National University of Computer and Emerging Sciences

Lab Manual 10
Data Minning Lab

Department of Computer Science
FAST-NU, Lahore, Pakistan

Table of Contents

1 Objectives

After performing this lab, students shall be able to understand the following concepts:

- ✓ Single Perceptron
- ✓ Perceptron Learning Rule
- ✓ Training of a Perceptron
- ✓ Logic Gates Implementation using Perceptron Learning

2 Single Perceptron

A perceptron is a neural network unit (an artificial neuron) that does certain computations to detect features or business intelligence in the input data. A Perceptron is an algorithm for supervised learning of binary classifiers. This algorithm enables neurons to learn and process elements in the training set one at a time.

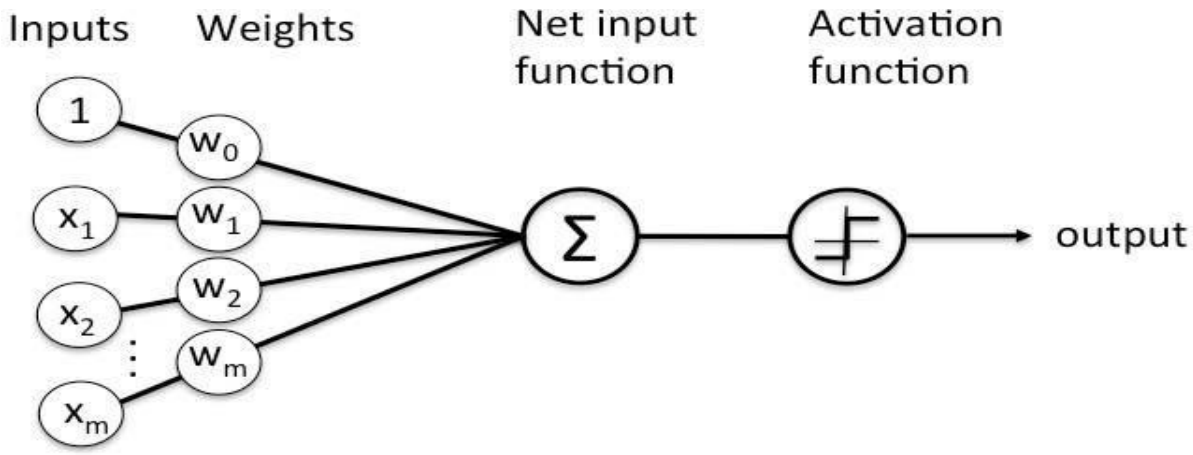


Figure 1 Single Perceptron Illustration

2.1 Types of Perceptrons

Single layer Perceptrons can learn only linearly separable patterns.

Multilayer Perceptrons or feedforward neural networks with two or more layers have the greater processing power.

2.2 Perceptron Function

Perceptron is a function that maps its input "x," which is multiplied with the learned weight coefficient; an output value "f(x)"is generated.

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

In the equation given above:

"w" = vector of real-valued weights

"b" = bias (an element that adjusts the boundary away from origin without any dependence on the input value)

"x" = vector of input x values

$$\sum_{i=1}^{m} w_i x_i$$

"m" = number of inputs to the Perceptron

The output can be represented as "1" or "0." It can also be represented as "1" or "-1" depending on which activation function is used.

2.3 Inputs of a Perceptron

A Perceptron accepts inputs, moderates them with certain weight values, then applies the transformation function to output the final result. The image below shows a Perceptron with a Boolean output.

