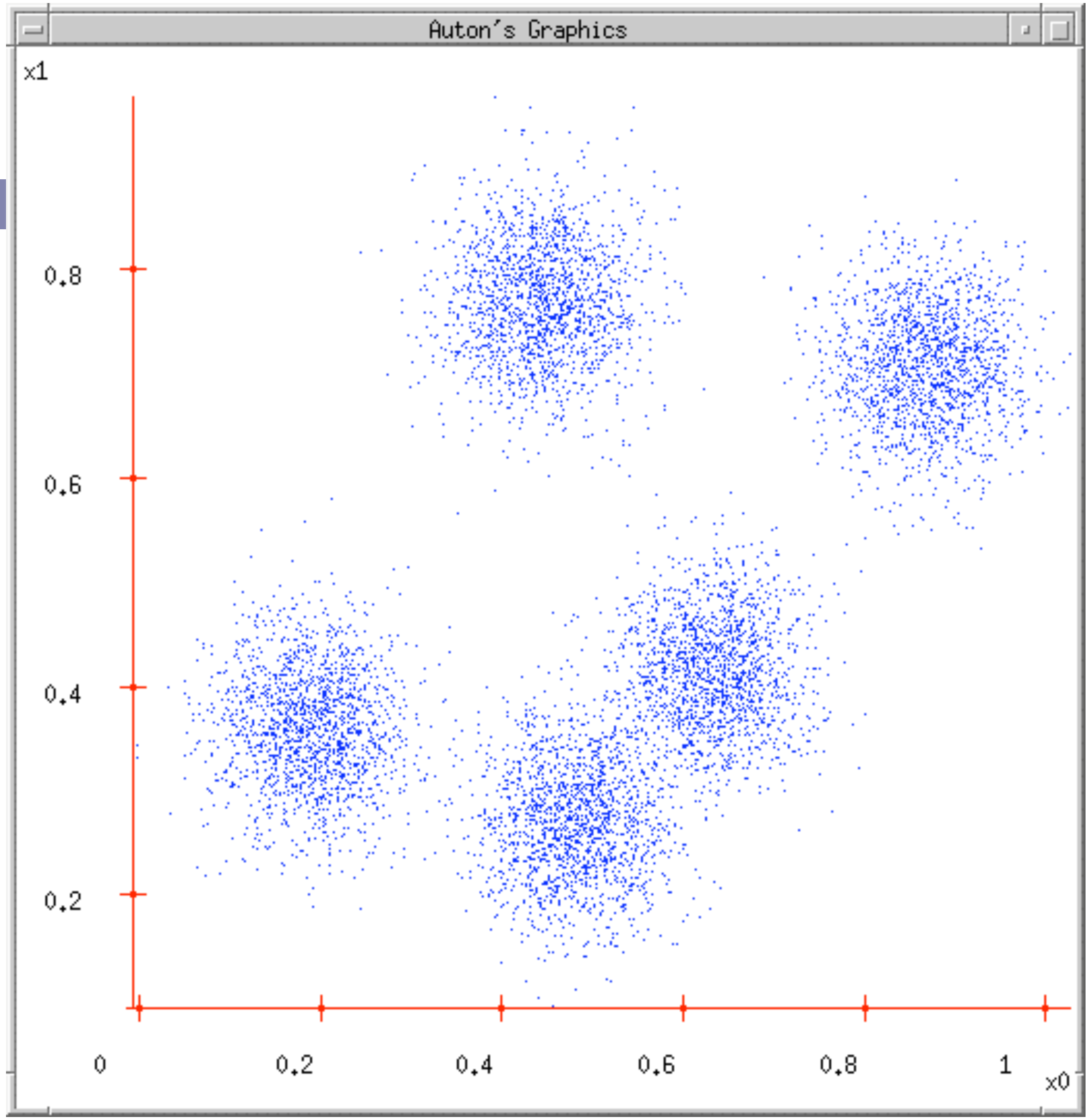
Unsupervised learning or Clustering – K-means Gaussian mixture models

Machine Learning – 10701/15781

Carlos Guestrin
Carnegie Mellon University

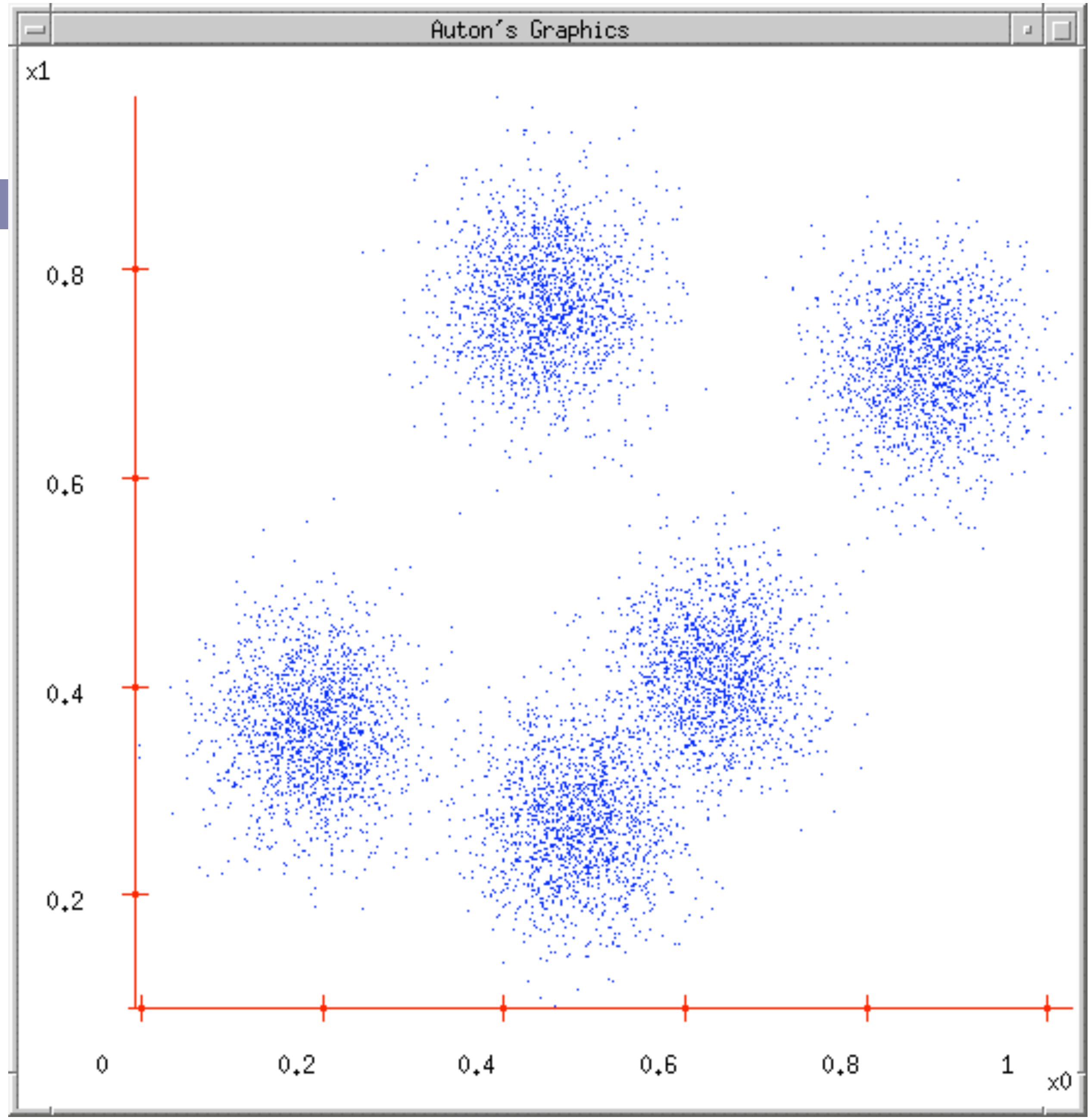
April 4th, 2007

Some Data



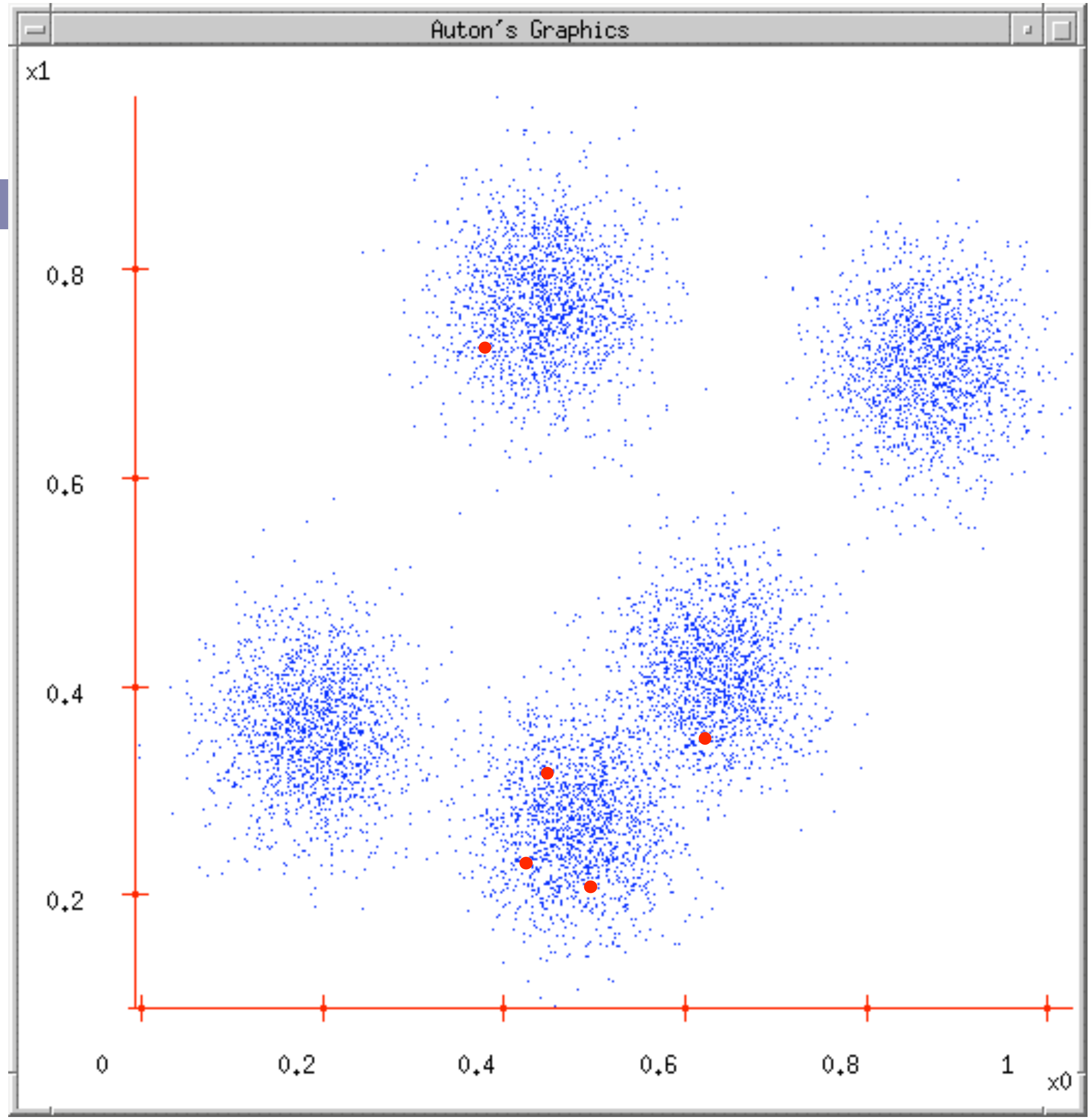
K-means

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



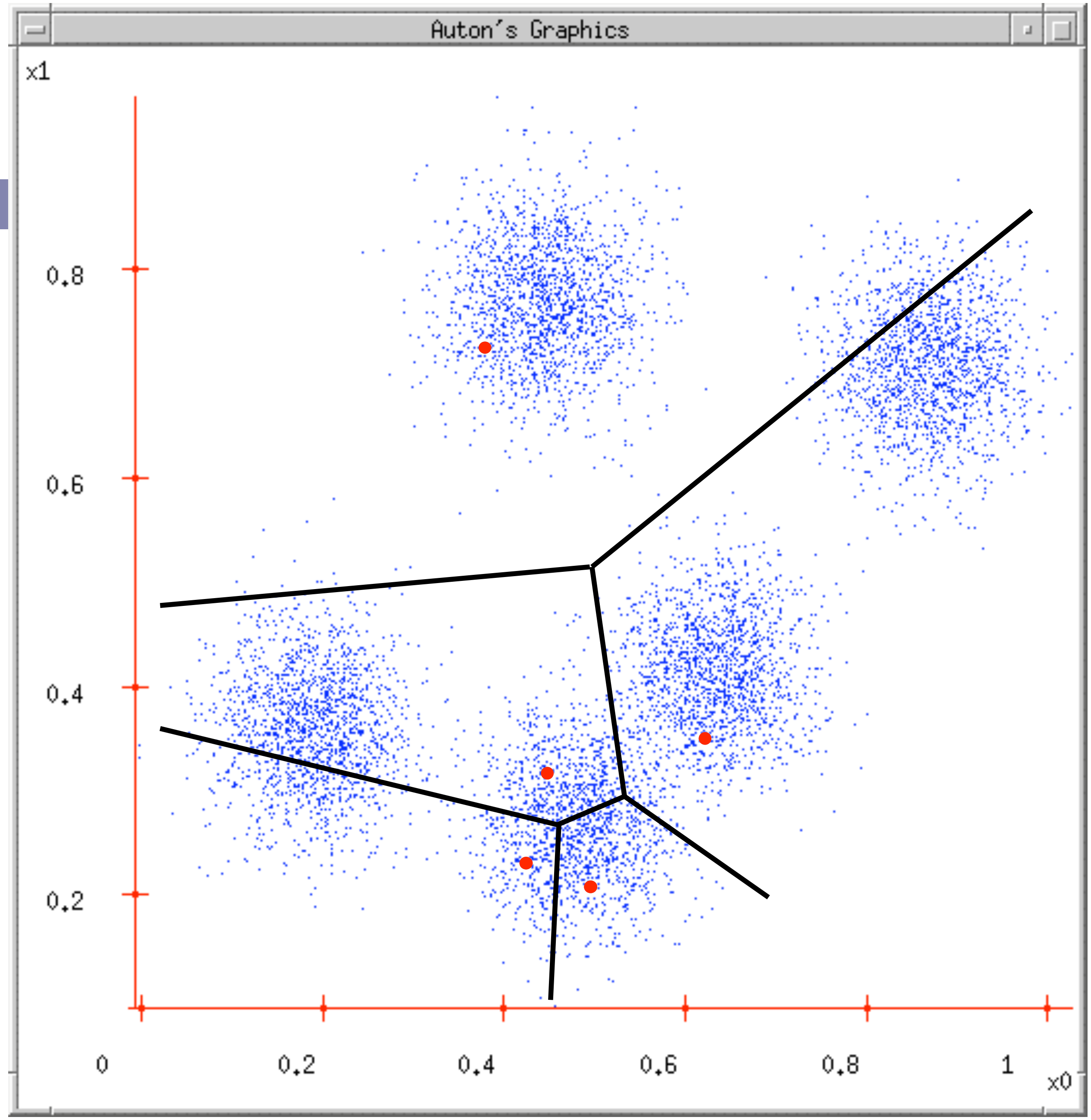
K-means

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



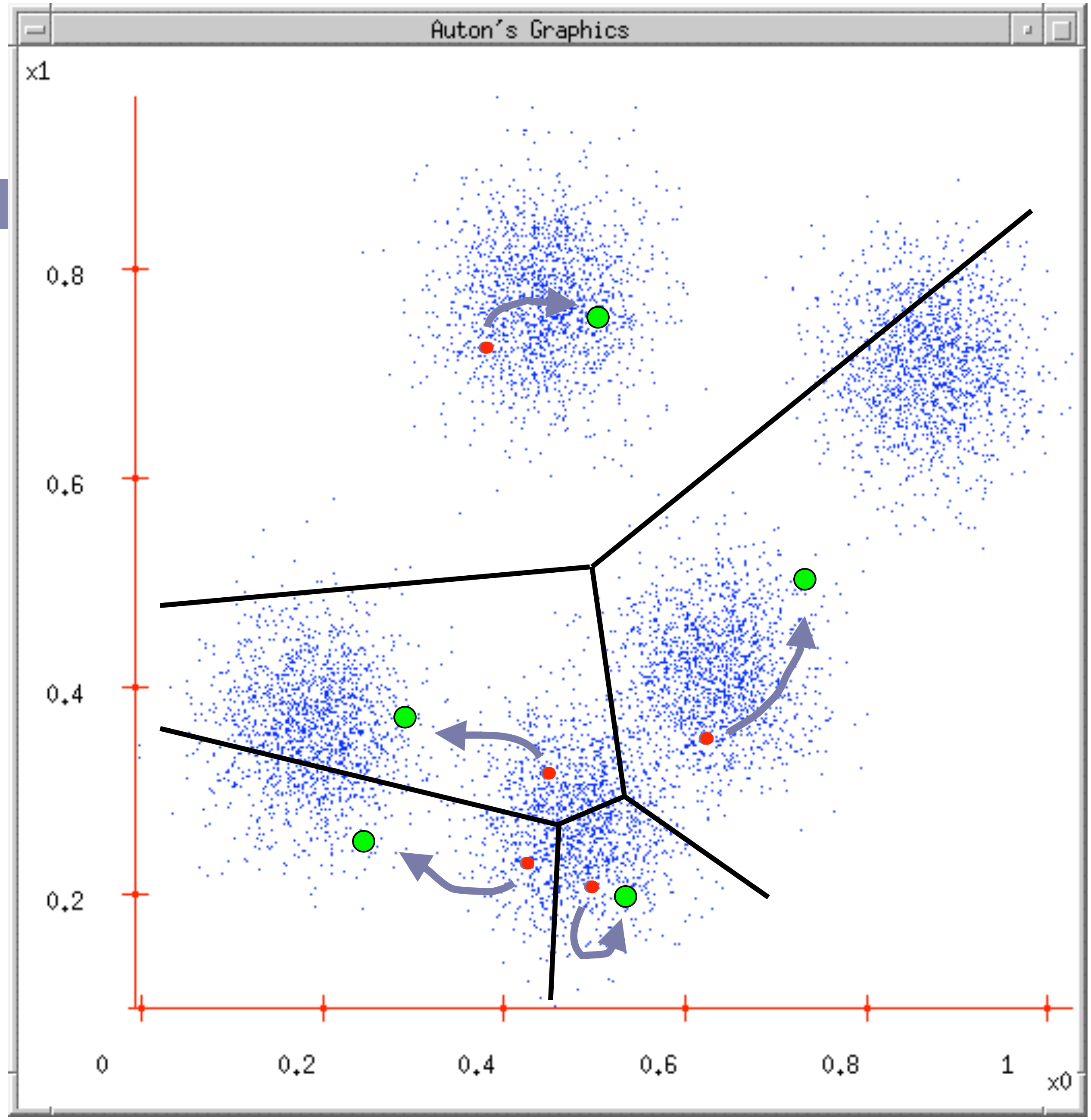
K-means

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)



