ZhiXuan Yang

HW1

DS310

1. (a) Since this problem is asking KNN neighbor interpolation, I don't have to worry about Variance of the given dataset. In other word, we are finding a value that giving perfect output. In this case, 7 nearest neighbor are {7.8, 7, 8, 7, 8, 9} and we are finding argmin $\sqrt{\sum_{i=1}^{N}(X_{ip}-X_{ir})^2}$,

min $\sqrt{(7-X_i)^2+(7-X_i)^2+(7-X_i)+(8-X_i)^2+(8-X_i)^2+(8-X_i)^2+(9-X_i)^2}$, and we find that this will be minimum when X_i = average of the data set.

$X_i = (7+7+7+8+8+8+9)/7 \approx 7.7$

Therefore, the predicted value of the function for the query sample is 7.7

2. (a) the good representation for this data set is convert categorical value into numerical value, and we have to standardization the 'Fever' data to 0-1 scale.

Example	Classification	Fever	Nausea	Diarrhea	Chills
1	Healthy	0	0	0	0
2	Flu	$\frac{1}{2}$	0	0	0
3	Flu	1	0	0	1
4	Salmonella	1	1	1	0
5	Salmonella	$\frac{1}{2}$	0	1	0
6	IBP	0	1	1	0
7	IBP	$\frac{1}{2}$	1	1	0

We assume, no = 0
average = 1
High = 2,
we do the standmarzation, we got
High = (2-0)/(2-0) = 1
Avg = (1-0)/(2-0) = $\frac{1}{2}$
low = (0-0)/(2-0) = 0

(b) For this case, we transformed categorical value into numerical, and we can each example is a vector. Therefore, a suitable distance measure will be Euclidean distance $\sqrt{\sum_{j=1}^{d}(p_j-q_j)^2}$ because We are looking for distances in Vector Space

(c) (High, no, no, no) = (1, 0, 0, 0)

Example 1 = $\sqrt{(0-1)^2+(0-0)^2+(0-0)^2+(0-0)^2} = 1$

Example 2 = $\sqrt{(\frac{1}{2}-1)^2+(0-0)^2+(0-0)^2+(0-0)^2} = \frac{1}{2}$

Example 3 = $\sqrt{(1-1)^2+(0-0)^2+(0-0)^2+(1-0)^2} = 1$

Example 4: $\sqrt{(1-1)^2+(1-0)^2+(1-0)^2+(0-0)^2} = \sqrt{2}$

Example 5: $\sqrt{(\frac{1}{2}-1)^2+(0-0)^2+(1-0)^2+(0-0)^2} = \sqrt{1.25}$

Example 6: $\sqrt{(0-1)^2+(1-0)^2+(1-0)^2+(0-0)^2} = \sqrt{3}$

Example 7: $\sqrt{(\frac{1}{2}-1)^2+(1-0)^2+(1-0)^2+(0-0)^2} = \sqrt{2.25}$

(d) Based on previous question, the 3-nearest-neighbor are ~~{flu, flu, healthy}~~

Flu, Flu, Healthy
$d=\frac{1}{2}$ $d=1$ $d=1$

3. (a) First, we have to training the data sets, then we have m training samples. $O(m)$, and we have d features, thus we have $O(dm)$ for compute all the training set. We also have to find k nearest neighbors in m training samples, therefore we got $O(kmm)$, since the function of KNN classifier is $(\sum_{i=1}^{N}(X_{ip}-X_{ir})^2)$, which means the training samples will run the number of data samples, times $O(p)$, and repeat previous proces. we got $O(dmp)+O(kmp)$, and finally we got $O((d+k)mp)$

(b) The disadvantage of KNN algorithm is that it needs a lot of resource to computing because it needs to compute the distance of every single example in the dataset. As the result, the way of optimizing is reduce the size of computing; and there are two ways to do this. (1) Hash algorithm. (2) K-Dimensional Tree. For Hash, a hash algorithm is a function that converts a data string into a numeric string output of fixed length. The output string is generally much smaller than the original data. Therefore, it reaches our goal that reduce the size of data, so Hash function is a way. For K-Dimensional Tree, this is a binary search tree where data in each node is a K-Dimensional point in space. Generally, it is a space partitioning data structure for organizing points in a K-Dimensional space, and after this, the complexity will be $O(\log m)$. Therefore, it also reached our goal that reduce the size of data required. However, Hash algorithm and K-Dimensional Tree can make KNN become faster.

4. First, we have to know that the meaning of sparse is that reduce the influence of irrelevant features and features with very little weight, and then pick the most important features. Therefore, we have to find a way to drive useless features' weights to 0. Therefore, we have involve L_1 Norm and L_2 Norm.

The function of sum of squares error (SSE) is the difference between the observed value and the predicted value.

$$\sum_{i=1}^{N}(d_i - y_i)^2 = \sum_{i=1}^{N}(e_i)^2$$

SSE function

This function calculate the difference between the observed value and the predicted value, and we have to minize this function. To make this object function is sparse, we can add L_1 Norm and L_2 Norm to this SSE function.

L_1 norm does feature selection. It does this by assigning insignificant input features with zero weight and useful features with a non zero weight.

L_1 norm: $\lambda \sum_{j=1}^{H} |w_j|$, and add this to SEE: $\sum_{i=1}^{N}(d_i - y_i)^2 + \lambda \sum_{j=1}^{H} |w_j|$ → L_1 Norm. W=weight

Here, λ is a constant, and this new SEE function reduce the irrelevant features weight to zero.

L_2 norm forces the weights to be small but does not make them zero and does non sparse solution.

L_2 norm: $\lambda \sum_{j=1}^{H} w_j^2$, and add this to SEE: $\sum_{i=1}^{N}(d_i - y_i)^2 + \lambda \sum_{j=1}^{H} w_j^2$ → L_2 Norm. W=weight

λ is a constant as well, and this pushes the weights close to zero.

Finally, (L_1 Norm) and (L_2 Norm) can modify the object function to be minized to ensure that the learned linear function is sparse.