26

K NEAREST NEIGHBORS

26.1 Principles

26.1.1 The algorithm

The core idea of the K-nearest neighbors (KNN) algorithm is to construct the prediction $\hat{y} = f(x)$ of input x based on the average output of its neighborhood in the training examples D. Particularly, denoting the K nearest neighbors by $N_k(x)$, for regression problem, we have

$$\hat{y} = \frac{1}{|N_k(x)|} \sum_{x_i \in N_k(x)} y_i;$$

for classification problem, we have

$$\hat{y} = \arg\max_c \sum_{x_i \in N_k(x)} I(y_i = c_i).$$

where we determine the class label of x based on the majority vote of the K neighbors.

The KNN algorithm is summarized in algorithm 37.

Algorithm 37: KNN classification and regression algorithm

```
Input: Training data set $(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$, a distance metric $d$.
       Parameter $K$. Test data $x$.
1 Find the K nearest neighbors of $x$ for the training data set using the provided
    metric $d$. Denote the neighbors by $N_k(x)$.
2 For classification problem, determine the class of $x$ based on majority vote of the
    $K$ neighbors:
        $\hat{y} = \arg\max_c \sum_{x_i \in N_k(x)} I(y_i = c_i).$
3 For regression problem, determine the predicted value based on the mean(or
    median) value of the $K$ neighbors:
        $\hat{y} = \frac{1}{|N_k(x)|} \sum_{x_i \in N_k(x)} y_i.$
Output: the predicted value $\hat{y}$.
```

The optimal hyperparameter of K can be determined by using cross-validation. For large K, the decision boundary would include points from other classes into the neighborhood; for small K, the decision boundary is less smooth and the testing result can be very sensitive to noise.

Unlike most of machine learning algorithms that have model parameters to be estimated, KNN is a memory-based approach where training examples are the model parameters and the computational cost at the training stage is zero. At the testing stage, the computational complexity for classifying new samples grows linearly with the number of samples in the training dataset, i.e, $O(kN)$, N is the number of the training samples, if we search the nearest neighbor in a brute force way. KNN can be a good alternative when we are facing high dimensional data but number of samples is insufficient to train a complex model such as neural networks.

If there are multiple testing cases and training sample size is larger, computational complexity can be optimized by using efficient spatial searching algorithms, such as KD-Tree and Ball-Tree.

In general, KNN tends to overfit, and the classification rule is less transparent compared to other methods like logistic regression and decision trees. For high-dimensional features, we can use dimensional reduction methods to reduce feature dimensionality that helps reduce computational time and the tendency to overfit.

26.1.2 Metrics and features

Although KNN is one of most simple machine learning algorithms, it has found applications in a wealth of domains, including image, audio, speech, etc. The key to its broad application lies in choosing suitable metrics and features that enable desired measure of distances specific to applications.

Besides Euclidean distance, we also have other commonly used metrics in $\mathbb{R}^n$.

Definition 26.1.1 (metrics for vector space $\mathbb{R}^n$).

- *Minkowski distance*

$$L_p\left(x_i, x_j\right) = \left(\sum_{l=1}^{n}\left|x_i^{(l)} - x_j^{(l)}\right|^p\right)^{\frac{1}{p}}$$

- *Euclidean distance*

$$L_2\left(x_i, x_j\right) = \left(\sum_{l=1}^{n}\left|x_i^{(l)} - x_j^{(l)}\right|^p\right)^{\frac{1}{2}}$$

- *Manhattan distance*

$$L_1\left(x_i, x_j\right) = \sum_{l=1}^{n}|x_i^{(l)} - x_j^{(l)}|)$$

- *Mahalanobis distance*

$$L_M\left(x_i, x_j\right) = \sqrt{((x_i - x_j)^T V^{-1}(x_i - x_j))},$$

where V is a symmetric positive definite matrix.

We also have following metrics designed for binary features. Note that for input data that contains categorical variables, we can convert categorical feature into binary feature.

Definition 26.1.2 (metrics for binary vector spaces). *[1]*

- *The Jaccard distance, also known as Intersection over Union*

$$J(A, B) = \frac{A \cap B}{A \cup B}.$$

$$J(x, y) = \frac{NNEQ}{NNZ},$$

where $NNEQ$ is number of non-equal dimensions, and NNZ is number of nonzero dimensions.

