COMS 4721: Machine Learning for Data Science
Lecture 16, 3/28/2017

Prof. John Paisley

Department of Electrical Engineering
& Data Science Institute

Columbia University

SOFT CLUSTERING VS HARD CLUSTERING MODELS

Review: K-means clustering algorithm

Given: Data $x_1, \dots, x_n$, where $x \in \mathbb{R}^d$

Goal: Minimize $\mathcal{L} = \sum_{i=1}^n \sum_{k=1}^K \mathbb{1}\{c_i = k\} \|x_i - \mu_k\|^2$.

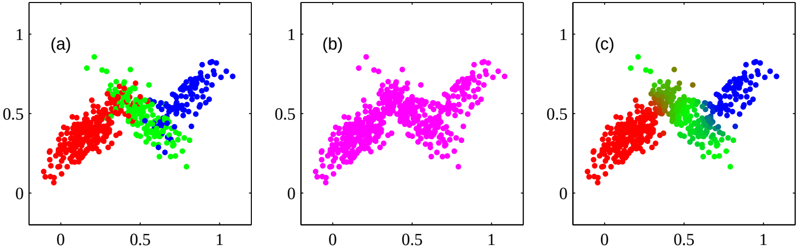
- Iterate until values no longer changing
  1. Update $\boldsymbol{c}$: For each i, set $c_i = \arg\min_k \|x_i - \mu_k\|^2$
  2. Update $\boldsymbol{\mu}$: For each k, set $\mu_k = \left(\sum_i x_i \mathbb{1}\{c_i = k\}\right) / \left(\sum_i \mathbb{1}\{c_i = k\}\right)$

K-means is an example of a *hard clustering* algorithm because it assigns each observation to only one cluster.

In other words, $c_i = k$ for some $k \in \{1, \dots, K\}$. There is no accounting for the "boundary cases" by hedging on the corresponding c_i.

SOFT CLUSTERING MODELS

A *soft clustering* algorithm breaks the data across clusters intelligently.



(left) True cluster assignments of data from three Gaussians.

(middle) The data as we see it.

(right) A soft-clustering of the data accounting for borderline cases.

WEIGHTED K-MEANS (SOFT CLUSTERING EXAMPLE)

Weighted K-means clustering algorithm

Given: Data $x_1, \ldots, x_n$, where $x \in \mathbb{R}^d$

Goal: Minimize $\mathcal{L} = \sum_{i=1}^{n} \sum_{k=1}^{K} \phi_i(k) \frac{\|x_i - \mu_k\|^2}{\beta} - \sum_i \mathcal{H}(\phi_i)$ over ϕ_i and μ_k

Conditions: $\phi_i(k) > 0, \sum_{k=1}^{K} \phi_i(k) = 1$, $\mathcal{H}(\phi_i) =$ entropy. Set $\beta > 0$.

- Iterate the following

  1. Update $\boldsymbol{\phi}$: For each i, update the cluster allocation weights

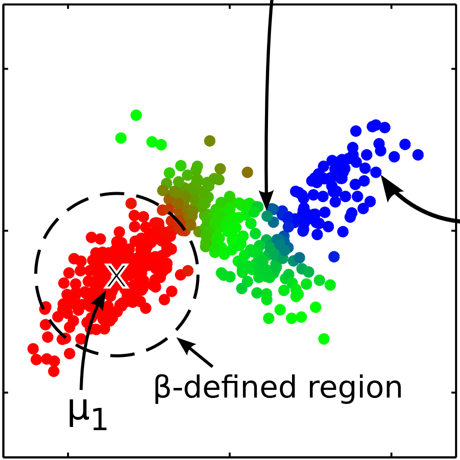
  $$\phi_i(k) = \frac{\exp\{-\frac{1}{\beta}\|x_i - \mu_k\|^2\}}{\sum_j \exp\{-\frac{1}{\beta}\|x_i - \mu_j\|^2\}}, \quad \text{for } k = 1, \ldots, K$$

  2. Update $\boldsymbol{\mu}$: For each k, update μ_k with the *weighted* average

  $$\mu_k = \frac{\sum_i x_i \phi_i(k)}{\sum_i \phi_i(k)}$$

Soft clustering with weighted k-means

ϕ_i = 0.75 on green cluster & 0.25 blue cluster



ϕ_i = 1 on blue cluster

When ϕ_i is binary, we get back the hard clustering model

Mixture models

Probabilistic soft clustering models

Probabilistic vs non-probabilistic soft clustering

The weight vector ϕ_i is *like* a probability of x_i being assigned to each cluster.

