ColumbiaX: Machine Learning

Lecture 15

Prof. John Paisley

Department of Electrical Engineering
& Data Science Institute

Columbia University

Maximum likelihood

Approaches to data modeling

Our approaches to modeling data thus far have been either probabilistic or non-probabilistic in motivation.

- Probabilistic models: Probability distributions defined on data, e.g.,
  1. Bayes classifiers
  2. Logistic regression
  3. Least squares and ridge regression (using ML and MAP interpretation)
  4. Bayesian linear regression
- Non-probabilistic models: No probability distributions involved, e.g.,
  1. Perceptron
  2. Support vector machine
  3. Decision trees
  4. K-means

In *every* case, we have some objective function we are trying to optimize (greedily vs non-greedily, locally vs globally).

MAXIMUM LIKELIHOOD

As we've seen, one *probabilistic* objective function is maximum likelihood.

Setup: In the most basic scenario, we start with

1. some set of model parameters θ
2. a set of data $\{x_1, \ldots, x_n\}$
3. a probability distribution $p(x|\theta)$
4. an i.i.d. assumption, $x_i \overset{iid}{\sim} p(x|\theta)$

Maximum likelihood seeks to find the θ that maximizes the likelihood

$$\theta_{\text{ML}} = \arg\max_{\theta} \; p(x_1, \ldots, x_n|\theta) \overset{(a)}{=} \arg\max_{\theta} \prod_{i=1}^{n} p(x_i|\theta) \overset{(b)}{=} \arg\max_{\theta} \sum_{i=1}^{n} \ln p(x_i|\theta)$$

(a) follows from i.i.d. assumption.
(b) follows since $f(y) > f(x) \;\Rightarrow\; \ln f(y) > \ln f(x)$.

MAXIMUM LIKELIHOOD

We've discussed maximum likelihood for a few models, e.g., least squares linear regression and the Bayes classifier.

Both of these models were "nice" because we could find their respective θ_{ML} analytically by writing an equation and plugging in data to solve.

Gaussian with unknown mean and covariance

In the first lecture, we saw if $x_i \overset{iid}{\sim} N(\mu, \Sigma)$, where $\theta = \{\mu, \Sigma\}$, then

$$\nabla_\theta \ln \prod_{i=1}^{n} p(x_i|\theta) = 0$$

gives the following maximum likelihood values for μ and Σ:

$$\mu_{\text{ML}} = \frac{1}{n}\sum_{i=1}^{n} x_i, \qquad \Sigma_{\text{ML}} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_{\text{ML}})(x_i - \mu_{\text{ML}})^T$$

Coordinate ascent and maximum likelihood

In more complicated models, we might split the parameters into groups θ_1, θ_2 and try to maximize the likelihood over both of these,

$$\theta_{1,\text{ML}}, \theta_{2,\text{ML}} = \arg\max_{\theta_1,\theta_2} \sum_{i=1}^{n} \ln p(x_i|\theta_1, \theta_2),$$

Although we can solve one *given* the other, we can't solve it *simultaneously*.

Coordinate ascent (probabilistic version)

We saw how K-means presented a similar situation, and that we could optimize using coordinate ascent. This technique is generalizable.

Algorithm: For iteration $t = 1, 2, \ldots,$

1. Optimize $\theta_1^{(t)} = \arg\max_{\theta_1} \sum_{i=1}^{n} \ln p(x_i|\theta_1, \theta_2^{(t-1)})$
2. Optimize $\theta_2^{(t)} = \arg\max_{\theta_2} \sum_{i=1}^{n} \ln p(x_i|\theta_1^{(t)}, \theta_2)$

COORDINATE ASCENT AND MAXIMUM LIKELIHOOD

There is a third (subtly) different situation, where we really want to find

$$\theta_{1,\mathrm{ML}} = \arg\max_{\theta_1} \sum_{i=1}^{n} \ln p(x_i|\theta_1).$$

Except this function is "tricky" to optimize directly. However, we figure out that we can add a second variable θ_2 such that

$$\sum_{i=1}^{n} \ln p(x_i, \theta_2|\theta_1) \qquad \text{(Function 2)}$$

is easier to work with. We'll make this clearer later.

- Notice in this second case that θ_2 is on the *left* side of the conditioning bar. This implies a prior on θ_2, (whatever "θ_2" turns out to be).
- We will next discuss a fundamental technique called the EM algorithm for finding $\theta_{1,\mathrm{ML}}$ by using Function 2 instead.

Expectation-Maximization Algorithm

A MOTIVATING EXAMPLE

Let $x_i \in \mathbb{R}^d$, be a vector with *missing data*. Split this vector into two parts:

1. x_i^o – observed portion (the sub-vector of x_i that is measured)
2. x_i^m – missing portion (the sub-vector of x_i that is still unknown)
3. The missing dimensions can be different for different x_i.

We assume that $x_i \overset{iid}{\sim} N(\mu, \Sigma)$, and want to solve

$$\mu_{\text{ML}}, \Sigma_{\text{ML}} = \arg\max_{\mu,\Sigma} \sum_{i=1}^{n} \ln p(x_i^o | \mu, \Sigma).$$

This is tricky. However, if we knew x_i^m (and therefore x_i), then

$$\mu_{\text{ML}}, \Sigma_{\text{ML}} = \arg\max_{\mu,\Sigma} \sum_{i=1}^{n} \ln \underbrace{p(x_i^o, x_i^m | \mu, \Sigma)}_{= p(x_i | \mu, \Sigma)}$$

is very easy to optimize (we just did it on a previous slide).

CONNECTING TO A MORE GENERAL SETUP

We will discuss a method for optimizing $\sum_{i=1}^{n} \ln p(x_i^o|\mu,\Sigma)$ and imputing its missing values $\{x_1^m,\ldots,x_n^m\}$. This is a very general technique.

General setup

Imagine we have two parameter sets θ_1,θ_2, where

$$p(x|\theta_1) = \int p(x,\theta_2|\theta_1)\,d\theta_2 \qquad \text{(marginal distribution)}$$

Example: For the previous example we can show that

$$p(x_i^o|\mu,\Sigma) = \int p(x_i^o,x_i^m|\mu,\Sigma)\,dx_i^m = N(\mu_i^o,\Sigma_i^o),$$

where μ_i^o and Σ_i^o are the sub-vector/sub-matrix of μ and Σ defined by x_i^o.

We need to define a general *objective function* that gives us what we want:

1. It lets us optimize the marginal $p(x|\theta_1)$ over θ_1,
2. It uses $p(x, \theta_2|\theta_1)$ in doing so purely for computational convenience.

The EM objective function

Before picking it apart, we claim that this objective function is

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

Some immediate comments:

- $q(\theta_2)$ is *any* probability distribution (assumed continuous for now)
- We assume we know $p(\theta_2|x, \theta_1)$. That is, given the data x and fixed values for θ_1, we can solve the conditional posterior distribution of θ_2.

Deriving the EM objective function

Let's show that this equality is actually true

$$\begin{aligned}\ln p(x|\theta_1) &= \int q(\theta_2) \ln \frac{p(x,\theta_2|\theta_1)}{q(\theta_2)}\, d\theta_2 + \int q(\theta_2) \ln \frac{q(\theta_2)}{p(\theta_2|x,\theta_1)}\, d\theta_2 \\ &= \int q(\theta_2) \ln \frac{p(x,\theta_2|\theta_1) q(\theta_2)}{p(\theta_2|x,\theta_1) q(\theta_2)}\, d\theta_2\end{aligned}$$

Remember some rules of probability:

$$p(a,b|c) = p(a|b,c)p(b|c) \quad \Rightarrow \quad p(b|c) = \frac{p(a,b|c)}{p(a|b,c)}.$$

Letting $a = \theta_1$, $b = x$ and $c = \theta_1$, we conclude

$$\begin{aligned}\ln p(x|\theta_1) &= \int q(\theta_2) \ln p(x|\theta_1)\, d\theta_2 \\ &= \ln p(x|\theta_1)\end{aligned}$$

THE EM OBJECTIVE FUNCTION

The EM objective function splits our desired objective into two terms:

$$\ln p(x|\theta_1) = \underbrace{\int q(\theta_2) \ln \frac{p(x,\theta_2|\theta_1)}{q(\theta_2)}\, d\theta_2}_{\text{A function only of } \theta_1\text{, we'll call it } \mathcal{L}} + \underbrace{\int q(\theta_2) \ln \frac{q(\theta_2)}{p(\theta_2|x,\theta_1)}\, d\theta_2}_{\text{Kullback-Leibler divergence}}$$

Some more observations about the right hand side:

1. The **KL diverence** is always ≥ 0 and only $= 0$ when $q = p$.
2. We are assuming that the integral in $\mathcal{L}$ can be calculated, leaving a function only of θ_1 (for a particular setting of the distribution q).

