Machine Learning

Category	
Files	
Created	@February 20, 2023 11:17 AM
Reminder	
Status	Open
URL	
Updated	@May 19, 2023 12:51 PM

▼ Syllabus -

CEN- 809: MACHINE LEARNING TECHNIQUES

L	T	P		
3	1	0	Internal:	40 Marks
			External:	60 Marks
Credits : 4			**Total:**	**100 Marks**
Duration of Exam : 3 Hours				

Course Outcomes:

1. Students will be able to understand the concept of machine learning.
2. Students will be able to analyze different types of regression techniques and its applications
3. Students will be able to apply different types of dimensionality reduction techniques used in machine learning
4. Students will be able to understand about the Artificial neural network and its uses in various field
5. Students will be able to understand about the deep learning and its component and their implementation to solve various types of problems

UNIT I

Introduction of Machine Learning, AI, ML and DL, Types of machine learning, Instance based learning, model-based learning, challenges in Machine Learning, Application of ML, ML development life cycle (MLDLC), fundamental of machine learning, univariate analysis, bivariate analysis, Multivariate Analysis

UNIT II

Linear Regression, Regression Metrics, MAE, MSE, RMSE, R squared , Adjusted R squared, Multiple Regression, Gradient descent, Batch Gradient , Stochastic Gradient , Mini Gradient, Polynomial Regression, Bias, variance, regularization techniques, Regularization Methods, Lasso Regression, Ridge Regression, Elastic Net Regression, Logistic regression, classification evaluation metrics, Accuracy, confusion matrix, Precision, Recall, F1-Score, macro and weighted F1-Score, SoftMax regression or multinomial logistic regression, polynomial logistic regression

UNIT III

Dimensionality reduction, Subset selection, Forward selection, Backward selection, Principal component analysis, Linear Discriminant Analysis, Fisher's criterion, t-Distributed Stochastic Neighbor Embedding (t-SNE), Kullback–Leibler divergence

UNIT IV

Introduction to Deep Learning, Types of Neural Network, McCulloch-Pitts Neuron Model, Boolean Functions Using M-P Neuron, The Perceptron, Logistic regression using perceptron , Activation Functions,

Multilayer Perceptron(MLP), Multilayer Forward propagation , Back Propagation: Input, output and hidden layer computation, Memorization

UNIT V

Gradient descent in deep learning, Vanishing gradient problem, Early stopping, dropout layer, regularization in deep learning, CNN, convolution, padding & strides, pooling, backpropagation in CNN, Recurrent neural network(RNN), types of RNN

Unit 1 -

What is machine learning

Machine learning is a subfield of artificial intelligence (AI) that involves building algorithms and models that enable computer systems to learn and improve their performance based on data inputs, without being explicitly programmed. Machine learning algorithms are designed to automatically identify patterns and relationships in large and complex datasets, and to use this information to make predictions, recommendations, or decisions.

The process of machine learning typically involves the following steps:

1. Data preparation: Collecting, cleaning, and organizing data to ensure it is relevant and usable for training the algorithm.
2. Model building: Selecting an appropriate machine learning algorithm and configuring it to achieve the desired outcome.
3. Training: Feeding the algorithm with the prepared data to enable it to learn from the patterns and relationships in the data.
4. Testing: Evaluating the performance of the algorithm using a separate set of data.
5. Deployment: Implementing the algorithm in a real-world application.

Machine learning has a wide range of applications in various fields, including image and speech recognition, natural language processing, recommendation systems, fraud detection, predictive maintenance, and autonomous vehicles, to name a few.

AI vs ML vs Dl

AI (Artificial Intelligence), ML (Machine Learning), and DL (Deep Learning) are related terms but they are not interchangeable.

AI refers to the development of computer systems that can perform tasks that usually require human intelligence, such as visual perception, speech recognition, decision-making, and language translation.

ML is a subset of AI that involves building systems that can learn from data, identify patterns, and make decisions with minimal human intervention.

DL is a subset of ML that uses neural networks with multiple layers to analyze large and complex datasets, and to extract features and patterns that are difficult to discern with traditional ML algorithms.

DL has achieved remarkable results in various fields, including computer vision, natural language processing, and game playing. However, it requires large amounts of labeled data, powerful hardware, and specialized expertise to train and deploy DL models.

In summary, AI is the broadest term that encompasses ML and DL, while ML is a more specific term that includes traditional ML methods and DL. DL is a

cutting-edge and powerful technique that has revolutionized many AI applications, but it is not always the best solution for every problem.

▼ Types of machine learning

there are three main types of machine learning -

1. **Supervised learning**: This type of machine learning involves training a model on labeled data, meaning that the correct output is already known. The model learns to generalize from the examples and can predict the correct output for new, unseen data. An example of supervised learning is image classification, where the algorithm is trained on a labeled dataset of images and their corresponding categories (e.g. cat, dog, bird). The algorithm learns to associate features in the images with their respective categories, and can then classify new images it has never seen before.
2. **Unsupervised learning**: This type of machine learning involves training a model on unlabeled data, meaning that there is no correct output. The model learns to find patterns and relationships in the data, and can be used for clustering, association, and dimensionality reduction tasks. A common example of unsupervised learning is clustering, where the algorithm groups similar data points together based on their proximity in feature space. For example, an e-commerce website might use clustering to group customers into segments based on their purchasing behavior, so that it can tailor its marketing campaigns to each segment.
3. **Reinforcement learning**: This type of machine learning involves training a model to make decisions by interacting with an environment, receiving rewards or penalties based on its actions, and adjusting its behavior to maximize the reward. Reinforcement learning is used for optimization and control tasks, such as game playing and robotics. An example of reinforcement learning is training a robot to navigate a maze, where the robot receives a reward for reaching the end of the maze and a penalty for hitting a wall. The robot learns to navigate the maze by adjusting its movements based on the rewards and penalties it receives.

I hope these examples help clarify the different types of machine learning! Let me know if you have any further questions.

▼ Instance Based Learning

Instance-based learning, also known as lazy learning, is a type of machine learning that involves storing and retrieving instances of training data to make predictions. In this approach, the model does not explicitly learn a general function that maps inputs to outputs, but rather stores the training instances in memory and compares them to new instances to find the most similar ones. The output for the new instance is then based on the outputs of the stored instances that are most similar.

Instance-based learning is useful in situations where the relationship between inputs and outputs is complex and difficult to model explicitly. It is also well-suited for problems with a large number of variables, as it

can handle high-dimensional data without the need for feature selection or extraction.

An example of instance-based learning is the k-nearest neighbor (k-NN) algorithm, which finds the k training instances that are closest to a new instance in terms of some distance metric (e.g. Euclidean distance). The output for the new instance is then based on the outputs of the k nearest neighbors, such as the majority class in a classification problem or the mean value in a regression problem.

While instance-based learning can be effective in certain situations, it has some drawbacks. It can be computationally expensive, especially when dealing with large datasets or high-dimensional data. It is also sensitive to the choice of distance metric and the value of k, which can affect the quality of the predictions.

▼ Model Based Learning

Model-based learning is a type of machine learning that involves building a model of the system or process that generates the data, and then using the model to make predictions or decisions. In this approach, the model is first trained on a set of input-output pairs, and then used to predict the output for new input values.

Model-based learning is useful in situations where the relationship between inputs and outputs is well understood and can be explicitly modeled. It is often used in engineering and physics-based applications, where the underlying principles governing the system are known.

An example of model-based learning is the linear regression algorithm, which models the relationship between a dependent variable and one or more independent variables using a linear function. The model is trained on a set of input-output pairs, and then used to predict the output for new input values.

While model-based learning can be effective in certain situations, it has some drawbacks. It assumes that the underlying model accurately captures the relationship between inputs and outputs, which may not always be the case. It can also be sensitive to outliers and noise in the data, which can affect the quality of the predictions.

▼ Challenges in Machine Learning

One of the biggest challenges in machine learning is the quality and quantity of data. Machine learning algorithms require large amounts of high-quality data to learn patterns and relationships, and to make accurate predictions. However, data can be noisy, incomplete, or biased, which can affect the performance of the algorithm.

Another challenge is selecting the right algorithm and model architecture for the problem at hand. Different algorithms and architectures have different strengths and weaknesses, and may perform better or worse depending on the specific dataset and task.

Machine learning models can also suffer from overfitting or underfitting, which can lead to poor performance on new, unseen data. Overfitting occurs when the model is too complex and learns the noise in the training data,

while underfitting occurs when the model is too simple and fails to capture the underlying patterns in the data.

Finally, machine learning models can be susceptible to adversarial attacks, where an attacker intentionally manipulates the input data to cause the model to make incorrect predictions. Adversarial attacks can have serious consequences in applications such as autonomous vehicles and cybersecurity, and are an active area of research in the field of machine learning.

▼ Application of Machine Learning

Machine learning has numerous applications in various fields, some of which are:

1. Image and speech recognition: ML algorithms can be used to recognize objects and patterns in images and speech, which is useful in applications such as self-driving cars, security surveillance, and virtual assistants.
2. Natural language processing: ML algorithms can be used to understand and generate human language, which is useful in applications such as chatbots, language translation, and voice assistants.
3. Recommendation systems: ML algorithms can be used to analyze user behavior and preferences, and make personalized recommendations for products, services, and content.
4. Fraud detection: ML algorithms can be used to detect fraudulent activity in financial transactions, insurance claims, and online transactions.
5. Predictive maintenance: ML algorithms can be used to predict when equipment or systems are likely to fail, enabling proactive maintenance and reducing downtime.
6. Autonomous vehicles: ML algorithms are used extensively in self-driving cars to analyze sensor data and make decisions about steering, acceleration, and braking.
7. Healthcare: ML algorithms can be used to analyze medical data and make predictions about patient outcomes, as well as to develop personalized treatment plans.

These are just a few examples of the many applications of machine learning. As the field continues to develop, it is likely that we will see even more innovative and impactful applications in the future.

▼ ML Development Life Cycle

The machine learning development life cycle consists of several stages, which are:

1. **Problem definition**: This stage involves identifying the problem to be solved and defining the scope of the project. It is important to clearly define the problem and the desired outcome, as well as any constraints or limitations that may affect the project.
2. **Data collection**: This stage involves collecting and acquiring the data that will be used to train and test the machine learning algorithm. It is important to ensure that the data is relevant, representative, and

of high quality, as this will have a significant impact on the performance of the algorithm.

3. **Data preparation**: This stage involves cleaning, transforming, and preparing the data for use in the machine learning algorithm. This may involve tasks such as removing duplicates, handling missing values, and normalizing or scaling the data.
4. **Model selection**: This stage involves selecting the appropriate machine learning algorithm and model architecture for the problem at hand. Different algorithms and architectures have different strengths and weaknesses, and may perform better or worse depending on the specific dataset and task.
5. **Model training**: This stage involves feeding the prepared data into the selected machine learning algorithm and training the model to make predictions. This may involve tuning hyperparameters, selecting regularization methods, and evaluating the performance of the model using various metrics.
6. **Model evaluation**: This stage involves testing the performance of the trained model on a separate set of data to evaluate its accuracy and generalization ability. This may involve using techniques such as cross-validation, holdout validation, or k-fold validation.
7. **Model deployment**: This stage involves deploying the trained model in a production environment to make predictions on new, unseen data. This may involve integrating the model into an existing system, creating a user interface for interacting with the model, or deploying the model to a cloud-based platform.
8. **Model monitoring**: This stage involves monitoring the performance of the deployed model over time and making updates or modifications as needed. This may involve retraining the model with new data, fine-tuning hyperparameters, or updating the model architecture.

By following a structured machine learning development life cycle, it is possible to build accurate and effective machine learning models that can be used to solve a wide range of problems in various fields.

▼ Univariate Analysis

Univariate analysis is a statistical method that involves analyzing a single variable at a time. The goal of univariate analysis is to describe and summarize the characteristics of the data, such as its central tendency, dispersion, and shape.

Some common techniques used in univariate analysis include:

1. Measures of central tendency: These include the mean, median, and mode, and are used to describe where the data is centered.
2. Measures of dispersion: These include the range, variance, and standard deviation, and are used to describe how spread out the data is.
3. Frequency distributions: These show the number or proportion of observations that fall into different categories or intervals.

4. Histograms: These are graphical representations of frequency distributions, where bars are used to represent the number or proportion of observations in each interval.
5. Box plots: These are graphical representations of the distribution of the data, showing the median, quartiles, and outliers.

Univariate analysis is often used as a first step in data analysis, to get a basic understanding of the data and identify any potential issues or anomalies. It can also be used to compare different groups or subsets of the data, and to test hypotheses about the data using statistical tests such as the t-test or chi-squared test.

▼ Bivariate Analysis

Bivariate analysis is a statistical method that involves analyzing the relationship between two variables. The goal of bivariate analysis is to determine whether there is a relationship between the two variables, and if so, what type of relationship it is.

Some common techniques used in bivariate analysis include:

1. Scatter plots: These are graphical representations of the relationship between the two variables, with one variable plotted on the x-axis and the other on the y-axis. Each point on the plot represents an observation, and the position of the point in relation to the axes indicates the values of the two variables for that observation. Scatter plots can be used to identify patterns or trends in the data, and to detect outliers or influential observations.
2. Correlation analysis: This involves calculating a correlation coefficient, which measures the strength and direction of the relationship between the two variables. The correlation coefficient ranges from -1 to 1, with values closer to -1 indicating a negative correlation, values closer to 1 indicating a positive correlation, and values close to 0 indicating no correlation. Correlation analysis can be used to determine whether there is a linear relationship between the two variables, and to estimate the strength of that relationship.
3. Regression analysis: This involves fitting a regression model to the data, which allows us to predict the value of one variable based on the value of the other variable. Regression analysis can be used to test hypotheses about the relationship between the two variables, and to estimate the parameters of the model.
4. Contingency tables: These are tables that show the frequency or proportion of observations that fall into different categories or combinations of categories for the two variables. Contingency tables can be used to test hypotheses about the relationship between the two variables, and to identify any patterns or trends in the data.

Bivariate analysis is a useful tool for exploring the relationship between two variables, and can provide insights into the underlying patterns and trends in the data. However, it is important to remember that correlation does not imply causation, and that other factors may be influencing the relationship between the two variables.

▼ Multivariate Analysis

▼ Unit 2 -

▼ Classification in Machine Learning

Classification is a type of supervised learning in machine learning that involves predicting the class or category of a given input based on a set of labeled examples. The goal of classification is to build a model that can accurately assign new, unseen inputs to the correct class based on their characteristics.

Some common algorithms used for classification include:

1. Logistic regression: This is a linear algorithm that models the probability of an input belonging to a particular class. It is often used for binary classification problems.
2. Decision trees: This is a tree-based algorithm that uses a series of if-then statements to classify inputs based on their characteristics. It is often used for multi-class classification problems.
3. Random forests: This is an ensemble algorithm that combines multiple decision trees to improve the accuracy and stability of the model.
4. Support vector machines (SVMs): This is a kernel-based algorithm that finds the optimal hyperplane that separates the input data into different classes. It is often used for binary classification problems, but can also be used for multi-class problems.

Classification has numerous applications in various fields, such as:

1. Image and speech recognition: Classification algorithms can be used to classify images or speech into different categories or classes, such as objects or spoken words.
2. Fraud detection: Classification algorithms can be used to detect fraudulent transactions or activities based on their characteristics.
3. Medical diagnosis: Classification algorithms can be used to diagnose diseases or conditions based on patient data and medical history.
4. Sentiment analysis: Classification algorithms can be used to classify text or social media posts into different sentiment categories, such as positive or negative.

Classification is an important tool in machine learning, and has numerous applications in various fields. By accurately classifying inputs based on their characteristics, it is possible to make more informed decisions and predictions in a wide range of contexts.

▼ Linear Regression

Linear regression is a type of supervised learning in machine learning that involves modeling the relationship between a dependent variable and one or more independent variables using a linear function. The goal of linear regression is to find the best-fit line that describes the relationship

between the variables, so that it can be used to predict the value of the dependent variable for new input values.

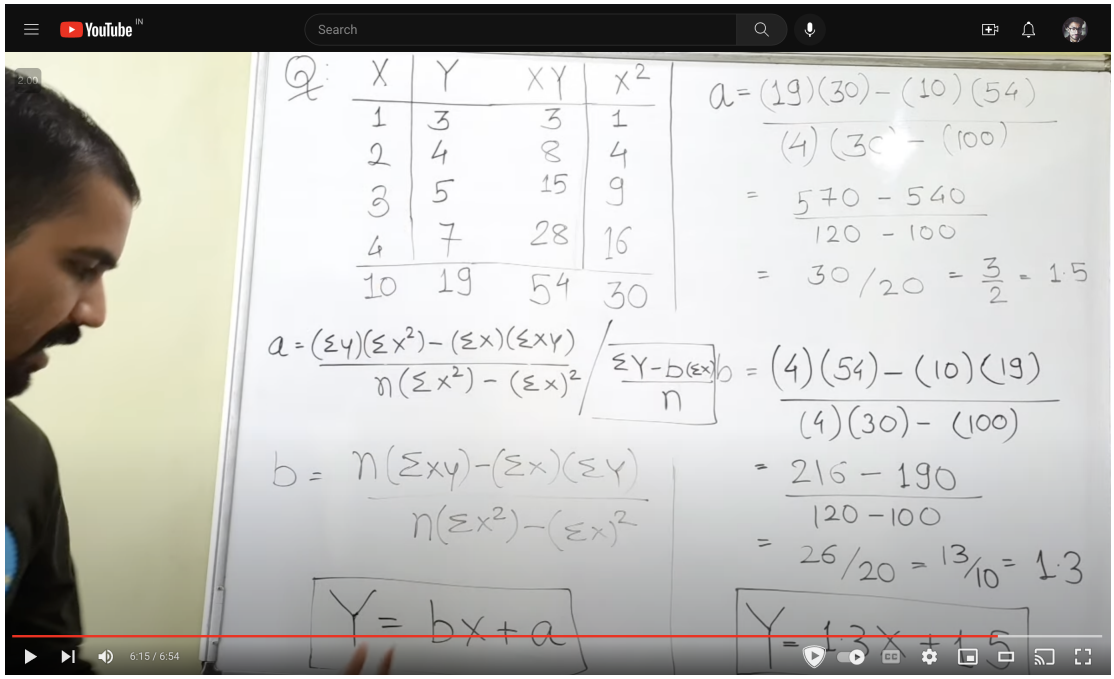
Some common variations of linear regression include:

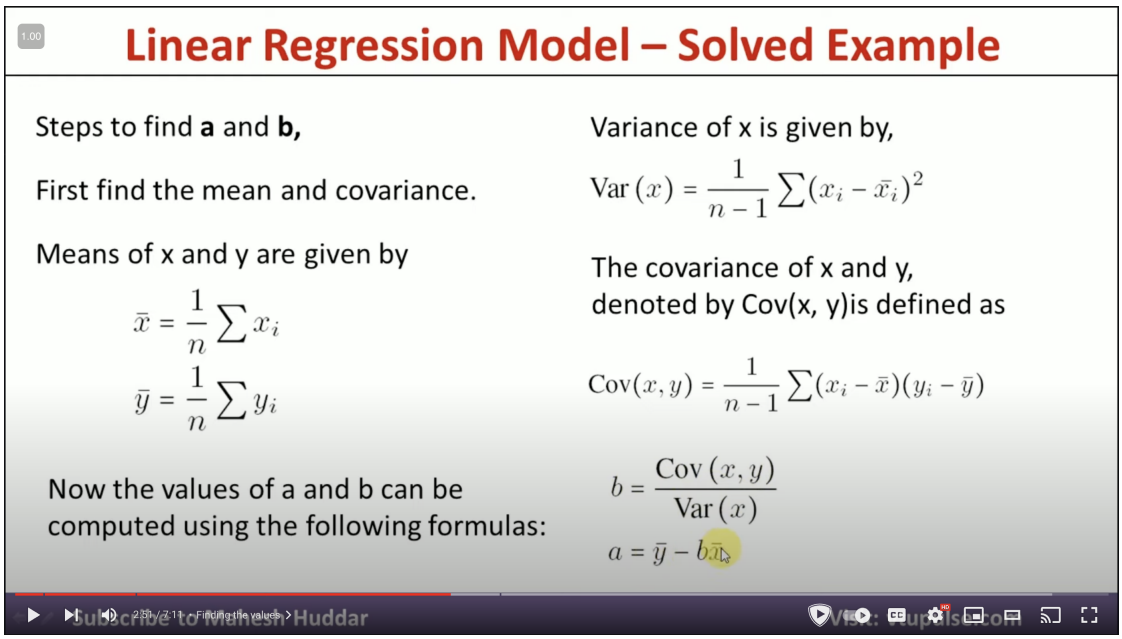
1. Simple linear regression: This involves modeling the relationship between a dependent variable and a single independent variable using a linear function. The equation for a simple linear regression model is y = mx + b, where y is the dependent variable, x is the independent variable, m is the slope of the line, and b is the y-intercept.
2. Multiple linear regression: This involves modeling the relationship between a dependent variable and multiple independent variables using a linear function. The equation for a multiple linear regression model is y = b0 + b1x1 + b2x2 + ... + bnxn, where y is the dependent variable, x1, x2, ..., xn are the independent variables, and b0, b1, b2, ..., bn are the coefficients of the model.
3. Polynomial regression: This involves modeling the relationship between a dependent variable and an independent variable using a polynomial function. The equation for a polynomial regression model is y = b0 + b1x + b2x^2 + ... + bnx^n, where y is the dependent variable, x is the independent variable, and n is the degree of the polynomial.

