Homework 1

CS420-Machine learning, Shikui Tu, Spring 2018

* Name:Zhiwen Qiang Student ID:515030910367 Email: qlightman@163.com

1 k–mean vs GMM

Give a variant of k–mean algorithm somewhat between the original k–mean and Expectation-Maximization (EM) for Gaussian Mixture Models (GMM). Please specify the computational details of the formulas. Pseudo-codes of the algorithm would be great.
Discuss the advantages or limitations of your algorithm.

1.1 Computational Details

FCM is a kind of fuzzy clustering algorithm, which allows data to belong to two or more clusters. It is based on minimization J_m, where J_m refer to:

$$J_m = \sum_{i=1}^{N}\sum_{j=1}^{C} u_{ij}^m ||x_i - c_j||^2, \qquad 1 \le m \le \infty \tag{1.1}$$

m is a real number greater than 1, u_{ij} is the degree of x_i in cluster j, x_i is the ith of d-dimensional data, c_j is the d-dimension center of the cluster.

The fuzzy operation is carried out through an iterative optimization of the objective function shown in equation 1.1, u_{ij} and c_j is defined in equation 1.2, 1.3.

$$u_{ij} = \frac{1}{\sum_{k=1}^{C}(\frac{||x_i - c_j||}{||x_i - c_k||})^{\frac{2}{m-1}}} \tag{1.2}$$

$$c_j = \frac{\sum_{i=1}^{N} u_{ij}^m x_i}{\sum_{i=1}^{N} u_{ij}^m} \tag{1.3}$$

The iteration won't stop until $max(|u_{ij}^{k+1} - u_{ij}^{(k)}|) < \varepsilon$, where ε is a termination criterion range form 0 to 1.

1.2 Pseudo-codes

Algorithm 1: FCM algorithm

input : A batch of data points.
output: Dataset being divided into several groups.

1 Initialize $U = [u_{ij}]$ matirx, U^0
2 **while** $||U^{k+1} - U^k|| >= \varepsilon$ **do**
3 At k-step: calculate the centers vectors $C^k = [c_j]$ with U^k
4 Update U^k,U^{k+1}

1.3 Advantages

- In overlapped data set, the result is comparatively better than k-means algorithm.
- Unlike k-means algorithm, the data point in FCM may belong to one or more clusters, which suits well in a lot more situations.

1.4 Limitations

- The number of iterations are usually large, thus requires more time to compute than k-means algorithm.
- This algorithm is sensitive to isolated points.

2 k–mean vs CL

Compare the k–mean algorithm with competitive learning (CL) algorithm. Could you apply the idea of Rival Penalized Competitive Learning (RPCL) to k-mean so that the number of clusters is automatically determined? If so, give the details of your algorithm and then implement it on a three-cluster dataset generated by yourself. If not, state the reasons.

2.1 Comparision

- Competitive learning algorithm is an adaptive way of learning. To be specific, in clustering problems, competitive learning algorithm usually view problem in an adaptive way, that is: data points come one by one. While in k-means algorithm, it usually deals with a batch of data points.
- Competitive learning algorithm is a form of unsupervised learning in artificial neural networks, which is a more general type of algorithms, it includes RPCL, FSCL, etc. While k-means algorithm is more specific and only deals with the clustering problems.
- K-means algorithm use a 'winner takes all' strategy, which may perform badly when starting with 'bad initializations'. On the other hand, FSCL algorithm solve the problems by penalizing the frequent winners. Also, k-means cannot determine the number of clusters, while RPCL can automatically determine the k value by penalizing the 'rivel'. So in a word, CL algorithm is more general and can deal with many situations where the k-means cannot handle.

2.2 Solution

I think it is possible to apply the RPCL to k-means so that the number of clusters is automatically determined. The way to locate the extra center is by

examing the distances between each centers. I assume that the extra center is cutting apart some of the points in other center, so after each time the center of each cluster is relocated, we need to select the two centers where their distance is the shortest. Then kick the center where its group is smaller than the other point away to some ratio. The pseudo-code of this algorithm is defined below:

Algorithm 2: RPCL-k-means algorithm

input : An three-cluster dataset.
output: Dataset being divided into several groups.

