

# Mixture Models

He He  
(adapted from David Rosenberg's slides)

CDS, NYU

May 5, 2020

# Contents

1  $k$ -Means Clustering

2 Gaussian Mixture Models

# $k$ -Means Clustering

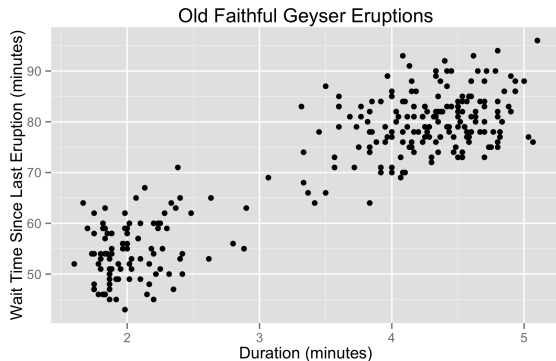
# Unsupervised learning

**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.

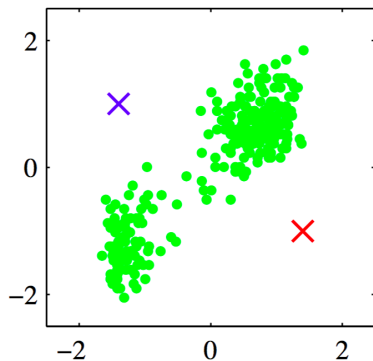
## Example: Old Faithful Geyser



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

## $k$ -Means: By Example

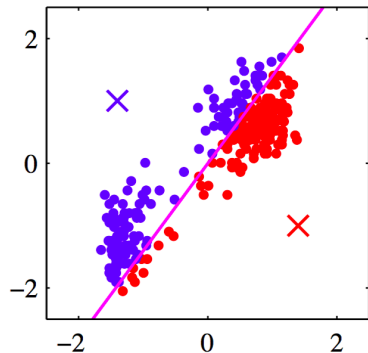
- Standardize the data.
- Choose two cluster centers.



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

## k-means: by example

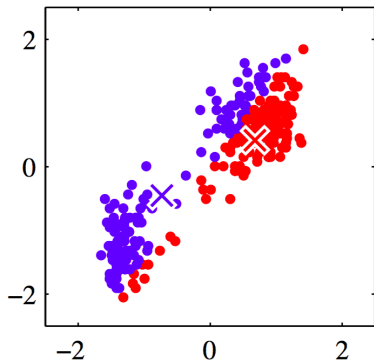
- Assign each point to closest center.



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

## $k$ -means: by example

- Compute new class centers.

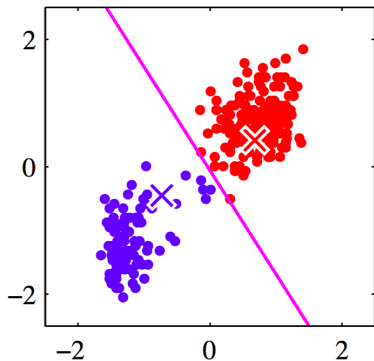


From Bishop's [Pattern recognition and machine learning](#), Figure 9.1(c).



## $k$ -means: by example

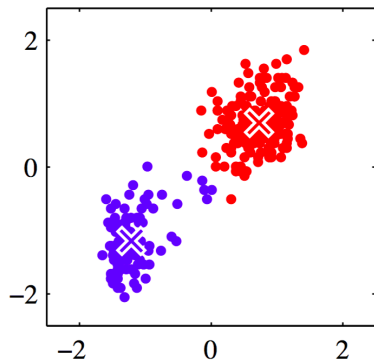
- Assign points to closest center.



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

## $k$ -means: by example

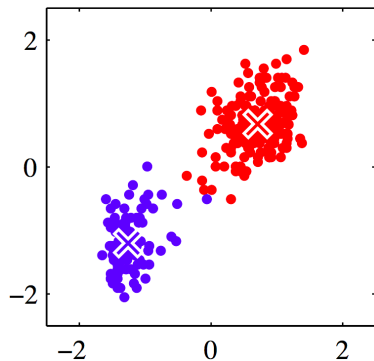
- Compute cluster centers.



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).

# $k$ -Means Clustering

## $k$ -means: by example

Iterate until convergence.

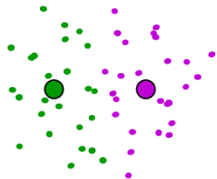


From Bishop's *Pattern recognition and machine learning*, Figure 6.102.

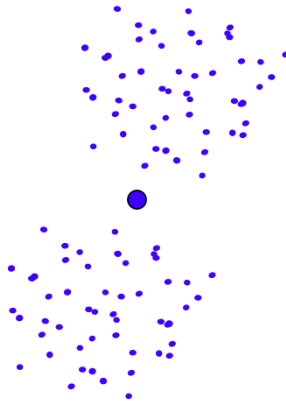
1. When do we achieve convergence? The cluster assignments no longer change.
2. Does it always converge? Yes, more on this later.
3. Does it always converge to good clusters? Not necessarily.

