A black background with text

AI-generated content may be incorrect.

INTERNSHIP REPORT

**ON**

**Machine Learning**

**Offered By**

**SkillDzire**

**NAME:** M.Sunil Varma

**REG. NO.:** 22K65A1204

Under the esteemed guidance

**Mrs Ch.Lalitha Syama Sundari**

Assistant professor

DEPARTMENT OF INFORMATION TECHNOLOGY

**SASI INSTITUTE OF TECHNOLOGY& ENGINEERING**

(Approved by AICTE, New Delhi, Permanently Affiliated to JNTUK, Kakinada and SBTET- Hyderabad, Accredited by NAAC with “A+” Grade, Ranked as ‘A’ Grade by Govt. of A.P., Recognized by UGC 2(f) & 12(B)) Kadakatla, TADEPALLIGUDEM– 534101.

**Academic Year 2024-25**



# DECLARATION

I, **M.SunilVarma**, hereby declare that this internship report titled **Machine Learning** is an original work carried out by me during my internship at **SkillDzire**, from December 2024 to 2025. This report is submitted in partial fulfillment of the requirements for the completion of my **Summer Internship Program** at Sasi Institute of Technology & Engineering.

I confirm that this report has not been previously submitted by me or any other individual, either in part or in full, for any degree, diploma, or certificate at this or any other institution.

All information, findings, and interpretations within this report are based on my personal experience and knowledge gained during my tenure with **Machine Learning**, unless otherwise cited. I have duly acknowledged all sources and references used in this report.

I understand that any unauthorized use, reproduction, or misrepresentation of this report constitutes an act of academic misconduct

**M.SunilVarma**

**22K65A1204**

|  |  |
| --- | --- |
| **Academic Supervisor** | **Head of the Department** |
| **Mrs.Ch.Lalitha syama sundari** | **Dr. K. Subhash Bhagavan** |
| **Assistant Professor** | **Associate Professor** |
| **External Examiner** | |

## Abstract

The Machine Learning program offered by SkillDzire provided a comprehensive and hands-on journey into the core concepts and real-world applications of machine learning. The course emphasized Python as the primary programming language due to its simplicity and widespread use in the ML community. Through this training, I gained deep insights into essential ML concepts such as supervised and unsupervised learning, model training, testing, and evaluation using libraries like Scikit-learn, Pandas, NumPy, and Matplotlib. The course also covered critical algorithms including linear regression, decision trees, SVM, K-means clustering, and neural networks. I learned about data preprocessing, feature selection, and the importance of handling imbalanced datasets to improve model accuracy. Real-time projects were integrated into the curriculum to enhance practical experience, including the development of predictive models and classification systems. The program further introduced me to basic concepts of deep learning using TensorFlow and Keras. Additionally, we explored model deployment techniques and tools like Flask to serve ML models as web applications. The SkillDzire certification has equipped me with both theoretical understanding and hands-on experience, making me industry-ready to apply machine learning solutions to complex data-driven problems with confidence.

I

# ACKNOWLEDGEMENT

I express my deep sense of gratitude to my beloved Principal, **Prof. Mohammed Ismail** for his valuable guidance and for permitting us to carry out this internship.

I express my deep sense of gratitude to **Dr. K. Subhash Bhagavan**, Professor and Head of the Department for the valuable guidance and suggestions, keen interest shown thorough encouragement extended throughout the period of internship work

I express my deep sense of gratitude to **Ms.Ch. Lalitha Syama Sundari**, Assistant Professor and Department Internship Coordinator for the valuable guidance and suggestions, keen interest shown thorough encouragement extended throughout the period of internship work.

I express immense pleasure and gratitude to my beloved Guide **Ms.Ch. Lalitha Syama Sundari**, Associate Professor for their benevolent guidance and kind cooperation throughout my training along with completing this Internship and provided me the various knowledge about their stations.

With Gratitude Manukonda.SunilVarma

22K65A1204

II

### TABLE OF CONTENTS

**TITLE PAGE-NO**

**VISION AND MISSION ii**

**VISION AND MISSION OF DEPARTMENT ii**

**POs, PSOs, PEOs and COs List of Figures**

**List of Tables**

**iii iv**

**v**

|  |  |
| --- | --- |
| **Chapter 1: INTRODUCTION TO Machine Learning** | **1** |
| 1.1: What is Machine Learning? | 1 |
| 1.2: Wellsprings of Machine Learning | 2 |
| 1.3:Varieties of Machine Learning | 2 |
| **Chapter-2: Boolean Functions in Machine Learning** | **11** |
| 2.1:Boolean Algebra | 11 |
| 2.2:Diagrammatic Representations | 12 |
| 2.3:Classes of Boolean Functions | 13 |
| **Chapter- 3: Version Spaces** | **16** |
| 3.1:Version Spaces and Mistake Bounds | 17 |
| 3.2:Version Graphs | 18 |
| 3.3 :Learning as Search of a Version Space. | 19 |
| 3.4:Candidate Elimination Method | 20 |
| **Chapter-**4: **Neural Networks** | **23** |

|  |  |
| --- | --- |
| 4.1:Threshold Logic Units | 24 |
| 4.2 Linear Machine | 25 |
| 4.3:Training Feedforward Networks by Backpropagation | 27 |
| 4.4: Synergies with Knowledge-Based Methods | 29 |
| **Chapter-5:Statistical Learning** | **31** |
| 5.1:Statistical Decision Theory | 32 |
| 5.2:Gaussian Distributions | 32 |
| 5.3:Conditionally Independent Binary Components | 33 |
| 5.4:Learning Belief Network | 33 |
| 5.5:Nearest-Neighbor Methods | 34 |
| **Chapter-6:Decision Trees** | **37** |
| 6.1:Definitions | 37 |
| 6.2:Supervised Learning of Univariate Decision Tree | 38 |
| 6.3:Avoiding Overfitting | 40 |
| **Chapter-7:Inductive Logic Programming (ILP)** | **44** |
| 7.1:Notation and Definitions | 44 |
| 7.2:A Generic ILP Algorithm | 45 |
| 7.3:Inducing Recursive Programs | 48 |
| 7.4:Relationships with Decision Trees | 49 |

#### INSTITUTE VISION AND MISSION

**Institute Vision**

Aspire to be a leading institute in professional education by creating technocrats to propel societal transformations through inventions and innovations.

**Institute Mission**

1. To impart technology integrated active learning environment that nurtures the technical life skills.
2. To enhance scientific temper through active research leading to innovations sustainable environment.
3. To create responsible citizens with highest ethical standards.

**DEPARTMENT VISION AND MISSION**

**Department Vision**

To become recognized Centre for excellence for quality Information Technology education and create professionals with ability to solve social needs through invention and innovation.

**Department Mission**

1. Provide quality teaching learning environment oriented towards employ ability and career development.
2. Conduct training/events for overall development of stake holders with collaborations.
3. Impart value base education to serve the society with high integrity and good character.

## PEOs, POs, and PSOs

#### Program Educational Objectives

These PEO’s are meant to prepare our students to thrive and to lead in their career. Our graduates will be able

|  |  |
| --- | --- |
| P1 | Graduates will have strong knowledge about IT applications with leadership  Qualities |
| P2 | Graduates will pursue successful career in IT and allied industries and provide  solutions for global needs |
| P3 | Graduates with life-long learning attitude and practice professional ethics |

#### Program Outcomes

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usag**e: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. **The engineer and the world:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the

consequent responsibilities relevant to the professional engineering practice.

1. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
2. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
3. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
4. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments
5. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

#### Program Specific Outcomes

1. **Application Development:** Develop risk free innovative IT applications for industrial needs.
2. **Successful Career and Entrepreneurship:** Explore technical knowledge in diverse areas of IT and experience an environment conducive in cultivating skills for successful career, entrepreneurship and higher studies

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No** | **Figure Name** | **Page No.** |
| 1.1 | An AI System | 1 |
| 1.2 | An Input-Output Function | 4 |
| 3.1 | Implementing the Version Space | 11 |
| 3.2 A Version Graph for Terms | | 12 |
| 4.1 A Threshold Logic Unit (TLU) | | 16 |
| 4.2 Weight Space | | 18 |
| 5.1 The Two-Dimensional Gaussian Distribution | | 22 |
| 6.1 A Decision Tree | | 26 |
| 6.2 A Decision Tree with Subtree Replication | | 28 |
| 7.1 Sufficient, Necessary, and Consistent Programs | | 33 |

# Introduction to Machine Learning

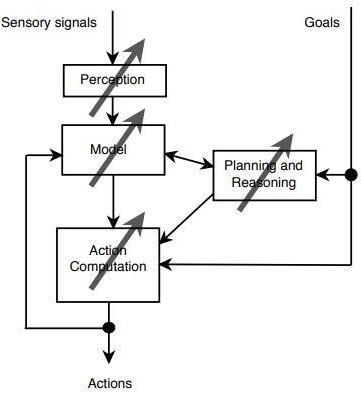
Machine Learning (ML) is a subset of artificial intelligence (AI) focused on building systems that can learn from data and make decisions without being explicitly programmed for every possible scenario. Rather than following static instructions, machine learning algorithms enable systems to identify patterns and adapt to new inputs. This adaptive learning is crucial in today’s data-driven world, where traditional programming methods would struggle to handle the complexities of massive and dynamic datasets.

Fig 1.1 An AI System

##### **What is Machine Learning?**

Machine learning is the field of study that gives computers the ability to learn without being explicitly programmed. It involves designing algorithms and models that can learn patterns from data and make predictions or decisions based on that learning.

##### Key Characteristics**:**

* + - * + **Learning from Data**: The essence of ML is the ability to derive knowledge from data. For example, a model might be trained on past weather data to predict future weather patterns.
        + **Generalization**: ML models should be able to apply the patterns they've learned to new, unseen data.
        + **Adaptability**: Models can improve over time as they are exposed to more data.

##### Types of Machine Learning**:**

* + - * + **Supervised Learning**: The model is trained on a labeled dataset (i.e., data with known outcomes) to learn to predict the output from input data.
        + **Unsupervised Learning**: The model is trained on data without labels, aiming to discover underlying patterns or structures in the data.
        + **Reinforcement Learning**: The model learns by interacting with an environment and receiving feedback through rewards or penalties.
      * **Applications of ML**: It is used in a variety of fields, such as healthcare (e.g., disease diagnosis), finance (e.g., fraud detection), marketing (e.g., customer segmentation), and autonomous driving.

##### Wellsprings of Machine Learning

Machine learning has its origins in several foundational fields, including statistics, computer science, optimization theory, and cognitive science. These disciplines laid the groundwork for creating algorithms that can find patterns in data and improve themselves with experience.

##### History**:**

* Early work in machine learning began with attempts to simulate human intelligence and cognitive processes.
* Statistical methods in the mid-20th century began to be applied to machine learning problems, paving the way for modern supervised learning techniques.
* The rise of powerful computers and the availability of vast amounts of data in the 21st century accelerated the development and adoption of machine learning techniques.

##### Key Influences**:**

* + - * + **Artificial Intelligence (AI)**: ML is a core subfield of AI, focusing on systems that learn and adapt autonomously.
        + **Statistics**: Concepts such as probability, regression, and hypothesis testing form the basis of many ML algorithms.
        + **Optimization**: Many ML algorithms involve finding optimal solutions or minimizing loss functions, drawing heavily from optimization theory.
        + **Cognitive Science**: Early ML systems were inspired by the human brain’s ability to learn from experience and make decisions.

##### **Varieties of Machine Learning**

Machine learning can be categorized based on the type of data used and the learning process. The three main categories are:

##### Supervised Learning**:**

* + **Definition**: A model is trained on labeled data (input-output pairs). The goal is to learn the mapping from inputs to outputs.
  + **Common Algorithms**: Linear regression, decision trees, support vector machines (SVM), neural networks.
  + **Applications**: Image classification, spam email detection, medical diagnoses.

##### Unsupervised Learning**:**

* + **Definition**: The model is given data without explicit labels and must find patterns, groupings, or structures within the data on its own.
  + **Common Algorithms**: K-means clustering, hierarchical clustering, principal component analysis (PCA), autoencoders.
  + **Applications**: Market segmentation, anomaly detection, dimensionality reduction.

##### Reinforcement Learning**:**

* + **Definition**: An agent learns by interacting with an environment and receiving feedback in the form of rewards or penalties.
  + **Key Concepts**: Exploration vs. exploitation, reward signals, Markov Decision Processes (MDPs).
  + **Applications**: Robotics, game playing (e.g., AlphaGo), autonomous driving.

##### Semi-supervised and Self-supervised Learning**:**

* + **Definition**: These methods lie between supervised and unsupervised learning. Semi-supervised learning uses a small amount of labeled data along with a large amount of unlabeled data. Self-supervised learning generates its own labels from the input data.

##### **Learning Input-Output Functions**

Machine learning can be viewed as a process of learning a function that maps inputs to outputs. This section would explore how this function is learned and how different types of inputs and outputs are handled.

##### **Types of Learning**

Learning in machine learning can be broadly classified into different categories

based on the way the model interacts with the data:

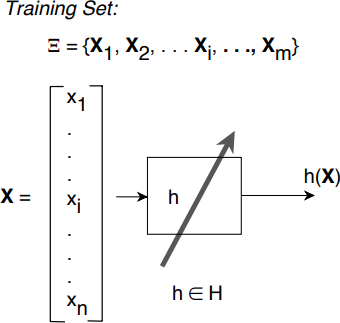


Fig 1.2 An Input-Output Function

##### **Supervised Learning:**

* + - * + The most straightforward approach, where the system learns from input-output pairs and generalizes this mapping to unseen data.
        + **Examples**: Image classification, medical diagnosis, financial prediction.

##### **Unsupervised Learning:**

* + - * + Involves learning patterns or structure from input data without any labeled outputs. The focus is on uncovering hidden patterns, groupings, or relationships in the data.
        + **Examples**: Customer segmentation, clustering, anomaly detection.

##### **Reinforcement Learning:**

* + - * + Involves an agent interacting with an environment, learning to make decisions based on feedback in the form of rewards or penalties.
        + **Examples**: Robotics, self-learning AI agents in games, autonomous vehicles.

##### **Transfer Learning:**

* + - * + Involves leveraging knowledge learned from one task to improve learning on a related task. This is particularly useful when data is scarce for a new problem but abundant for similar problems.

##### **Few-shot Learning:**

* + - * + A type of learning where the model is expected to learn from only a few examples.

##### **Input Vectors**

An input vector refers to the representation of the input data fed into a machine learning model. Each data point can be viewed as a vector of features or variables, which are the dimensions of the data that the model uses to make predictions or classifications.

* **Feature Engineering**: The process of selecting, transforming, or creating new features from raw data that improve the performance of the model.
  + **Examples**: Extracting numerical features from text (e.g., word frequencies), normalizing features to scale them, creating categorical features.
* **Feature Selection**: The process of identifying and using the most relevant features while ignoring redundant or irrelevant ones.
* **Dimensionality Reduction**: Techniques like PCA or t-SNE that reduce the number of features in a dataset to make models more efficient.

##### **Outputs**

The output of a machine learning model is what the model predicts or produces after being trained on input data. Depending on the task, outputs can take many forms:

* **Regression Tasks**: Outputs are continuous values, such as predicting house prices based on features like location, size, etc.
* **Classification Tasks**: Outputs are discrete categories, such as labeling emails as spam or not spam.
* **Clustering Tasks**: Outputs are group assignments, where similar data points are grouped together (e.g., customer segments in marketing).
* **Reinforcement Learning**: The output is a series of actions that maximize the cumulative reward in an environment.