Linear regression has numerous applications in various fields, such as:

1. Predictive modeling: Linear regression can be used to predict the value of a dependent variable for new input values based on the relationship between the variables in the training data.
2. Trend analysis: Linear regression can be used to analyze trends in data over time, such as changes in stock prices or weather patterns.
3. Quality control: Linear regression can be used to monitor and control the quality of a product or process by predicting the outcome based on input variables.
4. Marketing analysis: Linear regression can be used to analyze the relationship between marketing campaigns and customer behavior, such as sales or website traffic.

Linear regression is a powerful tool in machine learning, and can provide valuable insights into the relationship between variables in a wide range of contexts.





▼ Regression Metrics

Regression metrics are used to evaluate the performance of regression models in machine learning. These metrics measure how well the model is able to predict the value of the dependent variable for new input values, and can be used to compare different models and select the best one for a given problem. Some common regression metrics include:

1. Mean Squared Error (MSE): This is the average of the squared differences between the predicted and actual values of the dependent variable. It is a popular metric for continuous variables, and penalizes large errors more than small errors.
2. Root Mean Squared Error (RMSE): This is the square root of the MSE, and is a popular metric for continuous variables as well. It has the

advantage of being in the same units as the dependent variable, making it more interpretable than the MSE.

3. Mean Absolute Error (MAE): This is the average of the absolute differences between the predicted and actual values of the dependent variable. It is a popular metric when outliers are present in the data, as it is less sensitive to large errors than the MSE.
4. R-squared (R2): This is a measure of the proportion of the variance in the dependent variable that is explained by the independent variables in the model. It ranges from 0 to 1, with higher values indicating a better fit. It is a popular metric for linear regression models, but can also be used for other types of regression models.
5. Adjusted R-squared: This is a modified version of the R-squared metric that takes into account the number of independent variables in the model. It penalizes models that have too many independent variables, and is a more conservative measure of model fit.
6. Mean Squared Log Error (MSLE): This is the average of the squared differences between the logarithm of the predicted and actual values of the dependent variable. It is a popular metric when the dependent variable is highly skewed or has a large range of values.
7. Median Absolute Error (MedAE): This is the median of the absolute differences between the predicted and actual values of the dependent variable. It is a popular metric when the data contains outliers or extreme values.

Each of these regression metrics has its own strengths and weaknesses, and the choice of metric depends on the specific problem and context. By using appropriate regression metrics, it is possible to evaluate the performance of regression models and make informed decisions about which model to use for a given problem.

```
https://drive.google.com/file/d/1N-IqYXoFhoqovpOJMnzZUYJlZbVhjEAe/view?usp=share_link
```

▼ Multiple Linear Regression

Multiple linear regression is a type of supervised learning in machine learning that involves modeling the relationship between a dependent variable and multiple independent variables using a linear function. The goal of multiple linear regression is to find the best-fit line or hyperplane that describes the relationship between the variables, so that it can be used to predict the value of the dependent variable for new input values.

The equation for a multiple linear regression model is y = b0 + b1x1 + b2x2 + ... + bnxn, where y is the dependent variable, x1, x2, ..., xn are the independent variables, and b0, b1, b2, ..., bn are the coefficients of the model.

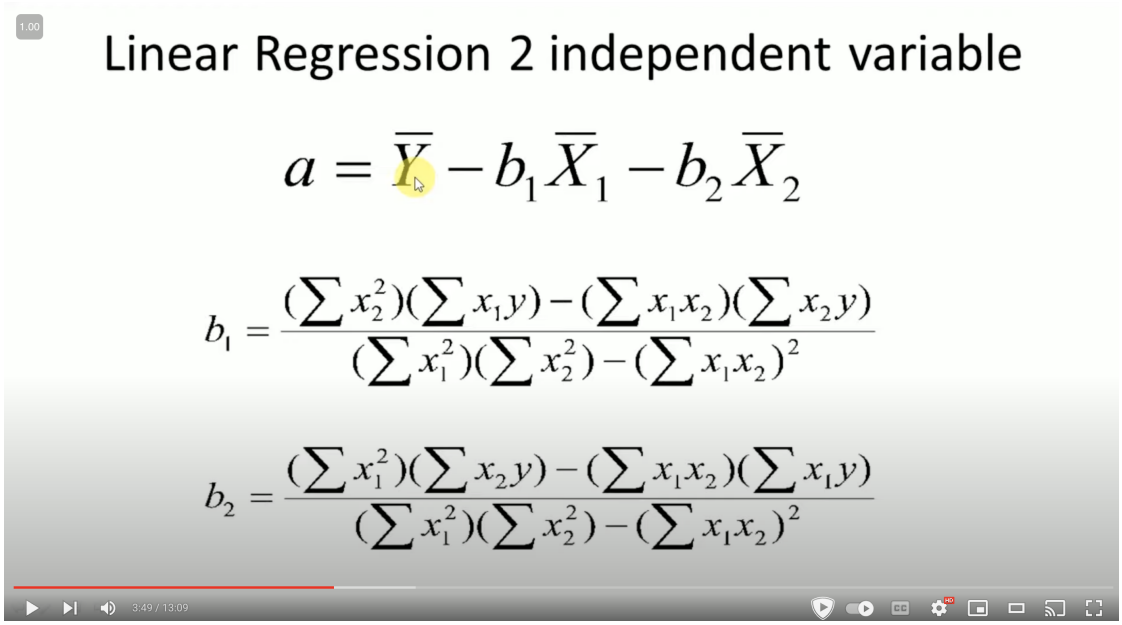
Some common variations of multiple linear regression include:

1. Polynomial regression: This involves modeling the relationship between a dependent variable and multiple independent variables using a polynomial function. The equation for a polynomial regression model is y = b0 + b1x1 + b2x2^2 + ... + bnxn^n, where y is the dependent variable, x1, x2, ..., xn are the independent variables, and n is the degree of the polynomial.

Multiple linear regression has numerous applications in various fields, such as:

1. Predictive modeling: Multiple linear regression can be used to predict the value of a dependent variable for new input values based on the relationship between the variables in the training data.
2. Marketing analysis: Multiple linear regression can be used to analyze the relationship between marketing campaigns and customer behavior, such as sales or website traffic.
3. Financial analysis: Multiple linear regression can be used to predict stock prices or analyze the relationship between economic variables.
4. Medical research: Multiple linear regression can be used to analyze the relationship between medical variables and patient outcomes.

By accurately modeling the relationship between a dependent variable and multiple independent variables, multiple linear regression can provide valuable insights and predictions in a wide range of contexts.



Numericals -

https://drive.google.com/file/d/1xlJHGY0Zbj1ytn2L6EOnv8i0VcbvJ7aH/view?usp=sharing

```
https://drive.google.com/file/d/1qEaIubMS-OKiM1vvkPeXyXEpprLoJ0bb/view?usp=share_link
```

▼ Polynomial Regression

Polynomial regression involves modeling the relationship between a dependent variable and an independent variable using a polynomial function. The equation for a polynomial regression model is y = b0 + b1x + b2x^2 + ... + bnx^n, where y is the dependent variable, x is the independent variable, and n is the degree of the polynomial.

Polynomial regression is a type of linear regression, as the relationship between the variables is still linear. However, by using a polynomial function, polynomial regression can capture more complex relationships between the variables than simple linear regression.

Polynomial regression has numerous applications in various fields, such as:

1. Predictive modeling: Polynomial regression can be used to predict the value of a dependent variable for new input values based on the relationship between the variables in the training data.
2. Trend analysis: Polynomial regression can be used to analyze trends in data over time, such as changes in stock prices or weather patterns.
3. Quality control: Polynomial regression can be used to monitor and control the quality of a product or process by predicting the outcome based on input variables.
4. Medical research: Polynomial regression can be used to analyze the relationship between medical variables and patient outcomes.

By accurately modeling the relationship between a dependent variable and an independent variable using a polynomial function, polynomial regression can provide valuable insights and predictions in a wide range of contexts.

▼ Gradient Descent

Gradient descent is an optimization algorithm commonly used in machine learning for finding the values of parameters that minimize a cost function. The basic idea behind gradient descent is to iteratively adjust the values of the parameters in the direction of the steepest descent of the cost function, until a minimum is reached.

There are two main types of gradient descent:

1. Batch gradient descent: This involves computing the gradient of the cost function with respect to the parameters over the entire training set. The parameters are then updated by taking a step in the direction of the negative gradient, scaled by a learning rate parameter alpha. Batch gradient descent can be slow for large data sets, but it generally converges to a global minimum.
2. Stochastic gradient descent: This involves computing the gradient of the cost function with respect to the parameters for each individual training example. The parameters are then updated based on each

individual gradient, scaled by a learning rate parameter alpha. Stochastic gradient descent is much faster than batch gradient descent, but it may not converge to a global minimum and can be sensitive to the learning rate parameter.

There are also variations of gradient descent, such as mini-batch gradient descent, which involves computing the gradient of the cost function with respect to a small batch of training examples.

Gradient descent is a powerful tool in machine learning, and is used to optimize a wide range of models, such as linear regression, logistic regression, and neural networks. By iteratively adjusting the values of the parameters in the direction of the steepest descent of the cost function, gradient descent can find the optimal values of the parameters that minimize the cost function and improve the accuracy of the model.

When implementing gradient descent, it is important to choose an appropriate learning rate parameter alpha. If the learning rate is too small, the algorithm may converge very slowly, while if it is too large, the algorithm may overshoot the minimum and fail to converge. Additionally, it is important to choose an appropriate number of iterations, as stopping the algorithm too early may result in suboptimal parameter values.

One potential issue with gradient descent is that it can get stuck in local minima, which are points in the cost function where the gradient is zero but are not the global minimum. To address this issue, various techniques such as momentum and adaptive learning rates have been developed.

▼ Batch Gradient Descent

Batch gradient descent is an optimization algorithm used in machine learning for finding the values of parameters that minimize a cost function. It involves computing the gradient of the cost function with respect to the parameters over the entire training set. The parameters are then updated by taking a step in the direction of the negative gradient, scaled by a learning rate parameter alpha. Batch gradient descent can be slow for large data sets, but it generally converges to a global minimum.

When implementing batch gradient descent, it is important to choose an appropriate learning rate parameter alpha. If the learning rate is too small, the algorithm may converge very slowly, while if it is too large, the algorithm may overshoot the minimum and fail to converge. Additionally, it is important to choose an appropriate number of iterations, as stopping the algorithm too early may result in suboptimal parameter values.

Batch gradient descent is used to optimize a wide range of models, such as linear regression, logistic regression, and neural networks. By iteratively adjusting the values of the parameters in the direction of the steepest descent of the cost function, batch gradient descent can find the optimal values of the parameters that minimize the cost function and improve the accuracy of the model.

▼ Stochastic Gradient Descent

Stochastic gradient descent is an optimization algorithm commonly used in machine learning for finding the values of parameters that minimize a cost function. It involves computing the gradient of the cost function with respect to the parameters for each individual training example. The parameters are then updated based on each individual gradient, scaled by a learning rate parameter alpha.

Stochastic gradient descent is much faster than batch gradient descent, but it may not converge to a global minimum and can be sensitive to the learning rate parameter. When implementing stochastic gradient descent, it is important to choose an appropriate learning rate parameter alpha. If the learning rate is too small, the algorithm may converge very slowly, while if it is too large, the algorithm may overshoot the minimum and fail to converge. Additionally, it is important to choose an appropriate number of iterations, as stopping the algorithm too early may result in suboptimal parameter values.

Stochastic gradient descent is used to optimize a wide range of models, such as linear regression, logistic regression, and neural networks. By iteratively adjusting the values of the parameters in the direction of the steepest descent of the cost function, stochastic gradient descent can find the optimal values of the parameters that minimize the cost function and improve the accuracy of the model.

▼ Mini Batch Gradient Descent

Mini-batch gradient descent is an optimization algorithm used in machine learning for finding the values of parameters that minimize a cost function. It is a variation of batch gradient descent that involves computing the gradient of the cost function with respect to a small batch of training examples, rather than the entire training set. The parameters are then updated based on the average gradient of the batch, scaled by a learning rate parameter alpha.

Mini-batch gradient descent is faster than batch gradient descent, as it can take advantage of parallel processing and is less sensitive to the learning rate parameter. Additionally, mini-batch gradient descent can converge to a better minimum than stochastic gradient descent, as it takes into account more information about the cost function than individual training examples.

When implementing mini-batch gradient descent, it is important to choose an appropriate batch size, as choosing a very small batch size can result in noisy gradients, while choosing a very large batch size can result in slow convergence. Additionally, it is important to choose an appropriate learning rate parameter alpha and number of iterations.

Mini-batch gradient descent is used to optimize a wide range of models, such as linear regression, logistic regression, and neural networks. By iteratively adjusting the values of the parameters in the direction of the steepest descent of the cost function, mini-batch gradient descent can find the optimal values of the parameters that minimize the cost function and improve the accuracy of the model.

▼ Bias Variance

Bias and variance are two types of errors that can occur in a machine learning model.

Bias refers to the error that is introduced by approximating a real-world problem with a simpler model. A high bias model is one that is too simple and cannot capture the complexity of the underlying problem. This can result in underfitting, where the model is unable to accurately predict the outcomes on the training and test data.

Variance refers to the error that is introduced by the model's sensitivity to small fluctuations in the training data. A high variance model is one that is too complex and can fit the training data too closely. This can result in overfitting, where the model performs well on the training data but poorly on the test data.

The goal of a machine learning model is to balance bias and variance to achieve good generalization performance, where the model can accurately predict outcomes on unseen data. This can be achieved through techniques such as regularization, cross-validation, and ensemble methods.

Regularization is a technique that adds a penalty term to the model's loss function, which discourages the model from overfitting the training data. Cross-validation is another technique that involves splitting the data into training and validation sets, and using the validation set to tune the model's hyperparameters. Ensemble methods involve combining multiple models to reduce the bias and variance of the overall model.

By balancing bias and variance, a machine learning model can achieve good generalization performance and accurately predict outcomes on unseen data.

▼ Ridge Regression

Ridge regression is a regularization technique used in linear regression to prevent overfitting. It works by adding a penalty term to the loss function that shrinks the coefficients of the model towards zero. This penalty term is proportional to the square of the magnitude of the coefficients, which means that larger coefficients are penalized more heavily than smaller coefficients.

The equation for ridge regression is:

minimize ||y - Xw||^2 + alpha * ||w||^2

where y is the vector of target values, X is the matrix of input features, w is the vector of coefficients, and alpha is the regularization parameter. The term ||w||^2 is the L2 norm of the coefficients, which is the sum of the squares of the individual coefficients.

Ridge regression can be used to handle multicollinearity, which is the presence of highly correlated independent variables in the data. In the presence of multicollinearity, the coefficients of the linear regression model can become unstable and sensitive to small changes in the data. Ridge regression addresses this issue by reducing the magnitudes of the coefficients, which can improve the stability and generalization performance of the model.

One disadvantage of ridge regression is that it does not perform variable selection, which means that it cannot completely eliminate irrelevant features from the model. However, it can reduce the impact of irrelevant features on the model's predictions by shrinking their coefficients towards zero.

Overall, ridge regression is a powerful tool in linear regression that can improve the stability and generalization performance of the model by reducing the magnitudes of the coefficients.

▼ Lasso Regression

Lasso regression is a regularization technique used in linear regression to prevent overfitting. It works by adding a penalty term to the loss function that shrinks the coefficients of the model towards zero. This penalty term is proportional to the absolute value of the magnitude of the coefficients, which means that larger coefficients are penalized more heavily than smaller coefficients.

The equation for lasso regression is:

minimize ||y - Xw||^2 + alpha * ||w||

where y is the vector of target values, X is the matrix of input features, w is the vector of coefficients, and alpha is the regularization parameter. The term ||w|| is the L1 norm of the coefficients, which is the sum of the absolute values of the individual coefficients.

Lasso regression can be used for variable selection, as it tends to shrink the coefficients of irrelevant features towards zero, effectively removing them from the model. This can lead to a more interpretable and simple model, as well as improved generalization performance on new data.

One limitation of lasso regression is that it can only be used for linear models, as it requires the loss function to be convex. Additionally, it may not perform well in the presence of multicollinearity, as it tends to arbitrarily select one of the correlated features and ignore the others.

Overall, lasso regression is a useful tool in linear regression for variable selection and preventing overfitting, and can lead to simpler and more interpretable models.

▼ ElasticNet Regression

ElasticNet regression is a regularization technique used in linear regression to prevent overfitting. It is a combination of L1 regularization (lasso regression) and L2 regularization (ridge regression).

The objective of ElasticNet regression is to minimize the sum of the squared errors between the predicted and actual values, while also minimizing the sum of the absolute values of the coefficients multiplied by a tuning parameter alpha.

The equation for ElasticNet regression is:

minimize ||y - Xw||^2 + alpha * ((1 - l1_ratio) * ||w||^2 / 2 + l1_ratio * ||w||)

where y is the vector of target values, X is the matrix of input features, w is the vector of coefficients, alpha is the regularization parameter, and l1_ratio is a parameter that controls the balance between L1 and L2 regularization.

ElasticNet regression can be used to handle multicollinearity and perform variable selection, as it can reduce the magnitudes of the coefficients and shrink irrelevant features towards zero. It is particularly useful in cases where the number of predictor variables is larger than the number of observations, as it can help to avoid overfitting.

However, ElasticNet regression can be computationally expensive and requires tuning of the regularization parameter and l1_ratio. In addition, it may not perform well in cases where the number of relevant features is much smaller than the total number of features.

Overall, ElasticNet regression is a powerful tool in linear regression for preventing overfitting and performing variable selection, and can lead to more accurate and interpretable models.

https://drive.google.com/file/d/18ajVHQS7ukOuFqZgR_Kwzm4o1QQ-YV-0/view?usp=share_link

▼ Logistic Regression

Logistic regression is a statistical method used to analyze a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes).

The goal of logistic regression is to find the best fitting model to describe the relationship between the dichotomous characteristic of interest (dependent variable = response or outcome variable) and a set of independent (predictor or explanatory) variables. Logistic regression generates the coefficients (and its standard errors and significance levels) of a formula to predict a logit transformation of the probability of presence of the characteristic of interest.

The logit transformation is defined as the logged odds:

Logit(p) = ln[p/(1-p)]

where p is the probability of presence of the characteristic of interest. The odds are defined as:

Odds = p / (1-p)

which is the ratio of the probability of presence to the probability of absence. The odds range from 0 to infinity. The logit transform ranges from

negative infinity to positive infinity. The logistic regression equation can be written as:

Logit(p) = b0+b1X1+b2X2+...+bkXk

where b0 is the intercept, b1 to bk are the coefficients of the predictor variables, and X1 to Xk are the values of the predictor variables.

The logistic regression model can be used to predict the presence or absence of the characteristic of interest based on values of the independent variables. The logistic regression model can also be used to estimate the probability of presence of the characteristic of interest.

Logistic regression has many applications in various fields, such as:

1. Medical research: Logistic regression can be used to analyze the relationship between medical variables and patient outcomes, such as the presence or absence of a disease.
2. Marketing analysis: Logistic regression can be used to analyze the relationship between marketing campaigns and customer behavior, such as the likelihood of a customer buying a product.
3. Environmental science: Logistic regression can be used to analyze the relationship between environmental variables and the presence or absence of a species.
4. Credit scoring: Logistic regression can be used to analyze the relationship between credit risk and a set of predictor variables.

By accurately modeling the relationship between a dichotomous outcome and a set of predictor variables, logistic regression can provide valuable insights and predictions in a wide range of contexts.

▼ Confusion Matrix

A confusion matrix is a table used to evaluate the performance of a classification model. The matrix displays the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) produced by the classifier.

The four values in the confusion matrix can be used to calculate various performance metrics of the classification model, such as accuracy, precision, recall, and F1 score.

