DD2434 Assignment 1

Cheng Yang

November 2020

Problem 7

github link: https://github.com/yangBryants/AML-assignment1-programming-task

(1) Data pre-processing

There are 101 instances in the zoo dataset, and each one corresponds to one animal(in particular, there is one instance of "girl") with 17 attributes.

There are 2 instances of "frog" with the same attributes so we first move the duplicate items and get 100 instances in total. Then we remove the attribute "*type*" which will be used as the color for each point in the visualization. We don't need to use the "*animal_name*" to tag each data point because the names of the 100 instances are different from each other. Now we have got a 100×16 dataset.

Note that all attributes are Boolean, except 'legs'. So we firstly standardize the dataset using the function *sklearn.preprocessing.StandardScaler*. The standard score of a sample x is calculated as: $z = (x - u)/s$, where u is the mean of the training samples and s is the standard deviation of the training samples.

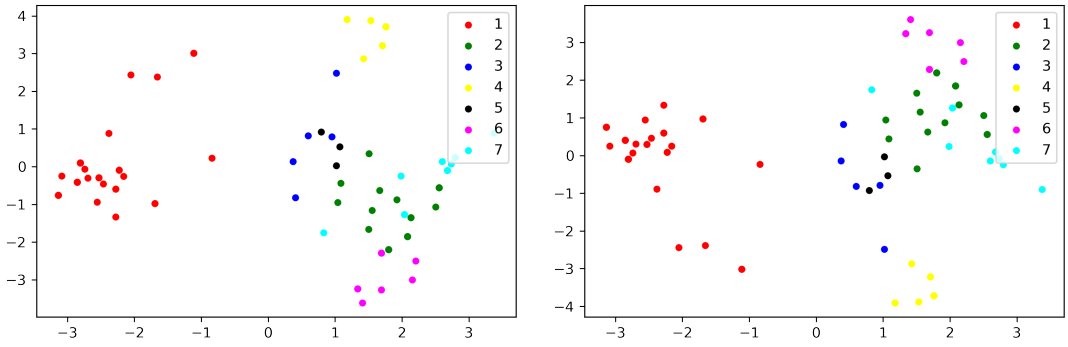
Finally, we want to see how many instances there are for each category:

(type: numebr) 1: 41, 2: 20, 3: 5, 4: 13, 5: 3, 6: 8, 7: 10

There number of different classes varies, eg, there are only three instances with label 5 but 41 instances with label 1.

(2) The visualization of the dataset using dimensionality reduction

The first thing to point out is that our goal is to separate the points as much as possible, but this may



(a) PCA (b) MDS

Figure 1: PCA and MDS results

not be objectively correct. Because our animal categories (colors or numbers) are manually labeled, they may be related to other animal characteristics, and not entirely determined by the attributes in the data set. In other words, if we care too much about the original animal category, overfitting may occur and reduce the generalization ability of the model.

The result of the PCA and MDS dimensionality reduction is shown in Figure 1(a) and Figure 1(b). Different types of instances are marked as dots with different colores. In the following description, we use colors to replace the animal type number.

From question 5, we know that the results of PCA and classical MDS should be the same. We can see that the two pictures are symmetric about the x axis. This is because the second principal components returned by the two algorithms are somehow opposite to each other. This does not affect our classification results.

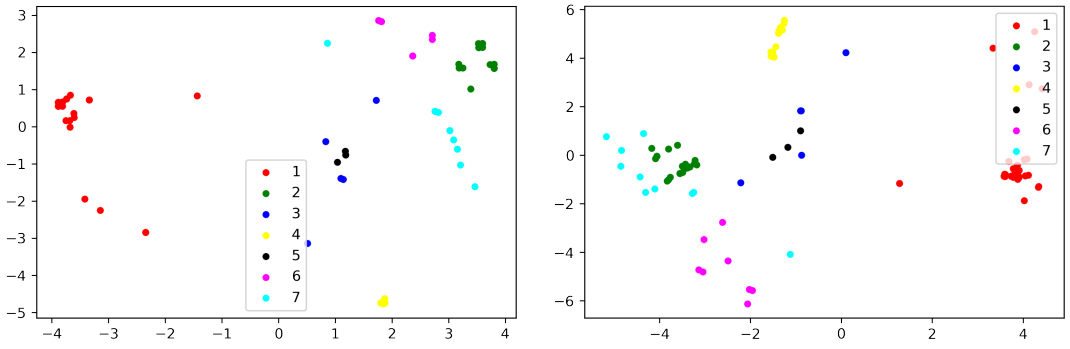
There are some problems with the clustering here:

1. Black dots are mixed with blue ones, which may be related to the small number of black dots (only 3).
2. Cyan and green dots are mixed.
3. One blue dot very close to the yellow dots.

