

PRML（ProjectⅡ）

Pen-gesture Recognition with Hidden Markov Models



学 院（系）： 电子信息与电气工程学部

专 业： 计算机科学与技术

班 级： 电计1704

学 生 姓 名： 谢玉宁

学 号： 201792260

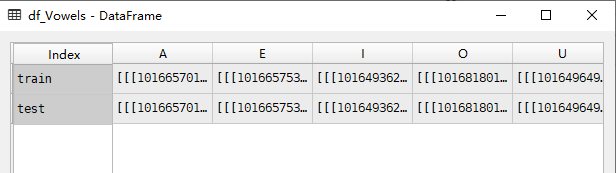
完 成 日 期： 2020年6月19日

# *Keywords*:

*K-Means++, HMM, Scaling, Multiple Observation Sequences, DTW, Viterbi decoding, Clustering feature, Direction feature, left-right model.*

1. Get training data and testing data.

Take the odd indexed entries as training data and the even ones as testing data. I used Beautifusoup to get the data set from .xml file and store it as form of pandas.DataFrame.



1. Spatial clustering algorithm (K-Means++)
2. K-Means++

To cluster the training data, I used K-Means++ algorithm. This algorithm is an improved K-Means, which applies a more reasonable method to initialize the center of each cluster than K-Means which randomly pick some points as initial centers. To better separate the data by clustering, K-Means tries to make the initial centers far from each other, therefore it applies *Roulette Wheel method* to make it more likely get initial centers far from each other.

1. The clustering results

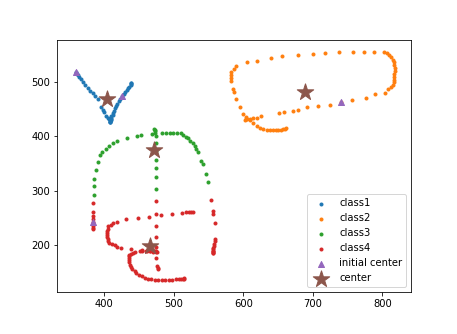


Fig 1 K=4, 1 Sample K-Means++ clustering result

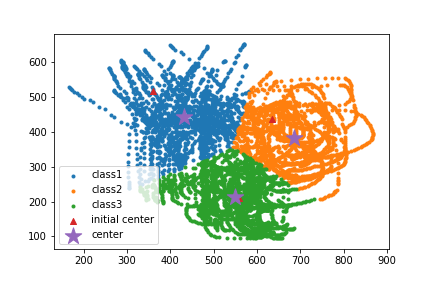


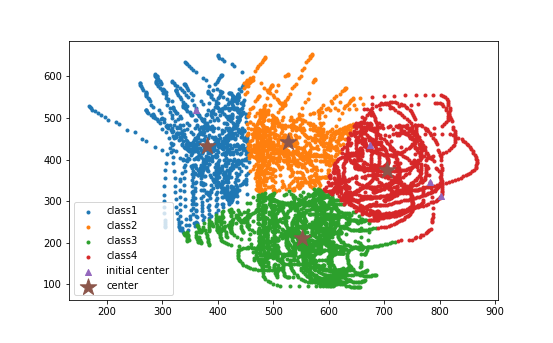
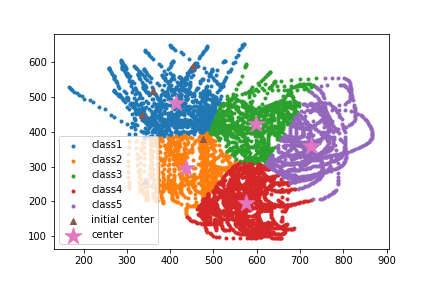
Fig 2 K=3, K-Means++ clustering result

Fig 3 K=3, K-Means++ clustering result  Fig 4 K-Means++ clustering result

1. Train a separate HMM for each vowel
2. Scaling α and β

When training the HMM, since each *a* and *b* term is less than 1(generally significantly less than 1), it can be seen that as t starts to get big, each term of *α* starts to head exponentially to zero. For sufficiently large t, alpha computation will exceed the precision range of essentially any machine. So I scaled every*α*within the dynamic range of computer. also, I used the sum of logged coefficients as log-likelihood.