- Accuracy: The proportion of correct predictions out of the total number of predictions. Accuracy = (TP + TN) / (TP + TN + FP + FN)
- Precision: The proportion of true positive predictions out of the total number of positive predictions. Precision = TP / (TP + FP)
- Recall: The proportion of true positive predictions out of the total number of actual positives. Recall = TP / (TP + FN)
- F1 Score: The harmonic mean of precision and recall. F1 Score = 2 * (precision * recall) / (precision + recall)

Confusion matrices can be especially useful when evaluating models for imbalanced datasets, where one class may have significantly more data than the other. In these cases, accuracy may not be a reliable metric, and other metrics such as precision and recall may be more informative.

Overall, confusion matrices provide a useful tool for evaluating the performance of classification models and identifying areas where the model may need improvement.

▼ Unit 3 -

▼ Dimensionality Reduction

Dimensionality reduction is a technique used in machine learning and data analysis to reduce the number of features or variables in a dataset while preserving important information. The goal is to simplify the dataset and eliminate redundant or irrelevant features, which can help improve computational efficiency, decrease storage requirements, and mitigate the effects of the curse of dimensionality.

Dimensionality reduction methods can be broadly categorized into two types: feature selection and feature extraction.

1. **Feature Selection:** This approach selects a subset of the original features based on some criteria. It aims to identify the most relevant features and discard the rest. Common feature selection techniques include filtering methods (e.g., variance threshold, mutual information) and wrapper methods (e.g., recursive feature elimination, genetic algorithms).
2. **Feature Extraction:** This approach creates new low-dimensional representations by combining or transforming the original features. These new representations, called latent variables or components, are typically a linear combination of the original features. Feature extraction techniques, such as Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA), aim to find the most informative combinations of features that explain the maximum variance or discriminate between classes.

Principal Component Analysis (PCA) is one of the most widely used dimensionality reduction techniques. It identifies the orthogonal directions (principal components) in the data that capture the most significant variability. By retaining only a subset of these components, PCA allows for a lower-dimensional representation of the data while preserving as much information as possible.

Other popular dimensionality reduction methods include t-distributed Stochastic Neighbor Embedding (t-SNE), which is useful for visualizing high-dimensional data in two or three dimensions, and autoencoders, which are neural network models that can learn compressed representations of the input data.

When applying dimensionality reduction techniques, it is crucial to strike a balance between reducing the dimensionality and preserving enough information for the specific task at hand. Additionally, it is important to consider the interpretability and impact on the downstream tasks, as dimensionality reduction can sometimes lead to a loss of interpretability or affect the performance of certain algorithms.

▼ Forward and Backward Selection

Forward selection and backward selection are two feature selection techniques used in machine learning and statistics. These methods aim to find an optimal subset of features from a larger set of available features.

1. **Forward Selection:** Forward selection starts with an empty set of selected features and progressively adds features one at a time. At each step, the algorithm evaluates the performance of the model using the currently selected features and adds the feature that contributes the most to the model's performance. The process continues until a desired number of features is reached or until adding additional features does not improve the model significantly.

The steps involved in forward selection are as follows:

- Start with an empty set of selected features.
- For each feature not yet selected, train a model using the currently selected features plus the feature being considered.
- Evaluate the performance of the model using a predefined metric (e.g., accuracy, mean squared error).
- Select the feature that improves the model's performance the most.
- Add the selected feature to the set of selected features.
- Repeat the above steps until the desired number of features is reached or until no significant improvement is observed.

1. **Backward Selection:** Backward selection starts with the full set of available features and removes one feature at a time. Similar to forward selection, the algorithm evaluates the model's performance at each step and removes the feature that contributes the least to the model's performance. The process continues until a desired number of features is reached or until removing additional features does not significantly affect the model's performance.

The steps involved in backward selection are as follows:

- Start with a set of all available features.
- For each feature in the set, train a model using the remaining features (excluding the feature being considered).
- Evaluate the performance of the model using a predefined metric.
- Remove the feature that has the least impact on the model's performance.
- Repeat the above steps until the desired number of features is reached or until removing additional features does not significantly affect the model's performance.

Both forward selection and backward selection are greedy algorithms that iteratively modify the feature set based on the current best choice. The selection criteria can vary depending on the specific machine learning algorithm and evaluation metric used. It is important to evaluate the selected feature subset's performance on an independent validation set to avoid overfitting and ensure generalizability.

▼ Subset Selection

Subset selection is a feature selection technique that aims to find the best subset of features from a larger set of available features. It involves evaluating all possible combinations of features and selecting the subset that yields the best model performance according to a specified criterion.

The subset selection process can be computationally expensive, especially for large feature sets, as it requires evaluating a large number of feature combinations. Therefore, it is typically only feasible for datasets with a relatively small number of features. There are two main approaches to subset selection: exhaustive search and heuristic search.

1. **Exhaustive Search:** In exhaustive search, all possible feature combinations are evaluated to determine the optimal subset. This approach guarantees finding the best subset but can be computationally expensive. The number of possible feature combinations grows exponentially with the number of features, making exhaustive search impractical for datasets with a large number of features.
2. **Heuristic Search:** Heuristic search methods aim to find a good or near-optimal feature subset without evaluating all possible combinations. These methods use certain criteria or algorithms to guide the search process and explore the feature space more efficiently. Examples of heuristic search algorithms include forward selection, backward elimination, and genetic algorithms. These algorithms iteratively build or refine the feature subset based on the evaluation of subsets in a more efficient manner than exhaustive search.

The choice of the evaluation criterion is crucial in subset selection. Common criteria include:

- **Prediction performance:** The criterion can be based on the performance of a specific machine learning algorithm, such as accuracy, mean squared error, or area under the receiver operating characteristic curve (AUC-ROC). The subset selection algorithm aims to find the subset that maximizes or minimizes the chosen performance metric.
- **Information-based criteria:** These criteria, such as Akaike information criterion (AIC) or Bayesian information criterion (BIC), measure the trade-off between model complexity (number of features) and goodness of fit. The goal is to find the subset that minimizes the criterion, indicating a good balance between model complexity and performance.
- **Stability and robustness:** Some subset selection methods consider the stability or robustness of the selected features. For example, stability selection combines multiple subsamples of the data and evaluates the frequency at which each feature is selected across the subsamples. Features selected more frequently are considered more stable and reliable.

Subset selection can help in reducing overfitting, improving model interpretability, and enhancing computational efficiency by eliminating irrelevant or redundant features. However, it is important to validate the selected subset on an independent dataset to ensure its generalizability

and assess its performance compared to other feature selection or dimensionality reduction techniques.

▼ LDA

Linear Discriminant Analysis (LDA) and Fisher's criteria are both techniques used for dimensionality reduction and feature extraction, particularly in the context of classification problems.

Linear Discriminant Analysis (LDA):
LDA is a technique that finds a linear combination of features that maximizes the separation between different classes in a dataset. It aims to project the data onto a lower-dimensional space while maximizing the class separability. LDA assumes that the data is normally distributed and that the classes have equal covariance matrices.

The main steps involved in LDA are as follows:

1. Compute the mean vectors of each class.
2. Compute the scatter matrices, including the within-class scatter matrix (within-class variance) and the between-class scatter matrix (between-class variance).
3. Solve the generalized eigenvalue problem using the scatter matrices to obtain the discriminant directions.
4. Project the data onto the discriminant directions to obtain the reduced-dimensional representation.

The discriminant directions, also known as linear discriminants, are chosen to maximize the ratio of between-class scatter to within-class scatter. The number of linear discriminants is equal to the number of classes minus one or the desired reduced dimension, whichever is smaller.

LDA is often used as a dimensionality reduction technique and can also serve as a classification algorithm by assigning class labels based on the projected data. LDA assumes linearity and the underlying assumptions of the data, such as normality and equal covariance matrices, should be verified before applying it.

Fisher's Criteria:
Fisher's criteria, also known as Fisher's linear discriminant, is a dimensionality reduction technique that aims to find a projection of the data that maximizes the separability between classes while minimizing the intra-class scatter. It is closely related to LDA, and sometimes the terms LDA and Fisher's criteria are used interchangeably.

The steps involved in Fisher's criteria are similar to LDA:

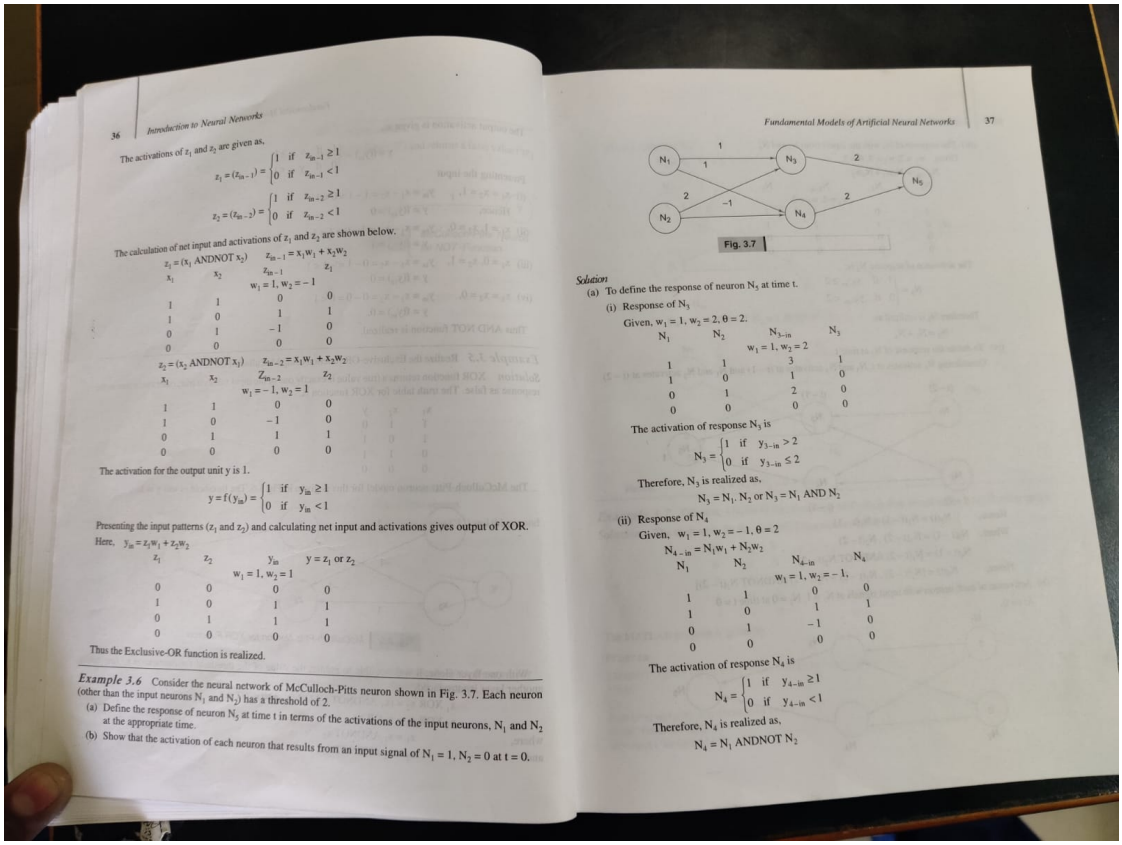
1. Compute the mean vectors of each class.
2. Compute the scatter matrices, including the within-class scatter matrix and the between-class scatter matrix.
3. Solve the generalized eigenvalue problem using the scatter matrices to obtain the projection vectors (linear discriminants).
4. Project the data onto the projection vectors to obtain the reduced-dimensional representation.

Fisher's criteria differ from LDA in that it does not assume equal covariance matrices across classes. Instead, it aims to maximize the ratio of between-class scatter to within-class scatter, considering only the relevant information for discrimination.

Fisher's criteria can be used for dimensionality reduction and classification tasks. It is particularly useful when the classes have unequal covariance matrices or when the underlying data assumptions do not hold for LDA. However, like LDA, Fisher's criteria assume linearity and should be validated against the assumptions of the data.

▼ Unit 4

▼ McCulloh Pits



36 Introduction to Neural Networks

The activations of z_1 and z_2 are given as,

$$z_1 = (z_{in-1}) = \begin{cases} 1 & \text{if } z_{in-1} \geq 1 \\ 0 & \text{if } z_{in-1} < 1 \end{cases}$$

$$z_2 = (z_{in-2}) = \begin{cases} 1 & \text{if } z_{in-2} \geq 1 \\ 0 & \text{if } z_{in-2} < 1 \end{cases}$$

The calculation of net input and activations of z_1 and z_2 are shown below.

$z_1 = (x_1$ ANDNOT $x_2)$ $\quad z_{in-1} = x_1w_1 + x_2w_2$

x_1	x_2	z_{in-1}	z_1
		$w_1 = 1, w_2 = -1$	
1	1	0	0
1	0	1	1
0	1	−1	0
0	0	0	0

$z_2 = (x_2$ ANDNOT $x_1)$ $\quad z_{in-2} = x_1w_1 + x_2w_2$

x_1	x_2	Z_{in-2}	z_2
		$w_1 = -1, w_2 = 1$	
1	1	0	0
1	0	−1	0
0	1	1	1
0	0	0	0

The activation for the output unit y is 1.

$$y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} \geq 1 \\ 0 & \text{if } y_{in} < 1 \end{cases}$$

Presenting the input patterns (z_1 and z_2) and calculating net input and activations gives output of XOR.

Here, $y_{in} = z_1w_1 + z_2w_2$

z_1	z_2	y_{in}	$y = z_1$ or z_2
		$w_1 = 1, w_2 = 1$	
0	0	0	0
1	0	1	1
0	1	1	1
0	0	0	0

Thus the Exclusive-OR function is realized.

Example 3.6 Consider the neural network of McCulloch-Pitts neuron shown in Fig. 3.7. Each neuron (other than the input neurons N_1 and N_2) has a threshold of 2.

(a) Define the response of neuron N_5 at time t in terms of the activations of the input neurons, N_1 and N_2 at the appropriate time.

(b) Show that the activation of each neuron that results from an input signal of $N_1 = 1$, $N_2 = 0$ at t = 0.

Fundamental Models of Artificial Neural Networks 37

Fig. 3.7

Solution

(a) To define the response of neuron N_5 at time t.

(i) Response of N_3

Given, $w_1 = 1, w_2 = 2, \theta = 2$.

N_1	N_2	N_{3-in}	N_3
		$w_1 = 1, w_2 = 2$	
1	1	3	1
1	0	1	0
0	1	2	0
0	0	0	0

The activation of response N_3 is

$$N_3 = \begin{cases} 1 & \text{if } y_{3-in} > 2 \\ 0 & \text{if } y_{3-in} \leq 2 \end{cases}$$

Therefore, N_3 is realized as,

$N_3 = N_1 . N_2$ or $N_3 = N_1$ AND N_2

(ii) Response of N_4

Given, $w_1 = 1, w_2 = -1, \theta = 2$

$N_{4-in} = N_1w_1 + N_2w_2$

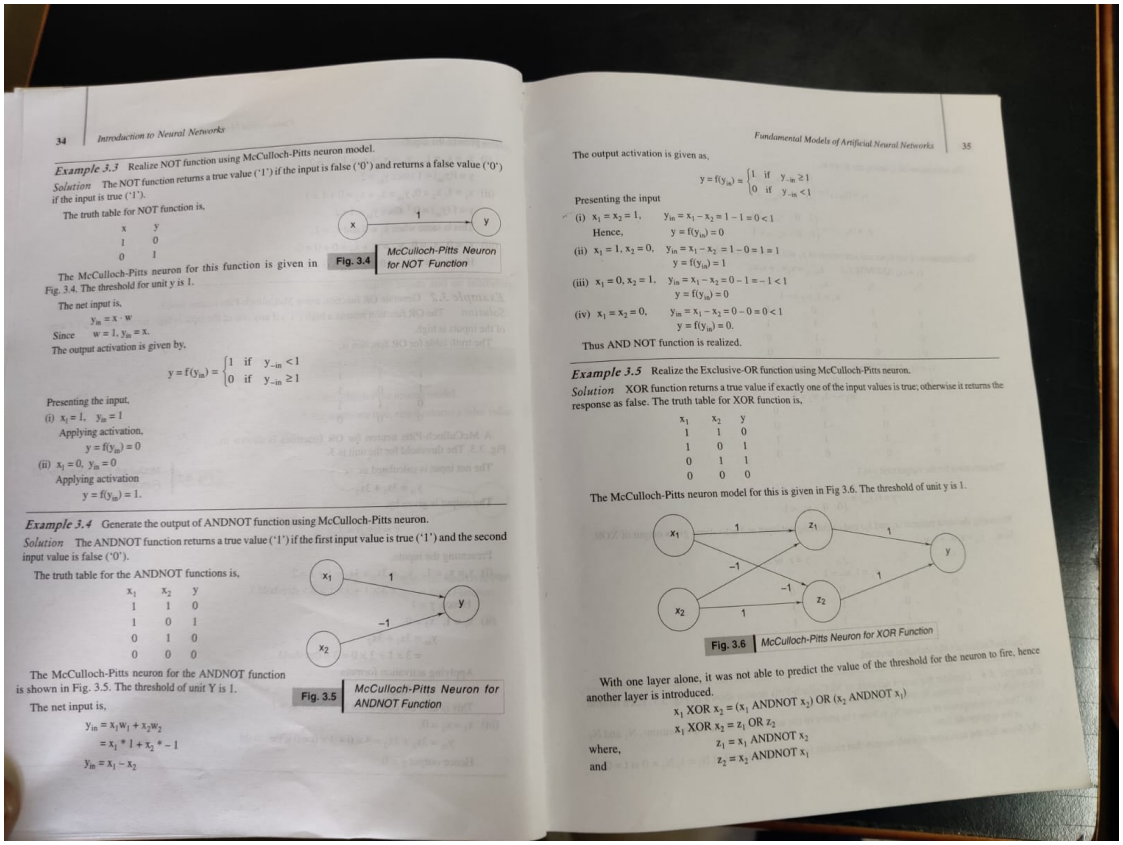
N_1	N_2	N_{4-in}	N_4
		$w_1 = 1, w_2 = -1,$	
1	1	0	0
1	0	1	1
0	1	−1	0
0	0	0	0

The activation of response N_4 is

$$N_4 = \begin{cases} 1 & \text{if } y_{4-in} \geq 1 \\ 0 & \text{if } y_{4-in} < 1 \end{cases}$$

Therefore, N_4 is realized as,

$N_4 = N_1$ ANDNOT N_2



34 Introduction to Neural Networks

Example 3.3 Realize NOT function using McCulloch-Pitts neuron model.

Solution The NOT function returns a true value ('1') if the input is false ('0') and returns a false value ('0') if the input is true ('1').

The truth table for NOT function is,

x	y
1	0
0	1

The McCulloch-Pitts neuron for this function is given in Fig. 3.4. The threshold for unit y is 1.

Fig. 3.4 McCulloch-Pitts Neuron for NOT Function

The net input is,

$y_{in} = x \cdot w$

Since $w = 1$, $y_{in} = x$.

The output activation is given by,

$$y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{-in} < 1 \\ 0 & \text{if } y_{-in} \geq 1 \end{cases}$$

Presenting the input,

(i) $x_1 = 1$, $y_{in} = 1$

Applying activation,

$y = f(y_{in}) = 0$

(ii) $x_1 = 0$, $y_{in} = 0$

Applying activation

$y = f(y_{in}) = 1$.

Example 3.4 Generate the output of ANDNOT function using McCulloch-Pitts neuron.

Solution The ANDNOT function returns a true value ('1') if the first input value is true ('1') and the second input value is false ('0').

The truth table for the ANDNOT functions is,

x_1	x_2	y
1	1	0
1	0	1
0	1	0
0	0	0

The McCulloch-Pitts neuron for the ANDNOT function is shown in Fig. 3.5. The threshold of unit Y is 1.

Fig. 3.5 McCulloch-Pitts Neuron for ANDNOT Function

The net input is,

$y_{in} = x_1 w_1 + x_2 w_2$

$= x_1 * 1 + x_2 * -1$

$y_{in} = x_1 - x_2$

Fundamental Models of Artificial Neural Networks 35

The output activation is given as,

$$y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{-in} \geq 1 \\ 0 & \text{if } y_{-in} < 1 \end{cases}$$

Presenting the input

(i) $x_1 = x_2 = 1$, $y_{in} = x_1 - x_2 = 1 - 1 = 0 < 1$

Hence, $y = f(y_{in}) = 0$

(ii) $x_1 = 1, x_2 = 0$, $y_{in} = x_1 - x_2 = 1 - 0 = 1 = 1$

$y = f(y_{in}) = 1$

(iii) $x_1 = 0, x_2 = 1$, $y_{in} = x_1 - x_2 = 0 - 1 = -1 < 1$

$y = f(y_{in}) = 0$

(iv) $x_1 = x_2 = 0$, $y_{in} = x_1 - x_2 = 0 - 0 = 0 < 1$

$y = f(y_{in}) = 0$.

