INSPIRED
BY PURPOSE

ACE College of Engineering
(Approved by AICTE, Affiliated to KTU)
An ISO 9001: 2015 certified institution
Promoted by Manarul Huda Trust
Thiruvallam P.O., Thiruvananthapuram - 695 027, Kerala, India.
+91 471 7117777 / 87 / 97 info@acetvm.com www.acetvm.com

PEECT413
MACHINE LEARNING



Semester: 4
Electronics and Communication Engineering

Prepared by: Hima Vijayan V P,
Assistant Professor in ECE

PEECT413 Machine Learning

S4 ECE, 2024 Scheme KTU

Module 1

Module 1 Syllabus

Supervised, unsupervised machine learning techniques, dimensionality reduction techniques-PCA, SVD Instance-Based vs Model-Based Learning, Machine Learning models, Hyper parameters, regularization, Training - Batch and Online Learning, Challenges of Machine Learning: Data Issues-Quality, Relevancy, Over fitting, under fitting. Bias, variance, Performance metrics: Accuracy Recall, Precision, ROC curve

1 Introduction

In today's world, technology is growing very fast, and we are getting in touch with different new technologies day by day. Here, one of the booming technologies of computer science is Artificial Intelligence which is ready to create a new revolution in the world by making intelligent machines. Artificial Intelligence is composed of two words Artificial and Intelligence, where Artificial defines "man-made," and intelligence defines "thinking power", hence AI means "a man-made thinking power." So, we can define AI as: "It is a branch of computer science by which we can create intelligent machines which can behave like a human, think like humans, and able to make decisions." Artificial Intelligence exists when a machine can have human based skills such as learning, reasoning, and solving problems.

1.1 Why Artificial Intelligence is Important?

- With the help of AI, you can create such software or devices which can solve real-world problems very easily and with accuracy such as health issues, marketing, traffic issues, etc.
- With the help of AI, you can create your personal virtual Assistant, such as Cortana, Google Assistant, Siri, etc.
- With the help of AI, you can build such Robots which can work in an environment where survival of humans can be at risk.

Dept. of ECE, ACETVM

- AI opens a path for other new technologies, new devices, and new Opportunities.

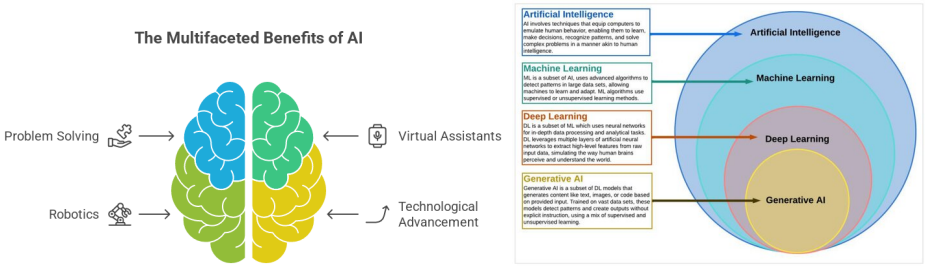
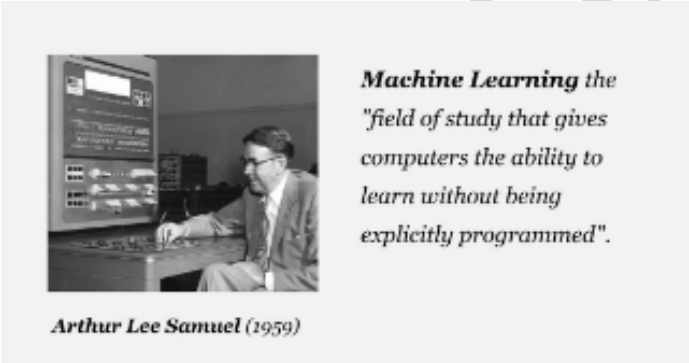


Figure 1: The Multifaceted Benefits of AI



Arthur Samuel defined machine learning as "the field of study that gives computers the ability to learn without being explicitly programmed". Coined in 1959, his definition emphasized that this learning happens through experience, with computers improving their performance over time by analyzing data and adjusting their programs accordingly. A famous example was his checkers-playing program, which improved by playing tens of thousands of games against itself.

Machine Learning is the science (and art) of programming computers so they can learn from data. Here is a slightly more general definition:

1. Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed —Arthur Samuel, 1959
2. A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E. —Tom Mitchell, 1997

Spam filter is a Machine Learning program that, given examples of spam emails (e.g., flagged by users) and examples of regular (nonspam, also called "ham") emails, can learn to flag spam. T

Dept. of ECE, ACETVM

The examples that the system uses to learn are called the training set. Each training example is called a training instance (or sample).

In this case, the task T is to flag spam for new emails, the experience E is the training data, and the performance measure P needs to be defined; for example, you can use the ratio of correctly classified emails. This particular performance measure is called accuracy, and it is often used in classification tasks. If you just download a copy of Wikipedia, your computer has a lot more data, but it is not suddenly better at any task. Thus, downloading a copy of Wikipedia is not Machine Learning

2 Why Use Machine Learning?

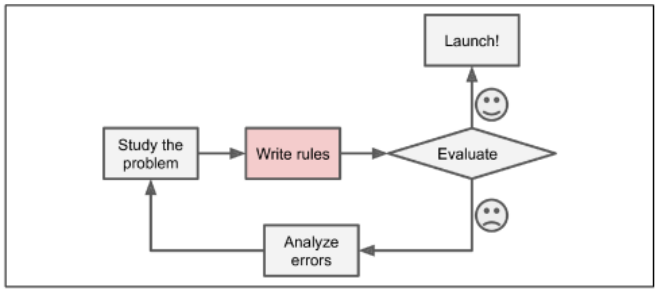


Figure 2: The traditional approach

Consider how you would write a spam filter using traditional programming techniques.

1. First you would consider what spam typically looks like. You might notice that some words or phrases (such as "4U," "credit card," "free," and "amazing") tend to come up a lot in the subject line. Perhaps you would also notice a few other patterns in the sender's name, the email's body, and other parts of the email.

2. You would write a detection algorithm for each of the patterns that you noticed, and your program would flag emails as spam if a number of these patterns were detected.

3. You would test your program and repeat steps 1 and 2 until it was good enough to launch.

Since the problem is difficult, your program will likely become a long list of complex rules—pretty hard to maintain.

In contrast, a spam filter based on Machine Learning techniques automatically learns which words and phrases are good predictors of spam by detecting unusually frequent patterns of words in the spam examples compared to the

Dept. of ECE, ACETVM

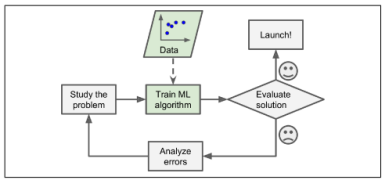


Figure 1-2. The Machine Learning approach

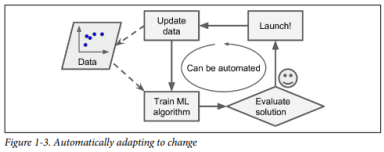


Figure 1-3. Automatically adapting to change

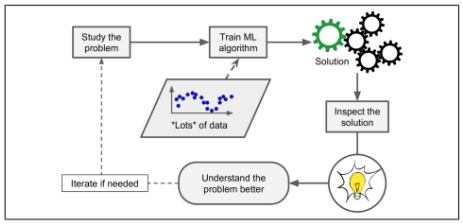


Figure 1-4. Machine Learning can help humans learn

ham examples . The program is much shorter, easier to maintain, and most likely more accurate.

Machine Learning can help humans learn . ML algorithms can be inspected to see what they have learned (although for some algorithms this can be tricky). For instance, once a spam filter has been trained on enough spam, it can easily be inspected to reveal the list of words and combinations of words that it believes are the best predictors of spam. Sometimes this will reveal unsuspected

Machine Learning is great for:

- Problems for which existing solutions require a lot of fine-tuning or long lists of rules: one Machine Learning algorithm can often simplify code and perform better than the traditional approach.
- Complex problems for which using a traditional approach yields no good solution: the best Machine Learning techniques can perhaps find a solution.
- Fluctuating environments: a Machine Learning system can adapt to new data.
- Getting insights about complex problems and large amounts of data.

3 Types of Machine Learning Systems

There are so many different types of Machine Learning systems that it is useful to classify them in broad categories, based on the following criteria:

Dept. of ECE, ACETVM

- Whether or not they are trained with human supervision (supervised, unsupervised, semisupervised, and Reinforcement Learning)
- Whether or not they can learn incrementally on the fly (online versus batch learning).
- Whether they work by simply comparing new data points to known data points, or instead by detecting patterns in the training data and building a predictive model, much like scientists do (instance-based versus model-based learning)

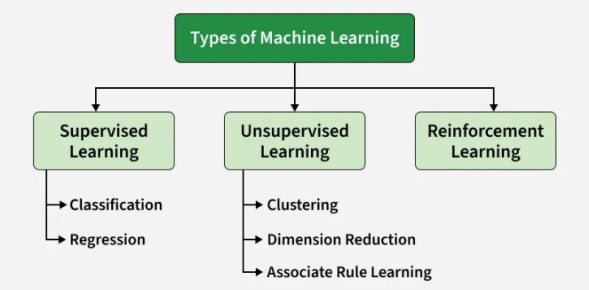


Figure 3: Types of Machine Learning Systems

3.1 Supervised

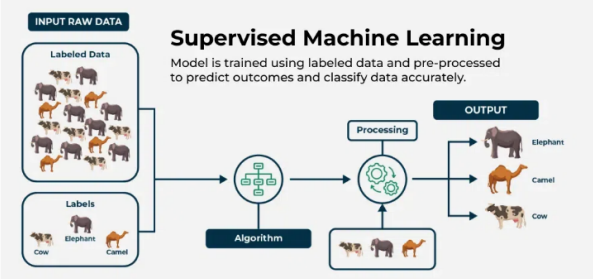


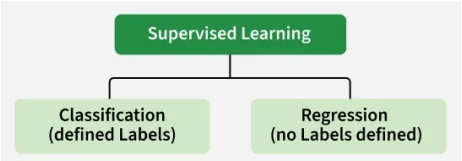
Figure 4: Supervised Learning

In supervised learning, the training set you feed to the algorithm includes the desired solutions, called labels. A typical supervised learning task is classification.

Supervised learning can be applied to two main types of problems:

- Classification: Where the output is a categorical variable (e.g., spam vs. non-spam emails, yes vs. no).

Dept. of ECE, ACETVM



- Regression: Where the output is a continuous variable (e.g., predicting house prices, stock prices).

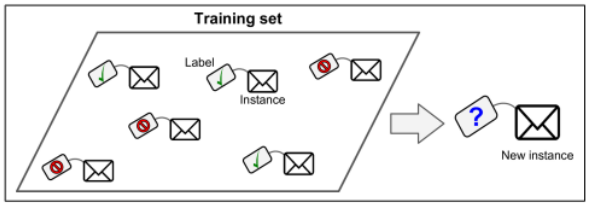


Figure 5: A labeled training set for spam classification (an example of supervised learning)

The spam filter is a good example of this: it is trained with many example emails along with their class (spam or ham), and it must learn how to classify new emails. Another typical task is to predict a target numeric value, such as the price of a car, given a set of features (mileage, age, brand, etc.) called predictors. This sort of task is called regression.To train the system, you need to give it many examples of cars, including both their predictors and their labels (i.e., their prices).

While training the model, data is usually split in the ratio of 80:20 i.e. 80% as training data and the rest as testing data. In training data, we feed input as well as output for 80% of data. The model learns from training data only. We use different supervised learning algorithms (which we will discuss in detail in the next section) to build our model.



Figure 6: A regression problem: predict a value, given an input feature (there are usually multiple input features, and sometimes multiple output values)

Dept. of ECE, ACETVM

3.1.1 Advantages

- Simplicity & clarity: Easy to understand and implement since it learns from labeled examples.
- High accuracy: When sufficient labeled data is available, models achieve strong predictive performance.
- Versatility: Works for both classification like spam detection, disease prediction and regression like price forecasting.
- Generalization: With enough diverse data and proper training, models can generalize well to unseen inputs.
- Wide application: Used in speech recognition, medical diagnosis, sentiment analysis, fraud detection and more.

3.1.2 Disadvantages

- Requires labeled data: Large amounts of labeled datasets are expensive and time-consuming to prepare.
- Bias from data: If training data is biased or unbalanced, the model may learn and amplify those biases.
- Overfitting risk: Model may memorize training data instead of learning general patterns, especially with small datasets.
- Limited adaptability: Performance drops significantly when applied to data distributions very different from training data.
- Not scalable for some problems: In tasks with millions of possible labels like natural language, supervised labeling becomes impractical.

Some of the most important supervised learning algorithms:

- k-Nearest Neighbors
- Linear Regression
- Logistic Regression
- Support Vector Machines (SVMs)
- Decision Trees and Random Forests
- Neural networks

Dept. of ECE, ACETVM

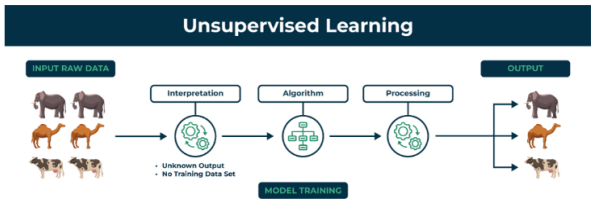


Figure 7: Unsupervised Machine Learning

3.2 Unsupervised learning

Unsupervised Learning works with unlabeled data, meaning there are no predefined outputs. The algorithm finds hidden patterns, groups or relationships within the data on its own. It's mainly used for clustering, dimensionality reduction and data visualization.

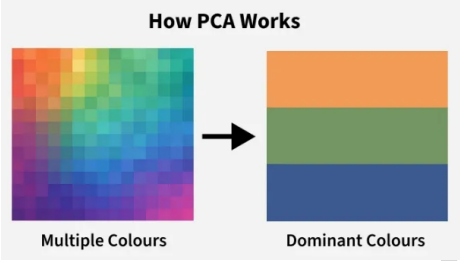
Example: If you have customer data without labels, the algorithm can group similar customers based on purchase behavior useful for segmentation and marketing.

There are two main categories of unsupervised learning that are mentioned below:

- Clustering : Clustering is the process of grouping data points into clusters based on their similarity. This technique is useful for identifying patterns and relationships in data without the need for labeled examples. Common techniques include:
    1. K-Means : K-Means Clustering groups similar data points into clusters without needing labeled data. It is used to uncover hidden patterns when the goal is to organize data based on similarity.
    2. DBSCAN : DBSCAN is a density-based clustering algorithm that groups data points that are closely packed together and marks outliers as noise based on their density in the feature space. It identifies clusters as dense regions in the data space separated by areas of lower density.
    3. Mean-shift : Meanshift is falling under the category of a clustering algorithm in contrast of Unsupervised learning that assigns the data points to the clusters iteratively by shifting points towards the mode (mode is the highest density of data points in the region, in the context of the Meanshift). As such, it is also known as the Mode-seeking algorithm
- Dimensionality Reduction Techniques: Dimensionality reduction helps reduce the number of features while preserving important information. Common techniques include:
    1. Principal Component Analysis:

Dept. of ECE, ACETVM

PCA (Principal Component Analysis) is a dimensionality reduction technique and helps us to reduce the number of features in a dataset while keeping the most important information. It changes complex datasets by transforming correlated features into a smaller set of uncorrelated components.



It helps us to remove redundancy, improve computational efficiency and make data easier to visualize and analyze.

2. Independent Component Analysis : Independent Component Analysis (ICA) is a technique used to separate mixed signals into their independent, non-Gaussian components. ICA is widely used in fields like audio, image processing and biomedical signal analysis to isolate distinct sources from mixed signals.

- 3. Association Rule Learning: Association rule learning is a technique for discovering relationships between items in a dataset. It identifies rules that indicate the presence of one item implies the presence of another item with a specific probability.

4 Applications of Unsupervised Learning

- Clustering and segmentation: Group similar data points, customers, or images.
- Anomaly detection: Spot unusual patterns or outliers in data.
- Dimensionality reduction: Simplify large datasets while retaining key information.
- Recommendation and marketing: Identify user preferences and improve product suggestions.
- Data preprocessing and analysis: Clean data, detect patterns and support exploratory data analysis (EDA).

Dept. of ECE, ACETVM

5 Examples of Applications of Machine Learning

- Analyzing images of products on a production line to classify them automatically : image classification.
- Detecting tumors in brain scans : Semantic segmentation.
- Automatically classifying news articles: Natural language processing (NLP).
- Automatically flagging offensive comments on discussion forums : text classification(NLP).
- Summarizing long documents automatically: Branch of NLP called text summarization.
- Creating a chatbot or a personal assistant: NLP components.
- Forecasting your company's revenue next year, based on many performance metrics: Regression task.
- Making your app react to voice commands: Speech recognition.
- Detecting credit card fraud: Anomaly detection.
- Segmenting clients based on their purchases so that you can design a different marketing strategy for each segment: clustering.
- Representing a complex, high-dimensional dataset in a clear and insightful diagram: Data visualization, often involving dimensionality reduction techniques.
- Recommending a product that a client may be interested in, based on past purchases: Recommender system.
- Building an intelligent bot for a game: Reinforcement Learning.

5.1 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a dimensionality reduction technique that transforms a set of possibly correlated features into a smaller set of new, uncorrelated features called principal components. It aims to reduce the dimensionality of the data while retaining as much of the important variation (information) as possible.

Intuitively, PCA finds new axes (directions) in the feature space such that:

- The first principal component captures the maximum variance in the data.
- The second principal component captures the maximum remaining variance and is orthogonal to the first.

Dept. of ECE, ACETVM

- Subsequent components continue this pattern, each being orthogonal to the previous ones.

By keeping only the first few principal components, we can represent the data in a lower-dimensional space with minimal loss of information.

Mathematical formulation

Suppose we have a dataset with n observations and p features, arranged in a data matrix

$$\mathbf{X} \in R^{n \times p},$$

where each row corresponds to an observation and each column to a feature. Before applying PCA, the data is usually standardized so that each feature has zero mean and unit variance.

Covariance matrix The (sample) covariance matrix of the standardized data is

$$\mathbf{C} = \frac{1}{n-1}\mathbf{X}^\top \mathbf{X} \in R^{p \times p}.$$

Eigen-decomposition PCA is based on the eigen-decomposition of the covariance matrix:

$$\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i, \quad i = 1, \ldots, p,$$

where λ_i are the eigenvalues and $\mathbf{v}_i$ are the corresponding eigenvectors. The eigenvalues are sorted in descending order,

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0,$$

so that $\mathbf{v}_1$ is the first principal component direction, $\mathbf{v}_2$ is the second, and so on.

Principal components and projection The k leading principal component directions are collected in the matrix

$$\mathbf{W}_k = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \ldots & \mathbf{v}_k \end{bmatrix} \in R^{p \times k}.$$

The projection of the original data onto the k-dimensional principal component space is

$$\mathbf{Z} = \mathbf{X}\mathbf{W}_k \in R^{n \times k},$$

where each row of $\mathbf{Z}$ is the representation of the corresponding observation in the reduced k-dimensional space.

Dept. of ECE, ACETVM

Explained variance The total variance in the data is proportional to the sum of all eigenvalues,

$$\text{Total variance} = \sum_{i=1}^{p} \lambda_i.$$

The proportion of variance explained by the k principal components is

$$\text{Explained variance ratio}(k) = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{p} \lambda_i}.$$

In practice, k is chosen so that this ratio exceeds a desired threshold (for example, 90%), ensuring that most of the information is retained while significantly reducing dimensionality.

6 Singular Value Decomposition (SVD)

Singular Value Decomposition (SVD) is a matrix factorization technique that writes any real matrix A as a product of three matrices:

$$A = U\Sigma V^T,$$

where:

- A is an $m \times n$ matrix (original data or transformation),
- U is an $m \times m$ orthogonal matrix (columns: left singular vectors),
- Σ is an $m \times n$ diagonal matrix with non-negative entries $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$ (singular values),
- V is an $n \times n$ orthogonal matrix (columns: right singular vectors).

Geometrically, SVD tells us that the linear transformation defined by A can be seen as:

1. a rotation or reflection given by V^T,
2. followed by a scaling along coordinate axes given by Σ,
3. followed by another rotation or reflection given by U.

Large singular values indicate directions where the data varies a lot, while small singular values correspond to directions that can often be discarded for dimensionality reduction or noise filtering.

7 Batch and Online Learning

Another criterion used to classify Machine Learning systems is whether or not the system can learn incrementally from a stream of incoming data.

Dept. of ECE, ACETVM

7.1 Batch learning

- In batch learning, the system is incapable of learning incrementally: it must be trained using all the available data. This will generally take a lot of time and computing resources, so it is typically done offline. First the system is trained, and then it is launched into production and runs without learning anymore; it just applies what it has learned. This is called offline learning.
- If you want a batch learning system to know about new data (such as a new type of spam), you need to train a new version of the system from scratch on the full dataset (not just the new data, but also the old data), then stop the old system and replace it with the new one.
- Simply update the data and train a new version of the system from scratch as often as needed.
- But training using the full set of data can take many hours, so you would typically train a new system only every 24 hours or even just weekly. If your system needs to adapt to rapidly changing data.
- Training on the full set of data requires a lot of computing resources (CPU, memory space, disk space, disk I/O, network I/O, etc.). If you have a lot of data and you automate your system to train from scratch every day, it will end up costing you a lot of money. If the amount of data is huge, it may even be impossible to use a batch learning algorithm.

7.2 Online learning

- In online learning, you train the system incrementally by feeding it data instances sequentially, either individually or in small groups called mini-batches. Each learning step is fast and cheap, so the system can learn about new data on the fly, as it arrives.
- Online learning is great for systems that receive data as a continuous flow (e.g., stock prices) and need to adapt to change rapidly or autonomously. It is also a good option if you have limited computing resources: once an online learning system has learned about new data instances, it does not need them anymore, so you can discard them.
- Online learning algorithms can also be used to train systems on huge datasets that cannot fit in one machine's main memory (this is called out-of-core learning). The algorithm loads part of the data, runs a training step on that data, and repeats the process until it has run on all of the data.

Dept. of ECE, ACETVM

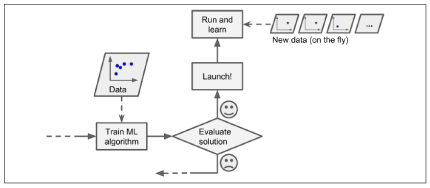


Figure 1-13. In online learning, a model is trained and launched into production, and then it keeps learning as new data comes in

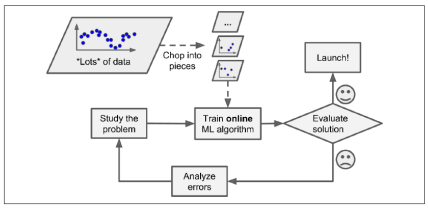


Figure 1-14. Using online learning to handle huge datasets

8 Instance-Based Versus Model-Based Learning

8.1 Instance-Based Learning

- This method, known as "lazy learning," memorizes the training data and makes predictions based on the most similar instances or data points. It doesn't create an abstract model during training but instead delays the work until a query or new data point is presented.
- In this approach, when a new data point comes in, the algorithm compares it to existing instances in the training set and makes predictions based on how closely the new point matches the old data.
- No model is explicitly created during the training process.
- Predictions are made using the entire training dataset by comparing new instances to existing data points.
- Works well with small datasets, as the algorithm needs to compare new data to all previously stored examples.
- The K-Nearest Neighbors algorithm is the most well-known example of instance-based learning. It classifies new data points by comparing them to the 'k' nearest points in the dataset, making predictions based on the majority class among those neighbours.
- For example, in movie recommendation systems, KNN can be used to find similar users based on their preferences. If a user enjoys action movies, the system might recommend films that people with similar tastes have liked.

8.2 Model-Based Learning

- This method, also known as "eager learning," is the process where the algorithm builds a model based on the training data. The model learns patterns from the training data to make predictions for unseen data points. Once the model is built, it can make predictions much faster compared to instance-based learning, since it doesn't rely on the entire dataset during prediction.

Dept. of ECE, ACETVM

- In this learning, the algorithm uses mathematical models (such as linear regression, decision trees, or neural networks) to learn relationships between input features and the target output.
- Builds a model during the training phase to make future predictions.
- The model extract patterns from the training data to predict unseen instances without having to rely on the training data directly.
- Effective for larger datasets, as the model simplifies and transforms the data into a set of learned rules or mathematical equations that can be used to make predictions efficiently.
- Linear regression is a classic example of model-based learning. It builds a model to predict the target variable based on a linear relationship between input features.
- Linear regression can be used to predict house prices based on features like the number of bedrooms, area (in sq feet), and location. Once the model is built, it can quickly provide predictions without needing to reference past instances.

9 Machine learning models

A Machine Learning Model is a computational program that learns patterns from data and makes decisions or predictions on new, unseen data. It is created by training a machine learning algorithm on a dataset and optimizing it to minimize errors. Key characteristics of ML models are:

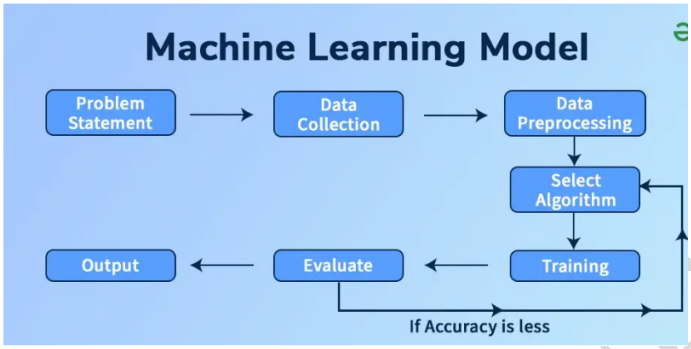
- Finds hidden patterns from historical information.
- Can forecast values or classify inputs.
- Learns from additional data and feedback.
- Reduces human effort and increases efficiency.

9.1 Components of Machine learning model

To build an effective Machine Learning model, it is important to understand its core components. These elements define how a model learns, predicts and improves over time.

- **Parameters**: Internal values learned automatically during training that define model knowledge and predictions like weights and biases in neural networks.

Dept. of ECE, ACETVM



- **Hyperparameters**: External configuration settings defined before training that control learning speed, complexity and model structure. It can include learning rate, number of epochs, batch size, etc. Unlike model parameters, which are learned during training, hyperparameters are set before training begins and stay fixed throughout. Choosing the right ones is key to achieving optimal results. A well-tuned model, optimized with carefully selected hyperparameters, can lead to:

  1. Better Generalization.
  2. Improved Training Efficiency.
  3. Improved Interpretability.

- **Loss Function**: Mathematical function that measures how far predictions are from actual outputs and guides model training. It can be MSE for regression and Cross-Entropy for classification.

- **Optimization**: Algorithms that adjust parameters iteratively to minimize loss and improve model accuracy and convergence, like Gradient Descent, Adam, RMSprop, etc.

- **Evaluation Metrics**: Quantitative measures to assess model performance on unseen data, enabling comparison and selection. Some examples are Accuracy, Precision, Recall, F1-Score, RMSE, R^2 Score, etc.

10 Training data vs Testing data

When we build any machine learning model, the data we use is divided into two important parts: training data and testing data. Training data teaches a model how to make predictions, and testing data checks how well the model has learned.

Dept of ECE, ACETVM

10.1 Training Data

Training data is the dataset used to teach a machine learning model. It usually contains labeled examples (where the correct output is already known). The model studies these examples, finds patterns, and slowly learns to make predictions on its own.

During training, the model:

1. looks at input and output pairs.
2. identifies relationships.
3. adjusts its internal rules.
4. improves its accuracy over time.

Models with large and good-quality training data usually perform better.

10.2 Testing data

Once the model has learned from training data, we need new, unseen data to check if it has learned correctly. This new dataset is called testing data. Testing data helps to:

1. measure accuracy.
2. check if the model is overfitting.
3. verify if the model can handle new information.

If a model performs well on testing data, it means it has truly understood the patterns instead of just memorizing.

11 Bias and Variance in Machine Learning

Bias and Variance are two fundamental concepts that help explain a model's prediction errors in machine learning. Bias refers to the error caused by oversimplifying a model while variance refers to the error from making the model too sensitive to training data.

11.1 Bias

Bias is the error that occurs when a model is too simple to capture the true patterns in the data.

- **High bias**: The model oversimplifies, misses patterns and underfits the data.
- **Low bias**: The model captures patterns well and is closer to the true values.

Dept. of ECE, ACETVM

11.1.1 How to Reduce Bias?

Some methods to lower bias in models are:

1. Use More Complex Models: Use models capable of capturing non-linear relationships such as neural networks or ensemble methods.
2. Add Relevant Features: Include additional informative features in the training data to give the model for capturing underlying patterns.
3. Adjust Regularization Strength: Reduce regularization to allow the model more flexibility in fitting the data.

11.2 Variance

Variance arises when a model becomes too sensitive to training data and it captures noises in data too. It fails to give prediction on unseen new data.

