Machine Learning Exercise Sheet 2

k-Nearest Neighbors and Decision Trees

Group__369

Fan XUE – fan98.xue@tum.de

Xing ZHOU – xing.zhou@tum.de

Jianzhe LIU – jianzhe.liu@tum.de

November 3, 2021

Problem 1

According to the question we have:

A(1, 1) B(2, 0.5) C(1, 2.5) D(3, 3.5) E(5.5, 3.5) F(5.5, 2.5)

a) According to L_1-Norm,

$$d_1(X,Y) = \sum_i |X_i - Y_i|$$

we have for example:

$$d_{AB} = |1-2| + |1-0.5| = 1.5$$

By using this formula we can compute every distance between each two points and fill the results in a table:

L_1-**Norm**	**A**	**B**	**C**	**D**	**E**	**F**	**N-Neighbor**	**Class**
A	0.0	1.5	1.5	4.5	7.0	6.0	B or C	1
B	1.5	0.0	3.0	4.0	6.5	5.5	A	1
C	1.5	3.0	0.0	3.0	5.5	4.5	A	1
D	4.5	4.0	3.0	0.0	2.5	3.5	E	2
E	7.0	6.5	5.5	2.5	0.0	1.0	F	2
F	6.0	5.5	4.5	3.5	1.0	0.0	E	2

We can see that with leave-one-out cross validation, all the points are correctly classified.

b) According to L_2-Norm,

$$d_2(X,Y) = \left(\sum_i (X_i - Y_i)^2\right)^{\frac{1}{2}}$$

we have for example:

$$d_{AB} = ((1-2)+(1-0.5))^{\frac{1}{2}} = 1.118 \approx 1.2$$

Same as before, we still can use this formula to compute every distance between each two points and fill the results in another table:

L_1-**Norm**	**A**	**B**	**C**	**D**	**E**	**F**	**N-Neighbor**	**Class**
A	0.00	1.12	1.50	3.20	5.15	4.74	B	1
B	1.12	0.00	2.24	3.16	4.61	4.03	A	1
C	1.50	2.24	0.00	2.24	4.61	4.50	A	1
D	3.20	3.16	2.24	0.00	2.50	2.69	C	1
E	5.15	4.61	4.61	2.50	0.00	1.00	F	2
F	4.74	4.03	4.50	5.69	1.00	0.00	E	2

We can see that when we use L_2-Norm to validate our data, an error occurs: Point D is classified as Class 1 instead of 2.

c) According to these two cases we can draw the conclusion that a point can be classified into different class if we use different distance measurement. In our case, point D is close to Class 2 in the view of L_1-Norm yet close to Class 1 in the view of L_2-Norm.

Problem 2

a) If we use unweighted K-NN and $K_{new} = N_A + N_B + N_C$, then for X_{new}, the result would definitely be **Class C**. Because K-NN algorithm will always classify the new data into its major neighbor's class. If we use K that concludes every data in this training set, then the majority of the classes are equal to the major neighbors.

b) If we use weighted K-NN, whether the problem would be solved remains uncertain. If some cases it may, but in our case, we are lacking further information about our data's feature, thus not being able to conduct further calculations.

Problem 3

a) After observing this training model we can find that each data in this model has many features, and those features has huge differences on ranges. So if this model performs bad on test set, the reason could be followings:

1. Those features in different ranges have different impact on the result. Solution should be: Using data Standardization:

   $$x_i^{'} = \frac{x_i - \mu_i}{\sigma_i}$$

   or Mahalanobis distance:

   $$\sqrt{(\mathbf{u}-\mathbf{v})^T \Sigma^{-1} (\mathbf{u}-\mathbf{v})}$$

2. Considering so many data, the choice of K can also affect the final result. Solution to this point is simple, we should use various K on validation set and find a better K for this model.
3. Still, when there are so many data, it's possible that a shift occurs between the training set and the test set. For example, in the training set there are only few samples about vans yet in the test set the majority of the samples are vans. Solution to this would be, equally divide the samples into different set to make minimize the shift.
4. When there are too many features in a single sample, it would bring the curse of dimension, which will lead to the lack of samples. As a result, the distance between samples will be amplified, and this could cause bad performance on test set. We can only use another algorithm to try to minimize this problem but we can't solve it so far.

b) If we use decision tree, the above-mentioned problems would be:

1. Features in different ranges will no longer affect the result, because in each node of the decision tree, only one feature will be discussed, that is to say, the other feature will have no impact on this exact node.
2. I would say, the parameter K in K-NN algorithm is just like the way we divide the decision trees. In K-NN, we can use some methods to find a better K, here in DT, we can also find a better way to divide the nodes, by using Misclassification rate, entropy or Gini index.
3. The shift on both sets also exists here and the solution is almost the same.
4. As said above, the curse of dimension will always more or less affect the result.

Problem 4

Can be **proved** by:

$$\begin{aligned}
d_1(x,y)^2 &= \left(\sum_i |X_i - Y_i|\right)^2 \\
&= \sum_i (X_i - Y_i)^2 + 2\sum_i \sum_{j,j\neq i} |X_i - Y_i||X_j - Y_j| \\
&\geq \sum_i (X_i - Y_i)^2 \\
&= d_2(x,y)^2
\end{aligned}$$

Since d_1 and d_2 are all > 0, we can draw the conclusion that:

$$d_1(x,y) \geq d_2(x,y)$$

Problem 5

Can be **disproved** by following example:

Assuming that we have 3 different points: $y(2,\ 2)$ $x_1(1,\ 0.5)$ $x_2(2,\ 0)$

In L_2-Norm we have:

$$d_2(y,x_1) = \sqrt{(2-1)^2 + (2-0.5)^2} = 1.803 \approx 1.8$$

$$d_2(y,x_2) = \sqrt{(2-2)^2 + (2-0)^2} = 2$$

Obviously we have: $d_2(y,x_1) < d_2(y,x_2)$.

