Supervised Learning: Decision Trees

Construction, Pruning, Entropy, Information Gain, and Gini Impurity

Sarwan Ali

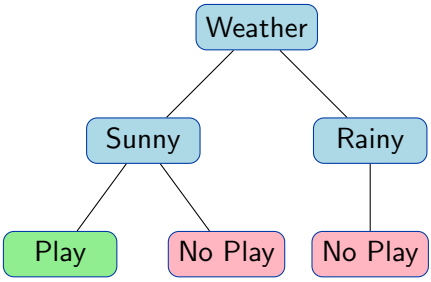
Department of Computer Science
Georgia State University

 Understanding Decision Trees 

Today's Learning Journey

What are Decision Trees?

- **Definition:** A tree-like model of decisions and their possible consequences
- **Structure:** Hierarchical representation with nodes and branches
- **Purpose:** Classification and regression tasks
- **Interpretability:** Easy to understand and visualize



Key Components:

- Root Node: Starting point
- Internal Nodes: Decision points
- Leaf Nodes: Final outcomes
- Branches: Possible paths

Advantages and Disadvantages

Advantages

- Easy to understand and interpret
- No need for data preprocessing
- Handles both numerical and categorical data
- Non-parametric method
- Can capture non-linear relationships
- Feature selection is automatic

Disadvantages

- Prone to overfitting
- Unstable (small data changes = different tree)
- Biased toward features with more levels
- Can create overly complex trees
- May not perform well with linear relationships

Decision Tree Construction Algorithm

General Algorithm (Top-Down Approach):

1. **Start with root node** containing all training examples
2. **Select best attribute** to split the data using splitting criteria
3. **Create branches** for each value of the selected attribute
4. **Partition examples** into subsets based on attribute values
5. **Recursively repeat** for each subset until stopping criteria met

Stopping Criteria:

- All examples in a node have the same class label
- No more attributes to split on
- Maximum depth reached
- Minimum number of samples per node/leaf
- Improvement in purity is below threshold

Attribute Selection Criteria

How do we choose the best attribute to split?

We need measures to evaluate the **quality** of a split:

Entropy	**Information Gain**	**Gini Impurity**
Measures disorder/impurity in the dataset	Reduction in entropy after splitting	Alternative measure of node impurity

Goal: Choose the attribute that best separates the classes (maximizes information gain or minimizes impurity)

Understanding Entropy

Entropy measures the amount of uncertainty or disorder in a dataset.

Mathematical Definition:

$$H(S) = -\sum_{i=1}^{c} p_i \log_2(p_i)$$

Where:

- S = dataset
- c = number of classes
- p_i = proportion of examples belonging to class i

Properties:

- $H(S) = 0$ when all examples belong to the same class (pure)
- $H(S)$ is maximum when classes are equally distributed
- For binary classification: $H(S) = -p \log_2(p) - (1-p) \log_2(1-p)$

Entropy Examples

Example 1: Pure Dataset

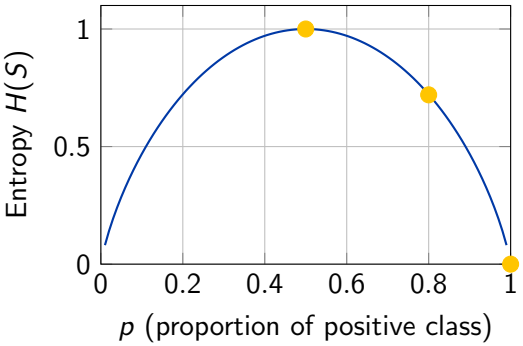
- 10 positive examples, 0 negative
- $p_+ = 1.0$, $p_- = 0.0$
- $H(S) = -1.0 \log_2(1.0) - 0.0 \log_2(0.0) = 0$

Example 2: Balanced Dataset

- 5 positive, 5 negative examples
- $p_+ = 0.5$, $p_- = 0.5$
- $H(S) = -0.5 \log_2(0.5) - 0.5 \log_2(0.5) = 1.0$

Example 3: Imbalanced Dataset

- 8 positive, 2 negative examples
- $p_+ = 0.8$, $p_- = 0.2$
- $H(S) = -0.8 \log_2(0.8) - 0.2 \log_2(0.2) = 0.72$



Information Gain Concept

Information Gain measures the reduction in entropy achieved by splitting the dataset on a particular attribute.