Initialize m_i, $i = 1, \ldots, k$ **while** m_i *don't converge* **do**
  for *all* x *in* X **do**
    if($||x - m_i = min_j||x - m_j||)b_i \leftarrow 1$
    else $b_i \leftarrow 0$
  for *all* m_i, $i = 1, \ldots, k$ **do**
    $m_i \leftarrow \sum b_i x / \sum b_i$
  for *all* m_i, $i = 1, \ldots, k$ **do**
    select the two points where their distance is the shortest.
    kick the point where its group is smaller than the other point
    away to some ratio.

2.3 Results

Figure 2.1 is the three-cluster dataset I generated, figure 2.2 is the results obtained from normal k-means algorithm. The given k value is 4 and it is obvious that the result isn't good. In figure 2.3 we present the results obtaind form the RPCL-k-means algorithm, we can see that point resprenting the extra center is being 'kicking' away, which proves the correctness of my algorithm. (Pictures of other interations can be found in the 'picture' folder)

3 model selection of GMM

Write a report on experimental comparisons on model selection performance between BIC,AIC and VBEM. Specifically, you need to randomly generate datasets based on GMM, by varying some factors, e.g., sample sizes, dimensionality, number of clusters, and so on.

3.1 BIC

Using the sklearn mixture module, we implement a example in Figure 3.1, first we randomly generate datasets based on GMM, in this case, the dataset consists of 3 components, then run the algorithm on the dataset for $k = 1 \ldots 6$, the result is shown in the bar chart, the rectangle with $*$ above it is the best match. It is

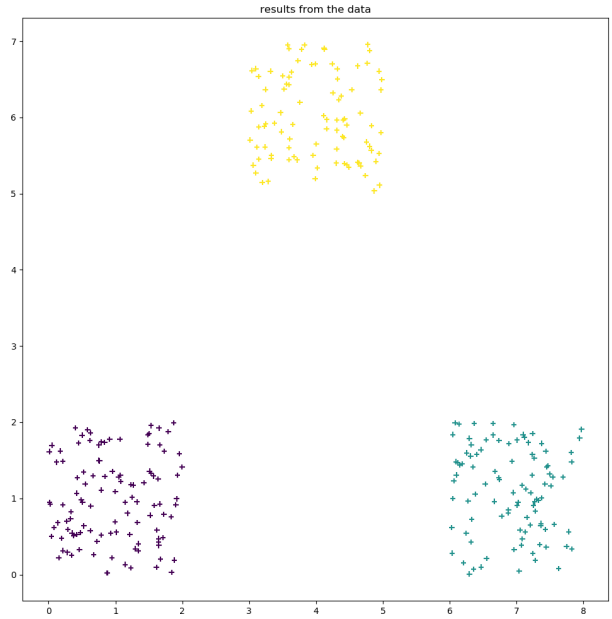

Figure 2.1: The three-cluster dataset I generated.

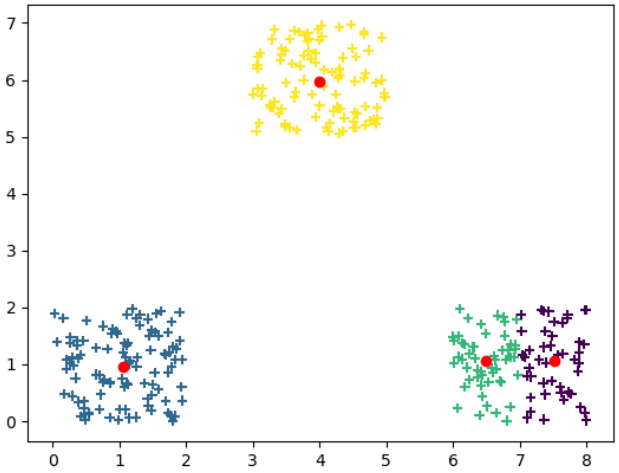

Figure 2.2: Results obtained from normal k-means algorithm.

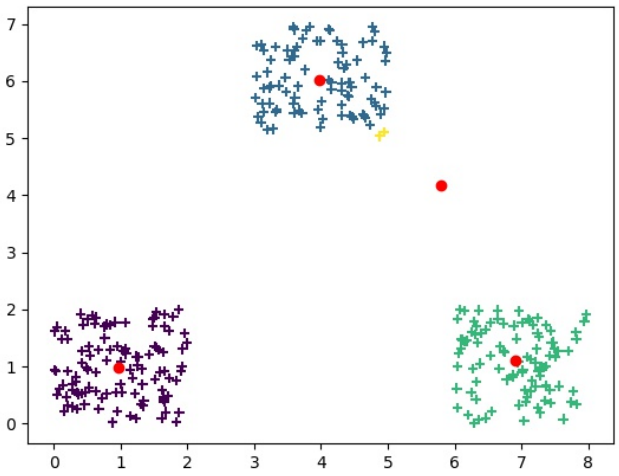

Figure 2.3: Results obtained from RPCL-k-means algorithm in 6 interations

the model with 3 components and full covariance, which is the true generative model.

