# Competency Based Learning Material (CBLM)

## AI in Immersive Technology

**Level-6**

## Module: Apply Math Skills for Machine Learning

**Code: OU-ICT-AIIT-01-L6-V1**

**National Skills Development Authority**
**Chief Advisor's Office**
**Government of the People's Republic of Bangladesh**

## Copyright

National Skills Development Authority
Chief Advisor's Office
Level: 10-11, Biniyog Bhaban,
E-6 / B, Agargaon, Sher-E-Bangla Nagar Dhaka-1207, Bangladesh.
Email: ec@nsda.gov.bd
Website: www.nsda.gov.bd.
National Skills Portal: http:\\skillsportal.gov.bd

This Competency Based Learning Materials (CBLM) on "Apply Math Skills for Machine Learning" under the AI in Immersive Technology, Level-6" qualification is developed based on the national competency standard approved by National Skills Development Authority (NSDA)

This document is to be used as a key reference point by the competency-based learning materials developers, teachers/trainers/assessors as a base on which to build instructional activities.



It serves as the document for providing training consistent with the requirements of industry in order to meet the qualification of individuals who graduated through the established standard via competency-based assessment for a relevant job.

This document has been developed by NSDA in association with industry representatives, academia, related specialist, trainer and related employee.

Public and private institutions may use the information contained in this CBLM for activities benefitting Bangladesh.

Approved by the Authority ..... meeting held on .....

# How to use this Competency Based Learning Material (CBLM)

The module, Apply Math Skills for Machine Learning contains training materials and activities for you to complete. These activities may be completed as part of structured classroom activities or you may be required you to work at your own pace. These activities will ask you to complete associated learning and practice activities in order to gain knowledge and skills you need to achieve the learning outcomes.

1. Review the **Learning Activity** page to understand the sequence of learning activities you will undergo. This page will serve as your road map towards the achievement of competence.

2. Read the **Information Sheets.** This will give you an understanding of the jobs or tasks you are going to learn how to do. Once you have finished reading the **Information Sheets** complete the questions in the **Self-Check.**

3. **Self-**Checks are found after each **Information Sheet**. **Self-Checks** are designed to help you know how you are progressing. If you are unable to answer the questions in the **Self-Check** you will need to re-read the relevant **Information Sheet**. Once you have completed all the questions check your answers by reading the relevant **Answer Keys** found at the end of this module.

4. Next move on to the **Job Sheets. Job Sheets** provide detailed information about *how to do the job* you are being trained in. Some **Job Sheets** will also have a series of **Activity Sheets**. These sheets have been designed to introduce you to the job step by step. This is where you will apply the new knowledge you gained by reading the Information Sheets. This is your opportunity to practise the job. You may need to practise the job or activity several times before you become competent.

5. Specification **sheets**, specifying the details of the job to be performed will be provided where appropriate.

6. A review of competency is provided on the last page to help remind if all the required assessment criteria have been met. This record is for your own information and guidance and is not an official record of competency

When working though this Module always be aware of your safety and the safety of others in the training room. Should you require assistance or clarification please consult your trainer or facilitator.

When you have satisfactorily completed all the Jobs and/or Activities outlined in this module, an assessment event will be scheduled to assess if you have achieved competency in the specified learning outcomes. You will then be ready to move onto the next Unit of Competency or Module

## Table of Contents

# Module Content

| | |
|---|---|
| **Unit of Competency** | **Apply Math Skills for Machine Learning** |
| **Unit Code** | OU-ICT-AIIT-01-L6-V1 |
| **Module Title** | **Apply Math Skills for Machine Learning** |
| Module Descriptor | This unit covers the knowledge, skills and attitude required to apply Math Skills for Machine. It specifically includes the requirements of applying statistical measures, using multivariable calculus, applying Linear Algebra, and using optimization methods. |
| Nominal Hours | 20 Hours |
| Learning Outcome | After completing the practice of the module, the trainees will be able to perform the following jobs:<br>1. Apply Statistical measures.<br>2. Use Multivariable Calculus<br>3. Apply Linear Algebra<br>4. Use Optimization Methods |

**Assessment Criteria**

1. Measures of central tendency
2. Random variable and probability distribution
3. Hypothesis Testing
4. Multi variable Functions are exercised
5. Derivatives and gradients are exercised
6. Activation functions
7. Cost function is exercised
8. Plotting of functions are applied
9. Minimum and Maximum values of a function are determined
10. Matrices are exercised
11. Vectors are exercised
12. Objective function is applied
13. Likelihood function is applied
14. Error function is applied
15. Gradient Descent Algorithm and its variants are applied

## Learning Outcome 1: Apply Statistical measures

| | |
|---|---|
| Assessment Criteria | 1. Measures of central tendency<br>2. Random variable and probability distribution<br>3. Hypothesis Testing |
| Conditions and Resources | 1. Real or simulated workplace<br>2. CBLM<br>3. Handouts<br>4. Laptop<br>5. Multimedia Projector<br>6. Paper, Pen, Pencil, Eraser<br>7. Internet facilities<br>8. White board and marker<br>9. Audio Video Device |
| Contents | 1 Measures of central tendency<br>2 Random variable and probability distribution<br>3 Hypothesis Testing |
| Activities/job/Task | 1. Apply Statistical measures<br>2. Measures of Central Tendency:<br>a. Calculate the mean, median, and mode for the following dataset: [10, 15, 20, 25, 30, 35, 40, 45, 50].<br>b. Compute the standard deviation and variance for the same dataset.<br>3. Random Variables and Probability Distribution:<br>a. Define a random variable and explain its significance in probability theory.<br>b. Describe the concept of a probability distribution. Provide examples of discrete and continuous probability distributions.<br>c. Calculate the correlation coefficient and covariance matrix for the following dataset:<br>X = [2, 4, 6, 8, 10]<br>Y = [1, 3, 5, 7, 9]<br>4. Probability Distributions and p-value:<br>a. Explain the characteristics and applications of the Binomial, Poisson, and Normal probability distributions.<br>b. Given a Binomial distribution with n=10 and p=0.3, calculate the probability of getting exactly 3 successes.<br>c. Define the p-value and explain its significance in hypothesis testing. |

| | |
|---|---|
| | 5. Bayes' Theorem and Performance Metrics:<br>a. Describe Bayes' Theorem and explain how it is used in probabilistic reasoning.<br>b. Define Precision, Recall, Positive Predictive Value (PPV), and Negative Predictive Value (NPV). Discuss their roles in evaluating classification models.<br>c. Explain the concepts of Confusion Matrix and ROC Curve in evaluating the performance of classification models.<br><br>6. Hypothesis Testing:<br>a. Define A/B Testing and explain its application in statistical hypothesis testing.<br>b. Design an A/B test scenario for a website aiming to improve user engagement. Outline the hypothesis, experimental setup, and metrics for evaluation.<br>c. Explain the concept of Monte Carlo Simulation and discuss its advantages in solving complex probabilistic problems. |
| Training Methods | 1. Discussion<br>2. Presentation<br>3. Demonstration<br>4. Guided Practice<br>5. Individual Practice<br>6. Project Work<br>7. Problem Solving<br>8. Brainstorming |
| Assessment Methods | Assessment methods may include but not limited to<br>1. Written Test<br>2. Demonstration<br>3. Oral Questioning<br>4. Portfolio |

**Learning Experience 1: Apply Statistical measures**

In order to achieve the objectives stated in this learning guide, you must perform the learning steps below. Beside each step are the resources or special instructions you will use to accomplish the corresponding activity.

| Learning Activities | Recourses/Special Instructions |
|---|---|
| 1. Trainee will ask the instructor about the learning materials | a) Instructor will provide the learning materials 'Apply Statistical measures' |
| 2. Read the Information sheet and complete the Self Checks & Check answer sheets on "Apply Statistical measures" | b) Read Information sheet 1: Apply Statistical measures<br>c) Answer Self-check 1: Apply Statistical measures<br>d) Check your answer with Answer key 1: Apply Statistical measures |
| 3. Read the Job/Task Sheet and Specification Sheet and perform job/Task | e) Job/Task Sheet and Specification Sheet<br>Task Sheet-1.1: Apply Statistical measures.<br>Task Sheet-1.2: Measures of Central Tendency: a. Calculate the mean, median, and mode for the following dataset: [10, 15, 20, 25, 30, 35, 40, 45, 50]. b. Compute the standard deviation and variance for the same dataset<br>Task Sheet-1.3: Random Variables and Probability Distribution: a. Define a random variable and explain its significance in probability theory. b. Describe the concept of a probability distribution. Provide examples of discrete and continuous probability distributions. c. Calculate the correlation coefficient and covariance matrix for the following dataset: X = [2, 4, 6, 8, 10] Y = [1, 3, 5, 7, 9]<br>Task Sheet-1.4: Probability Distributions and p-value: a. Explain the characteristics and applications of the Binomial, Poisson, and Normal probability distributions. b. Given a Binomial distribution with n=10 and p=0.3, calculate the probability of getting exactly 3 successes. c. Define the p-value and explain its significance in hypothesis testing. |

| | |
|---|---|
| | Task Sheet-1.5: Bayes' Theorem and Performance Metrics: a. Describe Bayes' Theorem and explain how it is used in probabilistic reasoning. b. Define Precision, Recall, Positive Predictive Value (PPV), and Negative Predictive Value (NPV). Discuss their roles in evaluating classification models. c. Explain the concepts of Confusion Matrix and ROC Curve in evaluating the performance of classification models.<br>Task Sheet-1.6: Hypothesis Testing: a. Define A/B Testing and explain its application in statistical hypothesis testing. b. Design an A/B test scenario for a website aiming to improve user engagement. Outline the hypothesis, experimental setup, and metrics for evaluation. c. Explain the concept of Monte Carlo Simulation and discuss its advantages in solving complex probabilistic problems. |

## Information Sheet 1: Apply Statistical measures

**Learning Objective:**

After completion of this information sheet, the learners will be able to explain, define and interpret the following contents:

1. Measures of central tendency
2. Random variable and probability distribution
3. Hypothesis Testing

## 1. Measures of central tendency

Central tendency is a statistical measure that identifies the central point or typical value around which a dataset is distributed. It provides a single value that represents the "center" of a distribution or a set of data points.

Central tendency is crucial in AI for understanding the underlying patterns and distributions within data, which is essential for training accuracy machine learning models. By providing summary statistics such as the mean, median, and mode, central tendency helps in preprocessing tasks like outlier detection and data normalization, improving model performance. Moreover, central tendency measures serve as baseline predictors, aiding in the initial assessment of model effectiveness and guiding feature selection processes. They also facilitate decision-making by offering concise representations of data distributions, supporting stakeholders in making informed choices based on statistical insights. Overall, central tendency is a fundamental concept in AI that underpins various aspects of model development, evaluation, and deployment.

The most common measures of central tendency are the mean, median, and mode.

### a. Mean

Mean is the simple mathematical average of a set of two or more numbers.

The mean for a given set of numbers can be computed in more than one way, including the arithmetic mean method, which uses the sum of the numbers in the series, and the geometric mean method, which is the average of a set of products. However, all the primary methods of computing a simple average produce the same approximate result most of the time.

- **Understanding Mean**

The mean is a statistical indicator that can be used to gauge performance over time. Specific to investing, the mean is used to understand the performance of a company's stock price over a period of days, months, or years.

An analyst who wants to measure the trajectory of a company's stock value in, say, the last 10 days would sum up the closing price of the stock in each of the 10 days. The sum total would then be divided by the number of days to get the arithmetic mean. The geometric mean will be calculated by multiplying all of the values together. The nth root of the product total is then taken—in this case, the $10^{th}$ root—to get the mean.

- **Formulas for Arithmetic Mean and Geometric Mean**

Calculations for both the arithmetic and geometric means are fairly similar. The calculated amount for one will not substantially vary from another. However, there are subtle differences between the two approaches that do lead to different numbers.

**i. Arithmetic Mean**

The formula for calculating the arithmetic mean is to add up all figures and divide by the quantity of figures used. For example, the arithmetic mean of the numbers 4 and 9 is found by adding 4 and 9 together, then dividing by 2 (the quantity of numbers we are using). The arithmetic mean in this example is 6.5.

- **Pros of Arithmetic Mean:**
  - It is easier to calculate
  - It is simpler for following along and audit results.
  - Its calculated value is a finite number.
  - It has more widespread use in algebraic computations.
  - It is often the fastest type of mean to calculate.

- **Cons:**
  - It is highly affected by material outliers or extreme numbers outside of a data set.
  - It is not as useful for skewed distributions.
  - It is not useful when using time series data (or other series of data with varying basis).
  - It weighs every item equally, diminishing the importance of more impactful data points.

### ii. Geometric Mean

The geometric mean is more complicated and uses a more complex formula. To get the formula for calculating the geometric mean is to multiply all values within a data set. Then, take the root of the sum equal to the quantity of values within that data set. For example, to calculate the geometric of the values 4 and 9, multiply the two numbers together to get 36. Then, take the square root (since there are two values). The geometric mean in this example is 6.

- **Pros:**
  - It is less likely to be impacted by extreme outliers.
  - It returns a more accurate measurement for more volatile data sets.
  - It considers the effects of compounding.
  - It is more accurate when using a data set over a long period of time (due to compounding).

- **Mean of Grouped Data**

Mean of grouped data is the data set formed by aggregating individual observations of a variable into different groups. Grouped data is data that is grouped together in different categories. Mean is considered as the average of the data. For the mean of grouped data, it might be difficult to find the exact value however, we can always estimate it. Let us learn more about the mean of grouped data, the methods to find the mean of grouped data, and solve a few examples to understand this concept better.

Mean of grouped data is the process of finding the average of a set of data that are grouped together in different categories. To determine the mean of a grouped data, a frequency table is required to set across the frequencies of the data which makes it simple to calculate. There are three main methods of calculating the mean of grouped data, they are - direct method, assumed mean method, and step deviation method. Each of these methods has its own formulas and ways to calculate the mean.

- **Mean of Grouped Data Formula**

The mean formula is defined as the sum of the observations divided by the total number of observations. There are two different formulas for calculating the mean for ungrouped data and the mean for grouped data. Let us look at the formula to calculate the mean of grouped data. The formula is: $\bar{x} = \Sigma fi/N$
Where,

- $\bar{x}$ = the mean value of the set of given data.
- f = frequency of the individual data
- N = sum of frequencies

Hence, the average of all the data points is termed as mean.

- **Direct Methods of Calculating Mean of Grouped Data**

The direct method is the simplest method to find the mean of the grouped data. If the values of the observations are $x11, x22, x33, \ldots\ldots xn$ with their corresponding frequencies are $f11, f22, f33, \ldots\ldots fn$ then the mean of the data is given by,

$$\bar{x} = \frac{(x1f1 + x2f2 + x3f3 + \ldots\ldots xn)}{f1 + f2 + f3 + \ldots\ldots fn}$$

$\bar{x} = \sum xifi \,/\, \sum fi$ where i = 1, 2, 3, 4,.....n

Here are the steps that can be followed to find the mean for grouped data using the direct method,

- Create a table containing four columns such as class interval, class marks (corresponding), denoted by xi, frequencies fi corresponding), and xifi.
- Calculate Mean by the Formula Mean = ∑xifi/ ∑fi. Where fi is the frequency and xi is the midpoint of the class interval.

- Calculate the midpoint, xi, we use this formula xi= (upper class limit + lower class limit)/2.

**Example of Ungrouped Data:**

Suppose we have the following dataset representing the daily temperatures (in degrees Celsius) for a particular week:

{20,22,19,25,18,23,21}

To calculate the mean temperature for the week, we add up all the temperatures and then divide by the total number of days:

Mean=(20+22+19+25+18+23+21)/7

Mean ≈ 21.14

So, the mean temperature for the week is approximately 21.14 Celsius.

**Example of Mean for grouped data:**

| Class Interval | 0 - 10 | 10 - 20 | 20 - 30 | 30 - 40 | 40 - 50 |
|---|---|---|---|---|---|
| Frequency (fi) | 9 | 13 | 8 | 15 | 10 |

**Solution:** The first step is to create the table with the midpoint or marks and the product of the frequency and midpoint. To calculate the midpoint we find the average between the class intervals by using the formula mentioned above.

Midpoint xi = 0 - 10 = 5 ([10 + 0]/2), 10 - 20 = 15 ([20 + 10]/2) and so on.

xifi = For the class interval 0 - 10 = 5 × 9 = 45, For the class interval 10 - 20 = 13 × 15 = 195 and so on.

| **Class Interval** | **Frequency (fi)** | **Class Mark (xi)** | **xifi** |
|---|---|---|---|
| 0 - 10 | 9 | 5 | 45 |
| 10 - 20 | 13 | 15 | 195 |
| 20 - 30 | 8 | 25 | 200 |

| | | | |
|---|---|---|---|
| 30 - 40 | 15 | 35 | 525 |
| 40 - 50 | 10 | 45 | 450 |
| Total | 55 | | 1415 |

Once we have determined the totals, let us use the formula to calculate the estimated mean.

Estimated Mean = $\sum xifi/ \sum fi = 1415/55 = 25.73$.

**Importance of Mean**

Within business and investing, mean is used extensively to analyze performance. Examples of situations in which you may encounter mean include:

- Determining whether an equity is trading above or below its average over a specified time period.
- Looking back to see how comparative trading activity may determine future outcomes. For example, seeing the average rate of return for broad markets during prior recessions may guide decision making in future economic downturns.
- Seeing whether trading volume or the quantity of market orders is in line with recent market activity.
- Analyzing the operational performance of a company. For instance, some financial ratios like days sales outstanding require determining the average accounts receivable balance for the numerator.
- Quantifying macroeconomic data like average unemployment over a period of time to determine the general health of an economy.

## b. Median

The term median refers to a metric used in statistics. It is the middle number in a sorted ascending or descending list of numbers and can be more descriptive of that data set than the average. It is the point above and below which half (50%) of the observed data falls, and so represents the midpoint of the data. The median is often compared with other descriptive statistics such as the mean (average), mode, and standard deviation.

- **Understanding the Median**

Statistics is a branch of mathematics. It involves the collection and study of data, which allows researchers to make inferences or determinations about a certain topic. The analysis

of quantitative data can be used to study anything from demographics, populations, and investments among other things.

A median is the middle number in a sorted list of numbers (either ascending or descending) used in statistical studies. To determine the median value in a sequence of numbers, the numbers must first be sorted or arranged in value order from lowest to highest or highest to lowest.

- If there is an odd amount of numbers, the median value is the number that is in the middle, with the same amount of numbers below and above.
- If there is an even amount of numbers in the list, the middle pair must be determined, added together, and divided by two to find the median value.

The median can be used to determine an approximate average, or mean, but is not to be confused with the actual mean.

The median is sometimes used as opposed to the mean when there are outliers in the sequence that might skew the average of the values. The median of a sequence can be less affected by outliers than the mean.

- **Median vs. Mean**

As noted above, it's important not to confuse the terms median and mean. The two may sound the same (which is a common misconception) but they are very different. A median is a number that falls in the middle of a group. Remember, this is done by ordering the numbers from smallest to largest and locating the one that falls in the middle.

A mean, on the other hand, is the average of a data set. Also called the arithmetic mean, it is the average of the sum of the numbers in a group. In order to figure out the mean, you must take the sum of the numbers in the group and divide the sum by the total number of data points. For instance, let's say a data set consists of the numbers 3, 5, 7, and 19. To figure out the mean:

Add the numbers together: $3 + 5 + 7 + 19 = 34$

Divide the sum by the number of data points: $34 \div 4 = 8.5$

In this case, the mean is 8.5. The median, on the other hand, would be 6 or $(5 + 7) \div 2$. That's because there's an even number of data points, which we add together and divide by 2 to get the result.

- **Calculate the Median**

The median is the middle value in a set of data. First, organize and order the data from smallest to largest. To find the midpoint value, divide the number of observations by two.

If there is an odd number of observations, round that number up, and the value in that position is the median. If the number of observations is even, take the average of the values found above and below that position.

**Example of a Median**

To find the median value in a list with an odd amount of numbers, one would find the number that is in the middle with an equal amount of numbers on either side of the median. To find the median, first arrange the numbers in order, usually from lowest to highest.

For example, in a data set of {3, 13, 2, 34, 11, 26, 47}, the sorted order becomes {2, 3, 11, 13, 26, 34, 47}. The median is the number in the middle {2, 3, 11, 13, 26, 34, 47}, which in this instance is 13 since there are three numbers on either side.

To find the median value in a list with an even amount of numbers, one must determine the middle pair, add them, and divide by two. Again, arrange the numbers in order from lowest to highest.

For example, in a data set of {3, 13, 2, 34, 11, 17, 27, 47}, the sorted order becomes {2, 3, 11, 13, 17, 27, 34, 47}. The median is the average of the two numbers in the middle {2, 3, 11, 13, 17, 26 34, 47}, which in this case is 15 or (13 + 17) ÷ 2 = 15.

The median is the number that lies in the middle of an ordered dataset that goes from lowest to highest. It should not be confused with the mean, which is determined by adding the numbers in a set together and dividing by the total number of data points. Many experts prefer using the median over the mean because it often provides a more accurate representation of the distribution in a data set.

- **Calculate Median for Grouped Data:**

In a grouped data, it is not possible to find the median for the given observation by looking at the cumulative frequencies. The middle value of the given data will be in some class interval. So, it is necessary to find the value inside the class interval that divides the whole distribution into two halves. In this scenario, we have to find the median class.

To find the median class, we have to find the cumulative frequencies of all the classes and n/2. After that, locate the class whose cumulative frequency is greater than (nearest to) n/2. The class is called the median class.

After finding the median class, use the below formula to find the median value.

$$\text{Median} = l + \left(\frac{\frac{n}{2} - cf}{f}\right) * h$$

Where

- l is the lower limit of the median class
- n is the number of observations
- f is the frequency of median class
- h is the class size
- cf is the cumulative frequency of class preceding the median class.

Now, let us understand how to find the median of a grouped data using the formula with the help of an example.

**Where Is the Median in a Normal Distribution?**

In the normal distribution or bell curve the median, mean, and mode are all the same value and fall at the highest point in the center of the curve.

**When Are the Mean and Median Different?**

In a skewed data set, the mean and median will typically be different. The mean is calculated by adding up all of the values in the data and dividing by the number of observations. If there are sizable outliers, or if the data clumps around certain values, the mean (average) will not be the midpoint of the data.

For instance, in a set of data {0, 0, 0, 1, 1, 2, 10, 10} the average would be 24/8 = 3. The median, however, would be 1 (the midpoint value).

This is why many economists favor the median for reporting a nation's income or wealth, since it is more representative of the actual income distribution.

## c. Mode

The mode is the value that appears most frequently in a data set. A set of data may have one mode, more than one mode, or no mode at all. Other popular measures of central tendency include the mean, or the average of a set, and the median, the middle value in a set.

- In statistics, the mode is the most commonly observed value in a set of data.
- For the normal distribution, the mode is also the same value as the mean and median.
- In many cases, the modal value will differ from the average value in the data.

Example:

For example, in the following list of numbers, 16 is the mode since it appears more times in the set than any other number:

3, 3, 6, 9, **16, 16, 16**, 27, 27, 37, 48

A set of numbers can have more than one mode (this is known as *bimodal* if there are two modes) if there are multiple numbers that occur with equal frequency and more times than the others in the set.

**Mode for Grouped Data:**

Often, we may want to calculate the mode of data that is grouped in some way. Recall that the mode represents the value that occurs most frequently. For example, suppose we have the following grouped data:

| **Range** | **Frequency** |
|---|---|
| 1-10 | 2 |
| 11-20 | 7 |
| 21-30 | 10 |
| 31-40 | 3 |
| 41-50 | 1 |

While it's not possible to calculate the exact mode since we don't know the raw data values, it is possible to estimate the mode using the following formula:

**Mode of Grouped Data = L + W[(Fm – F1)/( (Fm-F1) + (Fm – F2) )]**

where:

- **L**: Lower limit of modal class
- **W**: Width of modal class
- **Fm**: Frequency of modal class
- **F1**: Frequency of class immediately before modal class
- **F2**: Frequency of class immediately after modal class

**Note**: The **modal class** is simply the class with the highest frequency. In the example above, the modal class would be 21-30 since it has the highest frequency.

