Mixtures of Gaussians continued

Machine Learning – CSE446

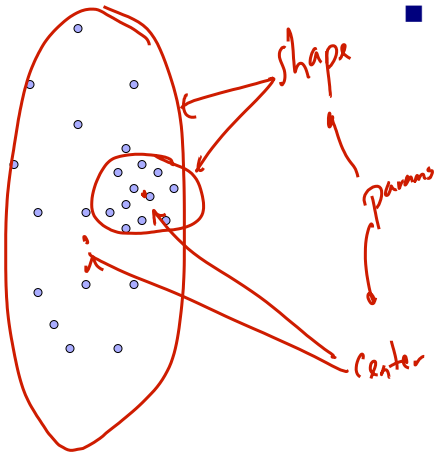
Carlos Guestrin

University of Washington

May 17, 2013

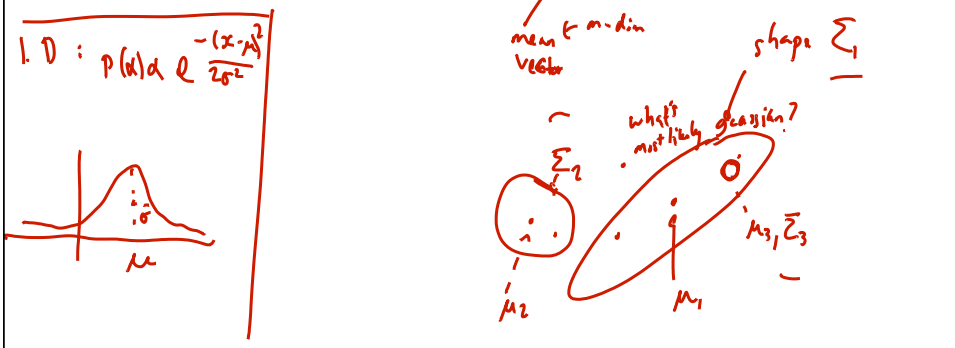
(One) bad case for k-means

- Clusters may overlap
- Some clusters may be "wider" than others



Gaussians in *m* Dimensions

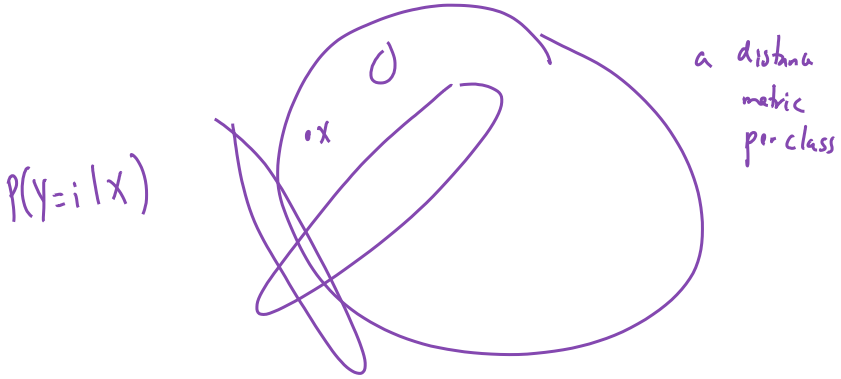
$$P(\mathbf{x}) = \frac{1}{(2\pi)^{m/2} \|\Sigma\|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1} (\mathbf{x}-\mu)\right]$$



Suppose You Have a Gaussian For Each Class

$$\|a-b\|_\Sigma = (a-b)\Sigma(a-b)$$

$$P(\mathbf{x} \mid y=i) \propto \frac{1}{(2\pi)^{m/2} \|\Sigma_i\|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mu_i)^T \Sigma_i^{-1} (\mathbf{x}-\mu_i)\right]$$



Gaussian Bayes Classifier

- You have a Gaussian over **x** for each class y=i:

$$P(\mathbf{x} \mid y=i) = \frac{1}{(2\pi)^{m/2} \|\Sigma_i\|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mu_i)^T \Sigma_i^{-1} (\mathbf{x}-\mu_i)\right]$$

- But you need probability of class y=i given **x**:

$P(y=i \mid x)$

- Thank you Bayes Rule!!

