Lecture 2: k-Nearest Neighbors

ECE 2410 – Introduction to Machine Learning

Farzad Farnoud

University of Virginia

Spring 2026

Today's Agenda

1. Representing Data
2. Matrices, Vectors, and Operations
3. kNN
4. Distances and Norms
5. Python and NumPy
6. Practice
7. Summary

Last Time: Predict the Major

The Activity

We predicted a student's major based on the majors of their $k = 3$ "closest" neighbors.

Key insight:

- Relies on the assumption that there is a relationship between inputs and outputs
- Similar inputs $\rightarrow$ similar outputs
- Had we tried to predict cat vs dog person, the error would have been higher

Today

We'll formalize this intuition in our first ML algorithm: **k-Nearest Neighbors (kNN)**

A Bit of Terminology

- **Training data** — solved examples we learn from
- **Test data** — new examples to evaluate on
- **Validation data** — used to choose between approaches and tune model settings
- **Production data** — real-world data after deployment

- **Features (inputs, x)** — measurable inputs (e.g., where you seat)
- **Label (output, y)** — what we want to predict (e.g., major)

- **Generalization** — ability to perform well on *new* data

Key Insight

We care about performance on **test data**, not just training data!

The Classification Problem

Given:

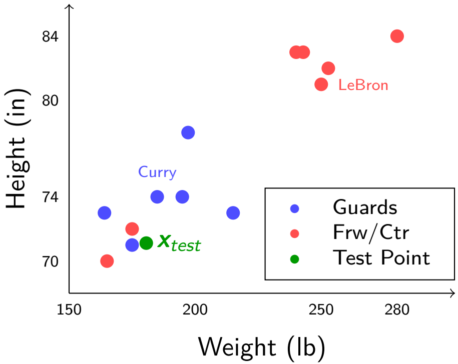
- Training data: n examples
- Each example: $(\boldsymbol{x}_i, y_i)$
- $\boldsymbol{x}_i$ = features (input)
- y_i = label/class (output)

Goal:

- Given a new point $\boldsymbol{x}_{test}$
- Predict its label $\hat{y}_{test}$

Example:

- Predict NBA player position based on height and weight



Is the green point a guard or forward?

Vectors and Matrices: The Basics

Before we proceed: we need a language to describe data and algorithms effectively.

A **Vector** is an ordered list of D numbers
D is the **dimension**; written as row or column, but it's the same object

$$\boldsymbol{x} = \begin{bmatrix} x_1 & \cdots & x_D \end{bmatrix} \quad \text{or} \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_D \end{bmatrix}$$

Example: A player's data

$$\boldsymbol{x} = \begin{bmatrix} 250 & 81 \end{bmatrix} \quad \text{(Weight, Height)}$$

A **matrix** is a rectangular array of numbers with N rows and D columns.

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1D} \\ a_{21} & a_{22} & \cdots & a_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{ND} \end{bmatrix}$$

We write:

- $\boldsymbol{x} \in \mathbb{R}^D$
- $\boldsymbol{A} \in \mathbb{R}^{N \times D}$

From Data to Matrices and Vectors

Raw Data Table:

Name	Weight	Height	Pos
K. Abdul-Jabbar	225	86"	C
M. Abdul-Rauf	162	73"	G
...	...	...	...
L. James	250	81"	F
S. Curry	185	74"	G

Data record i, $(\boldsymbol{x}_i, y_i)$:

- **Features ($\boldsymbol{x}_i$):** Measurable properties $(x_{i1} = \mathit{Height}, x_{i2} = \mathit{Weight})$
- **Label (y_i):** What we want to predict (Position)

From Data to Matrices and Vectors

Raw Data Table:

Name	Weight	Height	Pos
K. Abdul-Jabbar	225	86”	C
M. Abdul-Rauf	162	73”	G
…	…	…	…
L. James	250	81”	F
S. Curry	185	74”	G

$\longleftrightarrow$ Row $i \leftrightarrow (\boldsymbol{x}_i, y_i)$

Matrices & Vectors:

$$\boldsymbol{X} = \begin{bmatrix} 225 & 86 \\ 162 & 73 \\ \vdots & \vdots \\ 250 & 81 \\ 185 & 74 \end{bmatrix} \qquad \boldsymbol{y} = \begin{bmatrix} \text{C} \\ \text{G} \\ \vdots \\ \text{F} \\ \text{G} \end{bmatrix}$$

Data record i, $(\boldsymbol{x}_i, y_i)$:

