Assignment 2

CMSC 678 — Introduction to Machine Learning

Faisal Rasheed Khan

VB02734

vb02734@umbc.edu

Question 1---

1. (20 points) Consider the following joint distribution p(x, y):

y

X =

	-1	0	1
a	0.3	0.02	0.03
b	0.03	0.2	0.02
c	0.02	0.03	0.2
d	0.05	0.05	0.05

(a) Compute the marginal distribution p(x).

A. Marginal Distribution of X,

p(X) = $\sum_y p(X, Y = y)$

p(X=a) = p(a,-1) + p(a,0) + p(a,1)

p(X=a) = 0.3 + 0.02 + 0.03

p(X=a) = 0.35

p(X=b) = p(b,-1) + p(b,0) + p(b,1)

p(X=b) = 0.03 + 0.2 + 0.02

p(X=b) = 0.25

p(X=c) = p(c,-1) + p(c,0) + p(c,1)

p(X=c) = 0.02 + 0.03 + 0.2

p(X=c) = 0.25

p(X=d) = p(d,-1) + p(d,0) + p(d,1)

p(X=d) = 0.05 + 0.05 + 0.05

p(X=d) = 0.15

	a	b	c	d
P(X)	0.35	0.25	0.25	0.15

(b) Compute the marginal distribution p(y).

A. Marginal Distribution of Y,

p(Y) = $\sum_x p(Y, X = x)$

p(Y=-1) = p(-1,a) + p(-1,b) + p(-1,c) + p(-1,d)

p(Y=-1) = 0.3 + 0.03 + 0.02 + 0.05

p(Y=-1) = 0.4

p(Y=0) = p(0,a) + p(0,b) + p(0,c) + p(0,d)

p(Y=0) = 0.02 + 0.2 + 0.03 + 0.05

p(Y=0) = 0.3

p(Y=1) = p(1,a) + p(1,b) + p(1,c) + p(1,d)

p(Y=1) = 0.03 + 0.02 + 0.2 + 0.05

p(Y=1) = 0.3

	-1	0	1
P(Y)	0.4	0.3	0.3

(c) For y = 0, compute p(x|y = 0).

A. $p(x \mid y) = \frac{p(x,y)}{p(y)}$

$p(x \mid y=0) = \frac{p(x,y=0)}{p(y=0)}$

$p(a \mid 0) = \frac{p(a,0)}{p(0)} = \frac{0.02}{0.3} = 0.067$

$p(b \mid 0) = \frac{p(b,0)}{p(0)} = \frac{0.2}{0.3} = 0.67$

$p(c \mid 0) = \frac{p(c,0)}{p(0)} = \frac{0.03}{0.3} = 0.1$

$p(d \mid 0) = \frac{p(d,0)}{p(0)} = \frac{0.05}{0.3} = 0.167$

	a	b	c	d
P(X | Y=0)	0.067	0.67	0.1	0.167

(d) Let x1, x2, x3 be three different (empirical) samples, where x1 = a, x2 = c, and x3 = d. Let p(y|xi) be the posterior distribution of y given the particular value of xi , e.g., for x2, this would be p(y|x = c). Compute

$$\sum_{i=1}^{N} E_{y\sim p(y|xi)} \, [\log p(y, x_i)]$$

A. = $\sum_{i=1}^{N} \sum_{y} p\,(\mathrm{y} \mid \mathrm{xi})\,[\log \mathrm{p}(\mathrm{y}, \mathrm{xi}\,)]$

$= \sum_{i=1}^{N} \mathrm{p}(-1 \mid \mathrm{xi})\,[\log \mathrm{p}(-1, \mathrm{xi}\,)] + \mathrm{p}(0 \mid \mathrm{xi})\,[\log \mathrm{p}(0, \mathrm{xi}\,) + \mathrm{p}(-1 \mid \mathrm{xi})\,[\log \mathrm{p}(0, \mathrm{xi}\,)]$

