Due: 2019/4/14

Homework 1

Problem 1.

$$C_i^{\Delta} = \left(1 - A_{i-1}^{\Delta}\right) C_i + C_{i-1}^{\Delta}$$
$$A_i^{\Delta} = \left(1 - A_{i-1}^{\Delta}\right) A_i + A_{i-1}^{\Delta}$$

K-mean vs GMM (Variant of K-mean).

Solution.

(1) Derivation In this section, I will modify the soft assignment to hard assignment. In another word, I will assign the data point to a specific cluster, which is similar to K-mean method.

Similar to GMM model, I introduce a latent variable $\boldsymbol{z}$ to represent the cluster to which a point $\boldsymbol{x}$ belongs, i.e.

$$\boldsymbol{z} = (z_1, \dots, z_K),$$
$$z_k \in \{0, 1\},\ \forall\ k \in \{1, \dots, K\},$$
$$\sum_k^K z_k = 1,$$
$$p(z_k = 1) = \pi_k,\ 0 \geq \pi_k \leq 1,\ \sum_k^K \pi_k = 1,$$
$$\boldsymbol{x}_n = (x_1, \dots, x_p)^T,\ \forall\ n \in \{1, \dots, N\}.$$

I then assume that the points in the same cluster follow the same Gaussian distribution:

$$p(\boldsymbol{x}|z_k = 1) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k),$$

I get the distribution for data point $\boldsymbol{x}$:

$$p(\boldsymbol{x}) = \sum_k^K p(\boldsymbol{z})p(\boldsymbol{x}|\boldsymbol{z}) = \sum_k^K \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k).$$

I further define the r_{nk}, which is similar to the definition in the K-mean algorithm.

$$r_{nk} = \begin{cases} 1 & , k = \underset{j}{\operatorname{argmax}}\ \pi_j \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \\ 0 & , otherwise. \end{cases}$$

I call this the E step in our algorithm, since I evaluate the cluster to which the point $\boldsymbol{x}_n$ should belong in this step. What's more, with these r_{nk}, $\forall n \in \{1, \dots, N\}$, I can re-estimate the parameters of the K Gaussian distributions. And this is our M step.

The M step can be described in mathematic way:

$$N_k = \sum_{n=1}^{N} r_{nk}$$

$$\boldsymbol{\mu}_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk}\boldsymbol{x}_n$$

$$\boldsymbol{\Sigma}_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk}(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{new})(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{new})^T$$

$$\pi_k^{new} = \frac{N_k}{N}$$

(2) Algorithm

The algorithm is described in detail with pseudo code in the appendix A.

(3) Advantages

Compare our algorithm with GMM, the calculation of ours is reduced since each data point I only consider the most suitable distribution. This is achieved by hard assignment.

When compared with K-mean, our algorithm outperforms K-mean in dealing with the anisotropism inside the data points, because our algorithm utilizes the variance of the data.

(4) Limitations

I did not consider the features of competitive learning in this algorithm. Our algorithm still faces the similar limitations as the GMM and K-mean. The result is influenced by the initialization and it usually is not a global optimization. Though the convergence is guaranteed, it may take a long time to converge. The parameter K still needs to be set manually.

□

Problem 2. K-mean vs CL

Solution.

(1) Derivation

The competitive learning algorithm is similar to K-mean. The different between them is that the CL is an adaptive algorithm and the dataset is a sequence of data coming one by one, while the inputs of K-mean are batches of data.

Thus, I can apply the idea of RPCL into the K-mean. I can modify the definition of r_{nk} as follows:

$$r_{nk} = \begin{cases} 1 & , k = c,\ c = \underset{j}{\operatorname{argmax}}\ \pi_j \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \\ -\eta & , k = b,\ b = \underset{j \neq c}{\operatorname{argmax}}\ \pi_j \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \\ 0 & , otherwise. \end{cases}$$

For r_{nk} which is equal to 1, I apply the same method to update the $\boldsymbol{\mu}_k$. In another words, I neglect the r_{nk} which is equal to $-\eta$ in the following equation:

$$\boldsymbol{\mu}_k = \frac{\sum_n r_{nk} \boldsymbol{x}_n}{\sum_n r_{nk}}.$$

As for $r_{nk} = -\eta$, I apply the thoughts of RPCL. In this way, the rival clusters would move away from the cluster, as the equation

$$\boldsymbol{\mu}_j = \boldsymbol{\mu}_j + \sum_k \sum_n r_{nk} (\boldsymbol{\mu}_k - \boldsymbol{\mu}_j)$$

shows.

