

k -Means Clustering, Gaussian Mixture Models, and Principal Component Analysis

Gautam Kamath

Clustering

- Canonical unsupervised learning problem
 - Versus supervised learning
- (Draw two cluster picture)
- Given $X = \{X_1, \dots, X_n\}$, partition into sets $C_1, \dots, C_k$
 - E.g., say if $n = 5, k = 3$, then $C_1 = \{X_1, X_5\}$, $C_2 = \{X_2\}$, $C_3 = \{X_3, X_4\}$
 - k is a hyperparameter
- Goals (informally):
 - Points in a cluster are similar, points in different clusters are dissimilar

k -Means Clustering

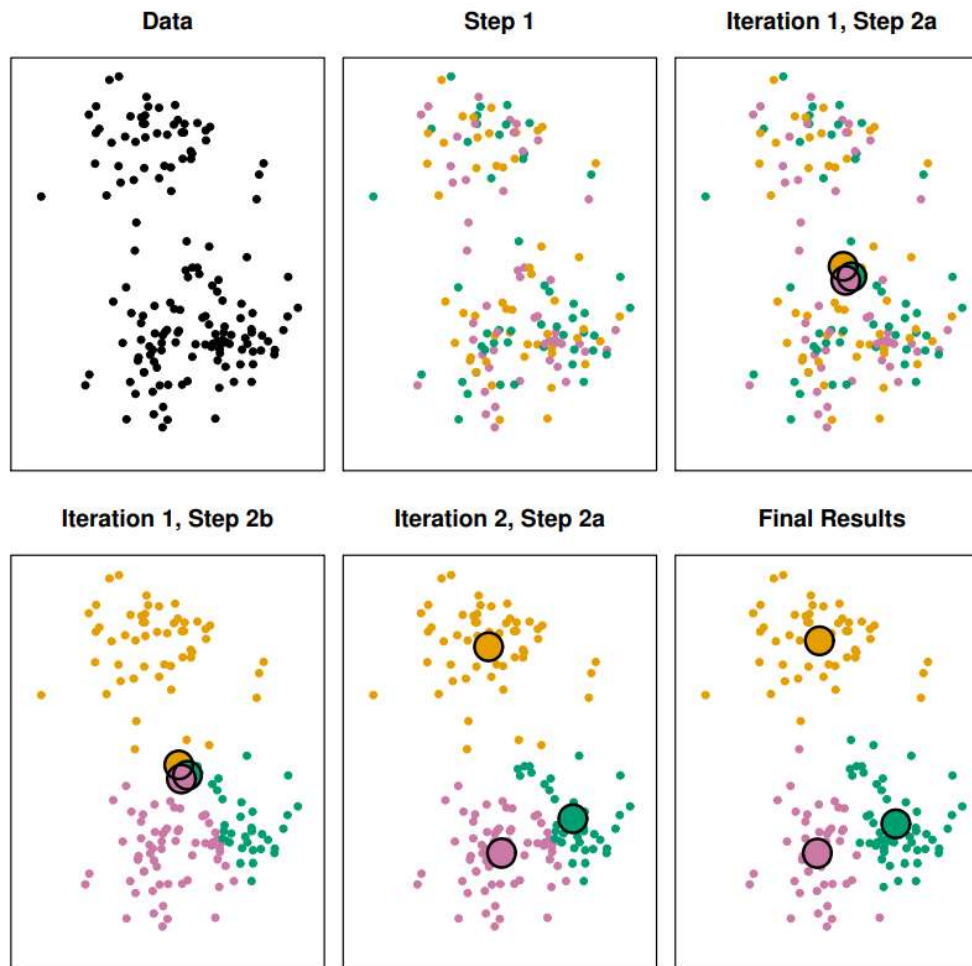
- $\min_{\text{partitions } C_1, \dots, C_k} \sum_{j=1}^k W(C_j)$
 - $W(\cdot)$ is a function that measures “cost” for points in cluster j
- For k -means: $W(C_j) = \frac{1}{|C_j|} \sum_{X_i, X'_i \in C_j} \|X_i - X'_i\|_2^2$
 - Equivalently: $W(C_j) = 2 \sum_{X_i \in C_j} \|X_i - \mu_j\|_2^2$, where $\mu_j = \frac{1}{|C_j|} \sum_{X_i \in C_j} X_i$
- How to optimize?
- Slow algorithm: Try all partitions (there’s a lot, roughly k^n)
- Usually: Lloyd’s algorithm

