Bayesian Networks –(Structure) Learning

Machine Learning – 10701/15781

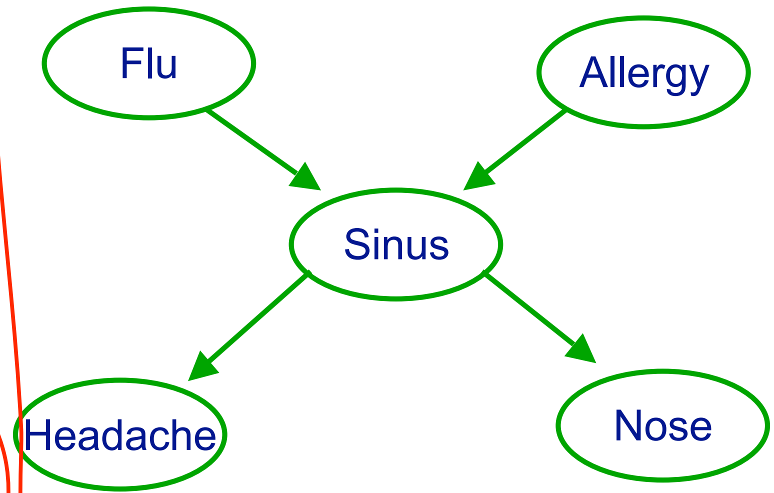
Carlos Guestrin

Carnegie Mellon University

April 2nd, 2007

Review

- Bayesian Networks
  - Compact representation for probability distributions
  - Exponential reduction in number of parameters
- Fast probabilistic inference using variable elimination
  - Compute P(X|e)
  - Time exponential in tree-width, not number of variables
- Today
  - Learn BN structure

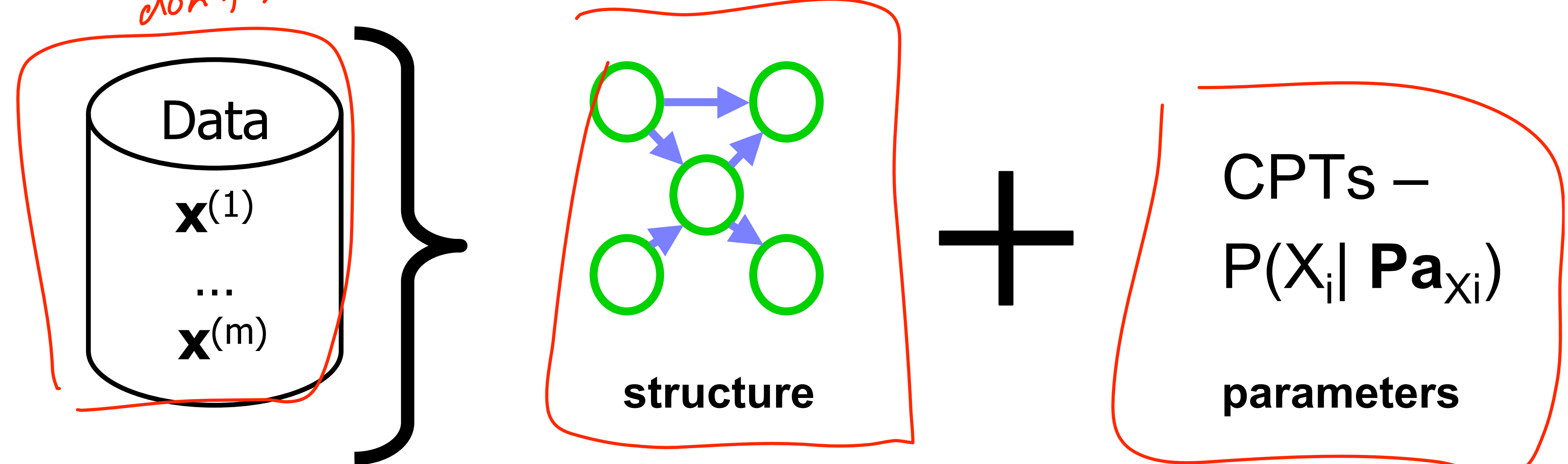


Learning Bayes nets

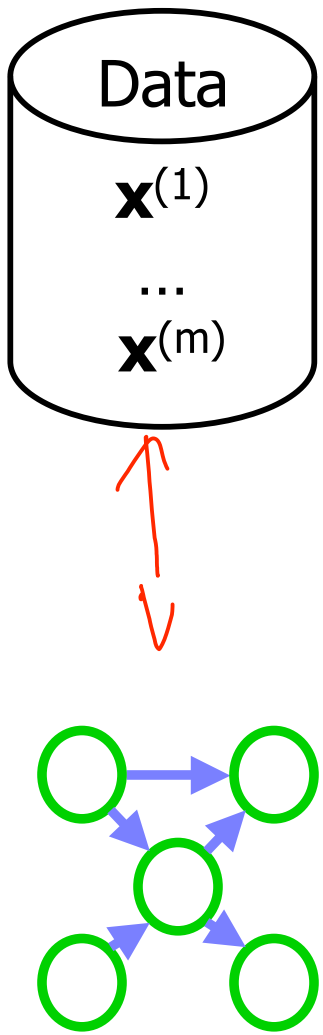
	Known structure	Unknown structure
Fully observable data ⟨A=t, H=f, S=t, F=t⟩	very easy!! ☺ ✓	learning "good" structure hard … next week
Missing data	hard … talk about in two weeks	really really hard … we'll talk about it next semester

⟨A=?, H=f, S=t, F=?, N=t⟩

? don't know



Learning the CPTs



For each discrete variable X_i

Want to learn $P(X_i \mid Pa_{X_i})$

$P(S=t \mid F=t, A=f) = \frac{\text{Count}(S=t, F=t, A=f)}{\text{Count}(F=t, A=f)}$

Maximum likelihood estimates

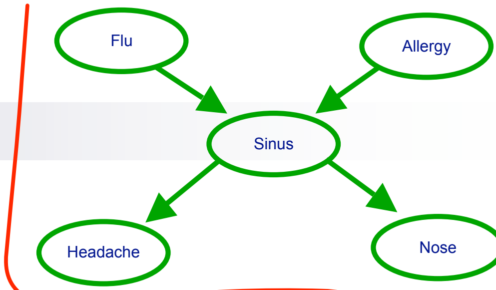
set of parents

MLE: $P(X_i = x_i \mid X_j = x_j) = \frac{\text{Count}(X_i = x_i, X_j = x_j)}{\text{Count}(X_j = x_j)}$

Information-theoretic interpretation of maximum likelihood

- Given structure, log likelihood of data:

$$\log P(\mathcal{D} \mid \theta_{\mathcal{G}}, \mathcal{G})$$



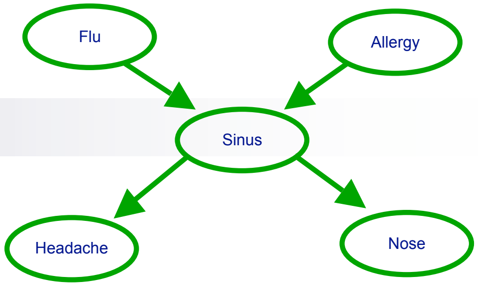
$|D| = m$

$$= \log \prod_{j=1}^{m} P(f^{(j)}, a^{(j)}, s^{(j)}, h^{(j)}, n^{(j)} \mid \theta_G, G)$$

what is G

$$= \log \prod_{j=1}^{m} P(f^{(j)} \mid \theta_F, G) \cdot P(a^{(j)} \mid \theta_A, G) \cdot P(s^{(j)} \mid f^{(j)}, a^{(j)}, \theta_{S|FA}, G) \cdot P(h^{(j)} \mid s^{(j)}, \theta_{H|S}, G) \, P(n^{(j)} \mid s^{(j)}, \theta_{N|S}, G)$$

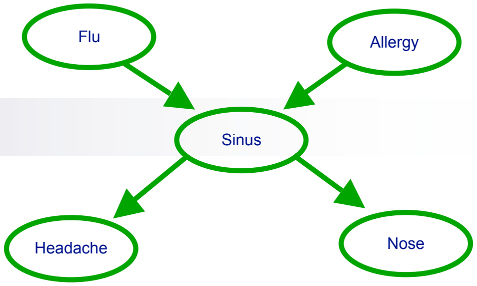
Information-theoretic interpretation of maximum likelihood



- Given structure, log likelihood of data:

$$\log P(\mathcal{D} \mid \theta_{\mathcal{G}}, \mathcal{G}) = \sum_{j=1}^{m} \sum_{i=1}^{n} \log P\left(X_i = x_i^{(j)} \mid \mathbf{Pa}_{X_i} = \mathbf{x}^{(j)}\left[\mathbf{Pa}_{X_i}\right]\right)$$

Information-theoretic interpretation of maximum likelihood 2



- Given structure, log likelihood of data:

$$\log \hat{P}(\mathcal{D} \mid \theta, \mathcal{G}) = m \sum_i \sum_{x_i, \mathbf{Pa}_{x_i,\mathcal{G}}} \hat{P}(x_i, \mathbf{Pa}_{x_i,\mathcal{G}}) \log \hat{P}(x_i \mid \mathbf{Pa}_{x_i,\mathcal{G}})$$

Decomposable score

- Log data likelihood

$$\log \widehat{P}(\mathcal{D} \mid \theta, \mathcal{G}) = m \sum_i \widehat{I}(x_i, \mathbf{Pa}_{x_i,\mathcal{G}}) - M \sum_i \widehat{H}(X_i)$$

- Decomposable score:
  - Decomposes over families in BN (node and its parents)
  - Will lead to significant computational efficiency!!!
  - Score(G : D) = $\sum_i$ FamScore($X_i | \mathbf{Pa}_{X_i}$: D)

How many trees are there?

Nonetheless – Efficient optimal algorithm finds best tree

Scoring a tree 1: equivalent trees

$$\log \hat{P}(\mathcal{D} \mid \theta, \mathcal{G}) = M \sum_i \hat{I}(x_i, \mathbf{Pa}_{x_i,\mathcal{G}}) - M \sum_i \hat{H}(X_i)$$

Scoring a tree 2: similar trees

$$\log \hat{P}(\mathcal{D} \mid \theta, \mathcal{G}) = M \sum_i \hat{I}(x_i, \mathbf{Pa}_{x_i,\mathcal{G}}) - M \sum_i \hat{H}(X_i)$$

Chow-Liu tree learning algorithm 1

- For each pair of variables X_i, X_j
  - Compute empirical distribution:

$$\hat{P}(x_i, x_j) = \frac{\text{Count}(x_i, x_j)}{m}$$

  - Compute mutual information:

$$\hat{I}(X_i, X_j) = \sum_{x_i, x_j} \hat{P}(x_i, x_j) \log \frac{\hat{P}(x_i, x_j)}{\hat{P}(x_i)\hat{P}(x_j)}$$

- Define a graph
  - Nodes $X_1, \ldots, X_n$
  - Edge (i,j) gets weight $\hat{I}(X_i, X_j)$

Chow-Liu tree learning algorithm 2

$$\log \hat{P}(\mathcal{D} \mid \theta, \mathcal{G}) = M \sum_i \hat{I}(x_i, \mathbf{Pa}_{x_i,\mathcal{G}}) - M \sum_i \hat{H}(X_i)$$

- Optimal tree BN
  - Compute maximum weight spanning tree
  - Directions in BN: pick any node as root, breadth-first-search defines directions

Can we extend Chow-Liu 1

- Tree augmented naïve Bayes (TAN)
  [Friedman et al. '97]
  - Naïve Bayes model overcounts, because correlation between features not considered
  - Same as Chow-Liu, but score edges with:

$$\hat{I}(X_i, X_j \mid C) = \sum_{c, x_i, x_j} \hat{P}(c, x_i, x_j) \log \frac{\hat{P}(x_i, x_j \mid c)}{\hat{P}(x_i \mid c)\hat{P}(x_j \mid c)}$$

Can we extend Chow-Liu 2

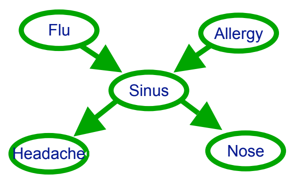
- (Approximately learning) models with tree-width up to *k*
  - [Narasimhan & Bilmes '04]
  - But, $O(n^{k+1})$…
    - and more subtleties

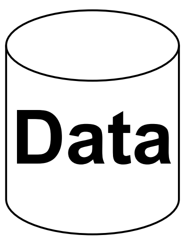
What you need to know about learning BN structures so far

- Decomposable scores
  - Maximum likelihood
  - Information theoretic interpretation
- Best tree (Chow-Liu)
- Best TAN
- Nearly best k-treewidth (in $O(N^{k+1})$)

Scoring general graphical models – Model selection problem

What's the best structure?





<x_1^{(1)},…,x_n^{(1)}>
…
<x_1^{(m)},…,x_n^{(m)}>

The more edges, the fewer independence assumptions, the higher the likelihood of the data, but will overfit…

Maximum likelihood overfits!

$$\log \widehat{P}(\mathcal{D} \mid \theta, \mathcal{G}) = M \sum_i \widehat{I}(x_i, \mathbf{Pa}_{x_i,\mathcal{G}}) - M \sum_i \widehat{H}(X_i)$$

- Information never hurts:

- Adding a parent always increases score!!!

Bayesian score avoids overfitting

- Given a structure, distribution over parameters

$$\log P(D \mid \mathcal{G}) = \log \int_{\theta_{\mathcal{G}}} P(D \mid \mathcal{G}, \theta_{\mathcal{G}}) P(\theta_{\mathcal{G}} | \mathcal{G}) d\theta_{\mathcal{G}}$$

- Difficult integral: use Bayes information criterion (BIC) approximation (equivalent as M! 1)

$$\log P(D \mid \mathcal{G}) \approx \log P(D \mid \mathcal{G}, \theta_{\mathcal{G}}) - \frac{\text{NumberParams}(\mathcal{G})}{2} \log M + \mathcal{O}(1)$$

- Note: regularize with MDL score
- Best BN under BIC still NP-hard

How many graphs are there?

$$\sum_{k=1}^{n} \binom{n}{k} = 2^n - 1$$

Structure learning for general graphs

- In a tree, a node only has one parent

- **Theorem**:
  - The problem of learning a BN structure with at most *d* parents is NP-hard for any (fixed) *d*¸*2*

- Most structure learning approaches use heuristics
  - Exploit score decomposition
  - (Quickly) Describe two heuristics that exploit decomposition in different ways

Learn BN structure using local search

Starting from Chow-Liu tree

Local search, possible moves:
- Add edge
- Delete edge
- Invert edge

Score using BIC

What you need to know about learning BNs

- Learning BNs
  - Maximum likelihood or MAP learns parameters
  - Decomposable score
  - Best tree (Chow-Liu)
  - Best TAN
  - Other BNs, usually local search with BIC score

