Yuheng Zhi
515021910480
zyh1996@sjtu.edu.cn

Homework Assignment 1
Machine Learning, 18 Spring

2018-04-12

1 k-means 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. **solution**

1.1 Analysis

First, we need to observe the common points and differences of *k-means* and *EM for GMM*.
From the aspect of the model assumption, k-means is a special case of EM for GMM. They are both Gaussian Mixtures, but k-means give some constraints on the covariance matrices of GM, i.e., all covariance matrices Σ share the same value $\Sigma = \sigma\mathbf{I}$
From the aspect of parameter estimation algorithm itself, k-means can be regarded as a hard cut version of EM. In the E-step, k-means doesn't bother to calculate the actual posterior probability for each sample-cluster pair, but simply assigns all the *responsibility* of each sample to the nearest cluster center.
So the idea of this problem is to choose somewhere between the k-means and EM for GMM, either from the aspect of model assumption or algorithm. What I do is to give a loose the model assumption of k-means a little and use the EM algorithm for parameter estimation, i.e., not hard cut. While k-means use the same variance for all the clusters, I allow clusters to have different degree of variance, but they must still be diagonal and all the diagonal elements within each of them must be the same. Mathematically,

$$\Sigma_j = \sigma_j \mathbf{I}, \text{ for } j = 1 \; to \; K. \tag{1}$$

I call this distribution ***Spherical Gaussian Mixture (SGM)***

1.2 Algorithm

Algorithm 1 EM algorithm for SGM.

Require: Data points X, number of clusters K.
Ensure: $\mu_j,\ \sigma_j$ for $j = 1 \; to \; K$
  repeat
    for $i = 1$ to n **do**
      for $j = 1$ to K **do**

$$\gamma_{ij}^t \leftarrow \frac{\mathcal{N}(x_i|\mu_j^t, {\sigma_j^t}^2 I)}{\sum_{k=1}^{K} \mathcal{N}(x_i|\mu_j^t, {\sigma_j^t}^2 I)}$$

      end for
    end for
    for $j = 1$ to K **do**

$$\mu_j^{t+1} \leftarrow \frac{\sum_{i=1}^{n} \gamma_{ij}^t x_i}{\sum_{i=1} n\gamma_{ij}^t}$$
$$\sigma_j^{t+1\ 2} \leftarrow \frac{\sum_{i=1}^{n} \gamma_{ij}^t (x_i - \mu_j^t)^T (x_i - \mu_j^t)}{\sum_{i=1}^{n} \gamma_{ij}^t}$$

    end for
  until Convergence

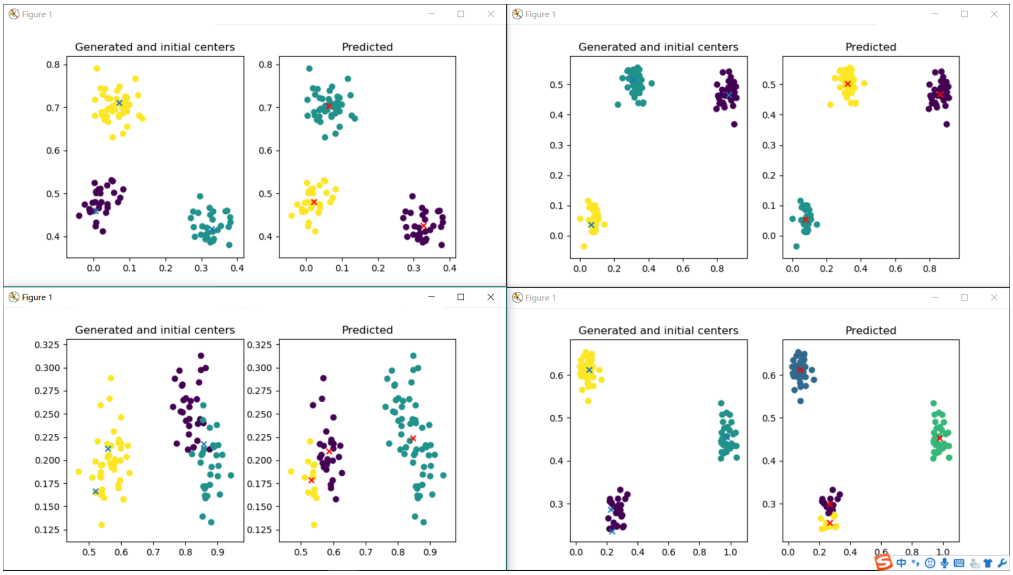

Abbildung 1: The result of k-means which employs RPCL as the initialization method.

