Clustering with Gaussian Mixtures

Andrew W. Moore
Professor
School of Computer Science
Carnegie Mellon University
www.cs.cmu.edu/~awm
awm@cs.cmu.edu
412-268-7599

Unsupervised Learning

- You walk into a bar.
  A stranger approaches and tells you:
  > "I've got data from k classes. Each class produces observations with a normal distribution and variance $\sigma^2 I$. Standard simple multivariate gaussian assumptions. I can tell you all the $P(w_i)$'s ."
- So far, looks straightforward.
  > "I need a maximum likelihood estimate of the μ_i's ."
- No problem:
  > "There's just one thing. None of the data are labeled. I have datapoints, but I don't know what class they're from (any of them!)
- Uh oh!!

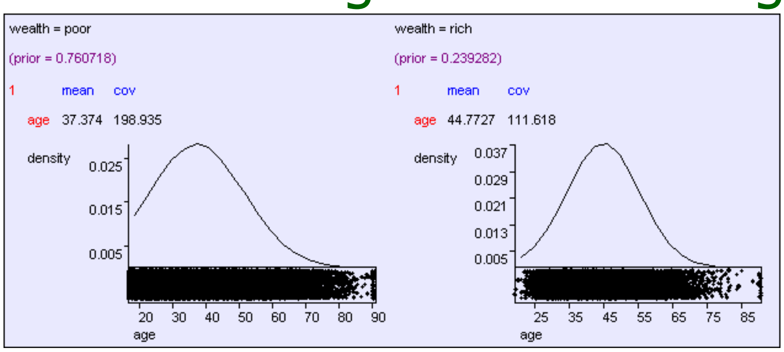
Gaussian Bayes Classifier Reminder

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

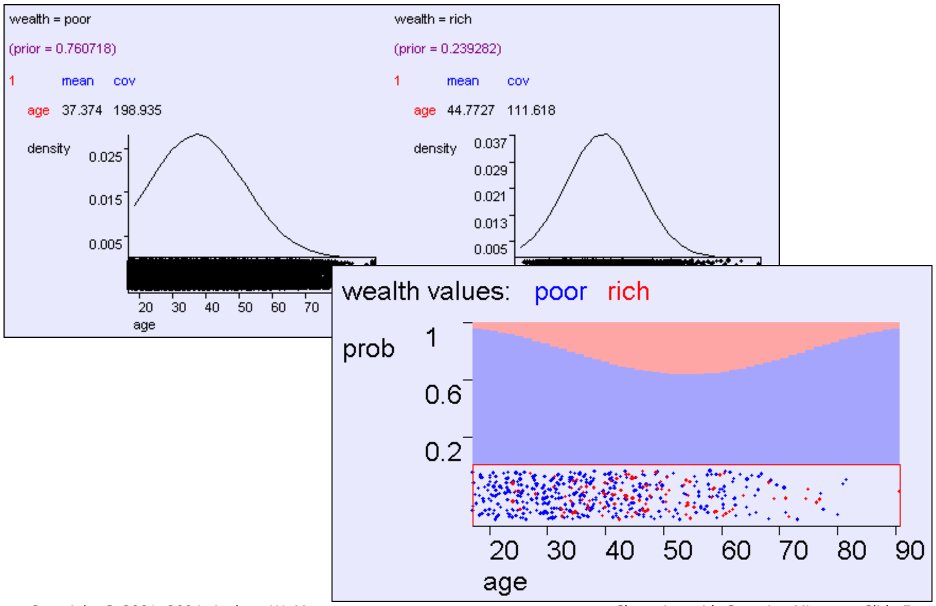
$$P(y=i\,|\,\mathbf{x}) = \frac{\dfrac{1}{(2\pi)^{m/2}\,\|\,\boldsymbol{\Sigma}_i\,\|^{1/2}}\exp\left[-\dfrac{1}{2}\left(\mathbf{x}_k-\boldsymbol{\mu}_i\right)^T\boldsymbol{\Sigma}_i\left(\mathbf{x}_k-\boldsymbol{\mu}_i\right)\right]p_i}{p(\mathbf{x})}$$

How do we deal with that?

Predicting wealth from age

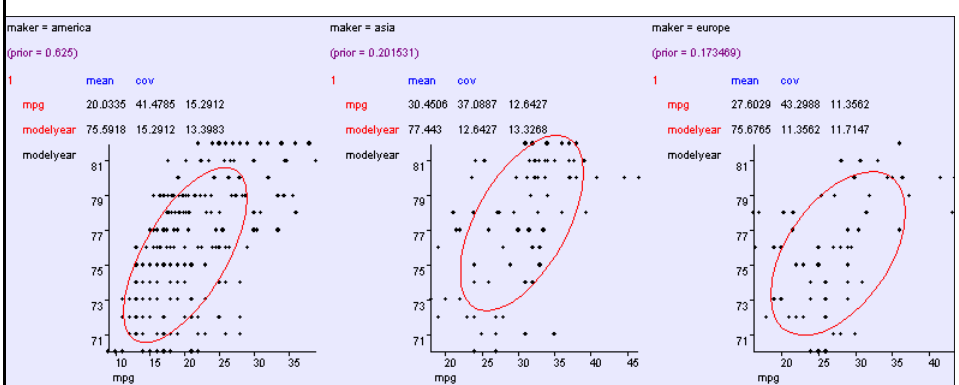


Predicting wealth from age



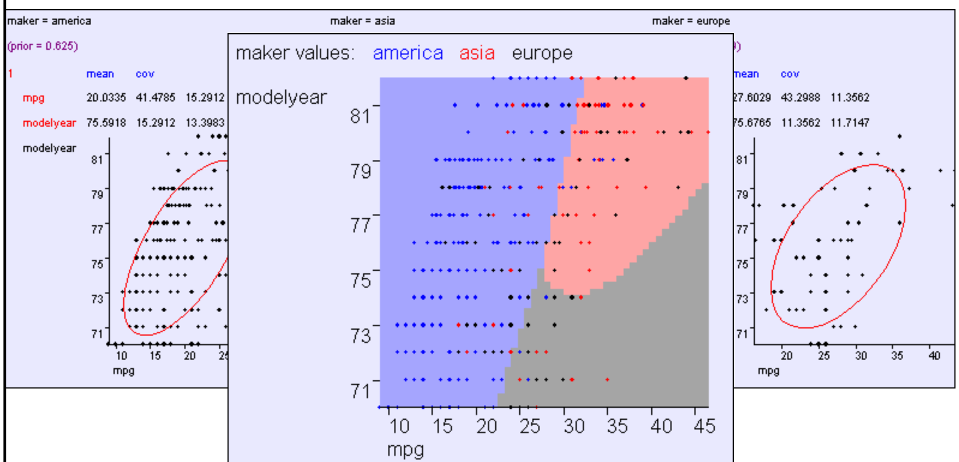
Learning modelyear , mpg ---> maker

$$\Sigma = \begin{pmatrix} \sigma^2{}_1 & \sigma_{12} & \cdots & \sigma_{1m} \\ \sigma_{12} & \sigma^2{}_2 & \cdots & \sigma_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{1m} & \sigma_{2m} & \cdots & \sigma^2{}_m \end{pmatrix}$$



