
第一次作业

李子龙

上海交通大学

计算机科学与工程系

logcreative@outlook.com

1 k-mean 算法

证明. 对两步分别证明。

(a) **E 步** 如果将每一个点 x_n 赋予类 k'_n 使得其相对于其他所有的类最近, 即

$$\|x_n - \mu_{k'_n}\| = \min_k \|x_n - \mu_k\|$$

就意味着它将比上一次赋予的类 k_n 在现在的这种聚类分布下距离不会增加:

$$\|x_n - \mu_{k'_n}\| \leq \|x_n - \mu_{k_n}\|$$

那么在对这个点求和的时候, 根据指示函数 r_{nk} 的定义, 该项也不会增加:

$$j_n = \sum_{k=1}^K r'_{nk} \|x_n - \mu_k\|^2 \leq \sum_{k=1}^K r_{nk} \|x_n - \mu_k\|^2$$

这里

$$r_{nk} = \begin{cases} 1, & x_n \text{ 属于类 } k; \\ 0, & \text{其他情况.} \end{cases}$$

那么损失函数也不会增加:

$$J(\mu_1, \dots, \mu_K) = \sum_{n=1}^N j_n \quad (1.1)$$

(b) **M 步** 记每个聚类 k 内的点损失函数贡献值为

$$j_k = \sum_{n=1}^N r_{nk} \|x_n - \mu_k\|^2 \quad (1.2)$$

求和可以交换, 损失函数改写为

$$J(\mu_1, \dots, \mu_K) = \sum_{k=1}^K j_k \quad (1.3)$$

将用引理 1 证明, 使用聚类内点的平均点作为新的聚类点将不会增加该项。

引理 1 (距离平方和最小). 当 μ 是所有数据点的均值时, 距离平方和

$$f = \sum_{t=1}^N \|\mu - x_t\|^2 \quad (1.4)$$

最小。

证明. 将公式 (1.4) 改写

$$\begin{aligned} f &= \sum_{t=1}^N \|\mu - x_t\|^2 \\ &= \sum_{t=1}^N (\mu - x_t)^T (\mu - x_t) \\ &= \sum_{t=1}^N (\mu^T \mu - 2x_t^T \mu + x_t^T x_t) \end{aligned}$$

对其求导,

$$\begin{aligned} \frac{\partial f}{\partial \mu} &= \sum_{t=1}^N (2\mu^T - 2x_t^T) = 0 \\ \mu &= \frac{1}{N} \sum_{t=1}^N x_t \end{aligned} \quad (1.5)$$

公式 (1.5) 表明当 μ 是所有数据点的均值时, 导数为 0, 距离平方和最小。 \square

由于对于每一类而言, 公式 (1.2) 都不会增加, 而类别指示函数 r_{nk} 不会改变, 所以损失函数 (1.3) 不会增加。

\square

2 k-mean 与 GMM 之间

解. 为了将 GMM 退化为 k-mean, 需要对 GMM 有三个方面的特殊化处理:

$$\pi_k = \frac{1}{K} \quad (2.1)$$

$$\Sigma = I \quad (2.2)$$

$$p(k|x_n) = \begin{cases} 1, & \text{如果 } k = \arg \max_k \mathcal{N}(x_n|\mu_k, \Sigma_k) \\ 0, & \text{其他情况.} \end{cases} \quad (2.3)$$

公式 (2.1) 是混合权重的归一, 公式 (2.2) 是协方差一致使得其只计算欧氏距离, 公式 (2.3) 是硬赋值只取可能性最大的那个聚类 k 。

为了得到其中一个中间变种, 这里我们只退化 (2.1), 使 GMM 中的 $\pi_k = \frac{1}{K}$, 就可以得到一个更加一般的带方差项软赋值的 k-mean 算法 1。此时的对数似然值定义为

$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \Sigma) = \sum_{n=1}^N \left[-\ln K + \ln \sum_{k=1}^K \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \Sigma_k) \right] \quad (2.4)$$