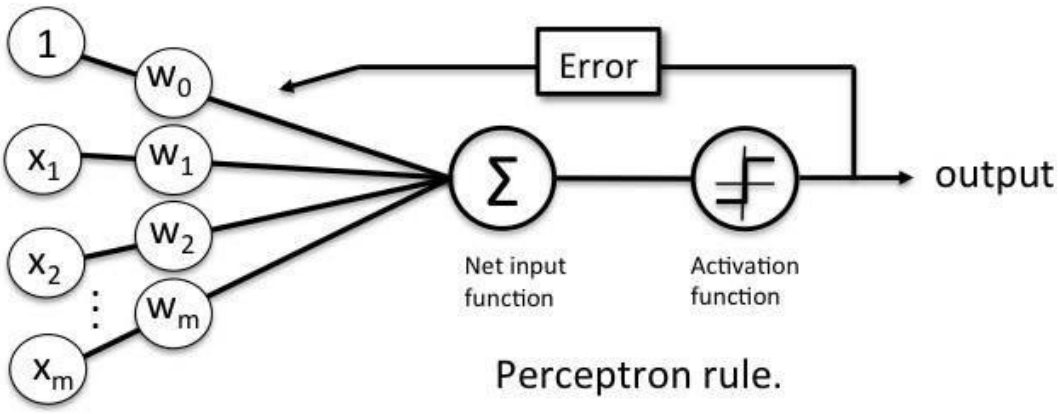


Figure 2 Inputs in a Perceptron Learning Process

A Boolean output is based on inputs such as salaried, married, age,

past credit profile, etc. It has only two values: Yes and No or True and False. The summation function "∑" multiplies all inputs of "x" by weights "w" and then adds them up as follows:

$$w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n$$

2.4 Activation Functions of Perceptron

The activation function applies a step rule (convert the numerical output into +1 or -1) to check if the output of the weighting function is greater than zero or not.

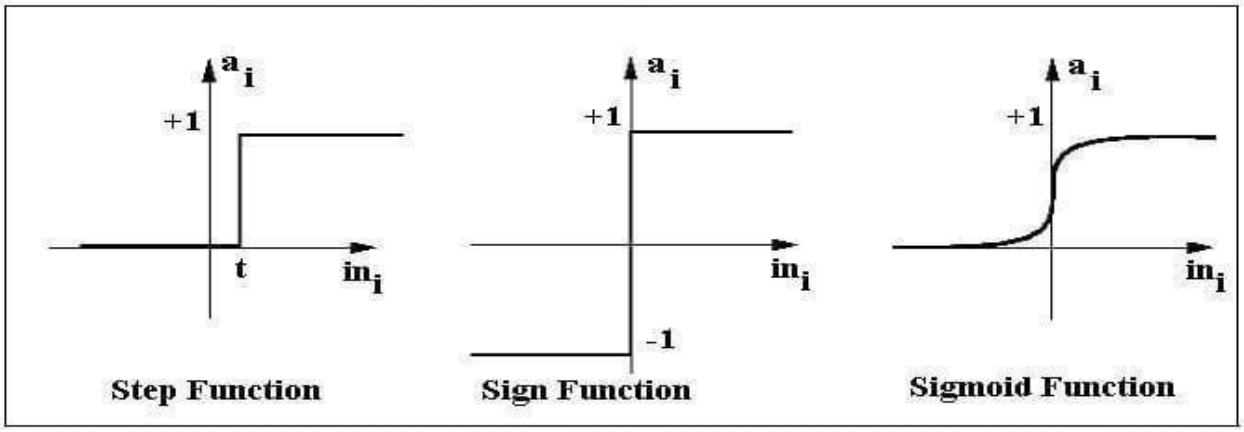


Figure 3 Various Activation Functions

For example:

If ∑ wixi> 0 => then final output "o" = 1 (issue bank loan)
Else, final output "o" = -1 (deny bank loan)

Step function gets triggered above a certain value of the neuron output; else it outputs zero.

Sign Function outputs +1 or -1 depending on whether neuron output is greater than zero or not.

Sigmoid is the S-curve and outputs a value between 0 and 1.

2.5 Perceptron at a glance

- Weights are multiplied with the input features and decision is made if the neuron is fired or not.
- Activation function applies a step rule to check if the output of the weighting function is greater than zero.
- Linear decision boundary is drawn enabling the distinction between the two linearly separable classes +1 and -1.

- If the sum of the input signals exceeds a certain threshold, it outputs a signal; otherwise, there is no output.
- Types of activation functions include the sign, step, and sigmoid functions.

For more information about perceptrons, check out the link given below:

https://becominghuman.ai/lets-build-a-perceptron-in-python-eff3462eb22

3 Perceptron Training Algorithms

In this course, we have covered two training algorithms namely, perceptron learning rule and gradient descent. These algorithms would automatically learn the optimal weight coefficients. The input features are then multiplied with these weights to determine if a neuron fires or not.

