

**Vidyavardhini’s**

**College of Engineering & Technology**

Vasai Road (W)

**Department of**

**Artificial Intelligence and Data Science**

**Laboratory Manual**

|  |  |  |  |
| --- | --- | --- | --- |
| Semester | VI | Class | T.E |
| Course Code | CSL604 | | |
| Course Name | Machine Learning Lab | | |

****

**Vidyavardhini’s College of Engineering & Technology**

**Vision**

To be a premier institution of technical education; always aiming at becoming a valuable resource for industry and society.

**Mission**

* To provide a technologically inspiring environment for learning.
* To promote creativity, innovation and professional activities.
* To inculcate ethical and moral values.
* To cater personal, professional and societal needs through quality education.

**Program Outcomes (POs):**

Engineering Graduates will be able to:

* **PO1. Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
* **PO2. Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
* **PO3. Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
* **PO4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
* **PO5. Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
* **PO6. The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
* **PO7. Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
* **PO8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
* **PO9. Individual and teamwork:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
* **PO10. Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
* **PO11. Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
* **PO12. Life-long learning:** Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

**Course Objectives**

|  |  |
| --- | --- |
| 1 | To introduce the basic libraries, tools using platform suitable to machine learning such as COLAB. |
| 2 | To implement various regression technique as a Supervised Learning. |
| 3 | To implement Neural Network based models. |
| 4 | To implement Clustering techniques. |

**Course Outcomes**

|  |  |  |  |
| --- | --- | --- | --- |
| **CO** | **At the end of course students will be able to:** | **Action verbs** | **Bloom’s Level** |
| CSL604.1 | Identify different libraries used for Data processing like Numpy , Pandas and Matplolib. | Identify | Understand (level 2) |
| CSL604.2 | Apply System of Linear equations, Length of vector and the concept of Symmetric Positive Definite Matrices on the given data to understand mathematical modeling of Machine Learning Models. | Apply | Apply (level 3) |
| CSL604.3 | Apply the different methods of Linear model for Regression and Classification. | Apply | Apply (level 3) |
| CSL604.4 | Apply Hebbian Learning Rule and Expectation-Maximization algorithm for clustering. | Apply | Apply (level 3) |
| CSL604.5 | Apply concept of Neural Network to design simple network and understand Perceptron Learning Rule | Apply | Apply (level 3) |
| CSL604.6 | Use Dimensionality Reduction techniques for dealing with data with large  number of attributes | Apply | Apply (level 3) |

**Mapping of Experiments with Course Outcomes**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **List of Experiments** | **Course Outcomes** | | | | | |
| **CSL604.1** | **CSL604.2** | **CSL604.3** | **CSL604.4** | **CSL604.5** | **CSL604.6** |
| Introduction to platforms such as Anaconda, COLAB. | 3 | - | - | - | - | - |
| Implementation of Linear Regression | - | 3 | - | - | - | - |
| Implementation of Logistic Regression | - | 3 | - | - | - | - |
| Implementation of Support Vector Machine | - | - | 3 | - | - | - |
| Implementation of Hebbian Learning Algorithms. | - | - | 3 | - | - | - |
| Implementation of McCulloch Pitts Model | - | - | - | 3 | - | - |
| Implementation of Single Layer Perceptron Learning algorithm | - | - | - | 3 | - | - |
| Implementation of Error Backpropagation Perceptron Training Algorithm | - | - | - | - | 3 | - |
| Implementation of Principal Component Analysis | - | - | - | - | 3 | - |
| Applications of above algorithms as a case study (E.g. Hand Writing Recognition using MNIST data set, classification using IRIS data set, etc) |  |  |  |  |  | 3 |

**INDEX**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sr. No.** | **Name of Experiment** | **D.O.P.** | **D.O.C.** | **Page No.** | **Remark** |
| 1 | Introduction to platforms such as Anaconda, COLAB. |  |  |  |  |
| 2 | Implementation of Linear Regression |  |  |  |  |
| 3 | Implementation of Logistic Regression |  |  |  |  |
| 4 | Implementation of Support Vector Machine |  |  |  |  |
| 5 | Implementation of Hebbian Learning Algorithms. |  |  |  |  |
| 6 | Implementation of McCulloch Pitts Model |  |  |  |  |
| 7 | Implementation of Single Layer Perceptron Learning algorithm |  |  |  |  |
| 8 | Implementation of Error Backpropagation Perceptron Training Algorithm |  |  |  |  |
| 9 | Implementation of Principal Component Analysis |  |  |  |  |
| 10 | Applications of above algorithms as a case study (E.g. Hand Writing Recognition using MNIST data set, classification using IRIS data set, etc) |  |  |  |  |

D.O.P: Date of performance

D.O.C : Date of correction

|  |
| --- |
| Experiment No. 1 |
| Introduction to platforms such as Anaconda, COLAB. |
| Date of Performance: |
| Date of Submission: |
| Marks: |
| Sign: |

**Aim:** Introduction to platforms such as Anaconda, COLAB.

**Objective:** Ability to understand the different platforms and their function used for data preprocessing and model development..

**Theory:**

**Anaconda**

Anaconda is a popular platform that simplifies the process of installing, managing, and deploying data science tools and libraries. It is widely used for scientific computing, machine learning, and data analysis.

**Key Features of Anaconda:**

**Package Management:** Comes with conda, a package manager that helps install and manage Python packages and their dependencies.

**Pre-installed Libraries:** Includes popular data science libraries like NumPy, pandas, matplotlib, scikit-learn, and TensorFlow.

**Jupyter Notebooks:** Supports running Jupyter Notebook, an interactive environment for writing and running code, especially useful for data analysis and visualization.

**Environment Management:** Allows you to create isolated environments for different projects, ensuring there are no conflicts between dependencies.

**Pros:**

User-friendly and beginner-friendly.

Works offline after installation.

Robust environment management.

**Cons:**

The download and installation size is large.

May require manual updates for packages.

**Google Colab**

Google Colab (Collaboratory) is a free, cloud-based platform provided by Google that allows you to write and execute Python code in a Jupyter Notebook-like environment. It is especially useful for machine learning and data science tasks.

**Key Features of Google Colab:**

**Cloud-Based:** No installation required; everything runs in the browser.

GPU and TPU Support: Provides free access to GPUs and TPUs for accelerated computations.

**Seamless Integration:** Easily integrates with Google Drive for saving and accessing notebooks.

**Collaboration:** Multiple users can work on the same notebook in real time.

**Pros:**

Free access to powerful computing resources (GPUs/TPUs).