$= \quad \mathrm{p}(-1 \mid \mathrm{a})\,[\log \mathrm{p}(-1, \mathrm{a}\,)] + \mathrm{p}(0 \mid \mathrm{a})\,[\log \mathrm{p}(0, \mathrm{a}\,)] + \mathrm{p}(1 \mid \mathrm{a})\,[\log \mathrm{p}(1, \mathrm{a}\,)] +$

$\mathrm{p}(-1 \mid \mathrm{c})\,[\log \mathrm{p}(-1, \mathrm{c}\,)] + \mathrm{p}(0 \mid \mathrm{c})\,[\log \mathrm{p}(0, \mathrm{c}\,)] + \mathrm{p}(1 \mid \mathrm{c})\,[\log \mathrm{p}(1, \mathrm{c}\,)] +$

$\mathrm{p}(-1 \mid \mathrm{d})\,[\log \mathrm{p}(-1, \mathrm{d}\,)] + \mathrm{p}(0 \mid \mathrm{d})\,[\log \mathrm{p}(0, \mathrm{d}\,)] + \mathrm{p}(1 \mid \mathrm{d})\,[\log \mathrm{p}(1, \mathrm{d}\,)]$

$= \quad \frac{p(-1,a)}{p(a)}[\log \mathrm{p}(-1, \mathrm{a}\,)] + \frac{p(0,a)}{p(a)}\,[\log \mathrm{p}(0, \mathrm{a}\,)] + \frac{p(1,a)}{p(a)}\,[\log \mathrm{p}(1, \mathrm{a}\,)] +$

$\frac{p(-1,c)}{p(c)}\,[\log \mathrm{p}(-1, \mathrm{c}\,)] + \frac{p(0,c)}{p(c)}\,[\log \mathrm{p}(0, \mathrm{c}\,)] + \frac{p(1,c)}{p(c)}\,[\log \mathrm{p}(1, \mathrm{c}\,)] +$

$\frac{p(-1,d)}{p(d)}[\log \mathrm{p}(-1, \mathrm{d}\,)] + \frac{p(0,d)}{p(d)}\,[\log \mathrm{p}(0, \mathrm{d}\,)] + \frac{p(1,d)}{p(d)}\,[\log \mathrm{p}(1, \mathrm{d}\,)]$

$= \quad \frac{0.3}{0.35}\log 0.3 + \frac{0.02}{0.35}\log 0.02 + \frac{0.03}{0.35}\log 0.03 +$

$$\frac{0.02}{0.25}\log 0.02 + \frac{0.03}{0.25}\log 0.03 + \frac{0.2}{0.25}\log 0.2 +$$

$$\frac{0.05}{0.15}\log 0.05 + \frac{0.05}{0.15}\log 0.05 + \frac{0.03}{0.15}\log 0.05$$

$$= \quad \frac{1}{0.35}(-0.9 - 3.89 - 3..47) +$$

$$\frac{1}{0.25}(-3.89 - 3.47 - 0.32) +$$

$$\frac{1}{0.15}(-0.149 - 0.149 - 0.149)$$

$$= \quad -23.6 - 30.72 - 2.99$$

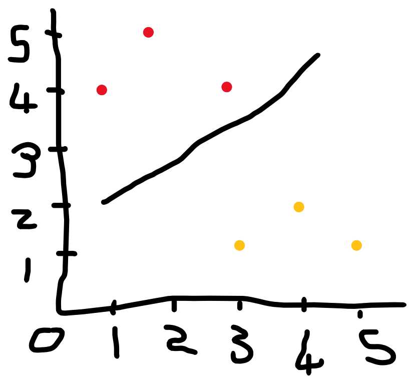
$$= \quad -57.31$$

Question 2---

2. (20 points) For 1 ≤ i ≤ N, let xi ∈ R 2 be a two-dimensional input, and yi ∈ {−1, +1} the binary label for xi .
   (a) Describe, in both words and with a concrete example, when the perceptron algorithm would be guaranteed to converge (find an optimal solution to the training set). For the concrete example, give N ≥ 5 data points {(xi , yi)} for which the perceptron algorithm would converge.