Lloyd's Algorithm

1. Initialize partition $C_1, \dots, C_k$ (could be random or carefully chosen)
2. For each cluster C_j , compute centroid $\mu_j = \frac{1}{|C_j|} \sum_{X_i \in C_j} X_i$
3. For each point X_i assign it to cluster with nearest centroid
 - Assign it to cluster with index $\arg \min_j \|X_i - \mu_j\|_2^2$
4. Go to step 2, repeat until convergence

Main idea: given clusters, compute centers. Then given centers, compute clusters. Repeat.

Example



Comments on k -Means/Lloyd's Algorithm

- Drawbacks
 - Can be slow to converge
 - May only converge to a local optimum
 - NP-hard to optimize even to a constant factor approximation
- Solutions
 - Repeat many times with different initializations, take best

k -Means with Restarts



Comments on k -Means/Lloyd's Algorithm

- Drawbacks
 - Can be slow to converge
 - May only converge to a local optimum
 - NP-hard to optimize even to a constant factor approximation
- Solutions
 - Repeat many times with different initializations, take best
 - Do better initializations (e.g., k -means++)

Generative Models

- Given $X_1, \dots, X_n$ drawn i.i.d. from some distribution p_θ
- Goal: Output $\hat{p} \approx p_\theta$
- Try to estimate the distribution which generated the dataset
- A simple case: $X_1, \dots, X_n \sim N(\mu, 1)$
 - $p_\mu(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2}\right)$
- MLE: $\hat{\mu} = \arg \max_{\mu} \sum_{i=1}^n \log p_\mu(X_i) = \arg \max_{\mu} \sum_{i=1}^n -(X_i - \mu)^2 = \frac{1}{n} \sum_{i=1}^n X_i$
- Output $\hat{p} = N(\hat{\mu}, 1)$

Mixture Model

- $p_{\theta}(x) = \sum_{j=1}^k \pi_j p_{\theta^{(j)}}^{(j)}(x)$
 - π_j 's are *mixing weights*: $\pi_j \geq 0$ and $\sum_{j=1}^k \pi_j = 1$
 - $p_{\theta^{(j)}}^{(j)}(x)$ is the PDF for component j
 - Density is a convex combination of a collection of other densities
- Intuitive way is to think about the sampling procedure
 1. Draw sample $\in \{1, \dots, k\}$ according to distribution π
 2. Output a sample from $p_{\theta^{(j)}}^{(j)}(x)$
- (Draw picture of male and female human height distributions)

Gaussian Mixture Model

- $p_{\theta}(x) = \sum_{j=1}^k \pi_j N(\mu_j, \Sigma_j, x)$
 - Note: $N(\mu_j, \Sigma_j, x)$ is the PDF of $N(\mu_j, \Sigma_j)$ at the point x
- (Draw picture of 3-GMM)
- Different from clustering
 - Clustering is a non-stochastic setting
 - Goal is a bit different: identify clusters vs estimate parameters

Gaussian Mixture Models

- Problem would be easy if we knew which component each came from
- Let Z_i be the component that X_i was sampled from, $Z_i \in \{1, \dots, k\}$
- If we knew then

$$\hat{\theta} = \arg \max_{\theta} \sum_{i=1}^n \log p_{\theta}(x_i) = \arg \max_{\theta = \{(\pi_j, \mu_j, \Sigma_j)\}} \sum_{i=1}^n \log \left(\sum_{j=1}^k \mathbf{1}_{\{Z_i=j\}} \pi_j N(\mu_j, \Sigma_j, x) \right)$$
$$\hat{\pi}_j = \frac{1}{n} \sum_{Z_i=j} 1, \hat{\mu}_j = \frac{1}{\sum_{Z_i=j} 1} \sum_{Z_i=j} X_i, \hat{\Sigma}_j = \frac{1}{\sum_{Z_i=j} 1} \sum_{Z_i=j} (X_i - \hat{\mu}_j)(X_i - \hat{\mu}_j)^T$$

- If we knew the components, then just take empirical estimates... but we don't.