Easy to share and collaborate.

Pre-installed popular libraries like TensorFlow, Keras, and pandas.

**Cons:**

Requires an internet connection.

Limited runtime (usually disconnects after a few hours of inactivity).



**Implementation:**

**Conclusion:**

|  |
| --- |
| Experiment No. 2 |
| Implementation of Linear Regression Algorithm |
| Date of Performance: |
| Date of Submission: |
| Marks: |
| Sign: |

**Aim:** Implementation of Linear Regression Algorithm.

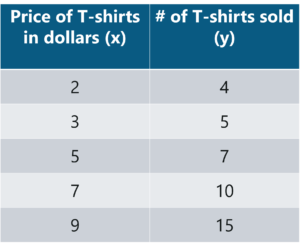
**Objective:** To implement Linear Regression in order to build a model that studies the relationship between an independent and dependent variable. The model will be evaluated by using least square regression method where RMSE and R-squared will be the model evaluation parameters..

**Theory:**

The least-squares method is a crucial statistical method that is practiced to find a regression line or a best-fit line for the given pattern. This method is described by an equation with specific parameters. The method of least squares is generously used in evaluation and regression. In regression analysis, this method is said to be a standard approach for the approximation of sets of equations having more equations than the number of unknowns. The method of least squares actually defines the solution for the minimization of the sum of squares of deviations or the errors in the result of each equation. Find the formula for sum of squares of errors, which help to find the variation in observed data. The least-squares method is often applied in data fitting.

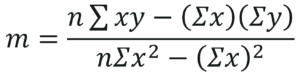
## Least Squares Regression Example

Tom who is the owner of a retail shop, found the price of different T-shirts vs the number of T-shirts sold at his shop over a period of one week.



Let us use the concept of least squares regression to find the line of best fit for the above data.

**Step 1:** Calculate the slope ‘m’ by using the following formula:



After you substitute the respective values, m = 1.518 approximately.

**Step 2:** Compute the y-intercept value

Y-Intercept formula - Least Squares Regression Method - Edureka

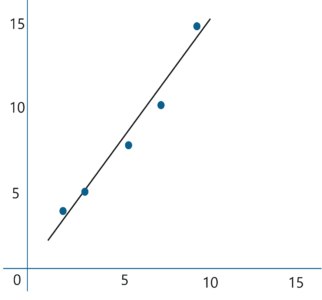
After you substitute the respective values, c = 0.305 approximately.

**Step 3:** Substitute the values in the final equation

Regression line formula - Least Squares Regression Method - Edureka



Let’s construct a graph that represents the**y=mx + c** line of best fit:



Now Tom can use the above equation to estimate how many T-shirts of price $8 can he sell at the retail shop.

*y = 1.518 x 8 + 0.305 = 12.45* T-shirts

This comes down to 13 T-shirts!

**Dataset:**

The data set contains the following variables:

* **Gender:** Male or female represented as binary variables
* **Age:** Age of an individual
* **Head size in cm^3:** An individuals head size in cm^3
* **Brain weight in grams:** The weight of an individual’s brain measured in grams

These variables need to be analyzed in order to build a model that studies the relationship between the head size and brain weight of an individual.

**Step 1: Import the required libraries**

**Step 2: Import the data set**

**Step 3: Assigning ‘X’ as independent variable and ‘Y’ as dependent variable**

**Step 4: Calculate the values of the slope and y-intercept**

**Step 5: Plotting the line of best fit**

**Step 6: Model Evaluation**

**Implementation:**

**Code:**

import numpy as np

import matplotlib.pyplot as plt

# Generate sample data (for example, let's say we're predicting y from x)

np.random.seed(0)  # For reproducibility

X = np.random.rand(100, 1) \* 10  # 100 random data points for X

y = 2 \* X + 1 + np.random.randn(100, 1) \* 2  # Linear relation with noise

# Add a column of ones to X to account for the intercept term

X\_b = np.c\_[np.ones((X.shape[0], 1)), X]  # Adding a bias column (X0 = 1)

# Calculate the optimal parameters (theta) using the Normal Equation

theta = np.linalg.inv(X\_b.T.dot(X\_b)).dot(X\_b.T).dot(y)

# Extract the intercept and slope from theta

intercept, slope = theta[0], theta[1]

# Print out the coefficients

print(f"Intercept: {intercept[0]}")

print(f"Slope: {slope[0]}")

# Plotting the data points

plt.scatter(X, y, color='blue', label='Data points')

# Plotting the regression line

plt.plot(X, X\_b.dot(theta), color='red', label=f'Linear regression line: y = {slope[0]:.2f}x + {intercept[0]:.2f}')

plt.xlabel("X")

plt.ylabel("y")

plt.legend()

# Save the plot as an image

plt.savefig("linear\_regression\_plot.png")

# Show the plot

plt.show()

**A graph with blue dots and red line

Description automatically generated**

**Conclusion:**

Comment on the Least Square Method used for regression.

The Least Squares Method is a foundational and widely used technique for regression analysis, offering simplicity, interpretability, and optimal performance under linearity and normality assumptions. However, it is sensitive to outliers, struggles with multicollinearity, and relies on strict assumptions like homoscedasticity. Despite its limitations, it remains a powerful tool for linear modeling, with applications in diverse fields. Extensions like Ridge Regression, LASSO, and robust regression methods address some of its weaknesses, making it adaptable to more complex scenarios. Overall, it is a cornerstone of statistical modeling but should be applied with caution and awareness of its assumptions.

|  |
| --- |
| Experiment No. 3 |
| Implementation of Logistic Regression Algorithm |
| Date of Performance: |
| Date of Submission: |
| Marks: |
| Sign: |

**Aim:** Implementation of Logistic Regression Algorithm.

**Objective:** Able to perform various feature engineering tasks, apply logistic regression on the given dataset and maximize the accuracy.