2 k-means vs CL

k-means and CL are similar in: (a) they all modify the cluster centers using numeric average and (b) they give equal importance to all the directions. They differ by: CL updates its parameters on seeing every single data point. RPCL is a special case of CL which gives penalty to the second winner in order to determine the number of clusters automatically.
I enhanced k-means employing RPCL as its **initialization**: initialize RPCL with **MAX_K** uniform random points; for every data point, it will **pull** the nearest center a bit and **push** the second nearest center a slight bit. I run this initialization for 20 epochs and use the centers left for k-means.
I use **MAX_K** $= 10$, **learning_rate** $= 0.1$, $\gamma = 0.05$ for RPCL and visualization is in Fig. 2

3 model selection of GMM

Here I present some visual experimental results among aic selection, bic selection and VBEM and summarize out some points.

Experiment Settings The data is generated by $K = 5$ components using random covariance matrix $\Sigma = \sigma U(0, 1, shape = (2, 2))$ with $\sigma \in \{0.1, 0.001\}$. I test AIC and BIC using a range of **n_components** $= 1...2 \times K$.

Model complexity The model complexity under these experiments looks like

$$AIC >> VBEM > BIC. \tag{2}$$

It seems that all models converge to pretty good points when the clusters are quite separate from each other. But AIC seems to lack the ability to really reject unnecessary centers - it tends to keep them all.
When the clusters become closer, the result of AIC becomes a disaster - various points remain on the board. At the same time, BIC and VBEM show similar reasonable abilities of selecting the most effective components.
The difference of BIC and VBEM really shows up when the number of points get higher and clusters are close to each other, even overlapped, as in Fig. 3. BIC tends to be quite aggressive about merging clusters. I should safely summarize that within this range of parameters and in comparatively low-dimension space, BIC tens to produce the most appropriate (or visually reasonable) results. Its decisions look like what a person would arrive at. VBEM comes second. It is less robust than BIC but provides more flexibility under some circumstances. AIC is not what I'm looking for. At least, I didn't find a best situation/range o parameters it performs well.

