CS 5786 - Competition 2 Report

FLAG (Ziyi Wang(zw376), Tanming Cui(tc634), Yuning Yang(yy693), Lu Chen(lc856))

1. *Understanding Problem and Data*

Problem: The problem could be explained as the graphical model in figure 1. First the bot(C) is chosen according to a distribution pi, then the bot would move to different unseen locations (Sn) step by step. Every location is only decided by its previous location and which bot this is. (the movement pattern of that specific bot) At each step, we can only observe its distance to the top left corner (Dn).

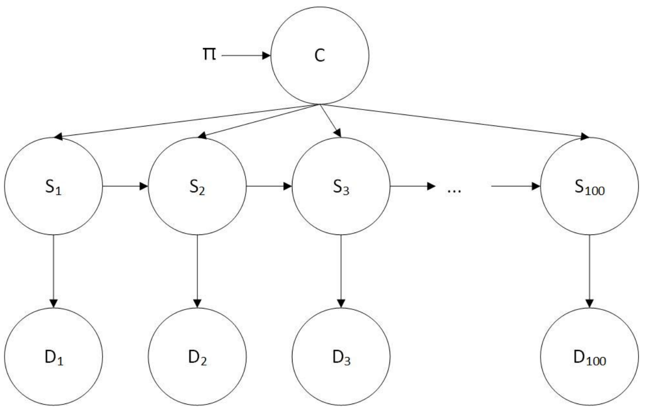


Figure 1: Graphical model for the problem

If we could distinguish the observation sequence of each bot, that we cluster the different runs (observation features) into 3 clusters and find the movement pattern (transition probabilities between states) of each cluster. And then we could simplify this problem as a mixture of 3 HMM and we can put each run in one cluster and train each cluster to get state of 100th step for each run.

Once the bot is chosen, each run is a generative process of normal HMM as we discussed on class. The unseen states are the locations of each step, S1, S2, ...S100. And the location is a sequence. Each location have influence on the next location and is only decided by its previous location. For each location, we can only observe the distance feature of it, D1, D2, ...D100, which is decided by its corresponding location (state). The model shows like figure 2.

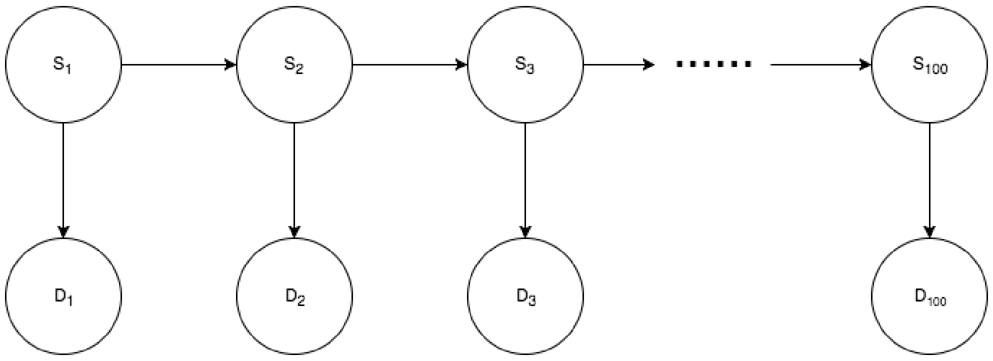


Figure 2: HMM model for each run

Data: Label table shows the final location of 5*5 grids for 200 runs. And the final location's euclidean distance from the top left corner of these 200 runs can be got from observation table. According to the distance and locations of these 200 runs, we can match the distance and location as table1.