**Theory:**

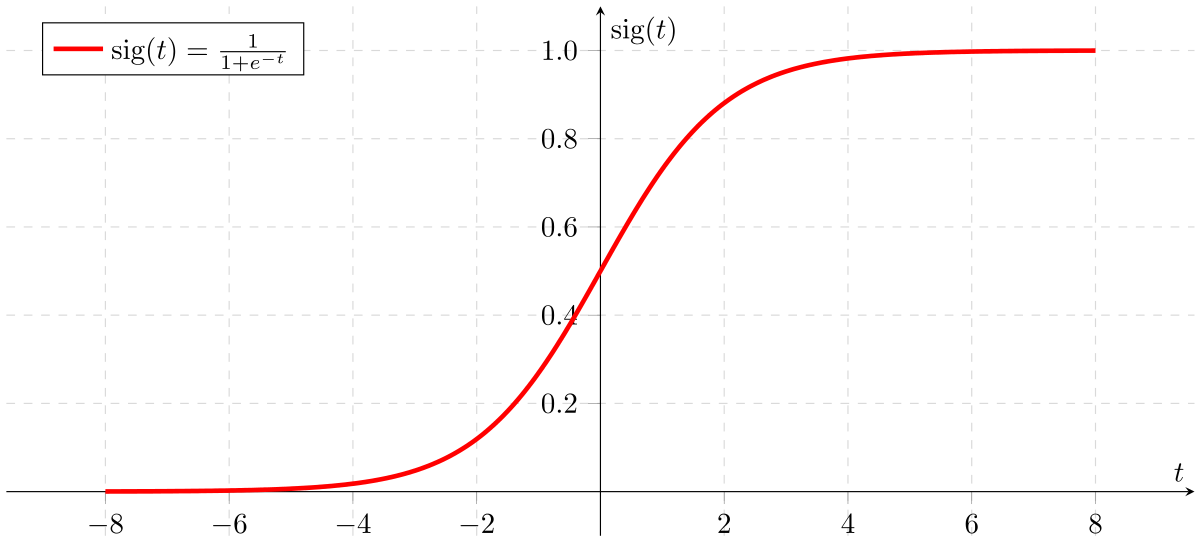
Logistic Regression was used in the biological sciences in early twentieth century. It was then used in many social science applications. Logistic Regression is used when the dependent variable(target) is categorical and is binary in nature. In order to perform binary classification, the logistic regression techniques makes use of Sigmoid function.

For example,

To predict whether an email is spam (1) or (0)

Whether the tumor is malignant (1) or not (0)

Consider a scenario where we need to classify whether an email is spam or not. If we use linear regression for this problem, there is a need for setting up a threshold based on which classification can be done. Say if the actual class is malignant, predicted continuous value 0.4 and the threshold value is 0.5, the data point will be classified as not malignant which can lead to serious consequence in real time.



From this example, it can be inferred that linear regression is not suitable for classification problem. Linear regression is unbounded, and this brings logistic regression into picture. Their value strictly ranges from 0 to 1.

**Implementation:**

**Code:**

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import load\_breast\_cancer

from sklearn.model\_selection import train\_test\_split

class LogisticRegression:

    def \_\_init\_\_(self):

        self.params = None

    def fit(self, X, y):

        bias = np.ones(len(X))

        X\_bias = np.c\_[bias, X]

        inner\_part = np.transpose(X\_bias) @ X\_bias

        inverse\_part = np.linalg.inv(inner\_part)

        outer\_part = inverse\_part @ np.transpose(X\_bias)

        self.params = outer\_part @ y

        return self.params

    def predict(self, X):

        bias\_testing = np.ones(len(X))

        X\_test = np.c\_[bias\_testing, X]

        z = X\_test @ self.params

        sigmoid = 1 / (1 + np.exp(-z))

        y\_hat = (sigmoid >= 0.5).astype(int)

        return sigmoid, y\_hat

# Load dataset

dataset = load\_breast\_cancer()

X = dataset.data

y = dataset.target

# Split dataset

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.1, random\_state=42)

# Train model

model = LogisticRegression()

model.fit(X\_train, y\_train)

# Predict

sigmoid\_vals, y\_pred = model.predict(X\_test)

# Plot sigmoid values

plt.figure(figsize=(10, 5))

plt.scatter(range(len(sigmoid\_vals)), sigmoid\_vals, c=y\_test, cmap='coolwarm', label="Sigmoid Probabilities")

plt.axhline(y=0.5, color='black', linestyle='--', label="Decision Boundary")

plt.xlabel("Test Samples")

plt.ylabel("Sigmoid Probability")

plt.title("Sigmoid Probabilities of Predictions")

plt.legend()

plt.show()

# Plot actual vs predicted

plt.figure(figsize=(10, 5))

plt.scatter(range(len(y\_test)), y\_test, color='blue', label="Actual Labels", alpha=0.6)

plt.scatter(range(len(y\_pred)), y\_pred, color='red', marker='x', label="Predicted Labels", alpha=0.6)

plt.xlabel("Test Samples")

plt.ylabel("Class (0 or 1)")

plt.title("Actual vs Predicted Labels")

plt.legend()

plt.show()

**Output:**

A diagram of a test

Description automatically generated with medium confidence

A graph of a comparison of labels

Description automatically generated with medium confidence

**Conclusion:**

Comment on the accuracy obtained.

* The accuracy score indicates how well the model is classifying the test samples.
* If accuracy is above 85-90%, the model performs well for this dataset.
* If accuracy is below 70%, it suggests that the model might need improvements like feature scaling, regularization, or a more robust optimization algorithm.
* The model in its current state does not use gradient descent, which is typically required for logistic regression in real-world scenarios.

|  |
| --- |
| Experiment No. 4 |
| Implementation of Support Vector Machine Algorithm |
| Date of Performance: |
| Date of Submission: |
| Marks: |
| Sign: |

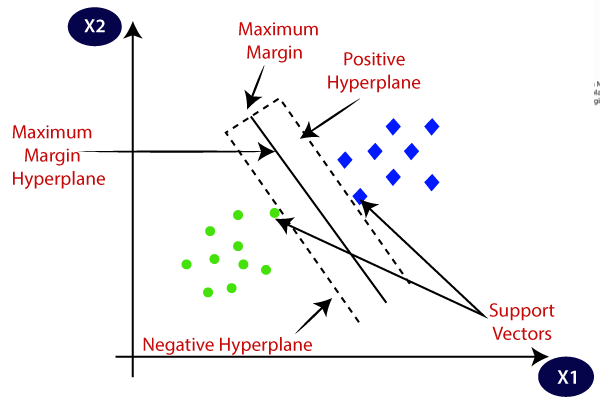
**Aim:** Implementation of Support Vector Machine Algorithm.

**Objective:** Ablility to perform various feature engineering tasks, apply Support Vector Machine and create the Confusion Matrix.