- **High variance**: The model is too sensitive to small changes and may overfit.
- **Low variance**: The model is more stable but might miss some patterns.

11.2.1 How to Reduce Variance?

Some methods to lower variance are:

1. Simplify the Model: Use a simpler model or prune overly deep decision trees to avoid overfitting.
2. Increase Training Data: Collect more data to stabilize learning and make the model generalize better.
3. Apply Regularization: Use L1 or L2 regularization to constrain model complexity and prevent overfitting.
4. Use Ensemble Methods: Implement techniques like bagging or random forests to combine multiple models and balance bias–variance trade-offs.

11.3 Bias Variance Tradeoff

The total prediction error depends on the tradeoff between bias and variance:

Model Type	**Bias**	**Variance**	**Result**
Underfitting	High	Low	Poor training and test performance
Optimal	Moderate	Moderate	Best generalization
Overfitting	Low	High	Poor test performance

Dept. of ECE, ACETVM

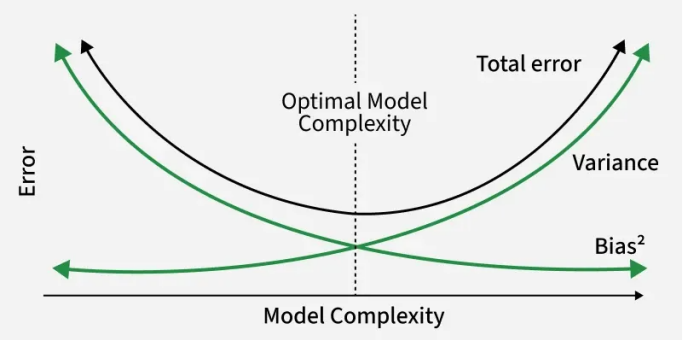


Figure 8: Bias Variance Trade off

12 Underfitting and Overfitting

Machine learning models should learn useful patterns from training data. When a model learns too little or too much, we get underfitting or overfitting.Underfitting means that the model is too simple and does not cover all real patterns in the data. Overfitting means that the model learns not just the underlying pattern, but also noise or random quirks in the training data. model memorizes training data A good model finds the right spot, it is complex enough to capture real patterns, but not so complex that it "memorizes" noise.

12.1 Underfitting

Underfitting happens when the model fails to learn important patterns. It performs poorly on both training and testing data. Underfitting happens due to:

- Model is too simple
- Very high regularization
- Features are weak or missing
- Not enough training
- High bias

12.1.1 Bias–Variance Inside Underfitting

Underfitting mainly occurs due to high bias:

- High bias means model makes strong assumptions
- Ignores patterns
- Learns an overly simple representation

Dept. of ECE, ACETVM

- Variance is low because the model gives similar outputs even if the data changes
- Underfitting = High Bias + Low Variance

12.2 What is Overfitting?

Overfitting happens when the model learns too much from the training data, including noise and outliers. It performs very well on training data but poorly on test data. Overfitting happens due to:

- Model too complex
- Too many features
- Very little data
- No regularization
- High variance

12.2.1 Bias–Variance Inside Overfitting

Overfitting is mainly caused by high variance:

- High variance means model reacts too strongly to training data
- Learns noise as patterns
- Low bias because the model is extremely flexible
- Overfitting = Low Bias + High Variance

12.3 How to Address Overfitting and Underfitting?

Techniques to Reduce Underfitting

- Use a more complex model
- Add new features and perform feature engineering
- Reduce regularization
- Train for more epochs
- Scale features properly

Techniques to Reduce Overfitting

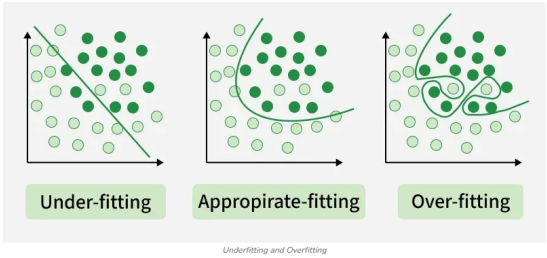
- Collect more training data
- Reduce model complexity

Dept. of ECE, ACETVM



Underfitting and Overfitting

- **Underfitting :** Straight line trying to fit a curved dataset but cannot capture the data's patterns, leading to poor performance on both training and test sets.
- **Overfitting**: A squiggly curve passing through all training points, failing to generalize performing well on training data but poorly on test data.
- **Appropriate Fitting**: Curve that follows the data trend without overcomplicating to capture the true patterns in the data.



Underfitting and Overfitting

- **Underfitting (High Bias):** A model that is too simple (like a straight line for curved data) misses key patterns and performs poorly on both training and testing data.
- **Overfitting (High Variance):** A model that is too complex (like a high-degree polynomial) learns noise, fits training data too closely, and performs poorly on new data.
- **Ideal Fit (Balanced):** A moderately complex model captures the main trend without following noise, giving good performance on both training and testing data.

- Use regularization (L1/L2)
- Apply dropout (for neural networks)
- Use early stopping
- Clean noisy data

13 Regularization in Machine Learning

Regularization is a technique used in machine learning to prevent overfitting, which otherwise causes models to perform poorly on unseen data. By adding a penalty for complexity, regularization encourages simpler and more generalizable models.

13.1 Benefits of Regularization

- Prevents Overfitting: Regularization helps models focus on underlying patterns instead of memorizing noise in the training data.
- Enhances Performance: Prevents excessive weighting of outliers or irrelevant features helps in improving overall model accuracy.

Dept of ECE ACETVM

- Stabilizes Models: Reduces sensitivity to minor data changes which ensures consistency across different data subsets.
- Prevents Complexity: Keeps model from becoming too complex which is important for limited or noisy data.
- Handles Multicollinearity: Reduces the magnitudes of correlated coefficients helps in improving model stability.
- Promotes Consistency: Ensures reliable performance across different datasets which reduces the risk of large performance shifts.

13.1.1 Types of regularization

1. L1 regularization: This technique adds the absolute value of the model parameters to the loss function. This is also known as lasso regularization.
2. L2 regularization: This technique adds the squared value of the model parameters to the loss function. This is also known as ridge regularization.
3. Elastic net regularization: This technique combines both L1 and L2 regularization by adding a weighted sum of their penalty terms to the loss function.
4. Dropout regularization: This technique randomly drops out some of the units or connections in a neural network during the training process.

14 Challenges in Machine learning

1. **Insufficient Quantity of Training Data**: It takes a lot of data for most Machine Learning algorithms to work properly. Even for very simple problems you typically need thousands of examples, and for complex problems such as image or speech recognition you may need millions of examples.
2. **Nonrepresentative Training Data**: In order to generalize well, it is crucial that your training data be representative of the new cases you want to generalize to. This is true whether you use instance-based learning or model-based learning.It is crucial to use a training set that is representative of the cases you want to generalize to. This is often harder than it sounds: if the sample is too small, you will have sampling noise (i.e., nonrepresentative data as a result of chance), but even very large samples can be nonrepresentative if the sampling method is flawed. This is called sampling bias.
3. **Poor-Quality Data**: If training data is full of errors, outliers, and noise (e.g., due to poor quality measurements), it will make it harder for the system to detect the underlying patterns, so your system is less likely to

Dept. of ECE, ACETVM

perform well. It is often well worth the effort to spend time cleaning up your training data. The truth is, most data scientists spend a significant part of their time doing just that.

4. **Irrelevant Features**: Your system will only be capable of learning if the training data contains enough relevant features and not too many irrelevant ones. A critical part of the success of a Machine Learning project is coming up with a good set of features to train on. This process, called feature engineering, involves the following steps:
   - Feature selection (selecting the most useful features to train on among existing features)
   - Feature extraction (combining existing features to produce a more useful one—as we saw earlier, dimensionality reduction algorithms can help)
   - Creating new features by gathering new data
5. **Overfitting the Training Data**: Overgeneralizing is something that we humans do all too often, and unfortunately machines can fall into the same trap if we are not careful. In Machine Learning this is called overfitting: it means that the model performs well on the training data, but it does not generalize well.
6. **Underfitting the Training Data**: It occurs when your model is too simple to learn the underlying structure of the data.

15 Performance matrices

When building machine learning models, it's important to understand how well they perform. Evaluation metrics help us to measure the effectiveness of our models. Whether we are solving a classification problem, predicting continuous values or clustering data, selecting the right evaluation metric allows us to assess how well the model meets our goals.

15.1 Classification Metrics

Classification problems aim to predict discrete categories. To evaluate the performance of classification models, several standard metrics are used: *accuracy*, *precision*, *recall*, and the *F1-score*.

15.1.1 Accuracy

Accuracy is a fundamental metric used for evaluating the performance of a classification model. It gives the proportion of correct predictions made by the model out of all predictions.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

Dept. of ECE, ACETVM

Although accuracy provides a quick snapshot of model performance, it can be misleading for imbalanced datasets. For instance, in a dataset with 90% samples from class A and 10% from class B, a classifier that always predicts class A achieves 90% accuracy but completely fails to identify class B instances. Thus, high accuracy may give a false sense of good performance when the minority class is frequently misclassified.

15.1.2 Precision

Precision measures how many of the positive predictions made by the model are actually correct. It is particularly useful when the cost of false positives is high, such as in medical diagnosis where predicting a disease that is not present can lead to unnecessary follow-up tests or anxiety.

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

where TP denotes true positives and FP denotes false positives. High precision indicates that when the model predicts a positive outcome, it is very likely to be correct.

15.1.3 Recall (Sensitivity)

Recall, also called sensitivity or true positive rate, measures how many of the actual positive cases were correctly identified by the model. It is important when missing a positive case (false negative) is more costly than raising some false alarms.

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

where FN denotes false negatives. In applications such as disease detection or fraud detection, recall is a key metric because failing to detect positive cases can have serious consequences.

15.1.4 F1-score

The F1-score is the harmonic mean of precision and recall and provides a single value that balances both metrics. It is especially useful when a balance between precision and recall is desired, and when the class distribution is imbalanced.

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

The F1-score lies in the range $[0, 1]$, with higher values indicating better overall performance in terms of both precision and recall. A model with high accuracy but very low recall on the minority class will typically have a low F1-score, highlighting that many positive instances are being missed.

Dept. of ECE, ACETVM

15.1.5 Area Under Curve (AUC) and ROC Curve

Area Under the Curve (AUC) and the Receiver Operating Characteristic (ROC) curve are widely used to evaluate binary classification models. The AUC value represents the probability that the model ranks a randomly chosen positive example higher than a randomly chosen negative example, and it ranges from 0 to 1, with higher values indicating better performance. AUC–ROC is particularly useful when comparing models across different classification thresholds.

True Positive Rate (TPR) The True Positive Rate (TPR), also known as sensitivity or recall, measures how many actual positive instances are correctly identified by the model. It answers the question: out of all actual positive cases, how many did the model correctly identify?

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

where TP is the number of true positives and FN is the number of false negatives.

True Negative Rate (TNR) The True Negative Rate (TNR), also called specificity, measures how many actual negative instances are correctly identified by the model. It answers the question: out of all actual negative cases, how many did the model correctly identify as negative?

$$\text{TNR} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

where TN is the number of true negatives and FP is the number of false positives.

False Positive Rate (FPR) The False Positive Rate (FPR) measures how many actual negative instances are incorrectly classified as positive. It is especially important when the cost of false positives is high, such as in fraud detection.

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

where FP denotes false positives and TN denotes true negatives.

False Negative Rate (FNR) The False Negative Rate (FNR) measures how many actual positive instances are incorrectly classified as negative. It answers the question: out of all the actual positive cases, how many were misclassified as negative?

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

where FN denotes false negatives and TP denotes true positives.

Dept. of ECE, ACETVM

ROC Curve and AUC The ROC curve is a graphical representation of the True Positive Rate versus the False Positive Rate at different classification thresholds. It visualizes the trade–off between sensitivity (TPR) and specificity $(1 - \text{FPR})$ as the threshold varies. The Area Under the ROC Curve (AUC) quantifies the overall ability of the model to distinguish between positive and negative classes.

Typical interpretations of AUC values are:

- AUC = 1: perfect model, always correctly classifies positives and negatives.
- AUC = 0.5: model performs no better than random guessing.
- AUC $<$ 0.5: model performs worse than random guessing (effectively inverted).

16 Exercises

1. How would you define Machine Learning?
2. Can you name four types of problems where it shines?
3. What is a labeled training set?
4. What are the two most common supervised tasks?
5. Can you name four common unsupervised tasks?
6. What type of Machine Learning algorithm would you use to allow a robot to walk in various unknown terrains?
7. What type of algorithm would you use to segment your customers into multiple groups?
8. Would you frame the problem of spam detection as a supervised learning problem or an unsupervised learning problem?
9. What is an online learning system?
10. What is out-of-core learning?
11. What type of learning algorithm relies on a similarity measure to make predictions?
12. What is the difference between a model parameter and a learning algorithm's hyperparameter?
13. What do model-based learning algorithms search for? What is the most common strategy they use to succeed? How do they make predictions?
14. Can you name four of the main challenges in Machine Learning?

Dept. of ECE, ACETVM

15. If your model performs great on the training data but generalizes poorly to new instances, what is happening? Can you name three possible solutions?
16. What is a test set, and why would you want to use it?
17. What is the purpose of a validation set?
18. What is the train-dev set, when do you need it, and how do you use it?
19. What can go wrong if you tune hyperparameters using the test set?

16.1 Solutions

1. Machine Learning is about building systems that can learn from data. Learning means getting better at some task, given some performance measure.
2. Machine Learning is great for complex problems for which we have no algorithmic solution, to replace long lists of hand-tuned rules, to build systems that adapt to fluctuating environments, and finally to help humans learn (e.g., data mining).
3. A labeled training set is a training set that contains the desired solution (a.k.a. a label) for each instance.
4. The two most common supervised tasks are regression and classification.
5. Common unsupervised tasks include clustering, visualization, dimensionality reduction, and association rule learning.
6. Reinforcement Learning is likely to perform best if we want a robot to learn to walk in various unknown terrains, since this is typically the type of problem that Reinforcement Learning tackles. It might be possible to express the problem as a supervised or semisupervised learning problem, but it would be less natural.
7. If you don't know how to define the groups, then you can use a clustering algorithm (unsupervised learning) to segment your customers into clusters of similar customers. However, if you know what groups you would like to have, then you can feed many examples of each group to a classification algorithm (supervised learning), and it will classify all your customers into these groups.
8. Spam detection is a typical supervised learning problem: the algorithm is fed many emails along with their labels (spam or not spam).
9. An online learning system can learn incrementally, as opposed to a batch learning system. This makes it capable of adapting rapidly to both changing data and autonomous systems, and of training on very large quantities of data.

Dept. of ECE, ACETVM

10. Out-of-core algorithms can handle vast quantities of data that cannot fit in a computer's main memory. An out-of-core learning algorithm chops the data into mini-batches and uses online learning techniques to learn from these minibatches.

11. An instance-based learning system learns the training data by heart; then, when given a new instance, it uses a similarity measure to find the most similar learned instances and uses them to make predictions.

12. A model has one or more model parameters that determine what it will predict given a new instance (e.g., the slope of a linear model). A learning algorithm tries to find optimal values for these parameters such that the model generalizes well to new instances. A hyperparameter is a parameter of the learning algorithm itself, not of the model (e.g., the amount of regularization to apply).

13. Model-based learning algorithms search for an optimal value for the model parameters such that the model will generalize well to new instances. We usually train such systems by minimizing a cost function that measures how bad the system is at making predictions on the training data, plus a penalty for model complexity if the model is regularized. To make predictions, we feed the new instance's features into the model's prediction function, using the parameter values found by the learning algorithm.

14. Some of the main challenges in Machine Learning are the lack of data, poor data quality, nonrepresentative data, uninformative features, excessively simple models that underfit the training data, and excessively complex models that overfit the data.

15. If a model performs great on the training data but generalizes poorly to new instances, the model is likely overfitting the training data (or we got extremely lucky on the training data). Possible solutions to overfitting are getting more data, simplifying the model (selecting a simpler algorithm, reducing the number of parameters or features used, or regularizing the model), or reducing the noise in the training data.

16. A test set is used to estimate the generalization error that a model will make on new instances, before the model is launched in production.

17. A validation set is used to compare models. It makes it possible to select the best model and tune the hyperparameters.

18. The train-dev set is used when there is a risk of mismatch between the training data and the data used in the validation and test datasets (which should always be as close as possible to the data used once the model is in production). The traindev set is a part of the training set that's held out (the model is not trained on it). The model is trained on the rest of the training set, and evaluated on both the train-dev set and the validation set. If the model performs well on the training set but not on

Dept. of ECE, ACETVM

the train-dev set, then the model is likely overfitting the training set. If it performs well on both the training set and the train-dev set, but not on the validation set, then there is probably a significant data mismatch between the training data and the validation + test data, and you should try to improve the training data to make it look more like the validation + test data.

19. If you tune hyperparameters using the test set, you risk overfitting the test set, and the generalization error you measure will be optimistic (you may launch a model that performs worse than you expect).

Dept. of ECE, ACETVM

PEECT413 Machine Learning

S4 ECE, 2024 Scheme KTU

Module 1

Module 2 Syllabus

Regression: linear regression, logistic regression error functions in regression, MSE, L1, L2, cross-entropy, and multivariate regression. Classification: Naive Bayes classifier, Support Vector machines, Decision trees -random forests, Ensemble methods: boosting, bagging.

1 Classification and Regression

There are two major types of supervised machine learning problems, called classification and regression.

In **classification**, the goal is to predict a class label, which is a choice from a predefined list of possibilities. Classification is sometimes separated into binary classification, which is the special case of distinguishing between exactly two classes, and multiclass classification, which is classification between more than two classes. You can think of binary classification as trying to answer a yes/no question. Classifying emails as either spam or not spam is an example of a binary classification problem. In this binary classification task, the yes/no question being asked would be "Is this email spam?"

For **regression** tasks, the goal is to predict a continuous number, or a floating-point number in programming terms (or real number in mathematical terms). Predicting a person's annual income from their education, their age, and where they live is an example of a regression task. When predicting income, the predicted value is an amount, and can be any number in a given range. Another example of a regression task is predicting the yield of a corn farm given attributes such as previous yields, weather, and number of employees working on the farm. The yield again can be an arbitrary number.

An easy way to distinguish between classification and regression tasks is to ask whether there is some kind of continuity in the output. If there is continuity between possible outcomes, then the problem is a regression problem.

2 Linear Models for regression

Linear models make a prediction using a linear function of the input features

Dept. of ECE, ACETVM

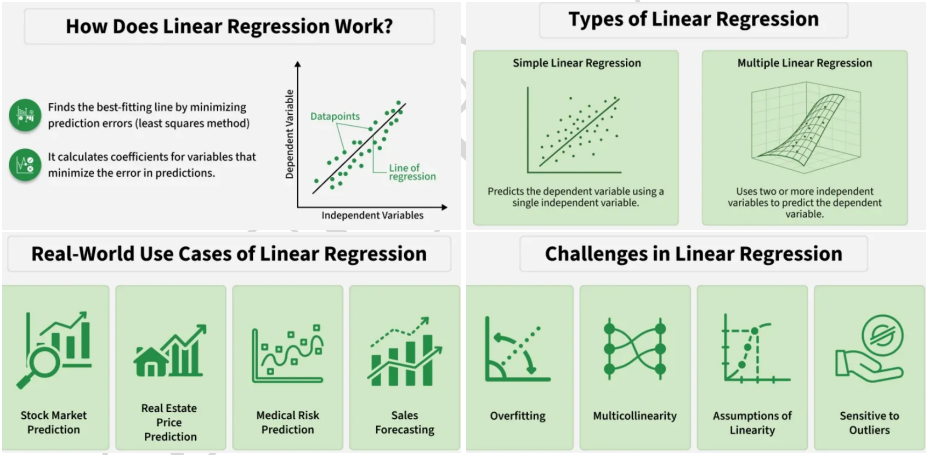
3 Linear Regression in Machine learning

Linear regression is a type of supervised machine-learning algorithm that learns from the labelled datasets and maps the data points with the most optimized linear functions, which can be used for prediction on new datasets. It assumes that there is a linear relationship between the input and output, meaning the output changes at a constant rate as the input changes. This relationship is represented by a straight line.

For example we want to predict a student's exam score based on how many hours they studied. We observe that as students study more hours, their scores go up. In the example of predicting exam scores based on hours studied. Here

- Independent variable (input): Hours studied because it's the factor we control or observe.
- Dependent variable (output): Exam score because it depends on how many hours were studied.

We use the independent variable to predict the dependent variable.



3.1 Best Fit Line in Linear Regression

In linear regression, the best-fit line is the straight line that most accurately represents the relationship between the independent variable (input) and the dependent variable (output). It is the line that minimizes the difference between the actual data points and the predicted values from the model.

3.1.1 Goal of the Best-Fit Line

The goal of linear regression is to find a straight line that minimizes the error (the difference) between the observed data points and the predicted values. This line helps us predict the dependent variable for new, unseen data.

Dept. of ECE, ACETVM

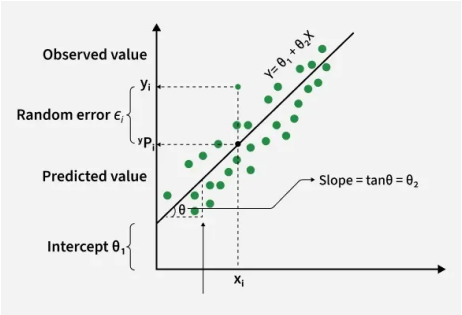


Figure 1: Linear regression

Here, Y is called a dependent or target variable and X is called an independent variable, also known as the predictor of Y.

- θ_1 represents the intercept, which is the value of Y when $X = 0$.
- θ_2 represents the slope, which shows how much Y changes for a unit change in X.

There are many types of functions or models that can be used for regression. A linear function is the simplest type of function. Here, X may be a single feature or multiple features representing the problem.

3.2 Equation of the Best-Fit Line

For simple linear regression (with one independent variable), the best-fit line is represented by the equation

$$y = mx + b$$

Where:

- y is the predicted value (dependent variable).
- x is the input (independent variable).
- m is the slope of the line (how much y changes when x changes).
- b is the intercept (the value of y when $x = 0$).

The best-fit line will be the one that optimizes the values of m (slope) and b (intercept) so that the predicted y values are as close as possible to the actual data points.

Dept. of ECE, ACETVM

3.3 Minimizing the Error: The Least Squares Method

To find the best-fit line, we use a method called *Least Squares*. The idea behind this method is to minimize the sum of squared differences between the actual values (data points) and the predicted values from the line. These differences are called residuals.

The formula for residuals is:

$$\text{Residual} = y_i - \hat{y}_i$$

Where:

- y_i is the actual observed value.
- $\hat{y}_i$ is the predicted value from the line for that x_i.

The least squares method minimizes the sum of the squared residuals:

$$\sum (y_i - \hat{y}_i)^2$$

This method ensures that the line best represents the data where the sum of the squared differences between the predicted values and actual values is as small as possible.

3.4 Interpretation of the Best-Fit Line

Slope (m): The slope of the best-fit line indicates how much the dependent variable (y) changes with each unit change in the independent variable (x). For example, if the slope is 5, it means that for every 1-unit increase in x, the value of y increases by 5 units.

Intercept (b): The intercept represents the predicted value of y when $x = 0$. It is the point where the line crosses the y-axis.

In linear regression some hypotheses are made to ensure reliability of the model's results.

Limitations:

- *Assumes Linearity:* The method assumes the relationship between the variables is linear. If the relationship is non-linear, linear regression might not work well.
- *Sensitivity to Outliers:* Outliers can significantly affect the slope and intercept, skewing the best-fit line.

3.5 Hypothesis function in Linear Regression

In linear regression, the hypothesis function is the equation used to make predictions about the dependent variable based on the independent variables. It represents the relationship between the input features and the target output.

Dept. of ECE, ACETVM

For a simple case with one independent variable, the hypothesis function is:

$$h(x) = \beta_0 + \beta_1 x$$

Where:

- $h(x)$ or $\hat{y}$ is the predicted value of the dependent variable (y).
- x is the independent variable.
- β_0 is the intercept, representing the value of y when x is 0.
- β_1 is the slope, indicating how much y changes for each unit change in x.

For multiple linear regression (with more than one independent variable), the hypothesis function expands to:

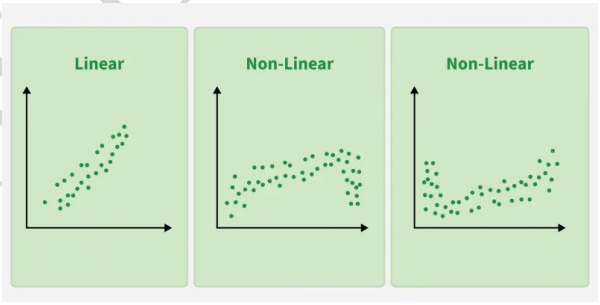
$$h(x_1, x_2, \ldots, x_k) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k$$

Where:

- $x_1, x_2, \ldots, x_k$ are the independent variables.
- β_0 is the intercept.
- $\beta_1, \beta_2, \ldots, \beta_k$ are the coefficients, representing the influence of each respective independent variable on the predicted output.

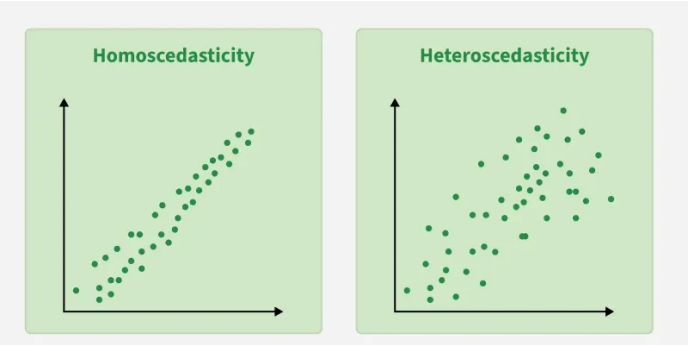
3.6 Assumptions of the Linear Regression

- **Linearity:** The relationship between inputs (X) and the output (Y) is a straight line.



- **Independence of Errors:** The errors in predictions should not affect each other.
- **Constant Variance (Homoscedasticity):**The errors should have equal spread across all values of the input. If the spread changes (like fans out or shrinks), it's called heteroscedasticity and it's a problem for the model.

Dept. of ECE, ACETVM



- **Normality of Errors:** The errors should follow a normal (bell-shaped) distribution.
- **No Multicollinearity (for multiple regression):** Input variables shouldn't be too closely related to each other.
- **No Autocorrelation:** Errors shouldn't show repeating patterns, especially in time-based data.
- **Additivity:** The total effect on Y is just the sum of effects from each X, no mixing or interaction between them.'

3.7 Types of Linear Regression

When there is only one independent feature it is known as Simple Linear Regression or Univariate Linear Regression and when there are more than one feature it is known as Multiple Linear Regression or Multivariate Regression.

3.7.1 Simple Linear Regression

Simple linear regression is used when we want to predict a target value (dependent variable) using only one input feature (independent variable). It assumes a straight-line relationship between the two.

Formula

$$\hat{y} = \theta_0 + \theta_1 x$$

Where:

- $\hat{y}$ is the predicted value.
- x is the input (independent variable).
- θ_0 is the intercept (value of $\hat{y}$ when $x = 0$).
- θ_1 is the slope or coefficient (how much $\hat{y}$ changes with one unit of x).

Example:
Predicting a person's salary (y) based on their years of experience (x).

Dept of ECE, ACETVM

3.8 Multiple Linear Regression

Multiple linear regression involves more than one independent variable and one dependent variable. The equation for multiple linear regression is:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

where:

- $\hat{y}$ is the predicted value.
- $x_1, x_2, \ldots, x_n$ are the independent variables.
- $\theta_1, \theta_2, \ldots, \theta_n$ are the coefficients (weights) corresponding to each predictor.
- θ_0 is the intercept.

The goal is to find the best-fit line that predicts Y accurately for given inputs X.

Use Cases

- **Real Estate:** Predict property prices using location, size and other factors.
- **Finance:** Forecast stock prices using interest rates and inflation data.
- **Agriculture:** Estimate crop yield from rainfall, temperature and soil quality.
- **E-commerce:** Analyze how price, promotions and seasons affect sales.

