Mixture Models

He He
(adapted from David Rosenberg's slides)

CDS, NYU

May 5, 2020

Contents

1. k-Means Clustering

2. Gaussian Mixture Models

Logistics

- Final exam
  - Released on May 14th, 6pm EST, available for 24 hours.
  - Duration: two hours.
  - Format: multiple choice and short answers on Gradescope.
- Review given by Joshua and Shubham during the usual section and tutorial hours.

Today's lecture

- A peek into unsupervised learning—clustering and mixture models.
- Final remarks and conclusion.

k-Means Clustering

Unsupervised learning

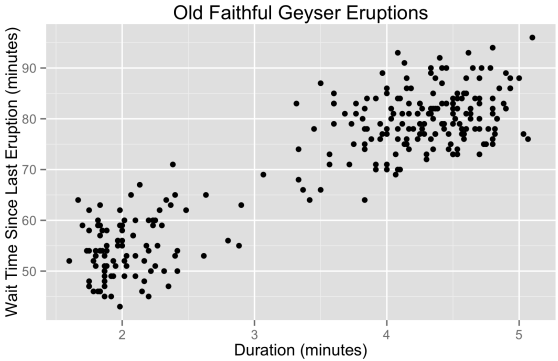
Goal Discover interesting structure in the data.

Formulation Density estimation: $p(x; \theta)$ (often with latent variables).

Examples

- Discover clusters: cluster data into groups.
- Discover factors: project high-dimensional data to a small number of "meaningful" dimensions, i.e. dimensionality reduction.
- Discover graph structures: learn joint distribution of correlated variables, i.e. graphical models.

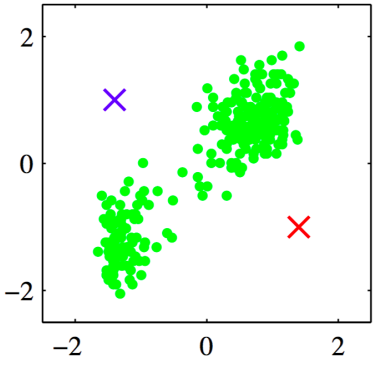
Example: Old Faithful Geyser



- Looks like two clusters.
- How to find these clusters algorithmically?

k-Means: By Example

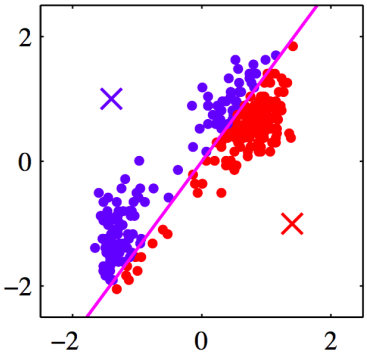
- Standardize the data.
- Choose two cluster centers.



From Bishop's Pattern recognition and machine learning, Figure 9.1(a).

k-means: by example

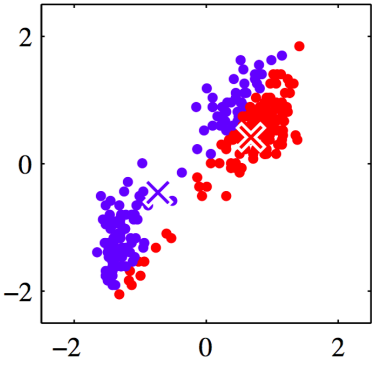
- Assign each point to closest center.



From Bishop's Pattern recognition and machine learning, Figure 9.1(b).

k-means: by example

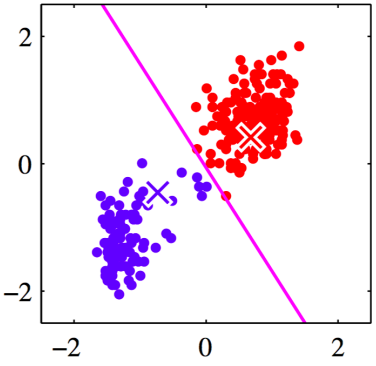
- Compute new cluster centers.



From Bishop's Pattern recognition and machine learning, Figure 9.1(c).

k-means: by example

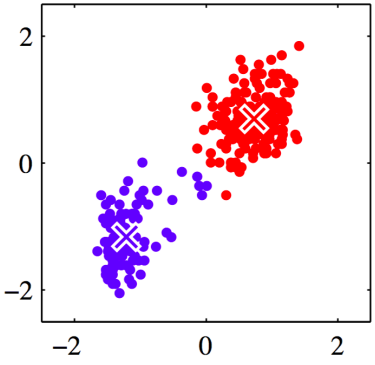
- Assign points to closest center.