**Theory:**

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



Hyperplane and Support Vectors in the SVM algorithm:

**Hyperplane:** There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane.

We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

**Support Vectors:**

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

**Implementation of Support Vector Machine**

1. Data Pre-processing step
2. Fitting the SVM classifier to the training set
3. Predicting the test set result
4. Creating the confusion matrix
5. Visualizing the training set result
6. Visualizing the test set result

**Implementation:**

**Conclusion:**

|  |
| --- |
| Experiment No. 5 |
| Implementation of Hebbian Learning Algorithms. |
| Date of Performance: |
| Date of Submission: |
| Marks: |
| Sign: |

**Aim:** Implementation of Hebbian Learning Algorithms.

**Objective:** Able to interpret the implementation of Hebbian Learning Algorithms.

**Theory:**

Hebbian learning is a foundational concept in neuroscience and artificial neural networks. It describes how the strength of connections (weights) between neurons is adjusted based on their activity. This learning mechanism is often summarized as:

"Neurons that fire together, wire together."

Historical Context

The Hebbian learning rule was first proposed by Donald Hebb in his 1949 book "The Organization of Behavior". It was an attempt to explain how neurons adapt during the learning process and how memory and associations are formed in the brain.

**Core Concepts of Hebbian Learning**

**Synaptic Plasticity:**

The strength of the synaptic connection between two neurons is modified based on their simultaneous activation.

If both the pre-synaptic and post-synaptic neurons are active simultaneously, the connection strengthens.

**Locality:**

The weight adjustment depends only on the states of the connected neurons, without external supervision or global feedback.

**Unsupervised Learning:**

Hebbian learning does not require labeled data or error correction. Instead, it adjusts weights based solely on the correlation of neuronal activity.

**Variants of Hebbian Learning**

**Standard Hebbian Learning:**

The weight change is proportional to the product of the activations of the pre- and post-synaptic neurons.

**Oja’s Rule (Stabilized Hebbian Learning):**

To prevent unbounded weight growth, Oja introduced a normalization term.

**Covariance Rule:**

Instead of directly using activations, this rule subtracts the mean activation

**Hebbian Anti-Hebbian Learning:**

Positive correlations increase weights (Hebbian rule), while negative correlations decrease weights (anti-Hebbian rule).

**Implementation:**

**Conclusion:**

|  |
| --- |
| Experiment No. 6 |
| Implementation of McCulloch Pitts Model |
| Date of Performance: |
| Date of Submission: |
| Marks: |
| Sign: |

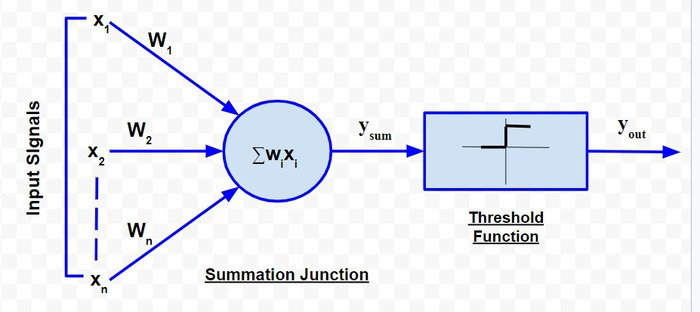
**Aim:** To implement McCulloch Pitts Model for ANN.

**Objective:** Able to design a neural network and use activation function as learning rule defined by McCulloch Piits Model.

**Theory:**

**McCulloch-Pitts Model of Neuron**

The McCulloch-Pitts neural model, which was the earliest ANN model, has only two types of inputs — **Excitatory and Inhibitory.** The excitatory inputs have weights of positive magnitude and the inhibitory weights have weights of negative magnitude. The inputs of the McCulloch-Pitts neuron could be either 0 or 1. It has a threshold function as an activation function. So, the output signal *yout* is 1 if the input *ysum* is greater than or equal to a given threshold value, else 0. The diagrammatic representation of the model is as follows:



Simple McCulloch-Pitts neurons can be used to design logical operations. For that purpose, the connection weights need to be correctly decided along with the threshold function (rather than the threshold value of the activation function).

**Example:**

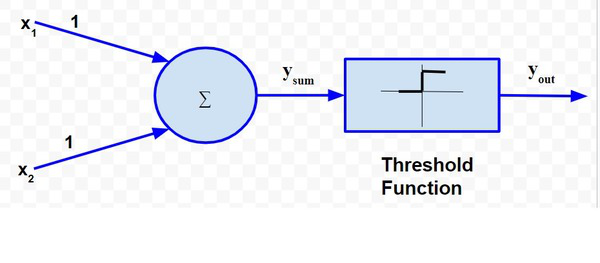
John carries an umbrella if it is sunny or if it is raining. There are four given situations. I need to decide when John will carry the umbrella. The situations are as follows:

* First scenario: It is not raining, nor it is sunny
* Second scenario: It is not raining, but it is sunny
* Third scenario: It is raining, and it is not sunny
* Fourth scenario: It is raining as well as it is sunny

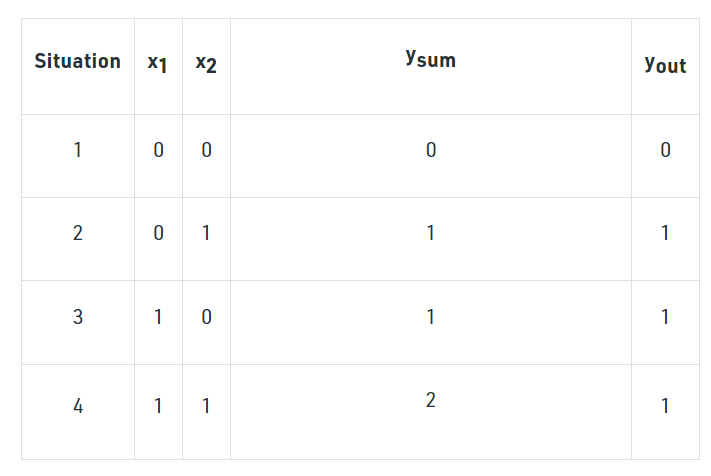
To analyse the situations using the McCulloch-Pitts neural model, I can consider the  input signals as follows:

* X1: Is it raining?
* X2 : Is it sunny?

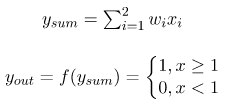
So, the value of both scenarios can be either 0 or 1. We can use the value of both weights X1 and X2as 1 and a threshold function as 1. So, the neural network model will look like:



**Truth Table for this case will be:**



So,



The truth table built with respect to the problem is depicted above. From the truth table, I can conclude that in the situations where the value of *yout* is 1, John needs to carry an umbrella. Hence, he will need to carry an umbrella in scenarios 2, 3 and 4.

