Lecture 4.2: Non-parametric method

谢丹
清华大学数学系

October 13, 2025

Parametric vs. Non-parametric Methods

From assumption-based to data-driven approaches

Parametric Methods

- **Discriminant functions**: MSE, Perceptron, SVM
- **Generative models**: LDA, QDA, Naive Bayes
- **Discriminative functions**: Logistic Regression

Model structure is fixed with parameters $\mathbf{w}$
Training optimizes $E(\mathbf{w})$ (possibly with constraints, e.g., SVM)

Non-parametric Methods

- ***k*-Nearest Neighbors (KNN)**
- **Decision Trees**
- **Random Forests**

No strong assumptions about functional form
Model complexity grows with data size

What is K-Nearest Neighbors?

Core Idea

- **Simple, intuitive** machine learning algorithm
- Used for both **classification** and **regression**
- **Instance-based** learning (lazy learning)
- "Tell me who your neighbors are, and I'll tell you who you are"

Example

Classifying a new fruit: Look at the 3 most similar fruits you already know. If 2 are apples and 1 is orange, guess it's an apple.

The K-NN Algorithm Step-by-Step

1. Choose the number of neighbors **K**
2. Calculate distances to all training points
3. Identify the K nearest neighbors
4. Poll the neighbors (majority vote for classification)
5. Make prediction

Note: The K here is not the number of classes, and is a choice.

Distance Metrics

Euclidean Distance

$$d = \sqrt{\sum_{i=1}^{D}(x_i - y_i)^2}$$

Most common, straight-line distance

Manhattan Distance

$$d = \sum_{i=1}^{D}|x_i - y_i|$$

Grid-like distance

Minkowski Distance

$$d = \left(\sum_{i=1}^{D}|x_i - y_i|^p\right)^{1/p}$$

Generalized form

Hamming Distance

For categorical data - counts differing positions

The Bias-Variance Trade-off

Small K (K=1)

- **Low bias**
- **High variance**
- Complex decision boundary
- Overfitting

Large K (K=99)

- **High bias**
- **Low variance**
- Smooth decision boundary
- Underfitting

Practical K Selection

Guidelines for Choosing K

- Typically use **odd numbers** to avoid ties
- Common starting point: $K = \sqrt{n}$ where n is sample size
- Use **cross-validation** to find optimal K
- Consider **dataset size** and **noise level**

Example

For a dataset with 1000 samples: $\sqrt{1000} \approx 32$, so try K=31, 33, 35, etc.

Classification vs Regression

Classification

Majority Vote

- Each neighbor votes for its class
- Most frequent class wins
- Can use weighted voting

Example: K=5, votes: [A, A, B, A, B] → Prediction: A

Regression

Weighted Average

- Average of neighbors' values
- Can use distance-weighted average
- Closer neighbors have more influence

Example: K=3, values: [10, 12, 11] → Prediction: 11

Feature Scaling Importance

Critical Step!

Features must be scaled before applying K-NN

Example

Without Scaling:

- Salary: 30,000-100,000
- Age: 20-70
- Salary dominates distance!

Example

With Scaling:

- Standardization
- Normalization
- All features contribute equally

Common scaling methods: StandardScaler, MinMaxScaler, RobustScaler

Pros and Cons

Advantages

- **Simple** to understand and implement
- **No training phase** - fast "training"
- **Adapts easily** to new data
- **Versatile** - classification and regression
- **No assumptions** about data distribution

Disadvantages

- **Computationally expensive** for prediction
- **Sensitive to irrelevant features**
- **Curse of dimensionality**
- **Memory intensive** - stores all data
- **Sensitive to outliers**

What are Decision Trees?

Definition

A **non-parametric supervised learning** method that learns simple decision rules from data features to predict target variables.

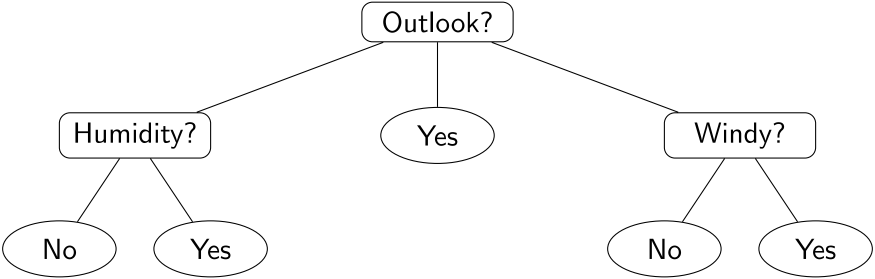


Figure: Example: Should we play tennis?