1. Multiple Observation Sequences

The modification of the reestimation procedure is input a series of observation sequences to train the mode. Implement in function *learn\_from\_MOS(args)*

1. Evaluation

I used the log-likelihood as evaluation in case the probability of the sequence underflow.

1. Classifying results
   1. Using cluster feature

I transferred every spatial points of the training sequence to a cluster number so I got a series of sequences as training input. Results are as follows.

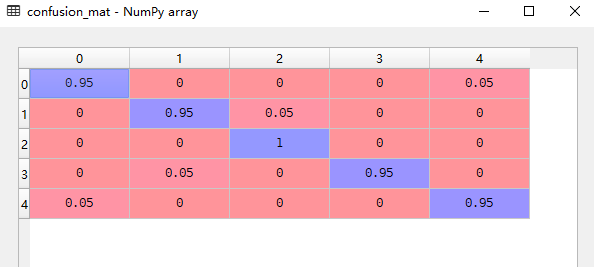


Fig 5 K-Means++ clustering feature confusion matrix (K=8, States num=4), accuracy= 96%

* 1. Using direction feature

I also used direction feature of data to classify their category. For each two adjacent points, the number after subtracts the number before is the direction of two points, so I got a series of sequences using direction feature as training input. Results are as follows.

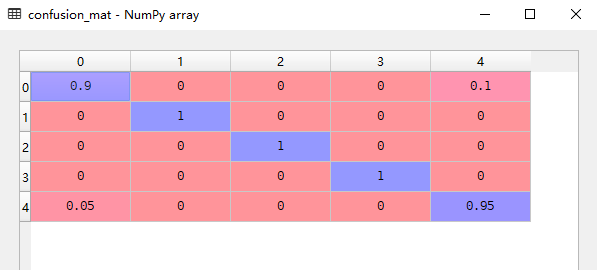


Fig 6 direction feature confusion matrix (direction num=6, States num=4), accuracy= 97%

1. Relationship between final accuracy and the parameters

I varied the number of clusters, the number of directions and the number of hidden nodes and repeat the quantiﬁcation.

The relationship between results and the variables:

1. Using cluster feature (average from 10 experiments)

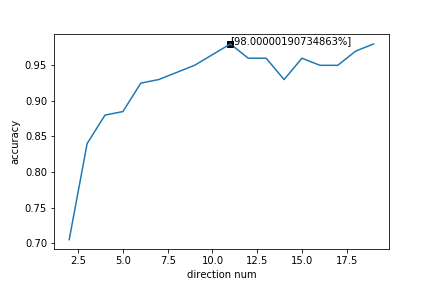


Fig 6 relation between accuracy and cluster numbers.

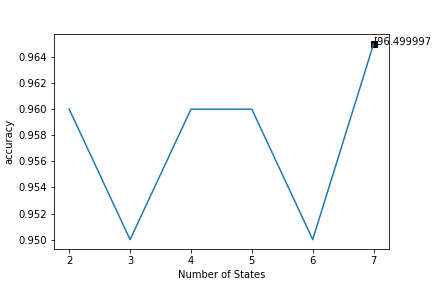


Fig 7 relation between accuracy and cluster numbers.

1. Using direction feature (average from 10 experiments)

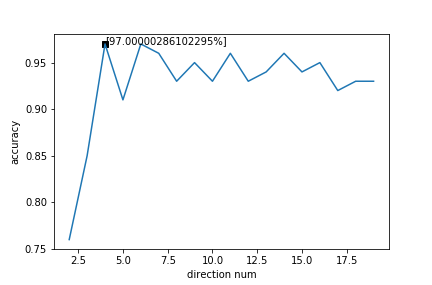


Fig 8 relation between accuracy and direction numbers.

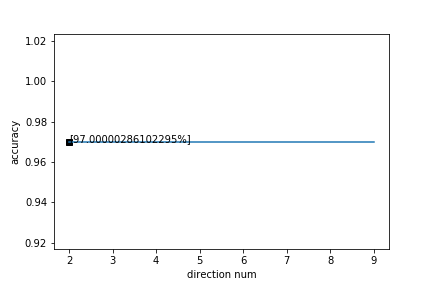


Fig 9 relation between accuracy and number of hidden nodes.