**Implementation:**

**Conclusion:**

* 1. What is the use of McCulloch Pitts model?
  2. How it will be used to develop ANN?

|  |
| --- |
| Experiment No. 7 |
| Implementation of Single Layer Perceptron Learning Algorithm |
| Date of Performance: |
| Date of Submission: |
| Marks: |
| Sign: |

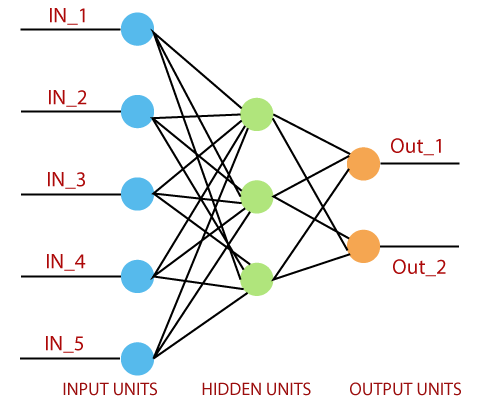
**Aim:** Implementation of Single Layer Perceptron Learning Algorithm

**Objective:** Able to implement and understand the aspects of Single Layer Perceptron Learning Algorithm.

**Theory:**

The perceptron is a single processing unit of any neural network. **Frank Rosenblatt** first proposed in **1958** is a simple neuron which is used to classify its input into one or two categories. Perceptron is a linear classifier, and is used in supervised learning. It helps to organize the given input data.

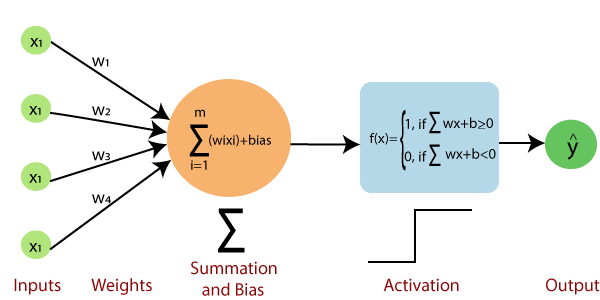
A perceptron is a neural network unit that does a precise computation to detect features in the input data. Perceptron is mainly used to classify the data into two parts. Therefore, it is also known as **Linear Binary Classifier**.



Perceptron uses the step function that returns +1 if the weighted sum of its input 0 and -1.

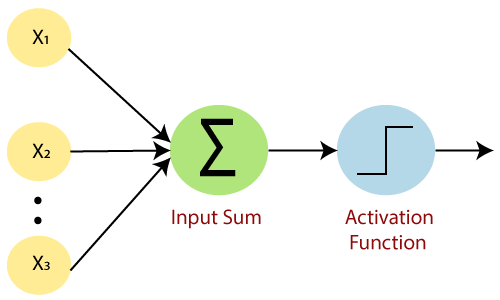
The activation function is used to map the input between the required value like (0, 1) or (-1, 1).

A regular neural network looks like this:

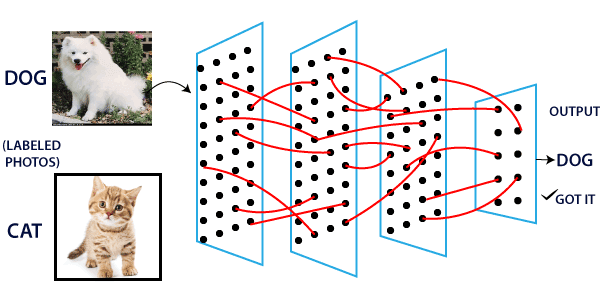


### The perceptron consists of 4 parts.

* **Input value or One input layer:** The input layer of the perceptron is made of artificial input neurons and takes the initial data into the system for further processing.
* **Weights and Bias:**  
  **Weight:** It represents the dimension or strength of the connection between units. If the weight to node 1 to node 2 has a higher quantity, then neuron 1 has a more considerable influence on the neuron.  
  **Bias:** It is the same as the intercept added in a linear equation. It is an additional parameter which task is to modify the output along with the weighted sum of the input to the other neuron.
* **Net sum:** It calculates the total sum.
* **Activation Function:** A neuron can be activated or not, is determined by an activation function. The activation function calculates a weighted sum and further adding bias with it to give the result.



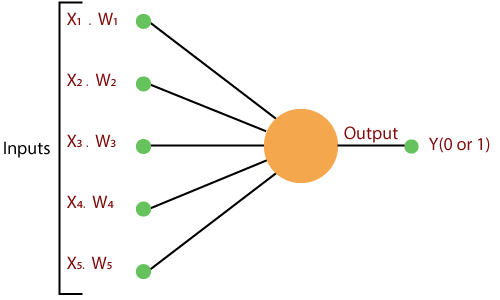
A standard neural network looks like the below diagram.



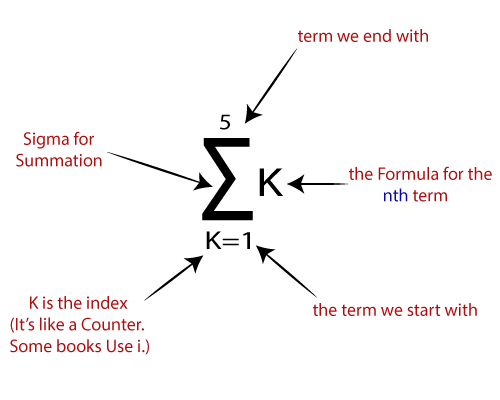
### How does it work?

The perceptron works on these simple steps which are given below:

**a.** In the first step, all the inputs x are multiplied with their weights **w**.



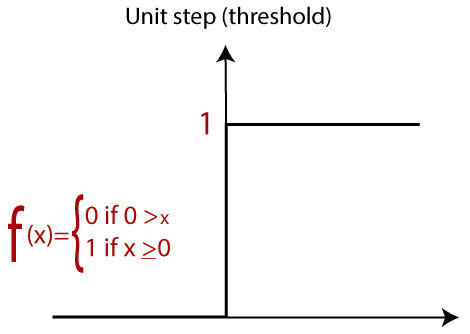
**b.** In this step, add all the increased values and call them the **Weighted sum**.



**c.** In our last step, apply the weighted sum to a correct **Activation Function**.

**For Example:**

A Unit Step Activation Function



**Implementation:**

**Conclusion**

|  |
| --- |
| Experiment No. 8 |
| Implementation of Error Backpropagation Perceptron Training Algorithm |
| Date of Performance: |
| Date of Submission: |
| Marks: |
| Sign: |

**Aim:** Implementation of Error Backpropagation Perceptron Training Algorithm

**Objective:** Able to design a neural network and use activation function as learning rule that converges using a backpropagation algorithm.

