Section 5: K-means

MCB 112

10/07/2022

(Adapted from 2021 notes)

Outline

- Hard and soft K-means
- Mixture Gaussian and Negative Binomial fitting
- Example of 1D hard K-means and 1D mixture gaussian fitting
  - See Jupyter notebook (w05_section_jupyter_notebook.ipynb)

How do we categorize gene counts (i.e. high-dimensional data) into cell type clusters?

Count Matrix

Cell ID	Gene 1	Gene 2	…	Gene Z
1				
2				
…				
N				

Cell Type
?
?
?
…

Clustering through K-means/expectation maximization (EM)

Advantages

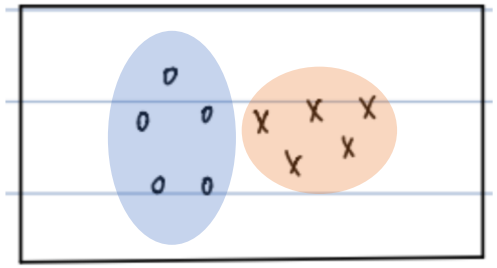
- Simple to implement on a large dataset
- Guarantee convergence
- Generalize to clusters of different shapes and sizes

Disadvantages

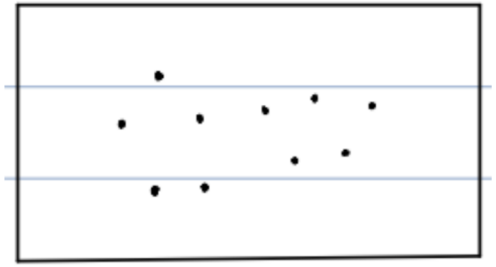
- Choice of K (elbow plot)
- Depend on initial centroid positions (multiple EM runs)
- Influenced by outliers (remove outliers first)

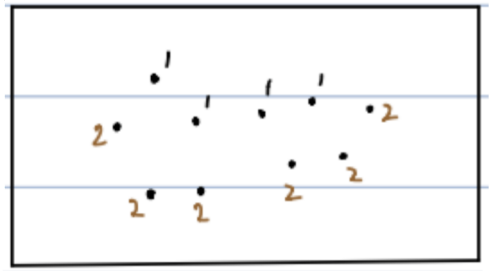
Simple example of K-means clustering

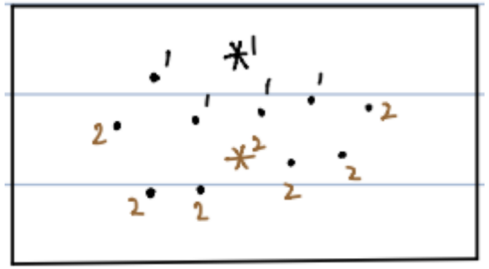
Ground Truth



Our Data





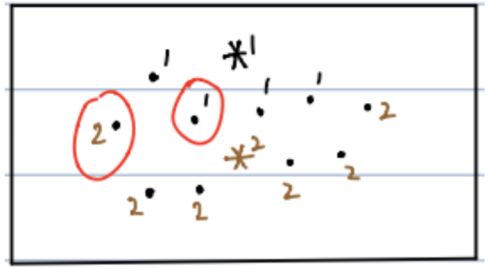


Step 1: Choose *K* and

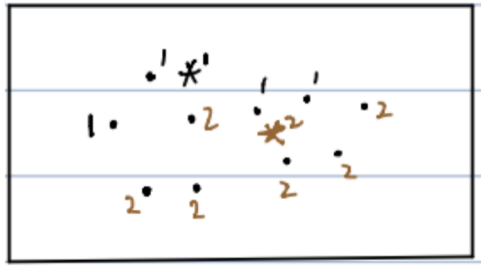
- Assign data randomly to clusters and calculate centroids (shown here)

OR

- Randomly sample centroids and assign data to closest centroids



Step 2: Calculate the distance d_{ik} between each data point *i* and each cluster *k*. Update cluster assignment by choosing the closest cluster.



Step 3: Update centroids and iterate until convergence.

Hard vs. Soft K-means

- Assume we have multi-dimensional data ***X***. For each X_i for i = 1, 2, …, N, it has Z dimensions (e.g. each cell i has gene expression counts of Z genes). We set K clusters.

	Hard K-means	**Soft K-means**
Step 1: Assign data randomly and calculate centroids, μ_{kz}	$\mu_{kz} = \frac{\sum_{i \in C_k} X_{iz}}{\lvert C_k \rvert}$	
Step 2: Update cluster assignment based on distances, d_{ik}, and update centroids, μ_{kz}	$d_{ik} = \sqrt{\sum_Z (X_{iz} - \mu_{kz})^2}$ $\mu_{kz} = \frac{\sum_{i \in C_k} X_{iz}}{\lvert C_k \rvert}$	$r_{ik} = \frac{\exp(-\beta d_{ik}^2)}{\sum_{k'} \exp(-\beta d_{ik'}^2)}$ where $\sum_k r_{ik} = 1$ for each i $\mu_{kz} = \frac{\sum_i r_{ik} X_{iz}}{\sum_i r_{ik}}$ (weighted mean of X_i's)
Step 3: Set a criteria for convergence (objective function)	$\min \sum_{k=1}^{K} \sum_{i=1}^{N} d_{ik}^2$ Check for convergence • Same cluster assignment • Same centroid position (set a threshold)	
Step 4: Run multiple times, choose the optimal objective function and corresponding cluster assignment		

Hard K-means
If a point is close to two clusters, we force it to be in one cluster

Soft K-means
Adjust "stiffness" β
- β → ∞, same as hard K-means
- β → 0, meaningless

What to do with an empty cluster?