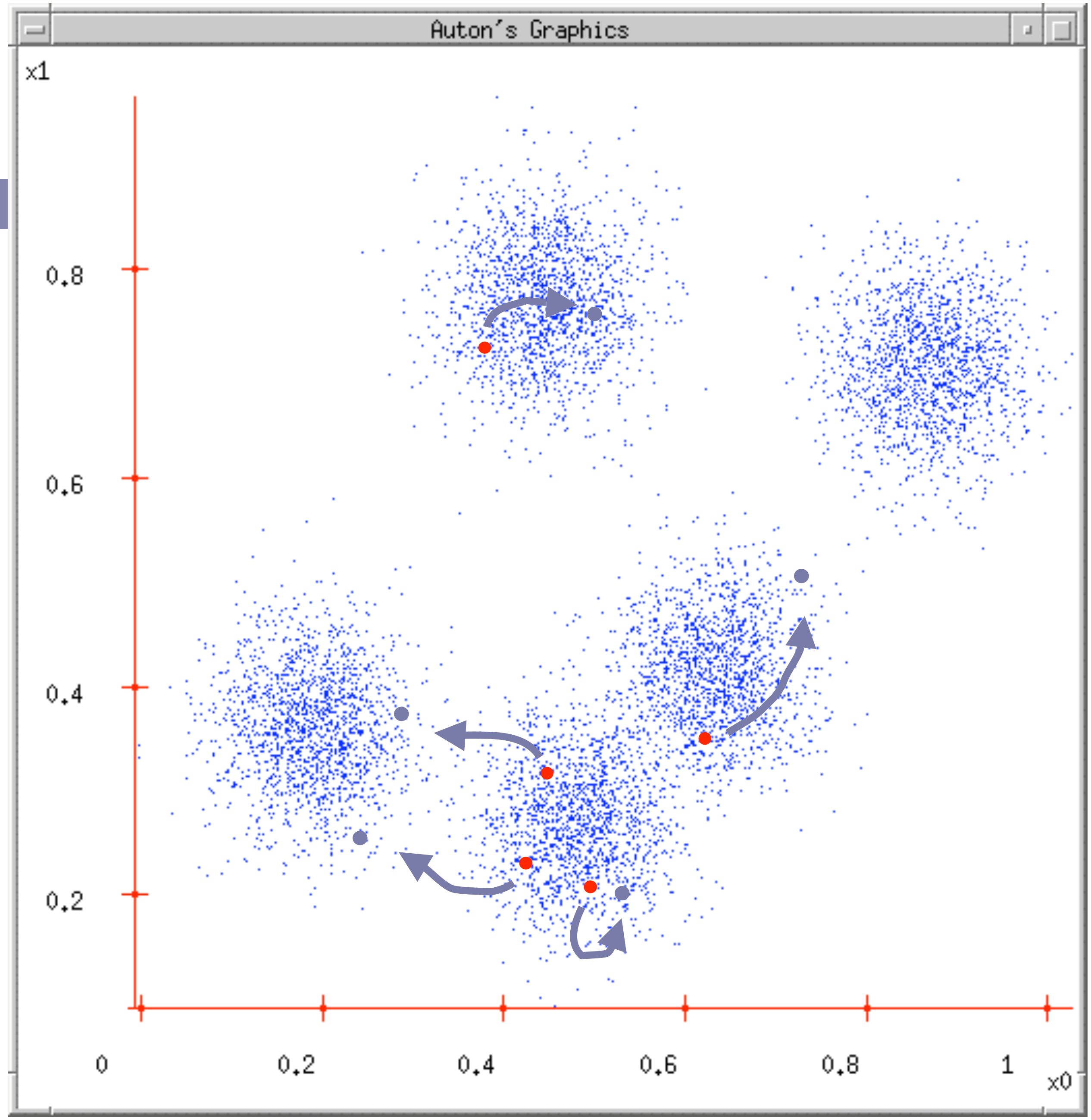
K-means

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!



K-means

- Randomly initialize *k* centers
  - $\mu^{(0)} = \mu_1^{(0)}, \ldots, \mu_k^{(0)}$
- **Classify**: Assign each point $j \in \{1, \ldots m\}$ to nearest center:
  - $C^{(t)}(j) \leftarrow \arg\min_i ||\mu_i - x_j||^2$
- **Recenter**: μ_i becomes centroid of its point:
  - $\mu_i^{(t+1)} \leftarrow \arg\min_\mu \sum_{j:C(j)=i} ||\mu - x_j||^2$
  - Equivalent to $\mu_i \leftarrow$ average of its points!

What is K-means optimizing?

- Potential function F(μ,C) of centers μ and point allocations C:
  - $$F(\mu, C) = \sum_{j=1}^{m} ||\mu_{C(j)} - x_j||^2$$

- Optimal K-means:
  - $\min_{\mu}\min_{C} F(\mu,C)$

Does K-means converge??? Part 1

- Optimize potential function:

$$\min_{\mu} \min_{C} F(\mu, C) = \min_{\mu} \min_{C} \sum_{i=1}^{k} \sum_{j:C(j)=i} ||\mu_i - x_j||^2$$

- Fix μ, optimize C

Does K-means converge??? Part 2

- Optimize potential function:

$$\min_{\mu} \min_{C} F(\mu, C) = \min_{\mu} \min_{C} \sum_{i=1}^{k} \sum_{j:C(j)=i} ||\mu_i - x_j||^2$$

- Fix C, optimize μ

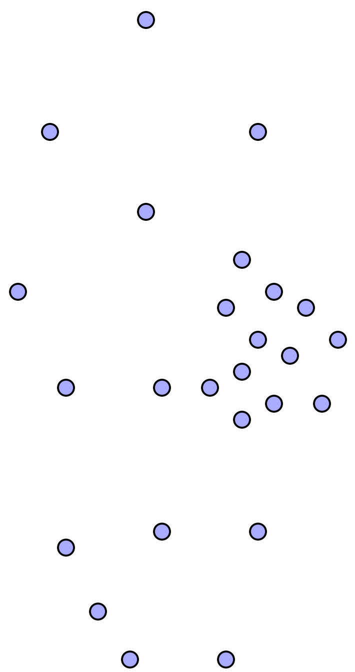
Coordinate descent algorithms

$$\min_{\mu} \min_{C} F(\mu, C) = \min_{\mu} \min_{C} \sum_{i=1}^{k} \sum_{j:C(j)=i} ||\mu_i - x_j||^2$$

- Want: $\min_a \min_b F(a,b)$
- Coordinate descent:
  - fix a, minimize b
  - fix b, minimize a
  - repeat
- Converges!!!
  - if F is bounded
  - to a (often good) local optimum
    - as we saw in applet (play with it!)

- K-means is a coordinate descent algorithm!

(One) bad case for k-means



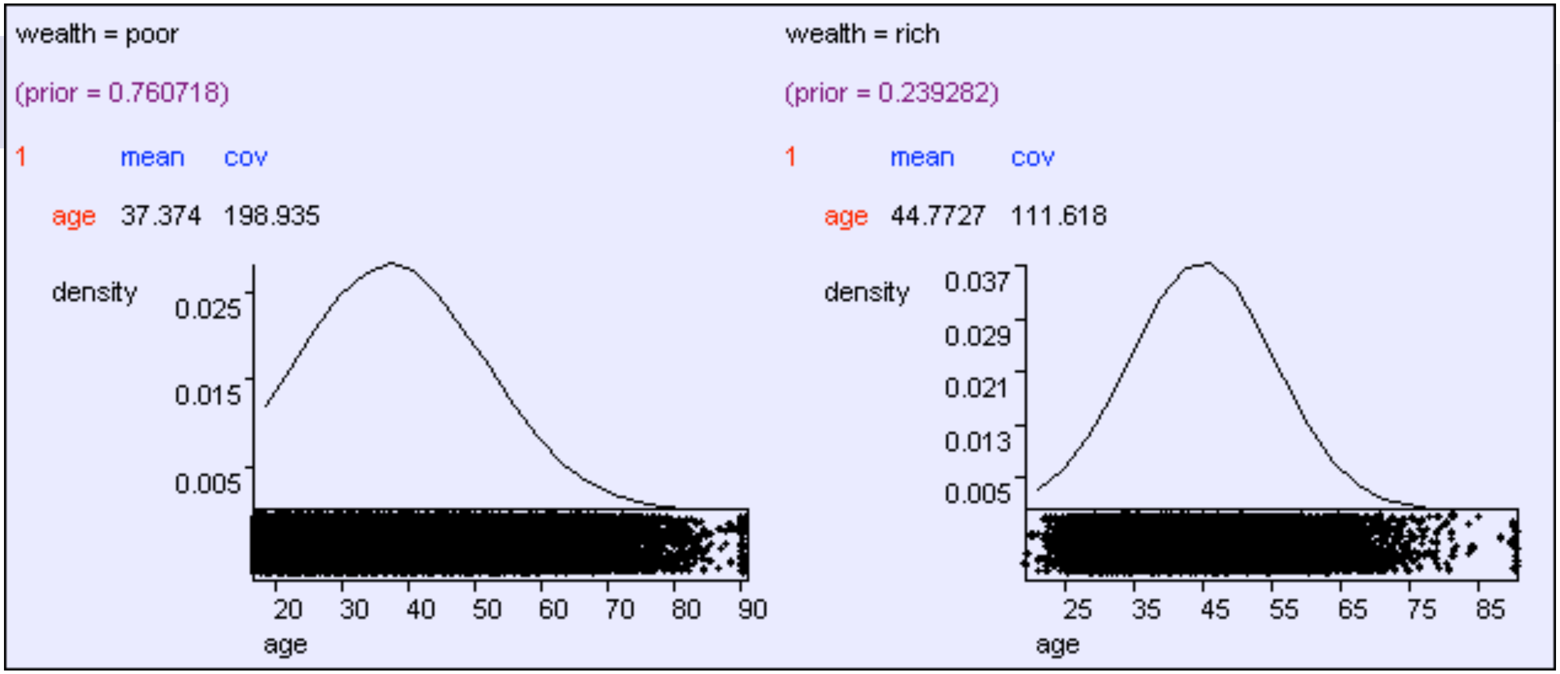
- Clusters may overlap
- Some clusters may be "wider" than others

Gaussian Bayes Classifier Reminder

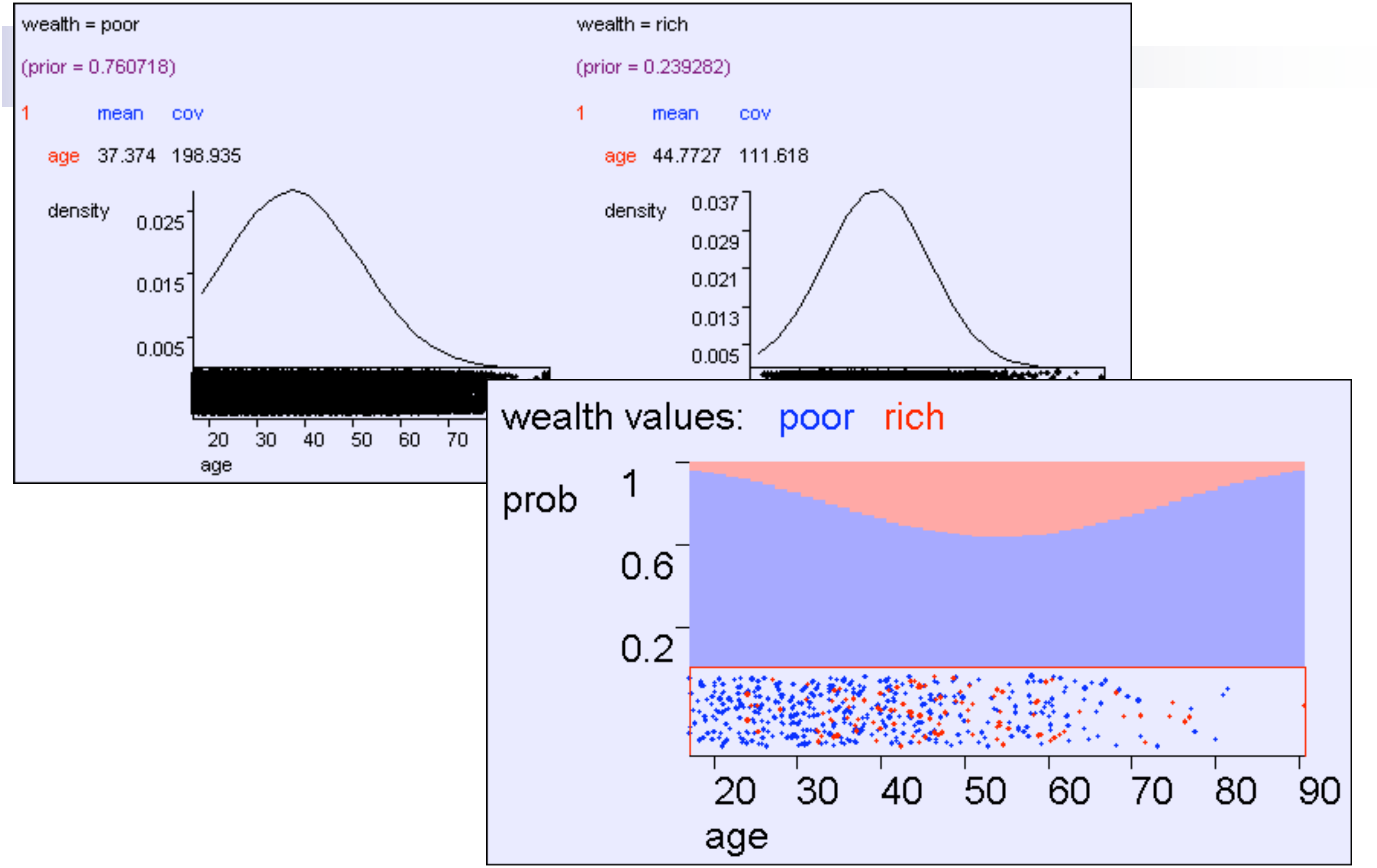
$$P(y=i \mid \mathbf{x}_j) = \frac{p(\mathbf{x}_j \mid y=i)P(y=i)}{p(\mathbf{x}_j)}$$