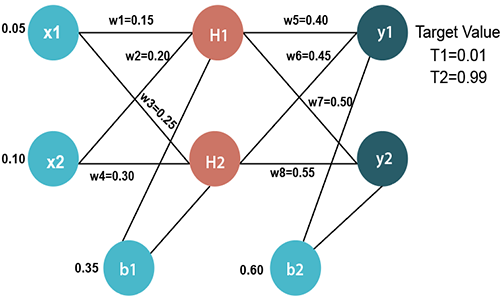
**Theory:**

**Backpropagation** is one of the important concepts of a neural network. Our task is to classify our data best. For this, we have to update the weights of parameter and bias, but how can we do that in a deep neural network? In the linear regression model, we use gradient descent to optimize the parameter. Similarly here we also use gradient descent algorithm using Backpropagation.

For a single training example, **Backpropagation** algorithm calculates the gradient of the **error function**. Backpropagation can be written as a function of the neural network. Backpropagation algorithms are a set of methods used to efficiently train artificial neural networks following a gradient descent approach which exploits the chain rule.

The main features of Backpropagation are the iterative, recursive and efficient method through which it calculates the updated weight to improve the network until it is not able to perform the task for which it is being trained. Derivatives of the activation function to be known at network design time is required to Backpropagation.

Now, how error function is used in Backpropagation and how Backpropagation works? Let start with an example and do it mathematically to understand how exactly updates the weight using Backpropagation.



### Input values

X1=0.05  
X2=0.10

### Initial weight

W1=0.15     w5=0.40  
W2=0.20     w6=0.45  
W3=0.25     w7=0.50  
W4=0.30     w8=0.55

### Bias Values

b1=0.35     b2=0.60

### Target Values

T1=0.01  
T2=0.99

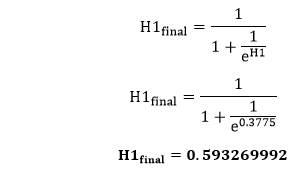
Now, we first calculate the values of H1 and H2 by a forward pass.

### Forward Pass

To find the value of H1 we first multiply the input value from the weights as

                              H1=x1×w1+x2×w2+b1  
                        H1=0.05×0.15+0.10×0.20+0.35  
                                    **H1=0.3775**

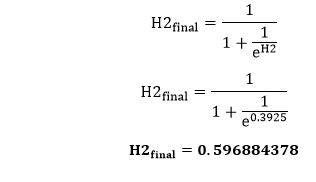
To calculate the final result of H1, we performed the sigmoid function as



We will calculate the value of H2 in the same way as H1

                              H2=x1×w3+x2×w4+b1  
                        H2=0.05×0.25+0.10×0.30+0.35  
                                    **H2=0.3925**

To calculate the final result of H1, we performed the sigmoid function as



Now, we calculate the values of y1 and y2 in the same way as we calculate the H1 and H2.

To find the value of y1, we first multiply the input value i.e., the outcome of H1 and H2 from the weights as

                              y1=H1×w5+H2×w6+b2  
                        y1=0.593269992×0.40+0.596884378×0.45+0.60  
                                    **y1=1.10590597**

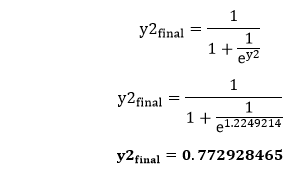
To calculate the final result of y1 we performed the sigmoid function as



We will calculate the value of y2 in the same way as y1

                              y2=H1×w7+H2×w8+b2  
                        y2=0.593269992×0.50+0.596884378×0.55+0.60  
                                    **y2=1.2249214**

To calculate the final result of H1, we performed the sigmoid function as

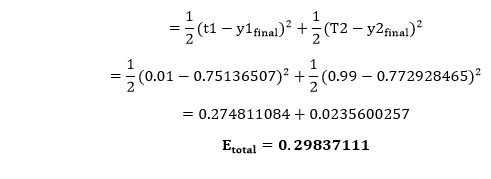


Our target values are 0.01 and 0.99. Our y1 and y2 value is not matched with our target values T1 and T2.

Now, we will find the **total error**, which is simply the difference between the outputs from the target outputs. The total error is calculated as

Backpropagation Process in Deep Neural Network

So, the total error is



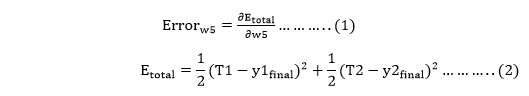
Now, we will backpropagate this error to update the weights using a backward pass.

### Backward pass at the output layer

To update the weight, we calculate the error correspond to each weight with the help of a total error. The error on weight w is calculated by differentiating total error with respect to w.

Backpropagation Process in Deep Neural Network

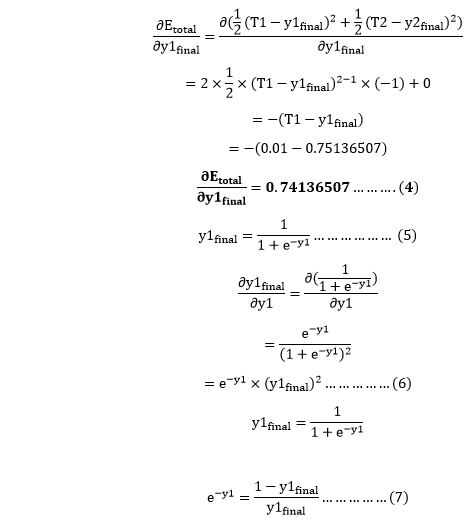
We perform backward process so first consider the last weight w5 as



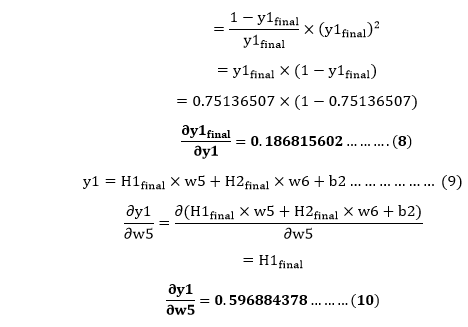
From equation two, it is clear that we cannot partially differentiate it with respect to w5 because there is no any w5. We split equation one into multiple terms so that we can easily differentiate it with respect to w5 as

Backpropagation Process in Deep Neural Network

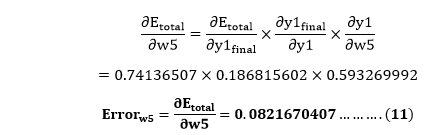
Now, we calculate each term one by one to differentiate Etotal with respect to w5 as



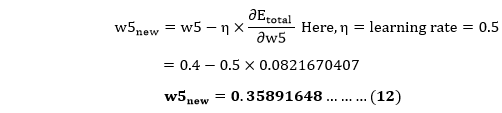
Putting the value of e-y in equation (5)



So, we put the values of Backpropagation Process in Deep Neural Network in equation no (3) to find the final result.