A **mixture model** is a probabilistic model where ϕ_i actually *is* a probability distribution according to the model.

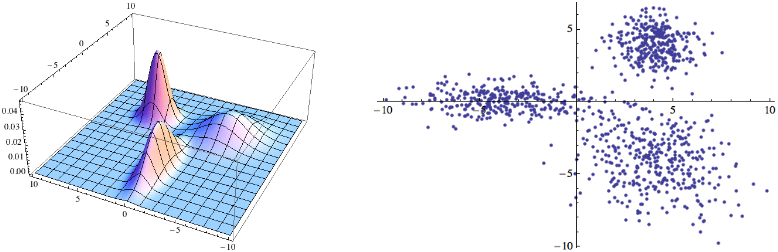
Mixture models work by defining:

- A prior distribution on the cluster assignment indicator c_i
- A likelihood distribution on observation x_i given the assignment c_i

Intuitively we can connect a mixture model to the Bayes classifier:

- Class prior $\rightarrow$ cluster prior. This time, we *don't* know the "label"
- Class-conditional likelihood $\rightarrow$ cluster-conditional likelihood

MIXTURE MODELS



(a) A probability distribution on $\mathbb{R}^2$. (b) Data sampled from this distribution.

Before introducing math, some key features of a mixture model are:

1. It is a generative model (defines a probability distribution on the data)
2. It is a weighted combination of simpler distributions.
   - Each simple distribution is in the same distribution family (i.e., a Gaussian).
   - The "weighting" is defined by a discrete probability distribution.

Generating data from a mixture model

Data: $x_1, \ldots, x_n$, where each $x_i \in \mathcal{X}$ (can be complicated, but think $\mathcal{X} = \mathbb{R}^d$)

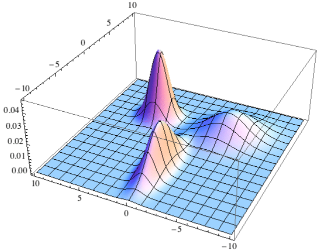
Model parameters: A K-dim distribution π and parameters $\theta_1, \ldots, \theta_K$.

Generative process: For observation number $i = 1, \ldots, n$,

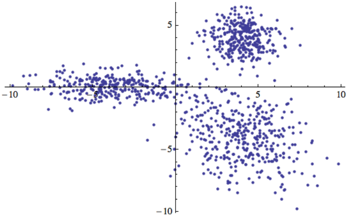
1. Generate cluster assignment: $c_i \overset{iid}{\sim} \text{Discrete}(\pi) \Rightarrow \text{Prob}(c_i = k|\pi) = \pi_k$.
2. Generate observation: $x_i \sim p(x|\theta_{c_i})$.

Some observations about this procedure:

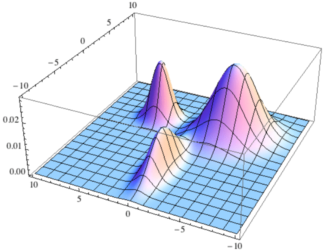
- First, each x_i is randomly assigned to a cluster using distribution π.
- c_i indexes the cluster assignment for x_i
  - This picks out the index of the parameter θ used to generate x_i.
  - If two x's share a parameter, they are clustered together.



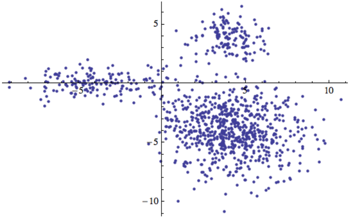
(a) Uniform mixing weights



(b) Data sampled from this distribution.