(cluster data likelihoods)

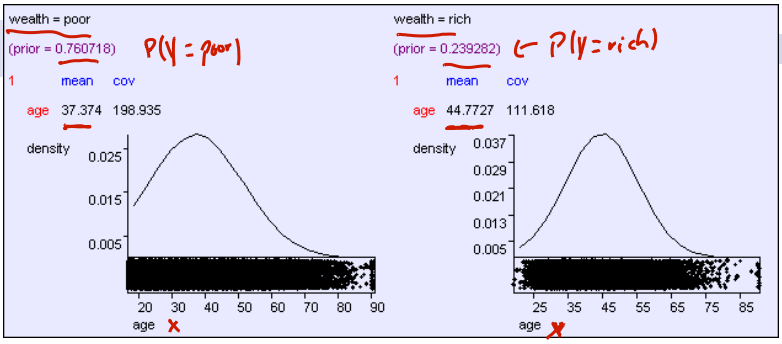
prior: before we see x, what fraction of points we expect to fall in cluster i

$$P(y=i \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid y=i) P(y=i)}{p(\mathbf{x})}$$

normalizer

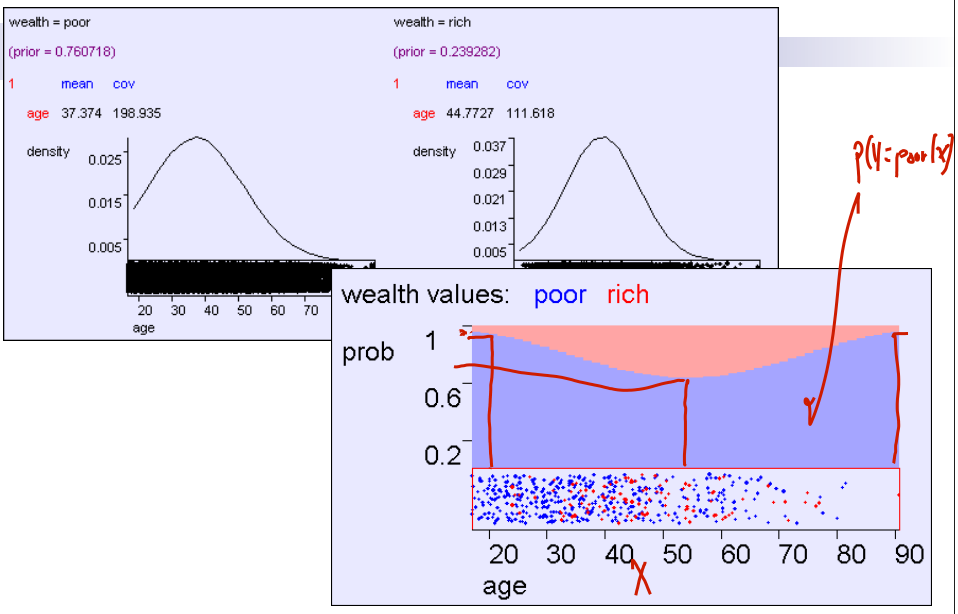
$\propto \frac{1}{(2\pi)^{m/2} |\Sigma_i|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1} (x-\mu_i)\right) \cdot P(y=i)$