- **Features ($\boldsymbol{x}_i$):** Measurable properties ($x_{i1} = Height, x_{i2} = Weight$)
- **Label (y_i):** What we want to predict (Position)

Structure:

- $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ (Features)
- $\boldsymbol{y} \in \mathbb{R}^{N}$ (Labels)

The Data Matrix and the Label Vector

Data Matrix: $\boldsymbol{X} = \begin{bmatrix} — & \boldsymbol{x}_1 & — \\ — & \boldsymbol{x}_2 & — \\ & \vdots & \\ — & \boldsymbol{x}_N & — \end{bmatrix} \in \mathbb{R}^{N \times D}$

- What are N, D, and row i?

Label Vector: $\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \in \mathbb{R}^N$

- y_i = label for $\boldsymbol{x}_i$

NBA Example:

$$\boldsymbol{X} = \begin{bmatrix} 185 & 74 \\ 175 & 72 \\ 250 & 81 \\ \vdots & \vdots \end{bmatrix} \quad \boldsymbol{y} = \begin{bmatrix} \text{G} \\ \text{G} \\ \text{F} \\ \vdots \end{bmatrix}$$

Accessing Elements

Vector Notation:

$$\boldsymbol{x}_i = \begin{bmatrix} x_{i1} & x_{i2} & \cdots & x_{iD} \end{bmatrix}$$

- $\boldsymbol{x}_{i2}$ = second element of $\boldsymbol{x}_i$ =

Matrix Notation:

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

- a_{ij} = element in row i, column j
- Math uses **1-indexing**
- a_{23} = row 2, column 3

Extracting from Matrices:

- Row i of A: $A_{i,:}$ or A_i
- Column j of A: $A_{:,j}$

Example:

$$\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

- $a_{12} = 2$
- Row 1: $A_{1,:} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$
- Column 2: $A_{:,2} = \begin{bmatrix} 2 \\ 5 \end{bmatrix}$

In-Class Activity: Practice the Notation!

Dataset: Student Study Habits

Name	Hours/Week	Coffees/Day	Grade
Alice	15	2	A
Bob	8	1	B
Carol	20	4	A
Dave	5	0	C

Task: Considering the $(X, \boldsymbol{y})$ representation, for each quantity:

- Describe it in words
- Write it out mathematically

Example: $\boldsymbol{x}_1$

- *Alice's feature vector*
- $\begin{bmatrix} 15 & 2 \end{bmatrix}$

Your turn: Solve the following individually, then discuss with your neighbor. Revise your work and turn in the final answers.

1. $\boldsymbol{x}_2$
2. $x_{3,2}$
3. $\boldsymbol{X}$
4. $\boldsymbol{y}$
5. y_1
6. $\boldsymbol{X}_{:,1}$

In-Class Activity: Solutions

Answers:

1. $\boldsymbol{x}_2$ = *Bob's feature vector*
   $\begin{bmatrix} 8 & 1 \end{bmatrix}$
2. $x_{3,2}$ = *Carol's coffee count*
   4
3. $\boldsymbol{X}$ = *Feature matrix (all data)*
   $\begin{bmatrix} 15 & 2 \\ 8 & 1 \\ 20 & 4 \\ 5 & 0 \end{bmatrix}$
4. $\boldsymbol{y}$ = *Label vector (all grades)*
   $\begin{bmatrix} \text{A} \\ \text{B} \\ \text{A} \\ \text{C} \end{bmatrix}$
5. y_1 = *Alice's grade*
   A
6. $\boldsymbol{X}_{:,1}$ = *Study hours column*
   $\begin{bmatrix} 15 \\ 8 \\ 20 \\ 5 \end{bmatrix}$

The Inner (dot) Product

Definition

The **inner product** (or dot product) of two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^D$ is the sum of element-wise products:

$$\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \sum_{j=1}^{D} a_j b_j$$

Interpretation: Measures **similarity** or alignment. *Higher dot product = more similar!*

Example: Movie Ratings $\begin{bmatrix} \text{Action} \\ \text{Romance} \end{bmatrix}$: $\quad \boldsymbol{a} = \begin{bmatrix} 5 \\ 1 \end{bmatrix}, \boldsymbol{b} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}, \boldsymbol{c} = \begin{bmatrix} 1 \\ 5 \end{bmatrix}$

$\langle \boldsymbol{a}, \boldsymbol{b} \rangle = 5(4) + 1(2) = \mathbf{22}$ High similarity! $\quad \langle \boldsymbol{a}, \boldsymbol{c} \rangle = 5(1) + 1(5) = \mathbf{10}$ Low similarity.

