Fundamental Algorithms

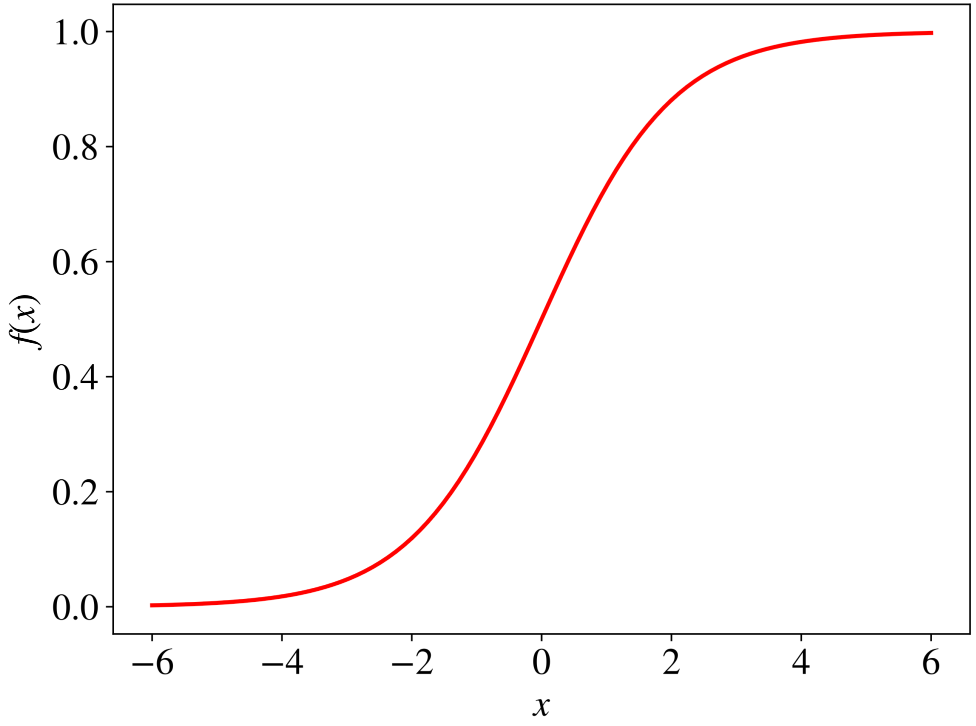
Chapter 3: Five Essential Supervised Learning Algorithms

2. Logistic Regression

Purpose: Classification (despite the name!)

Problem: Linear expression $\mathbf{w}^{\mathbf{T}}\mathbf{x} + b$ ranges from $-\infty$ to $+\infty$
But we need probabilities between 0 and 1

Solution: Use logistic function



Logistic Regression: Model

Logistic Function:

$$f_{\mathbf{w},b}(x) = \frac{1}{1 + e^{-(\mathbf{w}^{\mathbf{T}}\mathbf{x}+b)}}$$

Properties:

- Output range: (0, 1)
- Interpreted as $P(y = 1|x)$
- Decision rule: predict class 1 if $f(x) \geq 0.5$

Connection to Gradient Descent:

- We'll use gradient descent to find optimal $\mathbf{w}$ and b

Logistic Regression: Training

Uses Maximum Likelihood Estimation

Likelihood Function:

$$L_{\mathbf{w},b} = \prod_{i=1}^{N} f_{\mathbf{w},b}(x_i)^{y_i}(1 - f_{\mathbf{w},b}(x_i))^{1-y_i}$$

Log-Likelihood (easier to optimize):

$$\log L_{\mathbf{w},b} = \sum_{i=1}^{N} y_i \log f_{\mathbf{w},b}(x_i) + (1 - y_i)\log(1 - f_{\mathbf{w},b}(x_i))$$

Optimization: Gradient Descent (as we just learned!)

