Technische Universität Braunschweig

Machine learning

Mini-project: Weighted KNN

Menglin Xi
Carl-Friedrich-Gauß-Fakultät Mathematik

Machine Learning, WiSe 2020, Dr. -Ing. Mehdi Maboudi

Catalog

1. Data Set

2. What is WKNN?

3. Weighted functions

4. Distance conditions

5. An Example

6. Compare WKNN vs KNN

7. Code and result

About my Dataset: Iris and Avocado Price

Which library ?

```
import numpy as np
import random
import matplotlib.pyplot as plt
from collections import Counter
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

Dataset Iris: sklearn link

Dataset Avocado Price: Kaggle link

Dataset	Training Set	Testing Set
Iris	120 Samples	30 Samples
Avocado	14599 samples	3650 Samples

Training Set : Testing Set = 4 : 1

What is weighted KNN ?

In KNN：

when we find the K nearest neighbors of the observation point(x,y) and get the targets of these K points, we take the target with the highest frequency as the result of the new observation(x, y).

This means that the K nearest points we choose have equal influence on the classification results of the new observation(x, y).

In weighted KNN:

We want the K nearest points have different influence with decision, which are particularly close to the new observation(x, y) , should get a higher weight in the decision than such neighbors that are far away from (x, y).

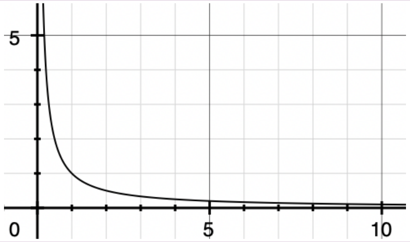
So what is the relationship between weight and distance?

Which function can be used?

Weight Functions

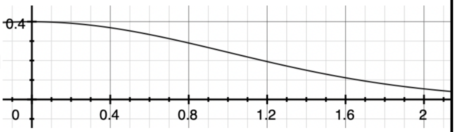
inversion kernel

$$W_{inversion} = \frac{1}{|d|}, |d| \neq 0$$



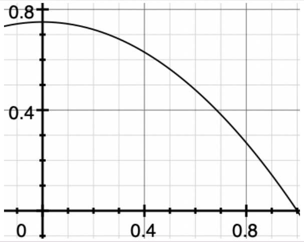
Gauss kernel

$$W_{Gauss} = \frac{1}{\sqrt{2\pi}} exp(-\frac{d^2}{2})$$



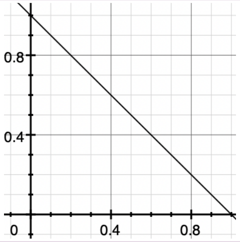
Epanechnikov kernel

$$W_{Epanechnikov} = \frac{3(1-d^2)}{4} * I, |d| \leqslant 1$$



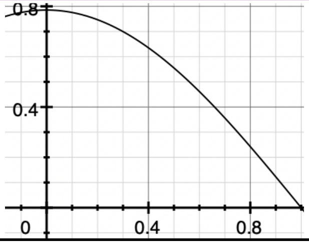
triangular kernel

$$W_{trianguler} = (1 - |d|) * I, |d| \leqslant 1$$



