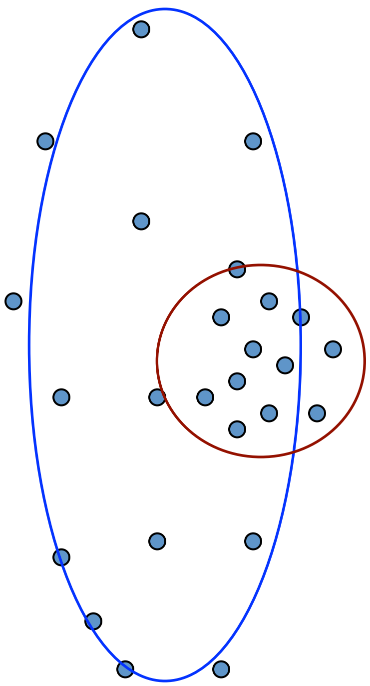
Mixture Models & EM algorithm
Lecture 21

David Sontag

New York University

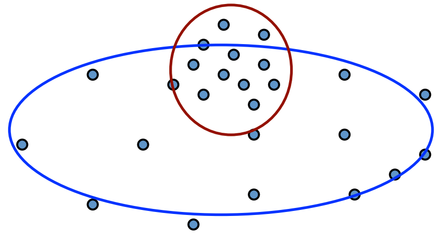
Slides adapted from Carlos Guestrin, Dan Klein, Luke Zettlemoyer, Dan Weld, Vibhav Gogate, and Andrew Moore

The Evils of "**Hard Assignments**"?



- Clusters may overlap
- Some clusters may be "wider" than others
- Distances can be deceiving!

Probabilistic Clustering



- Try a probabilistic model!
  - allows overlaps, clusters of different size, etc.
- Can tell a *generative story* for data
  - P(X|Y) P(Y)
- Challenge: we need to estimate model parameters without labeled Ys

Y	X_1	X_2
??	0.1	2.1
??	0.5	-1.1
??	0.0	3.0
??	-0.1	-2.0
??	0.2	1.5
…	…	…

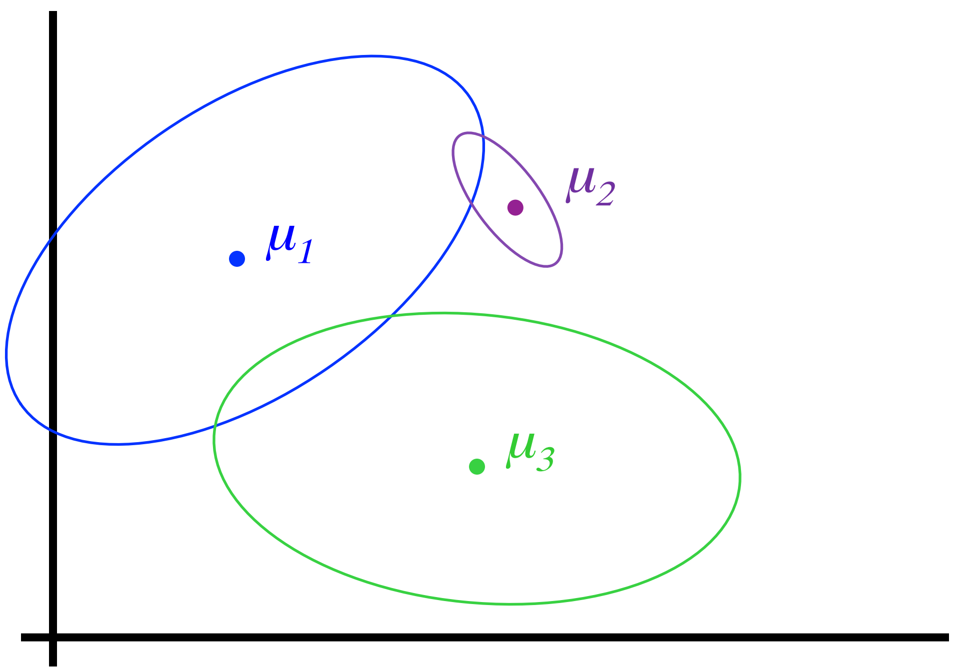
The General GMM assumption

- P(Y): There are k components
- P(X|Y): Each component generates data from a **multivariate Gaussian** with mean μ_i and covariance matrix Σ_i

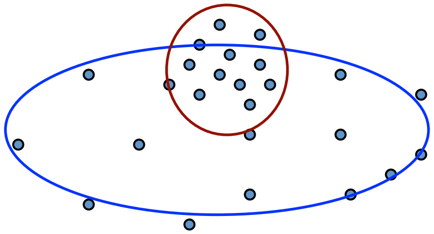
Each data point is sampled from a ***generative process*:**

1. Choose component i with probability *P(y=i)*
2. Generate datapoint ~ N(m_i, Σ_i)

Gaussian mixture model (GMM)



What Model Should We Use?



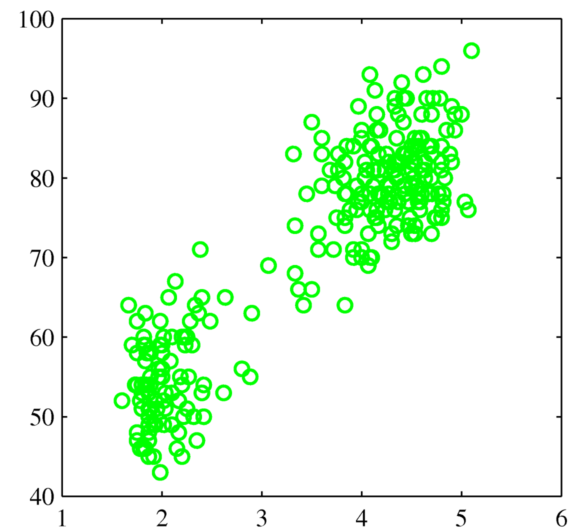
- Depends on X!
- Here, maybe Gaussian Naïve Bayes?
  - Multinomial over clusters Y
  - (Independent) Gaussian for each X_i given Y

Y	X_1	X_2
??	0.1	2.1
??	0.5	-1.1
??	0.0	3.0
??	-0.1	-2.0
??	0.2	1.5
	…	…

$$p(Y_i = y_k) = \theta_k$$

$$P(X_i = x \mid Y = y_k) = \frac{1}{\sigma_{ik}\sqrt{2\pi}} \; e^{\frac{-(x-\mu_{ik})^2}{2\sigma_{ik}^2}}$$

Could we make fewer assumptions?

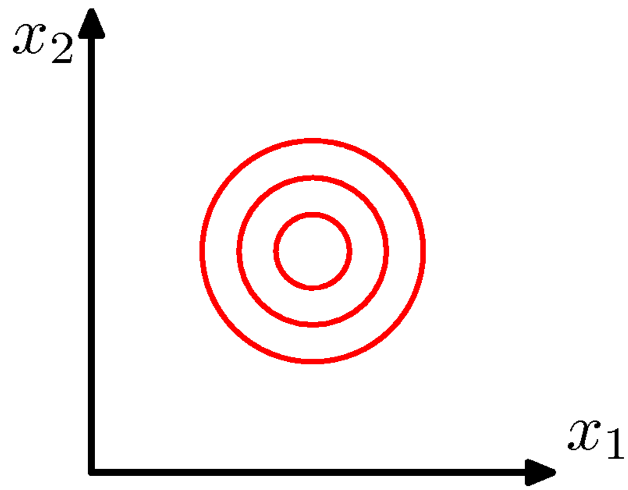


- What if the X_i co-vary?
- What if there are multiple peaks?
- Gaussian Mixture Models!
  - P(Y) still multinomial
  - P(**X**|Y) is a multivariate Gaussian distribution:

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

Multivariate Gaussians

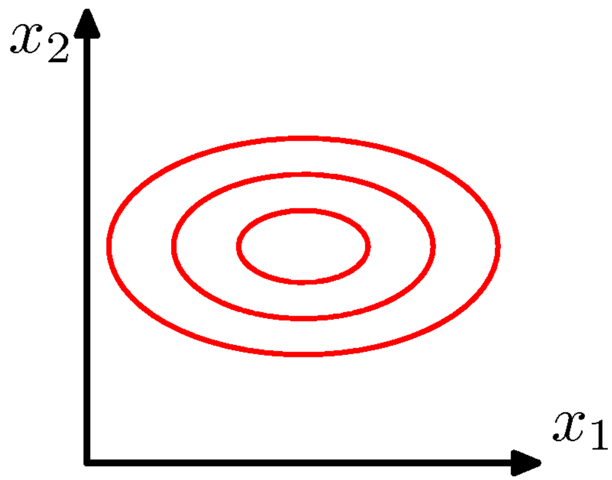
$$P(X=\boldsymbol{x}_j)= \frac{1}{(2\pi)^{m/2} \| \Sigma \|^{1/2}} \exp\left[-\frac{1}{2}\left(\mathbf{x}_j-\mu\right)^T \Sigma^{-1}\left(\mathbf{x}_j-\mu\right)\right]$$



$\Sigma \propto$ identity matrix

Multivariate Gaussians

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

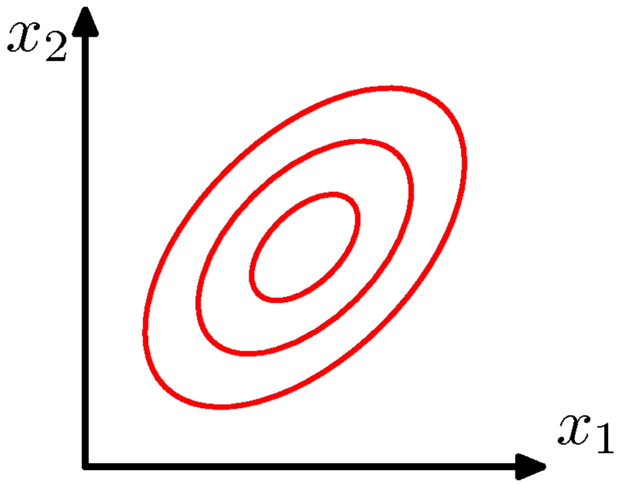


Σ = diagonal matrix

X_i are independent *ala* Gaussian NB

Multivariate Gaussians

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

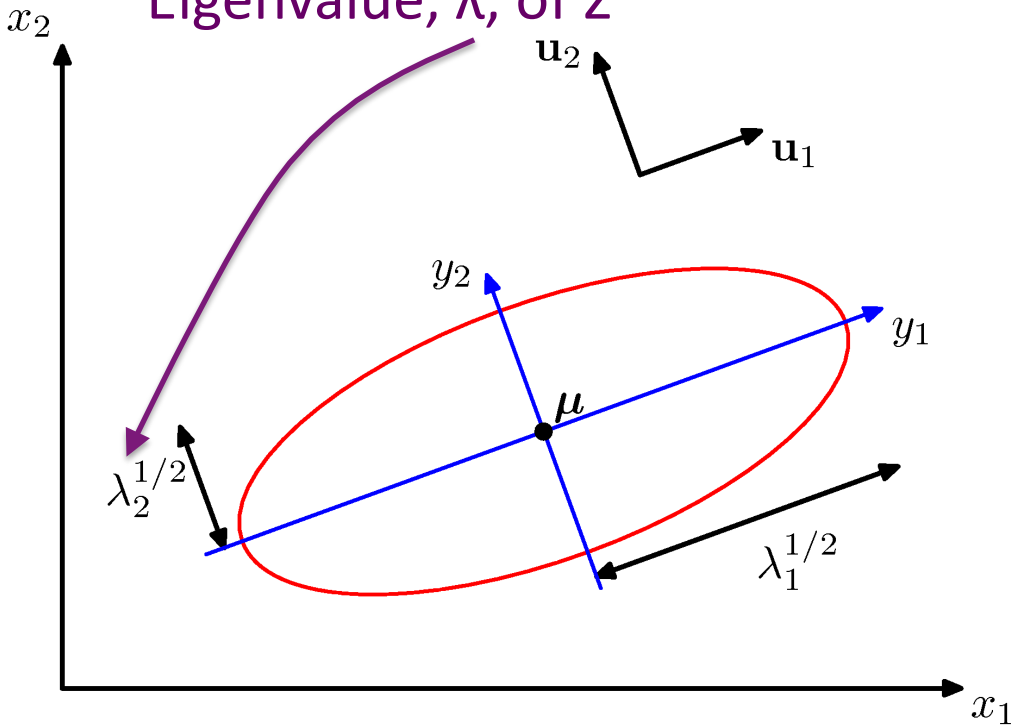


Σ = arbitrary (semidefinite) matrix:

- specifies rotation (change of basis)
- eigenvalues specify relative elongation

Multivariate Gaussians

Eigenvalue, λ, of Σ

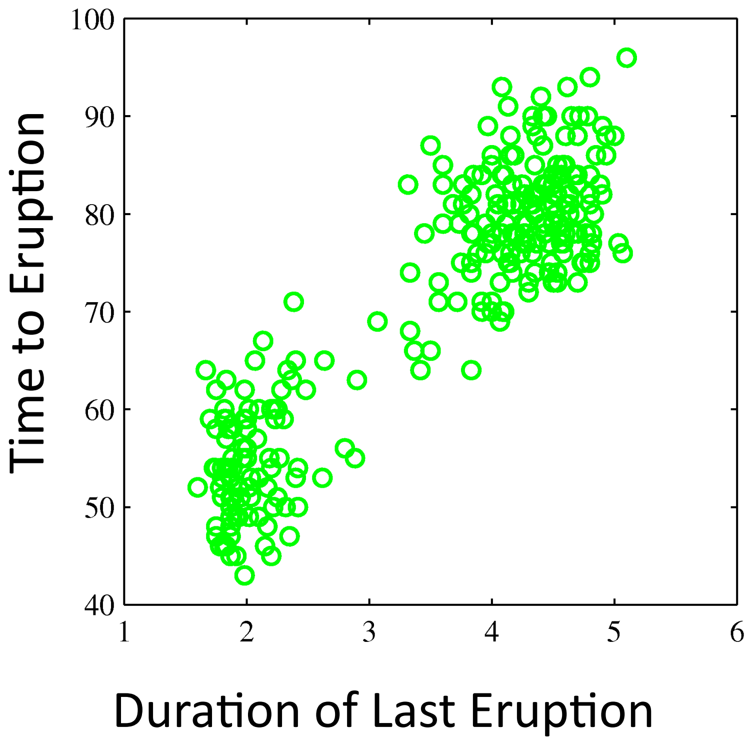


Covariance matrix, Σ, = degree to which x_i vary together

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

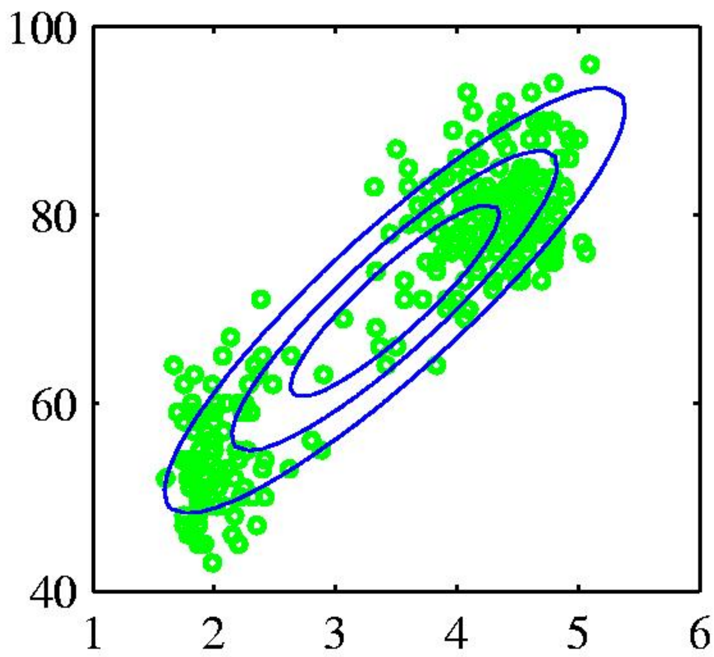
Mixtures of Gaussians (1)

