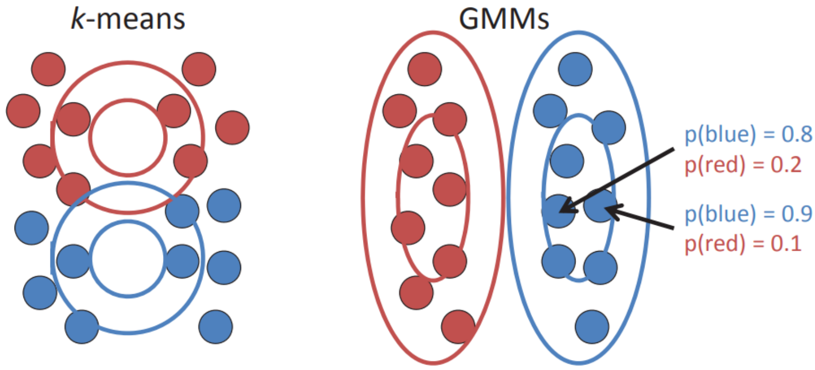
Gaussian Mixture Models, Expectation Maximization

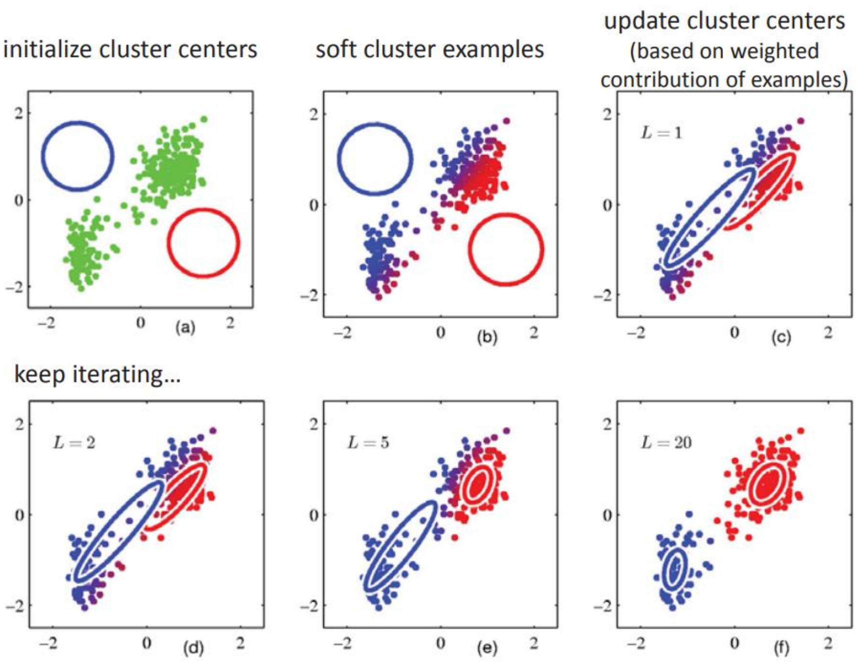
Gaussian Mixture Models

- Assume data came from **mixture of Gaussians** (**elliptical data**)
- Assign data to cluster with certain **probability** (**soft clustering**)



- Very similar at high-level to *k*-means: iterate between assigning examples and updating cluster centers

GMM Example

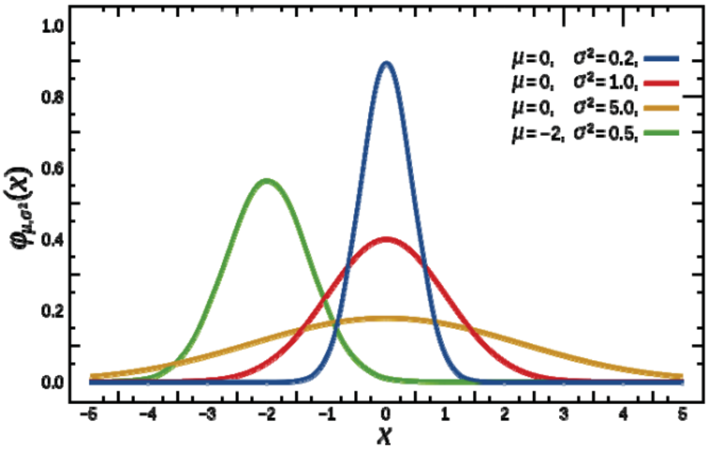


Univariate Gaussian Distribution

(scalar) random variable X

parameters: (scalar) mean μ, (scalar) variance σ^2

$$X \sim N(\mu, \sigma^2) \qquad p(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$