3.9 Cost function for Linear Regression

In Linear Regression, the cost function measures how far the predicted values ($\hat{Y}$) are from the actual values (Y). It helps identify and reduce errors to find the best-fit line. The most common cost function used is Mean Squared Error (MSE), which calculates the average of squared differences between actual and predicted values:

$$\text{Cost function } J = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$

Here,

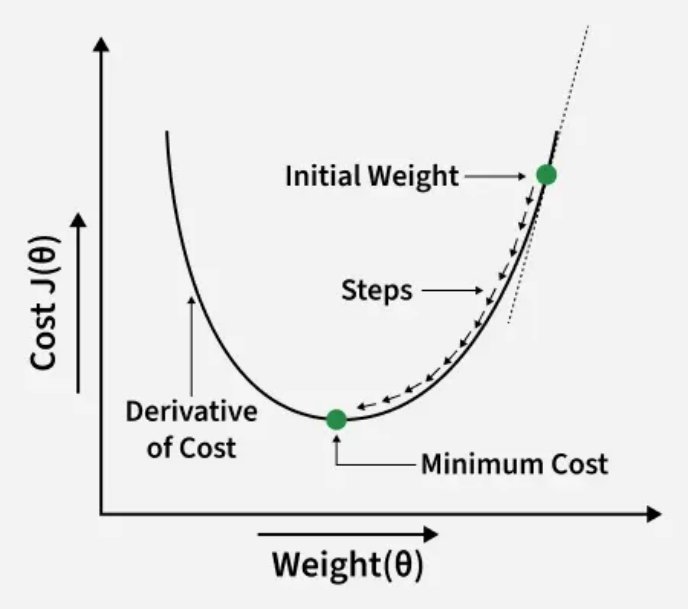
$$\hat{y}_i = \theta_1 + \theta_2 x_i$$

To minimize this cost, we use Gradient Descent, which iteratively updates θ_1 and θ_2 until the MSE reaches its lowest value. This ensures the line fits the data as accurately as possible.

Dept. of ECE, ACETVM

3.10 Gradient Descent for Linear Regression

Gradient descent is an optimization technique used to train a linear regression model by minimizing the prediction error. It works by starting with random model parameters and repeatedly adjusting them to reduce the difference between predicted and actual values.



How it works:

- Start with random values for slope and intercept.
- Calculate the error between predicted and actual values.
- Find how much each parameter contributes to the error (gradient).
- Update the parameters in the direction that reduces the error.
- Repeat until the error is as small as possible.

This helps the model find the best-fit line for the data.

4 Evaluation Metrics for Linear Regression

A variety of evaluation measures can be used to determine the strength of any linear regression model. These assessment metrics often give an indication of how well the model is producing the observed outputs.

Dept. of ECE, ACETVM

Mean Square Error (MSE)

Mean Squared Error (MSE) is an evaluation metric that calculates the average of the squared differences between the actual and predicted values for all the data points. The difference is squared to ensure that negative and positive differences do not cancel each other out.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Here,

- n is the number of data points.
- y_i is the actual or observed value for the i^{th} data point.
- $\hat{y}_i$ is the predicted value for the i^{th} data point.

MSE is a way to quantify the accuracy of a model's predictions. MSE is sensitive to outliers as large errors contribute significantly to the overall score.

5 Regularization Techniques for Linear Models

5.1 Lasso Regression (L1 Regularization)

Lasso Regression is a technique used for regularizing a linear regression model; it adds a penalty term to the linear regression objective function to prevent overfitting.

The objective function after applying lasso regression is:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2 + \lambda\sum_{j=1}^{n}|\theta_j|$$

- The first term is the least squares loss, representing the squared difference between predicted and actual values.
- The second term is the L1 regularization term; it penalizes the sum of absolute values of the regression coefficient θ_j.

5.2 Ridge Regression (L2 Regularization)

Ridge regression is a linear regression technique that adds a regularization term to the standard linear objective. The goal is to prevent overfitting by penalizing large coefficients in the linear regression equation. It is useful when the dataset has multicollinearity where predictor variables are highly correlated.

The objective function after applying ridge regression is:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2 + \lambda\sum_{j=1}^{n}\theta_j^2$$

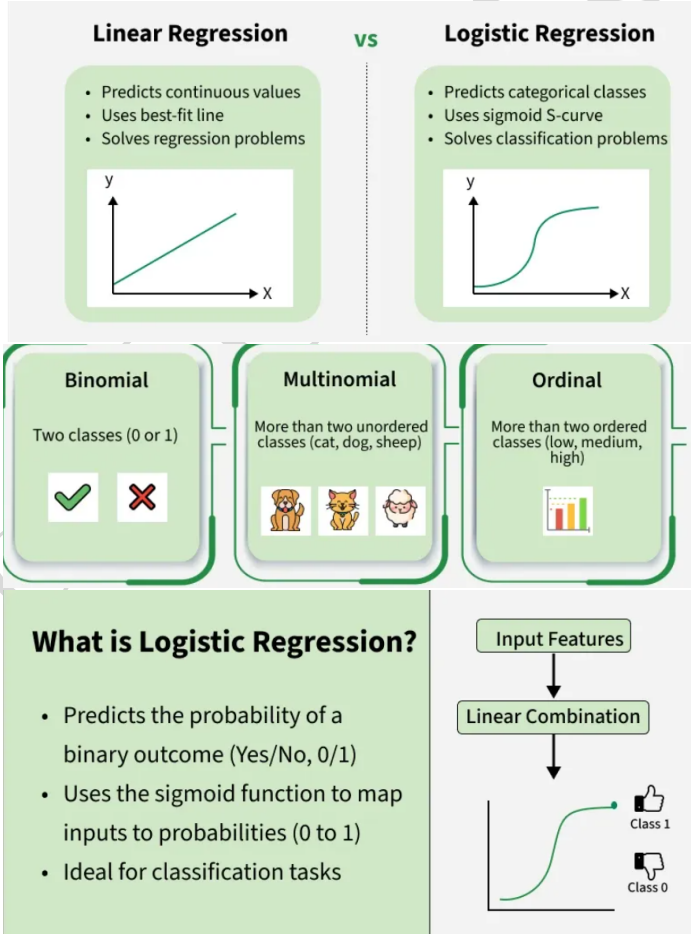
Dept. of ECE, ACETVM

- The first term is the least squares loss, representing the squared difference between predicted and actual values.
- The second term is the L2 regularization term; it penalizes the sum of square values of the regression coefficient θ_j.

6 Logistic Regression

Logistic Regression is a supervised machine learning algorithm used for classification problems. Unlike linear regression which predicts continuous values it predicts the probability that an input belongs to a specific class.

It is used for binary classification where the output can be one of two possible categories such as Yes/No, True/False or 0/1. It uses sigmoid function to convert inputs into a probability value between 0 and 1.



Dept. of ECE, ACETVM

Types of Logistic Regression

Logistic regression can be classified into three main types based on the nature of the dependent variable:

1. **Binomial Logistic Regression:** This type is used when the dependent variable has only two possible categories. Examples include Yes/No, Pass/Fail or 0/1. It is the most common form of logistic regression and is used for binary classification problems.
2. **Multinomial Logistic Regression:** This is used when the dependent variable has three or more possible categories that are not ordered. For example, classifying animals into categories like "cat", "dog" or "sheep". It extends the binary logistic regression to handle multiple classes.
3. **Ordinal Logistic Regression:** This type applies when the dependent variable has three or more categories with a natural order or ranking. Examples include ratings like "low", "medium" and "high". It takes the order of the categories into account when modeling.

Assumptions of Logistic Regression

Understanding the assumptions behind logistic regression is important to ensure the model is applied correctly. The main assumptions are:

1. **Independent observations:** Each data point is assumed to be independent of the others, meaning there should be no correlation or dependence between the input samples.
2. **Binary dependent variables:** It assumes that the dependent variable must be binary, meaning it can take only two values. For more than two categories, SoftMax functions are used.
3. **Linearity relationship between independent variables and log odds:** The model assumes a linear relationship between the independent variables and the log odds of the dependent variable, which means the predictors affect the log odds in a linear way.
4. **No outliers:** The dataset should not contain extreme outliers as they can distort the estimation of the logistic regression coefficients.
5. **Large sample size:** It requires a sufficiently large sample size to produce reliable and stable results.

Dept. of ECE, ACETVM

Understanding Sigmoid Function

1. The sigmoid function is an important part of logistic regression which is used to convert the raw output of the model into a probability value between 0 and 1.
2. This function takes any real number and maps it into the range 0 to 1 forming an "S" shaped curve called the sigmoid curve or logistic curve. Because probabilities must lie between 0 and 1, the sigmoid function is perfect for this purpose.
3. In logistic regression, we use a threshold value usually 0.5 to decide the class label.
   - If the sigmoid output is same or above the threshold, the input is classified as Class 1.
   - If it is below the threshold, the input is classified as Class 0.

This approach helps to transform continuous input values into meaningful class predictions.

6.1 How does Logistic Regression work?

Logistic regression model transforms the *linear regression* function continuous value output into categorical value output using a sigmoid function which maps any real-valued set of independent variables input into a value between 0 and 1. This function is known as the logistic function.

Suppose we have input features represented as a matrix:

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1m} \\ x_{21} & \cdots & x_{2m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nm} \end{bmatrix}$$

and the dependent variable is $\mathbf{Y}$ having only binary value i.e. 0 or 1:

$$Y = \begin{cases} 0 & \text{if Class 1} \\ 1 & \text{if Class 2} \end{cases}$$

then, apply the multi-linear function to the input variables X:

$$z = \left(\sum_{i=1}^{n} w_i x_i \right) + b$$

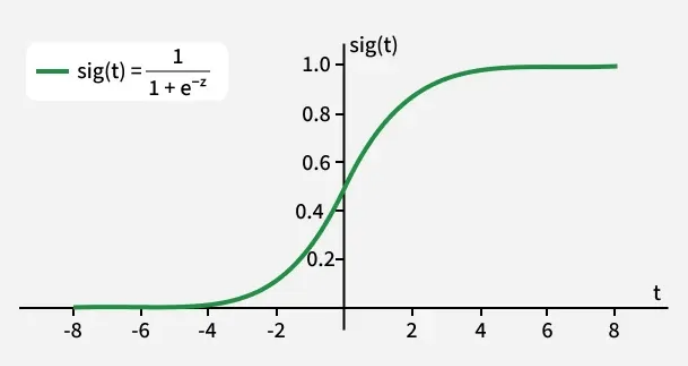
Here x_i is the *ith* observation of X, $\mathbf{w}_i = [w_1, w_2, w_3, \cdots, w_m]$ is the weights or coefficient and b is the bias term also known as intercept. Simply this can be represented as the dot product of weight and bias:

$$z = \mathbf{w} \cdot \mathbf{X} + b$$

Dept. of ECE, ACETVM

At this stage, z is a continuous value from the linear regression. Logistic regression then applies the sigmoid function to z to convert it into a probability between 0 and 1 which can be used to predict the class.

Now we use the *sigmoid function* where the input will be z and we find the probability between 0 and 1, i.e. predicted y.



As shown above the sigmoid function converts the continuous variable data into the probability i.e. between 0 and 1.

- $\sigma(z)$ tends towards 1 as $z \to \infty$
- $\sigma(z)$ tends towards 0 as $z \to -\infty$
- $\sigma(z)$ is always bounded between 0 and 1

where the probability of being a class can be measured as:

$$P(y = 1) = \sigma(z)$$

$$P(y = 0) = 1 - \sigma(z)$$

7 Cross-Entropy Loss Function

In classification problems, a machine learning model predicts the probability of each class for any given input. Because each data point truly belongs to only one class (probability 1 for one class, 0 for others). Cross-entropy loss is a way to measure how close a model's predictions are to the correct answers in classification problems.

It helps train models to make more confident and accurate predictions by rewarding correct answers and penalizing wrong ones. This makes it a key part of building reliable machine learning classifiers.

Dept. of ECE, ACETVM

Table 1: Comparison of linear and logistic regression

Aspect	Linear Regression	Logistic Regression
Definition	Linear regression is used to predict the continuous dependent variable using a given set of independent variables.	Logistic regression is used to predict the categorical dependent variable using a given set of independent variables.
Problem Type	It is used for solving regression problem.	It is used for solving classification problems.
Output Type	In this we predict the value of continuous variables.	In this we predict values of categorical variables.
Curve/Model Fitting	In this we find best fit line.	In this we find S-Curve.
Estimation Method	Least square estimation method is used for estimation of accuracy.	Maximum likelihood estimation method is used for estimation of accuracy.
Output Example	The output must be continuous value such as price, age etc.	Output must be categorical value such as 0 or 1, Yes or No, etc.
Relationship Requirement	It required linear relationship between dependent and independent variables.	It not required linear relationship.
Collinearity	There may be collinearity between the independent variables.	There should be little to no collinearity between independent variables.

7.1 Types of Cross-Entropy Loss Function

7.1.1 Binary Cross Entropy Loss

Binary Cross-Entropy Loss is a widely used loss function in binary classification problems. For a dataset with N instances, the Binary Cross-Entropy Loss is calculated as:

$$\text{BCE} = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log(p_i) + (1-y_i)\log(1-p_i)\right)$$

where

- N is number of samples,
- y_i is the true label for sample i (0 or 1),
- p_i is the model-predicted probability for class 1 for sample i.

7.1.2 Multiclass Cross Entropy Loss

Multiclass Cross-Entropy Loss, also known as *categorical cross-entropy* or *softmax loss*, is a widely used loss function for training models in multiclass classification problems. For a dataset with N instances, Multiclass Cross-Entropy

Dept. of ECE, ACETVM

Loss is calculated as

$$\text{CE} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C}(y_{i,j}\log(p_{i,j}))$$

where

- N is number of samples,
- C is the number of classes,
- $y_{i,j}$ is 1 if class j is correct for sample i, 0 otherwise,
- $p_{i,j}$ is the model-predicted probability of sample i being in class j.

7.2 How to Interpret Cross Entropy Loss?

The cross-entropy loss is a scalar value that quantifies how far off the model's predictions are from the true labels. For each sample in the dataset, the cross-entropy loss reflects how well the model's prediction matches the true label. A lower loss for a sample indicates a more accurate prediction, while a higher loss suggests a larger discrepancy.

Interpretability for Binary Classification:

- In binary classification, since there are two classes (0 and 1) it is straightforward to interpret the loss value.
- If the true label is 1, the loss is primarily influenced by how close the predicted probability for class 1 is to 1.0.
- If the true label is 0, the loss is influenced by how close the predicted probability for class 1 is to 0.0.

Interpretability for Multiclass Classification:

- In multiclass classification, only the true label contributes towards the loss, as other labels being zero do not add anything to the loss function.
- Lower loss indicates that the model is assigning high probabilities to the correct class and low probabilities to incorrect classes.

Key features of Cross Entropy loss

- **Probabilistic Interpretation:** Guides models to output probabilities near the true class labels.
- **Differentiable:** Supports optimization via gradient descent.
- **Standard for Neural Networks:** Especially with softmax (multiclass) or sigmoid (binary) output layers.

Dept. of ECE, ACETVM

- **Strong Penalization:** Assigns high penalty to confident but wrong predictions.
- **Library Support:** Implemented in all major ML libraries like PyTorch, TensorFlow, scikit-learn, etc.

8 Naive Bayes Classifiers

Naive Bayes is a machine learning classification algorithm that predicts the category of a data point using probability. It assumes that all features are independent of each other. Naive Bayes performs well in many real-world applications such as spam filtering, document categorisation and sentiment analysis.

8.1 Why is it Called Naive Bayes?

It is named as "Naive" because it assumes the presence of one feature does not affect other features. The "Bayes" part of the name refers to its basis in Bayes' Theorem.

Consider a fictional dataset that describes the weather conditions for playing a game of golf. Given the weather conditions, each tuple classifies the conditions as fit ("Yes") or unfit ("No") for playing golf. Here is a tabular representation of our dataset.

	Outlook	Temperature	Humidity	Windy	Play Golf
0	Rainy	Hot	High	False	Yes
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	No
4	Sunny	Cool	Normal	False	Yes
5	Sunny	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Rainy	Mild	High	False	No
8	Rainy	Cool	Normal	False	Yes
9	Sunny	Mild	Normal	False	Yes
10	Rainy	Mild	Normal	True	Yes
11	Overcast	Mild	High	True	Yes
12	Overcast	Hot	Normal	False	Yes
13	Sunny	Mild	High	True	No

The dataset is divided into two parts i.e. feature matrix and the response vector.

- **Feature matrix** contains all the vectors (rows) of dataset in which each vector consists of the value of dependent features. In the above dataset, features are 'Outlook', 'Temperature', 'Humidity' and 'Windy'.

Dept. of ECE, ACETVM

- **Response vector** contains the value of class variable (prediction or output) for each row of feature matrix. In the above dataset, the class variable name is 'Play golf'.

8.2 Assumption of Naive Bayes

The fundamental Naive Bayes assumption is that each feature makes an:

- **Feature independence:** This means that when we are trying to classify something, we assume that each feature (or piece of information) in the data does not affect any other feature.
- **Continuous features are normally distributed:** If a feature is continuous, then it is assumed to be normally distributed within each class.
- **Discrete features have multinomial distributions:** If a feature is discrete, then it is assumed to have a multinomial distribution within each class.
- **Features are equally important:** All features are assumed to contribute equally to the prediction of the class label.
- **No missing data:** The data should not contain any missing values.

8.3 Introduction to Bayes' Theorem

Bayes' Theorem provides a principled way to reverse conditional probabilities. It is defined as:

$$P(y \mid X) = \frac{P(X \mid y) \cdot P(y)}{P(X)}$$

Where:

- $P(y \mid X)$: Posterior probability, probability of class y given features X.
- $P(X \mid y)$: Likelihood, probability of features X given class y.
- $P(y)$: Prior probability of class y.
- $P(X)$: Marginal likelihood or evidence.

8.4 Key Features of Naive Bayes Classifiers

The main idea behind the Naive Bayes classifier is to use Bayes' Theorem to classify data based on the probabilities of different classes given the features of the data. It is used mostly in high-dimensional text classification.

- The Naive Bayes Classifier is a simple probabilistic classifier and it has very few number of parameters which are used to build the ML models that can predict at a faster speed than other classification algorithms.

Dept. of ECE, ACETVM

- It is a probabilistic classifier because it assumes that one feature in the model is independent of existence of another feature. In other words, each feature contributes to the predictions with no relation between each other.
- Naive Bayes Algorithm is used in spam filtration, sentiment analysis, classifying articles and many more.

9 Types of Naive Bayes Model

There are three types of Naive Bayes Model:

9.1 Gaussian Naive Bayes

In *Gaussian Naive Bayes*, continuous values associated with each feature are assumed to be distributed according to a Gaussian distribution. A Gaussian distribution is also called *Normal distribution*. When plotted, it gives a bell-shaped curve which is symmetric about the mean of the feature values.

9.2 Multinomial Naive Bayes

Multinomial Naive Bayes is used when features represent the frequency of terms (such as word counts) in a document. It is commonly applied in text classification, where term frequencies are important.

9.3 Bernoulli Naive Bayes

Bernoulli Naive Bayes deals with binary features, where each feature indicates whether a word appears or not in a document. It is suited for scenarios where the presence or absence of terms is more relevant than their frequency. Both models are widely used in document classification tasks.

9.4 Advantages

- Easy to implement and computationally efficient.
- Effective in cases with a large number of features.
- Performs well even with limited training data.
- It performs well in the presence of categorical features.
- For numerical features data is assumed to come from normal distributions.

Dept. of ECE, ACETVM

9.5 Disadvantages

- Assumes that features are independent, which may not always hold in real-world data.
- Can be influenced by irrelevant attributes.
- May assign zero probability to unseen events, leading to poor generalization.

9.6 Applications

- **Spam Email Filtering:** Classifies emails as spam or non-spam based on features.
- **Text Classification:** Used in sentiment analysis, document categorization and topic classification.
- **Medical Diagnosis:** Helps in predicting the likelihood of a disease based on symptoms.
- **Credit Scoring:** Evaluates creditworthiness of individuals for loan approval.
- **Weather Prediction:** Classifies weather conditions based on various factors.

9.7 Naive Bayes Working

9.7.1 1. Terminology

Consider a classification problem (like predicting if someone plays golf based on weather). Then:

- y is the class label (e.g. "Yes" or "No" for playing golf).
- $X = (x_1, x_2, \ldots, x_n)$ is the feature vector (e.g. Outlook, Temperature, Humidity, Wind).

A sample row from the dataset:

$$X = (\text{Rainy, Hot, High, False}), \quad y = \text{No}$$

This represents:
What is the probability that someone will not play golf given that the weather is Rainy, Hot, High humidity and No wind?

Dept. of ECE, ACETVM

9.7.2 2. The Naive Assumption

The "naive" in Naive Bayes comes from the assumption that all features are independent given the class. That is:

$$P(x_1, x_2, \ldots, x_n \mid y) = P(x_1 \mid y) \cdot P(x_2 \mid y) \cdots P(x_n \mid y)$$

Thus, Bayes' theorem becomes:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \cdot \prod_{i=1}^{n} P(x_i \mid y)}{P(X)}$$

Since the denominator is constant for a given input, we can write:

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \cdot \prod_{i=1}^{n} P(x_i \mid y)$$

9.7.3 3. Constructing the Naive Bayes Classifier

We compute the posterior for each class y and choose the class with the highest probability:

$$\hat{y} = \arg\max_{y} \; P(y) \cdot \prod_{i=1}^{n} P(x_i \mid y)$$

This becomes our Naive Bayes classifier.

9.8 Example: Weather Dataset

Let us take a dataset used for predicting if golf is played based on:

- Outlook: Sunny, Rainy, Overcast
- Temperature: Hot, Mild, Cool
- Humidity: High, Normal
- Wind: True, False

Example input:

$$X = (\text{Sunny, Hot, Normal, False})$$

Goal: Predict if golf will be played (Yes or No).

9.8.1 Pre-computation from Dataset

Class Probabilities: From dataset of 14 rows:

$$P(\text{Yes}) = \frac{9}{14}, \qquad P(\text{No}) = \frac{5}{14}$$

Conditional Probabilities:

Dept. of ECE, ACETVM

Feature	Value	P(Value | Yes)	P(Value | No)
Outlook	Sunny	2/9	3/5
Temperature	Hot	2/9	2/5
Humidity	Normal	6/9	1/5
Wind	False	6/9	2/5

9.8.2 Calculate Posterior Probabilities

For class = Yes:

$$P(\text{Yes} \mid \text{today}) \propto \frac{2}{9} \cdot \frac{2}{9} \cdot \frac{6}{9} \cdot \frac{6}{9} \cdot \frac{9}{14}$$

$$P(\text{Yes} \mid \text{today}) \approx 0.02116$$

For class = No:

$$P(\text{No} \mid \text{today}) \propto \frac{3}{5} \cdot \frac{2}{5} \cdot \frac{1}{5} \cdot \frac{2}{5} \cdot \frac{5}{14}$$

$$P(\text{No} \mid \text{today}) \approx 0.0068$$

9.8.3 Normalize Probabilities

To compare:

$$P(\text{Yes} \mid \text{today}) = \frac{0.02116}{0.02116 + 0.0068} \approx 0.756$$

$$P(\text{No} \mid \text{today}) = \frac{0.0068}{0.02116 + 0.0068} \approx 0.244$$

9.8.4 Final Prediction

Since:

$$P(\text{Yes} \mid \text{today}) > P(\text{No} \mid \text{today})$$

The model predicts: Yes (Play Golf).

9.9 Naive Bayes for Continuous Features

For continuous features, we assume a Gaussian distribution:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Where:

- μ_y is the mean of feature x_i for class y,
- σ_y^2 is the variance of feature x_i for class y.

This leads to what is called *Gaussian Naive Bayes*.

Dept. of ECE, ACETVM

10 Types of Naive Bayes Model

There are three types of Naive Bayes Model:

10.1 Gaussian Naive Bayes

In *Gaussian Naive Bayes*, continuous values associated with each feature are assumed to be distributed according to a Gaussian distribution. A Gaussian distribution is also called a *Normal distribution*. When plotted, it gives a bell-shaped curve which is symmetric about the mean of the feature values.

10.2 Multinomial Naive Bayes

Multinomial Naive Bayes is used when features represent the frequency of terms (such as word counts) in a document. It is commonly applied in text classification, where term frequencies are important.

10.3 Bernoulli Naive Bayes

Bernoulli Naive Bayes deals with binary features, where each feature indicates whether a word appears or not in a document. It is suited for scenarios where the presence or absence of terms is more relevant than their frequency. Both models are widely used in document classification tasks.

10.4 Advantages

- Easy to implement and computationally efficient.
- Effective in cases with a large number of features.
- Performs well even with limited training data.
- Performs well in the presence of categorical features.
- For numerical features data is assumed to come from normal distributions.

10.5 Disadvantages

- Assumes that features are independent, which may not always hold in real-world data.
- Can be influenced by irrelevant attributes.
- May assign zero probability to unseen events, leading to poor generalization.

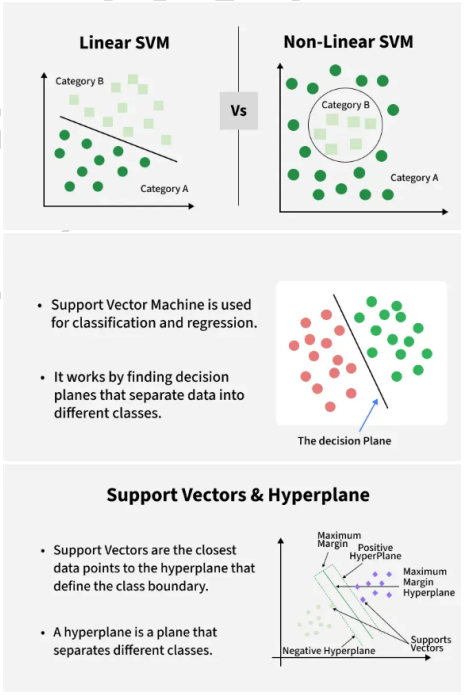
Dept. of ECE, ACETVM

10.6 Applications

- **Spam Email Filtering:** Classifies emails as spam or non-spam based on features.
- **Text Classification:** Used in sentiment analysis, document categorization and topic classification.
- **Medical Diagnosis:** Helps in predicting the likelihood of a disease based on symptoms.
- **Credit Scoring:** Evaluates creditworthiness of individuals for loan approval.
- **Weather Prediction:** Classifies weather conditions based on various factors.

11 Support Vector Machine (SVM) Algorithm

Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression tasks. It tries to find the best boundary known as hyperplane that separates different classes in the data. It is useful when you want to do binary classification like spam vs. not spam or cat vs. dog.



Dept. of ECE, ACETVM

The main goal of SVM is to maximize the margin between the two classes. The larger the margin the better the model performs on new and unseen data.

12 Support Vector Machine

12.1 Key Concepts of Support Vector Machine

- **Hyperplane:** A decision boundary separating different classes in feature space and is represented by the equation $w \cdot x + b = 0$ in linear classification.
- **Support Vectors:** The closest data points to the hyperplane, crucial for determining the hyperplane and margin in SVM.
- **Margin:** The distance between the hyperplane and the support vectors. SVM aims to maximize this margin for better classification performance.
- **Kernel:** A function that maps data to a higher-dimensional space enabling SVM to handle non-linearly separable data.
- **Hard Margin:** A maximum-margin hyperplane that perfectly separates the data without misclassifications.
- **Soft Margin:** Allows some misclassifications by introducing slack variables, balancing margin maximization and misclassification penalties when data is not perfectly separable.
- **C:** A regularization term balancing margin maximization and misclassification penalties. A higher C value forces stricter penalty for misclassifications.
- **Hinge Loss:** A loss function penalizing misclassified points or margin violations and is combined with regularization in SVM.
- **Dual Problem:** Involves solving for Lagrange multipliers associated with support vectors, facilitating the kernel trick and efficient computation.

12.2 How does Support Vector Machine Algorithm Work?

The key idea behind the SVM algorithm is to find the hyperplane that best separates two classes by maximizing the margin between them. This margin is the distance from the hyperplane to the nearest data points (support vectors) on each side.

The best hyperplane also known as the "hard margin" is the one that maximizes the distance between the hyperplane and the nearest data points from both classes. This ensures a clear separation between the classes. So from the above figure, we choose L2 as hard margin.

Let's consider a scenario like shown in Figure.3. Here, we have one blue ball in the boundary of the red ball.

Dept. of ECE, ACETVM

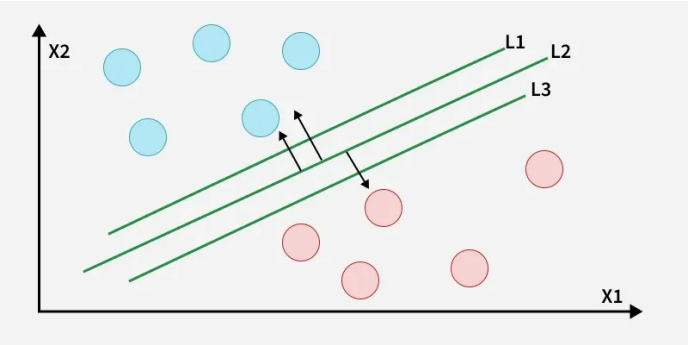


Figure 2: Multiple hyperplanes separate the data from two classes

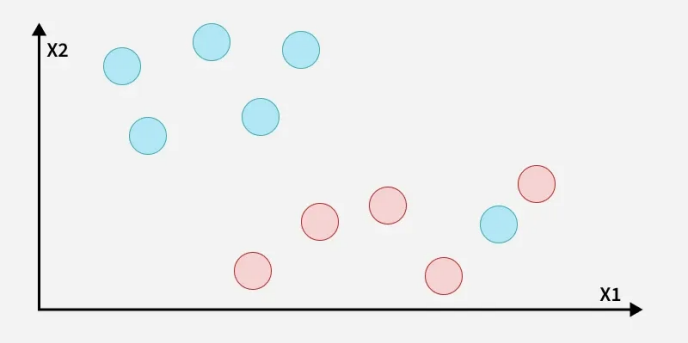


Figure 3: Selecting hyperplane for data with outlier

12.3 How does SVM classify the data?

The blue ball in the boundary of red ones is an outlier of blue balls. The SVM algorithm has the characteristics to ignore the outlier and finds the best hyperplane that maximizes the margin. SVM is robust to outliers.