Matrix-Vector Multiplication: $\boldsymbol{Ax}$

Rule for $\boldsymbol{Ax}$

Multiply column i of $\boldsymbol{A}$ by x_i and sum.

Example:

$$\begin{bmatrix} 1 & -1 \\ 3 & -4 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} = 2 \cdot \begin{bmatrix} 1 \\ 3 \\ -1 \end{bmatrix} + 1 \cdot \begin{bmatrix} -1 \\ -4 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}$$

Matrix-Vector Multiplication: A Recipe

Example: Party Planning

- We want to make **Tacos** and **Pizza**.
- **Matrix *A* (Recipes):** Ingredients needed per item.
- **Vector *x* (Menu):** How many of each we want.

$$\begin{array}{cc} & \begin{array}{cc} \text{🍕} & \text{🌮} \end{array} \\ \begin{array}{c} \text{🧀} \\ \text{🍚} \\ \text{🍖} \end{array} & \begin{bmatrix} 2 & 1 \\ 1 & 3 \\ 0.5 & 1 \end{bmatrix} \end{array} \times \begin{array}{c} \text{🍕} \\ \text{🌮} \end{array} \begin{bmatrix} 5 \\ 10 \end{bmatrix} = 5 \cdot \begin{bmatrix} 2 \\ 1 \\ 0.5 \end{bmatrix} + 10 \cdot \begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 10 \\ 5 \\ 2.5 \end{bmatrix} + \begin{bmatrix} 10 \\ 30 \\ 10 \end{bmatrix} = \begin{bmatrix} 20 \\ 35 \\ 12.5 \end{bmatrix} \begin{array}{c} \text{🧀} \\ \text{🍚} \\ \text{🍖} \end{array}$$

Result: Total shopping list!

Vector-Matrix Multiplication: $\boldsymbol{x}A$

Rule for $\boldsymbol{x}A$

Multiply element i of the vector with row i of the matrix and sum.

Example: Government Budget

- **Vector $\boldsymbol{x}$ (Investments):** Budget allocated to each sector.
- **Matrix $\boldsymbol{A}$ (Impact):** Each investment's contribution to outcomes.

$$\begin{array}{c} \begin{matrix}\text{📚} & \text{🏥} & \text{🏗️}\end{matrix} \\ \begin{bmatrix}2 & 3 & 1\end{bmatrix}\end{array} \times \begin{array}{cc} & \begin{matrix}\text{😊} & \text{🛡️} & \text{💰}\end{matrix} \\ \begin{matrix}\text{📚}\\ \text{🏥}\\ \text{🏗️}\end{matrix} & \begin{bmatrix}3 & 1 & 2\\ 2 & 4 & 1\\ 1 & 2 & 5\end{bmatrix}\end{array} = \begin{bmatrix}6 & 2 & 4\end{bmatrix} + \begin{bmatrix}6 & 12 & 3\end{bmatrix} + \begin{bmatrix}1 & 2 & 5\end{bmatrix} = \begin{array}{c}\begin{matrix}\text{😊} & \text{🛡️} & \text{💰}\end{matrix} \\ \begin{bmatrix}13 & 16 & 12\end{bmatrix}\end{array}$$

Total impact: 13 happiness, 16 security, 12 wealth units

Recall: Generalization

Generalization

The ability of a model to perform well on **new, unseen data** (test set), not just the data it was trained on.

- **Training Set:** Examples given to the model to learn from (e.g., historical player data).
- **Test Set:** New examples hidden from the model during training, used to evaluate performance (e.g., new rookies).

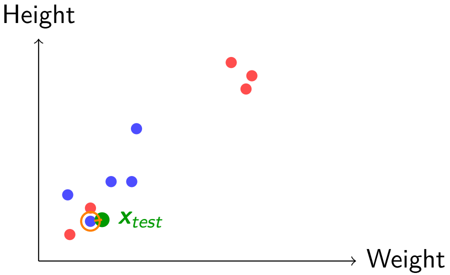
We care about how well we predict the unseen, not memorize observations!

Algorithm

Find the training point closest to $\boldsymbol{x}_{test}$ and copy its label.