But different from the RPCL, the η should decrease during the optimization process. If the η is a constant, the $\boldsymbol{\mu}$ will continue changing when it should converge. Because the η makes the k-th cluster $\boldsymbol{\mu}_k$ away from the its real position even when the rival has been removed far away enough. Thus I consider modify the penalizing parameter to η / t, where t is the iteration times. As the iteration time increases, the penalizing parameter decreases.

There's another important point that we should label the deserted clusters in time. Because we update the variance of the distribution of clusters during the iteration. If we don't eliminate the cluster which is moved far away from data in time, the variance of it could become very large when we estimate the Gaussian distribution. This is critical in implementation.

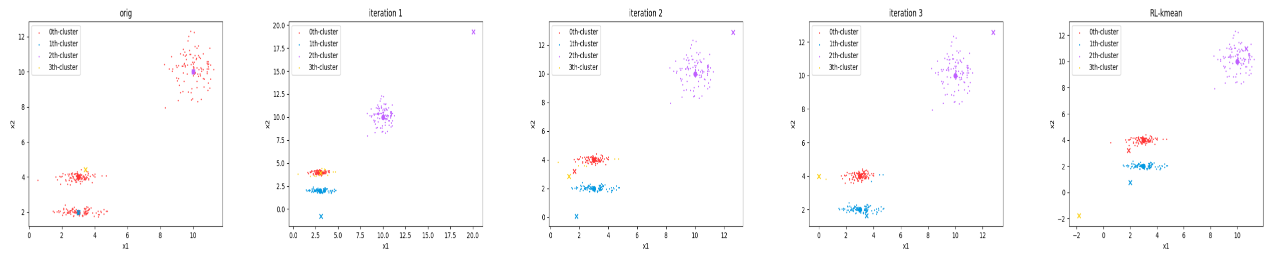
(2) Algorithm

The algorithm is described in detail with pseudo code in the appendix B.

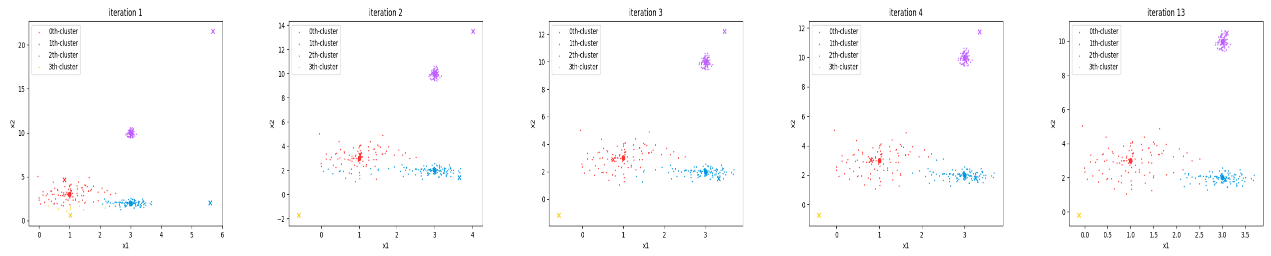
(3) Implementation

I implemented the algorithm above in Python. Then I test the performance with a dataset consisting three randomly generated clusters. The result is shown in Fig. (1).