A soft margin allows for some misclassifications or violations of the margin to improve generalization. The SVM optimizes the following equation to balance margin maximization and penalty minimization:

$$\text{Objective Function} = \left(\frac{1}{\text{margin}}\right) + \lambda \sum \text{penalty}$$

The penalty used for violations is often *hinge loss* which has the following behavior:

- If a data point is correctly classified and within the margin there is no penalty (loss = 0).
- If a point is incorrectly classified or violates the margin the hinge loss increases proportionally to the distance of the violation.

Dept. of ECE ACETVM

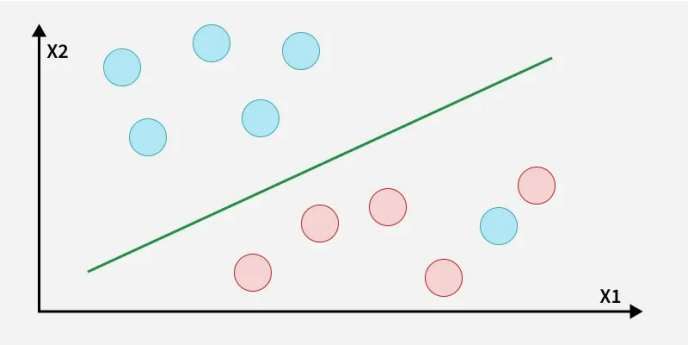


Figure 4: Hyperplane which is the most optimized one

Till now we were talking about linearly separable data that separates groups of blue balls and red balls by a straight line/linear line.

12.4 What if data is not linearly separable?

When data is not linearly separable i.e. it cannot be divided by a straight line, SVM uses a technique called *kernels* to map the data into a higher-dimensional space where it becomes separable. This transformation helps SVM find a decision boundary even for non-linear data. A kernel is a function that maps data

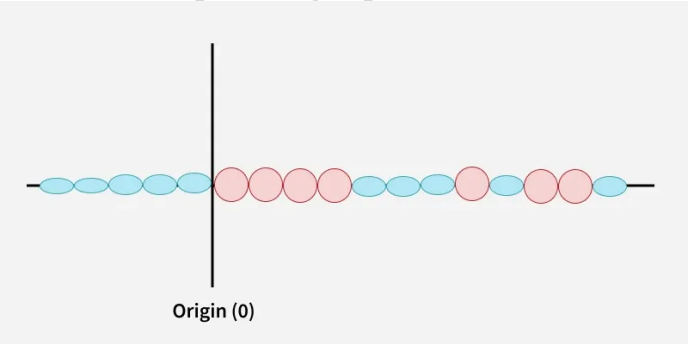


Figure 5: Original 1D dataset for classification

points into a higher-dimensional space without explicitly computing the coordinates in that space. This allows SVM to work efficiently with non-linear data by implicitly performing the mapping. For example consider data points that are not linearly separable. By applying a kernel function SVM transforms the data points into a higher-dimensional space where they become linearly separable.

- **Linear Kernel:** For linear separability.
- **Polynomial Kernel:** Maps data into a polynomial space.

Dept. of ECE, ACETVM

- **Radial Basis Function (RBF) Kernel:** Transforms data into a space based on distances between data points.

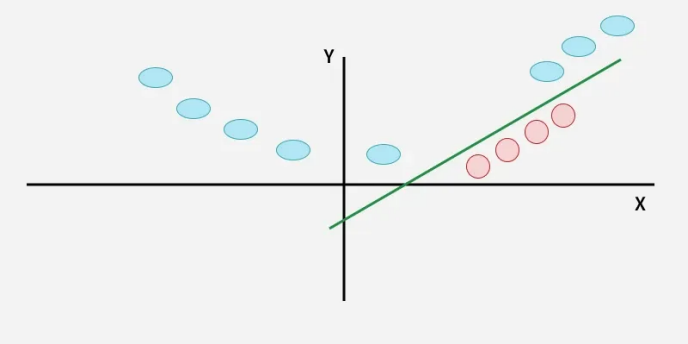


Figure 6: Mapping 1D data to 2D to become able to separate the two classes

In this case the new variable y is created as a function of distance from the origin.

13 Mathematical Computation of SVM

Consider a binary classification problem with two classes, labeled as +1 and -1. We have a training dataset consisting of input feature vectors X and their corresponding class labels Y. The equation for the linear hyperplane can be written as:

$$w^T x + b = 0$$

Where:

- w is the normal vector to the hyperplane (the direction perpendicular to it).
- b is the offset or bias term representing the distance of the hyperplane from the origin along the normal vector w.

13.1 Distance from a Data Point to the Hyperplane

The distance between a data point x_i and the decision boundary can be calculated as:

$$d_i = \frac{w^T x_i + b}{\|w\|}$$

where $\|w\|$ represents the Euclidean norm of the weight vector w.

Dept. of ECE, ACETVM

13.2 Linear SVM Classifier

Distance from a data point to the hyperplane:

$$\hat{y} = \begin{cases} 1, & w^T x + b \geq 0 \\ -1, & w^T x + b < 0 \end{cases}$$

Where $\hat{y}$ is the predicted label of a data point.

13.3 Optimization Problem for SVM

For a linearly separable dataset the goal is to find the hyperplane that maximizes the margin between the two classes while ensuring that all data points are correctly classified. This leads to the following optimization problem:

$$\min_{w,b} \frac{1}{2}\|w\|^2$$

Subject to the constraint:

$$y_i(w^T x_i + b) \geq 1 \quad \text{for } i = 1, 2, 3, \ldots, m$$

Where:

- y_i is the class label (+1 or -1) for each training instance.
- x_i is the feature vector for the i-th training instance.
- m is the total number of training instances.

The condition $y_i(w^T x_i + b) \geq 1$ ensures that each data point is correctly classified and lies outside the margin.

13.4 Soft Margin in Linear SVM Classifier

In the presence of outliers or non-separable data the SVM allows some misclassification by introducing slack variables ζ_i. The optimization problem is modified as:

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \zeta_i$$

Subject to the constraints:

$$y_i(w^T x_i + b) \geq 1 - \zeta_i \quad \text{and} \quad \zeta_i \geq 0 \quad \text{for } i = 1, 2, \ldots, m$$

Where:

- C is a regularization parameter that controls the trade-off between margin maximization and penalty for misclassifications.
- ζ_i are slack variables that represent the degree of violation of the margin by each data point.

Dept of ECE, ACETVM

13.5 Dual Problem for SVM

The dual problem involves maximizing the Lagrange multipliers associated with the support vectors. This transformation allows solving the SVM optimization using kernel functions for non-linear classification.

The dual objective function is given by:

$$\max_{\alpha} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j t_i t_j K(x_i, x_j) - \sum_{i=1}^{m} \alpha_i$$

Where:

- α_i are the Lagrange multipliers associated with the i-th training sample.
- t_i is the class label for the i-th training sample.
- $K(x_i, x_j)$ is the kernel function that computes the similarity between data points x_i and x_j. The kernel allows SVM to handle non-linear classification problems by mapping data into a higher-dimensional space.

The dual formulation optimizes the Lagrange multipliers α_i and the support vectors are those training samples where $\alpha_i > 0$.

13.6 Advantages of Support Vector Machine (SVM)

1. **High-Dimensional Performance:** SVM excels in high-dimensional spaces, making it suitable for image classification and gene expression analysis.
2. **Nonlinear Capability:** Utilizing kernel functions like RBF and polynomial SVM effectively handles nonlinear relationships.
3. **Outlier Resilience:** The soft margin feature allows SVM to ignore outliers, enhancing robustness in spam detection and anomaly detection.
4. **Binary and Multiclass Support:** SVM is effective for both binary classification and multiclass classification, suitable for applications in text classification.
5. **Memory Efficiency:** It focuses on support vectors making it memory efficient compared to other algorithms.

13.7 Disadvantages of Support Vector Machine (SVM)

1. **Slow Training:** SVM can be slow for large datasets, affecting performance in SVM in data mining tasks.
2. **Parameter Tuning Difficulty:** Selecting the right kernel and adjusting parameters like C requires careful tuning, impacting SVM algorithms.

Dept. of ECE, ACETVM

3. **Noise Sensitivity:** SVM struggles with noisy datasets and overlapping classes, limiting effectiveness in real-world scenarios.

4. **Limited Interpretability:** The complexity of the hyperplane in higher dimensions makes SVM less interpretable than other models.

5. **Feature Scaling Sensitivity:** Proper feature scaling is essential, otherwise SVM models may perform poorly.

13.8 SVM Decision Boundary

Once the dual problem is solved, the decision boundary is given by:

$$w = \sum_{i=1}^{m} \alpha_i t_i K(x_i, x) + b$$

Where w is the weight vector, x is the test data point and b is the bias term. Finally the bias term b is determined by the support vectors, which satisfy:

$$t_i(w^T x_i - b) = 1 \quad \Rightarrow \quad b = w^T x_i - t_i$$

Where x_i is any support vector.

This completes the mathematical framework of the Support Vector Machine algorithm which allows for both linear and non-linear classification using the dual problem and kernel trick.

13.9 Types of Support Vector Machine

Based on the nature of the decision boundary, Support Vector Machines (SVM) can be divided into two main parts:

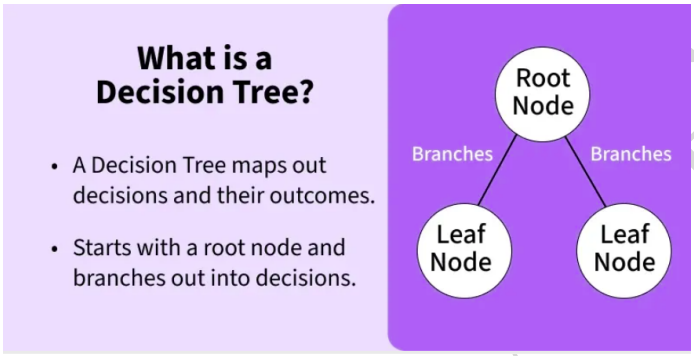
- **Linear SVM:** Linear SVMs use a linear decision boundary to separate the data points of different classes. When the data can be precisely linearly separated, linear SVMs are very suitable. This means that a single straight line (in 2D) or a hyperplane (in higher dimensions) can entirely divide the data points into their respective classes. A hyperplane that maximizes the margin between the classes is the decision boundary.

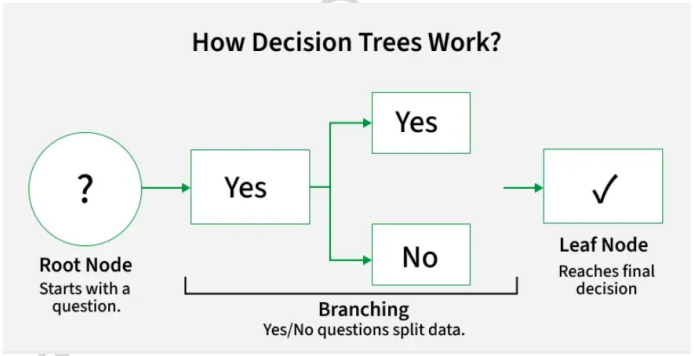
- **Non-Linear SVM:** Non-linear SVM can be used to classify data when it cannot be separated into two classes by a straight line (in the case of 2D). By using kernel functions, nonlinear SVMs can handle nonlinearly separable data. The original input data is transformed by these kernel functions into a higher-dimensional feature space where the data points can be linearly separated. A linear SVM is used to locate a nonlinear decision boundary in this modified space.

Dept. of ECE, ACETVM

14 Decision trees

A Decision Tree helps us to make decisions by mapping out different choices and their possible outcomes. It's used in machine learning for tasks like classification and prediction. In this article, we'll see more about Decision Trees, their types and other core concepts.



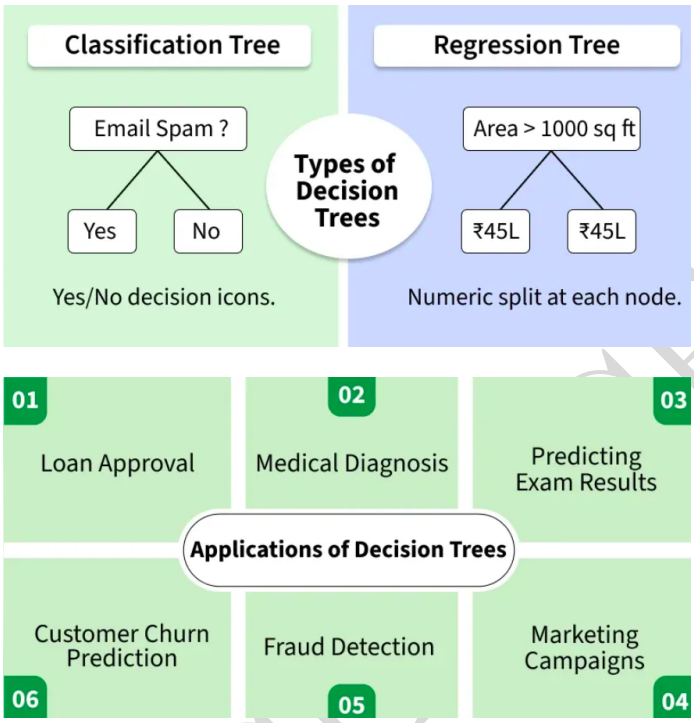


A decision tree helps with decision-making by showing possible outcomes clearly. It has a tree-like structure that starts with one main question called the root node, and then branches into different possibilities based on features in the data.

There are mainly two types of decision trees based on the target variable:

- **Classification Trees**: Used for predicting categorical outcomes, such as spam or not spam. These trees split the data based on features to classify data into predefined categories.
- **Regression Trees**: Used for predicting continuous outcomes, such as house prices. Instead of assigning categories, they provide numerical predictions based on the input features.

Dept. of ECE, ACETVM



14.1 Tree Components

A decision tree consists of the following components:

- **Root Node**: Starting point representing the whole dataset.
- **Branches**: Lines connecting nodes, showing the flow from one decision to another.
- **Internal Nodes**: Points where decisions are made based on data features.
- **Leaf Nodes**: End points of the tree where the final decision or prediction is made.

14.2 How Decision Trees Work

1. **Start with the Root Node**: Begin with a main question at the root node, derived from the dataset features.
2. **Ask Yes/No Questions**: From the root, the tree asks a series of yes/no questions to split the data into subsets based on specific attributes.
3. **Branching Based on Answers**: Each question leads to different branches.
    - If the answer is yes, the tree follows one path.

Dept of ECE ACETVM

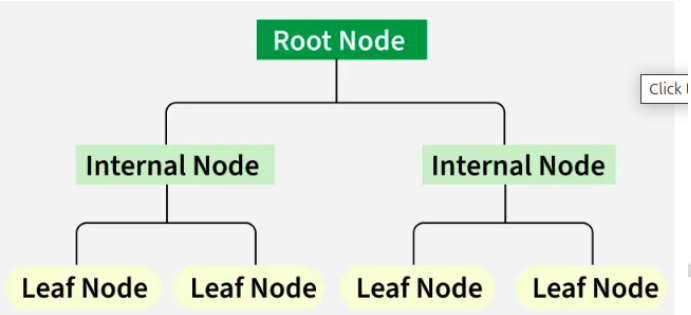


Figure 7: Decision Tree

- If the answer is no, the tree follows another path.

4. **Continue Splitting**: This branching continues through further decisions, helping to reduce the data step by step.
5. **Reach the Leaf Node**: The process ends when there are no more useful questions to ask, leading to a leaf node where the final decision or prediction is made.

14.3 Simple Coffee Example

Consider a simple example for deciding whether to drink coffee based on the time of day and how tired a person feels.

1. In the morning, the tree asks: "Tired?"
   - If yes, it suggests drinking coffee.
   - If no, it suggests no coffee is needed.
2. In the afternoon, it again asks: "Tired?"
   - If yes, it suggests drinking coffee.
   - If no, it suggests no coffee is needed.

15 Splitting Criteria in Decision Trees

In a decision tree, the process of splitting data at each node is important. The splitting criteria find the best feature to split the data on. Common splitting criteria include Gini Impurity and Entropy.

- **Gini Impurity**: This criterion measures how "impure" a node is. The lower the Gini impurity, the better the feature splits the data into distinct categories.

Dept. of ECE, ACETVM

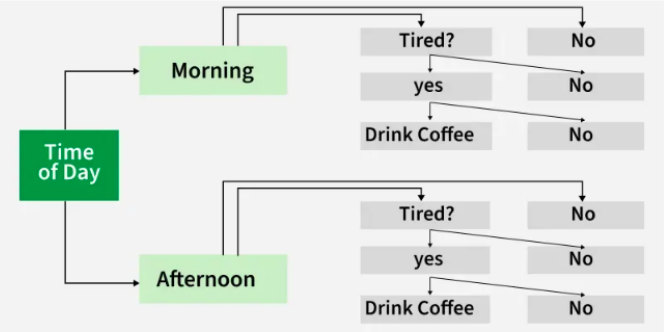


Figure 8: Example

- **Entropy**: This measures the amount of uncertainty or disorder in the data. The tree tries to reduce the entropy by splitting the data on features that provide the most information about the target variable.

These criteria help decide which features are useful for making the best split at each decision point in the tree.

16 Pruning in Decision Trees

Pruning is an important technique used to prevent overfitting in decision trees. Overfitting occurs when a tree becomes too deep and starts to memorize the training data rather than learning general patterns, which leads to poor performance on new, unseen data.

- This technique reduces the complexity of the tree by removing branches that have little predictive power.
- It improves model performance by helping the tree generalize better to new data and makes the model simpler and faster to deploy.
- It is useful when a decision tree is too deep and starts to capture noise in the data.

17 Advantages of Decision Trees

- **Easy to Understand**: Decision trees are visual, which makes it easy to follow the decision-making process.
- **Versatility**: They can be used for both classification and regression problems.
- **No Need for Feature Scaling**: Unlike many machine learning models, they do not require scaling or normalizing the data.

Dept. of ECE, ACETVM

- **Handles Non-linear Relationships**: They capture complex, non-linear relationships between features and outcomes effectively.
- **Interpretability**: The tree structure is easy to interpret, helping users understand the reasoning behind each decision.
- **Handles Missing Data**: They can handle missing values using strategies such as assigning the most common value or ignoring missing data during splits.

18 Disadvantages of Decision Trees

- **Overfitting**: They can overfit the training data if they are too deep, memorizing the data instead of learning general patterns, which leads to poor performance on unseen data.
- **Instability**: Small changes in the data may lead to significant differences in the tree structure and predictions.
- **Bias towards Features with Many Categories**: They can become biased toward features with many distinct values, which may cause the model to focus too much on them while missing other important features.
- **Difficulty in Capturing Complex Interactions**: They may struggle to capture complex interactions between features, making them less effective for certain types of data.
- **Computationally Expensive for Large Datasets**: For large datasets, building and pruning a decision tree can be computationally intensive, especially as the tree depth increases.

19 Applications of Decision Trees

Decision trees are used across various fields due to their simplicity, interpretability and versatility. Some key applications are:

1. **Loan Approval in Banking**: Banks use decision trees to assess whether a loan application should be approved, based on factors such as credit score, income, employment status and loan history.
2. **Medical Diagnosis**: In healthcare they assist in diagnosing diseases, for example by predicting whether a patient has diabetes based on clinical data such as glucose levels, BMI and blood pressure.
3. **Predicting Exam Results in Education**: Educational institutions use them to predict whether a student will pass or fail based on factors such as attendance, study time and past grades.

Dept. of ECE, ACETVM

4. **Customer Churn Prediction**: Companies use decision trees to predict whether a customer will leave or stay based on behavior patterns, purchase history and interactions.

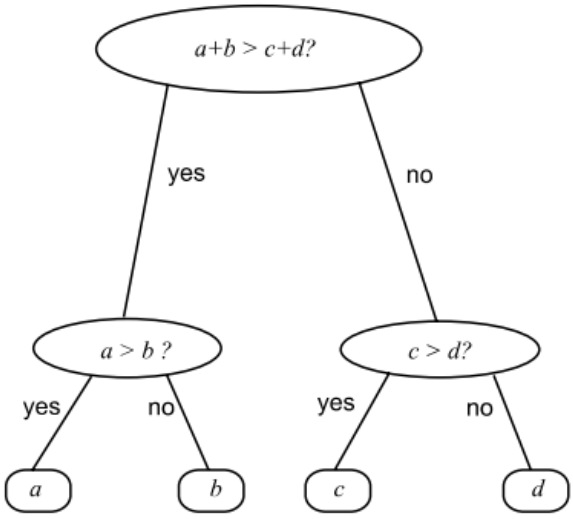
5. **Fraud Detection**: In finance, decision trees are used to detect fraudulent activities such as credit card fraud by analyzing past transaction data and patterns.

A decision tree can also be used to help build automated predictive models that have applications in machine learning, data mining and statistics. By mastering decision trees, we can gain a deeper understanding of data and make more informed decisions across different fields.

Example 1

There are four coins a, b, c, d out of which three coins are of equal weight and one coin is heavier. Find the heavier coin.

Figure 6.1 gives the decision tree corresponding to this problem. The root node compares the weight of $a + b$ with $c + d$. If $a + b$ is heavier than $c + d$, then the outcome is "yes" and it takes the left branch. Otherwise, $c + d$ is heavier and it takes the right branch. The node in the left branch compares the weight of a with b. If the outcome of this comparison is "yes", then it chooses a as the heavier coin. If the outcome is "no", then it chooses b as the heavier coin. This same procedure is done with c and d if the outcome is "no" for the root node.



The example depicted in Figure **??** represents a simple decision tree in the context of decision making. Some of the points illustrated are:

1. There are four leaf nodes corresponding to the four possible outcomes. Each leaf node corresponds to one of the coins being heavier.

Dept. of ECE, ACETVM

2. It requires two weighings to make the final decision or to reach a leaf node. Note that each decision node (non-terminal or internal node) corresponds to a weighing operation. Further, there are two decision nodes from the root to any leaf.

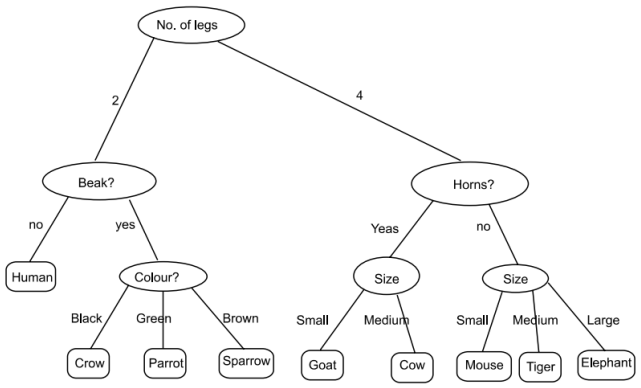
3. Each path from the root to a leaf corresponds to a rule. For example, the rule corresponding to the left-most path is "if $a + b > c + d$ and if $a > b$ then a is heavier".

Example 2

Let us consider a set of animals described in Table 6.1. Figure 6.2 gives a decision tree for the classification of the set of animals.

Table 2: Description of a set of animals

	Legs	Horns	Size	Colour	Beak	Sound
Cow	4	yes	medium	white	no	moo
Crow	2	no	small	black	yes	caw
Elephant	4	no	large	black	no	trumpet
Goat	4	yes	small	brown	no	bleat
Mouse	4	no	small	black	no	squeak
Parrot	2	no	small	green	yes	squawk
Sparrow	2	no	small	brown	yes	chirp
Tiger	4	no	medium	orange	no	roar

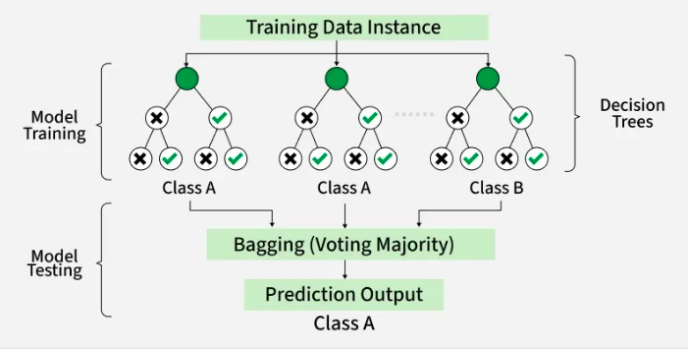


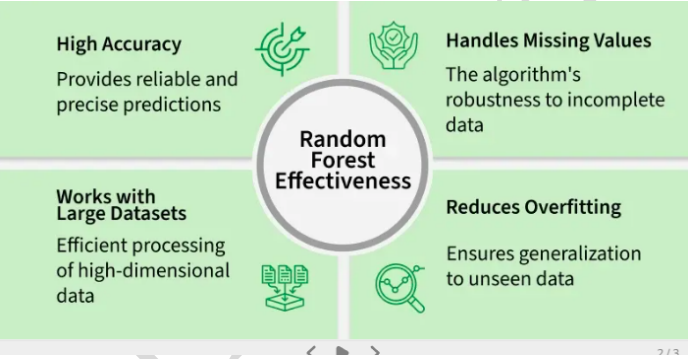
20 Random Forest Algorithm in Machine Learning

Random Forest is a machine learning algorithm that uses many decision trees to make better predictions. Each tree looks at different random parts of the

Dept. of ECE, ACETVM

data and their results are combined by voting for classification or averaging for regression which makes it as ensemble learning technique. This helps in improving accuracy and reducing errors.

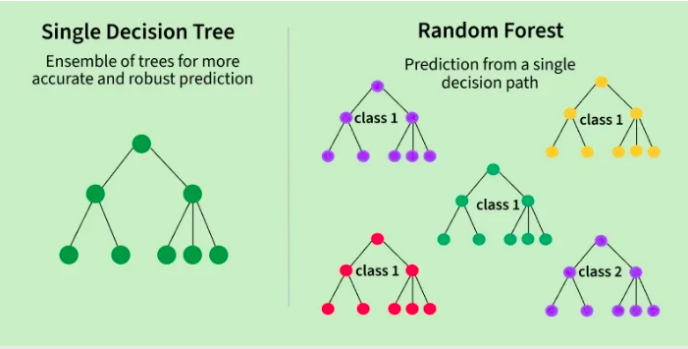




21 Working of Random Forest Algorithm

- **Create Many Decision Trees**: The algorithm makes many decision trees, each using a random part of the data, so every tree is a bit different.
- **Pick Random Features**: When building each tree it does not look at all the features (columns) at once. It picks a few at random to decide how to split the data, which helps the trees stay different from each other.
- **Each Tree Makes a Prediction**: Every tree gives its own answer or prediction based on what it learned from its part of the data.
- **Combine the Predictions**: For classification, the final answer is the category that most trees agree on (majority voting). For regression, the final answer is the average of all the trees' predictions.

Dept. of ECE, ACETVM



- **Why It Works Well**: Using random data and features for each tree helps avoid overfitting and makes the overall prediction more accurate and trustworthy.

22 Key Features of Random Forest

- **Handles Missing Data**: It can work even if some data is missing, so you do not always need to fill in the gaps yourself.
- **Shows Feature Importance**: It tells you which features (columns) are most useful for making predictions, which helps you understand your data better.
- **Works Well with Big and Complex Data**: It can handle large datasets with many features without slowing down or losing accuracy.
- **Used for Different Tasks**: You can use it for both classification (predicting types or labels) and regression (predicting numbers or amounts).

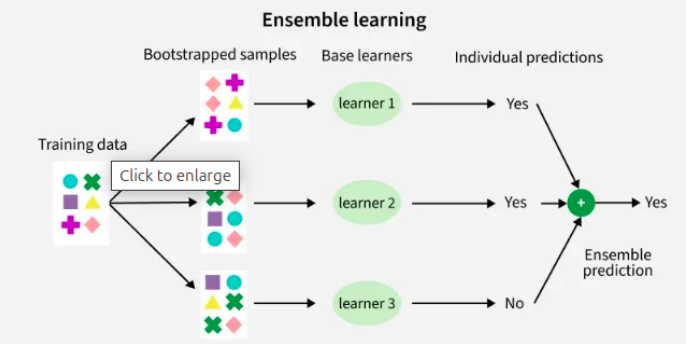
23 Assumptions of Random Forest

- **Each tree makes its own decisions**: Every tree in the forest makes its own predictions without relying on others.
- **Random parts of the data are used**: Each tree is built using random samples and features to reduce mistakes.
- **Enough data is needed**: Sufficient data ensures the trees are different and learn unique patterns and variety.
- **Different predictions improve accuracy**: Combining the predictions from different trees leads to a more accurate final result.