(c) Uneven mixing weights



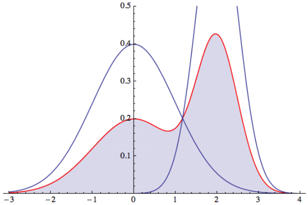
(d) Data sampled from this distribution.

GAUSSIAN MIXTURE MODELS

ILLUSTRATION

Gaussian mixture models are mixture models where $p(x|\theta)$ is Gaussian.

Mixture of two Gaussians

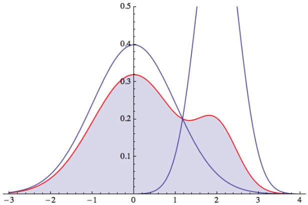


The red line is the density function.

$$\pi = [0.5,\ 0.5]$$

$$\begin{aligned}(\mu_1, \sigma_1^2) &= (0, 1)\\ (\mu_2, \sigma_2^2) &= (2, 0.5)\end{aligned}$$

Influence of mixing weights



The red line is the density function.

$$\pi = [0.8,\ 0.2]$$

$$\begin{aligned}(\mu_1, \sigma_1^2) &= (0, 1)\\ (\mu_2, \sigma_2^2) &= (2, 0.5)\end{aligned}$$

GAUSSIAN MIXTURE MODELS (GMM)

The model

Parameters: Let π be a K-dimensional probability distribution and (μ_k, Σ_k) be the mean and covariance of the kth Gaussian in $\mathbb{R}^d$.

Generate data: For the ith observation,

1. Assign the ith observation to a cluster, $c_i \sim \text{Discrete}(\pi)$
2. Generate the value of the observation, $x_i \sim N(\mu_{c_i}, \Sigma_{c_i})$

Definitions: $\boldsymbol{\mu} = \{\mu_1, \ldots, \mu_K\}$ and $\boldsymbol{\Sigma} = \{\Sigma_1, \ldots, \Sigma_k\}$.

Goal: We want to learn π, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$.

Maximum likelihood

Objective: Maximize the likelihood over model parameters π, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ by treating the c_i as auxiliary data using the EM algorithm.

$$p(x_1, \ldots, x_n | \pi, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \prod_{i=1}^{n} p(x_i | \pi, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \prod_{i=1}^{n} \sum_{k=1}^{K} p(x_i, c_i = k | \pi, \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

The summation over values of each c_i "integrates out" this variable.

We can't simply take derivatives with respect to π, μ_k and Σ_k and set to zero to maximize this because there's no closed form solution.

We could use gradient methods, but EM is cleaner.

Q: Why not instead just include each c_i and maximize $\prod_{i=1}^n p(x_i, c_i|\pi, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ since (we can show) this is easy to do using coordinate ascent?

A: We would end up with a hard-clustering model where $c_i \in \{1, \ldots, K\}$. Our goal here is to have soft clustering, which EM does.

EM and the GMM

We will not derive everything from scratch. However, we can treat $c_1, \ldots, c_n$ as the auxiliary data that we integrate out.

Therefore, we use EM to

$$\text{maximize} \quad \sum_{i=1}^n \ln p(x_i|\pi, \boldsymbol{\mu}, \boldsymbol{\Sigma}) \quad \text{by using} \quad \sum_{i=1}^n \ln p(x_i, c_i|\pi, \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Let's look at the outlines of how to derive this.