Predicting wealth from age



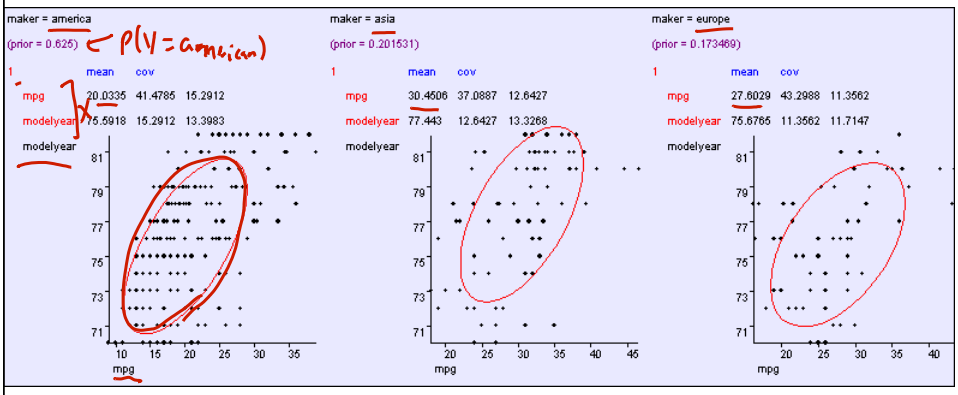
$P(y=\text{poor} \mid x) = P(y=\text{poor} \mid \text{age})$

Predicting wealth from age



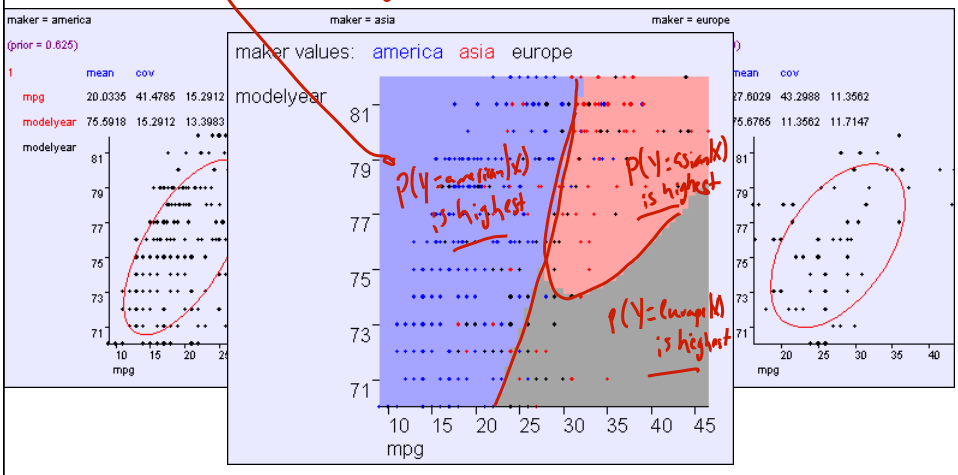
Learning modelyear , mpg ---> maker

$$\Sigma = \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1m} \\ \sigma_{12} & \sigma_2^2 & \cdots & \sigma_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{1m} & \sigma_{2m} & \cdots & \sigma_m^2 \end{pmatrix}$$



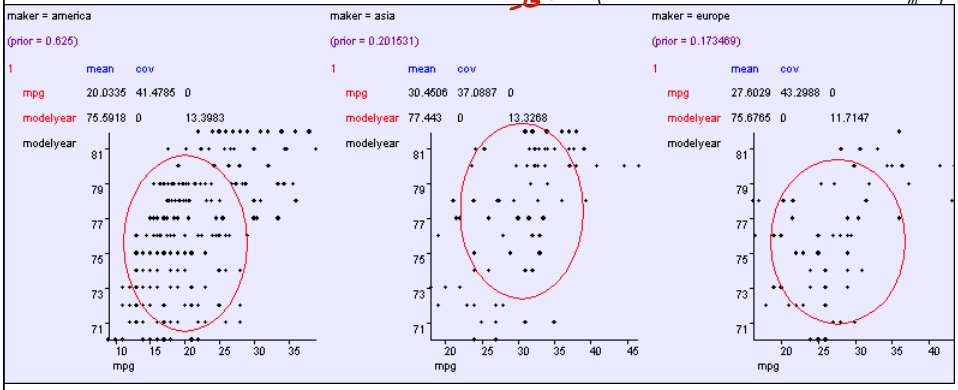
General: $O(m^2)$ parameters

$$\Sigma = \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1m} \\ \sigma_{12} & \sigma_2^2 & \cdots & \sigma_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{1m} & \sigma_{2m} & \cdots & \sigma_m^2 \end{pmatrix}$$