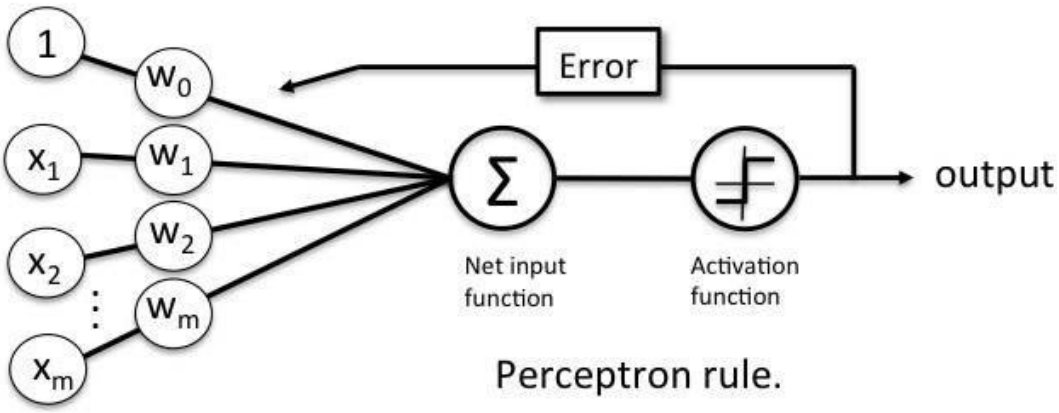


Figure 4 Learning Process

The perceptron receives multiple input signals and if the sum of the input signals exceeds a certain threshold, it either outputs a signal or does not return an output.

In this lab session, we only focus on the perceptron learning rule. This rule requires the data to be linearly separable. The weight update rule is given by:

$$\mathbf{newW} = \mathbf{oldW} + (y - \hat{y}).\ \mathbf{X}$$

The variable written in bold represents vectors. X is the input sample (vector) for which a mismatch is found.

Note: you will apply this rule to every misclassified sample until all samples are classified correctly.

4 Logic Gates

Logic gates are the building blocks of a digital system, especially neural network. In short, they are the electronic circuits that help in addition, choice, negation, and combination to form

complex circuits.
Using the logic gates, Neural Networks can learn on their own without you having to manually code the logic. Most logic gates have two inputs and one output.
Each terminal has one of the two binary conditions, low (0) or high (1), represented by different voltage levels. The logic state of a terminal changes based on how the circuit processes data.
Based on this logic, logic gates can be categorized into seven types:

- AND
- NAND
- OR
- NOR
- NOT
- XOR

4.1 Implementation of Logic Gates with Perceptron

The logic gates that can be implemented with Perceptron are discussed below.

AND

If the two inputs are TRUE (+1), the output of Perceptron is positive, which amounts to TRUE. This is the desired behavior of an AND gate.

```
x1= 1 (TRUE), x2= 1 (TRUE)
w0 = -.8, w1 = 0.5, w2 = 0.5
=> o(x1, x2) => -.8 + 0.5*1 + 0.5*1 = 0.2 > 0
```

OR

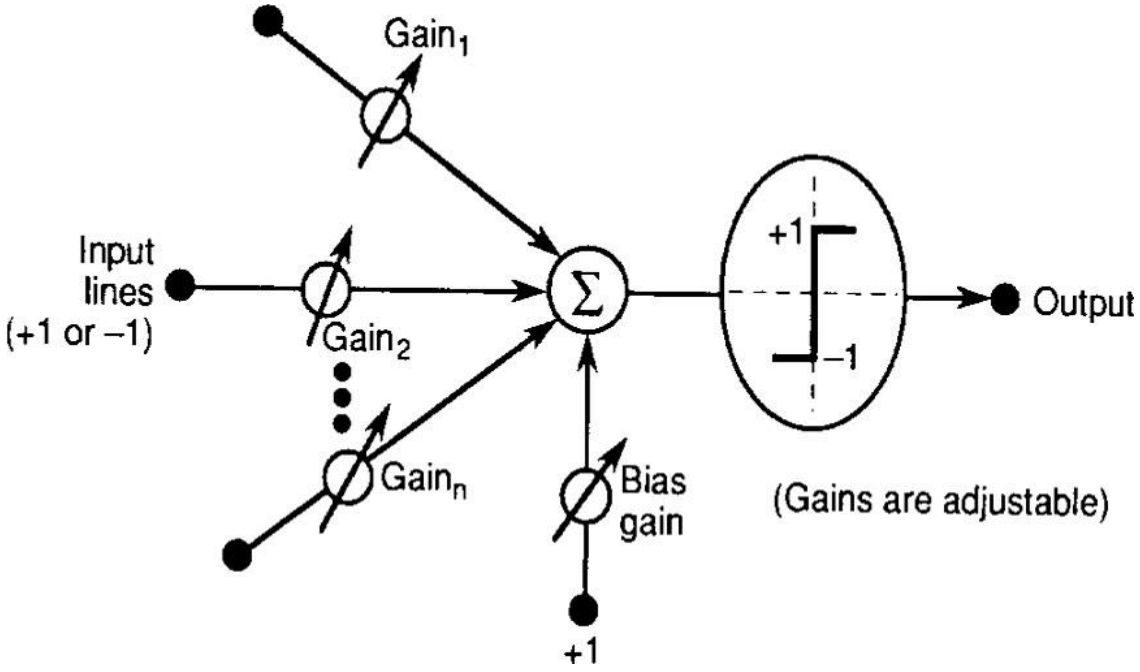
If either of the two inputs are TRUE (+1), the output of Perceptron is positive, which amounts to TRUE.
This is the desired behavior of an OR gate.

```
x1 = 1 (TRUE), x2 = 0 (FALSE)
w0 = -.3, w1 = 0.5, w2 = 0.5
=> o(x1, x2) => -.3 + 0.5*1 + 0.5*0 = 0.2 > 0
```