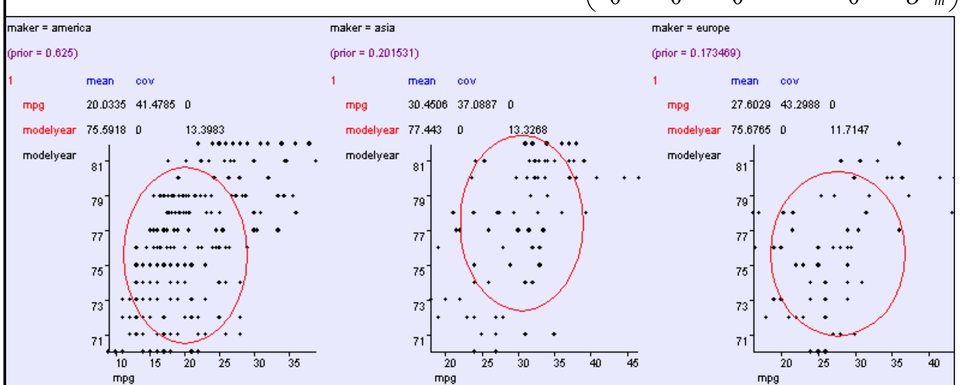
General: $O(m^2)$ parameters

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma^2{}_1 & \sigma_{12} & \cdots & \sigma_{1m} \\ \sigma_{12} & \sigma^2{}_2 & \cdots & \sigma_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{1m} & \sigma_{2m} & \cdots & \sigma^2{}_m \end{pmatrix}$$



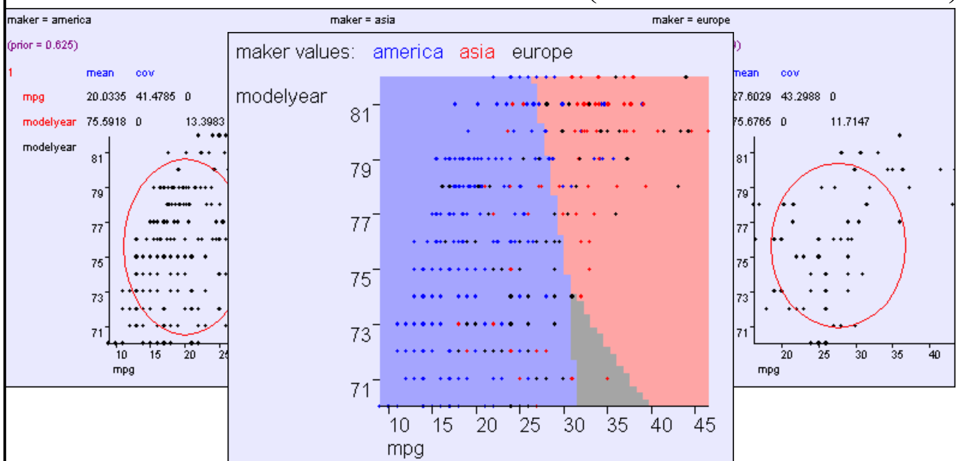
Aligned: $O(m)$ parameters

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma^2{}_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \sigma^2{}_2 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \sigma^2{}_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \sigma^2{}_{m-1} & 0 \\ 0 & 0 & 0 & \cdots & 0 & \sigma^2{}_m \end{pmatrix}$$



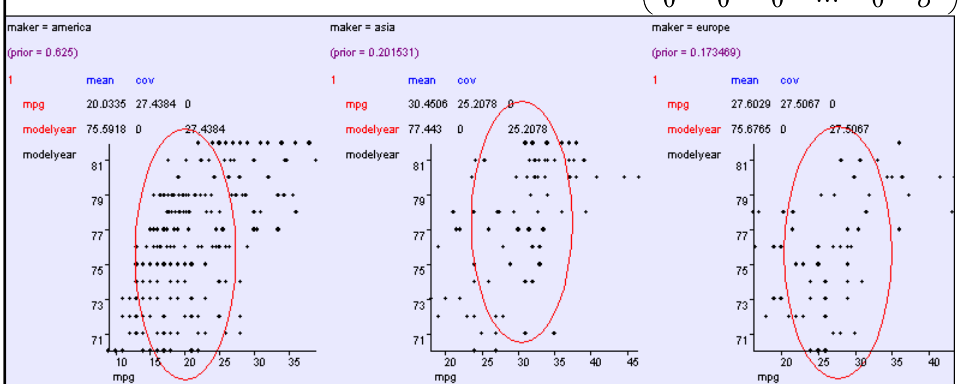
Aligned: *O(m)* parameters

$$
\Sigma = \begin{pmatrix}
\sigma^2{}_1 & 0 & 0 & \cdots & 0 & 0 \\
0 & \sigma^2{}_2 & 0 & \cdots & 0 & 0 \\
0 & 0 & \sigma^2{}_3 & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & \cdots & \sigma^2{}_{m-1} & 0 \\
0 & 0 & 0 & \cdots & 0 & \sigma^2{}_m
\end{pmatrix}
$$