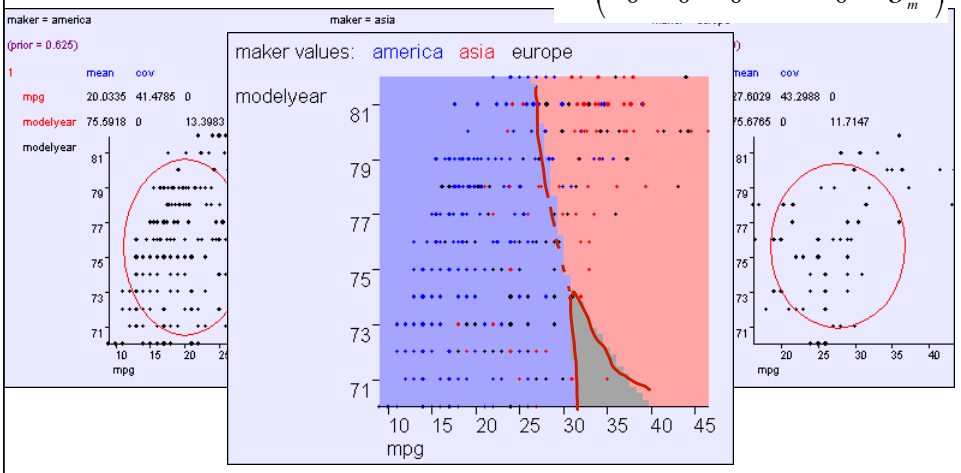
Aligned: $O(m)$ parameters

$$\Sigma = \begin{pmatrix} \sigma_1^2 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \sigma_2^2 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \sigma_3^2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \sigma_{m-1}^2 & 0 \\ 0 & 0 & 0 & \cdots & 0 & \sigma_m^2 \end{pmatrix}$$



Aligned: *O(m)* parameters

$$\Sigma = \begin{pmatrix} \sigma_1^2 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \sigma_2^2 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \sigma_3^2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \sigma_{m-1}^2 & 0 \\ 0 & 0 & 0 & \cdots & 0 & \sigma_m^2 \end{pmatrix}$$