The following examples show how to calculate the mode of grouped data in different scenarios.

Suppose we have the following frequency distribution that shows the exam scored receive by 40 students in a certain class:

| **Exam Score** | **Frequency** |
|---|---|
| 51-60 | 4 |
| 61-70 | 8 |
| 71-80 | 15 |
| 81-90 | 8 |
| 91-100 | 5 |

In this example, the **modal class** is 71-80.

Knowing this, we can calculate the following values:

- **L**: Lower limit of modal class: **71**
- **W**: Width of modal class: **9**
- **Fm**: Frequency of modal class: **15**
- **F1**: Frequency of class immediately before modal class: **8**
- **F2**: Frequency of class immediately after modal class: **8**

We can plug these values into the formula to calculate the mode of the distribution:

- Mode = L + W[(Fm – F1)/( (Fm-F1) + (Fm – F2) )]
- Mode = 71 + 9[(15-8) / ( (15-8) + (15-8) )]
- Mode = **75.5**

We estimate that the modal exam score is **75.5**.

### ▪ Standard Deviation

Standard deviation is a measure of the dispersion or spread of a set of data points. It quantifies how much the data values differ from the mean of the dataset. A higher standard

deviation indicates greater variability, while a lower standard deviation indicates less variability. The standard deviation is calculated as the square root of the variance.

**Formula for Standard Deviation:**

For a dataset *X* with *N* data points, the standard deviation $\sigma$ is calculated using the following formula:

$$\sigma=\sum_{i=1}^{N} \quad \frac{1}{N}(xi - \bar{x})^2$$

Where:

- *Xi* represents each individual data point.
- *X* is the mean (average) of the data points.
- *N* is the total number of data points.

## 2. Random variable and probability distribution

### a) Probability Distribution

A probability distribution is a statistical function that describes all the possible values and likelihoods that a random variable can take within a given range. This range will be bounded between the minimum and maximum possible values, but precisely where the possible value is likely to be plotted on the probability distribution depends on a number of factors. These factors include the distribution's mean (average), standard deviation, skewness, and kurtosis.

**Example of a Probability Distribution**

As a simple example of a probability distribution, let us look at the number observed when rolling two standard six-sided dice. Each die has a 1/6 probability of rolling any single number, one through six, but the sum of two dice will form the probability distribution depicted in the image below. Seven is the most common outcome (1+6, 6+1, 5+2, 2+5, 3+4, 4+3). Two and twelve, on the other hand, are far less likely (1+1 and 6+6).

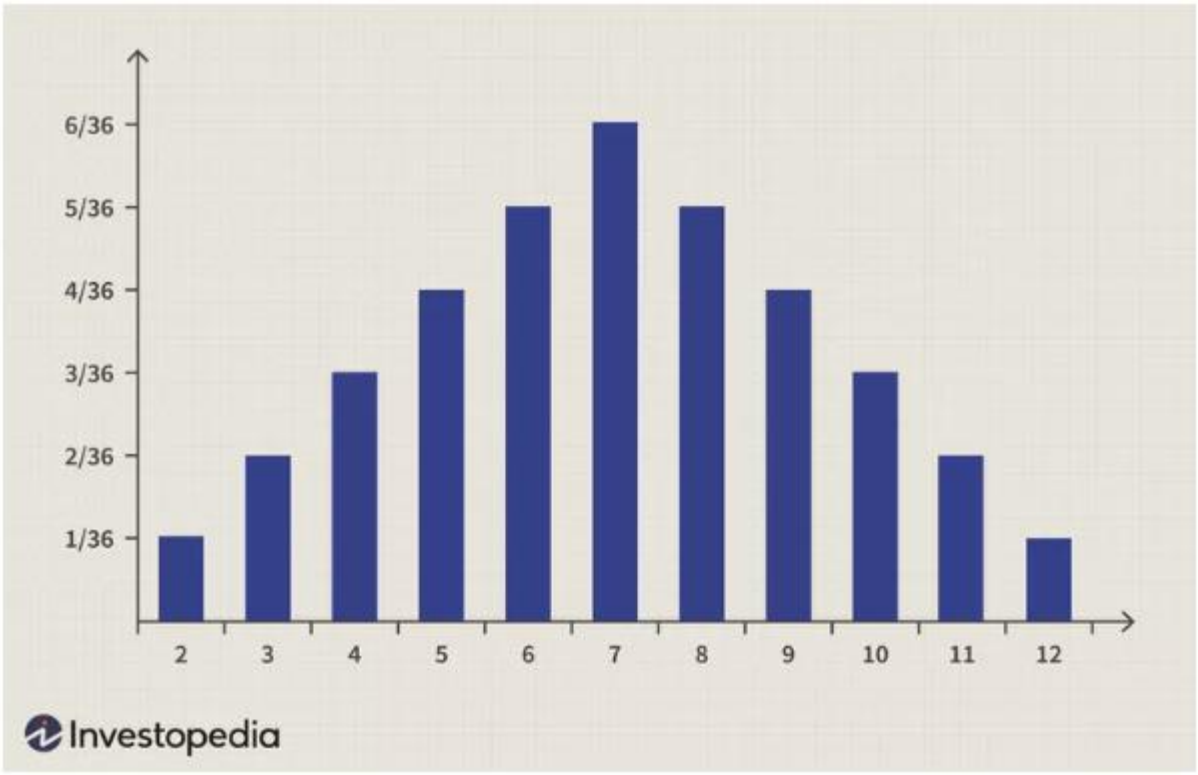



Image by Sabrina Jiang © Investopedia 2020

▪ **Types of Probability Distributions**

There are many different classifications of probability distributions. Some of them include the normal distribution, chi-square distribution, binomial distribution, and Poisson distribution. The different probability distributions serve different purposes and represent different data generation processes.

**1. Binomial**

The binomial distribution, for example, evaluates the probability of an event occurring several times over a given number of trials and given the event's probability in each trial. It may be generated, for example, by keeping track of how many free throws a basketball player makes in a game, where 1 = a basket and 0 = a miss.

Another typical example would be to use a fair coin and figure out the probability of that coin coming up heads in 10 straight flips. A binomial distribution is *discrete*, as opposed to continuous, since only 1 or 0 is a valid response.

## 2. Normal

The most commonly used distribution is the normal distribution, which is used frequently in finance, investing, science, and engineering. The normal distribution is fully characterized by its mean and standard deviation, meaning the distribution is not skewed and does exhibit kurtosis.

This makes the distribution symmetric and it is depicted as a bell-shaped curve when plotted. A normal distribution is defined by a mean (average) of zero and a standard deviation of 1.0, with a skew of zero and kurtosis = 3.

In a normal distribution, approximately 68% of the data collected will fall within +/- one standard deviation of the mean; approximately 95% within +/- two standard deviations; and 99.7% within three standard deviations. Unlike the binomial distribution, the normal distribution is continuous, meaning that all possible values are represented (as opposed to just 0 and 1 with nothing in between).

## 3. Poisson Distribution

The Poisson distribution is a discrete probability distribution that describes the number of events occurring in a fixed interval of time or space when these events happen with a known constant rate and are independent of the time since the last event. It's named after the French mathematician Siméon Denis Poisson.

## 4. Probability Mass Function (PMF)

Let $X$ be a random variable representing the number of events occurring in a fixed interval. The probability mass function (PMF) of the Poisson distribution is given by:

$$p(X = k) = \frac{e^{-\lambda}.\lambda^{k}}{k!}$$

Where:

- $k$ is the number of events that occur in the interval.
- $e$ is the base of the natural logarithm (approximately equal to 2.71828).
- $\lambda$ is the average rate (or mean) of occurrences in the interval. This parameter represents both the mean and the variance of the distribution.

Suppose you are studying the number of customers arriving at a coffee shop during the morning rush hour, and you find that on average, 10 customers arrive every 15 minutes. You can model this scenario using a Poisson distribution with $\lambda$=10. Then, you can calculate the probability of different numbers of customers arriving in the next 15 minutes using the PMF formula.

## 5. Bayes Theorem

Bayes' theorem is named after Thomas Bayes, a nonconformist English clergyman who did early work in probability and decision theory during the 18th century. Let X be a data tuple. In Bayesian terms, X is considered "evidence." As usual, it is described by measurements made on a set of n attributes.

Let's Consider an Example. Suppose that we have a database of a Computer shop named "*All Electronics' ' where* the purchase history of the computer buyer is stored. The database looks like the following table

| *RID* | *age* | *income* | *student* | *credit_rating* | *Class:buys_computer* |
|---|---|---|---|---|---|
| 1 | youth | high | no | fair | no |
| 2 | youth | high | no | excellent | no |
| 3 | middle_aged | high | no | fair | yes |
| 4 | senior | medium | no | fair | yes |
| 5 | senior | low | yes | fair | yes |
| 6 | senior | low | yes | excellent | no |
| 7 | middle_aged | low | yes | excellent | yes |
| 8 | youth | medium | no | fair | no |
| 9 | youth | low | yes | fair | yes |
| 10 | senior | medium | yes | fair | yes |
| 11 | youth | medium | yes | excellent | yes |
| 12 | middle_aged | medium | no | excellent | yes |
| 13 | middle_aged | high | yes | fair | yes |
| 14 | senior | medium | no | excellent | no |

Let H be some hypothesis such as that the data tuple X belongs to a specified class C. For classification problems, we want to determine P(H|X), the probability that the hypothesis H holds given the "evidence" or observed data tuple X. In other words, we are looking for the probability that tuple X belongs to class C, given that we know the attribute description of X. Bayes Classification Methods P(H|X) is the posterior probability, or a posteriori probability, of H conditioned on X. For example, suppose our world of data tuples is confined to customers described by the attributes age and income, respectively, and that X is a 35-year-old customer with an income of $40,000. Suppose that H is the hypothesis that our customer will buy a computer. Then P(H|X) reflects the probability that customer X will buy a computer given that we know the customer's age and income. In contrast, P(H) is the prior probability, or a priori probability, of H.

For our example, this is the probability that any given customer will buy a computer, regardless of age, income, or any other information, for that matter. The posterior probability, P(H|X), is based on more information (e.g., customer information) than the prior probability, P(H), which is independent of X. Similarly, P(X|H) is the posterior probability of X conditioned on H. That is, it is the probability that a customer, X, is 35 years old and earns $40,000, given that we know the customer will buy a computer. P(X) is the prior probability of X. Using our example, it is the probability that a person from our set of customers is 35 years old and earns $40,000. "How are these probabilities estimated?" P(H), P(X|H), and P(X) may be estimated from the given data, as we shall see next. Bayes' theorem is useful in that it provides a way of calculating the posterior probability, P(H|X), from P(H), P(X|H), and P(X). Bayes' theorem is

$$P(H|X) = P(X|H)P(H) / P(X)$$

Now that we have that out of the way, in the next section, we will look at how Bayes' theorem is used in the na¨ıve Bayesian classifier.

## 6. Naive Bayes Classification

The na¨ıve Bayesian classifier, or simple Bayesian classifier, works as follows:

■ Let D be a training set of tuples and their associated class labels. As usual, each tuple is represented by an n-dimensional attribute vector, X = (x1, x2,..., xn), depicting n measurements made on the tuple from n attributes, respectively, A1, A2,..., An.

■ Suppose that there are m classes, C1, C2,..., Cm. Given a tuple, X, the classifier will predict that X belongs to the class having the highest posterior probability, conditioned on X. That is, the na¨ıve Bayesian classifier predicts that tuple X belongs to the class Ci if and only if $P(Ci|X) > P(Cj|X)$ for $1 \leq j \leq m, j \neq i$. Thus, we maximize P(Ci |X). The class Ci for which

P(Ci |X) is maximized is called the maximum posteriori hypothesis. By Bayes' theorem P(Ci |X) = P(X|Ci)P(Ci) /P(X) .

▪ As P(X) is constant for all classes, only P(X|Ci)P(Ci) needs to be maximized. If the class prior probabilities are not known, then it is commonly assumed that the classes are equally likely, that is, P(C1) = P(C2) = ⋯ = P(Cm), and we would therefore maximize P(X|Ci). Otherwise, we maximize P(X|Ci)P(Ci). Note that the class prior probabilities may be estimated by P(Ci) = |Ci,D|/|D|, where |Ci,D| is the number of training tuples of class Ci in D.

▪ Given data sets with many attributes, it would be extremely computationally expensive to compute P(X|Ci). To reduce computation in evaluating P(X|Ci), the na¨ıve assumption of class-conditional independence is made. This presumes that the attributes' values are conditionally independent of one another, given the class label of the tuple (i.e., that there are no dependence relationships among the attributes). Thus, P(X|Ci) = Yn k=1 P(xk |Ci) (8.12) = P(x1|Ci) × P(x2|Ci) × ⋯ × P(xn|Ci). We can easily estimate the probabilities P(x1|Ci), P(x2|Ci),..., P(xn|Ci) from the training tuples. Recall that here xk refers to the value of attribute Ak for tuple X. For each attribute, we look at whether the attribute is categorical or continuous-valued. For instance, to compute P(X|Ci), we consider the following:

➢ If Ak is categorical, then P(xk |Ci) is the number of tuples of class Ci in D having the value xk for Ak , divided by |Ci,D|, the number of tuples of class Ci in D.

➢ If Ak is continuous-valued, then we need to do a bit more work, but the calculation is pretty straightforward. A continuous-valued attribute is typically assumed to have a Gaussian distribution with a mean μ and standard deviation σ, defined by g(x, μ, σ ) = 1 √ 2πσ e − (x−μ)2 2σ 2 , so that P(xk |Ci) = g(xk , μCi , σCi ). These equations may appear daunting, but hold on! We need to compute μCi and σCi , which are the mean (i.e., average) and standard deviation, respectively, of the values of attribute Ak for training tuples of class Ci . We then plug these two quantities into together with xk , to estimate P(xk |Ci).

▪ To predict the class label of X, P(X|Ci)P(Ci) is evaluated for each class Ci . The classifier predicts that the class label of tuple X is the class Ci if and only if P(X|Ci)P(Ci) > P(X|Cj)P(Cj) for 1 ≤ j ≤ m, j 6= i. (8.15) In other words, the predicted class label is the class Ci for which P(X|Ci)P(Ci) is the maximum. "How effective are Bayesian classifiers?" Various empirical studies of this classifier in comparison to decision tree and neural network classifiers have found it to be comparable in some domains. In theory, Bayesian classifiers have the minimum error rate in comparison to all other classifiers. However, in practice this is not always the case, owing to inaccuracies in the assumptions made for its use, such as class-conditional independence, and the lack of available probability data. Bayesian classifiers are also useful in that they provide a theoretical justification for other classifiers that do not explicitly use Bayes' theorem. For example, under certain assumptions, it can be shown that many neural network and curve-fitting algorithms output the maximum posteriori hypothesis, as does the na¨ıve Bayesian classifier.

Example: We wish to predict the class label of a tuple using na¨ıve Bayesian classification, given the same training data as in Example  for decision tree induction. The training data were shown earlier in Table  The data tuples are described by the attributes age, income, student, and credit rating. The class label attribute, buys computer, has two distinct values (namely, {yes, no}). Let C1 correspond to the class buys computer = yes and C2 correspond to buys computer = no. The tuple we wish to classify is

X = (age = youth, income = medium, student = yes, credit rating = fair)

We need to maximize P(X|Ci)P(Ci), for i = 1, 2. P(Ci), the prior probability of each class, can be computed based on the training tuples:

P(buys computer = yes) = 9/14 = 0.643

P(buys computer = no) = 5/14 = 0.357

To compute P(X|Ci), for i = 1, 2, we compute the following conditional probabilities:

P(age = youth | buys computer = yes) = 2/9 = 0.222

P(age = youth | buys computer = no) = 3/5 = 0.600

P(income = medium | buys computer = yes) = 4/9 = 0.444

P(income = medium | buys computer = no) = 2/5 = 0.400

P(student = yes | buys computer = yes) = 6/9 = 0.667

P(student = yes | buys computer = no) = 1/5 = 0.200

P(credit rating = fair | buys computer = yes) = 6/9 = 0.667

P(credit rating = fair | buys computer = no) = 2/5 = 0.400

Using these probabilities, we obtain

P(X|buys computer = yes) = P(age = youth | buys computer = yes) × P(income = medium | buys computer = yes) × P(student = yes | buys computer = yes) × P(credit rating = fair | buys computer = yes) = 0.222 × 0.444 × 0.667 × 0.667 = 0.044.

Similarly, P(X|buys computer = no) = 0.600 × 0.400 × 0.200 × 0.400 = 0.019.

To find the class, Ci , that maximizes P(X|Ci)P(Ci), we compute

P(X|buys computer = yes)P(buys computer = yes) = 0.044 × 0.643 = 0.028

P(X|buys computer = no)P(buys computer = no) = 0.019 × 0.357 = 0.007

Therefore, the na¨ıve Bayesian classifier predicts buys computer = yes for tuple X. “What if I encounter probability values of zero?” we estimate P(X|Ci) as the product of the probabilities P(x1|Ci), P(x2|Ci),..., P(xn|Ci), based on the assumption of class-conditional independence. These probabilities can be estimated from the training tuples (step 4). We need to compute P(X|Ci) for each class (i = 1,2,...,m) to find the class Ci for which P(X|Ci)P(Ci) is the maximum (step 5). Let’s consider this calculation. For each attribute–value pair (i.e., Ak = xk , for k = 1,2,...,n) in tuple X, we need to count the number of tuples having that attribute–value pair, per class (i.e., per Ci , for i = 1,..., m). In Example we have two classes (m = 2), namely buys computer = yes and buys computer = no.

Therefore, for the attribute–value pair student = yes of X, say, we need two counts—the number of customers who are students and for which buys computer = yes (which contributes to P(X|buys computer = yes)) and the number of customers who are students and for which buys computer = no (which contributes to P(X|buys computer = no)). But what if, say, there are no training tuples representing students for the class buys computer = no, resulting in P(student = yes|buys computer = no) = 0? In other words, what happens if we should end up with a probability value of zero for some P(xk |Ci)? Plugging this zero value into Equation would return a zero probability for P(X|Ci), even though, without the zero probability, we may have ended up with a high probability, suggesting that X belonged to class Ci ! A zero probability cancels the effects of all the other (posteriori) probabilities (on Ci) involved in the product.

There is a simple trick to avoid this problem. We can assume that our training database, D, is so large that adding one to each count that we need would only make a negligible difference in the estimated probability value, yet would conveniently avoid the case of probability values of zero. This technique for probability estimation is known as the Laplacian correction or Laplace estimator, named after Pierre Laplace, a French mathematician who lived from 1749 to 1827. If we have, say, q counts to which we each add one, then we must remember to add q to the corresponding denominator used in the probability calculation.

Using the Laplacian correction to avoid computing probability values of zero. Suppose that the class buys a computer = yes in some training database, D, containing 1000 tuples, we have 0 tuples with income = low, 990 tuples with income = medium, and 10 tuples with income = high. The probabilities of these events, without the Laplacian correction, are 0, 0.990 (from 990/1000), and 0.010 (from 10/1000), respectively. Using the Laplacian correction for the three quantities, we pretend that we have 1 more tuple for each income-value pair. In this way, we instead obtain the following probabilities (rounded up to three decimal places): 1 1003 = 0.001, 991 1003 = 0.988, and 11 1003 = 0.011, respectively. The

"corrected" probability estimates are close to their "uncorrected" counterparts, yet the zero probability value is avoided.

▪ **Metrics for Evaluating Classifier Performance**

This section presents measures for assessing how good or how "accurate" your classifier is at predicting the class label of tuples. We will consider the case of where the class tuples are more or less evenly distributed, as well as the case where classes are unbalanced (e.g., where an important class of interest is rare such as in medical tests). The classifier evaluation measures presented in this section are summarized. They include accuracy (also known as recognition rate), sensitivity (or recall), specificity, precision, F1, and Fβ. Note that although accuracy is a specific measure, the word "accuracy" is also used as a general term to refer to a classifier's predictive abilities. Using training data to derive a classifier and then estimate the accuracy of the resulting learned model can result in misleading overoptimistic estimates due to overspecialization of the learning algorithm to the data.

Instead, it is better to measure the classifier's accuracy on a test set consisting of class-labeled tuples that were not used to train the model. Before we discuss the various measures, we need to become comfortable with some terminology. Recall that we can talk in terms of positive tuples (tuples of the main class of interest) and negative tuples (all other tuples).6 Given two classes, for example, the positive tuples may be buys computer = yes while the negative tuples are buys computer = no. Suppose we use our classifier on a test set of labeled tuples. P is the number of positive tuples and N is the number of negative tuples. For each tuple, we compare the classifier's class label prediction with the tuple's known class label. There are four additional terms we need to know that are the "building blocks" used in computing many evaluation measures. Understanding them will make it easy to grasp the meaning of the various measures.

▪ **True positives (TP):** These refer to the positive tuples that were correctly labeled by the classifier. Let To be the number of true positives.
▪ **True negatives(TN):** These are the negative tuples that were correctly labeled by the classifier. Let TN be the number of true negatives.
▪ **False positives (FP):** These are the negative tuples that were incorrectly labeled as positive (e.g., tuples of class buys computer = no for which the classifier predicted buys computer = yes). Let FP be the number of false positives.
▪ **False negatives (FN):** These are the positive tuples that were mislabeled as negative (e.g., tuples of class buys computer = yes for which the classifier predicted buys computer = no). Let FN be the number of false negatives.

▪ **Confusion Matrix**

The confusion matrix is a useful tool for analyzing how well your classifier can recognize tuples of different classes. TP and TN tell us when the classifier is getting things right, while FP and FN tell us when the classifier is getting things wrong( i.e mislabeling). Given m classes (where m ≥ 2), a confusion matrix is a table of at least size m by m. An entry, CMi,j in the first m rows and m columns indicates the number of tuples of class i that were labeled by the classifier as class j. For a classifier to have good accuracy, ideally most of the tuples would be represented along the diagonal of the confusion matrix, from entry CM1,1 to entry CMm,m, with the rest of the entries being zero or close to zero. That is, ideally, FP and FN are around zero. The table may have additional rows or columns to provide totals. For example, in the confusion matrix

| | | Predicted class | | |
|---|---|---|---|---|
| | | *yes* | *no* | Total |
| **Actual class** | *yes* | *TP* | *FN* | *P* |
| | *no* | *FP* | *TN* | *N* |
| | Total | $P'$ | $N'$ | $P + N$ |

P and N are shown. In addition, P 0 is the number of tuples that were labeled as positive (TP + FP) and N0 is the number of tuples that were labeled as negative (TN + FN). The total number of tuples is TP + TN + FP + TN, or P + N, or P 0 + N0 . Note that although the confusion matrix shown is for a binary classification problem, confusion matrices can be easily drawn for multiple classes in a similar manner. Now let's look at the evaluation measures, starting with accuracy. The accuracy of a classifier on a given test set is the percentage of test set tuples that are correctly classified by the classifier. That is,

accuracy $= \frac{TP + TN}{P + N}$

In the pattern recognition literature, this is also referred to as the overall recognition rate of the classifier, that is, it reflects how well the classifier recognizes tuples of the various classes. An example of a confusion matrix for the two classes buying computer = yes (positive) and buying computer = no (negative) is given in the following table.

| Classes | buys_computer = yes | buys_computer = no | Total | Recognition(%) |
|---|---|---|---|---|
| buys_computer = yes | 6954 | 46 | 7000 | 99.34 |
| buys_computer = no | 412 | 2588 | 3000 | 86.27 |
| Total | 7366 | 2634 | 10,000 | 95.42 |

Confusion matrix for the classes buys computer = yes and buys computer = no, where an entry in row i and column j shows the number of tuples of class i that were labeled by the classifier as class j. Ideally, the non diagonal entries should be zero or close to zero.Totals are shown as well as the recognition rates per class and overall. By glancing at a confusion matrix, it is easy to see if the corresponding classifier is confusing two classes. For example, we see that it mislabeled 412 "no" tuples as "yes." Accuracy is most effective when the class distribution is relatively balanced. We can also speak of the error rate or misclassification rate of a classifier, M, which is simply 1 − accuracy(M), where accuracy(M) is the accuracy of M. This also can be computed as

$$error\ rate\ = \frac{FP\ +\ FN}{P + N}$$

If we were to use the training set (instead of a test set) to estimate the error rate of a model, this quantity is known as the resubstitution error. This error estimate is optimistic of the true error rate (and similarly, the corresponding accuracy estimate is optimistic) because the model is not tested on any samples that it has not already seen. We now consider the class imbalance problem, where the main class of interest is rare. That is, the data set distribution reflects a significant majority of the negative class and a minority positive class.