Mathematical Definition:

$$IG(S, A) = H(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} H(S_v)$$

Where:

- S = original dataset
- A = attribute we're considering for splitting
- S_v = subset of S where attribute A has value v
- $|S_v|$ = number of examples in subset S_v

Interpretation:

- Higher information gain = better split
- Choose attribute with maximum information gain
- Represents expected reduction in entropy

Information Gain Example

Tennis Playing Dataset:

Outlook	Temperature	Humidity	Wind	Play Tennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Total: 14 examples (9 Yes, 5 No) $H(S) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 0.940$

Information Gain Calculation

Computing Information Gain for "Outlook":

Subsets after splitting on Outlook:

- **Sunny:** 5 examples (2 Yes, 3 No)

$$H(S_{sunny}) = -\frac{2}{5}\log_2(\frac{2}{5}) - \frac{3}{5}\log_2(\frac{3}{5}) = 0.971$$

- **Overcast:** 4 examples (4 Yes, 0 No)

$$H(S_{overcast}) = -\frac{4}{4}\log_2(\frac{4}{4}) = 0$$

- **Rain:** 5 examples (3 Yes, 2 No)

$$H(S_{rain}) = -\frac{3}{5}\log_2(\frac{3}{5}) - \frac{2}{5}\log_2(\frac{2}{5}) = 0.971$$

Information Gain:

$$IG(S, Outlook) = 0.940$$

$$- \left[\frac{5}{14} \times 0.971 + \frac{4}{14} \times 0 + \frac{5}{14} \times 0.971\right]$$

$$= 0.940 - 0.693 = 0.247$$

Similarly, we calculate IG for other attributes and choose the one with highest IG.

Gini Impurity

Gini Impurity is an alternative measure of node impurity, widely used in CART algorithm.

Mathematical Definition:

$$Gini(S) = 1 - \sum_{i=1}^{c} p_i^2$$

Where:

- S = dataset
- c = number of classes
- p_i = proportion of examples belonging to class i

Properties:

- $Gini(S) = 0$ when all examples belong to the same class (pure)
- $Gini(S)$ is maximum when classes are equally distributed
- For binary classification: $Gini(S) = 1 - p^2 - (1-p)^2 = 2p(1-p)$
- Range: $[0, 0.5]$ for binary classification

Gini Impurity vs Entropy

Gini Examples:

- **Pure:** $p = 1.0$

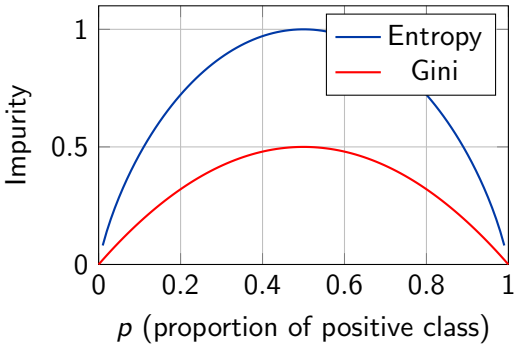
$$Gini = 1 - 1.0^2 = 0$$

- **Balanced:** $p = 0.5$

$$Gini = 1 - 0.5^2 - 0.5^2 = 0.5$$

- **Imbalanced:** $p = 0.8$

$$Gini = 1 - 0.8^2 - 0.2^2 = 0.32$$



Gini Gain:

$$GiniGain(S, A) = Gini(S) - \sum_v \frac{|S_v|}{|S|} Gini(S_v)$$

Both measures are similar, but Gini is computationally faster

Comparison: Entropy vs Gini

Aspect	Entropy	Gini Impurity
Formula	$-\sum p_i \log_2(p_i)$	$1 - \sum p_i^2$
Range	$[0, \log_2(c)]$	$[0, 1 - \frac{1}{c}]$
Computation	Slower (logarithm)	Faster (no logarithm)
Sensitivity	More sensitive to changes	Less sensitive
Usage	ID3, C4.5 algorithms	CART algorithm
Pure node	0	0
Binary balanced	1.0	0.5

Which to choose?

- Both produce similar trees in practice
- Gini is faster to compute
- Entropy has theoretical foundation in information theory

The Overfitting Problem

Overfitting in Decision Trees:

What happens without pruning?

- Tree grows until each leaf is pure
- Memorizes training data including noise
- Poor generalization to new data
- High variance in predictions

Signs of overfitting:

- Very deep trees
- Small number of samples per leaf
- High training accuracy, low test accuracy
- Complex rules that don't generalize

Solution: Pruning - removing parts of the tree that don't improve performance

Types of Pruning

Pre-pruning (Early Stopping)

- Stop growing tree during construction
- Based on stopping criteria:
  - Maximum depth
  - Minimum samples per node/leaf
  - Minimum information gain
  - Maximum number of leaf nodes
- **Advantages:** Fast, simple
- **Disadvantages:** May stop too early

Post-pruning

- Grow full tree, then remove branches
- Methods:
  - Reduced Error Pruning
  - Cost Complexity Pruning
  - Error-Based Pruning
- **Advantages:** More accurate, avoids premature stopping
- **Disadvantages:** Computationally expensive

Most common approach: Use validation set to determine optimal pruning

Cost Complexity Pruning

Cost Complexity Pruning (Minimal Cost-Complexity Pruning):
Balances tree complexity with accuracy using a complexity parameter α.

