Data Science – Fall 2025/2026

INTRODUCTION TO Machine Learning

Lecture 5:
Nearset Neighbors

Nearset Neighbors Method: Definition

The *Nearest Neighbors method* is a machine learning approach that makes decisions based on the data points that are closest to a new, unknown point.

To classify or predict something, the algorithm looks at the **nearest** existing examples (based on distance) and uses them to determine the answer.

In other words, to guess something about a new data point, the idea is to look at the most similar points around it and use their values.

Nearset Neighbors Method: Definition

The Nearest Neighbors method is both, supervised and unsupervised method ,depending on how it is used.

Supervised

- k-Nearest Neighbors (kNN) for classification or regression

Unsupervised

- Nearest Neighbor Search (finding similar items)
- k-Nearest Neighbors for anomaly detection

Nearset Neighbors Method: Definition

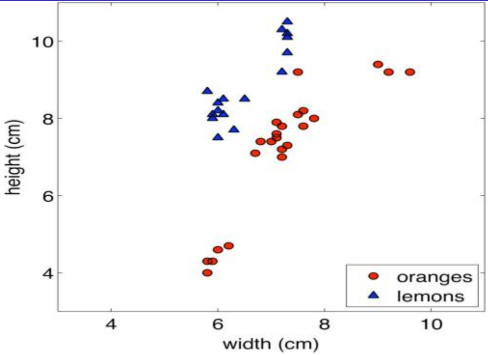
The Nearest Neighbors method (especially kNN) is:

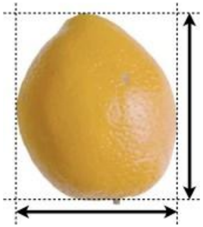
1. Based on distance: Decisions depend on measuring how **close** (similar) a new point is to existing points using a distance metric such as: Euclidian, Minkowski, Manhattan.

2. A non-Parametric Model:
A non-Parametric model does not assume any specific formula between the explanatory variables and the target variable. Instead of using a fixed mathematical shape (like a line or curve), the model lets the data determine the structure. As more data is available, the model can become more flexible and more complex.

N.B. In this course, we will mainly use the Nearest Neighbors method for classification, but it can also be applied to regression. Since both tasks require labeled data, this means that k-Nearest Neighbors is used as a supervised learning technique.

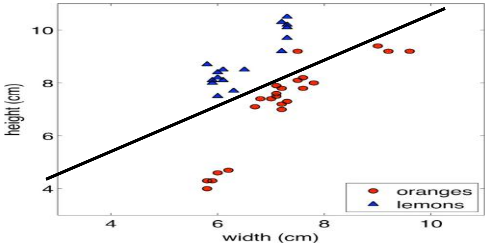
Classification: Oranges and Lemons





We can distinguish between oranges and lemons based on their measurements.
In the context of k-Nearest Neighbors, for a given a new unlabeled fruit,
It will be plot on the graph based on its height and width.
To classify it, we simply look at the "k" closest points to it (nearest neighbors). For example, the 3 closest point and whichever fruit type is most common among those neighbors would be the predicted type for the new fruit.

Classification: Oranges and Lemons



We can construct simple linear decision boundary:

$$y = \text{sign}(w_0 + w_2 x_2 + w_1 x_1)$$

While k-Nearest Neighbors classifies a new fruit by asking "What are my closest neighbors?", the linear classifier asks a direct question:
On which side of this specific line does the new fruit lie ?

Key Difference

•**k-NN** (non-Parametric Model)**:** No pre-defined boundary. The decision is made locally based on surrounding points.

•**Linear Classifier** (Parametric Model)**:** A single, global boundary is created. The decision is a simple "which side?"

Instance-based Learning

Instance-based Learning is a machine learning approach where the algorithm memorizes the training examples (instances) rather than learning an explicit model or rule. When a new observation is presented to classify, the algorithm compares it directly to all stored Instances using a similarity measure (like distance), and makes predictions based on the most similar stored examples.

The k-Nearest Neighbors (kNN) algorithm is the most common example of instance-based learning, because it stores all training data and makes decisions by finding the k closest instances to a new point.

Instance-based VS Model-based Learning

1. Lazy Learning: No work is done until a prediction is needed. The algorithm doesn't do any real learning or training when we give it the data. It simply stores the data in its memory and goes to sleep. This is the case of Instance-based methods.

2. Eager Learner: The linear classifier is an eager learner. As soon as we give it the data, it does all the hard work of calculating the perfect line (w_0, w_1, w_2). It learns a model.