Unsupervised learning or Clustering – K-means Gaussian mixture models

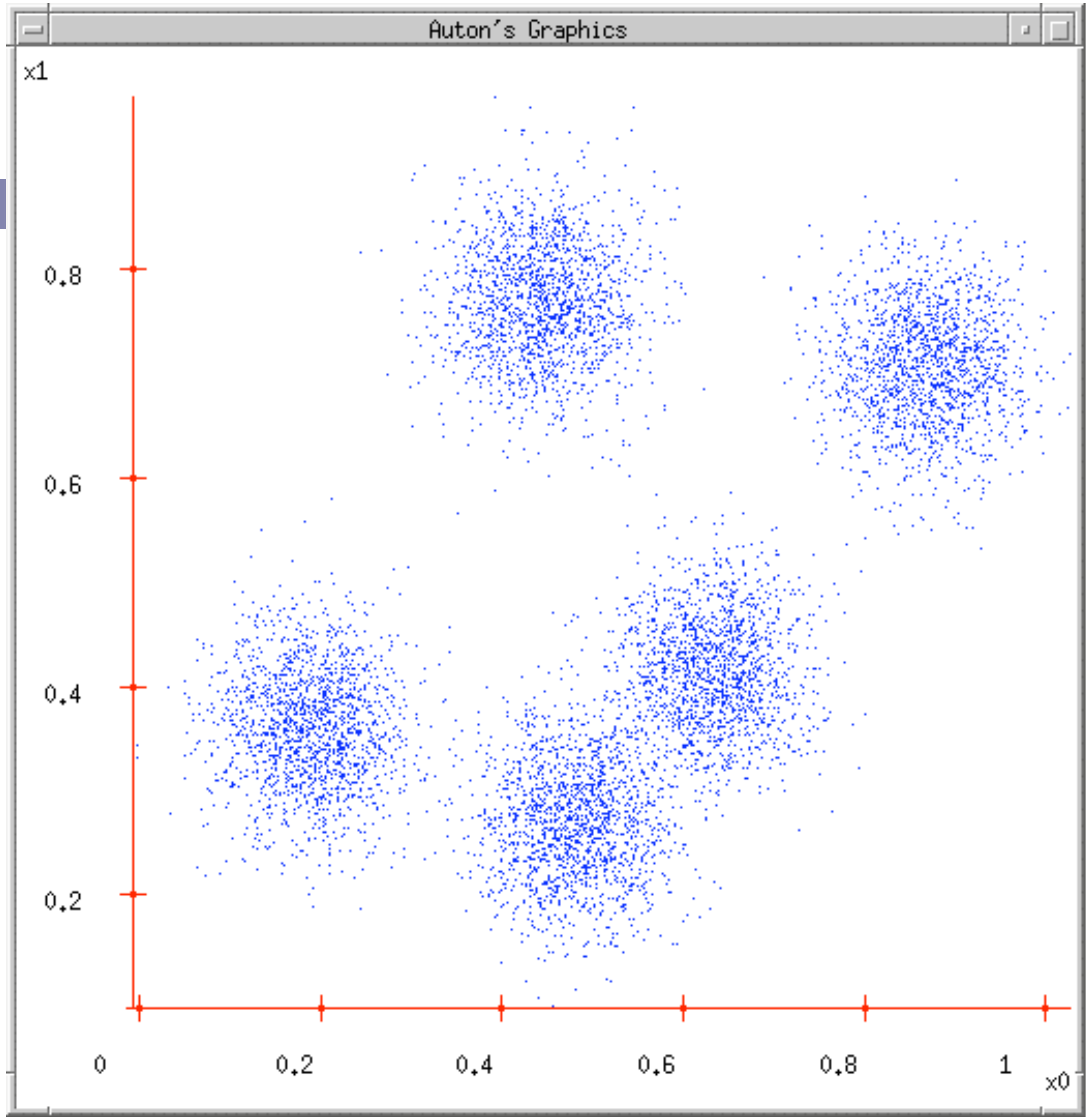
Machine Learning – 10701/15781

Carlos Guestrin

Carnegie Mellon University

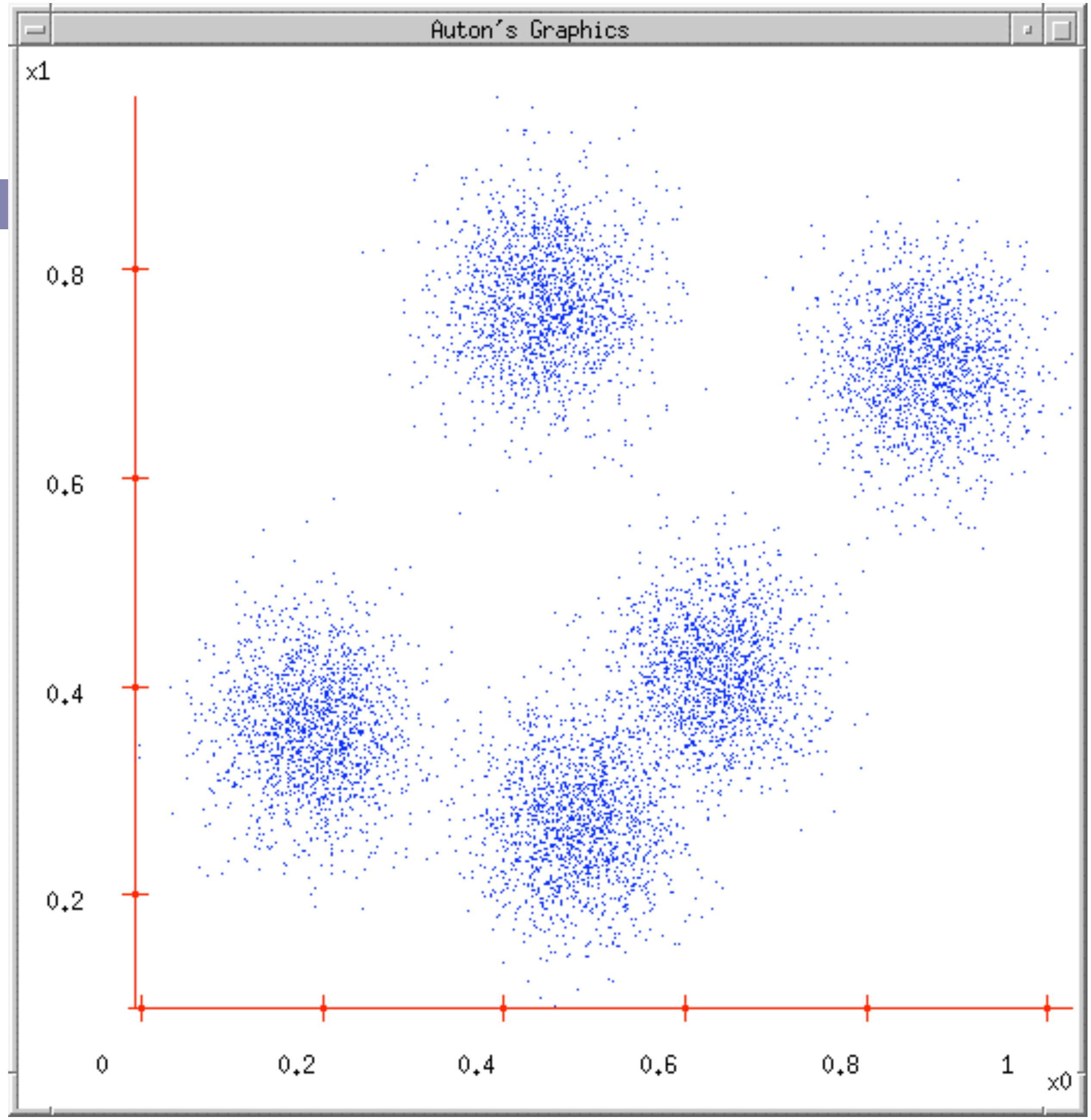
April 2nd, 2007

Some Data



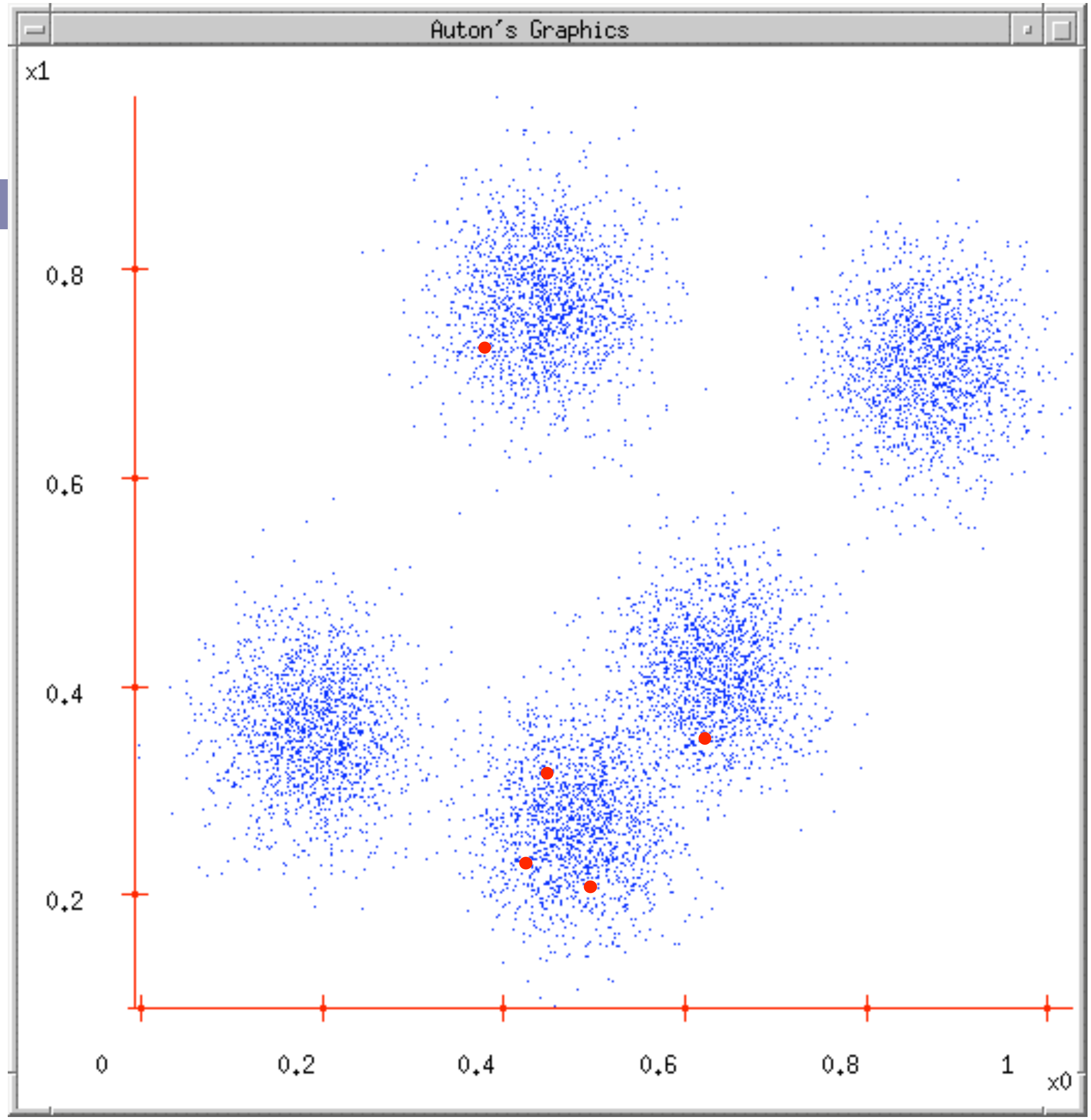
K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*



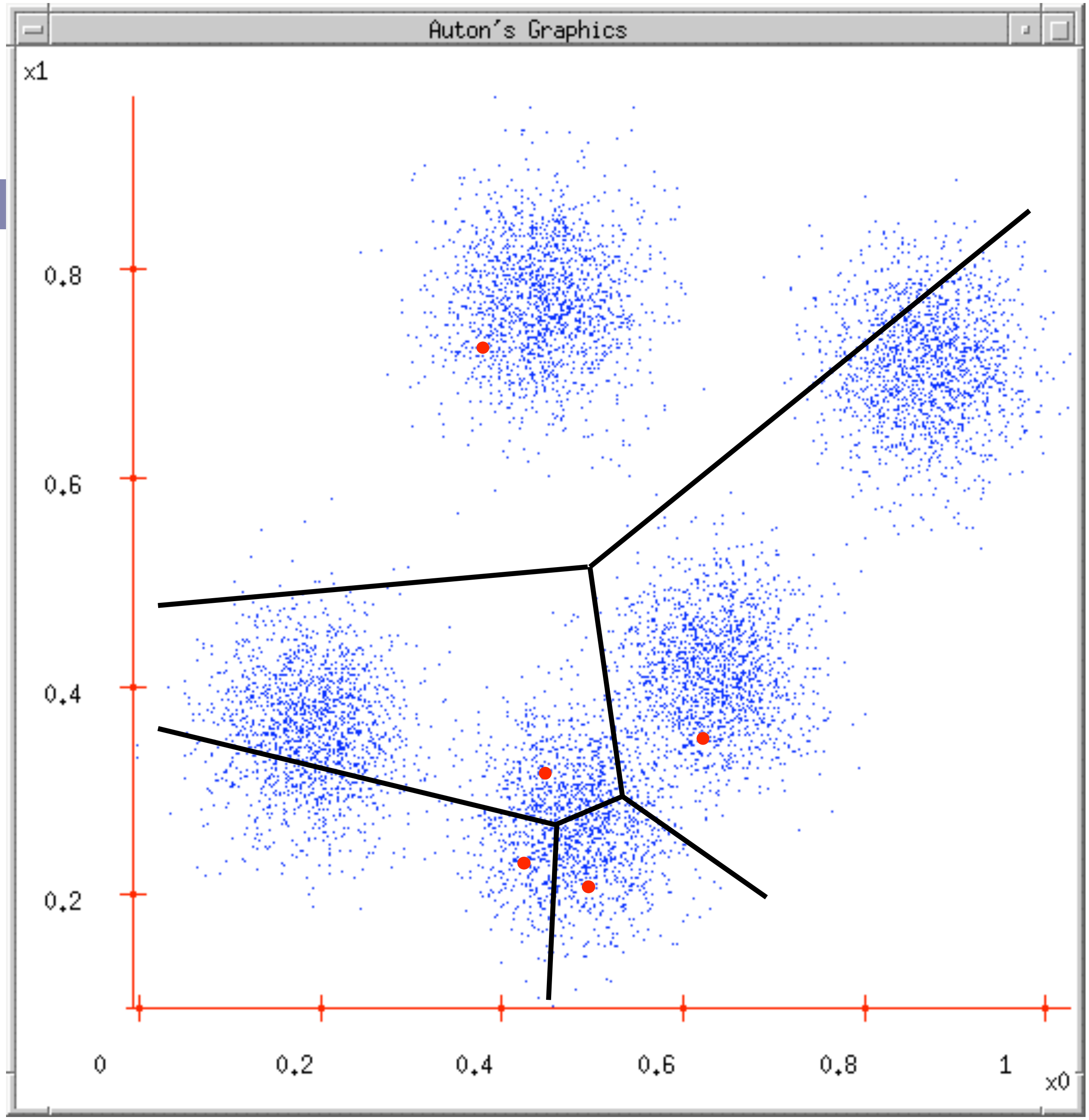
K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations



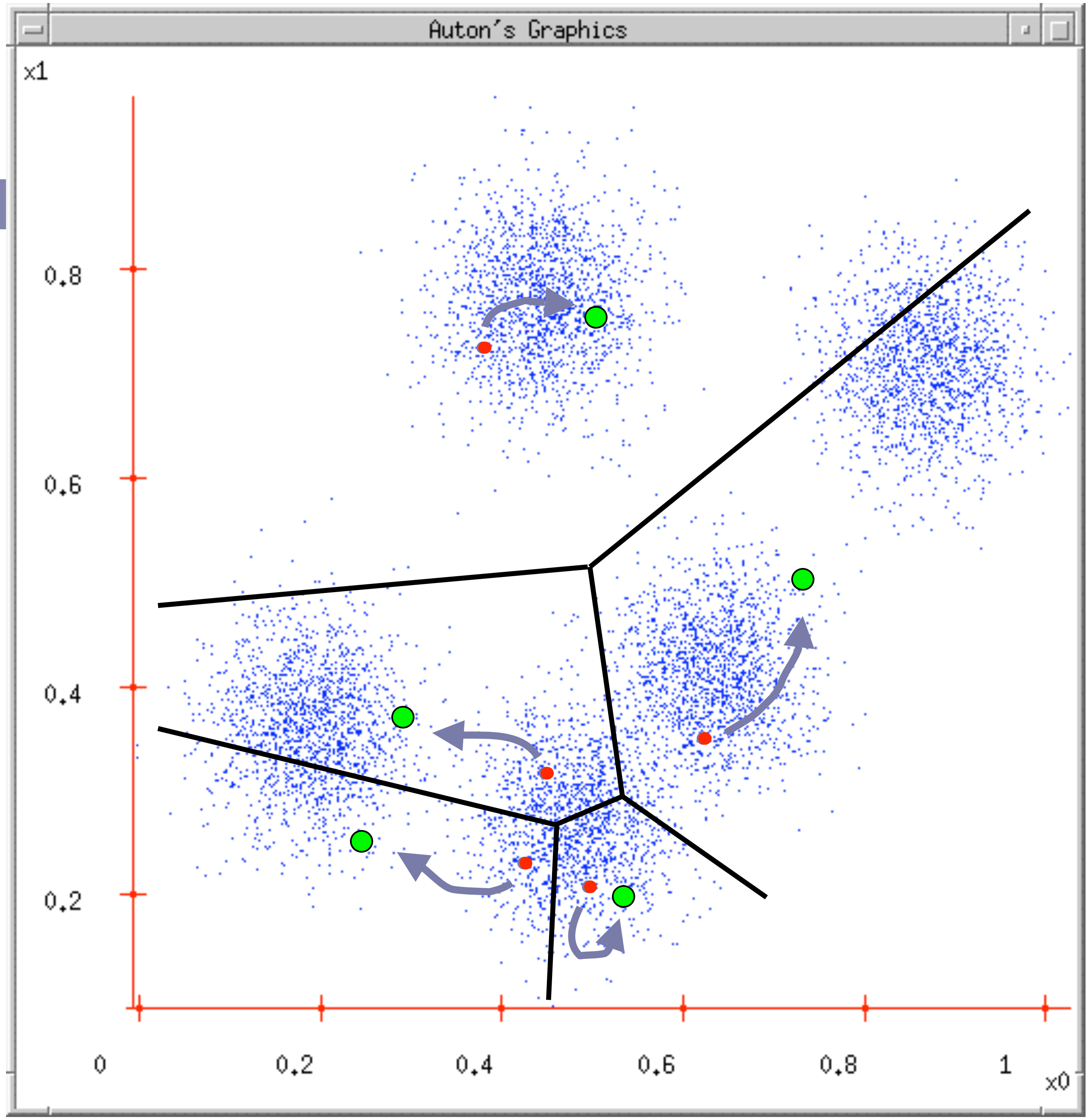
K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to. (Thus each Center "owns" a set of datapoints)



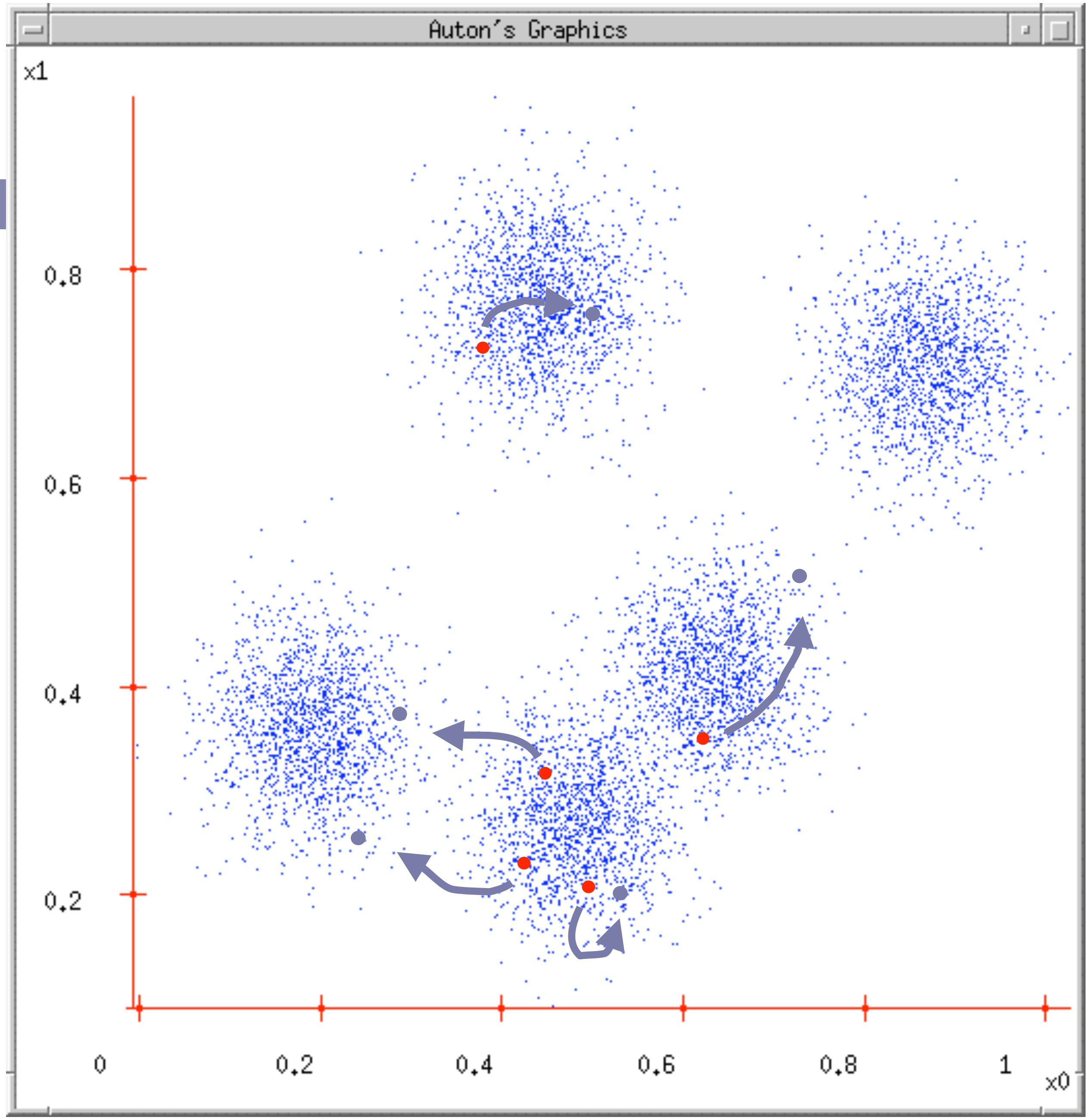
K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns



K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns…
5. …and jumps there
6. …Repeat until terminated!