Thus AND NOT function is realized.

Example 3.5 Realize the Exclusive-OR function using McCulloch-Pitts neuron.

Solution XOR function returns a true value if exactly one of the input values is true; otherwise it returns the response as false. The truth table for XOR function is,

x_1	x_2	y
1	1	0
1	0	1
0	1	1
0	0	0

The McCulloch-Pitts neuron model for this is given in Fig 3.6. The threshold of unit y is 1.

Fig. 3.6 McCulloch-Pitts Neuron for XOR Function

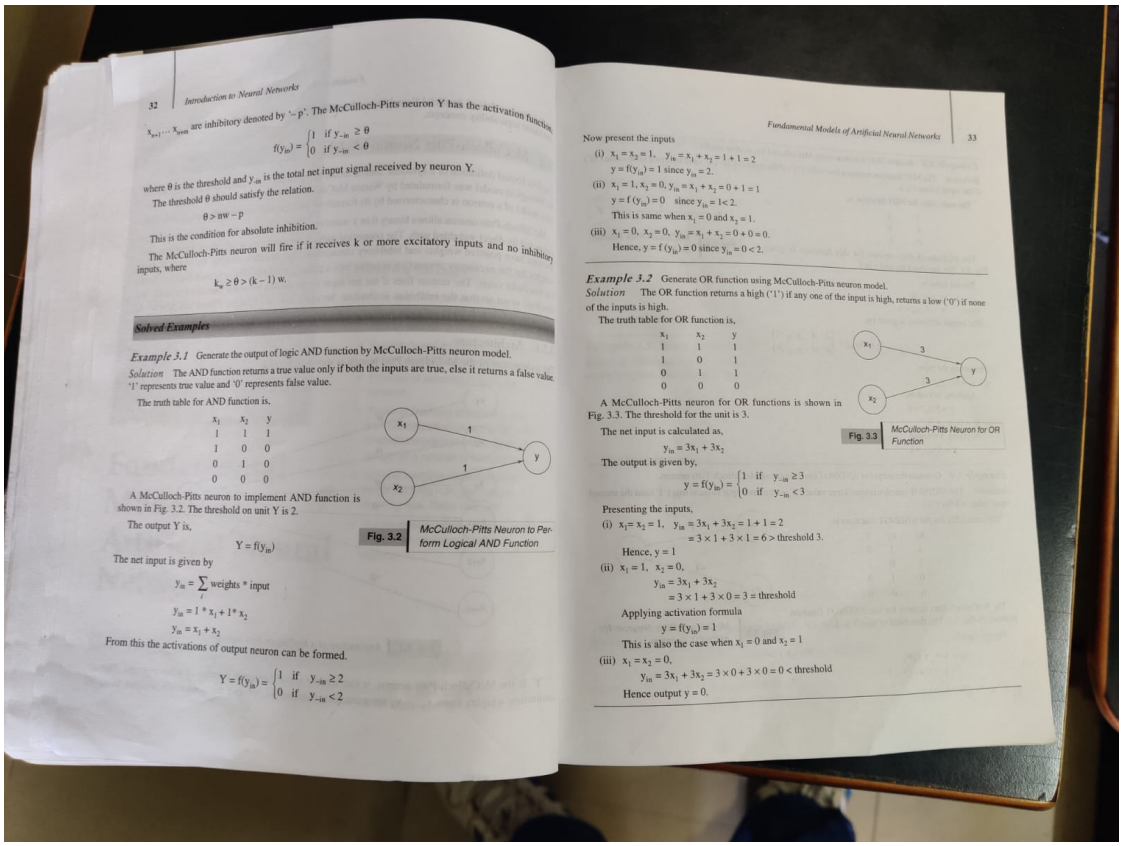
With one layer alone, it was not able to predict the value of the threshold for the neuron to fire, hence another layer is introduced.

x_1 XOR $x_2 = (x_1$ ANDNOT $x_2)$ OR $(x_2$ ANDNOT $x_1)$

where, x_1 XOR $x_2 = z_1$ OR z_2

and $z_1 = x_1$ ANDNOT x_2

$z_2 = x_2$ ANDNOT x_1



32 Introduction to Neural Networks

$x_{n+1}, \ldots, x_{n+m}$ are inhibitory denoted by '–p'. The McCulloch-Pitts neuron Y has the activation function

$$f(y_{in}) = \begin{cases} 1 & \text{if } y_{-in} \geq \theta \\ 0 & \text{if } y_{-in} < \theta \end{cases}$$

where θ is the threshold and y_{in} is the total net input signal received by neuron Y.

The threshold θ should satisfy the relation.

$\theta > nw - p$

This is the condition for absolute inhibition.

The McCulloch-Pitts neuron will fire if it receives k or more excitatory inputs and no inhibitory inputs, where

$k_w \geq \theta > (k - 1)\, w$.

Solved Examples

Example 3.1 Generate the output of logic AND function by McCulloch-Pitts neuron model.

Solution The AND function returns a true value only if both the inputs are true, else it returns a false value. '1' represents true value and '0' represents false value.

The truth table for AND function is,

x_1	x_2	y
1	1	1
1	0	0
0	1	0
0	0	0

A McCulloch-Pitts neuron to implement AND function is shown in Fig. 3.2. The threshold on unit Y is 2.

Fig. 3.2 McCulloch-Pitts Neuron to Perform Logical AND Function

The output Y is,

$Y = f(y_{in})$

The net input is given by

$y_{in} = \sum_i$ weights * input

$y_{in} = 1 * x_1 + 1 * x_2$

$y_{in} = x_1 + x_2$

From this the activations of output neuron can be formed.

$$Y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{-in} \geq 2 \\ 0 & \text{if } y_{-in} < 2 \end{cases}$$

Fundamental Models of Artificial Neural Networks 33

Now present the inputs

(i) $x_1 = x_2 = 1$, $y_{in} = x_1 + x_2 = 1 + 1 = 2$

$y = f(y_{in}) = 1$ since $y_{in} = 2$.

(ii) $x_1 = 1, x_2 = 0$, $y_{in} = x_1 + x_2 = 0 + 1 = 1$

$y = f(y_{in}) = 0$ since $y_{in} = 1 < 2$.

This is same when $x_1 = 0$ and $x_2 = 1$.

(iii) $x_1 = 0$, $x_2 = 0$, $y_{in} = x_1 + x_2 = 0 + 0 = 0$.

Hence, $y = f(y_{in}) = 0$ since $y_{in} = 0 < 2$.

Example 3.2 Generate OR function using McCulloch-Pitts neuron model.

Solution The OR function returns a high ('1') if any one of the input is high, returns a low ('0') if none of the inputs is high.

The truth table for OR function is,

x_1	x_2	y
1	1	1
1	0	1
0	1	1
0	0	0

A McCulloch-Pitts neuron for OR functions is shown in Fig. 3.3. The threshold for the unit is 3.

Fig. 3.3 McCulloch-Pitts Neuron for OR Function

The net input is calculated as,

$y_{in} = 3x_1 + 3x_2$

The output is given by,

$$y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{-in} \geq 3 \\ 0 & \text{if } y_{-in} < 3 \end{cases}$$

Presenting the inputs,

(i) $x_1 = x_2 = 1$, $y_{in} = 3x_1 + 3x_2 = 1 + 1 = 2$

$= 3 \times 1 + 3 \times 1 = 6 >$ threshold 3.

Hence, $y = 1$

(ii) $x_1 = 1$, $x_2 = 0$,

$y_{in} = 3x_1 + 3x_2$

$= 3 \times 1 + 3 \times 0 = 3 =$ threshold

Applying activation formula

$y = f(y_{in}) = 1$

This is also the case when $x_1 = 0$ and $x_2 = 1$

(iii) $x_1 = x_2 = 0$,

$y_{in} = 3x_1 + 3x_2 = 3 \times 0 + 3 \times 0 = 0 <$ threshold

Hence output $y = 0$.

▼ Types of neural network

There are several types of neural networks, each designed for specific tasks and modeling different types of data. Here are some commonly used types of neural networks:

1. **Feedforward Neural Network (FNN)**: This is the simplest type of neural network, where information flows only in one direction, from the input layer through one or more hidden layers to the output layer. FNNs are commonly used for regression and classification tasks.
2. **Convolutional Neural Network (CNN)**: CNNs are primarily used for analyzing visual data such as images. They are designed to automatically extract relevant features from the input using convolutional layers and pooling layers. CNNs have been highly successful in image recognition, object detection, and image generation tasks.
3. **Recurrent Neural Network (RNN)**: RNNs are suitable for sequential data, such as time series or natural language processing tasks. They have feedback connections that allow information to persist and flow through the network over time. RNNs can capture temporal dependencies and are used in tasks like speech recognition, machine translation, and sentiment analysis.
4. **Long Short-Term Memory (LSTM) Network**: LSTMs are a type of RNN that can effectively handle long-term dependencies. They are designed to address the vanishing gradient problem in traditional RNNs and have been successful in tasks involving sequential data with long-term dependencies, such as speech recognition and language modeling.
5. **Gated Recurrent Unit (GRU) Network**: GRUs are another variation of RNNs that address the vanishing gradient problem. They are simpler than LSTMs but can still capture long-term dependencies. GRUs have been widely used in natural language processing tasks, machine translation, and speech recognition.
6. **Autoencoder**: Autoencoders are unsupervised learning models that aim to learn efficient representations of the input data. They consist of an encoder that maps the input to a lower-dimensional latent space and a decoder that reconstructs the input from the latent representation. Autoencoders have applications in dimensionality reduction, anomaly detection, and generative modeling.
7. **Generative Adversarial Network (GAN)**: GANs are composed of two neural networks, a generator and a discriminator, that are trained in a competitive setting. The generator tries to generate realistic data instances, while the discriminator tries to distinguish between real and generated instances. GANs have been successful in generating realistic images, videos, and other types of data.

These are just a few examples of the many types of neural networks available. There are also variations and combinations of these networks, such as recurrent convolutional neural networks (RCNNs), transformer networks, and deep belief networks (DBNs), each with its own specific architectures and applications.

▼ Backpropogation

Here is a step-by-step breakdown of backpropagation calculations in the context of a feedforward neural network:

1. **Forward Pass**:
   - Compute the weighted sum (z) and activation (a) of each neuron in each layer using the following formulas:
     - Weighted sum (z):

$$z_j^{(l)} = \sum_{i=1}^{n} w_{ij}^{(l)} a_i^{(l-1)} + b_j^{(l)}$$

     - Activation (a): , where g() is the activation function.

$$a_j^{(l)} = g(z_j^{(l)})$$

2. **Loss Calculation**:
   - Compute the loss (L) using an appropriate loss function, such as mean squared error (MSE) or cross-entropy loss, depending on the problem.
3. **Backward Pass (Backpropagation)**:
   - Compute the gradients of the loss with respect to the outputs of the output layer neurons (δ):
     - For the output layer (L):

$$\delta_j^{(L)} = \frac{\partial L}{\partial a_j^{(L)}} \cdot g'(z_j^{(L)})$$

     - For other layers (l):

$$\delta_j^{(l)} = \sum_{k=1}^{n} \delta_k^{(l+1)} w_{jk}^{(l+1)} \cdot g'(z_j^{(l)})$$

4. **Gradient Calculation**:
   - Compute the gradients of the loss with respect to the weights (dw) and biases (db) of each neuron in each layer using the following formulas:
     - Weight gradient (dw):

$$\frac{\partial L}{\partial w_{ij}^{(l)}} = \delta_j^{(l)} a_i^{(l-1)}$$

     - Bias gradient (db):

$$\frac{\partial L}{\partial b_j^{(l)}} = \delta_j^{(l)}$$

5. **Update Parameters**:
    - Update the parameters (weights and biases) of the network using an optimization algorithm like gradient descent:
        - Weight update: , where α is the learning rate.

$$w_{ij}^{(l)} \leftarrow w_{ij}^{(l)} - \alpha \frac{\partial L}{\partial w_{ij}^{(l)}}$$

        - Bias update: ![Bias update](**https://latex.codecogs.com/png.image?\dpi{150}&space;b_j^{(l)}&space;\leftarrow&space;b_j^{(l)}&space;-&space;\alpha&space;\frac{\partial&space;L**

▼ Unit 5

▼ Gradient Descent Deep Learning

Gradient descent is a fundamental optimization algorithm used in deep learning to train neural networks. It is an iterative optimization method that aims to minimize the loss function by updating the network's parameters based on the gradients of the loss with respect to those parameters. Here's how gradient descent works in the context of deep learning:

1. **Initialization**: The parameters of the neural network (weights and biases) are initialized randomly or using a specific initialization scheme.
2. **Forward Propagation**: An input sample is fed forward through the network, and the output is computed using the current parameter values. This involves a series of matrix multiplications and activation function applications in each layer.
3. **Loss Computation**: The output of the network is compared with the ground truth or target value to compute the loss. The choice of loss function depends on the specific task, such as mean squared error (MSE) for regression or cross-entropy loss for classification.
4. **Backward Propagation (Backpropagation)**: The gradients of the loss with respect to the parameters of the network are computed using the chain rule. This involves propagating the gradients backward through the network, layer by layer, starting from the output layer to the input layer.
5. **Parameter Update**: The parameters of the network are updated using the gradients computed during backpropagation. The update is performed by subtracting a fraction of the gradients from the current parameter values, scaled by a learning rate hyperparameter. The learning rate

determines the step size of the update and influences the convergence of the optimization process.

6. **Repeat Steps 2-5**: Steps 2 to 5 are repeated for a mini-batch or a single training sample, depending on the chosen training strategy (e.g., batch gradient descent, stochastic gradient descent, or mini-batch gradient descent). The process continues until all training samples have been processed (one pass through the entire dataset), known as an epoch.
7. **Convergence and Stopping Criteria**: The training process continues for a predefined number of epochs or until a convergence criterion is met. Convergence can be assessed based on the reduction in loss or the stability of the network's performance on a validation set.
8. **Model Evaluation**: Once the training is complete, the trained model can be used to make predictions on new, unseen data.

By iteratively adjusting the parameters based on the gradients of the loss, gradient descent aims to find a set of parameter values that minimize the loss function and improve the performance of the neural network on the given task.

It's important to note that variations of gradient descent, such as stochastic gradient descent (SGD) and its extensions (e.g., Adam, RMSprop), are commonly used in practice to enhance the training efficiency and convergence of deep neural networks. These variations introduce additional modifications to the parameter update process, such as momentum and adaptive learning rates, to achieve faster convergence and better generalization.

▼ CNN and RNN

Convolutional Neural Networks (CNN):
CNNs are a type of deep learning model specifically designed for processing grid-like data, such as images. They consist of multiple layers, including convolutional, pooling, and fully connected layers. CNNs leverage the concept of local receptive fields and weight sharing to extract hierarchical features from input data. They have been remarkably successful in various computer vision tasks, including image classification, object detection, and image segmentation.

Convolution:
Convolution is a fundamental operation in CNNs. It involves sliding a small filter (also known as a kernel) over the input data, computing element-wise multiplications and summations at each location. The result is a feature map that captures spatial relationships and local patterns within the data.

Padding and Strides:
Padding and strides are used in CNNs to control the spatial dimensions of feature maps. Padding involves adding extra pixels around the input image, which helps retain spatial information and prevent the reduction in feature map size. Strides determine the step size at which the convolutional filter is applied over the input data, influencing the output feature map's spatial resolution.

Pooling:
Pooling layers in CNNs are used to downsample feature maps and reduce their spatial dimensions. Max pooling is a commonly used technique where the input is divided into non-overlapping regions, and only the maximum value within each region is retained, discarding the rest. Pooling helps extract dominant features, reduce computational complexity, and provide translation invariance.

Backpropagation in CNN:
Backpropagation is an algorithm used to train CNNs. It involves propagating the error backward through the network, adjusting the weights and biases of the layers using gradient descent. During backpropagation, the gradients are calculated using the chain rule, allowing the model to update the parameters and improve its performance over iterations.

Recurrent Neural Network (RNN):
RNNs are a type of neural network designed to process sequential data, where the output of a previous step is fed as input to the current step. This recurrent structure enables RNNs to capture temporal dependencies and handle variable-length sequences. RNNs have been widely used in tasks such as natural language processing, speech recognition, and time series analysis.

Types of RNN:

1. *Vanilla RNN:* The basic form of RNN, where each step's hidden state depends only on the current input and the previous hidden state. However, it suffers from the vanishing gradient problem, limiting its ability to capture long-term dependencies.
2. *Long Short-Term Memory (LSTM):* LSTM addresses the vanishing gradient problem by introducing a more complex gating mechanism. It has a memory cell that allows the network to selectively retain or forget information over time, making it capable of capturing long-term dependencies.
3. *Gated Recurrent Unit (GRU):* GRU is similar to LSTM but with a simplified architecture. It combines the forget and input gates of LSTM into a single update gate. GRU has shown comparable performance to LSTM while being computationally more efficient.
4. *Bidirectional RNN (BiRNN):* BiRNN processes sequences in both forward and backward directions simultaneously. By considering past and future context, BiRNNs can capture dependencies from both directions, often leading to improved performance in tasks requiring comprehensive sequence understanding.

These variations of RNNs offer different architectural modifications and improvements to tackle challenges related to sequential data processing, enabling more effective learning and modeling of temporal dependencies.

▼ Q) Which is faster batch gd or stochastic gd (not considering convergence)?

In terms of raw computational speed, batch gradient descent (BGD) is generally faster than stochastic gradient descent (SGD). The reason for this is that BGD processes the entire training set in each iteration,

whereas SGD processes one training example or a small mini-batch of examples at a time.

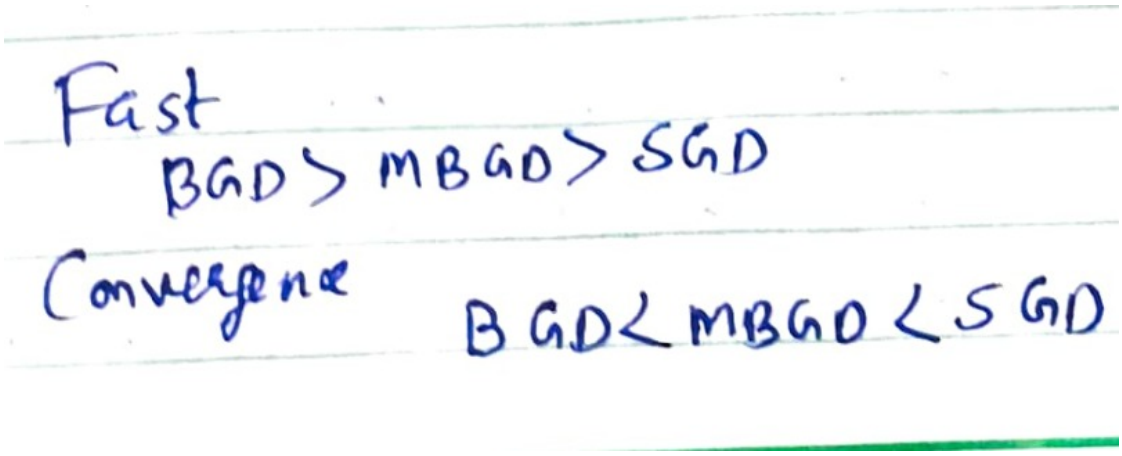
Here are a few reasons why BGD can be faster than SGD in terms of computational speed:

Vectorization: BGD can take advantage of efficient vectorized operations provided by libraries like NumPy. By processing the entire training set in parallel, BGD can leverage optimized linear algebra libraries and parallel computing capabilities, which can significantly speed up the computations.

Data Transfer Overhead: With BGD, the entire training set is loaded into memory once, and subsequent computations are performed on this data already residing in memory. This reduces the data transfer overhead between memory and processor, which can be a significant bottleneck, especially when dealing with large datasets.

Parallel Processing: BGD lends itself well to parallel processing, as computations can be distributed across multiple processors or threads. This parallelization can further speed up the computation, particularly on systems with multiple cores or GPUs.

It's important to note that the speed advantage of BGD in terms of computational efficiency does not necessarily translate to faster convergence or better generalization performance. SGD, with its more frequent updates, can often converge faster to a good solution, even though each update is computationally more expensive. So, while BGD may be faster in terms of raw computation, SGD's faster convergence and better generalization performance make it a popular choice in many cases, especially for large-scale deep learning tasks.



▼ Unit 1 chatgpt theory

Introduction of Machine Learning:
Machine Learning (ML) is a subset of Artificial Intelligence (AI) that focuses on the development of algorithms and models that enable computer systems to learn and make predictions or decisions without explicit programming. In traditional programming, humans write specific instructions for a computer to

perform tasks, but in machine learning, algorithms are trained on data to identify patterns and make predictions or take actions based on those patterns.