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

Predicting wealth from age

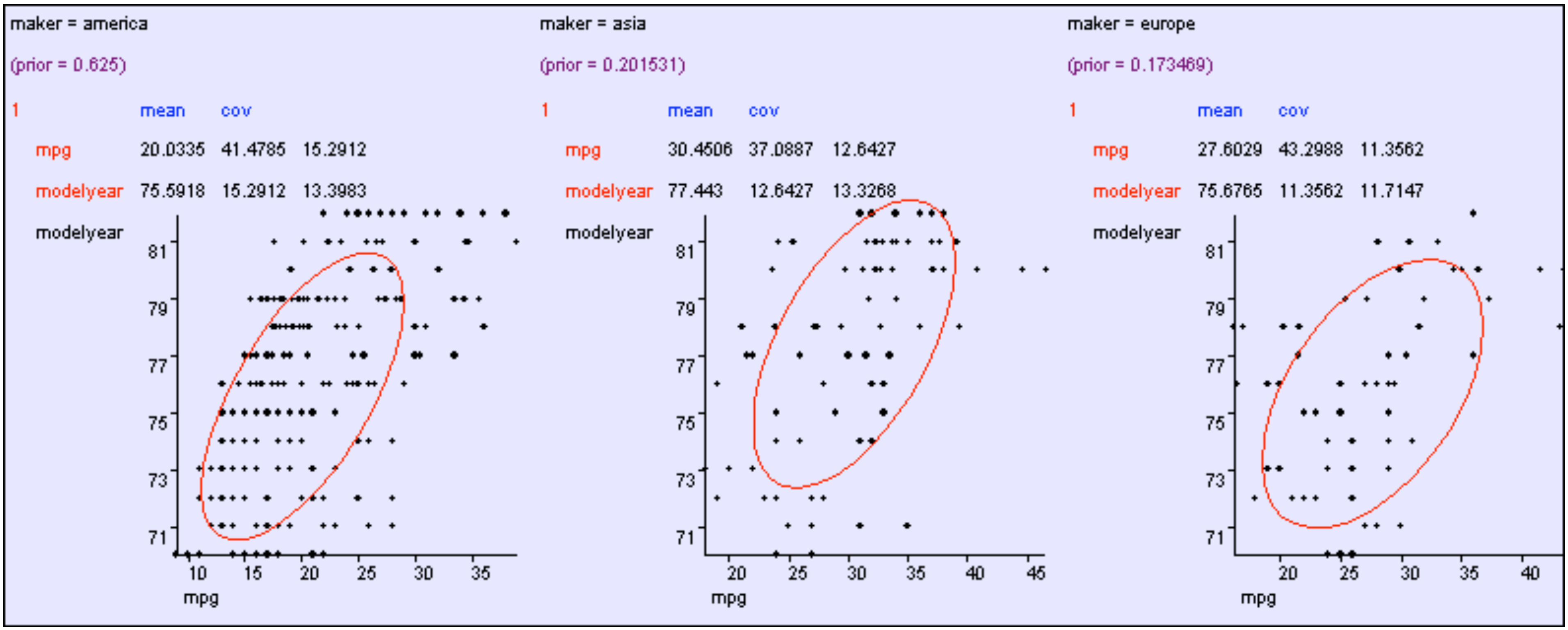


Predicting wealth from age



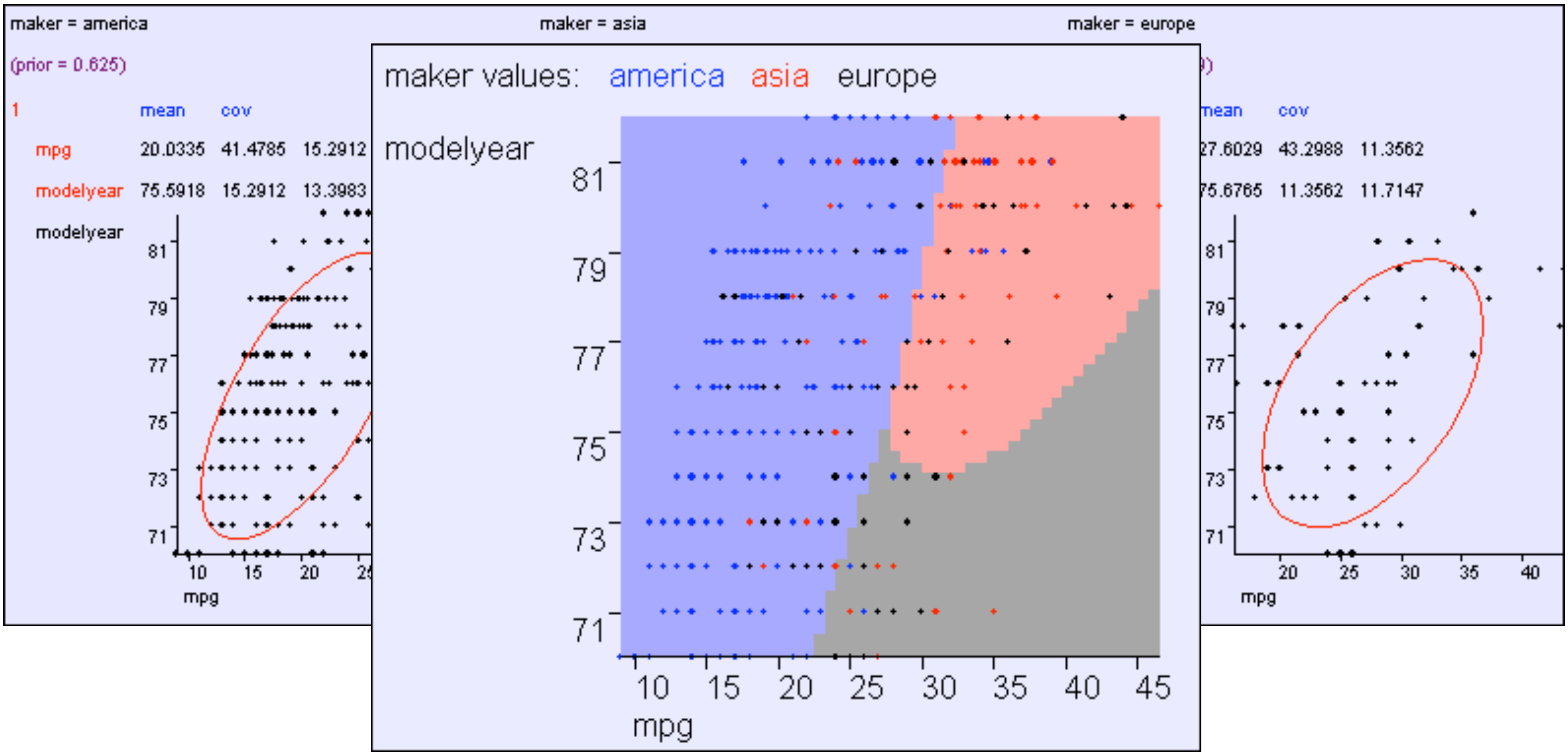
Learning modelyear , mpg ---> maker

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



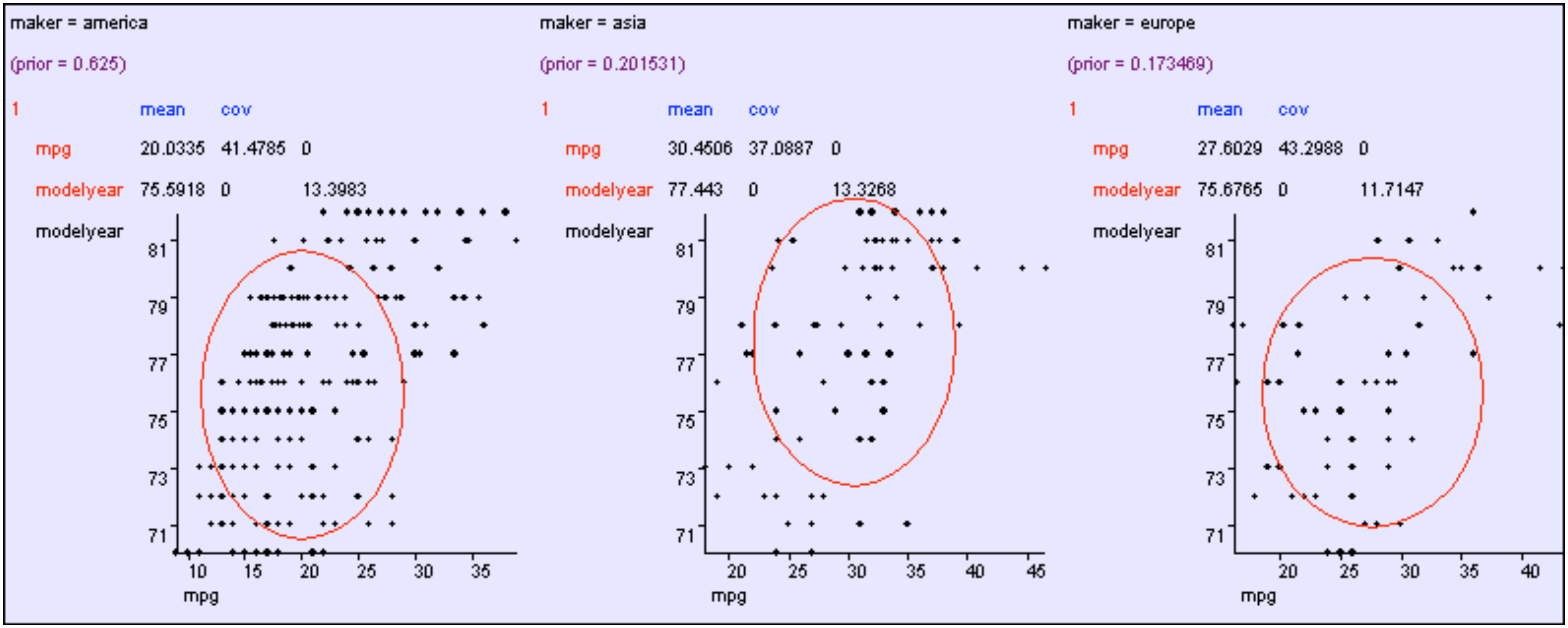
General: $O(m^2)$ parameters

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



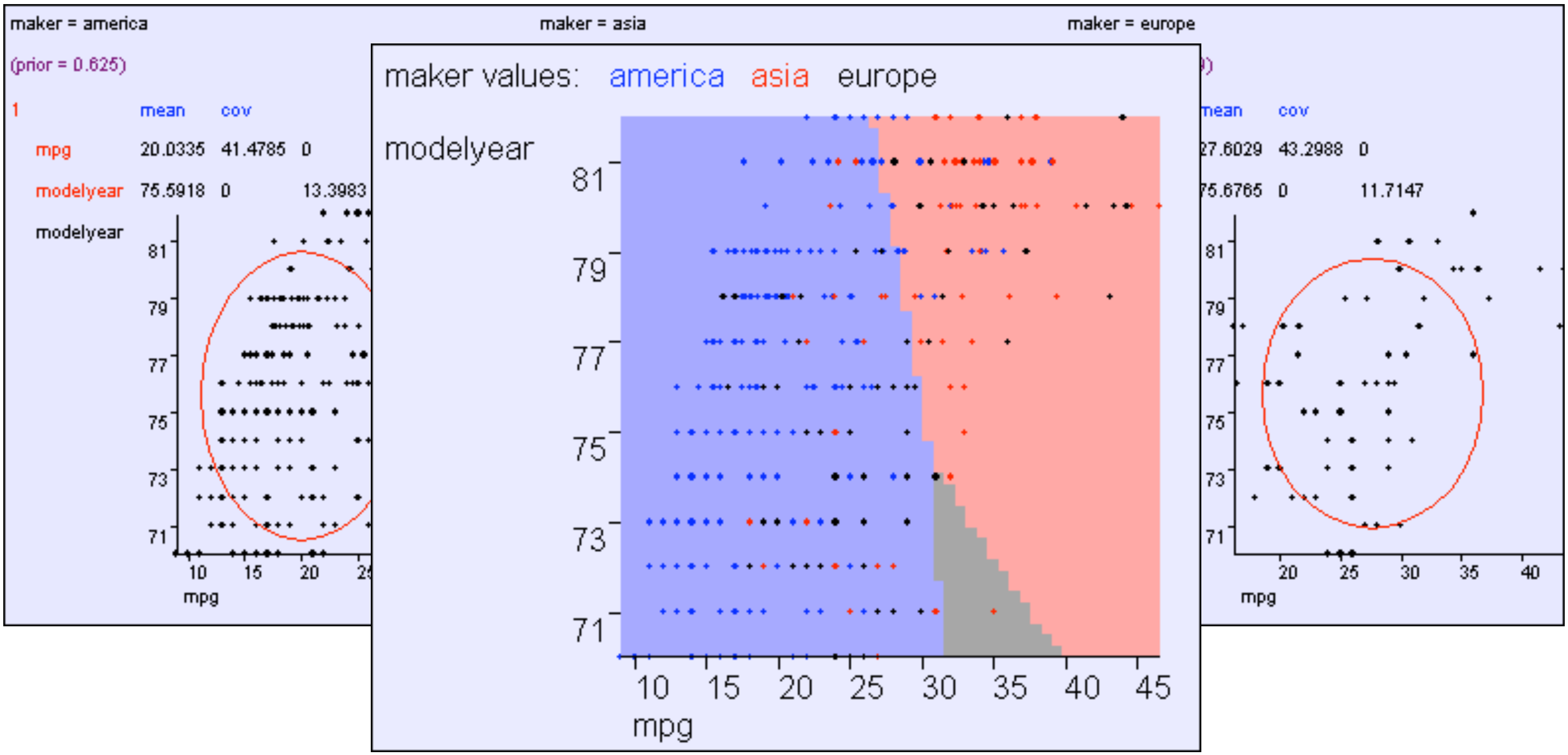
Aligned: *O(m)* parameters

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



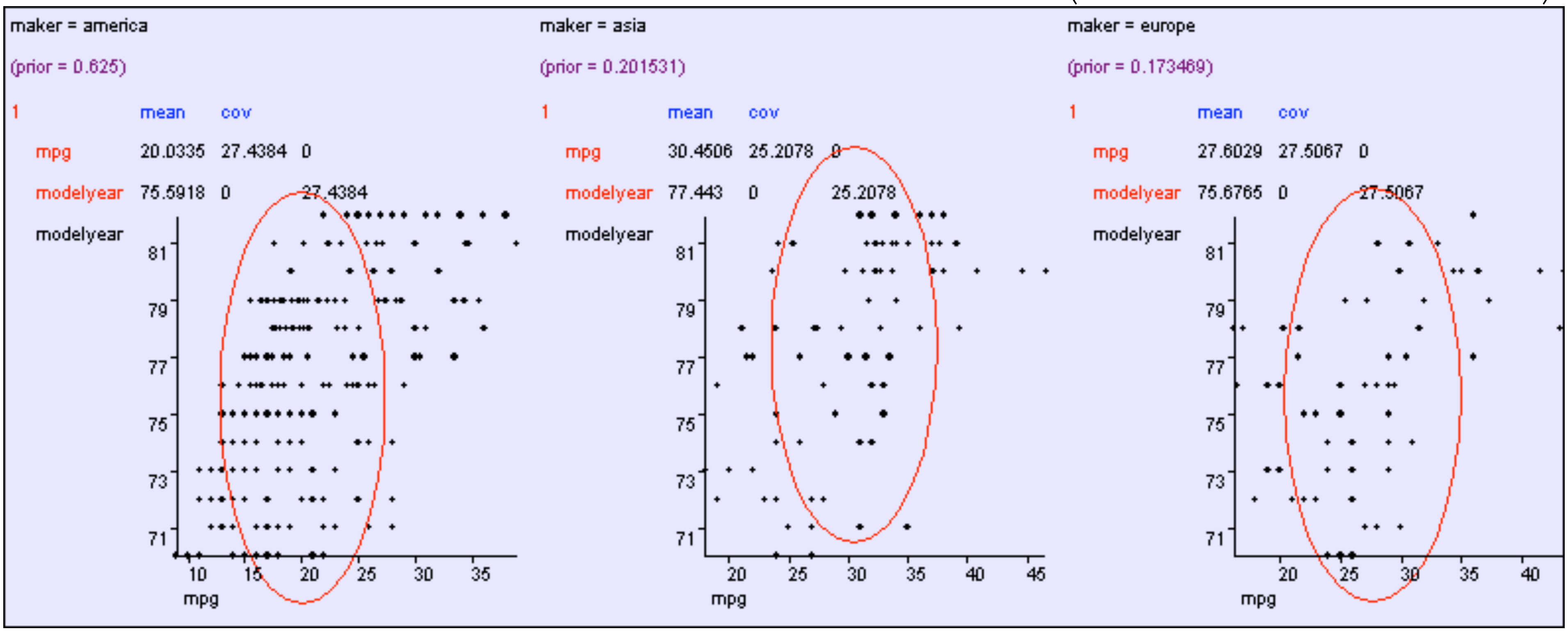
Aligned: *O(m)* parameters

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



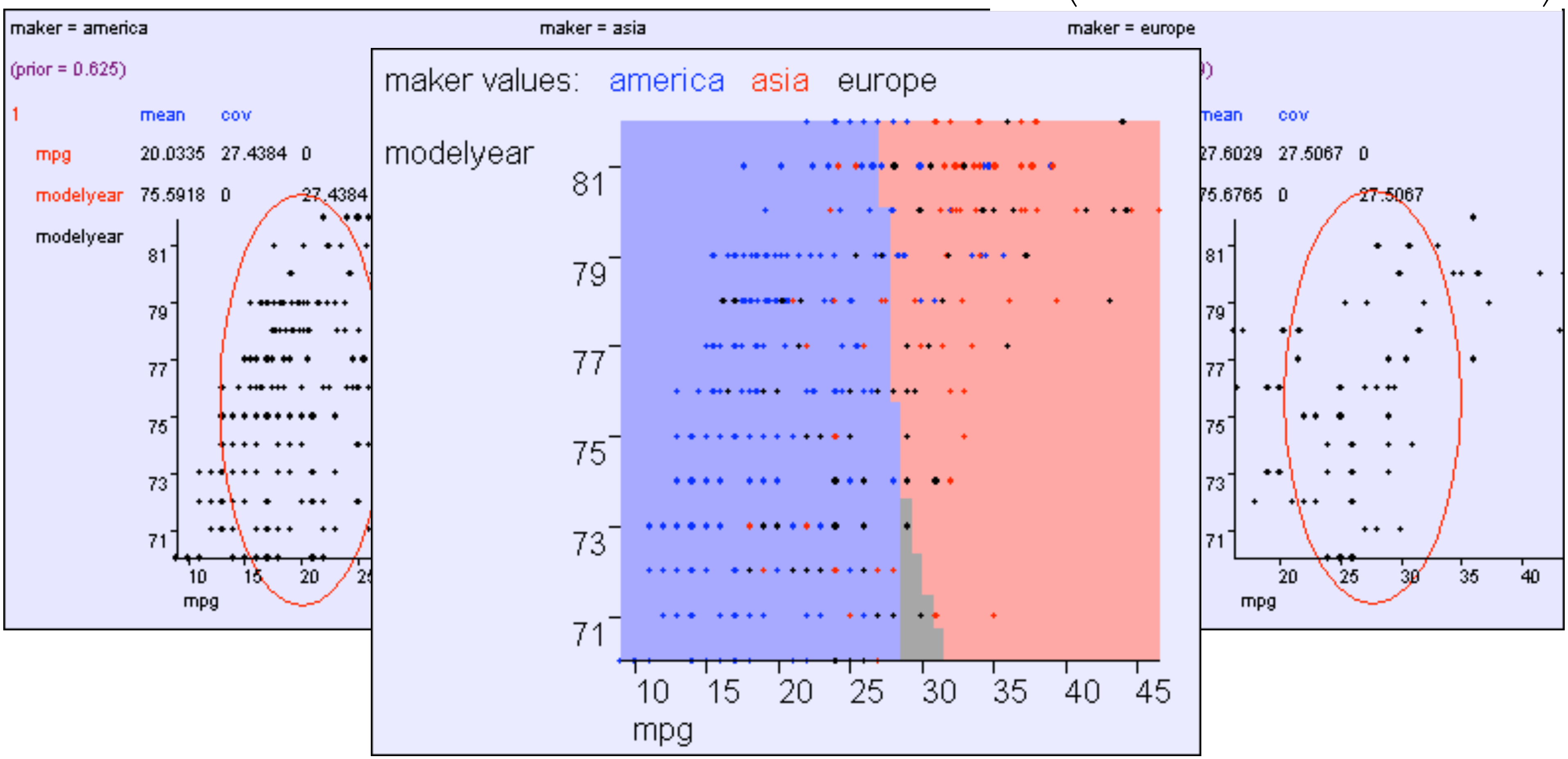
Spherical: *O(1)* cov parameters

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



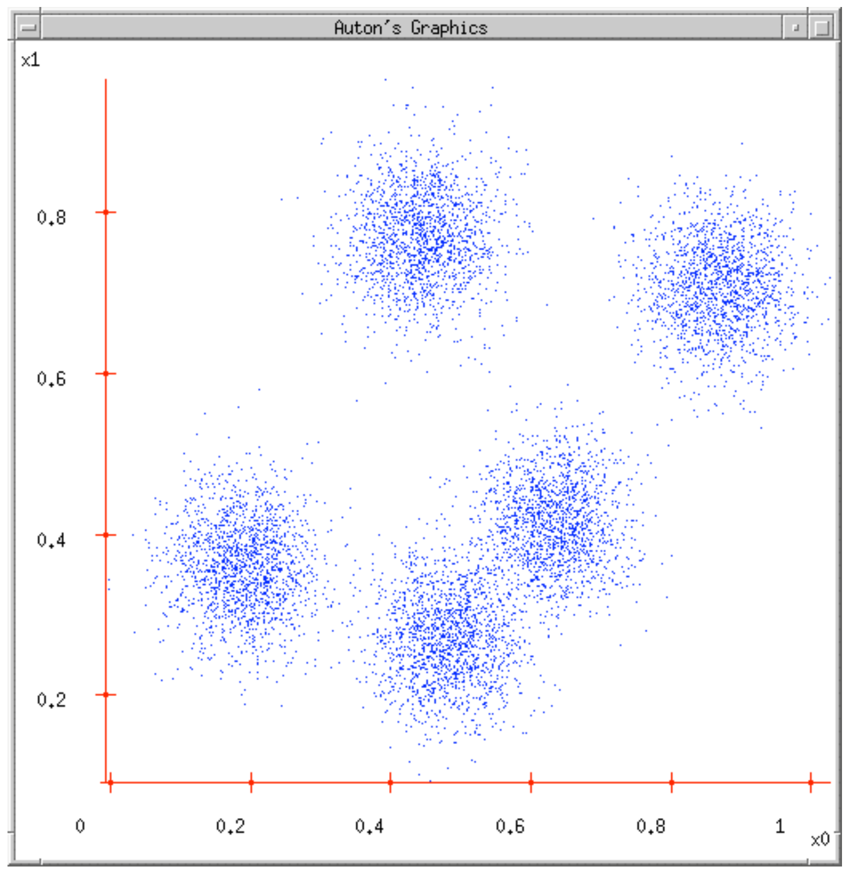
Spherical: *O(1)* cov parameters

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

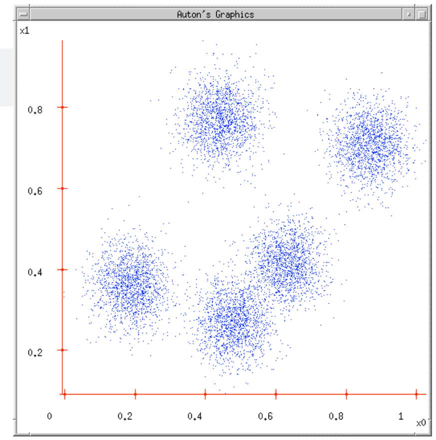


Next… back to Density Estimation

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's!!!