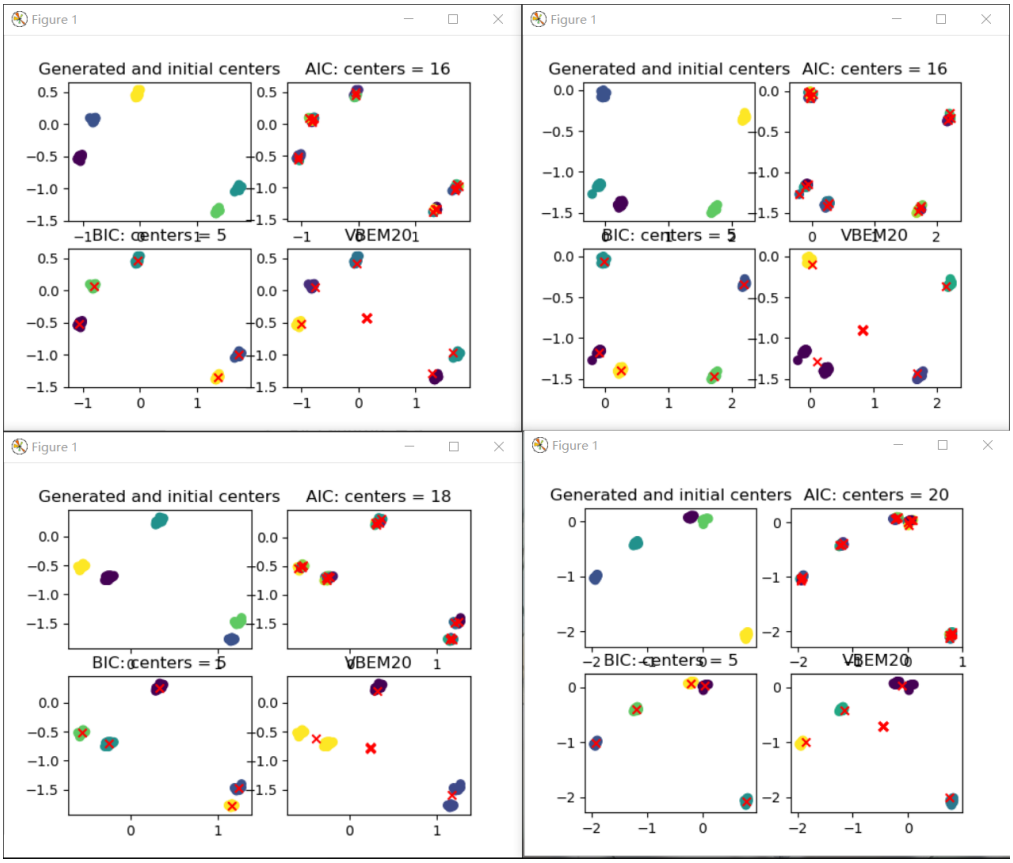

Abbildung 2: 100 points with cov ratio = 0.001

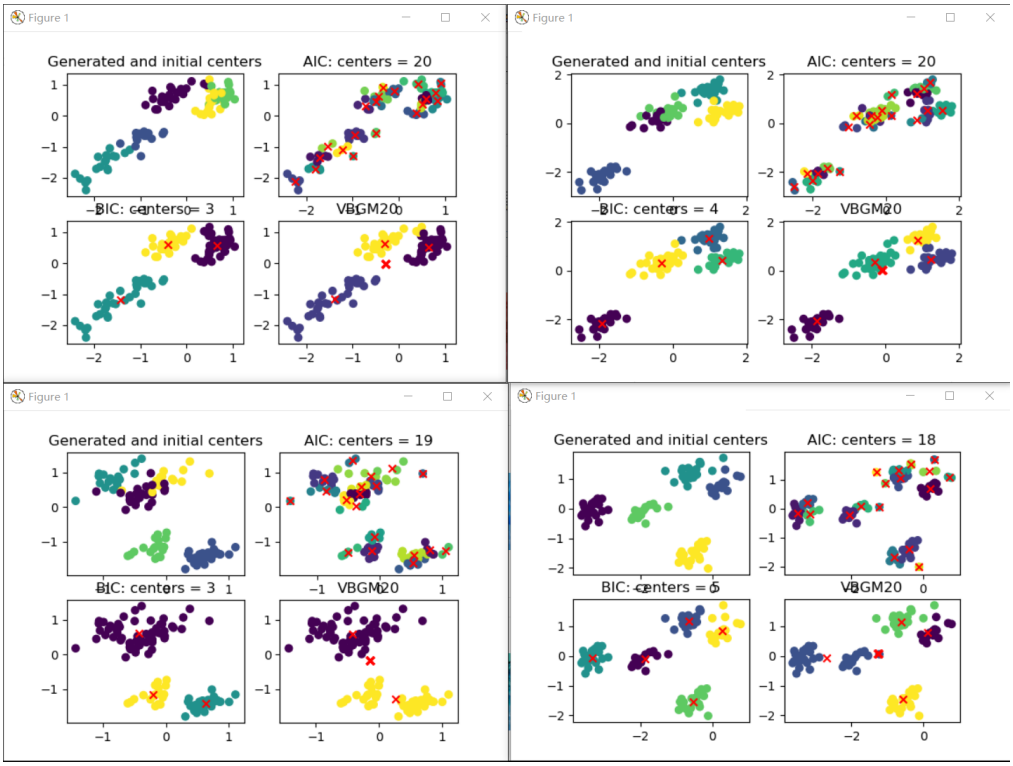

Abbildung 3: 100 points with cov ratio = 0.1