THE EM ALGORITHM AND THE GMM

From the last lecture, the generic EM objective is

$$\ln p(x|\theta_1) = \int q(\theta_2) \ln \frac{p(x, \theta_2|\theta_1)}{q(\theta_2)}\, d\theta_2 \; + \; \int q(\theta_2) \ln \frac{q(\theta_2)}{p(\theta_2|x, \theta_1)}\, d\theta_2$$

The EM objective for the Gaussian mixture model is

$$\begin{aligned}\sum_{i=1}^{n} \ln p(x_i|\pi, \boldsymbol{\mu}, \boldsymbol{\Sigma}) \quad &= \quad \sum_{i=1}^{n} \sum_{k=1}^{K} q(c_i = k) \ln \frac{p(x_i, c_i = k|\pi, \boldsymbol{\mu}, \boldsymbol{\Sigma})}{q(c_i = k)} \quad + \\ & \quad\quad \sum_{i=1}^{n} \sum_{k=1}^{K} q(c_i = k) \ln \frac{q(c_i = k)}{p(c_i|x_i, \pi, \boldsymbol{\mu}, \boldsymbol{\Sigma})}\end{aligned}$$

Because c_i is discrete, the integral becomes a sum.

EM SETUP (ONE ITERATION)

First: Set $q(c_i = k) \;\; \Leftarrow \;\; p(c_i = k|x_i, \pi, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ using Bayes rule:

$$p(c_i = k|x_i, \pi, \boldsymbol{\mu}, \boldsymbol{\Sigma}) \propto p(c_i = k|\pi)p(x_i|c_i = k, \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

We can solve the posterior of c_i given π, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$:

$$q(c_i = k) = \frac{\pi_k N(x_i|\mu_k, \Sigma_k)}{\sum_j \pi_j N(x_i|\mu_j, \Sigma_j)} \quad \Longrightarrow \quad \phi_i(k)$$

E-step: Take the expectation using the updated q's

$$\mathcal{L} = \sum_{i=1}^{n} \sum_{k=1}^{K} \phi_i(k) \ln p(x_i, c_i = k|\pi, \mu_k, \Sigma_k) \; + \; \text{constant w.r.t. } \pi, \boldsymbol{\mu}, \boldsymbol{\Sigma}$$

M-step: Maximize $\mathcal{L}$ with respect to π and each μ_k, Σ_k.

M-STEP CLOSE UP

Aside: How has EM made this easier?

Original objective function:

$$\mathcal{L} = \sum_{i=1}^{n} \ln \sum_{k=1}^{K} p(x_i, c_i = k|\pi, \mu_k, \Sigma_k) = \sum_{i=1}^{n} \ln \sum_{k=1}^{K} \pi_k N(x_i|\mu_k, \Sigma_k).$$

The log-sum form makes optimizing π, and each μ_k and Σ_k difficult.

Using EM here, we have the M-Step:

$$\mathcal{L} = \sum_{i=1}^{n} \sum_{k=1}^{K} \phi_i(k) \underbrace{\{\ln \pi_k + \ln N(x_i|\mu_k, \Sigma_k)\}}_{\ln p(x_i, c_i = k|\pi, \mu_k, \Sigma_k)} + \text{constant w.r.t. } \pi, \boldsymbol{\mu}, \boldsymbol{\Sigma}$$

The sum-log form is easier to optimize. We can take derivatives and solve.

Algorithm: Maximum likelihood EM for the GMM

Given: $x_1, \dots, x_n$ where $x \in \mathbb{R}^d$

Goal: Maximize $\mathcal{L} = \sum_{i=1}^{n} \ln p(x_i|\pi, \boldsymbol{\mu}, \boldsymbol{\Sigma})$.

- Iterate until incremental improvement to $\mathcal{L}$ is "small"

1. **E-step**: For $i = 1, \dots, n$, set

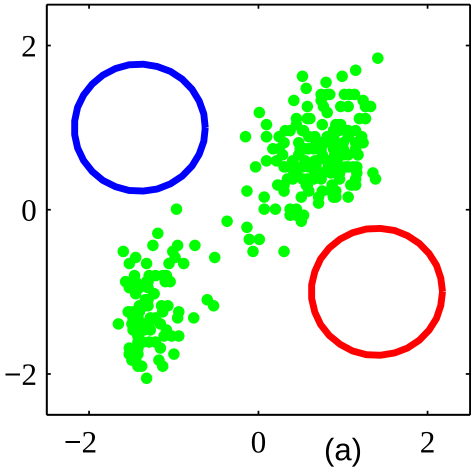
$$\phi_i(k) = \frac{\pi_k N(x_i|\mu_k, \Sigma_k)}{\sum_j \pi_j N(x_i|\mu_j, \Sigma_j)}, \quad \text{for } k = 1, \dots, K$$