Expectation Maximization (EM)

- Problem would have been easy if we knew which component each point came from
 - May be impossible to tell in some cases (draw point between two Gaussians)
 - Instead, say it came from multiple components fractionally (50-50 drawing)
 - Expectation Maximization: a “soft” version of k -means
 - A point belongs to *multiple* components instead of just one
1. Given θ , fractionally assign points X_i to mixture components
 2. Given fractional assignment of X_i to clusters, compute best θ

Deriving the EM Updates, starting at
 $\arg \max_{\theta} \sum_i \log p_{\theta}(X_i)$

$$\begin{aligned} & \log p_{\theta}(X_i) \\ &= \log \sum_{j=1}^k p_{\theta}(X_i, Z_i = j) \\ &= \log \sum_{j=1}^k \frac{q_i(Z_i = j)}{q_i(Z_i = j)} p_{\theta}(X_i, Z_i = j) \\ & \quad \text{(think } q_i \text{ as a "guess" for the} \\ & \quad \text{distribution of } Z_i) \\ &= \log \sum_{j=1}^k q_i(Z_i = j) \frac{p_{\theta}(X_i, Z_i = j)}{q_i(Z_i = j)} \\ &= \log E_{Z_i \sim q_i} \left[\frac{p_{\theta}(X_i, Z_i)}{q_i(Z_i)} \right] \end{aligned}$$

$$\begin{aligned} & \geq E_{Z_i \sim q_i} \log \left[\frac{p_{\theta}(X_i, Z_i)}{q_i(Z_i)} \right] \\ & \quad \text{(by Jensen's inequality)} \\ &= \sum_{j=1}^k q_i(Z_i = j) \log p_{\theta}(X_i, Z_i = j) - \\ & \quad \sum_{j=1}^k q_i(Z_i = j) \log q_i(Z_i = j) \end{aligned}$$

In summary

$$\begin{aligned} & \arg \max_{\theta} \sum_i \log p_{\theta}(X_i) \\ \geq & \arg \max_{\theta, \{q_i\}} \sum_i \left(\sum_{j=1}^k q_i(Z_i = j) \log p_{\theta}(X_i, Z_i = j) - \sum_{j=1}^k q_i(Z_i = j) \log q_i(Z_i = j) \right) \end{aligned}$$

E Step (Expectation Step)

$$\arg \max_{\theta, \{q_i\}} \sum_i \left(\sum_{j=1}^k q_i(Z_i = j) \log p_{\theta}(X_i, Z_i = j) - \sum_{j=1}^k q_i(Z_i = j) \log q_i(Z_i = j) \right)$$

- E step: fix θ , optimize q_i 's (let's focus on a single q_i for simplicity)

$$\begin{aligned} & \arg \max_{q_i} \sum_{j=1}^k q_i(Z_i = j) (\log p_{\theta}(X_i, Z_i = j) - \log q_i(Z_i = j)) \\ &= \arg \max_{q_i} \sum_{j=1}^k q_i(Z_i = j) (\log p_{\theta}(Z_i = j | X_i) + \log p_{\theta}(X_i) - \log q_i(Z_i = j)) \end{aligned}$$

(Drop constant $\log p_{\theta}(X_i)$)

$$= \arg \min_{q_i} \mathbb{E}_{Z_i \sim q_i} \left[\log \frac{q_i(Z_i)}{p_{\theta}(Z_i | X_i)} \right]$$

This is the KL Divergence between distributions q_i and $p_{\theta}(\cdot | X_i)$. It is always non-negative, and minimized when $q_i(Z_i) = p_{\theta}(Z_i | X_i)$, so choose q_i in this way.