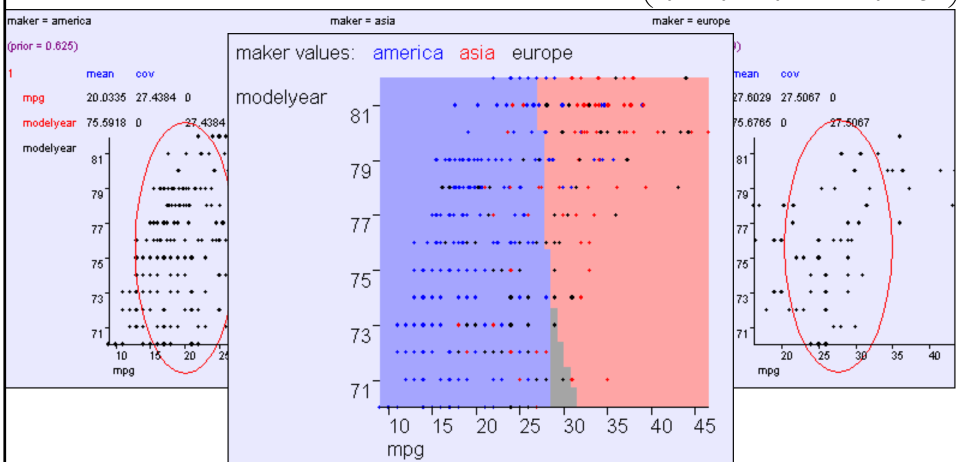
Spherical: *O(1)* cov parameters

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



Spherical: *O(1)* cov parameters

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

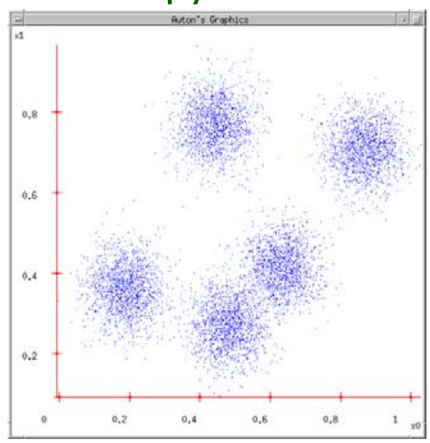


Making a Classifier from a Density Estimator

	Categorical inputs only	Real-valued inputs only	Mixed Real / Cat okay
Inputs → Classifier → Predict category	Joint BC Naïve BC	Gauss BC	Dec Tree
Inputs → Density Estimator → Prob-ability	Joint DE Naïve DE	Gauss DE	
Inputs → Regressor → Predict real no.			

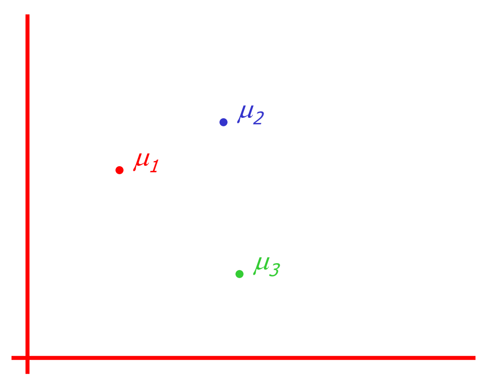
Next… back to Density Estimation

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



The GMM assumption

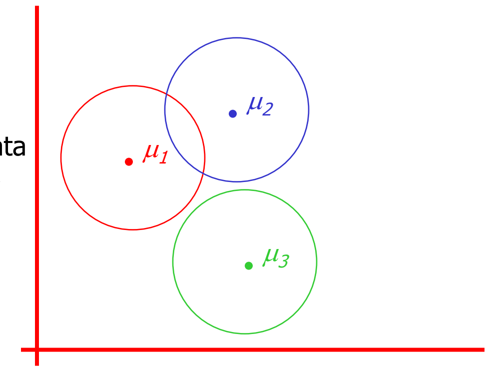
- There are k components. The i'th component is called ω_i
- Component ω_i has an associated mean vector μ_i



The GMM assumption

- There are k components. The i'th component is called ω_i
- Component ω_i has an associated mean vector μ_i
- Each component generates data from a Gaussian with mean μ_i and covariance matrix $\sigma^2 \mathbf{I}$

Assume that each datapoint is generated according to the following recipe:

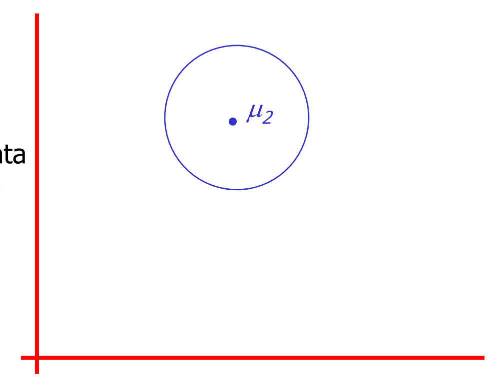


The GMM assumption

- There are k components. The i'th component is called ω_i
- Component ω_i has an associated mean vector μ_i
- Each component generates data from a Gaussian with mean μ_i and covariance matrix $\sigma^2 \mathbf{I}$

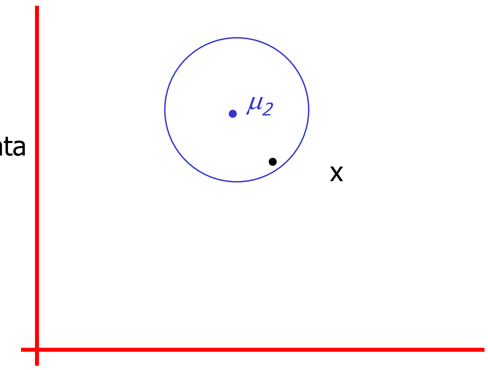
Assume that each datapoint is generated according to the following recipe:

1. Pick a component at random. Choose component i with probability $P(\omega_i)$.



The GMM assumption

- There are k components. The i'th component is called ω_i
- Component ω_i has an associated mean vector μ_i
- Each component generates data from a Gaussian with mean μ_i and covariance matrix $\sigma^2 \mathbf{I}$

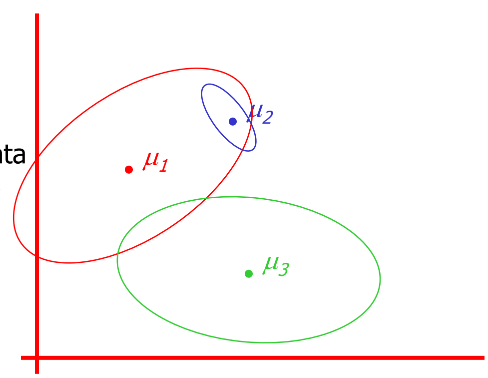


Assume that each datapoint is generated according to the following recipe:

1. Pick a component at random. Choose component i with probability $P(\omega_i)$.
2. Datapoint ~ N(μ_i, $\sigma^2 \mathbf{I}$)

The General GMM assumption

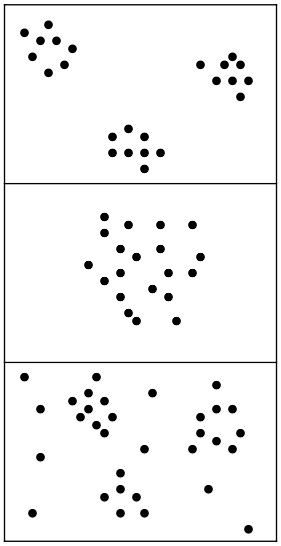
- There are k components. The i'th component is called ω_i
- Component ω_i has an associated mean vector μ_i
- Each component generates data from a Gaussian with mean μ_i and covariance matrix Σ_i



Assume that each datapoint is generated according to the following recipe:

1. Pick a component at random. Choose component i with probability $P(\omega_i)$.
2. Datapoint ~ N(μ_i, Σ_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*

Computing likelihoods in unsupervised case

We have $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_N$

We know $P(w_1)\ P(w_2)\ ..\ P(w_k)$

We know σ

$P(\boldsymbol{x}|w_i, \boldsymbol{\mu}_i, \ldots \boldsymbol{\mu}_k)$ = Prob that an observation from class w_i would have value $\boldsymbol{x}$ given class means $\boldsymbol{\mu}_1 \ldots \boldsymbol{\mu}_x$

Can we write an expression for that?

likelihoods in unsupervised case

We have $\mathbf{x}_1\ \mathbf{x}_2 \ldots \mathbf{x}_n$

We have $P(w_1)$ .. $P(w_k)$. We have σ.

