Machine Learning 101

Rajdeep Chatterjee, Ph.D.
Amygdala AI, Bhubaneswar, India *

January 2025

Distance-based Learning

1 Distance Metric

A **distance metric** is a function that defines a distance between elements of a set. It is widely used in various fields such as machine learning, pattern recognition, and clustering to measure similarity or dissimilarity between objects.

2 Criteria for a Distance Metric

For a function $d(x,y)$ to qualify as a distance metric, it must satisfy the following criteria:

1. **Non-negativity:** $d(x,y) \geq 0 \quad \forall x,y$. The distance is always non-negative.
2. **Identity of Indiscernibles:** $d(x,y) = 0 \iff x = y$. The distance between two distinct points is zero only if they are the same point.
3. **Symmetry:** $d(x,y) = d(y,x)$. The distance is the same in both directions.
4. **Triangle Inequality:** $d(x,z) \leq d(x,y) + d(y,z)$. The direct distance between two points is less than or equal to the sum of intermediate distances.

3 Types of Distance Metrics

Here are some common types of distance metrics along with their equations and examples:

*Amygdala AI, is an international volunteer-run research group that advocates for *AI for a better tomorrow* `http://amygdalaai.org/`.

3.1 1. Euclidean Distance

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Example: The Euclidean distance between points $x = (1,2)$ and $y = (4,6)$ in 2D space is:

$$d(x,y) = \sqrt{(1-4)^2 + (2-6)^2} = 5$$

3.2 2. Manhattan Distance (Taxicab Distance)

$$d(x,y) = \sum_{i=1}^{n} |x_i - y_i|$$

Example: For the same points $x = (1,2)$ and $y = (4,6)$, the Manhattan distance is:

$$d(x,y) = |1-4| + |2-6| = 7$$

3.3 3. Minkowski Distance

$$d(x,y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$$

Example: When $p = 1$, it becomes Manhattan distance; when $p = 2$, it becomes Euclidean distance.

3.4 4. Cosine Similarity (Not a Metric, but Used to Measure Similarity)

$$\text{Cosine Similarity} = \frac{x \cdot y}{\|x\|\|y\|}$$

Example: If $x = (1,0,1)$ and $y = (0,1,1)$, the cosine similarity is:

$$\frac{(1)(0) + (0)(1) + (1)(1)}{\sqrt{(1^2 + 0^2 + 1^2)}\sqrt{(0^2 + 1^2 + 1^2)}} = \frac{1}{\sqrt{2} \cdot \sqrt{2}} = 0.5$$

3.5 5. Hamming Distance

$$d(x,y) = \sum_{i=1}^{n} \mathbf{1}(x_i \neq y_i)$$

Example: For binary strings $x = (1,0,1,1)$ and $y = (1,1,0,1)$, the Hamming distance is:

$$d(x,y) = 0 + 1 + 1 + 0 = 2$$

4 Distance-Based Learning and Nearest Neighbor

4.1 Distance-Based Learning

Distance-based learning refers to methods that rely on distance metrics to classify or cluster data. For example, in k-Nearest Neighbors (k-NN), the class of a point is determined by the majority class among its k nearest neighbors.

4.2 k-Nearest Neighbors (k-NN)

The k-NN algorithm works as follows:

1. Calculate the distance between the query point and all points in the dataset.
2. Select the k nearest neighbors based on the smallest distances.
3. Assign the class based on the majority vote among these neighbors.

Example: Given a query point q and labeled data points, if q's three nearest neighbors belong to classes $\{A, A, B\}$, then q is classified as A (majority class).

5 k-Nearest Neighbors (KNN)

5.1 Pseudo-code for KNN

Algorithm 1 k-Nearest Neighbors (KNN) Algorithm

```
1: Input: Dataset D = {(x_i, y_i)}_{i=1}^N, query point q, number of neighbors k
2: Output: Predicted class label y_q for query point q
3: Compute the distance d(q, x_i) for all points x_i in the dataset using a chosen distance metric (e.g., Euclidean distance).
4: Sort the distances in ascending order.
5: Select the k-nearest points to q.
6: Perform a majority vote among the k-nearest neighbors to determine the class label y_q.
7: Return y_q.
```

5.2 Choosing a Suitable k

To choose a suitable k, consider the following:

1. **Cross-Validation:** Use cross-validation on your dataset to test different values of k and choose the one that minimizes the classification error.
2. **Odd Values:** Prefer odd values of k to avoid ties in classification (especially for binary classification).
3. **Small k:** A small k can make the model sensitive to noise, leading to overfitting.

4. **Large k:** A large k can make the model overly general, leading to underfitting.
5. **Rule of Thumb:** Start with $k = \sqrt{N}$, where N is the number of data points, and adjust based on performance.