A. Perceptron Algorithm would guarantee to converge if the data is linearly separable. The data should be separated by a good hyperplane. If the data is linearly separable then perceptron algorithm guarantees to converge

Consider the points,



Nicely Separable

X	y_i
(1,4)	-1
(2,5)	-1
(3,4)	-1
(3,1)	1
(4,2)	1
(5,1)	1

🔴 $y_i = -1$

🟡 $y_i = 1$

t=	Weights, $w^{(t)}$	Data point $x_{(t\%N)+1}$	Activation score	Predicted Value $\hat{y}$	New Weights, $w^{(t+1)}$
0	(0,0,…0)	(1,4)	0(1)+0(4)=0	-1	No update, [0,0]
1	[0,0]	(2,5)	0(2)+0(5)=0	-1	No update, [0,0]
2	[0,0]	(3,4)	0(3)+0(4)=0	-1	No update, [0,0]
3	[0,0]	(3,1)	0(3)+0(1)=0	-1	[0,0]+[3,1]=[3,1]
4	[3,1]	(4,2)	3(4)+1(2)=14	1	No update, [3,1]
5	[3,1]	(5,1)	3(5)+1(1)=16	1	No update, [3,1]
6	[3,1]	(3,1)	3(3)+1(1)=10	1	No update, [-0.5,1.5]

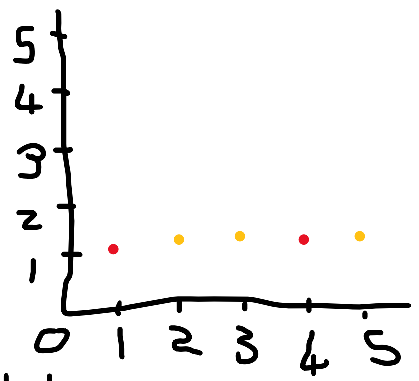
for misclassified data

Therefore the data is linearly separable and no misclassified data.

(b) Describe, in words and with a concrete example, when the perceptron algorithm would not be guaranteed to converge. For the concrete example, give N ≥ 5 data points {(xi , yi)} for which the perceptron algorithm would not converge.

A. Perceptron Algorithm would not guarantee to converge if the data is not linearly separable. If the data is not linearly separable then perceptron algorithm doesn't guarantee to converge

Consider the points,



Not Separable

X	y_i
(1,1)	-1
(2,1)	1
(3,1)	1
(4,1)	-1
(5,1)	1

🟥 $y_i = -1$

🟨 $y_i = 1$

t=	Weights, $w^{(t)}$	Data point $x_{(t\%N)+1}$	Activation score	Predicted Value $\hat{y}$	New Weights, $w^{(t+1)}$
0	(0,0,…0)	(1,1)	0(1)+0(1)=0	-1	No update, [0,0]
1	[0,0]	(2,1)	0(2)+0(1)=0	-1	[0,0]+[2,1]=[2,1]
2	[2,1]	(3,1)	2(3)+1(1)=7	1	No update, [2,1]
3	[2,1]	(4,1)	2(4)+1(1)=9	1	[2,1]+(-1)[4,1]=[-2,0]
4	[-2,0]	(5,1)	-2(5)+0(1)=-10	-1	[-2,0]+[5,1]=[3,1]
5	[3,1]	(2,1)	6(2)+1(1)=13	1	No update, [3,1]
6	[3,1]	(4,1)	3(4)+1(1)=13	1	[3,1]+(-1)[4,1]=[-1,0]
7	[-1,0]	(5,1)	-1(5)+0(1)=-5	-1	[-1,0]+[5,1]=[4,1]

The two misclassification points (4,1) and (5,1) never converges the perceptron algorithm

(c) Let the dataset D be

i=	x_i	y_i
1	-1,1	1
2	-1,-1	-1
3	0.5,0.5	1
4	1,-1	1
5	0.5,-1	-1

Run, by hand, the perceptron algorithm on these five data points. Record your computations for each time step (data point examined, activation score for that data point and current weight vector, the predicted value, and the new weights) in the following table.