We can define, for any $\mathbf{x}$, $P(\mathbf{x}|w_i, \mu_1, \mu_2 .. \mu_k)$

Can we define $P(\mathbf{x} \mid \mu_1, \mu_2 .. \mu_k)$?

Can we define $P(\mathbf{x}_1, \mathbf{x}_1, .. \mathbf{x}_n \mid \mu_1, \mu_2 .. \mu_k)$?

[YES, IF WE ASSUME THE X_1'S WERE DRAWN INDEPENDENTLY]

Unsupervised Learning: Mediumly Good News

We now have a procedure s.t. if you give me a guess at $\mu_1, \mu_2 .. \mu_k$,

I can tell you the prob of the unlabeled data given those μ's.

Suppose x's are 1-dimensional.

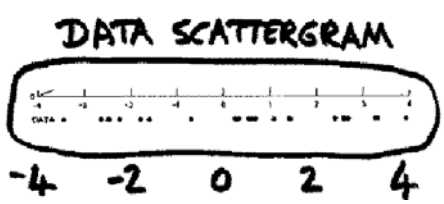
(From Duda and Hart)

There are two classes; w_1 and w_2

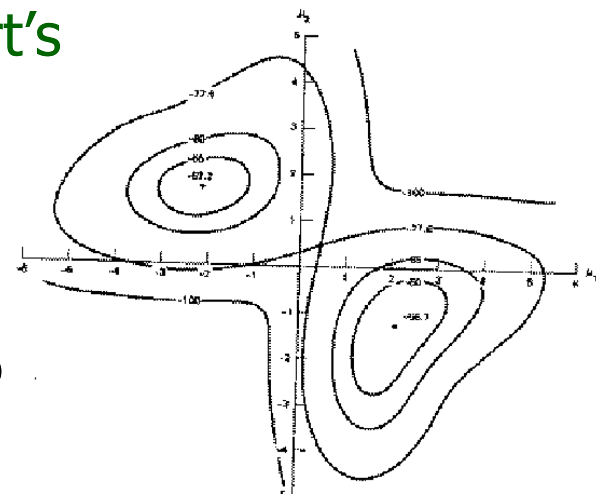
$P(w_1) = 1/3$ $P(w_2) = 2/3$ $\sigma = 1$.

There are 25 unlabeled datapoints

$x_1 =$ 0.608
$x_2 =$ -1.590
$x_3 =$ 0.235
$x_4 =$ 3.949
:
$x_{25} =$ -0.712



Duda & Hart's Example



Graph of
log P(x_1, x_2 .. x_{25} | μ_1, μ_2)
against μ_1 ($\rightarrow$) and μ_2 ($\uparrow$)

Max likelihood = (μ_1 =-2.13, μ_2 =1.668)

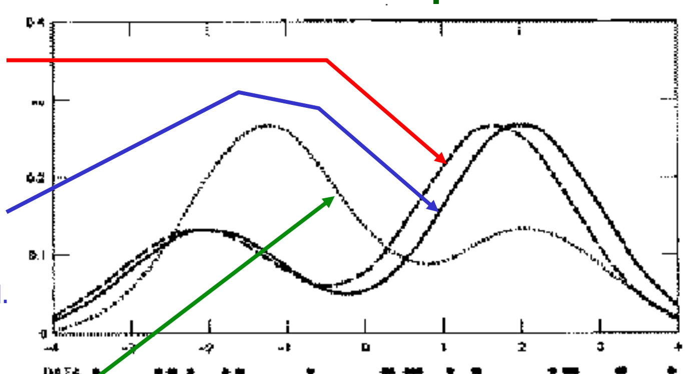
Local minimum, but very close to global at (μ_1 =2.085, μ_2 =-1.257)*

* corresponds to switching w_1 + w_2.

Duda & Hart's Example

We can graph the prob. dist. function of data given our μ_1 and μ_2 estimates.

We can also graph the true function from which the data was randomly generated.



- They are close. Good.
- The 2nd solution tries to put the "2/3" hump where the "1/3" hump should go, and vice versa.
- In this example unsupervised is almost as good as supervised. If the x_1 .. x_{25} are given the class which was used to learn them, then the results are (μ_1=-2.176, μ_2=1.684). Unsupervised got (μ_1=-2.13, μ_2=1.668).

Finding the max likelihood $\mu_1,\mu_2..\mu_k$

We can compute P(data | $\boldsymbol{\mu}_1,\boldsymbol{\mu}_2..\boldsymbol{\mu}_k$)

How do we find the $\boldsymbol{\mu}_i$'s which give max. likelihood?

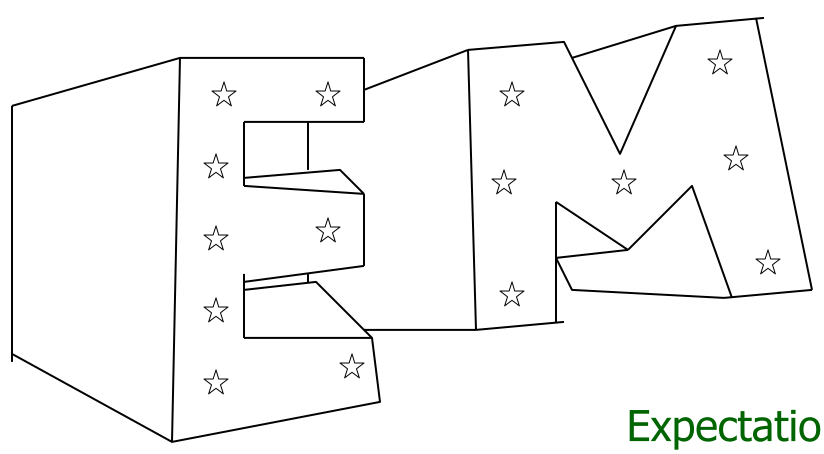
- The normal max likelihood trick:

  Set $\frac{☆}{☆\ \mu_i}$ log Prob (….) = 0

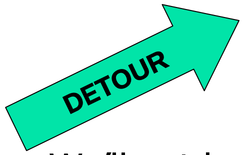
  and solve for μ_i's.