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 $\sigma^2 \cdot I$. 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 $\boldsymbol{\mu}_i$'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!!

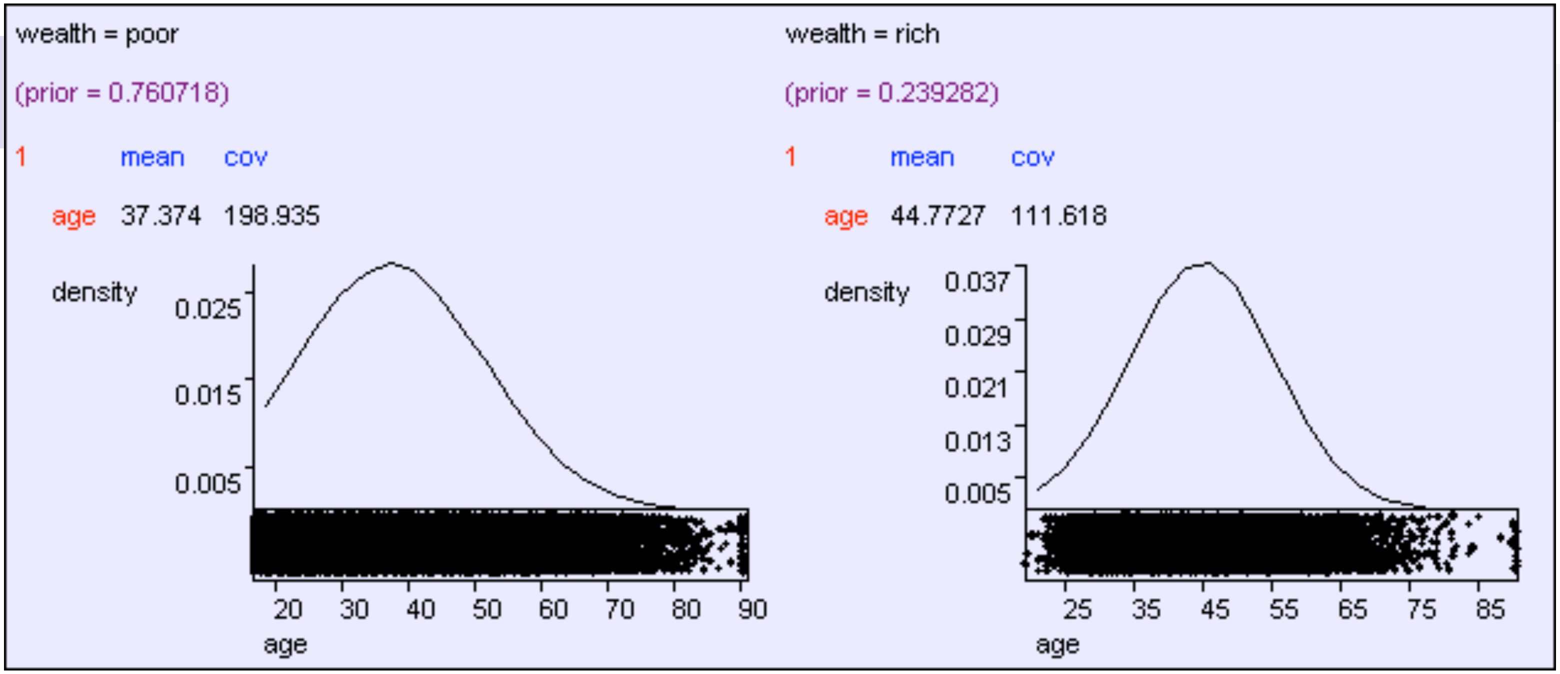
Gaussian Bayes Classifier Reminder

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

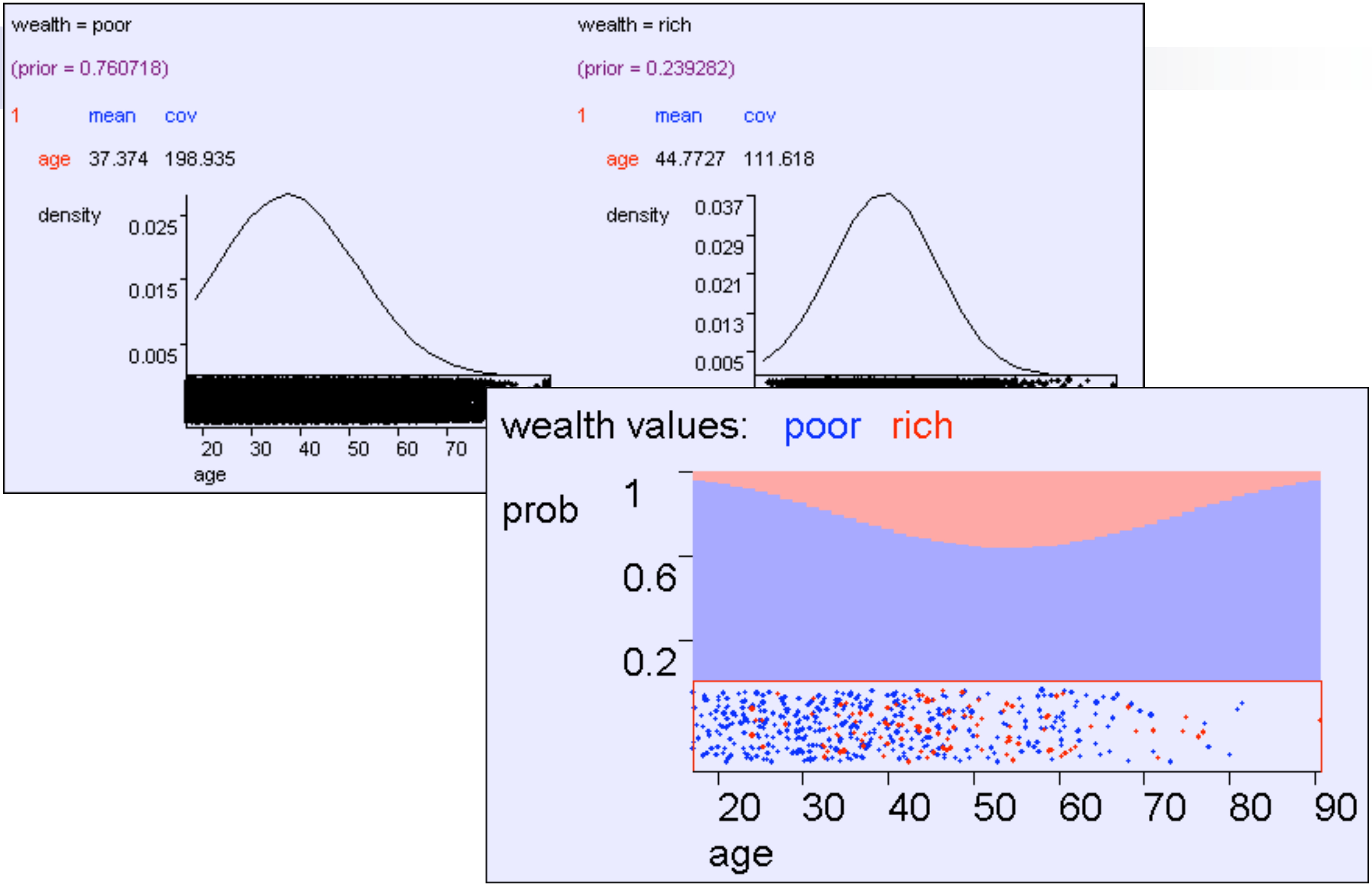
$$P(y=i \mid \mathbf{x}) = \frac{\frac{1}{(2\pi)^{m/2} \| \boldsymbol{\Sigma}_i \|^{1/2}} \exp\left[-\frac{1}{2}\left(\mathbf{x}_k - \boldsymbol{\mu}_i\right)^T \boldsymbol{\Sigma}_i \left(\mathbf{x}_k - \boldsymbol{\mu}_i\right)\right] p_i}{p(\mathbf{x})}$$

How do we deal with that?