For t = 0, please fix w (t) to have the correct dimensionality. The strange indexing x(t%N)+1 simply says to iterate through the data in the order provided above. Note the second column of each row should equal the last column of the previous row

A. Perceptron Algorithm,

Initially weights are 0 i.e. w = (0,0,…,0)

Y be the target value

$\hat{y}$ be the predicted value

$$\hat{y} = \begin{cases} +1 & \text{if } w^T x_i > 0 \\ -1 & \text{if } w^T x_i \leq 0 \end{cases}$$

If(y = $\hat{y}$) do nothing

If(y≠$\hat{y}$) then

$$w \leftarrow w + y_i x_i$$

We consider bias(b)=0 and learning rate(α)=1 below while calculating,

t=	Weights, $w^{(t)}$	Data point $x_{(t\%N)+1}$	Activation score	Predicted Value $\hat{y}$	New Weights, $w^{(t+1)}$
0	(0,0,…0)	(-1,1)	0(-1)+0(1)=0	-1	[0,0] + 1*[-1,1] = [-1,1]
1	[-1,1]	(-1,-1)	-1(-1)+1(-1)=0	-1	No update, [-1,1]
2	[-1,1]	(0.5,0.5)	-1(0.5)+1(0.5) =0	-1	[1,1]+1*[0.5,0.5]= [-0.5,1.5]
3	[-0.5,1.5]	(1,-1)	-0.5(1)+1.5(-1)=-2	-1	[-0.5,1.5]+1*[1,-1]=[0.5,0.5]
4	[0.5,0.5]	(0.5,-1)	0.5(0.5)+0.5(-1)=-0.25	-1	No update, [0.5,0.5]
5	[0.5,0.5]	(-1,1)	0.5(-1)+ 0.5(1)=0	-1	[0.5,0.5]+1*[-1,1]=[-0.5,1.5]
6	[-0.5,1.5]	(-1,-1)	-0.5(-1)+1.5(-1)=-1	-1	No update, [-0.5,1.5]

Question 3--

3. (35 points) First, let f(x) be a K-dimensional feature vector, i.e., f(x) ∈ R^K . The particular meaning of x doesn't matter for this question: it could be an image, or a text document, or collection of robotic sensor readings. In class, we discussed one method of turning a (binary) linear model into a probabilistic (binary) classifier: given vector weights w ∈ R^K , we pass our linear scoring model w^T f(x) through the sigmoid function σ(z) = 1 /(1+exp (−z)) and use the following decision function:

   output "class 1" if σ(w^T f(x)) ≥ 0.5

   output "class 2" if σ(w^T f(x)) < 0.5. [Eq-1]

   This question considers the multi-class extension of that model. Second, assume we have L classes, with corresponding weights θl (1 ≤ l ≤ L) per class. We can group these weights into a matrix θ ∈ $R^{L \times K}$. Notice that each θl is a K-dimensional vector. Consider a probabilistic linear classification model p($\hat{y}$|x), computed as

   $$p(y = \hat{y}|x) = \frac{\exp(\theta_{\hat{y}}^T f(x))}{\sum_{y^l} \exp(\theta_{y^l}^T f(x))} , \qquad \text{[Eq-2]}$$

   where y^ and y^l are potential values for the label.

Maximize log likelihood $(\arg\max)_y$ p(y|xi) = $(\arg\max)_y$ log p(y|xi) [Eq-3]

Cross entropy loss, $l^{xent} = -\sum_{j \in y} \vec{y}^*[j]$ log p(y = j|xi). [Eq-4]

(a) Argue that a binary instantiation (i.e., L = 2) of model [Eq-2] provides the same decision function as [Eq-1]. In arguing this, you do not need to prove it in a strict mathematical sense, but you need to make it clear that you understand it. (Imagine how you would explain it to a classmate.)

A. The Eq2 follows a conditional distribution for 2 instances L{1,2}.
The probability sum of both the individual instances over the distribution x is 1.
Therefore we can give the same decision function as of the above.