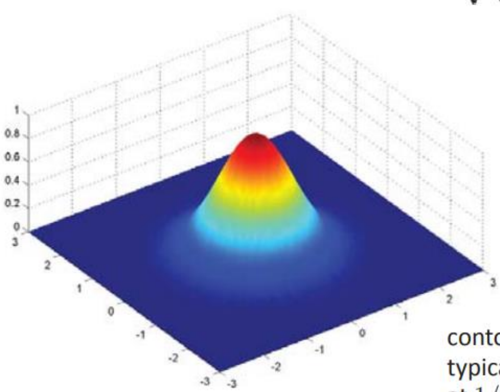
Multivariate Gaussian Distribution

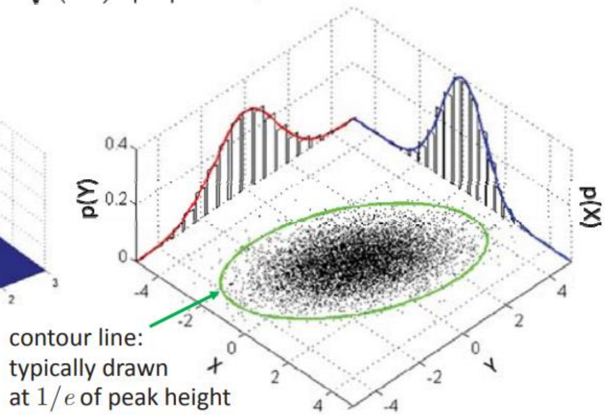
random variable vector $\boldsymbol{X} = [X_1, \ldots, X_n]^T$

parameters: mean vector $\boldsymbol{\mu} \in \mathbb{R}^n$

covariance matrix $\boldsymbol{\Sigma}$ (symmetric, positive definite)

$$\boldsymbol{X} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \quad p(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}-\boldsymbol{\mu})\right)$$





Covariance Matrix

Recall for pair of r.v.'s X and Y, covariance is defined as

$$\mathrm{cov}(X, Y) = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])]$$

For $\boldsymbol{X} = [X_1, \ldots, X_n]^T$, **covariance matrix** summarizes covariances across all pairs of variables:

$$\boldsymbol{\Sigma} = \mathbb{E}[(\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}])(\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}])^T]$$

$$\boldsymbol{\Sigma} \text{ is } n \times n \text{ matrix s.t. } \Sigma_{ij} = \mathrm{cov}(X_i, X_j)$$

___ parameters

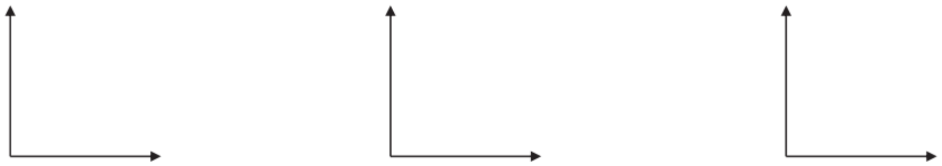
$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1n} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \cdots & \sigma_n^2 \end{bmatrix}$$

___ params

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & & 0 \\ & \sigma_2^2 & \\ 0 & & \ddots \\ & & & \sigma_n^2 \end{bmatrix}$$

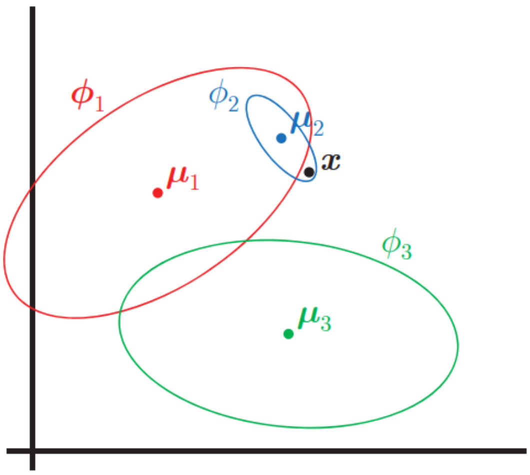
___ params

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma^2 & & 0 \\ & \sigma^2 & \\ 0 & & \ddots \\ & & & \sigma^2 \end{bmatrix}$$



GMMs as Generative Model

- There are k components
- Component j
  - has associated mean vector $\boldsymbol{\mu}_j$ and covariance matrix $\boldsymbol{\Sigma}_j$
  - generates data from $N(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$
- Each example $\boldsymbol{x}^{(i)}$ is generated according to following recipe:
  - pick component j at random with probability ϕ_j
  - sample $\boldsymbol{x}^{(i)} \sim N(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$