5*5	1	2	3	4	5
1	1.000000	2.000000	3.000000	4.000000	5.000000
2	1.414214	2.236068	3.162278	4.123106	5.099020
3	2.236068	2.828427	3.605551	4.472136	5.385165
4	3.162278	3.605551	4.242641	5.000000	5.830952
5	4.123106	4.472136	5.000000	5.656854	6.403124

Table 1: matching of distance and location

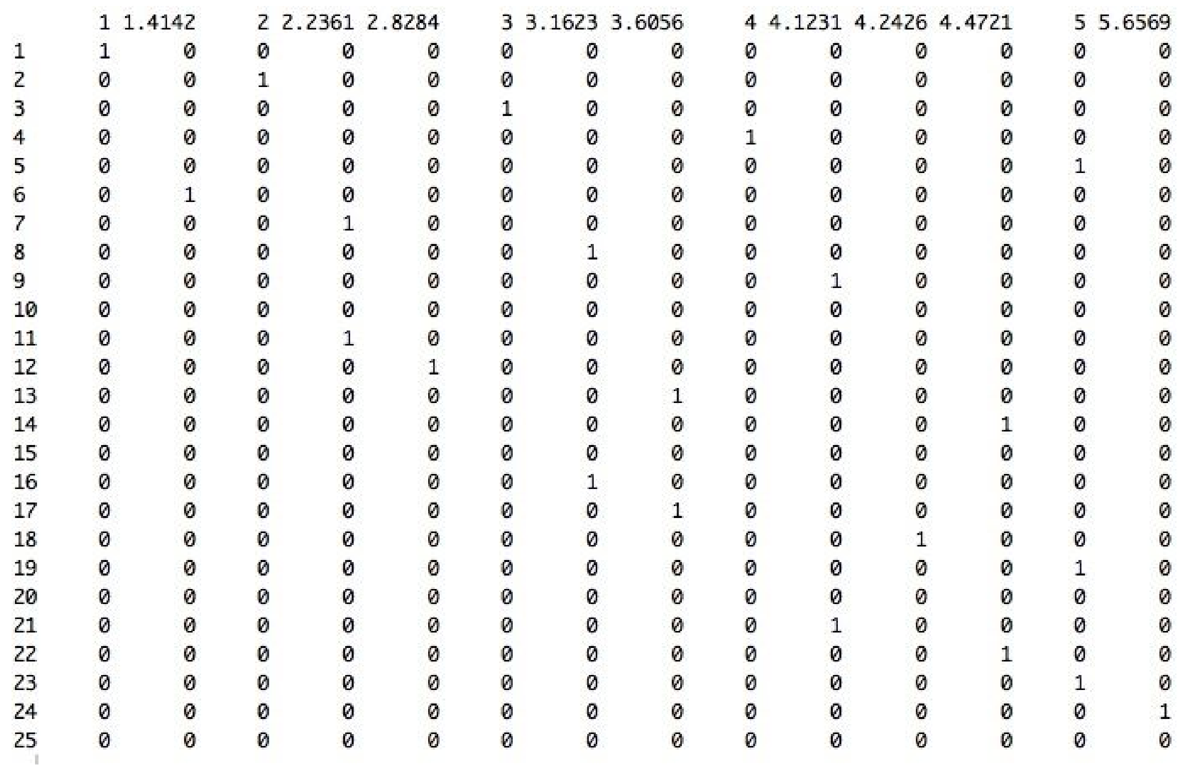
However, after summarize the observation data set, we could find that there are only 14 unique symbols in it, which is 1, 2, 3, 4, 5, 1.4142, 2.2361, 3.1623, 4.1231, 2.8284, 3.6056, 4.4721, 4.2426, 5.6569. This means that the bot can never reach the location 10, 15, 20, 25. Besides, the first distance is every sequence is always 3.6056.

2. Main Model Development

2.1 Preprocessing: Parameter Initialization

2.1.1 Initialize Emission Matrix

The emission matrix shows the probability of getting one distance if the location is given. Under the situation of this competition, there will be only one distance if the location is known. Since there are 25 kinds of locations and 14 kinds of distance, we generated a matrix with 25 rows and 14 columns with the elements of 0 and 1. The table of emission matrix is shown as table 2. If we can observe distance j (j = 1 matches dist = 1, j = 2 matches dist = 1.4142, they are matched in ascending order) at location i, then entry (i,j) is 1. Otherwise, it's 0. Every entry for location 10, 15, 20, 25 is 0 since no distance matches these 4 locations.