Consider a binary instantiation(L=2) model for Eq(2).

$$p(y = 1|x) = \frac{\exp(\theta_1^T f(x))}{\exp(\theta_1^T f(x)) + \exp(\theta_2^T f(x))}$$

$$p(y = 2|x) = \frac{\exp(\theta_2^T f(x))}{\exp(\theta_1^T f(x)) + \exp(\theta_2^T f(x))}$$

$0 \leq p(y = 1|x) \leq 1$
+
$0 \leq p(y = 2|x) \leq 1$
=1
i.e. $p(y = 1|x) + p(y = 1|x) = 1$

Generalized Linear Model,

$$Y = \sum_k \theta_k * f_k(x)$$

For class i≠k

$$\frac{\log p(y=i)}{\log p(y=k)} = Y + b + constant$$

Considering the likelihood,
For L =2,
$\frac{\log p(y=1)}{\log p(y=0)} = \theta$ f(x) for which SoftMax function provides the Eq1 decision boundary

Applying SoftMax function for the above σ(θ f(x)) gives boundary
output "class 1" if σ(θ f(x)) ≥ 0.5
output "class 2" if σ(θ f(x)) < 0.5.

Therefor we can define a decision function which is same as Eq2.
Predicted class = $argmax_y$ p(y|x)

Class 1	if $p(y = 1|x) \geq 0.5$
Class 2	if $p(y = 1|x) < 0.5$
Or	
Class 2	if $p(y = 2|x) \geq 0.5$
Class 1	if $p(y = 2|x) < 0.5$

(b) Show that, for the probabilistic classifier [Eq-2], maximizing the conditional log-likelihood ([Eq-3]) is the same as minimizing cross-entropy loss ([Eq-4]).

A. We have,
Maximize log likelihood (arg max)$_y$ p(y|xi) = (arg max)$_y$ log p(y|xi)
Cross entropy loss, $l^{xent} = -\sum_{j\in y} \vec{y}^*[j]$ log p(y = j|xi).

$\vec{y}^*$ is a one hot vector with all values zero except one value which is 1 (in the above loss equation its j.
So multiplying $\vec{y}^*$ with the terms, gives only one value where $\vec{y}^*$ vector value is defined as 1.
$\vec{y}^*[j]$=1
Therefore,
Cross entropy loss, $l^{xent} = -\sum_{j\in y} \vec{y}^*[j]$ log p(y = j|xi)
$l^{xent} = -$ log p(y = j|xi)
The above cross entropy we got after applying the values of vector $\vec{y}^*$ gives equation which is same in maximizing conditional log-likelihood.
Therefore, maximizing the conditional log-likelihood is the same as minimizing cross-entropy loss

(c) Given N data points {(x1, y1), . . . ,(xN , yN)}, formulate the empirical risk objective for [Eq-2] using cross-entropy loss. You can think of this as filling out the following template

$$opt_{\clubsuit} \frac{1}{N}\sum_{i=1}^{N} \blacklozenge$$

L(♣) , [Eq-5]

where you must specify

- opt: the optimization goal (e.g., max, min, or something else)
- ♣: the variables or values being optimized
- ♦: the function (and its arguments) being optimized.

A. $min_\theta \frac{1}{N}\sum_{i=1}^{N} L(\vec{y}^*, p(y \mid xi))$

Opt : here optimizing goal is to minimize the function L(θ)
♣: weights of the probabilistic classifier θ being optimized
♦: the function $L(\vec{y}^*, p(y \mid xi)$ (and its arguments θ) being optimized.

$$\boldsymbol{min_\theta \frac{1}{N}\sum_{i=1}^{N} - \sum_{j\in y}\vec{y}^*[j]\ \log p(y = j|xi)}$$

$$min_\theta \frac{-1}{N}\sum_{i=1}^{N} \log p(y = j|xi)$$

$\vec{y}^*$ is a one hot vector with all values zero except one value which is 1 (in the above loss equation its j.
So multiplying $\vec{y}^*$ with the terms, gives only one value where $\vec{y}^*$ vector value is defined as 1.
$\vec{y}^*[j]$=1
Therefore,

$$\text{L}(\theta) = min_\theta \frac{-1}{N}\sum_{i=1}^{N} \log \frac{\exp(\theta_{\hat{y}}^T f(x))}{\sum_{y'} \exp(\theta_{y'}^T f(x))}$$

(d) Derive the gradient ∇♣L(♣).