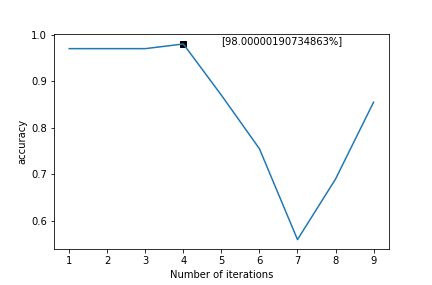
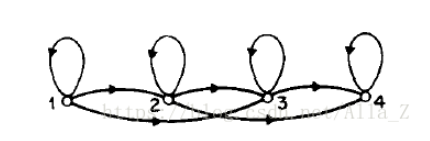


Fig 10 relation between accuracy and iteration times.

1. Using left-right HMM model

Apply left-right HMM to prevent overfit, because when the number of states nodes is too large, the parameters will be too many, therefore the model will be complex so the training procedure will make the model overfit.

However, if we constrain the states of model to be able to transfer to single direction, the parameters we train will decrease a lot.



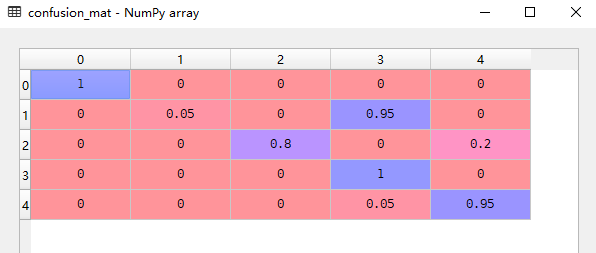


Fig 11 N=12, dirnum=10, iternum=10, confusion matrix. Ergodic model

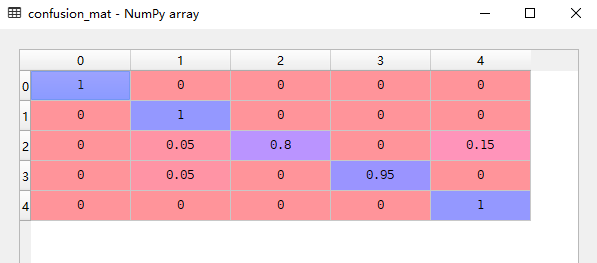


Fig 12 N=12, dirnum=10, iternum=10, confusion matrix. Left-right model

1. Founding according to results above.

The accuracy grows at first when the number of observation feature are small, but as the number of features grow, the accuracy will decrease and fluctuate.

For Ergodic model, the iteration time have a bad influence on the accuracy. This is probably because overfitting, but for left-right model, the accuracy can keep as high as that when iteration time is small, as this model has less parameters and is too easy to overfit.

The number of states node doesn’t have an obvious impact on the model by direction feature but have great influence on model by cluster feature.

1. Option: Dynamic Time Warping

I used the DTW to calculate the distance between one test data and each training data and got the total distance between test data and each vowel character, then chose the vowel of the smallest distance as prediction, and then calculate the confusion matrix and accuracy.

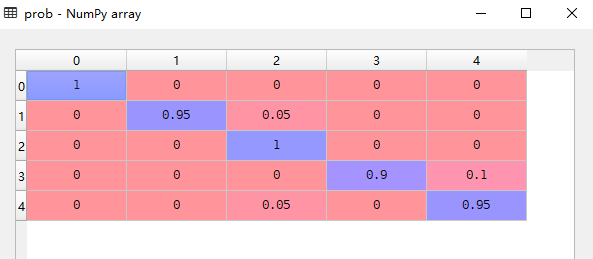


Fig 13 DTW confusion matrix.

1. Option: Decoding problem implemented by Viterbi algorithm

The algorithm is complemented by function: decoding(self, observ\_seq) but I didn’t implement a HMM model which can classify the 5 vowels.

**Sources**

K-Means++: K\_Means\_pp.py

HMM: HMM.py

DTW: DTW.py