Adapted from 2020 notes.

Goals:

1. compare hard and soft kmeans (mixture Gaussian and negative binomial fitting)

2. example of 1D hard kmeans and 1D mixture Gaussian fitting (Jupyter notebook)

Q: How do we categorize/visualize gene counts (high-dimensional data) into cell type clusters?

Example data:

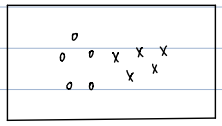
cell id	gene 1 counts	gene 2 counts	. . . . .	gene N counts	→ cell type ?
1					
2					
3					
↓					

A: Clustering through expectation maximization / kmeans:

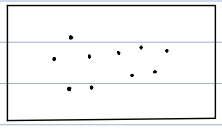
advantages: 1. simple to implement on large dataset
2. guarantees convergence
3. generalize to clusters of different shapes and sizes

disadvantages: 1. choice of k (loss vs. cluster plot)
2. depend on initial centroid positions (multiple EM runs)
3. influenced by outliers (remove first)

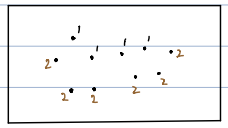
simple example: assume we have two different markers (O, X) with 2-D coordinates (x, y)



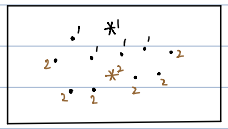
(ground truth)

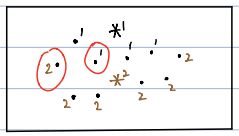


(data we have)



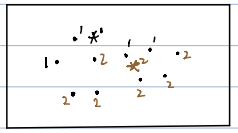
Step 1: choose k and
- ✓ ① assign data randomly to clusters and calculate centroids
- ② randomly sample centroids and assign data to closest centroids





Step 2: To which cluster is each point closest to? Update cluster assignment

point i: cluster 1 **2** ···· k
distance 3 **2** ···· d_k



Step 3: Update centroids and iterate until convergence

Assume we have multi-dimensional data X. For each X_i for $i = 1, 2, \dots, N$, it has z dimensions (e.g. each cell has RNA seq gene expression counts of z genes. We set K clusters, for $K = 1, 2, \dots, k$.

Steps:	hard kmeans ($\geq$1D)	1D soft kmeans
1. assign data randomly and calculate centroids	$\mu_{k,z} = \frac{\sum_{i \in C_k} X_{i,z}}{\lvert C_k \rvert}$	once we get d_{ik}, we calculate the responsibility r_{ik} (similar to probability to assign X_i to μ_k)
2. update cluster assignment by calculating distances	$d_{ik} = \sqrt{\sum_{j=1}^{z} (X_{i,j} - \mu_{k,j})^2}$	$r_{ik} = \frac{\exp(-\beta d_{ik}^2)}{\sum_{k'=1}^{K} \exp(-\beta d_{ik'}^2)}$ ↳ essentially Gaussian
update μ_k	$\mu_{k,z} = \frac{\sum_{i \in C_k} X_{i,z}}{\lvert C_k \rvert}$	where $\sum_{k'=1}^{K} r_{ik} = 1$ for each i $\mu_k = \frac{\sum_{i=1}^{N} r_{ik} \cdot X_i}{\sum_{i=1}^{N} r_{ik}}$ (weighted average of all X_i)
3. set a criteria for convergence (objective function)	$\sum_{k=1}^{K} \sum_{i=1}^{N} d_{ik}^2$ (minimize)	

1) check if point assignments change

2) check if centroid locations change

local minimizer need to set a reasonable threshold ← 3) check if objective function changes

4. run kmeans multiple times and choose optimal objective function and corresponding cluster assignment

hard: if a point is close to 2 clusters, we force it to be only one cluster

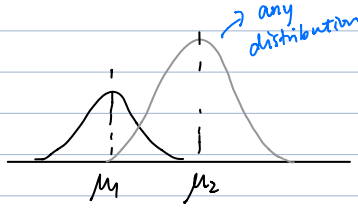
stiffness β { → ∞ : hard kmeans; → 0 : useless

① reinitialize randomly

Also mention the case of empty cluster: ② find the fartherest point and assign it as the new centroid

---

Mixture model: can be any distribution



1. Q components with probability π_q for $q = 1, 2, \dots, Q$

model parameters ↑

$\pi_q = P(q \mid \theta)$ and $\sum_{q=1}^{Q} \pi_q = 1$

2. $P(X_i, q \mid \theta) = P(X_i \mid q, \theta) \cdot P(q \mid \theta) = \pi_q \cdot P(X_i \mid q, \theta)$

$P(X_i \mid \theta) = \sum_{q=1}^{Q} P(X_i, q \mid \theta) = \sum_{q=1}^{Q} \pi_q \, P(X_i \mid q, \theta)$

Q: What's the probability of cell i belonging to certain cell type given the RNA seq data?