BIGGER PICTURE

Q: What does it mean to iteratively optimize $\ln p(x|\theta_1)$ w.r.t. θ_1?

A: One way to think about it is that we want a method for generating:

1. A sequence of values for θ_1 such that $\ln p(x|\theta_1^{(t)}) \geq \ln p(x|\theta_1^{(t-1)})$.
2. We want $\theta_1^{(t)}$ to converge to a local maximum of $\ln p(x|\theta_1)$.

It doesn't matter how we generate the sequence $\theta_1^{(1)}, \theta_1^{(2)}, \theta_1^{(3)}, \ldots$

We will show how EM generates #1 and just mention that EM satisfies #2.

The EM objective function

$$\ln p(x|\theta_1) = \underbrace{\int q(\theta_2) \ln \frac{p(x,\theta_2|\theta_1)}{q(\theta_2)}\, d\theta_2}_{\text{define this to be } \mathcal{L}(x,\theta_1)} + \underbrace{\int q(\theta_2) \ln \frac{q(\theta_2)}{p(\theta_2|x,\theta_1)}\, d\theta_2}_{\text{Kullback-Leibler divergence}}$$

Definition: The EM algorithm

Given the value $\theta_1^{(t)}$, **find** the value $\theta_1^{(t+1)}$ as follows:

E-step: Set $q_t(\theta_2) = p(\theta_2|x,\theta_1^{(t)})$ and calculate

$$\mathcal{L}_t(x,\theta_1) = \int q_t(\theta_2) \ln p(x,\theta_2|\theta_1)\, d\theta_2 - \underbrace{\int q_t(\theta_2) \ln q_t(\theta_2)\, d\theta_2}_{\text{can ignore this term}}.$$

M-step: Set $\theta_1^{(t+1)} = \arg\max_{\theta_1} \mathcal{L}_t(x,\theta_1)$.

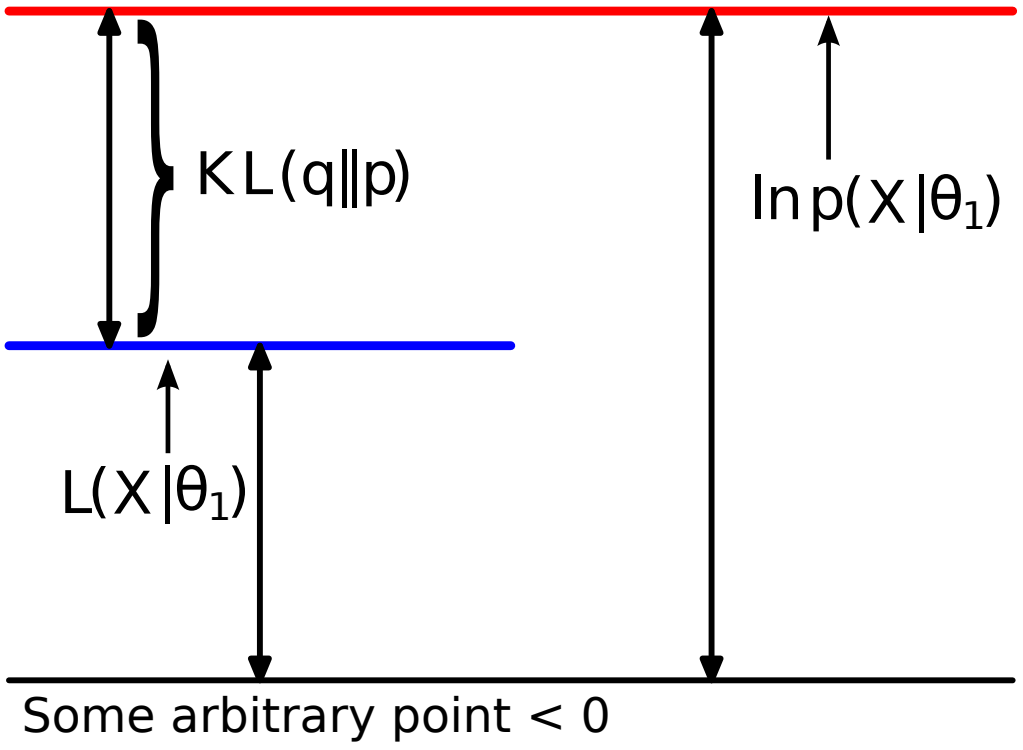
PROOF OF MONOTONIC IMPROVEMENT

Once we're comfortable with the moving parts, the proof that the sequence $\theta_1^{(t)}$ monotonically improves $\ln p(x|\theta_1)$ just requires *analysis*:

$$
\begin{aligned}
\ln p(x|\theta_1^{(t)}) &= \mathcal{L}(x, \theta_1^{(t)}) + \underbrace{KL\left(q(\theta_2) \,\|\, p(\theta_2|x_1, \theta_1^{(t)})\right)}_{=0 \text{ by setting } q=p} \\
&= \mathcal{L}_t(x, \theta_1^{(t)}) \qquad \leftarrow \text{E-step} \\
&\leq \mathcal{L}_t(x, \theta_1^{(t+1)}) \qquad \leftarrow \text{M-step} \\
&\leq \mathcal{L}_t(x, \theta_1^{(t+1)}) + \underbrace{KL\left(q_t(\theta_2) \,\|\, p(\theta_2|x_1, \theta_1^{(t+1)})\right)}_{>0 \text{ because } q \neq p} \\
&= \mathcal{L}(x, \theta_1^{(t+1)}) + KL\left(q(\theta_2) \,\|\, p(\theta_2|x_1, \theta_1^{(t+1)})\right) \\
&= \ln p(x|\theta_1^{(t+1)})
\end{aligned}
$$

One iteration of EM

Start: Current setting of θ_1 and $q(\theta_2)$



For reference:

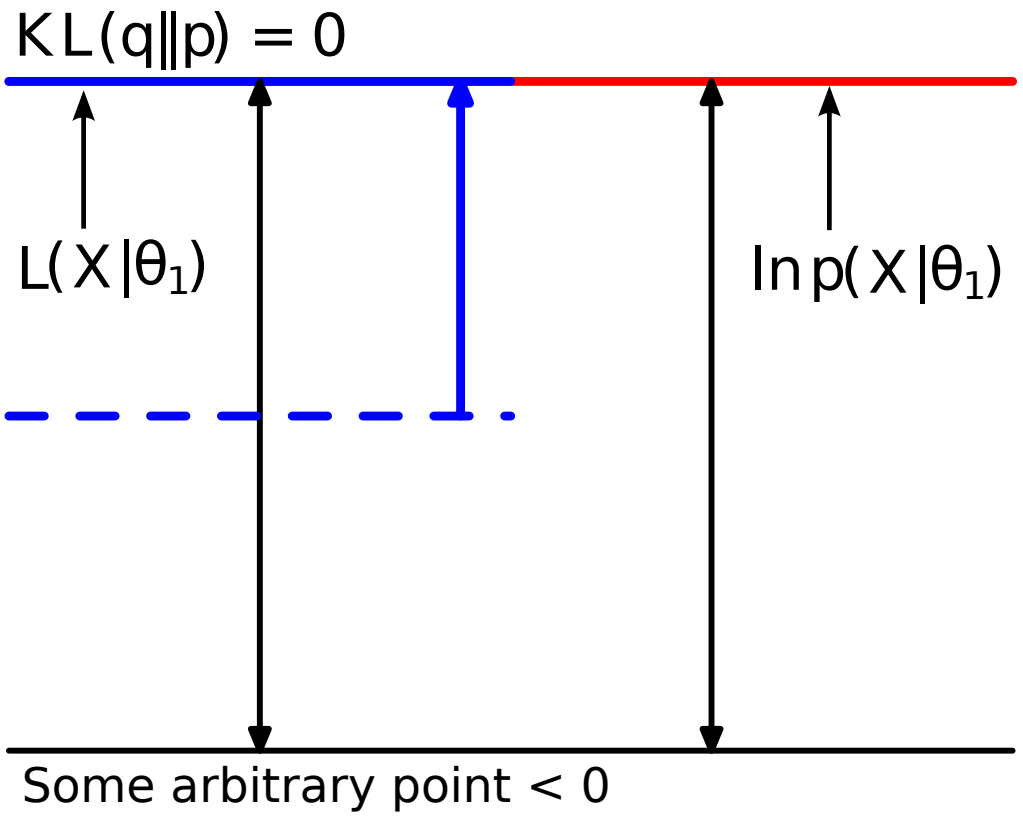
$$\ln p(x|\theta_1) = \mathcal{L} + KL$$

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

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

One iteration of EM

E-step: Set $q(\theta_2) = p(\theta_2|x, \theta_1)$ and update $\mathcal{L}$.