AI, ML, and DL:
Artificial Intelligence (AI) is a broad field that encompasses the development of intelligent machines capable of performing tasks that typically require human intelligence. Machine Learning (ML) is a subset of AI, focusing on algorithms and models that learn from data. Deep Learning (DL) is a specific subfield of ML that uses artificial neural networks to model and learn complex patterns.

Types of Machine Learning:

1. Supervised Learning: In supervised learning, the algorithm is trained on labeled data, where the input data is paired with corresponding output labels. The goal is to learn a mapping function that can predict labels for new, unseen data.
2. Unsupervised Learning: Unsupervised learning involves training algorithms on unlabeled data, where the algorithm must discover patterns or relationships in the data without any pre-existing knowledge. Clustering and dimensionality reduction are common tasks in unsupervised learning.
3. Reinforcement Learning: Reinforcement learning involves an agent interacting with an environment and learning to make optimal decisions through a trial-and-error process. The agent receives feedback in the form of rewards or penalties based on its actions and aims to maximize cumulative rewards.

Instance-Based Learning:
Instance-based learning, also known as lazy learning, is a type of machine learning where the algorithm stores the training instances and makes predictions based on the similarity between new instances and the stored training instances. It doesn't build an explicit model but retrieves and reuses stored instances for making predictions.

Model-Based Learning:
Model-based learning, also known as eager learning, involves building a model from the training data to make predictions. The model represents the patterns and relationships in the data and is used to generalize and make predictions on unseen instances.

Challenges in Machine Learning:

1. Data quality and preprocessing: Obtaining high-quality and relevant data, cleaning and preprocessing it to remove noise, missing values, and inconsistencies.
2. Overfitting and underfitting: Overfitting occurs when a model performs well on the training data but fails to generalize to unseen data. Underfitting occurs when a model is too simple to capture the underlying patterns in the data.
3. Feature selection and dimensionality: Selecting the most relevant features from the data and dealing with high-dimensional datasets that can lead to computational and performance issues.

4. Scalability: Handling large datasets efficiently and training models that can scale to handle increasing amounts of data.

Application of Machine Learning:
Machine Learning has a wide range of applications across various domains, including:

1. Image and speech recognition
2. Natural language processing and sentiment analysis
3. Fraud detection and cybersecurity
4. Recommender systems
5. Healthcare and medical diagnosis
6. Financial forecasting
7. Autonomous vehicles
8. Predictive maintenance
9. Customer segmentation and targeted marketing

Machine Learning Development Life Cycle (MLDLC):
The Machine Learning Development Life Cycle consists of several stages:

1. Data collection and preprocessing: Gathering relevant data and cleaning it by removing noise, handling missing values, and transforming it into a suitable format for analysis.
2. Feature engineering and selection: Identifying relevant features from the data that can contribute to the learning process and engineering new features if necessary.
3. Model selection and training: Choosing an appropriate algorithm or model architecture, dividing the data into training and validation sets, and training the model on the training data.
4. Model evaluation and tuning: Assessing the performance of the trained model using appropriate evaluation metrics. If the model's performance is not satisfactory, tuning the hyperparameters of the model to improve its performance.
5. Deployment: Integrating the trained model into a production environment where it can make predictions or decisions on new, unseen data. This may involve creating APIs, deploying the model on cloud platforms, or embedding it in applications or systems.
6. Monitoring and maintenance: Continuously monitoring the performance of the deployed model, analyzing its predictions, and updating the model as new data becomes available. It also involves handling issues such as concept drift or data drift that may affect the model's performance over time.

Fundamentals of Machine Learning:

1. Training data: Machine learning algorithms require labeled or unlabeled training data to learn patterns and make predictions. The quality and quantity of training data play a crucial role in the performance of the model.

2. Feature representation: Features are the individual measurable properties or characteristics of the data. Selecting relevant features and representing them in a suitable format is important for effective learning.

3. Learning algorithm: The learning algorithm is the core component of machine learning. It processes the training data, identifies patterns, and builds a model or makes predictions based on the data.

4. Generalization: Machine learning aims to build models that can generalize well to unseen data. Generalization refers to the ability of a model to perform accurately on new, unseen instances.

Univariate Analysis:
Univariate analysis involves analyzing and understanding the characteristics and patterns of a single variable. It focuses on examining the distribution, central tendency, dispersion, and other statistical properties of a single variable in isolation. Common techniques used in univariate analysis include histograms, box plots, and summary statistics.

Bivariate Analysis:
Bivariate analysis explores the relationship between two variables. It examines how changes in one variable are related to changes in another variable. It can involve analyzing correlations, scatter plots, cross-tabulations, or computing statistical measures such as correlation coefficients.

Multivariate Analysis:
Multivariate analysis involves analyzing the relationships and interactions among multiple variables simultaneously. It aims to understand how variables collectively affect each other. Multivariate analysis techniques include principal component analysis (PCA), factor analysis, and cluster analysis.

These analysis techniques help in gaining insights, identifying patterns, and understanding the relationships within the data, which can be useful for feature selection, model building, and decision-making in machine learning.

▼ Unit 2 chatgpt theory

Linear Regression:
Linear regression is a statistical modeling technique used to establish a linear relationship between a dependent variable and one or more independent variables. It assumes a linear function between the variables, allowing for prediction or estimation of the dependent variable based on the independent variables. The goal is to find the best-fit line that minimizes the difference between the predicted and actual values.

Regression Metrics:
Regression metrics are used to evaluate the performance of regression models. Some commonly used metrics include:

1. Mean Absolute Error (MAE): MAE measures the average absolute difference between the predicted and actual values. It provides a measure of the average magnitude of errors.

2. Mean Squared Error (MSE): MSE measures the average squared difference between the predicted and actual values. It amplifies larger errors

compared to MAE.

3. Root Mean Squared Error (RMSE): RMSE is the square root of MSE, providing a measure of the average magnitude of errors in the original scale of the dependent variable.
4. R-squared (R^2): R-squared represents the proportion of variance in the dependent variable that is explained by the independent variables. It ranges from 0 to 1, with a higher value indicating a better fit.
5. Adjusted R-squared: Adjusted R-squared is a modified version of R-squared that adjusts for the number of predictors in the model. It penalizes overfitting and provides a more accurate measure of the model's goodness of fit.

Multiple Regression:
Multiple regression extends linear regression to include multiple independent variables. It models the relationship between the dependent variable and multiple predictors simultaneously. The goal is to estimate the coefficients of the independent variables and assess their significance in predicting the dependent variable.

Gradient Descent:
Gradient descent is an optimization algorithm used to minimize the loss function and find the optimal values of the model's parameters. It iteratively updates the parameter values in the direction of the steepest descent of the loss function gradient.

Batch Gradient Descent:
Batch gradient descent computes the gradient of the loss function with respect to the entire training dataset. It updates the model's parameters after evaluating the gradient for the entire dataset. It can be computationally expensive for large datasets.

Stochastic Gradient Descent:
Stochastic gradient descent computes the gradient of the loss function with respect to a single training example at each iteration. It updates the model's parameters immediately after processing each training example. It is computationally more efficient but may result in noisy updates.

Mini-Batch Gradient Descent:
Mini-batch gradient descent is a combination of batch gradient descent and stochastic gradient descent. It computes the gradient based on a small batch of training examples at each iteration. It strikes a balance between the efficiency of stochastic gradient descent and stability of batch gradient descent.

Polynomial Regression:
Polynomial regression is a form of regression analysis where the relationship between the independent and dependent variables is modeled as an nth-degree polynomial. It allows for a non-linear relationship between the variables by introducing polynomial terms.

Bias and Variance:
Bias refers to the error introduced by approximating a real-world problem with

a simplified model. It measures the model's tendency to consistently underpredict or overpredict the true values.

Variance refers to the variability in model predictions for different training datasets. It measures the sensitivity of the model to changes in the training data. High variance can indicate overfitting, where the model is too complex and captures noise in the training data.

Regularization Techniques:
Regularization techniques are used to prevent overfitting and improve the generalization of machine learning models. They add additional constraints or penalties to the loss function to control the model's complexity.

Regularization Methods:

1. Lasso Regression (L1 regularization): Lasso regression adds an L1 penalty to the loss function, encouraging sparsity in the model by forcing some of the coefficients to become exactly zero. It performs feature selection by eliminating irrelevant or less important features.
2. Ridge Regression (L2 regularization): Ridge regression adds an L2 penalty to the loss function, which encourages smaller coefficient values. It reduces the impact of individual features without completely eliminating them.
3. Elastic Net Regression: Elastic Net regression combines both L1 and L2 regularization. It adds a linear combination of L1 and L2 penalties to the loss function, allowing for feature selection while also handling multicollinearity among the predictors.

Logistic Regression:
Logistic regression is a classification algorithm used to predict the probability of a binary or multiclass outcome. It models the relationship between the independent variables and the log-odds of the outcome using a logistic function. Logistic regression is widely used in various fields, including medical diagnosis, fraud detection, and sentiment analysis.

Classification Evaluation Metrics:
Classification evaluation metrics are used to assess the performance of classification models. Some commonly used metrics include:

1. Accuracy: Accuracy measures the proportion of correctly classified instances out of the total instances. It provides an overall measure of the model's correctness.
2. Confusion Matrix: A confusion matrix is a table that summarizes the performance of a classification model by showing the counts of true positives, true negatives, false positives, and false negatives.
3. Precision: Precision measures the proportion of correctly predicted positive instances out of the total predicted positive instances. It indicates the model's ability to avoid false positives.
4. Recall (Sensitivity or True Positive Rate): Recall measures the proportion of correctly predicted positive instances out of the total actual positive instances. It indicates the model's ability to identify all positive instances.

5. F1-Score: F1-score is the harmonic mean of precision and recall. It provides a balanced measure of the model's performance by considering both precision and recall.
6. Macro and Weighted F1-Score: Macro F1-score calculates the F1-score for each class independently and then takes the average. Weighted F1-score calculates the F1-score for each class and weights it by the number of instances in each class.

Softmax Regression or Multinomial Logistic Regression:
Softmax regression, also known as multinomial logistic regression, is an extension of logistic regression for multiclass classification. It models the relationship between the independent variables and the probabilities of each class using the softmax function. Softmax regression assigns a probability distribution over the classes and predicts the class with the highest probability.

Polynomial Logistic Regression:
Polynomial logistic regression extends logistic regression by incorporating polynomial terms of the independent variables. It allows for non-linear decision boundaries in classification tasks by capturing higher-order interactions between the variables.

▼ Unit 3 chatgpt theory

Dimensionality Reduction:
Dimensionality reduction is the process of reducing the number of input variables or features in a dataset. It is used to overcome the curse of dimensionality, improve computational efficiency, and eliminate irrelevant or redundant features. The goal is to retain the most informative features while reducing noise and preserving the structure of the data.

Subset Selection:
Subset selection is a dimensionality reduction technique that involves selecting a subset of the original features from the dataset. The selected subset is typically chosen based on certain criteria, such as relevance to the target variable or predictive power. It can be performed using various methods like forward selection, backward selection, or a combination of both.

Forward Selection:
Forward selection is a feature selection technique that starts with an empty set of features and iteratively adds the most relevant feature to the subset based on some evaluation criterion. The process continues until a stopping criterion is met or all features are included in the subset.

Backward Selection:
Backward selection is a feature selection technique that starts with a full set of features and iteratively removes the least relevant feature from the subset based on an evaluation criterion. The process continues until a stopping criterion is met or only a desired number of features remain in the subset.

Principal Component Analysis (PCA):
Principal Component Analysis is a popular dimensionality reduction technique that transforms a high-dimensional dataset into a lower-dimensional space while preserving the most important patterns or variances in the data. It

achieves this by projecting the data onto a new set of orthogonal axes called principal components. The principal components are ranked in order of the amount of variance they explain in the original data.

Linear Discriminant Analysis (LDA):
Linear Discriminant Analysis is a dimensionality reduction technique that aims to find a linear combination of features that maximizes class separability in a supervised learning setting. LDA seeks to project the data onto a lower-dimensional space while maximizing the between-class scatter and minimizing the within-class scatter.

Fisher's Criterion:
Fisher's criterion, also known as Fisher's linear discriminant, is a measure used in LDA to evaluate the separability between classes. It quantifies the ratio of between-class scatter to within-class scatter, indicating how well the classes are separated in the feature space.

t-Distributed Stochastic Neighbor Embedding (t-SNE):
t-SNE is a dimensionality reduction technique commonly used for visualizing high-dimensional data. It aims to represent the complex relationships and structures of the data in a lower-dimensional space, typically 2D or 3D. t-SNE preserves the local structure of the data by modeling the similarity between data points in the original space and the lower-dimensional space.

Kullback-Leibler Divergence:
Kullback-Leibler (KL) divergence is a measure of dissimilarity or divergence between two probability distributions. It is used in various machine learning tasks, including dimensionality reduction, to compare the similarity or difference between the distributions of the original data and the reduced or transformed data.

These techniques and measures play important roles in handling high-dimensional data, reducing feature space, and extracting meaningful information to improve the efficiency and performance of machine learning models.

▼ Unit 4 chatgpt theory

Introduction to Deep Learning:
Deep Learning is a subfield of Machine Learning that focuses on building and training neural networks with multiple layers to learn hierarchical representations of data. It is inspired by the structure and function of the human brain and aims to model complex patterns and relationships in data. Deep Learning has revolutionized various fields, including computer vision, natural language processing, and speech recognition.

Types of Neural Networks:

1. Feedforward Neural Networks (FNN): FNNs are the simplest form of neural networks, where information flows only in one direction, from input to output layer. They are used for tasks such as classification and regression.
2. Convolutional Neural Networks (CNN): CNNs are widely used for image and video processing tasks. They leverage specialized layers like convolutional layers, pooling layers, and fully connected layers to learn spatial hierarchies and extract meaningful features from images.

3. Recurrent Neural Networks (RNN): RNNs are designed to handle sequential data, such as time series or natural language data. They have feedback connections that allow information to flow in cycles, enabling them to capture temporal dependencies.
4. Long Short-Term Memory (LSTM) Networks: LSTMs are a type of RNN that can better handle long-term dependencies by utilizing memory cells and gating mechanisms. They are commonly used in tasks involving sequential data with long-range dependencies.
5. Generative Adversarial Networks (GAN): GANs consist of two neural networks, a generator and a discriminator, which compete against each other. GANs are used for tasks like generating synthetic data, image-to-image translation, and data augmentation.

McCulloch-Pitts Neuron Model:
The McCulloch-Pitts Neuron Model is a simplified model of a biological neuron, proposed by Warren McCulloch and Walter Pitts in 1943. It represents a binary logic gate that takes binary inputs and produces binary outputs. The model considers the weighted sum of inputs and applies a threshold function to determine the output.

Boolean Functions Using M-P Neuron:
The McCulloch-Pitts Neuron Model can be used to implement basic boolean functions such as AND, OR, and NOT gates. For example, the AND gate can be implemented by setting appropriate weights and a threshold such that the output is 1 only when both inputs are 1.

The Perceptron:
The Perceptron is a type of neural network model developed by Frank Rosenblatt in the late 1950s. It consists of a single layer of McCulloch-Pitts neurons with adjustable weights. The Perceptron can learn to classify linearly separable patterns and adjust its weights through a process called supervised learning.

Logistic Regression using Perceptron:
Logistic regression is a popular classification algorithm that can be implemented using a Perceptron. By using an appropriate activation function (such as the sigmoid function), the Perceptron can be trained to output probabilities and make predictions based on them.

Activation Functions:
Activation functions introduce non-linearities into neural networks and help in learning complex patterns. Some common activation functions include:

1. Sigmoid function: Maps the input to a value between 0 and 1, commonly used in binary classification problems.
2. Rectified Linear Unit (ReLU): Returns 0 for negative inputs and the input value for positive inputs. ReLU is widely used in hidden layers of deep neural networks.
3. Hyperbolic tangent (tanh): Similar to the sigmoid function, but maps the input to a value between -1 and 1.
4. Softmax: Used in the output layer for multi-class classification problems, the softmax function converts a vector of values into a probability

distribution.

Multilayer Perceptron (MLP):
A Multilayer Perceptron (MLP is a type of feedforward neural network that consists of multiple layers of neurons, including an input layer, one or more hidden layers, and an output layer. Each neuron in the network is connected to neurons in the previous and next layers. MLPs can learn complex non-linear relationships in data and are widely used for various tasks, including classification, regression, and pattern recognition.

Multilayer Forward Propagation:
In multilayer forward propagation, the input data is fed through the network layer by layer in a forward direction to compute the output. Each neuron in a layer receives inputs from the previous layer, applies a weighted sum of inputs, and passes the result through an activation function to produce its output. This process is repeated for each layer until the output layer is reached, and the final output of the network is obtained.

Back Propagation: Input, Output, and Hidden Layer Computation:
Backpropagation is a training algorithm for neural networks that adjusts the weights of the network based on the difference between the predicted output and the desired output. It involves two main steps: forward propagation and backward propagation.

1. Forward propagation: The input data is fed through the network using the multilayer forward propagation process, as described above, to compute the output of the network.
2. Backward propagation: The error between the predicted output and the desired output is calculated. The error is then propagated back through the network, starting from the output layer and moving backward. The error is used to adjust the weights of the neurons in each layer, aiming to minimize the overall error.

During backpropagation, the computation of errors and weight adjustments is performed in three main stages:

- Output layer computation: The error at the output layer is computed by comparing the predicted output with the desired output. This error is used to adjust the weights connecting the output layer to the previous hidden layer.
- Hidden layer computation: The error is propagated backward from the output layer to the hidden layers. The error at each hidden layer is computed based on the contribution of the neurons in that layer to the overall error.
- Weight adjustment: The computed errors are used to adjust the weights of the connections between neurons in each layer. This adjustment is typically done using gradient descent optimization algorithms, such as the gradient descent algorithm or its variations (e.g., stochastic gradient descent).

Memorization:
In the context of deep learning, memorization refers to the phenomenon where a neural network learns to simply memorize the training data without truly understanding the underlying patterns. Memorization can occur when the network

has excessive capacity, leading to overfitting. In such cases, the network performs well on the training data but fails to generalize to unseen data.

Memorization can be undesirable as it compromises the network's ability to make accurate predictions on new, unseen data. Techniques such as regularization, early stopping, and data augmentation are commonly employed to prevent or mitigate the problem of memorization and encourage the network to learn meaningful representations and generalize better.

▼ Unit 5 chatgpt theory

Gradient Descent in Deep Learning:
Gradient descent is an optimization algorithm commonly used in deep learning to update the parameters of a neural network during training. It works by iteratively adjusting the parameters in the direction of steepest descent of the loss function with respect to those parameters. This process continues until convergence or until a specified number of iterations is reached. There are different variants of gradient descent, such as stochastic gradient descent (SGD), mini-batch gradient descent, and Adam optimizer, which incorporate additional techniques to improve convergence speed and stability.

Vanishing Gradient Problem:
The vanishing gradient problem refers to the issue where the gradients become extremely small as they propagate backward through many layers in a deep neural network during training. This problem can hinder the learning process, especially in deep networks with many layers, as the updates to the parameters become negligible. Consequently, the early layers of the network may not learn effectively, leading to slower convergence or even no learning at all. This problem is particularly prevalent in recurrent neural networks (RNNs) due to the repeated application of the same weights during backpropagation.

Early Stopping:
Early stopping is a technique used during training to prevent overfitting in a neural network. It involves monitoring the performance of the model on a validation dataset during training and stopping the training process when the validation performance starts to deteriorate. By stopping the training at this point, we can avoid overfitting, where the model becomes too specialized to the training data and performs poorly on unseen data.

Dropout Layer:
Dropout is a regularization technique commonly used in neural networks, including deep learning models. It aims to prevent overfitting by randomly setting a fraction of the output features of a layer to zero during training. This effectively turns off some neurons during each training step, which forces the network to learn more robust and redundant representations. Dropout helps in reducing the interdependencies between neurons, which can lead to more generalized and less overfit models.

Regularization in Deep Learning:
Regularization is a technique used to prevent overfitting in machine learning models, including deep learning models. It introduces additional constraints or penalties on the model's parameters during training to discourage complex and overfitting solutions. Common regularization techniques in deep learning include L1 and L2 regularization, which add penalties based on the magnitudes

of the weights to the loss function. Regularization helps in reducing the model's complexity and makes it more robust to noise and outliers in the data.

CNN (Convolutional Neural Network):
Convolutional Neural Networks (CNNs) are a class of deep learning models designed specifically for processing structured grid-like data, such as images and audio. CNNs are widely used in computer vision tasks. They are composed of convolutional layers, pooling layers, and fully connected layers. The convolutional layers use filters to perform convolutions over the input data, capturing local patterns and features. Pooling layers downsample the spatial dimensions, reducing the computational complexity and extracting the most salient features. Fully connected layers perform the final classification or regression based on the extracted features.