Old Faithful Data Set

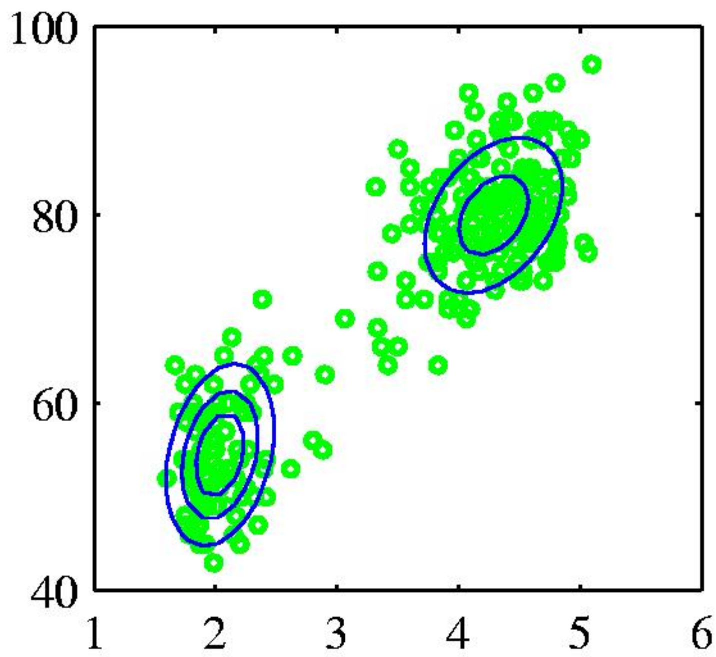


Mixtures of Gaussians (1)

Old Faithful Data Set



Single Gaussian



Mixture of two Gaussians

Mixtures of Gaussians (2)

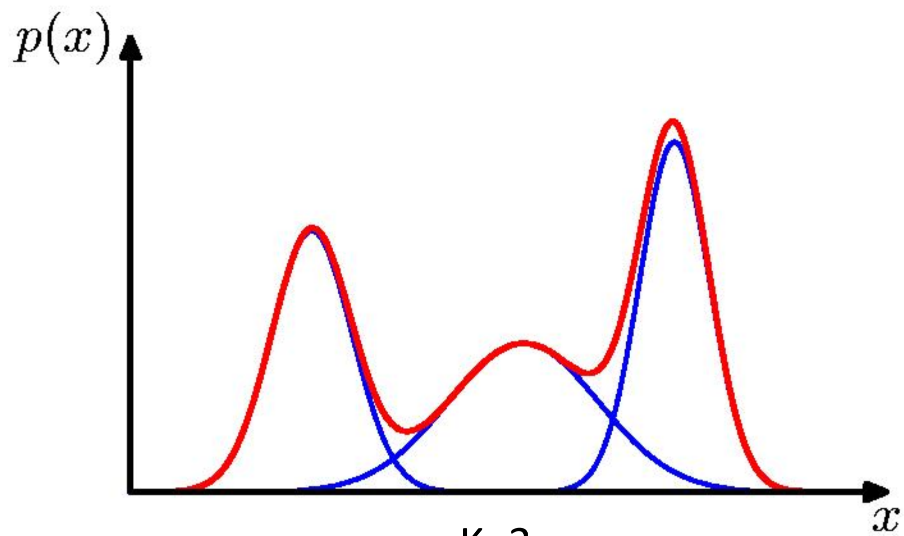
Combine simple models into a complex model:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Component

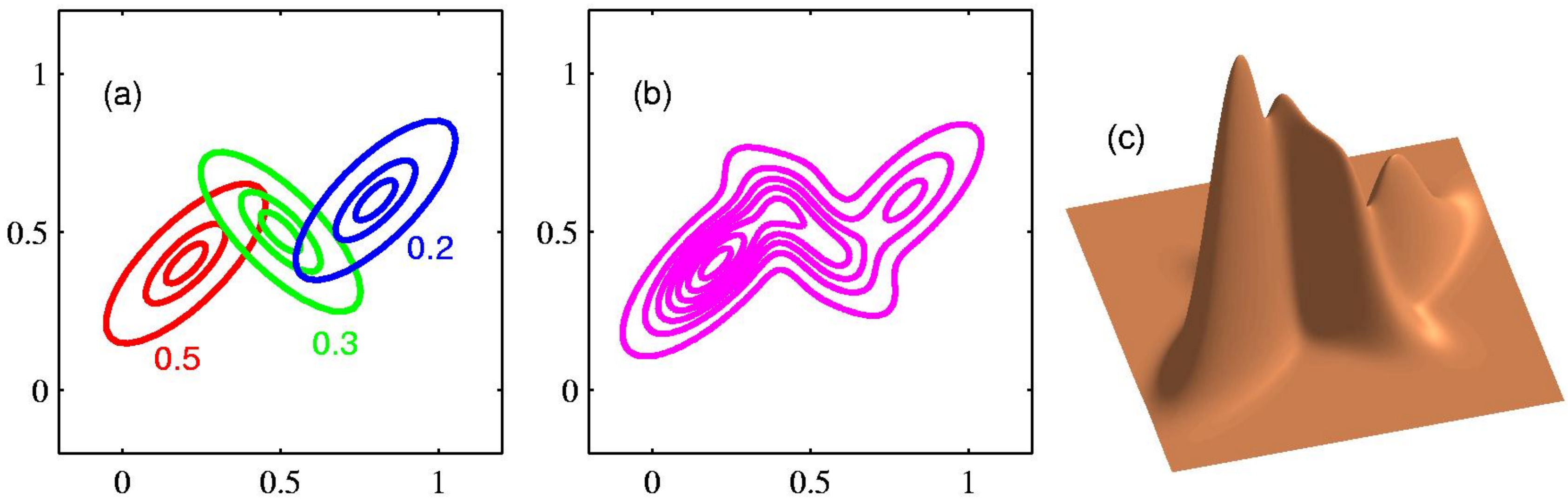
Mixing coefficient

$$\forall k : \pi_k \geqslant 0 \qquad \sum_{k=1}^{K} \pi_k = 1$$



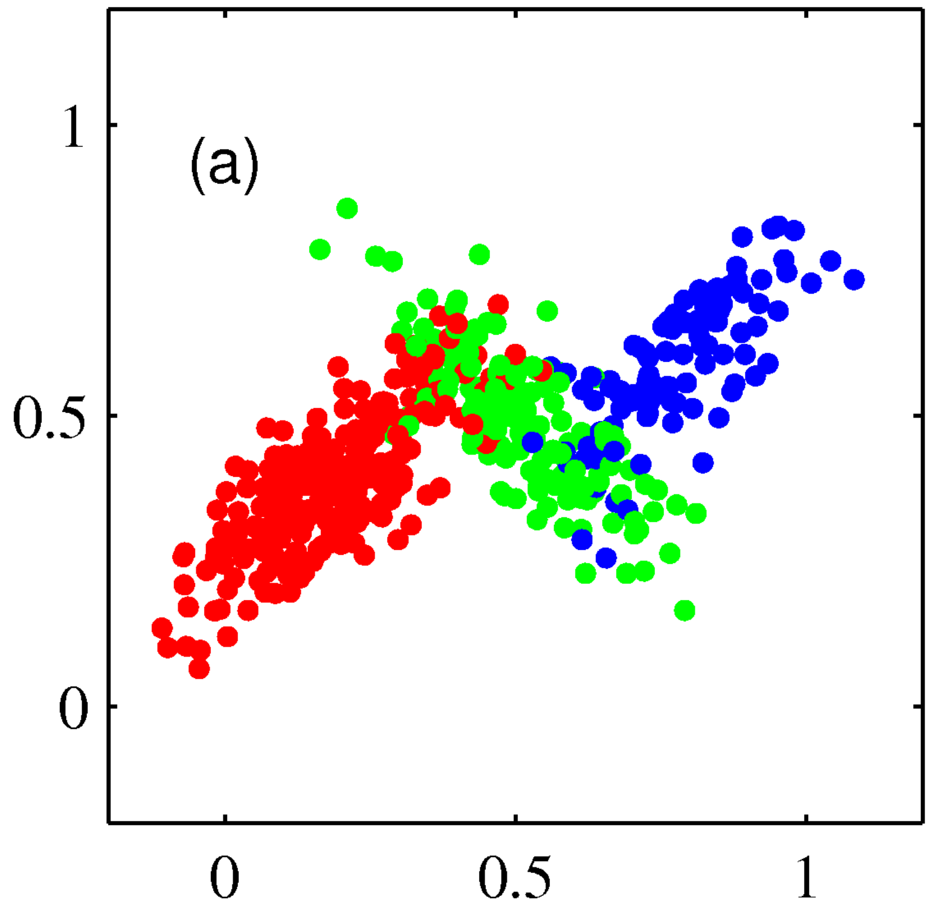
K=3

Mixtures of Gaussians (3)