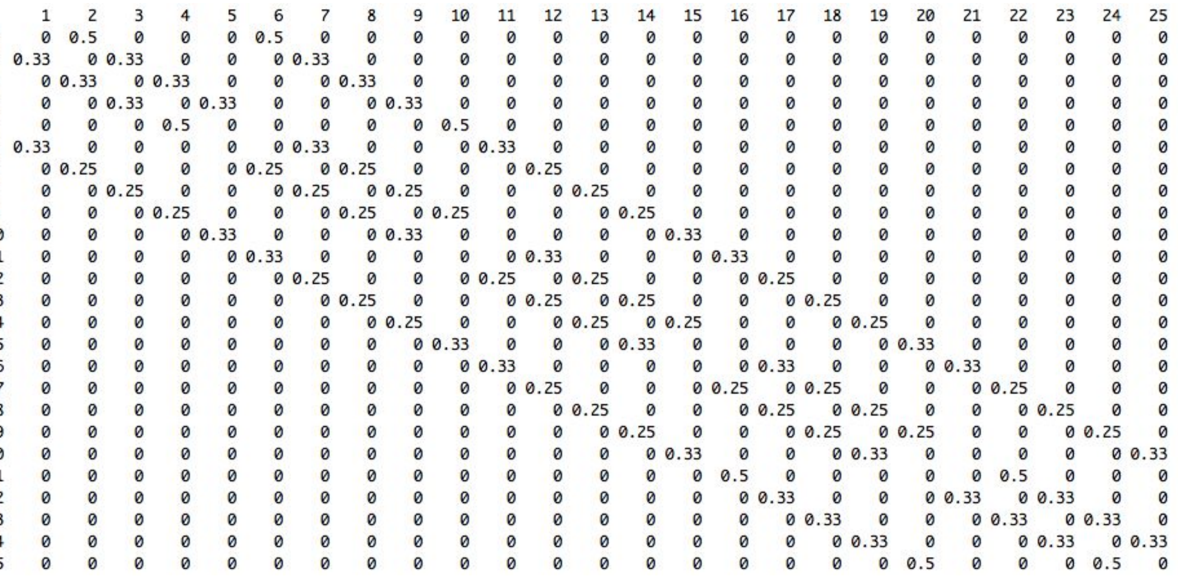


	1	1.4142	2	2.2361	2.8284	3	3.1623	3.6056	4	4.1231	4.2426	4.4721	5	5.6569
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	1	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	1	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	1	0
6	0	1	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	1	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	1	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	1	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	1	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	1	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	1	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	1	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	1	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	1	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	1	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	1	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	1	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	1	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	1	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	1
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2: Initial Emission Matrix

2.1.2 Initialize Transition Matrix

The transition matrix shows the probability of transiting from one state to another. There are 25 states in total so the matrix has 25 rows and 25 columns. We assumed that the bot had equal probabilities to move to its neighbours. Under this hypothesis, we initialized the transition matrix as table 3.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	0	0.5	0	0	0	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0.33	0	0.33	0	0	0	0.33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0.33	0	0.33	0	0	0	0.33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0.33	0	0.33	0	0	0	0.33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0.5	0	0	0	0	0	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0.33	0	0	0	0	0	0.33	0	0	0	0.33	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0.25	0	0	0	0.25	0	0.25	0	0	0	0.25	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0.25	0	0	0	0.25	0	0.25	0	0	0	0.25	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0.25	0	0	0	0.25	0	0.25	0	0	0	0.25	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0.33	0	0	0	0.33	0	0	0	0	0	0.33	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0.33	0	0	0	0	0	0.33	0	0	0	0.33	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0.25	0	0	0	0.25	0	0.25	0	0	0	0.25	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0.25	0	0	0	0.25	0	0.25	0	0	0	0.25	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0.25	0	0	0	0.25	0	0.25	0	0	0	0.25	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0.33	0	0	0	0.33	0	0	0	0	0	0.33	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0.33	0	0	0	0	0	0.33	0	0	0	0.33	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0.25	0	0	0	0.25	0	0.25	0	0	0	0.25	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0.25	0	0	0	0.25	0	0.25	0	0	0	0.25	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0.25	0	0	0	0.25	0	0.25	0	0	0	0.25	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.33	0	0	0	0.33	0	0	0	0	0	0.33
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	0	0	0	0	0	0.5	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.33	0	0	0	0.33	0	0.33	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.33	0	0	0	0.33	0	0.33	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.33	0	0	0	0.33	0	0.33
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	0	0	0	0.5	0