  \# Here you get non-linear non-analytically-solvable equations
- Use gradient descent

  Slow but doable
- Use a much faster, cuter, and recently very popular method…



Expectation
Maximalization



The E.M. Algorithm

- 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, including inference of Hidden Markov Models (future lecture).

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 ?

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

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

$P(a,b,c,d \mid \mu) = K(½)^a(\mu)^b(2\mu)^c(½-3\mu)^d$

$\log P(a,b,c,d \mid \mu) = \log K + a\log ½ + b\log \mu + c\log 2\mu + d\log (½-3\mu)$

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

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

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

So if class got

A	B	C	D
14	6	9	10

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

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?

REMEMBER

P(A) = ½

P(B) = μ

P(C) = 2μ

P(D) = ½-3μ

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?

REMEMBER

P(A) = ½

P(B) = μ

P(C) = 2μ

P(D) = ½-3μ

We can answer this question circularly:

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

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

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 μ(t) the estimate of μ on the t'th iteration
b(t) the estimate of *b* on t'th iteration

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

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

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.

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]

In our example, suppose we had
h = 20
c = 10
d = 10
μ(0) = 0

t	μ(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

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

Back to Unsupervised Learning of GMMs

Remember:

We have unlabeled data $x_1\ x_2\ \ldots\ x_R$

We know there are k classes

We know $P(w_1)\ P(w_2)\ P(w_3)\ \ldots\ P(w_k)$

We <u>don't</u> know $\boldsymbol{\mu}_1\ \boldsymbol{\mu}_2\ ..\ \boldsymbol{\mu}_k$

We can write P(data | $\boldsymbol{\mu}_1 \ldots \boldsymbol{\mu}_k$)

$$= p(x_1 \ldots x_R | \mu_1 \ldots \mu_k)$$

$$= \prod_{i=1}^{R} p(x_i | \mu_1 \ldots \mu_k)$$

$$= \prod_{i=1}^{R} \sum_{j=1}^{k} p(x_i | w_j, \mu_1 \ldots \mu_k) P(w_j)$$

$$= \prod_{i=1}^{R} \sum_{j=1}^{k} K \exp\left(-\frac{1}{2\sigma^2}(x_i - \mu_j)^2\right) P(w_j)$$

E.M. for GMMs

For Max likelihood we know $\frac{\partial}{\partial \mu_i} \log \Pr ob(\text{data}|\mu_1 \ldots \mu_k) = 0$

Some wild'n'crazy algebra turns this into : "For Max likelihood, for each j,

$$\mu_j = \frac{\sum_{i=1}^{R} P(w_j | x_i, \mu_1 \ldots \mu_k)\, x_i}{\sum_{i=1}^{R} P(w_j | x_i, \mu_1 \ldots \mu_k)}$$

responsibility?

How do we get this?

See

http://www.cs.cmu.edu/~awm/doc/gmm-algebra.pdf

This is n nonlinear equations in $\boldsymbol{\mu}_j$'s."

If, for each $\mathbf{x}_i$ we knew that for each w_j the prob that $\boldsymbol{\mu}_j$ was in class w_j is $P(w_j|x_i,\mu_1 \ldots \mu_k)$ Then… we would easily compute μ_j.

If we knew each μ_j then we could easily compute $P(w_j|x_i,\mu_1 \ldots \mu_k)$ for each w_j and x_i.

…I feel an EM experience coming on!!

E.M. for GMMs

Iterate. On the *t*'th iteration let our estimates be

$$\lambda_t = \{ \mu_1(t), \mu_2(t) \ldots \mu_c(t) \}$$

responsibility:

$E(w_i | x_k, \lambda_t)$

$= P(w_i = 1 | x_k, \lambda_t)$

E-step

Compute "expected" classes of all datapoints for each class

Just evaluate a Gaussian at x_k

$$P(w_i | x_k, \lambda_t) = \frac{p(x_k | w_i, \lambda_t) P(w_i | \lambda_t)}{p(x_k | \lambda_t)} = \frac{p(x_k | w_i, \mu_i(t), \sigma^2 \mathbf{I}) p_i(t)}{\sum_{j=1}^{c} p(x_k | w_j, \mu_j(t), \sigma^2 \mathbf{I}) p_j(t)}$$

加t是因为 $P(w_i)$ 也是一个可以迭代更新的项

M-step.

Compute Max. like μ given our data's class membership distributions

$$\mu_i(t+1) = \frac{\sum_k P(w_i | x_k, \lambda_t) x_k}{\sum_k P(w_i | x_k, \lambda_t)}$$

→ subject to max likelihood

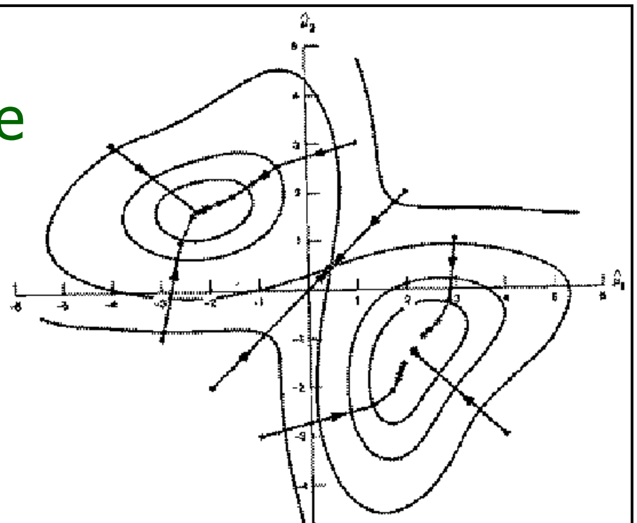
$P(w_i | \lambda_t)$

$= P(w_i)$

二者独立

E.M. Convergence

- Your lecturer will (unless out of time) give you a nice intuitive explanation of why this rule works.
- As with all EM procedures, convergence to a local optimum guaranteed.