Convolution, Padding, and Strides:
In the context of CNNs, convolution refers to the mathematical operation of applying a filter (also known as a kernel or a feature detector) to the input data. The filter slides across the input data, performing element-wise multiplications and summations, producing a feature map. Convolutional layers use multiple filters to capture different features in the input.

Padding is the process of adding additional border pixels to the input data before performing convolutions. It helps in retaining the spatial dimensions of the input and avoids the reduction of spatial resolution that occurs during convolutions. Padding can be "valid" (no padding) or "same" (adding padding to preserve spatial dimensions).

Strides determine the step size at which the filter slides or moves across the input data during convolution. A stride of 1 means the filter moves one pixel at a time, resulting in output with the same spatial dimensions as the input. A stride of 2 means the filter moves two pixels at a time, reducing the spatial dimensions by a factor of 2. Strides larger than 1 can help in reducing the spatial dimensions and computational complexity of the network.

Pooling:
Pooling layers are commonly used in CNNs to reduce the spatial dimensions of the feature maps generated by the convolutional layers. Pooling helps in extracting the most important features while reducing the sensitivity to small spatial variations. The two most common types of pooling are max pooling and average pooling. Max pooling selects the maximum value within a pooling window, while average pooling calculates the average value within the window. Pooling windows slide over the feature maps, reducing the spatial dimensions based on the pooling size and stride.

Backpropagation in CNN:
Backpropagation is the process of computing the gradients of the loss function with respect to the parameters of the neural network. In CNNs, backpropagation is used to update the weights of the convolutional layers and fully connected layers during training. The gradients are computed by applying the chain rule, starting from the output layer and propagating the errors backward through the network. The computed gradients are then used to update the weights using an optimization algorithm such as gradient descent.

Recurrent Neural Network (RNN):
Recurrent Neural Networks (RNNs) are a type of neural network architecture

designed for sequential data processing. Unlike feedforward networks, RNNs have feedback connections, allowing them to process input sequences of variable lengths. RNNs maintain an internal state or memory that captures information about past inputs, which is updated and passed along as the network processes each new input. This memory allows RNNs to capture temporal dependencies and context in the data, making them suitable for tasks such as natural language processing and speech recognition.

Types of RNN:
There are different variants of RNNs, each with specific architectural modifications to address the challenges of training recurrent networks. Some common types of RNNs include:

1. Vanilla RNN: The basic form of RNN, where the hidden state is updated using a simple recurrent connection. However, vanilla RNNs suffer from the vanishing gradient problem and have difficulty capturing long-term dependencies.
2. Long Short-Term Memory (LSTM): LSTM addresses the vanishing gradient problem by introducing a memory cell and three gating mechanisms: input gate, forget gate, and output gate. These gates control the flow of information and help LSTM networks retain or forget information over long sequences.
3. Gated Recurrent Unit (GRU): GRU is another variant of RNN that tackles the vanishing gradient problem and improves the capability to capture long-term dependencies. It combines the forget and input gates of the LSTM into a single update gate and merges the cell state and hidden state.

These variants of RNNs have been successful in capturing temporal dependencies and have been widely used in various applications involving sequential data.

▼ Sessional 1 -

CEN-809

Roll No. 19

B.Tech. (Computer Engineering) VIII Semester

FIRST Sessional Examination, 2023

MACHINE LEARNING TECHNIQUES

Time: One Hours Paper No. CEN-809 Maximum Marks: 15

Instructions: Attempt all questions. Marks are indicated against each question.

C.O.	S.No.	QUESTIONS	M.M.
CO1	Q1	Explain the life cycles of machine learning?	3
CO2	Q2	Find a multiple regression model for the following data using Matrices formulation:	4

x1	4	7	2
x2	5	2	6
Y(ouiput)	6	11	4

CO2	Q3	Find the approximate linear regression line using Gradient Descent technique for the following linear regression model. Assume the initial value of m_0=10, b_0=0 and learning rate =0.1. (NOTE :Apply only two iteration)	3

X	3	4	5
Y	6	7	8

CO2	Q4	Write shorts notes on (a) Stochastic Gradient Descent (b) Ridge Regression (c) Elastic Net Regression (d) R Squared vs Adjusted R-squared error (e) Bias-Variance Trade-Off	5

SUBJECT :- Machine Learning Assignment

Title CEN-809 Date 7/05/23 Page No. 1

Name = SURYA KANT ROLL No. - 19BCS060

Q1. Explain Machine Learning development life cycle. (MLDLC).

The Machine Learning Development Life Cycle (MLDLC) is a systematic and iterative process that guides the development and deployment of machine learning models. It encompasses various stages and tasks that data scientists and machine learning engineers follow to create effective and reliable models. The MLDLC typically consists of the following phases:

1. Problem Definition: In this initial phase, the problem to be solved using machine learning is defined. The goals, objectives, and requirements of the project are identified, along with the ~~requirements of the defined~~ available resources and constraints.
2. Data Acquisition and preparation:- In this phase, relavant data for training the machine learning model is collected. This includes data gathering, data cleaning, data integration, and feature engineering. The data is examined for quality, completeness, and consistency to ensure its suitability for the task at hand
3. Exploratory Data Analysis (EDA): EDA involves analyzing and visualizing the collected data to gain insights and a deeper understanding of its characteristics. This phase helps in identifying patterns, correlation, correlations, outliers, and any other relavant information that can guide subsequent steps.
4. Model Selection: Based on the problem statement and the available data, a suitable machine learning algorithm or a combination of algorithm or a combination of algorithms is selected. The selection is based on factors such as the type of problem (classification, regression, etc), the size and nature of the data, and the desired model performance.
5. Model Training: In this phase, the selected algorithm is applied to the training data to create a machine learning model. The model learns from the data and adjust its internal parameters to minimize the prediction errors. This process involves techniques like optimization algorithms, gradient descent, and backpropagation.

DELTA®

Title _____ Date _____ Page No. 2

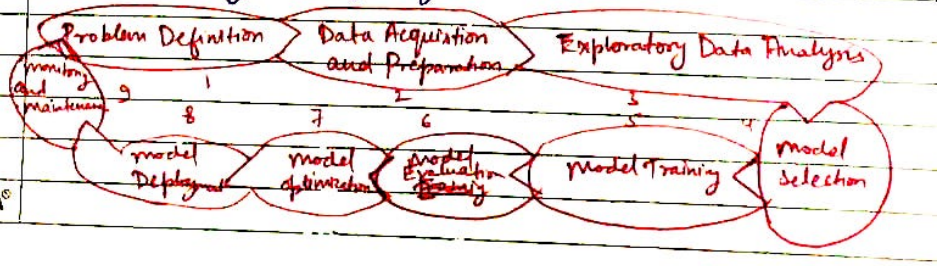
6. Model Evaluation: Once the model is trained, it needs to be evaluated to assess its performance and effectiveness. Evaluation metrics are defined based on the problem domain, such as accuracy, precision, recall or mean squared error. The model is tested on a separate dataset called the validation or test set to measure its generalization ability.

7. Model ~~Evaluation~~ Optimization :- If the model's performance is unsatisfactory, it may require optimization. This can involve fine-tuning the hyperparameters, exploring different algorithms, or adjusting the feature engineering process. The goal is to improve the model's performance and ensure it meets the desired quality standards.

8. Model Deployment: After achieving satisfactory performance, the model is ready to be deployed in a real-world environment where it can be used to make predictions on new, unseen data. This phase involves considerations such as scalability, reliability, and security. The deployment can be in the form of an API, a web app, or integration into existing systems.

3. Monitoring and Maintenance: Once the model is deployed, it needs to be continuously monitored to ensure its performed remains optimal. Monitoring involves tracking key metrics, detecting concept drift, handling data drift, and retraining or updating the model as needed. Ongoing maintenance and periodic reevaluation are essential to keep the model accurate and up-to-date.

The MLDLC is an iterative process, meaning that the steps are often revisited and refined based on the feedback and insights gained from previous stages. This iterative nature allows for continuous improvement of the models and ensures they remain effective in addressing the problem they were designed for.



DELTA

Q2 Find a multiple regression model for the following data using Matrices formulation:

x_1	4	7	2
x_2	5	2	6
Y (output)	6	11	4

$$X = \begin{bmatrix} 1 & 4 & 5 \\ 1 & 7 & 2 \\ 1 & 2 & 6 \end{bmatrix} \quad Y = \begin{bmatrix} 6 \\ 11 \\ 4 \end{bmatrix} \quad X^T = \begin{bmatrix} 1 & 1 & 1 \\ 4 & 7 & 2 \\ 5 & 2 & 6 \end{bmatrix}$$

$$B = (X^TX)^{-1}X^TY \Rightarrow Y = BX$$

$$X^TX = \begin{bmatrix} 1 & 1 & 1 \\ 4 & 7 & 2 \\ 5 & 2 & 6 \end{bmatrix} \cdot \begin{bmatrix} 1 & 4 & 5 \\ 1 & 7 & 2 \\ 1 & 2 & 6 \end{bmatrix} = \begin{bmatrix} 3 & 13 & 13 \\ 13 & 69 & 46 \\ 13 & 46 & 65 \end{bmatrix}$$

$$(X^TX)^{-1} = \begin{bmatrix} 2369/9 & -247/9 & -299/9 \\ -247/9 & 26/9 & 31/9 \\ -299/9 & 31/9 & 38/9 \end{bmatrix}$$

$$(X^TX^{-1}).X^T = \begin{bmatrix} 2369/9 & -247/9 & -299/9 \\ -247/9 & 26/9 & 31/9 \\ -299/9 & 31/9 & 38/9 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 4 & 7 & 2 \\ 5 & 2 & 6 \end{bmatrix} = \begin{bmatrix} -38/3 & 14/3 & 9 \\ 4/3 & -1/3 & -1 \\ 5/3 & -2/3 & -1 \end{bmatrix}$$

$$(X^TX^{-1})X^T.Y = \begin{bmatrix} -38/3 & 14/3 & 9 \\ 4/3 & -1/3 & -1 \\ 5/3 & -2/3 & -1 \end{bmatrix} \begin{bmatrix} 6 \\ 11 \\ 4 \end{bmatrix} = \begin{bmatrix} 34/3 \\ 1/3 \\ -4/3 \end{bmatrix}$$

$$B = \begin{bmatrix} B_0 \\ B_1 \\ B_2 \end{bmatrix} = \begin{bmatrix} 34/3 \\ 1/3 \\ -4/3 \end{bmatrix} = \begin{bmatrix} 11.33 \\ 0.33 \\ -1.33 \end{bmatrix} \Rightarrow \boxed{y = 11.33 + 0.33x_1 - 1.33x_2}$$

Ans 3 -

DATE
PAGE

Q2

$b_{new} = b_{old} - \eta \times Error$ $\quad m_{new} = m_{old} - \eta \times Error$

$L = MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - m_{old}x_i - b_{old})$

$\frac{dL}{db} = -\frac{2}{n}\sum(y_i - m_{old}x_i - b_{old})$

$\frac{dL}{dm} = -\frac{2}{n}\sum x_i(y_i - m_{old}x_i - b_{old})$

x_i	3	4	5
y_i	6	7	8

$b_{new} = b_{old} + \frac{2\eta}{n}\sum(y_i - m_{old}x_i - b_{old})$

$m_{new} = m_{old} + \frac{2\eta}{n}\sum x_i(y_i - m_{old}x_i - b_{old})$

(i) $m_0 = 10$, $b_0 = 0$ $\quad \boxed{\eta = 0.1}$ not changed

$m_{new} = 10 + \frac{2 \times 0.1}{3}\left(3(6 - 10\times3 - 0) + 4(7 - 10\times4 - 0) + 5(8 - 10\times5 - 0)\right)$

$= 10 + \frac{0.2}{3}(3x - 24 + 4x - 33 + 5x - 42)$

$= 10 + \frac{0.2}{3}(-414)$

$= 10 - 27.6 \Rightarrow \boxed{m_1 = -17.6}$

$b_{new} = 0 + \frac{2}{3} \times 0.1\left(6 - 10\times3 - 0 + 7 - 10\times4 - 0 + 8 - 10\times5 - 0\right)$

$= \frac{0.2}{3} \times (-24 - 33 - 42)$

$= \frac{0.2}{3} \times -99 = -6.6$

$\Rightarrow \boxed{b_1 = -6.6}$

$m_1, b_1 = \boxed{-17.6, -6.6}$

(ii) $m_1 = -17.6 \qquad b_1 = -6.6$

$$m_2 = -17.6 + \frac{2}{3} \times 0.1 \left(3(6 - 3 \times (-17.6) - 6.6) + 4(7 - 4 \times (8 - 17.6) - (-6.6)) + 5 \times (8 - 5 \times (-17.6) - (6.6)) \right)$$

$$= -17.6 + \frac{0.2}{3} \left(3 \times 65.4 + 4 \times 84 + 5 \times 102.6 \right)$$

$$= -17.6 + \frac{0.2}{3} \times (1045.2)$$

$$= -17.6 + 69.68$$

$$m_2 = 52.08$$

$$b_2 = -6.6 + \frac{2}{3} \times 0.1 \left(6 - 3 \times (-17.6) + 6.6 + 7 + 4 \times 17.6 + 6.6 + 8 + 5 \times 17.6 + 6.6 \right)$$

$$= -6.6 + \frac{0.2}{3} \times (69.4 + 84 + 102.6)$$

$$= -6.6 + \frac{0.2}{3} \times 252$$

$$= -6.6 + 16.8$$

$$b_2 = 10.2$$

$m_2, b_2 = 52.08, 10.2$

η must be small

Q4 Short notes :-

a) Stochastic Gradient Descent.

SGD is stochastic in nature i.e. it picks up a 'random' instance of training data at each step and then computes the gradient, making it much faster as there is much fewer data to manipulate at a single time, unlike Batch GD.

for i in range(m)

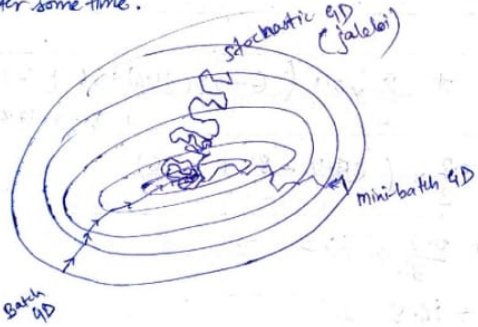
$$\theta_j = \theta_j - \alpha(\hat{y}_i - y_i)x_{ji}$$

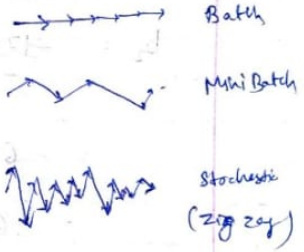
Batch GD:

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)x_{ji}$$

There is a downside of Stochastic nature of SGD i.e. once it reaches close to the minimum value then it doesn't settle down, instead bounces around which gives us a good value for model parameters but not optimal which can be solved by reducing the learning rate at each step which can reduce the bouncing and SGD might settle down at global minimum after some time.





b) Ridge Regression

Ridge Regression or shrinkage makes use of L2 regularization. In the linear regression objective function we try to minimize the sum of squares of errors. In ridge regression we add a constraint on the sum of squares of the regression coefficients. Thus in ridge regression our objective function is :-

$$\text{Min}\left(\underbrace{\Sigma\varepsilon^2}_{\text{error}} + \lambda\Sigma\beta^2\right) = \text{Min}\,\Sigma\left(y - (\beta_1 + \beta_2 X_2 + \beta_3 X_3 + \ldots + \beta_k X_k)\right)^2 + \lambda\Sigma\beta^2$$

Here, λ is the regularisation parameter which is a non negative number. Here we do not assume normality on the error terms.

On solving above objective func we get the estimates of β as :- $\hat{\beta} = (X^TX + \lambda I)^{-1}X^TY$ (I → identity matrix.)

c) **Elastic Net Regression**

Elastic Net regression is preffered over both ridge and lasso regression when one is dealing with highly correlated independant variables. It is a combination of both L1 and L2 regularization. The objective function in case of Elastic Net Regression is :-

$$\text{Min}\left(\sum \epsilon^2 + \lambda_1 \sum \beta^2 + \lambda_2 \sum |\beta|\right) = \text{Min} \sum \left(y - (\beta_1 + \beta_2 x_2 + \dots \beta_k x_k)\right)^2 + \lambda_1 \sum \beta^2 + \lambda_2 \sum |\beta|$$

λ_2 (Ridge) λ_1 (Lasso)

Like ridge and lasso, it does not assume normality.

d) **R Squared vs Adjusted R-squared error.**

R^2 determines how much of the total variation in Y (dependant variable) is explained by the variation in X (independent variable.) (R^2 is b/w 0 and 1

$$\text{R-square} = 1 - \frac{\sum (Y_{actual} - Y_{predicted})^2}{\sum (Y_{actual} - Y_{mean})^2} = 1 - \frac{SS_{regression}}{SS_{total}}$$

$$= 1 - \frac{SS_r}{SS_m}$$

SS_r = squared sum error of regression

SS_m = sq sum error of mean line

The only drawback of R^2 is that if new predictors (x) are added to our model, R^2 only increases or remains constant but it never decreases. We can't judge that by increasing complexity. So we use Adjusted R square. R^2_a is the modified form of R^2 that has been adjusted for the number of predictors in the model. It incorporates model's degree of freedom. The adjusted R^2 only increases if the new term improves the model accuracy. While in R^2, it assumes that while adding more data variance of data increases. But it was a problem because when we add an irrelavant feature in the dataset then at that time R^2 sometimes starts increasing increasily.

$$R^2_a = 1 - \left(\left[\frac{n-1}{n-k-1} \right] \times (1 - R^2) \right)$$

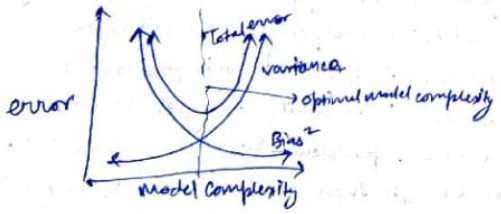
n = no. of observations
k = no. of independent variables
R^2_a = adjusted R square

Now as K increases by adding some features so the denominator will decrease n-1 remains constant. R^2 score will remain constant or increase slightly so the complete answer will increase and when we subtract this from one then the resultant score will decrease. So this is the case when we add an irrelavant feature in the dataset.

And if we add relavant features then R_2 score will increase and $1-R2$ will decrease heavily and the denominator $(n-k-1)$ will also decrease so the complete term decreases, and on substracting from one the score increases. Hence, this metric becomes one of the most important metrics to use during the evaluation of the model.

e) Bias-variance trade-off.

While building the machine learning model, it is really important to take care of bias and variance in order to avoid overfitting and underfitting in the Model. If the model is very simple with fewer parameters, it may have low variance and high bias. Whereas, if the model has large no of parameters, it will have high variance and low bias. So, it is required to make a balance between bias and variance errors, and thus this balance between the bias error and variance error is known as the Bias-Variance trade-off.



For an accurate prediction of the model, algorithms need a low variance and low bias. But this not possible because bias and variance are related to each other:
- If we decrease variance, it will increase bias
- If we decrease bias, it will increase the variance.

A high variance algo may perform well with training data. But it may lead to overfitting to noisy data. Whereas high bias algo generates a much simple model that may not even capture important regularities in the data. So, to make an optimal model we find a sweet spot between bias and variance i.e. Bias-variance tradeoff.

▼ Sessional 2 -

CEN-809

B.Tech. (Computer Engineering) VIII Semester
SECOND Sessional Examination, 2023
MACHINE LEARNING TECHNIQUES
Paper No. CEN-809

Roll No.....

Time: One Hours

Maximum Marks: 15

Instructions: Attempt all questions. Marks are indicated against each question.