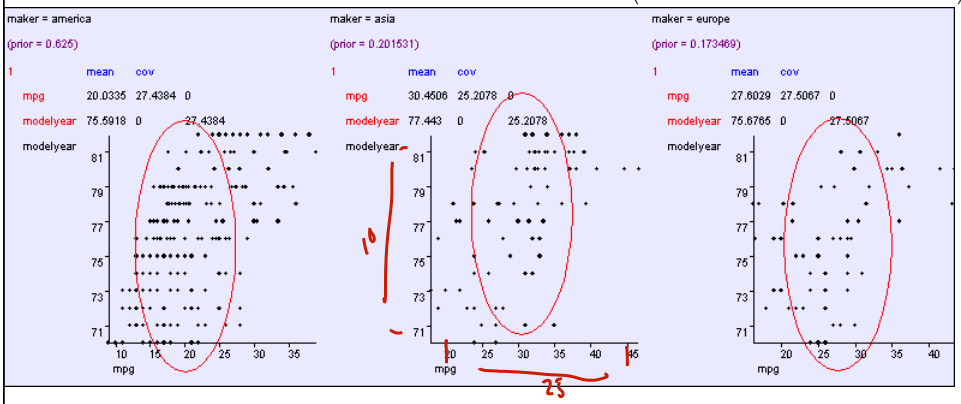


Spherical: *O(1)* cov parameters

spherical

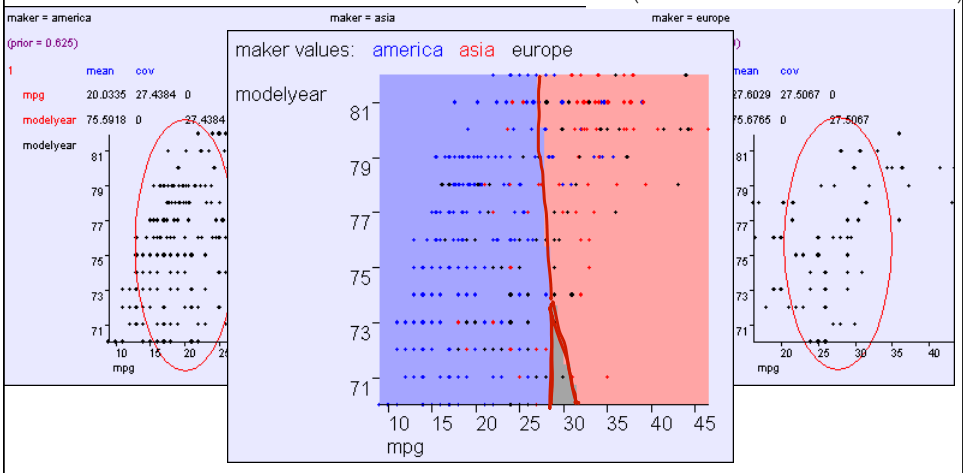
$$\Sigma = \begin{pmatrix} \sigma^2 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \sigma^2 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \sigma^2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \sigma^2 & 0 \\ 0 & 0 & 0 & \cdots & 0 & \sigma^2 \end{pmatrix}$$

all diag entries $= \sigma^2$



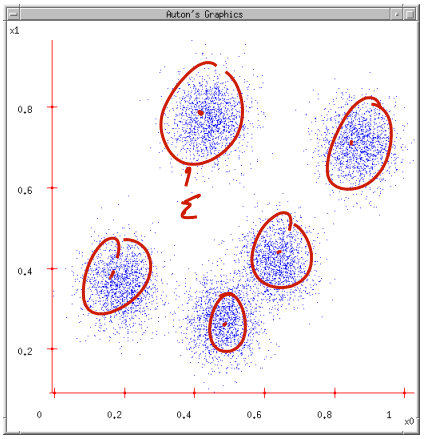
Spherical: *O(1)* cov parameters

$$\Sigma = \begin{pmatrix} \sigma^2 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \sigma^2 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \sigma^2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \sigma^2 & 0 \\ 0 & 0 & 0 & \cdots & 0 & \sigma^2 \end{pmatrix}$$

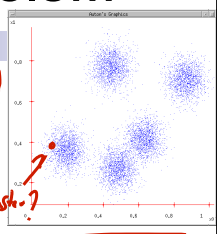


Next… back to Density Estimation

What if we want to do density estimation with multimodal or clumpy data?



But we don't see class labels!!!



- MLE:
  - argmax $\prod_j P(y^j,x^j)$
- But we don't know y^j!!!
- Maximize marginal likelihood:
  - argmax $\prod_j P(x^j)$ = argmax $\prod_j \sum_{i=1}^{k} P(y^j=i,x^j)$

Special case: spherical Gaussians and hard assignments

$$P(y=i \mid \mathbf{x}^j) \propto \frac{1}{(2\pi)^{m/2} \|\Sigma_i\|^{1/2}} \exp\left[-\frac{1}{2}\left(\mathbf{x}^j-\mu_i\right)^T \Sigma_i^{-1}\left(\mathbf{x}^j-\mu_i\right)\right] P(y=i)$$

- If P(X|Y=i) is spherical, with same σ for all classes:

$$P(\mathbf{x}^j \mid y=i) \propto \exp\left[-\frac{1}{2\sigma^2}\left\|\mathbf{x}^j-\mu_i\right\|^2\right]$$