For example, in fraud detection applications, the class of interest (or positive class) is "fraud," which occurs much less frequently than the negative "nonfraudulant" class. In medical data, there may be a rare class, such as "cancer." Suppose that you have trained a classifier to classify medical data tuples, where the class label attribute is "cancer" and the possible class values are "yes" and "no." An accuracy rate of, say, 97% may make the classifier seem quite accurate, but what if only, say, 3% of the training tuples are actually cancer?

Clearly, an accuracy rate of 97% may not be acceptable—the classifier could be correctly labeling only the noncancer tuples, for instance, and misclassifying all the cancer tuples. Instead, we need other measures, which access how well the classifier can recognize the positive tuples (cancer = yes) and how well it can recognize the negative tuples (cancer = no). The sensitivity and specificity measures can be used, respectively, for this purpose. Sensitivity is also referred to as the true positive (recognition) rate (i.e., the proportion of positive tuples that are correctly identified), while specificity is the true negative rate (i.e., the proportion of negative tuples that are correctly identified). These measures are defined as

$$\text{sensitivity} = \frac{TP}{p}$$

$$\text{specificity} = \frac{TN}{N}$$

It can be shown that accuracy is a function of sensitivity and specificity:

$$\text{accuracy} = \text{sensitivity}\frac{P}{P+N} + \text{specificity}\frac{N}{P+N}$$

Example: Sensitivity and specificity. Figure 8.16 shows a confusion matrix for medical data where the class values are yes and no for a class label attribute, cancer. The sensitivity of the classifier is 90 300 = 30.00%. The specificity is 9560 9700 = 98.56%. The classifier's overall accuracy is 9650 10,000 = 96.50%. Thus, we note that although the classifier has a high accuracy, it's ability to correctly label the positive (rare) class is poor given its low sensitivity.

It has high specificity, meaning that it can accurately recognize negative tuples.

| Classes | yes | no | Total | Recognition(%) |
|---|---|---|---|---|
| yes | 90 | 210 | 300 | 30 |
| no | 140 | 9560 | 9700 | 98.56 |
| Total | 230 | 9770 | 10,000 | 96.40 |

▪ **Precision**

Precision is a measure of the accuracy of the positive predictions made by a classification model. It is calculated as the ratio of true positive predictions to the total number of positive predictions (both true positives and false positives).

Precision=True PositivesTrue Positives+False Positives

$$\text{Precision} = \frac{True\ Positives + False\ Positives}{True\ Positives}$$

▪ **Recall (Sensitivity)**

Recall, also known as sensitivity or true positive rate, measures the proportion of actual positives that are correctly identified by a classification model. It is calculated as the ratio of true positive predictions to the total number of actual positives (true positives and false negatives).

Recall=True PositivesTrue Positives+False Negatives

$$\text{Recall}= \frac{True\ Positives+False\ Negatives}{True\ Positives}$$

**Example of Precision and Recall:**

The precision of the classifier for the yes class is 90 230 = 39.13%. The recall is 90 300 = 30.00%, which is the same calculation for sensitivity A perfect precision score of 1.0 for a class C means that every tuple that the classifier labeled as belonging to class C does indeed belong to class C. However, it does not tell us anything about the number of class C tuples that the classifier mislabeled. A perfect recall score of 1.0 for C means that every item from class C was labeled as such, but it does not tell us how many other tuples were incorrectly labeled as belonging to class C. There tends to be an inverse relationship between precision and recall, where it is possible to increase one at the cost of reducing the other. For example, our medical classifier may achieve high precision by labeling all cancer tuples that present a certain way as cancer, but may have low recall if it mislabels many other instances of cancer tuples. Precision and recall scores are typically used together, where precision values are compared for a fixed value of recall, or vice versa. For example, we may compare precision values at a recall value of, say, 0.75. An alternative way to use precision and recall is to combine them into a single measure. This is the approach of the F measure (also known as the F1 score or F-score) and the Fβ measure. They are defined as

$$F = 2 \times \text{precision} \times \text{recall} / \text{precision} + \text{recall}$$

$$F\beta = (1 + \beta\ 2\ ) \times \text{precision} \times \text{recall}/\ \beta\ 2 \times \text{precision} + \text{recall}$$

where β is a non-negative real number. The F measure is the harmonic mean of precision and recall (the proof of which is left as an exercise). It gives equal weight to precision and recall. The Fβ measure is a weighted measure of precision and recall. It assigns β times as much weight to recall as to precision. Commonly used Fβ measures are F2 (which weights recall twice as much as precision) and F0.5 (which weights precision twice as much as recall). “Are there other cases where accuracy may not be appropriate?” In classification problems, it is commonly assumed that all tuples are uniquely classifiable, that is, that each training tuple can belong to only one class. Yet, owing to the wide diversity of data in large databases, it is not always reasonable to assume that all tuples are uniquely classifiable. Rather, it is more probable to assume that each tuple may belong to more than one class. How then can the accuracy of classifiers on large databases be measured? The accuracy measure is not

appropriate, because it does not take into account the possibility of tuples belonging to more than one class.

- **Positive Predictive Value (PPV)**

Positive Predictive Value, also known as precision, is the probability that instances classified as positive are truly positive. It is calculated as the ratio of true positive predictions to the total number of instances classified as positive.

PPV=True PositivesTrue Positives+False Positives

$$\text{PPV}=\frac{True\ Positives+False\ Positives}{True\ Positives}$$

- **Negative Predictive Value (NPV)**

Negative Predictive Value measures the probability that instances classified as negative are truly negative. It is calculated as the ratio of true negative predictions to the total number of instances classified as negative.

NPV=True NegativesTrue Negatives+False Negatives

$$\text{NPV}=\frac{True\ Negatives+False\ Negatives}{True\ Negatives}$$

- **ROC**

ROC (Receiver Operating Characteristic) curve is a graphical representation used to evaluate the performance of a classification model at various thresholds. It plots the true positive rate (TPR) against the false positive rate (FPR) at different classification thresholds. The area under the ROC curve (AUC) is commonly used as a summary statistic to compare different models.

Let's consider an example to illustrate ROC:

Suppose we have a binary classification problem where we want to predict whether emails are spam (positive class) or not spam (negative class). We've built a machine learning model (let's say a logistic regression model) that outputs probabilities for each email being spam.

Here's a step-by-step guide to constructing the ROC curve:

I. **Train the Model:** Train your logistic regression model using a labeled dataset that includes features of emails and their corresponding labels (spam or not spam).

II. **Predict Probabilities:** Use the trained model to predict probabilities of each email being spam.

III. **Calculate TPR and FPR:** Sort the emails based on their predicted probabilities. Start with a threshold of 0, then gradually increase the threshold. For each threshold, classify emails with predicted probabilities above the threshold as spam and below as not spam. Calculate True Positive Rate (TPR) and False Positive Rate (FPR) at each threshold.

IV. **Plot ROC Curve:** Plot the TPR against the FPR at different thresholds. Each point on the curve represents a different threshold setting.

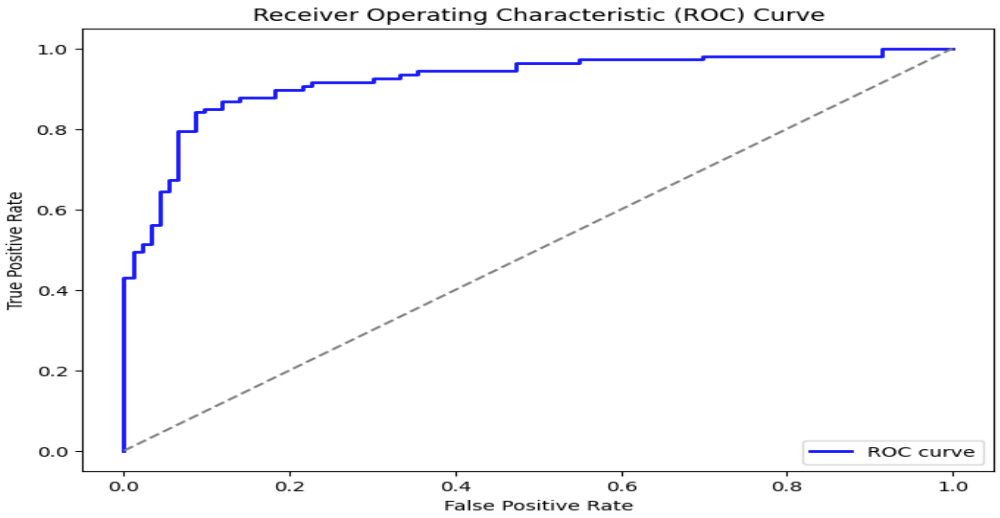
V. **Calculate AUC:** Compute the area under the ROC curve. A perfect classifier would have an AUC of 1, while a random classifier would have an AUC of 0.5.

Here's Python code to illustrate ROC curve with an example using scikit-learn:

```
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, roc_auc_score
import matplotlib.pyplot as plt

# Generate synthetic dataset
X, y = make_classification(n_samples=1000, n_features=20, n_classes=2,
random_state=42)

# Split dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Predict probabilities on the test set
probs = model.predict_proba(X_test)[:, 1]

# Calculate fpr, tpr, thresholds
fpr, tpr, thresholds = roc_curve(y_test, probs)
```

```
# Plot ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label='ROC curve')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()

# Calculate AUC
auc = roc_auc_score(y_test, probs)
print(f'AUC: {auc}')
```

**Output: ROC Curve Plot:** The code will generate a plot showing the ROC curve. It will have the True Positive Rate (TPR) on the y-axis and the False Positive Rate (FPR) on the x-axis. The curve will be plotted in blue, and the diagonal line (representing random guessing) will be plotted in gray dashed line.



**AUC Calculation:** The code will calculate the Area Under the ROC Curve (AUC) based on the predicted probabilities and the true labels of the test set. It will print out the AUC value.

```
AUC: 0.85
```

## b) Random Variable

A random variable is a variable whose value is unknown or a function that assigns values to each of an experiment's outcomes. Random variables are often designated by letters and can be classified as discrete, which are variables that have specific values, or continuous, which are variables that can have any values within a continuous range.

Random variables are often used in econometric or regression analysis to determine statistical relationships among one another.

The correlation coefficient and the covariance matrix are both statistical tools used to measure the relationship between variables in a dataset. Let's discuss each one:

- **Covariance Matrix**

The covariance matrix is a square matrix that summarizes the covariance between multiple variables in a dataset. . If you have *n* variables, the covariance matrix will be an $n \times n$ matrix where each element represents the covariance between two variables.

If X1,X2,…..Xn are n random variables, the covariance between Xi and Xj is

$$\text{Cov(Xi, Xj)} = \frac{1}{N}\sum_{k=1}^{N} (X_i^{(k)} - \underline{X_i})(X_j^{(k)} - \underline{X_j})$$

Where:

- $N$ is the number of observations.
- $X_i^{(k)}$ and $X_j^{(k)}$ are the $k$th observations of variables Xi and Xj respectively.

Python Code:

```
import numpy as np

# Sample dataset
data = np.array([[1, 2, 3],
```

```
                [4, 5, 6],
                [7, 8, 9],
                [10, 11, 12]])

# Calculate covariance matrix
cov_matrix = np.cov(data, rowvar=False)

print("Covariance Matrix:")
print(cov_matrix)
```

In this code:

- We import the NumPy library.
- We define a sample dataset data consisting of rows as observations and columns as variables/features.
- We then calculate the covariance matrix using np.cov() function, specifying rowvar=False to indicate that each column represents a variable/feature.
- Finally, we print out the covariance matrix.

- **Correlation Coefficient**

The correlation coefficient measures the strength and direction of the linear relationship between two variables. It is a standardized measure, meaning it's scale-invariant and ranges between -1 and 1.

The correlation coefficient between variables

$X$ and $Y$ is given by:

$$\text{Corr}(X,Y) = \frac{Cov(X,Y)}{\sigma_x \sigma_y}$$

- $\text{Cov}(X,Y)$ is the covariance between $X$ and Y
- $\sigma_x$ and $\sigma_y$ are the standard deviations of X and Y respectively.

**Python Code:**

```
import numpy as np

# Sample dataset
data = np.array([[1, 2, 3],
                 [4, 5, 6],
                 [7, 8, 9],
                 [10, 11, 12]])

# Calculate correlation coefficient matrix
correlation_matrix = np.corrcoef(data, rowvar=False)

print("Correlation Coefficient Matrix:")
print(correlation_matrix)
```

In this code:

- We import the NumPy library.
- We define a sample dataset consisting of rows as observations and columns as variables/features.
- We then calculate the correlation coefficient matrix using np.corrcoef() function, specifying rowvar=False to indicate that each column represents a variable/feature.
- Finally, we print out the correlation coefficient matrix.

**Interpretation:**

- If the correlation coefficient is close to 1, it indicates a strong positive linear relationship between the variables.
- If the correlation coefficient is close to -1, it indicates a strong negative linear relationship between the variables.
- If the correlation coefficient is close to 0, it indicates little to no linear relationship between the variables.

**Use Cases:**

- Covariance matrices are often used in multivariate analysis and machine learning algorithms such as principal component analysis (PCA) and linear discriminant analysis (LDA).

▪ Correlation coefficients are widely used in various fields to understand relationships between variables, such as finance, economics, and social sciences.

▪ **P-Value**

In statistics, particularly in hypothesis testing, the p-value (short for "probability value") is a measure that helps determine the strength of evidence against the null hypothesis. The null hypothesis is a statement or assumption that there is no significant difference or effect.

Here's a basic overview of the p-value:

**Definition**: The p-value is the probability of obtaining test results at least as extreme as the observed results, assuming that the null hypothesis is true.

**Interpretation:**

▪ If the p-value is small (typically below a predetermined significance level, often denoted as $\alpha$, commonly set at 0.05), it suggests that the observed results are unlikely to have occurred under the assumption of the null hypothesis. In this case, we may reject the null hypothesis in favor of an alternative hypothesis.

▪ If the p-value is large, it indicates that the observed results are reasonably likely to occur even if the null hypothesis is true. In such cases, we fail to reject the null hypothesis.

## 3. Hypothesis Testing

Hypothesis testing is a statistical method used to make inferences about a population based on sample data. It involves formulating a null hypothesis (*H0*) and an alternative hypothesis (*H*1), collecting data, and using statistical tests to determine whether there is enough evidence to reject the null hypothesis in favor of the alternative hypothesis.

The steps involved in hypothesis testing typically include:

- Formulating the null and alternative hypotheses.
- Choosing an appropriate statistical test based on the type of data and research question.
- Collecting sample data.
- Calculating a test statistic and determining its probability under the null hypothesis (p-value).
- Comparing the p-value to a predetermined significance level (α) to make a decision about the null hypothesis.

### a) A/B Testing

A/B testing, also known as split testing, is a method used in marketing, web development, and product management to compare two versions of a webpage, advertisement, or product feature to determine which one performs better. It involves dividing users or participants into two groups (A and B), showing each group a different version (A and B) of the item being tested, and then measuring and comparing their responses.

The steps involved in A/B testing typically include:

- Identifying the goal or metric to be optimized (e.g., conversion rate, click-through rate).
- Creating two versions (A and B) of the item being tested, with one differing element (e.g., color, layout, wording).
- Randomly assigning users or participants to the two groups.
- Running the experiment and collecting data on the chosen metric for each group.
- Analyzing the results using statistical methods (e.g., hypothesis testing) to determine if there is a statistically significant difference between the two versions.

### b) Monte Carlo Simulation

Monte Carlo simulation is a computational technique used to model the behavior of complex systems or processes through repeated random sampling. It involves generating a large number of random samples from probability distributions that represent uncertain inputs or parameters of the system, running simulations using these samples, and then analyzing the results to estimate outcomes or evaluate risk.

The steps involved in Monte Carlo simulation typically include:

- Identifying the system or process to be modeled and defining its inputs, outputs, and parameters.
- Specifying probability distributions for the uncertain inputs or parameters.
- Generating random samples from these distributions using a pseudo-random number generator.
- Running simulations of the system using the sampled values and recording the outcomes of interest.
- Analyzing the simulation results to estimate probabilities, expected values, or other relevant statistics.

Monte Carlo simulation is widely used in finance, engineering, operations research, and many other fields to tackle problems involving uncertainty, variability, and risk. It provides valuable insights into the behavior of complex systems and helps in decision-making under uncertainty.

## Self-Check Sheet - 1: Apply Statistical Measures

Q1: What is the mean of a dataset?

Q2: What are the limitations of using the mean to represent data?

Q3: What is the median of a dataset?

Q4: How do you calculate the median for the dataset [3, 8, 12, 16, 20]?

Q5: Find the mode of the dataset [3, 3, 6, 9, 9, 9, 15, 18].

Q6: How is standard deviation related to variance?

Q7: What does a high standard deviation indicate about a dataset?

Q8: What is a random variable?

Q9: What is the correlation coefficient?

Q10: What is a Poisson distribution?

Q11: What is a p-value in hypothesis testing?

Q12: What is a confusion matrix?

Q13: How does A/B testing work?

Q14: Can you give an example of a successful A/B test?

Q15: How does Monte Carlo testing work?

## Answer Sheet - 1: Apply Statistical Measures

**Q1: What is the mean of a dataset?**

**A**: The mean, or average, is the sum of all values in the dataset divided by the number of values. It represents the central point of a data distribution.

**Q2: What are the limitations of using the mean to represent data?**

**A:** The mean is sensitive to outliers. Extreme values can skew the mean and may not represent the data accurately in such cases.

**Q3: What is the median of a dataset?**

**A:** The median is the middle value of a dataset when the numbers are arranged in ascending or descending order. If the dataset has an even number of values, the median is the average of the two middle numbers.

**Q4: How do you calculate the median for the dataset [3, 8, 12, 16, 20]?**
**A:** Since there are 5 numbers, the middle value (third number) is the median:

Median=12

**Q5: Find the mode of the dataset [3, 3, 6, 9, 9, 9, 15, 18].**

**A:** The mode is the most frequent value, which is 9.

**Q6: How is standard deviation related to variance?**

**A:** Standard deviation is the square root of the variance. While variance measures the spread of data, the standard deviation is easier to interpret since it is in the same unit as the original data.

**Q7: What does a high standard deviation indicate about a dataset?**

**A:** A high standard deviation indicates that the data points are spread out over a wider range, while a low standard deviation means the data points are closer to the mean.

**Q8: What is a random variable?**

**A:** A random variable is a variable that takes on different values based on the outcome of a random event. It can be discrete (taking specific values, like rolling a die) or continuous (taking any value within a range, like measuring temperature).

**Q9: What is the correlation coefficient?**

**A:** The correlation coefficient (usually represented as rrr) measures the strength and direction of a linear relationship between two variables. Its value ranges from -1 to +1:

- r =+1: Perfect positive correlation
- r=−1: Perfect negative correlation
- r=0: No linear correlation

**Q10: What is a Poisson distribution?**

**A:** A Poisson distribution models the number of events occurring within a fixed interval of time or space when these events happen at a constant rate and independently of each other. It is often used to model rare events, such as the number of earthquakes in a year.

**Q11: What is a p-value in hypothesis testing?**

**A:** The p-value represents the probability of obtaining results as extreme as those observed, assuming the null hypothesis is true. A small p-value (usually less than 0.05) suggests strong evidence against the null hypothesis, indicating the results are statistically significant.

**Q12: What is a confusion matrix?**

**A:** A confusion matrix is a table that summarizes the performance of a classification model by showing the true positives, false positives, true negatives, and false negatives. It helps in calculating performance metrics like accuracy, precision, recall, and F1-score.

**Q13: How does A/B testing work?**
**A:** A/B testing involves these steps:

1. **Hypothesis**: Start with a hypothesis about how a change might improve a metric (e.g., click-through rate).
2. **Create Variants**: Develop two versions (A and B), where A is the original and B has a single variation.
3. **Split Traffic**: Randomly assign half of the users to version A and half to version B.
4. **Measure Results**: Collect data on user interactions, like clicks, conversions, or other KPIs.
5. **Analyze**: Use statistical analysis to determine if version B performs significantly better than version A.

**Q14: Can you give an example of a successful A/B test?**

**A:** A company might run an A/B test on their homepage call-to-action button. The control version (A) uses "Get Started" text, while the variation (B) uses "Start Free Trial." After running the test for two weeks, version B results in a 15% higher conversion rate, leading the company to adopt the new text.

**Q16: How does Monte Carlo testing work?**
**A:** Monte Carlo testing works by:

1. Defining a model or system with uncertain parameters.
2. Generating random inputs based on probability distributions of the uncertain parameters.
3. Running simulations or trials using those inputs.
4. Collecting and analyzing the outcomes to estimate the likelihood of different scenarios or to identify potential risks.

## Task Sheet-1.1: Apply Statistical measures.

**Step**

1. **Understand the Problem Statement**
    - Familiarize with the type of data you will be working with (e.g., image data, sensor data, user interaction data).
    - Identify which statistical measures are most relevant to your analysis (mean, median, mode, variance, etc.).
2. **Data Collection and Preparation**
    - Collect a relevant dataset for analysis (can be real-world data or a pre-existing dataset).
    - Clean and preprocess the data (handle missing values, normalize data, and remove outliers if needed).
3. **Apply Statistical Measures**
    - **Mean, Median, and Mode**: Calculate these measures to understand the central tendency of your dataset.
    - **Variance and Standard Deviation**: Compute the spread of the data points from the mean.
    - **Correlation**: Check how two variables in the dataset relate to one another.
    - **Regression Analysis**: Perform regression to predict one variable based on another (e.g., predicting user engagement based on interaction time).
4. **Data Interpretation and Reporting**
    - Analyze the results and interpret the statistical findings in the context of your machine learning application.
    - Prepare a report summarizing your findings and how they can inform the development of AI-driven immersive technologies.

5. **Review and Validate**

    - Cross-check calculations for accuracy.

    - Validate findings through visualization (graphs or charts) and ensure the statistical measures align with expected outcomes.

6. **Prepare and Present Results**

    - Use data visualization tools like Matplotlib or Seaborn in Python to visualize data distributions.

    - Present the findings in a clear, concise manner for stakeholders or clients.

**Task Sheet-1.2: Measures of Central Tendency: a. Calculate the mean, median, and mode for the following dataset: [10, 15, 20, 25, 30, 35, 40, 45, 50]. b. Compute the standard deviation and variance for the same dataset.**

**Step**

1. **Understand the Problem Statement**
    - You will calculate the **mean**, **median**, **mode**, **variance**, and **standard deviation** for the provided dataset.
    - These measures will help you understand the central tendency (where most data points lie) and the variability (how spread out the data points are).
2. **Step 1: Calculate the Mean**
    - The **mean** is the average of all data points.
    - Formula: Mean=∑Data PointsN\text{Mean} = \frac{\sum \text{Data Points}}{N}Mean=N∑Data Points
    - Where ∑\sum∑ is the sum of all the data points and NNN is the number of data points.
3. **Step 2: Calculate the Median**
    - The **median** is the middle value when the data is arranged in ascending order.
    - If there is an odd number of data points, the median is the middle value. If the number is even, the median is the average of the two middle numbers.
4. **Step 3: Calculate the Mode**
    - The **mode** is the value that appears most frequently in the dataset.
    - If no number repeats, the dataset is said to have no mode.
5. **Step 4: Calculate the Variance**
    - **Variance** measures how far each data point is from the mean.