Steps:

1. Compute distance from $\boldsymbol{x}_{test}$ to every training point
2. Find the training point with minimum distance
3. Assign its label to $\boldsymbol{x}_{test}$



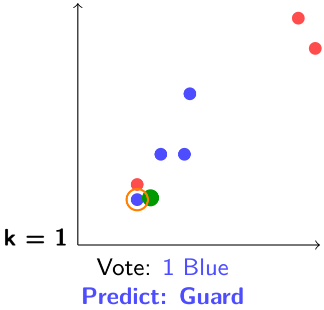
Predict: Guard

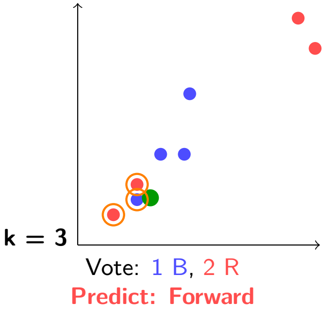
k-Nearest Neighbors (k-NN)

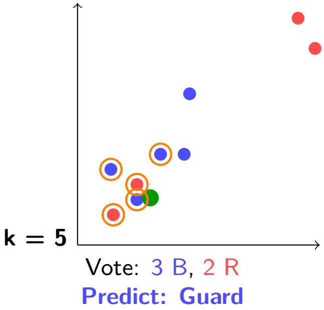
Algorithm

Find the k closest training points and take a **majority vote**.

Scenario: Test player at 180 lbs, 71 inches. Guard or Forward?







Later: How do we choose k?

The Argmin Operation

Definition

$\arg\min_i f(i)$ returns the **index** i that minimizes $f(i)$, not the minimum value itself.

Example: Let distances $= [5.2, 3.1, 8.7, 2.4, 6.0]$

- $\min(\text{distances}) = 2.4$ (the smallest value)
- $\arg\min(\text{distances}) = 3$ (the index of 2.4, using 0-indexing)

In Python/NumPy:

- `np.min(distances)` $\rightarrow$ `2.4`
- `np.argmin(distances)` $\rightarrow$ `3`

For kNN

We need argmin to find *which* training point is closest, then look up its label.

How Do We Measure "Closest"?

Key Question

What does it mean for two points to be "close"?

Consider two points:

$$\boldsymbol{x} = (x_1, x_2, \ldots, x_d) \quad \text{and} \quad \boldsymbol{z} = (z_1, z_2, \ldots, z_d)$$

Common distance metrics:

Euclidean (L2) Distance:

$$d_2(\boldsymbol{x}, \boldsymbol{z}) = \sqrt{\sum_{j=1}^{d} (x_j - z_j)^2}$$

"Straight line" distance

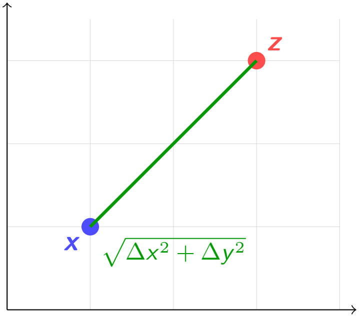
Manhattan (L1) Distance:

$$d_1(\boldsymbol{x}, \boldsymbol{z}) = \sum_{j=1}^{d} |x_j - z_j|$$

"City block" distance

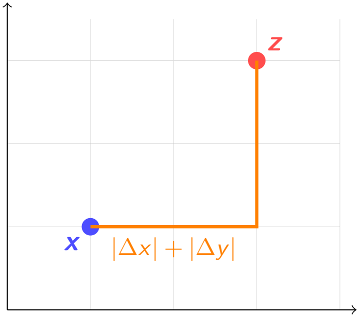
Visualizing L1 vs L2 Distance

L2 (Euclidean)



Straight Line

L1 (Manhattan)



Grid Path

The General Lp Norm

Norm is the distance from the origin. Measures the size of a vector.

$$\|\boldsymbol{x}\| = d(\boldsymbol{x}, \mathbf{0})$$

Distance is the norm of the difference vector.

$$d(\boldsymbol{x}, \boldsymbol{z}) = \|\boldsymbol{x} - \boldsymbol{z}\|$$

Lp Norm

$$\|\boldsymbol{x}\|_p = \left(\sum_{j=1}^{d} |x_j|^p \right)^{1/p}$$

Lp Distance

$$d_p(\boldsymbol{x}, \boldsymbol{z}) = \left(\sum_{j=1}^{d} |x_j - z_j|^p \right)^{1/p}$$

We usually talk about norms for simplicity.