Tree Anatomy

Components

- **Root Node**: First feature test
- **Internal Nodes**: Intermediate decisions
- **Branches**: Outcomes of tests
- **Leaves**: Final classifications

Key Concepts

- **Recursive partitioning**
- **Purity measures**
- **Stopping criteria**

Measuring Node Impurity

Goal

Find splits that minimize **purity** in child nodes.

Gini Impurity

$$I_G(p) = 1 - \sum_{i=1}^{K} p_i^2$$

- Range: [0, 0.5] for binary
- Favors larger partitions

Entropy

$$I_E(p) = - \sum_{i=1}^{K} p_i \log_2 p_i$$

- Information theory basis
- Measures disorder

p_i is the probability of class i, which is defined as

$$p_i = \frac{N_i}{N}$$

Impurity Calculation for Splitting

Averaging impurities across child nodes

The overall impurity for a split is computed as the weighted average of child node impurities:

$$I_{\text{split}} = \frac{N_{\text{left}}}{N} \cdot I_{\text{left}} + \frac{N_{\text{right}}}{N} \cdot I_{\text{right}}$$

where:

- $I_{\text{left}}, I_{\text{right}}$: impurity measures for left and right child nodes
- $N_{\text{left}}, N_{\text{right}}$: number of samples in each child node
- $N = N_{\text{left}} + N_{\text{right}}$: total number of samples

Classification and Regression Trees (CART)

Algorithm Outline

1. Start with all data at root node
2. For each feature, find best split threshold
3. Choose feature with minimal impurity
4. Recursively split child nodes
5. Stop when stopping criteria met

The decision at the leaf node is given by the class with maximal probabilities.

Feature Type Handling

Numerical Features

- **Binary splits**: $x_j \leq t$ vs $x_j > t$
- Sort values, test midpoints
- Efficient: $O(n \log n)$ per feature

Categorical Features

- **Binary**: category $\in S$ vs $\notin S$
- **Multi-way**: one branch per category
- Can cause overfitting with high cardinality

When to Stop Splitting?

Pre-Pruning (Early Stopping)

- Maximum tree depth
- Minimum samples per leaf
- Minimum samples to split
- Minimum impurity decrease

Typical Values

- max_depth: 3-10
- min_samples_leaf: 1-20
- min_samples_split: 2-20

Post-Pruning (Cost Complexity)

Minimize:

$$R_\alpha(T) = R(T) + \alpha|T|$$

where:

- $R(T)$: misclassification rate
- $|T|$: number of leaves
- α: complexity parameter

Process

1. Grow full tree
2. Collapse nodes greedily
3. Choose α via CV

Tennis Playing Example

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Root Node Calculation

Parent: 9 Yes, 5 No $\rightarrow$ Gini $= 1 - (9/14)^2 - (5/14)^2 = 0.459$

Split Calculations

Outlook Split

- Sunny: [2 Yes, 3 No] $\rightarrow$ Gini $= 0.48$
- Overcast: [4 Yes, 0 No] $\rightarrow$ Gini $= 0$
- Rainy: [3 Yes, 2 No] $\rightarrow$ Gini $= 0.48$
- Weighted: $\frac{5}{14} \times 0.48 + \frac{4}{14} \times 0 + \frac{5}{14} \times 0.48 = 0.343$

Windy Split

- False: [6 Yes, 2 No] $\rightarrow$ Gini $= 0.375$
- True: [3 Yes, 3 No] $\rightarrow$ Gini $= 0.5$
- Weighted: $\frac{8}{14} \times 0.375 + \frac{6}{14} \times 0.5 = 0.429$

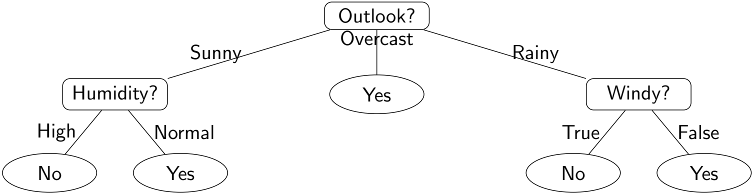
Humidity Split

- High: [3 Yes, 4 No] $\rightarrow$ Gini $= 0.490$
- Normal: [6 Yes, 1 No] $\rightarrow$ Gini $= 0.245$
- Weighted: $\frac{7}{14} \times 0.490 + \frac{7}{14} \times 0.245 = 0.367$

Best Split

Outlook has minimal inpurities $\rightarrow$ Split first!