For reference:

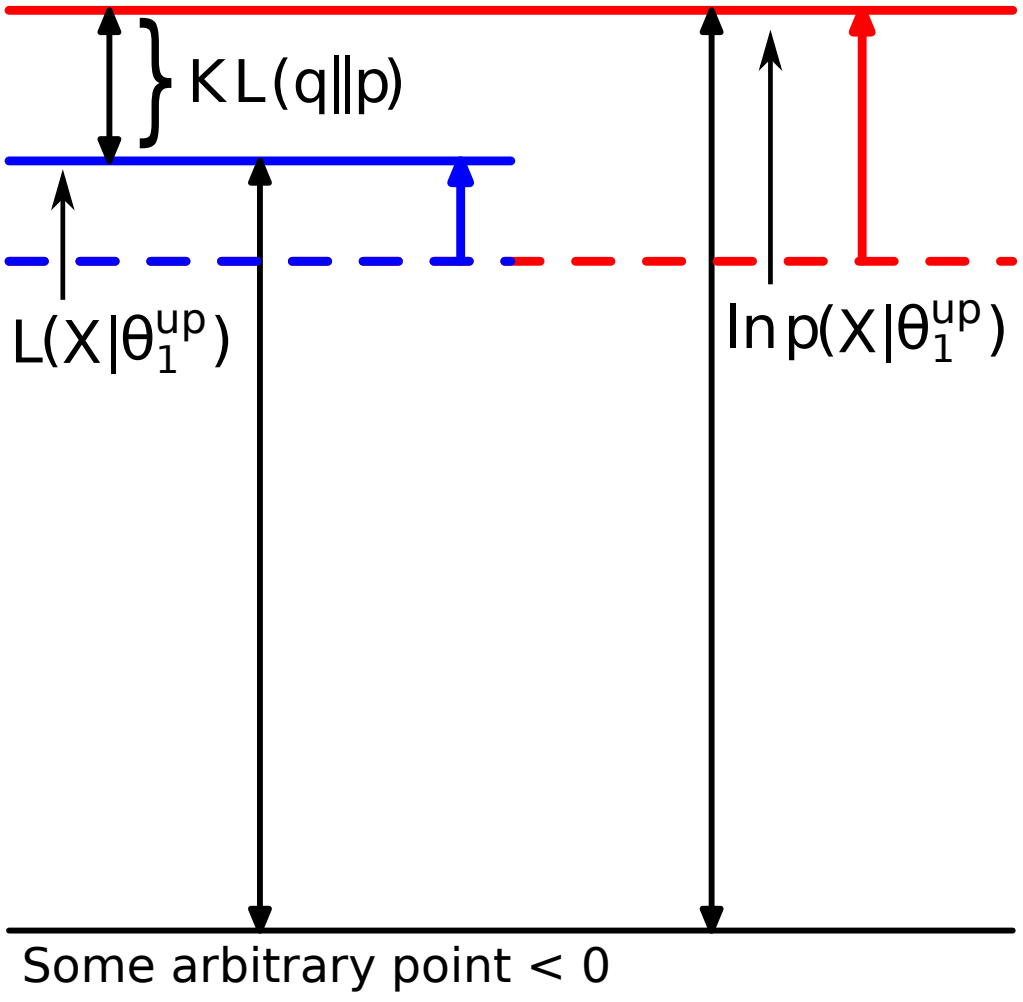
$$\ln p(x|\theta_1) = \mathcal{L} + KL$$

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

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

One iteration of EM

M-step: Maximize $\mathcal{L}$ wrt θ_1. Now $q \neq p$.



For reference:

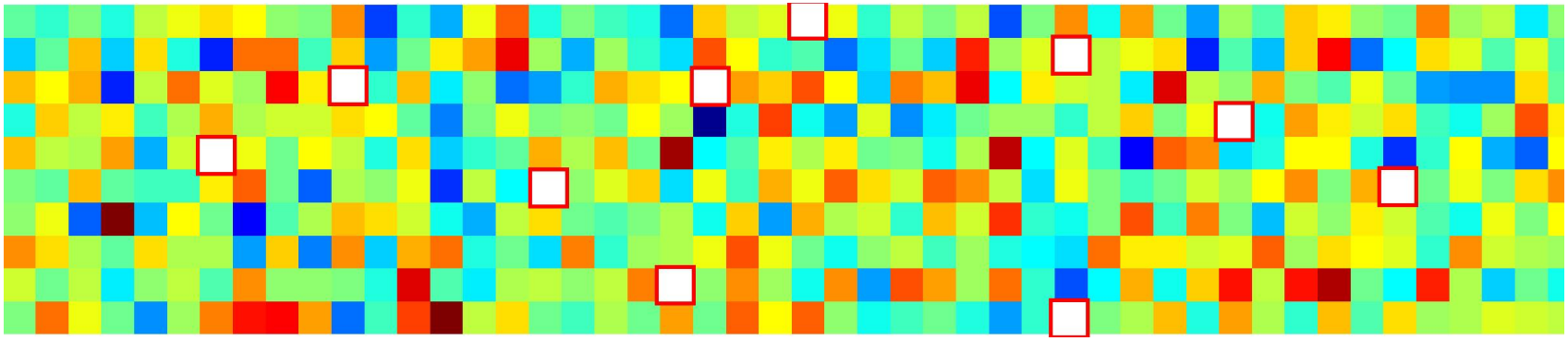
$$\ln p(x|\theta_1) = \mathcal{L} + KL$$

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

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

EM FOR MISSING DATA

THE PROBLEM



We have a data matrix with missing entries. We model the columns as

$$x_i \overset{iid}{\sim} N(\mu, \Sigma).$$

Our goal could be to

1. Learn μ and Σ using maximum likelihood
2. Fill in the missing values "intelligently" (e.g., using a model)
3. Both

We will see how to achieve both of these goals using the EM algorithm.

EM FOR SINGLE GAUSSIAN MODEL WITH MISSING DATA

The original, 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 this specific problem and notation is

$$\begin{aligned} \sum_{i=1}^{n} \ln p(x_i^o|\mu, \Sigma) \quad &= \quad \sum_{i=1}^{n} \int q(x_i^m) \ln \frac{p(x_i^o, x_i^m|\mu, \Sigma)}{q(x_i^m)} \, dx_i^m \quad + \\ & \qquad \sum_{i=1}^{n} \int q(x_i^m) \ln \frac{q(x_i^m)}{p(x_i^m|x_i^o, \mu, \Sigma)} \, dx_i^m \end{aligned}$$

We can calculate everything required to do this.

E-STEP (PART ONE)

Set $q(x_i^m) = p(x_i^m | x_i^o, \mu, \Sigma)$ using current μ, Σ

Let x_i^o and x_i^m represent the observed and missing dimensions of x_i. For notational convenience, think

$$x_i = \begin{bmatrix} x_i^o \\ x_i^m \end{bmatrix} \sim N\left(\begin{bmatrix} \mu_i^o \\ \mu_i^m \end{bmatrix}, \begin{bmatrix} \Sigma_i^{oo} & \Sigma_i^{om} \\ \Sigma_i^{mo} & \Sigma_i^{mm} \end{bmatrix} \right)$$

Then we can show that $p(x_i^m | x_i^o, \mu, \Sigma) = N(\widehat{\mu}_i, \widehat{\Sigma}_i)$, where

$$\widehat{\mu}_i = \mu_i^m + \Sigma_i^{mo}(\Sigma_i^{oo})^{-1}(x_i^o - \mu_i^o), \quad \widehat{\Sigma}_i = \Sigma_i^{mm} - \Sigma_i^{mo}(\Sigma_i^{oo})^{-1}\Sigma_i^{om}.$$

It doesn't look nice, but these are just functions of sub-vectors of μ and sub-matrices of Σ using the relevant dimensions defined by x_i.

E-STEP (PART TWO)

E-step: $\mathbb{E}_{q(x_i^m)}[\ln p(x_i^o, x_i^m|\mu, \Sigma)]$

For each i we will need to calculate the following term,

$$\begin{aligned}\mathbb{E}_q[(x_i-\mu)^T\Sigma^{-1}(x_i-\mu)] &= \mathbb{E}_q[\text{trace}\{\Sigma^{-1}(x_i-\mu)(x_i-\mu)^T\}] \\ &= \text{trace}\{\Sigma^{-1}\mathbb{E}_q[(x_i-\mu)(x_i-\mu)^T]\}\end{aligned}$$

The expectation is calculated using $q(x_i^m) = p(x_i^m|x_i^o, \mu, \Sigma)$. So only the x_i^m portion of x_i will be integrated.