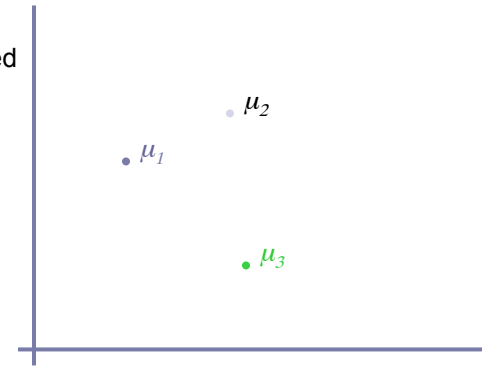
- If each x_j belongs to one class C(j) (hard assignment), marginal likelihood:

$$\prod_{j=1}^{N}\sum_{i=1}^{k} P(\mathbf{x}^j, y=i) \propto \prod_{j=1}^{N} \exp\left[-\frac{1}{2\sigma^2}\left\|\mathbf{x}^j-\mu_{C(j)}\right\|^2\right]$$

- Same as K-means!!!

The GMM assumption

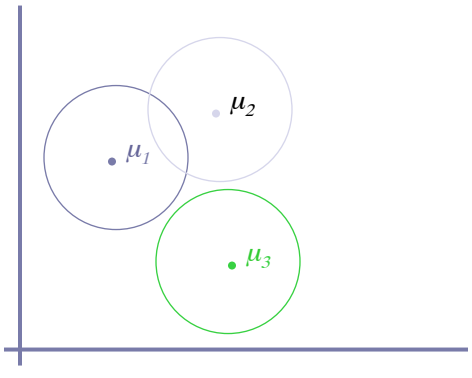
- There are k components
- Component *i* has an associated mean vector μ_i



The GMM assumption

- There are k components
- Component *i* has an associated mean vector μ_i
- Each component generates data from a Gaussian with mean m_i and covariance matrix $\sigma^2 I$

Each data point is generated according to the following recipe:

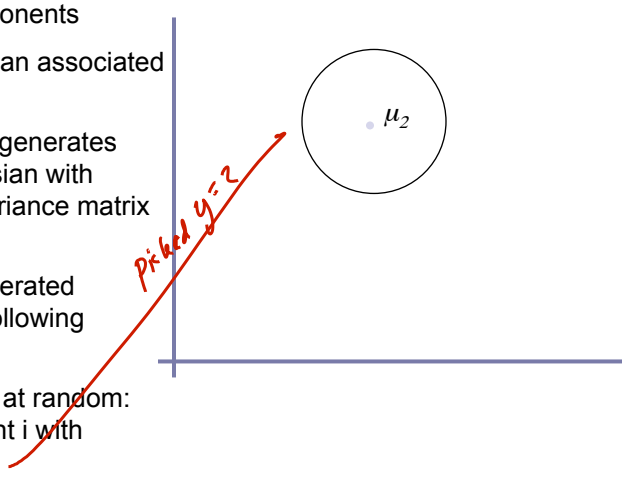


The GMM assumption

- There are k components
- Component *i* has an associated mean vector μ_i
- Each component generates data from a Gaussian with mean m_i and covariance matrix $\sigma^2 \boldsymbol{I}$

Each data point is generated according to the following recipe:

1. Pick a component at random: Choose component i with probability *P(y=i)*

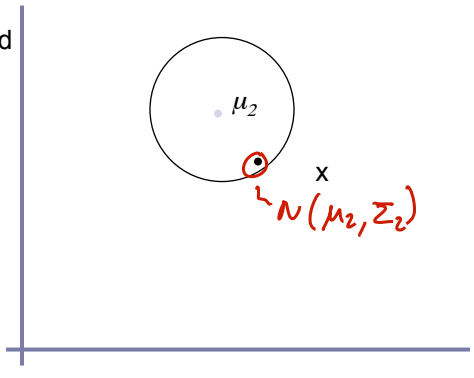


The GMM assumption

- There are k components
- Component *i* has an associated mean vector μ_i
- Each component generates data from a Gaussian with mean m_i and covariance matrix $\sigma^2 \boldsymbol{I}$

