KNN

Baboo J. Cui

June 20, 2019

Contents

1 Introduction

KNN is short for K-nearest-neighbor:

- supervised learning
- very easy
- classification is based on measuring distance between characteristics

Its advantages:

- accurate(in terms of algorithm not result)
- insensitive to extreme values
- no training process

Its disadvantages:

- computational costly(traverse the whole data set)
- time consuming
- high requirement for storage
- data must be in the same dimension

2 Ideology

Given a data set with label on each element, when a new sample arrives, find **top** k elements in set that are **closest** to it. The label for new sample is simply set to be the mass label of the k elements.

- k usually is less than 20
- k should be odd number

3 Algorithm

KNN algorithm is as follow:

1. calculate **distance**(usually $\mathcal{L}_2$, may be others) between coming sample and all elements in data set
2. sorting data set by distance from closest to farthest
3. pick up the first k elements in data set
4. get the label frequency in k elements
5. the prediction is the label with highest frequency

Here is a list of tricky situation:

- when label is not number, assign each label with a number for convenience
- when dynamic range of each characters differs too much, normalization is necessary, it can be done by:

$$\text{newVal} = \frac{\text{oldVal} - \min}{\max - \min}$$

4 Example

Here is a quick example of KNN for finding the type of point:

```
import math
import numpy as np

if __name__ == '__main__':
    data_set = [[0.1, 0.2, 'A'],
                [0.2, 0.1, 'A'],
                [1.1, 1.2, 'B'],
                [1.0, 0.9, 'B']]
    num_char = 2
    k = 3

    sample = [1, 1]

    for element in data_set:
        tmp = 0
        for val in range(num_char):
            tmp = tmp + (element[val]-sample[val])**2
        tmp_distance = math.sqrt(tmp)
        element.append(tmp_distance)

    data_set = np.array(data_set)

    idx = np.lexsort([data_set[:, 3]])
    data_set = data_set[idx, :]

    label = []

    for i in range(k):
        label.append(data_set[i, -2])

    # quiet tricky
    print(max(label, key=lambda x: len(x)))
```

This will print "B" as expected.