The practical implementation of perceptron algorithm on a dataset can be found on the link given below:
https://machinelearningmastery.com/implement-perceptron-algorithm-scratch-python/

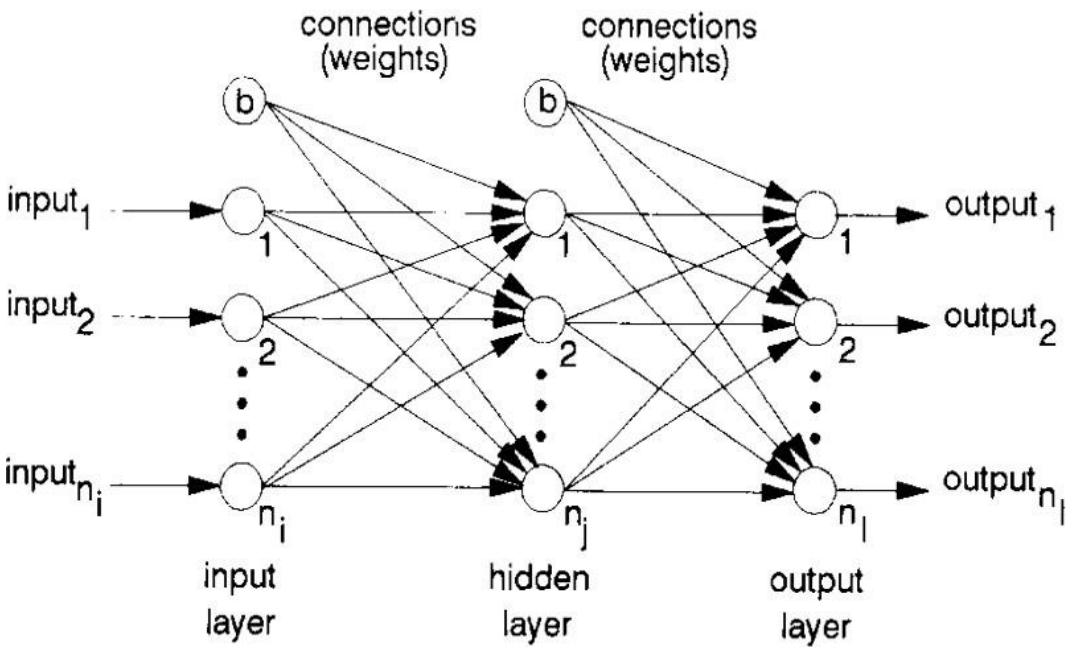
5 Artificial Neural Network (ANN)

An Artificial neural network is a computational network, based on biological neural networks that construct the structure of the human brain. Similar to how a human brain has neurons interconnected to each other, artificial neural networks also have neurons that are linked to each other in various layers of the networks. A typical single neuron (also known as perceptron) is represented below.



ANNs are designed to spot patterns in the data. This makes ANNs an optimal solution for classifying (sorting data into predetermined categories), clustering (finding like characteristics among data and pulling that data together into categories) and making predictions from data (such as helping determine infection rates for COVID, the next catastrophic weather event or box-office smash).