- Calculate gradients of log-likelihood
- Update parameters iteratively

Gradient Descent for Logistic Regression

Cost Function (Negative Log-Likelihood):

$$J(\mathbf{w}, b) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(f_{\mathbf{w},b}(x_i)) + (1 - y_i)\log(1 - f_{\mathbf{w},b}(x_i))]$$

Gradients:

$$\frac{\partial J}{\partial w_j} = \frac{1}{N}\sum_{i=1}^{N}(f_{\mathbf{w},b}(x_i) - y_i)x_{ij} \text{ and } \frac{\partial J}{\partial b} = \frac{1}{N}\sum_{i=1}^{N}(f_{\mathbf{w},b}(x_i) - y_i)$$

Update Rules:

$$w_j := w_j - \alpha\frac{\partial J}{\partial w_j} \text{ and } b := b - \alpha\frac{\partial J}{\partial b}$$

Key Insight: Same form as linear regression, but $f(\mathbf{x})$ is now the sigmoid function!

Logistic Regression: Inductive Bias

What does Logistic Regression assume?

Core Assumption: Classes are **linearly separable** in the feature space

- Decision boundary is a hyperplane: $\mathbf{w}^\mathbf{T}\mathbf{x} + b = 0$
- Features combine linearly to influence probability

Inductive Bias Examples:

- ✅ **Good for:** Email spam detection (word frequencies combine linearly)
- ✅ **Good for:** Medical diagnosis (symptoms independently contribute)
- ❌ **Bad for:** XOR problem (non-linear decision boundary needed)
- ❌ **Bad for:** Image classification (pixels don't combine linearly)

Logistic Regression: Implementation

Complete Implementation: Check the Noteboook

3. Decision Tree Learning

The Classical Model of Learning

This is a classic and natural model of learning, closely related to the fundamental computer science notion of **"divide and conquer."**

Core Concept: Learn to ask the right questions in the right order to make predictions.

The Question-Answer Learning Game

Scenario: Predict whether a student will enjoy an unknown course

Your Tool: Binary questions about the student/course

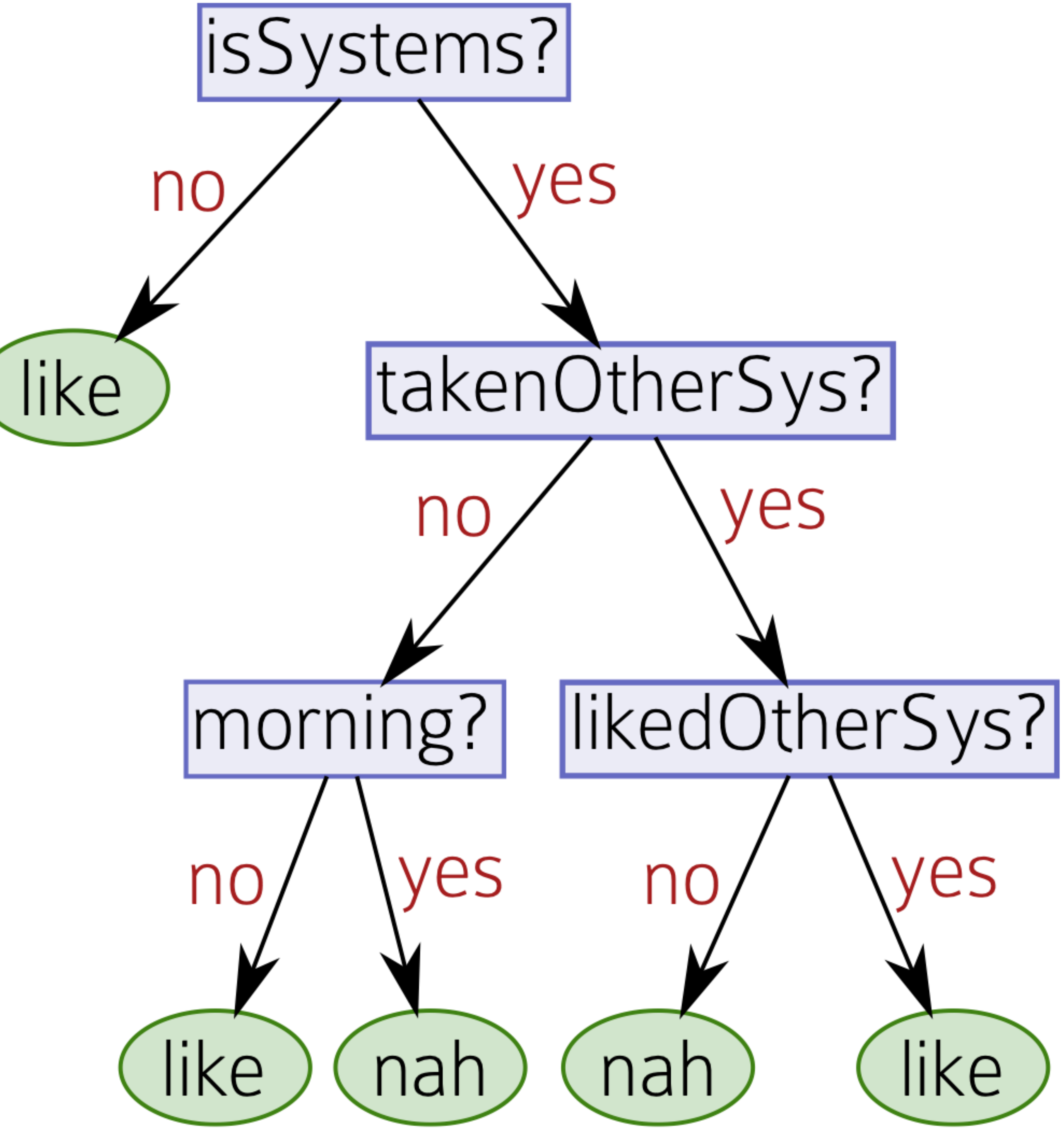
Example Interaction:

```
You: Is the course under consideration in Systems?
Me: Yes
You: Has this student taken any other Systems courses?
Me: Yes
You: Has this student liked most previous Systems courses?
Me: No
You: I predict this student will NOT like this course.
```

