

# Introduction to Machine Learning

## Lecture 3

Yonghao Lee

November 17, 2025

# Contents

<b>1</b>	<b>Applications of Distance Methods</b>	<b>3</b>
1.1	Retrieval . . . . .	3
1.2	Anomaly Detection . . . . .	3
<b>2</b>	<b>K-Means Clustering</b>	<b>4</b>
2.1	The Mathematical Objective . . . . .	4
2.2	The K-Means Algorithm . . . . .	4
2.3	Limitations . . . . .	5
<b>3</b>	<b>From Non-Parametric to Parametric Models</b>	<b>5</b>
3.1	Parametric Methods . . . . .	5
3.2	First Parametric Model: Decision Stump . . . . .	6
3.3	Learning a Decision Stump . . . . .	7
3.4	What Threshold to Choose . . . . .	7
<b>4</b>	<b>Learning Theory for Parametric Models</b>	<b>7</b>
4.1	The "Elephant in the Room" . . . . .	7
4.2	No Free Lunch Theorem . . . . .	7
4.3	Vapnik-Chervonenkis (VC) Dimension . . . . .	8
<b>5</b>	<b>Decision Trees</b>	<b>8</b>
5.1	Hypothesis Class: Axis-Aligned Decision Rule . . . . .	8
5.2	Inductive Bias . . . . .	9
5.3	Trees as AA Decision Rules . . . . .	9
5.4	How to Learn Decision Trees . . . . .	9
5.5	Learning Procedure . . . . .	10
5.6	Decision Rule by Accuracy . . . . .	10

# 1 Applications of Distance Methods

This section expands on the k-Nearest Neighbors (kNN) concept, showing how the core idea of "distance" can be applied to other tasks.

## 1.1 Retrieval

**Key Concept 1.1: Goal:** *Given a query item (e.g., an image, a document), find the  $K$  most similar items from a database.*

- **Algorithm:**

1. Compute the distance between the query item and every item in the training database.
2. Sort the database items by this distance in ascending order.
3. Retrieve the top  $K$  items (the  $K$  nearest neighbors).

- **Key Dependency:** The success of this method depends entirely on the chosen representation (features) and the distance function.

## 1.2 Anomaly Detection

**Key Concept 1.2: Goal:** *To identify rare, unusual, or "abnormal" data points that are significantly different from the "normal" data.*

**Algorithm 1.1** (kNN-based Anomaly Detection): *1. **Set a Threshold ( $T$ ):** This is a pre-defined value that acts as our boundary for "normal."*

*2. **Input a New Target Point ( $x_{\text{target}}$ ):** This is the point we want to test.*

*3. **Find Neighbors:** Find the  $K$  nearest neighbors of  $x_{\text{target}}$  from the training data by computing all distances and sorting.*

*4. **Calculate Anomaly Score:** Compute the average distance from  $x_{\text{target}}$  to its  $K$  neighbors.*

*5. **Classify:***

- *If (Average Distance  $> T$ ): The point is an **Anomaly**.*
- *If (Average Distance  $< T$ ): The point is **Normal**.*

### Common Point of Confusion

**Core Assumption of kNN Anomaly Detection** A common point of confusion is the underlying assumption of this method.

- **The Core Idea:** The method works on the assumption of density. Normal data points are "clumpy" and live in dense neighborhoods, while anomalies are isolated and far away.
- **Normal Point:** A new "normal" point will land inside a dense neighborhood. Its neighbors will be close, giving a low score.
- **Anomalous Point:** A new anomaly will land far from dense regions. Its nearest neighbors will still be far away, giving a high score.
- **The Threshold:** Choosing  $T$  is crucial. Too high: miss anomalies. Too low: flag normal points incorrectly.

## 2 K-Means Clustering

K-Means is an **unsupervised learning** algorithm. This is used when we have data  $x_i$  but no labels  $y_i$ . The goal is to discover the group structure in the data.