Common Lp Norms

Lp Norm

$$\|\boldsymbol{x}\|_p = \left(\sum_{j=1}^{d} |x_j|^p\right)^{1/p}$$

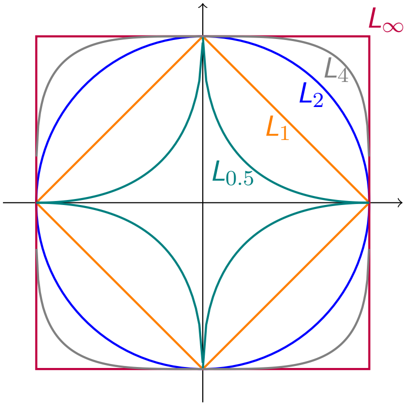
- $p = 1$ (Manhattan / Taxicab): $\|\boldsymbol{x}\|_1 = |x_1| + |x_2| + \cdots + |x_d|$
- $p = 2$ (Euclidean): $\|\boldsymbol{x}\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_d^2} = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle}$ (square root of the inner product of the vector with itself)
- $p = \infty$ (Max / Chebyshev): $\|\boldsymbol{x}\|_\infty = \max_j |x_j|$

Example: $\boldsymbol{x} = \begin{bmatrix} 3 & -4 \end{bmatrix}$

- $\|\boldsymbol{x}\|_1 = |3| + |-4| = 7$
- $\|\boldsymbol{x}\|_2 = \sqrt{9 + 16} = 5$
- $\|\boldsymbol{x}\|_\infty = \max(3, 4) = 4$

Unit Circles (Contours of Norms)

Each curve represents the set of points at a unit distance away from the origin in L_p metric.



L2 (Euclidean):

- **Rotationally Invariant**: Distance doesn't change if you rotate the axes.
- "As the crow flies"

L1 (Manhattan):

- **Axis Dependent**: Axes have specific, distinct meanings (e.g., Age vs. Salary).
- Less susceptible to outliers in one dimension (each axis contributes linearly).

Why NumPy?

Without NumPy:

- Python lists are slow for math
- Must write explicit loops
- Memory inefficient

With NumPy:

- Fast vectorized operations
- Clean, readable code
- Built-in math functions

Example: Add two vectors

Python lists:

```
[a[i]+b[i] for i in range(n)]
```

NumPy arrays:

```
a + b
```

NumPy is often **100x faster!**

Computing Distances in NumPy

The `np.linalg.norm` function:

```
np.linalg.norm(x - z, ord=2)
```

- Computes the Lp norm of the difference vector
- `ord=2` (default): Euclidean distance
- `ord=1`: Manhattan distance
- `ord=np.inf`: Maximum absolute difference

Example:

```
x = np.array([3, 4])
z = np.array([0, 0])
np.linalg.norm(x - z) → 5.0
```

(This is $\sqrt{3^2 + 4^2} = \sqrt{25} = 5$)

Today's Notebook Activities

Prerequisites from HW01:

- §7: NumPy arrays, vectors, matrices
- §7.3: Slicing and indexing (`M[i, :]`)
- §8: Boolean indexing, `np.argmax`

Today's NEW activities (in notebook):

1. **Activity 1: Computing Distances**
   - Use `np.linalg.norm` to compute Euclidean distance
2. **Activity 2: Finding the Nearest Neighbor**
   - Compute distances to all training points
   - Use `np.argmin` to find the nearest
3. **Activity 3: Classify All Test Points**
   - Implement 1-NN for multiple test points

Note

Complete HW01 §7–8 first if you haven't already!

Key Takeaways

1. **k-Nearest Neighbors:**
   - Find k closest training points
   - Take majority vote to classify
   - Simple but powerful!
2. **Distance Metrics:**
   - L2 (Euclidean): straight-line distance
   - L1 (Manhattan): city-block distance
   - Use `np.linalg.norm(x-z, ord=p)`
3. **NumPy Essentials:**
   - Arrays have `.shape` and `.dtype`
   - Indexing: `M[row, col]`, slicing with `:`
   - Vectorized operations are fast!

Next Time

We'll implement kNN from scratch and apply it to real data!

Summary

k-Nearest Neighbors Algorithm:

1. Find k closest training points to test point
2. Take majority vote of their labels
3. Prediction depends on distance metric and k

Key Concepts:

- Vectors represent data points
- Matrices organize datasets (rows = samples, columns = features)
- Distance metrics: L1 (Manhattan), L2 (Euclidean), Lp
- Matrix operations: Ax (combines columns), xA (combines rows)