Cost Function:

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

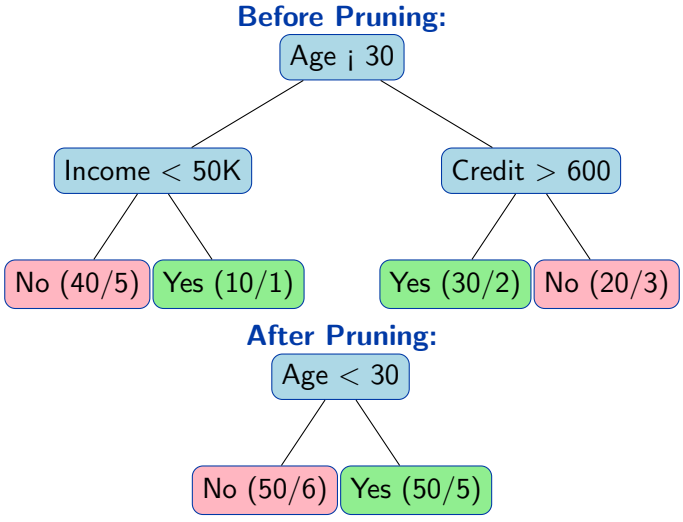
Where:

- $R(T)$ = error rate of tree T
- $|T|$ = number of leaf nodes in tree T
- α = complexity parameter (tuning parameter)

Algorithm:

1. Grow full tree
2. For each internal node, calculate cost improvement from pruning
3. Find subtree that minimizes $R_\alpha(T)$ for given α
4. Use cross-validation to select optimal α

Pruning Example



Numbers in parentheses: (total samples/misclassified)

Step-by-Step Tree Construction

Complete Example: Tennis Dataset

Step 1: Calculate initial entropy

$$H(S) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 0.940$$

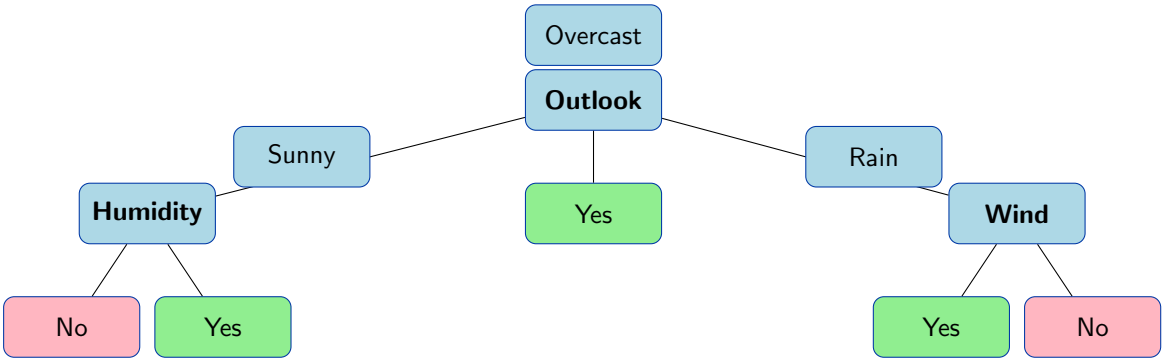
Step 2: Calculate Information Gain for each attribute

- $IG(S, Outlook) = 0.247$
- $IG(S, Temperature) = 0.029$
- $IG(S, Humidity) = 0.152$
- $IG(S, Wind) = 0.048$

Step 3: Choose attribute with highest IG $\rightarrow$ **Outlook**

Step 4: Split dataset and repeat for each branch

Final Decision Tree



Decision Rules:

- If Outlook = Overcast → Play Tennis = Yes
- If Outlook = Sunny and Humidity = Normal → Play Tennis = Yes
- If Outlook = Rain and Wind = Weak → Play Tennis = Yes
- Otherwise → Play Tennis = No

Key Takeaways

Decision Trees Summary:

Construction Process:

- Top-down, greedy approach
- Select best attribute using splitting criteria
- Recursively build subtrees
- Apply stopping criteria

Splitting Criteria:

- **Entropy:** Information theory based
- **Information Gain:** Reduction in entropy
- **Gini Impurity:** Faster alternative

Pruning:

- **Pre-pruning:** Early stopping
- **Post-pruning:** Remove after building
- Prevents overfitting
- Improves generalization

Advantages:

- Interpretable and explainable
- Handles mixed data types
- No assumptions about data distribution
- Feature selection built-in

Applications and Extensions

Real-World Applications:

- **Medical Diagnosis:** Disease prediction
- **Finance:** Credit scoring, fraud detection
- **Marketing:** Customer segmentation
- **Manufacturing:** Quality control
- **Web:** Content recommendation
- **Education:** Student performance prediction

Why Decision Trees?