From Bishop's Pattern recognition and machine learning, Figure 9.1(d).

k-means: by example

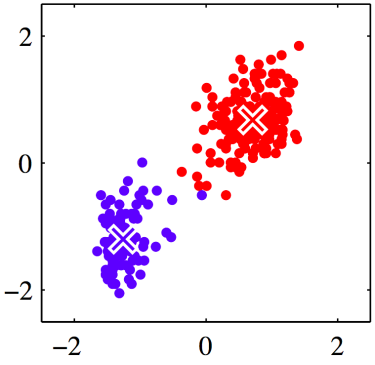
- Compute cluster centers.



From Bishop's Pattern recognition and machine learning, Figure 9.1(e).

k-means: by example

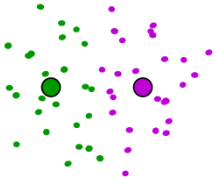
- Iterate until convergence.



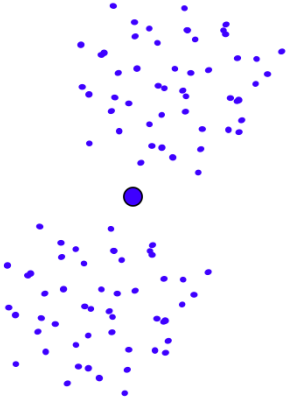
From Bishop's Pattern recognition and machine learning, Figure 9.1(i).

Suboptimal Local Minimum

- The clustering for $k = 3$ below is a local minimum, but suboptimal:



Would be better to have one cluster here



… and two clusters here

From Sontag's DS-GA 1003, 2014, Lecture 8.

Formalize k-Means

- Dataset $\mathcal{D} = \{x_1, \ldots, x_n\} \subset \mathcal{X}$ where $\mathcal{X} = \mathrm{R}^d$.
- Goal: Partition data $\mathcal{D}$ into k disjoint sets $C_1, \ldots, C_k$.
- Let $c_i \in \{1, \ldots, k\}$ be the cluster assignment of x_i.
- The **centroid** of C_i is defined to be

$$\mu_i = \arg\min_{\mu \in \mathcal{X}} \sum_{x \in C_i} \|x - \mu\|^2. \qquad \text{mean of } C_i \tag{1}$$

- The k-means objective is to minimize the distance between each example and its cluster centroid:

$$J(c, \mu) = \sum_{i=1}^{n} \|x_i - \mu_{c_i}\|^2. \tag{2}$$

k-Means: Algorithm

1. Initialize: Randomly choose initial centroids $\mu_1, \ldots, \mu_k \in \mathbb{R}^d$.
2. Repeat until convergence (i.e. c_i doesn't change anymore):
   1. For all i, set

$$c_i \leftarrow \arg\min_j \|x_i - \mu_j\|^2. \qquad \text{Minimize } J \text{ w.r.t. } c \text{ while fixing } \mu \tag{3}$$

   2. For all j, set

$$\mu_j \leftarrow \frac{1}{|C_j|} \sum_{x \in C_j} x. \qquad \text{Minimze } J \text{ w.r.t. } \mu \text{ while fixing } c. \tag{4}$$

- Recall the objective: $J(c, \mu) = \sum_{i=1}^n \|x_i - \mu_{c_i}\|^2$.
- k-means is coordinate descent on J.

Avoid bad local minima

k-means converges to a local minimum.

- k-means is coordinate descent on J, thus J will monotonically decrease.
- But J is non-convex, thus no guarantee to converging to the global minimum.

Avoid getting stuck with bad local minima:

- Re-run with random initial centroids.
- k-**means++**: choose initial centroids that spread over all data points.
  - Randomly choose the first centroid from the data points $\mathcal{D}$.
  - Sequentially choose subsequent centroids from points that are farther away from current centroids:
    - Compute distance between each x_i and the closest already chosen centroids.
    - Randomly choose next centroid with probability proportional to the computed distance squared.

Summary

We've seen

- Clustering—an unsupervised learning problem that aims to discover group assignments.
- k-means:
  - Algorithm: alternating between assigning points to clusters and computing cluster centroids.
  - Objective: minmizing some loss function by cooridinate descent.
  - Converge to a local minimum.

