Complete Guide to Machine Learning Algorithms

Classification Algorithms

1. Logistic Regression

How it works:

1. **Linear Combination**: Creates a linear combination of input features: $z = \beta_0 + \beta_1x_1 + \beta_2x_2 + ... + \beta_nx_n$
2. **Sigmoid Function**: Applies sigmoid function to map z to probability: p = 1/(1 + e^(-z))
3. **Decision Boundary**: Classifies based on threshold (usually 0.5)
4. **Cost Function**: Uses log-likelihood as cost function
5. **Optimization**: Uses gradient descent to minimize cost and find optimal coefficients

Step-by-step process:

- Initialize random weights
- Calculate predicted probabilities using sigmoid
- Compute cost using cross-entropy loss
- Calculate gradients
- Update weights using gradient descent
- Repeat until convergence

2. Decision Trees

How it works:

1. **Root Selection**: Start with entire dataset at root
2. **Feature Selection**: Choose best feature to split on using criteria like Gini impurity or entropy
3. **Splitting**: Split data based on feature threshold that maximizes information gain
4. **Recursive Process**: Repeat splitting process for each subset
5. **Stopping Criteria**: Stop when max depth reached, minimum samples met, or pure nodes achieved
6. **Prediction**: Follow path from root to leaf based on feature values

Step-by-step process:

- Calculate impurity measure for current node
- For each feature, find best split point
- Choose split that maximizes information gain
- Create child nodes and distribute data
- Recursively apply to child nodes
- Assign class label to leaf nodes

3. Random Forest

How it works:

1. **Bootstrap Sampling**: Create multiple bootstrap samples from training data
2. **Feature Randomness**: For each tree, randomly select subset of features at each split
3. **Tree Building**: Build decision trees using bootstrap samples and random features
4. **Ensemble**: Combine predictions from all trees
5. **Voting**: Use majority voting for final classification

Step-by-step process:

- Generate B bootstrap samples
- For each sample, build decision tree with random feature selection
- Train all trees independently
- For prediction, pass input through all trees
- Aggregate predictions using majority vote

4. Support Vector Machine (SVM)

How it works:

1. **Hyperplane**: Find optimal hyperplane that separates classes with maximum margin
2. **Support Vectors**: Identify data points closest to hyperplane (support vectors)
3. **Margin Maximization**: Maximize distance between hyperplane and nearest points
4. **Kernel Trick**: Use kernel functions for non-linear separation
5. **Optimization**: Solve quadratic optimization problem

Step-by-step process:

- Transform data using kernel function (if needed)
- Set up optimization problem to maximize margin
- Identify support vectors
- Calculate optimal weights and bias
- Use decision function for classification

5. K-Nearest Neighbors (KNN)

How it works:

1. **Distance Calculation**: Calculate distance between query point and all training points
2. **Neighbor Selection**: Select K nearest neighbors based on distance metric
3. **Voting**: Use majority voting among K neighbors for classification
4. **Prediction**: Assign class label based on majority vote

Step-by-step process:

- Store all training data
- For new point, calculate distances to all training points
- Sort distances and select K nearest neighbors
- Count class labels among K neighbors
- Assign most frequent class label

6. Naive Bayes

How it works:

1. **Bayes' Theorem**: Apply P(A|B) = P(B|A) × P(A) / P(B)
2. **Independence Assumption**: Assume features are conditionally independent
3. **Prior Probability**: Calculate prior probability for each class
4. **Likelihood**: Calculate likelihood of features given each class
5. **Posterior**: Calculate posterior probability for each class
6. **Classification**: Choose class with highest posterior probability

Step-by-step process:

- Calculate prior probabilities for each class
- For each feature, calculate likelihood given each class
- For new instance, multiply prior by all feature likelihoods
- Normalize to get posterior probabilities
- Classify as class with highest posterior

Regression Algorithms

1. Linear Regression

How it works:

1. **Linear Relationship**: Assumes linear relationship between features and target
2. **Best Fit Line**: Finds line that minimizes sum of squared residuals
3. **Normal Equation**: Uses mathematical formula or gradient descent for optimization
4. **Prediction**: Uses linear equation $y = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n$

Step-by-step process:

- Initialize weights randomly
- Calculate predictions using linear equation
- Compute mean squared error
- Calculate gradients
- Update weights using gradient descent
- Repeat until convergence

2. Polynomial Regression

How it works:

1. **Feature Engineering**: Transform features to polynomial terms
2. **Linear Model**: Apply linear regression to polynomial features
3. **Degree Selection**: Choose polynomial degree based on complexity vs accuracy trade-off
4. **Regularization**: Often requires regularization to prevent overfitting

Step-by-step process:

- Generate polynomial features (x, x^2, x^3, etc.)
- Apply linear regression to expanded feature set
- Use cross-validation to select optimal degree
- Train model with chosen degree
- Make predictions using polynomial equation

3. Ridge Regression

How it works:

1. **L2 Regularization**: Adds penalty term $\lambda\sum\beta_i^2$ to cost function
2. **Shrinkage**: Shrinks coefficients toward zero
3. **Bias-Variance Trade-off**: Increases bias to reduce variance
4. **Hyperparameter**: Tune λ using cross-validation

Step-by-step process:

- Add L2 penalty to mean squared error
- Solve modified optimization problem
- Coefficients shrink proportionally to λ
- Use cross-validation to find optimal λ
- Train final model with chosen λ

4. Lasso Regression

How it works:

1. **L1 Regularization**: Adds penalty term $\lambda\sum|\beta_i|$ to cost function
2. **Feature Selection**: Can set some coefficients exactly to zero
3. **Sparsity**: Produces sparse models with fewer features
4. **Optimization**: Uses coordinate descent or specialized algorithms

Step-by-step process:

- Add L1 penalty to mean squared error
- Use coordinate descent to optimize
- Some coefficients become exactly zero
- Automatically performs feature selection
- Tune λ using cross-validation

5. Support Vector Regression (SVR)

How it works:

1. **ε-insensitive Loss**: Ignores errors within ε-tube around prediction
2. **Support Vectors**: Uses points outside ε-tube for model
3. **Kernel Functions**: Can handle non-linear relationships
4. **Margin Maximization**: Finds function with maximum margin around predictions

Step-by-step process:

- Define ε-tube around target values
- Identify support vectors (points outside tube)
- Solve quadratic optimization problem
- Apply kernel transformation if needed
- Use support vectors for prediction

Clustering Algorithms

1. K-Means Clustering

How it works:

1. **Initialization**: Randomly place K centroids in feature space
2. **Assignment**: Assign each point to nearest centroid
3. **Update**: Move centroids to center of assigned points
4. **Iteration**: Repeat assignment and update until convergence
5. **Convergence**: Stop when centroids no longer move significantly

Step-by-step process:

- Choose number of clusters K
- Initialize K centroids randomly
- Assign each point to closest centroid
- Calculate new centroid positions
- Repeat until centroids stabilize
- Final clusters based on centroid assignments

2. Hierarchical Clustering

How it works:

1. **Distance Matrix**: Calculate pairwise distances between all points
2. **Initialization**: Start with each point as its own cluster
3. **Merging**: Repeatedly merge closest clusters
4. **Linkage Criteria**: Use single, complete, or average linkage
5. **Dendrogram**: Creates tree structure showing cluster hierarchy
6. **Cutting**: Cut dendrogram at desired number of clusters

Step-by-step process:

- Calculate distance matrix
- Initialize each point as separate cluster
- Find pair of closest clusters
- Merge closest clusters
- Update distance matrix
- Repeat until single cluster or desired number reached

3. DBSCAN (Density-Based Clustering)

How it works:

1. **Core Points**: Points with at least MinPts neighbors within ε distance
2. **Border Points**: Non-core points within ε of core points
3. **Noise Points**: Points that are neither core nor border
4. **Cluster Formation**: Core points and their neighbors form clusters
5. **Density Connection**: Clusters connected through density-reachable points

Step-by-step process:

- For each point, count neighbors within ε distance
- Classify points as core, border, or noise
- Start cluster with unvisited core point
- Add all density-reachable points to cluster
- Repeat for all unvisited core points
- Noise points remain unclassified

4. Gaussian Mixture Models (GMM)

How it works:

1. **Mixture Components**: Assumes data comes from mixture of Gaussian distributions
2. **Parameters**: Estimates mean, covariance, and mixing coefficients
3. **EM Algorithm**: Uses Expectation-Maximization for parameter estimation
4. **Soft Assignment**: Points have probabilistic membership in clusters
5. **Model Selection**: Use criteria like AIC/BIC for number of components

Step-by-step process:

- Initialize parameters for K Gaussian components
- E-step: Calculate probability of each point belonging to each component
- M-step: Update parameters based on weighted assignments
- Repeat E-step and M-step until convergence
- Assign points to most probable cluster

5. Mean Shift

How it works:

1. **Kernel Density**: Estimates probability density using kernel functions
2. **Mode Seeking**: Finds local maxima of density function
3. **Gradient Ascent**: Moves points toward higher density regions
4. **Convergence**: Points converge to density modes (cluster centers)
5. **Automatic K**: Automatically determines number of clusters

Step-by-step process:

- Place kernel at each data point
- Calculate density at each point
- For each point, find direction of steepest density increase
- Move point in direction of gradient
- Repeat until convergence to density modes
- Group points that converge to same mode

Key Considerations

Algorithm Selection Criteria:

- **Dataset Size**: Some algorithms scale better with large datasets
- **Feature Dimensionality**: Curse of dimensionality affects some algorithms more
- **Interpretability**: Tree-based methods more interpretable than neural networks
- **Computational Resources**: Training and prediction time requirements
- **Data Distribution**: Assumptions about data distribution and relationships
- **Performance Requirements**: Accuracy vs speed trade-offs

Common Preprocessing Steps:

1. **Data Cleaning**: Handle missing values and outliers
2. **Feature Scaling**: Normalize or standardize features
3. **Feature Engineering**: Create new features from existing ones
4. **Feature Selection**: Remove irrelevant or redundant features
5. **Data Splitting**: Divide into training, validation, and test sets

Evaluation Metrics:

- **Classification**: Accuracy, precision, recall, F1-score, ROC-AUC
- **Regression**: MSE, RMSE, MAE, R-squared
- **Clustering**: Silhouette score, inertia, adjusted rand index

This comprehensive guide covers the fundamental machine learning algorithms across all three main categories, providing both theoretical understanding and practical implementation insights.