### 2.1 The Mathematical Objective

**Key Concept 2.1: Goal:** To partition the data into  $K$  clusters,  $S = \{S_1, \dots, S_K\}$ , where each cluster  $k$  is represented by a mean vector  $c_k$  (centroid).

The objective is to find the cluster assignments ( $S_k$ ) and the means ( $c_k$ ) that minimize the total intra-cluster sum of squares:

$$L(S_1, \dots, S_K) = \min_{S_1, \dots, S_K, c_1, \dots, c_K} \sum_{k=1}^K \sum_{x_i \in S_k} \|x_i - c_k\|^2$$

### 2.2 The K-Means Algorithm

Finding the perfect solution is computationally infeasible (exponential). Instead, K-Means uses a simple two-step iterative process:

**Algorithm 2.1 (K-Means):** 1. **Initialize:** Randomly choose  $K$  initial means ( $c_1, \dots, c_K$ ).  
2. **Assignment Step:** Assign each sample  $x_i$  to the nearest mean:

$$S_k = \{x_i : \|x_i - c_k\|^2 \leq \|x_i - c_j\|^2 \text{ for all } j \neq k\}$$

3. **Update Step:** Recalculate  $c_k$  as the average of the samples in  $S_k$ :

$$c_k = \frac{1}{|S_k|} \sum_{x_i \in S_k} x_i$$

4. **Repeat:** Continue until assignments do not change.

### Common Point of Confusion

#### Key Concepts in K-Means Implementation

##### 1. Initialization

- Results depend on starting points.
- Bad: all zeros or random coordinates.
- Better: pick  $K$  actual samples as starting means.
- Best: run the whole algorithm multiple times and choose the best clustering (lowest  $L$ ).

##### 2. Mean vs. Sample

- The new  $c_k$  is not a sample—it's the average of the points.
- Choosing an actual sample is K-Medoids, not K-Means.

##### 3. Cluster Label Meaning

- K-Means returns clusters, but it doesn't know what they "mean".
- Humans must inspect clusters to label them meaningfully.

## 2.3 Limitations

- Results depend on the choice of distance and representation.
- Not guaranteed to converge to the global optimum.
- It is not clear how to choose the correct  $K$ .
- Clusters do not have inherent meanings.

## 3 From Non-Parametric to Parametric Models

This marks a major shift from non-parametric methods (like kNN) to **parametric methods**.

### 3.1 Parametric Methods

**Definition 3.1:** A *parametric model* is a model that is defined by a set of parameters (variables). The learning task is to find the values of these parameters that best fit the training data.

### Common Point of Confusion

#### "Mapping Parameters to Models" & "Finding the ERM Model"

- **"Mapping Parameters to Models":** This means the parameters are the "settings" and the "model" is the specific prediction rule you get.
  - **Model Class (Template):** "I will use one threshold ( $\theta$ ) on one feature."
  - **Parameter (Setting):**  $\theta = 38$ .
  - **Model (Product):** The specific rule: "If temp  $< 38$ , predict Healthy."
  - A different parameter ( $\theta = 37.5$ ) maps to a different model ("If temp  $< 37.5$ ...").
- **"Finding the ERM Model":** This is the definition of "learning".
  - **ERM** = Empirical Risk Minimization = The model with the lowest error (risk) on the training data (empirical).
  - **"Finding the values of parameters..."** = The process of testing all the settings (e.g., trying  $\theta = 37, \theta = 38, \theta = 39$ ...) to see which one produces the model with the best accuracy on the training data.

## 3.2 First Parametric Model: Decision Stump

A **Decision Stump** is the simplest parametric model.

- **Rule:** It makes a decision based on a **single 1D feature** ( $x$ ) and a **single parameter**, the threshold ( $\theta$ ).
- **Example:** If  $x$  (Temperature)  $< 38$ , predict "Healthy". Else, predict "Sick".