M Step (Maximization Step)

$$\arg \max_{\theta, \{q_i\}} \sum_i \left(\sum_{j=1}^k q_i(Z_i = j) \log p_{\theta}(X_i, Z_i = j) - \sum_{j=1}^k q_i(Z_i = j) \log q_i(X_i, Z_i = j) \right)$$

- M step: Fix q_i 's, optimize θ

$$\arg \max_{\theta} \sum_{i=1}^n \sum_{j=1}^k q_i(Z_i = j) \log p_{\theta}(X_i, Z_i = j)$$

- Often solvable in closed form

The Algorithm

1. Initialize θ parameters
2. Run E step
3. Run M step
4. Repeat 2, 3

EM for GMMs

- E step: $q_i(Z_i = j) = p_\theta(Z_i = j|X_i) = \frac{p_\theta(Z_i=j, X_i)}{p_\theta(X_i)} = \frac{\pi_j N(\mu_j, \Sigma_j, X_i)}{\sum_{\ell=1}^k \pi_\ell N(\mu_\ell, \Sigma_\ell, X_i)}$
 - Compute for all X_i , for all $j \in \{1, \dots, k\}$

- M step (for simplicity, 1D, variance = 1. $p_\theta(x) = \sum \pi_j N(\mu_j, 1, x)$):

$$\begin{aligned} & \arg \max_{\theta} \sum_{i=1}^n \sum_{j=1}^k q_i(Z_i = j) \log p_\theta(X_i, Z_i = j) \\ &= \arg \max_{\theta} \sum_{i=1}^n \sum_{j=1}^k q_i(Z_i = j) \log \left(\pi_j \frac{1}{\sqrt{2\pi}} \exp \left(-\frac{(X_i - \mu_j)^2}{2} \right) \right) \\ &= \arg \max_{\theta} \sum_{i=1}^n \sum_{j=1}^k q_i(Z_i = j) \left(\log \pi_j - \frac{(X_i - \mu_j)^2}{2} \right) \end{aligned}$$

Focus on optimizing μ_j

$$\arg \max_{\theta} \sum_{i=1}^n \sum_{j=1}^k q_i(Z_i = j) \left(-\frac{(X_i - \mu_j)^2}{2} \right)$$

Optimize by taking derivative wrt μ_j and setting = 0

$$\sum_{i=1}^n -q_i(Z_i = j)(X_i - \mu_j) = 0$$

Rearranging...

$$\mu_j = \frac{\sum_i q_i(Z_i = j) X_i}{\sum_i q_i(Z_i = j)}$$

Focus on optimizing π_j

$$\begin{aligned} & \arg \max_{\theta} \sum_{i=1}^n \sum_{j=1}^k q_i(Z_i = j) \log \pi_j \quad \left(\text{s.t. } \sum_{\ell=1}^k \pi_{\ell} = 1 \right) \\ & \arg \max_{\theta} \sum_{i=1}^n \sum_{j=1}^k q_i(Z_i = j) \log \pi_j + \lambda \left(\sum_{j=1}^k \pi_j - 1 \right) \end{aligned}$$

Differentiate wrt π_j , set equal to 0

$$\begin{aligned} & \sum_{i=1}^n \frac{q_i(Z_i = j)}{\pi_j} + \lambda = 0 \\ & \pi_j = -\frac{1}{\lambda} \sum_{i=1}^n q_i(Z_i = j) \end{aligned}$$

