The K-Nearest Neighbors Algorithm

Latreche Sara

June 9, 2025

Contents

1 The k-Nearest Neighbors (kNN) Algorithm

Definition

The **k-Nearest Neighbors** (kNN) algorithm is a non-parametric supervised learning method used for classification and regression. It is based on the principle that similar data points tend to be near each other in feature space.

1.1 Problem Setup: Classification

Consider a training dataset

$$D = \{(x^{(i)}, y^{(i)})\}_{i=1}^{n},$$

where each $x^{(i)}$ is a feature vector in a domain X and $y^{(i)}$ is the corresponding label or output.

The goal is to predict the label y for a new input $x \in X$.

Note

kNN performs no explicit training: it simply memorizes the dataset. The computation happens at prediction time by comparing distances between points.

1.2 Distance Metric

A distance metric $d : X \times X \to \mathbb{R}^+$ is required to quantify similarity between points. It must satisfy, for all $x, x', x'' \in X$:

$$\begin{cases} d(x,x) = 0, \\ d(x,x') = d(x',x), \\ d(x,x'') \le d(x,x') + d(x',x''). \end{cases}$$

A common choice is the Euclidean distance in $\mathbb{R}^d$:

$$d(x,x') = \sqrt{\sum_{j=1}^{d} (x_j - x'_j)^2}.$$

1.3 The kNN Algorithm

Given a query point x, the algorithm finds the k points in the training set closest to x according to the distance d. The prediction function $h(x)$ depends on the task:

- **Classification**: output the majority class among the k nearest neighbors.
- **Regression**: output the average value of the target variables of the k nearest neighbors.

Algorithm 1 kNN Algorithm for Classification

```
Require: Training data $D = \{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$, query point $x$, number of neighbors $k$
Ensure: Predicted class $h(x)$
1: Compute distances $d(x, x^{(i)})$ for all $i = 1, \dots, n$
2: Find indices $N_k(x)$ of the $k$ nearest neighbors
3: Retrieve labels $\{y^{(i)} : i \in N_k(x)\}$
4: $h(x) \leftarrow$ the majority class among these labels
5: return $h(x)$
```

Algorithm 2 kNN Algorithm for Regression

```
Require: Training data $D = \{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$, query point $x$, number of neighbors $k$
Ensure: Predicted value $h(x)$
1: Compute distances $d(x, x^{(i)})$ for all $i = 1, \dots, n$
2: Find indices $N_k(x)$ of the $k$ nearest neighbors
3: Retrieve target values $\{y^{(i)} : i \in N_k(x)\}$
4: $h(x) \leftarrow \frac{1}{k} \sum_{i \in N_k(x)} y^{(i)}$
5: return $h(x)$
```

1.4 Numerical Example: Classification

Example

Consider the following training data in $\mathbb{R}^2$ with two classes $+1$ and -1:

$$\{(0,0),+1\}, \quad \{(1,0),+1\}, \quad \{(0,1),-1\}, \quad \{(1,1),-1\}.$$

Predict the class of the point $x=(0.9,0.2)$ with $k=3$.
Solution:

- Compute Euclidean distances:

$$d(x,(0,0)) = \sqrt{0.9^2+0.2^2} \approx 0.922,$$

$$d(x,(1,0)) = \sqrt{(0.9-1)^2+(0.2-0)^2} = \sqrt{0.01+0.04} = \sqrt{0.05} \approx 0.224,$$
$$d(x,(0,1)) = \sqrt{0.9^2+(0.2-1)^2} = \sqrt{0.81+0.64} = \sqrt{1.45} \approx 1.204,$$
$$d(x,(1,1)) = \sqrt{(0.9-1)^2+(0.2-1)^2} = \sqrt{0.01+0.64} = \sqrt{0.65} \approx 0.806.$$

- The three nearest neighbors are:

$$(1,0) \quad (+1), \quad (1,1) \quad (-1), \quad (0,0) \quad (+1).$$

- Classes of neighbors: $+1,-1,+1$. Majority class is $+1$.

Hence, $h(x)=+1$.

1.5 Numerical Example: Regression

Example

Consider the training data with real-valued targets:

$$\{(1), 2.0\}, \quad \{(2), 2.5\}, \quad \{(3), 3.5\}, \quad \{(4), 4.0\}.$$

Predict the output at $x = 2.7$ with $k = 2$ using Euclidean distance on the scalar input.
Solution:

- Distances:
$$\begin{aligned} d(2.7, 1) &= |2.7 - 1| = 1.7, \\ d(2.7, 2) &= 0.7, \\ d(2.7, 3) &= 0.3, \\ d(2.7, 4) &= 1.3. \end{aligned}$$
- The two closest points are 3 (3.5) and 2 (2.5).
- Average output:
$$h(2.7) = \frac{3.5 + 2.5}{2} = 3.0.$$

1.6 Choice of Parameters and Algorithm Limitations

1.6.1 Choice of Distance Metric

Imagine you have a dataset of fruits, each described by features such as weight, sweetness, and color intensity (numerical values). We want to classify a new fruit based on these features.