- This algorithm is REALLY USED. And in high dimensional state spaces, too. E.G. Vector Quantization for Speech Data

E.M. for General GMMs

Iterate. On the t'th iteration let our estimates be

$$\lambda_t = \{ \mu_1(t), \mu_2(t) \ldots \mu_c(t), \Sigma_1(t), \Sigma_2(t) \ldots \Sigma_c(t), p_1(t), p_2(t) \ldots p_c(t) \}$$

$p_i(t)$ is shorthand for estimate of $P(\omega_i)$ on t'th iteration

E-step

Compute "expected" classes of all datapoints for each class

$$\mathrm{P}(w_i | x_k, \lambda_t) = \frac{\mathrm{p}(x_k | w_i, \lambda_t)\mathrm{P}(w_i|\lambda_t)}{\mathrm{p}(x_k|\lambda_t)} = \frac{\mathrm{p}(x_k | w_i, \mu_i(t), \Sigma_i(t)) p_i(t)}{\sum_{j=1}^{c} \mathrm{p}(x_k | w_j, \mu_j(t), \Sigma_j(t)) p_j(t)}$$

Just evaluate a Gaussian at x_k

$P(w_i|\lambda_t)$

$= p_i(t)$

已经开始忘了

M-step.

Compute Max. like **μ** given our data's class membership distributions

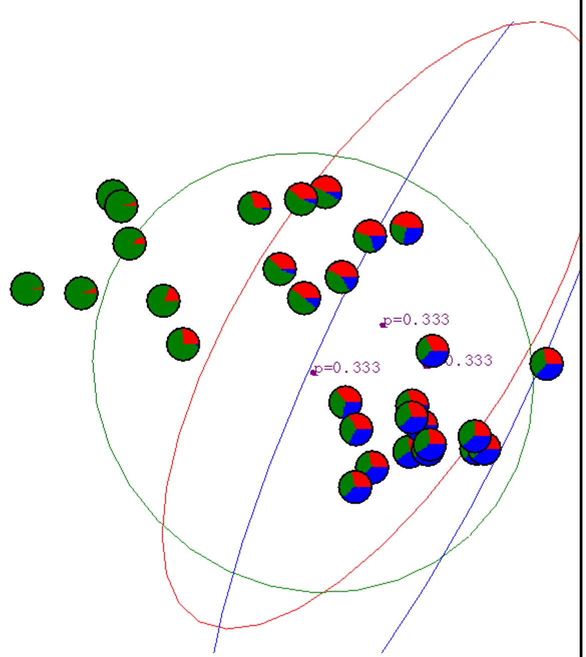
$$\mu_i(t+1) = \frac{\sum_k \mathrm{P}(w_i | x_k, \lambda_t) x_k}{\sum_k \mathrm{P}(w_i | x_k, \lambda_t)} \qquad \Sigma_i(t+1) = \frac{\sum_k \mathrm{P}(w_i | x_k, \lambda_t)[x_k - \mu_i(t+1)][x_k - \mu_i(t+1)]^T}{\sum_k \mathrm{P}(w_i | x_k, \lambda_t)}$$