The Question-Answer Learning Game

Learning Goal: Figure out:

1. What questions to ask
2. What order to ask them
3. What answer to predict once you have enough information



The Tree Structure

Decision Tree Components:

- **Internal nodes (rectangles):** Questions/splits
- **Leaves (ovals):** Final predictions
- **Edges:** Answers (Yes/No, Left/Right)

Navigation Rules:

- **Left child:** "No" answer
- **Right child:** "Yes" answer
- **Traverse** until you reach a leaf

Training Data and Features

Terminology:

- **Features:** Questions you can ask (e.g., "Is Systems course?")
- **Feature values:** Responses to questions (Yes/No, categorical, numerical)
- **Label:** The target prediction (Like/Hate, Class A/B)
- **Example:** One data point with feature values + label
- **Training data:** Set of examples with known labels

Example Dataset:

Course Type	Difficulty	Prev Experience	Grade	Label
Systems	Hard	Yes	B+	Hate
Theory	Easy	No	A	Like

The Greedy Learning Strategy

Key Insight: Millions of possible trees exist - we can't try them all!

Greedy Approach: Build tree top-down

1. **Ask:** "If I could only ask ONE question, which would be most useful?"
2. **Split:** Partition data based on best question
3. **Recurse:** Apply same strategy to each partition
4. **Stop:** When data is pure or no features remain

"Most Useful" Question:

- Look at histogram of labels for each feature value
- Choose feature that best separates different labels
- Maximize classification accuracy if we had to guess now

Feature Selection by Scoring

Example: Evaluating "Is this a Systems course?"

Step 1: Partition training data

- **NO set:** Non-systems courses , **YES set:** Systems courses

Step 2: Build histograms

- NO set: [Like: 8, Hate: 2] → Guess "Like", YES set: [Like: 2, Hate: 8] → Guess "Hate"

Step 3: Calculate accuracy

- NO set: 8/10 correct (guessing "Like"), YES set: 8/10 correct (guessing "Hate")
- **Total score:** 16/20 = 80% accuracy

Step 4: Repeat for all features, choose highest score

Algorithm 1 DecisionTreeTrain(*data*, *remaining features*)

```
guess ← most frequent answer in data                    // default answer for this data
if the labels in data are unambiguous then
    return Leaf(guess)                                  // base case: no need to split further
else if remaining features is empty then
    return Leaf(guess)                                  // base case: cannot split further
else                                                    // we need to query more features
    for all f ∈ remaining features do
        NO ← the subset of data on which f=no
        YES ← the subset of data on which f=yes
        score[f] ← # of majority vote answers in NO
                      + # of majority vote answers in YES
                        // the accuracy we would get if we only queried on f
    end for
    f ← the feature with maximal score(f)
    NO ← the subset of data on which f=no
    YES ← the subset of data on which f=yes
    left ← DecisionTreeTrain(NO, remaining features \ {f})
    right ← DecisionTreeTrain(YES, remaining features \ {f})
    return Node(f, left, right)
end if
```