- 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:

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

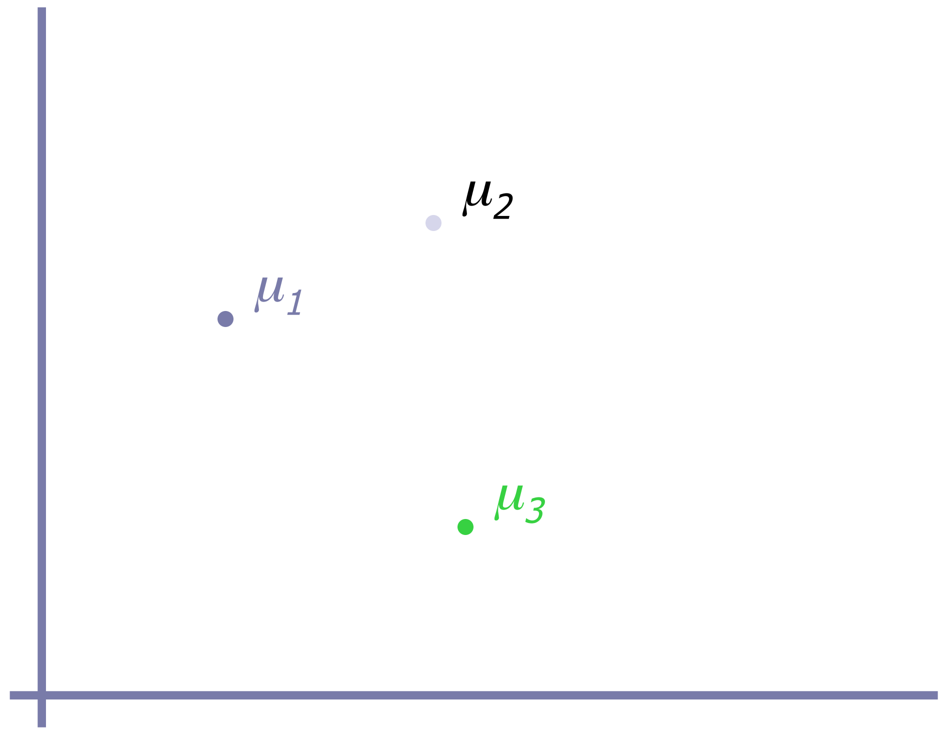
- If each x_j belongs to one class C(j) (hard assignment), marginal likelihood:

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

- Same as K-means!!!

The GMM assumption

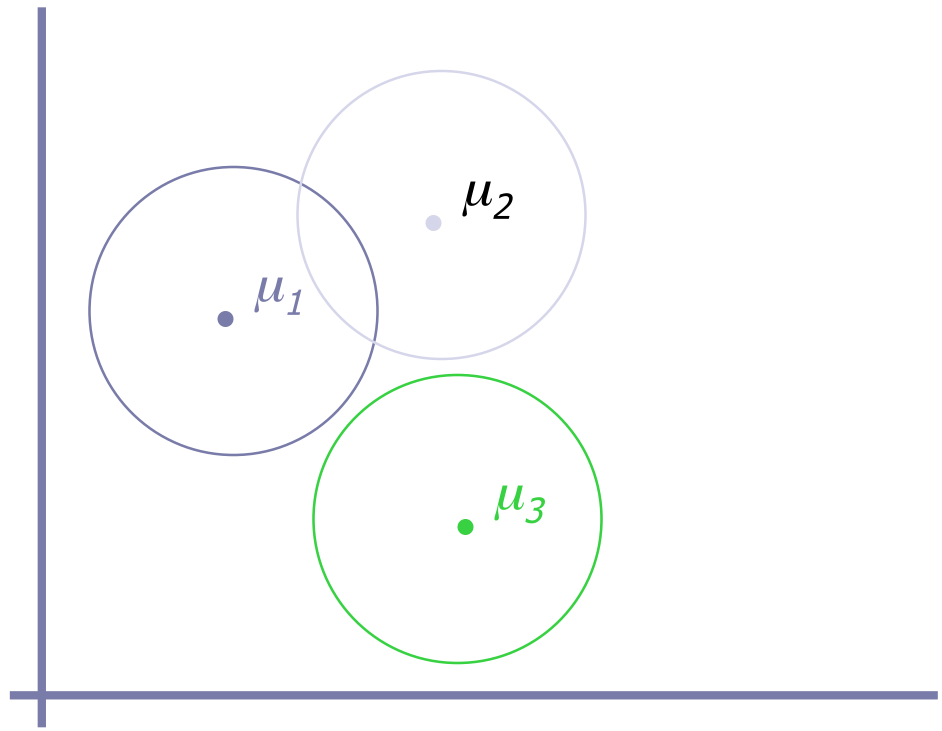
- There are k components
- Component *i* has an associated mean vector μ_i



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 μ_i and covariance matrix $\sigma^2 \boldsymbol{I}$

Each data point is generated according to the following recipe:

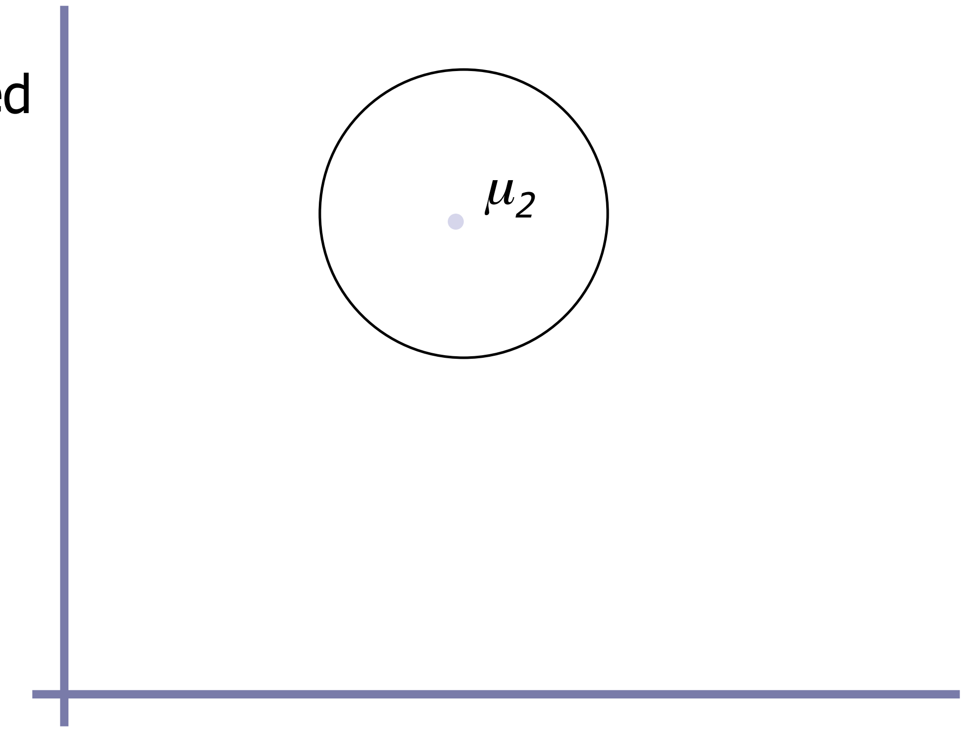


The GMM assumption

- There are k components
- Component *i* has an associated mean vector μ_i
- Each component generates data from a Gaussian with mean μ_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)*

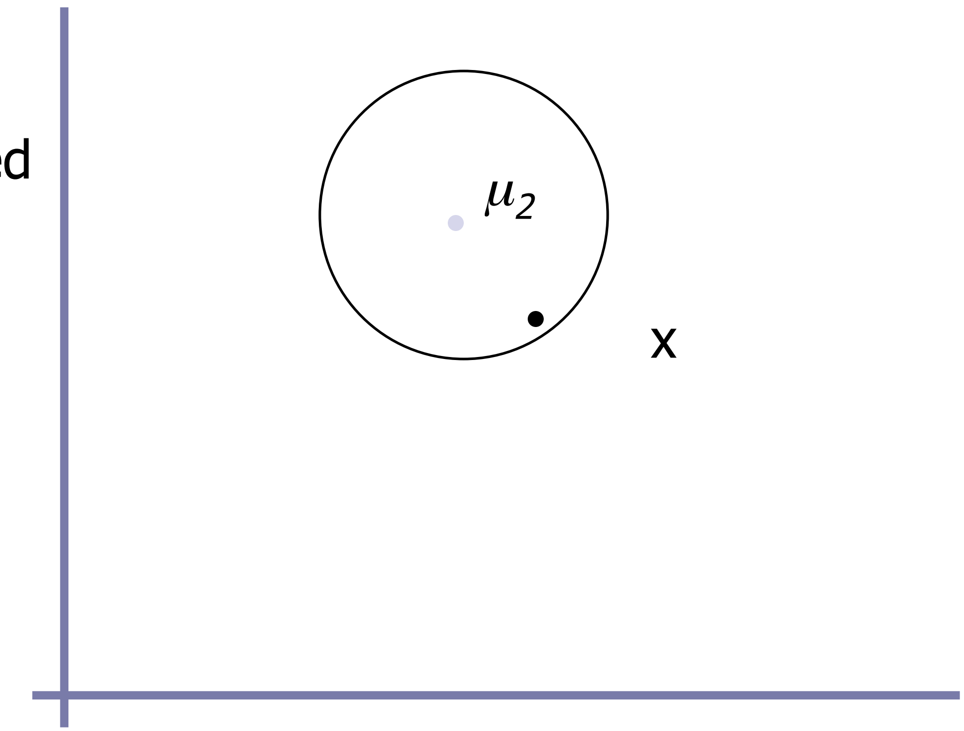


The GMM assumption

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

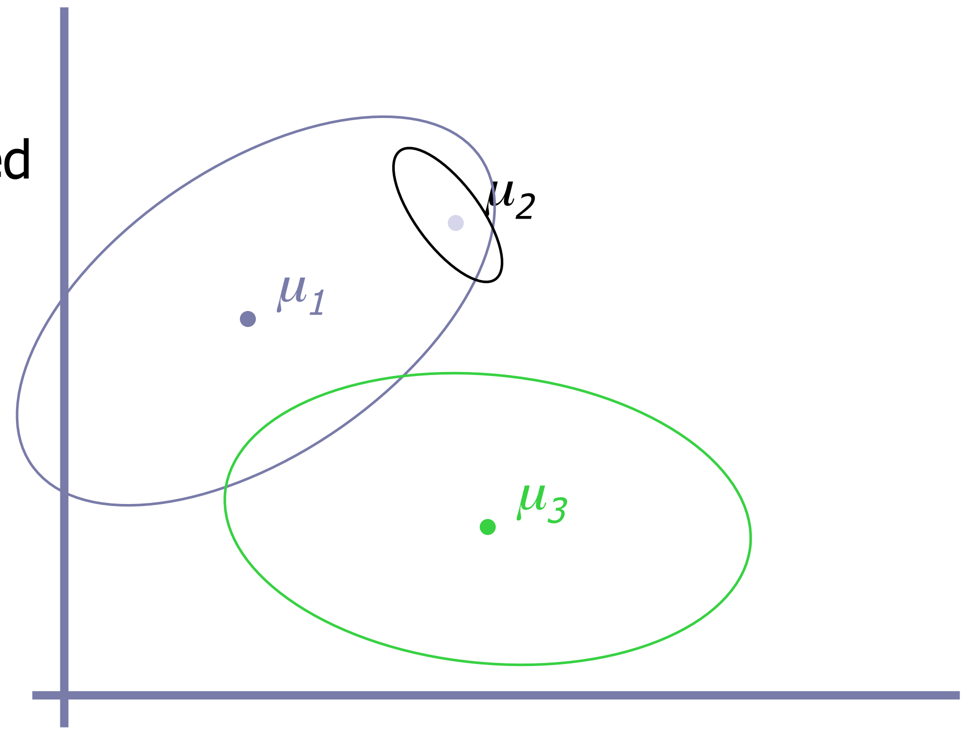
Each data point is generated according to the following recipe:

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



The General GMM assumption

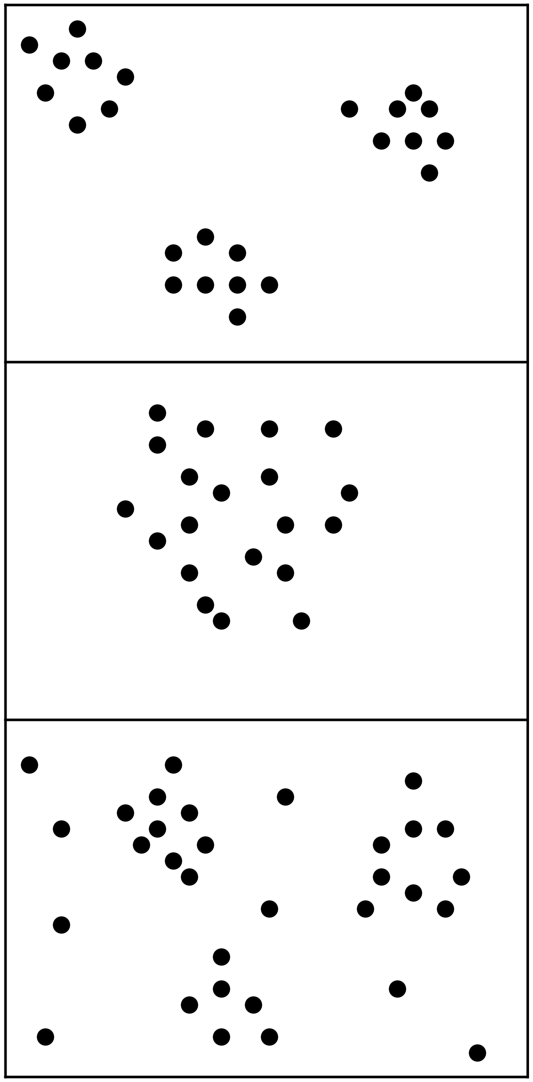
- There are k components
- Component i has an associated mean vector μ_i
- Each component generates data from a Gaussian with mean μ_i and covariance matrix Σ_i



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(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*

Marginal likelihood for general case

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

- Marginal likelihood:

$$\prod_{j=1}^{m} P(\mathbf{x}_j) = \prod_{j=1}^{m} \sum_{i=1}^{k} P(\mathbf{x}_j, y = i)$$

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

Special case 2: spherical Gaussians and soft assignments

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

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

- Uncertain about class of each x_j (soft assignment), marginal likelihood:

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

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.

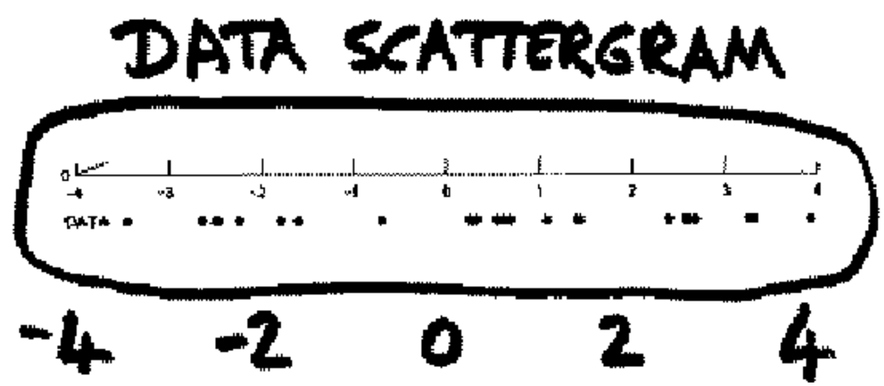
(From Duda and Hart)

There are two classes; w_1 and w_2

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

There are 25 unlabeled datapoints

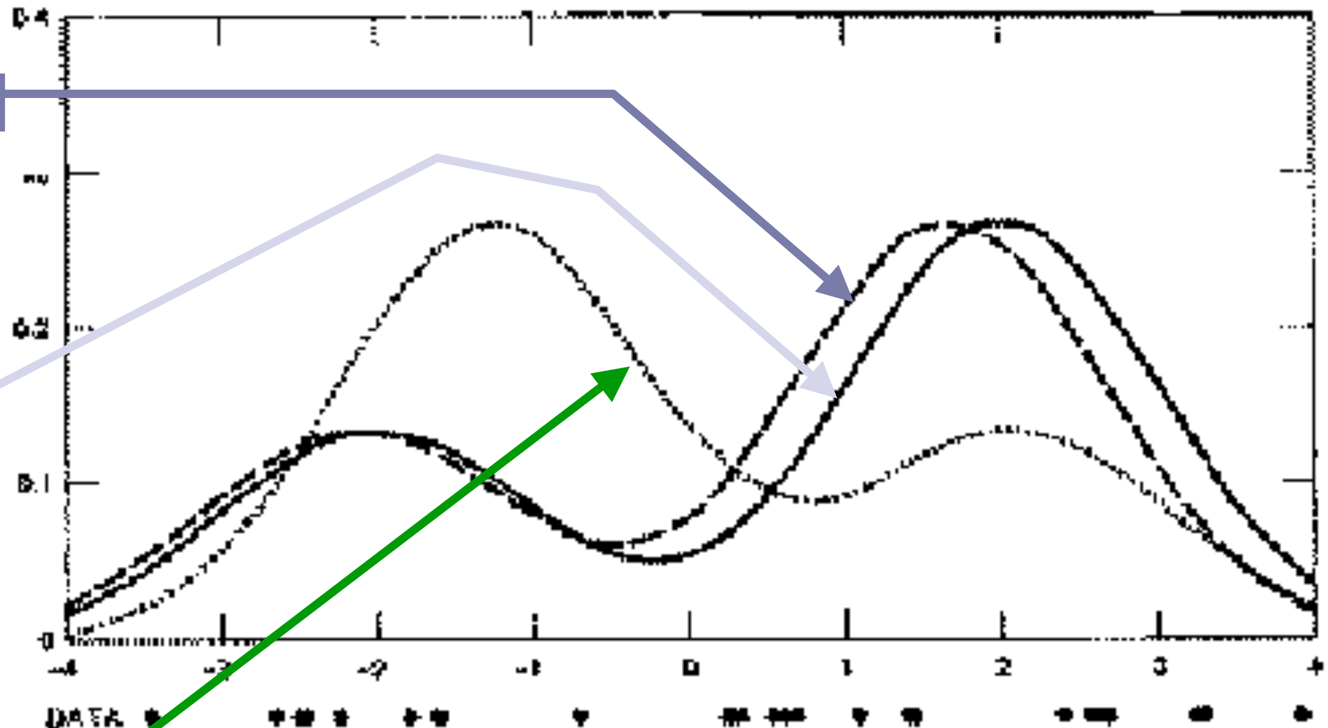
$x_1 = 0.608$
$x_2 = -1.590$
$x_3 = 0.235$
$x_4 = 3.949$
:
$x_{25} = -0.712$



Duda & Hart's Example

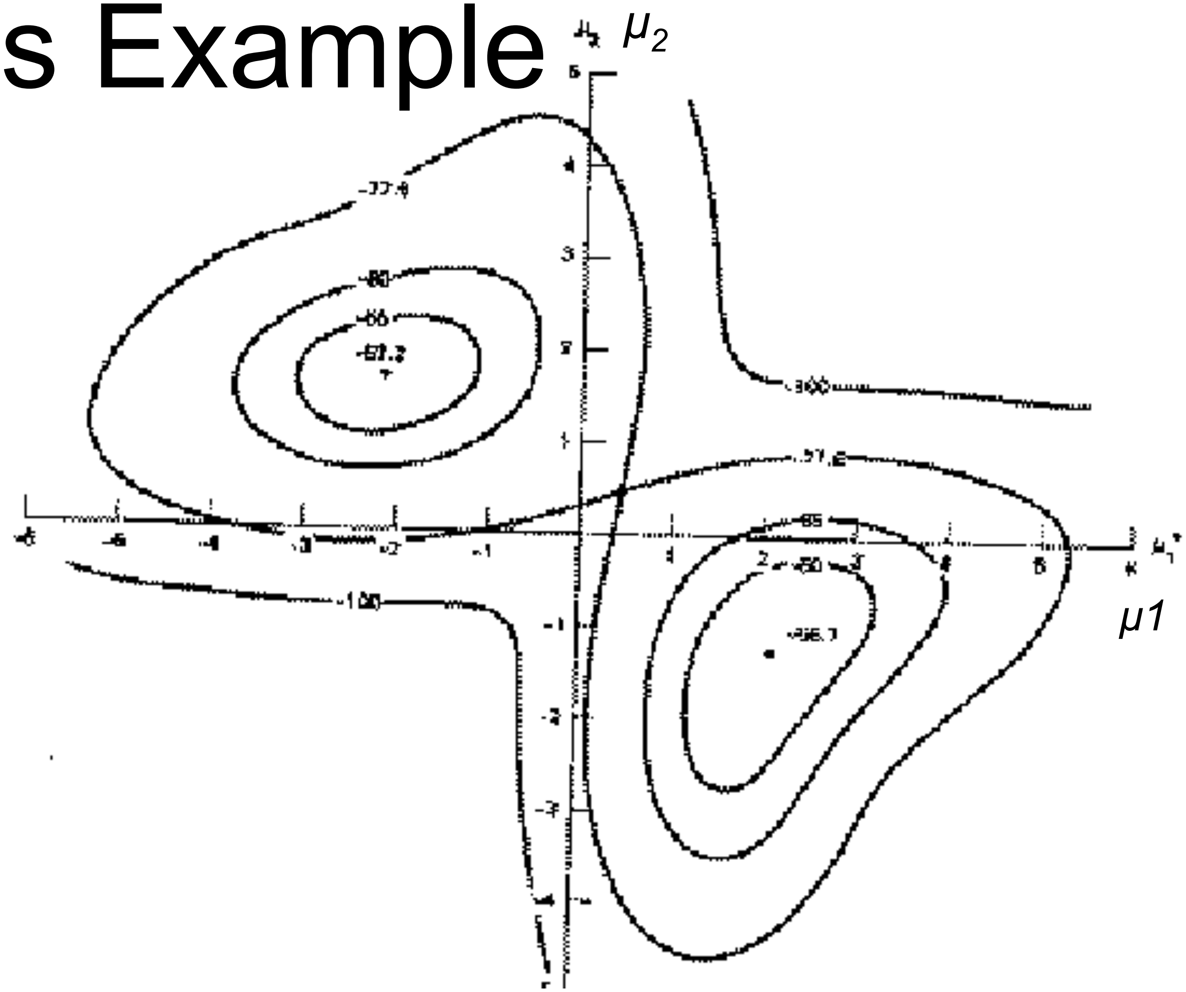
We can graph the prob. dist. function of data given our μ_1 and μ_2 estimates.

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?

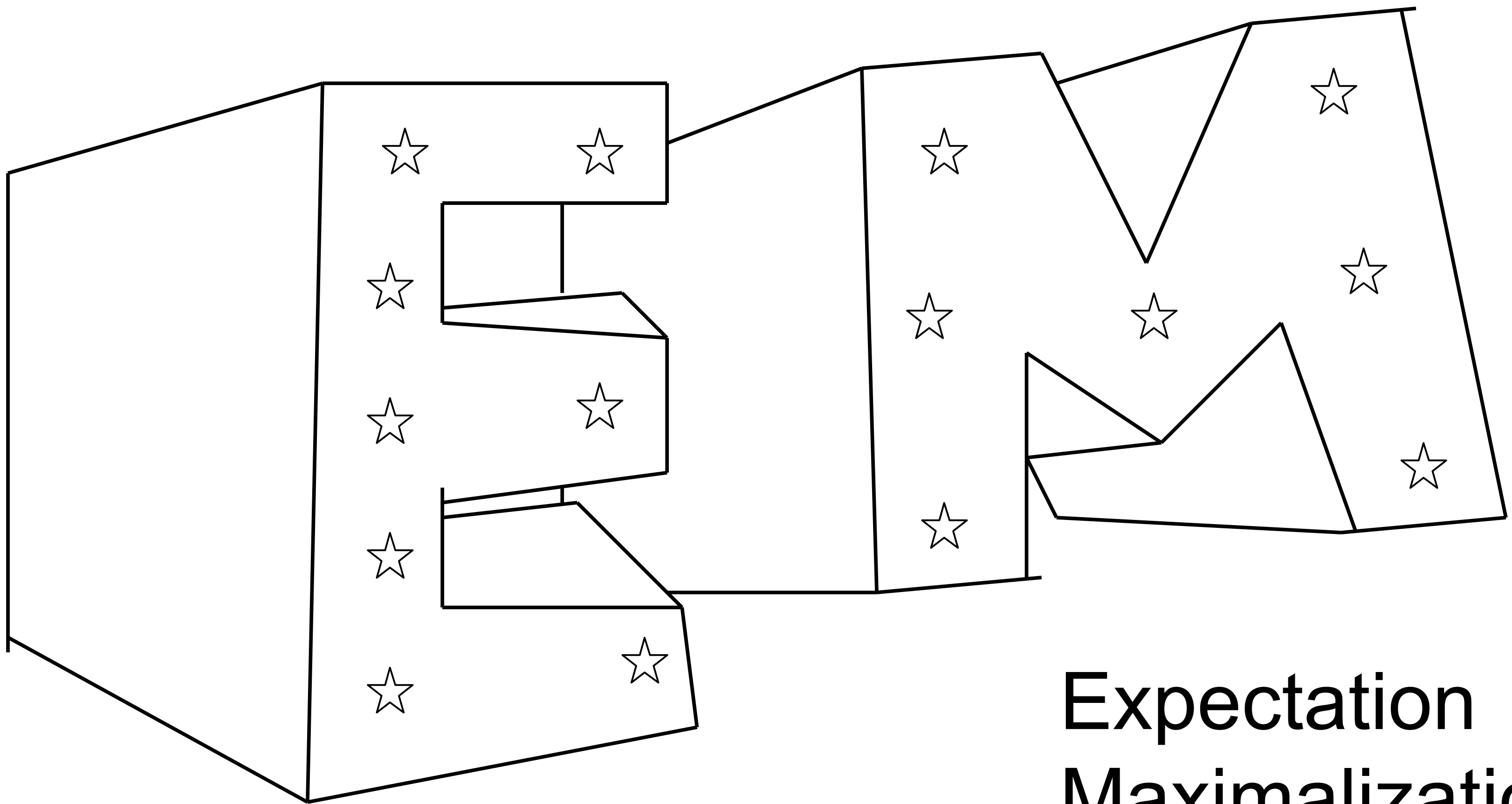
- The normal max likelihood trick:

  Set $\frac{\partial}{\partial \mu_i}$ log 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

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, 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	$P(B) = \mu$
w_3 = Gets a C	$P(C) = 2\mu$
w_4 = Gets a D	$P(D) = ½-3\mu$

(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

REMEMBER

$P(A) = ½$

$P(B) = \mu$

$P(C) = 2\mu$

$P(D) = ½-3\mu$

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?

