UNIT – II – MACHINE LEARNING ALGORITHMS

~by Avinash R. Kauran

Machine Learning Algorithms: Fundamentals and Applications

Introduction

Machine learning algorithms form the backbone of modern artificial intelligence systems. This lecture covers six fundamental algorithms and techniques that are essential for any machine learning practitioner. We'll explore their principles, applications, advantages, and limitations.

1. Decision Trees and Random Forests

Decision Trees
Decision trees are hierarchical structures that make decisions by following a path from root to leaf nodes. They work by splitting data based on features that provide the most information gain.

Key Concepts:
* Information Gain and Entropy: Measures the reduction in uncertainty after splitting data
* Gini Impurity: Alternative to entropy, measures the probability of incorrect classification
* Pruning: Technique to prevent overfitting by removing unnecessary branches

Random Forests
Random forests are ensemble models composed of multiple decision trees. They improve upon single decision trees through:

* Bagging (Bootstrap Aggregating): Training trees on random subsets of data
* Feature Randomization: Considering random subsets of features at each split
* Voting/Averaging: Combining predictions from multiple trees

Advantages:
- Robust against overfitting
- Can handle missing values
- Provides feature importance rankings

Limitations:
- Can be computationally intensive
- Less interpretable than single decision trees
- May require significant memory for large datasets

2. Support Vector Machines (SVM)

SVMs are powerful algorithms that find the optimal hyperplane to separate classes in high-dimensional space.

Core Concepts:

1. Maximum Margin Hyperplane
   - Finds the boundary that maximizes distance between classes
   - Support vectors are the closest points to the boundary

2. Kernel Trick

- Maps data to higher dimensions where it becomes linearly separable
- Common kernels: Linear, Polynomial, RBF (Radial Basis Function)

3. Soft Margin
- Allows for some misclassification to prevent overfitting
- Controlled by regularization parameter C

Applications:
- Text classification
- Image recognition
- Bioinformatics

3. Artificial Neural Networks (ANN)

ANNs are inspired by biological neural networks and consist of interconnected nodes organized in layers.

Architecture:

1. Input Layer
- Receives raw features
- Normalizes/standardizes data

2. Hidden Layers
- Perform non-linear transformations
- Extract hierarchical features
- Deep learning involves multiple hidden layers

3. Output Layer
- Produces final predictions
- Uses activation functions based on task type

Key Components:

* Activation Functions
- ReLU (Rectified Linear Unit)
- Sigmoid
- Tanh
- Softmax (for classification)

* Backpropagation
- Updates weights based on prediction errors
- Uses chain rule to compute gradients

* Learning Rate
- Controls step size during optimization
- Critical for convergence

4. Ensemble Learning: Boosting and Bagging

Ensemble methods combine multiple models to create a stronger predictor.

Bagging (Bootstrap Aggregating)
- Trains models on random subsets of data
- Models vote or average predictions
- Reduces variance without increasing bias
- Example: Random Forests

Boosting
- Trains models sequentially
- Each model focuses on previous models' mistakes
- Popular algorithms:
  * AdaBoost
  * Gradient Boosting (GBM)
  * XGBoost
  * LightGBM

5. K-Nearest Neighbors (K-NN)

K-NN is a simple but effective instance-based learning algorithm.

Working Principle:
1. Store all training examples
2. For new instances:
   - Calculate distance to all training examples
   - Find K nearest neighbors
   - Vote/average for prediction

Considerations:
- Choice of K
- Distance metrics (Euclidean, Manhattan, etc.)
- Feature scaling importance
- Curse of dimensionality

Advantages:
- Simple to understand and implement
- No training phase
- Naturally handles multi-class problems

Limitations:
- Computationally expensive for large datasets
- Sensitive to irrelevant features
- Requires feature scaling

6. Gradient Descent for Optimization

Gradient descent is a fundamental optimization algorithm used in many machine learning models.

Variants:

1. Batch Gradient Descent
   - Uses entire dataset for each update
   - Computationally expensive but stable

2. Stochastic Gradient Descent (SGD)
   - Updates parameters using single examples
   - Faster but noisier convergence

3. Mini-batch Gradient Descent
   - Compromise between batch and SGD
   - Most commonly used in practice

Key Concepts:

* Learning Rate Scheduling
  - Adaptive learning rates
  - Learning rate decay
  - Momentum

* Optimization Algorithms
  - Adam
  - RMSprop
  - Adagrad

Summary and Best Practices

When choosing an algorithm, consider:
1. Data size and dimensionality
2. Type of problem (classification, regression, etc.)
3. Computational resources
4. Interpretability requirements
5. Training time constraints

Practical Implementation Tips

For each algorithm:
1. Start with default parameters
2. Use cross-validation for parameter tuning
3. Monitor for overfitting
4. Consider computational trade-offs
5. Validate assumptions about your data

Remember to always split your data into training, validation, and test sets to ensure robust evaluation of your models.