### 3.3 Learning a Decision Stump

#### Common Point of Confusion

##### The "Infinite Hypotheses" Problem

- **The Problem:** Theoretically, there are *infinite* possible values for the threshold  $\theta$  (e.g., 37.5, 37.51, 37.511...). This creates a problem for learning theory, which suggests we'd need infinite data if we have infinite models.
- **The Practical Solution:** We don't need to check infinite thresholds. The decision rule only changes when the threshold  $\theta$  *crosses over* a data point.
- **Algorithm:**
  1. Sort the  $n$  training samples by their feature  $x$ .
  2. Only check thresholds that fall *between* consecutive sorted points (e.g., if we have points at 37 and 37.5, we only need to test one  $\theta$  in between, like 37.25).
  3. This gives a finite, small number of thresholds to test.
  4. We choose the one that gives the best accuracy (the ERM stump).

### 3.4 What Threshold to Choose

- Sometimes, a *range* of thresholds all give the same perfect accuracy (e.g., any  $\theta$  between 37.5 and 38).
- **Which to pick?** The most robust idea is to pick the one that gives the **maximal margin of error**.
- This means picking the threshold exactly in the middle of the gap (e.g.,  $(37.5 + 38)/2 = 37.75$ ). This is the "safest" choice.

## 4 Learning Theory for Parametric Models

### 4.1 The "Elephant in the Room"

This refers to the major, obvious problem: our old learning theory (from Week 1) is broken.

- **Old Theory:** Required  $n$  (samples) to be related to  $|H|$  (number of hypotheses).
- **New Problem:** A simple Decision Stump has  $|H| = \infty$ .
- **Conclusion:** The old theory is not good enough. We need a new theory that measures model complexity in a "smarter way" than just counting.

### 4.2 No Free Lunch Theorem

This concept explains why an infinitely powerful model is useless.

- A model class that is "too rich" (too flexible) can fit *any* training data perfectly.
- But for a new, unseen point, the class will contain "perfect" models that predict *every possible label* for that new point.
- The training data gives us no information to choose between these models. The model has learned nothing.
- **Conclusion:** To learn, a model must have some "bias" or be limited in its flexibility.

### 4.3 Vapnik-Chervonenkis (VC) Dimension

VC Dimension is the "smarter" way to measure a model's capacity or power.

**Definition 4.1** (Shattering): A hypothesis class  $H$  "**shatters**" a set of  $d$  points if it can find a model that perfectly implements **every single possible labeling** for those  $d$  points. For  $d$  points, there are  $2^d$  possible label combinations.

**Definition 4.2** (VC Dimension): The VC Dimension of a hypothesis class  $H$  is the **largest number**  $d$  of points that  $H$  can shatter.

- If a model has infinite VC dim, it cannot be learned (the "No Free Lunch" problem).
- A model is "learnable" if it has a finite VC dimension.

#### Common Point of Confusion

##### VC Dim: "Smarter" Capacity vs. "Dumb" Counting

- **Counting (Dumb):** We count the parameters. For a stump,  $\theta$  is continuous, so we get  $\infty$  models. This is useless.
- **Capacity (Smart):** We test the model's *power*.
  - Can a stump shatter 1 point? **Yes.** We can label it (+) or (-) by moving  $\theta$ .
  - Can a stump shatter 2 points ( $x_1, x_2$ )? **No.** It can achieve (+,+), (-,-), and (-,+). But it is *impossible* for a single threshold to achieve (+, -).
- **Conclusion:** The largest number of points a stump can shatter is 1. Its VC dimension is **1**. This finite number correctly captures its simplicity.

## 5 Decision Trees

### 5.1 Hypothesis Class: Axis-Aligned Decision Rule

**Key Concept 5.1:** This is a new hypothesis class (a new type of model). Its rules are based on carving up the feature space into non-overlapping rectangles using only **axis-aligned** (horizontal and vertical) lines. Every sample inside one rectangle gets the same prediction.

