Artificial Intelligence

BS (CS) _Spring_2025

Lab_12 Manual

Learning Objectives:

1. Implementing K-Nearest Neighbor (KNN) algorithm to classify the data set

Lab Manual

K-Nearest Neighbor (KNN) algorithm:

Introduction:

K-Nearest Neighbor (KNN) is a simple, versatile, instance-based algorithm used for both classification and regression in domains such as finance (credit scoring, loan risk), healthcare, political science (voter prediction), handwriting, image and video recognition. It labels a query point by finding the k closest training examples in feature space and using their majority vote (or average for regression).

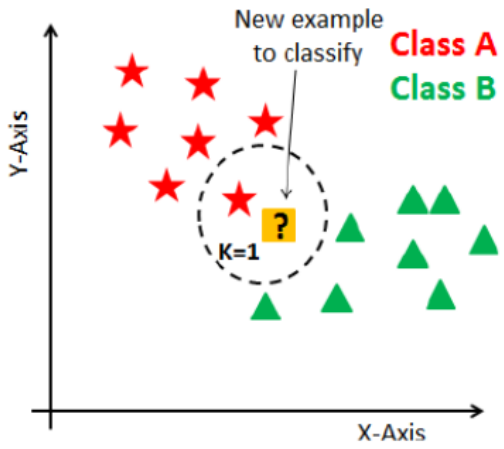
As a non-parametric method, KNN makes no assumptions about data distribution, its "model" is just the dataset itself. And as a lazy learner, it skips any model-building during training, doing all the work at prediction time. This makes training very fast but inference slower and more memory-intensive, since each prediction must scan the entire training set.

How does the KNN algorithm work?:

In KNN, K is the number of nearest neighbors. The number of neighbors is the core deciding factor. K is generally an odd number if the number of classes is 2.

When K = 1:

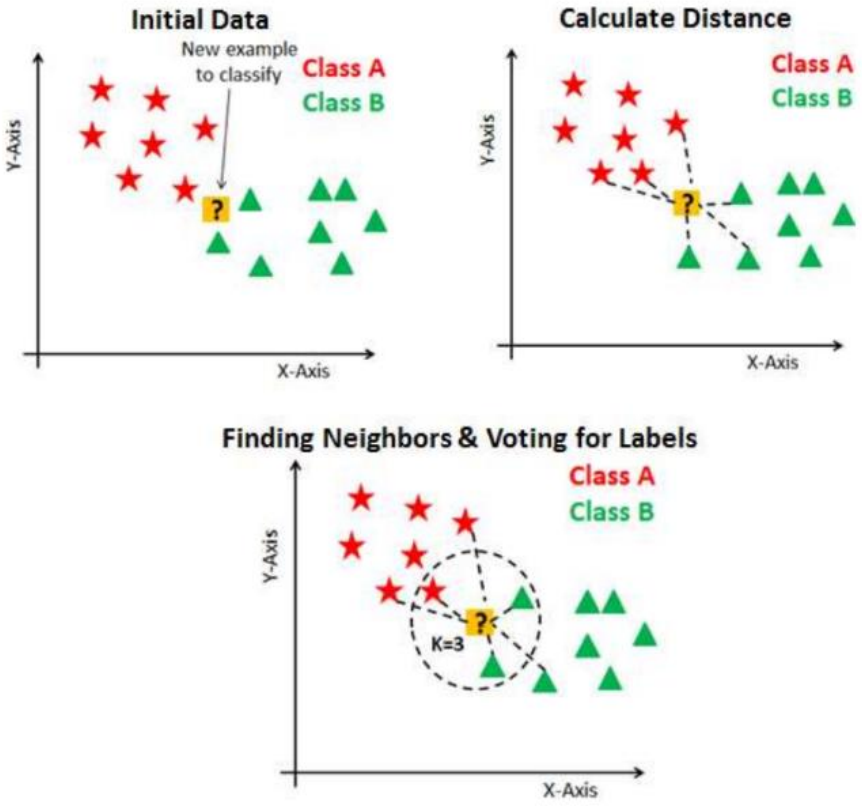
the algorithm reduces to the basic nearest neighbor method: for a query point P_1, you compute its distance to every training example (using, say, Euclidean, Manhattan, or Hamming distance), find the single closest point, and simply assign P_1 that point's label.



When K > 1, the process becomes:

1. Compute distances between P_1 and all training points.
2. Select the K nearest points (the smallest distances).
3. Vote: each neighbor casts one vote for its class; the class with the most votes is the prediction for P_1.

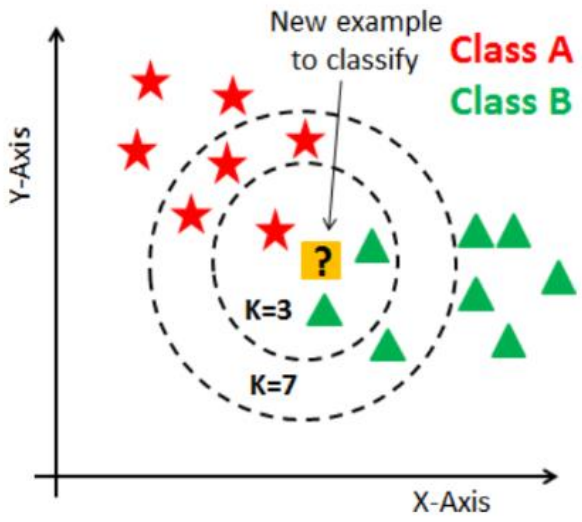
Using an odd K (especially with two classes) avoids ties. A small K makes the decision boundary sensitive to noise (high variance), while a larger K yields smoother, more stable boundaries (higher bias).



How to decide K value:

Choosing the Right K

- **Small K (e.g. 1–5)**
  - Looks only at a handful of nearby points
  - Can jump around if one of those points is unusual
  - Very responsive to small changes in the data
- **Large K (e.g. 10–20+)**
  - Averages over many neighbors
  - Gives steadier, smoother results
  - May miss finer local patterns



KNN Pseudocode

Input: training set T, query point q, parameter k

1. For each x in T, compute d(x, q)
2. Sort all x by d(x, q) ascending
3. Let N = {k closest x's}
4. Return majority_label(N) (or average for regression)

Distance formulas:

1. **Euclidean distance** (ℓ_2 norm)

$$d_{\text{Euclidean}}(\mathbf{v}_1, \mathbf{v}_2) = \sqrt{\sum_{i=1}^{n}(v_{1,i} - v_{2,i})^2}$$

2. **Manhattan distance** (ℓ_1 norm)

$$d_{\text{Manhattan}}(\mathbf{v}_1, \mathbf{v}_2) = \sum_{i=1}^{n}\left| v_{1,i} - v_{2,i}\right|$$

3. **Minkowski distance** (ℓ_p norm)

$$d_{\text{Minkowski}}(\mathbf{v}_1, \mathbf{v}_2) = \left(\sum_{i=1}^{n}\left|v_{1,i} - v_{2,i}\right|^p\right)^{\frac{1}{p}}$$