In supervised learning, the output is typically compared to the true label (or ground truth) to calculate the model’s performance, usually via loss functions like Mean Squared Error (MSE) or Cross-Entropy.

# Boolean Functions in Machine Learning

In machine learning, Boolean functions are used to describe decision-making processes, particularly in **classification problems**, **feature selection**, and the construction of **decision trees**. These functions map input features to outputs that are often binary (0 or 1, true or false). Understanding how to represent and work with Boolean functions is crucial for designing models that deal with binary classification tasks, decision rules, and logic-based models.

##### **Representation**

* + 1. **Boolean Algebra in Machine Learning**

In machine learning, **Boolean algebra** helps represent logical relationships between features and outcomes. For example, a binary classifier might use Boolean operations to combine features and determine the class label (e.g., **yes** or **no**).

##### Logical Operations in ML**:**

* + - * + **AND (**⋀**)**: Represents an intersection or conjunction of conditions. For example, "if age > 50 AND income > 30k, then the prediction is ‘yes’".
        + **OR (**⋁**)**: Represents a disjunction where the condition is true if at least one of the features is satisfied. E.g., "if age > 50 OR income > 30k, then predict ‘yes’".
        + **NOT (¬)**: Reverses a condition. E.g., "if NOT (age > 50), then predict ‘no’".
      * **Use in Decision Trees**: Boolean algebra simplifies decision trees by using basic operations to combine conditions. Each decision node in a decision tree might involve an AND/OR combination of features.
      * **Simplification**: Boolean algebra is used to **simplify decision rules** in ML models, allowing for the creation of more compact and efficient decision- making rules (e.g., simplifying a series of logical conditions in decision tree pruning).

##### **Diagrammatic Representations in Machine Learning**

In machine learning, diagrammatic representations of Boolean functions are crucial for visualizing decision-making processes, especially when explaining or debugging models.

* + - * **Truth Tables**: In the context of binary classification, truth tables can represent all possible combinations of input features and their associated output. For a classifier that takes two binary features, a truth table could show all possible feature combinations and the corresponding predicted outcome.
      * **Decision Trees**: A **decision tree** can be viewed as a series of Boolean decisions applied to features. Each decision node tests whether a certain condition holds (e.g., "Is feature x > threshold?"), and the tree branches according to the answer (true or false). The decision rules can be represented as a series of Boolean expressions.
      * **Logic Gates**: In certain models, such as **neural networks** or **rule-based classifiers**, Boolean logic gates (AND, OR, NOT) can be used as building blocks for more complex decisions. For example, a perceptron (the simplest form of a neural network) behaves like a logic gate that combines weighted inputs using Boolean operations to make a binary decision.
      * **Karnaugh Maps (K-Maps)**: Though typically used in hardware design, K-maps can also help simplify **Boolean expressions** for machine learning models by reducing the complexity of decision rules. They’re used to minimize the number of features or conditions in classification models.

##### **Classes of Boolean Functions in Machine Learning**

In machine learning, Boolean functions help define different **types of decision boundaries** and **classification rules**. These functions can be categorized into different forms based on how they are used to classify data points.

##### **Terms and Clauses in Decision Rules**

In decision rule-based classifiers, **terms** refer to individual conditions or comparisons (e.g., “age > 50”), while **clauses** are combinations of terms (e.g., “age > 50 AND income > 30k”).

* + - * **Example**: A decision rule like **"age > 50 AND income > 30k"** represents a **term** (age > 50) and another **term** (income > 30k), combined with the AND operator.

##### **Disjunctive Normal Form (DNF) Functions**

A Boolean function is in **Disjunctive Normal Form (DNF)** if it is expressed as an OR (disjunction) of AND (conjunction) terms. In machine learning, DNF is useful for **rule-based classifiers** where we have multiple conditions that lead to a positive outcome.

* + - * **Example**: In a binary classification problem, a DNF function could represent a rule like:
        + "Predict ‘yes’ if (age > 50 AND income > 30k) OR (age <= 50 AND income > 40k)."

DNF is highly relevant for models that use **rule-based learning**, like decision trees, where each path in the tree can be seen as a conjunction of features.

##### **Conjunctive Normal Form (CNF) Functions**

A Boolean function is in **Conjunctive Normal Form (CNF)** if it is expressed as an AND (conjunction) of OR (disjunction) terms. CNF is less common in typical machine learning algorithms, but it can be useful for certain forms of **logical rule- based classification**, especially when using **Satisfiability Solvers** (like in constraint satisfaction problems).

* + - * **Example**: A CNF might represent a rule like:
        + "Predict ‘no’ if (age > 50 OR income > 30k) AND (age <= 50 OR income <= 20k)."

##### **Decision Lists**

**Decision lists** are a sequence of ordered rules used for classification. Each rule in a decision list is a Boolean expression, and the output is determined by the first matching rule.

* + - * **Example**: A decision list might look like:
        + "If age > 50, predict ‘yes’."
        + "If income > 40k, predict ‘yes’."
        + "Otherwise, predict ‘no’."

Decision lists are useful in situations where there is a **priority** among rules or when conditions are complex.

##### **Symmetric and Voting Functions**

In some machine learning tasks, such as **ensemble learning** (e.g., **Random Forests** or **Boosting**), **voting functions** aggregate the outputs of several classifiers.

* + - * **Symmetric Functions**: These are Boolean functions that produce the same output for any permutation of their input variables. In machine learning, this can be useful when dealing with symmetric data, such as in ensemble learning, where the order of the classifiers doesn’t matter.
      * **Voting Functions**: These involve taking a vote from multiple classifiers. For example, in a majority voting scheme, the most frequent output from multiple classifiers (each making Boolean predictions) is chosen as the final decision.

##### **Linearly Separable Functions**

A Boolean function is **linearly separable** if there exists a hyperplane (or line in two dimensions) that separates the inputs into two classes. **Linear classifiers** (e.g., **Perceptrons**) can be used to model these functions.

* + - * **Example**: A **linearly separable function** might involve a classification problem where a decision boundary can separate the positive and negative instances in feature space. In such cases, a linear model like **Logistic Regression** or **Support Vector Machines (SVM)** can perform well.

# Using Version Spaces for Learning

In machine learning, **Version Spaces** are a conceptual framework used to describe the set of hypotheses (models) consistent with a set of training examples. These hypotheses are iteratively refined as more examples are encountered, and the space of possible models narrows down. The **Candidate Elimination Method** is one specific algorithm that uses this framework to incrementally eliminate hypotheses that do not fit the observed data.

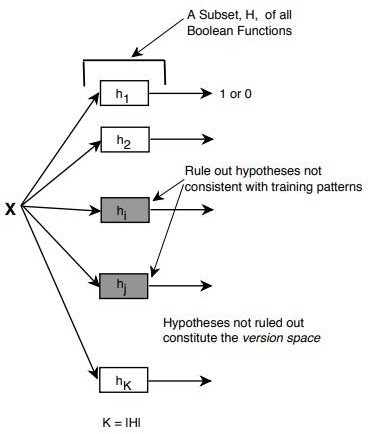


Fig 3.1 Implementing the Version Space

##### **Version Spaces and Mistake Bounds Version Spaces:**

* + - **Definition**: A **Version Space** is the set of all hypotheses that are consistent with a given set of training examples. It is essentially the collection of all possible models that could explain the observed data.
    - **Learning Process**: As more training examples are provided, the version space is updated. If a hypothesis is inconsistent with a new training example, it is removed from the version space. Over time, the version space shrinks until only one hypothesis (or a small set) remains, which is used for prediction.
    - **Mistake Bounds**: The **mistake bound** is a theoretical concept used to measure how many mistakes (incorrect predictions) a learning algorithm might make before arriving at the correct hypothesis. In the context of version spaces, the mistake bound can be used to determine the number of incorrect predictions the algorithm will make before converging to a hypothesis that correctly classifies all future examples.