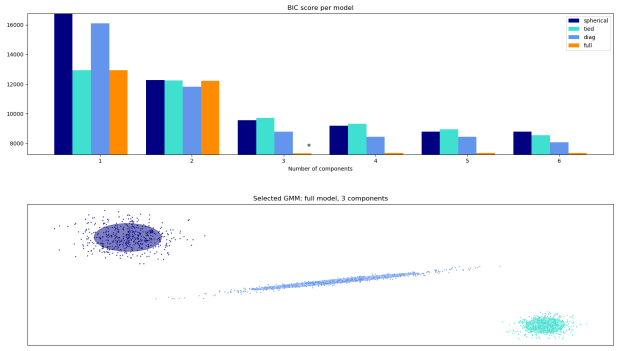

Figure 3.1: BIC score per model

3.2 AIC

Using the sklearn mixture module, we implement a example in Figure 3.2, first we randomly generate datasets based on GMM, in this case, the dataset consists of 3 components, then run the algorithm on the dataset for $k = 1 \ldots 6$, the result is shown in the bar chart, the rectangle with $*$ above it is the best match. It is the model with 4 components and full covariance, which is the true generative model.

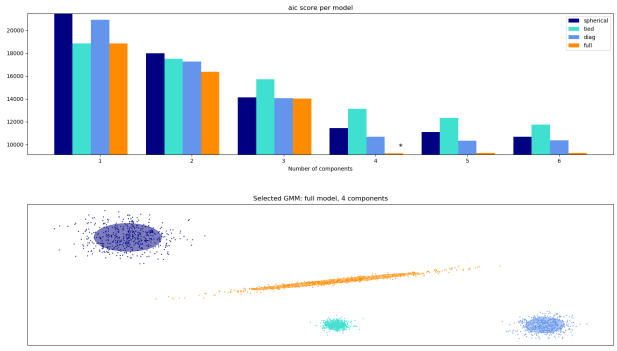

Figure 3.2: AIC score per model

3.3 VBEM

In this part, I implemented the sklearn mixture.BayesianGaussianMixture module. This variational algorithm needs more hyper- parameters than EM algorithm. The most important of these being the concentration parameter weight_concentration_prior, which has a strong impact on the effective number of active components obtained. I use the weight parameter ranging from 0.01 to 1000. Another important parameter is weight_concentration_prior_type, which consists of 'dirichlet_distribution' type and 'dirichlet_process' type.

Figure 3.3, 3.4, 3.5, 3.6 are some results obtained from changing the two parameters mentioned above, here we can notice that large values for the concentration weight prior lead to more uniform weights when the type of prior is 'dirichlet_distribution' while this is not necessarily the case for the 'dirichlet_process' type. Overall this algorithm can select the optimal k automatically. (Pictures of other parameters can be found in the 'picture' folder)