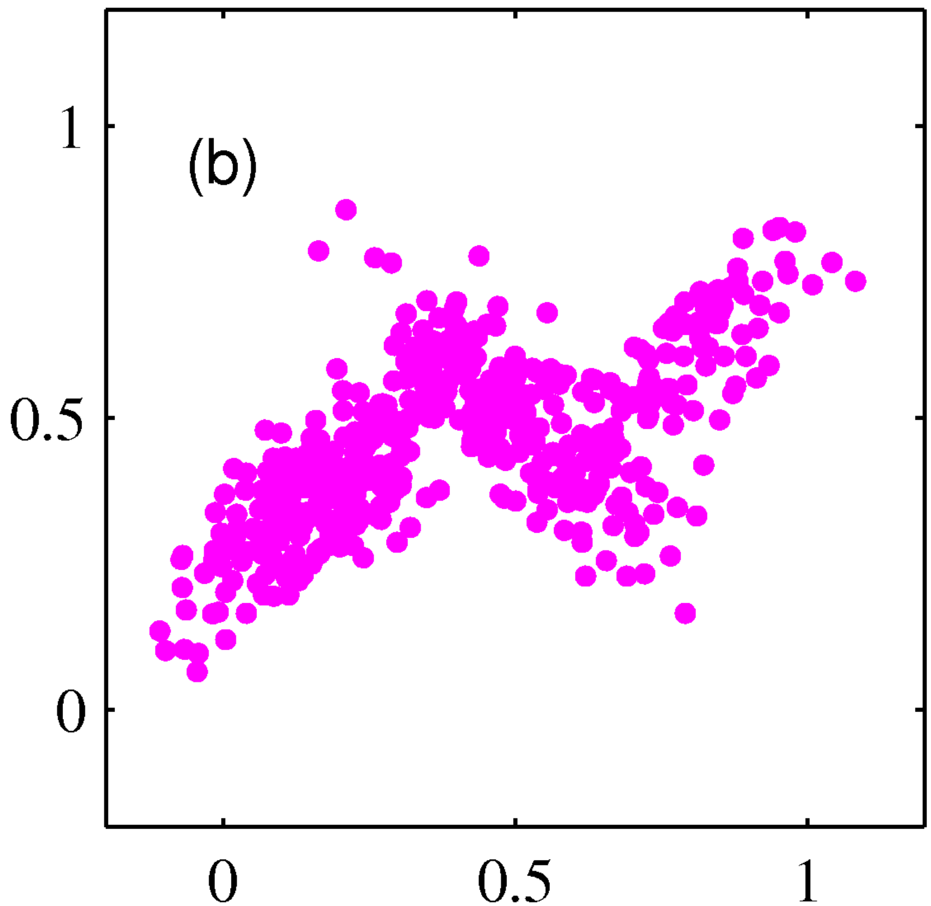
Eliminating Hard Assignments to Clusters

Model data as mixture of multivariate Gaussians



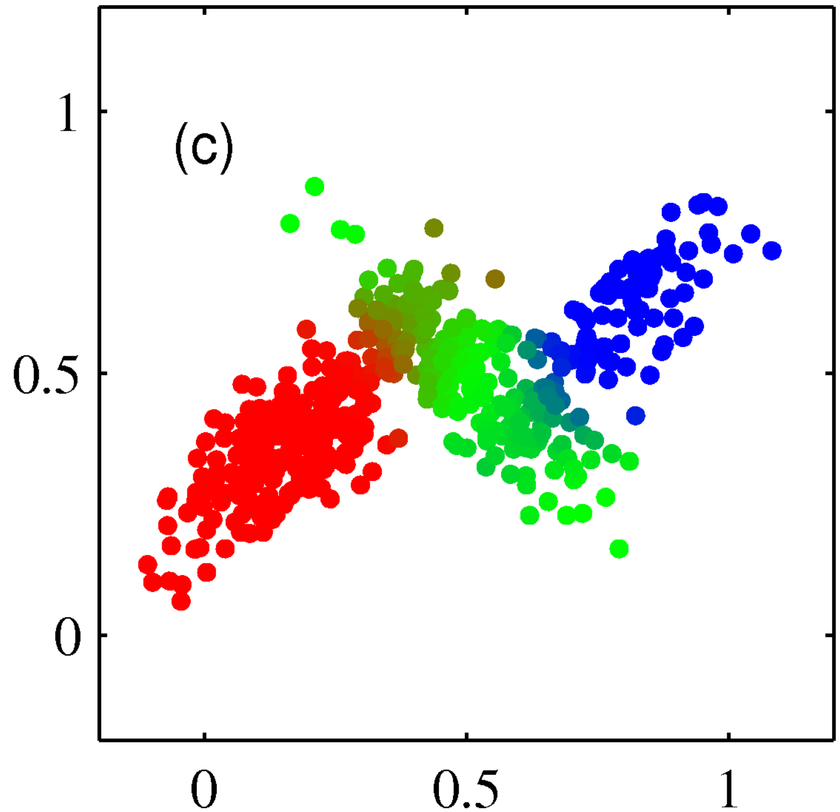
Eliminating Hard Assignments to Clusters

Model data as mixture of multivariate Gaussians



Eliminating Hard Assignments to Clusters

Model data as mixture of multivariate Gaussians



Shown is the *posterior probability* that a point was generated from ith Gaussian: $\Pr(Y = i \mid x)$

ML estimation in **supervised** setting

- Univariate Gaussian

$$\mu_{MLE} = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad \sigma^2_{MLE} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \widehat{\mu})^2$$

- ***Mixture*** of ***Multi***variate Gaussians

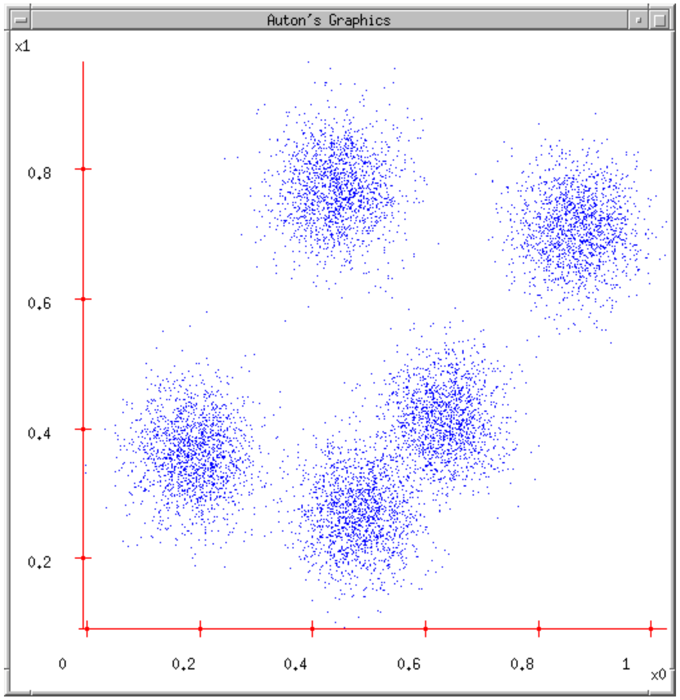
ML estimate for each of the Multivariate Gaussians is given by:

$$\mu_{ML}^{k} = \frac{1}{n}\sum_{j=1}^{n} x_n \qquad \Sigma_{ML}^{k} = \frac{1}{n}\sum_{j=1}^{n}\left(\mathbf{x}_j - \mu_{ML}^{k}\right)\left(\mathbf{x}_j - \mu_{ML}^{k}\right)^T$$

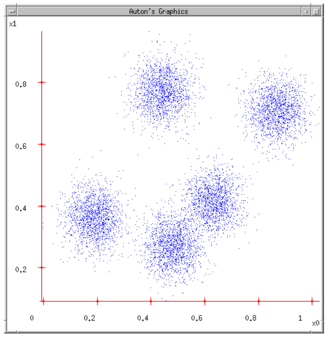
Just sums over *x* generated from the *k*'th Gaussian

That was easy! But what if ***unobserved data***?