Now, we will calculate the updated weight w5new with the help of the following formula



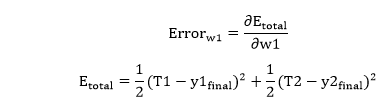
In the same way, we calculate w6new,w7new, and w8new and this will give us the following values

**w5new=0.35891648**  
                        **w6new=408666186**  
                        **w7new=0.511301270**  
                        **w8new=0.561370121**

### Backward pass at Hidden layer

Now, we will backpropagate to our hidden layer and update the weight w1, w2, w3, and w4 as we have done with w5, w6, w7, and w8 weights.

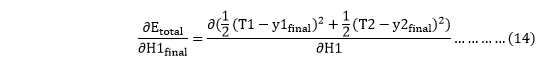
We will calculate the error at w1 as



From equation (2), it is clear that we cannot partially differentiate it with respect to w1 because there is no any w1. We split equation (1) into multiple terms so that we can easily differentiate it with respect to w1 as

Backpropagation Process in Deep Neural Network

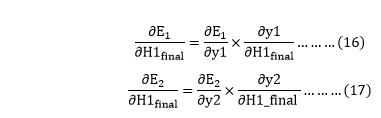
Now, we calculate each term one by one to differentiate Etotal with respect to w1 as



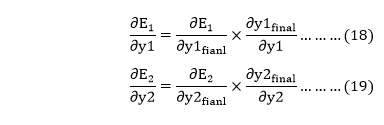
We again split this because there is no any H1final term in Etoatal as

Backpropagation Process in Deep Neural Network

Backpropagation Process in Deep Neural Network will again split because in E1 and E2 there is no H1 term. Splitting is done as

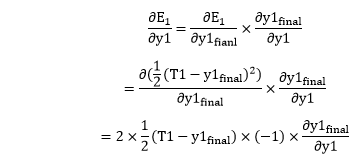


We again Split bothBackpropagation Process in Deep Neural Network because there is no any y1 and y2 term in E1 and E2. We split it as

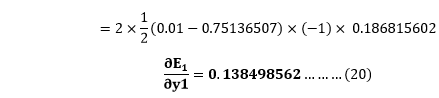


Now, we find the value of Backpropagation Process in Deep Neural Network by putting values in equation (18) and (19) as

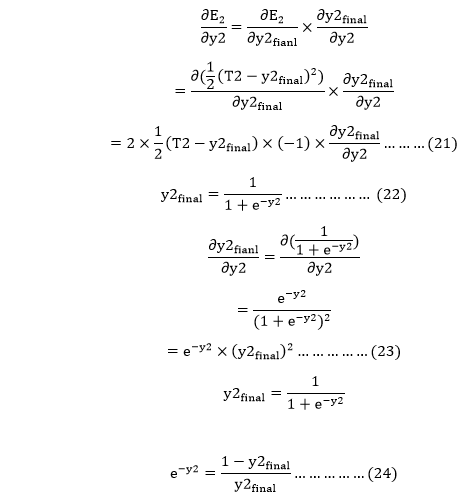
From equation (18)



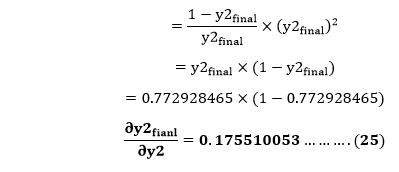
From equation (8)



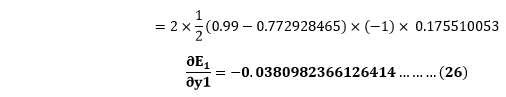
From equation (19)



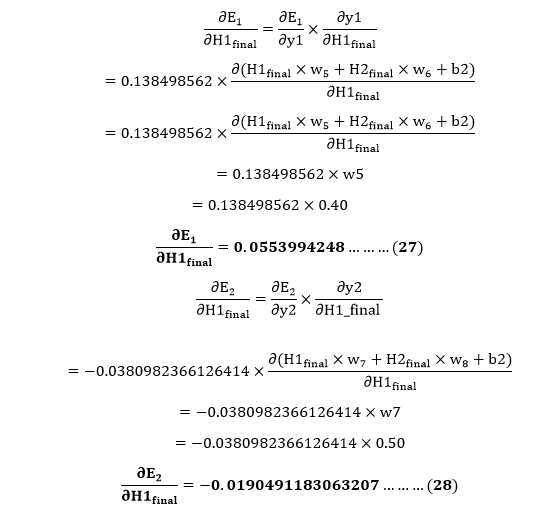
Putting the value of e-y2 in equation (23)



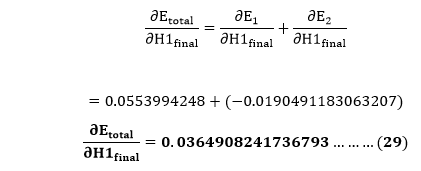
From equation (21)



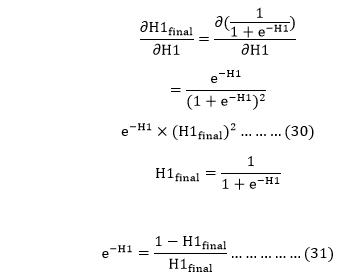
Now from equation (16) and (17)



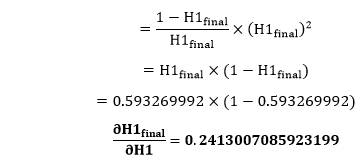
Put the value of Backpropagation Process in Deep Neural Network in equation (15) as



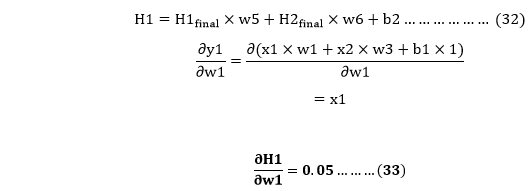
We haveBackpropagation Process in Deep Neural Networkwe need to figure outBackpropagation Process in Deep Neural Networkas



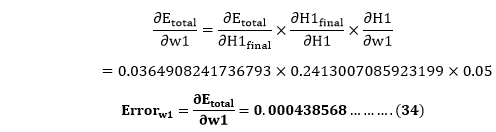
Putting the value of e-H1 in equation (30)



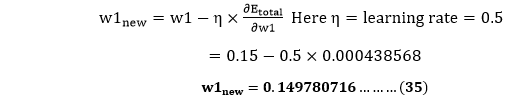
We calculate the partial derivative of the total net input to H1 with respect to w1 the same as we did for the output neuron:



So, we put the values of Backpropagation Process in Deep Neural Network in equation (13) to find the final result.