Let's see if the weighted distance matrix can solve these problems. Theoretically, we hope to give greater weight to more important features, that is, to stretch its corresponding vector, which will cause the distance between data points with a large difference in this feature to increase. Because MDS will maintain the distance, So the distance between the data points in the final two-dimensional coordinates will increase, which will make our clustering results better.

We use random forest classification with animal "*type*" tags as the target to calculate the importances of features. The more important the feature is, the greater the weight is. Then we multiply the data column by the corresponding feature importance, and then calculate the weight distance matrix as the input of the MDS algorithm. It should be pointed out that the random forest needs to be tuned. We use 50 decision trees, and the other parameters are default.

The result is shown in Figure 2(a), we can see that the data points of the same color are closer to



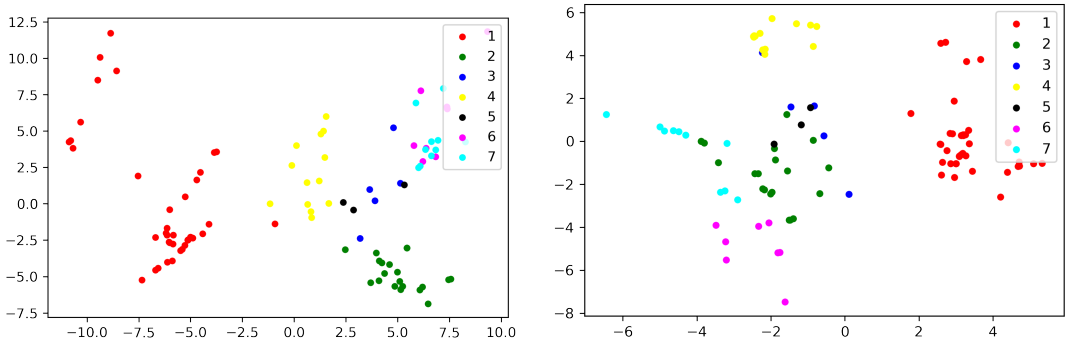
(a) MDS (b) Isomap, $k = 90$

Figure 2: MDS and Isomap results with weighted distance matrix

each other. There are still some problems: the black points are still between the blue points, and there is one cyan outlier.

Next we implement the Isomap algorithm. We use the Floyd-Warshall algorithm to calculate the

shortest path distance. We solve the problem in the problem 6 by increasing the value of k. In the normal Isomap, the minimum value of k is 18. If the input is the same weight distance matrix as before in MDS, the minimum value of k will increase to 37 because of the fact the weight will generally increase the distance between the data points, making it more difficult to connect the data points.



(a) $k = 18$ (b) $k = 60$

Figure 3: Isomap results

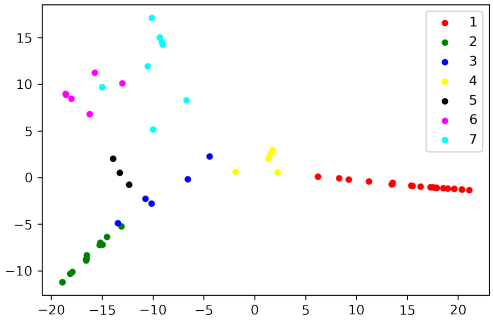


Figure 4: Isomap using the sklearn module with $k = 4$

To search for the best k, we plot the results of k from 18 to 97 and from 37 to 96 , and the best k is selected from them, respectively. For the normal Isomap, when k is small, the magenta point and the cyan points can't be seperated which is shown in Figure 3(a) , but when k is relatively large, the cayan point and the green point can't be separated which is shown in Figure 3(b). To make a trade-off, we select the optimum k at 60. For the Isomap with weighted distance matrix, we set the optimal k at 90 which is shown in Figure 2(b).

Compared with MDS, Isomap does not improve the accuracy of clustering very much. For example, the problem of black dots mixed in blue dots we mentioned before is not solved, and the clustering results of other dots are not as good as MDS with weighted weighted distance matrix. In addition, the parameter k needs to be adjusted, which increases the time of computation.

The Isomap result using the sklearn module is shown in Figure 4. It works better than what we have done before, which means that there is still a lot of room for improving the algorithm.

To summarize, the MDS with weighted distance matrix works best in this scenario among our algorithms.