2. **M-step**: For $k = 1, \dots, K$, define $n_k = \sum_{i=1}^{n} \phi_i(k)$ and update the values

$$\pi_k = \frac{n_k}{n}, \quad \mu_k = \frac{1}{n_k} \sum_{i=1}^{n} \phi_i(k) x_i \quad \Sigma_k = \frac{1}{n_k} \sum_{i=1}^{n} \phi_i(k)(x_i - \mu_k)(x_i - \mu_k)^T$$

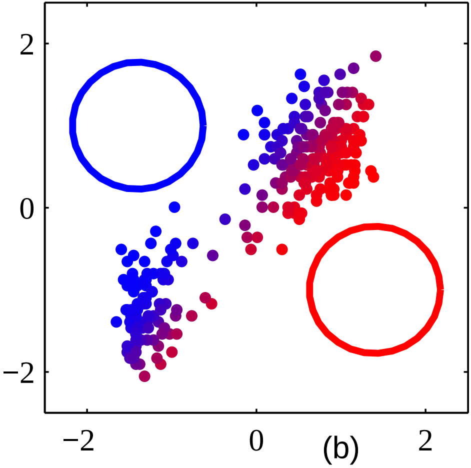
Comment: The updated value for μ_k is used when updating Σ_k.

Gaussian mixture model: Example run



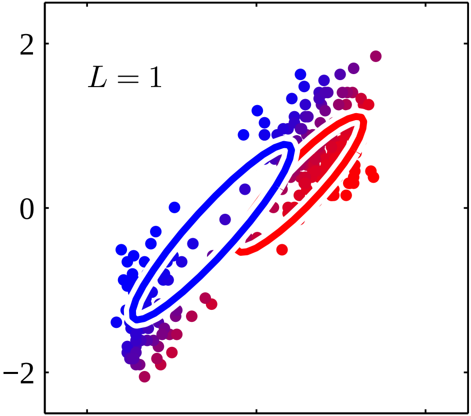
A random initialization

Gaussian mixture model: Example run



Iteration 1 (E-step)

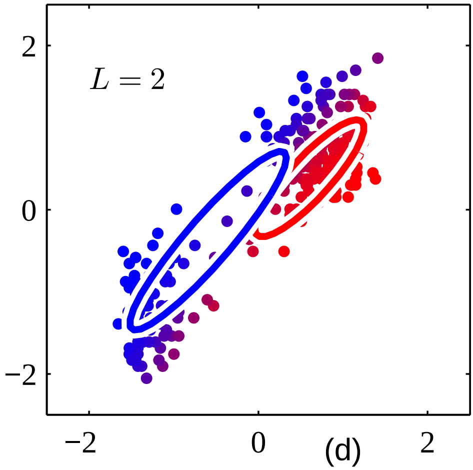
Assign data to clusters



(c)

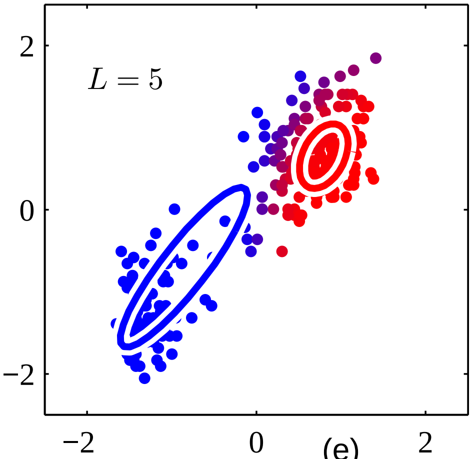
Iteration 1 (M-step)

