach = targ

for misclassified patterns only.

(b) Initialization: $w_1^0 = 0 \quad w_2^0 = 0 \quad \theta_0^0 = 0$

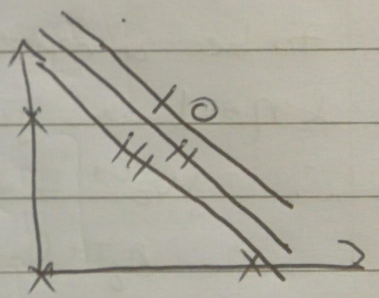
x_1	x_2	out	Teach	w_1	w_2	θ
1	1	$0 \equiv C_1$	$-1 \equiv C_0$	−1	−1	1
0	0	$-1 \equiv C_0$	$1 \equiv C_1$	−1	−1	0
0	1	$-1 \equiv C_0$	$1 \equiv C_1$	−1	0	−1
1	0	$0 \equiv C_1$	No change - correct classification			
1	1	$0 \equiv C_1$	$-1 \equiv C_0$	−2	−1	0
1	0	$-2 \equiv C_0$	$1 \equiv C_1$	−1	−1	−1
0	0	$+1 \equiv C_1$	No change.			

We see that it has converged as all patterns are correctly classified.

Given $w_1, w_2, \theta = -1, -1, -1$

Date

Input	$w_1x_1 + w_2x - \theta$	Class		output.
0 0	1	C_1	✓	1
0 1	0	C_1	✓	1
1 0	0	C_1	✓	1
1 1	-1	C_0	✓	0



(c) No it is not unique.

Any of the lines shown in the diagram are valid seperating ~~lines~~ line segments.

They can be achieved if $\eta \neq 1$.

Date

Logistic & Softmax Regression

1. $$-\frac{\partial}{\partial w_j} E^n(w) = +\frac{\partial}{\partial w_j}\left[t^n \ln y^n + (1-t^n)\ln(1-y^n)\right]$$

$$= \frac{t^n}{y^n}\frac{\partial y^n}{\partial w_j} + \frac{1-t^n}{1-y^n}(-1)\cdot\frac{\partial y^n}{\partial w_j}$$

But: $$\frac{\partial y^n}{\partial w_j} = \frac{\partial}{\partial w_j}\left[\frac{1}{1+e^{-\sum x_i^n w_i}}\right] = \frac{-1}{(1+e^{-\sum x_i^n w_i})^2} e^{-\sum x_i^n w_i}\left[-x_j^n\right]$$

$$= y_n(1-y_n)(x_j^n)$$

$$\therefore -\frac{\partial E^n}{\partial w_j} = \frac{t^n}{y^n} y^n(1-y^n)\left[-x_j^n\right] + \frac{1-t^n}{1-y^n} y^n(1-y^n)\left[-x_j^n\right]$$

$$= \left[t^n(1-y^n) + (1-t^n)(y^n)\right]\left[-x_j^n\right]$$

$$= \left[t^n - t^n y^n - y^n + t^n y^n\right]\left[-x_j^n\right]$$

$$= +\left(t^n - y^n\right)x_j^n$$

2. $$\frac{\partial}{\partial w_{jk}} E^n(w) = \frac{\partial}{\partial w_{jk}}\left[\sum_{l=1}^{c} t_l^n \ln y_l^n\right]$$

$$= \sum_{l=1}^{c} t_l^n \frac{\partial \ln y_l^n}{\partial w_{jk}}$$

$$= \sum_{l=1}^{c} \frac{t_l^n}{y_l^n}\frac{\partial y_l^n}{\partial w_{jk}}$$

$$\frac{\partial}{\partial w_{jk}} y_l^n = \frac{\partial}{\partial w_{jk}} \frac{e^{a_l^n}}{\sum_{l'} e^{a_{l'}^n}}$$

Indicator $I_{kl} = 1$ when $k = l$

$$= \frac{I_{kl}\, e^{a_l^n}\, x_j^n}{\sum_{l'} e^{a_{l'}^n}} + \frac{e^{a_l^n}\, e^{a_k^n}\, x_j^n}{\left(\sum_{l'} e^{a_{l'}^n}\right)^2}$$

$\frac{e^{a_l^n}}{\sum_{l'} e^{a_{l'}^n}} = y_l^n$

$\#\ y_k^n y_l^n$

What makes you happy?
#HappyCollegeDays

Date

$$\frac{\partial E^n}{\partial w_{jk}} = \sum_{l=1}^{c} \frac{t_l^n}{y_l^n} \left[I_{kl} y_l - x_j^n + y_l y_k x_j^n \right]$$

$$-\frac{\partial E}{\partial w_{jk}} = \sum_{l=1}^{c} t_l^n \left[I_{kl} - y_k \right] x_j^n$$

$$= x_j^n \left[\left(\sum_{l=1}^{c} t_l^n I_{kl} \right) - \left(\sum_{l=1}^{c} t_l^n y_k^n \right) \right]$$

$$= x_j^n \left[t_k^n - y_k^n \sum_{l=1}^{c} t_l^n \right]$$

$\hookrightarrow \sum t_l = 1$ by definitio

$$= (t_k^n - y_k^n)\, x_j^n$$