GMM Optimization

Assume supervised setting (known cluster assignments)

MLE for univariate Gaussian

$$\hat{\mu} = \frac{1}{n}\underbrace{\sum_{i=1}^{n}}_{\text{sum over points generated from this Gaussian}} x^{(i)} \qquad \hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}\left(x^{(i)} - \hat{\mu}\right)^2$$

MLE for multivariate Gaussian

$$\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}^{(i)} \qquad \hat{\boldsymbol{\Sigma}} = \frac{1}{n}\sum_{i=1}^{n}\left(\boldsymbol{x}^{(i)} - \hat{\boldsymbol{\mu}}\right)\left(\boldsymbol{x}^{(i)} - \hat{\boldsymbol{\mu}}\right)^T$$

Expectation Maximization

- Clever method for maximizing marginal likelihoods

$$\arg\max_{\theta} \prod_{i=1}^{n} P\left(\boldsymbol{x}^{(i)}\right) = \arg\max_{\theta} \prod_{i=1}^{n}\sum_{j=1}^{k} P\left(\boldsymbol{x}^{(i)}, z^{(i)} = j\right)$$

- Excellent approach for unsupervised learning
- Can do “trivial” things (upcoming example)
- One of most general unsupervised approaches with many other uses (e.g. HMM inference)

Overview

- Begin with guess for model parameters
- Repeat until convergence
  - Update latent variables based on our expectations [E-step]
  - Update model parameters to maximize log likelihood [M-step]

Silly Example

Let events be "grades in a class"

component 1 = gets an A P(A) = ½

component 2 = gets a B P(B) = p

component 3 = gets a C P(C) = $2p$

component 4 = gets a D P(D) = ½ $- 3p$ (note $0 \leq p \leq 1/6$)

Assume we want to estimate p from data. In a given class, there were

a A's, b B's, c C's, d D's.

What is the MLE of *p* given a, b, c, d?

so if class got

a	*b*	*c*	*d*
14	6	9	10

Same Problem with Hidden Information

Someone tells us that

# of high grades (A's + B's) = h

# of C's = c

# of D's = d

What is the MLE of p now?

Remember

P(A) = ½

P(B) = p

P(C) = $2p$

P(D) = ½ $- 3p$

We can answer this question circularly:

EXPECTATION If we know value of p,
we could compute **expected** values of a and b.

MAXIMIZATION If we know expected values of a and b,
we could compute **maximum** likelihood value of p.

EM for Our Silly Example

- Begin with initial guess for p
- Iterate between Expectation and Maximization to improve our estimates of p and a & b

- Define $p^{(t)}$ = estimate of p on t^{th} iteration
  $b^{(t)}$ = estimate of b on t^{th} iteration
- Repeat until convergence

E-step $$b^{(t)} = \frac{p^{(t)}}{\frac{1}{2} + p^{(t)}} h \qquad = \mathbb{E}[b|p^{(t)}]$$

M-step $$p^{(t+1)} = \frac{b^{(t)} + c}{6\left(b^{(t)} + c + d\right)} \qquad = \text{MLE of } p \text{ given } b^{(t)}$$

EM Convergence

- Good news: converging to local optima is guaranteed
- Bad news: local optima

Aside (idea behind convergence proof)

- likelihood must increase or remain same between each iteration [not obvious]
- likelihood can never exceed 1 [obvious]
- so likelihood must converge [obvious]

In our example, suppose we had

$$h = 20,\ c = 10,\ d = 10$$
$$p^{(0)} = 0$$

Error generally decreases by constant factor each time step (e.g. convergence is linear)

t	$p^{(t)}$	$b^{(t)}$
0	0	0
1	0.0833	2.857
2	0.0937	3.158
3	0.0947	3.185
4	0.0948	3.187
5	0.0948	3.187
6	0.0948	3.187

Final Comments

EM is not magic

- Still optimizing non-convex function with lots of local optima
- Computations are just easier (often, significantly so!)

Problems

- EM susceptible to local optima

$\Rightarrow$ reinitialize at several different initial parameters

Extensions

- EM looks at maximum log likelihood of data

$\Rightarrow$also possible to look at maximum *a posteriori*

Clustering methods: Comparison

	Hierarchical	**K-means**	**GMM**
Running time	naively, O(N^3)	fastest (each iteration is linear)	fast (each iteration is linear)
Assumptions	requires a similarity / distance measure	strong assumptions	strongest assumptions
Input parameters	none	K (number of clusters)	K (number of clusters)
Clusters	subjective (only a tree is returned)	exactly K clusters	exactly K clusters