To this end, recall $q(x_i^m) = N(\widehat{\mu}_i, \widehat{\Sigma}_i)$. We define

1. $\widehat{x}_i$: A vector where we replace the missing values in x_i with $\widehat{\mu}_i$.
2. $\widehat{V}_i$: A matrix of 0's, plus sub-matrix $\widehat{\Sigma}_i$ in the missing dimensions.

M-step: Maximize $\sum_{i=1}^{n} \mathbb{E}_q[\ln p(x_i^o, x_i^m|\mu, \Sigma)]$

We'll omit the derivation, but the expectation can now be solved and

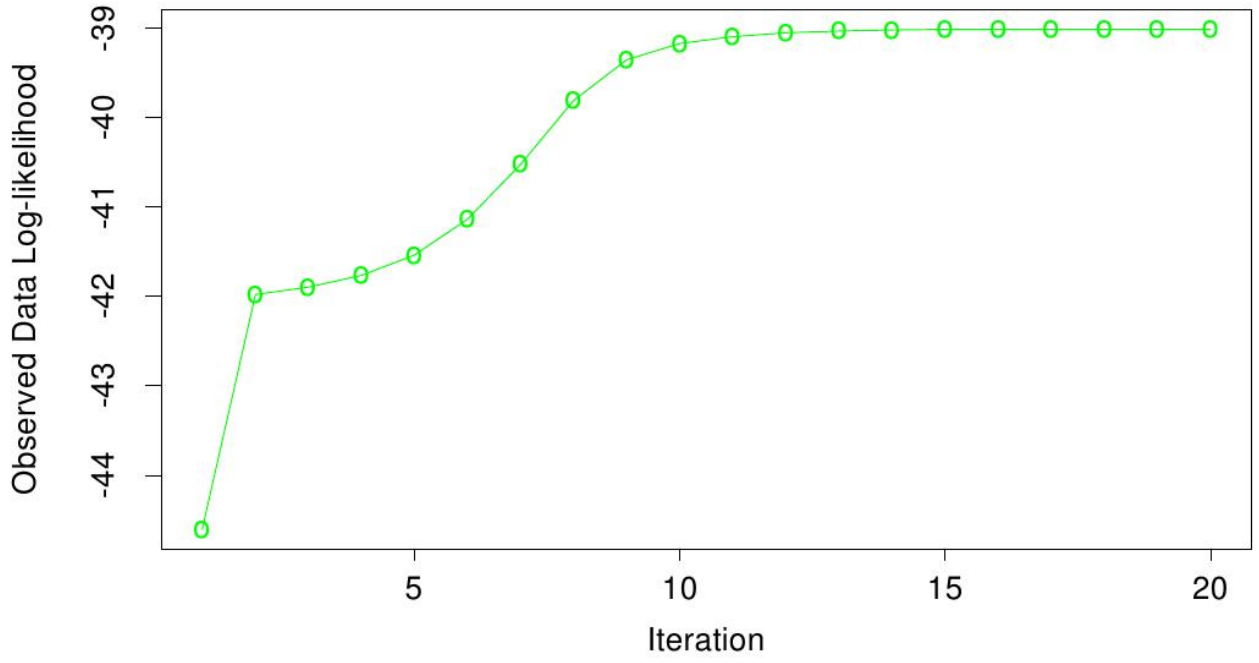
$$\mu_{\text{up}}, \Sigma_{\text{up}} = \arg\max_{\mu,\Sigma} \sum_{i=1}^{n} \mathbb{E}_q[\ln p(x_i^o, x_i^m|\mu, \Sigma)]$$

can be found. Recalling the $\widehat{\ }$ notation,

$$\begin{aligned} \mu_{\text{up}} &= \frac{1}{n}\sum_{i=1}^{n} \widehat{x}_i, \\ \Sigma_{\text{up}} &= \frac{1}{n}\sum_{i=1}^{n} \{(\widehat{x}_i - \mu_{\text{up}})(\widehat{x}_i - \mu_{\text{up}})^T + \widehat{V}_i\} \end{aligned}$$

Then return to the E-step to calculate the new $p(x_i^m|x_i^o, \mu_{\text{up}}, \Sigma_{\text{up}})$.

IMPLEMENTATION DETAILS



We need to initialize μ and Σ, for example, by setting missing values to zero and calculating μ_{ML} and Σ_{ML}. (We can also use random initialization.)

The EM objective function is then calculated after each update to μ and Σ and will look like the figure above. Stop when the change is "small."

The output is μ_{ML}, Σ_{ML} and $q(x_i^m)$ for all missing entries.

ColumbiaX: Machine Learning

Lecture 16

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

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

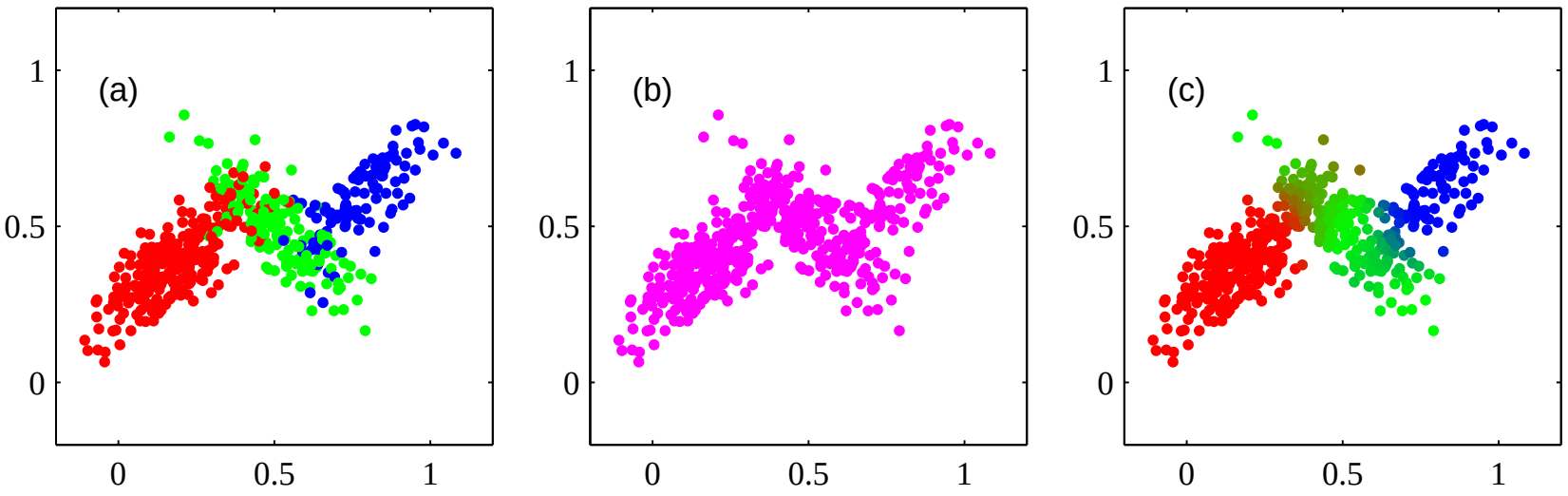
- Iterate until values no longer changing
  1. Update $\boldsymbol{c}$: For each i, set $c_i = \arg\min_k \|x_i - \mu_k\|^2$
  2. Update $\boldsymbol{\mu}$: For each k, set $\mu_k = \left(\sum_i x_i \mathbb{1}\{c_i = k\}\right) / \left(\sum_i \mathbb{1}\{c_i = k\}\right)$

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 $k \in \{1, \ldots, K\}$. 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 $\mathcal{L} = \sum_{i=1}^{n} \sum_{k=1}^{K} \phi_i(k) \frac{\|x_i - \mu_k\|^2}{\beta}$ over ϕ_i and μ_k

Conditions: $\phi_i(k) > 0$ and $\sum_{k=1}^{K} \phi_i(k) = 1$. Set parameter $\beta > 0$.

