CSL7020: Machine Learning-1

(Programming Assignment -1)

Submitted By

Name	KAVINKUMAR M.
RollNumber	M22AI565
Department	Mtech in DCS

M. Kavinkumar.
ML - Fractal 3

1) Perceptron.

Following training samples are given.

x_1	x_2	class
1	1	+1
-1	-1	-1
0	0.5	-1
0.1	0.5	-1
0.2	0.2	+1
0.9	0.5	+1

Assuming weights vectors of initial decision boundary $w^T x = 0$ as $w = [1, 1]$

Solve the following.

1) In how many steps perceptron learning algorithm will converge ?
2) what will be the final decision boundary ? Show step-wise update of weights vector using computation as well as hand drawn plot.

Solution :-

Given :

$w^T x = 0$;

$w = [1, 1]$

Therefore, $\begin{bmatrix} 1 \\ 1 \end{bmatrix} x = 0.$

$\Rightarrow \boxed{x_1 + x_2 = 0}$

$\Rightarrow \boxed{b = 0}$

$y_{in} = w_i^T x_i + b$

$= w_1 x_1 + w_1 x_2 + b$

Lets assume learning rate as 1

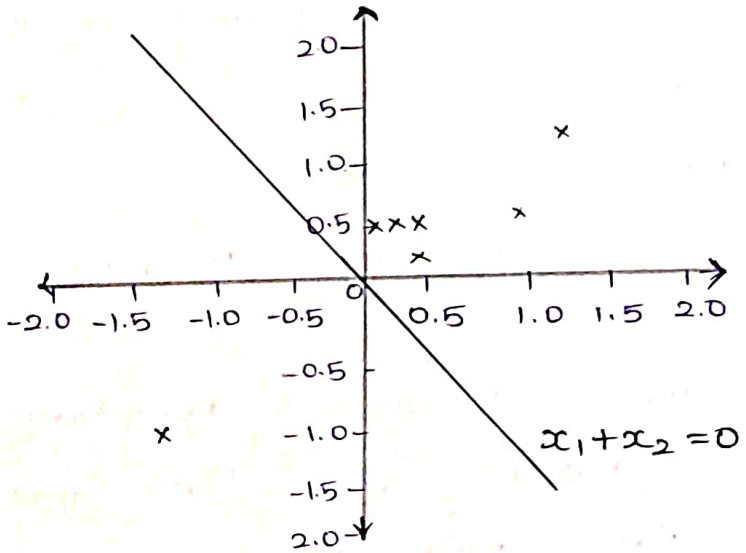
$$y = \begin{cases} 1 & \text{if } y_{in} > 0 \\ 0 & \text{if } y_{in} = 0 \\ -1 & \text{if } y_{in} < 0 \end{cases}$$

$\Delta w_1 = \alpha t x_1$

$\Delta w_2 = \alpha t x_2$

$\Delta b = \alpha t$

Initial decision boundary :-



Iteration 1

x_1	x_2	class (t)	y_{in}	y	Δw_1	Δw_2	Δb	w_1	w_2	b.
1	1	+1	2	+1	0	0	0	1	1	0.
-1	-1	-1	-2	-1	0	0	0	1	1	0
0	0.5	-1	0.5	+1	0	-0.5	-1	1	0.5	-1
0.1	0.5	-1	-0.65	-1	0	0	0	1	0.5	-1
0.2	0.2	+1	-0.1	-1	0.2	0.2	1	1.2	0.7	0
0.9	0.5	+1	1.43	+1	0	0	0	1.2	0.7	0

Iteration 2

x_1	x_2	t	y_{in}	y	Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	+1	1.9	+1	0	0	0	1.2	0.7	0
-1	-1	-1	-1.9	+1	0	0	0	1.2	0.7	0
0	0.5	-1	-0.78	+1	0	-0.5	-1	1.2	0.2	-1
0.1	0.5	-1	-0.72	-1	0	0	0	1.2	0.2	-1
0.2	0.2	+1	-0.72	-1	0.2	0.2	1	1.4	0.4	0
0.9	0.5	+1	1.46	+1	0	0	0	1.4	0.4	0