##### **Mistake Bound Analysis:**

The mistake bound often depends on factors like:

* + - **Size of the version space**: The number of possible hypotheses that could fit the data.
    - **The nature of the training data**: How noisy the data is and how well the hypotheses generalize to unseen data.
    - **The learning algorithm used**: For example, some algorithms might converge faster than others, reducing the mistake bound.

For a consistent learner (one that always eventually finds the correct hypothesis), the mistake bound provides a limit on how many mistakes can occur before the algorithm identifies the correct model.

##### **Version Graphs**

Fig 3.2 A Version Graph for Terms

 **Version Graphs** are a more structured way to represent version spaces.

Rather than storing all hypotheses in an unorganized manner, a version graph organizes hypotheses based on their generality and specificity.

* + - **Nodes**: Each node in a version graph represents a hypothesis (a possible model).
    - **Edges**: An edge between two nodes indicates that one hypothesis is more specific or general than the other. If one hypothesis is a generalization (broader) of another, it is connected to the more specific hypothesis.
    - **Version Graphs for Learning**: These graphs help visualize the space of possible hypotheses and make it easier to reason about the relationships between hypotheses. By organizing hypotheses this way, learners can explore the hypothesis space more efficiently. They can navigate the version graph to eliminate inconsistent hypotheses based on new training data and gradually refine the hypotheses set.
    - **Graph Structure**: The version graph typically has a hierarchical structure where:
      * The most general hypotheses (the least restrictive) are at the top.
      * The most specific hypotheses (the most restrictive) are at the bottom.

Version graphs are particularly useful in **inductive learning** where the goal is to identify the most specific hypothesis that still explains the training data.

##### **Learning as Search of a Version Space**

In this approach, **learning is framed as a search process** through the version space. The goal of the learner is to efficiently explore the version space to find a hypothesis that best fits the training data.

##### **Search Process:**

* + - **Initial Step**: Start with a broad version space that includes all hypotheses.
    - **Refinement**: As each training example is processed, the learner eliminates hypotheses that are inconsistent with the new example.
    - **Convergence**: Over time, the version space narrows down, eventually converging to a hypothesis (or a small set of hypotheses) that correctly classifies future examples.

In practice, this search is done by using algorithms like the **Candidate Elimination Method**, which is designed to update the version space based on new examples.

##### **Challenges in Search:**

* + - **Exploration vs. Exploitation**: In some cases, searching the version space may involve trade-offs between exploring new hypotheses and exploiting the current best hypothesis.
    - **Efficiency**: Searching a large version space can be computationally expensive. Therefore, methods like pruning or using heuristics to narrow down the search space are important.

##### **The Candidate Elimination Method**

The **Candidate Elimination Method** is a specific algorithm used to search and refine the version space. It operates in a way that progressively narrows down the version space based on the training data, with the goal of eventually identifying the best hypothesis.

##### **How it Works:**

1. **Initialization**: Start with two sets of hypotheses:
   * **S**: The set of most specific hypotheses (initially, this is the hypothesis that classifies all examples incorrectly).
   * **G**: The set of most general hypotheses (initially, this is the hypothesis that classifies all examples correctly).

##### **Processing each training example:**

* + For each example, update the **S** and **G** sets:
    - **S** is refined to remove any hypotheses that are inconsistent with the example.
    - **G** is refined to include only hypotheses that could explain the example while remaining consistent with the training data.

##### **Refinement:**

* + **Specific Hypotheses**: If a hypothesis in **S** does not fit the new training example, it is generalized to be consistent with the example.
  + **General Hypotheses**: If a hypothesis in **G** does not fit, it is specialized (made more specific) to be consistent with the example.

1. **Convergence**: Over time, as more examples are processed, the sets **S** and **G** converge. The **S** set becomes more specific, and the **G** set becomes more general, ultimately leading to a refined hypothesis that fits the training data.

##### **Advantages:**

* **Efficient**: This method helps in narrowing down the hypothesis space quickly by systematically eliminating inconsistent hypotheses.
* **Clear Decision Boundaries**: Since the method works by maintaining general and specific hypotheses, it provides clear boundaries for classification.

##### **Disadvantages:**

* **Requires Consistent Data**: The Candidate Elimination Method assumes that the data is consistent (i.e., there exists a hypothesis that correctly classifies all examples). If the data is noisy or inconsistent, the method may fail to converge.
* **Computational Complexity**: The method can be computationally expensive for large hypothesis spaces, as it requires comparing multiple hypotheses with each new training example.

##### **Summary of Concepts:**

* **Version Space**: The set of all hypotheses consistent with the training data.
* **Mistake Bounds**: The theoretical limit on how many mistakes an algorithm might make before converging to the correct hypothesis.
* **Version Graph**: A hierarchical structure representing the relationships between hypotheses in a version space.
* **Learning as Search**: Learning involves searching through the version space to find the correct hypothesis.
* **Candidate Elimination Method**: An algorithm for refining the version space, iterating over the training examples to eliminate inconsistent hypotheses and converge on the best model.

Together, these concepts provide a framework for inductive learning, helping algorithms efficiently search for hypotheses that explain the observed data. The Candidate Elimination Method, in particular, is foundational for **concept learning** tasks in machine learnin

# Neural Networks

In machine learning, **Neural Networks (NNs)** are a powerful class of models designed to recognize patterns by learning from data. They consist of interconnected units (neurons), inspired by biological neural networks, that work together to solve tasks like classification, regression, and more complex tasks such as steering a van, as mentioned in your application example. Let's explore the specific sections you've highlighted in more details.

##### **Threshold Logic Units (TLUs)**

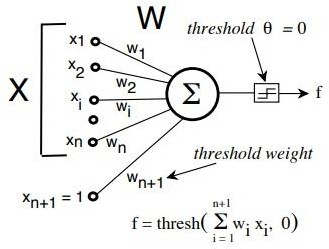
A **Threshold Logic Unit** (TLU) is a type of artificial neuron used in early neural network models. It acts as a basic building block for more complex neural networks.

Fig 4.1 A Threshold Logic Unit (TLU)

##### **Definitions and Geometry**

* + - * **Definition**: A TLU takes a set of inputs, applies weights to them, sums them up, and then applies a threshold function to decide whether the output should be 0 or 1 (binary classification). Mathematically, the output yyy of a TLU can be described as:

y=step(w1x1+w2x2+⋯+wnxn−θ)y = \text{step}(w\_1x\_1 + w\_2x\_2 + \dots + w\_nx\_n - \theta)y=step(w1x1+w2x2+⋯+wnxn−θ)

Where:

* + - * + x1,x2,…,xnx\_1, x\_2, \dots, x\_nx1,x2,…,xn are the input features.
        + w1,w2,…,wnw\_1, w\_2, \dots, w\_nw1,w2,…,wn are the weights associated with the inputs.
        + θ\thetaθ is the threshold.
      * **Geometry**: The decision boundary of a TLU is a hyperplane in the input space. This means that the function performed by the TLU can be visualized geometrically as separating the input space into two regions: one where the output is 1 and another where the output is 0.

##### **Special Cases of Linearly Separable Functions**

* + - * **Linearly Separable Functions:** A function is linearly separable if the data points can be separated into two classes by a straight line (in 2D) or a hyperplane (in higher dimensions). For linearly separable data, a TLU can be trained to correctly classify all examples.
      * **Example**: In a binary classification problem where the input data can be separated with a straight line (such as classifying points above or below a line), a single TLU can learn the decision boundary effectively.

##### **Error-Correction Training of a TLU**

* + - * **Training**: A simple approach to training a TLU is the **error-correction rule**. For each training example, the network compares the predicted output with the actual output. The weights are updated based on the error, typically using a simple rule like: wi←wi+Δwiw\_i \leftarrow w\_i + \Delta w\_iwi wi+Δwi Where: Δwi=η(t−o)xi\Delta w\_i = \eta (t - o) x\_iΔwi=η(t−o)xi

##### **Weight Space**

 The **weight space** is the multi-dimensional space where each point represents a set of weights for the TLU. During training, the weight vector is adjusted iteratively, moving through this space towards the optimal weights that minimize error.