Algorithm 2 DecisionTreeTest(*tree*, *test point*)

```
if tree is of the form Leaf(guess) then
    return guess
else if tree is of the form Node(f, left, right) then
    if f = no in test point then
        return DecisionTreeTest(left, test point)
    else
        return DecisionTreeTest(right, test point)
    end if
end if
```

Divide and Conquer Algorithm

Recursive Strategy:

```
DecisionTreeTrain(data, remaining_features):
    // Base cases
    if data has single label:
        return Leaf(most_common_label)
    if no remaining_features:
        return Leaf(most_common_label)

    // Find best split
    best_feature = highest_scoring_feature(remaining_features)

    // Partition data
    left_data = data where best_feature = "No"
    right_data = data where best_feature = "Yes"

    // Recurse
    left_subtree = DecisionTreeTrain(left_data, remaining_features - best_feature)
    right_subtree = DecisionTreeTrain(right_data, remaining_features - best_feature)

    return DecisionNode(best_feature, left_subtree, right_subtree)
```

Prediction Algorithm

Making Predictions:

```
DecisionTreePredict(tree, test_example):
    if tree is Leaf:
        return tree.prediction

    feature_value = test_example[tree.feature]

    if feature_value == "No":
        return DecisionTreePredict(tree.left, test_example)
    else:
        return DecisionTreePredict(tree.right, test_example)
```

Process:

1. Start at root with test example
2. Follow edges based on feature values

Why Decision Trees Work

Intuitive Appeal:

- **Human-like reasoning:** How experts make decisions
- **Interpretable:** Easy to understand and explain
- **No assumptions:** Works with any type of features
- **Automatic feature selection:** Finds most important questions

Limitations:

- **Greedy strategy:** May miss globally optimal tree
- **Overfitting:** Can memorize training data
- **Instability:** Small data changes can create very different trees

Decision Tree: Algorithm (Gini Impurity)

Building Process:

1. Start with all training data at root
2. For each feature and threshold, calculate Gini impurity
3. Choose split that minimizes weighted Gini impurity
4. Recursively split until stopping criteria

Gini Impurity Formula:

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

Where p_i is the proportion of samples belonging to class i in set S

For binary classification:

Decision Tree: Splitting with Gini

Weighted Gini After Split:

$$\text{Gini}_{split}(S^{left}, S^{right}) = \frac{|S^{left}|}{|S|}\text{Gini}(S^{left}) + \frac{|S^{right}|}{|S|}\text{Gini}(S^{right})$$

Gini Gain: Choose split that maximizes reduction in Gini impurity

$$\text{Gini Gain} = \text{Gini}(S) - \text{Gini}_{split}(S^{left}, S^{right})$$

Decision Tree: Splitting with Gini

Why Gini Impurity?

- Computationally efficient (no logarithms)
- Measures node "purity"
- 0 = pure node (all same class)
- Maximum when classes are equally distributed

Stopping Criteria:

- All labels in node are same (Gini = 0)
- No improvement in Gini gain (< threshold ε)
- Maximum depth reached
- Minimum samples per node

Decision Tree: Key Concepts

Internal Nodes: Feature tests

Leaves: Predictions/classes

Splitting Criteria:

- Information Gain (classification)
- Gini Impurity
- Mean Squared Error (regression)

Decision Tree: Key Concepts

Advantages:

- Highly interpretable
- Handles mixed data types
- No assumptions about data distribution
- Can capture non-linear relationships

Disadvantages:

- Prone to overfitting
- Can be unstable (small data changes → different tree)
- Biased toward features with more levels

Decision Tree: Inductive Bias

What does Decision Tree assume?

Core Assumption: "Few features matter most" - Simple rules with minimal features

- Decisions can be made by testing a small number of features
- Tree-like (hierarchical) decision making is appropriate
- Axis-aligned splits are sufficient

Inductive Bias Examples:

- ✅ **Good for:** Medical diagnosis (few key symptoms determine disease), Credit approval (income, credit score, employment status), Game playing (position evaluation with clear rules)
- ❌ **Bad for:** Parity functions (need ALL features: XOR, majority vote), High-dimensional data where many features matter equally