Predicting wealth from age

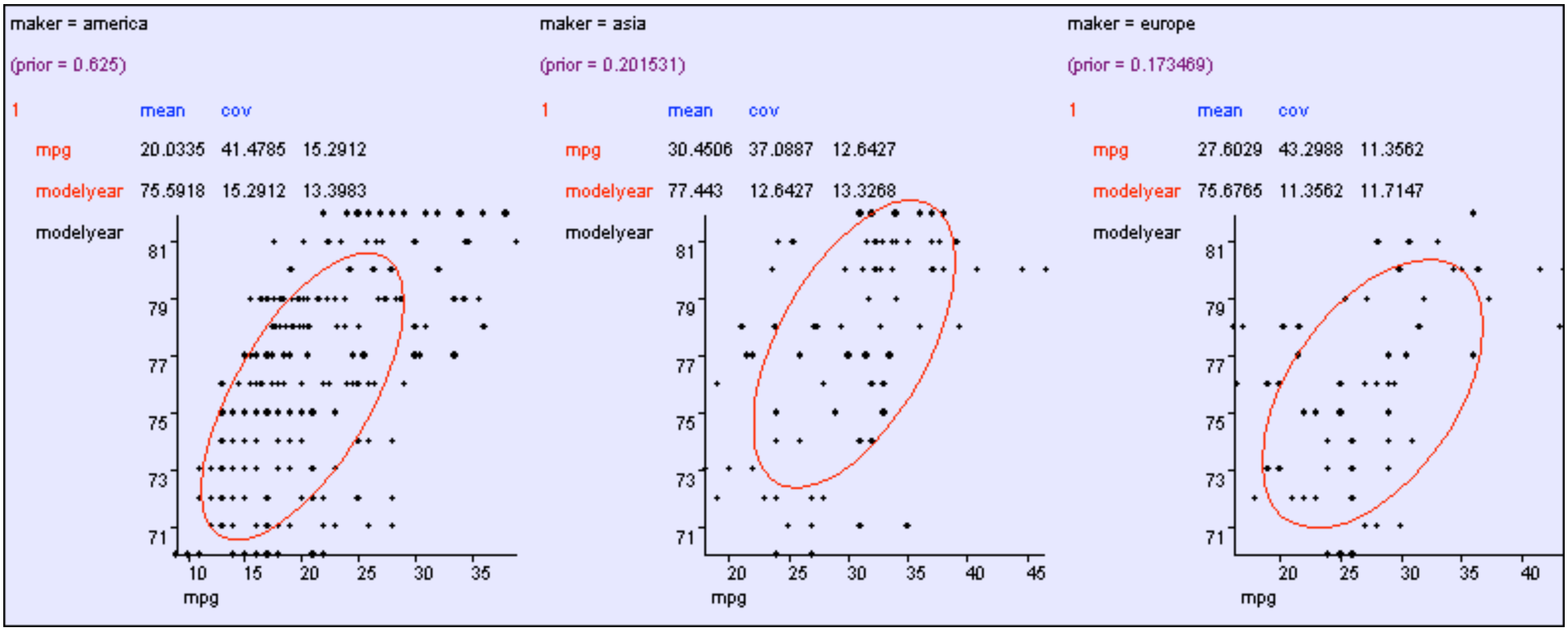


Predicting wealth from age



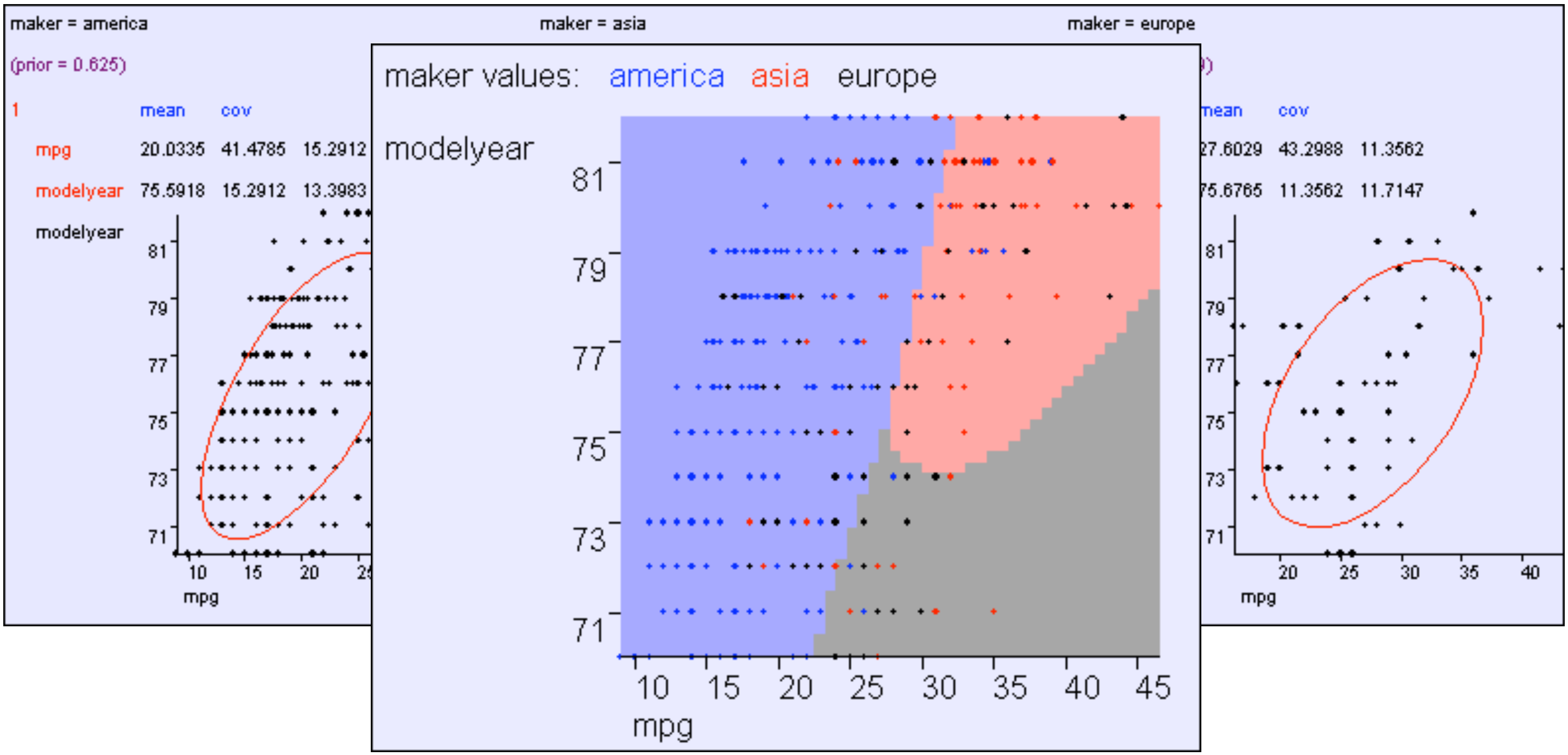
Learning modelyear , mpg ---> maker

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



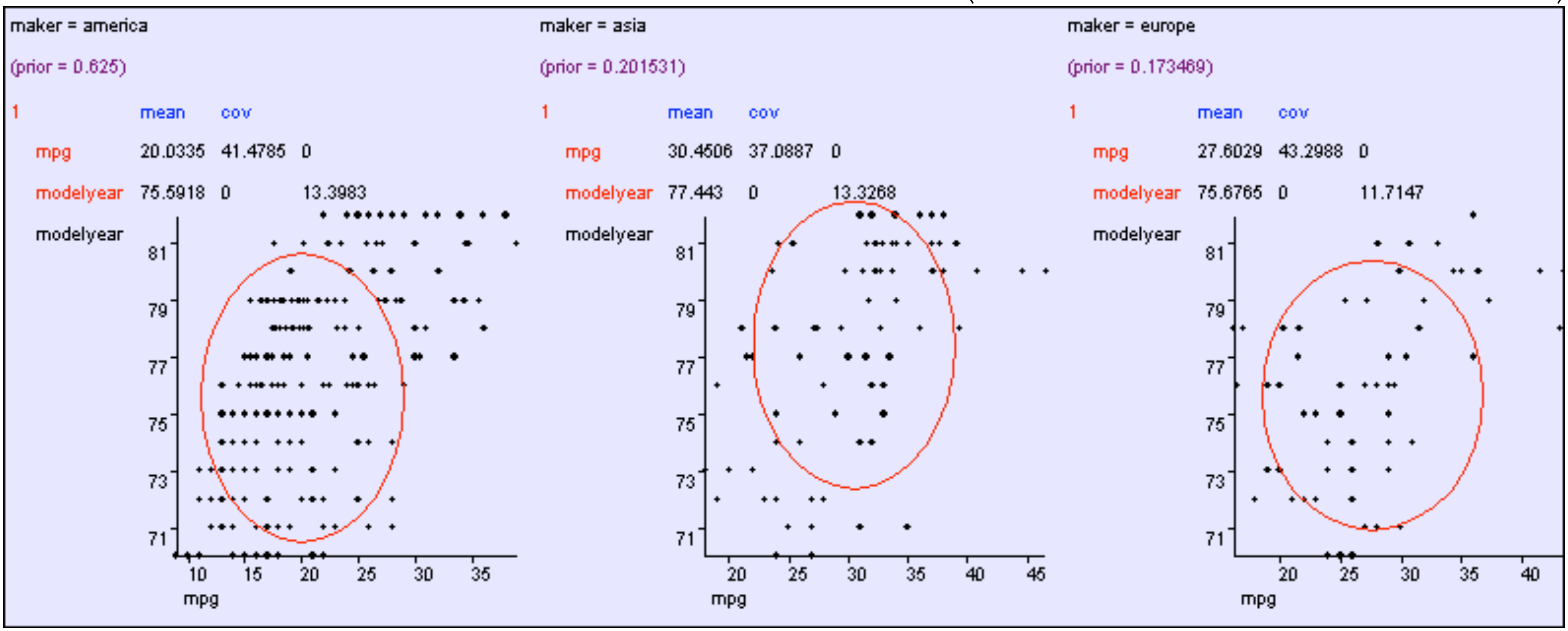
General: $O(m^2)$ parameters

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



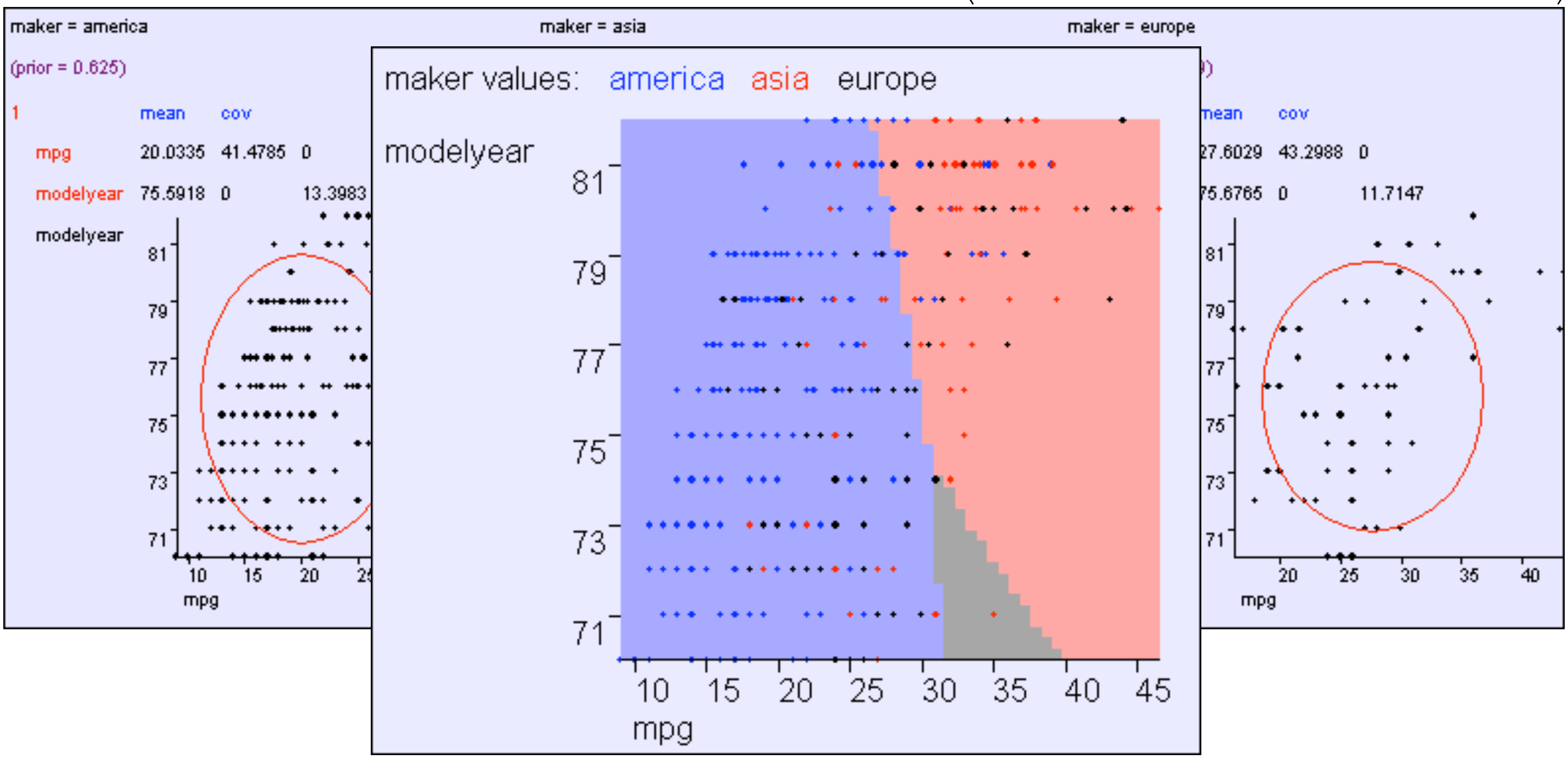
Aligned: *O(m)* parameters

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



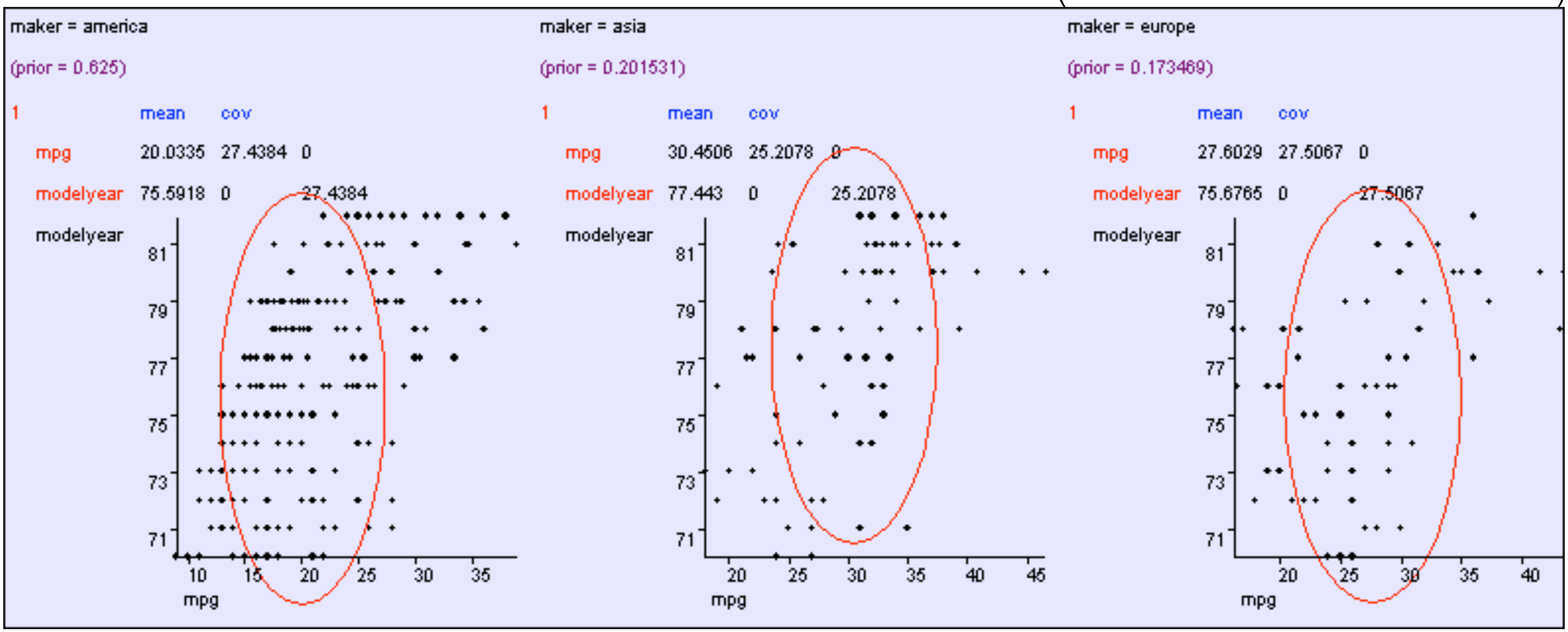
Aligned: *O(m)* parameters

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



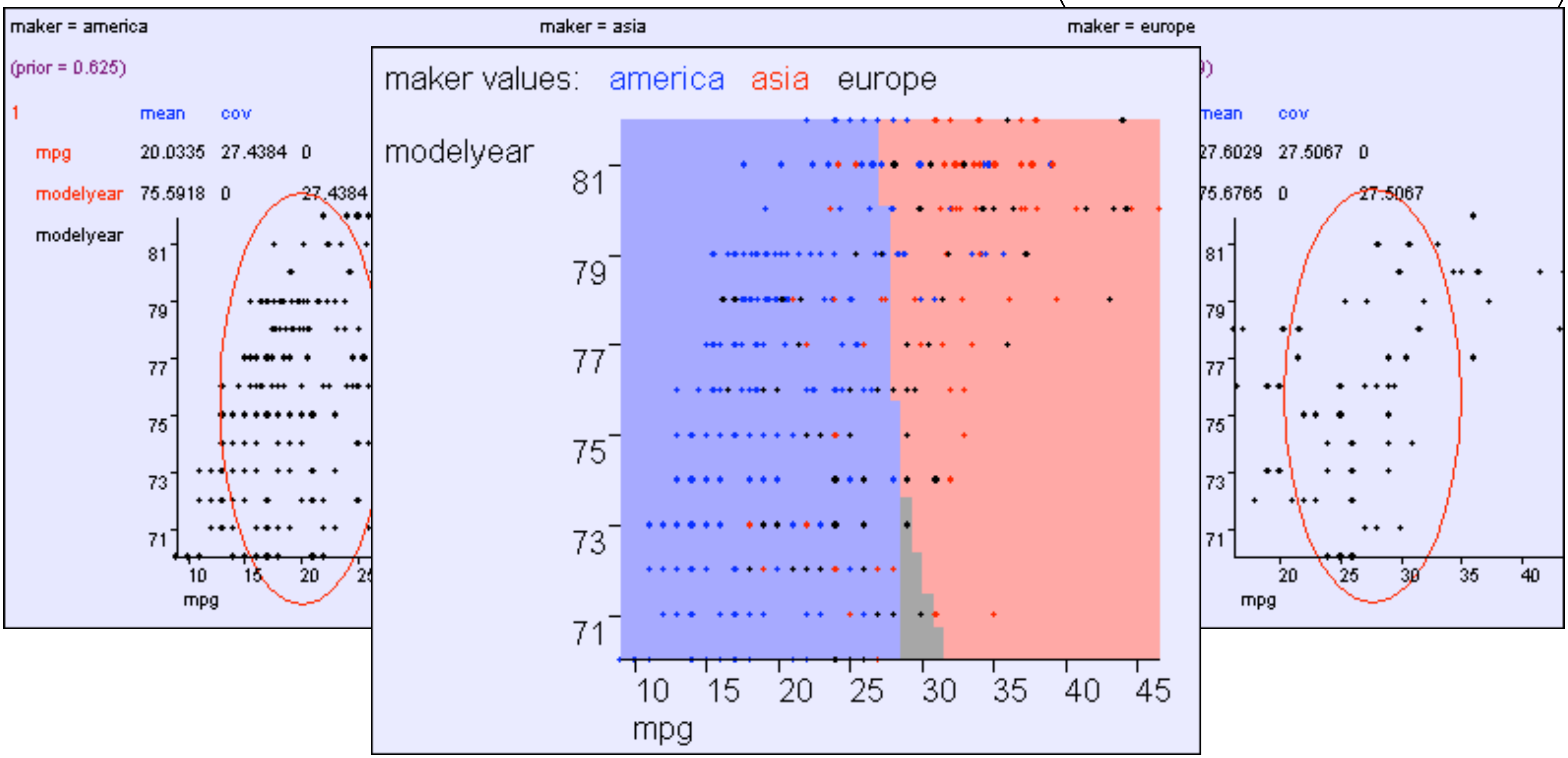
Spherical: *O(1)* cov parameters

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



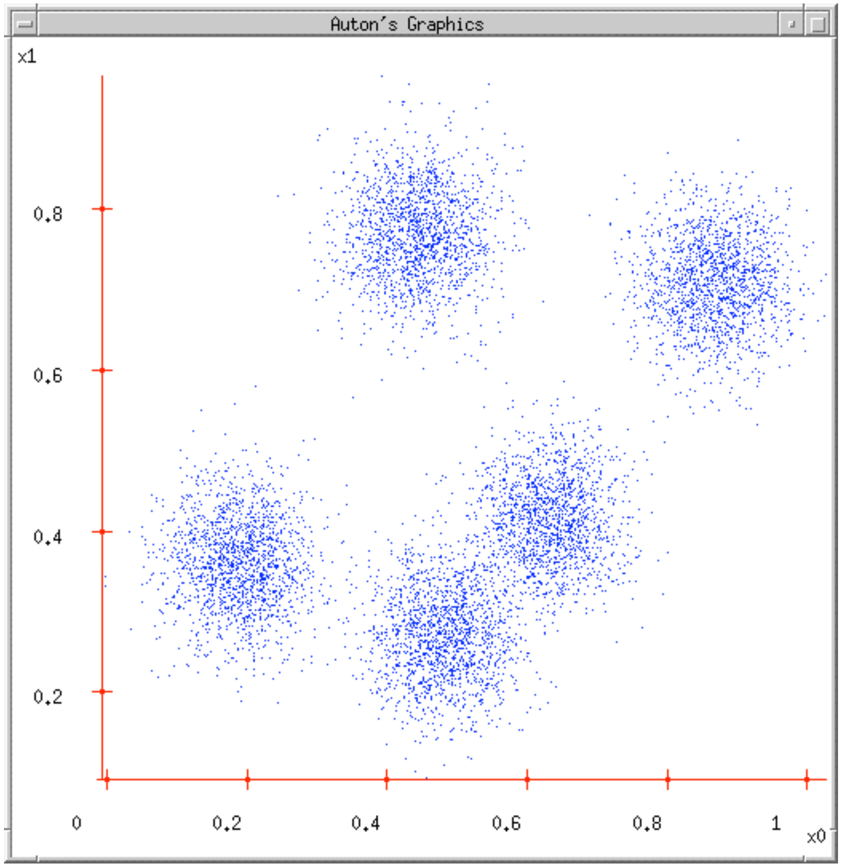
Spherical: *O(1)* cov parameters

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



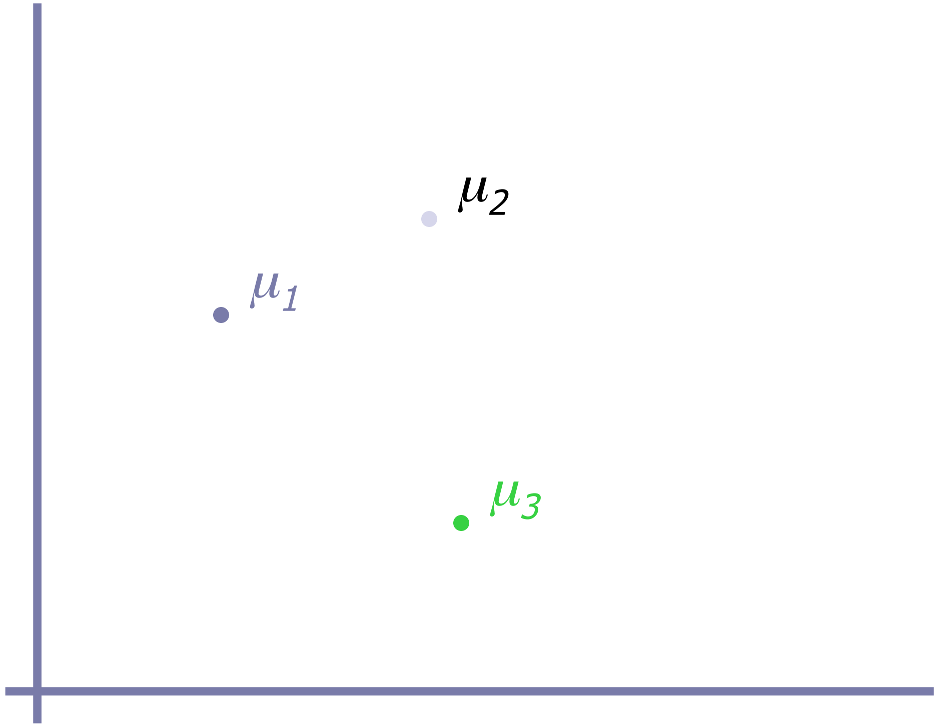
Next… back to Density Estimation

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



The GMM assumption

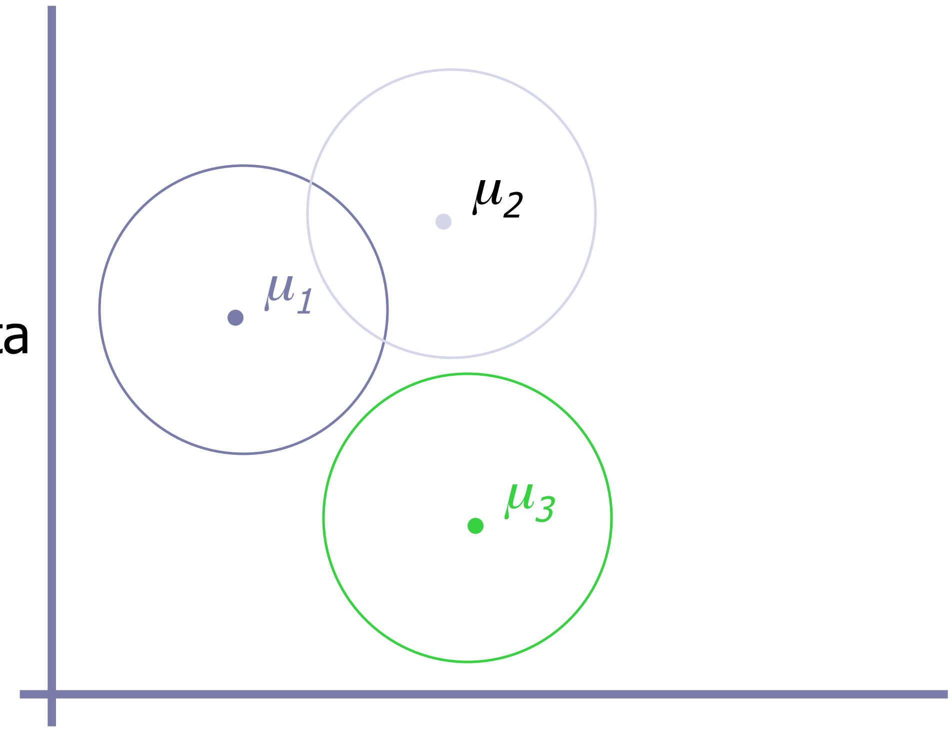
- There are k components. The i'th component is called ω_i
- Component ω_i has an associated mean vector μ_i



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 $\sigma^2\boldsymbol{I}$

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

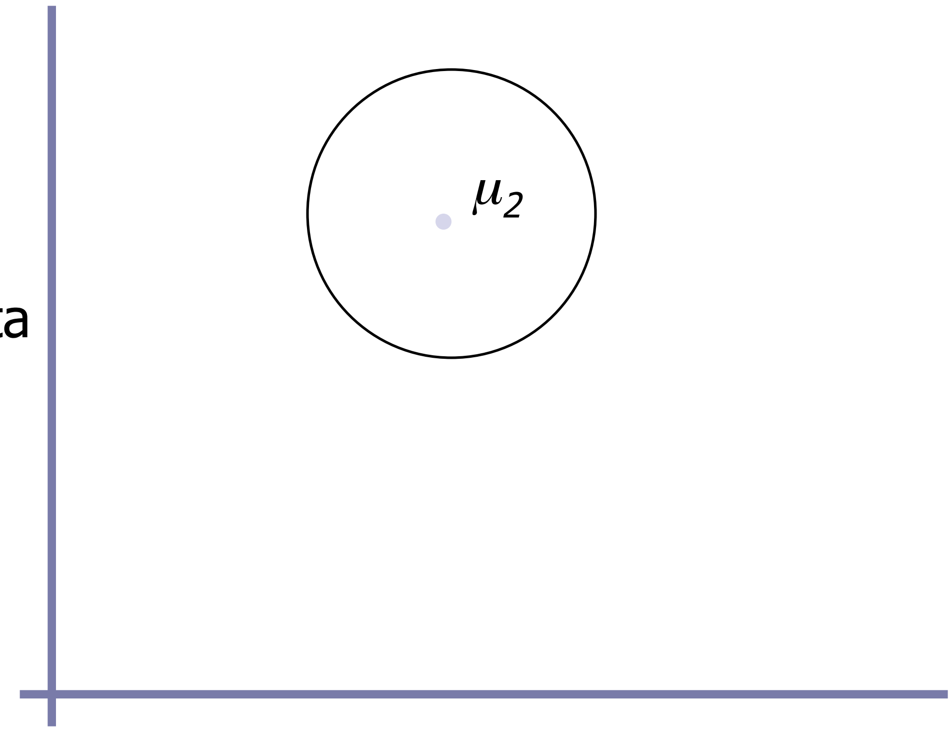


The GMM assumption

- 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 $P(y_i)$.