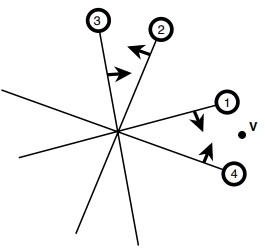


Fig 4.2 Weight Space

##### **The Widrow-Hoff Procedure**

* + - * The **Widrow-Hoff rule**, also known as the **delta rule**, is a gradient descent method for updating the weights of the TLU. It adjusts weights in the direction that minimizes the error between the predicted and target outputs.

##### **Training a TLU on Non-Linearly Separable Training Sets**

* + - * **Non-linearly separable** data refers to cases where no straight line or hyperplane can separate the data into two classes. In these cases, a single TLU won't work. This limitation leads to the development of more complex architectures like multi-layer networks (i.e., neural networks).

##### **Linear Machines**

* **Linear Machines** are models that separate data using linear decision boundaries. These include **linear classifiers** like **perceptrons**, which can be used to classify linearly separable data. However, they struggle with non-linearly separable data, which leads to the development of more complex neural network models.

##### **Networks of TLUs**

A network of TLUs refers to a collection of TLUs arranged in layers to solve more complex problems.

##### **Motivation and Examples**

* + - * The motivation behind networks of TLUs is to tackle more complex, non- linear problems. By stacking multiple TLUs into layers, networks can create complex decision boundaries, enabling the classification of non- linearly separable data.
      * **Example**: An XOR function, which is not linearly separable, can be solved by a simple network of TLUs.

##### **Madalines**

* + - * **Madalines** (Multiple Adaline Units) are a type of neural network that consists of several Adaline units, which are similar to TLUs but use linear activation functions. These networks were used to solve problems that single-layer networks couldn't handle.

##### **Piecewise Linear Machines**

* + - * These machines are neural networks that use piecewise linear activation functions. They can approximate any continuous function by combining several linear segments.

##### **Cascade Networks**

* + - * **Cascade Networks** are a type of network where the outputs of earlier layers are fed into subsequent layers. This approach allows the network to build complex decision boundaries step by step.

##### **Training Feedforward Networks by Backpropagation**

* + 1. **Notation**
       - **Feedforward Networks**: These networks are composed of layers of neurons, where each layer is fully connected to the next one, and information flows in one direction (from input to output).
       - **Notation**: The input to each neuron is represented as a vector, and weights are associated with the connections between neurons. The output of a neuron is computed as a weighted sum of its inputs, followed by an activation function.

##### **The Backpropagation Method**

* + - * **Backpropagation** is the key algorithm used for training multi-layer neural networks. It computes the gradient of the error with respect to each weight by applying the chain rule of calculus, propagating the error backwards from the output layer to the input layer.

##### **Computing Weight Changes in the Final Layer**

* + - * The weights in the **final layer** of a network are adjusted based on the error between the predicted output and the target. The weight updates are proportional to the error gradient and the input values to the layer.

##### **Computing Changes to the Weights in Intermediate Layers**

* + - * For **intermediate layers**, the weight updates depend on the error from the subsequent layer, multiplied by the derivative of the activation function. This allows the network to learn from both the direct and indirect contributions of neurons to the final output.

##### **Variations on Backprop**

* + - * **Stochastic Gradient Descent (SGD)**: A popular variant of backpropagation where weights are updated after processing each individual training example, instead of after processing the entire dataset.
      * **Mini-batch Gradient Descent**: A hybrid approach where weights are updated after processing small batches of data, balancing computational efficiency and convergence speed.

##### **An Application: Steering a Van**

* + - * **Backpropagation** can be used in applications like **autonomous driving**, where a neural network might be trained to steer a van by processing sensory inputs (such as camera images or lidar data) and outputting steering commands. This involves mapping inputs to desired outputs (steering angles) through a multi-layer neural network.

##### **Synergies Between Neural Networks and Knowledge-Based Methods**

* **Neural networks** and **knowledge-based methods** can work together to enhance learning. Knowledge-based systems use explicit rules and domain-specific knowledge to guide decision-making, while neural networks can learn patterns from data. By combining both, we can create more robust models that leverage both learned data and predefined knowledge.

##### **Summary of Key Concepts:**

1. **Threshold Logic Units (TLUs)**: Basic building blocks of neural networks that classify based on linear thresholds.
2. **Linear Machines**: Classifiers that separate data using linear decision boundaries.
3. **Networks of TLUs**: Multi-layer architectures that allow solving non-linear problems.
4. **Backpropagation**: The algorithm for training multi-layer neural networks by adjusting weights based on error gradients.
5. **Applications**: Neural networks can be used for complex tasks like classification, regression, and control systems (e.g., steering a van).
6. **Synergy with Knowledge-Based Methods**: Neural networks and knowledge-based methods can complement each other to improve model performance.

Neural networks, especially with techniques like backpropagation, have become the foundation for many modern machine learning tasks due to their ability to handle complex, non-linear problems.

# Statistical Learning

**Statistical learning** is a framework in machine learning where models are trained based on statistical theory. The goal is to make predictions about unknown data based on observed data. Statistical methods offer a robust approach to dealing with uncertainty and variability in real-world data. The main methods discussed in this section are **Statistical Decision Theory**, **Belief Networks**, and **Nearest- Neighbor Methods**.

##### **Using Statistical Decision Theory**

Statistical Decision Theory provides a framework for decision-making under uncertainty. It aims to model and optimize the decision-making process by considering potential outcomes, their associated probabilities, and the costs or benefits of those outcomes.

##### **Background and General Method**

The general method of **Statistical Decision Theory** involves:

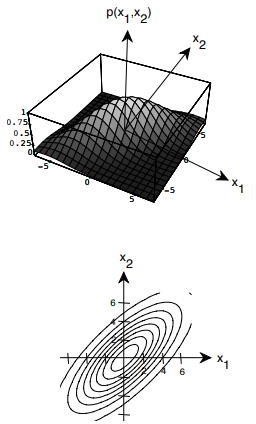
1. **Defining the decision problem**: The problem is modeled with a set of possible actions and outcomes.
2. **Assigning probabilities** to the possible outcomes of each decision

fig 5.1 The Two-Dimensional Gaussian Distribution

1. **Assigning costs or utilities** to each possible outcome, based on the decision made.
2. **Choosing the decision** that maximizes the expected utility or minimizes the expected loss.

This is especially useful in situations where we have incomplete knowledge about the data or the environment, and we aim to make the best decision given the uncertainty.

##### **Gaussian (or Normal) Distributions**

A **Gaussian distribution** (also known as a **normal distribution**) is a probability distribution commonly used in statistical learning due to its many useful properties, such as its symmetry and the central limit theorem.

##### Properties of Gaussian distributions**:**

 It is fully characterized by its **mean** (μ\muμ) and **variance**

(σ2\sigma^2σ2).

* The probability density function (PDF) is given by: f(x)=1σ2πexp⁡(−(x−μ)22σ2)f(x) = \frac{1}{\sigma\sqrt{2\pi}}

\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)f(x)=σ2π1 exp(−2σ2(x−μ)2)

* **Bell curve** shape: The distribution has a peak at the mean and tails that extend towards infinity.

Gaussian distributions are widely used in modeling real-world data, especially when the data exhibits variability around a central value. In many machine learning models (e.g., **Gaussian Naive Bayes**), it’s assumed that the features follow a Gaussian distribution.

##### **Conditionally Independent Binary Components**

In some learning problems, we assume that the features (or variables) are conditionally independent given the target class. This assumption is central to models like **Naive Bayes** classifiers.