- Formula: Variance=∑(xi−μ)2N\text{Variance} = \frac{\sum (x_i - \mu)^2}{N}Variance=N∑(xi−μ)2

- Where xix_ixi is each data point, μ\muμ is the mean, and NNN is the number of data points.

6. **Step 5: Calculate the Standard Deviation**

   - **Standard deviation** is the square root of the variance and provides a sense of the spread of the data points.

   - Formula: Standard Deviation=Variance\text{Standard Deviation} = \sqrt{\text{Variance}}Standard Deviation=Variance

7. **Data Interpretation and Reporting**

   - After performing these calculations, analyze the results. What does the mean tell you about the "center" of the data? How spread out is the dataset, based on the variance and standard deviation?

   - Summarize your findings in a report.

8. **Review and Validate**

   - Double-check the calculations for accuracy.

   - You may use tools like Excel or a scientific calculator to verify your results.

9. **Prepare and Present Results**

   - Present your results clearly, showing how each of the statistical measures was derived.

   - Use graphs like histograms or boxplots to visualize the data distribution.

**Task Sheet-1.3: Random Variables and Probability Distribution: a. Define a random variable and explain its significance in probability theory. b. Describe the concept of a probability distribution. Provide examples of discrete and continuous probability distributions. c. Calculate the correlation coefficient and covariance matrix for the following dataset: X = [2, 4, 6, 8, 10] Y = [1, 3, 5, 7, 9]**

**Step**

**1. Define and Explain Random Variables**

- A **random variable** is a variable that can take on different values based on the outcome of a random event or experiment.
- **Types of Random Variables**:
    - **Discrete Random Variable**: Takes on a finite number of distinct values (e.g., the number of heads in a coin toss).
    - **Continuous Random Variable**: Takes on an infinite number of values within a given range (e.g., the height of a person).
- **Significance**: Random variables are used to quantify uncertainty and model real-world processes in probability theory, which is foundational for predicting outcomes and making decisions in machine learning applications.

**2. Describe the Concept of a Probability Distribution**

- A **probability distribution** is a function that describes the likelihood of different outcomes for a random variable.
    - **Discrete Probability Distribution**: It gives the probability of each possible value for a discrete random variable. Example: The probability distribution of a dice roll (1, 2, 3, 4, 5, or 6).
    - **Continuous Probability Distribution**: It assigns probabilities to ranges of values rather than specific outcomes. Example: The normal distribution of human heights.
- **Examples**:
    - **Discrete**: Binomial distribution, Poisson distribution
    - **Continuous**: Normal distribution, Exponential distribution

**3. Calculate the Correlation Coefficient and Covariance Matrix**

Given the dataset:

- X=[2,4,6,8,10]X = [2, 4, 6, 8, 10]X=[2,4,6,8,10]
- Y=[1,3,5,7,9]Y = [1, 3, 5, 7, 9]Y=[1,3,5,7,9]

**Step 1: Calculate the Mean of X and Y**

- Mean of XXX:

$$\mu_X = \frac{2+4+6+8+10}{5} = 6$$

- Mean of YYY

$$\mu_Y = \frac{1+3+5+7+9}{5} = 5$$

**Step 2: Calculate the Covariance**

- **Covariance** measures how two variables change together.

$$\text{Cov}(X,Y) = \frac{1}{N}\sum_{i=1}^{N}(X_i - \mu_X)(Y_i - \mu_Y)$$

- Substituting values:

$$\text{Cov}(X,Y) = \frac{(2-6)(1-5) + (4-6)(3-5) + (6-6)(5-5) + (8-6)(7-5) + (10-6)(9-5)}{5}$$

$$\text{Cov}(X,Y) = \frac{(-4)(-4) + (-2)(-2) + (0)(0) + (2)(2) + (4)(4)}{5} = \frac{16+4+0+4+16}{5} = \frac{40}{5} = 8$$

**Step 3: Calculate the Correlation Coefficient**

- **Correlation coefficient** measures the strength and direction of the linear relationship between two variables.
- Formula:

$$r = \frac{\text{Cov}(X,Y)}{\sigma_X \sigma_Y}$$

Where σX\sigma_XσX and σY\sigma_YσY are the standard deviations of XXX and YYY, respectively.

**Calculate Standard Deviations**:

$$\sigma_X = \sqrt{\frac{(2-6)^2 + (4-6)^2 + (6-6)^2 + (8-6)^2 + (10-6)^2}{5}} = \sqrt{\frac{16+4+0+4+16}{5}} = \sqrt{8} = 2.828$$

$$\sigma_Y = \sqrt{\frac{(1-5)^2 + (3-5)^2 + (5-5)^2 + (7-5)^2 + (9-5)^2}{5}} = \sqrt{\frac{16+4+0+4+16}{5}} = \sqrt{8} = 2.828$$

Now, calculate the correlation:

$$r = \frac{8}{2.828 \times 2.828} = \frac{8}{8} = 1$$

**Step 4: Covariance Matrix**

- The covariance matrix is a square matrix showing the covariance between each pair of variables in the dataset.
- Since we only have two variables, XXX and YYY, the covariance matrix is:

$$\text{Covariance Matrix} = \begin{bmatrix} \text{Cov}(X,X) & \text{Cov}(X,Y) \\ \text{Cov}(Y,X) & \text{Cov}(Y,Y) \end{bmatrix}$$

Where:

$$\text{Cov}(X,X) = \sigma_X^2 = 8$$

$$\text{Cov}(Y,Y) = \sigma_Y^2 = 8$$

Thus, the covariance matrix is:

$$\begin{bmatrix} 8 & 8 \\ 8 & 8 \end{bmatrix}$$

**Task Sheet-1.4: Probability Distributions and p-value: a. Explain the characteristics and applications of the Binomial, Poisson, and Normal probability distributions. b. Given a Binomial distribution with n=10 and p=0.3, calculate the probability of getting exactly 3 successes. c. Define the p-value and explain its significance in hypothesis testing.**

**Step**

**1. Explain the Characteristics and Applications of the Binomial, Poisson, and Normal Probability Distributions**

- **Binomial Distribution:**
  - **Characteristics:**
    - Describes the number of successes in a fixed number of independent trials.
    - Each trial has two possible outcomes: success or failure.
    - Probability of success remains constant across trials.
    - Parameters: nnn (number of trials) and ppp (probability of success in each trial).
  - **Applications:**
    - Used in situations where the outcome of each trial is binary (e.g., coin tosses, success/failure in experiments).
- **Poisson Distribution:**
  - **Characteristics:**
    - Describes the number of events occurring in a fixed interval of time or space.
    - Events occur independently and at a constant average rate.
    - Parameters: λ\lambdaλ (average number of events in the interval).
  - **Applications:**
    - Used for modeling rare events, such as the number of calls at a call center, the number of accidents at an intersection, or the number of decay events in radioactive material.
- **Normal Distribution:**
  - **Characteristics:**
    - A symmetric, bell-shaped curve.
    - Describes data that clusters around a central value, with the distribution tapering off symmetrically towards both ends.
    - Parameters: mean (μ\muμ) and standard deviation (σ\sigmaσ).
  - **Applications:**
    - Used in situations where the data is expected to follow a natural pattern (e.g., height, weight, test scores, measurement errors).

**2. Calculate the Probability of Getting Exactly 3 Successes in a Binomial Distribution**
Given:

- Number of trials n=10n = 10n=10
- Probability of success p=0.3p = 0.3p=0.3
- We need to find the probability of getting exactly 3 successes.

**Step 1: Understand the Binomial Distribution Formula**
The probability of getting exactly kkk successes in nnn trials is given by the formula:

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

Where:

- $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ is the binomial coefficient,
- ppp is the probability of success,
- nnn is the number of trials,
- kkk is the number of successes.

**Step 2: Calculate the Probability for k=3k = 3k=3**

$$P(X = 3) = \binom{10}{3} (0.3)^3 (0.7)^{10-3}$$

First, calculate the binomial coefficient:

$$\binom{10}{3} = \frac{10!}{3!(10-3)!} = \frac{10 \times 9 \times 8}{3 \times 2 \times 1} = 120$$

Now calculate the probability:

$$P(X = 3) = 120 \times (0.3)^3 \times (0.7)^7 = 120 \times 0.027 \times 0.0823543 \approx 0.267$$

Thus, the probability of getting exactly 3 successes is approximately **0.267** or **26.7%**.

**3. Define the p-value and Explain Its Significance in Hypothesis Testing**

- **p-value:**
  - The **p-value** is the probability of obtaining a test statistic at least as extreme as the one observed, under the assumption that the null hypothesis is true.
  - A **low p-value** (typically p≤0.05p \leq 0.05p≤0.05) indicates strong evidence against the null hypothesis, suggesting it should be rejected.
  - A **high p-value** (typically p>0.05p > 0.05p>0.05) suggests weak evidence against the null hypothesis, meaning we fail to reject it.

**Significance in Hypothesis Testing:**

- In hypothesis testing, the p-value helps determine whether the observed data provides sufficient evidence to reject the null hypothesis.
- **Steps in Hypothesis Testing:**
  1. Formulate the null hypothesis (H0H_0H0) and alternative hypothesis (H1H_1H1).
  2. Choose a significance level (α\alphaα), commonly 0.05.
  3. Compute the p-value.
  4. If the p-value is less than α\alphaα, reject H0H_0H0; otherwise, fail to reject H0H_0H0.
- For example, in machine learning, p-values are often used to assess the significance of features or model parameters in statistical tests (e.g., t-tests, chi-square tests).

**Task Sheet-1.5: Bayes' Theorem and Performance Metrics: a. Describe Bayes' Theorem and explain how it is used in probabilistic reasoning. b. Define Precision, Recall, Positive Predictive Value (PPV), and Negative Predictive Value (NPV). Discuss their roles in evaluating classification models. c. Explain the concepts of Confusion Matrix and ROC Curve in evaluating the performance of classification models.**

**Step**

**1. Describe Bayes' Theorem and Explain How It Is Used in Probabilistic Reasoning**

- **Bayes' Theorem** is a method in probability theory that describes the probability of an event, based on prior knowledge of conditions related to the event. It provides a way of updating the probability of a hypothesis as more evidence becomes available.

**Formula:**

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Where:

- $P(A|B)$ is the **posterior** probability (probability of $A$ given $B$),
- $P(B|A)$ is the **likelihood** (probability of $B$ given $A$),
- $P(A)$ is the **prior** probability (initial belief about $A$),
- $P(B)$ is the **marginal likelihood** (probability of observing $B$).

- **Application in Probabilistic Reasoning:**
  - Bayes' Theorem is frequently used in machine learning, particularly in **Naive Bayes classifiers** and **decision theory**, to update beliefs about the data given new evidence.
  - For example, in spam filtering, Bayes' Theorem helps update the probability that an email is spam based on the words it contains.

**2. Define Precision, Recall, Positive Predictive Value (PPV), and Negative Predictive Value (NPV). Discuss Their Roles in Evaluating Classification Models**

- **Precision**:
  - **Definition**: Precision is the proportion of true positive predictions out of all positive predictions made by the model.
  - **Formula**:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Where:

- $TP$ = True Positives
- $FP$ = False Positives

  - **Role**: Precision is important when the cost of false positives is high. For example, in medical diagnostics, if a model falsely predicts a healthy person as sick (false positive), it could lead to unnecessary treatments.
- **Recall**:
  - **Definition**: Recall, also known as Sensitivity or True Positive Rate, is the proportion of actual positives correctly identified by the model.
  - **Formula**:

$$\text{Recall} = \frac{TP}{TP + FN}$$

Where:

- $FN$ = False Negatives

  - **Role**: Recall is crucial when false negatives are costly. For instance, in fraud detection, missing fraudulent transactions (false negatives) is highly undesirable.
- **Positive Predictive Value (PPV)**:
  - **Definition**: PPV is the proportion of positive predictions that are actually correct.

$$\text{PPV} = \frac{TP}{TP + FP}$$

  - **Formula**:
  - **Role**: PPV tells you how reliable the positive predictions are, useful in evaluating the quality of predictions when a model classifies an instance as positive.
- **Negative Predictive Value (NPV)**:
  - **Definition**: NPV is the proportion of negative predictions that are actually correct.
  - **Formula**:

$$\text{NPV} = \frac{TN}{TN + FN}$$

Where:

- $TN$ = True Negatives
- $FN$ = False Negatives

  - **Role**: NPV is important in scenarios where a high proportion of the predicted negatives should be reliable, such as ruling out a disease in a medical test.

**Overall Role in Evaluation**:

- These metrics are critical in classification tasks to balance performance based on different error types (false positives and false negatives). Depending on the application, you may prioritize one over the other.

**3. Explain the Concepts of Confusion Matrix and ROC Curve in Evaluating the Performance of Classification Models**

- **Confusion Matrix**:
  - A **Confusion Matrix** is a table used to evaluate the performance of a classification algorithm. It summarizes the results of predictions and compares them to actual values.
  - The matrix consists of four key components:
    - **True Positives (TP)**: Correctly predicted positive cases.
    - **False Positives (FP)**: Incorrectly predicted positive cases (type I error).
    - **True Negatives (TN)**: Correctly predicted negative cases.
    - **False Negatives (FN)**: Incorrectly predicted negative cases (type II error).

**Example of a Confusion Matrix**:

| | Predicted Positive | Predicted Negative |
|---|---|---|
| Actual Positive | $TP$ | $FN$ |
| Actual Negative | $FP$ | $TN$ |

  - **Role**: The confusion matrix is used to calculate important metrics like precision, recall, accuracy, and F1-score. It provides insights into the types of errors the model is making.

- **ROC Curve (Receiver Operating Characteristic Curve)**:
  - An **ROC Curve** is a graphical representation of the performance of a binary classification model. It plots the True Positive Rate (Recall) against the False Positive Rate (FPR), which is FPFP+TN\frac{FP}{FP + TN}FP+TNFP.
  - **Area Under the Curve (AUC)**:
    - AUC represents the degree of separability the model has in distinguishing between classes. A higher AUC indicates a better model.
    - **Role**: The ROC curve helps in comparing different classification models, particularly when dealing with imbalanced datasets. A model with a higher AUC is generally preferred.

**Task Sheet-1.6: Hypothesis Testing: a. Define A/B Testing and explain its application in statistical hypothesis testing. b. Design an A/B test scenario for a website aiming to improve user engagement. Outline the hypothesis, experimental setup, and metrics for evaluation. c. Explain the concept of Monte Carlo Simulation and discuss its advantages in solving complex probabilistic problems.**

**Step**

**1. Define A/B Testing and Explain Its Application in Statistical Hypothesis Testing**

- **A/B Testing**:
  - **Definition**: A/B testing (also known as split testing) is a controlled experiment comparing two versions (A and B) of a treatment or intervention to determine which performs better based on a given metric. It is widely used in areas like marketing, product design, and user experience (UX) optimization.
  - **Application in Statistical Hypothesis Testing**:
    - **Null Hypothesis ($H_0$)**: There is no significant difference between the two versions (A and B).
    - **Alternative Hypothesis ($H_1$)**: There is a significant difference between the two versions (A and B).
    - A/B testing helps to test the effectiveness of changes (like a new website design, marketing strategy, etc.) by comparing outcomes and deciding if the new version improves performance.
    - **Example**: Testing whether changing the color of a "Buy Now" button (from blue to green) increases the conversion rate on an e-commerce website.

**2. Design an A/B Test Scenario for a Website Aiming to Improve User Engagement**

- **Scenario**: A website wants to improve user engagement by changing the layout of its homepage. Version A (current layout) is compared to version B (new layout).
- **Step-by-Step Design**:
  - **Hypothesis**:
    - **Null Hypothesis ($H_0$)**: There is no significant difference in user engagement between the current homepage layout (A) and the new homepage layout (B).
    - **Alternative Hypothesis ($H_1$)**: The new homepage layout (B) significantly improves user engagement compared to the current layout (A).
  - **Experimental Setup**:
    - **Random Sampling**: Randomly assign website visitors to either version A or version B, ensuring the sample is large enough to yield statistically significant results.
    - **Randomized Control**: Both groups (A and B) should experience the website under similar conditions (e.g., time of day, similar demographic characteristics, etc.) to reduce bias.

        - **Duration**: Run the test for a sufficient period (e.g., 2 weeks) to account for variability and gather enough data.
        - **Test Group**: Half of the visitors will see the current homepage layout (A).
        - **Control Group**: The other half will see the new homepage layout (B).
    - **Metrics for Evaluation**:
        - **Primary Metric**: User engagement, which can be measured by the average time spent on the site or the number of pages viewed per session.
        - **Secondary Metrics**: Bounce rate (the percentage of visitors who leave the site after viewing only one page), conversion rate (the percentage of visitors who make a purchase or sign up).
        - **Statistical Analysis**: Perform statistical tests (e.g., t-test) to evaluate whether the differences in metrics between groups A and B are statistically significant.

**3. Explain the Concept of Monte Carlo Simulation and Discuss Its Advantages in Solving Complex Probabilistic Problems**

- **Monte Carlo Simulation**:
    - **Definition**: Monte Carlo simulation is a computational technique that uses random sampling to obtain numerical results for problems that might be deterministic in principle but are complex and difficult to solve directly.
    - **Process**:
        - Simulate a process by generating random variables within a specified range.
        - Run multiple simulations (often thousands or millions) to model the probability distribution of the output.
        - Analyze the results to estimate the behavior of the system and make predictions about the likelihood of different outcomes.
- **Advantages**:
    - **Handling Complex Problems**: Monte Carlo methods are particularly useful in situations where an analytical solution is difficult or impossible to obtain, especially in high-dimensional problems.
    - **Flexibility**: Can be applied to a wide range of problems (financial modeling, physics, engineering, AI, etc.).
    - **Risk Analysis**: Monte Carlo simulations are widely used in finance, risk management, and decision-making to understand the impact of uncertainty and variability on outcomes.
    - **Real-World Applications**: Common in estimating the probability of different scenarios in situations with uncertainty, such as financial forecasting, project management, and optimizing machine learning models.

## Learning Outcome 2: Use Multivariable Calculus

| | |
|---|---|
| Assessment Criteria | 1. Matrices are exercised<br>2. Vectors are exercised |
| Conditions and Resources | 1. Actual workplace or training environment.<br>2. CBLM<br>3. 3.Handouts<br>4. Laptop<br>5. Multimedia Projector<br>6. Paper, Pen, Pencil and Eraser<br>7. Internet Facilities<br>8. Whiteboard and Marker<br>9. Imaging Device (Digital camera, scanner etc.) |
| Contents | • Matrices<br>  o Transpose of a matrix<br>  o The inverse of a matrix<br>  o The determinant of a matrix<br>• Vectors<br>  o Dot product<br>  o Eigenvalues<br>  o Eigenvectors |
| Activities/job/Task | Use Multivariable Calculus<br>In this assignment, you will explore concepts related to multivariable functions, derivatives, activation functions, cost functions, and the plotting of functions. Please answer all questions thoroughly and provide explanations where necessary.<br><br>1. Multivariable Functions:<br>a. Define a multivariable function and explain its significance in mathematics and data analysis.<br>b. Consider the function f(x, y) = x^2 + 2xy + y^2. Calculate the partial derivatives ∂f/∂x and ∂f/∂y.<br><br>2. Derivatives and Gradients:<br>a. Define a derivative and explain its interpretation in calculus.<br>b. Compute the gradient of the function f(x) = 3x^2 - 2x + 1.<br><br>3. Activation Functions:<br>a. Explain the purpose of activation functions in neural networks.<br>b. Define and plot the following activation functions:<br>i. Step function<br>ii. Sigmoid function<br>iii. Logit function<br>iv. ReLU (Rectified Linear Unit) function |

| | |
|---|---|
| | 4. Cost Function:<br>a. Define a cost function and discuss its role in machine learning.<br>b. Consider a linear regression model with the hypothesis function $h(x) = \theta_0 + \theta_1 x$. Define the cost function $J(\theta)$ for this model.<br><br>5. Plotting of Functions:<br>a. Using Python or any other preferred tool, plot the function f(x) = x^3 - 2x^2 + 3x - 4 in the range [-2, 3].<br>b. Label the axes and provide a title for the plot.<br><br>6. Minimum and Maximum Values of a Function:<br>a. Define local minimum, local maximum, global minimum, and global maximum of a function.<br>1. b. Determine the local and global minimum and maximum values, if any, for the function f(x) = x^4 - 4x^3 + 6x^2. |
| Training Methods | 1. Discussion<br>2. Presentation<br>3. Demonstration<br>4. Guided Practice<br>5. Individual Practice<br>6. Project Work<br>7. Problem Solving<br>8. Brainstorming |
| Assessment Methods | 1. Written Test<br>2. Demonstration<br>3. Oral Questioning |

## Learning Experience 2: Use Multivariable Calculus

In order to achieve the objectives stated in this learning guide, you must perform the learning steps below. Beside each step are the resources or special instructions you will use to accomplish the corresponding activity.

| Learning Activities | Recourses/Special Instructions |
|---|---|
| 1. Trainee will ask the instructor about the learning materials | 1. Instructor will provide the learning materials 'Apply Statistical measures' |
| 2. Read the Information sheet and complete the Self Checks & Check answer sheets on "Apply Statistical measures" | 2. Read Information sheet 2: Use Multivariable Calculus<br>3. Answer Self-check 2: Use Multivariable Calculus<br>4. Check your answer with Answer key 2: Use Multivariable Calculus |
| 3. Read the Job/Task Sheet and Specification Sheet and perform job/Task | 5. Job/Task Sheet and Specification Sheet<br>Task Sheet-2.1: Multivariable Functions: a. Define a multivariable function and explain its significance in mathematics and data analysis. b. Consider the function f(x, y) = x^2 + 2xy + y^2. Calculate the partial derivatives ∂f/∂x and ∂f/∂y.<br>Task Sheet-2.2: Derivatives and Gradients: a. Define a derivative and explain its interpretation in calculus. b. Compute the gradient of the function f(x) = 3x^2 - 2x + 1.<br>Task Sheet-2.3: Activation Functions: a. Explain the purpose of activation functions in neural networks. b. Define and plot the following activation functions: i. Step function ii. Sigmoid function iii. Logit function iv. ReLU (Rectified Linear Unit) function<br>Task Sheet-2.4: Cost Function: a. Define a cost function and discuss its role in machine learning. b. Consider a linear regression model with the hypothesis function $h(x) = \theta_0 + \theta_1 x$. Define the cost function $J(\theta)$ for this model. |

| | |
|---|---|
| | Task Sheet-2.5: Plotting of Functions: a. Using Python or any other preferred tool, plot the function f(x) = x^3 - 2x^2 + 3x - 4 in the range [-2, 3]. b. Label the axes and provide a title for the plot<br>Task Sheet-2.6: Minimum and Maximum Values of a Function: a. Define local minimum, local maximum, global minimum, and global maximum of a function. b. Determine the local and global minimum and maximum values, if any, for the function f(x) = x^4 - 4x^3 + 6x^2 |

## Information Sheet 2: Use Multivariable Calculus

**Learning Objective:**

After completion of this information sheet, the learners will be able to explain, define and interpret the following contents:

1. Multivariable Functions
2. Derivatives and gradients
3. Activation Functions
4. Cost function
5. Function plotting
6. Minimum and maximum values of a function

## 1. Multivariable Functions

Multivariable functions, also known as multivariate functions, are functions that depend on more than one variable. These functions can take multiple inputs and produce a single output. They are commonly encountered in mathematics, physics, engineering, economics, and other fields where phenomena depend on multiple factors. Here's an example of a multivariable function:

Example: Multivariable Function in Economics

Consider a production function in economics that describes the output (quantity produced) of a firm as a function of two inputs: labor ($L$) and capital ($K$). The Cobb-Douglas production function is a classic example:

$$Q=f(L,K)=A.L^{\alpha}.K^{\beta}$$

Where:

- $Q$ is the quantity produced.
- $L$ is the quantity of labor.
- $K$ is the quantity of capital.
- $A$ is the total factor productivity or technological level.
- $\alpha$ and $\beta$ are the output elasticities of labor and capital, respectively.