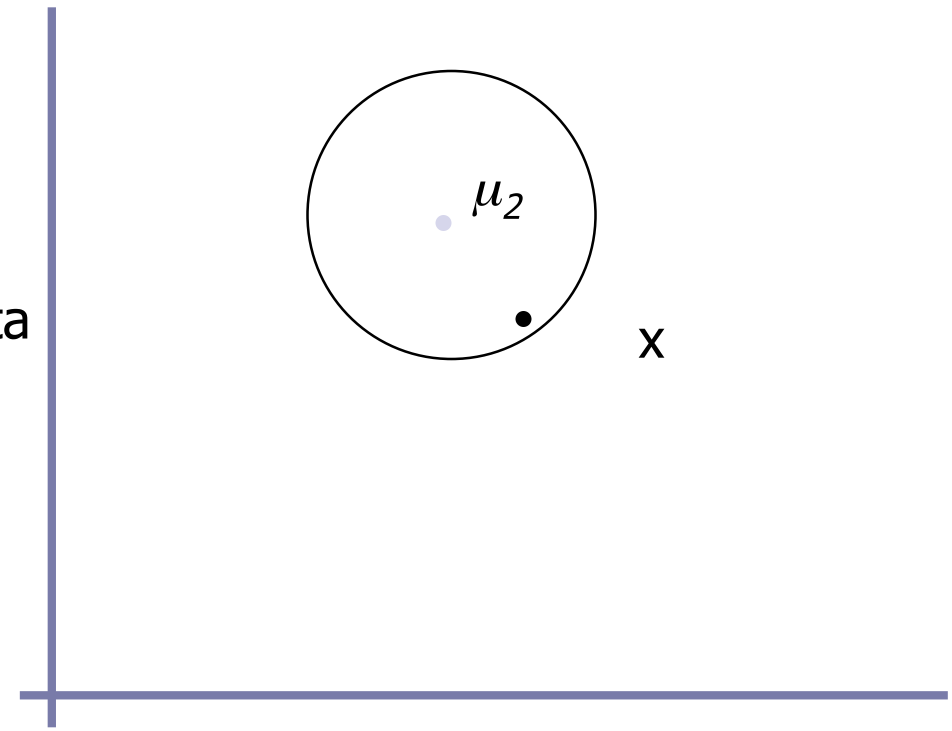


The GMM assumption

- 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 $\sigma^2 \boldsymbol{I}$

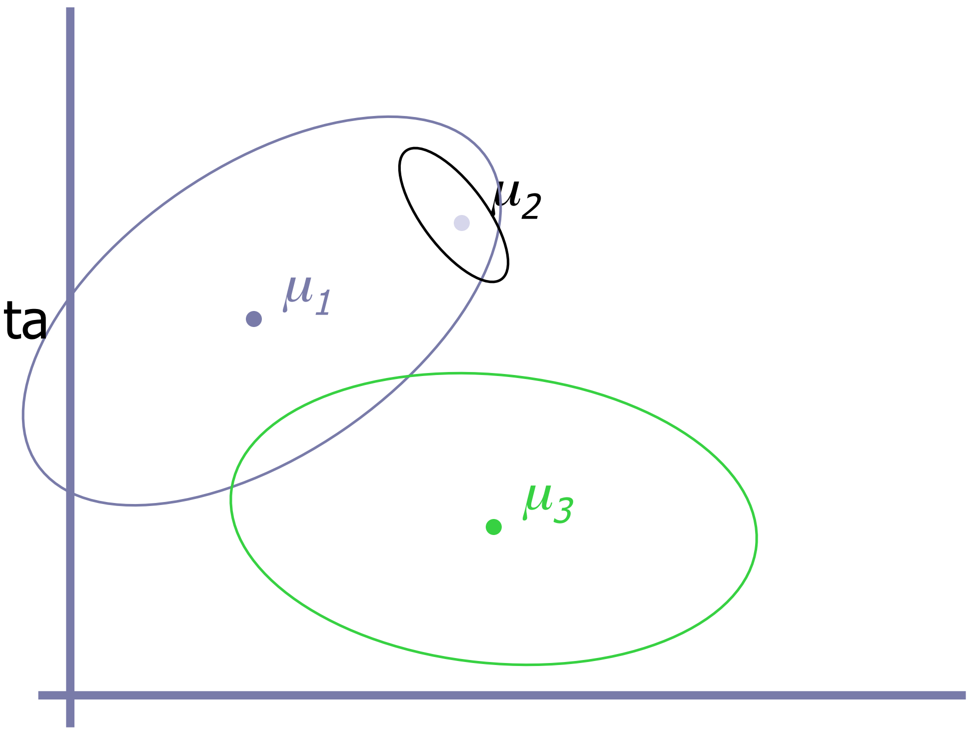
Assume that each datapoint is generated according to the following recipe:

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



The General GMM assumption

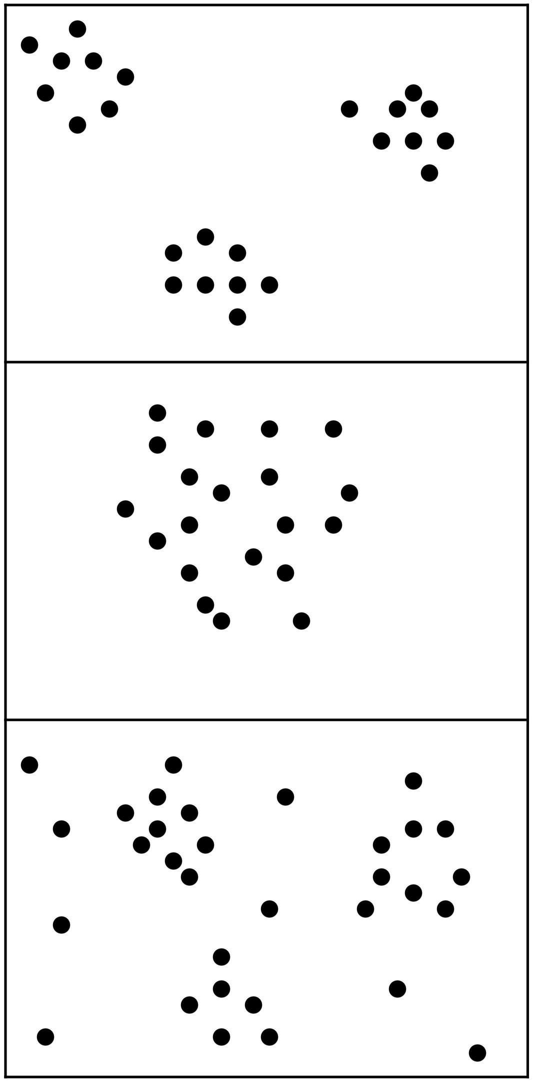
- There are k components. The i'th component is called ω_i
- Component ω_i has an associated mean vector μ_i
- Each component generates data from a Gaussian with mean μ_i and covariance matrix Σ_i



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

1. Pick a component at random. Choose component i with probability $P(y_i)$.
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 supervised learning case

We have $y_1, \boldsymbol{x}_1, y_2, \boldsymbol{x}_2, \ldots y_N, \boldsymbol{x}_N$

Learn $P(y_1)\ P(y_2)\ ..\ P(y_k)$

Learn $\sigma, \mu_1, \ldots, \mu_k$

By MLE: $P(y_1, \boldsymbol{x}_1, y_2, \boldsymbol{x}_2, \ldots y_N, \boldsymbol{x}_N \,|\, \boldsymbol{\mu}_i, \ldots \boldsymbol{\mu}_k, \sigma)$

Computing likelihoods in unsupervised case

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

We know $P(y_1)$ $P(y_2)$ .. $P(y_k)$

We know σ

$P(\boldsymbol{x}|y_i, \boldsymbol{\mu}_i, \ldots \boldsymbol{\mu}_k)$ = Prob that an observation from class y_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 $\boldsymbol{x}_1\ \boldsymbol{x}_2 \ldots \boldsymbol{x}_n$

We have $P(y_1)$ .. $P(y_k)$. We have σ.

We can define, for any $\boldsymbol{x}$, $P(\boldsymbol{x}|y_i\ , \boldsymbol{\mu}_1, \boldsymbol{\mu}_2 \ldots \boldsymbol{\mu}_k)$

Can we define $P(\boldsymbol{x} \mid \boldsymbol{\mu}_1, \boldsymbol{\mu}_2 \ldots \boldsymbol{\mu}_k)$?

Can we define $P(\boldsymbol{x}_1, \boldsymbol{x}_1, \ldots \boldsymbol{x}_n \mid \boldsymbol{\mu}_1, \boldsymbol{\mu}_2 \ldots \boldsymbol{\mu}_k)$?

[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.

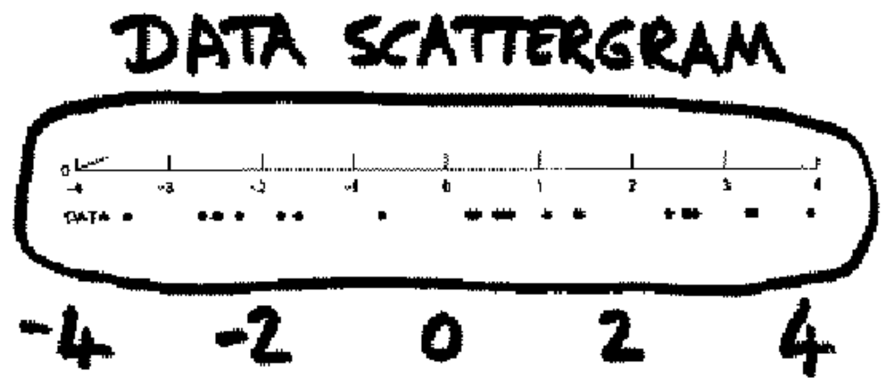
(From Duda and Hart)

There are two classes; w_1 and w_2

$P(y_1) = 1/3 \quad P(y_2) = 2/3 \quad \sigma = 1$.

There are 25 unlabeled datapoints

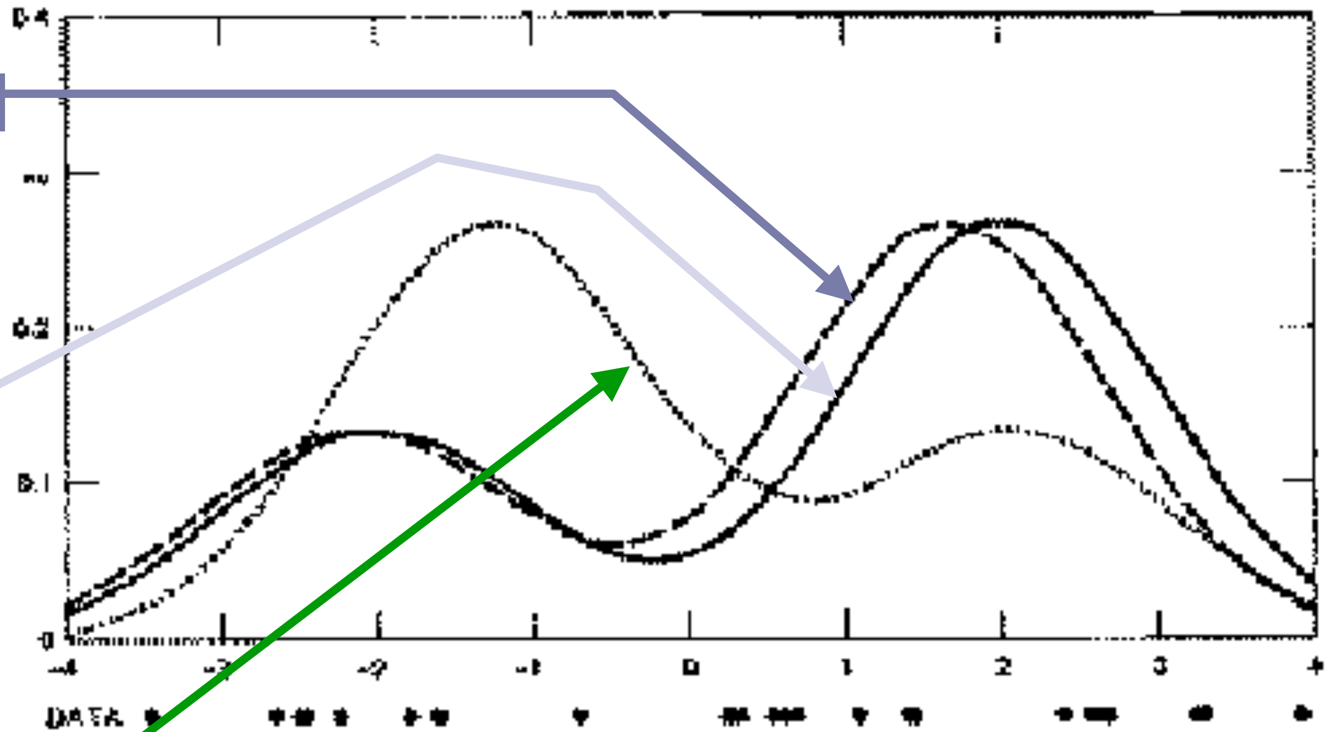
$x_1 = 0.608$
$x_2 = -1.590$
$x_3 = 0.235$
$x_4 = 3.949$
:
$x_{25} = -0.712$



Duda & Hart's Example

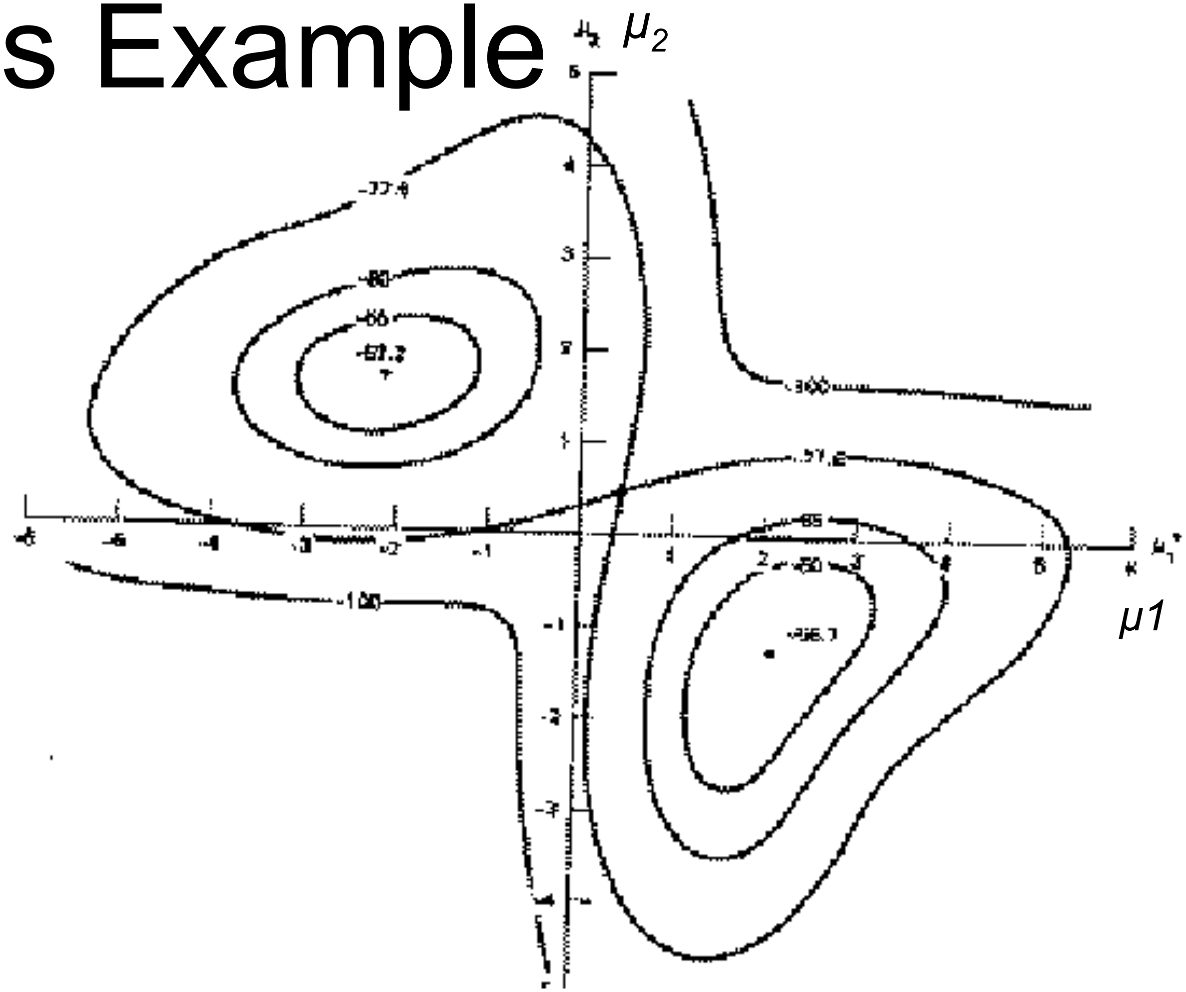
We can graph the prob. dist. function of data given our μ_1 and μ_2 estimates.

