1 K-Nearest Neighbors (KNN) Algorithm

1.1 Algorithm

Algorithm 1 K-Nearest Neighbors (KNN) Algorithm

1: **procedure** KNN($X_{train}, y_{train}, x_{query}, k$)
2: $n \leftarrow |X_{train}|$
3: Initialize distances $D = []$
4: **for** $i = 1$ to n **do**
5: $d_i \leftarrow$ distance($X_{train}[i], x_{query}$)
6: Append $(d_i, y_{train}[i])$ to D
7: **end for**
8: Sort D based on distances
9: $N_k \leftarrow$ first k elements of D
10: **if** classification task **then**
11: **return** most common class in N_k
12: **else if** regression task **then**
13: **return** average of target values in N_k
14: **end if**
15: **end procedure**

1.2 Detailed Explanation

1. **Input**:
    - X_{train}: Training data features
    - y_{train}: Training data labels or target values
    - x_{query}: Query point for which we want to make a prediction
    - k: Number of nearest neighbors to consider
2. **Procedure**:
    (a) **Initialize** (lines 2-3):
        - n is set to the number of training samples.
        - An empty list D is created to store distances.

    (b) **Calculate distances** (lines 4-7):
        - For each training sample, calculate its distance to the query point.
        - The distance function can be Euclidean, Manhattan, or any other metric.
        - Store each distance along with its corresponding label.

    (c) **Sort distances** (line 8):

- Sort the list D based on the calculated distances.

(d) **Select k nearest neighbors** (line 9):
- Take the first k elements from the sorted list D.
- These represent the k nearest neighbors to the query point.

(e) **Make prediction** (lines 10-14):
- For classification:
  - Return the most frequent class among the k nearest neighbors.
- For regression:
  - Return the average of the target values of the k nearest neighbors.

3. **Distance Calculation**: The most common distance metric is Euclidean distance:

$$\text{distance}(x_1, x_2) = \sqrt{\sum_{i=1}^{d} (x_{1i} - x_{2i})^2}$$

where d is the number of features.

4. **Choosing k**:
   - A small k can lead to overfitting (high variance).
   - A large k can lead to underfitting (high bias).
   - k is often chosen using cross-validation.
   - An odd k is preferred for binary classification to avoid ties.

5. **Complexity**:
   - Time complexity: $\mathrm{O}(n \log n)$ due to sorting.
   - Space complexity: $\mathrm{O}(n)$ to store distances.

6. **Advantages**:
   - Simple to understand and implement.
   - No assumptions about data distribution.
   - Can be used for both classification and regression.

7. **Disadvantages**:
   - Computationally expensive for large datasets.
   - Sensitive to irrelevant features and the scale of the data.
   - Requires feature scaling for best results.

8. **Variants**:

- Weighted KNN: Closer neighbors have more influence on the prediction.
- KD-Trees or Ball Trees: Data structures to speed up nearest neighbor search.