Iteration 3

x_1	x_2	Class(t)	y_{in}	y	Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	+1	1.8	+1	0	0	0	1.4	0.4	0
-1	-1	-1	-1.8	-1	0	0	0	1.4	0.4	0
0	0.5	-1	0.2	+1	0	-0.5	-1	1.4	-0.1	-1
0.1	0.5	-1	-0.81	-1	0	0	0	1.4	-0.1	-1
0.2	0.2	+1	-0.74	-1	0.2	0.2	1	1.6	0.1	0
0.9	0.5	+1	1.44	+1	0	0	0	1.6	0.1	0

Iteration 4

x_1	x_2	Class(t)	y_{in}	y	Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	+1	1.7	+1	0	0	0	1.6	0.1	0
-1	-1	-1	1.7	-1	0	0	0	1.6	0.1	0
0	0.5	-1	0.05	+1	0	0.5	-1	1.6	-0.4	-1
0.1	0.5	-1	-1.04	-1	0	0	0	1.6	-0.4	-1
0.2	0.2	+1	-0.76	-1	0.2	0.2	1	1.8	-0.2	0
0.9	0.5	+1	1.52	+1	0	0	0	1.8	-0.2	0

Iteration 5

x_1	x_2	t	y_{in}	y	Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	+1	1.6	+1	0	0	0	1.8	-0.2	0
-1	-1	+1	-1.6	+1	0	0	0	1.8	-0.2	0
0	0.5	-1	-0.1	-1	0	0	0	1.8	-0.2	0
0.1	0.5	-1	0.08	+1	-0.1	-0.5	-1	1.7	-0.7	-1
0.2	0.2	+1	-0.8	-1	0.2	0.2	0.2	1.9	-0.5	0
0.9	0.5	+1	1.46	+1	0	0	0	1.9	-0.5	0

Iteration 6

x_1	x_2	t	y_{in}	y	Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	+1	1.4	+1	0	0	0	1.9	-0.5	0
-1	-1	-1	-1.4	-1	0	0	0	1.9	-0.5	0
0	0.5	-1	-0.25	-1	0	0	0	1.9	-0.5	0
0.1	0.5	-1	-0.06	-1	0	0	0	1.9	-0.5	0
0.2	0.2	+1	0.28	+1	0	0	0	1.9	-0.5	0
0.9	0.5	+1	1.46	+1	0	0	0	1.9	-0.5	0

* The perceptron learning algorithm converged in 6 steps

* The final weight vector of the decision boundary is.