优点 这种算法可以很好地拓展 k-mean 算法，使其能够具有方差项（引入高斯分布），并且软赋值可以更好地考虑多个聚类。

缺点 这种算法无疑增加了一定的计算量。

Algorithm 1: 含方差软赋值的 k-mean 算法

Input: 数据点 $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$, 聚类数目 K

Output: 聚类的分类结果 $\mu_j, \Sigma_j, \forall j \in \mathbb{N} \cap [1, K]$

1 初始化均值矩阵 μ_k , 协方差矩阵 Σ_k , 根据公式 (2.4) 初始化对数似然值;

2 **repeat**

3 **for** $n \leftarrow 1$ to N **do**

4 **for** $k \leftarrow 1$ to K **do**

5

$$\gamma_{nk}^{(t)} \leftarrow \frac{\mathcal{N}(\mathbf{x}_n | \mu_k, \Sigma_k)}{\sum_{j=1}^K \mathcal{N}(\mathbf{x}_n | \mu_j, \Sigma_j)} \quad (2.5)$$

6 **for** $k \leftarrow 1$ to K **do**

7

$$\mu_k^{(t+1)} \leftarrow \frac{\sum_{n=1}^N \gamma_{nk}^{(t)} \mathbf{x}_n}{\sum_{n=1}^N \gamma_{nk}^{(t)}} \quad (2.6)$$

$$\Sigma_k^{(t+1)} \leftarrow \frac{\sum_{n=1}^N \gamma_{nk}^{(t)} (\mathbf{x}_n - \mu_k^{(t+1)}) (\mathbf{x}_n - \mu_k^{(t+1)})^T}{\sum_{n=1}^N \gamma_{nk}^{(t)}} \quad (2.7)$$

8 **until** 公式 (2.4) 中的值没有明显变化;

9 **return** μ, Σ

3 k-mean 与 CL

3.1 k-mean 与 CL 的比较

见表 1。以及，如果将 k-mean 按照 CL 的方法以在线版本的视角去看，可以看到两者在公式上的联系。

对于一个聚类 k 的前 N 个数据点，获得该聚类的均值点

$$\mu_k^{(N)} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n \quad (3.1)$$

表 1: k-mean 与 CL 的比较

	k-mean	CL
算法类型	批学习算法	在线学习算法
超参数	不需要	需要学习率 η
更新方法	每次需要遍历	每次只更新一个点
复杂度	高	相对低
聚类数目	无法决定	
初始化	收敛速度依赖初始化	

如果该聚类在下一轮后多了一个点，那么更新该聚类的均值点后

$$\begin{aligned}
 \mu_k^{(N+1)} &= \frac{1}{N+1} \sum_{n=1}^{N+1} x_n \\
 &= \frac{1}{N+1} (N\mu_k^{(N)} + x_{N+1}) \\
 &= \mu_k^{(N)} + \frac{1}{N+1} (x_{N+1} - \mu_k^{(N)})
 \end{aligned} \tag{3.2}$$

对比与 CL 的更新公式

$$\mu_k^{(N+1)} = \mu_k^{(N)} + \eta p_{k,n} (x_{N+1} - \mu_k^{(N)}) \tag{3.3}$$

这里

$$p_{k,n} = \begin{cases} 1, & \text{如果 } k = \arg \min_j \|x_n - \mu_k\|^2 \\ 0, & \text{其他情况.} \end{cases} \tag{3.4}$$

可见公式 (3.2) 是公式 (3.3) 在 $\eta = \frac{1}{N+1}$ 和该点被选中时 $p_{k,n} = 1$ 的特殊情形。

3.2 RPCL 版本的 k-mean

RPCL 改变了公式 (3.4) 使其疏远竞争点

$$p_{k,n} = \begin{cases} 1, & \text{如果 } k = c = \arg \min_j \|x_n - \mu_k\|^2 \\ -\gamma, & \text{如果 } k = r = \arg \min_{j \neq c} \|x_n - \mu_k\|^2 \\ 0, & \text{其他情况.} \end{cases} \tag{3.5}$$

