K-Nearest Neighbors (KNN): Mathematical Notes

Let us consider a small dataset for a classification problem. K-Nearest Neighbors (KNN) is a non-parametric, instance-based learning algorithm used for classification and regression.

Dataset: Exam Score Classification

Suppose we want to classify students as **Pass (+1)** or **Fail (-1)** based on two features:

- x_1 = Math Score
- x_2 = Programming Score

Student	x_1	x_2	Class (y)
1	2	3	+1
2	3	4	+1
3	5	6	-1
4	6	5	-1

Goal: Classify a new student with features $x_{new} = (4, 4)$ using KNN.

I. K-Nearest Neighbors (KNN) Algorithm

KNN does not build an explicit model. Instead, it stores the training dataset and classifies new points based on distance to the nearest neighbors.

Distance Metrics in KNN

The performance of KNN depends heavily on the chosen distance measure. Below are the most commonly used distance functions.

1. Euclidean Distance (L2 Norm)
For two points x_i and x_j in d-dimensional space:

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{d} (x_{ik} - x_{jk})^2}$$

For 2D space:

$$d(x_i, x_j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2}$$

Used when geometric distance is meaningful.

2. Manhattan Distance (L1 Norm)

$$d(x_i, x_j) = \sum_{k=1}^{d} |x_{ik} - x_{jk}|$$

For 2D space:

$$d(x_i, x_j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}|$$

Used when movement is grid-like (city block distance).

3. Minkowski Distance (General Form)

$$d(x_i, x_j) = \left(\sum_{k=1}^{d} |x_{ik} - x_{jk}|^p \right)^{1/p}$$

Special cases:

- $p = 1 \rightarrow$ Manhattan Distance
- $p = 2 \rightarrow$ Euclidean Distance

4. Chebyshev Distance (L Norm)

$$d(x_i, x_j) = \max_k |x_{ik} - x_{jk}|$$

Used when the maximum coordinate difference determines similarity.

5. Hamming Distance
Used for categorical or binary features:

$$d(x_i, x_j) = \sum_{k=1}^{d} I(x_{ik} \neq x_{jk})$$

where $I(\cdot)$ is the indicator function.

6. Cosine Distance
First compute cosine similarity:

$$\text{Cosine Similarity} = \frac{x_i \cdot x_j}{\|x_i\|\|x_j\|}$$

Cosine distance:

$$d(x_i, x_j) = 1 - \frac{x_i \cdot x_j}{\|x_i\|\|x_j\|}$$

Used in text mining and high-dimensional sparse data.

7. Mahalanobis Distance

$$d(x_i, x_j) = \sqrt{(x_i - x_j)^T S^{-1} (x_i - x_j)}$$

where:

- $S =$ covariance matrix

Used when features are correlated.

Summary of Distance Choices

- Euclidean $\rightarrow$ Most common for continuous features.
- Manhattan $\rightarrow$ Robust to outliers.
- Minkowski $\rightarrow$ Generalized distance framework.
- Chebyshev $\rightarrow$ Maximum coordinate difference.
- Hamming $\rightarrow$ Categorical data.
- Cosine $\rightarrow$ Direction-based similarity.
- Mahalanobis $\rightarrow$ Accounts for feature correlation.

II. Step-by-Step Numerical Calculation

We classify $x_{new} = (4, 4)$.

Step 1: Compute Euclidean Distances

Distance to Student 1 $(2, 3)$:

$$d_1 = \sqrt{(4-2)^2 + (4-3)^2}$$
$$d_1 = \sqrt{4+1} = \sqrt{5} \approx 2.24$$

Distance to Student 2 $(3, 4)$:

$$d_2 = \sqrt{(4-3)^2 + (4-4)^2}$$
$$d_2 = \sqrt{1+0} = 1$$

Distance to Student 3 $(5, 6)$:

$$d_3 = \sqrt{(4-5)^2 + (4-6)^2}$$
$$d_3 = \sqrt{1+4} = \sqrt{5} \approx 2.24$$

Distance to Student 4 $(6, 5)$:

$$d_4 = \sqrt{(4-6)^2 + (4-5)^2}$$
$$d_4 = \sqrt{4+1} = \sqrt{5} \approx 2.24$$

Step 2: Sort Distances

Student	Distance	Class
2	1.00	+1
1	2.24	+1
3	2.24	-1
4	2.24	-1

Step 3: Choose $K = 3$

The three nearest neighbors are:

- Student 2 (+1)
- Student 1 (+1)
- Student 3 (-1)

Majority class = +1

Final Classification

$$\hat{y} = +1$$

Therefore, the new student is classified as **Pass**.

Discussion

- **Lazy Learning Paradigm:** KNN is an instance-based or lazy learning algorithm. It does not construct an explicit parametric model during training. Instead, the entire training dataset is stored, and computation is deferred until prediction time. While this eliminates training cost, it increases inference-time complexity.
- **Non-Parametric Nature:** KNN makes no prior assumptions about the underlying data distribution. Unlike parametric models (e.g., Linear Regression, Naïve Bayes), it does not assume linearity or Gaussian structure. This flexibility allows KNN to model highly non-linear decision boundaries.
- **Sensitivity to the Choice of K:** The value of K controls the bias–variance tradeoff:
  - Small $K \rightarrow$ low bias, high variance (prone to overfitting).

  - Large $K \rightarrow$ high bias, low variance (may underfit).

  Selecting an optimal K typically requires cross-validation.

- **Feature Scaling Requirement:** Since KNN relies on distance calculations, features with larger numerical ranges can dominate the metric. Therefore, normalization or standardization (e.g., Min–Max scaling or Z-score normalization) is essential before applying KNN.

- **Computational Complexity:** For each prediction, KNN computes distances between the query point and all training samples. Time complexity per query:

$$O(nd)$$

  where:

  - n = number of training samples
  - d = number of features

  For large datasets, this can become computationally expensive. Data structures such as KD-trees or Ball trees can reduce search time in low-dimensional settings.

- **Curse of Dimensionality:** As dimensionality increases, distances between points tend to become similar, reducing the discriminative power of nearest neighbors. In high-dimensional spaces, data becomes sparse, and KNN performance may degrade significantly.

- **Memory Requirement:** Because all training data must be stored, KNN has high memory consumption compared to parametric models.

- **Robustness Considerations:** KNN can be sensitive to noise and outliers, particularly when K is small. Weighted KNN variants can mitigate this issue by assigning larger weights to closer neighbors.

Prepared By:
Md. Atikuzzaman
Lecturer
Department of Computer Science and Engineering
Green University of Bangladesh
Email: atik@cse.green.edu.bd