- MLE:
  - $\arg\max_\theta \prod_j P(y_j, x_j)$
  - θ: all model parameters
    - eg, class probs, means, and variances
- But we don't know y_j's!!!
- Maximize ***marginal likelihood***:
  - $\arg\max_\theta \prod_j P(x_j) = \arg\max \prod_j \sum_{k=1}^{K} P(Y_j=k, x_j)$



How do we optimize? Closed Form?



- Maximize ***marginal likelihood***:
  - $\text{argmax}_\theta \prod_j P(x_j) = \text{argmax} \prod_j \sum_{k=1}^{K} P(Y_j{=}k, x_j)$
- Almost always a hard problem!
  - Usually no closed form solution
  - Even when lgP(X,Y) is convex, lgP(X) generally isn't…
  - For all but the simplest P(X), we will have to do gradient ascent, in a big messy space with lots of local optimum…

Learning general mixtures of Gaussian

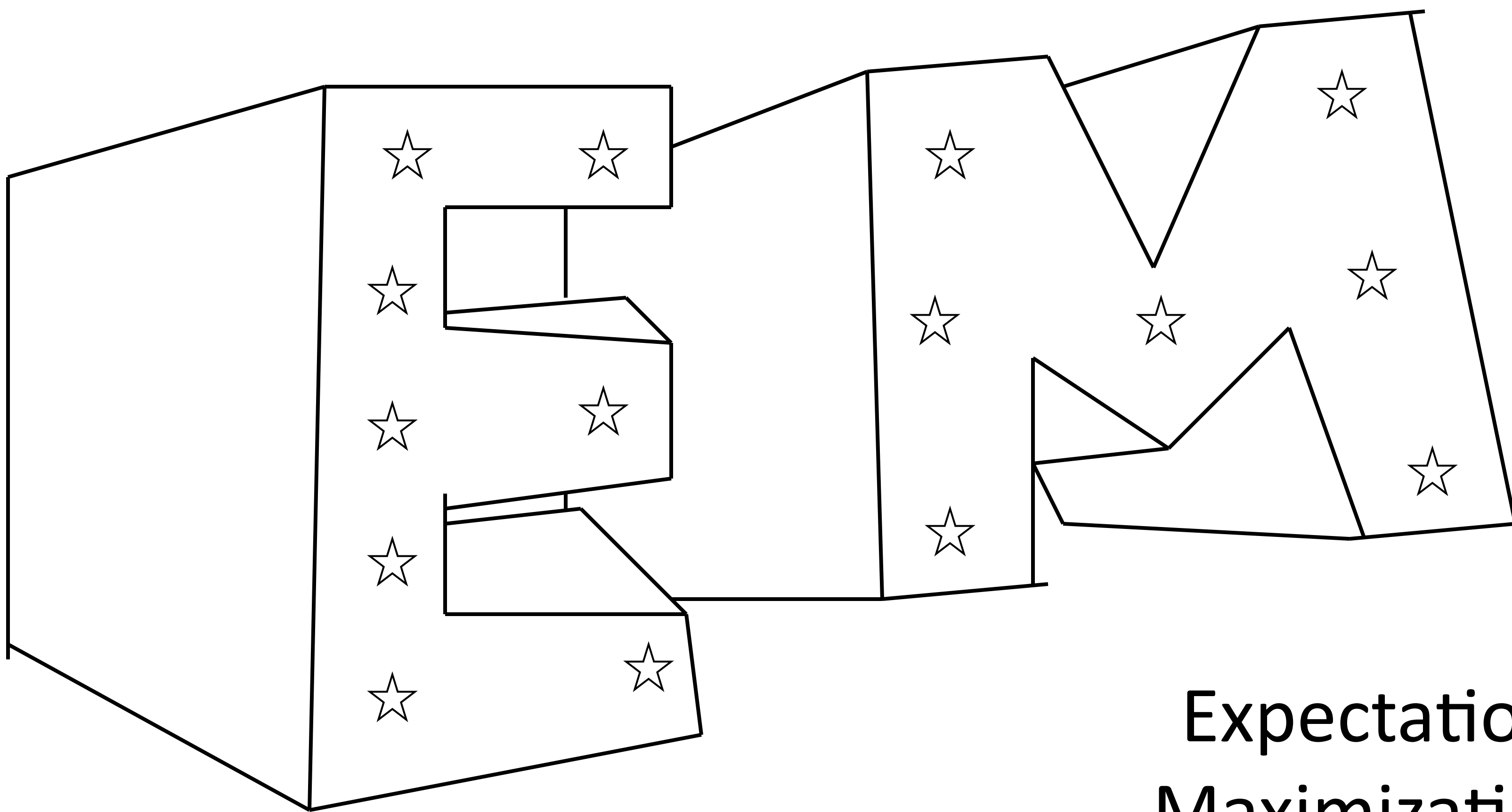
$$P(y=k\mid \mathbf{x}_j)\propto \frac{1}{(2\pi)^{m/2}\,||\Sigma_k||^{1/2}}\exp\left[-\frac{1}{2}\left(\mathbf{x}_j-\mu_k\right)^T\Sigma_k^{-1}\left(\mathbf{x}_j-\mu_k\right)\right]P(y=k)$$

- Marginal likelihood:

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

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

- Need to differentiate and solve for μ_k, Σ_k, and P(Y=k) for k=1..K
- There will be no closed form solution, gradient is complex, lots of local optimum
- ***Wouldn't it be nice if there was a better way!?!***



Expectation
Maximization

The EM Algorithm

- A clever method for maximizing marginal likelihood:
  - $\text{argmax}_\theta \prod_j P(x_j) = \text{argmax}_\theta \prod_j \sum_{k=1}^{K} P(Y_j=k, x_j)$
  - A type of gradient ascent that can be easy to implement (eg, no line search, learning rates, etc.)
- Alternate between two steps:
  - Compute an expectation
  - Compute a maximization
- Not magic: ***still optimizing a non-convex function with lots of local optima***
  - The computations are just easier (often, significantly so!)