5.3 Numerical Toy Example

Dataset:

$$D = \{(x_1 = [2,4], y_1 = A), (x_2 = [4,6], y_2 = B), (x_3 = [5,4], y_3 = B), (x_4 = [6,2], y_4 = A), (x_5 = [1,3], y_5 = A)\}$$

Query Point:

$$q = [3,5]$$

5.3.1 Step 1: Compute Distances

Use Euclidean distance:

$$d(x_i, q) = \sqrt{\sum_{j=1}^{n} (x_{i,j} - q_j)^2}$$

$$\begin{aligned}
d(x_1, q) &= \sqrt{(2-3)^2 + (4-5)^2} = \sqrt{2} \approx 1.41, \\
d(x_2, q) &= \sqrt{(4-3)^2 + (6-5)^2} = \sqrt{2} \approx 1.41, \\
d(x_3, q) &= \sqrt{(5-3)^2 + (4-5)^2} = \sqrt{5} \approx 2.24, \\
d(x_4, q) &= \sqrt{(6-3)^2 + (2-5)^2} = \sqrt{18} \approx 4.24, \\
d(x_5, q) &= \sqrt{(1-3)^2 + (3-5)^2} = \sqrt{8} \approx 2.83.
\end{aligned}$$

5.3.2 Step 2: Sort Distances

$$\text{Sorted Distances: } x_1, x_2, x_3, x_5, x_4.$$

5.3.3 Step 3: Select k-Nearest Neighbors

For $k = 3$, the nearest neighbors are:

$$x_1(A), x_2(B), x_3(B).$$

5.3.4 Step 4: Majority Vote

Classes of neighbors: $\{A, B, B\}$. Majority class is B.

5.3.5 Result:

The predicted class for q is B.

6 Frequently Asked Questions (FAQs) on k-Nearest Neighbors (KNN)

6.1 1. What is k-Nearest Neighbors (KNN)?

KNN is a simple, non-parametric, and lazy learning algorithm used for classification and regression. It classifies a data point based on the majority class of its k-nearest neighbors in the feature space.

6.2 2. What is the role of the parameter k?

The parameter k determines the number of nearest neighbors to consider for classification or regression:

- A small k (e.g., $k = 1$) makes the model sensitive to noise (overfitting).
- A large k leads to a smoother decision boundary but risks underfitting.

6.3 3. How is the distance between points calculated?

The distance between points can be calculated using various metrics, such as:

- **Euclidean Distance:** $d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$.
- **Manhattan Distance:** $d(x,y) = \sum_{i=1}^{n}|x_i - y_i|$.
- **Minkowski Distance:** $d(x,y) = \left(\sum_{i=1}^{n}|x_i - y_i|^p\right)^{1/p}$.
- **Hamming Distance:** For categorical data, $d(x,y) = \sum_{i=1}^{n}\mathbf{1}(x_i \neq y_i)$.

6.4 4. Is KNN suitable for high-dimensional data?

KNN performance may degrade with high-dimensional data due to the **curse of dimensionality**. As the number of dimensions increases, all points become equidistant, making it difficult to find meaningful nearest neighbors.

5. How do you handle missing data in KNN?

To handle missing data in KNN:

- Impute missing values using statistical methods (e.g., mean or median imputation).
- Use only the dimensions without missing values for distance computation.

6.5 6. How does KNN perform for imbalanced datasets?

KNN can struggle with imbalanced datasets, as the majority class may dominate the nearest neighbors. Solutions include:

- Using weighted distances to give closer points higher influence.
- Balancing the dataset using oversampling or undersampling techniques.

6.6 7. What are the advantages of KNN?

- Simple to implement and understand.
- Makes no assumptions about the data distribution (non-parametric).
- Effective for small datasets with low dimensionality.

6.7 8. What are the disadvantages of KNN?

- Computationally expensive during prediction, as it calculates distances for all data points.
- Sensitive to irrelevant features and the choice of distance metric.
- Poor performance with high-dimensional or noisy data.

6.8 9. How do you optimize KNN performance?

To optimize KNN performance:

- Perform feature scaling (e.g., normalization or standardization) to ensure all features contribute equally to distance computations.
- Select a suitable k using cross-validation.
- Reduce dimensionality using techniques like PCA (Principal Component Analysis).

6.9 10. Can KNN be used for regression?

Yes, KNN can be used for regression tasks. In this case, the predicted value is the average (or weighted average) of the target values of the k-nearest neighbors.

11. Is KNN sensitive to outliers?

Yes, KNN is sensitive to outliers, as outliers can affect the nearest neighbor calculation and the majority vote. Using robust distance metrics or preprocessing to remove outliers can help.

6.10 12. How do you deal with categorical data in KNN?

To handle categorical data:

- Convert categories to numerical values using encoding techniques (e.g., one-hot encoding).
- Use Hamming distance or other categorical distance measures.