Dept. of ECE, ACETVM

24 Ensemble methods

Ensemble methods are used in data mining due to their ability to enhance the predictive performance of machine learning models. A single model may either overfit the training data or underperform on unseen instances. Ensembles solve these problems by aggregating models and balancing their errors



25 Effectiveness of Ensembles

Ensembles are effective because they address three key challenges in machine learning:

- **Statistical Problem:** When the set of possible models is too large for the available data, multiple models can fit the training data well. A learning algorithm might pick just one of them, which may not generalize well. Ensembles reduce this risk by averaging across multiple models.
- **Computational Problem**: In cases where algorithms cannot efficiently find the optimal model, ensemble learning mitigates this by combining several approximate solutions.
- **Representational Problem**: If the true function is not present in the set of the base learner, ensembles can combine multiple models to better approximate complex target functions.

The main challenge is diversity among the models. For ensembles to be effective, each base model should make different types of errors. Even if individual models are relatively weak, the ensemble can still perform strongly if their mistakes are uncorrelated.

26 Methods for Constructing Ensemble Models

Ensemble methods can be classified into two main categories based on how the base models are trained and combined.

Dept. of ECE, ACETVM

1. Independent Ensemble Construction

In this approach, each base model is trained separately without relying on the others. Randomness is often introduced during the training process to ensure that the models learn different aspects of the data and make diverse errors. Once trained, their predictions are combined using aggregation techniques such as averaging or voting to produce the final output.

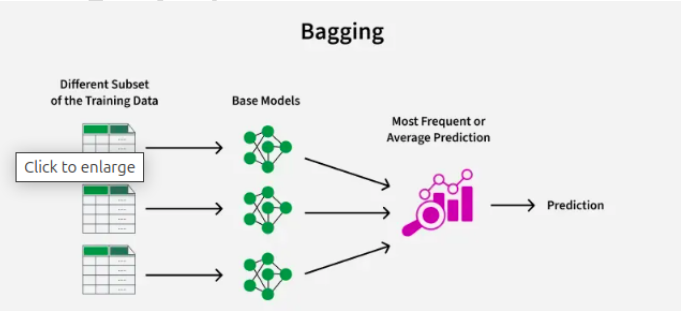
2. Coordinated Ensemble Construction

This approach builds models in a dependent or sequential manner, where each model is influenced by the performance of the previous ones. By focusing on correcting earlier mistakes, the ensemble becomes progressively more accurate. The predictions of these models are then combined in a way that uses their complementary strengths.

27 Types of Ensemble Classifiers

27.1 Bagging (Bootstrap Aggregation)

Bagging trains multiple models independently in parallel, using different bootstrap samples (random samples with replacement) from the training dataset. Each model learns independently on its own subset of data, reducing variance and improving overall prediction stability. The outputs of all models are then combined, typically by averaging (for regression) or majority voting (for classification).



How it Works

- Create multiple bootstrap datasets by randomly sampling with replacement.
- Train a base learner (often a decision tree) on each subset independently.
- Combine predictions from all models for the final output.

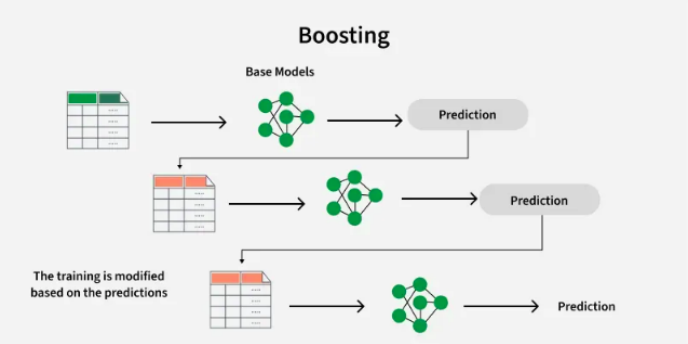
Dept. of ECE, ACETVM

Advantages:

- Reduces variance and helps prevent overfitting.
- Models are trained in parallel, making it efficient.

27.2 Boosting

Boosting builds models sequentially so that each model learns from the errors of the previous ones, improving bias and accuracy. After each iteration, misclassified samples receive higher weights, forcing subsequent models to focus on difficult instances. This process continues for multiple iterations, and the final prediction is formed by combining all models.



How it Works

- Starts with a weak base model (e.g., shallow decision tree).
- Increases weights for misclassified samples after each iteration.
- Combines the predictions of all models to generate the final output.

Advantages:

- Reduces bias and can turn weak learners into strong ones.
- Works well with structured data and provides high accuracy.

28 Advantages and Disadvantages

We have the following advantages and disadvantages of using ensemble learning techniques in data mining.

Dept. of ECE, ACETVM

Advantages

- **Improved Accuracy**: Combining multiple models reduces generalization errors and achieves higher predictive performance than individual models.
- **Robustness**: Less sensitive to data fluctuations and outliers, providing more stable and consistent predictions.
- **Versatility**: Can integrate different types of base models, making them flexible across various data mining tasks and domains.

Disadvantages

- **Lack of Interpretability**: Understanding ensemble behavior is more challenging compared to analyzing a single model.
- **Increased Complexity**: Requires more computational resources and makes deployment or debugging more difficult.
- **Longer Training Time**: Training multiple models and combining their outputs is time-consuming.

Ensemble learning in data mining improves model accuracy and generalization by combining multiple classifiers. Techniques like bagging, boosting and stacking help solve issues such as overfitting and model instability. Ensembles reduce interpretability, but their strong performance on real-world datasets makes them a widely used choice in data mining tasks.

29 Short Answer Questions

1. Define linear regression. Derive the normal–equation solution for the parameters of a univariate linear regression model.
2. Explain the role of the hypothesis function and decision boundary in logistic regression. How is a class label finally decided from the model output?
3. Define mean squared error (MSE). Why is MSE the most commonly used error function in linear regression? Mention two drawbacks of using MSE.
4. Differentiate between L1 and L2 loss functions used in regression. Give one advantage and one disadvantage of each.
5. What is cross-entropy loss in binary classification? Write the expression for cross-entropy in logistic regression and explain the meaning of each term.
6. With a neat expression, explain multivariate (multiple) linear regression. How does it differ from simple (univariate) linear regression?

Dept. of ECE, ACETVM

7. State the Naive Bayes assumption. Derive the decision rule for a Naive Bayes classifier for classifying a feature vector into one of k classes.
8. Explain how class-conditional probabilities and prior probabilities are estimated in a Naive Bayes classifier for categorical attributes.
9. What is a support vector in Support Vector Machines? Explain the concept of margin and soft-margin with suitable sketches.
10. List any four kernel functions used in SVMs. Briefly explain why the kernel trick is useful for non-linearly separable data.
11. Differentiate between classification trees and regression trees. Mention two issues commonly encountered in decision tree learning.
12. Explain the concepts of entropy and information gain in decision tree construction. How are they used to select a splitting attribute?
13. What is a random forest? Describe how randomness is introduced in the construction of individual trees and in the selection of features.
14. State any four advantages of random forests over a single decision tree in classification tasks.
15. Define ensemble learning. Distinguish between bagging and boosting in terms of
    (a) the way training sets are formed and
    (b) the way individual models are combined.
16. Explain the working principle of bagging with an example. How does bagging reduce variance?
17. Describe the main idea of the AdaBoost algorithm. How are sample weights and model weights updated across iterations?
18. What are stacking ensembles? Explain the roles of base learners and meta-learner in stacking.
19. Write short notes on:
    (a) bias–variance trade-off in regression,
    (b) effect of regularization (L1/L2) on model complexity.
20. For a binary classification problem, explain how logistic regression, Naive Bayes, and SVM differ in
    (a) the form of the decision function and
    (b) the way model parameters are estimated.

Dept. of ECE, ACETVM

Solutions (4 Marks Each)

1. **Linear regression and normal equations**

   Linear regression models the relationship between an input x and output y as

   $$\hat{y} = h_\theta(x) = \theta_0 + \theta_1 x.$$

   For a dataset $\{(x_i, y_i)\}_{i=1}^n$, the cost (MSE) is

   $$J(\theta) = \frac{1}{2n} \sum_{i=1}^{n} (h_\theta(x_i) - y_i)^2.$$

   Setting $\nabla_\theta J(\theta) = 0$ and writing in matrix form with $X \in \mathbb{R}^{n \times (d+1)}$, $y \in \mathbb{R}^n$, the normal equation is

   $$\theta = (X^T X)^{-1} X^T y.$$

2. **Hypothesis and decision boundary in logistic regression**

   Logistic regression models the posterior of class 1 as

   $$p(y = 1 \mid x) = \sigma(w^T x + b) = \frac{1}{1 + e^{-(w^T x + b)}}.$$

   The hypothesis is this probability mapping from features to $[0, 1]$. The decision boundary is defined by $p(y = 1 \mid x) = 0.5$, i.e.

   $$w^T x + b = 0,$$

   which is a line or hyperplane in feature space. Points with $p(y = 1 \mid x) \geq 0.5$ are assigned to class 1; others to class 0.

3. **Definition and role of MSE; drawbacks**

   Mean squared error for regression is

   $$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2.$$

   It is widely used because it is differentiable, convex for linear models, and has a closed-form solution via the normal equations. However, MSE is highly sensitive to outliers (large errors are squared) and assumes Gaussian noise, which may be violated, leading to sub-optimal robustness.

4. **L1 vs L2 loss: advantages and disadvantages**

   L1 loss uses absolute error

   $$L_1 = \sum_{i=1}^{n} |y_i - \hat{y}_i|,$$

Dept. of ECE, ACETVM

which is robust to outliers and leads to median-based estimates, but is not differentiable at zero and can be harder to optimize. L2 loss uses squared error

$$L_2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2,$$

which is smooth and easy to optimize but is sensitive to outliers and can be dominated by few large errors.

5. **Cross-entropy loss in logistic regression**

   For binary classification with true label $y \in \{0, 1\}$ and predicted probability $\hat{p} = p(y = 1 \mid x)$, the cross-entropy loss is

   $$L = -\big(y \log \hat{p} + (1 - y) \log(1 - \hat{p})\big).$$

   The first term penalizes the model when a positive sample is assigned low probability; the second term penalizes negative samples assigned high probability. Minimizing this loss maximizes the likelihood of the observed labels.

6. **Multivariate (multiple) linear regression**

   In multivariate linear regression with feature vector $x \in \mathbb{R}^d$,

   $$\hat{y} = h_\theta(x) = \theta_0 + \theta_1 x_1 + \cdots + \theta_d x_d = \theta^T \tilde{x},$$

   where $\tilde{x}$ includes a bias term. Parameters are learned by minimizing MSE as in the univariate case, using

   $$\theta = (X^T X)^{-1} X^T y.$$

   It differs from univariate regression by modelling the joint linear effect of multiple features simultaneously.

7. **Naive Bayes assumption and decision rule**

   Naive Bayes assumes conditional independence of features given the class:

   $$P(x_1, \ldots, x_d \mid y) = \prod_{j=1}^{d} P(x_j \mid y).$$

   Using Bayes' theorem,

   $$P(y_k \mid x) \propto P(y_k) \prod_{j=1}^{d} P(x_j \mid y_k).$$

   The classifier predicts the class

   $$\hat{y} = \arg\max_k \; P(y_k) \prod_{j=1}^{d} P(x_j \mid y_k),$$

   or equivalently the maximum of its logarithm.

Dept. of ECE, ACETVM

8. **Estimating probabilities in Naive Bayes**

   Prior probabilities are estimated as

$$\hat{P}(y_k) = \frac{\text{number of training samples in class } k}{n}.$$

   For categorical attributes, class-conditional probabilities are estimated by relative frequencies,

$$\hat{P}(x_j = v \mid y_k) = \frac{\text{count}(x_j = v, y_k)}{\text{count}(y_k)},$$

   often with Laplace smoothing to avoid zero probabilities. During prediction these estimates are plugged into the Naive Bayes decision rule.

9. **Support vector and margin in SVM**

   In a linear SVM, the decision function is $f(x) = w^T x + b$. Support vectors are training points that lie closest to the separating hyperplane and satisfy the margin constraints. The (hard-margin) SVM maximizes the geometric margin

$$\gamma = \frac{2}{\|w\|},$$

   subject to $y_i(w^T x_i + b) \geq 1$. Soft-margin SVM introduces slack variables ξ_i and a penalty parameter C to allow misclassifications while still maximizing a large margin.

10. **Kernel functions and kernel trick in SVM**

    Common kernels include linear $K(x, z) = x^T z$, polynomial $K(x, z) = (x^T z + c)^p$, radial basis function $K(x, z) = \exp(-\gamma\|x - z\|^2)$, and sigmoid $K(x, z) = \tanh(\kappa x^T z + c)$. The kernel trick replaces explicit mapping $\phi(x)$ by an inner product $K(x, z) = \phi(x)^T \phi(z)$, enabling SVMs to learn non-linear decision boundaries in high-dimensional feature spaces without computing $\phi(x)$ explicitly.

11. **Classification vs regression trees; common issues**

    A classification tree predicts discrete class labels using impurity measures such as entropy or Gini index at internal nodes. A regression tree predicts continuous outputs, typically using variance or MSE reduction as the split criterion. Common issues include overfitting due to deep trees, high variance with small changes in data, bias toward attributes with many levels, and instability in learned structure.

12. **Entropy and information gain for splitting**

    For a node with class distribution $\{p_1, \ldots, p_K\}$, entropy is

$$H(S) = -\sum_{k=1}^{K} p_k \log_2 p_k.$$

Dept. of ECE, ACETVM

Splitting on attribute A partitions data into subsets S_v. The information gain is

$$\text{Gain}(S, A) = H(S) - \sum_v \frac{|S_v|}{|S|} H(S_v).$$

The attribute with maximum information gain is selected for splitting, since it produces the largest reduction in class uncertainty.

13. **Random forest and sources of randomness**

A random forest is an ensemble of decision trees where each tree is trained on a bootstrap sample of the training data. At each split, the tree considers only a random subset of features instead of all features. Bootstrap sampling introduces randomness in the data seen by each tree, and feature subsampling decorrelates trees, leading to a more robust aggregated prediction via majority vote or averaging.

14. **Advantages of random forests over single trees**

Random forests reduce variance by averaging many de-correlated trees, improving generalization compared to a single deep tree. They handle high-dimensional data and complex interactions effectively with minimal feature engineering. They are relatively robust to noise and overfitting, provide internal estimates of generalization error via out-of-bag samples, and can compute feature importance scores.

15. **Ensemble learning; bagging vs boosting**

Ensemble learning combines multiple base learners to obtain better predictive performance than any individual model. In bagging, each model is trained independently on a bootstrap sample of the data and their outputs are combined by averaging (regression) or voting (classification); the goal is variance reduction. In boosting, models are trained sequentially on reweighted data that emphasizes previously misclassified examples, and predictions are combined using weighted voting, primarily reducing bias and building a strong learner from many weak learners.

16. **Working principle of bagging and variance reduction**

Bagging (bootstrap aggregating) generates B bootstrap datasets by sampling with replacement from the original training set. A base learner (often a decision tree) is trained on each bootstrap set, and the final prediction is the average or majority vote of all learners. Since each model sees a different sample, their errors tend to cancel out, reducing variance of the ensemble relative to any single model.

17. **Main idea of AdaBoost; updating weights**

AdaBoost trains weak learners sequentially on weighted training data. Initially all samples have equal weights. At iteration t a weak learner $h_t(x)$

Dept. of ECE, ACETVM

is trained and its weighted error ϵ_t is computed. The learner's weight is

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right).$$

Sample weights are then updated: misclassified points have their weights increased, correctly classified ones decreased, and weights are normalized to sum to one. The final classifier is

$$H(x) = \text{sign} \left(\sum_t \alpha_t h_t(x) \right).$$

18. **Stacking ensembles: base learners and meta-learner**

In stacking, multiple diverse base learners are trained on the original training data. Their predictions (on validation data or via cross-validation) are collected as new features. A meta-learner is then trained on this second-level dataset to learn how best to combine base predictions. During testing, base learners first produce predictions, which are fed to the meta-learner to output the final prediction, exploiting complementary strengths of different algorithms.

19. **Bias–variance trade-off and effect of regularization**

Total expected prediction error can be decomposed into bias, variance, and irreducible noise. High-bias models (too simple) underfit, while high-variance models (too complex) overfit; an optimal trade-off minimizes total error.[web:49] Regularization adds a penalty such as $\lambda \|w\|_2^2$ (L2) or $\lambda \|w\|_1$ (L1) to the loss, shrinking coefficients and effectively reducing model complexity.[web:43][web:46] Increasing regularization generally decreases variance but increases bias; choosing λ via validation yields good generalization.

20. **Logistic regression vs Naive Bayes vs SVM**

Logistic regression directly models $P(y \mid x)$ using a sigmoid applied to a linear combination of features, and its parameters are estimated by maximum likelihood (typically via gradient descent). Naive Bayes is a generative model that estimates $P(y)$ and $P(x \mid y)$ under conditional independence and uses Bayes' rule to compute $P(y \mid x)$.[web:41][web:42] SVMs construct a discriminative decision function $f(x) = w^T x + b$ (or its kernelized form) that maximizes the margin between classes; parameters are obtained by solving a constrained optimization problem. Logistic regression outputs calibrated probabilities; Naive Bayes is simple and works well with high-dimensional sparse data; SVM focuses on boundary points and can create complex non-linear boundaries via kernels.

Dept. of ECE, ACETVM

PEECT413 Machine Learning

Module 3 Notes

S4 ECE, 2024 Scheme KTU

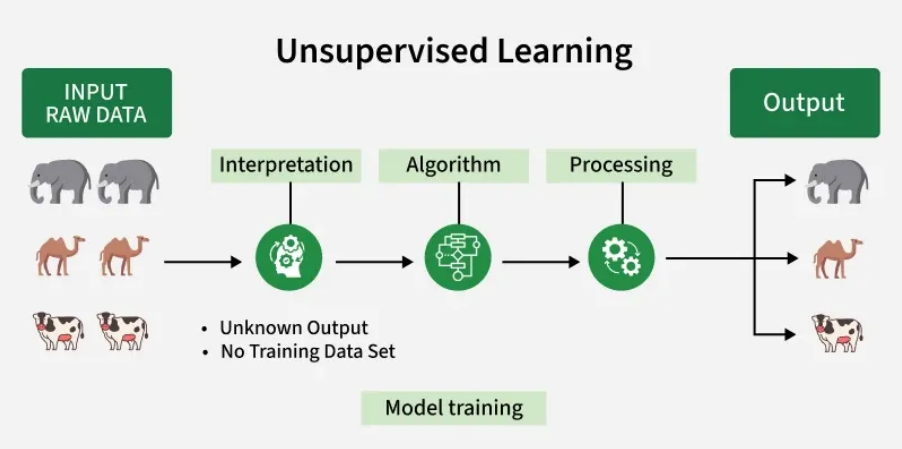
Module 3

Module 3 Syllabus

Unsupervised learning: Clustering-K-means, High, Hierarchical clustering, criterion functions for clustering, proximity measures, Euclidean, Manhattan, Minkowski Distances, Cosine Similarity. Reinforcement Learning: Agent based learning, Q-learning, Introduction to HMM models

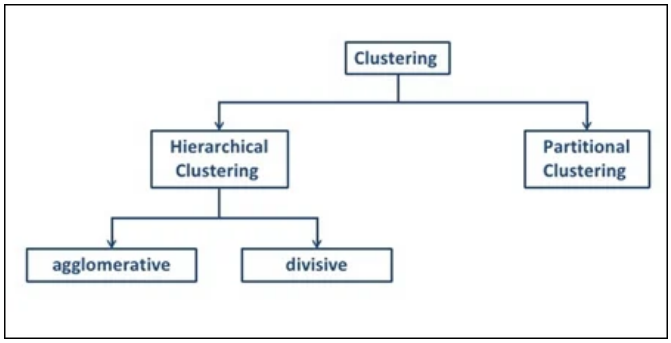
1. Unsupervised Learning

Unsupervised learning works with **unlabeled** data and tries to discover hidden structure or patterns in the data. There is no teacher or target output; the algorithm must organize data by itself based on similarity or density.



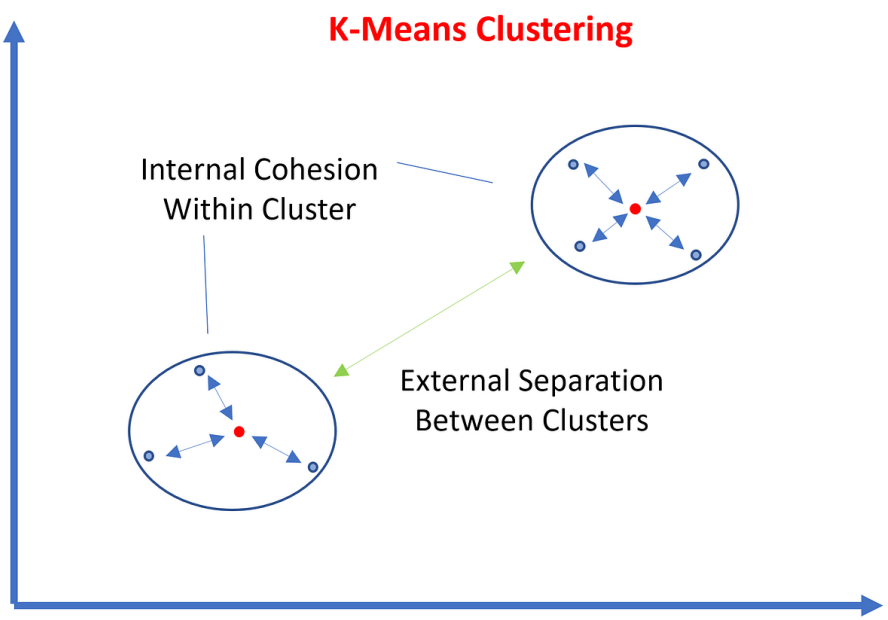
1.1 Clustering

Clustering aims to group similar data points into clusters so that points in the same cluster are more similar to each other than to points in other clusters. It is widely used in customer segmentation, document grouping, image compression, etc.



1.1.1 K-Means Clustering

Idea: K-means is a partition-based clustering algorithm that divides the data into k non-overlapping clusters, each represented by its centroid (mean vector). Each cluster is characterized by the average of its points, and every point belongs to exactly one cluster.



Objective Function (SSE): K-means minimizes the **sum of squared errors** (SSE) within clusters:

$$J = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2$$

where C_i is the i-th cluster and μ_i is its centroid. This ensures points are as close as possible to their cluster center (high compactness).

Algorithm Steps:

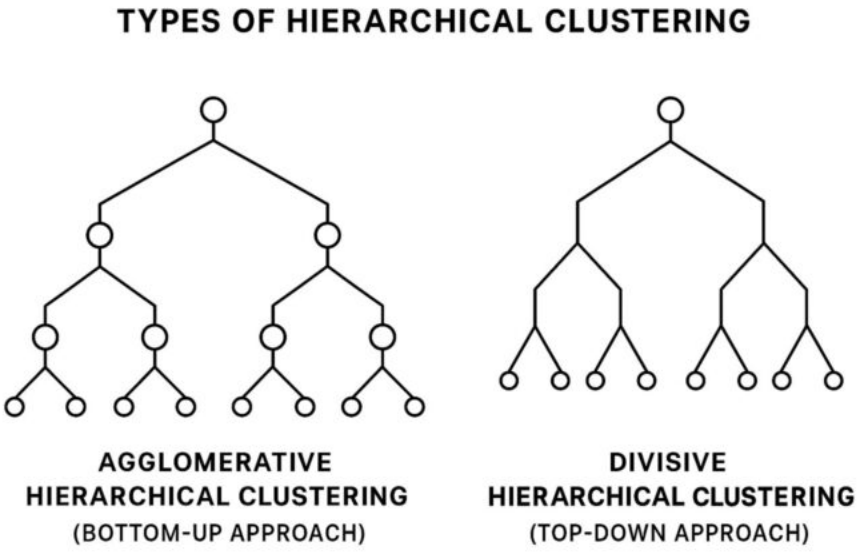
1. Choose the number of clusters k and initialize k centroids (often randomly or using heuristics like k-means++).
2. **Assignment step:** For each point, compute distance to all centroids and assign the point to the nearest centroid.
3. **Update step:** For each cluster, recompute the centroid as the mean of all points assigned to that cluster.
4. Repeat the assignment and update steps until:
    - Centroids stop changing significantly, or
    - A maximum number of iterations is reached.

Remarks:

- Works well when clusters are compact, spherical, and of similar size.
- Sensitive to:
    - Initial centroid choice.
    - Outliers and noise.
- May converge to a local (not global) minimum of the objective function.

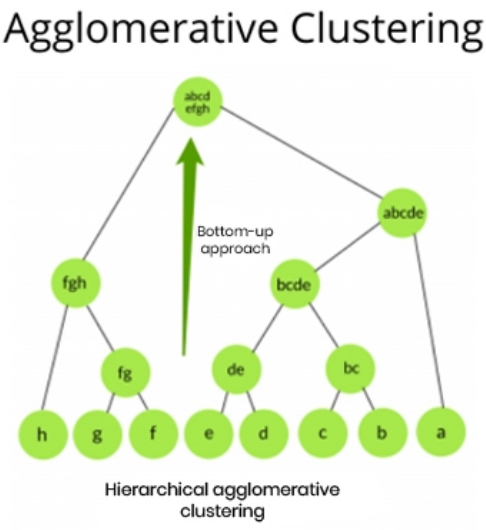
1.1.2 Hierarchical Clustering

Idea: Hierarchical clustering builds a **tree-like hierarchy** of clusters and does not require choosing the number of clusters k in advance. It allows us to view clustering at different levels of granularity.



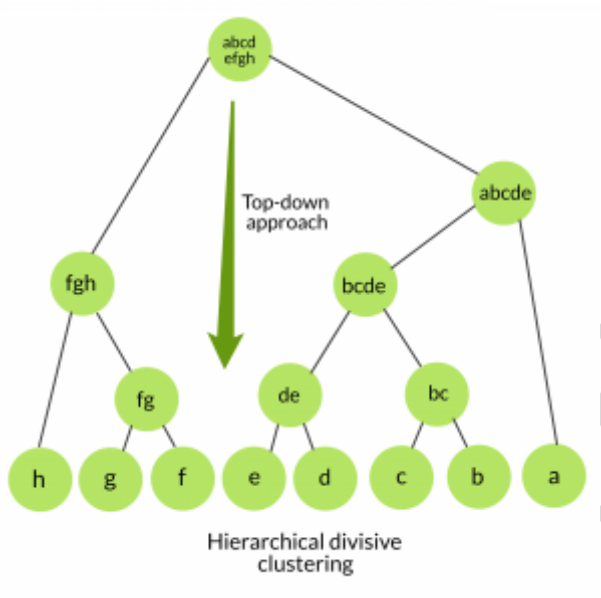
Types:

- **Agglomerative (bottom-up):**
  - Start with each data point as a separate cluster.
  - At each step, merge the two closest clusters based on a distance (linkage) criterion.
  - Continue until all points are merged into one big cluster.



- **Divisive (top-down):**
  - Start with all points in a single cluster.

- Recursively split clusters into smaller sub-clusters.
- Continue until each point is in its own cluster or a stopping condition is reached.



Linkage Criteria (for agglomerative):

- **Single linkage:** Distance between two clusters = minimum distance between any pair of points (one from each cluster).
- **Complete linkage:** Distance between two clusters = maximum distance between any pair of points.
- **Average linkage:** Distance = average pairwise distance between points in the two clusters.

Dendrogram:

- A **dendrogram** is a tree diagram that shows the order and distances at which clusters are merged or split.
- By choosing a horizontal cut at a certain height in the dendrogram, we can obtain a desired number of clusters.

1.2 Criterion Functions for Clustering

Criterion functions evaluate how good a clustering solution is. They capture the goals of clustering such as compactness, separation, and structure.

Within-Cluster Scatter (Compactness):

- Measures how close data points in a cluster are to each other (or to the centroid).
- For K-means, the commonly used criterion is:

$$J_{\text{SSE}} = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2$$

- Lower values indicate that clusters are dense and tight.

Between-Cluster Separation:

- Measures how far apart cluster centroids are from each other.
- A good clustering typically aims for:
  - Low within-cluster scatter (compact clusters).
  - High between-cluster separation (well-separated clusters).

1.3 Proximity Measures

Proximity measures quantify similarity or dissimilarity between data points and are fundamental in clustering, classification, and information retrieval.

Let $p = (p_1, p_2, \ldots, p_n)$ and $q = (q_1, q_2, \ldots, q_n)$ be two data points in n-dimensional space.

1.3.1 Euclidean Distance

$$d_{\text{Euclidean}}(p, q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

- Straight-line (as-the-crow-flies) distance between two points.
- Widely used in K-means and many geometric algorithms.

1.3.2 Manhattan Distance

$$d_{\text{Manhattan}}(p, q) = \sum_{i=1}^{n} |p_i - q_i|$$

- Also called **city-block** or L_1 distance.
- Represents the distance if movement is only allowed along axes (like a grid of city streets).