3. Memory-based: Because a lazy learner (like k-Nearest Neighbors) does not build a model during training, it must keep all the training data. When making a prediction, it needs access to every example it has seen so it can compare the new point to the stored instances. In contrast, a linear classifier learns an explicit model. Its decision boundary is defined by an equation such as: $y = sign(w_0 + w_1 \cdot x_1 + w_2 \cdot x_2)$
Once these weights are learned, the model no longer needs the original data. It has compressed or summarized the information into a compact set of parameters and can safely discard the training observations.

Instance-based VS Model-based Learning

4. Local approximation

Decisions are based on local patterns rather than global rules.

k-NN does not try to learn one global rule for all cases, instead, for each new observation, it focuses only on the small local neighborhood around it and makes a decision based on those nearby examples. In contrast, a linear classifier learns a single global rule (a straight-line boundary) that aims to correctly classify any new observation anywhere on the graph.

1-Nearest Neighbor (1-NN Algorithm)

Concept: Each training example exists as a point in p-dimensional space: $x \in \mathbb{R}^p$

Core principle: Estimate the label of a new query point using the label of its closest training example

Distance Measurement: We typically use **Euclidean Distance** to find the nearest neighbor.

Algorithm:

1. Find $(\mathbf{x}^*, t^*)$ (from the stored training dataset) closest to the test instance $\mathbf{x}$

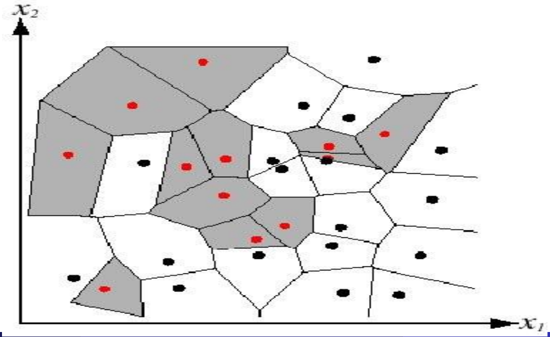
$$\mathbf{x}^* = \underset{\mathbf{x}^{(i)} \in \texttt{train. set}}{\operatorname{argmin}} \ \texttt{distance}(\mathbf{x}^{(i)}, \mathbf{x})$$

2. Output $y = t^*$

Nearest Neighbors: Decision Boundaries

A decision boundary is the line that separates different classes in a classification problem. It's where the model switches from predicting one class to another.

A Voronoi diagram divides space into regions. Each region contains all the locations that are closest to one particular point. This diagram shows exactly how **1-nearest neighbor classification** works: Every new point is assigned to the same region (and class) as the closest training point. Each boundary line in a Voronoi diagram consists of points that are equidistant between two training points.



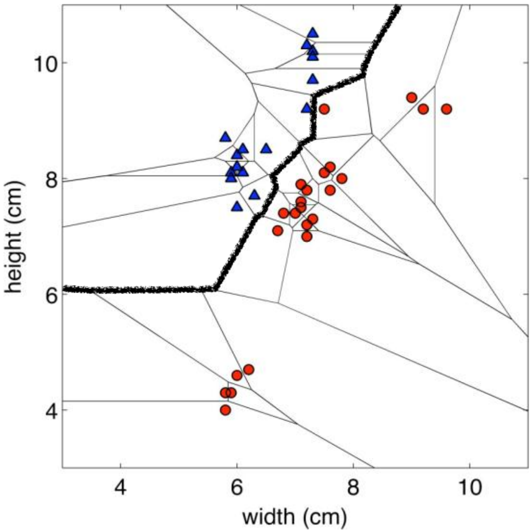
Nearest Neighbors: Decision Boundaries

Decision boundaries help diagnose model behavior by visually revealing whether a model is overfitting, underfitting, or making biased assumptions. Overfitting happens when the model creates an overly complicated, curved boundary that tries to fit every training point perfectly. This usually means the model is memorizing noise instead of learning the real pattern.
Underfitting shows as overly simplistic, straight boundaries that miss important data relationships, while biased models display boundaries that force inappropriate separations (like linear splits).

By examining boundary complexity and shape, we can immediately assess if a model has the right balance of simplicity and complexity to generalize well to new data.