- Iterate the following

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

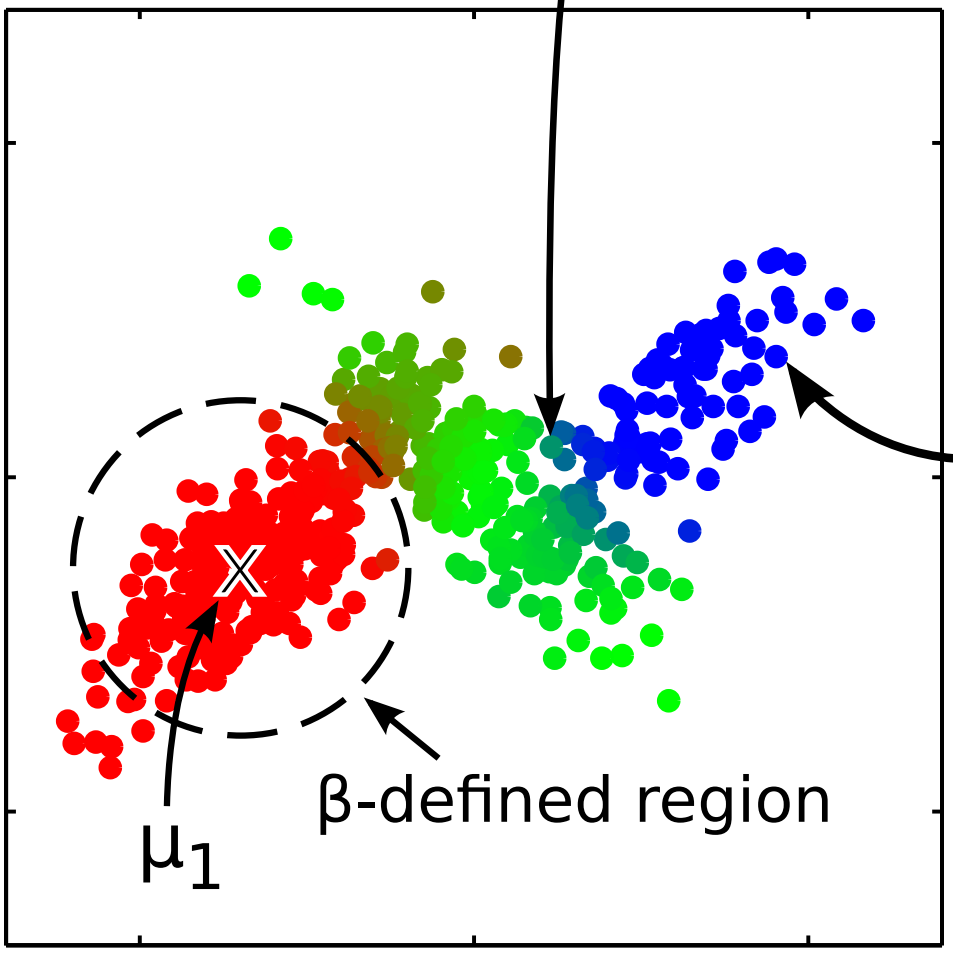
$$\phi_i(k) = \frac{\exp\{-\frac{1}{\beta}\|x_i - \mu_k\|^2\}}{\sum_j \exp\{-\frac{1}{\beta}\|x_i - \mu_j\|^2\}}, \quad \text{for } k = 1, \ldots, K$$

  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 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.

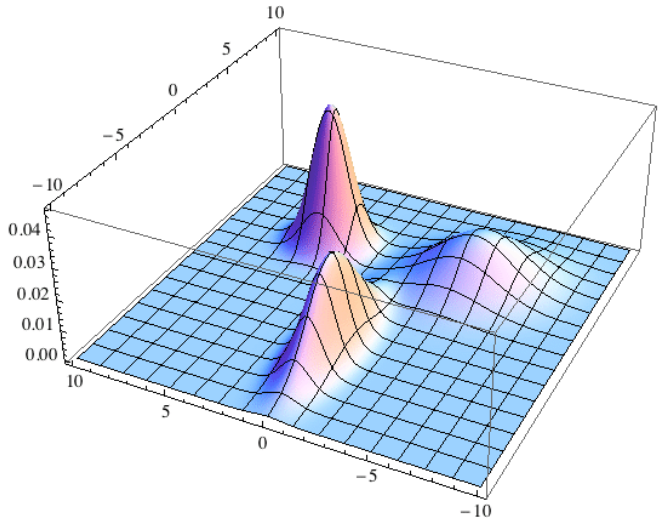
Mixture models work by defining:

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

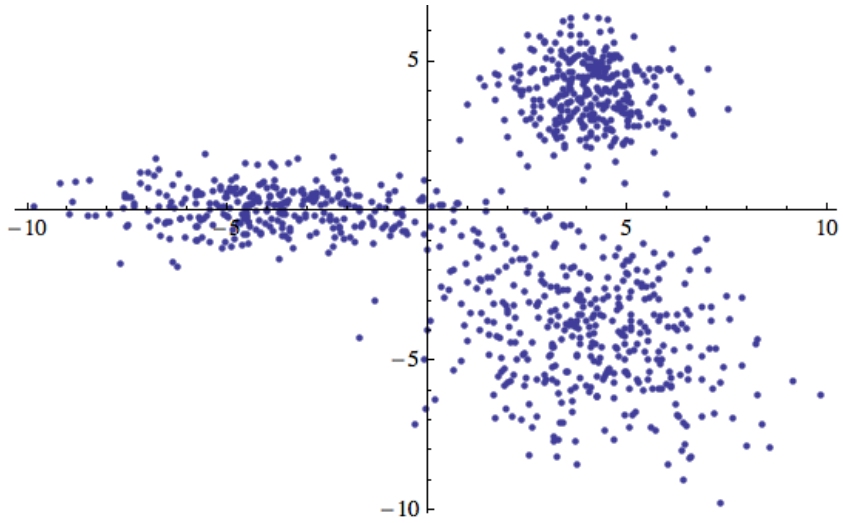
Intuitively we can connect a mixture model to the Bayes classifier:

- 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 $\theta_1, \ldots, \theta_K$.

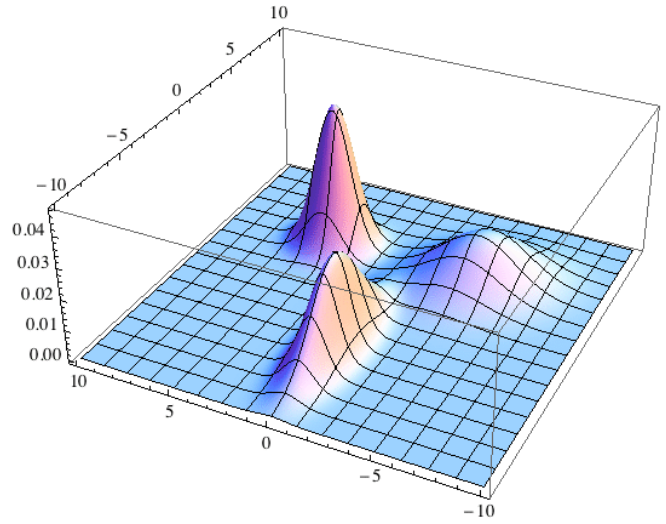
Generative process: For observation number $i = 1, \ldots, n$,

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

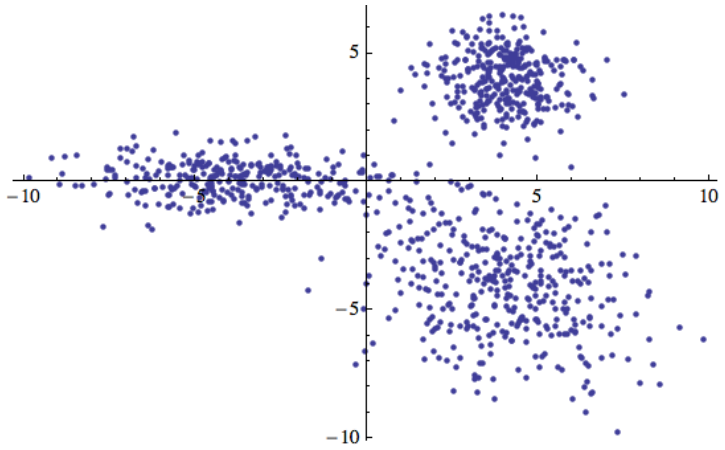
Some observations about this procedure:

- 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.