1.3.3 Minkowski Distance

$$d_{\text{Minkowski}}(p, q) = \left(\sum_{i=1}^{n} |p_i - q_i|^r \right)^{1/r}$$

- Generalization of both Euclidean and Manhattan distances.
- For $r = 1$: Minkowski becomes Manhattan distance.
- For $r = 2$: Minkowski becomes Euclidean distance.
- Higher r values put more emphasis on larger coordinate differences.

1.3.4 Cosine Similarity

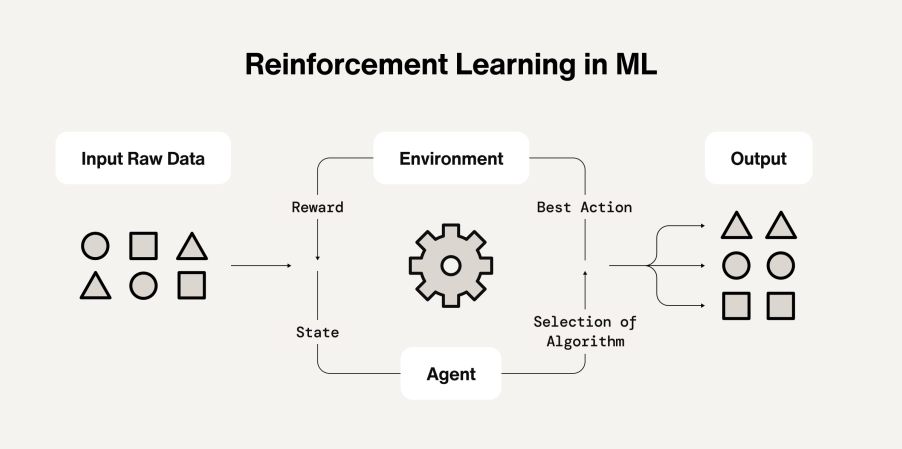
$$\text{CosineSimilarity}(p, q) = \frac{p \cdot q}{\|p\| \, \|q\|} = \frac{\sum_{i=1}^{n} p_i q_i}{\sqrt{\sum_{i=1}^{n} p_i^2} \, \sqrt{\sum_{i=1}^{n} q_i^2}}$$

- Measures the cosine of the angle between two vectors.
- Focuses on *direction* rather than magnitude.
- Commonly used in text mining, where document vectors are high-dimensional and sparse.

2. Reinforcement Learning

Reinforcement Learning (RL) is a learning paradigm where an **agent** learns to take actions in an environment to maximize cumulative reward through trial and error. The agent learns from feedback (rewards or penalties) rather than from labeled examples.

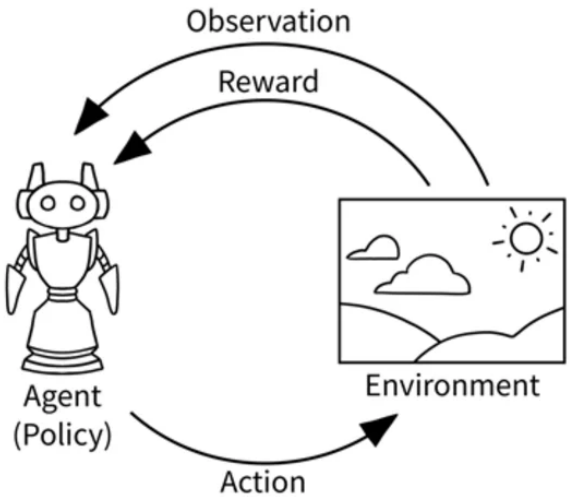
Dept. of ECE, ACETVM



2.1 Agent-Based Learning

Key Elements:

- **Agent:** The learner or decision maker.
- **Environment:** Everything outside the agent that it interacts with.
- **State (s):** A description of the current situation of the environment.
- **Action (a):** A choice the agent can make when in a particular state.
- **Reward (r):** Scalar feedback signal indicating the immediate benefit of an action.
- **Policy (π):** A mapping from states to actions (defines agent behaviour).



Interaction Loop: At each time step t:

1. Agent observes state s_t.
2. Agent selects action a_t according to policy π.
3. Environment returns reward r_{t+1} and next state s_{t+1}.
4. Agent updates its internal values (e.g., Q-values) to improve decisions over time.

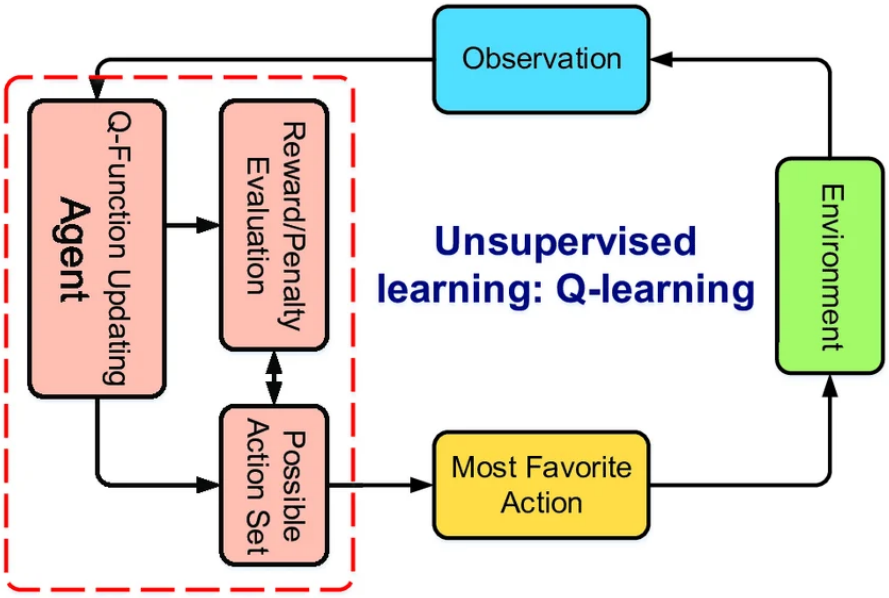
Goal: The main goal is to learn a policy that maximizes the expected **cumulative discounted reward** over time.

2.2 Q-Learning

Q-learning is a model-free RL algorithm that learns the **action-value function** $Q(s, a)$. It does not need a model of the environment's transition probabilities.

Q-Function:

$$Q(s, a) \approx \text{expected discounted sum of future rewards when taking action } a \text{ in state } s$$



Q-Learning Update Rule:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right]$$

where:

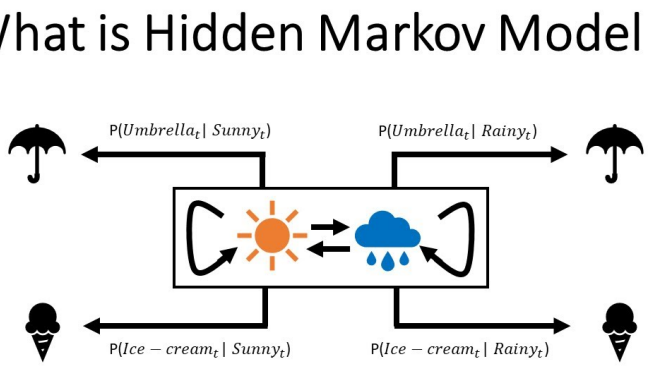
- α: Learning rate ($0 < \alpha \leq 1$), controls how much new information overrides old.
- γ: Discount factor ($0 \leq \gamma < 1$), controls importance of future rewards vs. immediate reward.
- r: Immediate reward received after taking action a in state s.
- s': Next state after the action.
- $\max_{a'} Q(s', a')$: Estimate of the best possible future value from the next state.

Characteristics:

- Off-policy algorithm: learns the optimal policy independently of the agent's current behaviour policy.
- Widely used in grid-world problems, games, and control tasks.

2.3 Introduction to Hidden Markov Models (HMMs)

Idea: A Hidden Markov Model (HMM) is a probabilistic model for sequential data where the system is assumed to be a Markov process with **hidden** (unobserved) states and **observable** outputs. While we cannot directly see the state sequence, we can observe outputs generated from those states.



Components of an HMM:

- A set of hidden states $S = \{s_1, s_2, \ldots, s_N\}$.
- Transition probability matrix $A = [a_{ij}]$, where

$$a_{ij} = P(s_{t+1} = s_j \mid s_t = s_i), \quad \sum_j a_{ij} = 1.$$

- Observation (emission) probability matrix B, where

$$b_j(o_t) = P(o_t \mid s_t = s_j), \quad \sum_o b_j(o) = 1.$$

- Initial state distribution $\pi_i = P(s_1 = s_i)$.

Typical Problems in HMMs:

- **Evaluation:** Compute probability of an observation sequence given the model.
- **Decoding:** Find the most likely sequence of hidden states (e.g., using Viterbi algorithm).
- **Learning:** Adjust model parameters (A, B, π) to maximize likelihood of observed data (e.g., using Baum-Welch algorithm).

Applications:

- Speech recognition (phoneme/state sequence hidden, audio frames observed).
- Bioinformatics (hidden functional states, observed DNA/protein sequences).
- Time-series, gesture recognition, and other sequential pattern tasks.

Short Answer Questions

1. Define unsupervised learning. Give one real-life example.
2. What is clustering? How is it different from classification?
3. State any two applications of clustering in real life.
4. Define K-means clustering.
5. What is the objective function used in K-means clustering?

6. List the main steps in K-means clustering algorithm.
7. Mention two limitations of K-means clustering.
8. What is hierarchical clustering?
9. Differentiate agglomerative and divisive hierarchical clustering.
10. What is a dendrogram?
11. Define criterion function in clustering.
12. What is within-cluster scatter?
13. List any two desirable properties of a good clustering.
14. Define Euclidean distance between two points.
15. Define Manhattan distance.
16. Write the general expression for Minkowski distance.
17. Write the formula for cosine similarity.
18. What is reinforcement learning? Give an example.
19. Define: agent, environment, state, action, reward.
20. What is a policy in reinforcement learning?
21. What is Q-learning?
22. Write the Q-learning update rule and identify each term.
23. What is a Hidden Markov Model (HMM)?
24. List the main components of an HMM.
25. Mention any two applications of HMM.

Part B – Medium Answer Questions

1. Explain K-means clustering algorithm with neat steps.
2. Derive and explain the objective function minimized in K-means clustering.
3. Write the algorithm for K-means clustering and comment on its convergence.
4. Compare K-means clustering with hierarchical clustering (any four points).
5. Explain agglomerative hierarchical clustering. Describe single, complete and average linkage.
6. With a neat sketch, explain how a dendrogram is used to obtain clusters.
7. Explain the terms: within-cluster scatter and between-cluster separation.
8. Discuss the role of criterion functions in clustering.
9. Define Euclidean, Manhattan and Minkowski distances with formulas and give one use-case for each.
10. Explain cosine similarity. Why is it preferred for text data?
11. Explain the basic elements of a reinforcement learning system with a block diagram.
12. Distinguish supervised, unsupervised and reinforcement learning.

Dept. of ECE, ACETVM

13. Explain the idea of agent-based learning with a simple example.
14. Explain Q-learning. What is the significance of learning rate and discount factor?
15. Write a short note on exploration vs exploitation in reinforcement learning.
16. Explain the basic elements of an HMM and list any two applications.

Dept. of ECE, ACETV

PEECT413 Machine Learning

Module 4 Notes

S4 ECE, 2024 Scheme KTU

Module 4

Module 4 Syllabus

Introduction to Artificial Neural Networks: Biological Neuron, Perceptron, Training, Limitations, XOR Problem, Multilayer Perceptron, Gradient Based Learning, Stochastic Gradient Descent, Activation Functions – Sigmoid, ReLU, Tanh. Backpropagation – Chain Rule, Regularization – L1, L2.

1 Biological Neuron

1.1 What is a Biological Neuron?

A biological neuron is a brain cell that processes and transmits information. The human brain has about 86 billion neurons working together, connected through trillions of synaptic connections.

1.2 Parts of a Neuron

Think of a neuron like a tiny computer:

- **Dendrites**: Input wires that receive signals from other neurons
- **Cell Body (Soma)**: The processor that adds up all incoming signals
- **Axon**: Output wire that sends the signal forward to other neurons
- **Synapse**: Connection point between two neurons where signals are transferred

1.3 How Does a Neuron Work?

1. Dendrites receive signals from neighbouring neurons
2. The cell body sums up all incoming signals
3. If the total exceeds a threshold, the neuron *fires* (sends an action potential)
4. The signal travels down the axon to the next neuron via synapses

Key Idea: A neuron fires only when the total input crosses a threshold – this idea directly inspired the artificial neuron.

1.4 From Biology to Artificial Neurons

2 Perceptron

2.1 What is a Perceptron?

A **perceptron** is the simplest artificial neuron, proposed by Frank Rosenblatt in 1958. It takes multiple inputs, multiplies each by a weight, sums them up, and produces a binary decision: *Yes* or *No*.

Real-world analogy: Deciding whether to watch a movie. You weigh factors: good reviews (+3), friend recommends (+2), expensive ticket (−2), far away (−1). Total = $+2 > 0$, so you go!

Dept. of ECE, ACET

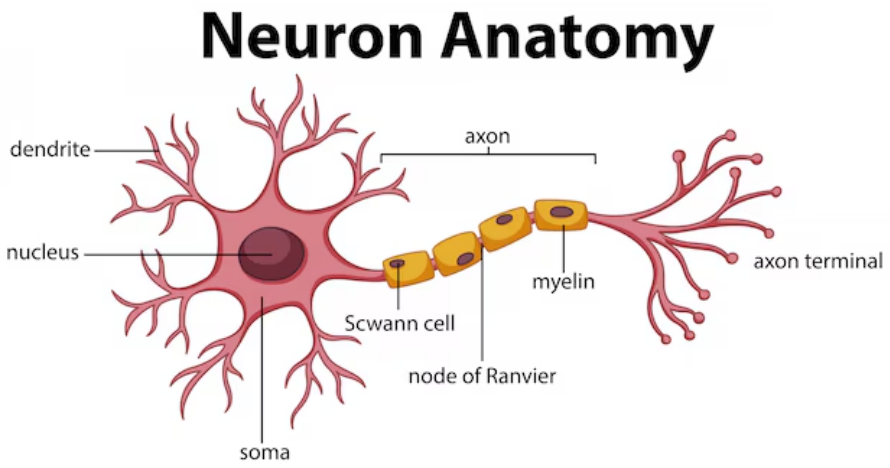


Figure 1: Biological neuron

Biological Neuron	**Artificial Neuron**
Dendrites (inputs)	Input signals x_i
Synaptic strength	Learnable weights w_i
Cell body (summation)	Weighted sum $\sum w_i x_i + b$
Threshold	Activation function $f(\cdot)$
Axon (output)	Output y

Table 1: Analogy between biological and artificial neurons

2.2 Mathematical Formulation

Step 1 – Compute the weighted sum:

$$z = \sum_{i=1}^{n} w_i x_i + b = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b \tag{1}$$

where:

- x_i – input features
- w_i – weights (how important each input is)
- b – bias (shifts the decision boundary)
- z – net input (pre-activation value)

Step 2 – Apply the step (sign) function:

$$y = \begin{cases} +1 & \text{if } z \geq 0 \quad \text{(Yes / Class 1)} \\ -1 & \text{if } z < 0 \quad \text{(No / Class 0)} \end{cases} \tag{2}$$

2.3 Geometric Interpretation

The perceptron defines a **linear decision boundary** (hyperplane):

$$w_1 x_1 + w_2 x_2 + b = 0 \tag{3}$$

Points on one side are classified as Class 1; points on the other side as Class 0.

Dept. of ECE, ACETVM

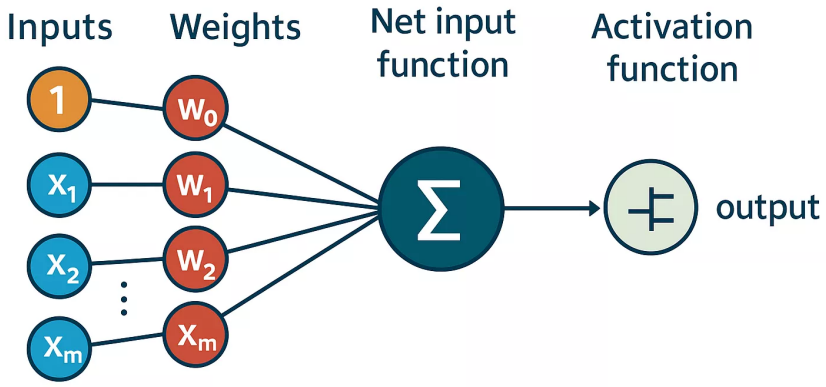
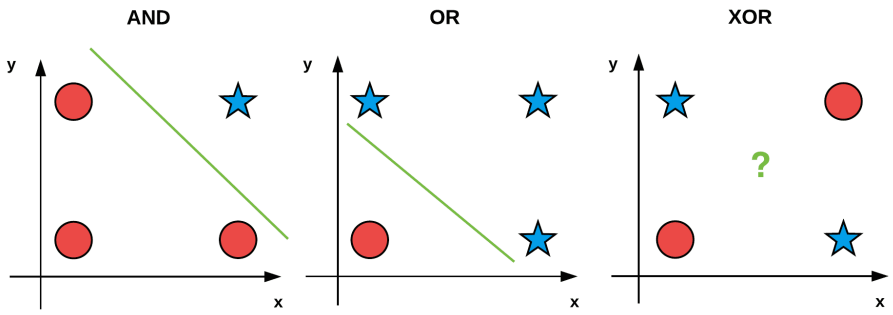


Figure 2: Perceptron



2.4 Example: AND Gate

Solution: Choose $w_1 = 1,\ w_2 = 1,\ b = -1.5$

Verification:

- $(0,0)$: $z = 0 + 0 - 1.5 = -1.5 < 0 \Rightarrow y = 0$ ✓
- $(0,1)$: $z = 0 + 1 - 1.5 = -0.5 < 0 \Rightarrow y = 0$ ✓
- $(1,0)$: $z = 1 + 0 - 1.5 = -0.5 < 0 \Rightarrow y = 0$ ✓
- $(1,1)$: $z = 1 + 1 - 1.5 = +0.5 \geq 0 \Rightarrow y = 1$ ✓

3 Perceptron Training

3.1 What is Training?

Training is the process of finding weights w_i and bias b so that the perceptron correctly classifies all training examples.

Analogy: Like learning to throw a basketball. Each miss gives you feedback (throw harder, aim left) and each small adjustment brings you closer to making the shot.

3.2 The Learning Rule

When the perceptron makes a wrong prediction on example $(\mathbf{x}, t)$, update weights as:

x_1	x_2	AND
0	0	0
0	1	0
1	0	0
1	1	1

Table 2: Truth table for AND gate

$$w_i^{\text{new}} = w_i^{\text{old}} + \eta\,(t-y)\,x_i \tag{4}$$
$$b^{\text{new}} = b^{\text{old}} + \eta\,(t-y) \tag{5}$$

where:

- η – learning rate (step size, e.g. 0.01–1.0)
- t – true label
- y – predicted label
- $(t-y)$ – error (0 if correct, non-zero if wrong)

3.3 Training Algorithm

```
Initialize: w = 0, b = 0

Repeat until convergence:
    For each training example (x, t):
        Compute output: y = sign(w . x + b)
        If y != t:
            Update: w = w + eta * (t - y) * x
            Update: b = b + eta * (t - y)
```

3.4 Convergence Theorem

Good news: If the training data is *linearly separable*, the perceptron is **guaranteed to converge** in a finite number of steps (Rosenblatt, 1962).

Maximum number of updates $\leq \frac{R^2}{\gamma^2}$, where $R = \max \|\mathbf{x}\|$ (largest input norm) and γ is the classification margin.

4 Limitations of the Perceptron

4.1 The Core Problem

A single perceptron can only classify data that is **linearly separable** – i.e. a single straight line (or hyperplane) can separate the two classes.

- AND gate, OR gate – linearly separable ✓
- XOR gate – **not** linearly separable ×

4.2 Consequences

- Cannot solve problems with curved or multiple decision boundaries
- Cannot represent complex, real-world patterns
- Only one layer ⇒ limited representational power

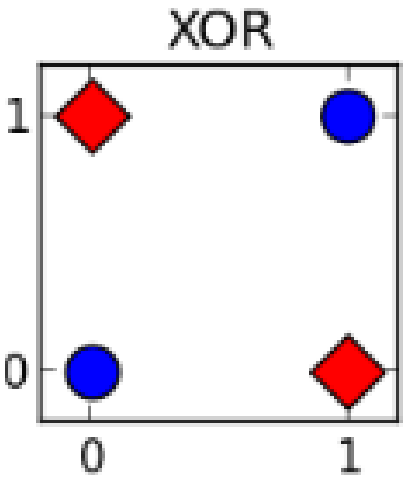
This limitation, highlighted by Minsky & Papert (1969), triggered the first "AI Winter" in the 1970s.

Dept of ECE, ACETVM

5 The XOR Problem

5.1 What is XOR?

XOR (exclusive OR) outputs 1 when the inputs are *different*, and 0 when they are the *same*.



x_1	x_2	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Table 3: Truth table for XOR gate

5.2 Why Perceptron Fails on XOR

Plot the four points in 2D:

- Class 0: $(0, 0)$ and $(1, 1)$ – diagonal corners
- Class 1: $(0, 1)$ and $(1, 0)$ – other diagonal corners

No single straight line can separate these two classes.

5.3 Mathematical Proof by Contradiction

Assume a linear separator exists: $w_1x_1 + w_2x_2 + b = 0$.
From the four XOR constraints:

1. $(0, 0) \in$ Class 0: $b < 0$
2. $(0, 1) \in$ Class 1: $w_2 + b > 0 \Rightarrow w_2 > -b > 0$
3. $(1, 0) \in$ Class 1: $w_1 + b > 0 \Rightarrow w_1 > -b > 0$
4. $(1, 1) \in$ Class 0: $w_1 + w_2 + b < 0$

From (2) and (3): $w_1 + w_2 > -2b$
From (4): $w_1 + w_2 < -b$
Therefore $-b > -2b \Rightarrow b > 0$.
But condition (1) requires $b < 0$. **Contradiction!**
No linear separator exists $\Rightarrow$ a single perceptron *cannot* solve XOR.

Dept of ECE, ACETV

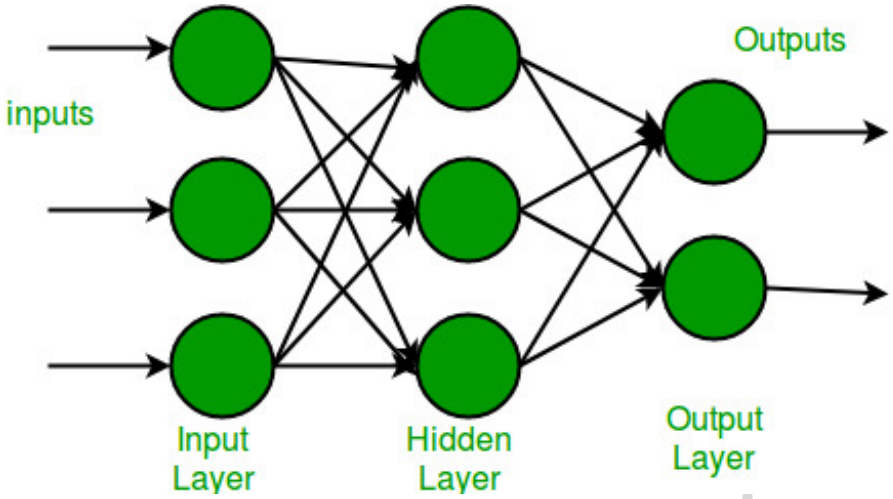


Figure 3: Multilayer Perceptron

6 Multilayer Perceptron (MLP)

6.1 The Solution: Add Hidden Layers

Stack multiple perceptrons in layers. Hidden layers between input and output transform the data into a linearly separable form.

$$\underbrace{\mathbf{x}}_{\text{Input Layer}} \longrightarrow \underbrace{\mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \ldots}_{\text{Hidden Layers}} \longrightarrow \underbrace{y}_{\text{Output Layer}} \tag{6}$$

6.2 Mathematical Notation

For each layer l ($l = 1, 2, \ldots, L$):

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)}\,\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)} \quad \text{(weighted sum)} \tag{7}$$

$$\mathbf{a}^{(l)} = f\left(\mathbf{z}^{(l)}\right) \quad \text{(apply activation function)} \tag{8}$$

where $\mathbf{a}^{(0)} = \mathbf{x}$ (input) and $\mathbf{a}^{(L)} = \hat{y}$ (predicted output).

6.3 Solving XOR with MLP

Network: 2 inputs → 2 hidden neurons → 1 output

Hidden layer learns:

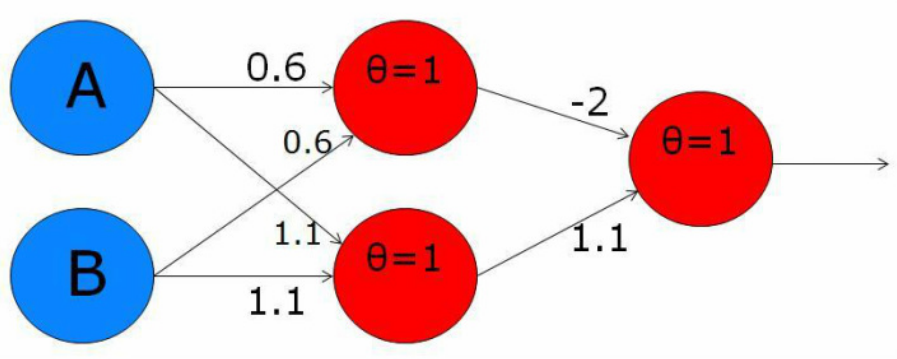
- Neuron 1: $h_1 \approx x_1$ OR x_2
- Neuron 2: $h_2 \approx \text{NOT}(x_1$ AND $x_2)$

Output layer combines: $y = h_1$ AND $h_2 = $ XOR!

x_1	x_2	h_1 (OR)	h_2 (NAND)	y (AND)	XOR
0	0	0	1	0	0
0	1	1	1	1	1
1	0	1	1	1	1
1	1	1	0	0	0

Table 4: MLP solving XOR via hidden layer features

Dept. of ECE, ACETVM



6.4 Universal Approximation Theorem

Theorem (Cybenko, 1989): An MLP with a *single* hidden layer containing a finite number of neurons can approximate *any* continuous function to arbitrary accuracy.

Implication: Neural networks are universal function approximators – given enough neurons, they can learn any pattern.

7 Gradient-Based Learning

7.1 What is Gradient Descent?

Goal: Find weights $\mathbf{w}$ that minimise the loss function $L(\mathbf{w})$.

Analogy: Imagine standing blindfolded on a hilly landscape and wanting to reach the lowest valley. You feel the slope under your feet and take a small step downhill. Repeat until you reach the bottom.

7.2 Loss Function

A loss function measures how wrong the network's predictions are. Common examples:

- **MSE** (regression): $L = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$
- **Binary Cross-Entropy** (classification): $L = -\frac{1}{N}\sum_{i}[t_i \log \hat{y}_i + (1-t_i)\log(1-\hat{y}_i)]$

7.3 The Update Rule

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla L(\mathbf{w}^{(t)}) \tag{9}$$

where:

- η – learning rate (step size)
- $\nabla L(\mathbf{w}) = \frac{\partial L}{\partial \mathbf{w}}$ – gradient (direction of steepest increase)

We move *opposite* to the gradient (downhill).

7.4 Variants of Gradient Descent

7.4.1 Batch Gradient Descent

Uses **all** N training examples per update:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla \left(\frac{1}{N}\sum_{i=1}^{N} L_i(\mathbf{w}) \right) \tag{10}$$

Pros: Stable, smooth convergence **Cons:** Very slow for large datasets

Dept. of ECE, ACETVM

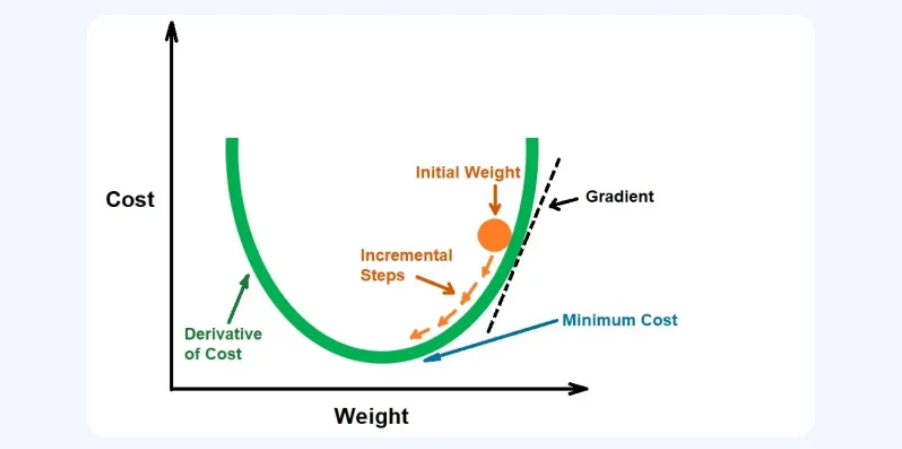


Figure 4: Gradient-Based Learning

7.4.2 Stochastic Gradient Descent (SGD)

Uses **one random** training example per update:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla L_i(\mathbf{w}) \tag{11}$$

Pros: Very fast, can escape local minima
Cons: Noisy, erratic convergence path
Why "stochastic"? Because the example is chosen randomly each time.