In this example, $Q$ is a multivariable function of both $L$ and $K$. The output quantity depends on both the amount of labor and the amount of capital employed in the production process. The

parameters $A$, $\alpha$, and $\beta$ determine the shape and characteristics of the production function, reflecting factors such as technological progress, returns to scale, and factor substitution possibilities.

## 2. Derivatives and Gradients

Derivatives and gradients are fundamental concepts in calculus and vector calculus, respectively, used to describe how functions change with respect to their input variables. Here, I'll provide a brief overview of derivatives and gradients, followed by an exercise to practice computing them.

### i. Derivatives

Derivatives measure the rate of change of a function with respect to its input variable. For a function $f(x)$, the derivative $f'(x)$ (or $\frac{dx}{df}$) represents how $f(x)$ changes as $x$ changes.If $f(x)$ represents a single-variable function, the derivative $f'(x)$ can be computed using the limit definition of the derivative:

$$f'(x)=\lim_{h\to 0} \frac{h}{f(x+h)-f(x)}$$

### ii. Gradients

Gradients extend the concept of derivatives to multivariable functions. For a multivariable function $f(\mathbf{x})$, the gradient

$\nabla f(\mathbf{x})$ (pronounced "nabla f of x") is a vector that points in the direction of the steepest increase of the function at the point $\mathbf{x}$, and its magnitude represents the rate of change of $f$ in that direction.

**Exercise: Compute Derivatives and Gradients**

Let's consider the following functions:

- $f(x)=x^2$
- $g(x,y)= x^2 + y^2$

For each function, compute:

- The derivative of $f(x)$ with respect to $x$.
- The gradient of $g(x,y)$ with respect to $x$ and $y$.

**Solutions:**

- **For f(x)=$X^2$**
  - ➢ The derivative of $f(x)$ with respect to $x$ is $f'(x)=2x$.
- **For $g(x,y)=x^2+y^2$**
  - ➢ The gradient of $g(x,y)$ with respect to $x$ is: $\frac{\delta g}{\delta x} = 2x$
  - ➢ The gradient of$g(x,y)$ with respect to $y$ is: $\frac{\delta g}{\delta y} = 2y$

These are the basic computations for derivatives and gradients. Practicing with more complex functions and understanding their applications in optimization, machine learning, and physics will deepen your understanding of these concepts.

## 3. Activation Function

In the context of artificial neural networks, an activation function is a mathematical function applied to the output of each neuron (or node) in a neural network. It introduces non-linearity into the network, allowing it to model complex relationships between inputs and outputs. Activation functions play a crucial role in determining the output of a neuron and, consequently, the overall behavior of the neural network.

- **Purpose of Activation Functions**

  - Introducing Non-Linearity: Without activation functions, the entire neural network would behave like a single linear transformation, regardless of its depth and width. Activation functions introduce non-linearity, enabling neural networks to learn complex patterns and relationships in data.
  - Supporting Complex Mapping: Non-linear activation functions allow neural networks to approximate arbitrary functions, making them powerful universal function approximators.
  - Enabling Backpropagation: Activation functions, along with the derivative of the activation function, are essential for the backpropagation algorithm, which is used to train neural networks by adjusting the model's parameters (weights and biases) to minimize the error between predicted and actual outputs.

- **Types of Activation Functions:**

### a. Sigmoid Function

A smooth, S-shaped function that maps input values to the range (0, 1). Examples include the logistic

Python Code

```
import numpy as np

def sigmoid(x):
    """
    Compute the sigmoid function for the input x.

    Parameters:
    x : array_like
        The input values.

    Returns:
    ndarray
        The output of the sigmoid function.
    """
    return 1 / (1 + np.exp(-x))

# Example usage:
x = np.array([-2, -1, 0, 1, 2])
sigmoid_output = sigmoid(x)
print("Input:", x)
print("Sigmoid Output:", sigmoid_output)
```

In this code:

We define a function called sigmoid that takes an input x and returns the result of applying the sigmoid function to each element of x.

- The sigmoid function is defined as $\frac{1}{1+e^{-x}}$

- We use NumPy's exp function to calculate the exponential of each element of -x and then compute the sigmoid function using element-wise division.
- Example usage is provided with an array x, and we print both the input array and the output of the sigmoid function for demonstration.

### b. Rectified Linear Unit (ReLU)

A piecewise linear function that returns the input if it is positive, and zero otherwise. ReLU is widely used due to its simplicity and effectiveness in training deep neural networks.

Python Code:

```
import numpy as np
def relu(x):
    """
    Compute the Rectified Linear Unit (ReLU) function for the input x.

    Parameters:
    x : array_like
        The input values.

    Returns:
    ndarray
        The output of the ReLU function.
    """
    return np.maximum(0, x)

# Example usage:
x = np.array([-2, -1, 0, 1, 2])
relu_output = relu(x)
print("Input:", x)
print("ReLU Output:", relu_output)
```

In this code:

- We define a function called relu that takes an input x and returns the result of applying the ReLU function to each element of x.
- The ReLU function is defined as ReLU($x$)=max(0,$x$), which means that for any negative input value, the output is 0, and for any non-negative input value, the output is the same as the input value.
- We use NumPy's maximum function to compute the element-wise maximum between 0 and x, effectively implementing the ReLU function.
- Example usage is provided with an array x, and we print both the input array and the output of the ReLU function for demonstration.

### c. Leaky ReLU

A variant of ReLU that allows a small, non-zero gradient when the input is negative. This addresses the "dying ReLU" problem, where neurons may become inactive and stop learning.

Python Code

```
def leaky_relu(x, alpha=0.01):
    """
    Compute the Leaky Rectified Linear Unit (Leaky ReLU) function for the input x.

    Parameters:
    x : array_like
        The input values.
    alpha : float, optional
        The slope of the negative part of the function. Default is 0.01.

    Returns:
    ndarray
        The output of the Leaky ReLU function.
    """
    return np.where(x > 0, x, alpha * x)
```

```
# Example usage:
x = np.array([-2, -1, 0, 1, 2])
leaky_relu_output = leaky_relu(x)
print("Input:", x)
print("Leaky ReLU Output:", leaky_relu_output)
```

In this code

- We define a function called leaky_relu that takes an input x and an optional parameter alpha (the slope of the negative part of the function) and returns the result of applying the Leaky ReLU function to each element of x.
- The Leaky ReLU function is defined as follows:
  - For $x>0$, the output is the same as the input: LeakyReLU($x$)=$x$.
  - For $x\leq 0$, the output is LeakyReLU($x$)=$\alpha \cdot x$, where $\alpha$ is a small constant (typically around 0.01).
- We use NumPy's where function to compute the output of the Leaky ReLU function based on the condition $x>0$.
- Example usage is provided with an array x, and we print both the input array and the output of the Leaky ReLU function for demonstration.

### d. Logit Function

The logit function is a mathematical function used in statistics and machine learning, particularly in the context of logistic regression and binary classification. It transforms the probability values (which typically range from 0 to 1) into log-odds values (which range from negative infinity to positive infinity).

Python Code :

```
import numpy as np

def logit(p):
    """
    Compute the logit function for the input probability values.
```

```
    Parameters:
    p : array_like
        The input probability values.

    Returns:
    ndarray
        The output of the logit function.
    """
    return np.log(p / (1 - p))

# Example usage:
p_values = np.array([0.1, 0.3, 0.5, 0.7, 0.9])
logit_values = logit(p_values)
print("Probability Values:", p_values)
print("Logit Values:", logit_values)
```

In this code :

- We define a function called logit that takes an input array p (probability values) and returns the result of applying the logit function to each element of p.
- The logit function is defined as logit(p) = $log(\frac{p}{1-p})$
- We use NumPy's log function to compute the natural logarithm of the ratio of p to $1-p$.
- Example usage is provided with an array p_values, and we print both the input array and the output of the logit function for demonstration.

### e. Step Function

The step function, also known as the Heaviside step function, is a simple mathematical function that returns a binary output based on the input value. It is commonly used as an activation function in binary classification problems, where the output is either 0 or 1.

**Python Code :**

```
import numpy as np

def step_function(x):
    """
    Compute the step function for the input x.

    Parameters:
    x : array_like
        The input values.

    Returns:
    ndarray
        The output of the step function.
    """
    return np.where(x >= 0, 1, 0)

# Example usage:
x_values = np.array([-2, -1, 0, 1, 2])
step_values = step_function(x_values)
print("Input Values:", x_values)
print("Step Function Values:", step_values)
```

In this Code:

- We define a function called step_function that takes an input array x and returns the result of applying the step function to each element of x.
- The step function is implemented using NumPy's where function, which returns 1 where the condition x >= 0 is True and 0 otherwise.
- Example usage is provided with an array x_values, and we print both the input array and the output of the step function for demonstration.

**Activation Functions in Practice:**

The choice of activation function depends on the nature of the problem, the architecture of the neural network, and empirical performance on validation data. Experimenting with different activation functions can lead to improvements in training speed, convergence, and generalization performance of neural networks.

## 4. Cost Function

A cost function, also known as a loss function or objective function, is a key concept in the field of machine learning and optimization. It quantifies the difference between the predicted output of a model and the actual target values in the training data. The goal of training a machine learning model is to minimize this cost function, which effectively means reducing the difference between the predicted and actual values.

Exercise: Suppose we have the following dataset representing the relationship between the number of hours studied and the exam scores:

| Hours Studied | Exam Score |
|---|---|
| 2 | 81 |
| 3 | 91 |
| 4 | 93 |
| 5 | 97 |
| 6 | 99 |

We want to build a linear regression model to predict the exam score based on the number of hours studied. The linear regression model is represented as:

$$y = mx + b$$

- Y is the predicted exam score,
- $x$ is the number of hours studied,
- $m$ is the slope of the line (weight),
- $b$ is the y-intercept (bias).

Let's assume that the model has been trained, and we have the following predicted exam scores:

| Hours Studied | Actual Score | Predicted Score |
|---|---|---|
| 2 | 81 | 82 |
| 3 | 91 | 88 |
| 4 | 93 | 94 |
| 5 | 97 | 100 |
| 6 | 99 | 106 |

## 5. Plotting Function

Plotting functions is a common task in data analysis, visualization, and understanding the behavior of mathematical functions. Python provides several libraries for plotting functions, with Matplotlib being one of the most popular ones. Here's a basic example of how to plot functions using Matplotlib:

Example: Plotting a Simple Mathematical Function

Let's plot the function

$$f(x)=x^2 \text{ over the range } [-5,5]$$

Python Code

```
import numpy as np
import matplotlib.pyplot as plt

# Define the range of x values
x = np.linspace(-5, 5, 100)  # Generate 100 evenly spaced points between -5 and 5

# Define the function
def f(x):
    return x**2

# Calculate the corresponding y values using the function
y = f(x)
```

```
# Plot the function
plt.plot(x, y, label='f(x) = x^2')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.title('Plot of f(x) = x^2')
plt.grid(True)
plt.legend()
plt.show()
```

## 6. Minimum and Maximum Value of a function

To find the minimum and maximum values of a function, you can use calculus techniques such as differentiation and critical points. Here's a step-by-step approach to finding the minimum and maximum values of a function:

- **Find Critical Points:**
    a) Find the first derivative of the function.
    b) Set the derivative equal to zero and solve for
    c) *x* to find critical points.
    d) Critical points are where the function may have minimum, maximum, or saddle points.

- **Determine Nature of Critical Points:**
    a) Use the second derivative test or analyze the behavior of the function around critical points to determine if they correspond to minima, maxima, or saddle points.
- **Evaluate Function Values:**
    a. Once you have identified critical points, evaluate the function at these points and also at the endpoints of the domain (if applicable).
    b. The lowest function value among these points will be the minimum, and the highest function value will be the maximum.

## Self-Check Sheet - 2: Use Multivariable Calculus

Q1: What is a multivariable function?

Q2: What is the derivative of a function?

Q3: What is a step function in neural networks?

Q4: What is the sigmoid function, and where is it used?

Q5: What is the ReLU function, and why is it popular?

Q6: How do you plot a multivariable function?

Q7: What tools can be used to plot functions?

Q8: How do you find the minimum and maximum values of a function?

Q9: What is a local minimum versus a global minimum?

## Answer Sheet - 2: Use Multivariable Calculus

**Q1: What is a multivariable function?**
**A:** A multivariable function is a function that has more than one input or independent variable. It maps a set of input values to a single output value. For example, f(x,y)= x^2 + y^2 is a multivariable function of two variables, x and y.

**Q2: What is the derivative of a function?**
**A:** The derivative of a function measures the rate of change of the function with respect to one of its variables. For example, if f(x)=x^2 the derivative with respect to x denoted f′(x), is 2x.

**Q3: What is a step function in neural networks?**
**A:** A step function is a type of activation function that outputs 0 if the input is less than a certain threshold and 1 if the input is greater than or equal to the threshold. It's a simple binary output function often used in early perceptrons.

**Q4: What is the sigmoid function, and where is it used?**
**A:** The sigmoid function is an S-shaped activation function defined as:

σ(x)=11+e-x
It maps input values to the range (0, 1) and is used in neural networks to introduce non-linearity. It is commonly used in logistic regression and binary classification tasks

**Q5: What is the ReLU function, and why is it popular?**
**A:** The ReLU function is defined as:
f(x)=max(0,x)

It outputs the input if it’s positive and 0 if it’s negative. ReLU is popular because it allows for faster training of deep neural networks and helps mitigate the vanishing gradient problem by not saturating for positive values.

**Q6: How do you plot a multivariable function?**
**A:** To plot a multivariable function, you can use 2D or 3D graphing techniques depending on the number of variables. For a function with two variables f(x,y) you can create a 3D surface plot where the x- and y-axes represent the inputs and the z-axis represents the output values.

**Q7: What tools can be used to plot functions?**

**A:** Popular tools for plotting functions include:

- **Matplotlib**: For 2D and 3D plots in Python.
- **Desmos**: An online graphing calculator.
- **MATLAB**: A powerful tool for mathematical computation and plotting.
- **WolframAlpha**: An online computational engine that can plot various functions.

**Q8: How do you find the minimum and maximum values of a function?**

**A:** The minimum and maximum values (extrema) of a function can be found by:

1. **Taking the derivative** of the function.
2. **Setting the derivative equal to zero** to find the critical points.
3. **Using the second derivative test** to classify the critical points as minima, maxima, or saddle points. For multivariable functions, you compute the gradient and set it to zero to find critical points.

**Q9: What is a local minimum versus a global minimum?**

**A:**

- A **local minimum** is a point where the function has a lower value than nearby points but may not be the lowest overall.
- A **global minimum** is the absolute lowest point of the function over its entire domain.

**Task Sheet-2.1: Multivariable Functions: a. Define a multivariable function and explain its significance in mathematics and data analysis. b. Consider the function f(x, y) = x^2 + 2xy + y^2. Calculate the partial derivatives ∂f/∂x and ∂f/∂y.**

**1. Define a Multivariable Function and Explain Its Significance in Mathematics and Data Analysis**

- **Multivariable Function**:
  - A **multivariable function** is a mathematical function that has more than one input variable (also known as independent variables). These functions can be written in the form:

$$f(x_1, x_2, \ldots, x_n)$$

Where $f$ is the function, and $x_1, x_2, \ldots, x_n$ are the independent variables.

variables.

- **Significance in Mathematics**:
  - Multivariable functions are essential in calculus, where they are used to study functions of two or more variables. This extends the concepts of differentiation and integration to higher dimensions, allowing for a more comprehensive understanding of how systems behave.
- **Significance in Data Analysis**:
  - In data analysis, multivariable functions model relationships between different variables. For example, in machine learning, multivariable functions can represent models like linear regression, where the output depends on multiple features (independent variables). By analyzing these functions, we can predict outcomes, identify trends, and optimize models.
  - In data science, functions like these help describe complex relationships between various factors, enabling better decision-making and insights.

**2. Consider the Function** $f(x, y) = x^2 + 2xy + y^2.$

**Calculate the Partial Derivatives** $\frac{\partial f}{\partial x}$ **and** $\frac{\partial f}{\partial y}$

- **Given Function**:

$$f(x, y) = x^2 + 2xy + y^2$$

- **Partial Derivatives**:
  - **Partial Derivative with Respect to xxx**:
    - To calculate ∂f∂x\frac{\partial f}{\partial x}∂x∂f, we differentiate the function with respect to xxx, treating yyy as a constant.

    - To calculate $\frac{\partial f}{\partial x}$, we differentiate the function with respect to $x$, treating $y$ as a constant.

$$\frac{\partial f}{\partial x} = \frac{\partial}{\partial x}(x^2 + 2xy + y^2)$$

Differentiating each term:

- $\frac{\partial}{\partial x}(x^2) = 2x$
- $\frac{\partial}{\partial x}(2xy) = 2y$ (since $y$ is constant with respect to $x$)
- $\frac{\partial}{\partial x}(y^2) = 0$ (since $y^2$ does not depend on $x$)

Therefore:

$$\frac{\partial f}{\partial x} = 2x + 2y$$

  - **Partial Derivative with Respect to yyy**:
    - To calculate ∂f∂y\frac{\partial f}{\partial y}∂y∂f, we differentiate the function with respect to yyy, treating xxx as a constant.

- To calculate $\frac{\partial f}{\partial y}$, we differentiate the function with respect to $y$, treating $x$ as a constant.

$$\frac{\partial f}{\partial y} = \frac{\partial}{\partial y}(x^2 + 2xy + y^2)$$

Differentiating each term:

- $\frac{\partial}{\partial y}(x^2) = 0$ (since $x^2$ does not depend on $y$)
- $\frac{\partial}{\partial y}(2xy) = 2x$ (since $x$ is constant with respect to $y$)
- $\frac{\partial}{\partial y}(y^2) = 2y$

Therefore:

$$\frac{\partial f}{\partial y} = 2x + 2y$$

**Task Sheet-2.2: Derivatives and Gradients: a. Define a derivative and explain its interpretation in calculus. b. Compute the gradient of the function f(x) = 3x^2 - 2x + 1.**

**Step:**

**1. Define a Derivative and Explain Its Interpretation in Calculus**

- **Definition of Derivative**:
  - A **derivative** of a function represents the rate at which the function's output changes as its input changes. In mathematical terms, it is the slope of the tangent line to the curve of the function at any given point.
  - The derivative is a measure of how a function behaves locally and is calculated as the limit of the average rate of change over an infinitesimally small interval.
- **Interpretation in Calculus**:
  - The derivative f′(x)f'(x)f′(x) gives the **instantaneous rate of change** of the function at any point xxx. It tells us how fast the function is changing with respect to xxx.
  - Geometrically, the derivative represents the **slope of the tangent line** at a point on the curve.
  - In machine learning, derivatives are used in **gradient descent** to optimize the parameters of a model by minimizing a loss function.

**2. Compute the Gradient of the Function** $f(x) = 3x^2 - 2x + 1$

- **Given Function**:

$$f(x) = 3x^2 - 2x + 1$$

**Gradient Calculation**:

- The **gradient** of a function refers to the vector of partial derivatives of the function with respect to each independent variable. In the case of a single-variable function, the gradient is simply the derivative.
- To compute the gradient of f(x)f(x)f(x), we calculate the derivative of the function f(x)f(x)f(x) with respect to xxx.

**Steps**:

- To find the derivative, we apply the power rule:
  - The derivative of $x^n$ is $nx^{n-1}$.
- **Derivative of** $f(x) = 3x^2 - 2x + 1$:

$$\frac{d}{dx}(3x^2) = 6x$$

$$\frac{d}{dx}(-2x) = -2$$

$$\frac{d}{dx}(1) = 0$$

- So, the **derivative** of the function $f(x)$ is:

$$f'(x) = 6x - 2$$

**Gradient**:

- Since this is a single-variable function, the **gradient** is simply the derivative:

$$\nabla f( \downarrow = f'(x) = 6x - 2$$

**Task Sheet-2.3: Activation Functions: a. Explain the purpose of activation functions in neural networks. b. Define and plot the following activation functions: i. Step function ii. Sigmoid function iii. Logit functioniv. ReLU (Rectified Linear Unit) function**

**Step**

**1. Explain the Purpose of Activation Functions in Neural Networks**

- **Purpose of Activation Functions**:
  - In neural networks, **activation functions** introduce non-linearity into the model. Without them, the neural network would essentially be a linear model, limiting its ability to model complex patterns in data.
  - Activation functions decide whether a neuron should be activated or not, based on the input it receives. They take the weighted sum of inputs and pass them through a non-linear transformation to produce the output.
  - Key reasons for using activation functions:
    - **Non-linearity**: Allows neural networks to learn complex patterns.
    - **Controlling output range**: Activation functions can limit the output to a specific range, which is important for controlling the flow of data through layers.
    - **Enabling backpropagation**: Activation functions have derivatives, which are essential for updating the weights during training using gradient descent.

**2. Define and Plot the Following Activation Functions:**

- **Step Function**:
  - **Definition**: The step function is a simple binary activation function that outputs one value (e.g., 1) if the input is above a certain threshold and another value (e.g., 0) if it is below the threshold. f(x)={1if x≥00if x<0f(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}
  - **Plotting**: The plot of the step function is a discontinuous function with a sharp jump at the threshold (usually 0).
- **Sigmoid Function**:
  - **Definition**: The sigmoid function is a smooth, S-shaped curve that maps any real-valued number to a value between 0 and 1. It is widely used in binary classification problems. f(x)=11+e−xf(x) = \frac{1}{1 + e^{-x}}
  - **Plotting**: The sigmoid curve smoothly approaches 0 as x→−∞x \to -\infty and approaches 1 as x→+∞x \to +\infty.
- **Logit Function**:
  - **Definition**: The logit function is the inverse of the sigmoid function. It maps values between 0 and 1 to the entire real number range. f(x)=log⁡(x1−x)f(x) = \log\left(\frac{x}{1-x}\right)

- **Plotting**: The logit function has an S-shape, but its range is from −∞-\infty to +∞+\infty, mapping values from the 0–1 interval to the entire real line.

- **ReLU (Rectified Linear Unit) Function**:
  - **Definition**: The ReLU function is defined as the positive part of its input. It is one of the most commonly used activation functions because it helps with training deep neural networks by mitigating the vanishing gradient problem. f(x)=max⁡(0,x)f(x) = \max(0, x)
  - **Plotting**: The ReLU function is a linear function for positive inputs and outputs 0 for negative inputs, making the plot a straight line along the x-axis for x<0x < 0 and a line with slope 1 for x≥0x \geq 0.

**Task Sheet-2.4: Cost Function: a. Define a cost function and discuss its role in machine learning. b. Consider a linear regression model with the hypothesis function $h(x) = \theta_0 + \theta_1 x$. Define the cost function $J(\theta)$ for this model.**

**Step**

**1. Define a Cost Function and Discuss Its Role in Machine Learning**

- **Definition of Cost Function**:
  - A **cost function** (also known as a **loss function**) is a mathematical function used in machine learning to quantify the difference between the predicted output of a model and the actual output (or target) in the dataset. The goal of training a machine learning model is to minimize this cost function to improve the model's performance.
- **Role of Cost Function in Machine Learning**:
  - The cost function guides the optimization process, helping the model learn from errors and improve over time.
  - During training, the model's parameters (like weights in neural networks or coefficients in linear regression) are adjusted to minimize the cost function. This is typically done using optimization algorithms such as **gradient descent**.
  - For supervised learning models like **linear regression**, the cost function quantifies how well the model fits the data, providing a way to assess model accuracy and determine the best set of parameters (such as weights or coefficients).