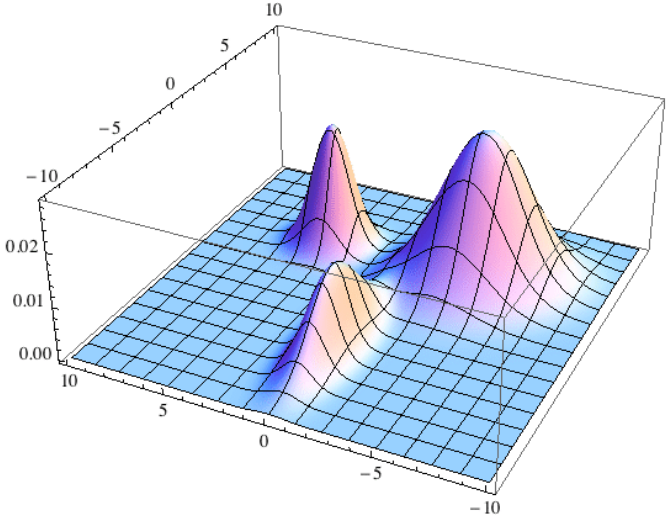
MIXTURE MODELS



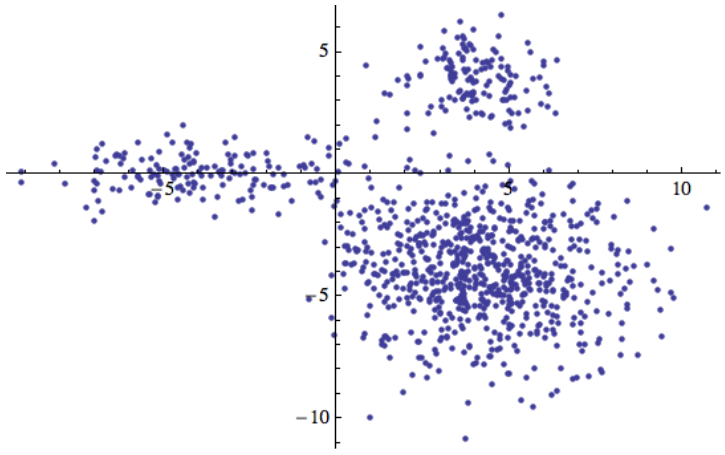
(a) Uniform mixing weights



(b) Data sampled from this distribution.



(c) Uneven mixing weights



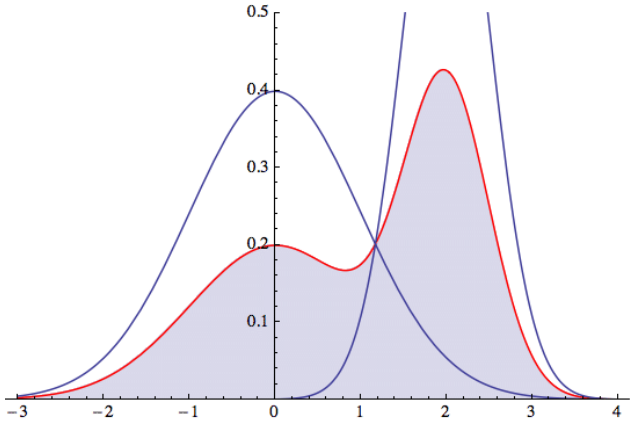
(d) Data sampled from this distribution.

GAUSSIAN MIXTURE MODELS

ILLUSTRATION

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

Mixture of two Gaussians

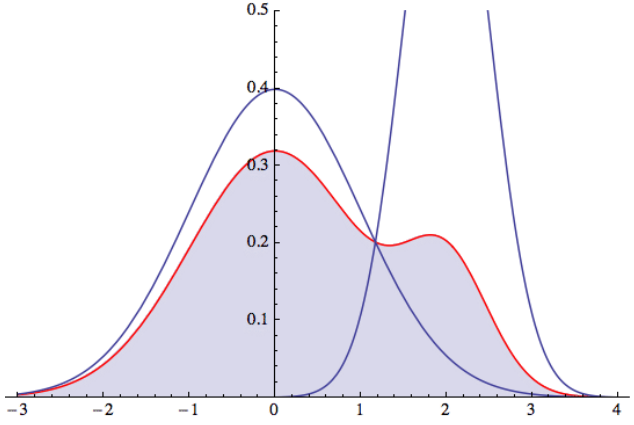


The red line is the density function.

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

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

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 the sum effectively 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

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

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 $\boldsymbol{\Sigma}$:

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

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

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

M-step: Maximize Q 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:

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

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

Algorithm: Maximum likelihood EM for the GMM

Given: $x_1, \ldots, x_n$ 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 $i = 1, \ldots, n$, set

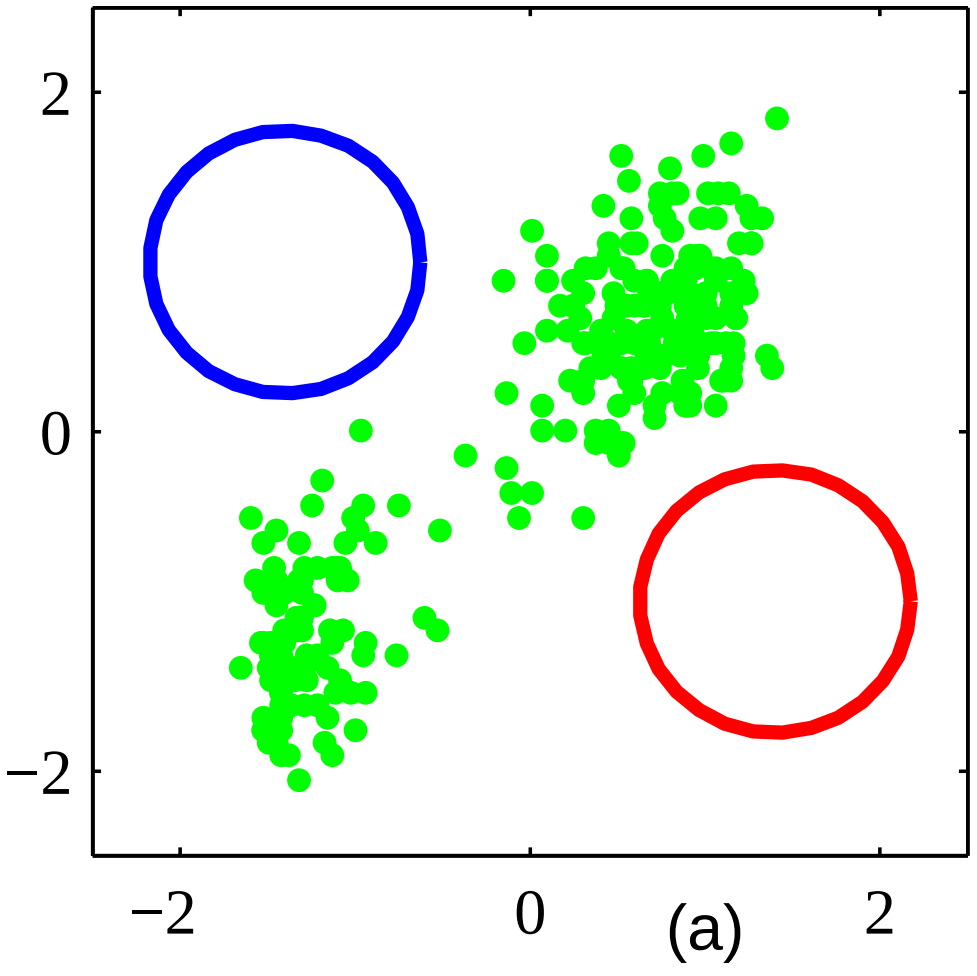
$$\phi_i(k) = \frac{\pi_k N(x_i|\mu_k, \Sigma_k)}{\sum_j \pi_j N(x_i|\mu_j, \Sigma_j)}, \quad \text{for } k = 1, \ldots, K$$