* + - * **Conditional Independence**: The assumption is that, given the class label, the individual features do not influence each other. Mathematically, for binary features X1,X2,…,XnX\_1, X\_2, \dots, X\_nX1,X2,…,Xn and class YYY, this assumption can be written as: P(X1,X2,…,Xn∣Y)=∏i=1nP(Xi∣Y)P(X\_1, X\_2, \dots, X\_n \mid Y) =\prod\_{i=1}^{n} P(X\_i \mid Y)P(X1,X2,…,Xn∣Y)=i=1∏nP(Xi∣Y)

This simplifies the model and makes it computationally feasible, though it may not always be true in practice. Nevertheless, the simplicity of this assumption often leads to good performance, especially when the features are not strongly dependent.

##### **Learning Belief Networks**

**Belief Networks** (also known as **Bayesian Networks**) are a type of probabilistic graphical model used to represent the conditional dependencies between variables in a compact form. These networks consist of nodes (representing variables) and directed edges (representing dependencies).

* **Learning Belief Networks** involves:
  + **Modeling the joint probability distribution** of a set of variables.
  + Using **Bayes' theorem** to update beliefs as new evidence is observed.
  + Inference: Computing the posterior probabilities of certain variables given observed data.

Belief networks are powerful tools for handling uncertainty and for building models where multiple variables interact in complex ways. They are used in areas such as decision support systems, diagnostics, and pattern recognition.

##### **Nearest-Neighbor Methods**

**Nearest-Neighbor Methods** are a class of algorithms used for classification and regression. They work by comparing new data points to the most similar, or "nearest," data points in the training set and making predictions based on the known outcomes of those neighbors.

##### Key Points of Nearest-Neighbor Methods:

* **k-Nearest Neighbors (k-NN)**: A widely used method where the class or value of a new data point is predicted based on the majority class (for classification) or the average value (for regression) of the k nearest data points in the training set.
  + **Distance Metric**: The proximity between points is typically measured using distance metrics like **Euclidean distance**: dist(x,x′)=∑i=1n(xi−xi′)2\text{dist}(x, x') = \sqrt{\sum\_{i=1}^{n}(x\_i - x'\_i)^2}dist(x,x′)=i=1∑n(xi−xi′)2 Other distance metrics can be used depending on the type of data, such as Manhattan distance or cosine similarity.
  + **Choosing k**: The number of neighbors (k) is a critical parameter. A small value of k makes the model sensitive to noise, while a large value can make the model overly smooth and less sensitive to local patterns.

##### **Advantages:**

* + Simple and intuitive.
  + Non-parametric: It makes no assumptions about the underlying data distribution.
  + Works well for both classification and regression tasks.

##### **Disadvantages:**

* + Computationally expensive, especially for large datasets, because it requires calculating distances to all training points.
  + Sensitive to the choice of distance metric and the scaling of features.

##### **Summary of Key Concepts**

1. **Statistical Decision Theory**: A framework for making decisions under uncertainty by modeling actions, outcomes, and probabilities.

##### **Gaussian Distributions:** A common probability distribution in statistical learning, used for modeling continuous data with symmetry around a mean.

##### **Conditionally Independent Binary Components:** Assumption of independence between features given the class, central to models like Naive Bayes.

##### **Belief Networks:** Graphical models that represent the probabilistic relationships between variables.

##### **Nearest-Neighbor Methods:** A non-parametric approach to classification and regression based on the similarity between new data points and training data.

These statistical learning methods provide the theoretical foundation for many machine learning algorithms, from basic classifiers to sophisticated probabilistic models. They help guide decision-making in uncertain environments, model complex dependencies, and make predictions based on observed data.

# Decision Trees

**Decision Trees** are one of the most popular and interpretable machine learning algorithms. They are used for both classification and regression tasks and work by partitioning the feature space into subsets and making predictions based on the majority class or average value of the data points in each subset. A decision tree is structured as a tree, where each internal node represents a test (or decision) on a feature, each branch represents the outcome of that test, and each leaf node represents a class label or a continuous value.

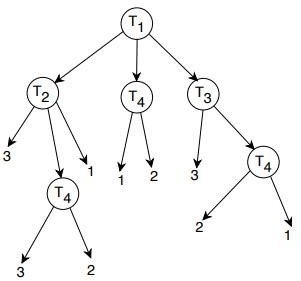


Fig 6.1 A Decision Tree

##### **Definitions**

A **decision tree** is a flowchart-like structure where:

* **Nodes** represent tests on attributes (features of the data).
* **Edges** (branches) represent the outcomes of those tests.
* **Leaves** represent a decision or classification label (for classification tasks) or a predicted value (for regression tasks).

##### **Important Terminology:**

* **Root Node**: The topmost node in a decision tree, where the first decision is made.
* **Internal Nodes**: Nodes that represent decision tests based on input features.
* **Leaf Nodes**: Terminal nodes that assign a class label or output a predicted value.
* **Branches**: Edges connecting nodes, representing possible outcomes of the test.

##### **Supervised Learning of Univariate Decision Trees**

**Univariate decision trees** use a single feature (attribute) at each decision node to split the data. This makes the tree interpretable, as each decision only considers one feature at a time.

##### **Selecting the Type of Test**

When building a decision tree, the first step is to decide what type of test to use at each node. Tests can involve:

* + - * **Threshold tests** for continuous features (e.g., "Is age > 30?").
      * **Categorical tests** for discrete features (e.g., "Is the color red?").

The choice of tests influences the structure of the tree and how well it generalizes to unseen data.

##### **Using Uncertainty Reduction to Select Tests**

The goal of a decision tree is to reduce uncertainty (or **entropy**) at each node. One popular criterion to decide how to split the data at each node is the **Information Gain** (or reduction in entropy).

* + - * **Entropy** is a measure of uncertainty or impurity in the dataset.
        + For classification, the entropy H(S)H(S)H(S) of a dataset SSS is defined as: H(S)=−∑i=1kpilog⁡2piH(S) = - \sum\_{i=1}^{k} p\_i

\log\_2 p\_iH(S)=−i=1∑kpilog2pi where pip\_ipi is the probability of a class label iii in the set SSS.

* + - * **Information Gain** is the reduction in entropy achieved by splitting the dataset based on a particular attribute.
        + It is calculated as the difference between the entropy of the original set and the weighted sum of the entropies of the subsets created by the split.

The attribute that maximizes **Information Gain** is chosen for the test at the current node.

##### **Non-Binary Attributes**

Decision trees can handle both **binary** (true/false) and **non-binary** (multiple categories) attributes. For non-binary attributes, a test could involve comparing the attribute to several possible values or ranges. The splitting criteria can be generalized by using **multi-way splits** instead of just binary splits.

##### **Networks Equivalent to Decision Trees**

Certain types of networks, such as **feedforward neural networks** or **regression trees**, can represent the same decision-making process as decision trees. These networks might have additional complexity or flexibility, but conceptually they can achieve similar outcomes.

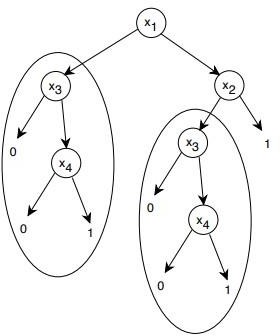


Fig 6.2 A Decision Tree with Subtree Replication

 **Decision Trees** can be viewed as shallow neural networks that are

specifically designed for **interpretable decision-making**.

 Some methods like **Madalines** (Multiple Adaptive Linear Elements) attempt to bridge the gap between decision trees and neural networks by using a combination of threshold units to replicate the decision-making process of a tree.

##### **Overfitting and Evaluation**

Overfitting occurs when a model learns too much from the training data, capturing noise and irregularities instead of generalizable patterns. This leads to poor performance on unseen data.

##### **Overfitting**

Overfitting happens when the decision tree becomes too complex, splitting the data into many small subsets that are too specific to the training data. While this results in perfect accuracy on the training set, the model performs poorly on new data.