Fruit	Weight (g)	Sweetness (1–10)	Color Intensity (1–100)
Apple	150	7	60
Orange	130	6	80
Pear	170	5	40

Table 1: Example fruit dataset

A new fruit is described as:

$$\text{new\_fruit} = \{\text{weight} : 160, \text{sweetness} : 6, \text{color\_intensity} : 70\}$$

We want to calculate the distance between the new fruit and the Apple to see how similar they are.

Distance Calculations **Euclidean distance** (straight-line distance in feature space):

$$\begin{aligned} d &= \sqrt{(w_{\text{new}} - w_{\text{apple}})^2 + (s_{\text{new}} - s_{\text{apple}})^2 + (c_{\text{new}} - c_{\text{apple}})^2} \\ &= \sqrt{(160-150)^2 + (6-7)^2 + (70-60)^2} \\ &= \sqrt{100+1+100} \approx 14.18 \end{aligned}$$

Manhattan distance (distance if you can only move along each feature axis):

$$d = |160-150| + |6-7| + |70-60| = 10 + 1 + 10 = 21$$

Why Does the Distance Metric Matter? The choice of distance metric affects which neighbors are considered "closest." For example, if features have very different scales (weight in grams vs. sweetness in 1–10), then large differences in weight may overshadow small differences in sweetness.

Feature Scaling (Normalization) To avoid features with larger scales dominating the distance calculation, normalize features between 0 and 1 using min-max scaling:

$$x' = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}$$

For example, if weight ranges from 100g to 200g, the normalized weight for the new fruit is:

$$\frac{160-100}{200-100} = 0.6$$

Normalize all features before calculating distances to ensure no single feature dominates.

1.6.2 Choosing the Parameter k

The parameter k defines how many neighbors the algorithm uses to decide the class of a new point.

- If $k = 1$, classification depends only on the closest neighbor, which can make the model sensitive to noise or outliers.
- If k is too large, the model may include neighbors far away, possibly from other classes, reducing accuracy.

Typically, k is chosen as an odd number between 3 and 10 to avoid ties and balance the bias-variance tradeoff.

Example Suppose we have 5 neighbors around the new fruit with these labels and distances (Euclidean):

Neighbor	Distance	Label
1	12.5	Apple
2	14.2	Pear
3	15.0	Apple
4	18.0	Orange
5	20.0	Apple

For $k = 3$, the nearest 3 neighbors are Apple (12.5), Pear (14.2), and Apple (15.0). Majority is Apple $\rightarrow$ classify new fruit as Apple.

For $k = 5$, neighbors include 3 Apples, 1 Pear, and 1 Orange $\rightarrow$ majority still Apple.

1.7 Limitations of kNN

Note

- Computationally expensive at prediction time for large datasets because distances must be computed to all training points.
- Curse of dimensionality: in high-dimensional spaces, distance metrics can become less meaningful.
- Sensitive to irrelevant or noisy features.

1.8 KNN in Recommendation Systems

K-Nearest Neighbors (KNN) is a simple yet powerful algorithm that is widely used in recommendation systems. It works by finding the most similar users or items based on their features or interactions and using these neighbors to predict preferences.

There are two common types of recommendation systems using KNN:

- **User-based collaborative filtering**: Finds users similar to the target user based on their preferences or behavior, then recommends items liked by those similar users.
- **Item-based collaborative filtering**: Finds items similar to the target item based on user ratings or attributes, then recommends items similar to those the user already likes.

KNN is popular in recommendation because:

- It is intuitive and easy to implement.

- It does not require a model training phase, making it adaptable to new data quickly.
- It can incorporate different similarity metrics (e.g., cosine similarity, Pearson correlation) depending on the context.

However, KNN also has limitations in recommendation systems:

- **Scalability**: Computing similarities for large datasets can be expensive.
- **Sparsity**: User-item matrices are often sparse, making it difficult to find good neighbors.
- **Cold start**: New users or items with little data cause poor recommendations.

Despite these challenges, KNN remains a foundational technique for many recommendation algorithms and serves as a baseline for more complex methods.

References

- MIT Open Learning Library. *Introduction to Machine Learning.* Available at: https://openlearninglibrary.mit.edu/courses/course-v1:MITx+6.036+1T2019/course/
- Junier, F. (n.d.). *K-Nearest Neighbors (KNN) Course Notes.* Available at: https://frederic-junier.org/NSI/premiere/chapitre22/knn-cours-.pdf