Next, probabilistic model of clustering.

- A generative model of x.
- Maximum likelihood estimation.

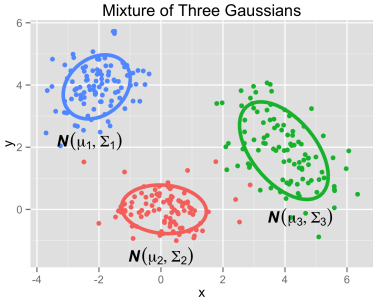
Gaussian Mixture Models

Probabilistic Model for Clustering

- Problem setup:
  - There are k clusters (or **mixture components**).
  - We have a probability distribution for each cluster.
- Generative story of a **mixture distribution**:
  1. Choose a random cluster $z \in \{1, 2, \ldots, k\}$.
  2. Choose a point from the distribution for cluster z.

Example:

1. Choose $z \in \{1, 2, 3\}$ with $p(1) = p(2) = p(3) = \frac{1}{3}$.
2. Choose $x \mid z \sim \mathcal{N}(X \mid \mu_z, \Sigma_z)$.



Gaussian mixture model (GMM)

Generative story of GMM with k mixture components:

1. Choose cluster $z \sim \text{Categorical}(\pi_1, \ldots, \pi_k)$.
2. Choose $x \mid z \sim \mathcal{N}(\mu_z, \Sigma_z)$.

Probability density of x:

- Sum over (marginalize) the **latent variable** z.

$$p(x) = \sum_z p(x, z) \tag{5}$$

$$= \sum_z p(x \mid z) p(z) \tag{6}$$

$$= \sum_k \pi_k \mathcal{N}(\mu_k, \Sigma_k) \tag{7}$$

Learning GMMs

How to learn the parameters π_k, μ_k, Σ_k?

- MLE (also called maximize marginal likelihood).
- Log likelihood of data:

$$L(\theta) = \sum_{i=1}^{n} \log p(x_i; \theta) \tag{8}$$

$$= \sum_{i=1}^{n} \log \sum_{z} p(x, z; \theta) \tag{9}$$

- Cannot push log into the sum... z and x are coupled.
- No closed-form solution for GMM—try to compute the gradient yourself!

Learning latent-variable models with MLE

In general, we can show

$$\frac{d}{d\theta}\sum_z \log p(x,z;\theta) = \mathbb{E}_{p(z|x)}\left[\frac{d}{d\theta}\log p(x,z)\right] \quad \text{Exercise} \tag{10}$$

- Expected gradient of joint log probability w.r.t. the posterior of z.
- Applies to general latent variable models.
- Hard to compute in general.
- For GMM, gradient ascent is doable but often slow.

Learning GMMs: observable case

Suppose we observe cluster assignments z. Then MLE is easy:

$$n_z = \sum_{i=1}^{n} 1(z_i = z) \qquad \text{\# examples in each cluster} \tag{11}$$

$$\hat{\pi}(z) = \frac{n_z}{n} \qquad \text{fraction of examples in each cluster} \tag{12}$$

$$\hat{\mu}_z = \frac{1}{n_z} \sum_{i:z_i=z} x_i \qquad \text{empirical cluster mean} \tag{13}$$

$$\hat{\Sigma}_z = \frac{1}{n_z} \sum_{i:z_i=z} (x_i - \hat{\mu}_z)(x_i - \hat{\mu}_z)^T. \qquad \text{empirical cluster covariance} \tag{14}$$

Learning GMMs: inference

The inference problem: observe x, want to know z.

$$p(z = j \mid x_i) = p(x, z = j)/p(x) \tag{15}$$

$$= \frac{p(x \mid z = j)p(z = j)}{\sum_k p(x \mid z = k)p(z = k)} \tag{16}$$

$$= \frac{\pi_j \mathcal{N}(x_i \mid \mu_j, \Sigma_j)}{\sum_k \pi_k \mathcal{N}(x_i \mid \mu_k, \Sigma_k)} \tag{17}$$

- $p(z \mid x)$ is a soft assignment.
- If we know the parameters μ, Σ, π, this would be easy to compute.

EM for GMM

Let's compute the cluster assignments and the parameters iteratively.

The expectation-minimization (EM) algorithm:

1. Initialize parameters μ, Σ, π randomly.
2. Run until convergence:
   1. E-step: fill in latent variables by inference.
      - compute soft assignments $p(z \mid x_i)$ for all i.
   2. M-step: standard MLE for μ, Σ, π given "observed" variables.
      - Equivalent to MLE in the observable case on data weighted by $p(z \mid x_i)$.