Same Problem with Hidden Information

REMEMBER

$P(A) = ½$

$P(B) = \mu$

$P(C) = 2\mu$

$P(D) = ½-3\mu$

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:

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

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

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

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

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

Back to Unsupervised Learning of GMMs – a simple case

Remember:

- We have unlabeled data $\mathbf{x}_1\ \mathbf{x}_2 \ldots \mathbf{x}_m$
- 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 $\boldsymbol{\mu}_1\ \boldsymbol{\mu}_2 .. \boldsymbol{\mu}_k$

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

$$= \mathrm{p}\left(x_1 \ldots x_m \middle| \mu_1 \ldots \mu_k\right)$$

$$= \prod_{j=1}^{m} \mathrm{p}\left(x_j \middle| \mu_1 \ldots \mu_k\right)$$

$$= \prod_{j=1}^{m} \sum_{i=1}^{k} \mathrm{p}\left(x_j \middle| \mu_i\right) \mathrm{P}\left(y = i\right)$$

$$\propto \prod_{j=1}^{m} \sum_{i=1}^{k} \exp\left(-\frac{1}{2\sigma^2}\left\|x_j - \mu_i\right\|^2\right) \mathrm{P}\left(y = i\right)$$

EM for simple case of GMMs: The E-step

- If we know $\mu_1,\ldots,\mu_k$ $\rightarrow$ easily compute prob. point x_j belongs to class y=i

$$\mathrm{p}\left(y = i \middle| x_j, \mu_1 \ldots \mu_k\right) \propto \exp\left(-\frac{1}{2\sigma^2}\left\|x_j - \mu_i\right\|^2\right)\mathrm{P}\left(y = i\right)$$

EM for simple case of GMMs: The M-step

- If we know prob. point x_j belongs to class $y=i$

  $\rightarrow$ MLE for μ_i is weighted average

  - imagine k copies of each x_j, each with weight $P(y=i|x_j)$:

$$\mu_i = \frac{\sum_{j=1}^{m} P(y=i|x_j)\, x_j}{\sum_{j=1}^{m} P(y=i|x_j)}$$

E.M. for GMMs

E-step

Compute “expected” classes of all datapoints for each class

$$\mathrm{p}\left(y=i \middle| x_j, \mu_1 \ldots \mu_k\right) \propto \exp\left(-\frac{1}{2\sigma^2}\left\|x_j - \mu_i\right\|^2\right) \mathrm{P}(y=i)$$

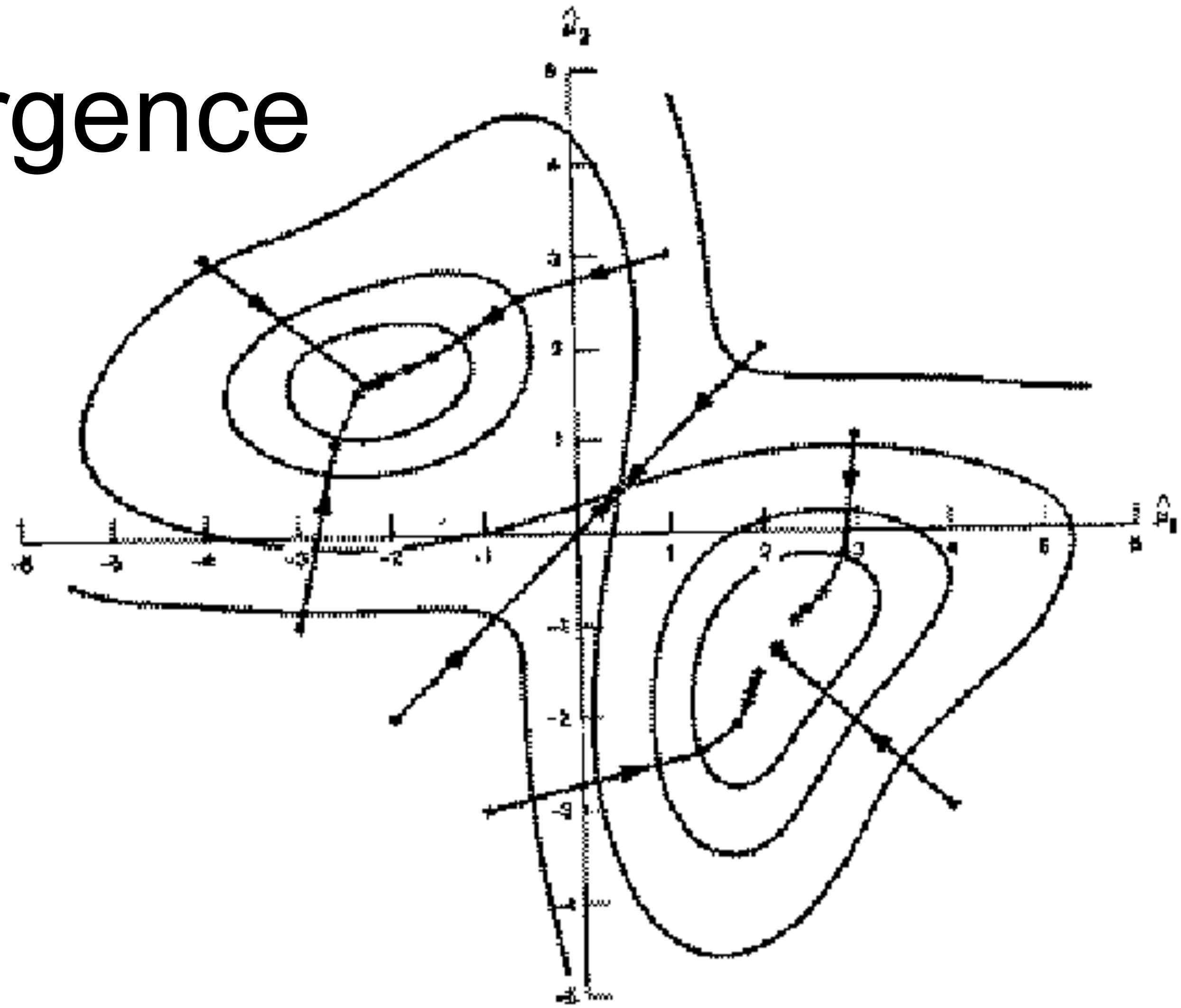
Just evaluate a Gaussian at x_j

M-step

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

$$\mu_i = \frac{\sum_{j=1}^{m} P\left(y=i \middle| x_j\right) x_j}{\sum_{j=1}^{m} P\left(y=i \middle| x_j\right)}$$

E.M. Convergence



- EM is coordinate ascent on an interesting potential function
- Coord. ascent for bounded pot. func. ! convergence to a local optimum guaranteed
- See Neal & Hinton reading on class webpage

- 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

$$\lambda_t = \{ \mu_1^{(t)}, \mu_2^{(t)} \ldots \mu_k^{(t)}, \Sigma_1^{(t)}, \Sigma_2^{(t)} \ldots \Sigma_k^{(t)}, p_1^{(t)}, p_2^{(t)} \ldots p_k^{(t)} \}$$

$p_i^{(t)}$ is shorthand for estimate of $P(y=i)$ on t'th iteration

E-step

Compute "expected" classes of all datapoints for each class

$$P\left(y = i \middle| x_j, \lambda_t\right) \propto p_i^{(t)} p\left(x_j \middle| \mu_i^{(t)}, \Sigma_i^{(t)}\right)$$

Just evaluate a Gaussian at x_j

M-step

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

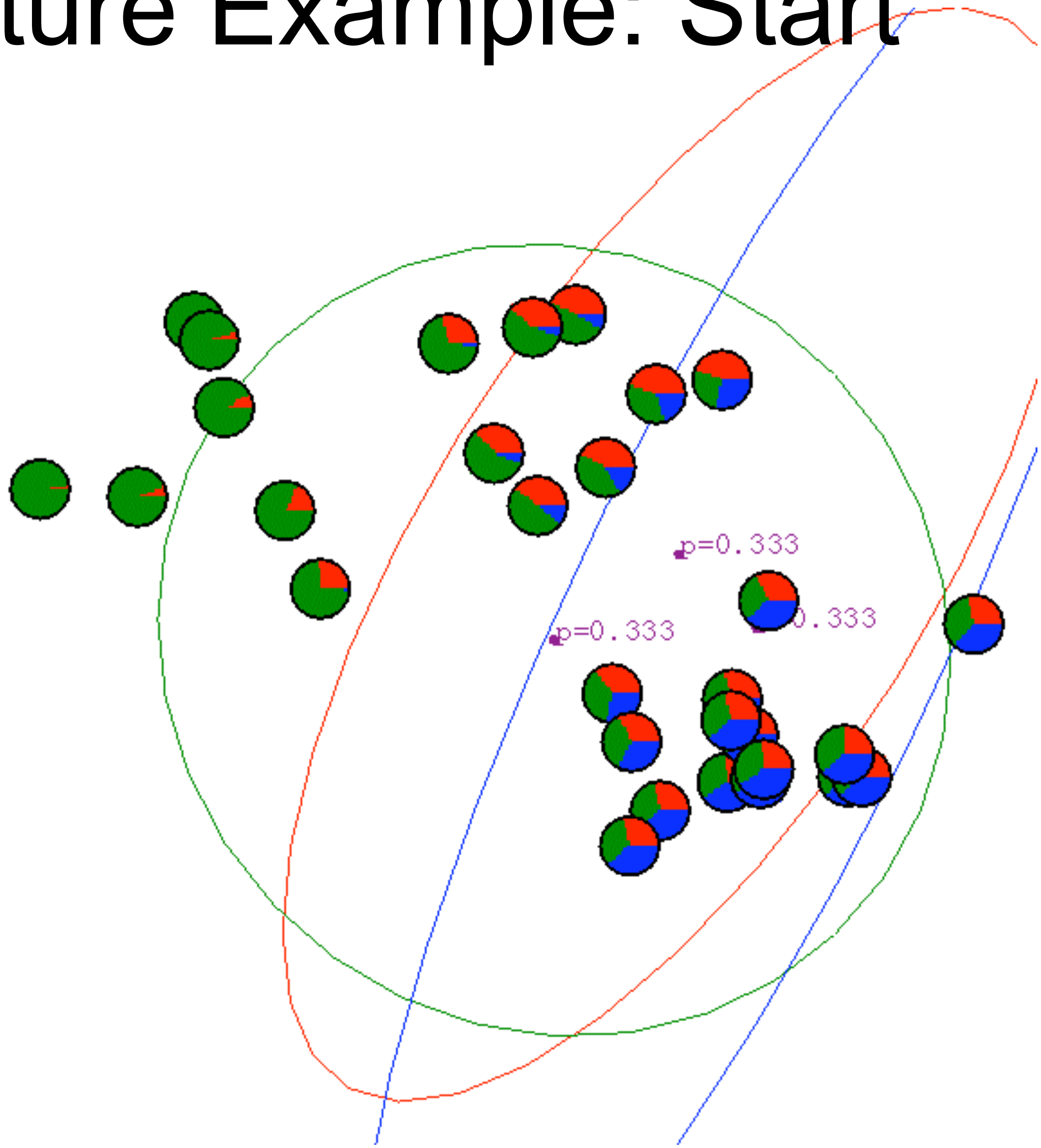
$$\mu_i^{(t+1)} = \frac{\sum_j P\left(y = i \middle| x_j, \lambda_t\right) x_j}{\sum_j P\left(y = i \middle| x_j, \lambda_t\right)}$$

$$\Sigma_i^{(t+1)} = \frac{\sum_j P\left(y = i \middle| x_j, \lambda_t\right) \left[x_j - \mu_i^{(t+1)}\right]\left[x_j - \mu_i^{(t+1)}\right]^T}{\sum_j P\left(y = i \middle| x_j, \lambda_t\right)}$$

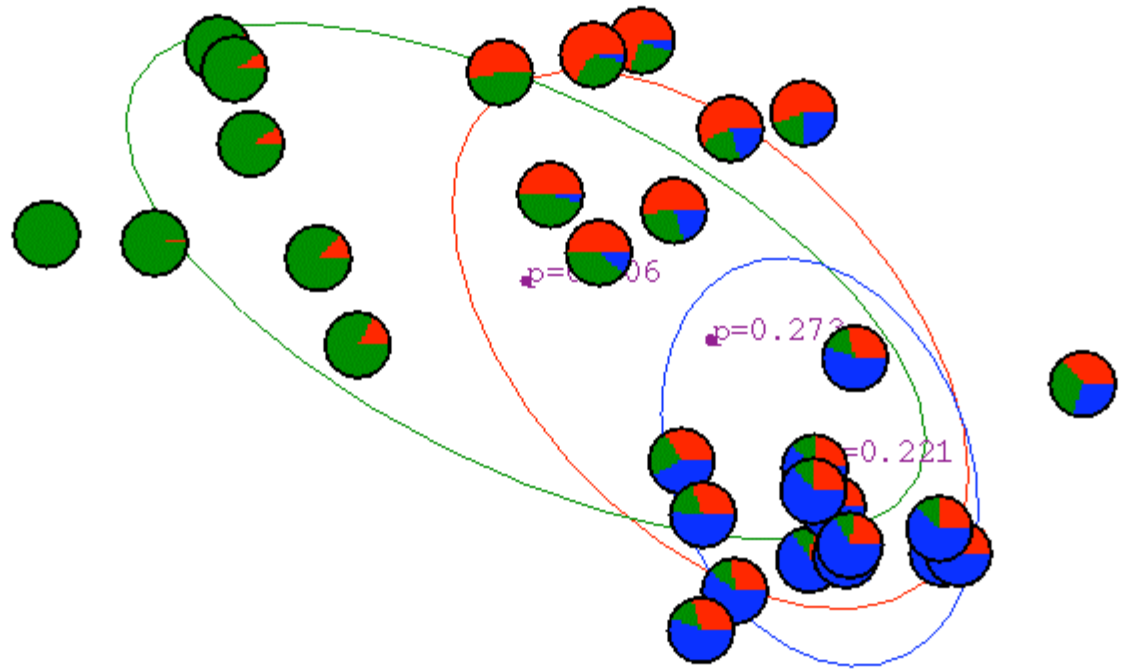
$$p_i^{(t+1)} = \frac{\sum_j P\left(y = i \middle| x_j, \lambda_t\right)}{m}$$