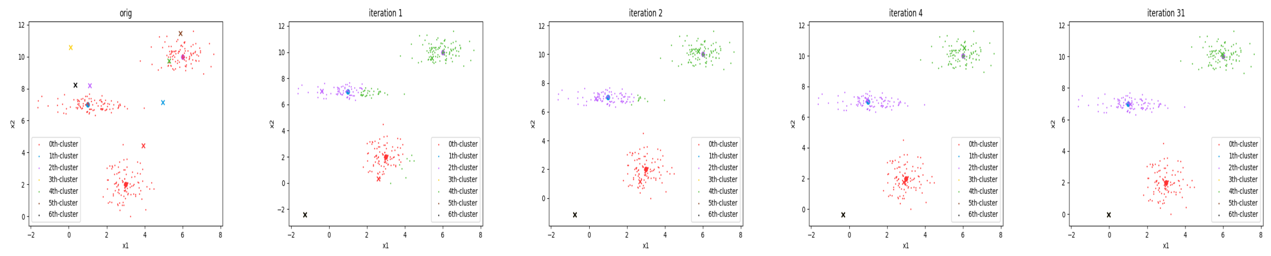
□



(a) Three cluster with four clusters being initialized.



(b) Three cluster with four clusters being initialized.

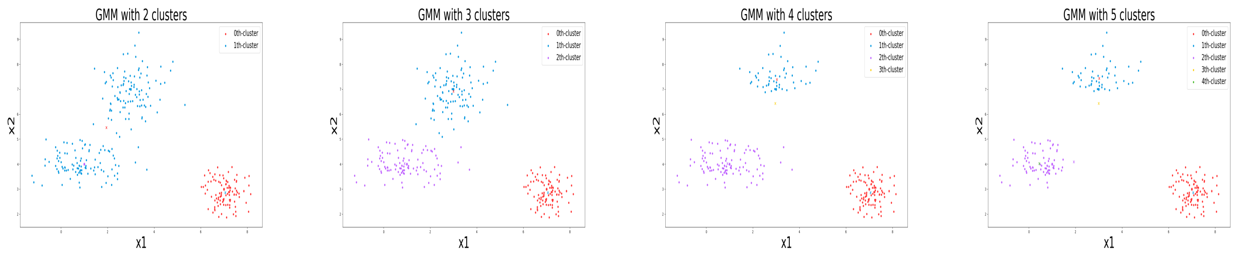


(c) Three cluster with seven clusters being initialized.

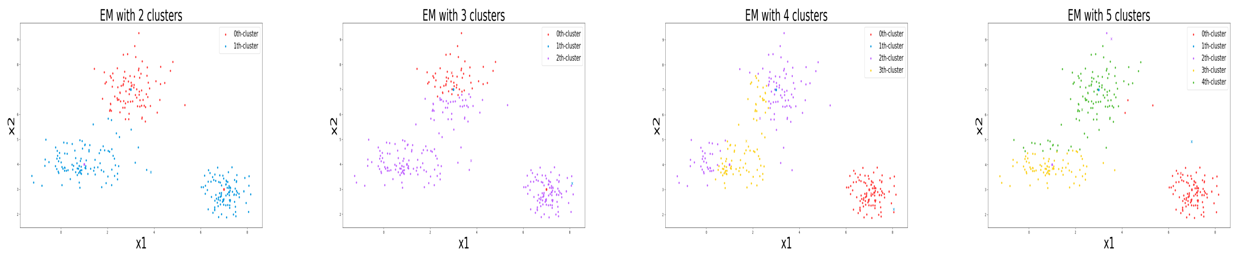
Figure 1: RP K-mean experiments.

Problem 3. model selection of GMM

Solution. In this section, I used the VBGMM model in sklearn library to do the experiment on VBEM algorithm. The experiment on basic EM algorithm with AIC and BIC criterion is carried twice using the GMM model in sklearn library and the EM algorithm implemented myself. And I will compare the results of them in detail.
The data is randomly generated from three Gaussian distribution. The initialization of GMM models are also randomly at the beginning. The results are shown in the Fig. (2). Compare the GMM model and EM algorithm implemented by myself, and we can find that GMM in scikit-learn library gives a better result than mine. The ”dot” is the position of the real means of three clusters, while the ”x” is the fitted means of clusters.
For VBGMM, we initialize the model with seven clusters, and the result is as Fig. ?? shows. It can be found that VBEM gives a good classification result.
According to the definition of the AIC and BIC criterion in the slides. Akalike’s Information



(a) Three cluster with four clusters being initialized.



(b) Three cluster with seven clusters being initialized.

Figure 2: GMM and EM with different cluster number initialization.