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).

Duda & Hart's Example



Graph of

log P(x_1, x_2 .. x_{25} | μ_1, μ_2)

against μ_1 (→) and μ_2 (↑)

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 y_1 with y_2.

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?

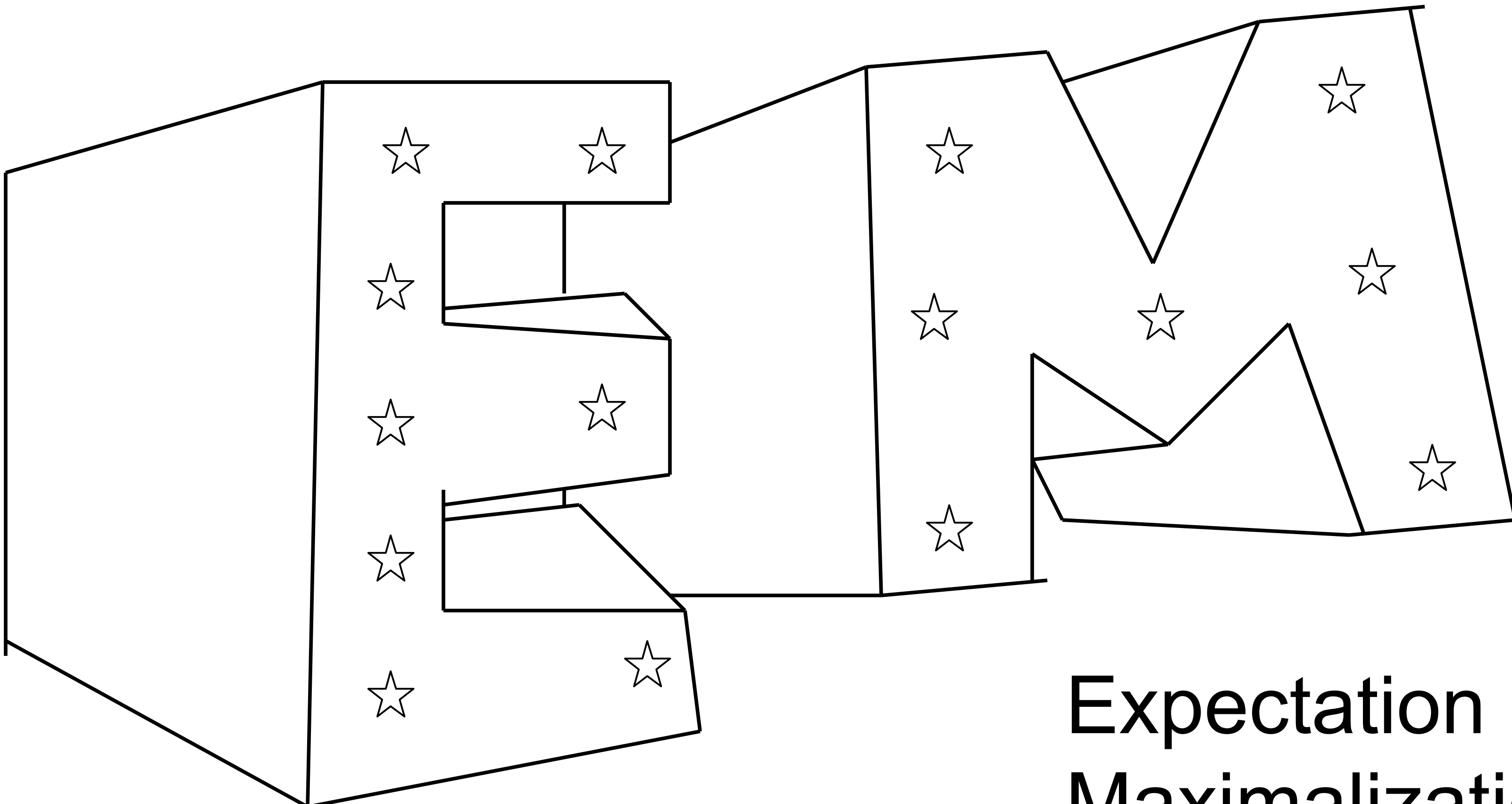
- The normal max likelihood trick:

  Set $\frac{\partial}{\partial \mu_i} \log \text{Prob} (\ldots.) = 0$

  and solve for μ_i's.

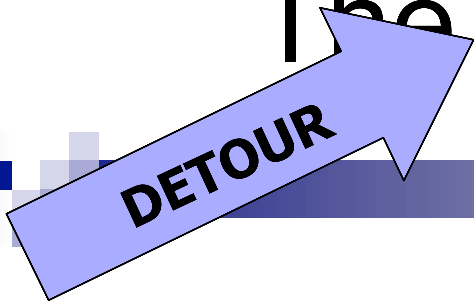
  # Here you get non-linear non-analytically- solvable equations
- Use gradient descent

  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 $0 \leq \mu \leq 1/6$)

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

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

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

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

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

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

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

$P(A) = ½$

$P(B) = \mu$

$P(C) = 2\mu$

$P(D) = ½-3\mu$

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

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

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

E-step

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

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

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

$h = 20$

$c = 10$

$d = 10$

$\mu(0) = 0$

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 <u>linear</u>: error decreases by a constant factor each time step.

Back to Unsupervised Learning of GMMs

Remember:

- We have unlabeled data $\boldsymbol{x}_1\ \boldsymbol{x}_2 \ldots \boldsymbol{x}_R$
- We know there are k classes
- We know $P(y_1)\ P(y_2)\ P(y_3) \ldots P(y_k)$
- We <u>don't</u> know $\boldsymbol{\mu}_1\ \boldsymbol{\mu}_2 \ldots \boldsymbol{\mu}_k$

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

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

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

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

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

E.M. for GMMs

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

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

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

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 y_j is $P(y_j|x_i,\mu_1\ldots\mu_k)$ Then… we would easily compute μ_j.

If we knew each μ_j then we could easily compute $P(y_j|x_i,\mu_1\ldots\mu_k)$ for each y_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 $\lambda_t = \{ \mu_1(t), \mu_2(t) \ldots \mu_c(t) \}$

E-step

Compute "expected" classes of all datapoints for each class

Just evaluate a Gaussian at x_k

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

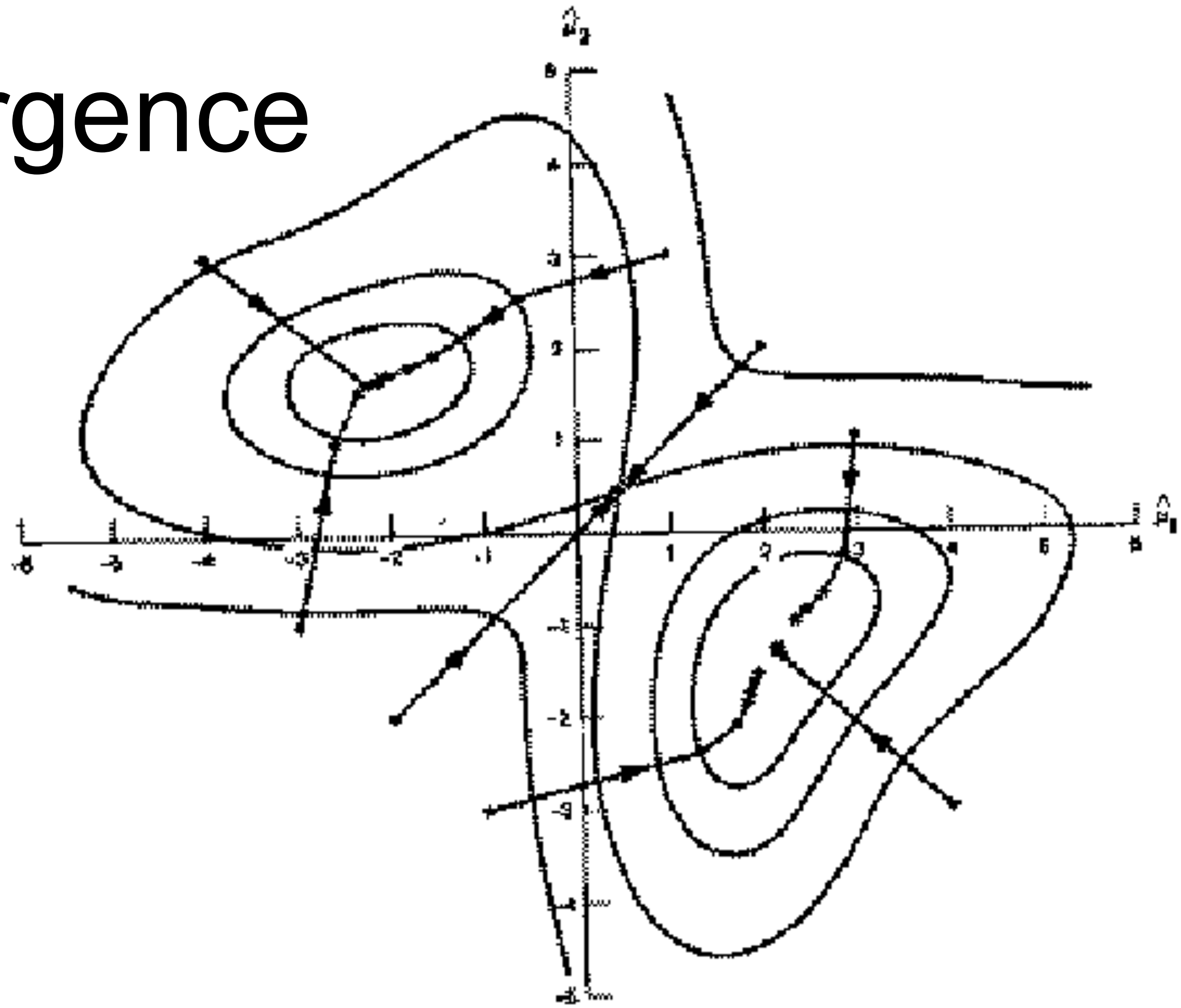
M-step.

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

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

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 *$P(y_i)$* on t'th iteration

E-step

Compute "expected" classes of all datapoints for each class

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

Just evaluate a Gaussian at x_k

M-step.

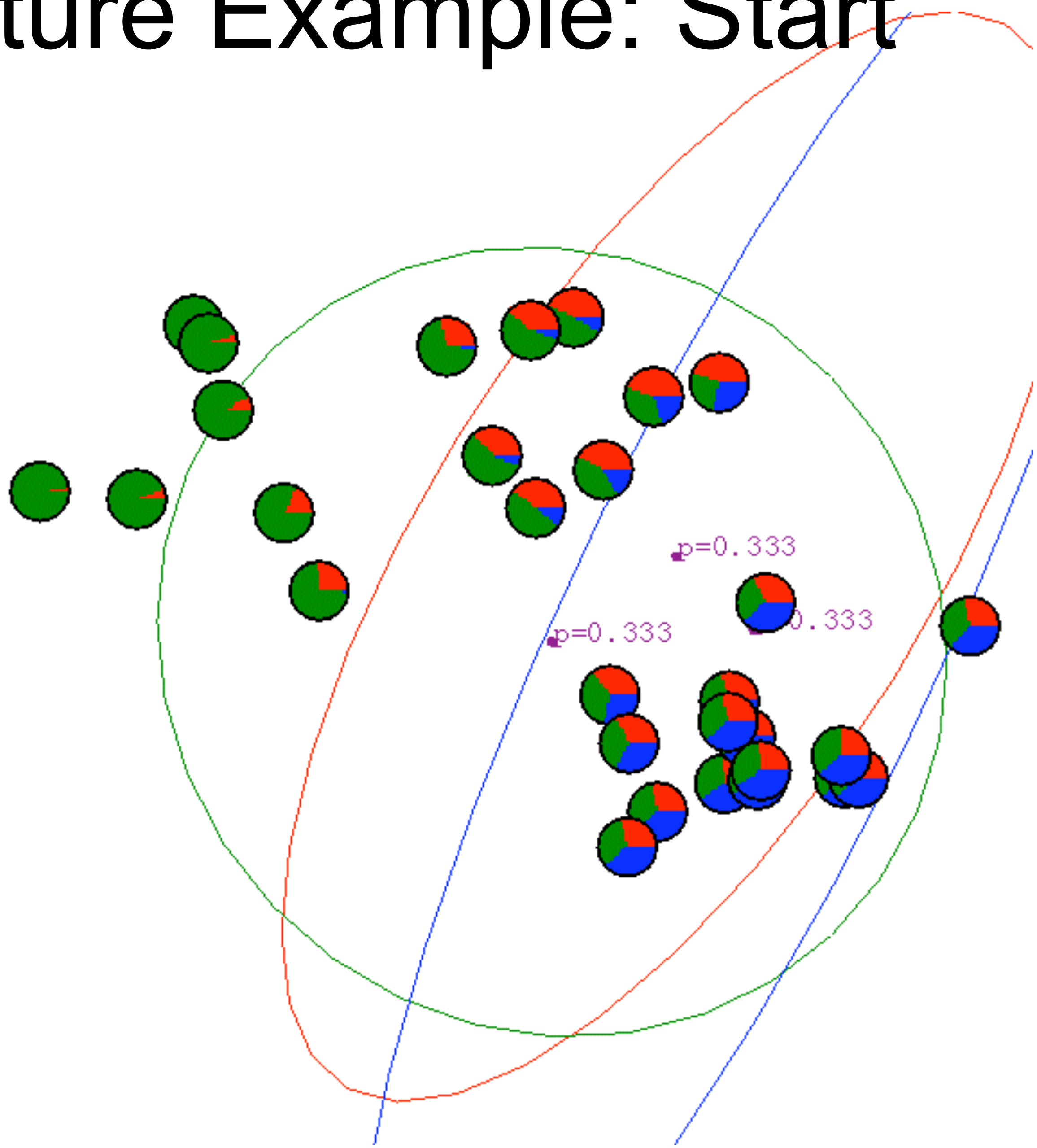
Compute Max. like **μ** given our data's class membership distributions