C.O.	S.No.	QUESTIONS	M.M.
CO3	Q1	Given the data in the following table, compute the principal component using PCA Algorithm to reduce the dimension from 2D space to 1D space. Table 1.1 Feature / Example 1 / Example 2 X1 / 2 / 3 X2 / 3 / 1	4
CO4	Q2	How to solve Logistic regression using perceptron method. Explain in details	3
CO4	Q3	Derive expressions for the weights and thresholds that can compute the table 1.2 input-output mappings. Using (i) McCulloch-Pitts neuron (ii) Perceptron model Table 1.2 in1 / in2 / out 0 / 0 / 1 0 / 1 / 0 1 / 0 / 0 1 / 1 / 0	4
CO3	Q4	Given the data in the above table 1.1 apply the Linear Discriminant Analysis algorithm to reduce the dimension from 2D space to 1D space. OR Write shorts notes on ANY TWO (a) t-Distributed Stochastic Neighbor Embedding (b) Kullback–Leibler divergence (c) Different types of Neural Network	4

Principal component analysis

DATE
PAGE

Q1 PCA algorithm :-

Feature	Example 1	Example 2
X1	2	3
X2	3	1

Step 1 : mean calculation :-

$$\bar{x}_1 = \frac{2+3}{2} = 2.5 \qquad \bar{x}_2 = \frac{3+1}{2} = 2$$

Step 2: covariance matrix

	x_1	x_2
x_1	$cov(x_1,x_1)$	$cov(x_1,x_2)$
x_2	$cov(x_2,x_1)$	$cov(x_2,x_2)$

$$cov(x_1,x_1) = \frac{1}{n-1}\sum (x_i-\bar{x}_1)^2 = \frac{1}{2-1}\left((2-2.5)^2+(3-2.5)^2\right) = 0.25+0.25 = 0.5$$

$$cov(x_1,x_2) = cov(x_2,x_1) = \frac{1}{n-1}\sum (x_{1i}-\bar{x}_1)(x_{2i}-\bar{x}_2) = \frac{1}{2-1}(2-2.5)(3-2)+(3-2.5)(1-2)$$

$$= -0.5 - 0.5 = -1$$

$$cov(x_2,x_2) = \frac{1}{n-1}\sum (x_{i_2}-\bar{x}_2)^2 = \frac{1}{2-1}\left((3-2)^2+(1-2)^2\right) = 1+1 = 2$$

$$\text{Covariance Matrix, } S = \begin{bmatrix} 0.5 & -1 \\ -1 & 2 \end{bmatrix}$$

Step 3: Eigen vector and Eigen values

$$|[S-\lambda I]| = 0$$

$$\left| \begin{bmatrix} 0.5 & -1 \\ -1 & 2 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right| = 0$$

$$\begin{vmatrix} 0.5-\lambda & -1 \\ -1 & 2-\lambda \end{vmatrix} = 0 \Rightarrow (0.5-\lambda)(2-\lambda) - (-1)(-1) = 0$$

$$1 - \frac{\lambda}{2} - 2\lambda + \lambda^2 - 1 = 0$$

$$\lambda^2 - \frac{5\lambda}{2} = 0$$

eigen values $\Rightarrow \lambda = 0, \frac{5}{2}$

$$(S-\lambda I)U_1 = 0$$

$$\begin{bmatrix} 0.5-\lambda & -1 \\ -1 & 2-\lambda \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\Rightarrow (0.5-\lambda)u_1 - u_2 = 0$$

$$(0.5-2.5)u_1 = u_2$$

$$\Rightarrow \frac{u_1}{1} = \frac{u_2}{-(2)} = t$$

$$u_1 = 1, \quad u_2 = -2$$

eigen vector $\Rightarrow U = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$

Step 4 unit eigen vector $= \frac{U}{\|U\|} = \frac{\begin{bmatrix} 1 \\ -2 \end{bmatrix}}{\sqrt{1^2+2^2}}$

$$e_1 = \frac{1}{\sqrt{5}} \begin{bmatrix} 1 \\ -2 \end{bmatrix} \Rightarrow e_1 = \begin{bmatrix} 1/\sqrt{5} \\ -2/\sqrt{5} \end{bmatrix} = \begin{bmatrix} 0.4472 \\ -0.8944 \end{bmatrix}$$

Step 5 First Principal Component :-

$$PC_{1_k} = e_1^T \begin{bmatrix} x_{1k} - \bar{x}_1 \\ x_{2k} - \bar{x}_2 \end{bmatrix}$$

$$PC_{1_1} = \begin{bmatrix} 0.4472 & -0.8944 \end{bmatrix} \begin{bmatrix} 2-2.5 \\ 3-2 \end{bmatrix} = \begin{bmatrix} 0.4472 & -0.8944 \end{bmatrix} \begin{bmatrix} -0.5 \\ 1 \end{bmatrix}$$

$$= -0.2236 - 0.8944$$

$$= -1.1180$$

$$PC_{1_2} = \begin{bmatrix} 0.4472 & -0.8944 \end{bmatrix} \begin{bmatrix} 3-2.5 \\ 1-2 \end{bmatrix} = \begin{bmatrix} 0.4472 & -0.8944 \end{bmatrix} \begin{bmatrix} 0.5 \\ -1 \end{bmatrix}$$

$$= 0.2236 + 0.8944$$

$$= 1.1180$$

$\Rightarrow$ **PC1 = [-1.118 | 1.118]**

DATE
PAGE

Q2 How to solve logistic regression using perceptron method.

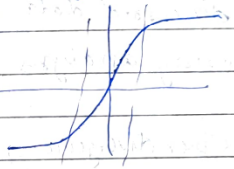
Logistic regression is a supervised learning algorithm that can be used to predict the probability of a binary outcome. The perceptron is a simple neural network that can be used to solve logistic regression problem.

by training a perceptron to predict the probability of a binary outcome.

The neural network input is $z = \sum x_i w_i = w^T x$
$= w_0 x_0 + w_1 x_1 + w_2 x_2 \dots$

The input is transformed using the activation function which generates values as probabilies from 0 to 1:

$$g(z) = \frac{1}{1+e^{-z}}, \quad g'(z) = g(z)(1-g)$$



By combining all above, we can formulate the hypothesis function for our classification problem $\rightarrow$ $h_{w(x)} = \frac{1}{1+e^{-w^T x}}$

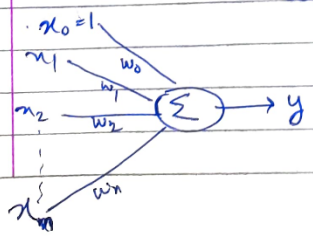
Cross entropy loss function: $L = \frac{1}{m} -y \log \hat{y} - (1-y)\log(1-\hat{y})$
(log loss function) — maximum likelihood (mL)
(for 1 row) $L = \frac{-1}{m} y \log \hat{y} + (1-y)\log(1-\hat{y})$

$\Rightarrow$ cost function (L) $= \frac{-1}{m} \sum y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)$
(for all rows)

now we come to gradient, there the algorithm converges towards global minimum and hence Gradient descent

$$w_{new} = w_{old} - \eta \frac{(y-\hat{y})}{m} x$$

η → learning rate, $\frac{dL}{dw}$



Q3 Derive expressions for the weights and thresholds that can compute the table input-output mappings.

in1	in2	out (NOR)
0	0	1
0	1	0
1	0	0
1	1	0

Ans (i) McCulloh – Pitts neuron model

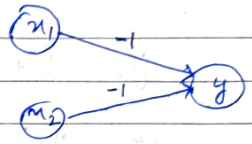
NOR functions returns a true value (1) if both inputs are 0

Let threshold of unit Y is 0

$y_{in} = x_1 w_1 + x_2 w_2$

$= -1 x_1 + -1 x_2$

$y_{in} = -x_1 - x_2$



$$y = f(y_{in}) = \begin{cases} 1; \text{ if } y_{in} \geq 0 \\ 0; \text{ if } y_{in} < 0 \end{cases}$$

Presenting the input

(i) $x_1 = 0, x_2 = 0 \Rightarrow y_{in} = -0 - 0 = 0 \geq 0 \Rightarrow y = 1$

(ii) $x_1 = 0; x_2 = 1 \Rightarrow y_{in} = -0 - 1 = -1 < 0 \Rightarrow y = 0$

(iii) $x_1 = 1, x_2 = 0 \Rightarrow y_{in} = -1 - 0 = -1 < 0 \Rightarrow y = 0$

(iv) $x_1 = 1, x_2 = 1 \Rightarrow y_{in} = -1 - 1 = -2 < 0 \Rightarrow y = 0$

Thus NOR function is realised.

(ii) Perceptron model, without bias convergence does not occur $\Rightarrow$ let $b = 0$ and $w_1 = 0$ & $w_2 = 0$, $\alpha = 1$, $\theta = 0$

Input (in1) x_1	Input (in2) x_2	b	Target (out) t
0	0	1	1
0	1	1	0
1	0	1	0
1	1	1	0

given input			Net	output	given Target	weight changes			weights		
x_1	x_2	b	y_{in}	y	t	Δw_1	Δw_2	Δb	$w_1=0$	$w_2=0$	$b=0$
0	0	1	0	0	1	0	0	1	0	0	1
0	1	1	1	1	0	0	0	0	0	0	1
1	0	1	1	1	0	0	0	0	0	0	1
1	1	1	1	1	0	0	0	0	1	0	1

Initially $b=0$, $w_1=0$, $w_2=0$, $\alpha=1$, $\theta=0$

for $(x_1, x_2) \rightarrow (0,0)$

$x = (0,0)$

$y_{in} = b + \sum x_i w_i = b + 0*w_1 + 0*w_2 = 0+0+0 = 0$

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

$\Rightarrow y = 0$

if $t=y$ do nothing

else $(t \neq y) \Rightarrow w_{i(new)} = w_{i old} + \alpha t x_i$, $b_{new} = b_{old} + \alpha t$

$t=1$, $y=0$

$\Rightarrow t \neq y \Rightarrow w_{1new} = 0 + 1 \times 1 \times 0$ $\quad b_{new} = 0 + 1 \times 1$

$w_{1new} = 0$ $\quad b_{new} = 0+1$

$w_{2new} = 0 + 1 \times 1 \times 0$

$w_{2new} = 0$

$x = (0,1)$ $\quad y_{in} = 1 + 0 \times 0 + 1 \times 0 = 1$ $\quad \Rightarrow y = 1$

but $y \neq t$ as $t=0$

$\Rightarrow w_{1new} = 0 + 1 \times 0 \times 0 = 0$ $\quad b_{new} = 1 + 1 \times 0$

$w_{2new} = 0 + 1 \times 0 \times 1 = 0$ $\quad = 1$

DATE
PAGE

'et's change weights and bias (initial) and try again

given input			Net	output	Target given	weight changes			weights		
x_1	x_2	b	y_{in}	y	t	Δw_1	Δw_2	Δb	$w_1=1$	$w_2=1$	$b=0$
0	0	1	0	0	1	0	0	1	1	1	1
0	1	1	2	1	0	0	0	0	1	1	1
1	0	1	2	1	0	0	0	0	1	1	1
1	1	1	3	1	0	0	0	0	1	1	1

initially $b=0, w_1=1, w_2=1$, $\alpha=1$, $\theta=0$

$x=(0,0)$ $y_{in} = 0 + (0\times1 + 0\times1) = 0$

$\Rightarrow y = 0$

$t=1 \Rightarrow y \neq t \Rightarrow$ $w_{1new} = 1 + 1\times1\times0 = 1$ $b_{new} = 0 + 1\times1 = 1$

$w_{2new} = 1 + 1\times1\times0 = 1$

$x=(0,1)$ $y_{in} = 1 + 0\times1 + 1\times1 = 2$

$\Rightarrow y = 1$

$t=0 \Rightarrow y \neq t \Rightarrow w_{1new} = 1 + 1\times0\times0 = 1$ $b_{new} = 1 + 1\times0 = 1$

$w_{2new} = 1 + 1\times0\times1 = 1$

Q4) LDA (Linear Discriminant Analysis)

DATE
PAGE

X_1	(2,2)	(3,3)
X_2	(3,3)	(1,1)

Step1 mean $\mu_1 = \left(\frac{2+3}{2}, \frac{2+3}{2}\right) = (2.5, 2.5)$

$\mu_2 = \left(\frac{3+1}{2}, \frac{3+1}{2}\right) = (2, 2)$

Step2: Scatter matrix within class

$$S_1 = \sum (x_1 - \mu_1)(x_1 - \mu_1)^T$$

$$(x - \mu_1) = \begin{bmatrix} 2-2.5 & 3-2.5 \\ 2-2.5 & 3-2.5 \end{bmatrix} = \begin{bmatrix} -0.5 & 0.5 \\ -0.5 & 0.5 \end{bmatrix}$$

$$S_1 = \begin{bmatrix} -0.5 & 0.5 \\ -0.5 & 0.5 \end{bmatrix}\begin{bmatrix} -0.5 & -0.5 \\ 0.5 & 0.5 \end{bmatrix} = \begin{bmatrix} 0.25+0.25 & 0.25+0.25 \\ 0.25+0.25 & 0.25+0.25 \end{bmatrix}$$

$$= \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$$

$$S_1 = \frac{S_1}{2} = \begin{bmatrix} 0.25 & 0.25 \\ 0.25 & 0.25 \end{bmatrix}$$

$$S_2 = \sum (x_2 - \mu_2)(x_2 - \mu_2)^T$$

$$(x_2 - \mu_2) = \begin{bmatrix} 3-2 & 1-2 \\ 3-2 & 1-2 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}$$

$$S_2 = \frac{S_2}{2} = \frac{1}{2}\begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1+1 & 1+1 \\ 1+1 & 1+1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$\Rightarrow S_2 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$S_w = S_1 + S_2 = \begin{bmatrix} 0.25 & 0.25 \\ 0.25 & 0.25 \end{bmatrix} + \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1.25 & 1.25 \\ 1.25 & 1.25 \end{bmatrix}$$

Step3 Scatter matrix Between Class

$$S_B = (M_1 - M_2)(M_1 - M_2)^T$$

$$M_1 - M_2 = \begin{bmatrix} 2.5-2 \\ 2.5-2 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

$$S_B = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \begin{bmatrix} 0.5 & 0.5 \end{bmatrix} = \begin{bmatrix} 0.25 & 0.25 \\ 0.25 & 0.25 \end{bmatrix}$$

Step4 :- LDA projection vector

$w \Rightarrow$ projection vector

$$S_W^{-1} S_B w = \lambda w$$

$$\left| S_W^{-1} S_B - \lambda I \right| = 0$$

$$S_W^{-1} S_B = \begin{bmatrix} 1.25 & 1.25 \\ 1.25 & 1.25 \end{bmatrix}^{-1} \begin{bmatrix} 0.25 & 0.25 \\ 0.25 & 0.25 \end{bmatrix} = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

inverse does not exist as $\det(S_W) = 0$

w = eigen vector = $[S_W]^{-1} [M_1 - M_2]$

Step5 Dimension reduction

$y = w^T x$ (w^T: projection vector; x: input data samples)

Q4 Short Notes

a) t-Distributed Stochastic Neighbor Embedding

t-SNE reduces the dimensionality of high dimensional data while preserving the local structure of the data. It aims to maintain the distance bw neighbouring points, ensuring points are mapped closely together. It transforms the data into lower-dimensional space (usually 2D-3D) that can be easily visualised.

Unlike PCA, t-SNE is a non-linear embedding method. It can capture complex relationships and non-linear structures.

t-SNE uses a probabilistic approach to create a low-dimensional representation of data.

While t-SNE provides valuable insights into the structure of high dimensional data, it should be used for visualization and exploratory analysis only rather than quantitative analysis.

b) Kullback-Leibler divergence, also known as relative entropy, is a measure of how one probability distribution differs from another. It is named after Solomon kullback and Richard Leibler, who first published it in 1951.

KL divergence is defined as the expected value of the logarithm of the ratio of 2 probability distributions. $D_{KL}(P||Q) = \sum_{x \in X} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$

$P \rightarrow$ 1st prob. distribution
$Q \rightarrow$ 2nd prob distri.
$x \rightarrow$ possible outcome

KL divergence is a non-symmetric measure, which means the $D_{KL}(P||Q) \neq D_{KL}(Q||P)$ because, KL divergence measure the information lost when P is used to approximate Q, but it does not measure the information lost when Q is used to approximate P.

▼ Assignment -

Assignment

Q1	*Explain* Machine Learning development life cycle (MLDLC).
Q2	*Find a multiple regression model for the following data using Matrices formulation:* ***x1*** 4 7 2 ***x2*** 5 2 6 ***Y(output)*** 6 11 4
Q3	What is Curse of Dimensionality?
Q4	Explain the Logistic regression using perceptron logic.
Q5	Explain: (a) Vanishing/Exploding Gradient Problem (b) Early stopping (c) How to in improve the performance of an ANN

SUBJECT :- Machine Learning Assignment

Title CEN-809 Date 7/05/23 Page No. 1

Name = SURYA KANT ROLL No. - 19BCS060

Q1. Explain Machine Learning development life cycle. (MLDLC).

The Machine Learning Development Life Cycle (MLDLC) is a systematic and iterative process that guides the development and deployment of machine learning models. It encompasses various stages and tasks that data scientists and machine learning engineers follow to create effective and reliable models. The MLDLC typically consists of the following phases:

1. Problem Definition: In this initial phase, the problem to be solved using machine learning is defined. The goals, objectives, and requirements of the project are identified, along with the ~~requirements of the defined~~ available resources and constraints.
2. Data Acquisition and preparation:- In this phase, relavant data for training the machine learning model is collected. This includes data gathering, data cleaning, data integration, and feature engineering. The data is examined for quality, completeness, and consistency to ensure its suitability for the task at hand
3. Exploratory Data Analysis (EDA): EDA involves analyzing and visualizing the collected data to gain insights and a deeper understading of its characteristics. This phase helps in identifying patterns, correlation, correlations, outliers, and any other relavant information that can guide subsequent steps.
4. Model Selection: Based on the problem statement and the available data, a suitable machine learning algorithm or a combination of algorithm or a combination of algorithms is selected. The selection is based on factors such as the type of problem (classification, regression, etc), the size and nature of the data, and the desired model performance.
5. Model Training: In this phase, the selected algorithm is applied to the training data to create a machine learning model. The model learns from the data and adjust its internal parameters to minimize the prediction errors. This process involves techniques like optimization algorithms, gradient descent, and backpropagation.

DELTA®

Title _____ Date _____ Page No. 2

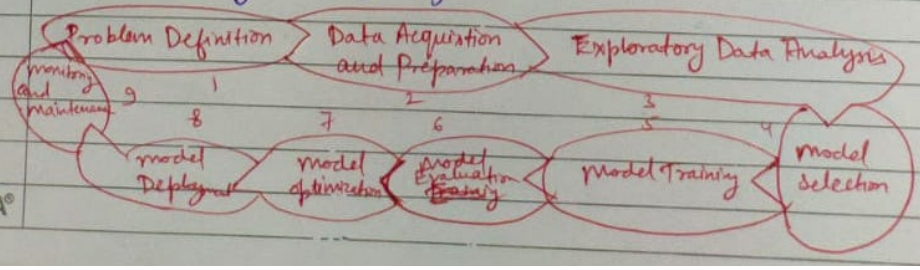
6. Model Evaluation: Once the model is trained, it needs to be evaluated to assess its performance and effectiveness. Evaluation metrics are defined based on the problem domain, such as accuracy, precision, recall or mean squared error. The model is tested on a separate dataset called the validation or test set to measure its generalization ability.

7. Model ~~Evaluation~~ Optimization :- If the model's performance is unsatisfactory, it may require optimization. This can involve fine-tuning the hyperparameters, exploring different algorithms, or adjusting the feature engineering process. The goal is to improve the model's performance and ensure it meets the desired quality standards.

8. Model Deployment: After achieving satisfactory performance environment where it can be used to make predictions on new, unseen data. This phase involves considerations such as scalability, reliability, and security. The deployment can be in the form of an API, a web app, or integration into existing systems.

9. Monitoring and Maintenance: Once the model is deployed, it needs to be continuously monitored to ensure its performed remains optimal. Monitoring involves tracking key metrics, detecting concept drift, handling data drift, and retraining or updating the model as needed. Ongoing maintenance and periodic reevaluation are essential to keep the model accurate and up-to-date.

The MLDLC is an iterative process, meaning that the steps are often revisited and refined based on the feedback and insights gained from previous stages. This iterative nature allows for continous improved of the models and ensure they remain effective in addressing the problem they were designed for.