Let's say we only have these 3 points, then x_1 is the nearest neighbor of y in L_2-Norm.

But in L_1-Norm we have:

$$d_1(y,x_1) = |2-1| + |2-0.5| = 2.5$$

$$d_1(y,x_2) = |2-2| + |2-0| = 2$$

Obviously we have: $d_1(y,x_1) > d_1(y,x_2)$.

That is to say, in L_1-Norm, x_1 isn't the nearest neighbor of y.

Problem 6

Yes. We can define two new features $y_1 = x_1 - x_2$ and $y_2 = x_2$. By doing this, we can split the dataset by judging the condition $y_1 \leqslant 0$. With only one split the dataset is split into two parts and each part only has one sort of class. It means, there exists a decision tree of depth 1 that classifies this dataset with 100% accuracy.

Problem 7

a)

$$\begin{aligned}
i_H(y) &= -p(y=W)\log p(y=W) - p(y=L)\log p(y=L) \\
&= -\frac{4}{10}\log\frac{4}{10} - \frac{6}{10}\log\frac{6}{10} \\
&= 0.971
\end{aligned}$$

b) Splitting by $x_1 = T$

$$\begin{aligned}
\Delta i_H &= i_H(y) - p(x_1=T)i_H(x_1=T) - p(x_1=I)i_H(x_1=I) \\
&= 0.971 - \frac{1}{2}(-\frac{2}{5}\log\frac{2}{5} - \frac{3}{5}\log\frac{3}{5}) - \frac{1}{2}(-\frac{2}{5}\log\frac{2}{5} - \frac{3}{5}\log\frac{3}{5}) \\
&= 0
\end{aligned}$$

Splitting by $x_2 = M$

$$\begin{aligned}
\Delta i_H &= i_H(y) - p(x_2=M)i_H(x_1=M) - p(x_2=P)i_H(x_2=P) \\
&= 0.971 - \frac{4}{10}(-\frac{2}{4}\log\frac{2}{4} - \frac{2}{4}\log\frac{2}{4}) - \frac{6}{10}(-\frac{2}{6}\log\frac{2}{6} - \frac{4}{6}\log\frac{4}{6}) \\
&= 0.020
\end{aligned}$$

Splitting by $x_3 = S$

$$\begin{aligned}
\Delta i_H &= i_H(y) - p(x_3=S)i_H(x_3=S) - p(x_3=C)i_H(x_3=C) \\
&= 0.971 - \frac{1}{2}(-\frac{3}{5}\log\frac{3}{5} - \frac{2}{5}\log\frac{2}{5}) - \frac{1}{2}(-\frac{1}{5}\log\frac{1}{5} - \frac{4}{5}\log\frac{4}{5}) \\
&= 0.125
\end{aligned}$$

According to the calculation the split judgement will be $x_3 = S$, since in this case, the Δi_H is the biggest. If $x_3 = S$, the instance will be classified as W. Otherwise it will be classified as L.

Problem 8

Let $i^2 = \frac{125}{i}$, we get $i = 5$. Figure 1 shows the 2-d space of the dataset.

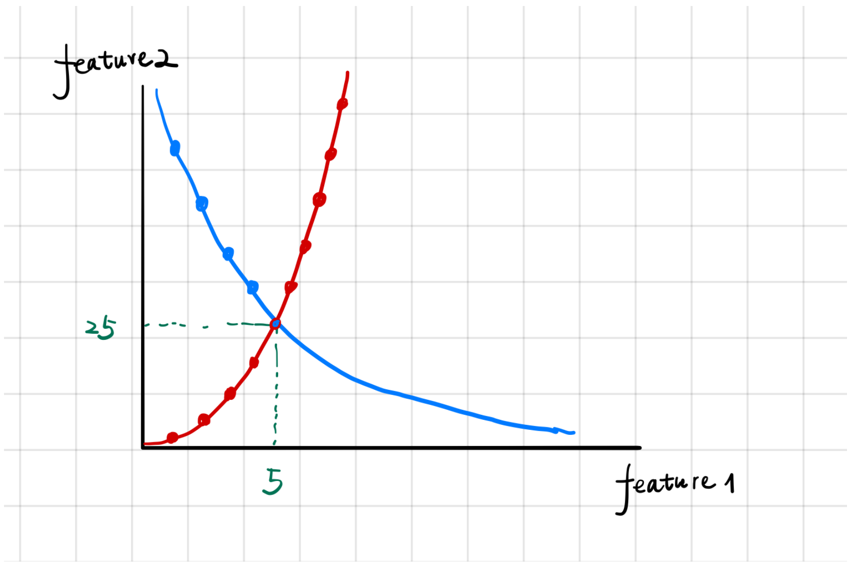

Figure 1: the 2-d space of the dataset

We can easily split the dateset into 4 parts. The first split uses the threshold feature2 $\leqslant$ 25.
The second split uses the threshold feature1 $\leqslant$ 5 for both child nodes.
In this way, the depth of the decision tree is 2. Only one datapoint (5, 25) is missclassified, which is unavoidable.

Problem 9

The result of programming will be shown in next few pages.