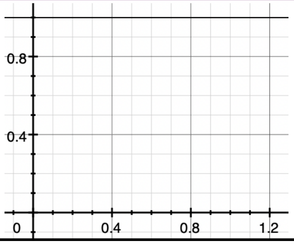
Cosine kernel

$$W_{cosine} = \frac{\pi}{4} \cos(\frac{\pi}{2} d) * I, |d| \leqslant 1$$



mean_one kernel

$$W_{mean-one} = 1$$



The smaller the distance, the greater the weight

Just choose one you like(1-5).

The last one is just like normal KNN.

Distance condition

1. We use Euclidean distance

2. For inversion kernel, the distance cannot be 0

3. For W_triangular, W_Epanechnikov, and W_cosine, the |d| <= 1, how can we make it ?
   We can use the distance of the (k+1)th nearest point as the divisor

$$d = \frac{d_i}{d_{k+1}}, i = 1,2,\ldots,k$$

An example

Classification

KNN	1	1	1	1	1	1
Target	A	B	B	C	A	A
wKNN	0.90	0.88	0.84	0.77	0.46	0.30

sum_knn	3	2	1
Target	A	B	C
sum_wknn	1.66	1.72	0.77

Regression

A,B,C are float price

KNN	WKNN
$y = \frac{3A + 2B + C}{6} = \frac{A}{2} + \frac{B}{3} + \frac{C}{6}$	$y = \frac{1.66A + 1.72B + 0.77C}{1.66 + 1.72 + 0.77} = \frac{2A}{5} + \frac{166B}{415} + \frac{77C}{415}$

Compare KNN and weighted KNN

Advantages of WKNN vs KNN

- Consider the influence of distance on the decision
- Different weight functions can better adapt to different data sets
- Reduce the distraction of relatively distant points
- Expect to achieve better results

Disadvantages of WKNN vs KNN

- Weight functions take more time to calculate
- Very weak immunity to nearby points
- WKNN is inefficient

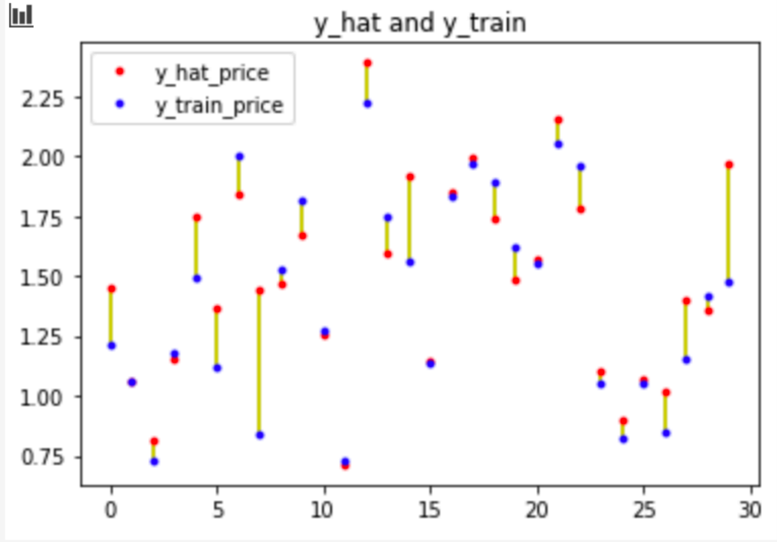
When to use WKNN

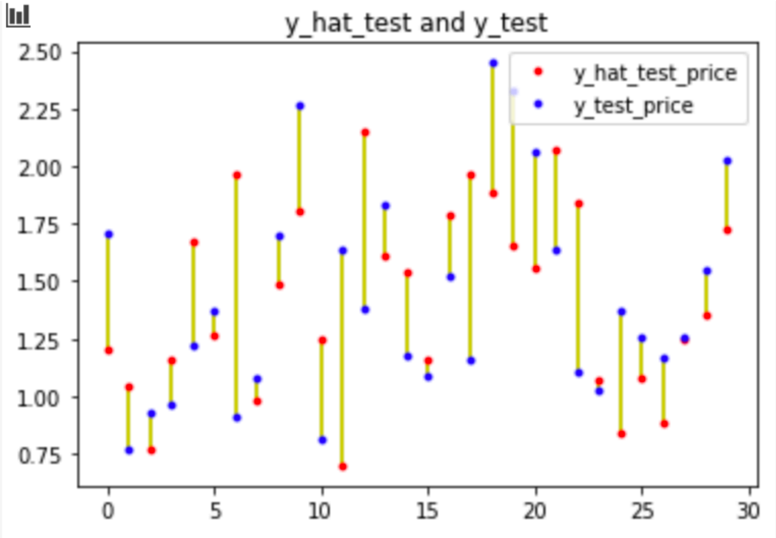
- Data set is small
- You have enough time and interest

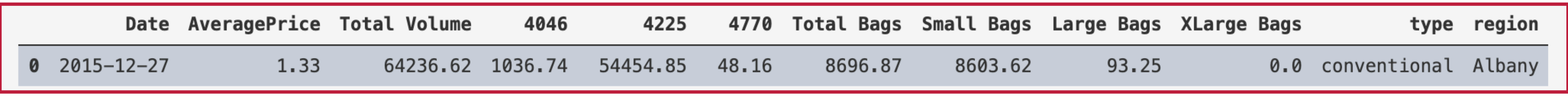
When to not use WKNN

- Data set is large
- Data is very dense or very sparse
- Not useful in Iris Dataset

Show some codes and results: Avocado Price







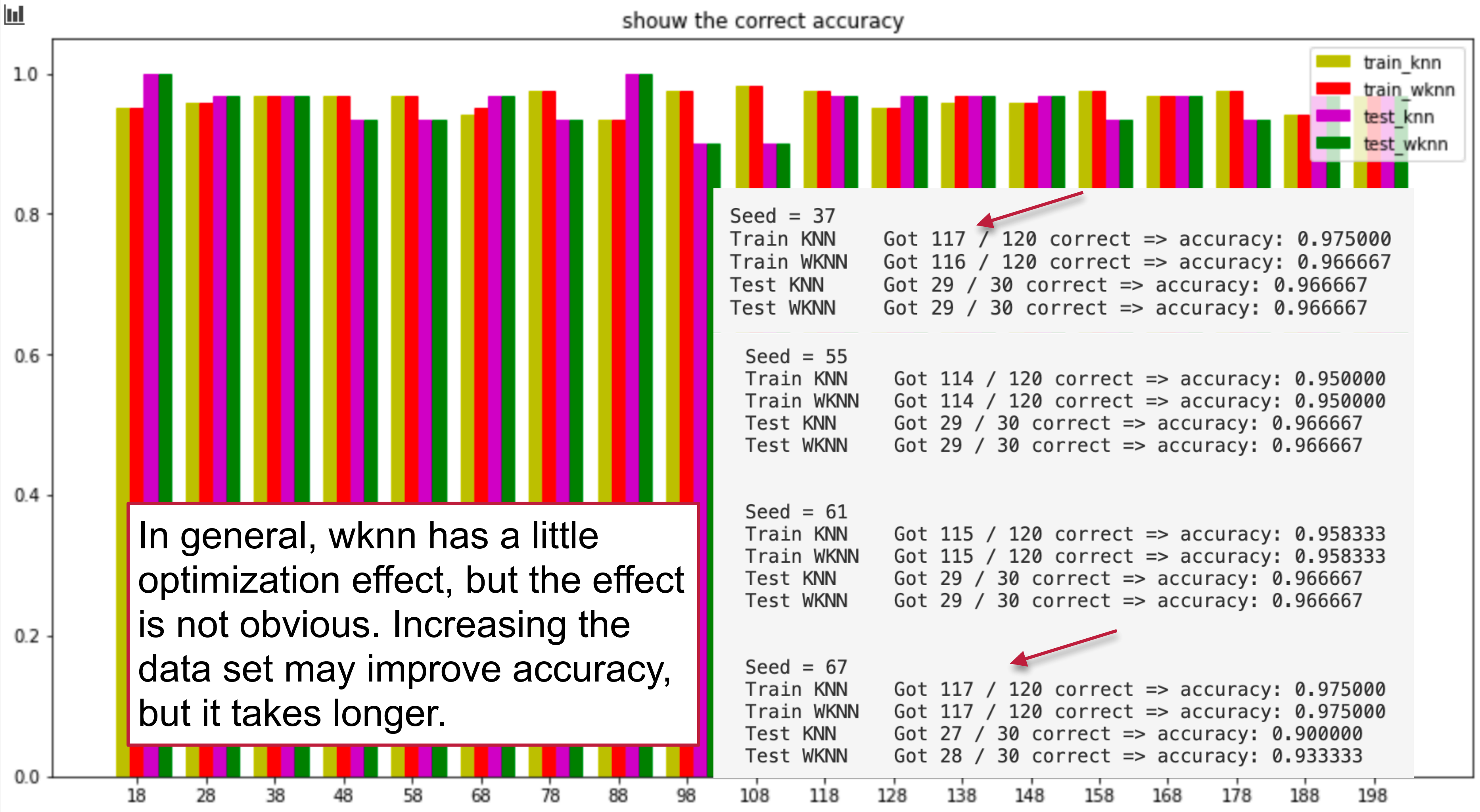
	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags	type	region
0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	conventional	Albany