DELTA

Q2 Find a multiple regression model for the following data using Matrices formulation:

x_1	4	7	2
x_2	5	2	6
Y (output)	6	11	4

$$X = \begin{bmatrix} 1 & 4 & 5 \\ 1 & 7 & 2 \\ 1 & 2 & 6 \end{bmatrix} \quad Y = \begin{bmatrix} 6 \\ 11 \\ 4 \end{bmatrix} \quad X^T = \begin{bmatrix} 1 & 1 & 1 \\ 4 & 7 & 2 \\ 5 & 2 & 6 \end{bmatrix}$$

$$B = (X^TX)^{-1}X^TY \Rightarrow Y = BX$$

$$X^TX = \begin{bmatrix} 1 & 1 & 1 \\ 4 & 7 & 2 \\ 5 & 2 & 6 \end{bmatrix} \cdot \begin{bmatrix} 1 & 4 & 5 \\ 1 & 7 & 2 \\ 1 & 2 & 6 \end{bmatrix} = \begin{bmatrix} 3 & 13 & 13 \\ 13 & 69 & 46 \\ 13 & 46 & 65 \end{bmatrix}$$

$$(X^TX)^{-1} = \begin{bmatrix} 2369/9 & -247/9 & -299/9 \\ -247/9 & 26/9 & 31/9 \\ -299/9 & 31/9 & 38/9 \end{bmatrix}$$

$$(X^TX^{-1}).X^T = \begin{bmatrix} 2369/9 & -247/9 & -299/9 \\ -247/9 & 26/9 & 31/9 \\ -299/9 & 31/9 & 38/9 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 4 & 7 & 2 \\ 5 & 2 & 6 \end{bmatrix} = \begin{bmatrix} -38/3 & 14/3 & 9 \\ 4/3 & -1/3 & -1 \\ 5/3 & -2/3 & -1 \end{bmatrix}$$

$$(X^TX^{-1})X^T.Y = \begin{bmatrix} -38/3 & 14/3 & 9 \\ 4/3 & -1/3 & -1 \\ 5/3 & -2/3 & -1 \end{bmatrix} \begin{bmatrix} 6 \\ 11 \\ 4 \end{bmatrix} = \begin{bmatrix} 34/3 \\ 1/3 \\ -4/3 \end{bmatrix}$$

$$B = \begin{bmatrix} B_0 \\ B_1 \\ B_2 \end{bmatrix} = \begin{bmatrix} 34/3 \\ 1/3 \\ -4/3 \end{bmatrix} = \begin{bmatrix} 11.33 \\ 0.33 \\ -1.33 \end{bmatrix} \Rightarrow \boxed{y = 11.33 + 0.33x_1 - 1.33x_2}$$

Title _____ Date _____ Page No. 4

Q3 What is Curse of Dimensionality? In machine learning

The "Curse of Dimensionality" is a phenomenon in machine learning and statistics that refers to the difficulties and limitations that arise when working with high-dimensional data. It describes the adverse effects of having a large number of features or dimensions in a dataset, which can lead to a degradation in the performance and efficiency of machine learning algorithms. Here are some key aspects of the curse of Dimensionality

1) Data Sparsity: As the number of dimensions increases, the amount of available data required to maintain an adequate sampling density becomes exponentially larger. This means that in high-dimensional spaces, data points are typically far apart from each other, resulting in sparsity. Sparse data can make it challenging to extract meaningful patterns or relationships from the data.

2) Increased computational Complexity :- The computational cost of processing and analyzing high-dimensionality data grows rapidly as the number of dimensions increases. Many algorithms, such as distance-based methods like k-nearest neighbours, ~~suffers~~ suffers from increased computational complexity and become inefficient and impractical in high-dimensionality spaces.

3) Overfitting: High-dimensional data tends to have more room for complex structures and patterns. This can leads to overfitting, where a model becomes overly sensitive to the noise or random variations present in the data, resulting in poor generalization to unseen data. Overfitting becomes more likely as the number of dimensions increases, making it harder to distinguish between meaningful patterns and random noises

4) Increased Model Complexity: With large number of dimensions, the complexity and size of the model also increase. This can lead to a higher risk of model complexity, making model harder to interpret and understand. It may also require a larger amount of training data to estimate the model parameters accurately.

DELTA®

Title _____ Date _____

Page No. 5

5. Feature Selection and Dimensionality Reduction: The curse of dimensionality highlights the importance of feature selection and dimensionality reduction techniques. These methods aim to reduce the number of dimensions by selecting relevant features or transforming the data into a lower-dimensional representation while preserving important information. Dimensionality reduction methods like Principal Component Analysis (PCA) or t-SNE can help mitigate the curse of dimensionality by compressing the data while retaining key patterns.

To combat the Curse of Dimensionality, it is crucial to carefully consider the dimensionality of the data and its impact on the chosen machine learning algorithms. Feature engineering, dimensionality reduction techniques, and careful model selection are essential to mitigate the challenges associated with high-dimensional data and improve the performance and efficiency of machine learning models.

Q4 Explain the logistic regression using perceptron logic

Logistic regression is a popular machine learning algorithm used for binary classification tasks. While it is not directly related to the perceptron algorithm, we can draw some connections between the two.

The perceptron is a basic algorithm for binary classification that dates back to the input data points into two classes. The perceptron takes an input vector, applies weight to each feature, sums them up, and passes the result through an activation function (often a step function) to make a prediction.

Logistic regression, on the other hand, is a probabilistic algorithm that estimates the probability of an input belonging to a particular class. It does this by applying a logistic function (also known as the sigmoid function) to a linear combination of the input features and their corresponding weights. The sigmoid function maps the linear combination to a value between 0 and 1, which can be interpreted as the probability of belonging to the positive class.

Title _____ Date _____

Page No. 6

To understand the connection between logistic regression and the perceptron, we can view logistic regression as a more refined version of the perceptron with a different activation function.

In the perceptron algorithm, the step function is used as the activation func, which makes a hard binary decision (0 or 1). However, this func is not differentiable, and its output is not probabilistic. Logistic regression addresses these limitations by using the logistic (sigmoid) func as the activation function.

Mathematically, the logistic reg model can be represented as follows:-

$z = w_0 + w_1x_1 + w_2x_2 + \dots + w_nx_n$

where z is the linear combination of the input features and their corresponding weights, w_0 is the bias term and $w_1, w_2 \dots w_n$ are the weights associated with each feature $x_1, x_2 \dots x_n$.

The logistic function, denoted as $\sigma(z)$, is then applied to the linear combination to produce the predicted probability:

$y\_pred = \sigma(z) = \frac{1}{1+e^{-z}}$, where e is the base of natural log.

The logistic function squashes the output of the linear combination into the range [0, 1], values close to 1 indicates a higher probability of belonging to the positive class, while values close to 0 indicates a higher probability of belonging to the negative class.

During training, the logistic regression model is typically optimized using methods such as maximum likelihood estimation or gradient descent to find the optimal values for the weights that minimizes the difference between the predicted probabilities and the actual class labels.

In summary, while logistic regression and perceptron algorithm share the goal of binary classification, logistic regression uses the sigmoid function to produce probabilistic outputs, allowing for more nuanced predictions.

DELTA®

Tit

Date

Page No. 7

Q5 Explain: (a) Vanishing / Exploding Gradient Problem
(b) Early stopping
(c) How to improve the performance of an ANN

(a) Vanishing / Exploding Gradient Problem:

The vanishing / exploding gradient problem is a challenge that can occur during the training of deep neural networks, particularly those with many layers. It refers to the phenomenon where the gradients computed during the backpropagation process either become extremely small (vanishing gradient) or exponentially large (exploding gradient) as they propagate from the output layer to the input layer.

When gradients vanish, it means that the updates to the network's weights become negligible, resulting in slow or stalled learning. This issue often arises in deep networks with many layers, as the gradients can diminish exponentially during backpropagation.

On the other hand, when gradients explode, the become significantly large, causing unstable updates and leading to overshooting of optimal weights. This can make the training process unstable or even fail to converge.

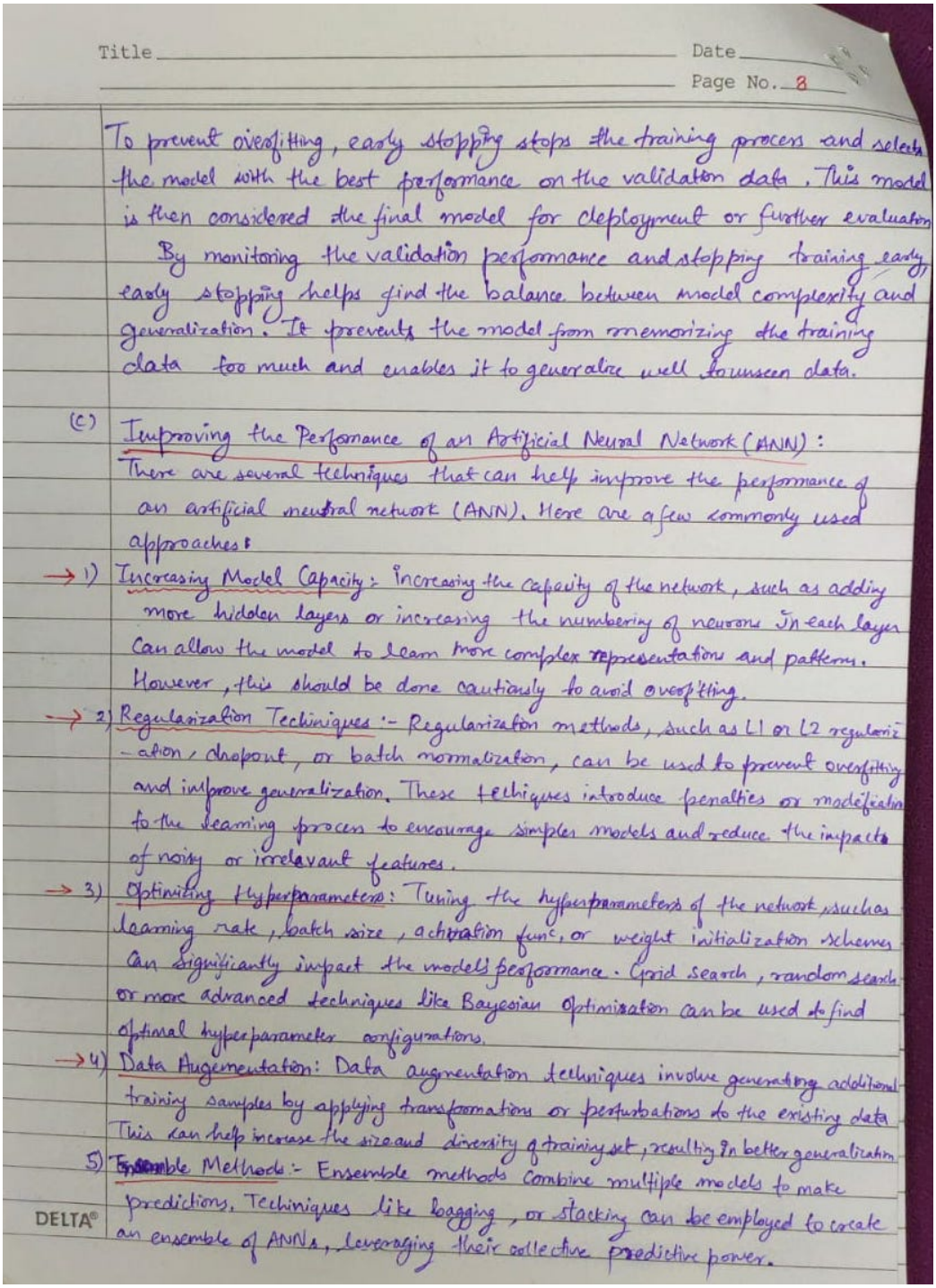
Both vanishing and exploding gradient problems hinder the training process and can prevent deep neural networks from effectively learning complex patterns and representations in the data.

(b) Early Stopping:-

Early stopping is a technique used during the training of machine learning models, including artificial neural networks (ANNs). It is a form of regularization that helps prevent overfitting by stopping the training process before the becomes too complex and starts to overfit the training data.

The idea behind early stopping is to monitor the performance of the model on a separate validate dataset during training. As the model improves on the training data, it should also demonstrate improved performance on the validation data. However, at some point, the model's performance on the validation data may start to deteriorate, indicating that it is starting to overfit.

DELTA®



Title _____ Date _____
Page No. 8

To prevent overfitting, early stopping stops the training process and selects the model with the best performance on the validation data. This model is then considered the final model for deployment or further evaluation.

By monitoring the validation performance and stopping training early, early stopping helps find the balance between model complexity and generalization. It prevents the model from memorizing the training data too much and enables it to generalize well to unseen data.

(c) Improving the Performance of an Artificial Neural Network (ANN):

There are several techniques that can help improve the performance of an artificial neural network (ANN). Here are a few commonly used approaches:

→ 1) Increasing Model Capacity: Increasing the capacity of the network, such as adding more hidden layers or increasing the numbering of neurons in each layer can allow the model to learn more complex representations and patterns. However, this should be done cautiously to avoid overfitting.

→ 2) Regularization Techniques: Regularization methods, such as L1 or L2 regularization, dropout, or batch normalization, can be used to prevent overfitting and improve generalization. These techniques introduce penalties or modifications to the learning process to encourage simpler models and reduce the impacts of noisy or irrelevant features.

→ 3) Optimizing Hyperparameters: Tuning the hyperparameters of the network, such as learning rate, batch size, activation func., or weight initialization schemes can significantly impact the model's performance. Grid search, random search or more advanced techniques like Bayesian optimization can be used to find optimal hyperparameter configurations.

→ 4) Data Augmentation: Data augmentation techniques involve generating additional training samples by applying transformations or perturbations to the existing data. This can help increase the size and diversity of training set, resulting in better generalization.

5) Ensemble Methods: Ensemble methods combine multiple models to make predictions. Techniques like bagging, or stacking can be employed to create an ensemble of ANNs, leveraging their collective predictive power.

DELTA®

▼ Formulae -

▼ for quadratic equation

this is for single x and single y column with quadratic relation

find a0, a1 and a2 by the three eq you get

y=a+a1*x+a2*x2

$$\sum y_i = na_0 + a_1(\sum x_i) + a_2(\sum x_i^2)$$
$$\sum y_i x_i = a_0(\sum x_i) + a_1(\sum x_i^2) + a_2(\sum x_i^3)$$
$$\sum y_i x_i^2 = a_0(\sum x_i^2) + a_1(\sum x_i^3) + a_2(\sum x_i^4)$$

▼ for multiple regression using matrix

$$\beta=(X'X)^{-1}\,X'y$$

We know that

$$X = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1k} \\ 1 & x_{21} & \cdots & x_{2k} \\ \vdots & \vdots & & \vdots \\ 1 & x_{n1} & \cdots & x_{nk} \end{bmatrix}$$

$$\beta = \begin{vmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{vmatrix} \qquad y = \begin{bmatrix} 6 \\ 11 \\ 4 \\ 3 \\ 5 \\ 9 \\ 10 \end{bmatrix} \qquad X = \begin{bmatrix} 1 & 4 & 5 & 4 \\ 1 & 7 & 2 & 3 \\ 1 & 2 & 6 & 4 \\ 1 & 1 & 9 & 6 \\ 1 & 3 & 4 & 5 \\ 1 & 7 & 3 & 4 \\ 1 & 8 & 2 & 5 \end{bmatrix}$$

$\lfloor 10 \rfloor$

After Solving $\beta=(X'X)^{-1}\,X'y$ we get

$$b = \begin{vmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{vmatrix} = \begin{vmatrix} 3.96239 \\ 1.06065 \\ 0.04396 \\ -0.48517 \end{vmatrix}$$

$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3$

$y = 3.96239 + 1.06065\,X_1 + 0.04396\,X_2 - 0.48517\,X_3$

▼ LDA

Linear Discriminant Analysis (LDA):

1. Compute the mean vectors of each class:
   - Mean vector of class k:
     , where nk is the number of samples in class k and $\mathbf{x}$ is a data point.

$$\mathbf{m}_k = \frac{1}{n_k} \sum_{\mathbf{x} \in C_k} \mathbf{x}$$

2. Compute the scatter matrices:
   - Within-class scatter matrix:

$$\mathbf{S}_W = \sum_{k=1}^{K} \sum_{\mathbf{x} \in C_k} (\mathbf{x} - \mathbf{m}_k)(\mathbf{x} - \mathbf{m}_k)^T$$

   - Between-class scatter matrix:
     , where $\mathbf{m}$ is the overall mean vector and K is the number of classes.

$$\mathbf{S}_B = \sum_{k=1}^{K} n_k (\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^T$$

3. Solve the generalized eigenvalue problem:
   - Construct the matrix $\mathbf{M}$ = $\mathbf{SW}$^(-1) $\mathbf{SB}$.
   - Compute the eigenvectors $\mathbf{w}_1$, $\mathbf{w}_2$, ..., $\mathbf{w}_d$ corresponding to the d largest eigenvalues of $\mathbf{M}$, where d is the desired reduced dimension.
4. Project the data onto the discriminant directions:
   - Reduced-dimensional representation:
     , where $\mathbf{x}$ is a data point and $\mathbf{y}$ is its reduced representation.

$$\mathbf{y} = [\mathbf{w}_1^T \mathbf{x}, \mathbf{w}_2^T \mathbf{x}, ..., \mathbf{w}_d^T \mathbf{x}]^T$$

Fisher's Criteria:

1. Compute the mean vectors of each class: Same as in LDA.
2. Compute the scatter matrices: Same as in LDA.
3. Solve the generalized eigenvalue problem:
   - Construct the matrix $\mathbf{M}$ = $\mathbf{SW}$^(-1) $\mathbf{SB}$.
   - Compute the eigenvectors $\mathbf{w}_1$, $\mathbf{w}_2$, ..., $\mathbf{w}_d$ corresponding to the d largest eigenvalues of $\mathbf{M}$, where d is the desired reduced dimension.
4. Project the data onto the projection vectors: Same as in LDA.

In both LDA and Fisher's criteria, the discriminant vectors $\mathbf{w}_1$, $\mathbf{w}_2$, ..., $\mathbf{w}_d$ are chosen based on the eigenvalues of the generalized eigenvalue problem.

The number of discriminant vectors to retain depends on the desired reduced dimension or the number of classes minus

▼ PCA

Certainly! Here are the formulas for Principal Component Analysis (PCA):

Principal Component Analysis (PCA):

1. Compute the mean vector of the data:
   - Mean vector:
     ![Mean vector](https://latex.codecogs.com/png.image?\\dpi{150}&space;\\mathbf{\\bar{x}}&space;=&space;\frac{1}{n}&space;\sum_{i=1}^{n}&space;\mathbf{x}_i), where n is the number of data points and $\mathbf{x}_i$ is a data point.
2. Compute the covariance matrix:
   - Covariance matrix:
     ![Covariance matrix](https://latex.codecogs.com/png.image?\\dpi{150}&space;\\mathbf{C}&space;=&space;\\frac{1}{n-1}&space;\\sum_{i=1}^{n}&space;(\\mathbf{x}_i&space;-&space;\\mathbf{\\bar{x}})(\mathbf{x}_i&space;-&space;\mathbf{\bar{x}})^T)
3. Perform eigendecomposition of the covariance matrix:
   - Compute the eigenvectors and eigenvalues of the covariance matrix:
     , where $\mathbf{V}$ is the matrix of eigenvectors and $\mathbf{D}$ is the diagonal matrix of eigenvalues.

$$\mathbf{C} = \mathbf{V}\mathbf{D}\mathbf{V}^T$$

4. Select the principal components:
   - Sort the eigenvalues in descending order and choose the d largest eigenvalues and their corresponding eigenvectors, where d is the desired reduced dimension.
5. Project the data onto the principal components:
   - Reduced-dimensional representation:
     , where $\mathbf{y}_i$ is the reduced representation of data point $\mathbf{x}_i$ and $\mathbf{V}_k$ is the matrix of the d largest eigenvectors.

$$\mathbf{y}_i = \mathbf{V}_d^T \mathbf{x}_i$$

PCA aims to find a new set of orthogonal axes, called principal components, that capture the maximum variance in the data. By selecting a subset of the principal components, you can achieve dimensionality reduction while preserving most of the important information in the data.

Please note that PCA assumes linearity and requires the data to be centered (subtracting the mean) before performing the analysis. Additionally, the number of principal components chosen affects the amount of variance retained in the reduced representation.

Materials -

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/25d9b77b-b5ea-434b-925f-db00bb133ced/Assignment_MLT_2023.pdf

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/bf473267-c4bd-41cc-a927-3901eb64c88e/Dimensionality_reduction.pdf

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/2cea2d2c-a33d-48a7-8be8-1c72bee11169/LDA_in_ML.pdf

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/d37cca56-03e2-44d5-bd33-9c9eb4ebb2d1/minhaaz_ml_notes.pdf

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/1c31b108-f09f-4d3c-b244-f95cc4b82e5f/ML_Notes.pdf

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/8194491b-b0d7-4efe-a536-40e7f2f1c80f/ML_NOTES-2.pdf

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/ca90863f-d024-4f10-a1df-426a793fa89b/ml_sess2_notes.pdf

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/24bf92de-c755-41b7-887c-2a0df3cfa1d6/NN_Book.pdf

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/5fb555a7-44e6-4d0e-9521-3d2e5254e0e2/PCA.pdf

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/92d8c965-b8d2-475e-8897-609e6fc6b95c/shahbuddin_ml_notes.pdf

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/57e68588-6802-44ae-911e-08ddd70fb5c1/surya_all_class_notes.pdf

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/972c4e93-b651-48fa-bbef-9d39a356898c/CamScanner_02-18-2023_20.30.09.pdf

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/cc800d96-5368-484a-964a-421a12e5e711/ML_NLP.pdf

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/39568c69-4e67-4677-9c38-c43bc71a7f0d/NLP_NETSEC.pdf

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/62976052-a6a1-4e98-b901-134a71f46528/surya_all_class_notes.pdf