* + - * **Signs of Overfitting**: The model has high accuracy on training data but low accuracy on validation/test data.

##### **Validation Methods**

To evaluate the performance of a decision tree and combat overfitting, various **validation methods** are used:

* + - * **Cross-validation**: Splitting the data into multiple subsets (folds) and training and testing the model on different combinations of these folds.
      * **Holdout Method**: Splitting the dataset into training and testing sets and using the testing set to evaluate the model.
      * **Bootstrap Sampling**: Randomly sampling from the training set to build multiple models and testing on the unseen data.

##### **Avoiding Overfitting in Decision Trees**

Several techniques can be used to prevent overfitting in decision trees:

* + - * **Pruning**: Reducing the size of the tree after it has been grown, removing branches that do not provide significant predictive value.
      * **Limiting tree depth**: Restricting the maximum depth of the tree to prevent excessive complexity.
      * **Minimum samples per leaf**: Setting a minimum number of data points required in a leaf node to prevent the tree from creating overly specific rules for small subsets of the data.

##### **Minimum-Description Length Methods**

The **Minimum-Description Length (MDL)** principle is a way to balance model complexity with accuracy. The idea is to select the tree that minimizes the total description length (the number of bits needed to describe both the tree and the data). This is closely related to **Occam’s Razor**, where simpler models are preferred if they perform similarly to more complex ones.

##### **The Problem of Replicated Subtrees**

In decision trees, **replicated subtrees** can occur when the same subset of data is processed by multiple branches of the tree. This redundancy can be inefficient and unnecessary. Identifying and eliminating such replicated subtrees helps reduce the tree's complexity.

##### **The Problem of Missing Attributes**

A common issue in real-world datasets is missing attribute values. Decision trees can handle missing data in several ways:

* **Imputation**: Replacing missing values with estimates, such as the mean or median value for continuous attributes or the most common value for categorical attributes.
* **Handling Missing Values During Splitting**: When splitting data at a node, decision trees can handle missing values by assigning them to the branch that most closely matches the missing attribute's distribution.

##### **Comparisons**

* **Advantages of Decision Trees**:
  + **Interpretability**: Easy to understand and visualize.
  + **Non-parametric**: No assumptions about the underlying data distribution.

##### Can handle both classification and regression tasks**.**

* **Disadvantages of Decision Trees**:
  + **Overfitting**: Susceptible to overfitting, especially with deep trees.
  + **Instability**: Small changes in the data can result in a very different tree.
  + **Bias**: Can be biased toward features with more levels or continuous attributes.

##### **Summary**

* **Decision Trees** are powerful models for classification and regression that partition the feature space based on tests.
* Key techniques such as **information gain**, **pruning**, and **cross-validation**

are essential for training robust decision trees.

* **Overfitting** is a critical challenge, and methods like pruning and limiting tree depth can help mitigate it.
* Decision trees are also susceptible to issues such as **replicated subtrees**

and **missing attributes**, but these can be addressed with proper techniques.

By understanding these key components and strategies, decision trees can be effectively applied to a variety of machine learning problems.

# Inductive Logic Programming (ILP)

**Inductive Logic Programming (ILP)** is a subfield of machine learning that focuses on learning logic-based models, such as **first-order logic** rules, from examples. Unlike traditional machine learning algorithms that typically operate with propositional (flat) data, ILP operates with relational (structured) data, where learning takes place over sets of objects and their relationships.

ILP combines elements of **inductive learning** (learning from examples) with **logic programming**, allowing for the induction of rules that generalize over structured data. The output of an ILP system is typically a set of logical rules that can explain or predict unseen examples based on the provided input data.

##### **Notation and Definitions**

To better understand ILP, it’s essential to familiarize oneself with the notation and definitions used in logic programming and inductive learning:

* **Literals**: A literal is a basic statement or its negation. For example, human(X) is a literal, and ¬human(X) represents the negation.
* **Atoms**: An atom is a basic relation or predicate applied to arguments. For instance, likes(john, pizza) is an atom.
* **Clauses**: A clause is a disjunction (OR) of literals, which can be interpreted as a set of logical rules. A **Horn clause** is a special type of clause that is used in ILP, which consists of a head (a positive literal) and a body (a conjunction of literals).
* **Background Knowledge**: This is the set of predefined facts or rules that the ILP system has access to during learning.
* **Positive and Negative Examples**: In ILP, examples are given in terms of

positive and negative instances. A positive example satisfies the concept (target) being learned, whereas a negative example does not.

##### Key Components of ILP:

1. **Training Data**: Examples represented in a logical form, such as sets of facts or tuples.
2. **Hypotheses**: The learned rules or models that generalize the patterns in the

data.

1. **Background Knowledge**: Domain-specific facts or rules that provide context to the learning task.
2. **Target Concept**: The concept or relationship that the ILP system is tasked

to learn, typically expressed as a logical rule.

##### **A Generic ILP Algorithm**

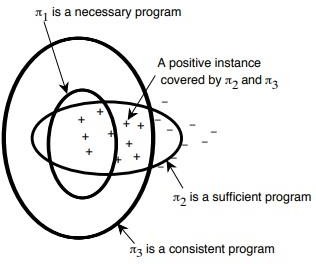
A generic ILP algorithm typically follows these steps:

Fig 7.1 Sufficient, Necessary, and Consistent Programs

##### **Input:**

* + A set of **positive examples**: Instances where the target concept is true.
  + A set of **negative examples**: Instances where the target concept is false.
  + **Background knowledge**: Domain-specific facts and rules that help guide the search for hypotheses.

1. **Hypothesis Space**: The space of potential hypotheses consists of logical rules that describe the target concept. Each hypothesis is a **logic clause** that relates to the given examples and background knowledge.
2. **Inductive Search**: The system searches for generalizations of the positive examples, starting from the most specific hypotheses (that only cover a small number of examples) and gradually generalizing. The algorithm uses various search strategies, such as **breadth-first search**, **depth-first search**, or **beam search**, to explore the hypothesis space.
3. **Refinement**: The candidate hypotheses are refined iteratively by adding or removing literals. For example, the system may start with a hypothesis that explains only a subset of the positive examples, and then incrementally add conditions (literals) to make the hypothesis cover more examples.
4. **Termination**: The process stops when an optimal hypothesis is found, or when a stopping condition is met (e.g., no further improvements can be made or the hypothesis reaches a predefined level of complexity).
5. **Output**: The final learned rule or set of rules that describe the target concept, such as a set of Horn clauses.

##### **Example of a Generic ILP Algorithm:**

For example, in the context of learning to predict whether a person is a "parent", the input could include background knowledge about family relationships (e.g., mother(X, Y) means X is the mother of Y), positive examples (e.g., parent(john)), and negative examples (e.g., ¬parent(mary)).

The ILP system could generate hypotheses such as:

* parent(X) :- mother(X, Y). This hypothesis suggests that if X is a mother of someone (Y), then X is a parent. The system can then refine this hypothesis by exploring more examples and relationships, such as considering fathers or additional background knowledge.

##### **An Example**

To better illustrate how ILP works, consider an example where the goal is to learn a rule for classifying animals based on their attributes. Suppose the system is provided with background knowledge about different animal species and their features (e.g., has\_wings(X) means X has wings, flies(X) means X flies, etc.), along with positive and negative examples of animals (e.g., eagle is a positive example, dog is a negative example).

##### Step-by-Step Example:

1. **Positive Example**: eagle(fly), sparrow(fly).
2. **Negative Example**: dog(no\_fly), cat(no\_fly).

##### Background Knowledge**:**

* + has\_wings(X) means X has wings.
  + flies(X) means X flies.

An ILP system might deduce the rule:

* flies(X) :- has\_wings(X). This rule indicates that if an animal has wings, it can fly.

##### **Inducing Recursive Programs**

One of the most powerful aspects of ILP is its ability to induce recursive logic. This is particularly useful when learning tasks involve hierarchical or recursive relationships, such as in natural language processing or reasoning tasks.