Criterion:

$$\ln p(\boldsymbol{X}_N|\hat{\Theta}_K) - d_k$$

Bayesian Information Criterion:

$$\ln p(\boldsymbol{X}_N|\hat{\Theta}_K) - \frac{1}{2} d_k \ln N$$

In the document of the sklearn library, it is said that "The lower the criterion is, the better the model is". In the Fig. (4), the upper two sub-figures are derived from GMM model in the sklearn library, and the other two figures are from our EM model. According to the scikit-learn document, we should choose the model with least AIC or BIC score. I change the sign of my AIC function and BIC function to get a positive value. In the upper part of the Fig (4), because I initialize the EM algorithm with random cluster position, the AIC and BIC changes every time we run the code. And sometimes they are not helpful in model selection. And if we initialize the EM algorithm with knowledge on the real distribution, the AIC and BIC are nearly the same each time and they are helpful to choose the best model with three cluster. Since the peak of BIC curve is evident than AIC curve, I think BIC is more useful than AIC.

In a word, we can make the conclusion that the models with three clusters are the best models. All of the experiments finally get similar predictions to the real distribution. □

During working on this homework, I learned a lot on the numpy library and sklearn library. Though coding really is tough, it makes me understand the algorithms in the class more thoroughly. The comparison between the famous library and the codes implemented by myself is interesting. Finally, thanks for the help from our teacher and assistants.

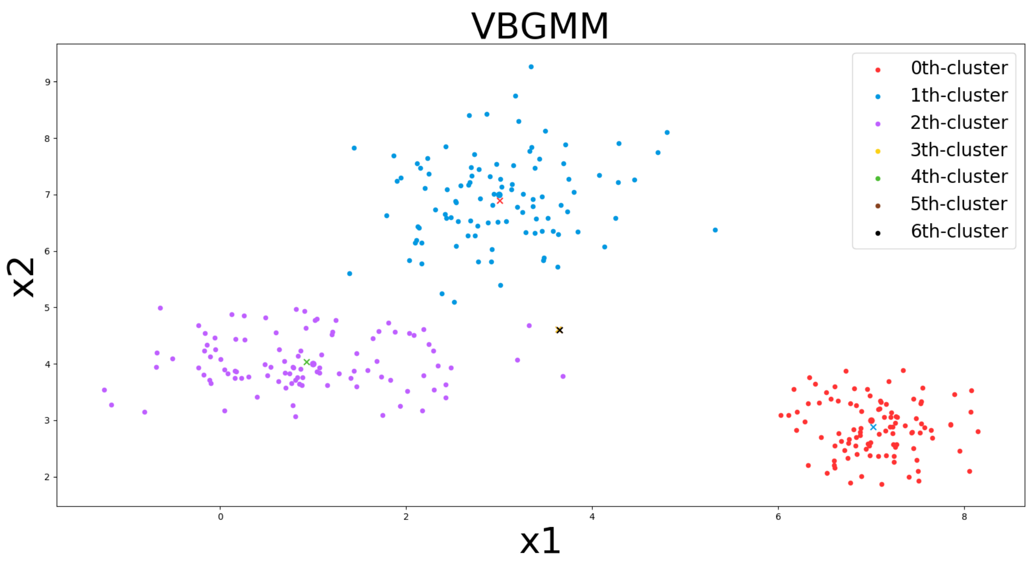


Figure 3: VBGMM running result.

A Proposed variant K-mean algorithm

Algorithm 1 Proposed variant K-mean algorithm

```
Input: Prior distribution of $\boldsymbol{z}$ (the original $\boldsymbol{\pi}$). Data points $\boldsymbol{x}_!, \dots, \boldsymbol{x}_N$.
Output: The estimated parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$.
 1: Initialize the $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K$ and $\boldsymbol{\Sigma}_1, \dots, \boldsymbol{\Sigma}_K$.
 2: repeat
 3:     for $n \in \{1 \rightarrow N\}$ do
 4:         $i = 0$;
 5:         $q = 0$;
 6:         for $k \in \{1 \rightarrow K\}$ do
 7:             $r_{nk} = 0$;
 8:             $p_{nk} = \pi_k \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$;
 9:             if $p_{nk} > q$ then
10:                 $q = p_{nk}$;
11:                 $i = k$;
12:             end if
13:         end for
14:         $r_{ri} = 1$;
15:     end for
16:     for $k \in \{1 \rightarrow K\}$ do
17:         $N_k = 0$;
18:         for $n \in \{1, \dots, N\}$ do
19:             if $r_{nk} == 1$ then
```

