Understanding of KNN

KNN, K-Nearest-Neighbor, means when given a set of training data, importing some new examples, finding k nearest training examples (neighbors), and then determining the type of the new examples according to the majority type of their neighbors.

Because there is an existing class *sklearn.neighbors.KNeighborsClassifier (n\_neighbors=5, weights=’uniform’, algorithm=’auto’, leaf\_size=30, p=2, metric=’minkowski’, metric\_params=None, n\_jobs=None )* for KNN, so when analyze the audio, I used it directly. According to the principle and parameters of KNN, we can find several main parameters for KNN. The first is k, the number of the neighbors. When the value of k is small, the training error decreases, under this situation, only import the training examples which are similar to the training ones can the prediction results be comparatively precise. However, at the same time, the generalization error increases, and the model will become complex and easy to overfit. When the value of k is large, the advantage is that the generalization error decreases, but the training error increases, and has the risk of making the model too simple (Andy\_shenzl, 2018). It is hard to determine which value of k is the best choice by experience. To select the best value, I decided to use cross-validation. It uses a train set and a valid set, for each k, the model train the classifier with the train set and test the trained model with the valid set, record the accuracy and select the k with the highest accuracy. (Cerisier, 2017)

Another main parameters is metric. There are many ways to calculate the distance: Euclidean Metric, Manhattan Metric, Chebyshev Metric and Minkowski Metric. Because the relationship between Minkowski Metric and other metrics, we can use different metrics through adjusting the value of p. According to Natasha Sharma (2018), Manhattan Distance is used when we need to calculate the distance between two data points in a grid like path, Euclidean Distance can be used to calculate the distance between two data points in a plane. Because we transfer the audio into pictures, so the main point of audio analysis is actually image identification. For the reason that an image is composed of thousands of pixels so we can view the image as a n-dimension matrix (n is the number of the pixels) and the distances between the pixels are straight line distances, I decided to calculate the distances between the corresponding pixels on the two images with the most simplest and common matrix, Euclidean Distance.

The third vital parameter is the algorithm, Brute Force, KD tree or Ball Tree. First, we rule out the Brute Force. The time complexity of this algorithm is exponential, so it is comparatively inefficient. Actually, the basic ideas of KD tree and Ball Tree are similar, they all use nodes to store the information and devise the space. The difference is that the former uses hyperrectangle and the latter uses hypersphere. Notably, KD tree is more efficient than ball tree when there are fewer dimensions, Ball tree is more efficient when the dimensions are high and the dataset are unbalanced. However, we do not need to determine which algorithm to use by ourselves, when use the knn package, I choose the parameter algorithm as “auto”, which means that it will choose the most appropriate one for us.

Although I adjust the parameters to serve the data better, the result is still unsatisfying. There are several factors contribute to the results. One is the objective condition that there are only 45 training data for the limitation of the collected samples, but the data have almost one hundred and twenty thousand dimensions. And this may cause underfitting. The model is unable to capture the characteristics of the data and has a poor generalization ability in return. Another objective condition is that the data with the same label concentrate, which may cause overfitting. This is the reason why it gets high score on the training data and low score on the testing data. After I shuffled the original data, it turns better. And another shortcoming based on the principle of KNN is that when calculating the distances, there may be several points have the same distance from the target point, so the error may be made.

References

Andy\_shenzl. (2018). Understanding of KNN algorithm. Retrieved July, 25, 2019 from <https://blog.csdn.net/Andy_shenzl/article/details/82774689>.

Cerisier. (2017). Understanding of the generalization error. Retrieved July, 25, 2019 from <https://blog.csdn.net/Cerisier/article/details/78122653>.

Natasha, Sharma. (2018). Importance of distance metrics in machine learning modelling. Retrieved July, 25, 2019 from <https://towardsdatascience.com/importance-of-distance-metrics-in-machine-learning-modelling-e51395ffe60d?gi=563d26d0eb40>.