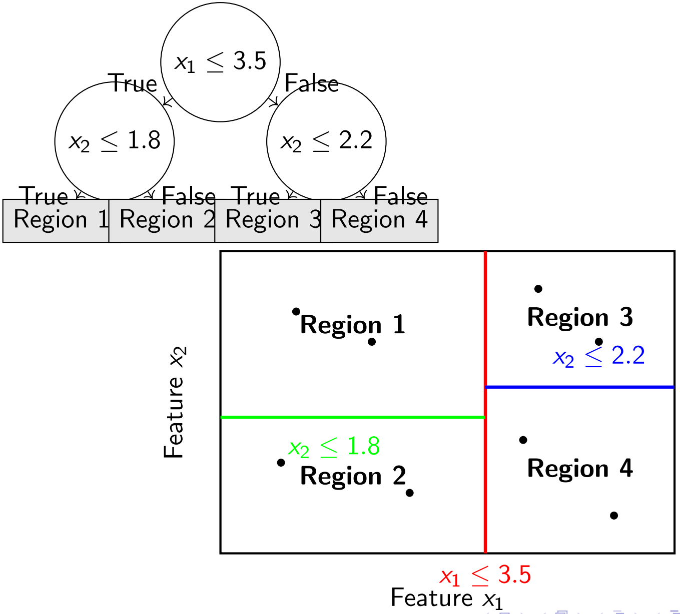
Final Tennis Decision Tree



Interpretation

- If Overcast → Always play
- If Sunny → Check humidity
- If Rainy → Check wind

For continuous variables, the feature space is separated into various regions.



Pros and Cons

Advantages

- **Interpretable**: Easy to explain
- **Few assumptions**: Handles mixed data
- **Non-parametric**: No distribution assumptions
- **Feature selection**: Built-in importance
- **Robust**: Handles outliers well

Limitations

- **High variance**: Unstable to small changes
- **Overfitting**: Complex trees don't generalize
- **Greedy**: Local optima
- **Axis-aligned**: Poor for diagonal boundaries
- **Biased**: Favors features with more levels

Solution: Ensemble Methods

Random Forests and Gradient Boosting overcome these limitations!

Random Forest: Intuition

The Wisdom of Crowds

- **Ensemble method** combining multiple decision trees
- Each tree trained on different data subsets and feature subsets
- Final prediction: majority vote (classification) or average (regression)
- Reduces overfitting through randomization

Random Forest Pseudo-code

procedure RANDOMFOREST(D, T, m)
  Input: Training data D, number of trees T, feature subset size m
  Output: Ensemble model F
  $F \leftarrow \emptyset$ ▷ Initialize empty forest
  for $t = 1$ to T **do**
    $D_t \leftarrow$ BootstrapSample(D) ▷ Sample with replacement
    $\text{Tree}_t \leftarrow \text{GrowTree}(D_t,\ m)$
    $F \leftarrow F \cup \{\text{Tree}_t\}$
  end for
  return F
end procedure

Tree Growing Procedure

```
procedure GrowTree(D, m)
    if StoppingCondition(D) then
        return CreateLeafNode(D)
    else
        features ← RandomSubset(all features, m)
        best_split ← FindBestSplit(D, features)
        left ← GrowTree(D_left)
        right ← GrowTree(D_right)
        return CreateDecisionNode(best_split, left, right)
    end if
end procedure
```

Stopping Conditions

- Maximum tree depth reached
- Minimum samples per leaf
- No improvement in impurity
- All samples belong to same class

Prediction Phase

procedure PREDICT(F, x)
    Input: Forest F, instance x
    Output: Prediction $\hat{y}$
    predictions $\leftarrow \emptyset$
    for tree $\in F$ **do**
        pred $\leftarrow$ tree.predict(x)
        predictions $\leftarrow$ predictions $\cup$ {pred}
    end for
    if Classification **then**
        return mode(predictions) $\triangleright$ Majority voting
    else (Regression)
        return mean(predictions) $\triangleright$ Averaging
    end if
end procedure