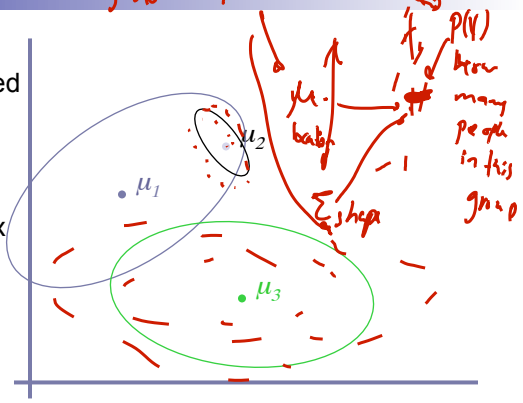
Each data point is generated according to the following recipe:

1. Pick a component at random: Choose component i with probability *P(y=i)*
2. Datapoint ~ $\mathrm{N}(\mu_i, \sigma^2 \boldsymbol{I})$



The General GMM assumption

- There are k components
- Component *i* has an associated mean vector μ_i
- Each component generates data from a Gaussian with mean μ_i and covariance matrix Σ_i



Each data point is generated according to the following recipe:

1. Pick a component at random: Choose component i with probability $P(y=i)$
2. Datapoint ~ $N(\mu_i, \Sigma_i)$

Unsupervised Learning with Mixtures of Gaussians
EM Algorithm

Machine Learning – CSE446
Carlos Guestrin
University of Washington

May 17, 2013

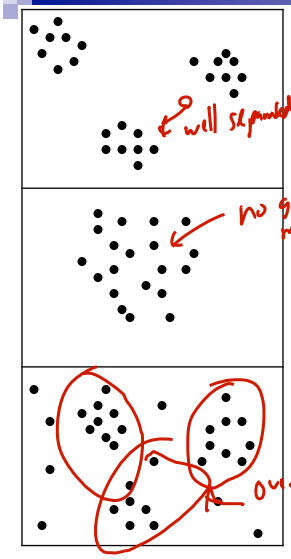
Supervised Learning of Mixtures of Gaussians

- Mixtures of Gaussians:
  - Prior class probabilities: P(y)
  - Likelihood function per class: P(**x**|y=i)
- Suppose, for each data point, we know location **x** and class y
  - Learning is easy… ☺
  - For prior P(y) ⟷ $P(y=i) = \frac{\text{(count (y=i) in data}}{N}$
  - For likelihood function:

μ_i is average of x^j for points in class i

$\Sigma_i \leftarrow \sigma_{uv} = \frac{\sum_{j \text{ in cluster } i} (x_u^j - \mu_{iu})(x_v^j - \mu_{iv})}{\text{num points in Cluster i}}$

Unsupervised Learning: not as hard as it looks



Sometimes easy

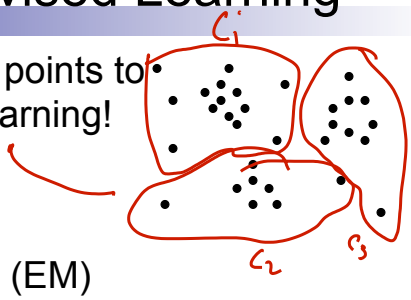
Sometimes impossible

and sometimes in between

*IN CASE YOU'RE WONDERING WHAT THESE DIAGRAMS ARE, THEY SHOW 2-d UNLABELED DATA (**X** VECTORS) DISTRIBUTED IN 2-d SPACE. THE TOP ONE HAS THREE VERY CLEAR GAUSSIAN CENTERS*

EM: "Reducing" Unsupervised Learning to Supervised Learning

- If we knew assignment of points to classes ➔ Supervised Learning!