Table 3: Initial Transition Matrix

2.1.3 Initialize Initial State Distribution Matrix

We also have to initialize the initial state distribution matrix, that with what possibility the bot would step on each location at the first step. According to the observation data set, the distance of the first steps of all the 3000 runs are 3.6056. Based on Table 1 we got above, we could assume the first location is 13 or 17. Therefore, we generated a 1 * 25 matrix with 0.5 at the 13th and 17th columns and 0 at other columns as the Initial State Distribution Matrix.

2.2 Clustering

After trying some initial clustering methods (details are discussed in failed attempts), we decided to find some clustering method that could give us a more stable and reliable clustering result. Inspired by the novel initialization proposed by Padhraic Smyth[1], we worked out our clustering based on his method with the combination of some methods learnt in class. Details are described as follows. We implemented all HMM related algorithms using HMM toolbox in MATLAB by Kevin Murphy[2], which is reliable and somehow faster than the built-in HMM functions.

2.2.1 Building Similarity Matrix

1) As mentioned in Padhraic Smyth's paper, first train each sample using uniform set of parameters with Baum Welch algorithm respectively. Then we would get 3000 probable transition matrices. In the competition, we used a default emission matrix as discussed in the previous part and a default transition matrix as the initial inputs to train each sample. The default transition matrix is constructed as follows: Assume the bot would move to its neighbors with equal probabilities. Therefore, the bot at the center would have a possibility of 1/4 of moving to its 4 neighbours. The bot at the top left corner would have a possibility of 1/2 of moving to its 2 neighbours... After that we would get a probable transition matrix for each sample.

2) Use each transition matrix to compute the log-likelihood of each sample using that model. Then we would get a 3000*3000 matrix, each entry (i,j) is the log likelihood of seeing observation i using the parameters of j. In this regard, each entry could represent the similarity of two samples. Besides, we add a small 0.000001 to each entry of the transition matrices to allow for a small possibility of transformations between different states. Or we would get many -Inf log-likelihoods and miss some trivial similarity. The additional 0.000001 would not affect the general pattern.

3) Compute the likelihood of each sample using each model. We simply take exp() of the log-likelihood matrix and get a 3000*3000 similarity matrix. Each entry is in the range [0,1]. If entry (i,j) is close to 1, it indicates that seeing i with model j is of large possibility and i and j are similar.

2.2.2 Spectral Clustering with GMM

In Padhraic Smyth's paper, after computing the log-likelihood matrix, the next step is to cluster the samples. Since we get a similarity matrix, we could now regard the samples as nodes in a weighted graph and further cluster the samples using spectral clustering. We first processed the similarity matrix and set its diagonal element as 0 to transform it into an adjacency matrix.

As it is known from the problem description, there are 3 bots, which indicates that there are 3 kinds of samples. Therefore, we used spectral clustering with k = 3 to reduce the dimensionality of the adjacency matrix to get a 3000*3 new matrix. Then we could cluster our samples with this new matrix.

