Kalinga University

Department of Computer Science & Information Technology

Course- BCAAIML
Subject- Machine Learning Basics
Subject Code – BCAAIML404 **Sem.- IV**

Unit 1

Algorithmic Models of Learning

Algorithmic learning models focus on defining and understanding how machines can learn patterns, make decisions, and improve their performance over time. The field combines elements of computer science, mathematics, and cognitive science.

1. Types of Algorithmic Learning

- **Supervised Learning**: Learning from labeled data. The algorithm learns a mapping from input (features) to output (labels).
  - Examples: Classification, Regression.
- **Unsupervised Learning**: Learning from unlabeled data to identify structure or patterns.
  - Examples: Clustering, Dimensionality Reduction.
- **Semi-supervised Learning**: Combines a small amount of labeled data with a large amount of unlabeled data.
- **Reinforcement Learning**: Agents learn to make decisions by receiving rewards or penalties for actions in an environment.
- **Online Learning**: Learning takes place sequentially, one instance at a time.
- **Batch Learning**: Learning occurs after accessing the entire dataset.

2. Key Concepts

- **Generalization**: The ability of the model to perform well on unseen data.
- **Overfitting**: When a model learns noise or irrelevant details in the training data, reducing its performance on new data.
- **Underfitting**: When a model is too simplistic to capture the underlying patterns in the data.
- **Bias-Variance Tradeoff**: Balancing the complexity of a model to optimize performance.

Learning Classifiers

A **classifier** is a model that categorizes data points into predefined classes. Classifiers are fundamental in supervised learning.

1. Types of Classifiers

- **Linear Classifiers**: Separate data using a linear boundary.
  - Examples: Logistic Regression, Support Vector Machines (Linear Kernel).
- **Non-Linear Classifiers**: Use complex boundaries for separation.
  - Examples: Decision Trees, Random Forests, Neural Networks.
- **Probabilistic Classifiers**: Output probabilities for each class.
  - Example: Naive Bayes.
- **Distance-Based Classifiers**: Classify based on proximity to examples.
  - Example: k-Nearest Neighbors (k-NN).

2. Performance Metrics

- **Accuracy**: Ratio of correctly predicted instances to total instances.
- **Precision**: Ratio of true positives to predicted positives.
- **Recall (Sensitivity)**: Ratio of true positives to actual positives.
- **F1 Score**: Harmonic mean of Precision and Recall.
- **Confusion Matrix**: A tabular representation of model predictions vs. true outcomes.

3. Techniques for Building Classifiers

- Feature Engineering: Selecting, transforming, or creating features to improve model performance.
- Feature Selection: Choosing the most relevant features to simplify the model.
- Hyperparameter Tuning: Optimizing the parameters controlling model behavior.

Learning Functions

Learning functions involves finding a mathematical mapping between inputs and outputs. This is foundational to supervised learning.

1. Types of Functions

- **Linear Functions**: Simpler models, e.g., $y=mx+by = mx + by=mx+b$.
- **Non-Linear Functions**: Used for complex relationships, e.g., polynomial or exponential functions.
- **Piecewise Functions**: Functions that operate differently over distinct ranges of input.
  - Example: Decision Tree.

2. Applications

- Regression Problems: Predicting continuous outputs like house prices or temperatures.
- Classification Problems: Output is binary or categorical.

Learning Relations

Learning relations focuses on identifying relationships or dependencies between variables in data.

1. Relation Learning Techniques

- **Association Rule Learning**: Extracts rules showing relationships among items in datasets.
    - Example: Market Basket Analysis (e.g., "If a customer buys bread, they are likely to buy butter").
- **Graph-Based Methods**: Represent relations as nodes and edges in a graph.
    - Example: Social Network Analysis.
- **Matrix Factorization**: Common in recommendation systems, where interactions are modeled as matrices.
    - Example: Singular Value Decomposition (SVD).

2. Applications

- Predicting associations in large databases.
- Modeling relationships in relational databases or graphs.

Learning Grammars

Grammar learning involves inferring the rules or syntax governing a language or structured data.

1. Types of Grammars

- **Context-Free Grammar (CFG)**: Rules that describe the syntax of a language. Commonly used in compilers.
- **Regular Grammar**: Simpler than CFG, used in text processing or finite automata.
- **Dependency Grammar**: Focuses on the relationships between words in a sentence.

2. Grammar Induction

Grammar induction is the process of learning grammar from data. This is often applied in:

- Natural Language Processing (NLP): Parsing sentences or generating language.
- Sequence Modeling: Understanding time-series or structured data sequences.

3. Applications

- Parsing in compilers.
- Speech recognition systems.
- Machine translation and text summarization.