```
[1036.74 54454.85 48.16 8696.87 8603.62 93.25 0.0 ‘conventional' 'Albany''2015-12-27']
```

```
[1036.74 54454.85 48.16 8696.87 8603.62 93.25 0.0 0 0 51]
```

The effect of regression price looks very bad, and the effect of converting time, region, and type into float values is not good enough.

Show some codes and results: Iris



```
Seed = 37
Train KNN    Got 117 / 120 correct => accuracy: 0.975000
Train WKNN   Got 116 / 120 correct => accuracy: 0.966667
Test KNN     Got 29 / 30 correct => accuracy: 0.966667
Test WKNN    Got 29 / 30 correct => accuracy: 0.966667

Seed = 55
Train KNN    Got 114 / 120 correct => accuracy: 0.950000
Train WKNN   Got 114 / 120 correct => accuracy: 0.950000
Test KNN     Got 29 / 30 correct => accuracy: 0.966667
Test WKNN    Got 29 / 30 correct => accuracy: 0.966667

Seed = 61
Train KNN    Got 115 / 120 correct => accuracy: 0.958333
Train WKNN   Got 115 / 120 correct => accuracy: 0.958333
Test KNN     Got 29 / 30 correct => accuracy: 0.966667
Test WKNN    Got 29 / 30 correct => accuracy: 0.966667

Seed = 67
Train KNN    Got 117 / 120 correct => accuracy: 0.975000
Train WKNN   Got 117 / 120 correct => accuracy: 0.975000
Test KNN     Got 27 / 30 correct => accuracy: 0.900000
Test WKNN    Got 28 / 30 correct => accuracy: 0.933333
```

In general, wknn has a little optimization effect, but the effect is not obvious. Increasing the data set may improve accuracy, but it takes longer.

References

All the pictures and tables are completed by myself.

I am happy to share my code and ppt with you.

Page 2, I give theta sets links, you can check it later.