# 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

# Formalize $k$ -Means

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

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

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

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

# k-Means: Algorithm

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

- 1 For all  $i$ , set

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

- 2 For all  $j$ , set

$$\mu_j \leftarrow \frac{1}{|C_j|} \sum_{x \in C_j} x. \quad \text{Minimize } J \text{ w.r.t. } \mu \text{ while fixing } c. \quad (4)$$

- 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.



$k$ -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.

1. It can be shown that  $k$ -means++ converges to a solution not much worse than the optimal solution. Specifically,  $\mathbb{E}[J_{k\text{-means}++}] \leq O(\log k)J^*$ .

# 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: minimizing some loss function by coordinate descent.
  - Converge to a local minimum.

Next, probabilistic model of clustering.

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

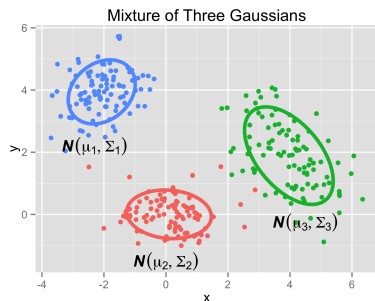
# Gaussian Mixture Models

# 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  $z \in \{1, 2, \dots, k\}$ .
  - 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  $z \sim \text{Categorical}(\pi_1, \dots, \pi_k)$ .
- 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  $\pi_k, \mu_k, \Sigma_k$ ?

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

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

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

- 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

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

- Expected **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.

In general, we can show

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

- Expected **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.

1. Had we know  $z$ , this would be much easier...



## Learning GMMs: observable case

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

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

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

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

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

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

$$p(z = j | x_i) = p(x, z = j) / p(x) \quad (15)$$

$$= \frac{p(x | z = j) p(z = j)}{\sum_k p(x | z = k) p(z = k)} \quad (16)$$

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

- $p(z | x)$  is a **soft assignment**.
- If we know the parameters  $\mu, \Sigma, \pi$ , this would be easy to compute.

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

The expectation-minimization (EM) algorithm:

- ① Initialize parameters  $\mu, \Sigma, \pi$  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  $\mu, \Sigma, \pi$  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:

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

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

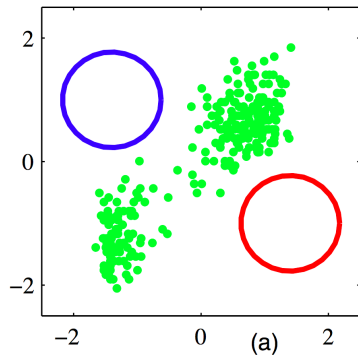
$$\gamma_i^j = \frac{\pi_j \mathcal{N}(x_i | \mu_j, \Sigma_j)}{\sum_{c=1}^k \pi_c \mathcal{N}(x_i | \mu_c, \Sigma_c)}.$$

- 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

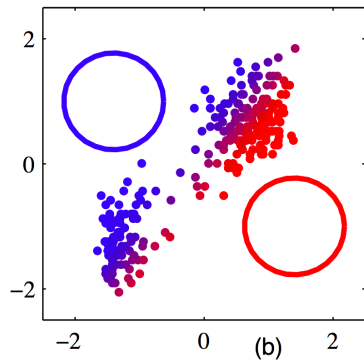
## • Initialization



From Bishop's [Pattern recognition and machine learning](#), Figure 9.8.

# EM for GMM

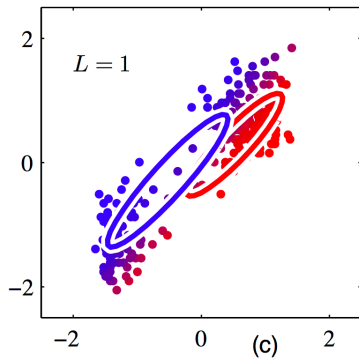
• First soft assignment:



From Bishop's [Pattern recognition and machine learning](#), Figure 9.8.

# EM for GMM

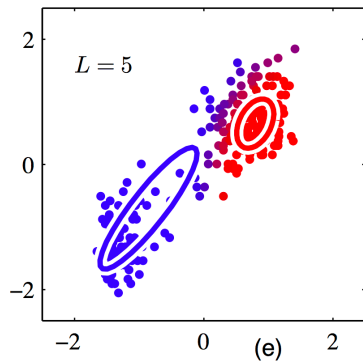
- First soft assignment:



From Bishop's [Pattern recognition and machine learning](#), Figure 9.8.

# EM for GMM

- After 5 rounds of EM:

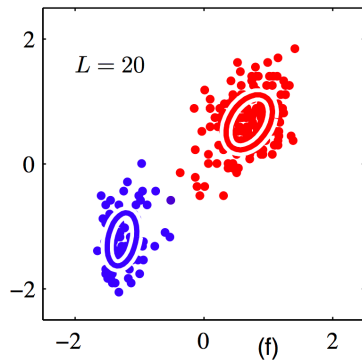


From Bishop's [Pattern recognition and machine learning](#), Figure 9.8.



# 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.