At first, we cluster the samples using K-Means. However, since K-Means is sensitive to the initial centroids and only considers the distances between samples and centroids, the result is not satisfying. Even though we could see there are three clusters by plotting out the 3 dimensions of the 3000*3 matrix, some of the points are mis-clustered (see Fig3.). Therefore, we further applied GMM based on the clustering result of K-Means. GMM considers the distribution of different dimensions and would get a more accurate clustering result in comparison with K-Means if used properly. And this clustering we get is slightly different from the clustering we get as described in failed attempts.

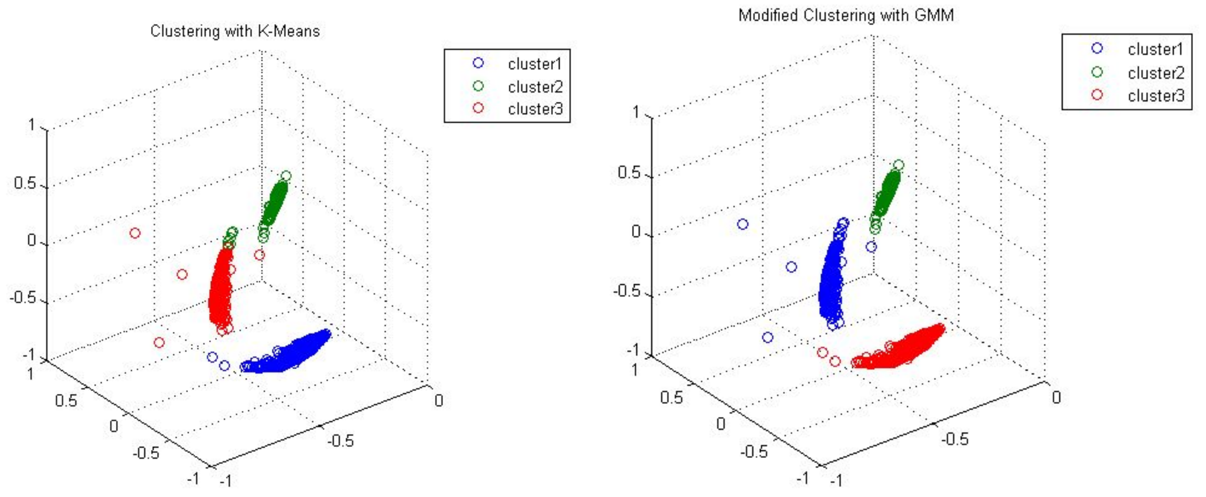


Figure 3. Clustering with K-Means (Left) & Modified Clustering with GMM (Right)

2.3 Learning Parameters for Each Cluster

Once we get the three clusters, we could train each cluster using Baum Welch algorithm to learn its specific transition matrix. And after we get the corresponding parameters, we could find out the most probable path of each observation sequence with inference using Viterbi algorithm.

Finally we can take the 100th state(location) of this path as our final result. However, since Baum Welch is a kind of EM algorithm, there is no guarantee that it would converge to global optimum and the final result is quite sensitive to initial parameters. And our initial guess of the transition matrix and initial state distribution might not be truth. This might lead to a false result.

With our fixed initial parameters as described in previous part (convergence tolerance: 0.000001), we ran Baum Welch and Viterbi on the three clusters respectively and compared our result for the first 200 runs with the labels professor provided. We found that most labels match but still some labels did not match. We submitted the result and got an accuracy of about 0.8.

By then we realized that this might be due to the limitation of EM algorithm and our initial guess might not lead to the correct results. Therefore, we further experimented some other initial matrices. For example, we experimented several initial state distributions. Since our default initial state distribution is 0.5 for both location 13 and location 17 in preliminary attempts, the initial state distribution after Baum Welch of one of our cluster seemed to be unstable. Sometimes it would be 0 for 13 and 1 for 17 after Baum Welch and sometimes it would be 0 for 17 and 1 for 13. And when it is 0 for 17 and 1 for 13, we would get more accurate results that match the labels. Initial state distribution matters in this case. Likewise, initial transition matrix would also influence the final result.