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

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

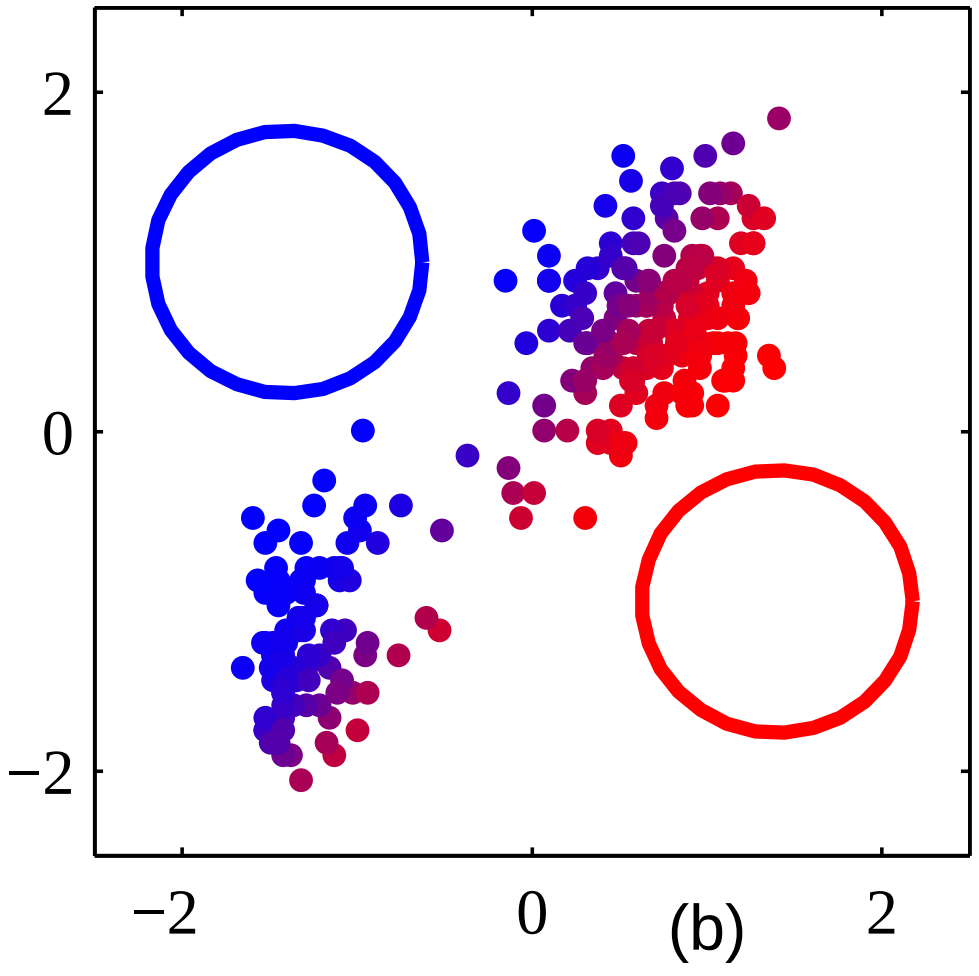
Comment: The updated value for μ_k is used when updating Σ_k.

GAUSSIAN MIXTURE MODEL: EXAMPLE RUN



A random initialization

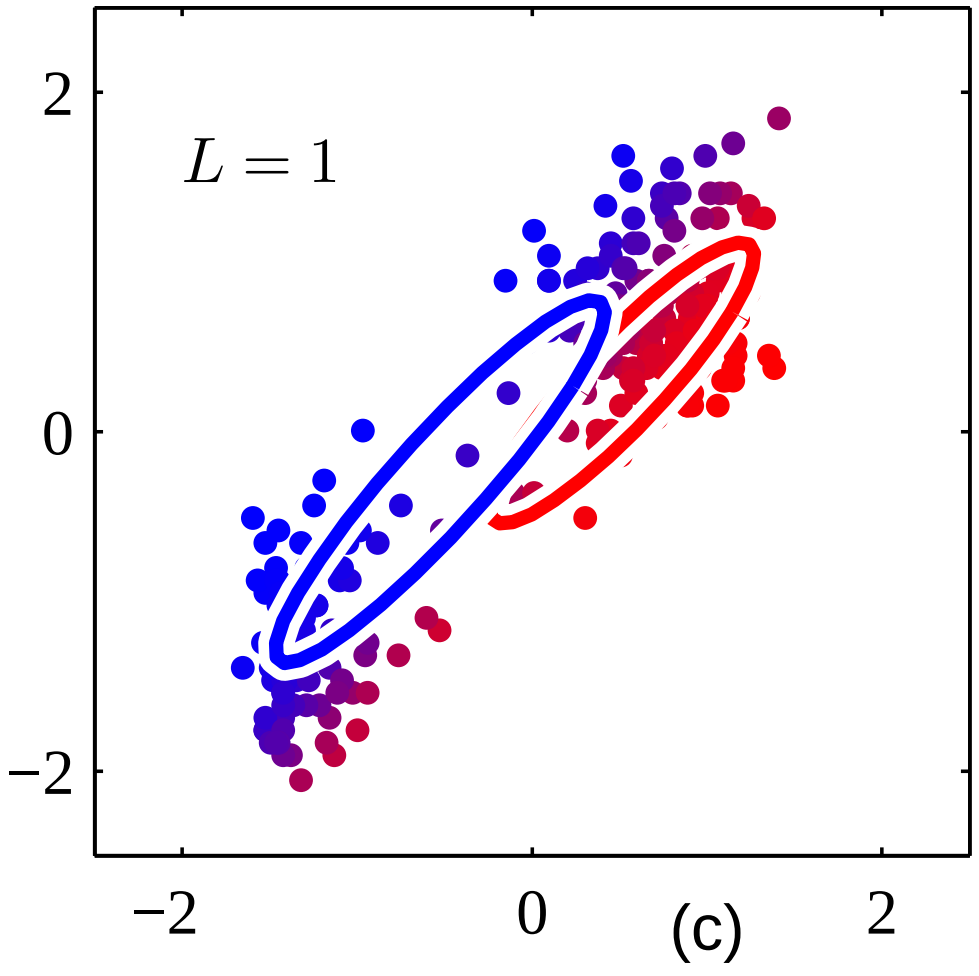
GAUSSIAN MIXTURE MODEL: EXAMPLE RUN



Iteration 1 (E-step)

Assign data to clusters

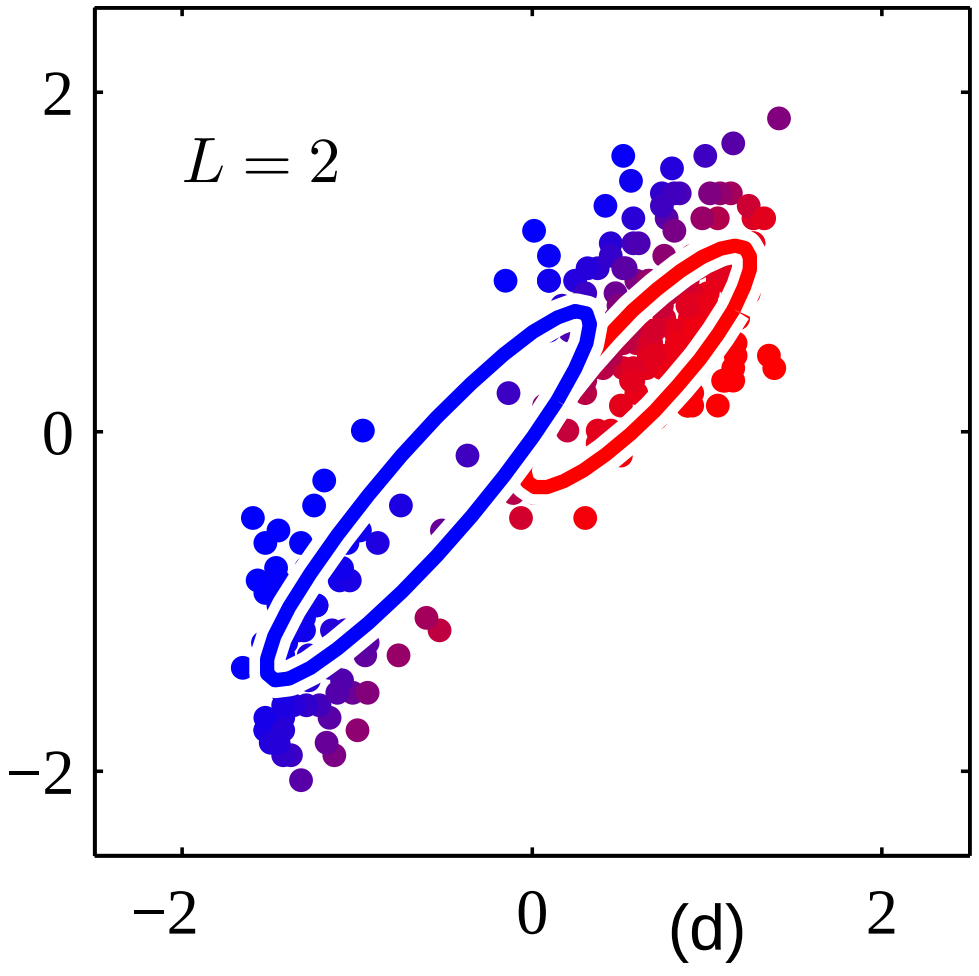
GAUSSIAN MIXTURE MODEL: EXAMPLE RUN



Iteration 1 (M-step)