Focus on optimizing π_j

$$\pi_j = -\frac{1}{\lambda} \sum_{i=1}^n q_i(Z_i = j)$$

But what is λ ? Note

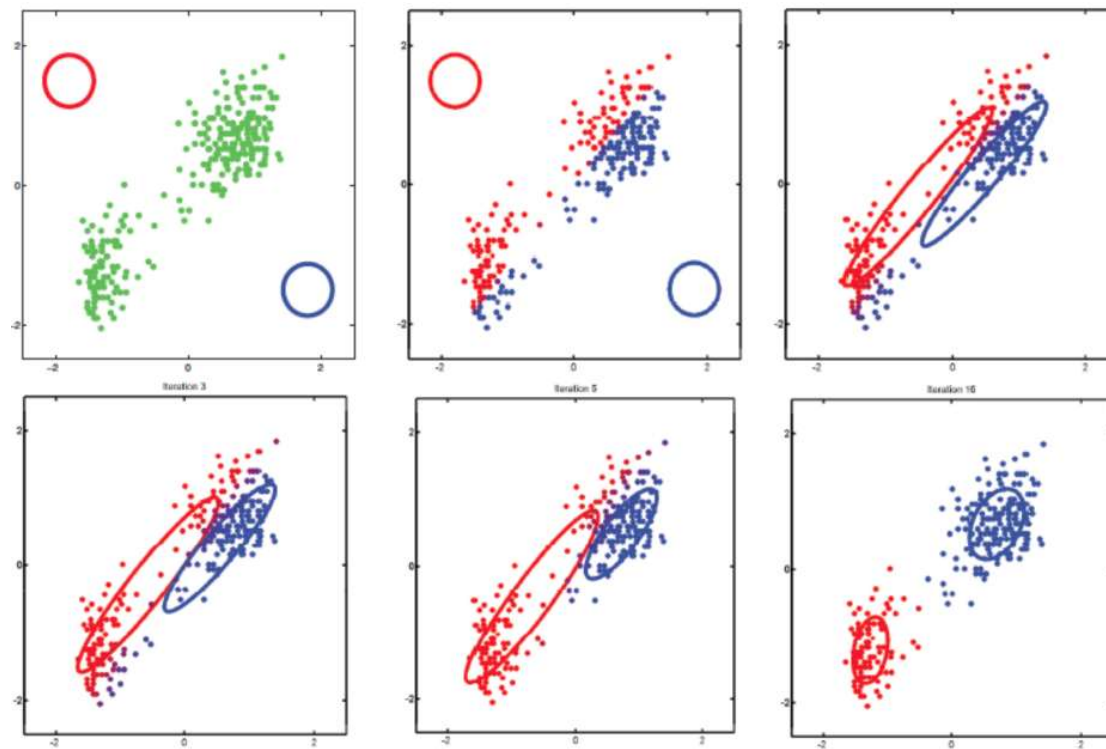
$$1 = \sum_{j=1}^k \pi_j = -\frac{1}{\lambda} \sum_{i=1}^n \sum_{j=1}^k q_i(Z_i = j) = -\frac{n}{\lambda}$$

So $\lambda = -n$.

Therefore

$$\pi_j = \frac{1}{n} \sum_{i=1}^n q_i(Z_i = j)$$

Example



High-Dimensional Data

- Not all dimensions are equally important
- (Draw one-dimensional data in two-dimensional space)
- Generally, can imagine low-d structure in high-d data
 - Caveat: we will only handle linear low-d structures
- How did we choose this one “interesting” direction?
- Direction with maximum *variance*
 - Suppose Z has mean 0, then $\text{Var}[Z] = \mathbf{E}[Z^2]$
 - For a dataset of X_i 's with mean 0, use empirical variance $\frac{1}{n} \sum_i X_i^2$ as a proxy

Principal Component Analysis

- Let $X \in \mathbf{R}^{n \times d}$ be a data matrix
 - Assume data is centered: each column has mean 0
 - Often assume data is standardized: each column has variance 1
- Direction v of max variance:

$$\arg \max_{\|v\|_2=1} \frac{1}{n} \sum_i \langle v, X_i \rangle^2$$

- $\frac{1}{n} \sum_i \langle v, X_i \rangle^2 = \frac{1}{n} \|Xv\|^2 = v^T \frac{X^T X}{n} v = v^T \hat{\Sigma} v$
- $\hat{\Sigma} = \frac{1}{n} X^T X$ is the empirical covariance matrix
- The top principal component $\arg \max_{\|v\|_2=1} v^T \hat{\Sigma} v$ is the top eigenvector of the empirical covariance matrix