EM: Two Easy Steps

Objective: $\text{argmax}_\theta \lg \prod_j \sum_{k=1}^K P(Y_j=k, x_j \mid \theta) = \sum_j \lg \sum_{k=1}^K P(Y_j=k, x_j \mid \theta)$

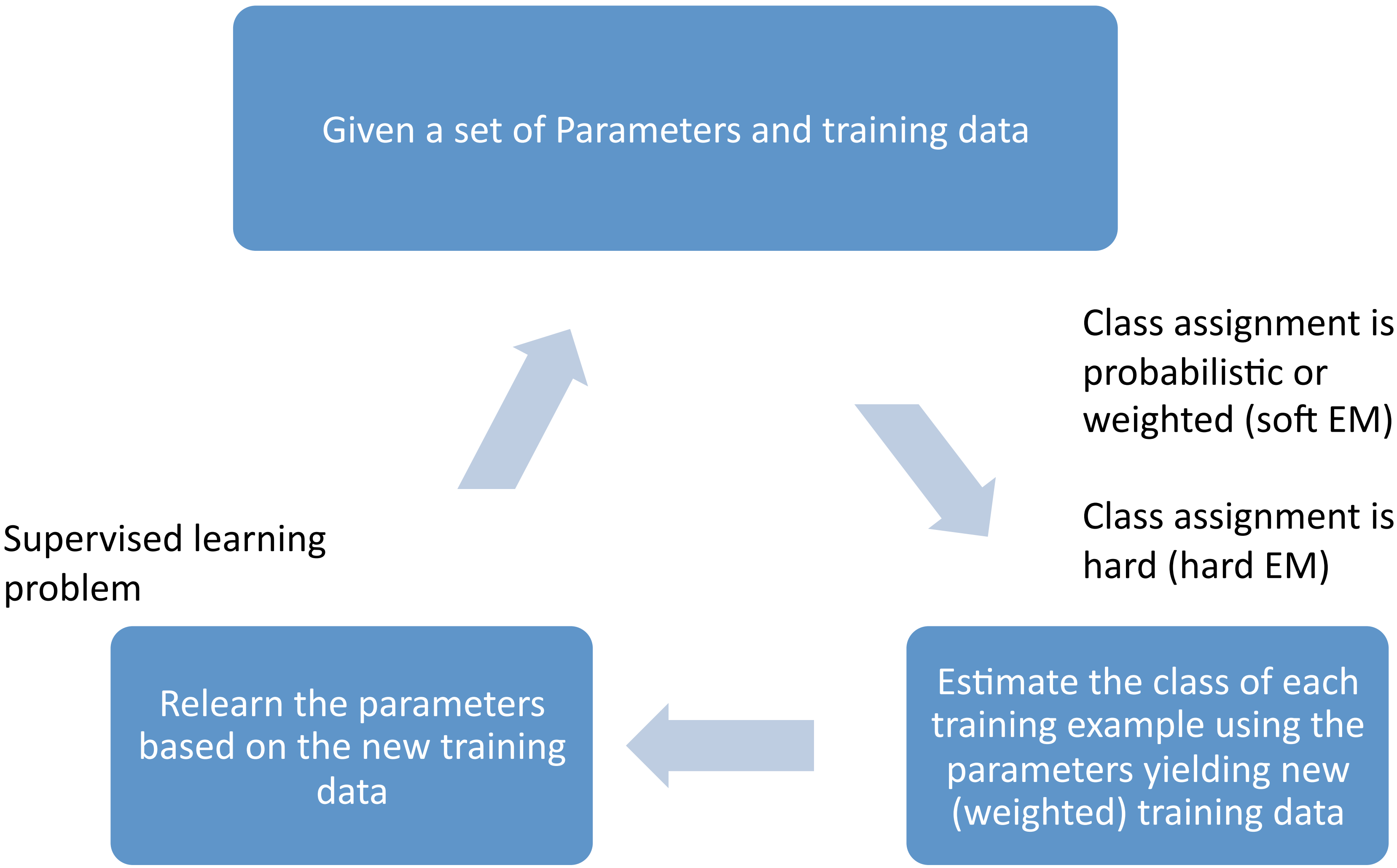
Data: $\{x_j \mid j=1 .. n\}$

Notation a bit inconsistent
Parameters = $\theta=\lambda$

- **E-step**: Compute expectations to "fill in" missing y values according to current parameters, θ
  - For all examples j and values k for Y_j, compute: $P(Y_j=k \mid x_j, \theta)$
- **M-step**: Re-estimate the parameters with "weighted" MLE estimates
  - Set $\theta = \text{argmax}_\theta \sum_j \sum_k P(Y_j=k \mid x_j, \theta) \log P(Y_j=k, x_j \mid \theta)$

Especially useful when the E and M steps have closed form solutions!!!

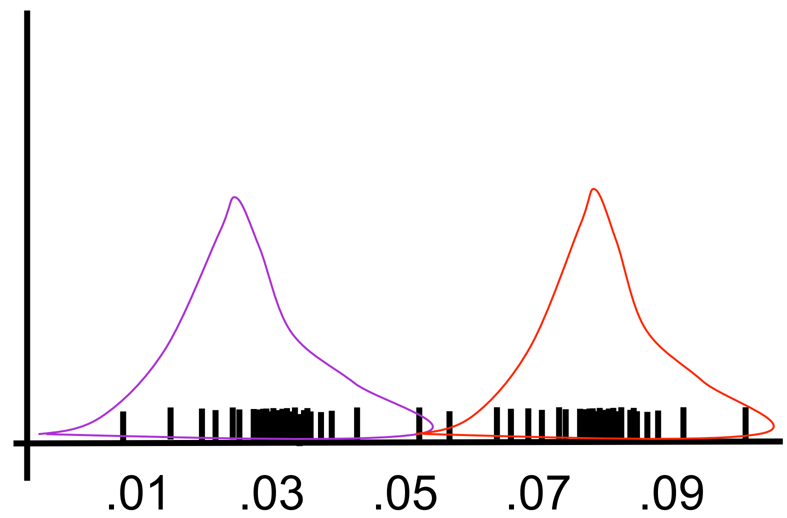
EM algorithm: Pictorial View



Simple example: learn means only!

Consider:

- 1D data
- Mixture of k=2 Gaussians
- Variances fixed to σ=1
- Distribution over classes is uniform
- Just need to estimate μ_1 and μ_2



$$\prod_{j=1}^{m}\sum_{k=1}^{K} P(x, Y_j = k) \propto \prod_{j=1}^{m}\sum_{k=1}^{K=2} \exp\left[-\frac{1}{2\sigma^2}\|x-\mu_k\|^2\right] P(Y_j = k)$$

EM for GMMs: only learning means

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

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

E-step

Compute "expected" classes of all datapoints

$$P\left(Y_j = k \middle| x_j, \mu_1 \ldots \mu_K\right) \propto \exp\left(-\frac{1}{2\sigma^2}\left\|x_j - \mu_k\right\|^2\right) \mathrm{P}\left(Y_j = k\right)$$

M-step

Compute most likely new **μ**s given class expectations

$$\mu_k = \frac{\sum_{j=1}^{m} P\left(Y_j = k \middle| x_j\right) x_j}{\sum_{j=1}^{m} P\left(Y_j = k \middle| x_j\right)}$$

E.M. for General GMMs

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

$$\lambda_t = \{ \mu_1^{(t)}, \mu_2^{(t)} \ldots \mu_K^{(t)}, \Sigma_1^{(t)}, \Sigma_2^{(t)} \ldots \Sigma_K^{(t)}, p_1^{(t)}, p_2^{(t)} \ldots p_K^{(t)} \}$$

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

E-step

Compute "expected" classes of all datapoints for each class

$$\mathrm{P}\left(Y_j = k \middle| x_j, \lambda_t\right) \propto p_k^{(t)} \mathrm{p}\left(x_j \middle| \mu_k^{(t)}, \Sigma_k^{(t)}\right)$$