7.4.3 Mini-Batch Gradient Descent

Uses a **small batch** of m examples (typically 32–256):

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla \left(\frac{1}{m} \sum_{i \in \text{batch}} L_i(\mathbf{w}) \right) \tag{12}$$

Best of both worlds – fast and stable. This is the **standard approach** in modern deep learning.

7.5 Choosing the Learning Rate

Learning Rate	Effect
Too small ($\eta = 0.0001$)	Very slow convergence
Too large ($\eta = 10$)	Overshoots, diverges
Just right ($\eta = 0.01$)	Smooth, steady convergence

Table 5: Effect of learning rate on training

Typical values: 0.001, 0.01, 0.1

Aspect	Batch Gradient Descent	Stochastic Gradient Descent (SGD)	Mini-Batch Gradient Descent
Gradient Calculation	Computes gradients using the entire dataset.	Updates the gradient using a single data point at a time.	Calculates gradients using a small subset (batch) of the dataset.
Speed	Slow due to processing the entire dataset for each update.	Fast since it updates after each data point.	Faster than batch, slower than SGD.
Memory Requirement	Requires enough memory to hold the entire dataset.	Requires minimal memory as it processes one data at a time.	Moderate memory requirements, enough for a mini-batch.
Convergence Behaviour	Converges smoothly, guarantee to reach the global minimum for convex surfaces, but may get stuck in local minima for non-convex problems.	Noisy convergence with a zigzag path, but helps escape local minima due to randomness.	Balanced—less noise than SGD but still able to escape local minima better than batch.
Variance	Low variance since all data points are considered at each update.	High variance due to reliance on a single data point per update.	Moderate variance, strikes a balance between batch and stochastic approaches.
Practical Use	Rarely used for large datasets due to inefficiency.	Useful for quick updates and initial exploration, but unstable for large-scale training.	Commonly used as it balances efficiency and stability, making it suitable for most tasks.
Example	Processes all 60,000 MNIST images in one update.	Updates after each individual image.	Divides 60,000 MNIST images into mini-batches (e.g., 128 images per batch) and updates after each batch.

Dept. of ECE, ACETVM

8 Activation Functions

8.1 What is an Activation Function?

An **activation function** $f(z)$ is applied to the weighted sum $z = \mathbf{w}^T\mathbf{x} + b$ at each neuron to produce its output:

$$a = f(z) \tag{13}$$

Why is it needed? Without a non-linear activation function, stacking multiple layers reduces to a single linear transformation (no matter how many layers):

$$\mathbf{a}^{(2)} = \mathbf{W}^{(2)} \underbrace{(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})}_{\mathbf{a}^{(1)}} + \mathbf{b}^{(2)} = \underbrace{\mathbf{W}^{(2)}\mathbf{W}^{(1)}}_{\mathbf{W}_{\text{eff}}}\mathbf{x} + \mathbf{b}_{\text{eff}} \tag{14}$$

This is just one linear layer! Non-linear activations break this collapse, enabling networks to learn complex patterns.

8.2 Sigmoid Function

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{15}$$

What it does: Squashes any real number to the range $(0, 1)$, like a probability.
Shape: S-shaped (sigmoidal) curve; output is 0.5 when $z = 0$.
Derivative:

$$\sigma'(z) = \sigma(z)\left(1 - \sigma(z)\right) \tag{16}$$

Advantage	Disadvantage
Smooth, differentiable	Vanishing gradient for large $\lvert z\rvert$
Output interpretable as probability	Not zero-centred (always > 0)
Bounded output (0,1)	Slow convergence in deep networks

Vanishing gradient problem: For large $|z|$, $\sigma'(z) \approx 0$, so gradients shrink to nearly zero as they propagate backward through many layers, making learning very slow.
Use case: Output layer for binary classification.

8.3 Tanh Function

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{17}$$

What it does: Squashes input to the range $(-1, +1)$, centred at zero.
Relation to sigmoid:

$$\tanh(z) = 2\,\sigma(2z) - 1 \tag{18}$$

Derivative:

$$\tanh'(z) = 1 - \tanh^2(z) \tag{19}$$

Advantage	Disadvantage
Zero-centred output	Vanishing gradient for large $\lvert z\rvert$
Stronger gradient than sigmoid	Computationally expensive
Faster convergence than sigmoid	

Use case: Hidden layers in recurrent neural networks (RNNs).

Dept. of ECE, ACETVM

8.4 ReLU – Rectified Linear Unit

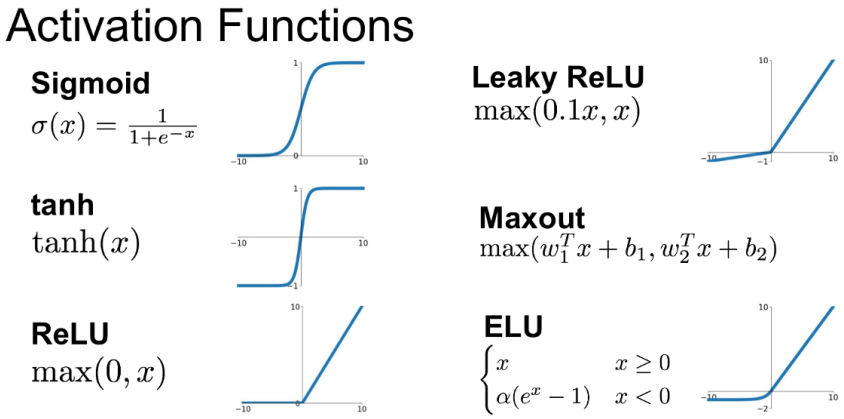
$$\text{ReLU}(z) = \max(0,\, z) = \begin{cases} z & \text{if } z > 0 \\ 0 & \text{if } z \leq 0 \end{cases} \tag{20}$$

What it does: Passes positive values unchanged; blocks negative values (outputs zero).
Derivative:

$$\text{ReLU}'(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{if } z \leq 0 \end{cases} \tag{21}$$

Advantage	Disadvantage
No vanishing gradient for $z > 0$	Dead ReLU (stuck at 0 forever)
Very fast to compute	Not zero-centred
Sparse activations (efficient)	Unbounded output
Default choice for deep nets	

Dead ReLU: If a neuron always receives negative input, it always outputs 0 and its gradient is 0 – it never learns again.
Use case: Default activation for hidden layers in deep networks and CNNs.



8.5 Comparison of Activation Functions

Function	Range	Best Feature	Main Problem	Use Case
Sigmoid	(0, 1)	Probability output	Vanishing gradient	Output (binary)
Tanh	$(-1, 1)$	Zero-centred	Vanishing gradient	RNNs
ReLU	$[0, \infty)$	Fast, no vanishing grad	Dead neurons	Deep networks

Table 6: Comparison of common activation functions

9 Backpropagation

9.1 What is Backpropagation?

Backpropagation is the algorithm used to train neural networks. It efficiently computes the gradient of the loss with respect to every weight in the network, so gradient descent can update them.

Analogy: A relay-race team loses the race. Backpropagation is like watching the replay and determining how much each runner contributed to the loss – so everyone knows exactly how to improve.

Key result: All gradients are computed in one forward pass plus one backward pass – very efficient!

9.2 Step 1 – Forward Pass

Compute activations layer by layer from input to output:

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)}\,\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)} \tag{22}$$

$$\mathbf{a}^{(l)} = f\Big(\mathbf{z}^{(l)}\Big) \tag{23}$$

with $\mathbf{a}^{(0)} = \mathbf{x}$ and $\mathbf{a}^{(L)} = \hat{y}$.
Then compute the loss $L(\hat{y},\, t)$.

9.3 Step 2 – Chain Rule

The **chain rule** is the mathematical backbone of backpropagation. For a composition $y = f(g(x))$:

$$\frac{dy}{dx} = \frac{dy}{dg} \cdot \frac{dg}{dx} \tag{24}$$

Extended chain rule for deep network $L \leftarrow a^{(L)} \leftarrow z^{(L)} \leftarrow \cdots \leftarrow w^{(l)}$:

$$\frac{\partial L}{\partial w^{(l)}} = \frac{\partial L}{\partial a^{(L)}} \cdot \frac{\partial a^{(L)}}{\partial z^{(L)}} \cdots \frac{\partial a^{(l)}}{\partial z^{(l)}} \cdot \frac{\partial z^{(l)}}{\partial w^{(l)}} \tag{25}$$

We multiply gradients layer by layer going *backward*.

9.4 Step 3 – Backward Pass

Define the **error (delta)** at layer l:

$$\boldsymbol{\delta}^{(l)} = \frac{\partial L}{\partial \mathbf{z}^{(l)}} \tag{26}$$

Output layer error:

$$\boldsymbol{\delta}^{(L)} = \nabla_{\mathbf{a}} L \;\odot\; f'\Big(\mathbf{z}^{(L)}\Big) \tag{27}$$

Hidden layer error (propagate backward):

$$\boldsymbol{\delta}^{(l)} = \Big(\mathbf{W}^{(l+1)}\Big)^{T} \boldsymbol{\delta}^{(l+1)} \;\odot\; f'\Big(\mathbf{z}^{(l)}\Big) \tag{28}$$

where $\odot$ denotes element-wise (Hadamard) multiplication.

9.5 Step 4 – Compute Gradients and Update Weights

$$\frac{\partial L}{\partial \mathbf{W}^{(l)}} = \boldsymbol{\delta}^{(l)} \;\Big(\mathbf{a}^{(l-1)}\Big)^{T} \tag{29}$$

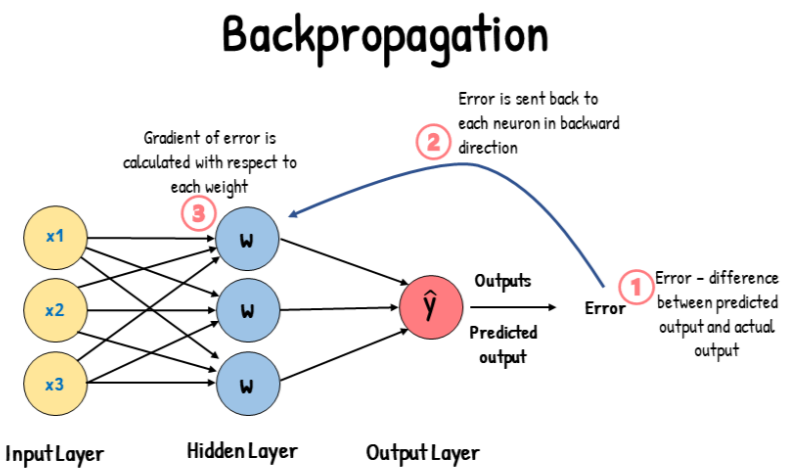
$$\frac{\partial L}{\partial \mathbf{b}^{(l)}} = \boldsymbol{\delta}^{(l)} \tag{30}$$

Weight update (gradient descent):

$$\mathbf{W}^{(l)} \leftarrow \mathbf{W}^{(l)} - \eta\, \frac{\partial L}{\partial \mathbf{W}^{(l)}} \tag{31}$$

$$\mathbf{b}^{(l)} \leftarrow \mathbf{b}^{(l)} - \eta\, \frac{\partial L}{\partial \mathbf{b}^{(l)}} \tag{32}$$

Dept. of ECE, ACETVM



9.6 Full Algorithm

```
FORWARD PASS:
    a(0) = x
    For l = 1 to L:
        z(l) = W(l) * a(l-1) + b(l)
        a(l) = f( z(l) )
    Compute loss L

BACKWARD PASS:
    delta(L) = grad_a(L) * f'( z(L) )     [output layer]
    For l = L-1 down to 1:
        delta(l) = W(l+1)^T * delta(l+1) * f'(z(l))

COMPUTE GRADIENTS:
    dW(l) = delta(l) * a(l-1)^T
    db(l) = delta(l)

UPDATE WEIGHTS:
    W(l) = W(l) - eta * dW(l)
    b(l) = b(l) - eta * db(l)
```

10 Regularization

10.1 The Overfitting Problem

Overfitting: The network memorises training data (including noise) rather than learning general patterns, so it performs poorly on unseen data.

Analogy: A student who memorises all past exam answers word for word. They ace the old papers but fail when new questions appear.

Signs of overfitting:

- Training accuracy: 99%, Test accuracy: 65%
- Large gap between training and validation loss

Regularization discourages the network from becoming too complex, improving generalisation.

10.2 L2 Regularization (Weight Decay / Ridge)

Idea: Add a penalty proportional to the *square* of each weight.

$$L_{\text{total}} = L_{\text{data}} + \frac{\lambda}{2} \sum_{l,i,j} \left(w_{ij}^{(l)} \right)^2 = L_{\text{data}} + \frac{\lambda}{2} \|\mathbf{w}\|_2^2 \tag{33}$$

Gradient with L2:

$$\frac{\partial L_{\text{total}}}{\partial w} = \frac{\partial L_{\text{data}}}{\partial w} + \lambda\, w \tag{34}$$

Weight update becomes:

$$w \leftarrow w - \eta \left(\frac{\partial L_{\text{data}}}{\partial w} + \lambda\, w \right) = (1 - \eta\lambda)\, w - \eta \frac{\partial L_{\text{data}}}{\partial w} \tag{35}$$

The factor $(1 - \eta\lambda) < 1$ causes weights to *decay* toward zero each step – hence the name **weight decay**.

Effect: All weights become small, producing smoother and simpler models.

Typical values: $\lambda = 0.0001,\ 0.001,\ 0.01$

10.3 L1 Regularization (Lasso)

Idea: Add a penalty proportional to the *absolute value* of each weight.

$$L_{\text{total}} = L_{\text{data}} + \lambda \sum_{l,i,j} \left| w_{ij}^{(l)} \right| = L_{\text{data}} + \lambda \|\mathbf{w}\|_1 \tag{36}$$

Gradient with L1:

$$\frac{\partial L_{\text{total}}}{\partial w} = \frac{\partial L_{\text{data}}}{\partial w} + \lambda\, \text{sign}(w) \tag{37}$$

Effect: Pushes many weights to *exactly zero*, creating a **sparse** model (automatic feature selection).

10.4 L1 vs. L2 – Key Differences

Aspect	L1 (Lasso)	L2 (Ridge)
Penalty term	$\lambda \sum \lvert w \rvert$	$\frac{\lambda}{2} \sum w^2$
Effect on weights	Many become exactly 0	All become small
Sparsity	Yes (sparse model)	No (dense model)
Feature selection	Yes (implicit)	No
Differentiability	Not at $w = 0$	Everywhere
When to use	Many irrelevant features	Default choice

Table 7: Comparison of L1 and L2 regularization

11 Quick Reference – Important Formulas

12 Summary

1. **Biological Neuron:** Inspiration for artificial networks – dendrites (inputs), soma (sum), axon (output).
2. **Perceptron:** Simplest artificial neuron; computes $z = \mathbf{w}^T\mathbf{x} + b$ and applies a step function.
3. **Training:** Perceptron learning rule adjusts weights on each error; converges if data is linearly separable.
4. **Limitations:** Cannot solve non-linearly separable problems (e.g. XOR).
5. **XOR Problem:** No straight line can separate XOR classes – mathematically provable.
6. **MLP:** Hidden layers transform data into linearly separable space; universal approximator.

Dept. of ECE, ACETVM

Concept	Formula
Perceptron output	$y = \text{sign}\left(\sum w_i x_i + b\right)$
Perceptron update	$w_i \leftarrow w_i + \eta(t - y)x_i$
Gradient descent	$w \leftarrow w - \eta \dfrac{\partial L}{\partial w}$
Sigmoid	$\sigma(z) = \dfrac{1}{1+e^{-z}}, \quad \sigma'(z) = \sigma(z)(1-\sigma(z))$
Tanh	$\tanh(z) = \dfrac{e^z - e^{-z}}{e^z + e^{-z}}, \quad \tanh'(z) = 1 - \tanh^2(z)$
ReLU	$\text{ReLU}(z) = \max(0, z), \quad \text{ReLU}'(z) = \mathbf{1}[z > 0]$
Backprop (output)	$\boldsymbol{\delta}^{(L)} = \nabla_a L \odot f'(\mathbf{z}^{(L)})$
Backprop (hidden)	$\boldsymbol{\delta}^{(l)} = (\mathbf{W}^{(l+1)})^T \boldsymbol{\delta}^{(l+1)} \odot f'(\mathbf{z}^{(l)})$
L1 regularization	$L = L_{\text{data}} + \lambda \sum \lvert w \rvert$
L2 regularization	$L = L_{\text{data}} + \dfrac{\lambda}{2} \sum w^2$

Table 8: Summary of key formulas

7. **Gradient Descent:** Iteratively minimise loss by moving weights opposite to the gradient.
8. **SGD:** Uses one random example per step – faster, can escape local minima.
9. **Activation Functions:** Sigmoid (probability output), Tanh (zero-centred), ReLU (default, fast, no vanishing gradient).
10. **Backpropagation:** Efficient gradient computation using the chain rule in one forward + one backward pass.
11. **L1 Regularization:** Penalises $|w|$; creates sparse models; useful for feature selection.
12. **L2 Regularization:** Penalises w^2; shrinks all weights; default regularization choice.

Dept. of ECE, ACETVM

PEECT413 Machine Learning

Module 1 - Tutorial 5

S4 ECE, 2024 Scheme KTU

Module 1

1 Question 9

9(a)

A model is used to detect spam emails. Out of 150 emails predicted as spam, 125 were actually spam. The model missed 5 actual spam emails, classifying them as not spam. Use this information to:

1. Construct the values for TP and FP.
2. Calculate the Precision of the model.

Solution

Step 1: Identify TP and FP

$$\text{Total predicted as Spam} = 150$$

$$\text{Actually Spam (correctly predicted)} = \text{TP} = 125 \tag{1}$$

$$\text{Not Spam (wrongly predicted as Spam)} = \text{FP} = 150 - 125 = 25 \tag{2}$$

$$\text{Missed actual Spam (False Negatives)} = \text{FN} = 5 \tag{3}$$

Confusion Matrix:

	Predicted: Spam	**Predicted: Not Spam**
Actual: Spam	TP = 125	FN = 5
Actual: Not Spam	FP = 25	TN = –

Table 1: Confusion Matrix for Spam Detection

Step 2: Precision Calculation

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{125}{125 + 25} = \frac{125}{150} = \mathbf{0.8333} \tag{4}$$

Conclusion: Model correctly identifies 83.33% of spam predictions.

9(b)

Distinguish between Instance-based learning and Model-based learning.

Solution

2 Question 10

10(a)

A binary classification model (predicting Positive = 1) is run on 20 samples, 10 of which are truly Positive (P=10) and 10 are truly Negative (N=10). The predicted scores (S) and true labels (L) are:

Using a decision threshold of $T = 0.50$, classify all samples and calculate the resulting point on the ROC curve.

pt. of ECE, ACET

Aspect	Instance-Based Learning	Model-Based Learning
Definition	Memorizes training data; predicts by comparing to stored instances	Builds mathematical model from training data
Learning	Lazy learner (deferred)	Eager learner (immediate)
Training Time	Fast	Slow
Prediction Time	Slow (compares all instances)	Fast
Memory	High (stores all data)	Low (stores model only)
Example	k-NN, Case-Based Reasoning	Linear Regression, SVM, Neural Networks

Table 2: Instance-Based vs Model-Based Learning

S	0.95	0.88	0.70	0.65	0.55	0.51	0.49	0.45	0.44	0.42	0.40	0.35	0.25	0.20	0.15	0.10	0.05	0.02	0.01	0.00
L	1	1	0	1	1	0	1	1	1	1	0	1	0	0	1	0	0	1	0	0

Table 3: Predicted Scores and True Labels

Solution

Step 1: Apply Threshold $T = 0.50$
Classify as Positive if $S \geq 0.50$:

$$\text{Predicted Positive} : S \in \{0.95, 0.88, 0.70, 0.65, 0.55, 0.51\} \quad (6 \text{ samples})$$
$$\text{Predicted Negative} : S < 0.50 \quad (14 \text{ samples})$$

Step 2: Confusion Matrix at T = 0.50

$$\text{TP} = \text{Predicted Positive AND Actual} = 1 = 4 \tag{5}$$
$$\text{FP} = \text{Predicted Positive AND Actual} = 0 = 2 \tag{6}$$
$$\text{TN} = \text{Predicted Negative AND Actual} = 0 = 7 \tag{7}$$
$$\text{FN} = \text{Predicted Negative AND Actual} = 1 = 7 \tag{8}$$

	Predicted: P	**Predicted: N**
Actual: P	TP = 4	FN = 7
Actual: N	FP = 2	TN = 7

Table 4: Confusion Matrix at T=0.50

Step 3: ROC Point Calculation

$$\text{TPR (Sensitivity)} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{4}{4+7} = \frac{4}{11} = \mathbf{0.3636} \tag{9}$$
$$\text{FPR (1-Specificity)} = \frac{\text{FP}}{\text{FP} + \text{TN}} = \frac{2}{2+7} = \frac{2}{9} = \mathbf{0.2222} \tag{10}$$

$$\boxed{\text{ROC Point} = (\text{FPR}, \text{TPR}) = (0.2222,\ 0.3636)} \tag{11}$$

10(b)

Distinguish between Batch Learning and Online Learning.

Solution

Dept. of ECE, ACETVM

Aspect	**Batch Learning**	**Online Learning**
Definition	Trains on entire dataset at once	Updates model incrementally
Data	Full dataset required	One sample or mini-batch
Training	Periodic, offline	Continuous, real-time
Adaptability	Low	High
Memory	High	Low
Example	Random Forest, SVM	SGD, Perceptron

Table 5: Batch vs Online Learning

Dept. of ECE, ACETVM

PEECT413 Machine Learning

Module 2- Tutorial 4

S4 ECE, 2024 Scheme KTU

Module 2

1 Question 11

11(a)

Suppose you are training a linear regression model on dataset with 3 samples. The actual values are $\mathbf{y} = [10, 20, 30]$ and the predicted values are $\hat{\mathbf{y}} = [12, 18, 33]$. Calculate the Mean Squared Error (MSE) and determine if the regularization term, L1(w) or L2(w), would penalize a weight of $w = 5$ heavily if the regularization parameter $\lambda = 0.1$ is applied?

11(b)(i)

In a binary logistic regression task, a model predicts a probability of 0.8 for a sample that actually belongs to the positive class ($y = 1$). Calculate the Cross-Entropy loss for this specific sample (use natural logarithm).

11(b)(ii)

P(Spam) = 0.4, P(Free|Spam) = 0.7, and P(Free|not Spam) = 0.1. If an email contains the word "Free", what is the posterior probability P(Spam|Free)?

2 Question 12

12(a)

What is Bootstrap Aggregating (Bagging)?

12(b)

In a binary logistic regression task, a model predicts a probability of 0.8 for a sample that actually belongs to the positive class ($y = 1$). Calculate the Cross-Entropy loss for this specific sample (use natural logarithm).

3 Solutions

11(a): MSE and L1/L2 Regularization Analysis

MSE Calculation

$$e = [10 - 12, 20 - 18, 30 - 33] = [-2, 2, -3] \tag{1}$$

$$e^2 = [4, 4, 9], \quad \text{SSE} = 17 \tag{2}$$

$$\text{MSE} = \frac{1}{2N} \times \text{SSE} = \frac{17}{6} = 2.8333 \tag{3}$$

L1 Regularization (Lasso)

$$L_1(w) = \text{MSE} + \lambda|w| = 2.8333 + 0.1 \times 5 = 3.3333 \tag{4}$$

Dept. of ECE, ACET

L2 Regularization (Ridge)

$$L_2(w) = \text{MSE} + \frac{\lambda}{2}w^2 = 2.8333 + 0.05 \times 25 = 4.0833 \tag{5}$$

Penalty Analysis

Metric	Value	Penalty	% of MSE
MSE	2.8333	0	0%
L1 Loss	**3.3333**	0.50	18%
L2 Loss	**4.0833**	1.25	44%

Conclusion: *Neither L1 nor L2 applies heavy penalty*

11(b)(i): Binary Logistic Regression

Model Definition

$$P(y = 1|\mathbf{x}) = \sigma(z) = \frac{1}{1 + e^{-z}}, \quad z = \mathbf{w}^\top \mathbf{x} + b \tag{6}$$

Cross-Entropy Loss

$$L_{\text{CE}} = -[y \ln p + (1 - y) \ln(1 - p)] \tag{7}$$

For $y = 1$, $p = 0.8$:

$$L_{\text{CE}} = -\ln(0.8) = 0.2231 \tag{8}$$

11(b)(ii): Naive Bayes Posterior Probability

Complete Probabilities

$$P(\neg Spam) = 1 - 0.4 = 0.6 \tag{9}$$

$$\begin{aligned} P(Free) &= P(Free|Spam)P(Spam) + P(Free|\neg Spam)P(\neg Spam) \\ &= (0.7)(0.4) + (0.1)(0.6) = 0.28 + 0.06 \\ &= \mathbf{0.34} \end{aligned} \tag{10}$$

Bayes' Theorem

$$P(Spam|Free) = \frac{P(Free|Spam) \cdot P(Spam)}{P(Free)} = \frac{0.28}{0.34} = 0.8235 \tag{11}$$

12(a): Bootstrap Aggregating (Bagging)

Algorithm

1. Generate B bootstrap samples $D_b \sim D$ (with replacement)
2. Train base model f_b on D_b for $b = 1, \ldots, B$
3. Aggregate predictions:

$$\hat{f}(\mathbf{x}) = \frac{1}{B} \sum_{b=1}^{B} f_b(\mathbf{x}) \tag{12}$$

Purpose: *Variance reduction through ensemble averaging*

12(b): Cross-Entropy Loss

Same as 11(b)(i):

$$L_{\text{CE}} = -\ln(0.8) = 0.2231 \tag{13}$$

Dept. of ECE, ACETVM

PEECT413 Machine Learning

Module 3- Tutorial3

S4 ECE, 2024 Scheme KTU

Module 3

1 Problem Statement

2 Question 13(a): Markov Property in State Transitions

2.1 Theoretical Foundation

The **Markov Property** (memorylessness) is a fundamental assumption in stochastic processes:

$$P(S_{t+1} = j \mid S_t = i, S_{t-1}, \ldots, S_0) = P(S_{t+1} = j \mid S_t = i) \tag{1}$$

Key Implications:

- Future state probability depends *only* on current state
- Past history is irrelevant given present
- Enables compact $n \times n$ transition matrix $\mathbf{\Pi}$

2.2 Transition Matrix Representation

For n states, $\mathbf{\Pi}_{ij} = P(S_{t+1} = j \mid S_t = i)$ where:

- Each row sums to 1: $\sum_j \mathbf{\Pi}_{ij} = 1$
- k-step prediction: $\mathbf{\Pi}^k$

2.3 Applications & Examples

Domain	Application
Weather	P(Rain tomorrow | Today)
PageRank	Web surfing model
MDPs	Policy evaluation $\pi(s) = \sum_{s'} P(s'\|s,a)\pi(s')$
HMMs	Hidden state transitions

Table 1: Markov Property Applications

3 Question 13(b): Minkowski Distance Calculation

3.1 Definition

Minkowski distance generalizes L_p norms:

$$d_p(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{d} |x_i - y_i|^p \right)^{1/p} \tag{2}$$

Special Cases: $p = 1$ Manhattan, $p = 2$ Euclidean, $p \to \infty$ Chebyshev

Dept of ECE, ACET

3.2 Numerical Computation: $p = 3$

$\mathbf{P_1} = (2, 3)$, $\mathbf{P_2} = (5, 7)$:

$$\begin{aligned} d_3 &= \left(|2-5|^3 + |3-7|^3\right)^{1/3} \\ &= \left(3^3 + 4^3\right)^{1/3} \\ &= (27 + 64)^{1/3} \\ &= 91^{1/3} \approx 4.4979 \end{aligned} \tag{3}$$

4 Question 13(c): Cosine Similarity

4.1 Mathematical Definition

$$\text{cosine similarity} = \cos\theta = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} \tag{4}$$

Range: $[-1, 1]$ (± 1 parallel, 0 orthogonal)

4.2 Computation

$\mathbf{A} = [1, 0]$, $\mathbf{B} = [0, 1]$:

$$\begin{aligned} \mathbf{A} \cdot \mathbf{B} &= 1 \cdot 0 + 0 \cdot 1 = 0 \\ \|\mathbf{A}\| &= \sqrt{1^2 + 0^2} = 1 \\ \|\mathbf{B}\| &= \sqrt{0^2 + 1^2} = 1 \\ \cos\theta &= \frac{0}{1 \cdot 1} = 0 \end{aligned} \tag{5}$$

5 Question 14(a): Complete Linkage Clustering

5.1 Agglomerative Hierarchical Clustering

Complete linkage: distance between clusters = **maximum** inter-point distance

5.2 Cluster Definitions

Cluster A: $\{(1, 1), (1, 2)\}$, Cluster B: $\{(4, 4), (5, 4)\}$

5.3 Euclidean Distance Matrix

All pairwise distances:

	$(1,1)$	$(1,2)$	$(4,4)$	$(5,4)$
$(1,1)$	0	1	4.24	5
$(1,2)$	1	0	3.61	5
$(4,4)$	4.24	3.61	0	1
$(5,4)$	5	5	1	0

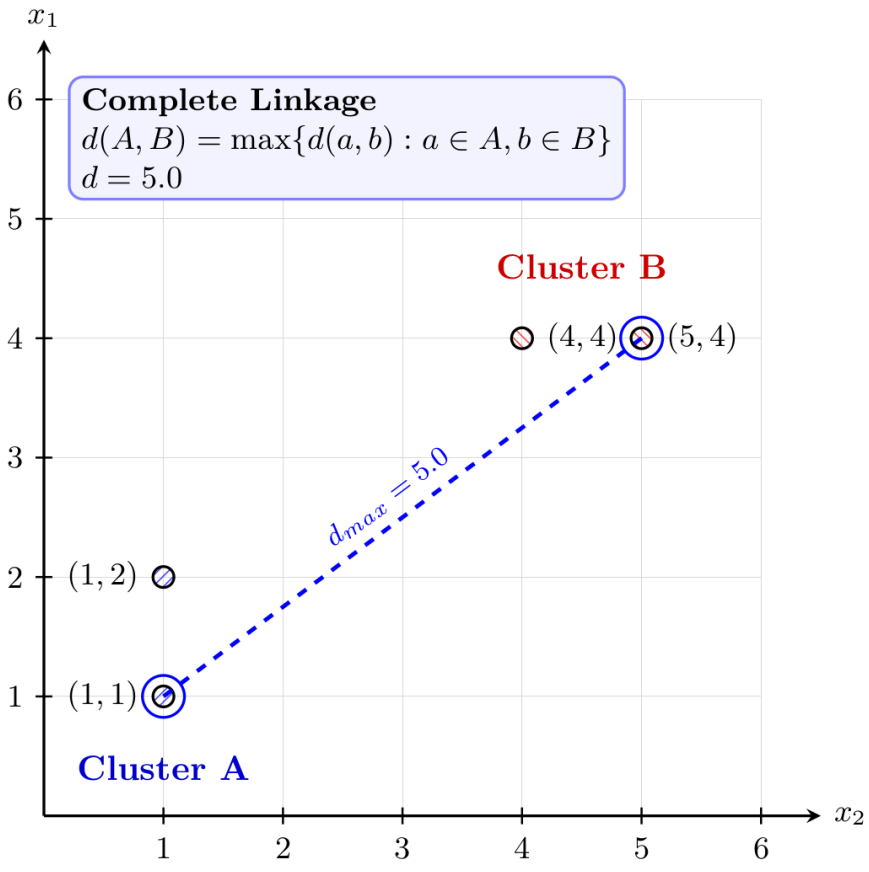
Complete linkage distance = max $= 5.0$

6 Question 14(b): K-Means Clustering Assignment

6.1 Manhattan Distance

L_1 norm: $d_1(\mathbf{x}, \mathbf{c}) = \sum_i |x_i - c_i|$

Dept. of ECE, ACETVM



6.2 Assignment Computation

$\mathbf{X} = (10, 12)$, $\mathbf{C_1} = (9, 8)$, $\mathbf{C_2} = (15, 15)$:

$$d(X, C_1) = |10 - 9| + |12 - 8| = 1 + 4 = 5$$
$$d(X, C_2) = |10 - 15| + |12 - 15| = 5 + 3 = 8$$

Assignment Rule: $\arg\min_c d(X, c) = C_1$ (distance $5 < 8$)

PEECT413 Machine Learning

Module 4- Tutorial 1

S4 ECE, 2024 Scheme KTU

Module 4

1 Problem Statement

Part (1a): Single-Layer Perceptron Forward Pass

A single-layer perceptron with weights $w_1 = 0.4$, $w_2 = -0.6$, bias $b = 0.5$, and inputs $x_1 = 0$, $x_2 = 0$. Calculate output y using step activation where $y = 1$ if $z \geq 0$ else $y = 0$.