Nearest Neighbors: Decision Boundaries

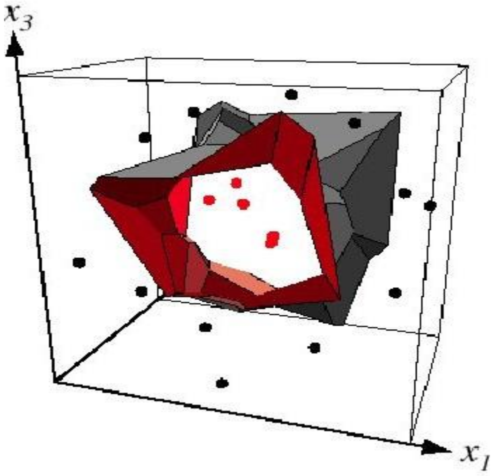
Separation curve



Example: 2D decision boundary

Nearest Neighbors: Decision Boundaries

Separation surface



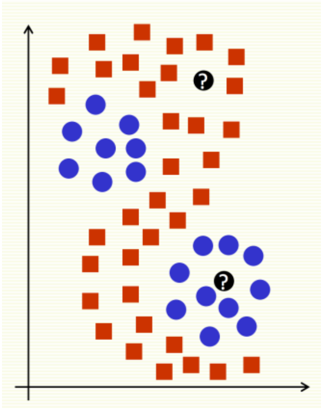
Example: 3D decision boundary

Nearest Neighbors: Multi-modal Data

- Nearest Neighbor approaches can work with multi-modal data

What is Multi-modal Data? Data where each class appears in multiple separate clusters

Class A: 🟥 🟥 (Cluster 1) + 🟥 🟥 (Cluster 2)
Class B: 🔵 🔵 (Cluster 1) + 🔵 🔵 (Cluster 2)



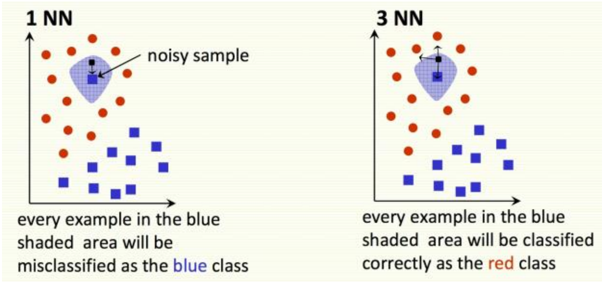
Example

Apples = Green apples + Red apples
Oranges = Tangerines + Navel oranges

Key Characteristics

- Complex, separate patterns
- Common in real-world data
- Multiple distinct groups per class
- Challenging for simple linear models

k-Nearest Neighbors


1 NN
noisy sample
every example in the blue shaded area will be misclassified as the blue class
3 NN
every example in the blue shaded area will be classified correctly as the red class

- Nearest neighbors sensitive to mis-labeled data ("class noise"). Solution?
- Smooth by having k nearest neighbors vote

k-Nearest Neighbors: The Algorithm

1. **Choose a value for k** (the nbr. of considered neighbors)

2. **Calculate the distance**
from the new point to every point in the training dataset
(commonly using Euclidean distance).

3. **Sort the distances** (from smallest to largest)

4. **Select the k closest points** (the k nearest neighbors)

5. **For classification:**
    1. Look at the labels of these k neighbors.
    2. The new point is assigned to the most common label
6. **For regression:**
    1. Take the **average** (or median) of the neighbors’ values
       to predict the new value.

k-Nearest Neighbors

How do we choose k ?

- Larger k may lead to better performance
- But if we set k too large we may end up looking at samples that are not neighbors
- We can use cross-validation to find k
- Rule of thumb is $k < sqrt(n)$, where n is the number of training examples
- Rule of thumb is not a strict rule, but practical to avoid:

  k too small → high variance, overfitting
  k too large → high bias, oversmoothing
  Using $\sqrt{n}$ gives a balance between these two extremes.

k-Nearest Neighbors: Issues & Remedies

1. Features with larger numeric ranges dominate the distance

If one feature has big values and another has small values, the big one affects the distance more.

Solution: normalize the data, such as:
•Scale each feature to the range [0, 1]
•Standardize: subtract the mean and divide by the standard deviation

N.B. Sometimes the original scale is actually meaningful.

k-Nearest Neighbors: Issues & Remedies

2. Irrelevant or highly correlated features cause noise

Extra features that don’t help can confuse the distance calculation.

Solutions:

- Remove useless features
- Adjust feature weights

k-Nearest Neighbors: Issues & Remedies

3. Non-numeric features

For symbolic categories (e.g., “red”, “blue”)

Solution

Special distance measure like **Hamming distance**.

Limitations of K-NN

1. **Slow at prediction time**

- To classify one new point, k-NN must compute the distance to every training example.
- This can be expensive when the dataset is large.

How to speed it up?

- Use fewer features (dimension reduction like PCA technique)
- Remove unnecessary or duplicate training points

Limitations of K-NN

2. High Storage Requirement

- k-NN must keep all training data because it doesn’t learn a model.