Update the Gaussians



Iteration 2

Assign data to clusters and update the Gaussians

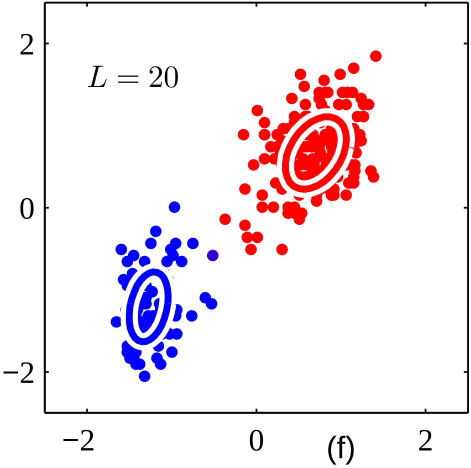


(e)

Iteration 5 (skipping ahead)

Assign data to clusters
and update the Gaussians

GAUSSIAN MIXTURE MODEL: EXAMPLE RUN



Iteration 20 (convergence)

Assign data to clusters
and update the Gaussians

GMM AND THE BAYES CLASSIFIER

The GMM feels a lot like a K-class Bayes classifier, where the label of x_i is

$$\text{label}(x_i) = \arg\max_k \; \pi_k N(x_i|\mu_k, \Sigma_k).$$

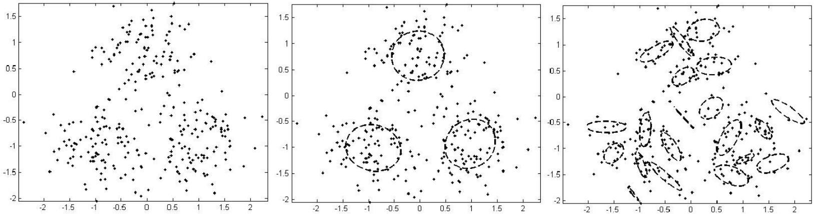
- π_k = class prior, and $N(\mu_k, \Sigma_k)$ = class-conditional density function.
- We learned π, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ using maximum likelihood there too.

For the Bayes classifier, we could find π, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ with a single equation because the class label was *known*. Compare with the GMM update:

$$\pi_k = \frac{n_k}{n}, \quad \mu_k = \frac{1}{n_k}\sum_{i=1}^{n} \phi_i(k)x_i \quad \Sigma_k = \frac{1}{n_k}\sum_{i=1}^{n} \phi_i(k)(x_i - \mu_k)(x_i - \mu_k)^T$$

They're almost identical. But since $\phi_i(k)$ is changing we have to update these values. With the Bayes classifier, "ϕ_i" encodes the label, so it was known.

CHOOSING THE NUMBER OF CLUSTERS



Maximum likelihood for the Gaussian mixture model can overfit the data. It will learn as many Gaussians as it's given.

There are a set of techniques for this based on the Dirichlet distribution.

A Dirichlet prior is used on π which encourages many Gaussians to disappear (i.e., not have any data assigned to them).

EM FOR A GENERIC MIXTURE MODEL

Algorithm: Maximum likelihood EM for mixture models

Given: Data $x_1, \dots, x_n$ where $x \in \mathcal{X}$

Goal: Maximize $\mathcal{L} = \sum_{i=1}^{n} \ln p(x_i|\pi, \boldsymbol{\theta})$, where $p(x|\theta_k)$ is problem-specific.

- Iterate until incremental improvement to $\mathcal{L}$ is "small"

  1. **E-step**: For $i = 1, \dots, n$, set

  $$\phi_i(k) = \frac{\pi_k\, p(x_i|\theta_k)}{\sum_j \pi_j\, p(x_i|\theta_j)}, \quad \text{for } k = 1, \dots, K$$

  2. **M-step**: For $k = 1, \dots, K$, define $n_k = \sum_{i=1}^{n} \phi_i(k)$ and set

  $$\pi_k = \frac{n_k}{n}, \qquad \theta_k = \arg\max_{\theta} \sum_{i=1}^{n} \phi_i(k) \ln p(x_i|\theta)$$

Comment: Similar to generalization of the Bayes classifier for any $p(x|\theta_k)$.