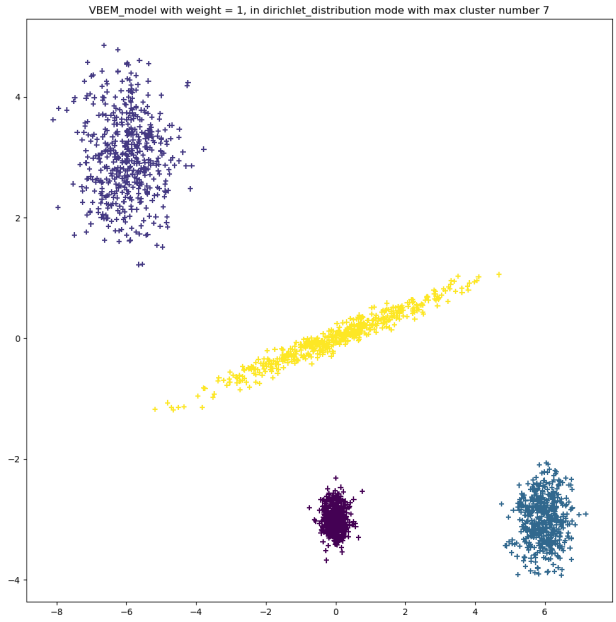

Figure 3.3: Dirichlet distribution with 1 weight

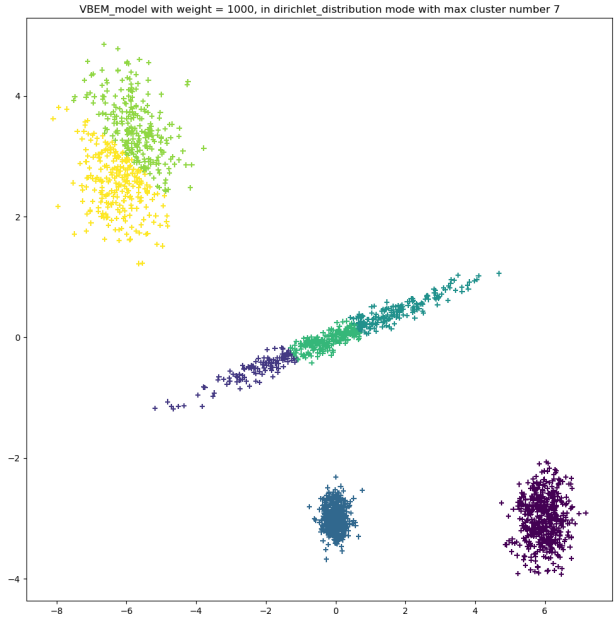

Figure 3.4: Dirichlet distribution with 1000 weight

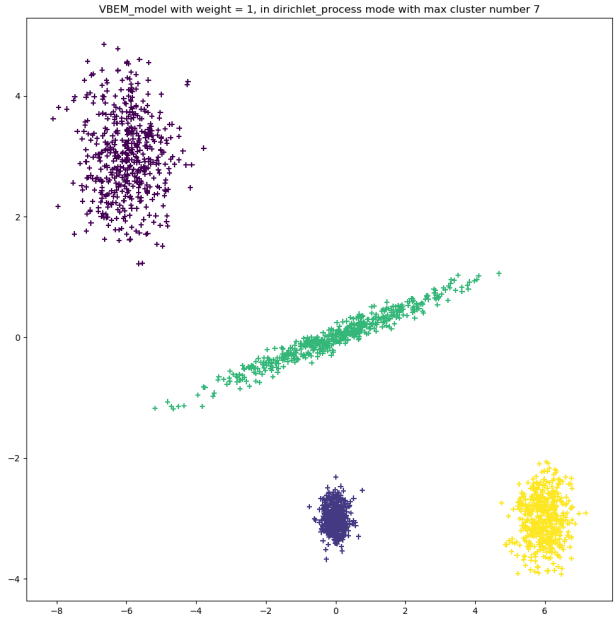

Figure 3.5: Dirichlet process with 1 weight

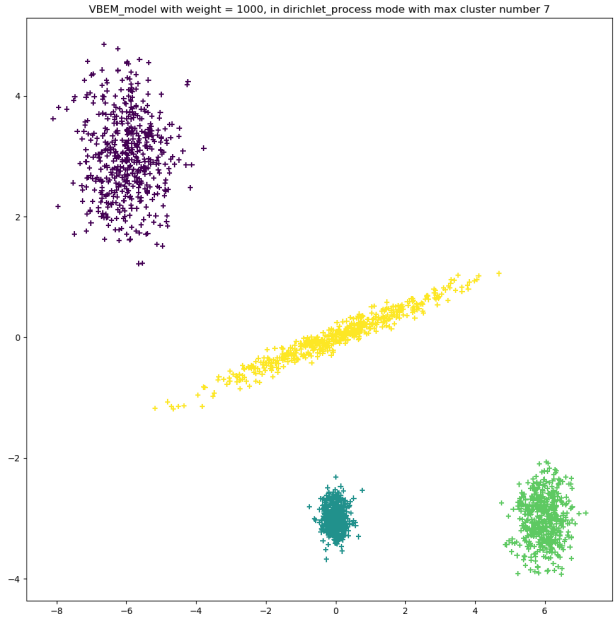

Figure 3.6: Dirichlet process with 1000 weight