M-step for GMM

- Recall the gradient is:

$$\frac{d}{d\theta}\sum_z \log p(x,z;\theta) = \mathbb{E}_{p(z|x)}\left[\frac{d}{d\theta}\log p(x,z)\right] \tag{18}$$

- Let $p(z \mid x)$ be the soft assignments:

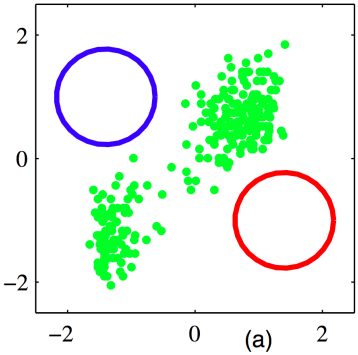
$$\gamma_i^j = \frac{\pi_j \mathcal{N}(x_i \mid \mu_j, \Sigma_j)}{\sum_{c=1}^k \pi_c \mathcal{N}(x_i \mid \mu_c, \Sigma_c)}.$$

- Exercise: show that

$$\begin{aligned}
\mu_c^{\text{new}} &= \frac{1}{n_c}\sum_{i=1}^n \gamma_i^c x_i \\
\Sigma_c^{\text{new}} &= \frac{1}{n_c}\sum_{i=1}^n \gamma_i^c (x_i - \mu_c^{\text{new}})(x_i - \mu_c^{\text{new}})^T \\
\pi_c^{\text{new}} &= \frac{n_c}{n}.
\end{aligned}$$

EM for GMM

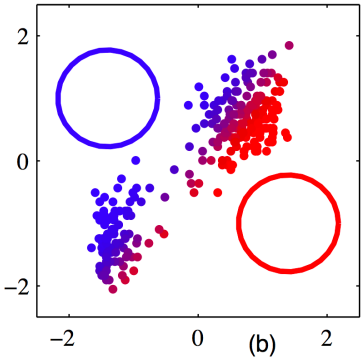
- Initialization



From Bishop's Pattern recognition and machine learning, Figure 9.8.

EM for GMM

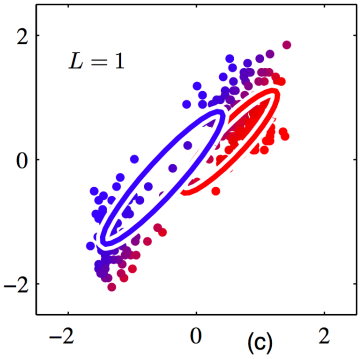
- First soft assignment:



From Bishop's Pattern recognition and machine learning, Figure 9.8.

EM for GMM

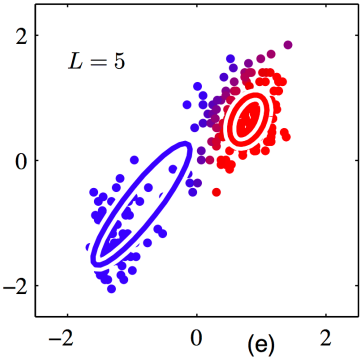
- First soft assignment:



From Bishop's Pattern recognition and machine learning, Figure 9.8.

EM for GMM

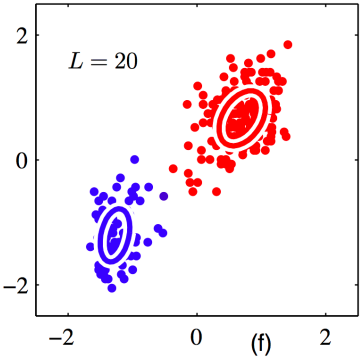
- After 5 rounds of EM:



From Bishop's Pattern recognition and machine learning, Figure 9.8.

EM for GMM

- After 20 rounds of EM:



From Bishop's Pattern recognition and machine learning, Figure 9.8.

EM for GMM: Summary

- EM is a general algorithm for learning latent variable models.
- Key idea: if data was fully observed, then MLE is easy.
  - E-step: fill in latent variables by computing $p(z \mid x, \theta)$.
  - M-step: standard MLE given fully observed data.
- Simpler and more efficient than gradient methods.
- Can prove that EM monotonically improves the likelihood and converges to a local minimum.
- k-means is a special case of EM for GMM with hard assignments, also called hard-EM.