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

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

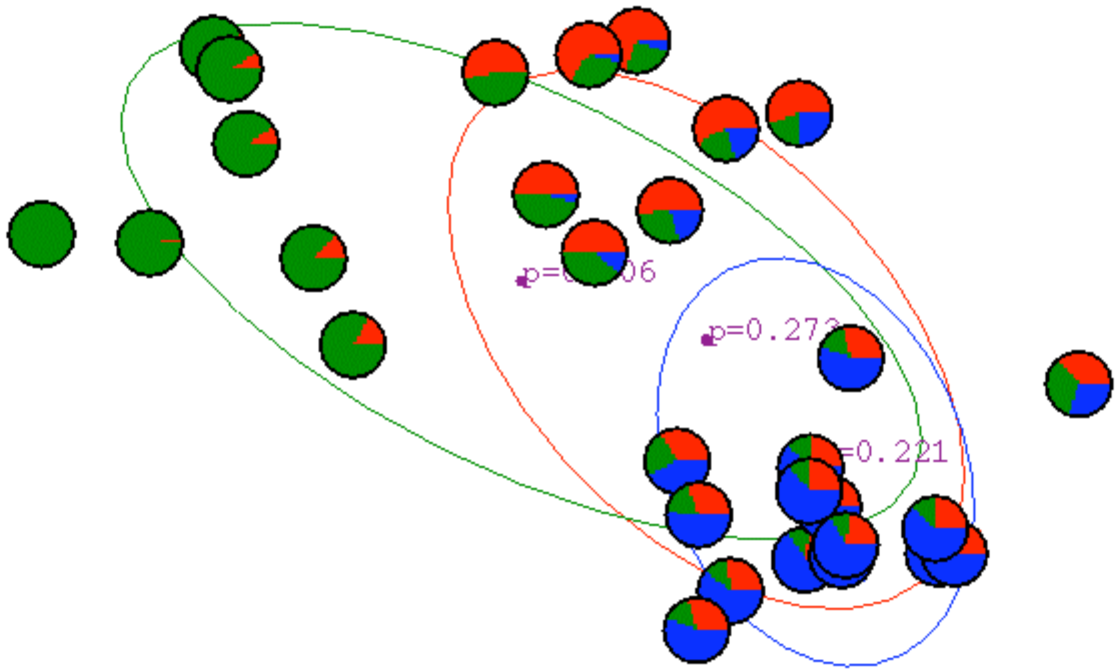
R = #records

Gaussian Mixture Example: Start

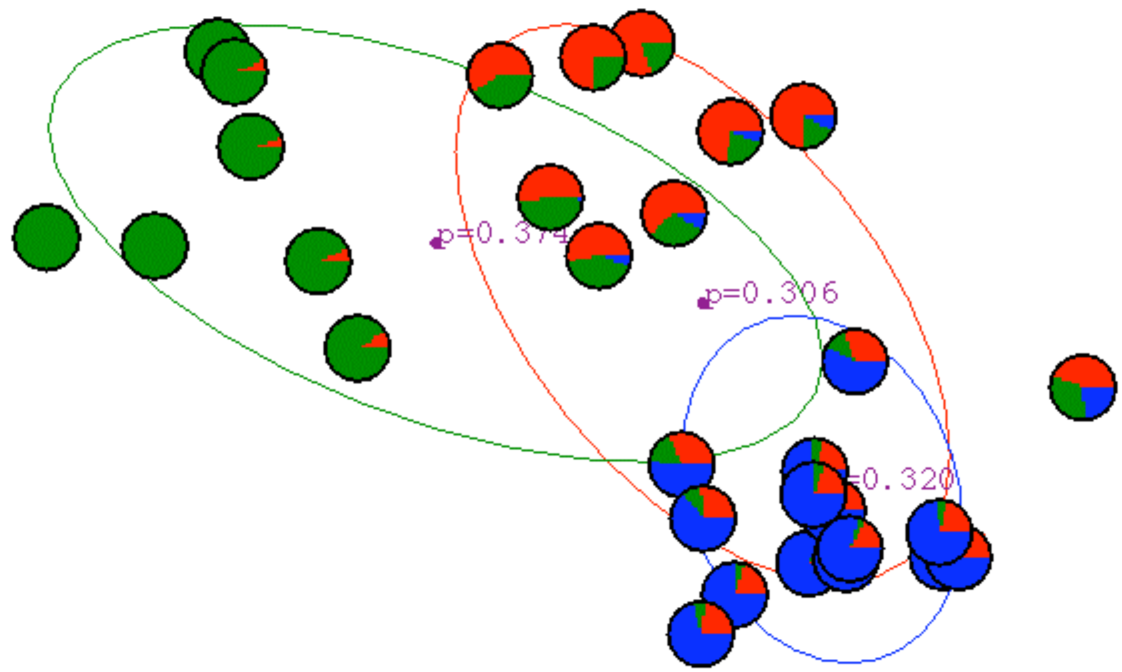


Advance apologies: in Black and White this example will be incomprehensible

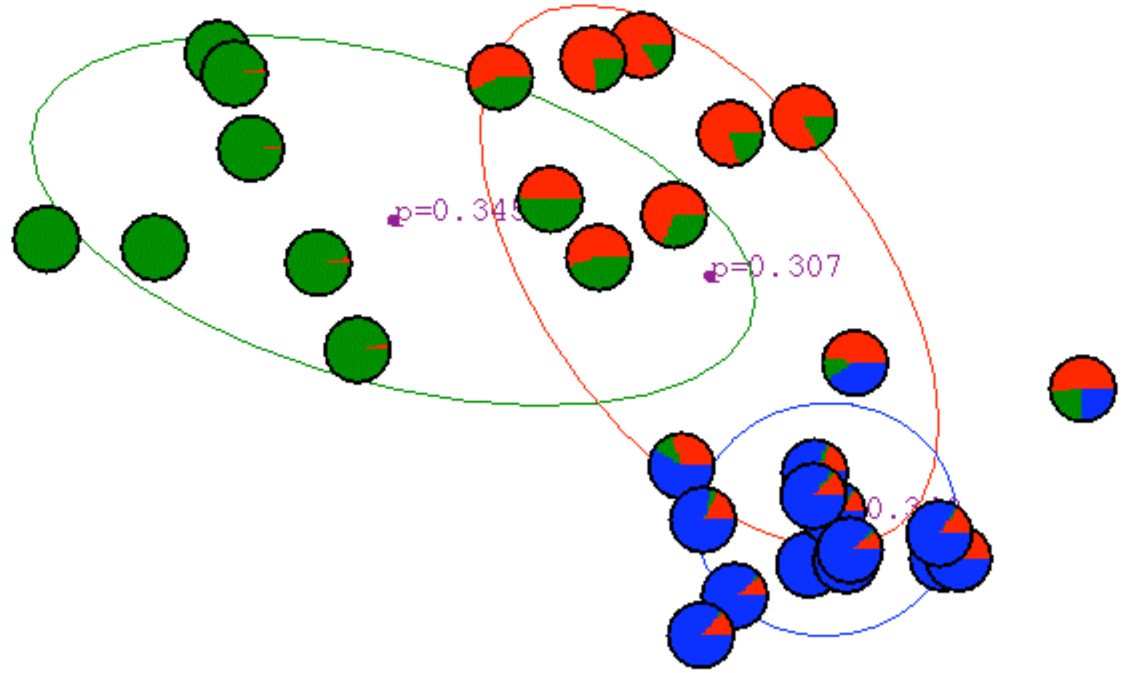
After first iteration



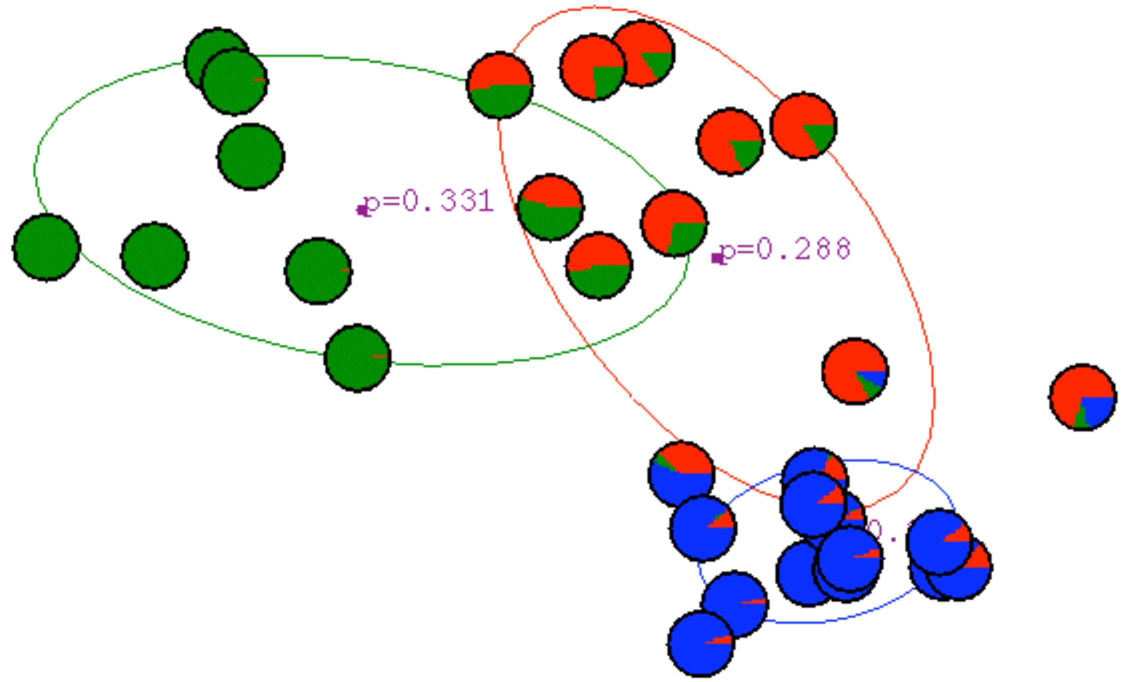
After 2nd iteration



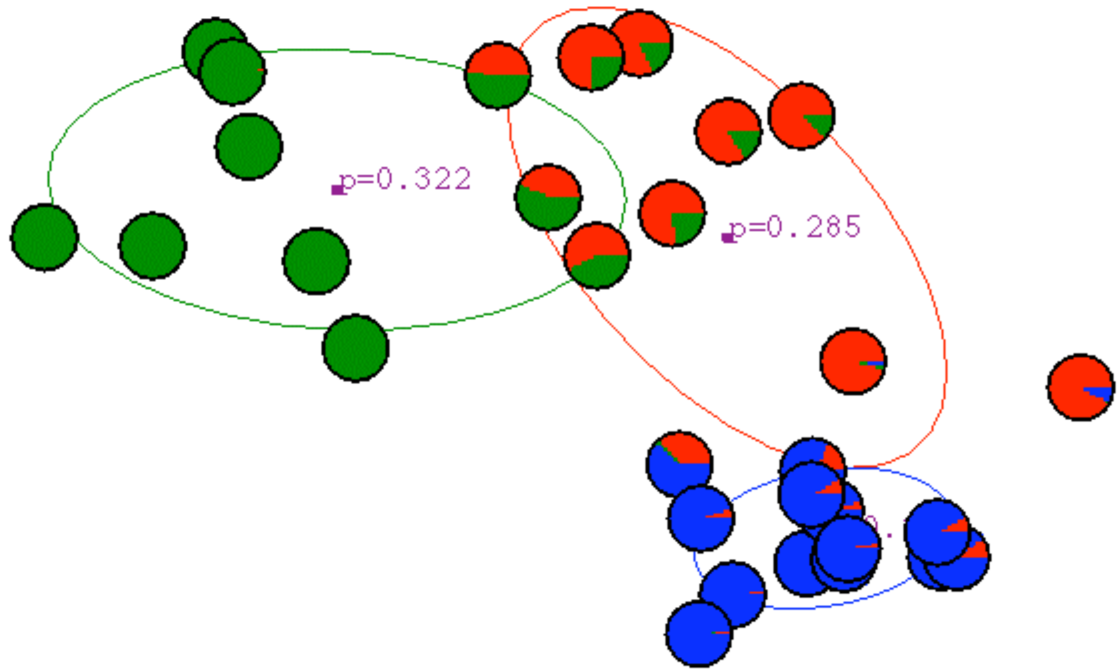
After 3rd iteration



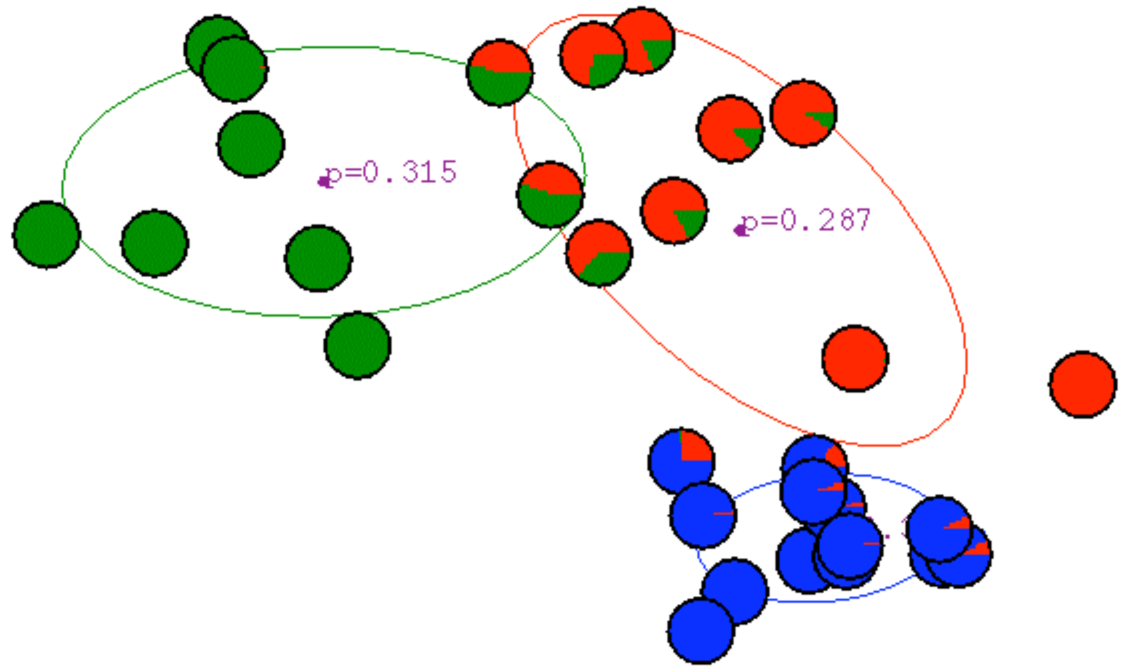
After 4th iteration



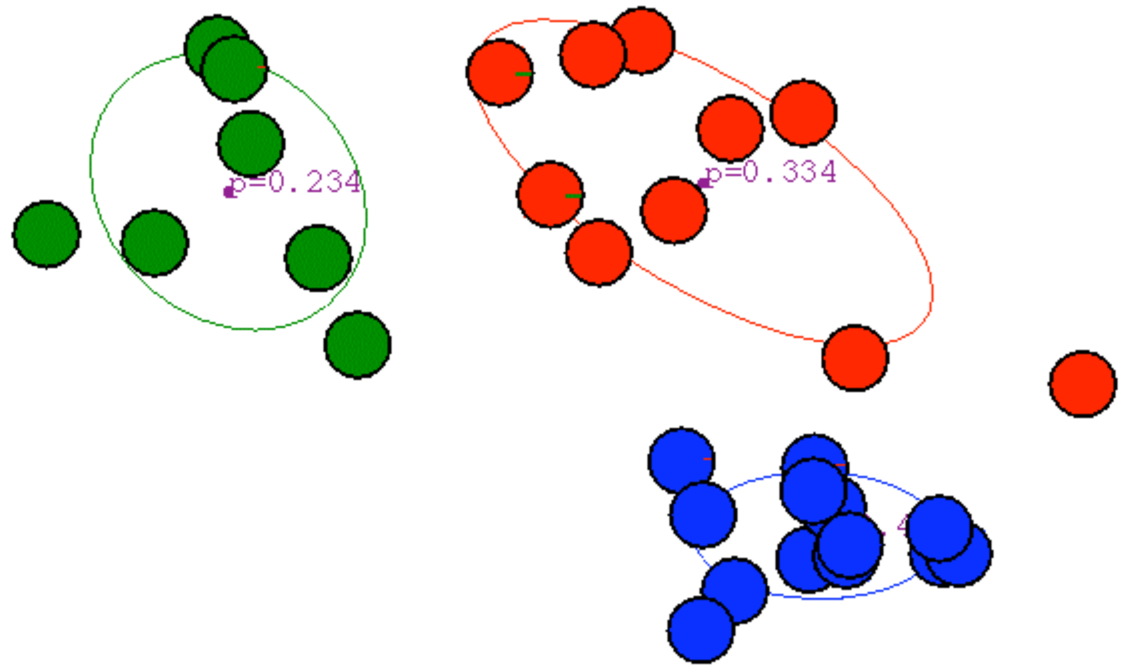
After 5th iteration



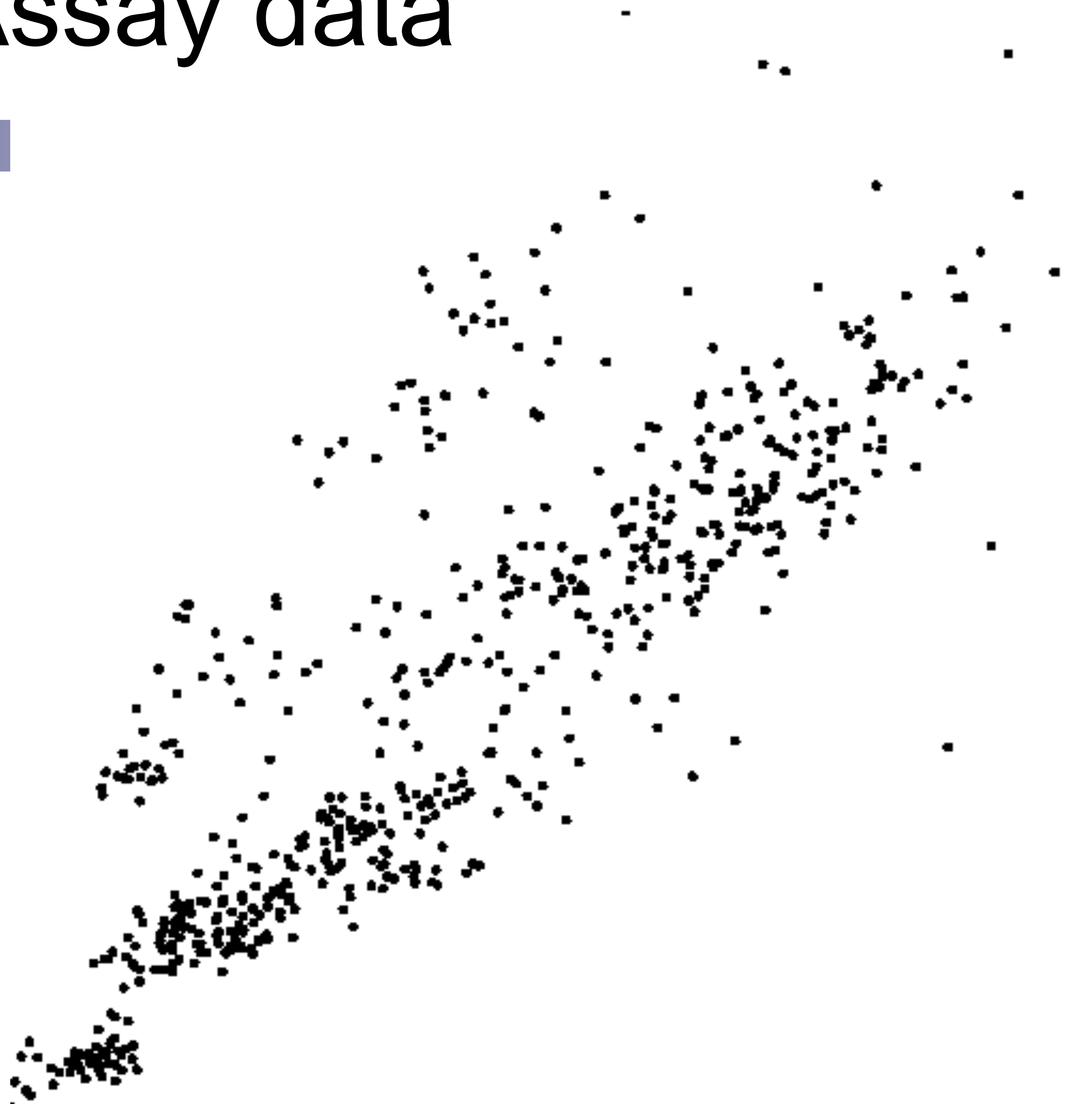
After 6th iteration



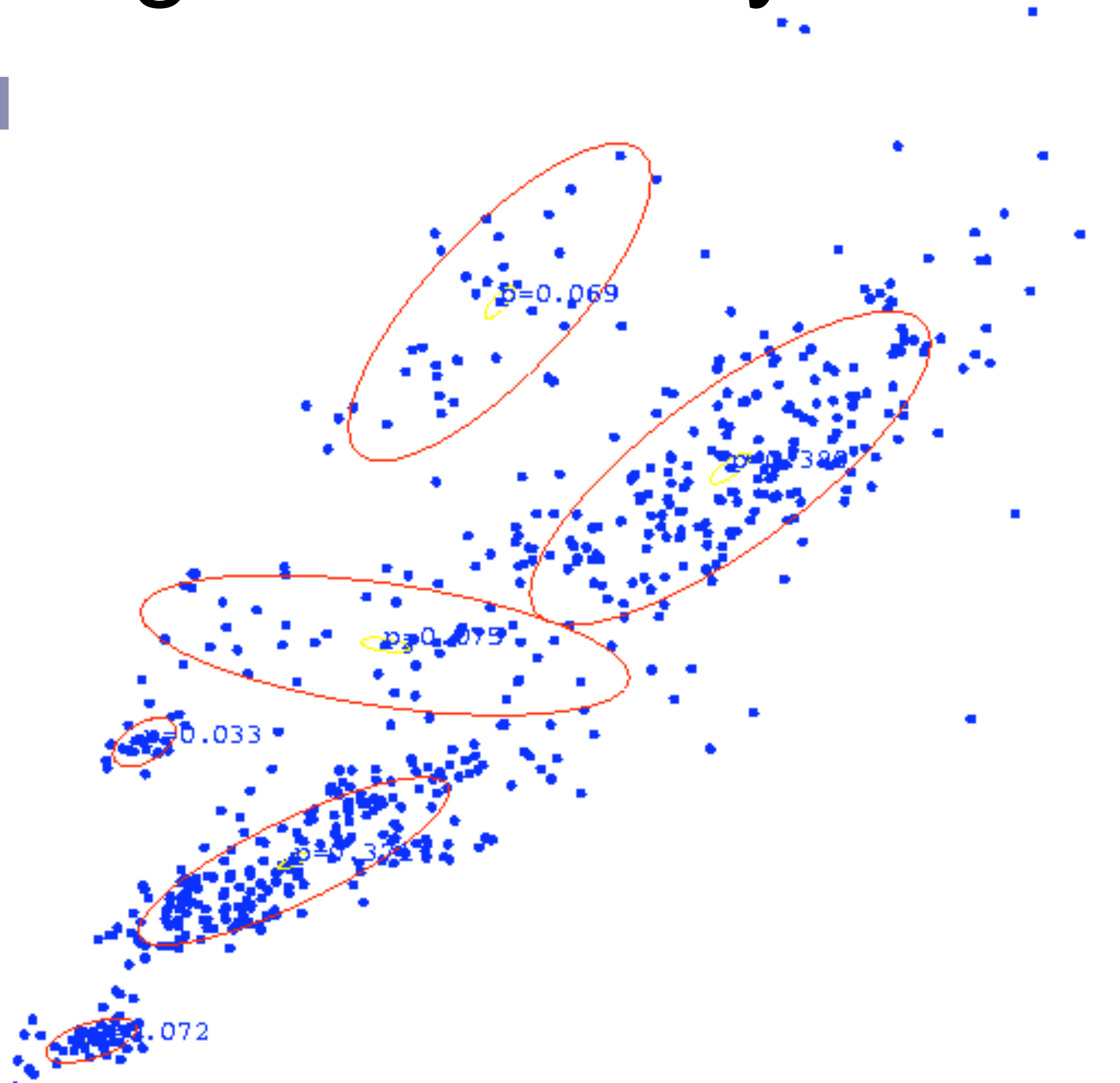
After 20th iteration



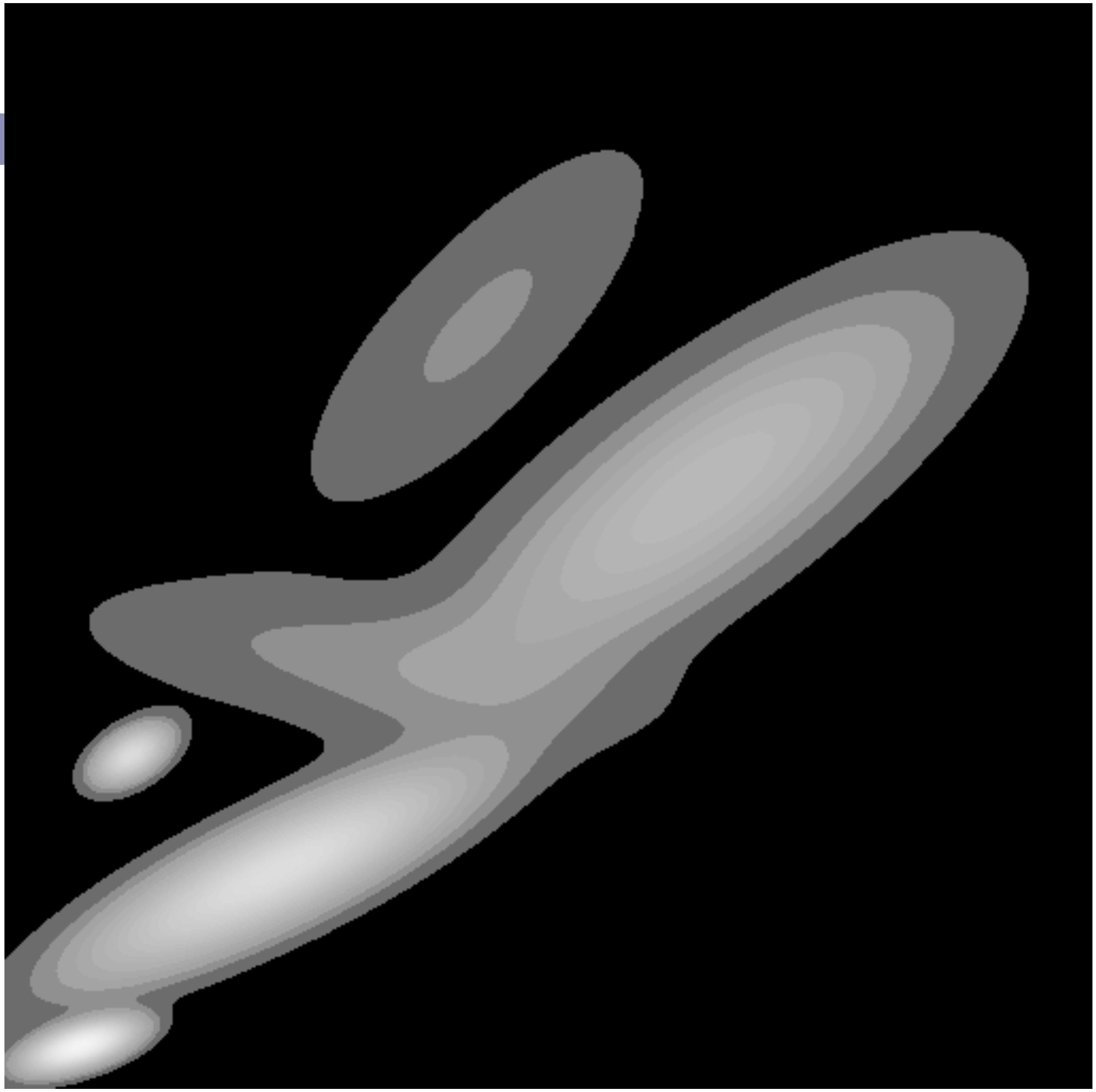
Some Bio Assay data



GMM clustering of the assay data

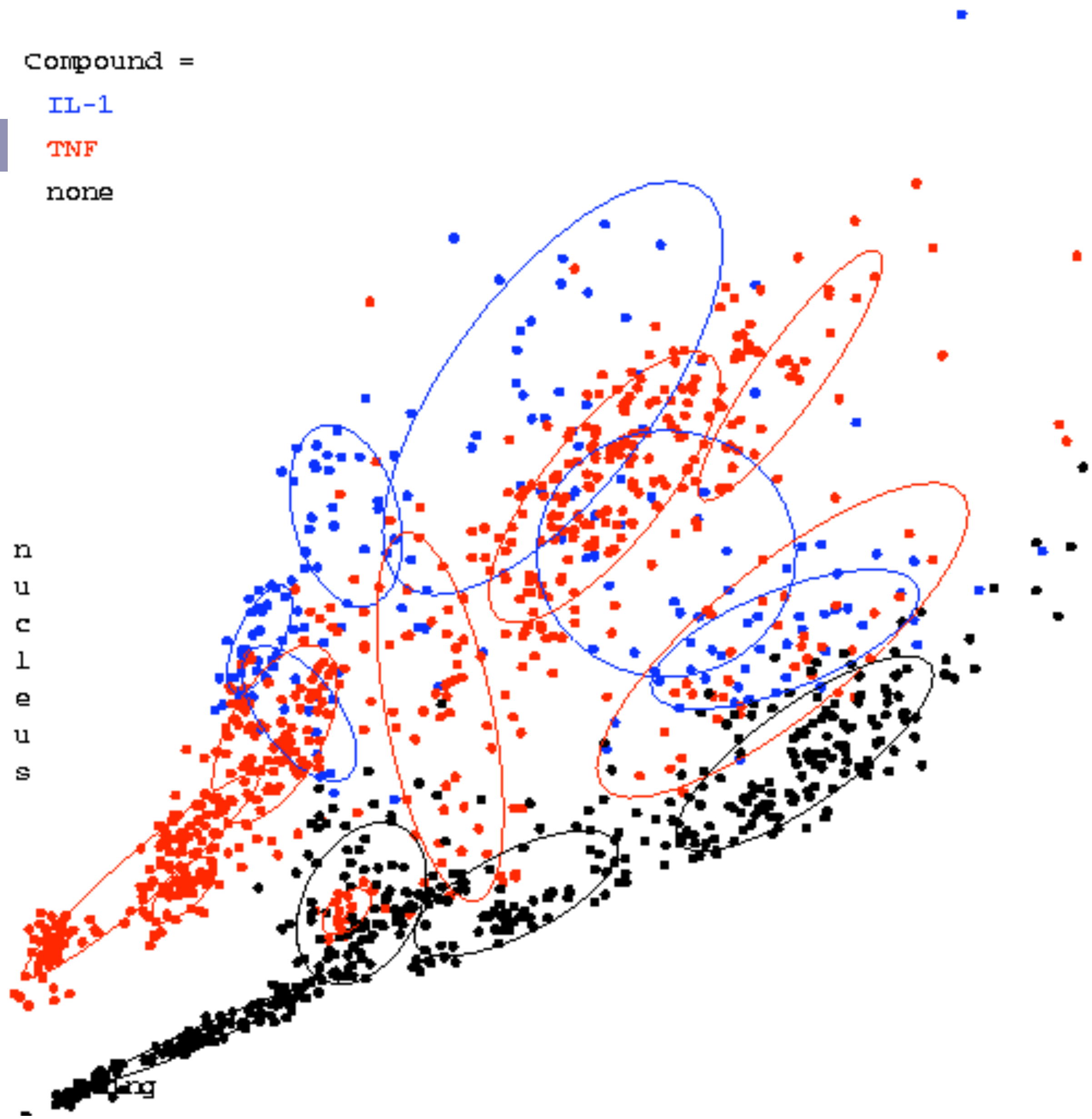


Resulting Density Estimator