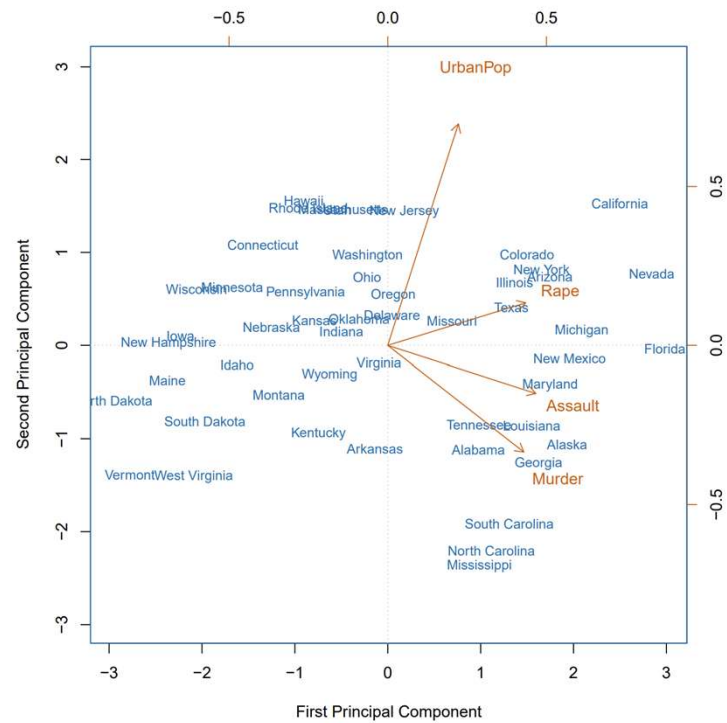
Beyond the first principal component

- Top principal component: $\arg \max_{\|v\|_2=1} v^T \hat{\Sigma} v$
- Second principal component: $\arg \max_{\|u\|_2=1} u^T \hat{\Sigma} u$ s.t. $\langle u, v \rangle = 0$
 - Top principal component in the subspace orthogonal to v
- k -th principal component: $\arg \max_{\|v^{(k)}\|_2=1} v^{(k)T} \hat{\Sigma} v^{(k)}$ s.t. $v^{(k)} \perp \text{span}\{v^{(1)}, \dots, v^{(k-1)}\}$
 - Top principal component in the subspace orthogonal to first $k - 1$ principal components

An actual dataset

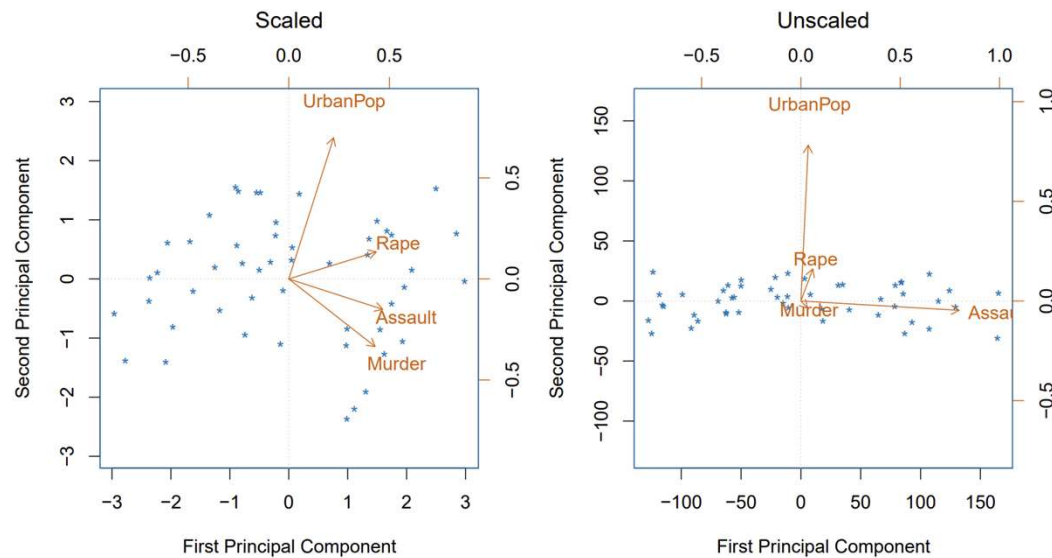
- $n = 50$ states
- $d = 4$ features: (murder, assault, rape, urban_population)
- What will the projection onto the top two principal components look like?
 - Plot of $[\langle x_i, v^{(1)} \rangle, \langle x_i, v^{(2)} \rangle]$

An actual dataset



Scaling

- Variances of (murder, rape, assault, urban_population):
 - 18.97, 87.73, 6945.16, 209.5



Computational aspects

- Any matrix X has a decomposition USV^T
 - Singular value decomposition (SVD)
 - U and V are orthogonal matrices, $S = \text{diag}(s)$ is diagonal
 - $O(nd^2)$ time to compute
- $\hat{\Sigma} = \frac{1}{n}VS^T SV^T$
 - Columns of V are principal components, corresponding eigenvalues are s_j^2/n
- If we just need a small # of principal components: power iteration
 - $z \leftarrow \frac{X^T(Xz)}{\|X^T(Xz)\|_2}$
 - $O(nd)$ time per iteration