- Regulatory compliance (explainable AI)
- Domain expert validation
- Feature importance analysis

Extensions & Variants:

- **Random Forest:** Ensemble of trees
- **Gradient Boosting:** Sequential improvement
- **XGBoost:** Optimized gradient boosting
- **C4.5:** Handles missing values
- **CART:** Binary splits only

Regression Trees:

- Predict continuous values
- Use MSE instead of entropy
- Leaf nodes contain mean values

Implementation Considerations

Hyperparameter Tuning:

Tree Structure:

- `max_depth`: Maximum tree depth
- `min_samples_split`: Minimum samples to split
- `min_samples_leaf`: Minimum samples per leaf
- `max_leaf_nodes`: Maximum number of leaves
- `max_features`: Features to consider per split

Splitting Criteria:

- `criterion`: 'gini', 'entropy', 'log_loss'
- `splitter`: 'best' vs 'random'
- `min_impurity_decrease`: Minimum impurity reduction

Pruning:

- `ccp_alpha`: Cost complexity parameter
- Cross-validation for optimal alpha

Best Practices:

- Use validation set for hyperparameter tuning
- Consider ensemble methods for better performance
- Visualize tree structure for interpretability
- Check feature importance scores

Performance Evaluation

Evaluation Metrics:

Classification:

- **Accuracy:** $\frac{TP+TN}{TP+TN+FP+FN}$
- **Precision:** $\frac{TP}{TP+FP}$
- **Recall:** $\frac{TP}{TP+FN}$
- **F1-Score:** $\frac{2 \times Precision \times Recall}{Precision+Recall}$
- **AUC-ROC:** Area under ROC curve

Regression:

- **MSE:** $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
- **MAE:** $\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$
- **R^2:** Coefficient of determination

Cross-Validation:

- K-fold cross-validation
- Stratified K-fold for imbalanced data
- Time series split for temporal data

Overfitting Detection:

- Training vs validation error
- Learning curves
- Validation curves

Feature Importance:

- Gini/Entropy-based importance
- Permutation importance
- SHAP values for explainability

Tree Visualization

Visualizing Decision Trees:

Interpretation Tips:

- **Node color:** Indicates dominant class
- **Samples:** Number of training examples
- **Value:** Class distribution
- **Gini/Entropy:** Node impurity

Tree Depth Analysis:

- Shallow trees: High bias, low variance
- Deep trees: Low bias, high variance
- Optimal depth: Balance bias-variance

Pruning Validation:

- Plot validation curve
- Find optimal complexity parameter
- Compare pruned vs unpruned

Common Pitfalls and Solutions

Common Pitfalls:

- **Overfitting:** Deep, complex trees
  - Solution: Pruning, max_depth
- **Bias toward categorical features:** More levels = higher IG
  - Solution: Gain ratio, balanced splitting
- **Instability:** Small data changes = different tree
  - Solution: Ensemble methods
- **Imbalanced data:** Bias toward majority class
  - Solution: Class weights, resampling

Best Practices:

- **Data preprocessing:**
  - Handle missing values
  - Encode categorical variables
  - Scale if using distance-based splits
- **Model selection:**
  - Use cross-validation
  - Grid search for hyperparameters
  - Consider ensemble methods
- **Interpretation:**
  - Visualize tree structure
  - Analyze feature importance
  - Validate with domain experts

Summary and Next Steps

What We've Learned:

- **Decision Tree Fundamentals:** Structure, construction, and interpretation
- **Splitting Criteria:** Entropy, Information Gain, and Gini Impurity
- **Pruning Techniques:** Pre-pruning and post-pruning to prevent overfitting
- **Practical Implementation:** Python code and best practices
- **Evaluation Methods:** Metrics and visualization techniques

Next Topics in Our ML Journey:

- **Ensemble Methods:** Random Forest, Gradient Boosting
- **Support Vector Machines:** Maximum margin classification
- **Neural Networks:** Deep learning foundations
- **Model Selection:** Cross-validation and hyperparameter tuning

Questions & Discussion