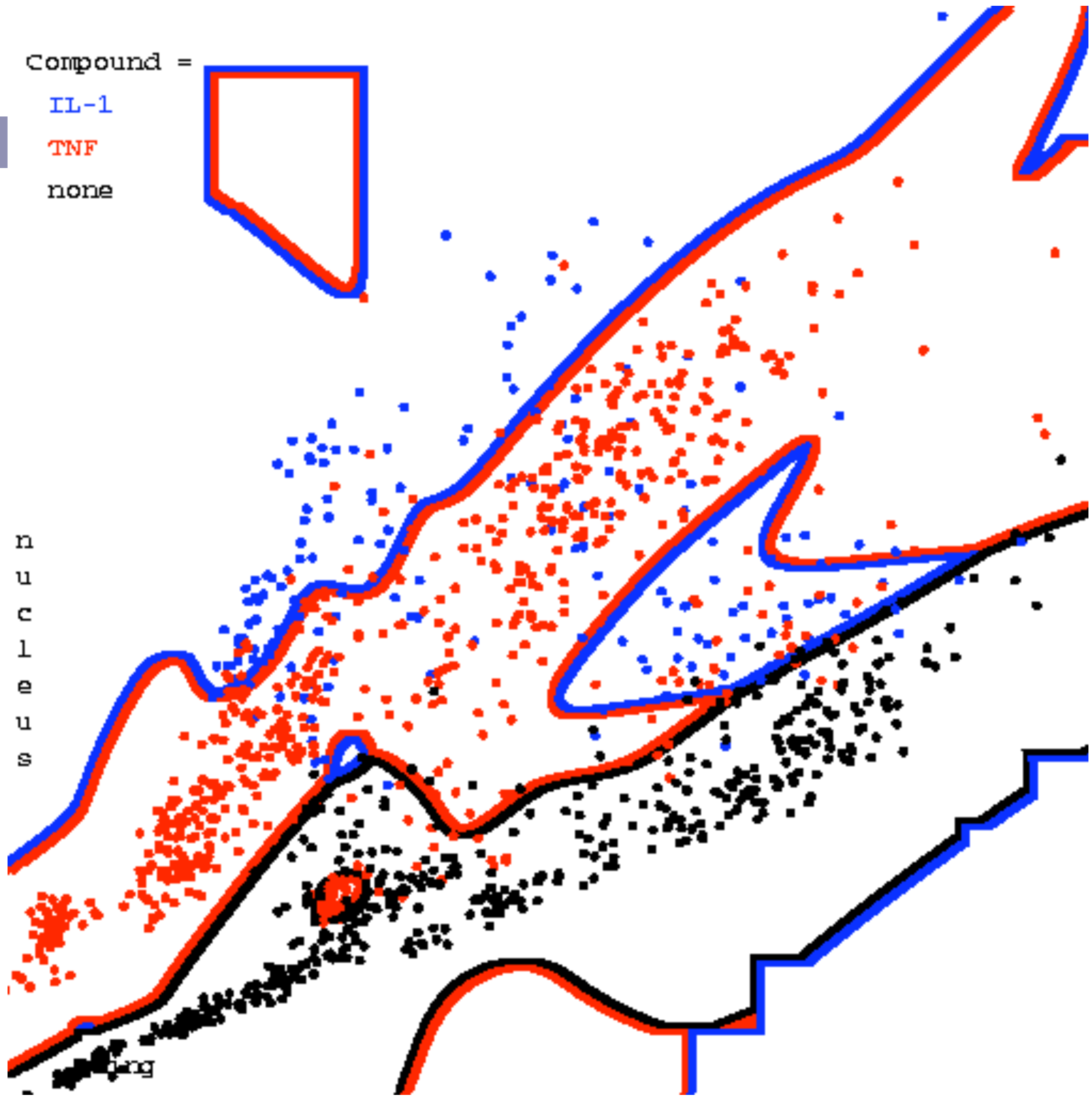


Three classes of assay

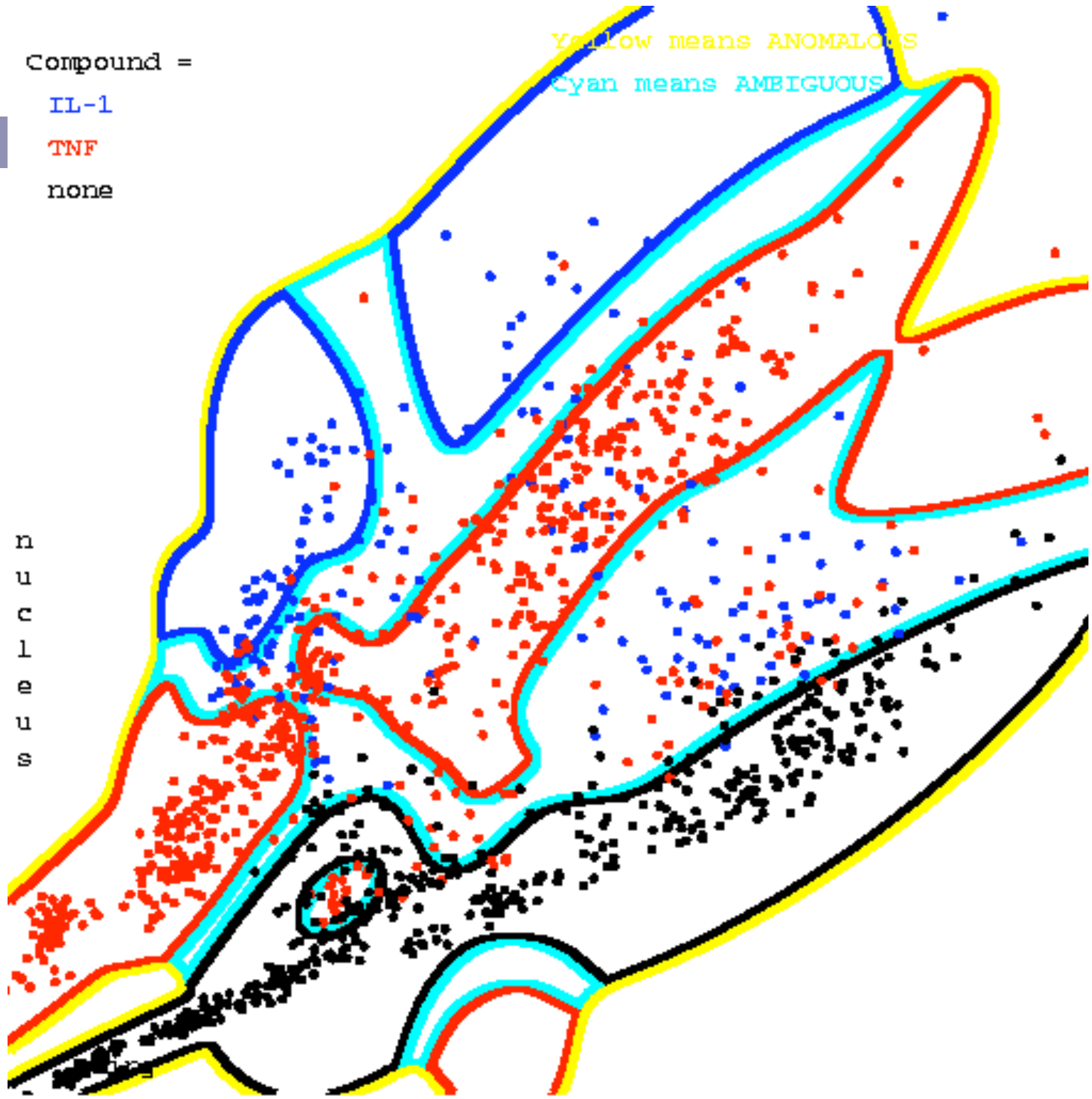
(each learned with it's own mixture model)



Resulting Bayes Classifier



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



Yellow means anomalous

Cyan means ambiguous

Final Comments

- Remember, E.M. can get stuck in local minima, and empirically it DOES.
- Our unsupervised learning example assumed $P(y_i)$'s known, and variances fixed and known. Easy to relax this.
- It's possible to do Bayesian unsupervised learning instead of max. likelihood.

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.

Acknowledgements

- K-means & Gaussian mixture models presentation derived from excellent tutorial by Andrew Moore:
  - http://www.autonlab.org/tutorials/
- K-means Applet:
  - http://www.elet.polimi.it/upload/matteucc/Clustering/tutorial_html/AppletKM.html
- Gaussian mixture models Applet:
  - http://www.neurosci.aist.go.jp/%7Eakaho/MixtureEM.html