有时间自己做一下

$$p_i(t+1) = \frac{\sum_k \mathrm{P}(w_i | x_k, \lambda_t)}{R}$$

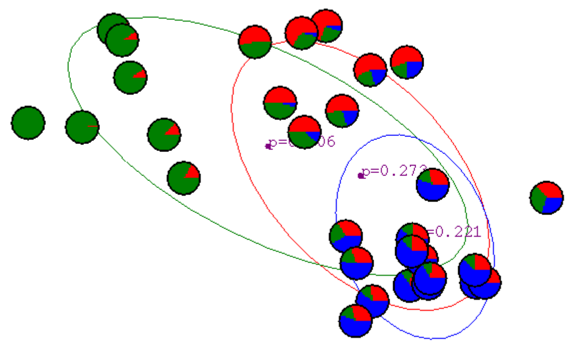
R = #records

Gaussian Mixture Example: Start

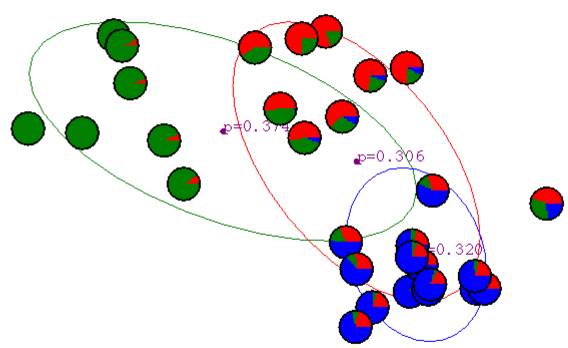


Advance apologies: in Black and White this example will be incomprehensible

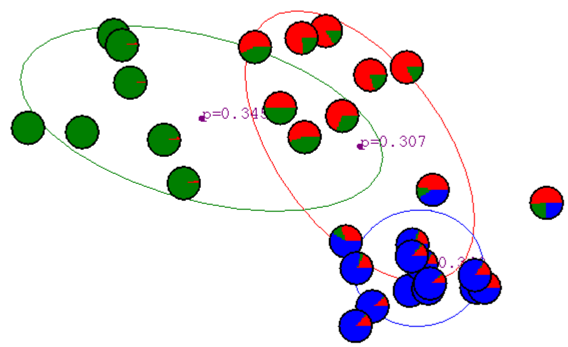
After first iteration



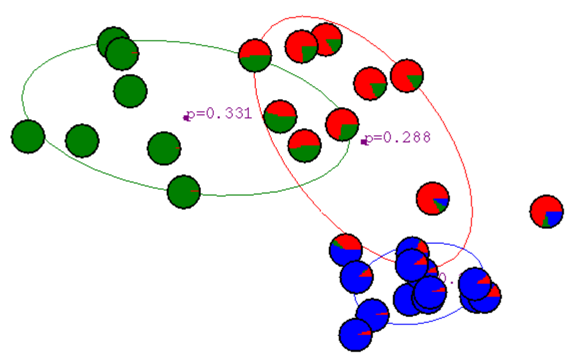
After 2nd iteration



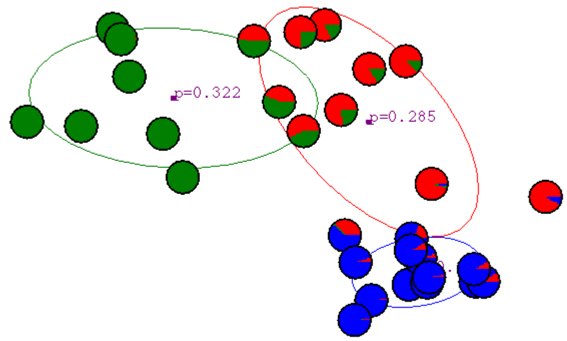
After 3rd iteration



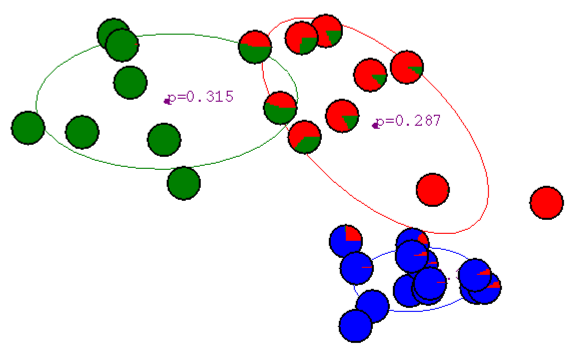
After 4th iteration



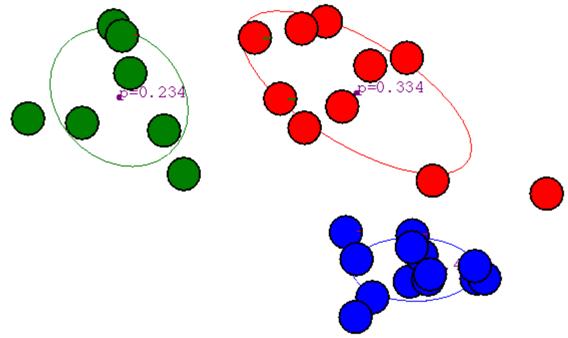
After 5th iteration



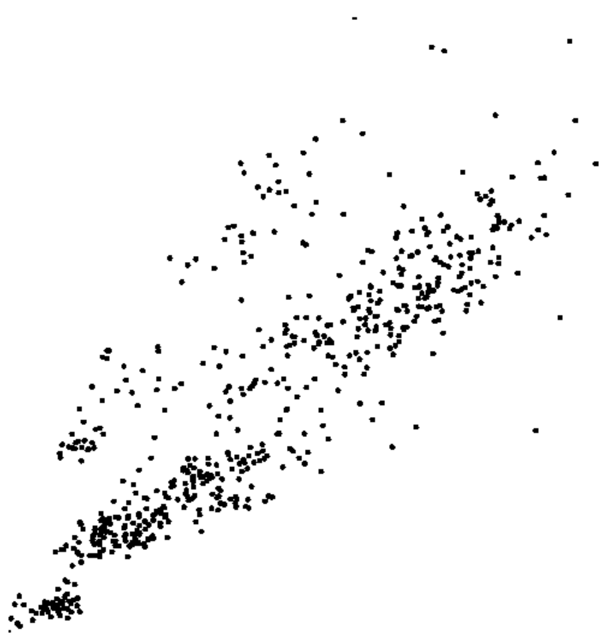
After 6th iteration



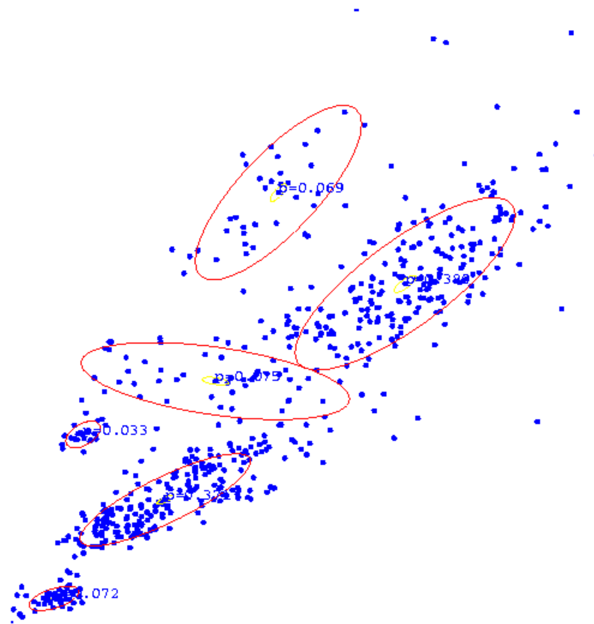
After 20th iteration



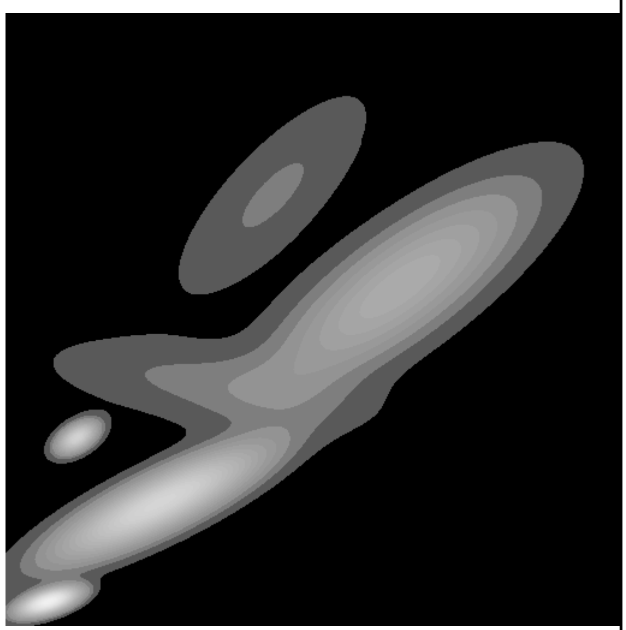
Some Bio Assay data



GMM clustering of the assay data



Resulting Density Estimator



Where are we now?

Inputs → Inference Engine Learn	$P(E_1 \mid E_2)$	Joint DE, Bayes Net Structure Learning
Inputs → Classifier	Predict category	Dec Tree, Sigmoid Perceptron, Sigmoid N.Net, Gauss/Joint BC, Gauss Naïve BC, N.Neigh, Bayes Net Based BC, Cascade Correlation
Inputs → Density Estimator	Prob-ability	Joint DE, Naïve DE, Gauss/Joint DE, Gauss Naïve DE, Bayes Net Structure Learning, GMMs
Inputs → Regressor	Predict real no.	Linear Regression, Polynomial Regression, Perceptron, Neural Net, N.Neigh, Kernel, LWR, RBFs, Robust Regression, Cascade Correlation, Regression Trees, GMDH, Multilinear Interp, MARS