Update the Gaussians

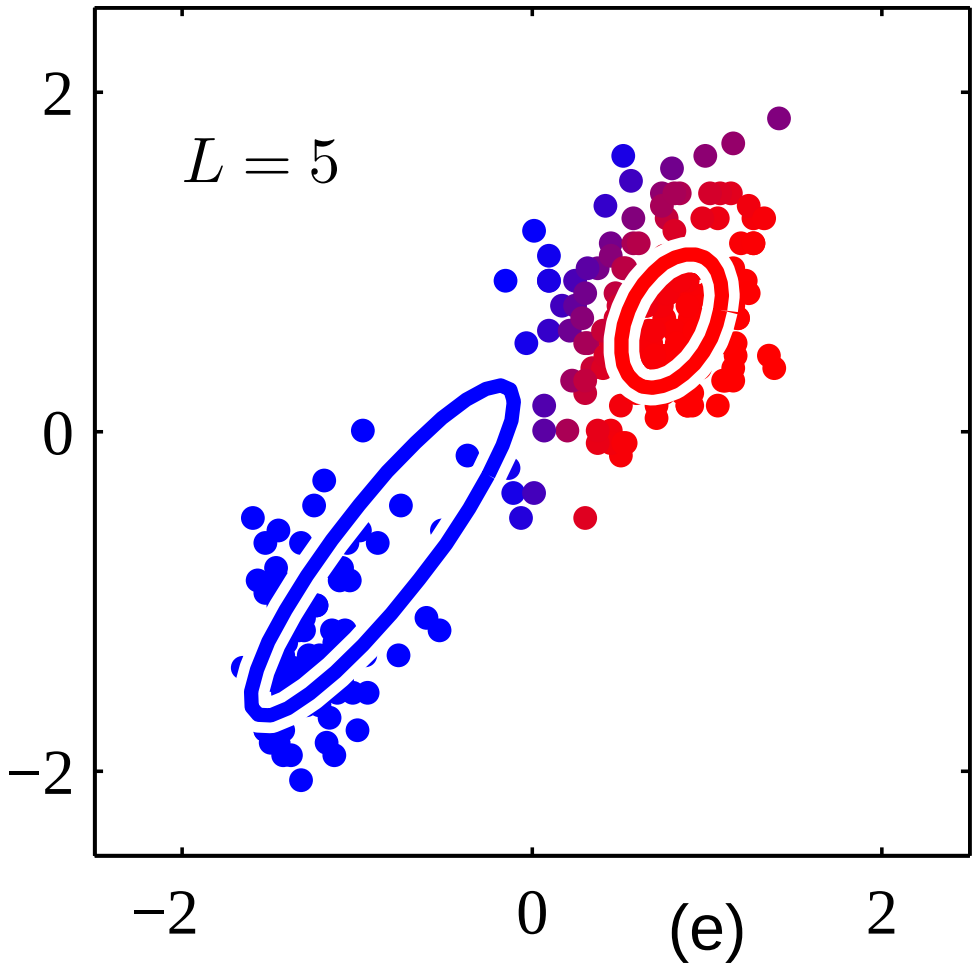
GAUSSIAN MIXTURE MODEL: EXAMPLE RUN



Iteration 2

Assign data to clusters
and update the Gaussians

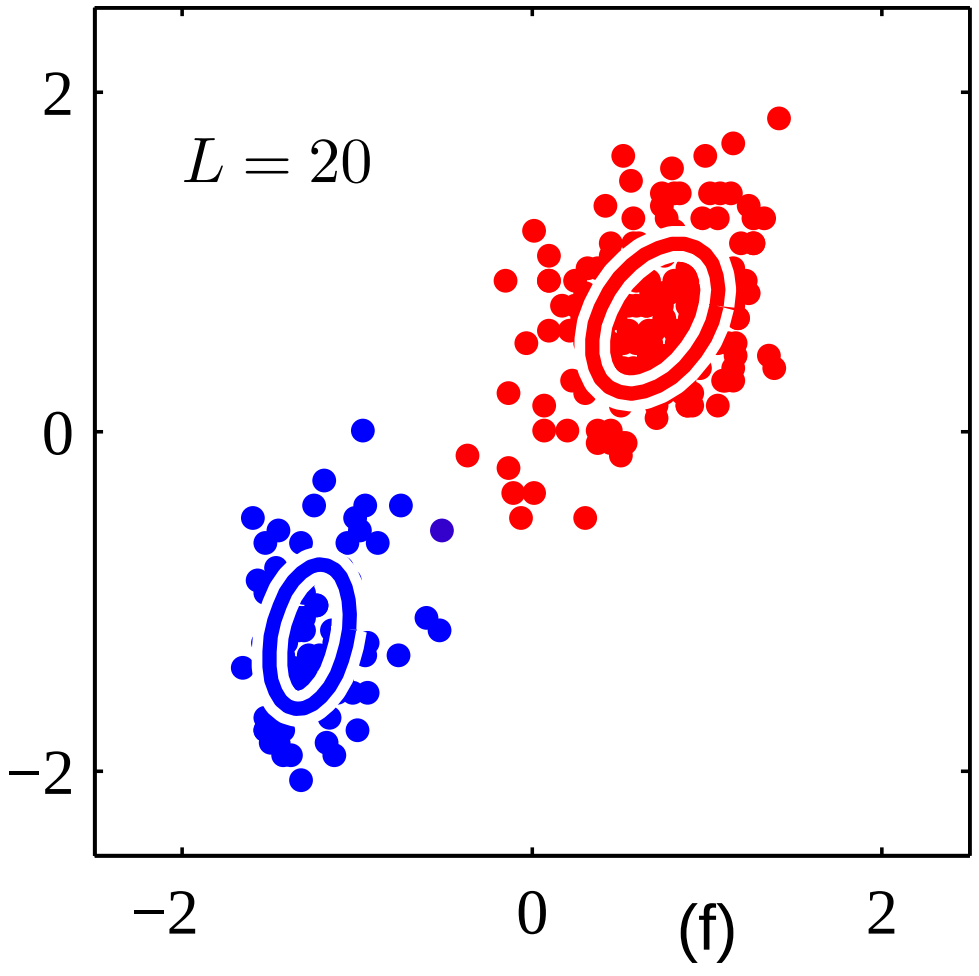
GAUSSIAN MIXTURE MODEL: EXAMPLE RUN



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

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

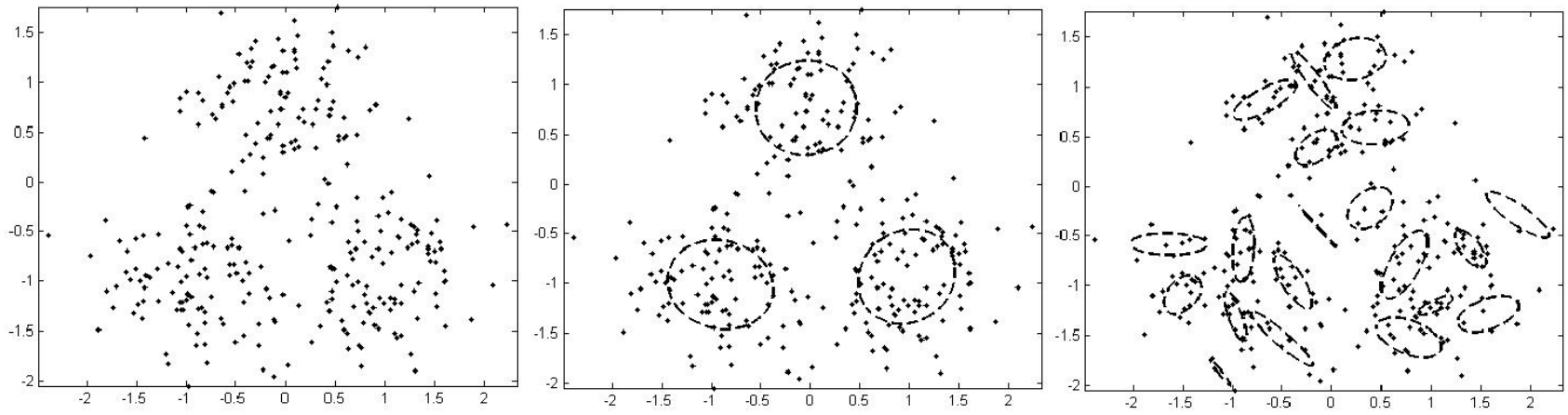
- π_k = class prior, and $N(\mu_k, \Sigma_k)$ = class-conditional density function.
- We learned π, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ using maximum likelihood here too.

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).

Algorithm: Maximum likelihood EM for mixture models

Given: Data $x_1, \ldots, x_n$ 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 $i = 1, \ldots, n$, set

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

2. **M-step**: For $k = 1, \ldots, K$, 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 $p(x|\theta_k)$.