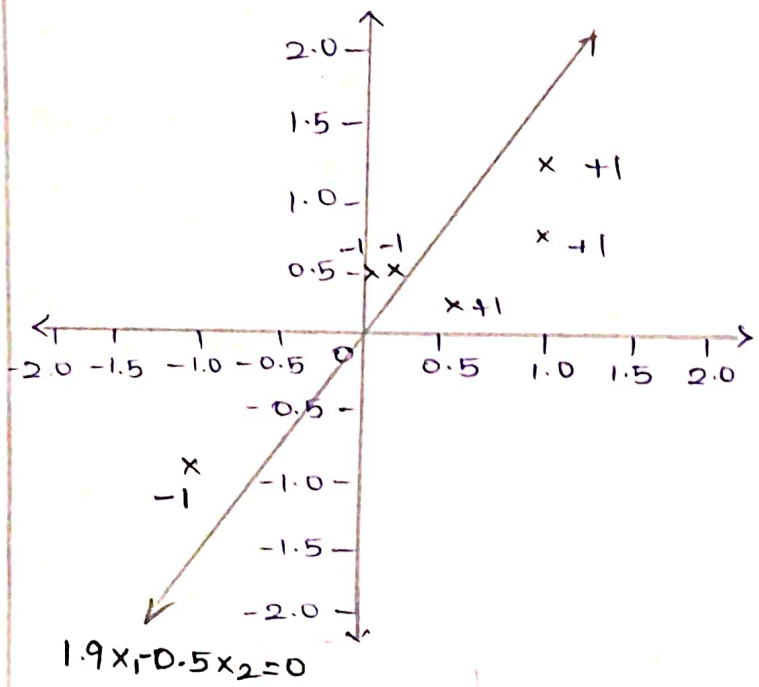
$w = [1.9, -0.5]$

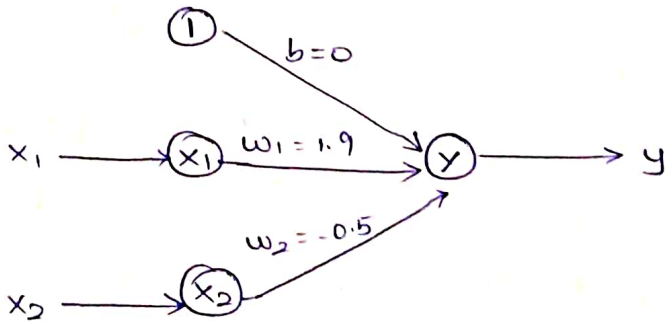
$\therefore 1.9 x_1 + (-0.5) x_2 = 0$

$\Rightarrow 1.9 x_1 - 0.5 x_2 = 0.$

Final Decision boundary.

* We can see that $1.9x_1 - 0.5x_2 = 0$ line seperate the two classes correctly.





Neural Network corresponding to the perceptron.

Q2) Neural Network Implementation

Gurmukhi Handwritten Digit Classification: Gurmukhi is one of the popular Indian scripts widely used in the Indian state of Punjab. In this part of the assignment, our goal is to develop a neural network solution (a simple NN, not a CNN) for classifying Gurmukhi digits.

Data Collection and Preparation

- Gurumukhi Handwritten Digit Dataset uploaded into Google Drive,Using python opencv packages to convert the train and test images into n-Dim Array
- Reshaped the image into 28x28 and converted into an ID vector with 784 Dim.
- All the train and validation images are converted into 784 Dim and Class labels are 0 to 9 . These class values are converted to binary format as 10 Dim vectors.

Neural Network Implementation

- Input Layer : The input data becomes 784 Dim vectors and class labels are 10 Dim vectors. So we can use this data to train the MLP to classify the Digits into the correct class.
- We can develop multiple combination of models with Different activation function , Different Optimizers etc.
- we are trying to reduce the categorical cross entropy loss because Multiclass Digit classification task
- Since we are classifying the image into multiple classes. Therefore we have use softmax classifier at the Output Layer

Model 1 : Softmax Classification

- The basic model we are developed using the Sgd optimizer and activation as softmax
- Using 20 epochs , we are getting 95% of test accuracy

Model 2 : MLP + Sigmoid Activation + SGD optimizer

- Two hidden layers are developed with 512 and 128 sigmoid activation units
- SGD optimizer are used for train the NN
- Using 20 epochs , we are getting 55% of test accuracy

Model 3 : MLP + Sigmoid Activation + ADAM optimizer

- Two hidden layers are developed with 512 and 128 sigmoid activation units
- Adam optimizer are used for train the NN
- Using 20 epochs , we are getting 94% of test accuracy

Model 4 : MLP + ReLU Activation + SGD optimizer

- Two hidden layers are developed with 512 and 128 ReLU activation units
- SGD optimizer are used for train the NN
- Using 20 epochs , we are getting 91% of test accuracy

Model 5 : MLP + ReLU Activation + ADAM optimizer

- Two hidden layers are developed with 512 and 128 ReLU activation units
- ADAM optimizer are used for train the NN
- Using 20 epochs , we are getting 94.9% of test accuracy