Now, we will calculate the updated weight w1new with the help of the following formula



In the same way, we calculate w2new,w3new, and w4 and this will give us the following values

**w1new=0.149780716**  
                        **w2new=0.19956143**  
                        **w3new=0.24975114**  
                        **w4new=0.29950229**

We have updated all the weights. We found the error 0.298371109 on the network when we fed forward the 0.05 and 0.1 inputs. In the first round of Backpropagation, the total error is down to 0.291027924. After repeating this process 10,000, the total error is down to 0.0000351085. At this point, the outputs neurons generate 0.159121960 and 0.984065734 i.e., nearby our target value when we feed forward the 0.05 and 0.1.

**Implementation:**

**Conclusion:**

|  |
| --- |
| Experiment No. 9 |
| Implementation of Principle Component Analysis as Dimensionality Reduction Technique. |
| Date of Performance: |
| Date of Submission: |
| Marks: |
| Sign: |

**Aim:** Implementation of Principle Component Analysis as Dimensionality Reduction Technique.

**Objective:** Able to perform Singular Value Decomposition for obtaining Dimensionality Reduction that yields Principal Component Analysis results.

**Theory:**

Principal Component Analysis is an unsupervised learning algorithm that is used for the dimensionality reduction in [machine learning](https://www.javatpoint.com/machine-learning). It is a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformation. These new transformed features are called the **Principal Components**. It is one of the popular tools that is used for exploratory data analysis and predictive modeling. It is a technique to draw strong patterns from the given dataset by reducing the variances.PCA generally tries to find the lower-dimensional surface to project the high-dimensional data.

PCA works by considering the variance of each attribute because the high attribute shows the good split between the classes, and hence it reduces the dimensionality. Some real-world applications of PCA are ***image processing, movie recommendation system, optimizing the power allocation in various communication channels.*** It is a feature extraction technique, so it contains the important variables and drops the least important variable.

The PCA algorithm is based on some mathematical concepts such as:

* Variance and Covariance
* Eigenvalues and Eigen factors

Some common terms used in PCA algorithm:

* **Dimensionality:** It is the number of features or variables present in the given dataset. More easily, it is the number of columns present in the dataset.
* **Correlation:** It signifies that how strongly two variables are related to each other. Such as if one changes, the other variable also gets changed. The correlation value

ranges from -1 to +1. Here, -1 occurs if variables are inversely proportional to each other, and +1 indicates that variables are directly proportional to each other.

* **Orthogonal:** It defines that variables are not correlated to each other, and hence the correlation between the pair of variables is zero.
* **Eigenvectors:** If there is a square matrix M, and a non-zero vector v is given. Then v will be eigenvector if Av is the scalar multiple of v.
* **Covariance Matrix:** A matrix containing the covariance between the pair of variables is called the Covariance Matrix.

### Principal Components in PCA

As described above, the transformed new features or the output of PCA are the Principal Components. The number of these PCs are either equal to or less than the original features present in the dataset. Some properties of these principal components are given below:

* The principal component must be the linear combination of the original features.
* These components are orthogonal, i.e., the correlation between a pair of variables is zero.
* The importance of each component decreases when going to 1 to n, it means the 1 PC has the most importance, and n PC will have the least importance.

### Steps for PCA algorithm

1. **Getting the dataset**  
   Firstly, we need to take the input dataset and divide it into two subparts X and Y, where X is the training set, and Y is the validation set.
2. **Representing data into a structure**  
   Now we will represent our dataset into a structure. Such as we will represent the two-dimensional matrix of independent variable X. Here each row corresponds to the data items, and the column corresponds to the Features. The number of columns is the dimensions of the dataset.
3. **Standardizing the data**  
   In this step, we will standardize our dataset. Such as in a particular column, the features with high variance are more important compared to the features with lower variance.  
   If the importance of features is independent of the variance of the feature, then we will divide each data item in a column with the standard deviation of the column. Here we will name the matrix as Z.
4. **Calculating the Covariance of Z**  
   To calculate the covariance of Z, we will take the matrix Z, and will transpose it. After transpose, we will multiply it by Z. The output matrix will be the Covariance matrix of Z.
5. **Calculating the Eigen Values and Eigen Vectors**  
   Now we need to calculate the eigenvalues and eigenvectors for the resultant covariance matrix Z. Eigenvectors or the covariance matrix are the directions of the axes with high information. And the coefficients of these eigenvectors are defined as the eigenvalues.
6. **Sorting the Eigen Vectors**  
   In this step, we will take all the eigenvalues and will sort them in decreasing order, which means from largest to smallest. And simultaneously sort the eigenvectors accordingly in matrix P of eigenvalues. The resultant matrix will be named as P\*.
7. **Calculating the new features Or Principal Components**  
   Here we will calculate the new features. To do this, we will multiply the P\* matrix to the Z. In the resultant matrix Z\*, each observation is the linear combination of original features. Each column of the Z\* matrix is independent of each other.
8. **Remove less or unimportant features from the new dataset.**  
   The new feature set has occurred, so we will decide here what to keep and what to remove. It means, we will only keep the relevant or important features in the new dataset, and unimportant features will be removed out.

## Applications of Principal Component Analysis

* PCA is mainly used as the dimensionality reduction technique in various AI applications such **as computer vision, image compression, etc.**
* It can also be used for finding hidden patterns if data has high dimensions. Some fields where PCA is used are Finance, data mining, Psychology, etc.

**Implementation:**

**Conclusion:**

1. **What is PCA ?**
2. **How it is used?**

|  |
| --- |
| Experiment No. 10 |
| Case Study: Applications of above algorithms as a case study (E.g. Hand Writing Recognition using MNIST data set, classification using IRIS data set, etc) |
| Date of Performance: |
| Date of Submission: |
| Marks: |
| Sign: |