A. $\text{L}(\theta) = min_\theta \frac{1}{N}\sum_{i=1}^{N} - \sum_{1\ to\ j} \vec{y}^*[j] \log \text{p}(\text{y} = \text{j}|\text{xi})$
$\text{L}(\theta) = min_\theta \frac{1}{N}\sum_{i=1}^{N} - \sum_{1\ to\ j} \vec{y}^*[j] \log \text{p}(\text{y} = \text{j}|\text{xi})$

$\text{L}(\theta) = min_\theta \frac{-1}{N}\sum_{i=1}^{N} 1[\vec{y}^*[\text{j}], \text{j} = \text{i}] \log \text{p}(\text{y} = \text{j}|\text{xi})$

$\text{L}(\theta) = min_\theta \frac{-1}{N}\sum_{i=1}^{N} [\overrightarrow{y}^*[\text{j}], \text{j} = \text{i}] \log \frac{\exp(\theta_{\hat{y}}^T f(x))}{\sum_{y'} \exp(\theta_{y'}^T f(x))}$

$\nabla_\theta \text{L}(\theta) = \nabla_\theta\ (\ min_\theta \frac{-1}{N}\sum_{i=1}^{N} \log \text{p}(\text{y} = \text{j}|\text{xi})\ )$

$\nabla_\theta \text{L}(\theta) = \nabla_\theta\ (\ min_\theta \frac{-1}{N}\sum_{i=1}^{N} \log \frac{\exp(\theta_{\hat{y}}^T f(x))}{\sum_{y'} \exp(\theta_{y'}^T f(x))})$

$\nabla_\theta \text{L}(\theta) = \nabla_\theta\ (\ min_\theta \frac{-1}{N}\sum_{i=1}^{N} \log \exp(\theta_{\hat{y}}^T f(x) - \log \sum_{y'} \exp(\theta_{y'}^T f(x))$

$\nabla_\theta \text{L}(\theta) = \nabla_\theta\ (\ min_\theta \frac{-1}{N}\sum_{i=1}^{N} (\theta_{\hat{y}}^T f(x) - \log \sum_{y'} \exp(\theta_{y'}^T f(x))$

$\nabla_\theta \text{L}(\theta) = min_\theta \frac{-1}{N}\sum_{i=1}^{N} \nabla\theta(\theta_{\hat{y}}^T f(x)) - \nabla\theta(\log \sum_{y'} \exp(\theta_{y'}^T f(x)))$

$\nabla_\theta \text{L}(\theta) = min_\theta \frac{-1}{N}\sum_{i=1}^{N} \nabla\theta\big(\theta_1 f(x1) + \theta_2 f(x2) + \cdots \theta_n f(xn)\big) - \frac{1}{\sum_{y'} \exp(\theta_{y'}^T f(x))} \nabla\theta(\sum_{y'} \exp(\theta_{y'}^T f(x)))$

Doing the second part of the derivative
$\nabla\theta(\sum_{y'} \exp(\theta_{y'}^T f(x)) = \nabla\theta\ (\exp(\theta_1 f(x1) + \exp(\theta_2 f(x2) + \cdots + \exp(\theta_n f(xn))$
$= (\exp(\theta_1 f(x1) + \exp(\theta_2 f(x2) + \cdots + \exp(\theta_n f(xn)))$
$\nabla\theta(\theta_1 f(x1) + \theta_2 f(x2) + \cdots + (\theta_n f(xn))$

Substituting the above derivative

$$\nabla_\theta L(\theta) = \ min_\theta \frac{-1}{N} \sum_{i=1}^{N} \nabla\theta\big(\theta_1 f(x1) + \theta_2 f(x2) + \cdots \theta_n f(xn)\big) - \frac{1}{\sum_{y|} \exp\,(\,\theta_{y|}^T\, f(x)\,)} \sum_{y|} \exp\,(\,\theta_{y|}^T\, f(x)\,))$$

$$\nabla_\theta L(\theta) = \ min_\theta \frac{-1}{N} \sum_{i=1}^{N} \nabla\theta\big(\theta_1 f(x1) + \theta_2 f(x2) + \cdots \theta_n f(xn)\big) - 1$$

$$\nabla_\theta L(\theta) = \ min_\theta \frac{-1}{N} \sum_{i=1}^{N} \nabla\theta(\theta^T f(xi)) - 1$$

Question 4--

4. (5 points) Implement a "most frequent" baseline classifier. This baseline identifies the label that was most common in training (int-train) and always produces that label during evaluation (regardless of the input).