For instance, consider a recursive rule:

* ancestor(X, Y) :- parent(X, Y).
* ancestor(X, Y) :- parent(X, Z), ancestor(Z, Y).

In this case, an ancestor of Y can be either a direct parent or a parent of a parent (i.e., a grandparent). The recursive structure allows the system to generalize over chains of relationships and generate more complex rules.

##### **Choosing Literals to Add**

In ILP, the process of choosing which literals to add to a rule is crucial for refining hypotheses. Literals can be added based on their **utility** in increasing the hypothesis’s explanatory power. Some strategies for choosing literals include:

* **Entropy-based measures**: Where literals that reduce uncertainty the most are preferred.
* **Greedy search**: Adding literals that maximize information gain or reduce error in the current hypothesis.

Choosing literals effectively involves balancing **complexity** (keeping the model simple) with **accuracy** (fitting the data well).

##### **Relationships Between ILP and Decision Tree Induction**

ILP and **decision tree induction** share similarities in that both are used for supervised learning tasks, but they differ in their approach and output.

* **Decision Trees**: Decision trees learn a series of binary tests on features and generate a tree structure to make predictions.
* **ILP**: ILP, in contrast, generates logical rules or **Horn clauses** that describe patterns in the data. These rules are more general than decision tree splits, as they can represent more complex relationships.

However, the core similarity is that both ILP and decision trees search for patterns in data and output rules that can be used to classify new instances.

##### **Summary**

Inductive Logic Programming (ILP) is a powerful framework for learning logical rules from structured data. Key features include:

* **Relational data**: ILP works with structured data, where examples are not just individual instances but can involve relationships between entities.
* **Logic-based rules**: The output of ILP is typically a set of **logical rules** that explain patterns in the data.
* **Recursive rules**: ILP is capable of learning recursive and hierarchical relationships, making it suitable for more complex tasks.
* **Connection with decision trees**: While decision trees are simpler, ILP can represent more complex patterns through logical rules.

ILP is especially useful in domains where **background knowledge** and **structured data** are available, such as bioinformatics, natural language processing, and knowledge discovery.

## Conclusion

Machine learning has established itself as a pivotal field in artificial intelligence, empowering systems to learn from data and make decisions independently. By distinguishing between types of learning, such as supervised, unsupervised, and reinforcement learning, we can understand how different approaches suit a wide range of applications, from predictive modeling to complex decision-making.

Boolean functions and version spaces illustrate machine learning’s logical foundations, where algorithms form structured rules and iteratively refine hypotheses. Neural networks, particularly with advanced training techniques like backpropagation, have demonstrated exceptional capability in capturing complex, non-linear relationships, making them suitable for tasks that require deep pattern recognition.

Statistical learning methods offer robust tools for handling data variability and uncertainty, relying on probabilistic models and inference techniques that optimize decision-making under uncertain conditions. Moreover, the interpretability of models like decision trees and the logical structure of inductive logic programming (ILP) provide transparency in predictions and are invaluable in applications where understanding the model’s decision process is crucial.

Overall, the adaptability of machine learning makes it indispensable across diverse fields, allowing systems to learn continuously and respond to new information. This foundation supports further advancements and opens up possibilities for sophisticated, adaptive, and efficient AI-driven solutions across industries.

## Appendix A

#### INDUSTRIAL INTERNSHIP EVALUATION FORM

For the Students of B.Tech. (IT), Sasi Institute of Technology &Engineering, Tadepalligudem, West Godavari District, Andhra Pradesh

|  |  |
| --- | --- |
|  | **Date:** |
| **Name of the Intern** | **: Manukonda SunilVarma** |
| **Reg. No.** | **:22K65A1204** |
| **Branch** | **: Information Technology** |
| **Internship Offered** | **:** From **2024 - 25** |

Evaluate this student intern on the following parameters by checking the appropriate attributes.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Evaluation Parameters** | **Attributes**  **Give Your Feedback with Tick Mark (**√ **)** | | | | |
| **Excellent** | **Very Good** | **Good** | **Satisfactory** | **Poor** |
| **Attendance**  (Punctuality) |  |  |  |  |  |
| **Productivity**  (Volume, Promptness) |  |  |  |  |  |
| **Quality of Work**  (Accuracy, Completeness, Neatness) |  |  |  |  |  |
| **Initiative**  (Self-Starter, Resourceful) |  |  |  |  |  |
| **Attitude**  (Enthusiasm, Desire to Learn) |  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Interpersonal Relations**  (Cooperative, Courteous, Friendly) |  |  |  |  |  |
| **Ability to Learn**  (Comprehension of New Concepts) |  |  |  |  |  |
| **Use of Academic Training** (Applies Education to Practical Usage) |  |  |  |  |  |
| **Communications Skills**  (Written and Oral Expression) |  |  |  |  |  |
| **Judgement**  (Decision Making) |  |  |  |  |  |

***Please summarize. Your comments are especially helpful.***

Areas where student excels:

Areas where student needs to improve:

Areas where student gained new skills, insights, values, confidence, etc.

Was student’s academic preparation sufficient for this internship?

Additional comments or suggestions for the student:

|  |  |
| --- | --- |
| **Overall Evaluation of the Intern’s Performance**  (Evaluation Scale shown below) | **Points Awarded** |
|  |

**Evaluation Scale:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attributes** | Excellent | Very Good | Good | Satisfactory | Poor |
| **Points** |  |  |  |  |  |

|  |  |
| --- | --- |
| **Name of Officer In-Charge (Guide/Supervisor)** | **:** |
| **Designation** | **:** |
|  | **Signature of Officer In-charge**  **(Guide/Supervisor)** |

## Appendix B

**PO's and PSO's relevance with Internship Work**

|  |  |  |
| --- | --- | --- |
| **PO** | **Program outcomes** | **Relevance** |
| PO1 | **Engineering Knowledge**: Apply knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems | Applied basic knowledge of engineering to understand about entrepreneurship |
| PO2 | **Problem Analysis:** Identify, formulate research literature and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences. | Performed research in various ways to analyze problems and find a soution |
| PO3 | **Design/development of solutions**: Design solutions for complex engineering problems and design systems components or processes that meet specified need with appropriate consideration for public health and safety, cultural, societal and environmental considerations. | Able to understand the market strategies and problems in the society |
| PO4 | **Conduct investigations of complex problems**: Research based knowledge and research methods including design of experiments, analysis and interpretation of data and synthesis of information to provide valid conclusions. | Investigation of various problems of farmers |
| PO5 | **Modern tool usage**: Create, select and apply appropriate techniques, resources and modern engineering and it tools including prediction and modelling to complex engineering activities with an understanding of the limitations. | Used many of the Tremendous tools for Development Process |

|  |  |  |
| --- | --- | --- |
| PO6 | **The engineer and society**: Apply reasoning informed by contextual knowledge to asses societal, health, safety, legal and cultural issues and consequent responsibilities relevant to professional engineering practice | It can be Implemented in various real-world problems |
| PO7 | **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge and need for sustainable development | It can be beneficial to apply the knowledge in the environment with sustainable nature |
| PO8 | **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and  norms of the engineering practice. | Able to identify standard norms |
| PO9 | **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings. | It is an Individual/Team work that solves problem through technology |
| PO10 | **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions. | Prepared & documented summer internship report on Technology Entrepreneurship Program |
| PO11 | **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments. | It is a one-year training process conducted by Indian School of Business With heavy costing. |

|  |  |  |
| --- | --- | --- |
| PO12 | **Life-long learning:** Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change. | It is a endless learning procedure because entrepreneur should learn everyday from everything. |
| PSO1 | **Application Development** | An application that helps farmers |
| PSO2 | **Successful career and Entrepreneurship** | Prepared & documented summer internship report on Technology Entrepreneurship Program |