Possible solutions

- Remove redundant or unhelpful points
- N.B. Some indexing structures can increase memory use

Limitations of K-NN

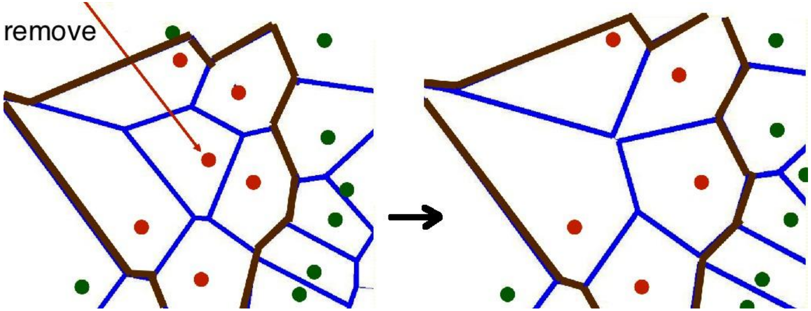
3. Problems with high-dimensional data

When the number of features is large:

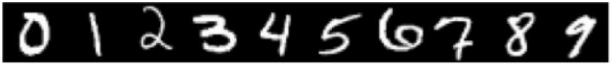
- Much more data to make k-NN work well is needed
- Distance calculations become less meaningful
- Computation becomes slower and more expensive

k-Nearest Neighbors Remedies: Remove Redundancy

- If all Voronoi neighbors have the same class, a sample is useless, remove it



Example: Digit Classification



- Yann LeCunn – MNIST Digit Recognition
  - Handwritten digits
  - 28x28 pixel images: $d = 784$
  - 60,000 training samples
  - 10,000 test samples
- Nearest neighbour is competitive

	Test Error Rate (%)
Linear classifier (1-layer NN)	12.0
K-nearest-neighbors, Euclidean	5.0
K-nearest-neighbors, Euclidean, deskewed	2.4
K-NN, Tangent Distance, 16x16	1.1
K-NN, shape context matching	0.67
1000 RBF + linear classifier	3.6
SVM deg 4 polynomial	1.1
2-layer NN, 300 hidden units	4.7
2-layer NN, 300 HU, [deskewing]	1.6
LeNet-5, [distortions]	0.8
Boosted LeNet-4, [distortions]	0.7

Fun Example: Where on Earth is this Photo From?

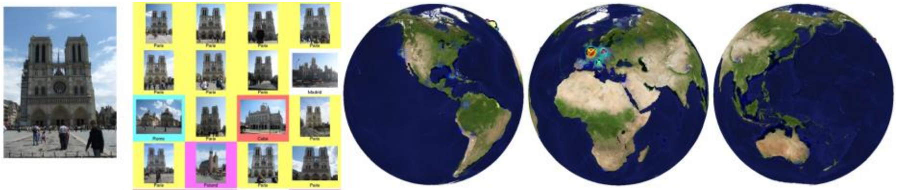
- Problem: Where (e.g., which country or GPS location) was this picture taken?

[Paper: James Hays, Alexei A. Efros. im2gps: estimating geographic information from a single image. CVPR'08. Project page: http://graphics.cs.cmu.edu/projects/im2gps/]

Fun Example: Where on Earth is this Photo From?

- Problem: Where (e.g., which country or GPS location) was this picture taken?

  Get 6M images from Flickr with GPs info (dense sampling across world)
  Represent each image with meaningful features
  Do kNN!

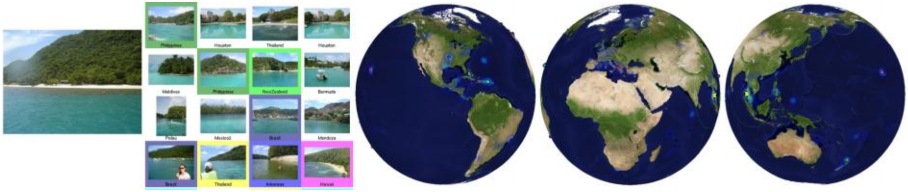


[Paper: James Hays, Alexei A. Efros. im2gps: estimating geographic information from a single image. CVPR'08. Project page: http://graphics.cs.cmu.edu/projects/im2gps/]

Fun Example: Where on Earth is this Photo From?

- Problem: Where (eg, which country or GPS location) was this picture taken?

  Get 6M images from Flickr with gps info (dense sampling across world)
  Represent each image with meaningful features
  Do kNN (large k better, they use $k = 120$)!



[Paper: James Hays, Alexei A. Efros. im2gps: estimating geographic information from a single image. CVPR'08. Project page: http://graphics.cs.cmu.edu/projects/im2gps/]