(a) In your report, document the implementation (this should be fairly straight-forward).

(b) In your report, report how well this baseline does on int-dev.

A.

- The challenges I faced during the implementation was reading the data from the zip file.
- Converted the zip file to json and read using pandas.
- But panda library retrieved only one training not 70000 training instances. So understood the mnist dataset, inorder to gain knowledge about the dataset provided.
- After that I got used to the dataset provided.
- Split the training instances of x and y according to the requirement
- Accessing issues for the data, resolved by using numpy library
- sklearn provides DummyClassifier to use strategy of most frequent classifier.
- The accuracy on int-dev was 0.1135, it wasn't much accurate as it implements the mostfrequent baseliner class.

Question 5--

5. Implement a multiclass perceptron. Train it on int-train and evaluate it on int-dev. For this question, you may use computation accelerators (e.g., blas, numpy, MATLAB) but you may not use any existing perceptron implementation.

(a) In your report, discuss your implementation and convergence criteria (this should also be fairly brief). (

b) In your report, discuss the concrete, quantifiable ways you validated your implementation was correct. One such way could be to run it on the toy data in Question 2, though there are other ways too.

(c) In your report, experiment with at least three different configurations, where you train each configuration on int-train and then evaluate on int-dev. Document this internal devel opment progress through quantifiable and readable means, i.e., graphs and/or tables showing how different model configurations perform on int-dev. A configuration could include learn ing biases or not, the convergence criteria, or the input featurization, among others.

A.

- Successfully implemented the multiperceptron class according to the given requirements. The implementation of multi perceptron follows the binary perceptron implementation.
- Spilt the data according to the problem statement.
- I tried implementing the perceptron algorithm which trains the data points and weights of the input features are updated accordingly.
- The main criteria is to find the weight vector size, it should be of the shape of x and shape of the labels
- The weight vectors are calculated for every class j
- The weights are updated if the the predicted label is not equal to actual label.
- Weights are calculated with dot product of weight vector and input feature
- The challenges here I faced were the shape of the input vectors, the weight vector should be calculated according to that starting with 0.
- Some of the errors like type, size not match etc. resolved those by checking with the print statement and error resolving criteria mentioned below the code.
- The perceptron Algorithm got a score of 0.3378,but it fared poor for the test data
- The model is not fetching good score is because of the learning rate and bias ,and not good for those features.
- Learning rate I chose 1 and bias 0
- While applying epochs=50, some improvement was there. Score improved to 0.4 .
- Visualized the input data using matplotlib

Question 6---

6.

- Visualized the data using signr (xi) = 1 xi > r
- xi ≤ r
- I have used the perceptron model to train over the 60000 images and tested it with 10000 testing images.

- The accuracy the model reached is 0.851. The unnecessary features will be reduced by this algorithm.
- But the most-frequent based classifer showed the same accuracy as before i.e 0.1135
- This perceptron model performed with much accuracy than my model (0.3378) which I have implemented.

References:

Dataset:
https://www.csee.umbc.edu/courses/graduate/678/spring23/materials/mnist_rowmajor.jsonl.gz

Welcome to PyTorch Tutorials — PyTorch Tutorials 1.13.1+cu117 documentation

sklearn.dummy.DummyClassifier — scikit-learn 1.2.2 documentation

Python | Pandas Series.tolist() - GeeksforGeeksciml-v0_99-ch02.pdf

04-linear-perceptron-grad.pdf (umbc.edu)

03-prob-decision.pdf (umbc.edu)