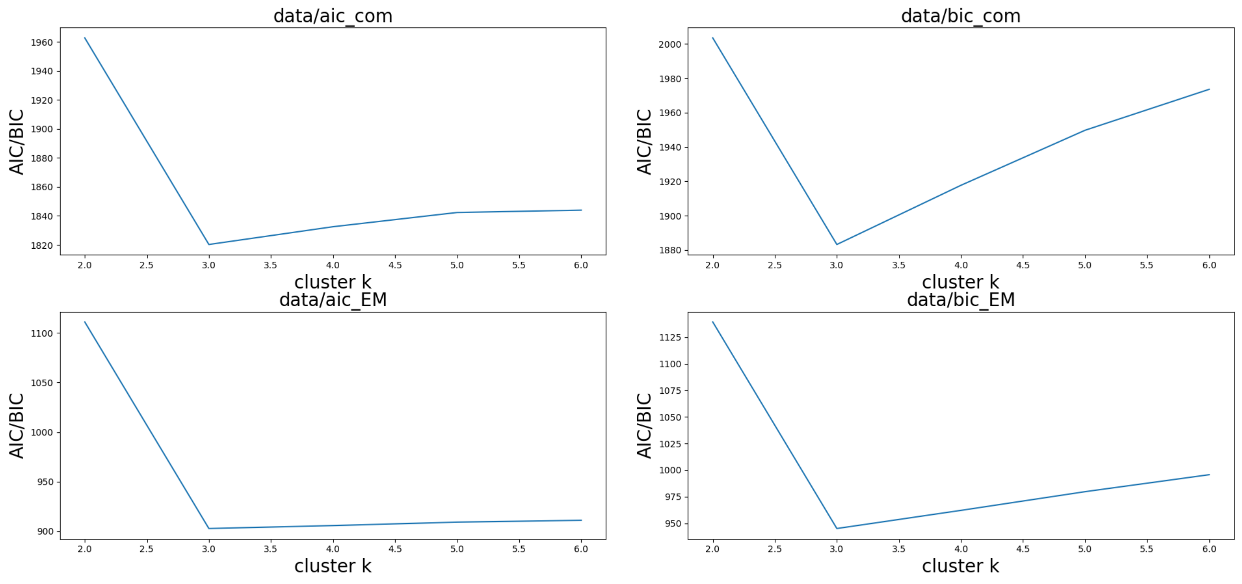
```
                $N_k = N_k + 1$;
            end if
        end for
        $\pi_k = N_k/N$;
        $\boldsymbol{\alpha} = 0$;
        for $n \in \{1, \ldots, N\}$ do
            if $r_{nk} == 1$ then
                $\boldsymbol{\alpha} = \boldsymbol{\alpha} + \boldsymbol{x}_n$;
            end if
        end for
        $\boldsymbol{\mu}_k = \alpha/N_k$;
        $\boldsymbol{\Phi} = \boldsymbol{O}$;
        for $n \in \{1, \ldots, N\}$ do
            if $r_{nk} == 1$ then
                $\boldsymbol{\Phi} = \boldsymbol{\Phi} + (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T$;
            end if
        end for
        $\boldsymbol{\Sigma}_k = \boldsymbol{\Phi}/N_k$;
    end for
until $\boldsymbol{\mu}$ *and* $\boldsymbol{\Sigma}$ *converge*
```