Just evaluate a Gaussian at x_j

M-step

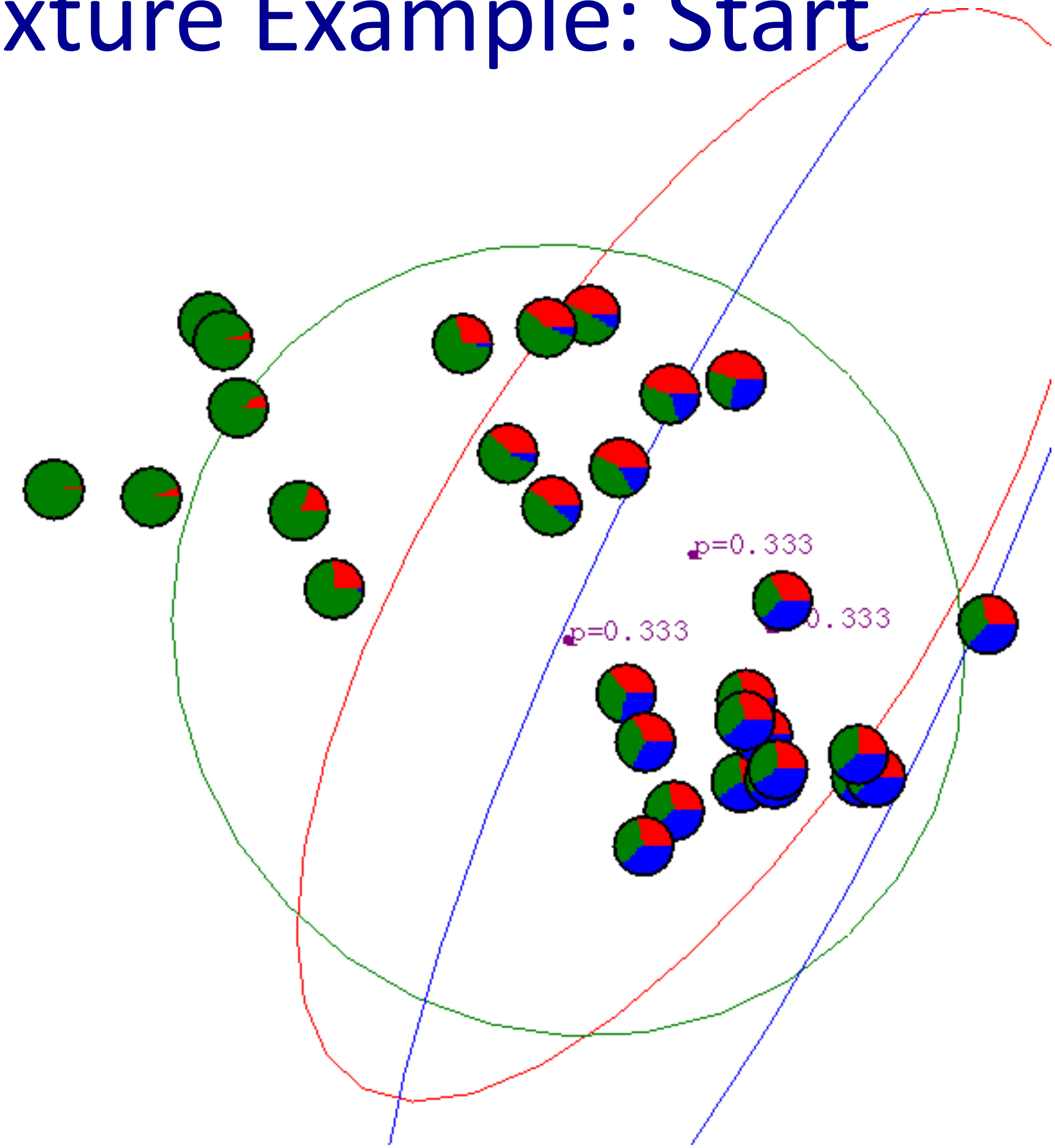
Compute weighted MLE for **μ** given expected classes above

$$\mu_k^{(t+1)} = \frac{\sum_j \mathrm{P}\left(Y_j = k \middle| x_j, \lambda_t\right) x_j}{\sum_j \mathrm{P}\left(Y_j = k \middle| x_j, \lambda_t\right)} \qquad \Sigma_k^{(t+1)} = \frac{\sum_j \mathrm{P}\left(Y_j = k \middle| x_j, \lambda_t\right) \left[x_j - \mu_k^{(t+1)}\right]\left[x_j - \mu_k^{(t+1)}\right]^T}{\sum_j \mathrm{P}\left(Y_j = k \middle| x_j, \lambda_t\right)}$$

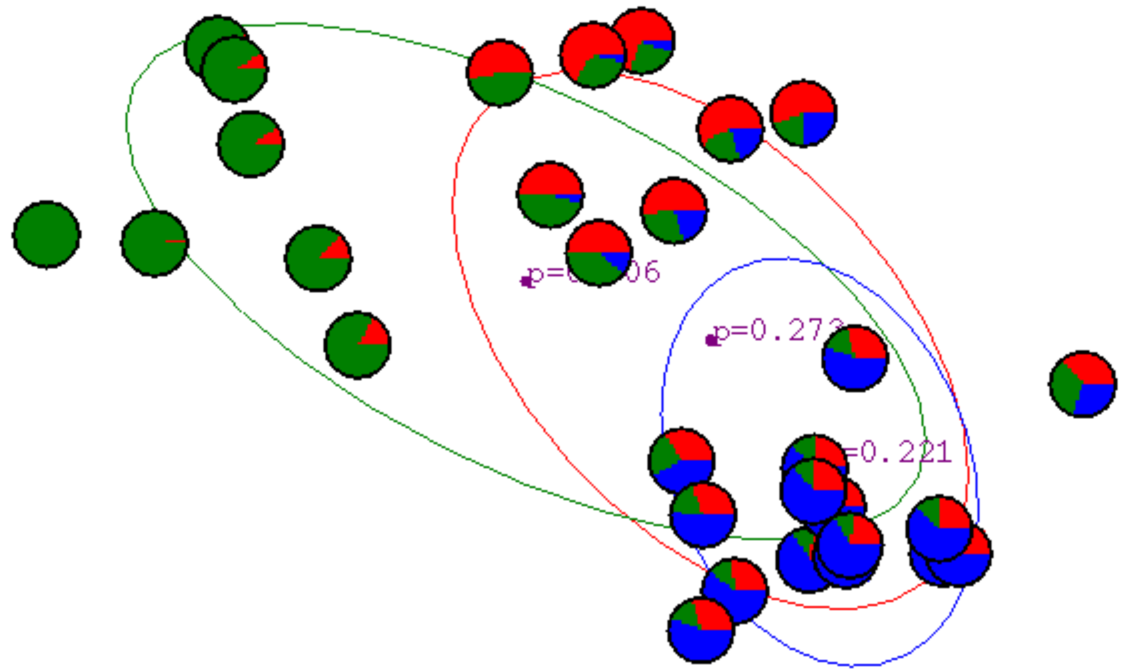
$$p_k^{(t+1)} = \frac{\sum_j \mathrm{P}\left(Y_j = k \middle| x_j, \lambda_t\right)}{m}$$