这里 γ 为可变参量，一般范围是 $[0.05, 0.1]$ 。

在第 3.1 节我们了解到 k-mean 是 CL 的一个特殊情形，为了将 RPCL 适配 k-mean，将公式 (3.5) 适配进去，得到算法 2。这个算法每次会进行批量运算，但会增加移开竞争对手的这个过程。

3.3 训练结果

生成数据的细节见第 4.1 节，设定生成参数样本总量 $N = 1000$ ，随机种子 20。

Algorithm 2: RPCL 版本的 k-mean 算法

Input: 数据点 $\mathbf{X} = \{x_n\}_{n=1}^N$, 最大聚类数目 K , 疏远参量 γ

Output: 聚类的分类结果 μ_j , $\forall j \in \mathbb{N} \cap [1, K]$

```
1 初始化均值  $\mu_k$ , 学习率  $\eta = 0.01K$ ;  
2 repeat  
3   for  $n \leftarrow 1$  to  $N$  do  
4     计算到每个聚类的距离  $\mathbf{E} = \{\|x_n - \mu_k\|\}_{k=1}^K$ ;  
5      $c \leftarrow \arg \min_k \mathbf{E}, r \leftarrow \arg \min_{k \neq c} \mathbf{E}$ ;  
6     for  $k \leftarrow 1$  to  $K$  do  
7       按照公式 (3.5) 计算  $p_{k,n}$ ;  
8   for  $k \leftarrow 1$  to  $K$  do  
9     if 聚类  $k$  不包含任何点 then  
10      本轮结束移除聚类  $k$ ;  
11    else  
12      
$$\mu_k^{(t)} = \frac{\sum_n p_{k,n} x_n}{\sum_n p_{k,n}} \quad \forall n : p_{k,n} = 1 \quad (3.6)$$

$$\mu_k^{(t+1)} = \mu_k^{(t)} + \eta \sum_n p_{k,n} (x_n - \mu_k^{(t)}) \quad \forall n : p_{k,n} < 0 \quad (3.7)$$
  
13    if 聚类数目有变动 then  
14      衰减学习率; /* 防止早期学习率过高 */  
15      按时间阶梯衰减学习率; /* 防止末期学习率过高 */  
16 until  $\mu_k$  没有明显变化;  
17 return  $\mu$ ;
```

训练代码见 [src/rpclk.py](#)。

如图 1(a), 对于 $K = 3$ 的等量聚类, 可以较好地分类。如图 1(b), 对于 $K = 12$ 的余量聚类, 较小的 γ 可能会导致不好的结果, 这种没有办法完全竞争的情形可以通过调高 γ 如图 1(c) 实现。