- Option 1: reinitialize centroid randomly
- Option 2: assign the farthest point as the new centroid

Mixture model

- *Q* components (do not need to be the same distribution) with probability π_q for q = 1, 2, …, Q
- $\pi_q = P(q \mid \theta)$ where θ is the model parameter(s) and $\sum_q \pi_q = 1$
- $P(X_i, q \mid \theta) = P(X_i \mid q, \theta) \cdot P(q \mid \theta) = \pi_q \cdot P(X_i \mid q, \theta)$
- $P(X_i \mid \theta) = \sum_q P(X_i, q \mid \theta) = \sum_q \pi_q \cdot P(X_i \mid q, \theta)$
- $P(\text{cell type} \mid \text{data } i) = P(\ q \mid X_i, \theta\ )$ → distribution { Gaussian; Negative Binomial (RNA-seq)

Soft K-means with mixture model

	Soft K-means with mixture model
Step 1: Assign data randomly and calculate centroids, μ_q	Centroids have initial π_q
Step 2: Update cluster assignment based on $P(q \mid X_i, \theta)$ for each q	$P(q \mid X_i, \theta) = \frac{P(X_i, q \mid \theta)}{P(X_i \mid \theta)} = \frac{\pi_q \cdot P(X_i \mid q, \theta)}{\sum_{q'} \pi_{q'} \cdot P(X_i \mid q', \theta)}$ • X_i is most likely to be in q with largest $P(q \mid X_i, \theta)$
Step 3: Update μ_q, π_q	$\mu_q = \frac{\sum_i X_i \cdot P(q \mid X_i, \theta)}{\sum_i P(q \mid X_i, \theta)}$, $\pi_q = \frac{\sum_i P(q \mid X_i, \theta)}{N}$

Soft K-means with mixture model (high-dimensional)

Soft K-means with mixture model (Z-dimensional)

Step 1: Assign data randomly and calculate centroids, μ_{qz}

Centroids have initial π_q

Step 2: Update cluster assignment based on $P(q \mid X_i, \theta)$ for each q

$$P(q \mid X_i, \theta) = \frac{P(X_i, q \mid \theta)}{P(X_i \mid \theta)} = \frac{\pi_q \cdot \prod_z P(X_{iz} \mid q, \theta)}{\sum_{q'}(\pi_{q'} \cdot \prod_z P(X_{iz} \mid q', \theta))}$$

- X_i is most likely to be in q with largest $P(q \mid X_i, \theta)$

Step 3: Update μ_{qz}, π_q

$$\mu_{qz} = \frac{\sum_i X_{iz} \cdot P(q \mid X_i, \theta)}{\sum_i P(q \mid X_i, \theta)}, \qquad \pi_q = \frac{\sum_i P(q \mid X_i, \theta)}{N}$$

Soft K-means with mixture model

Soft K-means with mixture model

Step 4: Set a criteria for convergence (objective function)

$$\max P(X \mid \theta) \quad \text{or} \quad \min -\log P(X \mid \theta)$$

$$P(X \mid \theta) = \prod_i P(X_i \mid \theta) \qquad (i.i.d)$$

$$= \prod_i \left(\sum_q P(X_i, q \mid \theta) \right)$$

$$= \prod_i \sum_q \pi_q \cdot P(X_i \mid q, \theta)$$

$$-\log P(X \mid \theta) = -\log \left(\prod_i \sum_q \pi_q \cdot P(X_i \mid q, \theta) \right)$$

$$= -\sum_i \log \left(\sum_q \pi_q \cdot P(X_i \mid q, \theta) \right)$$

Depend on what distribution we model the data with

Step 5: Run multiple times, choose the optimal cluster assignment

Log-likelihood implementation

$$-\log P(X \mid \theta) = -\log\left(\prod_i \sum_q \pi_q \cdot P(X_i \mid q, \theta)\right)$$

$$= -\sum_i \log\left(\sum_q \pi_q \cdot P(X_i \mid q, \theta)\right)$$

$$\log\left[\pi_1 P(X_i \mid \theta_1) + \cdots + \pi_Q P(X_i \mid \theta_Q)\right]$$

$$= \log\left[e^{\log(\pi_1 P(X_i \mid \pi_1))} + \cdots + e^{\log(\pi_Q P(X_i \mid \theta_Q))}\right]$$

`scipy.special.logsumexp()`:

- Input: an array X
- Compute: $\log \sum_{x \in X} e^x$

$P(X_i \mid q, \theta)$

- Gaussian distribution $$P(X_i \mid \mu_q, \sigma_q) = \frac{1}{\sqrt{2\pi}\sigma_q} \exp\left(-\frac{(X_i - \mu_q)^2}{2\sigma_q^2}\right)$$

- Negative binomial distribution (RNA-seq count data)
  - Models the number of failures in a sequence of i.i.d. Bernoulli trials (with success rate *p*) before *n* successes occur
  - Let *X* be the number of failures, $X \sim \mathrm{NB}(n, p)$

$$P(X = k \mid n, p) = \binom{k + n - 1}{n - 1} (1 - p)^k p^n$$

  - Each cell X_i has the probability in cluster *q* with NB(*n*, *p*) where

$$n = \frac{1}{\phi}, \qquad p = \frac{1}{1 + \mu_q \cdot \phi}$$

models dispersion — updated in each iteration

```
import scipy.stats
scipy.stats.nbinom.logpmf(x, n, p)
```