Part (1b): Hidden Layer Neuron

Given hidden layer net input $x = 0$, calculate output using step activation.

Part (1c): Gradient Descent Update

Current weight $w = -2.0$, learning rate $\eta = 0.1$, gradient $\frac{\partial L}{\partial w} = -0.1$, loss $L = 0.5$.

Part (1d): L2 Regularized Update

Apply L2 regularization with parameter $\lambda = 0.1$ to find $w = 10$? (appears to be $w = -2.0$ from context).

2 Solutions

(1a): Perceptron Forward Propagation

The perceptron computes linear combination:

$$z = w_1 x_1 + w_2 x_2 + b \tag{1}$$

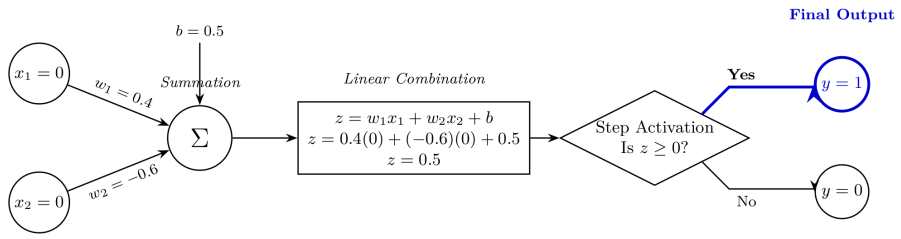
Substitute given values:

$$\begin{aligned} z &= 0.4 \cdot 0 + (-0.6) \cdot 0 + 0.5 && (2)\\ &= 0 + 0 + 0.5 && (3)\\ &= 0.5 && (4) \end{aligned}$$

Step activation function:

$$y = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases} \tag{5}$$

Since $z = 0.5 \geq 0$, we get $y = 1$.

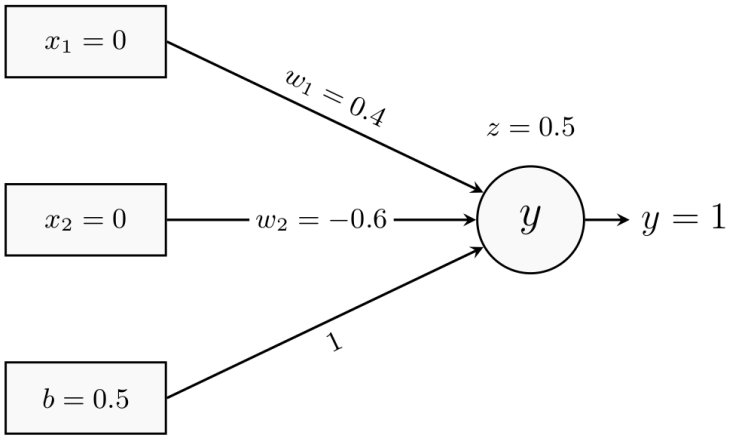


Dept. of ECE, ACET

(1b): Hidden Layer Neuron

Given net input $x = 0$ to hidden neuron. Using step activation:

$$h = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} = 1 \tag{6}$$



Single-Layer Perceptron
$z = \sum w_i x_i + b$

(1c) Backpropagation and Weight Updates - Standard Gradient Descent

Weight update rule:

$$w_{\text{new}} = w - \eta \frac{\partial L}{\partial w} \tag{7}$$

Given: $w = -2.0$, $\eta = 0.1$, $\frac{\partial L}{\partial w} = -0.1$

$$w_{\text{new}} = -2.0 - 0.1 \cdot (-0.1) \tag{8}$$
$$= -2.0 + 0.01 \tag{9}$$
$$= -1.99 \tag{10}$$

(1d): L2 Regularized Gradient Descent

L2 regularization adds penalty term to loss function:

$$J(w) = L(w) + \frac{\lambda}{2}\|w\|^2 \tag{11}$$

Gradient of regularized loss:

$$\frac{\partial J}{\partial w} = \frac{\partial L}{\partial w} + \lambda w \tag{12}$$

Substitute values $\lambda = 0.1$:

$$\frac{\partial J}{\partial w} = -0.1 + 0.1 \cdot (-2.0) \tag{13}$$
$$= -0.1 - 0.2 \tag{14}$$
$$= -0.3 \tag{15}$$

Dept of ECE, ACETVM

Weight update:

$$w_{\text{new}} = w - \eta \frac{\partial J}{\partial w} \tag{16}$$
$$= -2.0 - 0.1 \cdot (-0.3) \tag{17}$$
$$= -2.0 + 0.03 \tag{18}$$
$$= -1.97 \tag{19}$$

L2 Regularization Effect Visualization

L2 penalty shrinks weights toward zero, preventing overfitting:

$$\text{Penalty term: } \frac{\lambda}{2} w^2 = \frac{0.1}{2} \cdot (-2.0)^2 = 0.2 \tag{20}$$

Dept. of ECE, ACETVM

PEECT413 Machine Learning

Module 4- Tutorial 1

S4 ECE, 2024 Scheme KTU

Module 4

1 Problem Statement

Part (1a): Single-Layer Perceptron Forward Pass

A single-layer perceptron with weights $w_1 = 0.4$, $w_2 = -0.6$, bias $b = 0.5$, and inputs $x_1 = 0$, $x_2 = 0$. Calculate output y using step activation where $y = 1$ if $z \geq 0$ else $y = 0$.

Part (1b): Hidden Layer Neuron

Given hidden layer net input $x = 0$, calculate output using step activation.

Part (1c): Gradient Descent Update

Current weight $w = -2.0$, learning rate $\eta = 0.1$, gradient $\frac{\partial L}{\partial w} = -0.1$, loss $L = 0.5$.

Part (1d): L2 Regularized Update

Apply L2 regularization with parameter $\lambda = 0.1$ to find $w = 10$? (appears to be $w = -2.0$ from context).

2 Solutions

(1a): Perceptron Forward Propagation

The perceptron computes linear combination:

$$z = w_1 x_1 + w_2 x_2 + b \tag{1}$$

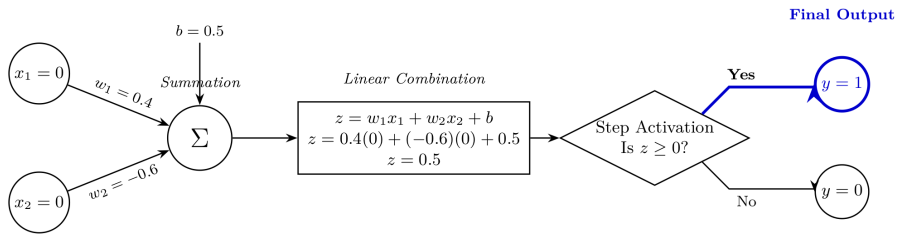
Substitute given values:

$$\begin{aligned} z &= 0.4 \cdot 0 + (-0.6) \cdot 0 + 0.5 && (2)\\ &= 0 + 0 + 0.5 && (3)\\ &= 0.5 && (4) \end{aligned}$$

Step activation function:

$$y = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases} \tag{5}$$

Since $z = 0.5 \geq 0$, we get $y = 1$.

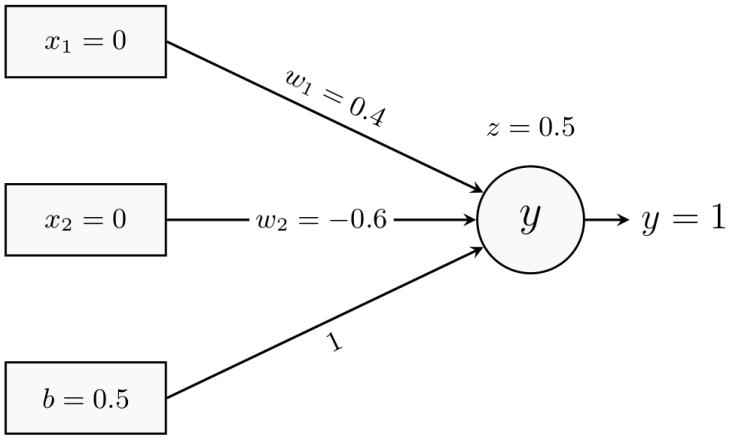


Dept. of ECE, ACET

(1b): Hidden Layer Neuron

Given net input $x = 0$ to hidden neuron. Using step activation:

$$h = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} = 1 \tag{6}$$



Single-Layer Perceptron
$z = \sum w_i x_i + b$

(1c) Backpropagation and Weight Updates - Standard Gradient Descent

Weight update rule:

$$w_{\text{new}} = w - \eta \frac{\partial L}{\partial w} \tag{7}$$

Given: $w = -2.0$, $\eta = 0.1$, $\frac{\partial L}{\partial w} = -0.1$

$$w_{\text{new}} = -2.0 - 0.1 \cdot (-0.1) \tag{8}$$
$$= -2.0 + 0.01 \tag{9}$$
$$= -1.99 \tag{10}$$

(1d): L2 Regularized Gradient Descent

L2 regularization adds penalty term to loss function:

$$J(w) = L(w) + \frac{\lambda}{2}\|w\|^2 \tag{11}$$

Gradient of regularized loss:

$$\frac{\partial J}{\partial w} = \frac{\partial L}{\partial w} + \lambda w \tag{12}$$

Substitute values $\lambda = 0.1$:

$$\frac{\partial J}{\partial w} = -0.1 + 0.1 \cdot (-2.0) \tag{13}$$
$$= -0.1 - 0.2 \tag{14}$$
$$= -0.3 \tag{15}$$

Dept of ECE, ACETVM

Weight update:

$$w_{\text{new}} = w - \eta \frac{\partial J}{\partial w} \tag{16}$$
$$= -2.0 - 0.1 \cdot (-0.3) \tag{17}$$
$$= -2.0 + 0.03 \tag{18}$$
$$= -1.97 \tag{19}$$

L2 Regularization Effect Visualization

L2 penalty shrinks weights toward zero, preventing overfitting:

$$\text{Penalty term: } \frac{\lambda}{2} w^2 = \frac{0.1}{2} \cdot (-2.0)^2 = 0.2 \tag{20}$$

Dept. of ECE, ACETVM

PEECT413 Machine Learning

Module 4- Tutorial2

S4 ECE, 2024 Scheme KTU

Module 4

1 Problem Statement

1.1 Part (a): Chain Rule Backpropagation

Using chain rule for backpropagation, if loss L depends on output y as $L = 0.5(y - \hat{y})^2$ and $y = \sigma(z)$ (step function), find expression for $\frac{dL}{dz}$.

1.2 Part (b): L2 Regularization

Apply L2 regularization to weight $w = 10$ with regularization parameter $\lambda = 0.1$. What penalty term is added to cost function J?

2 Detailed Mathematical Derivation

2.1 Part (a): Chain Rule Application

2.1.1 Loss Function

Mean Squared Error loss:

$$L = \frac{1}{2}(y - \hat{y})^2 \tag{1}$$

2.1.2 Step Activation Function

$$y = \sigma(z) = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases} \tag{2}$$

The derivative of step function is Dirac delta:

$$\frac{d\sigma(z)}{dz} = \delta(z) \tag{3}$$

2.1.3 Chain Rule Backpropagation

$$\begin{aligned} \frac{dL}{dz} &= \frac{dL}{dy} \cdot \frac{dy}{dz} \\ &= \frac{\partial L}{\partial y} \cdot \frac{d\sigma(z)}{dz} \\ &= -(y - \hat{y}) \cdot \delta(z) \end{aligned} \tag{4}$$

Key Insight: Step function is non-differentiable everywhere except $z = 0$. In practice: - $\delta(z) \approx 1$ at training point if $z \approx 0$ - $\delta(z) = 0$ otherwise (no gradient flow)

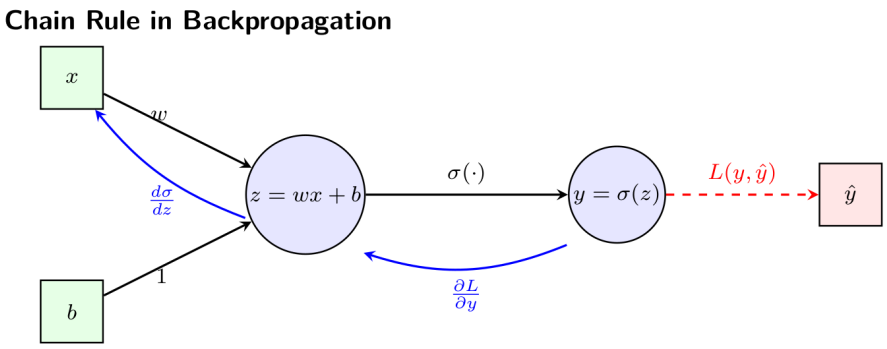
2.2 Part (b): L2 Regularization

2.2.1 Regularized Cost Function

$$J(w) = L(w) + \frac{\lambda}{2}\|w\|^2 = L + \frac{\lambda}{2}w^2 \tag{5}$$

Given: $w = 10$, $\lambda = 0.1$

Dept. of ECE, ACET



$$\frac{dL}{dz} = \frac{dL}{dy} \cdot \frac{d\sigma}{dz}$$

2.2.2 Penalty Term Calculation

$$\begin{aligned} \text{L2 Penalty} &= \frac{\lambda}{2} w^2 \\ &= \frac{0.1}{2} \times 10^2 \\ &= 0.05 \times 100 \\ &= 5 \end{aligned} \tag{6}$$

Thus: $J = L + 5$

2.2.3 Regularized Gradient

$$\frac{\partial J}{\partial w} = \frac{\partial L}{\partial w} + \lambda w \tag{7}$$

Weight decay term: $\lambda w = 0.1 \times 10 = 1$

Dept. of ECE, ACETVM

MODEL QUESTION PAPER

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY
FOURTH SEMESTER B. TECH DEGREE EXAMINATION, MONTH AND YEAR

Course Code: PEECT413

Course Name: MACHINE LEARNING

Max. Marks: 60 | Duration: 2 hours 30 minutes

PART A

		Answer all questions. Each question carries 3 marks	CO	Marks
1		Explain the term bias.	1,2	(3)
2		Describe when overfitting happens.	1,2	(3)
3		Define "Support Vectors" in the context of Support Vector Machines (SVM).	1,2	(3)
4		Identify and define the three types of nodes found in a standard Decision Tree.	3,4	(3)
5		Distinguish between the "Exploration" and "Exploitation" strategies in Reinforcement Learning.	4	(3)
6		Define a Hidden Markov Model (HMM).	4	(3)
7		State the primary limitation of a single-layer Perceptron and the problem that illustrated it.	1,2	(3)
8		What is Stochastic Gradient Descent (SGD) and how does it differ from Batch Gradient Descent?	1,2	(3)

PART B

Answer any one full question from each module. Each question carries 9 marks

Module 1

9	a)	A model is used to detect spam emails. Out of 150 emails predicted as spam, 125 were actually spam. The model missed 5 actual spam emails, classifying them as not spam. Use this information to: 1. Construct the values for TP and FP. 2. Calculate the Precision of the model.	1,2	(5)
	b)	Distinguish between Instance-based learning and Model-based learning.	1	(4)
10	a)	A binary classification model (predicting Positive = 1) is run on 20 samples, 10 of which are truly Positive (P=10) and 10 are truly Negative (N=10). The predicted scores (S) and true labels (L) are:	1	(6)

S	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.

			95	88	70	65	55	51	49	45	44	42	40	35	25	20	15	10	05	02	01	00
		L	1	1	0	1	1	0	1	1	1	1	0	1	0	0	1	0	0	1	0	0
		Using a decision threshold of T = 0.50, classify all samples and calculate the resulting point on the ROC curve.																				
	b)	Distinguish between batch learning and online learning.																				

		Module 2		
11	a)	Suppose you are training a linear regression model on a dataset with 3 samples. The actual values are y = [10, 20, 30] and the predicted values are y = [12, 18, 33]. Calculate the Mean Squared Error (MSE) and determine which regularization term, L1 (|w|) or L2 (w^2), would penalize a weight of w=5 more heavily if the regularization parameter $\lambda = 0.1$ is applied?	1,2	(7)
	b)	What is Bootstrap Aggregating (Bagging)?	2	(2)
12	a)	In a binary logistic regression task, a model predicts a probability of 0.8 for a sample that actually belongs to the positive class (y=1). Calculate the Cross-Entropy loss for this specific sample (use natural logarithm).	1,2	(3)
	b)	A Naive Bayes classifier is used to predict if an email is 'Spam'. Given: P(Spam) = 0.4, P('Free'}|Spam) = 0.7, and P('Free'|Not Spam) = 0.1. If an email contains the word 'Free', what is the posterior probability P(Spam|'Free')?	1,2	(6)
		Module 3		
13	a)	Explain the "Markov Property" in the context of state transitions.	4	(3)
	b)	Given two points $P_1 = (2, 3)$ and $P_2 = (5, 7)$, calculate the Minkowski distance between them with the parameter p = 3.	3	(3)
	c)	Calculate the Cosine Similarity between vector A = [1, 1, 0] and vector B = [0, 1, 1].	3	(3)
14	a)	Using Agglomerative Hierarchical Clustering with 'Complete Linkage', if Cluster A contains points {(1,1), (1,2)} and Cluster B contains points {(4,4), (5,4)}, what is the distance between these two clusters using Euclidean distance?	3	(5)
	b)	In a K-means clustering task, you have a data point X = (10, 12) and two cluster centroids $C_1 = (8, 9)$ and $C_2 = (15, 15)$. Using the Manhattan distance, which cluster should point X be assigned to?	3	(4)

Module 4				
15	a)	A single-layer Perceptron has weights $w_1 = 0.4$, $w_2 = -0.6$ and a bias $b = 0.5$. If the input vector is $X = [1, 2]$, calculate the net input (logit) z and the final output y using a Step Activation Function (where $y=1$ if $z \geq 0$, else 0).	1,2	(4)
	b)	Given a hidden layer neuron with a net input $z = 0$, calculate the output of the Sigmoid, Tanh, and ReLU activation functions respectively.	1,2	(3)
	c)	In a Gradient Descent update, the current weight is $w=2.0$, the learning rate $\eta=0.1$, and the gradient of the loss function at this point is $\partial w/\partial L=5.0$. What is the updated weight w_{new}?	1,2	(2)
16	a)	Using the Chain Rule for Backpropagation, if the loss L depends on output y as $L = 0.5(y-t)^2$, and $y = \sigma(z)$, find the expression for $\partial L/\partial z$.	1,2	(6)
	b)	Apply L2 Regularization to a weight $w = 10$. If the regularization parameter $\lambda = 0.1$, what is the penalty term added to the cost function J?	1,2	(3)
*****				

APJ Abdul Kalam Technological University

B.Tech ECE IV Semester Examination

PEECT413: MACHINE LEARNING

Model Question Paper - Complete Solutions with Explanations

Max Marks: 60 — Duration: 2.5 Hours — March 2026

---

1 PART A: All Questions (3 marks each = 24 marks)

Answer Strategy: Definition (1 mark) + Context/Example (1 mark) + Formula/Fix (1 mark)

Q1. Bias
Definition: Error from overly simplistic model assumptions.
Context: Causes underfitting (systematic errors).
Formula: Bias-variance decomposition: Error = Bias^2 + Variance + Noise.

Q2. Overfitting
When: Model complexity > training data capacity.
Signs: Low train error, high validation error.
Prevention: L2 regularization ($\lambda \sum w_i^2$), dropout, early stopping.

Q3. Support Vectors (SVM)
Definition: Training points nearest to hyperplane ($y_i(w^T x_i + b) = 1$).
Role: Define maximum-margin hyperplane.
SVM: $\min \frac{1}{2}\|w\|^2 + C \sum \xi_i$.

Q4. Decision Tree Nodes
Root Node: Initial best split (Gini/entropy).
Internal Node: Feature test with branches.
Leaf Node: Final class/regression value.

Q5. Exploration vs Exploitation (RL)
Exploration: Sample new actions (ϵ-greedy policy).
Exploitation: Choose highest $Q(s, a)$ value.
Tradeoff solved by Upper Confidence Bound (UCB).

Q6. Hidden Markov Model
Components: Initial π, transition A, emission B.
$P(O, S|\lambda) = \pi_{s_1} b_{s_1}(o_1) \prod a_{s_t, s_{t+1}} b_{s_{t+1}}(o_{t+1})$.

Q7. Single-layer Perceptron limitation
Cannot solve XOR (linearly inseparable).
XOR table shows no single hyperplane separates classes.

Q8. SGD vs Batch GD
SGD: $\theta \leftarrow \theta - \eta \nabla J(x_i, y_i)$ (single sample).

Batch: $\theta \leftarrow \theta - \frac{\eta}{N} \sum \nabla J(x_i, y_i)$ (full dataset).

2 PART B: One Full Question per Module (9 marks each = 36 marks)

Module 1: Classification Metrics

Q9(a): Confusion Matrix (5+4 marks)

$$\begin{aligned} \text{Predicted Spam} &= 150 \\ \text{TP (actual spam)} &= 125 \\ \text{FP} &= 150 - 125 = \mathbf{25} \\ \text{FN (missed spam)} &= 5 \\ \text{Precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{125}{150} = \mathbf{0.8333} \end{aligned}$$

Q9(b): Learning Types Instance-based: Lazy (KNN, stores data). Model-based: Eager (SVM, learns parameters).

Q10(a): ROC Calculation (6 marks) Scores **S** = [0.95,0.88,0.70,0.65,0.55,0.51,...] Labels **L** = [1,1,0,1,1,0,1,...]

At threshold T=0.50:

- Predicted positive (S0.50): First 6 samples
- True labels of these 6: [1,1,0,1,1,0]
- TP = 4 (positions 1,2,4,5), FP = 2 (3,6)

$$\text{FPR} = \frac{\text{FP}}{N} = \frac{2}{10} = 0.2, \quad \text{TPR} = \frac{4}{10} = 0.4$$

ROC point: (0.2, 0.4)

Module 2: Regression & Bayes

Q11(a): MSE + Regularization (7 marks)

$$\begin{aligned} y &= [10, 20, 30], \quad \hat{y} = [12, 18, 33] \\ e &= [2, -2, 3] \\ \text{MSE} &= \frac{2^2 + (-2)^2 + 3^2}{3} = \frac{17}{3} = \mathbf{5.6667} \end{aligned}$$

For $w = 5$, $\lambda = 0.1$: L1 = 0.5, L2 = 2.5 (L2 heavier).

Q12(a): Cross-Entropy (3 marks) $y = 1$, $\hat{y} = 0.8$:

$$CE = -\log(0.8) = \mathbf{0.22314}$$

Q12(b): Naive Bayes (6 marks)

$$P(\text{Spam}|\text{Free}) = \frac{0.7 \times 0.4}{0.7 \times 0.4 + 0.1 \times 0.6} = \frac{0.28}{0.34} = \mathbf{0.8235}$$

Module 3: Distance & Clustering

Q13(b): Minkowski p=3 (3 marks)

$$d_3[(2,3),(5,7)] = \left[3^3 + 4^3\right]^{1/3} = [27+64]^{1/3} = \mathbf{4.497}$$

Q13(c): Cosine (3 marks)

$$\cos\theta = \frac{[1,1,0]\cdot[0,1,1]}{\sqrt{2}\sqrt{2}} = \mathbf{0.5}$$

Q14(b): K-means Manhattan (4 marks)

$$d(X, C_1) = |10-8| + |12-9| = 5$$

$$d(X, C_2) = |10-15| + |12-15| = 8$$

Assign to C_1.

Module 4: Neural Networks

Q15(a): Perceptron (4 marks)

$$z = 0.4(1) - 0.6(2) + 0.5 = -0.3 < 0 \quad \Rightarrow \quad y = \mathbf{0}$$

Q15(b): Activations z=0 $\sigma(0) = \mathbf{0.5}$, $\tanh(0) = \mathbf{0}$, ReLU= $\mathbf{0}$.

Q15(c): GD Update (2 marks) $w_{\text{new}} = 2.0 + 0.5 = \mathbf{2.5}$.

Q16(a): Backprop (6 marks) $\frac{\partial L}{\partial z} = (y - t)$.

3 Complete Formulas Reference

Concept	Formula	Value
Precision	$\frac{TP}{TP+FP}$	83.33%
MSE	$\frac{1}{n}\sum(y_i - \hat{y}_i)^2$	5.667
CE Loss	$-y\log\hat{y} - (1-y)\log(1-\hat{y})$	0.223
Minkowski	$\left[\sum\lvert x_i - y_i\rvert^p\right]^{1/p}$	4.497
Cosine	$\frac{A\cdot B}{\lVert A\rVert\lVert B\rVert}$	0.5
ROC	(FPR, TPR)	(0.2, 0.4)
L1/L2	$\lambda\lvert w\rvert$ vs λw^2	L2 heavier
Perceptron	$y = 1(z \geq 0)$	y=0

Table 1: All key formulas with exam results

4 Answering Strategy Summary

PART A: Definition + example + formula (30-50 words)
PART B: Show every calculation step, box final answers
Time: Part A (30 min), Part B (120 min)
Total: 24 + 36 = 60 marks