September 24, 2017 Lecture 3, Sep. 13, 2017

KNN for Regression

Lecturer: Justin Domke *Scribe: Boya Ren*

1 Summary

Last time we talked about KNN for regression and today we will dive into more details and discuss its pros and cons.

2 Prediction Formula for KNN

Suppose each training sample can be represented as (x_i, y_i), where x_i is a vector of p dimensions and y_i is a single value. Then for a new input x, its output $\hat{Y}$ can be predicted as

$$\hat{Y} = \frac{1}{K} \sum_{x_i \in N_K(x)} y_i$$

Here $N_K(x)$ indicates the K nearest neighbors of x. So the output of an input vector is the averaged output of its nearest neighbors.

3 Examples

In this part we will go into more details to see how exactly KNN for regression works.

3.1

Suppose we have a dataset with $p = 1$ and the data points are $x_1 = 1$, $x_2 = 9$, $x_3 = 10$, $x_4 = 2$, $x_5 = 2.5$. Then neighbors of different input can be listed as below.

- $N_4(3) = \{x_1, x_2, x_4, x_5\}$
- $N_1(8) = \{x_2\}$
- $N_2(8) = \{x_2, x_3\}$

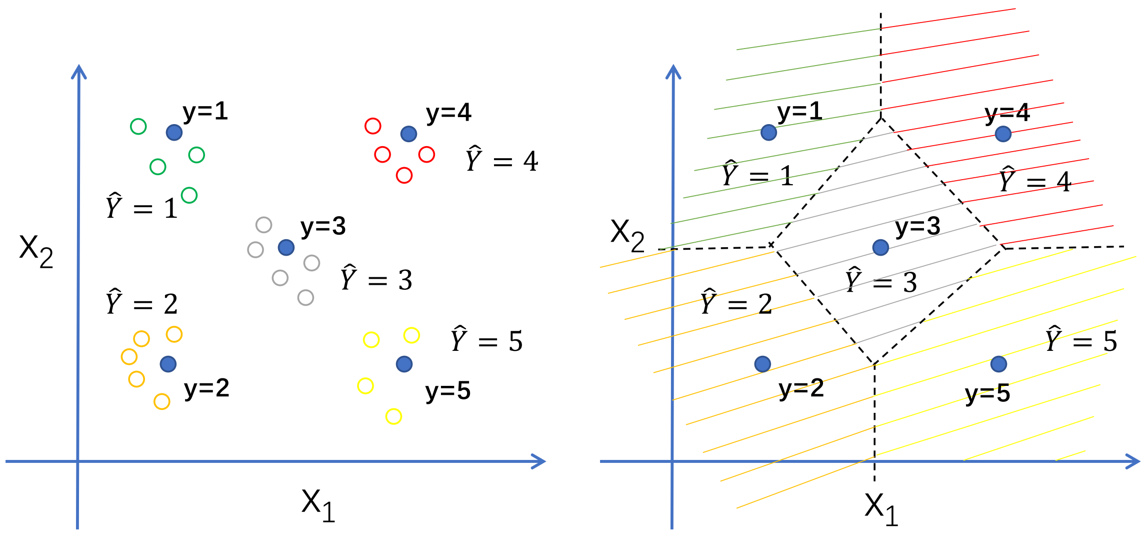


Figure 3.1: Two Dimensional Decision Boundaries

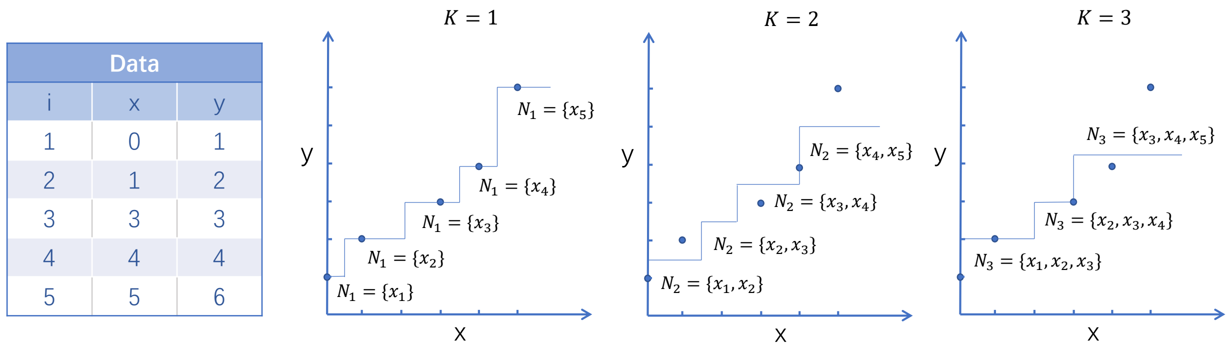
3.2

Suppose we have $p - 2$, $K = 1$ and $N = 5$ data points as shown in the graph 3.1 (the blue solid points).

As shown in the subfigure on the left, the hollow points will have the same predicted output as its closest training sample. In the case of $K = 1$, we only have the discrete output $\hat{Y} = 1,\ 2,\ 3,\ 4,\ 5$, so the whole input plane can be divided into five regions as shown in the right subfigure. The dashed lines are called **Decision Boundaries**, on which there is a tie between different ouputs, i.e. equal distances to training points.

3.3

Suppose we have $N = 5$ data points with input feature dimension $p = 1$, as listed in the table below.



Data		
i	x	y
1	0	1
2	1	2
3	3	3
4	4	4
5	5	6

Figure 3.2: Prediction Curves

The figures show how the output of predicted $\hat{Y}$ changes with input x, when $K = 1,\ 2,\ 3$.

Every jump in the lines indicate that another training data point beats the existing one in the KNN set N_K. So the output changes correspondingly when its neighbors change.