B Proposed RP K-mean algorithm

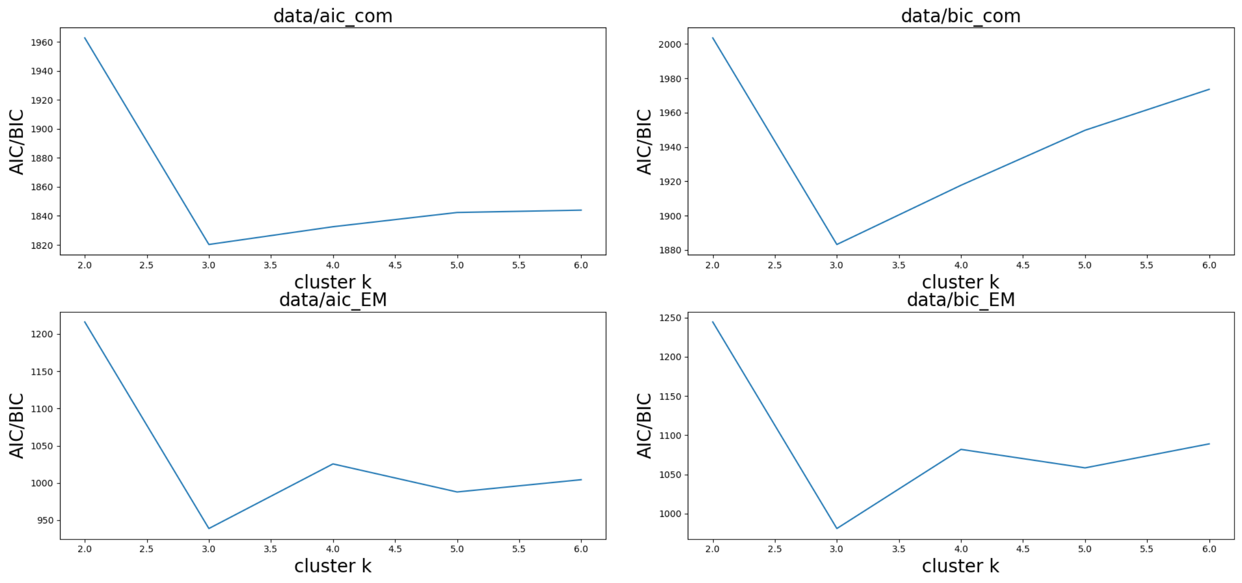
Algorithm 2 Proposed RP K-mean algorithm

```
Input: Data points $\boldsymbol{x}_!, \ldots, \boldsymbol{x}_N$.
Output: The estimated parameters $\boldsymbol{\mu}$.
Initialize the $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$.
$t = 1$
repeat
    for $n \in \{1 \to N\}$ do
        $i = 0$;
        $j = 0$;
        $q_1 = 0$;
        $q_2 = 0$
        for $k \in \{1 \to K\}$ do
            $r_{nk} = 0$;
            $d_{nk} =$;
            if $p_{nk} > q_1$ then
                $q_1 = p_{nk}$;
                $i = k$;
            else if $p_{nk} > q_2$ then
                $q_2 = p_{nk}$;
```

```
17:             $j = k$;
18:          end if
19:       end for
20:       $r_{ni} = 1$;
21:       $r_{nj} = -\eta/t$;
22:    end for
23:    for $k \in \{1 \to K\}$ do
24:       $N_k = 0$;
25:       for $n \in \{1, \dots, N\}$ do
26:          $N_k = N_k + r_{nk}$;
27:       end for
28:       $\boldsymbol{\alpha} = 0$;
29:       for $n \in \{1, \dots, N\}$ do
30:          if $r_{nk} == 1$ then
31:             $\boldsymbol{\alpha} = \boldsymbol{\alpha} + r_{nk}\boldsymbol{x}_n$;
32:          end if
33:       end for
34:       $\boldsymbol{\mu}_k = \boldsymbol{\alpha}/N_k$;
35:    end for
36:    for $j \in \{1 \to K\}$ do
37:       for $k \in \{1 \to K\}$ do
38:          for $n \in \{1, \dots, N\}$ do
39:             if $r_{nk} < 0$ then
40:                $\boldsymbol{\mu}_j = \boldsymbol{\mu}_j + r_{nk}(\boldsymbol{\mu}_j - \boldsymbol{\mu}_k)$;
41:             end if
42:          end for
43:       end for
44:    end for
45:    $t = t + 1$;
46: until $\boldsymbol{\mu}$ $converge$
```



(a) Initialize the clusters with knowledge on real distribution



(b) Random initialize the clusters

Figure 4: AIC and BIC criterion of GMM and EM.