Probabilistic Models

Probabilistic models are frameworks that incorporate uncertainty by using probability theory to model complex systems, data, or processes.

1. Definition

A probabilistic model uses probability distributions to represent relationships between variables and their uncertainties. It accounts for randomness and noise in data.

2. Types of Probabilistic Models

- **Bayesian Networks**: Directed acyclic graphs where nodes represent variables and edges denote probabilistic dependencies.
  - Example: Disease diagnosis systems modeling symptoms and diseases.
- **Markov Models**:
  - **Markov Chains**: Models a sequence of states with the Markov property (future states depend only on the current state).
  - **Hidden Markov Models (HMMs)**: Incorporates hidden states that generate observable sequences (common in speech recognition).
- **Gaussian Models**:
  - **Gaussian Mixture Models (GMMs)**: Models data as a mixture of multiple Gaussian distributions.
  - **Multivariate Gaussian Models**: Used for continuous data with multiple dimensions.

3. Applications

- Speech and language processing (e.g., HMMs in speech recognition).
- Financial risk modeling.
- Predictive analytics in healthcare and marketing.

Value Functions

Value functions are critical concepts in **Reinforcement Learning (RL)** and are used to evaluate the expected future rewards for an agent in a given state.

1. Definition

- A value function estimates the total cumulative reward an agent can expect from a given state or state-action pair.

2. Types of Value Functions

- **State-Value Function (V(s)V(s)V(s))**: Measures the expected reward starting from state sss, under a policy π\piπ:

$$V(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_t \mid S_0 = s \right]$$

Action-Value Function (Q(s,a)Q(s, a)Q(s,a)): Measures the expected reward from taking action aaa in state sss under policy π\piπ:

$$Q(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_t \mid S_0 = s, A_0 = a \right]$$

- **Advantage Function (A(s,a)A(s, a)A(s,a))**: Indicates the relative benefit of taking action aaa in state sss: A(s,a)=Q(s,a)−V(s)A(s, a) = Q(s, a) - V(s)A(s,a)=Q(s,a)−V(s)

3. Applications

- Reinforcement Learning: Policy optimization and decision-making.
- Game Theory: Evaluating strategic advantages.
- Robotics: Navigating environments by balancing rewards and risks.

Behaviors

Behavior modeling involves understanding and simulating the actions or decisions of agents, whether humans, animals, or machines, in specific environments.

1. Types of Behavior Modeling

- **Rule-Based Behavior**: Defined by fixed rules or logic.
  - Example: Expert systems.
- **Learning-Based Behavior**: Behavior emerges through training (e.g., reinforcement learning agents).
- **Probabilistic Behavior**: Actions are determined by probabilities rather than fixed decisions.
  - Example: Softmax action selection in RL.

2. Key Techniques

- **Behavior Cloning**: Using supervised learning to replicate observed behavior from demonstrations.

- **Inverse Reinforcement Learning (IRL)**: Inferring the reward function that explains an agent's observed behavior.
- **Stochastic Behavior Models**: Define actions probabilistically using distributions (e.g., Boltzmann models).

3. Applications

- Robotics: Simulating human-like navigation.
- Video Games: AI behaviors for non-player characters (NPCs).
- Autonomous Systems: Modeling real-world decision-making under uncertainty.

Programs for Experience

Programs for experience are systems or algorithms designed to learn and adapt from interactions or exposure to environments.

1. Key Concepts

- **Experience Replay**: A technique in reinforcement learning where past interactions are stored in memory and reused for learning.
  - Advantages: Improves sample efficiency and avoids forgetting.
- **Curriculum Learning**: Training the model on simpler tasks initially and progressively increasing complexity.
- **Exploration vs. Exploitation**: Balancing between exploring new actions (gathering more experience) and exploiting known actions for rewards.

2. Implementation Strategies

- **Reinforcement Learning Algorithms**:
  - **Q-Learning**: Off-policy algorithm to learn the action-value function.
  - **Deep Q-Learning**: Uses neural networks to approximate Q(s,a)Q(s, a)Q(s,a).
  - **Policy Gradient Methods**: Directly optimize policies based on experience.
- **Simulation Environments**:
  - Tools like OpenAI Gym or Unity ML-Agents simulate experiences for training agents.

3. Applications

- Autonomous Vehicles: Gaining experience in simulated driving environments.
- Robotics: Learning control strategies via trial and error.
- Gaming: Training agents to improve in complex, dynamic scenarios.

Conclusion

- **Probabilistic Models** allow systems to deal with uncertainty and make informed predictions.