请注意, 由于 k-mean 和 RPCL 算法都依赖于初始化, 所以分类结果的好坏也会与初始化相关。RPCL 化后一定程度上可以减少聚类数目至期望值。

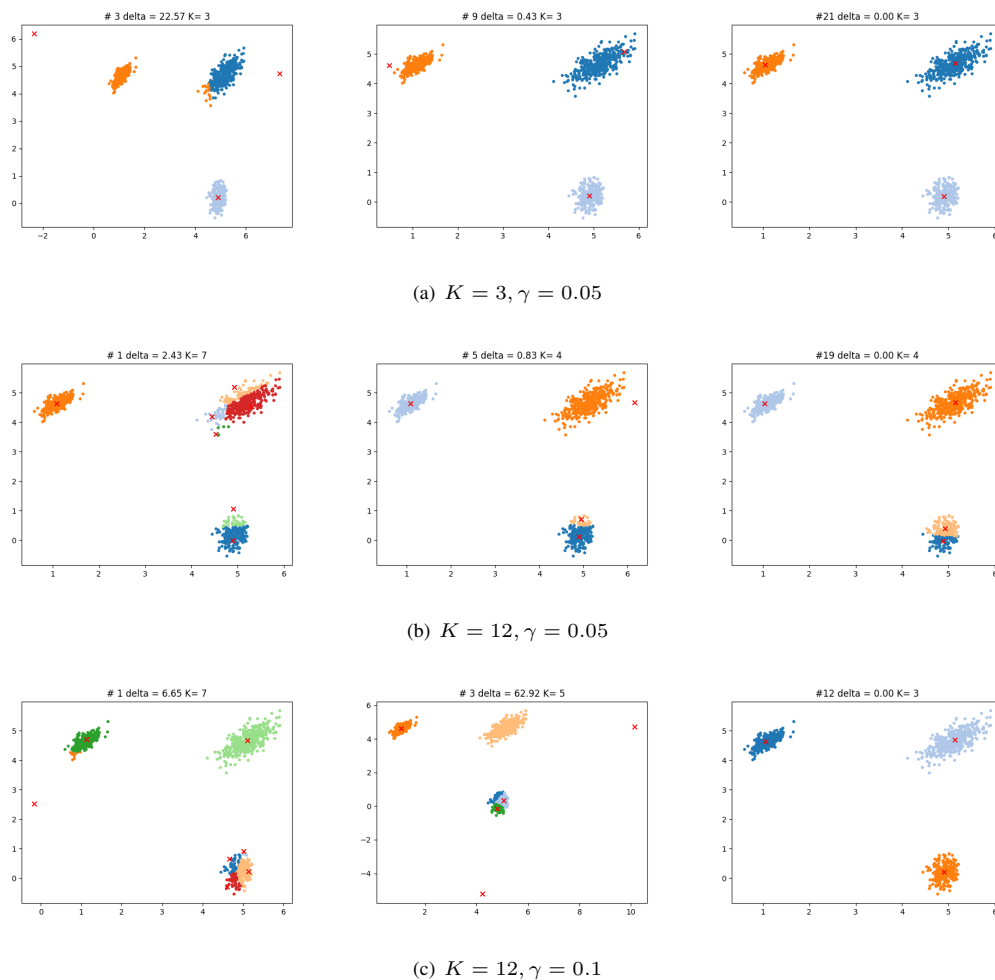


图 1: RPCL 版本的 k-mean 算法训练结果

4 GMM 模型选择

4.1 生成 GMM 数据

对于 L 维、 K 个子模型的高斯混合模型（GMM），其概率密度函数定义为

$$p(\mathbf{x}) = \sum_{k=1}^K w_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad (4.1)$$

其中

$$\sum_{k=1}^K w_k = 1 \quad (4.2)$$

算法 3 展示了通过随机化参量、多项分布获得每个子模型的样本数、多维高斯分布采样得到 GMM 的过程^[1]。生成数据代码见 [src/datagen.py](#)。

Algorithm 3: 生成 GMM 数据

- 1 如果指定随机种子，则初始化随机种子;
 - 2 随机生成权重 w_k ，归一化以满足公式 (4.2) 中的条件;
 - 3 借助 `numpy` 库生成随机的 K 个 $L \times (K + 1)$ 矩阵，计算协方差矩阵作为 GMM 数据的随机参量 Σ_k （这一步主要为了保证协方差矩阵的对称正定^[2]），对于均值 μ_k 直接随机坐标并扩大 $2K$ 倍以保持一定的距离;
 - 4 通过多项分布得到每个子模型的样本数 N_k ;
 - 5 对每个子模型通过多维高斯模型采样对应的样本数次数，合并得到全部的样本;
-

References

- [1] 派大西. 高斯混合模型(GMM): 样本生成与参数估计.py[EB/OL]. 2021. <https://zhuanlan.zhihu.com/p/411925257>.
- [2] NumPy Developers. `numpy.random.multivariate_normal`—numpy v1.23.dev0 manual [M/OL]. 2022. https://numpy.org/devdocs/reference/random/generated/numpy.random.multivariate_normal.html.