m = #records

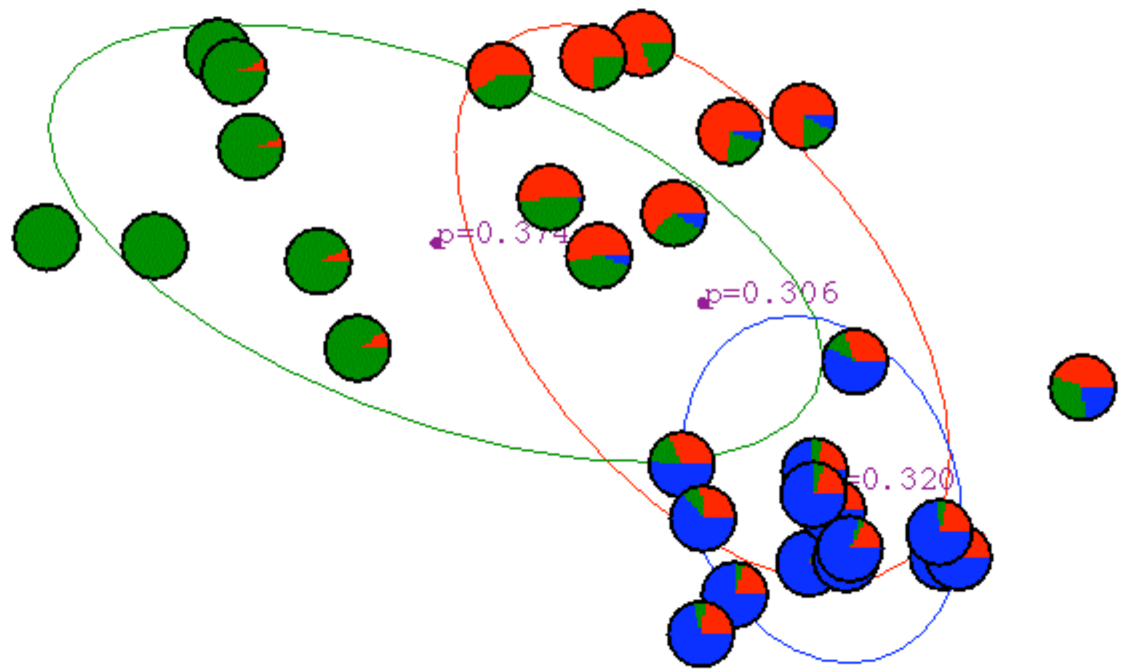
Gaussian Mixture Example: Start



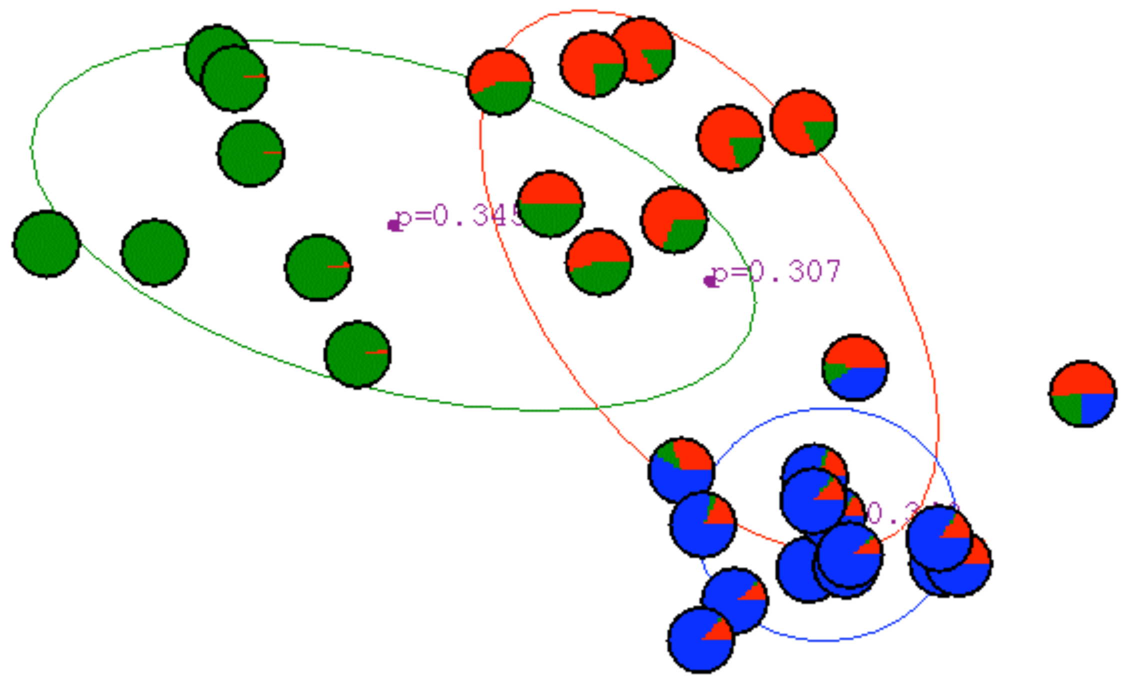
After first iteration



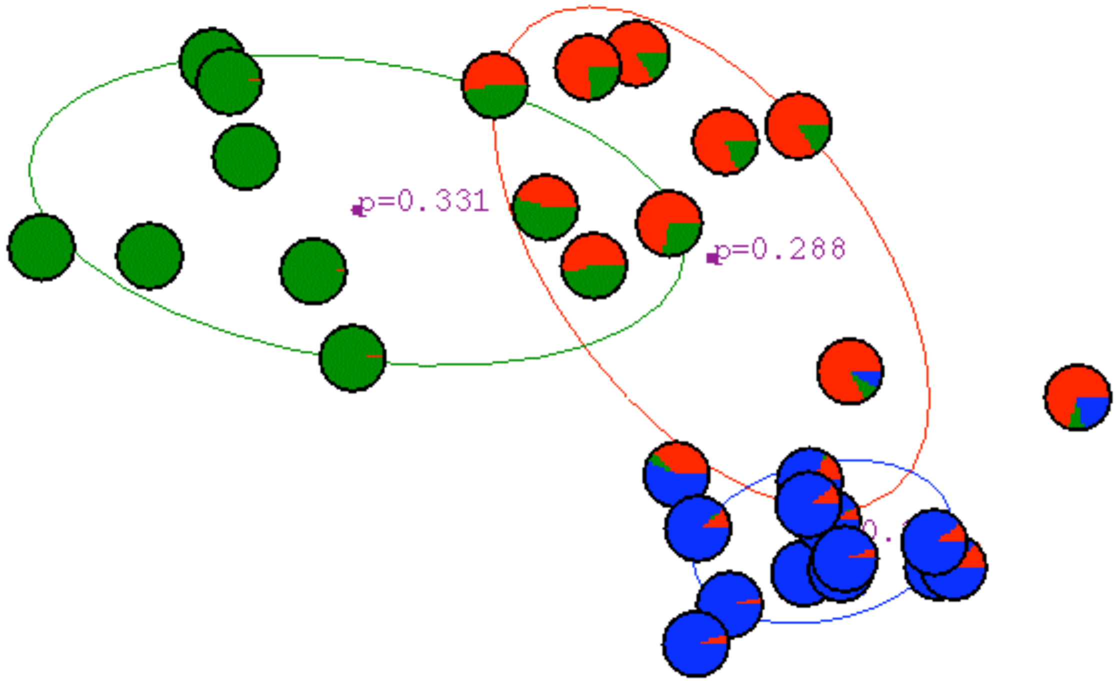
After 2nd iteration



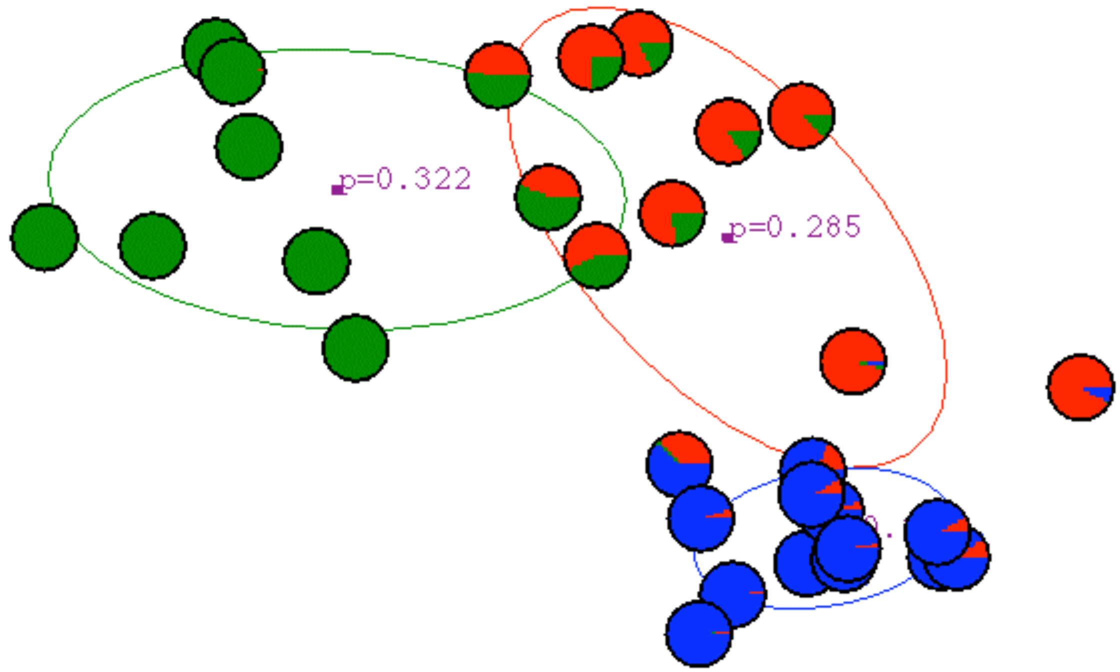
After 3rd iteration



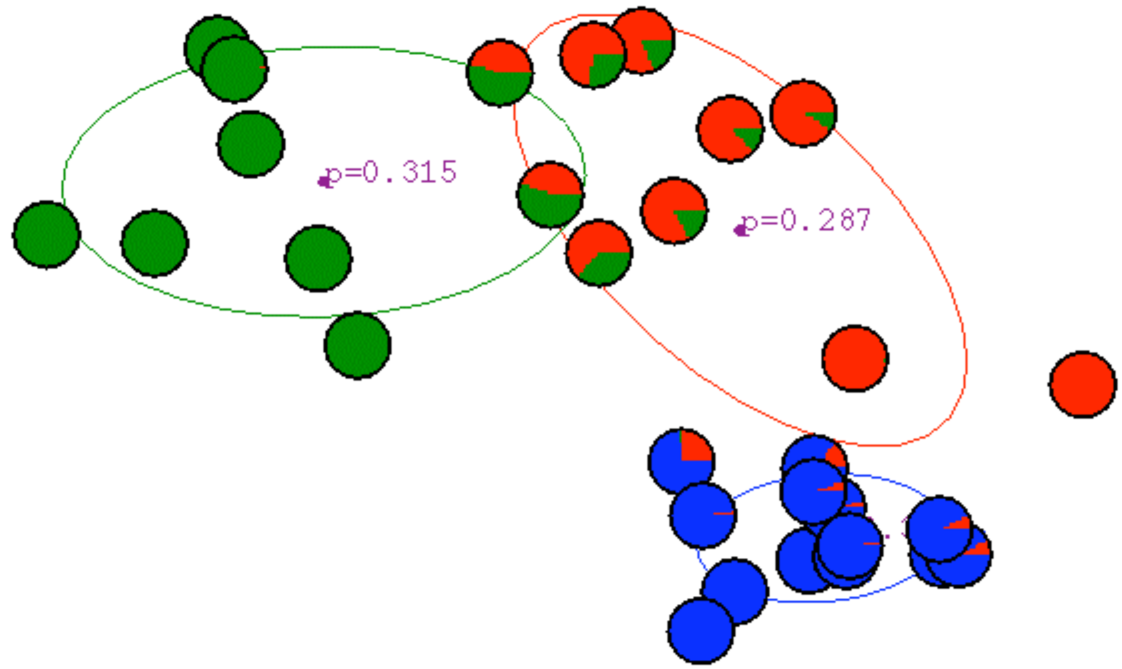
After 4th iteration



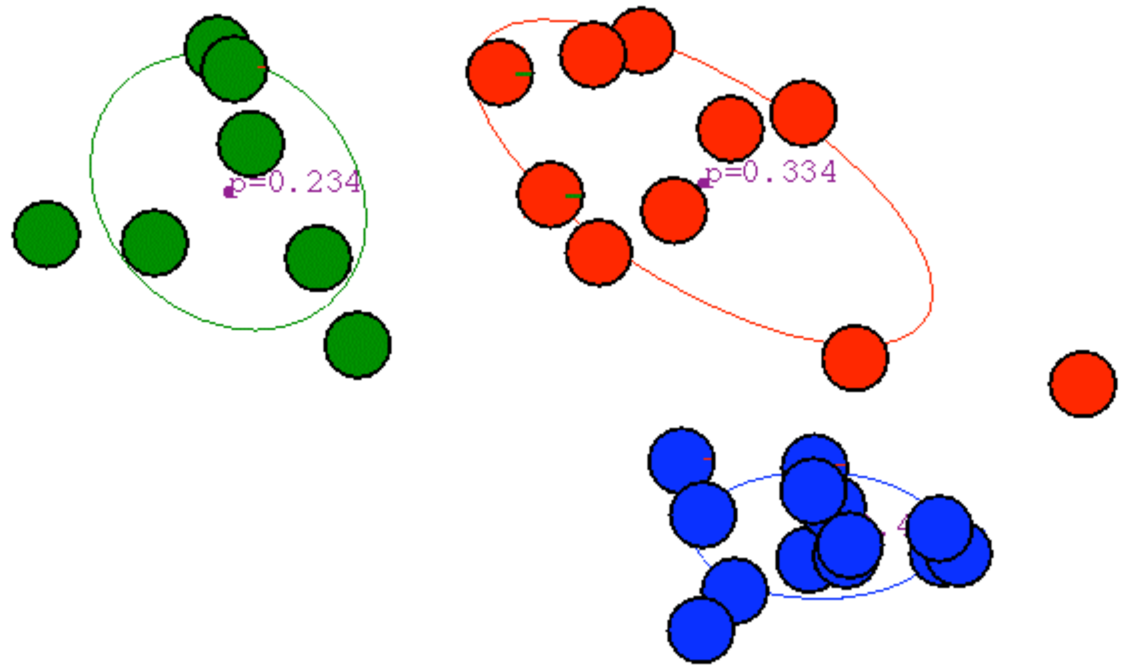
After 5th iteration



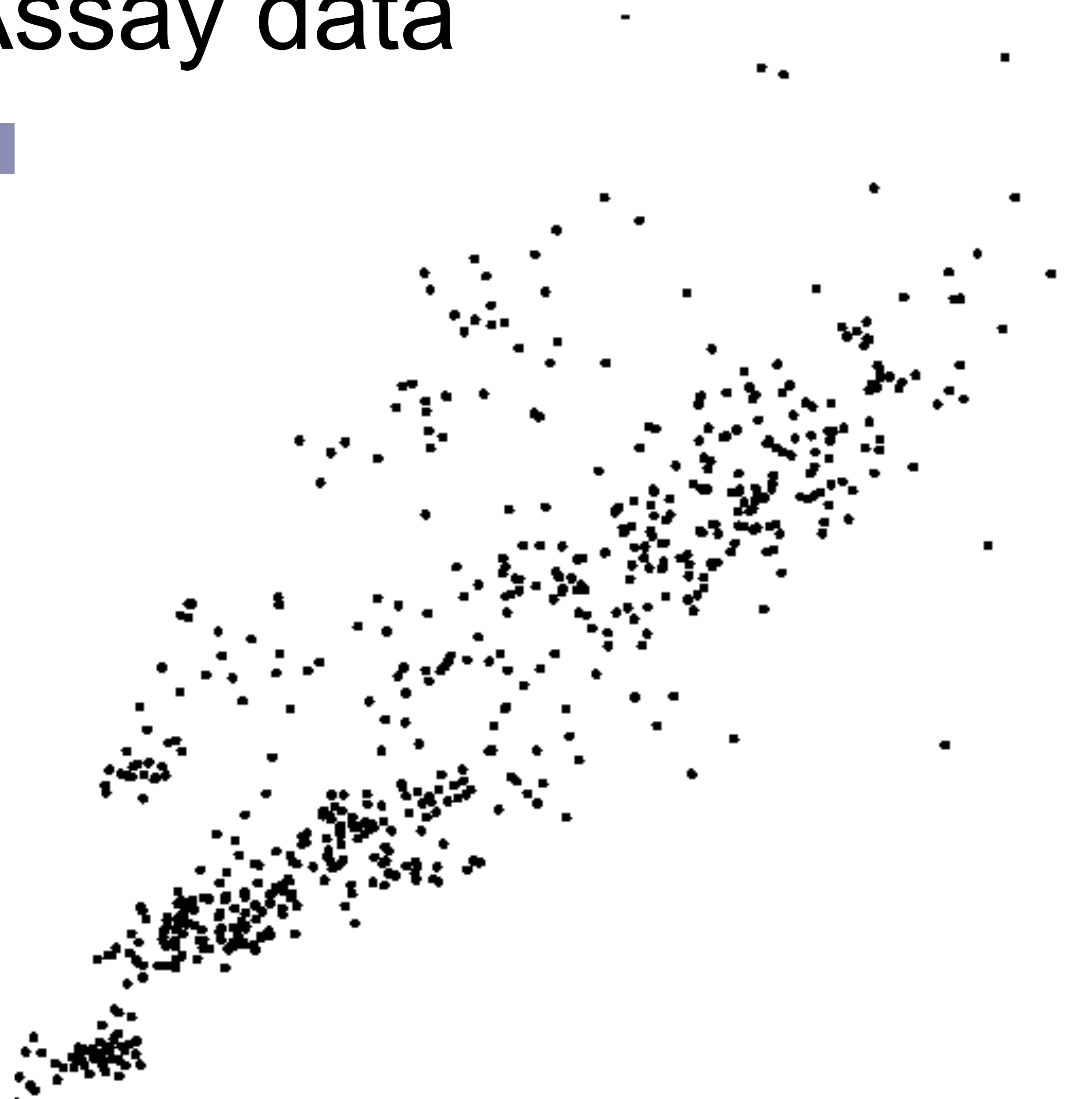
After 6th iteration



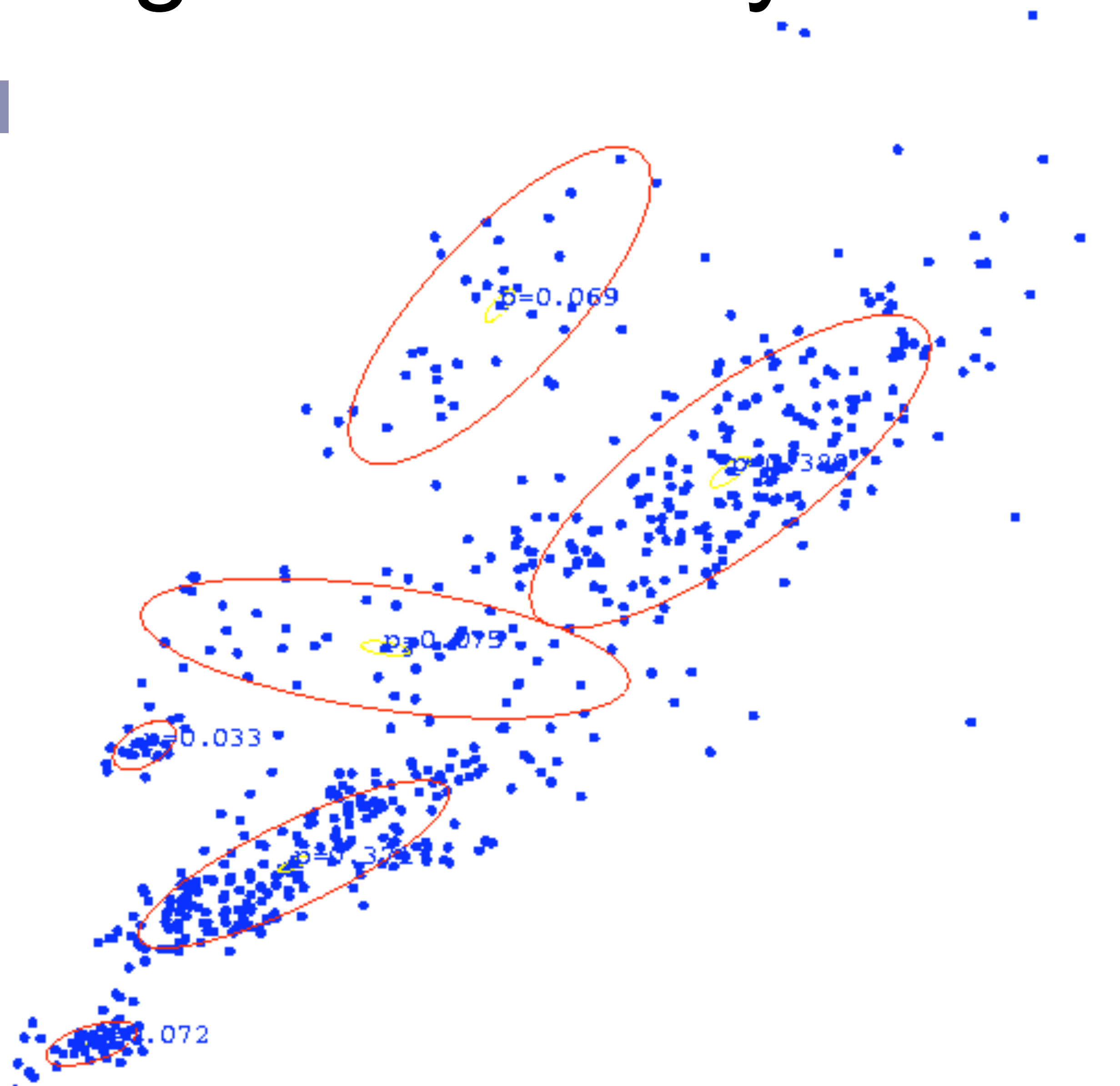
After 20th iteration



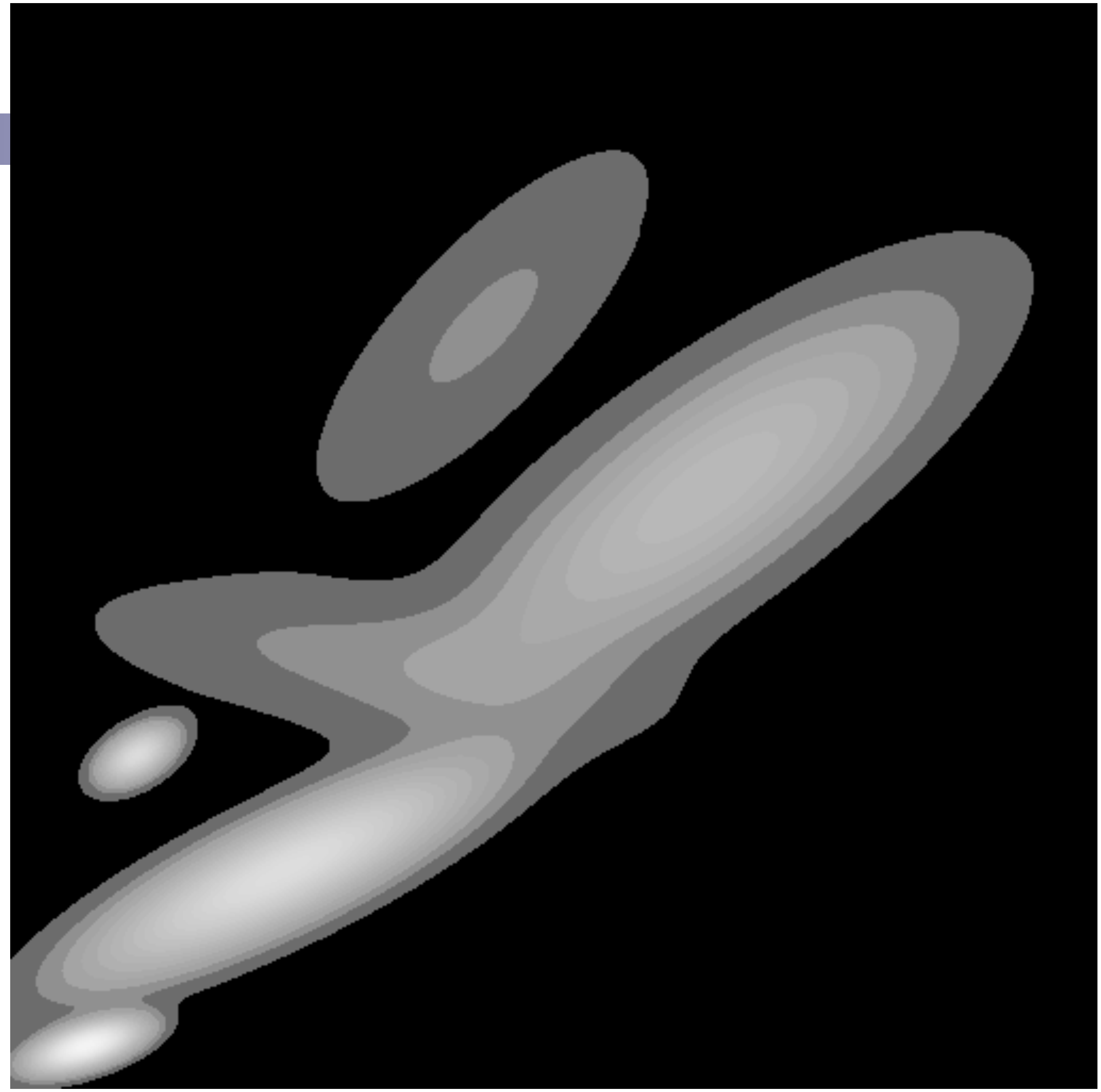
Some Bio Assay data



GMM clustering of the assay data

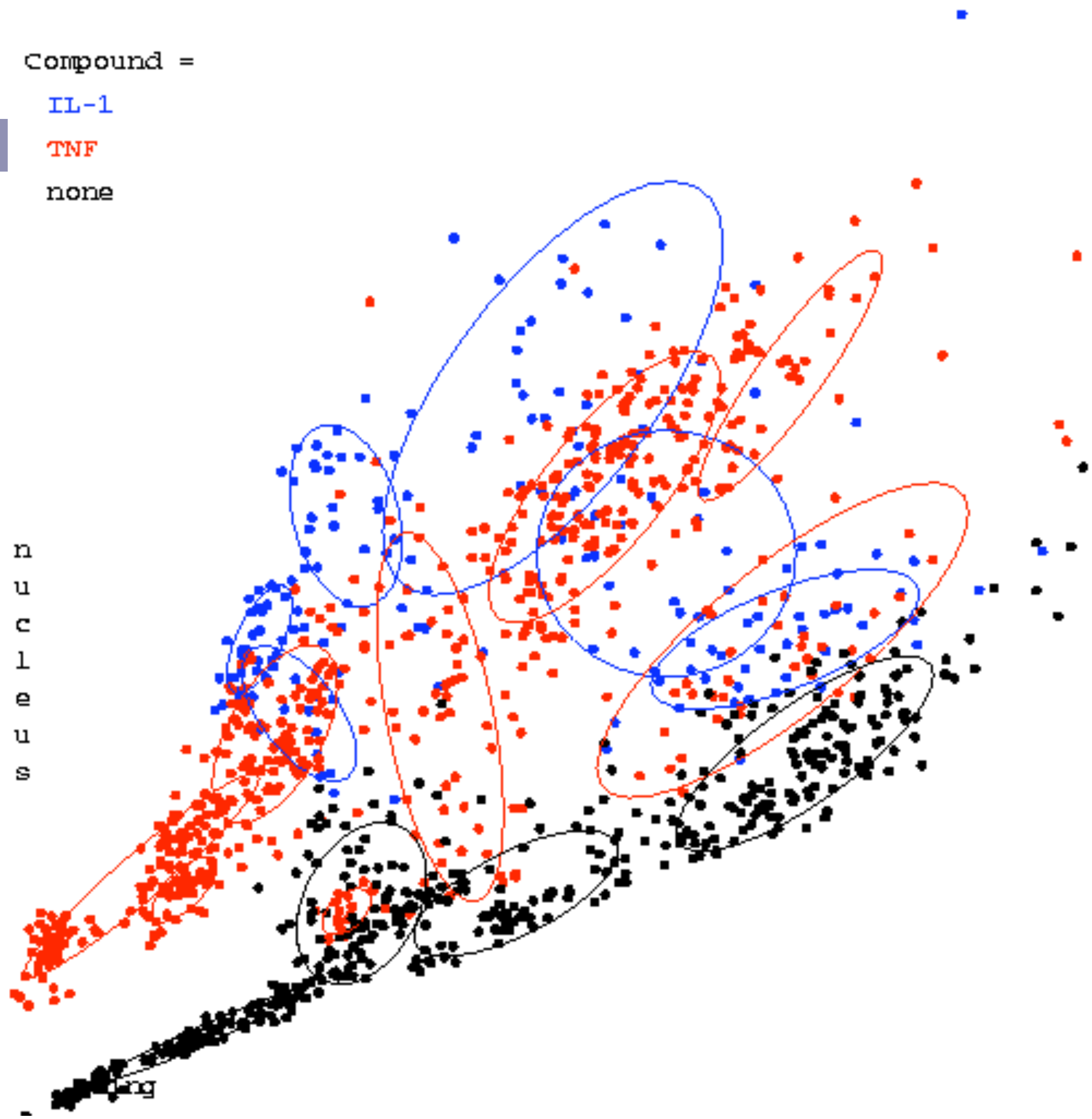


Resulting Density Estimator