2.4 Supervision

Therefore, we experimented with some more **random initial parameters (including transition matrices and initial state distributions)** and **checked the result with the labeled data**. The much data were matched, the much possible that we worked out a reliable model. Besides, we matched our result with the labeled data for each cluster respectively. If most of the labels within one cluster were matched, then the model for this cluster after this certain run might be reliable. Finally we combined the results of several runs and submitted our results. In this way, we also minimized our submission times to achieve better performance. In two of our submissions, we got the accuracy on public board of 0.98643 and 0.99214 respectively.

3. Model Interpretation

3.1 How does your **model fit** the problem description?

Our main model consists of 3 sub-HMM models. We use all of the runs of one bot to build each HMM model. First a bot is chosen according to a fixed distribution pi. This distribution could be computed using the number of samples in each cluster/number of all samples. Once the bot is chosen, one single run of this certain bot could be considered as a generative process of its corresponding HMM. Multiple runs of the same bot could be considered as generated by a set of fixed parameter. The transition matrix describes the movement pattern of the bot, that with what possibility the bot is going to move to another state(location) given his current state(location). The emission matrix describes the relationship between states(locations) and observations(distances), that what would the observation be given current state. And the initial state distribution describes how the bot would make his first step.

Besides, we implemented **new algorithm** to cluster the runs for fitting the problem better. Firstly, we fit models to each individual sequence, which will lead to noisier estimates of the model parameters (transition matrices & initial state distribution). So we need to cluster the parameters into 3 groups about the true values of the parameters. And the log-likelihood is a natural way to define the similarity of each two runs. And we refine the similarity matrix to adjacency matrix for spectral clustering. In this way, we can cluster runs into three groups and then are able to use the runs of one bot to build each model.

3.2 How does **model** account for the fact that there were **3 bots**?

We clustered the observation sequences into 3 clusters and each cluster refers to the movements of one bot. Each cluster has different transition matrices, which explains their different movement patterns.

Indeed, we further confirmed that there should be 3 bots, not 2,4, … bots. The package “NbClust” in R provides 30 indices for determining the optimal number of clusters. In our model, we got an adjacency matrix based on the Log-likelihood between each run. We treated this matrix as an input and got the plot showing the frequency of optimal number of clusters among all the indices. We could find that the peak of the plot is k = 3 (Figure 4), which means that the adjacency matrix we got indicated that there should be 3 different bots with different transition matrices.