- *Matching distance is defined as $NNEQ/N$*

To apply KNN to text, image and audio, we need to engineer features that lie in the Euclidean space. For example, for text classification, documents will be converted to TF-IDF representation. For image classification, features can be extracted from convolutional networks which will be covered in following chapters.

26.2 Application examples

We consider a binary classification problem based on two-dimensional inputs [Figure 26.2.1]. As we increase the parameter K in the KNN algorithm, we observe smoother decision boundary. We have following summary.

Remark 26.2.1 (tips on the performance of KNN).

- KNN performs much better if all of the data have similar scales (i.e., choose a suitable metric).
- KNN makes no assumptions about the functional form of the problem being solved.
- Increase the parameter K will increase bias; decrease the parameter K will increase the variance will increase.In case of very large value of K, we may include points from other classes into the neighborhood. In case of too small value of K the algorithm is very sensitive to noise. The decision boundary becomes smoother with increasing value of K.
- KNN is a memory-based approach and it does not requires training.
- The computational complexity for classifying new samples grows linearly with the number of samples in the training dataset in the worst-case scenario. To speed up the process, we can use advanced data structures and algorithms that help quickly identify neighbors.

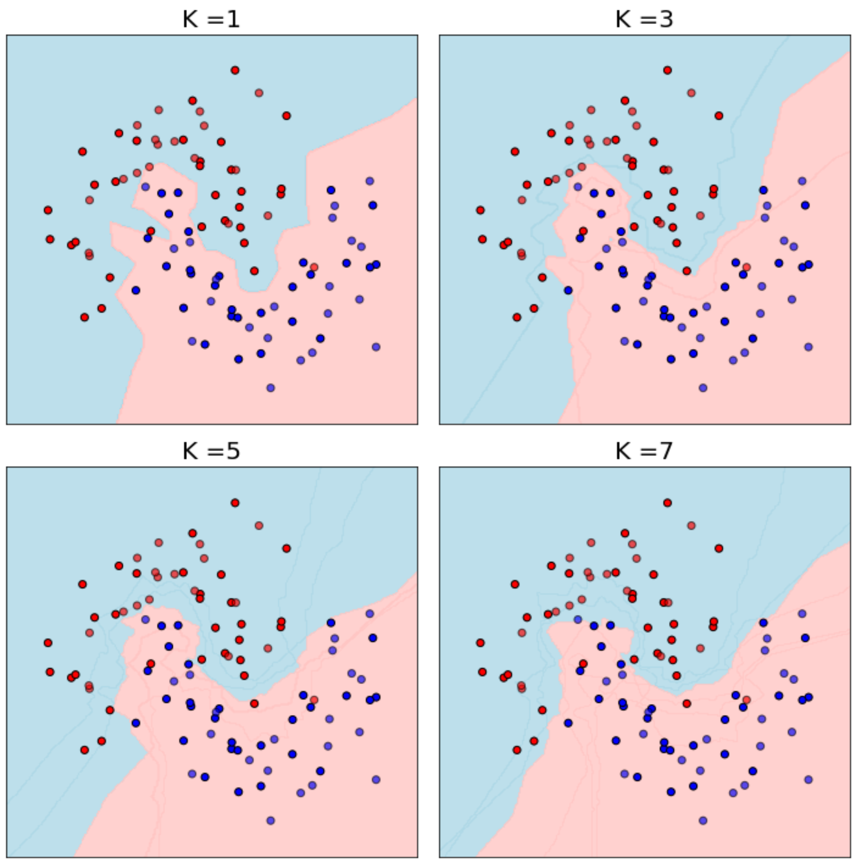

Figure 26.2.1: Binary classification via KNN algorithm with different choices $K = 1, 3, 5, 7$. Scattered points are training examples classified into two different classes (red and blue). Colored regions are corresponding decision boundaries.

BIBLIOGRAPHY

1. Choi, S.-S., Cha, S.-H. & Tappert, C. C. A survey of binary similarity and distance measures. *Journal of Systemics, Cybernetics and Informatics* **8,** 43–48 (2010).