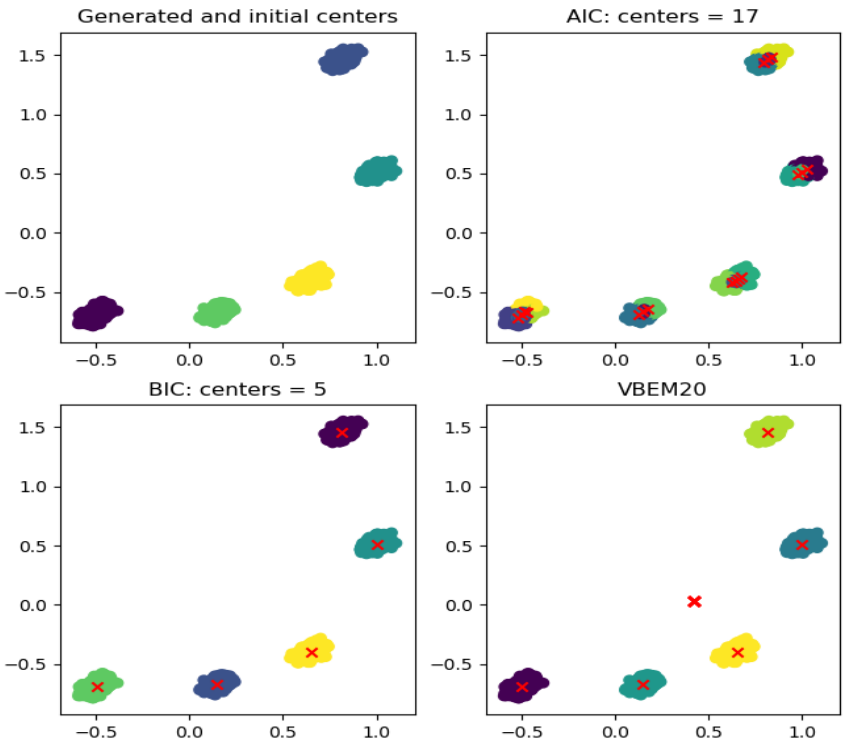

Abbildung 4: 10000 points with cov ratio = 0.001

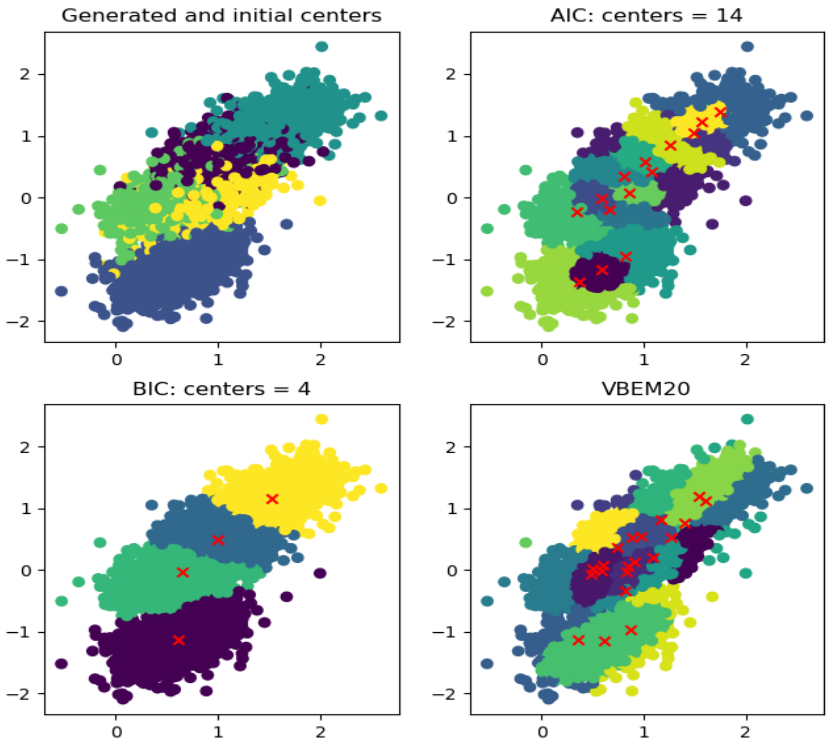

Abbildung 5: 10000 points with cov ratio = 0.1