**2. Consider a Linear Regression Model with the Hypothesis Function $h(x)=\theta0+\theta1x$h(x) = \theta_0 + \theta_1 x. Define the Cost Function $J(\theta)$J(\theta) for This Model.**

- **Linear Regression Hypothesis**:
  - In linear regression, the hypothesis function $h(x)$h(x) is defined as:

$h(x)=\theta0+\theta1x$h(x) = \theta_0 + \theta_1 x

Where:

  - $\theta0$\theta_0 is the intercept term (also called bias).
  - $\theta1$\theta_1 is the coefficient (or slope) associated with the feature $x$x.
  - $x$x represents the input feature (independent variable).
- **Cost Function for Linear Regression**:
  - The **cost function** $J(\theta)$J(\theta) for linear regression is usually defined as the **Mean Squared Error (MSE)** between the predicted values and the actual values in the training set.
  - The formula for the cost function is:

$J(\theta)=12m\sum i=1m(h(x(i))-y(i))2$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h(x^{(i)}) - y^{(i)})^2

Where:

  - $m$m is the number of training examples.
  - $x(i)$x^{(i)} is the input feature for the $i$i-th example.
  - $y(i)$y^{(i)} is the actual output (target value) for the $i$i-th example.

- h(x(i))=θ0+θ1x(i)h(x^{(i)}) = \theta_0 + \theta_1 x^{(i)} is the predicted output for the ii-th example.
- The term (h(x(i))−y(i))2(h(x^{(i)}) - y^{(i)})^2 represents the squared difference between the predicted value and the actual value, and the sum averages this error over all training examples.

The reason for squaring the differences is to penalize larger errors more heavily. The factor 12m\frac{1}{2m} is included for convenience, as it simplifies the derivative when using gradient descent.

**Task Sheet-2.5: Plotting of Functions: a. Using Python or any other preferred tool, plot the function f(x) = x^3 - 2x^2 + 3x - 4 in the range [-2, 3]. b. Label the axes and provide a title for the plot.**

**Step**

**1. Define the Function**

The given function is:

f(x)=x3−2x2+3x−4f(x) = x^3 - 2x^2 + 3x - 4

This is a cubic function, and we are tasked with plotting it within the range x∈[−2,3]x \in [-2, 3].

**2. Plot the Function Using Python (Matplotlib)**

- **Python Code** (using **Matplotlib** library):

```
import numpy as np
import matplotlib.pyplot as plt

# Define the function f(x) = x^3 - 2x^2 + 3x - 4
def f(x):
    return x**3 - 2*x**2 + 3*x - 4

# Define the range for x from -2 to 3
x = np.linspace(-2, 3, 100)

# Compute f(x) for each value of x
y = f(x)

# Plot the function
plt.plot(x, y, label=r'$f(x) = x^3 - 2x^2 + 3x - 4$')

# Label the axes
plt.xlabel('x')
plt.ylabel('f(x)')

# Provide a title for the plot
plt.title('Plot of the function $f(x) = x^3 - 2x^2 + 3x - 4$')

# Show a grid for better visualization
plt.grid(True)

# Display the plot
plt.legend()
```

plt.show()

- **Explanation**:
  - The code uses numpy to generate a range of values for xx in the range [−2,3][-2, 3].
  - The function f(x)f(x) is defined as f(x)=x3−2x2+3x−4f(x) = x^3 - 2x^2 + 3x - 4.
  - matplotlib.pyplot is used to plot the function, label the axes, and provide a title.
  - A grid is added to improve visualization, and a legend is included to indicate the function on the plot.

**3. Label the Axes and Provide a Title**

- **X-Axis Label**: This represents the input variable xx, so it is labeled as "x".
- **Y-Axis Label**: This represents the output of the function, so it is labeled as "f(x)".
- **Plot Title**: The title describes the function being plotted and is set as "Plot of the function f(x)=x3−2x2+3x−4f(x) = x^3 - 2x^2 + 3x - 4".

**Task Sheet-2.6: Minimum and Maximum Values of a Function: a. Define local minimum, local maximum, global minimum, and global maximum of a function. b. Determine the local and global minimum and maximum values, if any, for the function f(x) = x^4 - 4x^3 + 6x^2.**

**Step**

**1. Define Local Minimum, Local Maximum, Global Minimum, and Global Maximum of a Function**

- **Local Minimum**: A function has a **local minimum** at a point x=ax = a if the function value at aa is smaller than the values of the function at nearby points. Mathematically, for a function f(x)f(x), f(a)f(a) is a local minimum if:

f(a)≤f(x)for allx near af(a) \leq f(x) \quad \text{for all} \quad x \text{ near } a

- **Local Maximum**: A function has a **local maximum** at a point x=ax = a if the function value at aa is larger than the values of the function at nearby points. Mathematically, for a function f(x)f(x), f(a)f(a) is a local maximum if:

f(a)≥f(x)for allx near af(a) \geq f(x) \quad \text{for all} \quad x \text{ near } a

- **Global Minimum**: A function has a **global minimum** at a point x=ax = a if the function value at aa is the smallest value of the function for all xx in the domain of the function. Mathematically:

f(a)≤f(x)for allx in the domain of ff(a) \leq f(x) \quad \text{for all} \quad x \text{ in the domain of } f

- **Global Maximum**: A function has a **global maximum** at a point x=ax = a if the function value at aa is the largest value of the function for all xx in the domain of the function. Mathematically:

f(a)≥f(x)for allx in the domain of ff(a) \geq f(x) \quad \text{for all} \quad x \text{ in the domain of } f

**2. Find the Critical Points of the Function f(x)=x4−4x3+6x2f(x) = x^4 - 4x^3 + 6x^2**

To find the critical points, we need to compute the **first derivative** of the function and set it equal to zero.

- **Step 1: Find the first derivative** of f(x)f(x):

f′(x)=ddx(x4−4x3+6x2)=4x3−12x2+12xf'(x) = \frac{d}{dx} \left( x^4 - 4x^3 + 6x^2 \right) = 4x^3 - 12x^2 + 12x

- **Step 2: Set the first derivative equal to zero** and solve for xx:

4x3−12x2+12x=04x^3 - 12x^2 + 12x = 0

Factor the equation:

4x(x2−3x+3)=04x(x^2 - 3x + 3) = 0

Solve for xx:

x=0orx2−3x+3=0x = 0 \quad \text{or} \quad x^2 - 3x + 3 = 0

The quadratic x2−3x+3=0x^2 - 3x + 3 = 0 has no real roots (discriminant is negative), so the only critical point is x=0x = 0.

**3. Determine the Nature of the Critical Point (Local Minima, Maxima, or Saddle Point)**

- **Step 1: Find the second derivative** of f(x)f(x):

f′(x)=ddx(4x3−12x2+12x)=12x2−24x+12f'(x) = \frac{d}{dx} \left( 4x^3 - 12x^2 + 12x \right) = 12x^2 - 24x + 12

- **Step 2: Evaluate the second derivative at x=0x = 0**:

f′(0)=12(0)2−24(0)+12=12f'(0) = 12(0)^2 - 24(0) + 12 = 12

Since f′(0)>0f'(0) > 0, the function has a **local minimum** at x=0x = 0.

**4. Analyze the Behavior of the Function at the Endpoints**

Since the function is a polynomial of degree 4, its limits as x→∞x \to \infty and x→−∞x \to -\infty will both approach infinity:

- As x→∞x \to \infty, f(x)→∞f(x) \to \infty
- As x→−∞x \to -\infty, f(x)→∞f(x) \to \infty

Thus, there is **no global maximum** or **global minimum** at the endpoints.

**5. Conclusion: Local and Global Extrema**

- **Local Minimum**: At x=0x = 0, f(0)=04−4(0)3+6(0)2=0f(0) = 0^4 - 4(0)^3 + 6(0)^2 = 0. This is the **local minimum**.
- **Global Minimum**: The function does not have a global minimum because it tends to infinity as x→∞x \to \infty and x→−∞x \to -\infty.
- **Local Maximum**: There is **no local maximum** for this function in the real domain.
- **Global Maximum**: There is **no global maximum** because the function tends to infinity at both ends of the domain.

## Learning Outcome 3: Apply Linear Algebra

| | |
|---|---|
| Assessment Criteria | 1. Matrices are exercised<br>2. Vectors are exercised |
| Conditions and Resources | 1. Actual workplace or training environment<br>2. CBLM<br>3. Handouts<br>4. Laptop<br>5. Multimedia Projector<br>6. Paper, Pen, Pencil and Eraser<br>7. Internet Facilities<br>8. Whiteboard and Marker<br>9. Imaging Device (Digital camera, scanner etc.) |
| Contents | • Matrices<br>  o Transpose of a matrix<br>  o The inverse of a matrix<br>  o The determinant of a matrix<br>• Vector<br>  o Dot product<br>  o Eigenvalues<br>  o Eigenvectors |
| Activities/job/Task | 1. Apply Linear Algebra |
| Training Methods | 2. Discussion<br>3. Presentation<br>4. Demonstration<br>5. Guided Practice<br>6. Individual Practice<br>7. Project Work<br>8. Problem Solving<br>9. Brainstorming |
| Assessment Methods | 1. Written Test<br>2. Demonstration<br>3. Oral Questioning |

## Learning Experience 3: Apply Linear Algebra

In order to achieve the objectives stated in this learning guide, you must perform the learning steps below. Beside each step are the resources or special instructions you will use to accomplish the corresponding activity.

| Learning Activities | Recourses/Special Instructions |
|---|---|
| 1. Trainee will ask the instructor about the learning materials | 1. Instructor will provide the learning materials 'Apply Statistical measures' |
| 2. Read the Information sheet and complete the Self Checks & Check answer sheets on "Apply Statistical measures" | 2. Read Information sheet 2: Use Multivariable Calculus<br>3. Answer Self-check 2: Use Multivariable Calculus<br>4. Check your answer with Answer key 2: Use Multivariable Calculus |
| 3. Read the Job/Task Sheet and Specification Sheet and perform job/Task | 5. Job/Task Sheet and Specification Sheet Task Sheet-3.1: Apply Linear Algebra |

## Information Sheet 3: Apply Linear Algebra

**Learning Objective:**

After completion of this information sheet, the learners will be able to explain, define and interpret the following contents:

1. Matrices
2. Vectors

## 1. Matrices

In linear algebra, a matrix is a rectangular array of numbers (or other mathematical objects) arranged in rows and columns. Matrices are fundamental mathematical objects that are used to represent and manipulate linear transformations, solve systems of linear equations, and perform various other operations in mathematics, science, and engineering.

### a. Transpose Matrix

The transpose of a matrix is an operation that flips the rows and columns of the original matrix. It essentially reflects the matrix over its main diagonal. If the original matrix has dimensions $m \times n$, the transpose will have dimensions $n \times m$.

For a matrix A with elements $a_{ij}$, the transpose of A, denoted as $A^T$, is formed by placing the i-th row of A as the i-th column of $A^T$.Mathematically, the transpose operation is defined as

$(A^T)ij = A_{ji}$ In other words, the element in the i-th row and j-th column of the transpose matrix $A^T$ is equal to the element in the j-th row and i-th column of the original matrix A.

Consider the following matrix $A$:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

The transpose of A, denoted as $A^T$ will be:

$$A^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

Python Code:

```
import numpy as np

# Define a sample matrix
matrix = np.array([[1, 2, 3],
                   [4, 5, 6],
                   [7, 8, 9]])

# Transpose the matrix
transposed_matrix = np.transpose(matrix)

print("Original Matrix:")
print(matrix)

print("\nTransposed Matrix:")
print(transposed_matrix)
```

## b. Inverse Matrix

The inverse of a square matrix is a matrix that, when multiplied with the original matrix, yields the identity matrix. The inverse of a matrix A is denoted as A−1. However, not all matrices have inverses. A matrix must be square (i.e., have the same number of rows and columns) and must be non-singular (i.e., its determinant must be non-zero) to have an inverse.

Example: Consider the following matrix *A*:

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}$$

To find the inverse of *A*, you first calculate its determinant det(*A*) :

$$\det(A) = (2 \times 3) - (1 \times 1) = 5$$

Then, you find the adjugate matrix of *A* (which involves finding the cofactor matrix and taking its transpose), let's call it adj(*A*):

$$\text{adj}(A) = \begin{bmatrix} 3 & -1 \\ -1 & 2 \end{bmatrix}$$

Finally, you can calculate the inverse of *A* using the formula:

$$A^{-1} = \frac{1}{5} \begin{bmatrix} 3 & -1 \\ -1 & 2 \end{bmatrix} = \begin{bmatrix} \frac{3}{5} & -\frac{1}{5} \\ -\frac{1}{5} & \frac{2}{5} \end{bmatrix}$$

Python Code :

```
import numpy as np

# Define a sample matrix
matrix = np.array([[1, 2],
                   [3, 4]])

# Calculate the inverse of the matrix
inverse_matrix = np.linalg.inv(matrix)

print("Original Matrix:")
print(matrix)

print("\nInverse Matrix:")
print(inverse_matrix)
```

### c. Determinant

The determinant of a square matrix is a scalar value that represents certain properties of the matrix. It is denoted by det(*A*) or |*A*|. The determinant is only defined for square matrices (i.e., matrices with an equal number of rows and columns).

**Determinant of a 2x2 Matrix:**

For a 2x2 matrix:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

The determinant is calculated as:

$$\det(A) = ad - bc$$

Python Code

```
import numpy as np

# Define your matrix
A = np.array([[2, -1, 0],
              [-1, 2, -1],
              [0, -1, 2]])

# Calculate determinant
determinant = np.linalg.det(A)

print("Determinant of A:", determinant)
```

## 2. Vectors

Vectors play a fundamental role in machine learning, as they are used to represent data, features, parameters, and predictions in various machine learning models and algorithms.

### a. Dot Product

The dot product (also known as the inner product or scalar product) is a mathematical operation that takes two vectors and returns a scalar quantity. It is defined for vectors of the same size (i.e., having the same number of elements).

**Properties of the Dot Product**

- Commutativity: $a \cdot b = b \cdot a$
- Distributivity over Addition: $a \cdot (b+c) = a \cdot b + a \cdot c$
- Scalar Multiplication: $k(a \cdot b) = (k \cdot a) \cdot b = a \cdot (k \cdot b)$, where k is a scalar.
- Orthogonality: If **a** and **b** are orthogonal (i.e., the angle between them is 90∘ or $\pi/2$ radians), then $\mathbf{a} \cdot \mathbf{b} = 0$.

Python Code

```
import numpy as np

# Define two vectors
vector1 = np.array([1, 2, 3])
vector2 = np.array([4, 5, 6])

# Calculate the dot product
dot_product = np.dot(vector1, vector2)

print("Dot product of vector1 and vector2:", dot_product)
```

### b. Eigenvalues

Eigenvalues are a fundamental concept in linear algebra that play a crucial role in various mathematical and computational applications, including machine learning. Eigenvalues are associated with square matrices and provide important information about the behavior of linear transformations represented by those matrices.

Given a square matrix $A$, an eigenvalue of $A$ is a scalar $\lambda$ that satisfies the equation:

$$A\mathbf{v}=\lambda\mathbf{v}$$

Here, **v** is a non-zero vector called an eigenvector corresponding to the eigenvalue $\lambda$. In other words, when the linear transformation represented by $A$ is applied to the eigenvector **v**, the resulting vector is a scalar multiple of **v**, where the scalar is the eigenvalue $\lambda$.

Python code

```
import numpy as np

# Define your matrix
A = np.array([[2, -1, 0],
              [-1, 2, -1],
              [0, -1, 2]])

# Calculate eigenvalues
eigenvalues = np.linalg.eigvals(A)

print("Eigenvalues of A:")
print(eigenvalues)
```

### c. Eigenvectors

The eigenvector is a vector that is associated with a set of linear equations. The eigenvector of a matrix is also known as a latent vector, proper vector, or characteristic vector. These are defined in the reference of a square matrix. Eigenvectors are also useful in solving differential equations and many other applications related to them. In this article, let us discuss the eigenvector definition, equation, methods with examples in detail.

Eigenvector of a square matrix is defined as a non-vector in which when a given matrix is multiplied, it is equal to a scalar multiple of that vector. Let us suppose that A is an n x n square matrix, and if v be a non-zero vector, then the product of matrix A, and vector v is defined as the product of a scalar quantity λ and the given vector, such that:

$$Av =\lambda v$$

Where

v = Eigenvector and λ be the scalar quantity that is termed as eigenvalue associated with given matrix A

- **Eigenvector Equation**

The equation corresponding to each eigenvalue of a matrix is given by:

$$AX = \lambda X$$

It is formally known as the eigenvector equation.

In place of λ, substitute each eigenvalue and get the eigenvector equation which enables us to solve for the eigenvector belonging to each eigenvalue.

- **Eigenvector Method**

The method of determining the eigenvector of a matrix is given as follows:

If A be an n×n matrix and λ be the eigenvalues associated with it. Then, eigenvector v can be defined by the following relation: $Av = \lambda v$

If "I" be the identity matrix of the same order as A, then

$$(A - \lambda I)v = 0$$

The eigenvector associated with matrix A can be determined using the above method.

Here, "v" is known as eigenvector belonging to each eigenvalue and is written as:

$$v = \begin{bmatrix} v_1 \\ v_2 \\ . \\ . \\ v_n \end{bmatrix}$$

- **How to Find an Eigenvector?**

To find the eigenvectors of a matrix, follow the procedure given below:

1. Find the eigenvalues of the given matrix A, using the equation det ((A – λI) =0, where "I" is equivalent order identity matrix as A. Denote each eigenvalue of λ1, λ2, λ3….
2. Substitute the values in the equation AX = λ1 or (A – λ1 I) X = 0.
3. Calculate the value of eigenvector X, which is associated with the eigenvalue.
4. Repeat the steps to find the eigenvector for the remaining eigenvalues.

- **Types of Eigenvector**

The eigenvectors are of two types namely,

i. Left Eigenvector
ii. Right Eigenvector

**i. Left Eigenvector**

The left eigenvector is represented in the form of a row vector which satisfies the following condition:

$$AXL=\lambda XL$$

Where

A is a given matrix of order n and λ be one of its eigenvalues.

XL is a row vector of a matrix. I,e., [ x1 x2 x3 …. Xn]

**ii. Right Eigenvector**

The right eigenvector is represented in the form of a column vector which satisfies the following condition:

$$AXR=\lambda XR$$

Where

A is a given matrix of order n and λ be one of its eigenvalues.

XR is a column vector of a matrix. I,e.,

$$X_R = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_n \end{bmatrix}$$

- **Eigenvector Applications**

The important application of eigenvectors are as follows:

- Eigenvectors are used in Physics in simple mode of oscillation
- In Mathematics, eigenvector decomposition is widely used in order to solve the linear equation of first order, in ranking matrices, in differential calculus etc
- This concept is widely used in quantum mechanics
- It is applicable in almost all the branches of engineering

Vectors play a fundamental role in machine learning, as they are used to represent data, features, parameters, and predictions in various machine learning models and algorithms.

## d. Dot Product

The dot product (also known as the inner product or scalar product) is a mathematical operation that takes two vectors and returns a scalar quantity. It is defined for vectors of the same size (i.e., having the same number of elements).

**Properties of the Dot Product:**

- Commutativity: $a \cdot b = b \cdot a$
- Distributivity over Addition: $a \cdot (b+c) = a \cdot b + a \cdot c$
- Scalar Multiplication: $k(a \cdot b) = (k \cdot a) \cdot b = a \cdot (k \cdot b)$, where k is a scalar.
- Orthogonality: If **a** and **b** are orthogonal (i.e., the angle between them is 90∘ or $\pi/2$ radians), then $\mathbf{a} \cdot \mathbf{b} = 0$.

Python Code

```
import numpy as np

# Define two vectors
vector1 = np.array([1, 2, 3])
vector2 = np.array([4, 5, 6])

# Calculate the dot product
dot_product = np.dot(vector1, vector2)

print("Dot product of vector1 and vector2:", dot_product)
```

### e. Eigenvalues

Eigenvalues are a fundamental concept in linear algebra that play a crucial role in various mathematical and computational applications, including machine learning. Eigenvalues are associated with square matrices and provide important information about the behavior of linear transformations represented by those matrices.

Given a square matrix $A$, an eigenvalue of $A$ is a scalar $\lambda$ that satisfies the equation:

$$A\mathbf{v}=\lambda\mathbf{v}$$

Here, $\mathbf{v}$ is a non-zero vector called an eigenvector corresponding to the eigenvalue $\lambda$. In other words, when the linear transformation represented by $A$ is applied to the eigenvector $\mathbf{v}$, the resulting vector is a scalar multiple of $\mathbf{v}$, where the scalar is the eigenvalue $\lambda$.

Python code

```
import numpy as np

# Define your matrix
A = np.array([[2, -1, 0],
              [-1, 2, -1],
              [0, -1, 2]])
```

```
# Calculate eigenvalues
eigenvalues = np.linalg.eigvals(A)

print("Eigenvalues of A:")
print(eigenvalues)
```

## f. Eigenvectors

The eigenvector is a vector that is associated with a set of linear equations. The eigenvector of a matrix is also known as a latent vector, proper vector, or characteristic vector. These are defined in the reference of a square matrix. Eigenvectors are also useful in solving differential equations and many other applications related to them. In this article, let us discuss the eigenvector definition, equation, methods with examples in detail.

Eigenvector of a square matrix is defined as a non-vector in which when a given matrix is multiplied, it is equal to a scalar multiple of that vector. Let us suppose that A is an n x n square matrix, and if v be a non-zero vector, then the product of matrix A, and vector v is defined as the product of a scalar quantity λ and the given vector, such that:

$$Av = \lambda v$$

Where

v = Eigenvector and λ be the scalar quantity that is termed as eigenvalue associated with given matrix A

- **Eigenvector Equation**

The equation corresponding to each eigenvalue of a matrix is given by:

$$AX = \lambda X$$

It is formally known as the eigenvector equation.

In place of λ, substitute each eigenvalue and get the eigenvector equation which enables us to solve for the eigenvector belonging to each eigenvalue.

- **Eigenvector Method**

The method of determining the eigenvector of a matrix is given as follows:

If A be an n×n matrix and λ be the eigenvalues associated with it. Then, eigenvector v can be defined by the following relation: Av =λv

If "I" be the identity matrix of the same order as A, then

$$(A - \lambda I)v = 0$$

The eigenvector associated with matrix A can be determined using the above method.

Here, "v" is known as eigenvector belonging to each eigenvalue and is written as:

$$v = \begin{bmatrix} v_1 \\ v_2 \\ . \\ . \\ v_n \end{bmatrix}$$

- **How to Find an Eigenvector?**

To find the eigenvectors of a matrix, follow the procedure given below:

5. Find the eigenvalues of the given matrix A, using the equation det ((A – λI) =0, where "I" is equivalent order identity matrix as A. Denote each eigenvalue of λ1, λ2, λ3….
6. Substitute the values in the equation AX = λ1 or (A – λ1 I) X = 0.
7. Calculate the value of eigenvector X, which is associated with the eigenvalue.
8. Repeat the steps to find the eigenvector for the remaining eigenvalues.

- **Types of Eigenvector**

The eigenvectors are of two types namely,

iii. Left Eigenvector
iv. Right Eigenvector

### i. Left Eigenvector

The left eigenvector is represented in the form of a row vector which satisfies the following condition:

$$AXL=\lambda XL$$

Where

A is a given matrix of order n and λ be one of its eigenvalues.

XL is a row vector of a matrix. I,e., [ x1 x2 x3 …. Xn]

### ii. Right Eigenvector

The right eigenvector is represented in the form of a column vector which satisfies the following condition:

$$AXR=\lambda XR$$

Where

A is a given matrix of order n and λ be one of its eigenvalues.