## 5.2 Inductive Bias

**Definition 5.1:** *Inductive Bias* is the set of assumptions a model makes about a problem. It is a "preference" for a certain shape of solution.

- The inductive bias of an Axis-Aligned (AA) model is that it **assumes the true class boundary is also rectangular**.

### Common Point of Confusion

**"Good" vs. "Poor" Inductive Bias** This concept depends on the *data*, not just the model.

- **Good Inductive Bias:** The data's true pattern (e.g.,  $\text{temp} > 38$  AND  $\text{coughs} > 10$ ) is *rectangular*. Our model's bias (preference for rectangles) is a perfect match. The model will be simple and accurate.
- **Poor Inductive Bias:** The data's true pattern is *diagonal* (e.g.,  $\text{temp} + \text{coughs} > 50$ ). Our model's bias (preference for rectangles) is a terrible match. The model cannot learn this simple rule and will be forced to make a complex, inaccurate "staircase" approximation.

## 5.3 Trees as AA Decision Rules

This is the key connection: A **Decision Tree** is the *algorithm* used to *implement* an Axis-Aligned Decision Rule.

- The AA rule is the visual goal (the map with rectangles).
- The Decision Tree is the flowchart of questions (the "turn-by-turn" directions) that create those rectangles.
- **Every internal node (question) in the tree is just a Decision Stump.**
- Each leaf node in the tree corresponds to one final rectangle on the map.

## 5.4 How to Learn Decision Trees

- **Goal:** Find the tree that gives the best ERM (lowest error on training data).
- **Problem:** Finding the single *optimal* tree is computationally infeasible (too slow).
- **Solution: A "Greedy" Method.** Instead of planning the whole tree at once, we build it from the top down. At each step, we just make the single decision that looks best \*right now\*.
- **The Idea:** Learn one Decision Stump (one split) at a time.

## 5.5 Learning Procedure

This is the recursive, greedy algorithm for building the tree.

1. **Begin:** Start at a node (at the top, this is the **root node**) with a set of training samples (at the root, this is the **entire dataset**).
2. **Split:** Find the *single best decision stump* (the best feature  $<$  threshold rule) that splits the *current* set of samples. This creates new child nodes.
3. **Stop:** At a child node, if you decide to stop (e.g., the node is pure, or max depth is reached), it becomes a **leaf node**. Its prediction is the **majority label** of the samples that ended up there.
4. **Recurse:** For any new child node that is not a leaf, repeat this entire 3-step process.

### Common Point of Confusion

#### What is the "Root Node"?

- The Root Node is **not** your dataset. The Root Node is the **first question** (the first split) of your tree.
- This first question is *chosen by* looking at the **entire dataset**. The algorithm tests all features and all thresholds to find the single best split for the whole dataset.
- That best split *\*becomes\** the root node, which then divides the dataset and sends it to the child nodes.

## 5.6 Decision Rule by Accuracy

This is the first method for finding the "best split" at a node.

- **The Goal:** To find the one split (e.g.,  $\text{temp} < 37.5$ ) that maximizes the total accuracy.
- **The Method: A "Brute-Force Competition"**
  1. **For each feature (e.g., Temperature):**
    - Test every possible threshold (e.g.,  $< 36.5$ ,  $< 37$ ,  $< 37.5$ ...).
    - For each threshold, calculate the *total accuracy* (using the majority label of the two children it creates).
    - Find the "champion" split for this feature (e.g.,  $\text{temp} < 37.5$  gives 90)
  2. **For each other feature (e.g., Coughs):**
    - Find its "champion" split (e.g.,  $\text{Coughs} < 10$  gives 80)
  3. **Compare Champions:** The algorithm compares the best score from every feature (temp's 90)
  4. **Select Winner:** The split  $\text{temp} < 37.5$  wins and is chosen as the rule for this node.