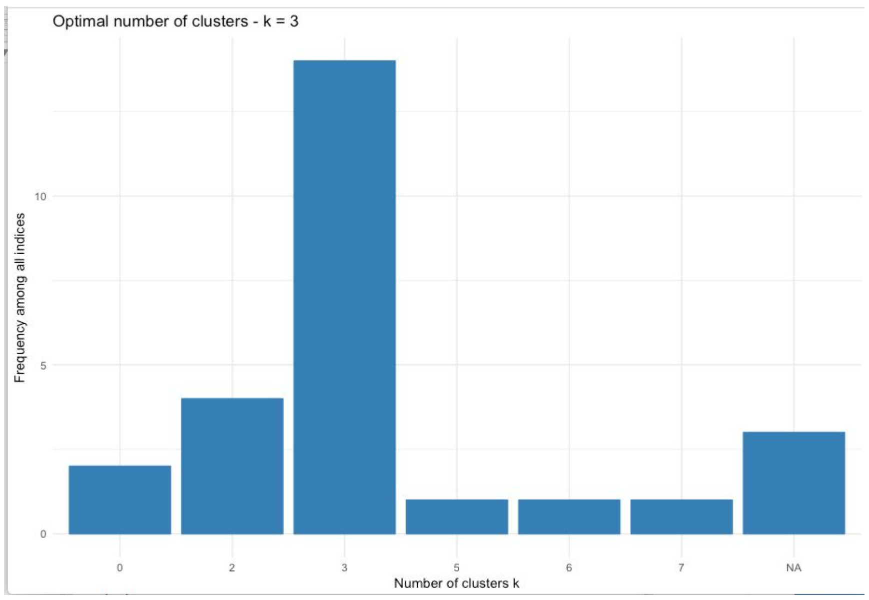


Figure 4. Optimal Number of Cluster - K = 3

3.3 Unlabeled examples: How were the unlabeled data points part of you model?

1）We cluster the labled and unlabeled data points to 3 groups. We fit model to each individual sequence and get model parameters. And we use log-likelihood to define the similarity of each two runs. And we refine the similarity matrix to adjacency matrix for spectral clustering. Then, we can cluster runs into three groups by applying GMM and k-means.

2) In each cluster, we build a HMM by using the unlabeled data and labeled data in this cluster. And we can predict the 100th steps for each run by using the HMM of their cluster.

4. Failed Attempts

*4.1 Training HMM Model with 25*3 States*

Due to three types of bots that are known, and our model is that of a mixture of HMMs. We can observe that this mixture can be regarded as a “three components” HMM where the transition matrix A of our model is block-diagonal. A_1, A_2, A_3 correspondingly indicates the component transition matrixes of bot 1, bot 2, bot 3, every bot has 25 possible positions on chess box, so there are 3*25=75 states in total. And this way, we could simplify the model and combine all the 3 bots within one HMM model.

$$A = \begin{pmatrix} A_1 & & \\ & A_2 & \\ & & A_3 \end{pmatrix}$$

Figure 5. Example of Transition Matrix that combines 3 bots

Therefore, our first attempt was to form a 75*75 initial transition matrix and train all the 3000 data together using Baum Welch, and each transition component is the same as the initial matrix described above. However, only about 20% of our result matched the labels of the first 200 runs. This is because we apply too little prior knowledge to our model and EM algorithm is hard to compute a decent estimation. Therefore, instead of training all the data together, we should first work out the hidden clustering behind the observations and then train each cluster respectively.

4.2 Clustering Using the Transition Probabilities of Observations

The first clustering we tried is to cluster using the transition probabilities of observations (distances). Since there are only 14 kinds of distance the in data set, for each run, we built a 14 * 14 matrix to calculate the probability of transition from one specific distance to another. Then we treated the 196 numbers in each matrix as 196 variables and generated a 3000 * 196 matrix. We applied K-means clustering to this matrix to classify all the data into three clusters. This plot shows the first three principal components of the matrix, and the first three engin vectors can explain 99.08% of the data.

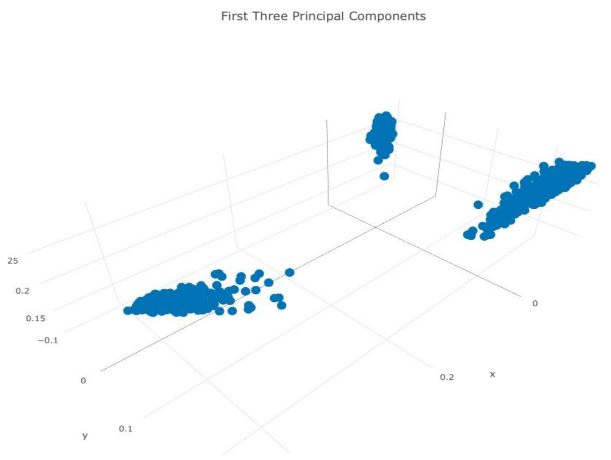


Figure 6. First Three Principal Components of The Transition of Observations

However, this method calculated the transition matrix using the distance instead of the locations. Even though we could see very good clusters in the plot, we consider this method not very reliable. We further tried some more principled ways as described in the main model.

References

[1] Smyth, Padhraic. "Clustering sequences with hidden Markov models." *Advances in neural information processing systems* (1997): 648-654.
[2] Kevin Murphy. Hidden Markov Model (HMM) Toolbox for Matlab.
https://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html