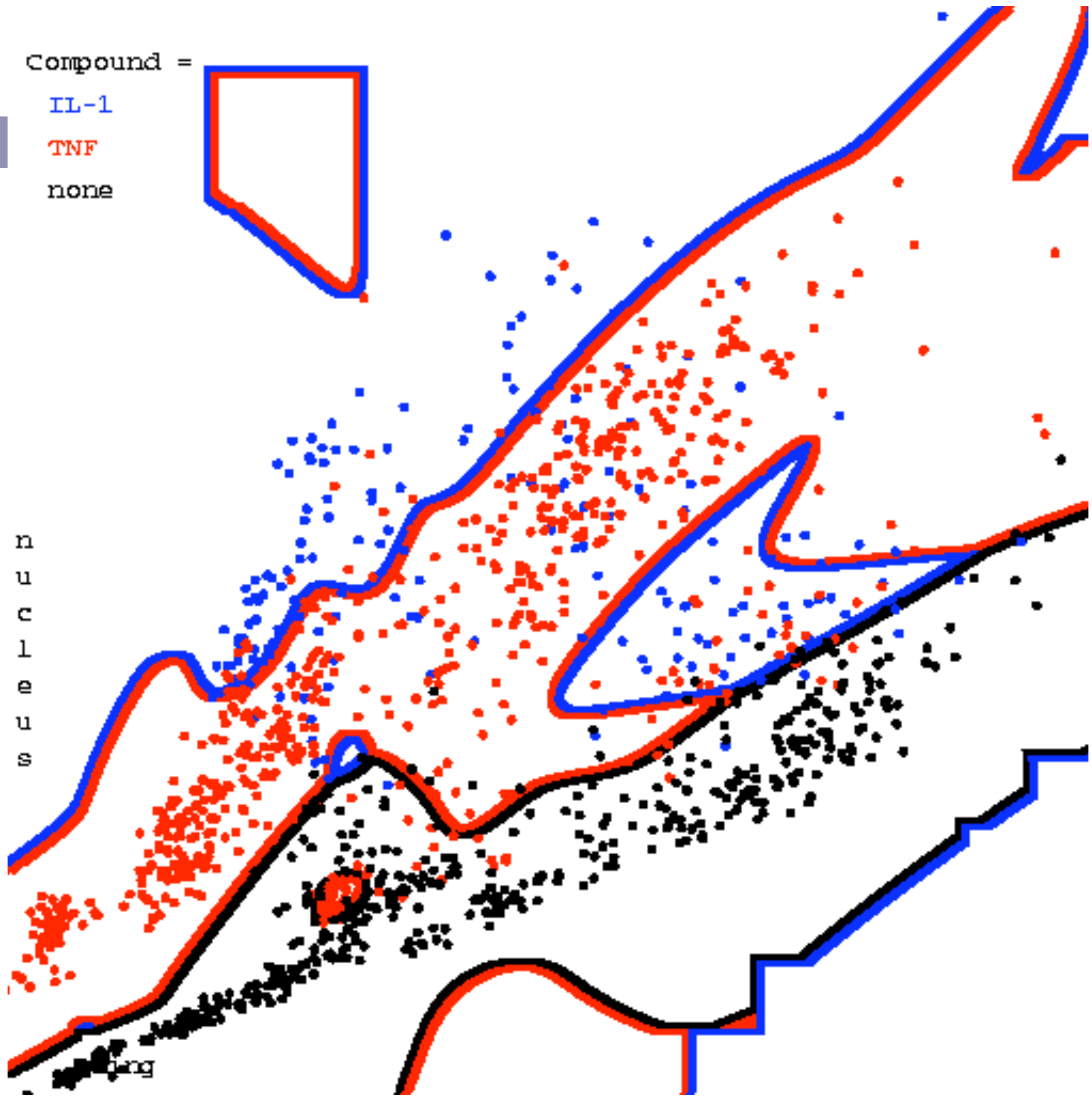


Three classes of assay

(each learned with it's own mixture model)



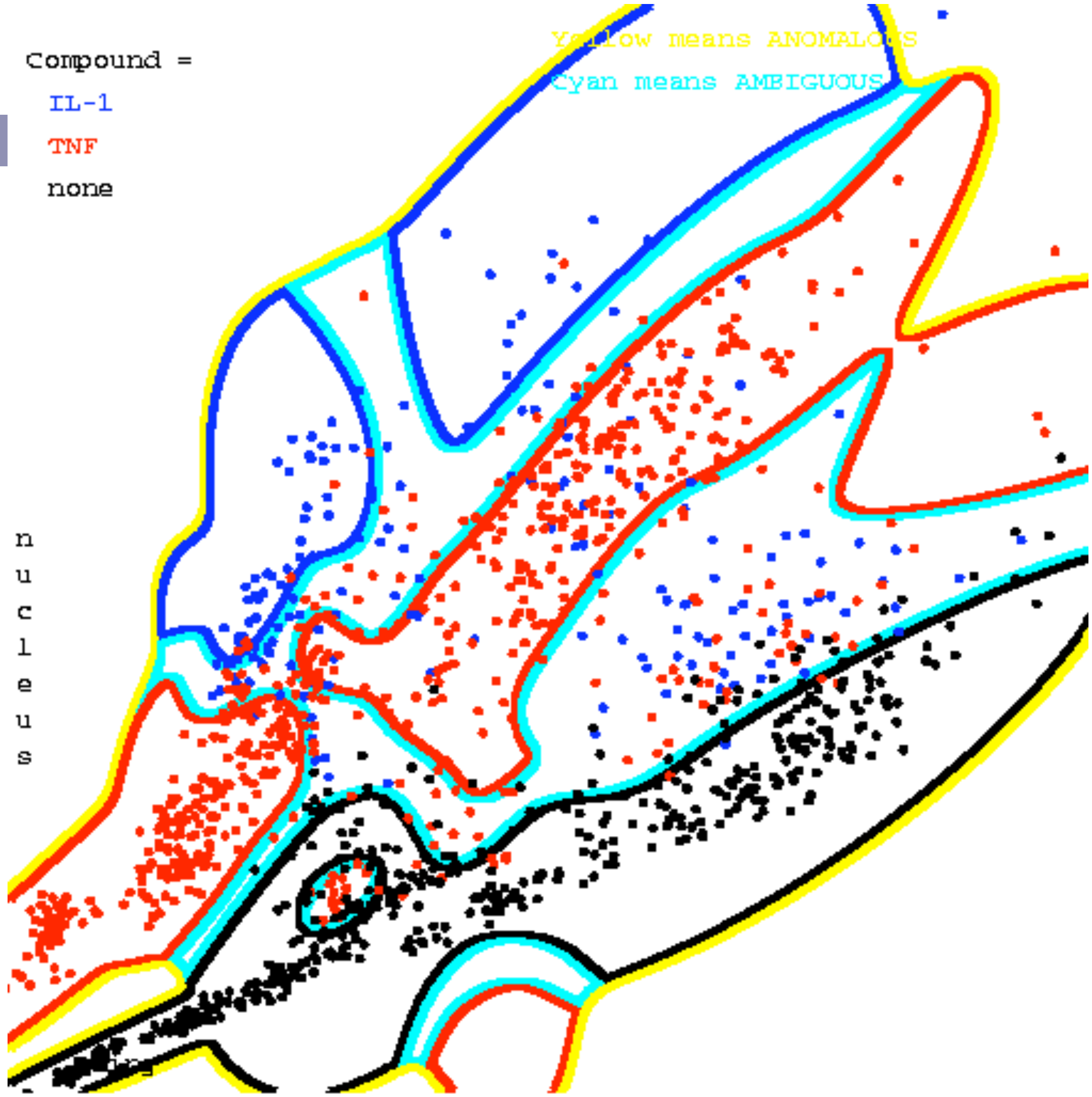
Resulting Bayes Classifier



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

Yellow means anomalous

Cyan means ambiguous



What you should know

- K-means for clustering:
  - algorithm
  - converges because it's coordinate ascent
- EM for mixture of Gaussians:
  - 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
- Remember, E.M. can get stuck in local minima, and empirically it <u>DOES</u>

Acknowledgements

- K-means & Gaussian mixture models presentation contains material 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