4. Nearest Neighbor Estimation

4.1 Introduction

The nearest neighbor estimator is a supervised, nonparametric learning method that classifies or predicts the target value of a data point using similar observations. This technique has been used for decades in both classification and regression problems. Unlike linear and logistic regression, it does not approximate any function for the target variable; instead, it simply stores the training examples. No computation or generalization is performed until a new instance requires a decision. Since the nearest neighbor estimator relies solely on training instances for estimations, it is also known as instance-based learning. It is sometimes called lazy learning because it delays processing until a new instance needs to be classified. The advantage of instance-based learning is that it estimates data points locally rather than approximating the target function across the entire input space.
In k-Nearest Neighbors (kNN), learning involves merely storing the training data. When a new instance appears, the algorithm retrieves a set of k similar instances from memory to classify the new query instance. The target variable for the new instance is determined based on the target variables of these k-nearest neighbors. A major drawback of this method is its execution time, as all computations occur during classification or prediction since there is no explicit training phase.

4.2 Euclidean Distance

The nearest neighbors of an instance x are the closest training points x_j determined by distance metrics. Commonly used distance metrics include Euclidean distance, Manhattan distance, and Minkowski distance.
Euclidean Distance:
Euclidean distance is the cartesian distance between any two points of d-dimension defined as

$$d(x,y) = \sqrt{\sum_{i=1}^{d}(x_i - y_i)^2} \tag{4.1}$$

Manhattan Distance:
Manhattan distance is calculated by summing the absolute difference between points.

$$d(x,y) = \sum_{i=1}^{d} |(x_i - y_i)| \tag{4.2}$$

Minkowski Distance:
Minkowski distance is the cartesian distance between any two points defined as

$$d(x,y) = (\sum_{i=1}^{d} (x_i - y_i)^p)^{1/p}) \tag{4.3}$$

4.3 k-Nearest Neighbor for Regression

First, let's consider the regression problem, where the target variable is continuous and the algorithm needs to learn the mapping $f : R^d \to \mathbb{R}$. Suppose the training set contains N observations, each with d dimensions, and we need to predict the value for a new instance x. We must calculate the distances between x and all other points, and then arrange the examples in ascending order based on these distances. Finally, we calculate the mean of the first k points. k Nearest Neighbour Algorithm:

1. Let $T_r = (x_i, y_i)$, $i = 1, 2, \ldots, N$ bet the training set with data points x_i and the price y_i . Let x be the new observation whose y value has to be predicted.
2. Calculate Euclidean distance of x with the data points in training set. Arrange the data points in ascending order.
   $(x_1, x_2, x_3, \ldots, x_N) : d(x, x_{1)} \leq d(x, x_2) \leq d(x, x_3) \leq \ldots d(x, x_N)$
3. Select first k members from the above set as the k-nearest neighbors $N(x) = x_1, x_2, \ldots, x_k$ of x The y value corresponding to x can be estimated as $\hat{y} = \frac{1}{k}\Sigma_{x_i \in N(x)} y_i$

4.4 k-Nearest Neighbor for Classification

In a k-nearest neighbors (kNN) classification problem, the target function is discrete. It aims to learn a discrete-valued target function $f : R^d \to C$, where C is the finite set of m values. The kNN algorithm assigns the most common value of the target variable among the k training instances nearest to x. For $k = 1$, it assigns the class label of the instance closest to x. In the figure, we classify the test instance x using the kNN algorithm with k set to 6. Among the 6 nearest neighbors of x, the majority (4 instances) belong to the blue class. Therefore, the kNN estimator assigns the blue class to the test instance kNN algorithm classifies the new instance as follows:

$$p(y|D, x, k) = 1/k \sum_{i \in N_k(x,D)} \mathbb{I}(y_i = c) \tag{4.4}$$

Where $N_k(x, D)$ denotes k nearest neighbors of the point x in the dataset D. $\mathbb{I}$ is the indicator variable.

$$\mathbb{I}(e) = \begin{cases} 1 & if\ e\ is\ true \\ 0 & if\ e\ is\ false \end{cases} \tag{4.5}$$

The instance-based approach incurs high computation costs. Each time a new instance needs a decision, the algorithm calculates the distance between this point and all other points, then sorts the training samples in ascending order based on these distances. Distance calculation requires $O(N)$ operations, and sorting requires at least $O(NlogN)$ operations. This makes kNN impractical for large datasets.
kNN requires scaling all variables to ensure equal weight for all features. If different variables have varying importance in determining the target variable, weighted kNN can be used to assign appropriate weights to each feature.

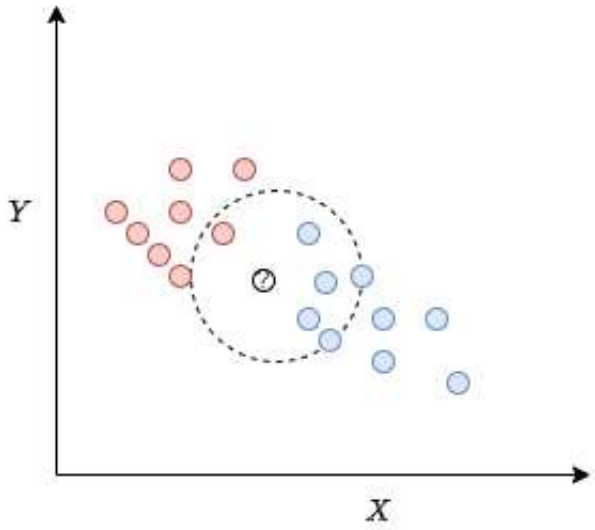

Figure 4.1: k-Nearest Neighbors