$P(\text{cell type} \mid \text{data } i) = P(q \mid X_i, \theta)$ ← distribution { Gaussian / Negative binomial (RNA seq)

Soft kmeans with mixture model

Steps:

1. assign data randomly and calculate centroids

Centroids have initial π_q and μ_q (depend on what distribution we will use)

$$P(q \mid X_i, \theta) = \frac{P(X_i, q \mid \theta)}{P(X_i \mid \theta)} = \frac{\pi_q \cdot P(X_i \mid q, \theta)}{\sum_{q=1}^{Q} \pi_q \cdot P(X_i \mid q, \theta)}$$

2. update cluster assignment by calculating $P(q \mid X_i, \theta)$ for each q.

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

update μ_q

$$\hat{\mu_q} = \frac{\sum_{i=1}^{N} X_i \cdot P(q \mid X_i, \theta)}{\sum_{i=1}^{N} P(q \mid X_i, \theta)} \quad \text{and} \quad \hat{\pi_q} = \frac{\sum_{i=1}^{N} P(q \mid X_i, \theta)}{N}$$

3. set a criteria for convergence (objective function)

we want to maximize $P(X \mid \theta)$ or minimize $-\log P(X \mid \theta)$

$$P(X \mid \theta) = \prod_{i=1}^{N} P(X_i \mid \theta) \quad (i.i.d)$$

$$= \prod_{i=1}^{N} \left[\sum_{q=1}^{Q} P(X_i, q \mid \theta) \right]$$

$$= \prod_{i=1}^{N} \sum_{q=1}^{Q} \pi_q \cdot P(X_i \mid q, \theta)$$

$$-\log P(X \mid \theta) = -\log \left(\prod_{i=1}^{N} \sum_{q=1}^{Q} \pi_q \cdot P(X_i \mid q, \theta) \right)$$

$$= -\sum_{i=1}^{N} \log \left(\sum_{q=1}^{Q} \pi_q \cdot P(X_i \mid q, \theta) \right)$$

log sumexp function: $\log \sum_{x=x} e^x$

it's input is an array of x

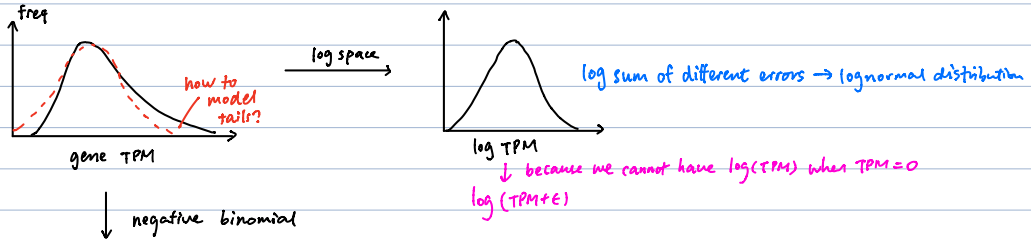
$$\log \left[\pi_1 P(X_i \mid \theta_1) + \pi_2 P(X_i \mid \theta_2) + \cdots + \pi_Q P(X_i \mid \theta_Q) \right]$$

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

assume constant, but may not be

$P(X_i | q, \theta)$ if Gaussian: $P(X_i | \mu_q, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x_i - \mu_q)^2}{2\sigma^2}\right)$

For scRNA-seq, we can model counts with either lognormal / negative binomial distributions.

① 

② Poisson ~ (μ) → it's discrete while TPM is continuous. (NB is better in modeling counts)

\+ sampling a transcript from a large pool has a Poisson distribution

NB to model the dispersion and the uncertainty in the mean of the distribution,

⇓ we use gamma-poisson distribution with 2 parameters

gamma-poisson ~ (μ, ϕ)

↓ mean # of occurance of a rare event

Define NB distribution: # of failures in a series of i.i.d Bernolli trials before n successes occurs, with success rate of p.
Let X be the # of failures, $X \sim NB(n, p)$

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

Each gene X_i has prob in cluster q with $NB(n, p)$ where $n = \frac{1}{\phi}$ ↑ models dispersion and $p = \frac{1}{1 + \mu_q \cdot \phi}$ updated

in $P(X_i | q, \theta) = P(X_i | \mu_q)$

- import scipy.stats
- scipy.stats.nbinom.logpmf(y, n, p)