- Expectation-Maximization (EM)
  - Guess assignment of points to classes or clusters
  - Recompute model parameters
  - Iterate

The E.M. Algorithm

DETOUR

- We'll get back to unsupervised learning soon
- But now we'll look at an even simpler case with hidden information
- The EM algorithm
  - Can do trivial things, such as the contents of the next few slides
  - An excellent way of doing our unsupervised learning problem, as we'll see
  - Many, many other uses…

Silly Example

Let events be "grades in a class"

w_1 = Gets an A	P(A) = ½
w_2 = Gets a B	P(B) = μ
w_3 = Gets a C	P(C) = 2μ
w_4 = Gets a D	P(D) = ½-3μ

(Note $0 \le \mu \le 1/6$)

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

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

What's the maximum likelihood estimate of μ given a,b,c,d ?

Trivial Statistics

Normalization Constant

P(A) = ½ P(B) = μ P(C) = 2μ P(D) = ½-3μ

$P(a,b,c,d \mid \mu) = K(\tfrac{1}{2})^a(\mu)^b(2\mu)^c(\tfrac{1}{2}-3\mu)^d$

$\log P(a,b,c,d \mid \mu) = \log K + a\log \tfrac{1}{2} + b\log \mu + c\log 2\mu + d\log(\tfrac{1}{2}-3\mu)$

FOR MAX LIKE μ, SET $\frac{\partial \text{LogP}}{\partial \mu} = 0$

$$\frac{\partial \text{LogP}}{\partial \mu} = \frac{b}{\mu} + \frac{2c}{2\mu} - \frac{3d}{1/2 - 3\mu} = 0$$

Gives max like $\mu = \frac{b+c}{6(b+c+d)}$

So if class got

A	B	C	D
14	6	9	10

Max like $\mu = \frac{1}{10}$

Boring, but true!

Same Problem with Hidden Information

Someone tells us that

Number of High grades (A's + B's) = *h*

Number of C's = *c*

Number of D's = *d*

What is the max. like estimate of μ now?

We can answer this question circularly:

REMEMBER
P(A) = ½
P(B) = μ
P(C) = 2μ
P(D) = ½-3μ

EXPECTATION

If we know the value of μ we could compute the expected value of *a* and *b*

Since the ratio a:b should be the same as the ratio ½ : μ

$$a = \frac{1/2}{1/2 + \mu} h \qquad b = \frac{\mu}{1/2 + \mu} h$$

MAXIMIZATION

If we know the expected values of *a* and *b* we could compute the maximum likelihood value of μ

$$\mu = \frac{b + c}{6(b + c + d)}$$

E.M. for our Trivial Problem

REMEMBER
P(A) = ½
P(B) = μ
P(C) = 2μ
P(D) = ½-3μ

We begin with a guess for μ

We iterate between EXPECTATION and MAXIMALIZATION to improve our estimates of μ and *a* and *b*.

Define $\mu^{(t)}$ the estimate of μ on the t'th iteration

$b^{(t)}$ the estimate of *b* on t'th iteration

$$\mu^{(0)} = \text{initial guess}$$

$$b^{(t)} = \frac{\mu^{(t)} h}{1/2 + \mu^{(t)}} = \mathrm{E}\left[b \mid \mu^{(t)}\right]$$

E-step

$$\mu^{(t+1)} = \frac{b^{(t)} + c}{6(b^{(t)} + c + d)}$$

$$= \text{max like est. of } \mu \text{ given } b^{(t)}$$

M-step

Continue iterating until converged.
Good news: Converging to local optimum is assured.
Bad news: I said "local" optimum.

just like k-means

E.M. Convergence

- Convergence proof based on fact that Prob(data | μ) must increase or remain same between each iteration [NOT OBVIOUS]
- But it can never exceed 1 [OBVIOUS]

So it must therefore converge [OBVIOUS]

would we get the same answer with a diff. starting point? Maybe... May be not... who knows

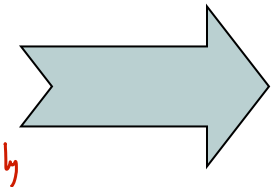
In our example, suppose we had

h = 20

c = 10

d = 10

$\mu^{(0)} = 0$ randomly



t	$\mu^{(t)}$	$b^{(t)}$
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

20.3187 got As

Convergence is generally linear: error decreases by a constant factor each time step.