Report on Homework 1

CS420, Machine Learning, Shikui Tu, Autumn 2017

Zelin Ye 515030910468

1 k-means vs GMM

Basically, k-means employs One-in-K assignment while GMM utilizes soft assignment. It turns out that k-means has bad robust. Therefore, I tend to introduce some of the ideas in GMM into k-means to form a new variant of k-means.

There are two main differences in my variant of k-means algorithm compared with the original:

- I introduce parts of the soft assignment into k-means to make it more robust.
- I affiliate a hyper parameter $Thres$ into original k-means. When the posteriori probablity is large than $Thres$, it can be retained, or it would be 0.

The pseudo-codes of my algorithm can refer to Alg. 1.

Algorithm 1: Improvements of k-means

input : The number of clusters K

output: $\pi_k, \mu_k, \Sigma_k, (k = 1, 2, ..., K)$

Initialize the means μ_k, covariances Σ_k, mixing coefficients π_k and threshold $Thres$.

Evaluating the initial value of the log likelihood.

while *the convergence criterion of parameters or log likelihood is not satisfied* **do**

E step. Evaluate the responsibilities using the current parameter values:

$$\omega \leftarrow \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)} \quad , \quad \gamma(z_{nk}) \leftarrow \begin{cases} \omega & \omega > Thres \\ 0 & \omega \leq Thres \end{cases}$$

$$z_n \leftarrow \frac{e^{z_{n_i}}}{\sum_{j=1}^{K} e^{z_{n_j}}}$$

M step. Re-estimate the parameters using the current responsibilities:

$$N_k \leftarrow \sum_{n=1}^{N} \gamma(z_{nk}) \quad , \quad \mu_k^{new} \leftarrow \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) x_n$$

$$\Sigma_k^{new} \leftarrow \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(x_n - \mu_k^{new})(x_n - \mu_k^{new})^T$$

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

Evaluate the log likelihood:

$$\ln p(X|\mu, \Sigma, \pi) \leftarrow \sum_{k=1}^{K} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k) \right\}.$$

return $\pi_k, \mu_k, \Sigma_k, (k = 1, 2, ..., K)$;

With the introduction of probability, my algorithm is somewhat similar to EM algorithm and would be more robust than k-means. Besides, the $Thres$ would eliminate the negligible posteriori probabilities and improve accuracy under the premise of robustness.

Generally speaking, more hyper parameters would result in more tricks, my algorithm is no exception. The choice of $Thres$ may greatly affect the result.

Additionally, my algorithm also encounters the initialization and unknown K problems. But I think such problems do not contradict with my core ideas, and could be resolved by introducing more components (e.g. the idea of RPCL).

2 k-means vs CL

2.1 Comparison between k-means and CL

- **Similarities**

  1. They can not estimate the total number of clusters from data.
  2. The effects of these two algorithms highly rely on initialization.
  3. Both of them have huge potential for expansion to improve their robustness.
  4. They consider data in distance framework instead of probability.

- **Differences**

  1. k-means is a batch learning algorithm while CL belongs to adaptive learning.
  2. CL has a hyper parameter η to adjust, making the algorithm a little tricky. But k-means has no hyper parameter.
  3. k-means would take more time to converge when it comes to large data size. CL avoids this issue by updating the center by one data point each time.
  4. Comparing to k-means, CL requires less computing resource (CPU, memory and etc).
  5. The process of k-means is more simple and succinct than CL.

2.2 Application of RPCL in k-means

Since k-means can not estimate the total number of clusters from data while RPCL can make extra centers far away to control the number of clusters, I tend to apply the idea of RPCL to **the initialization of k-means**.

Suppose the initial number of centers is m and the corresponding weight vectors are $v_i(i = 1, ..., m)$. For each center, its output could be defined as:

$$u_i \leftarrow \begin{cases} 1 & \text{if } u_i \text{ is the winner} \\ -1 & \text{if } u_i \text{ is the rival} \\ 0 & \text{otherwise} \end{cases}, \tag{1}$$

To make the winner approach the center of target cluster faster and keep rival away from it more efficiently, I take the geometric distribution of the data points into consideration. Therefore, a density function for each data point is defined as:

$$\rho(x_j) = \exp[-\sum_{i=1}^{N} \frac{d(x_i, x_j)}{\sum_{i=1}^{N} d(x_1, x_j)}], j = 1, 2, ..., N \tag{2}$$

where d denotes the distance function.

Then the adjustment for each weight vector v_i can be conducted according to u_i and $\rho(x_j)$:

$$\Delta v_i = \begin{cases} \alpha_i \rho(x_j)(x_j - v_i) & u_i = 1 \\ \beta_i \rho(x_j)(x_j - v_i) & u_i = -1 \\ 0 & \text{otherwise} \end{cases}, \tag{3}$$

where α and β are hyper parameters.

Once the weight vectors converge, I will eliminate negligible ones according to a threshold T. Then the number of the remaining centers should be the clusters number K, and these centers would locate in 'good' initial positions.

The pseudo-codes of my algorithm can refer to Alg. 2.

Algorithm 2: RPCL in k-means

input : The dataset
output: The initial centers and corresponding weight vectors

Initialize the number of centers m and weight vectors v_i, threshold T, density $\rho(x_j)$.
while *the convergence criterion is not satisfied* **do**
    for *i = 0 to n* **do**
        Calculating u_i according to Eqn. 1.
        Adjust the weight vectors of each centers via Eqn. 3
Assign each data point to its closest center.
for *i = 0 to m* **do**

$$\eta \leftarrow \frac{N_i}{N},$$

    where N_i denotes the number of data points in ith center, N is the number of data points in the dataset.
    if $\eta < T$ **then**
        Remove center v_i.
return v;

I test this application in a three-cluster dataset generated by myself, and the results of initial centers choice can refer to Fig. **??**.

3 Model Selection of GMM

3.1 Different Sample Sizes

3.2 Different Number of Clusters

References

[1] S. P. Bhavnani, D. Muñoz, and A. Bagai, "Data science in healthcare: implications for early career investigators," *Circulation: Cardiovascular Quality and Outcomes*, vol. 9, no. 6, pp. 683–687, 2016.

[2] Y. Liang and A. Kelemen, "Big data science and its applications in health and medical research: Challenges and opportunities," *Austin Journal of Biometrics & Biostatistics*, vol. 7, no. 3, 2016.

[3] S. O' Halloran, S. Maskey, G. McAllister, D. K. Park, and K. Chen, "Data science and political economy: application to financial regulatory structure," *RSF*, 2016.

[4] W. Xiao-fan, "Data science and social network: Big data, small world," *Science and Society*, vol. 1, p. 003, 2014.

[5] C. Vande Kerckhove, *Data science for modeling opinion dynamics on social media*. PhD thesis, UCL-Université Catholique de Louvain, 2017.

[6] "Numpy." http://www.numpy.org/.

[7] "Pandas." http://pandas.pydata.org/.

[8] "matplotlib." https://matplotlib.org/.

[9] "scikit-learn." http://scikit-learn.org/stable/.