- **Value Functions** are the backbone of reinforcement learning, enabling agents to evaluate and optimize long-term rewards.
- **Behavior Models** simulate or predict decision-making processes in agents.
- **Programs for Experience** focus on designing algorithms that learn from interactions with environments, ensuring adaptability and continuous improvement.

Bayesian Framework

The **Bayesian framework** is a probabilistic approach to modeling and decision-making that incorporates prior knowledge or beliefs with observed data.

Key Concepts

- **Bayes' Theorem**:
  - $$P(H|D) = \frac{P(D|H) \cdot P(H)}{P(D)}$$
    - P(H|D)P(H|D)P(H|D): Posterior probability (belief in HHH after observing data DDD).
    - P(D|H)P(D|H)P(D|H): Likelihood (probability of data DDD given hypothesis HHH).
    - P(H)P(H)P(H): Prior probability (belief in HHH before observing DDD).
    - P(D)P(D)P(D): Evidence (normalizing constant).

Components

1. **Prior (P(H)P(H)P(H))**:
   - Represents initial knowledge or assumptions about a hypothesis.
   - Example: In spam detection, a prior might reflect that 80% of emails are non-spam.
2. **Likelihood (P(D|H)P(D|H)P(D|H))**:
   - Probability of observing the data given the hypothesis.
3. **Posterior (P(H|D)P(H|D)P(H|D))**:
   - Updated belief in the hypothesis after observing data.
4. **Evidence (P(D)P(D)P(D))**:
   - Ensures posterior probabilities sum to 1.

Applications

- **Machine Learning**: Bayesian networks, Gaussian processes.
- **Medical Diagnosis**: Combining prior medical history with test results.
- **Natural Language Processing (NLP)**: Predicting word sequences using prior linguistic models.

---

2. Maximum A Posteriori (MAP)

The **Maximum A Posteriori (MAP)** framework extends Bayesian inference by selecting the hypothesis that maximizes the posterior probability.

Definition

The MAP estimate is given by:

$$H^{MAP} = \arg\max_H P(H|D)$$

Using Bayes' theorem:

$$P(H|D) = \frac{P(D|H) \cdot P(H)}{P(D)}$$

Since P(D)P(D)P(D) is constant, the MAP estimate simplifies to:

$$H^{MAP} = \arg\max_H [P(D|H) \cdot P(H)]$$

Key Features

- Incorporates prior knowledge (P(H)P(H)P(H)).
- Balances likelihood and prior information.
- Especially useful when data is scarce, as it avoids overfitting.

Comparison to Maximum Likelihood Estimation (MLE)

- **MAP** considers both prior and likelihood:

$$^{MAP} = \arg\max_H [P(D|H) \cdot P(H)]$$

MLE ignores prior and maximizes only the likelihood:

$$H^{MLE} = \arg\max_H P(D|H)$$

Applications

- **Spam Classification**: Using prior probabilities of spam and non-spam emails.
- **Computer Vision**: Incorporating prior knowledge of object shapes or sizes.
- **Robotics**: Estimating robot positions using sensor data and prior maps.

Minimum Description Length (MDL) Framework

The **Minimum Description Length (MDL)** framework is a principle for model selection based on information theory. It favors models that explain data with the shortest overall description.

Key Idea

- Inspired by **Occam's Razor**: Prefer simpler models that sufficiently explain the data.
- Encodes both:
    1. The model (hypothesis).
    2. The data given the model.

Formal Definition

For a hypothesis HHH, the total description length is:

$$L(H)+L(D|H)$$

- L(H): Length of encoding the hypothesis.
- L(D|H)L(D|H)L(D|H): Length of encoding the data given the hypothesis.

The goal is to find the model HHH that minimizes:

$$H^MDL=argminH[L(H)+L(D|H)]$$

Advantages

- Balances model complexity and fit to data.
- Penalizes overly complex models to avoid overfitting.
- Makes no assumptions about prior distributions.

Comparison to Bayesian Framework

- Similar to MAP but rooted in coding theory rather than probability.
- **MAP** requires prior probability distributions; **MDL** replaces priors with encoding lengths.

Applications

- **Model Selection**: Choosing the best regression model or neural network architecture.
- **Data Compression**: Optimizing encoding strategies for storage.
- **Anomaly Detection**: Identifying patterns that deviate from a compact model.

Summary of Frameworks

Framework	Key Principle	Focus	Strength	Example Application
Bayesian	Updates beliefs using prior and data	Posterior probability	Handles uncertainty well	Medical diagnosis
MAP	Maximizes posterior probability	Prior + Likelihood	Incorporates prior knowledge	Spam filtering
MDL	Minimizes total encoding length	Model simplicity + Data fit	Penalizes overfitting	Model selection