XR is a column vector of a matrix. I,e.,

$$X_R = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_n \end{bmatrix}$$

- **Eigenvector Applications**

The important application of eigenvectors are as follows:

- Eigenvectors are used in Physics in simple mode of oscillation
- In Mathematics, eigenvector decomposition is widely used in order to solve the linear equation of first order, in ranking matrices, in differential calculus etc
- This concept is widely used in quantum mechanics
- It is applicable in almost all the branches of engineering

Python Code

```
import numpy as np

# Define your matrix
A = np.array([[2, -1, 0],
              [-1, 2, -1],
              [0, -1, 2]])

# Calculate eigenvalues and eigenvectors
eigenvalues, eigenvectors = np.linalg.eig(A)

print("Eigenvalues of A:")
print(eigenvalues)

print("\nEigenvectors of A:")
print(eigenvectors)
```

## Self-Check Sheet - 3: Apply Linear Algebra

Q1: What is a matrix?

Q4: What is a square matrix?

Q5: What is the transpose of a matrix?

Q7: What is the inverse of a matrix?

Q9: What is the condition for a matrix to be invertible?

Q10: What is the determinant of a matrix?

Q11: What does the determinant of a matrix tell us?

Q13: What is a vector?

Q16: What are eigenvalues?

Q17: What are eigenvectors?

## Answer Sheet - 3: Apply Linear Algebra

**Q1: What is a matrix?**

**A:** A matrix is a rectangular array of numbers arranged in rows and columns. It is commonly denoted by A and written as A=[$a_{ij}$] where $a_{ij}$ represents the element at row i and column j.

**Q4: What is a square matrix?**

**A:** A square matrix is a matrix with the same number of rows and columns, such as

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

**Q5: What is the transpose of a matrix?**

**A:** The transpose of a matrix A, where A is obtained by swapping its rows and columns. For example,

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \text{ Transpose of A is } \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

**Q7: What is the inverse of a matrix?**

**A:** The inverse of a matrix A denoted $A^{-1}$, is a matrix such that A$A^{-1}$=I, I is the identity matrix. Only square matrices with a non-zero determinant have inverses.

**Q9: What is the condition for a matrix to be invertible?**

**A:** A matrix is invertible if and only if its determinant is non-zero.

**Q10: What is the determinant of a matrix?**

**A:** The determinant is a scalar value that can be computed from a square matrix and provides information about the matrix's properties, such as whether it is invertible. For a 2x2 matrix

$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ then the determinant is det(A)=ad−bc.

**Q11: What does the determinant of a matrix tell us?**

**A:** The determinant gives several insights:

- If det(A)≠0, the matrix is invertible.
- If det(A)=0, the matrix is singular and not invertible.
- It indicates the scaling factor of the transformation represented by the matrix.

**Q13: What is a vector?**

**A:** A vector is a quantity that has both magnitude and direction, often represented as a column or row of numbers. For example,$V = \binom{1}{2}$ is 2 dimensional vector.

**Q16: What are eigenvalues?**

**A:** Eigenvalues are scalars associated with a square matrix A, which satisfy the equation Av=λv, where λ is the eigenvalue and v is the corresponding eigenvector. The eigenvalue represents the factor by which the eigenvector is stretched.

**Q17: What are eigenvectors?**

**A:** Eigenvectors are non-zero vectors that only change in magnitude, not direction, when a matrix is applied to them. They satisfy the equation Av=λv, where v is the eigenvector and λ is the eigenvalue.

## Task Sheet-3.1: Apply Linear Algebra

**Step**

**1. Matrix Operations**

Matrix operations form the backbone of many machine learning algorithms. Learners will:

- **Multiply matrices**.
- **Find the inverse of a matrix** (if it exists).
- **Transpose a matrix**.
- Perform **dot products** with vectors.

**Example Task**: Given the following matrices:

A=[1234],B=[5678]A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}

Compute the following:

1. A×BA \times B (Matrix multiplication)
2. A−1A^{-1} (Inverse of matrix A)
3. ATA^T (Transpose of matrix A)

**Steps**:

1. **Matrix multiplication**: Use the definition of matrix multiplication.
2. **Matrix inverse**: Use the formula for the inverse of a 2x2 matrix.
3. **Matrix transpose**: Switch rows and columns.

**2. Eigenvalues and Eigenvectors**

Eigenvalues and eigenvectors are essential in many machine learning algorithms like Principal Component Analysis (PCA). Learners will:

- Compute the **eigenvalues** and **eigenvectors** of a matrix.

**Example Task**: Given the matrix CC:

C=[4123]C = \begin{bmatrix} 4 & 1 \\ 2 & 3 \end{bmatrix}

Find the eigenvalues and eigenvectors of matrix CC.

**Steps**:

1. Compute the characteristic polynomial of the matrix.
2. Solve for the eigenvalues.
3. Find the corresponding eigenvectors for each eigenvalue.

**3. Solving Systems of Linear Equations**

In machine learning, solving systems of equations is common when working with optimization problems or linear regression. Learners will:

- Use **Gaussian elimination** or **matrix inversion** to solve systems of linear equations.

**Example Task**: Solve the following system of equations:
x+2y=53x+4y=11x + 2y = 5 \\ 3x + 4y = 11
**Steps**:

1. Write the system as a matrix equation A·X=BA \cdot X = B, where: A=[1234],X=[xy],B=[511]A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad X = \begin{bmatrix} x \\ y \end{bmatrix}, \quad B = \begin{bmatrix} 5 \\ 11 \end{bmatrix}
2. Solve for XX using matrix inversion: X=A−1·BX = A^{-1} \cdot B.

**4. Apply Linear Algebra Concepts to a Machine Learning Problem**
Linear algebra is frequently used in machine learning algorithms, especially in tasks like regression, classification, and neural networks. Learners will apply linear algebra techniques to understand how algorithms like **linear regression** work.
**Example Task**: Using the least squares method for linear regression, learners will:

1. Use linear algebra to solve for the best-fit line to a set of points.
2. Apply the formula θ=(XTX)−1XTy\theta = (X^T X)^{-1} X^T y, where XX is the matrix of input data, and yy is the vector of output values.

## Learning Outcome 4: Use Optimization Methods

| | |
|---|---|
| Assessment Criteria | 1. Objective function is applied<br>2. Likelihood function is applied<br>3. Error function is applied<br>4. Gradient Descent Algorithm and its variants are applied |
| Conditions and Resources | 1. Actual workplace or training environment<br>2. CBLM<br>3. Handouts<br>4. Laptop<br>5. Multimedia Projector<br>6. Paper, Pen, Pencil and Eraser<br>7. Internet Facilities<br>8. Whiteboard and Marker<br>9. Imaging Device (Digital camera, scanner etc.) |
| Contents | 1. Objective function<br>2. Likelihood function<br>3. Error/Loss function<br>4. Gradient Descent Algorithm and its variants |
| Activities/job/Task | 1. Use Optimization Methods |
| Training Methods | 1. Discussion<br>2. Presentation<br>3. Demonstration<br>4. Guided Practice<br>5. Individual Practice<br>6. Project Work<br>7. Problem Solving<br>8. Brainstorming |
| Assessment Methods | 1. Written Test<br>2. Demonstration<br>3. Oral Questioning |

## Learning Experience 4: Use Optimization Methods

In order to achieve the objectives stated in this learning guide, you must perform the learning steps below. Beside each step are the resources or special instructions you will use to accomplish the corresponding activity.

| Learning Activities | Recourses/Special Instructions |
|---|---|
| 1. Trainee will ask the instructor about the learning materials | 1. Instructor will provide the learning materials 'Apply Statistical measures' |
| 2. Read the Information sheet and complete the Self Checks & Check answer sheets on "Apply Statistical measures" | 2. Read Information sheet 2: Use Multivariable Calculus<br>3. Answer Self-check 2: Use Multivariable Calculus<br>4. Check your answer with Answer key 2: Use Multivariable Calculus |
| 3. Read the Job/Task Sheet and Specification Sheet and perform job/Task | 5. Job/Task Sheet and Specification Sheet<br>Task Sheet-4.1: Use Optimization Methods |

## Information Sheet 4: Use Optimization Methods

**Learning Objective:**

After completion of this information sheet, the learners will be able to explain, define and interpret the following contents:

1. Objective function
2. Likelihood function
3. Error function
4. Gradient Descent Algorithm

## 1. Objective Function:

An objective function, also known as a loss function or cost function, is a mathematical function that quantifies the performance of a model in terms of its ability to achieve its goal or objective. In machine learning and optimization problems, the objective function is typically defined based on the specific task at hand and is used to guide the learning or optimization process towards the desired outcome.

Example: Let's consider a simple example of linear regression, where the goal is to fit a linear model to a set of data points. In this case, the objective function is often defined as the mean squared error (MSE) between the predicted values of the model and the actual values in the training data.

Given a dataset of $N$ data points $\{(x(i), y(i))\}$, where $x(i)$ represents the input features and $y(i)$ represents the corresponding target values, and a linear model parameterized by weights w and bias $b$, the objective function $J(w,b)$ for linear regression can be defined as:

$$J(w,b)=\sum_{i=1}^{N} (y(i) - (wTx(i) + b))^2$$

Where:

- $w^T$ represents the transpose of the weight vector w.
- x($i$) represents the feature vector of the $i$-th data point.
- $b$ is the bias term.
- The term $(wTx(i) + b))^2$ represents the predicted value of the linear model for the $i$-th data point. $(y(i) - (wTx(i) + b))^2$ represents the squared difference between the predicted value and the actual value for the $i$-th data point.
- $\sum_{i=1}^{N}$ represents the average of the squared differences over all

- $N$ data points.

The objective function $J(w,b)$ measures the average squared deviation between the predicted values of the linear model and the actual target values in the training data. The goal of the learning process is to minimize this objective function by adjusting the model parameters w and $b$ using optimization algorithms such as gradient descent.

In summary, the objective function quantifies the performance of a model in terms of its ability to fit the training data, and minimizing this function leads to better model performance.

## 2. Likelihood function

The likelihood function is a fundamental concept in statistics and machine learning, particularly in the context of parameter estimation. It quantifies the probability of observing a given set of data under a particular statistical model, as a function of the model parameters.
For linear regression, the likelihood function is derived from the assumption that the target variable $y$ follows a Gaussian (normal) distribution with mean $\mu$ and constant variance $\sigma^2$.
Given a set of observed data points $(X,y)$, the likelihood function represents the probability of observing the target values $y$ given the input features $X$ and the model parameters $\theta=(\mathbf{w},b)$, where $\mathbf{w}$ is the weight vector and $b$ is the bias term.

$L(\theta|X,y)$ for linear regression is defined as the product of the probabilities of observing each target value $yi$ given the corresponding input features $x_i$and the model parameters $\theta$:

$$L(\theta \mid X, y) = \prod i = 1N2\pi\sigma21exp(-2\sigma2(yi - (\boldsymbol{w}T\boldsymbol{x}i + b))2)$$

Python Code

```
import numpy as np

def likelihood(X, y, w, b, sigma_sq):
    N = len(y)
    y_pred = X.dot(w) + b
    error = y - y_pred
    likelihoods = (1 / np.sqrt(2 * np.pi * sigma_sq)) * np.exp(-0.5 * (error ** 2) / sigma_sq)
    return np.prod(likelihoods)

# Example usage:
```

```
X = np.array([[1, 2], [2, 3], [3, 4]])  # Input features (3 data points, 2 features each)
y = np.array([3, 4, 5])  # Target values
w = np.array([0.5, 0.5])  # Weight vector
b = 1  # Bias term
sigma_sq = 0.1  # Variance

likelihood_value = likelihood(X, y, w, b, sigma_sq)
print("Likelihood:", likelihood_value)
```

In this code:

- We define a function likelihood() that takes input features *X*, target values *y*, weight vector **w**, bias term *b*, and variance $\sigma^2$ as input and computes the likelihood function.
- We calculate the predicted target values $\hat{y}$ using the linear equation $\hat{y}$ ==**Xw**+*b*.
- We compute the likelihood of observing each target value *yi* given the corresponding predicted value $\hat{y}$ and the variance $\sigma^2$
- We return the product of all likelihoods as the final likelihood value.

This code demonstrates how to compute the likelihood function for linear regression given a set of observed data points and model parameters.

### 3. Error Function

In the context of machine learning, the error function, often referred to as the loss function or cost function, quantifies the difference between the predicted output of a model and the actual target values. The goal is to minimize this error function during the training process to improve the model's performance. A common error function used in regression tasks is the Mean Squared Error (MSE) function.

The MSE function measures the average of the squares of the differences between the predicted values ( $\hat{y}$) and the actual target values (*y*) across all data points. Mathematically, it is expressed as:

$$\text{MSE}=\frac{1}{N}\sum_{i=1}^{N} \quad (yi - y\text{^}i)^2$$

Where:

- N is the number of data points.
- Yi is the actual target value for the i-th data point.
- $y\text{^}i$ is the predicted value for the i-th data point.

- Python Code:

```
import numpy as np

def mean_squared_error(y_true, y_pred):
    """
    Compute the Mean Squared Error (MSE) between the true and predicted values.

    Args:
        y_true (numpy.ndarray): Array of true target values.
        y_pred (numpy.ndarray): Array of predicted target values.

    Returns:
        float: Mean Squared Error.
    """
    N = len(y_true)
    mse = np.mean((y_true - y_pred) ** 2)
    return mse

# Example usage:
y_true = np.array([3, 4, 5])  # True target values
y_pred = np.array([2.8, 4.2, 5.1])  # Predicted values

mse = mean_squared_error(y_true, y_pred)
print("Mean Squared Error:", mse)
```

In this code:

- We define a function mean_squared_error() that takes the true target values (y_true) and the predicted target values (y_pred) as input and computes the mean squared error.
- We calculate the squared differences between each pair of true and predicted values, compute their mean, and return the result as the Mean Squared Error.

This code demonstrates how to implement the Mean Squared Error function in Python, which is commonly used as an error function in regression tasks in machine learning

## 4. Gradient Descent Algorithm

**Gradient descent** (GD) is an iterative first-order optimisation algorithm, used to find a local minimum/maximum of a given function. This method is commonly used in *machine learning* (ML) and *deep learning* (DL) to minimise a cost/loss function (e.g. in a linear regression). Due to its importance and ease of implementation, this algorithm is usually taught at the beginning of almost all machine learning courses.

However, its use is not limited to ML/DL only, it's widely used also in areas like:

- control engineering (robotics, chemical, etc.)
- computer games
- mechanical engineering

That's why today, we will do a deep dive into the math, implementation and behaviour of first-order gradient descent algorithm. We will navigate the custom (cost) function directly to find its minimum. That means there will be no underlying data like in typical ML tutorials — we will be more flexible regarding a function's shape.

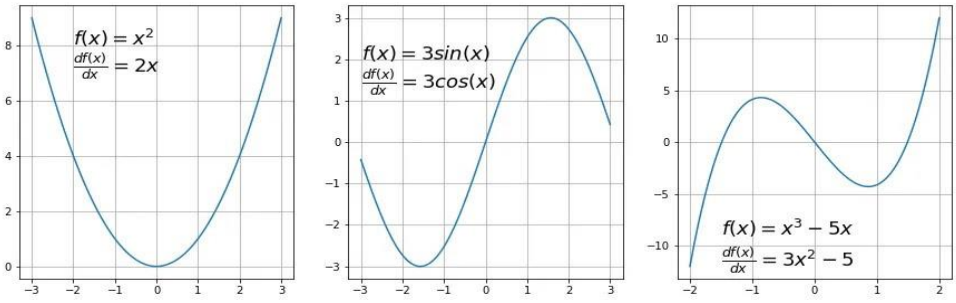
This method was proposed long before the era of modern computers by Augustin-Louis Cauchy in 1847. Since that time, there was an significant development in computer science and numerical methods. That led to numerous improved versions of Gradient Descent. However, in this article we're going to use a basic/vanilla version implemented in Python.

- **Function requirements**

Gradient descent algorithm does not work for all functions. There are two specific requirements. A function has to be:

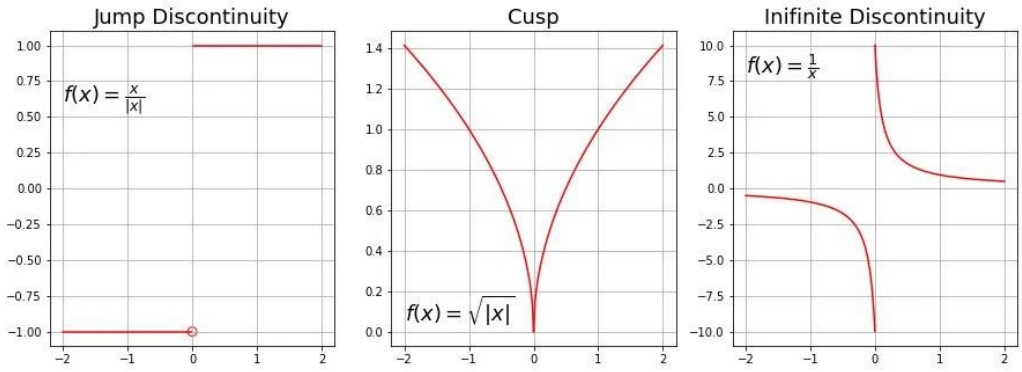
- **Differentiable**
- **Convex**

First, what does it mean it has to be **differentiable**? If a function is differentiable it has a derivative for each point in its domain — not all functions meet these criteria. First, let's see some examples of functions meeting this criterion:



Example of Differentiable Function

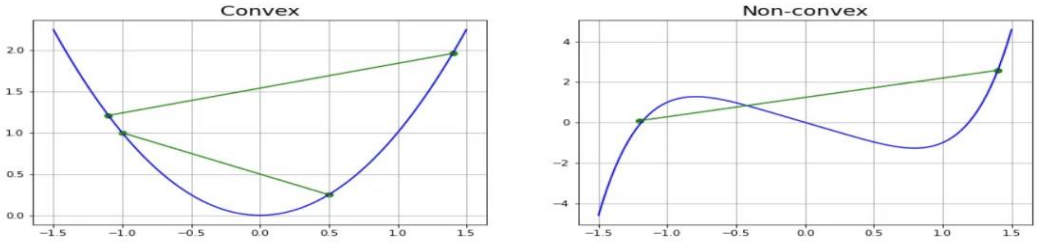
Typical non-differentiable functions have a step a cusp or a discontinuity:




Next requirement — function has to be convex. For a univariate function, this means that the line segment connecting two function's points lays on or above its curve (it does not cross it). If it does it means that it has a local minimum which is not a global one.
Mathematically, for two points $x_1$, $x_2$ laying on the function's curve this condition is expressed as:

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2)$$

where λ denotes a point's location on a section line and its value has to be between 0 (left point) and 1 (right point), e.g. λ=0.5 means a location in the middle.Below there are two functions with exemplary section lines.



Another way to check mathematically if a univariate function is convex is to calculate the second derivative and check if its value is always bigger than 0.

$$\frac{d^2 f(x)}{dx^2} > 0$$

Let's do a simple example .Let's investigate a simple quadratic function given by:

$$f(x) = x^2 - x + 3$$

Its first and second derivative are:

$$\frac{df(x)}{dx} = 2x - 1, \quad \frac{d^2 f(x)}{dx^2} = 2$$

Because the second derivative is always bigger than 0, our function is strictly convex.It is also possible to use quasi-convex functions with a gradient descent algorithm. However, often they have so-called saddle points (called also *minimax* points) where the algorithm can get stuck (we will demonstrate it later in the article). An example of a quasi-convex function is:

$$f(x) = x^4 - 2x^3 + 2$$

$$\frac{df(x)}{dx} = 4x^3 - 6x^2 = x^2(4x - 6)$$

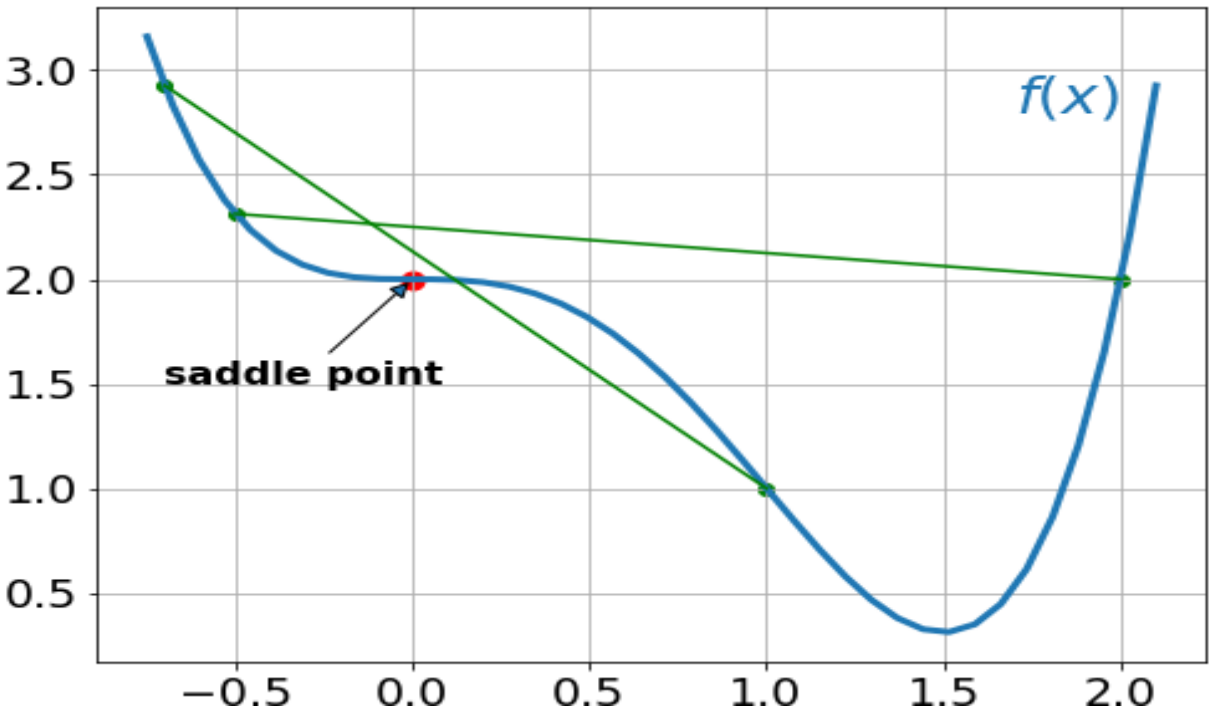
Let's stop here for a moment. We see that the first derivative equal zero at x=0 and x=1.5. This places are candidates for the function's extrema (minimum or maximum )— the slope is zero there. But first we have to check the second derivative first.

$$\frac{d^2 f(x)}{dx^2} = 12x^2 + 12x = 12x(x - 1)$$

The value of this expression is zero for x=0 and x=1. These locations are called an inflexion point — a place where the curvature changes sign — meaning it changes from convex to concave or vice-versa. By analyzing this equation we conclude that :

- for x<0: function is convex
- for 0<x<1: function is concave (the 2nd derivative < 0)
- for x>1: function is convex again

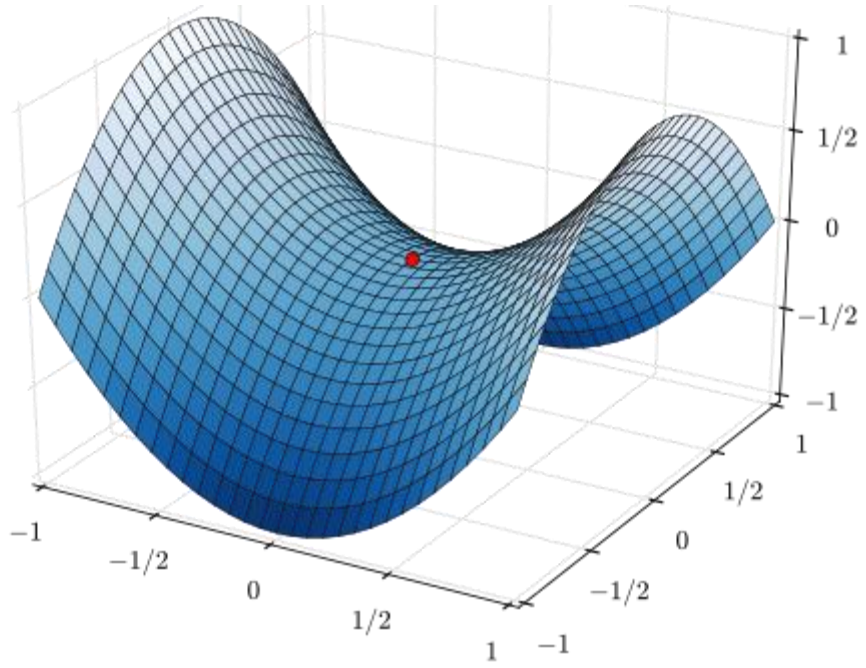
Now we see that point x=0 has both first and second derivative equal to zero meaning this is a saddle point and point x=1.5 is a global minimum.
Let's look at the graph of this function. As calculated before a saddle point is at x=0 and minimum at x=1.5.