4 Discussions

4.1 Why KNN Does not Work Well in High Dimensions?

We will briefly explain why KNN does not generalize very well with high dimensional data. First, we generate 1000 data points of p dimensions, where on each dimension, the value is uniformly distributed within $[0, 1]$ and all the dimensions are independent. Then we compute the distances between each point-pair and see how the distances are distributed. Fig. 4.1 shows the distribution of distances with different values of p.

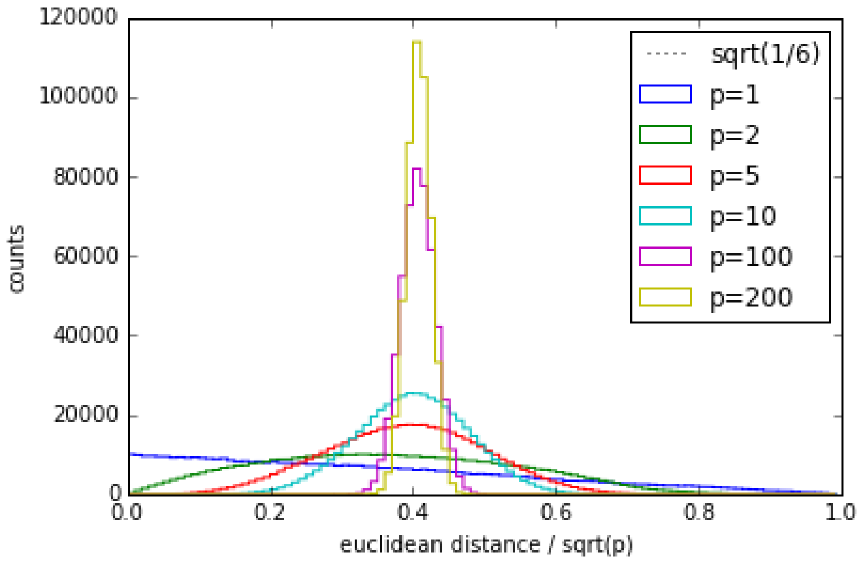


Figure 4.1: Distance Distribution

It can be seen that as p grows, the distances between different points become more and more concentrated, which means that it is more and more difficult to cluster the points by mere distance. Although real data does not have independent and uniformly distributed dimensions, but the probability that different data points have similar distances is definitely higher.

4.2 What is a Good Choice of K?

As we noticed, different choices of K has fairly large influence to the performance of KNN. Then how can we choose a reasonable K. Before coming to this question, let's recall the definition of Mean Square Error (MSE) and Mean Absolute Error (MAE).

$$MSE = \frac{1}{N}\sum_{i=1}^{N}[y_i - \hat{Y}(x_i)]^2$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{Y}(x_i)|$$

The basic idea to choose K is try to minimize MSE or MAE. Sometimes the result with MSE and MAE can be different, depending on how largely you want to penalize big errors. If we measure MSE on the training set, it will only grow up as K gets larger (shown in the left subfigure of 4.2), since the training data will be predicted perfectly with $K = 1$ and worse and worse when K grows. So it does not give us any useful information to test MSE on the training set. A reasonable way to do this is to leave out a validation set that is not using in training and measure MSE on it. It is seen below that in the ideal case, as K increases, MSE first goes down then up, and we can find the best K when the curve is at its lowest point.

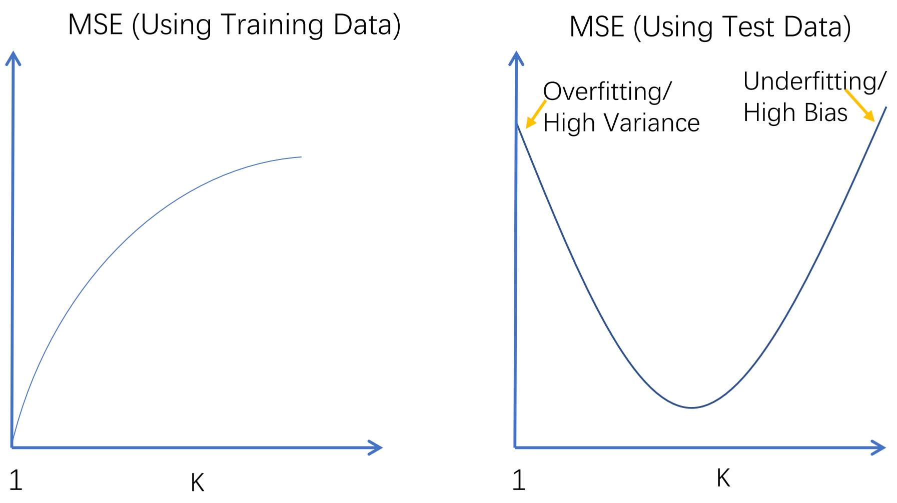


Figure 4.2: MSE with K

4.3 Time Complexity

The KNN algorithm can be briefly summarized as below.

Algorithm 1 KNN for Regression

```
Input: $x$
1: for $i = 1, 2, ..., N$ do
2:     $d_i = \|x - x_i\|_2$
3: Find $K$ smallest $d_i$ and average
```

For learning complexity, it can be either 0 or Np, depending on how you prepare the data. The test complexity is $O(Np)$ per query point, which can be fairly high for large number of queries. Imaging you have 50G of data stores on the server, and every second you receive 1000 hits on your web. Computation of KNN in this case can be tremendous.

5 Summary

The pros of KNN:

- Simple
- Tends to work well in low dimensions
- No “learning algorithm”
- Test complexity is low if test dataset is small

The cons of KNN:

- In general does not scale to high dimenstions
- Test complexity is high if test dataset is big