ANNs have an input layer and output layer. Between these two layers there are other hidden layers that perform the mathematical computations that help determine the decision or action the machine should take. Ultimately, these hidden layers are in place to transform the input data into something the output unit can use. A typical two layer NN is represented in the image below.



5.1 Back Propagation Algorithm

It is the training or learning algorithm. It learns by example. If you submit to the algorithm the example of what you want the network to do, it changes the network's weights so that it can produce desired output for a particular input on finishing the training.

The backprop algorithm cycles through two distinct passes, a forward pass followed by a backward pass through the layers of the network. The algorithm alternates between these passes several times as it scans the training data. Typically, the training data has to be scanned several times before the networks "learns" to make good classifications.

Forward Pass: Computation of outputs of all the neurons in the network. The input is fed to the input layer, the neurons perform a linear transformation on this input using the weights and biases. Post that, an activation function is applied on this linear transformation to add non-linearity in the model in order to learn complex pattern from the data. Finally, the output from the activation function moves to the next hidden layer and the same process is repeated. This forward movement of information is known as the forward pass or propagation.

$$Y = \text{Activation}(\Sigma(weight * input) + bias)$$

Backward pass: Propagation of error and adjustment of weights

The task is to make the output to the neural network as close to the actual (desired) output. Each of these neurons is contributing some error to the final output. How do you reduce the error? We try to minimize the value/weight of neurons that are contributing more to the error and this happens while traveling back to the neurons of the neural network and finding where the error lies. In order to minimize the error, the neural networks use a common algorithm known as "Gradient Descent", which helps to optimize the task quickly and efficiently.

The one round of forwarding and backpropagation iteration is known as one training iteration aka "**Epoch**". Most applications of feedforward networks and backprop require several epochs before errors are reasonably small.

5.2 Activation Functions

Activation function is one of the building blocks of Neural Network. Popular types of activation functions are:

- Binary Step Function
- Linear Function
- Sigmoid
- Tanh
- Rectified Linear Unit (ReLU)
- Leaky ReLU
- Parameterised ReLU
- ELU
- Softmax

Refer to the following links to understand activation functions in details:

- https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/
- https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0
- https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

Refer to the lecture for further details regarding ANNs.

6 Image Classification Using ANNs with Scikit-learn

Note:
Find the complete scikit-learn user guide here: https://scikit-learn.org/stable/user_guide.html
Find the Multilayer Perceptron scikit-learn here: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#
Refer to the Colab Notebook uploaded along this lab manual on Classroom.

Image classification is an amazing application of artificial intelligence. We can train a powerful algorithm to model a large image dataset. This model can then be used to classify a similar but unknown set of images.
We will build an Image classifier for the Fashion-MNIST Dataset. The Fashion-MNIST dataset is a collection of Zalando's article images. It contains 60,000 images for the training set and 10,000 images for the test set data (*we will discuss the test and training datasets along with the validation dataset later*). These are 28 x 28 grayscale images and belong to the labels of 10 different classes.

6.1 Labels

Each training and test example is assigned to one of the following labels:

Label	Description
0	T-shirt/top

1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

6.2 Classification Evaluation Metric

Learn more about Confusion matrix, Precision, Recall and Accuracy here:
https://medium.com/@kennymiyasato/classification-report-precision-recall-f1-score-accuracy-16a245a437a5

7 Exercise 1: Implement OR and AND gates using perceptron learning scheme. [10 Marks]

Perceptrons can be modelled for designing logic gates-based implementation for toys. You need to write a code to handle OR & AND gate logic using perceptron learning strategy. Refer to the contents of section 5 to help solve the task.

Exercise 2: Improve the accuracy of the Fashion-MNIST Neural Network model given in the Colab Notebook. [10 Marks]

You can apply following techiques to improve the accuracy:

1. Change No of hidden Layers
2. Increase no of Iteration
3. Change the learning rate

8 Submission Guidelines

Always read the submission instructions carefully.

- Rename your notebook to your roll number_labno_exercise1 and roll number_labno_exercise2 download the notebook as **.ipynb** extension.
- To download the required file, go to **File->Download .ipynb**
- Only submit the **.ipynb** file. DO NOT **zip** or **rar** your submission file
- Submit this file on Google Classroom under the relevant assignment

Data Minning Lab