Semi-convex function with a saddle point

For multivariate functions the most appropriate check if a point is a saddle point is to calculate a Hessian matrix which involves a bit more complex calculations and is beyond the scope of this article.
Example of a saddle point in a bivariate function is show below.

$$z = x^2 - y^2$$




Nicoguaro, CC BY 3.0, via Wikimedia Commons

- **Gradient**

Before jumping into code one more thing has to be explained — what is a gradient. Intuitively it is a slope of a curve at a given point in a specified direction.

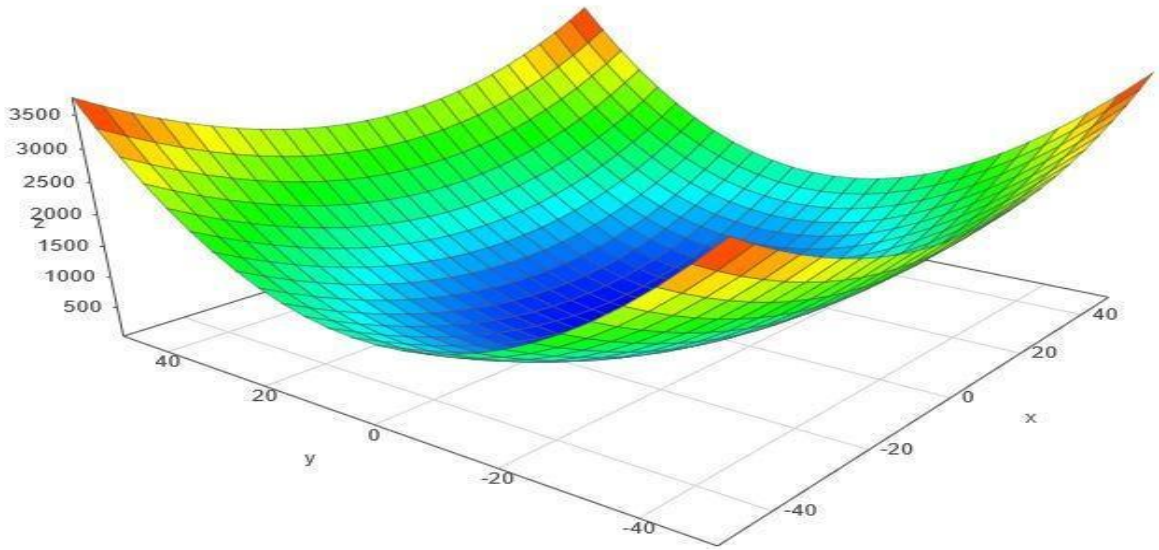
In the case of a univariate function, it is simply the first derivative at a selected point. In the case of a multivariate function, it is a vector of derivatives in each main direction (along variable axes).

Because we are interested only in a slope along one axis and we don't care about others these derivatives are called partial derivatives.
A gradient for an n-dimensional function f(x) at a given point p is defined as follows:

$$\nabla f(p) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(p) \\ \vdots \\ \frac{\partial f}{\partial x_n}(p) \end{bmatrix}$$

The upside-down triangle is a so-called nabla symbol and you read it "del". To better understand how to calculate it let's do a hand calculation for an exemplary 2-dimensional function below.

$$f(x) = 0.5x^2 + y^2$$




3D plot; Image by author

Let's assume we are interested in a gradient at point p(10,10):

$$\frac{\partial f(x,y)}{\partial x} = x, \quad \frac{\partial f(x,y)}{\partial y} = 2y$$

so consequently:

$$\nabla f(x,y) = \begin{bmatrix} x \\ 2y \end{bmatrix}$$

$$\nabla f(10,10) = \begin{bmatrix} 10 \\ 20 \end{bmatrix}$$

By looking at these values we conclude that the slope is twice steeper along the y axis.

- **Gradient Descent Algorithm**

Gradient Descent Algorithm iteratively calculates the next point using gradient at the current position, scales it (by a learning rate) and subtracts obtained value from the current position (makes a step). It subtracts the value because we want to minimise the function (to maximise it would be adding). This process can be written as:

$$p_{n+1} = p_n - \eta \nabla f(p_n)$$

There's an important parameter η which scales the gradient and thus controls the step size. In machine learning, it is called learning rate and have a strong influence on performance.

- The smaller learning rate the longer GD converges, or may reach maximum iteration before reaching the optimum point
- If learning rate is too big the algorithm may not converge to the optimal point (jump around) or even to diverge completely.

In summary, Gradient Descent method's steps are:

- choose a starting point (initialisation)
- calculate gradient at this point
- make a scaled step in the opposite direction to the gradient (objective: minimise)
- repeat points 2 and 3 until one of the criteria is met:
- maximum number of iterations reached
- step size is smaller than the tolerance (due to scaling or a small gradient).

Below, there's an exemplary implementation of the Gradient Descent algorithm (with steps tracking):

```
import numpy as np
from typing import Callable


def gradient_descent(start: float, gradient: Callable[[float], float],
                     learn_rate: float, max_iter: int, tol: float = 0.01):
    x = start
    steps = [start]  # history tracking

    for _ in range(max_iter):
        diff = learn_rate*gradient(x)
        if np.abs(diff) < tol:
            break
        x = x - diff
        steps.append(x)  # history tracing

    return steps, x
```

This function takes 5 parameters:

- starting point [float] - in our case, we define it manually but in practice, it is often a random initialisation
- gradient function [object] - function calculating gradient which has to be specified before-hand and passed to the GD function
- learning rate [float] - scaling factor for step sizes
- maximum number of iterations [int]
- tolerance [float] to conditionally stop the algorithm (in this case a default value is 0.01)

Example — a quadratic function

Let's take a simple quadratic function defined as:

$$f(x) = x^2 - 4x + 1$$

Because it is an univariate function a gradient function is:

$$\frac{df(x)}{dx} = 2x - 4$$

```
def func1(x:float):
    return x**2-4*x+1

def gradient_func1(x:float):
    return 2*x - 4
```

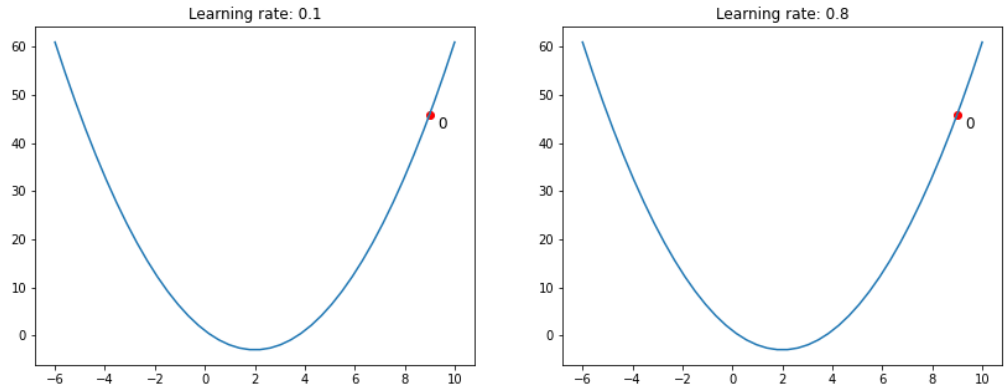
For this function, by taking a learning rate of 0.1 and starting point at x=9 we can easily calculate each step by hand. Let's do it for the first 3 steps:

$$\begin{aligned} x_1 &= 9 - 0.1 \cdot (2 \cdot 9 - 4) = 7.6 \\ x_2 &= 7.6 - 0.1 \cdot (2 \cdot 7.6 - 4) = 6.48 \\ x_3 &= 6.48 - 0.1 \cdot (2 \cdot 6.48 - 4) = 5.584 \end{aligned}$$

The Python code is called

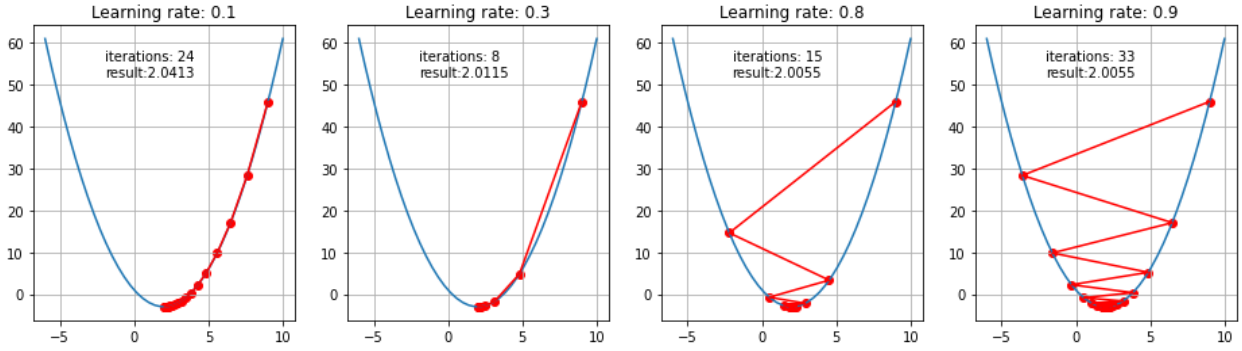
```
history, result = gradient_descent(9, gradient_func1, 0.1, 100)
```

The animation below shows steps taken by the GD algorithm for learning rates of 0.1 and 0.8. As you see, for the smaller learning rate, as the algorithm approaches the minimum the steps are getting gradually smaller. For a bigger learning rate, it is jumping from one side to another before converging.




First 10 steps taken by GD for small and big learning rate; Image by author

Trajectories, number of iterations and the final converged result (within tolerance) for various learning rates are shown below:



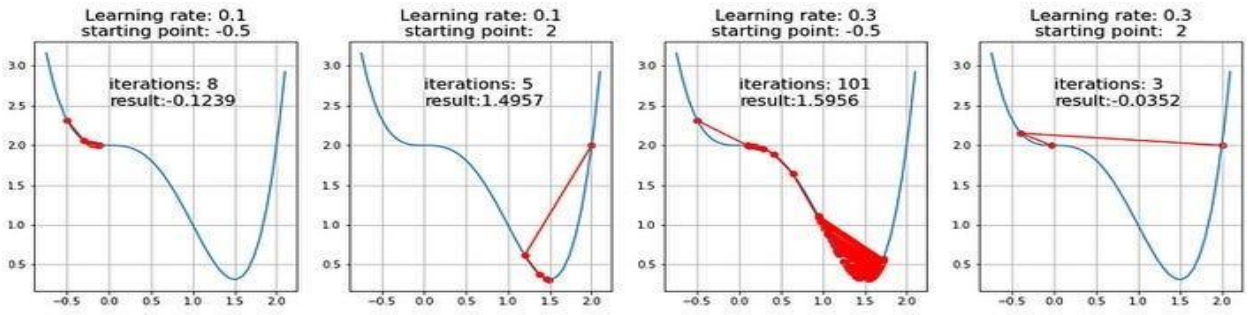

Results for various learning rates; Image by author

Example — a function with a saddle point

Now let's see how the algorithm will cope with a semi-convex function we investigated mathematically before.

$$f(x) = x^4 - 2x^3 + 2$$

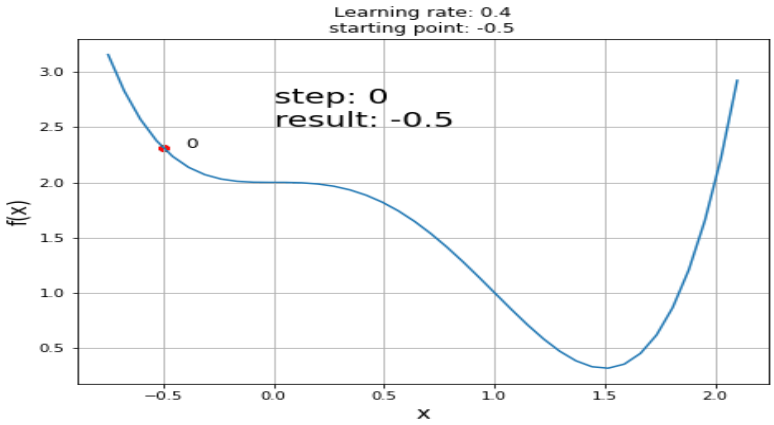
Below results for two learning rates and two different staring points.




GD trying to escape from a saddle point; Image by author

Below an animation for a learning rate of 0.4 and a starting point *x=-0.5*.

Now, you see that an existence of a saddle point imposes a real challenge for the first-order gradient descent algorithms like GD, and obtaining a global minimum is not guaranteed. Second-order algorithms deal with these situations better (e.g. Newton-Raphson method).




Animation of GD trying to escape from a saddle point; Image by author

Investigation of saddle points and how to escape from them is a subject of ongoing studies and various solutions were proposed.

## Self-Check Sheet - 4: Use Optimization Methods

Q1: What is an objective function?

Q2: What is the role of the objective function in machine learning?

Q3: What is the difference between a cost function and an objective function?

Q4: What is a likelihood function?

Q5: How does the likelihood function differ from the probability function?

Q6: What is Maximum Likelihood Estimation (MLE)?

Q7: How is the likelihood function used in logistic regression?

Q8: What is a loss function in machine learning?

Q9: What are common types of loss functions?

Q10: Why is the loss function minimized?

Q11: What is the learning rate in Gradient Descent?

Q12: What are the variants of Gradient Descent?

Q13: What is the difference between Batch Gradient Descent and Stochastic Gradient Descent?

Q14: What is the role of the gradient in Gradient Descent?

## Answer Sheet - 4: Apply Statistical Measures

### Q1: What is an objective function?

**A:** An objective function is a mathematical expression that defines the goal of an optimization problem. In machine learning, it is the function that the model tries to minimize (in case of loss) or maximize (in case of profit or likelihood).

### Q2: What is the role of the objective function in machine learning?

**A:** The objective function guides the learning process by quantifying the difference between predicted and actual values. It is minimized during training to improve the model's performance.

### Q3: What is the difference between a cost function and an objective function?

**A:** In many contexts, the terms are used interchangeably. However, an objective function refers to any function that we are trying to minimize or maximize, while a cost function specifically measures the error between predicted and true values and is typically minimized.

### Q4: What is a likelihood function?

**A:** A likelihood function measures the probability of observing the given data under specific model parameters. In statistical models, the likelihood is used to estimate the most probable parameters given the data.

### Q5: How does the likelihood function differ from the probability function?

**A:** A probability function calculates the probability of an outcome given fixed parameters. In contrast, a likelihood function treats the observed data as fixed and varies the parameters to maximize the likelihood of observing that data.

### Q6: What is Maximum Likelihood Estimation (MLE)?

**A:** Maximum Likelihood Estimation is a method for estimating the parameters of a statistical model by maximizing the likelihood function. The parameters that maximize the likelihood are considered the best fit for the data.

**Q7: How is the likelihood function used in logistic regression?**

**A:** In logistic regression, the likelihood function is used to estimate the probability that a given input belongs to a particular class. MLE is applied to find the model parameters that maximize the probability of the observed class labels.

**Q8: What is a loss function in machine learning?**

**A:** A loss function quantifies the difference between the predicted output of a model and the actual output. The goal of training a model is to minimize this loss function to improve accuracy.

**Q9: What are common types of loss functions?**
**A:** Common loss functions include:

- **Mean Squared Error (MSE)** for regression tasks.
- **Cross-Entropy Loss** for classification tasks.
- **Hinge Loss** for support vector machines.

**Q10: Why is the loss function minimized?**
**A:** Minimizing the loss function ensures that the model's predictions get closer to the actual values, leading to a more accurate model. This optimization helps the model generalize well on unseen data.

**Q11: What is the learning rate in Gradient Descent?**

**A:** The learning rate is a hyperparameter that controls the step size during the update process. A high learning rate may cause the algorithm to overshoot the minimum, while a low learning rate may lead to slow convergence.

**Q12: What are the variants of Gradient Descent?**
**A:**

1. **Batch Gradient Descent**: Uses the entire dataset to compute the gradient at each step. It converges smoothly but can be slow for large datasets.
2. **Stochastic Gradient Descent (SGD)**: Updates the parameters for each training example, leading to faster but noisier updates.
3. **Mini-Batch Gradient Descent**: Combines the advantages of both methods by updating parameters based on a small batch of data at each step.

**Q13: What is the difference between Batch Gradient Descent and Stochastic Gradient Descent?**

**A:** Batch Gradient Descent uses all data points to compute the gradient, leading to a more stable but slower convergence. Stochastic Gradient Descent, on the other hand, updates parameters after evaluating each data point, leading to faster but noisier convergence.

**Q14: What is the role of the gradient in Gradient Descent?**

**A:** The gradient of the objective function points in the direction of the steepest ascent. In Gradient Descent, the parameters are updated in the opposite direction (the direction of steepest descent) to minimize the objective function.

## Task Sheet-4.1: Use Optimization Methods

**Step**

### 1. Introduction to Optimization in Machine Learning

- **Optimization** in machine learning refers to the process of finding the best parameters (such as weights and biases in a neural network) that minimize (or maximize) a given objective function (like the loss function).
- **Objective Function**: This is typically a **loss function** in machine learning that quantifies how well a model is performing.

### 2. Gradient Descent Method

The **Gradient Descent (GD)** algorithm is a popular optimization method for minimizing the objective function in machine learning.

- **Steps**:
    1. Initialize the model parameters (weights) randomly or with some initial values.
    2. Compute the gradient (derivative) of the loss function with respect to the parameters.
    3. Update the parameters in the opposite direction of the gradient to reduce the loss function.
    4. Repeat the steps until convergence or a predefined number of iterations.
- **Example Task**: For a simple linear regression model, minimize the **Mean Squared Error (MSE)**:

MSE=1N∑i=1N(yi−(θ0+θ1xi))2MSE = \frac{1}{N} \sum_{i=1}^{N} (y_i - (\theta_0 + \theta_1 x_i))^2

Use Gradient Descent to update the parameters θ0\theta_0 and θ1\theta_1 iteratively.

**Steps**:

1. Define the loss function.
2. Compute the gradients of θ0\theta_0 and θ1\theta_1.
3. Update the parameters using the learning rate α\alpha:

θj=θj−α∂J(θ)∂θj\theta_j = \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j}

### 3. Stochastic Gradient Descent (SGD)

**Stochastic Gradient Descent (SGD)** is a variant of gradient descent that updates the model parameters based on a single data point (or a small batch of data points) rather than the entire dataset.

- **Steps**:
    1. Randomly select a data point.
    2. Compute the gradient based on that single data point.
    3. Update the parameters using the gradient and learning rate.
    4. Repeat the process for multiple iterations.
- **Advantages**:
    - Faster computation, especially for large datasets.
    - Often helps escape local minima due to noisy updates.
- **Example Task**: Train a simple linear regression model using SGD. Update the parameters based on one data point at a time.

**4. Newton's Method**

**Newton's Method** is a second-order optimization method that uses both the gradient and the second derivative (Hessian) to find the optimal solution more efficiently.

- **Steps**:
    1. Compute the gradient ∇f(x)\nabla f(x).
    2. Compute the Hessian matrix H(x)H(x) (second-order partial derivatives).
    3. Update the parameters using the formula:

xnew=x−H−1∇f(x)x_{new} = x - H^{-1} \nabla f(x)

    4. Repeat the process until convergence.
- **Advantages**:
    - Converges faster than gradient descent if the Hessian is easy to compute.
    - Can be more efficient for certain problems, especially with quadratic objective functions.

**5. Compare Optimization Methods**

In this step, learners will compare the performance of different optimization methods (Gradient Descent, Stochastic Gradient Descent, and Newton's Method) on the same problem. They will analyze:

- **Convergence speed** (how fast the methods approach the minimum).
- **Accuracy** (how close the methods get to the optimal solution).
- **Computational cost** (how many iterations are required to converge).

# Reference


1. Mathematics for Machine Learning, Author: Marc Peter Deisenroth, A. Aldo Faisal, Cheng Soon Ong (2020)
2. Deep Learning: A Practitioner's Approach, Author: Adam Gibson, Josh Patterson (2017)
3. Mathematics for 3D Game Programming and Computer Graphics. Author: Eric Lengyel (2011)
4. Mathematics of Continuous Media: A Handbook for Engineers and Scientists, Author: E.J. Hinch (2018)
5. Introduction to Machine Learning with Python: A Guide for Data Scientists, Author: Andreas C. Müller, Sarah Guido (2016)
6. AI Superpowers: China, Silicon Valley, and the New World Order, Author: Kai-Fu Lee (2018)

NB: After completion of all LO, then complete the following review of competency

## Review of Competency

Below is yourself assessment rating for module "Apply Math Skills for Machine Learning

| Assessment of performance Criteria | Yes | No |
|---|---|---|
| 1. Measures of central tendency | ☐ | ☐ |
| 2. Random variable and probability distribution | ☐ | ☐ |
| 3. Hypothesis Testing | ☐ | ☐ |
| 4. Multi variable Functions are exercised | ☐ | ☐ |
| 5. Derivatives and gradients are exercised | ☐ | ☐ |
| 6. Activation functions | ☐ | ☐ |
| 7. Cost function is exercised | ☐ | ☐ |
| 8. Plotting of functions are applied | ☐ | ☐ |
| 9. Minimum and Maximum values of a function are determined | ☐ | ☐ |
| 10. Matrices are exercised | ☐ | ☐ |
| 11. Vectors are exercised | ☐ | ☐ |
| 12. Objective function is applied | ☐ | ☐ |
| 13. Likelihood function is applied | ☐ | ☐ |
| 14. Error function is applied | ☐ | ☐ |
| 15. Gradient Descent Algorithm and its variants are applied | ☐ | ☐ |

I now feel ready to undertake my formal competency assessment.

Signed:

Date:

# Development of CBLM

The Competency based Learning Material (CBLM) of 'Apply Math Skills for Machine Learning' (Occupation: AI in Immersive Technology) for National Skills Certificate is developed by NSDA with the assistance of SAMAHAR Consultants Ltd.in the month of June, 2024 under the contract number of package SD-9C dated 15th January 2024.

| SL No. | Name and Address | Designation | Contact Number |
|---|---|---|---|
| 1 | A K M Mashuqur Rahman Mazumder | Writer | Cell: 01676323576<br>Email : mashuq.odelltech@odell.com.bd |
| 2 | Nafija Arbe | Editor | Cell: 01310568900<br>nafija.odelltech@odell.com.bd |
| 3 | Khan Mohammad Mahmud Hasan | Co-Ordinator | Cell: 01714087897<br>Email: kmmhasan@gmail.com |
| 4 | Md. Saif Uddin | Reviewer | Cell: 01723004419<br>Email:engrbd.saif@gmail.com |