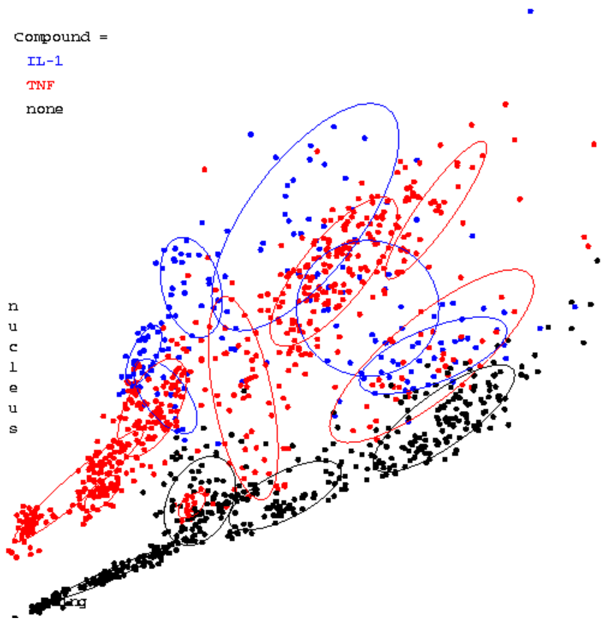
The old trick…

Inputs → Inference Engine Learn	$P(E_1 \mid E_2)$	Joint DE, Bayes Net Structure Learning
Inputs → Classifier	Predict category	Dec Tree, Sigmoid Perceptron, Sigmoid N.Net, Gauss/Joint BC, Gauss Naïve BC, N.Neigh, Bayes Net Based BC, Cascade Correlation, GMM-BC
Inputs → Density Estimator	Prob-ability	Joint DE, Naïve DE, Gauss/Joint DE, Gauss Naïve DE, Bayes Net Structure Learning, GMMs
Inputs → Regressor	Predict real no.	Linear Regression, Polynomial Regression, Perceptron, Neural Net, N.Neigh, Kernel, LWR, RBFs, Robust Regression, Cascade Correlation, Regression Trees, GMDH, Multilinear Interp, MARS

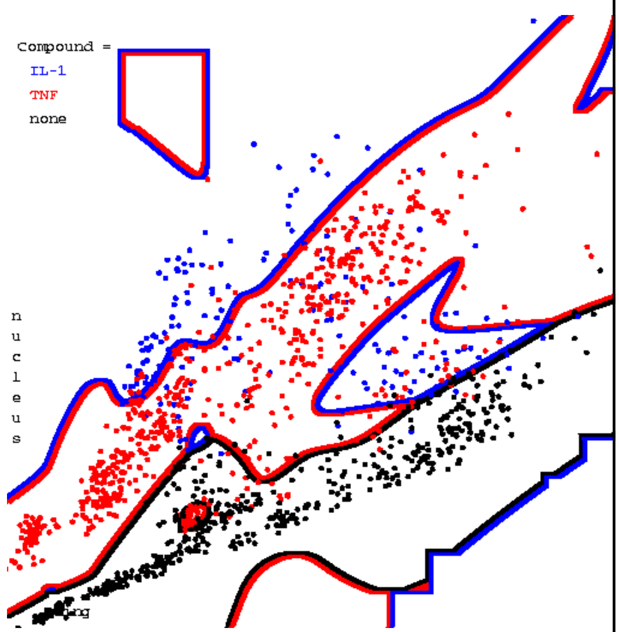
Three classes of assay

(each learned with it's own mixture model)

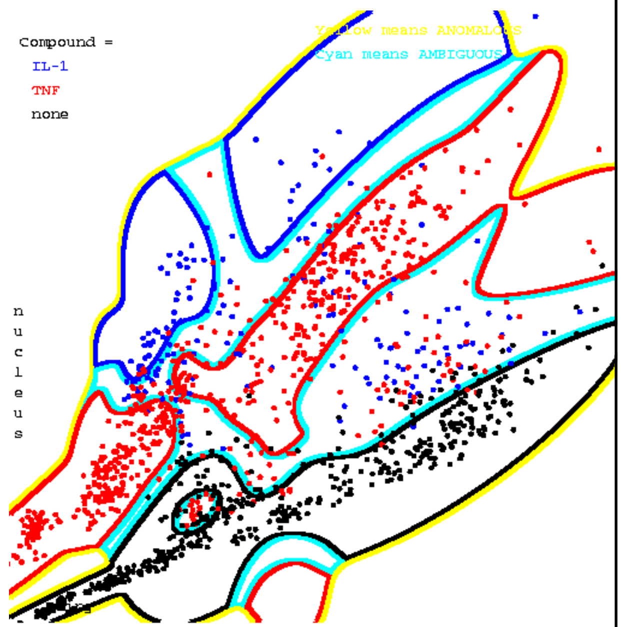
(Sorry, this will again be semi-useless in black and white)



Resulting Bayes Classifier



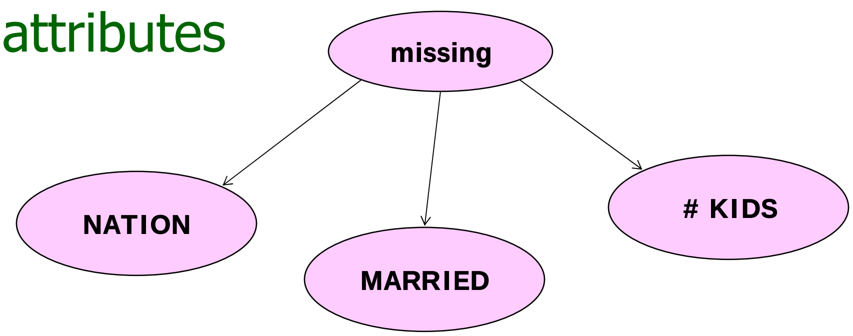
Resulting Bayes Classifier, using posterior probabilities to alert about ambiguity and anomalousness



Yellow means anomalous

Cyan means ambiguous

Unsupervised learning with symbolic attributes



It's just a "learning Bayes net with known structure but hidden values" problem.

Can use Gradient Descent.

<u>EASY</u>, fun exercise to do an EM formulation for this case too.

Final Comments

- Remember, E.M. can get stuck in local minima, and empirically it DOES.
- Our unsupervised learning example assumed $P(w_i)$'s known, and variances fixed and known. Easy to relax this.
- It's possible to do Bayesian unsupervised learning instead of max. likelihood.
- There are other algorithms for unsupervised learning. We'll visit K-means soon. Hierarchical clustering is also interesting.
- Neural-net algorithms called "competitive learning" turn out to have interesting parallels with the EM method we saw.

What you should know

- How to "learn" maximum likelihood parameters (locally max. like.) in the case of unlabeled data.
- Be happy with this kind of probabilistic analysis.
- Understand the two examples of E.M. given in these notes.

For more info, see Duda + Hart. It's a great book. There's much more in the book than in your handout.

Other unsupervised learning methods

- K-means (see next lecture)
- Hierarchical clustering (e.g. Minimum spanning trees) (see next lecture)
- Principal Component Analysis
    - simple, useful tool
- Non-linear PCA
    - Neural Auto-Associators
    - Locally weighted PCA
    - Others…