Model 6 : MLP + Batch Normalization on hidden layers +Sigmoid Activation + ADAM optimizer

- Two hidden layers are developed with 512 and 128 Sigmoid activation units
- ADAM optimizer are used for train the NN
- Batch normalization is introduced to avoid any internal covariance shift problems i.e Distribution of data is different for each layers
- Using 20 epochs , we are getting 80% of test accuracy

Model 7 : MLP + Drop out on hidden layers +Sigmoid Activation + ADAM optimizer

- Two hidden layers are developed with 512 and 128 Sigmoid activation units
- ADAM optimizer are used for train the NN
- Oftentimes we may encounter the overfitting problems in NN.To overcome this issue we may need to have enough difference in the train and test data. Drop out is the concept of dropping some connections in the training phase. So it helps to increase the validation accuracy
- Using 20 epochs , we are getting 83% of test accuracy

Model 8 : MLP + Drop out on hidden layers +RelLu Activation + ADAM optimizer

- Two hidden layers are developed with 512 and 128 ReLuactivation units
- ADAM optimizer are used for train the NN
- Oftentimes we may encounter the overfitting problems in NN.To overcome this issue we may need to have enough difference in the train and test data. Drop out is the concept of dropping some connections in the training phase. So it helps to increase the validation accuracy
- Using 20 epochs , we are getting 85% of test accuracy

Results

- We have developed the multiple models and identified it test Accuracy .Based on the accuracy metric the model 5 (MLP + ReLu + ADAM Optimizer) gives the higher accuracy.

Q3) CNN Implementation

You have to develop a CNN based classification architecture for classifying a given chart image to one of five chart classes, namely “Line”,“Dot Line”,“Horizontal Bar”,“Vertical Bar”, and “Pie” chart.

Data Collection and Preparation

- Training and validation images are uploaded into Google Drive. Python standard packages , numpy and pandas are used for loading the class labels CSV file.
- The Chart dataset contains very data points (1000 images with 5 classes). Data Augmentation is applied to the images to create the variance of the image
- Train and Test split is applied on 1000 data points with 80:20 ratio
- Keras Imagedatagenerator is utilized for train test split operations
- 800 images are utilized for training , 200 images are utilized for testing.
- Input Images are 28x28 RGB images

CNN implementation

- Input Layer is 28x28x3 Dimensions
- We have implemented the Two Layer.Convolutional Neural network with below architecture.
- The Hidden Layer 1 implemented with below details
    1. Filters : 32
    2. Kernel Size : 5x5
    3. Pool Size : 2x2
- The Hidden Layer 2 implemented with below details
    4. Filters : 16
    5. Kernel Size : 3x3
    6. Pool Size : 2x2
- BatchNormalization → Dropouts with 0.2 dropout rate → Flatten the output → Softmax activation at the output layer.
- ADAM optimizer is used .
- Since its classification problem we are trying to minimize the categorical cross entropy loss.
- Model was compiled and fitted using the fit_generator method . We are trying to run the 20 epochs to classify the images and accuracy values observed.
- End of 20 Epoch we observed that 97% Accuracy with 0.09 cross entropy loss

CNN implementation with pretrained Network VGG16

- We are using the VGG16 model for this Chart Classification task.
- All the pretrained VGG16 model weights are reused,and the output layer was replaced with a softmax classifier with 5 outputs.

- Model was compiled and fitted using the fit_generator method . We are trying to run the 5 epochs to classify the images and accuracy values observed.
- End of 5 Epoch we observed that 99% Accuracy with 0.003 cross entropy loss

Results

- We have developed the CNN Model and the pretrained model VGG16 for this Chart Classification task. The accuracy metric of the models are 97% to 99% respectively. The loss functions and its values are plotted using matplotlib. .The Pretrained model VGG16 gives highest accuracy for the chart classification task.