COMPSCI 762 2022 S1 Week 10 Questions

Luke Chang

May 21, 2022

Question 1 - K-Nearest Neighbour (KNN)

1.1

Let's consider a 2-dimension case, where each instance has two features – x and y, as show in Fig. 1. We have two data point $[0, 0]$ and $[1, 1]$, the Euclidean distance between them is $\sqrt{2}$ and the Manhattan distance is 2.

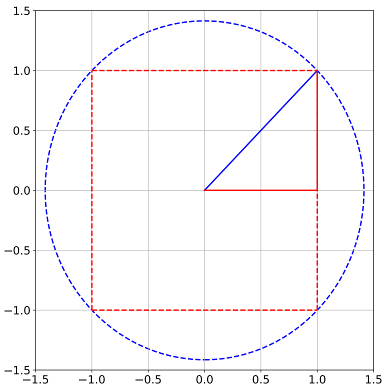

Figure 1: Blue indicates the Euclidean distance between data point $[0, 0]$ and $[1, 1]$, and Red represents the Manhattan distance of the same data points.

Next, let's apply two distance metrics in a sernario where we try to find two similar restaurants based on customer's satisfaction and cost, as shown in Table 1. The distance between a [Neutral, Neutral] restaurant and a [Neutral, Very expensive] restaurant is same as [Neutral, Neutral] and [Happy, Expensive] in Manhattan distance.

However, in Euclidean distance, [Neutral, Neutral] and [Happy, Expensive] have a shorter distance than [Neutral, Neutral] and [Neutral, Very expensive]. In this case, it makes sense to use Euclidean distance.

Manhattan distance works better in higher dimension than Euclidean distance.

Jaccard distance is often used for boolean vectors.

For a data set with mixed feature types, a transformation is required, e.g PCA.

1.2

$k = 1$ promotes more complicated decision boundary. Thus, it tends to overfit the data. However, it is very useful in high dimension cases and large data sets, where the distance is difficult to compute, or sorting data is very computational expensive. For example, finding the closest instance is a much easier task than finding the top 10 nearest point.

Table 1: Two features are encoded as integer values.

Encoded value	Satisfaction (F1)	Cost (F2)
0	Very unhappy	Very cheap
1	Unhappy	Cheap
2	Neutral	Neutral
3	Happy	Expensive
4	Very happy	Very expensive

Question 2 - K-Nearest Neighbour (KNN)

- Normalize your data. Distance metrics are extremely sensitive to unormalized data. In this case, F2 has 100 times of the weight than F1 and F3.
- Manhattan distance is an ok choice here. However, be aware of we are dealing with mixed data here. The optimal solution is that we transform the data, and then measure the distance on the transformed space.
- There is a caveat in applying kNN on the training data. Since every instance is already included in the training set, the the nearest neighbour is alway the instance itself. So in a 1NN model, the training set will always return 100% accuracy. To compute the true training set accuracy, we have to use $k + 1$, and remove the instance itself (nearest neighbour) from the queue.

Question 3 - Support Vector Machine (SVM)

1. Tuning C is no different to tuning other hyper-parameters we have used before. The standard procedure of a cross-validation is the way to go. Note that C is expected to be tuned as a coutinous log-uniform random variable, instead of linear, e.g., [0.01, 0.1, 1, 10, 100]. Here is the demo code: https://scikit-learn.org/stable/modules/grid_search.html
2. The rest of solutions are in the supplementary slides.