m = #training examples

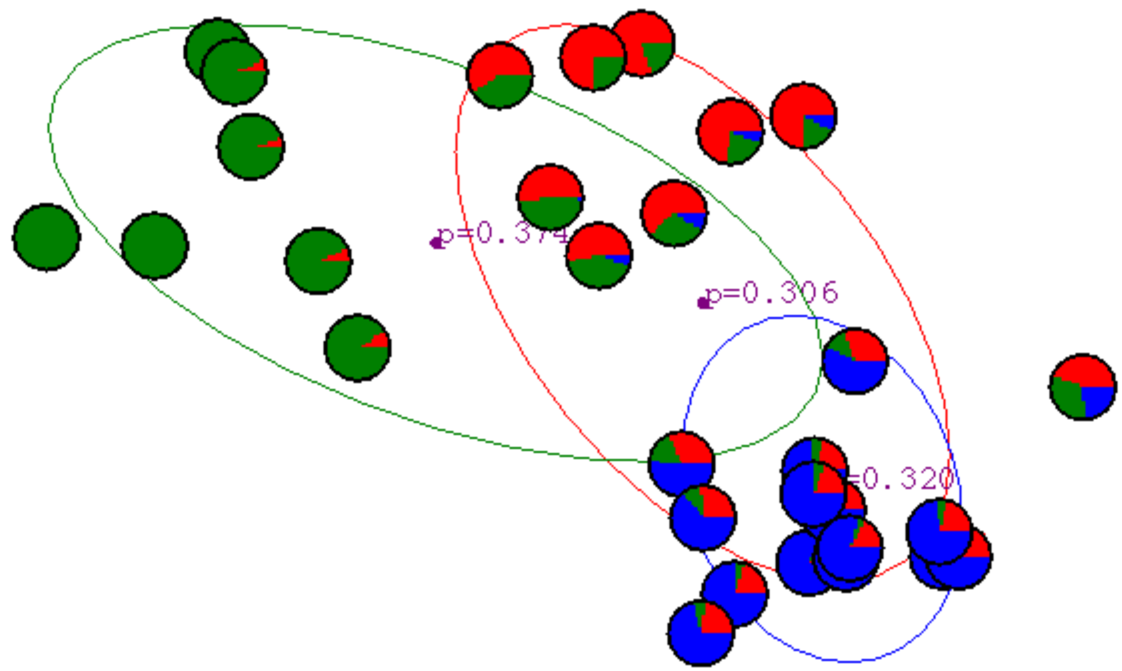
Gaussian Mixture Example: Start



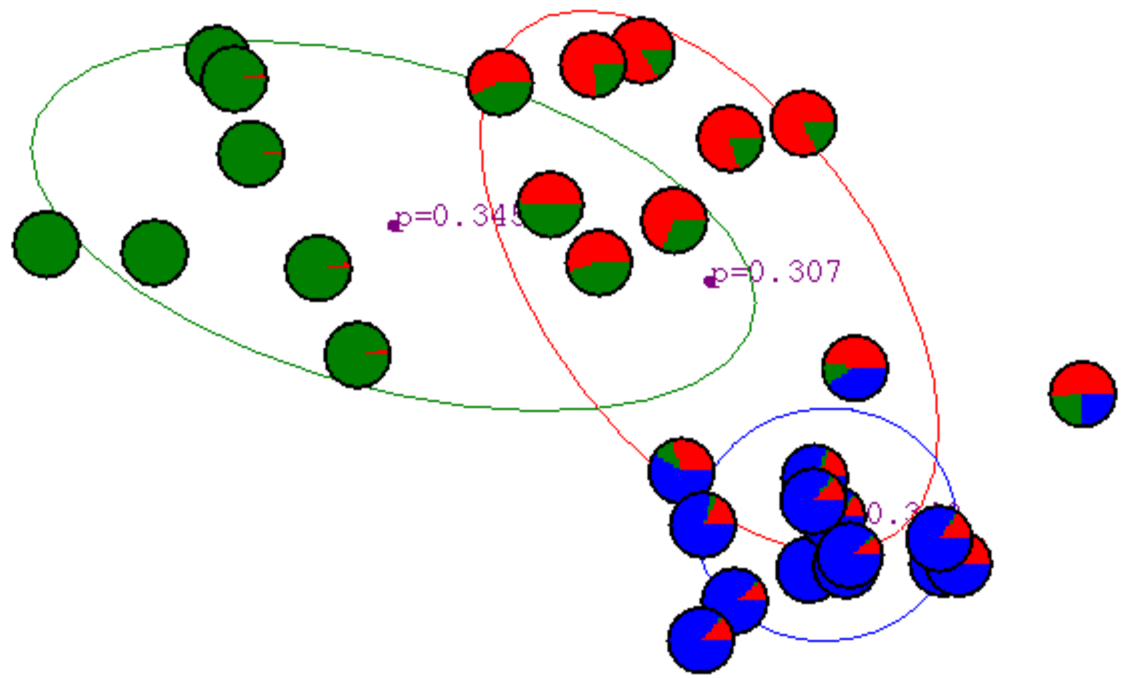
After first iteration



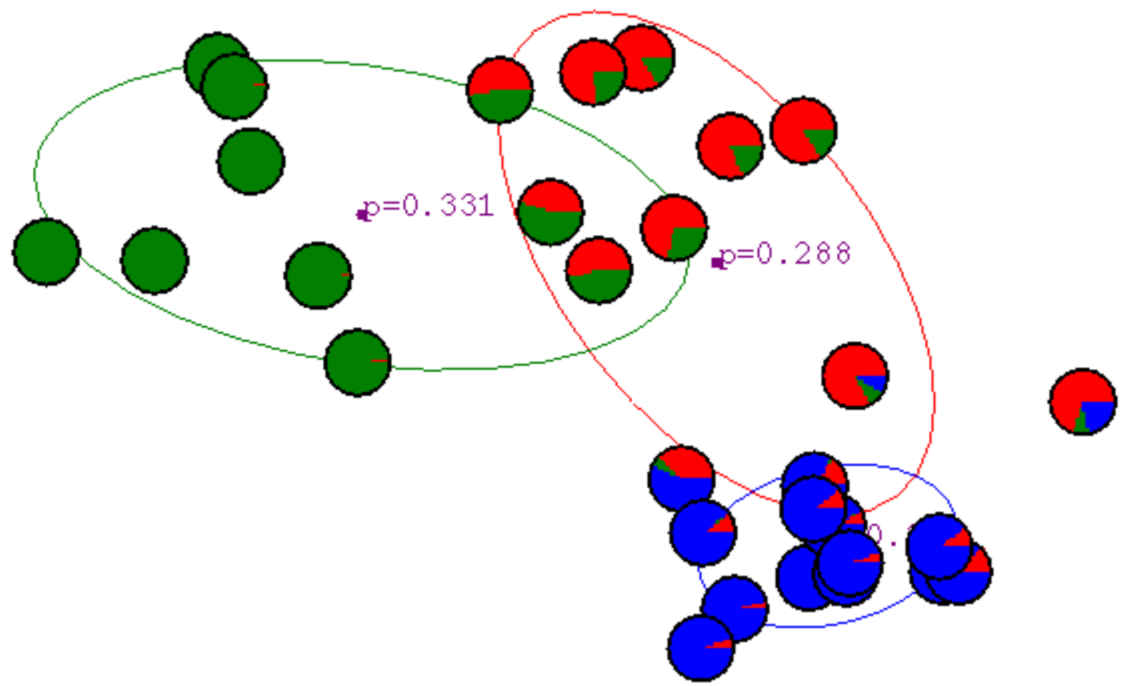
After 2nd iteration



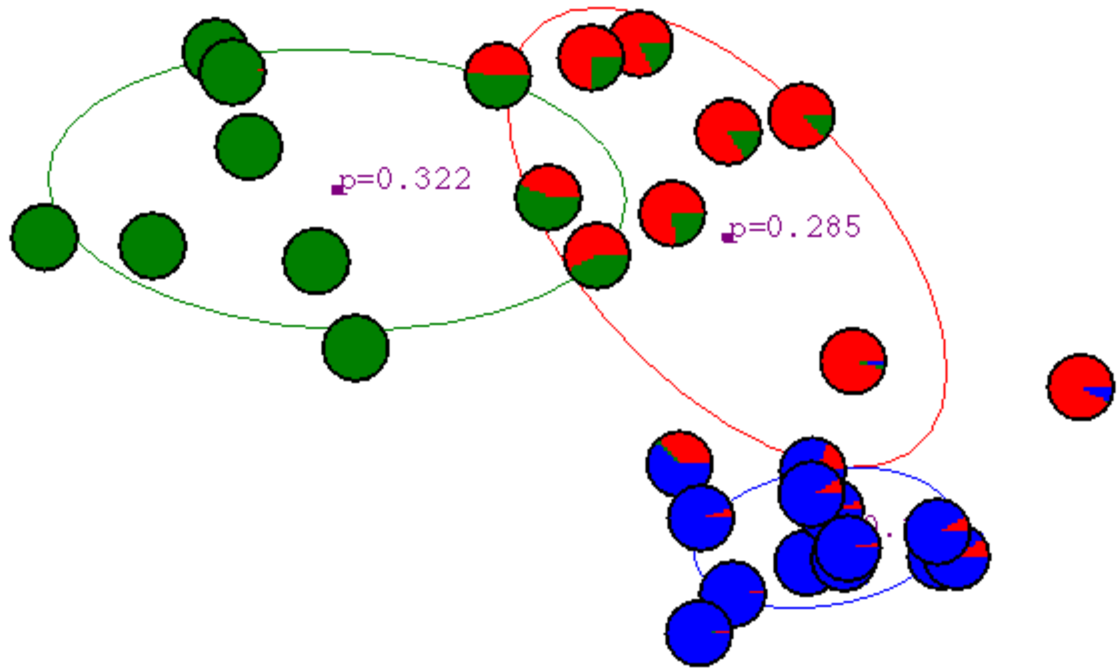
After 3rd iteration



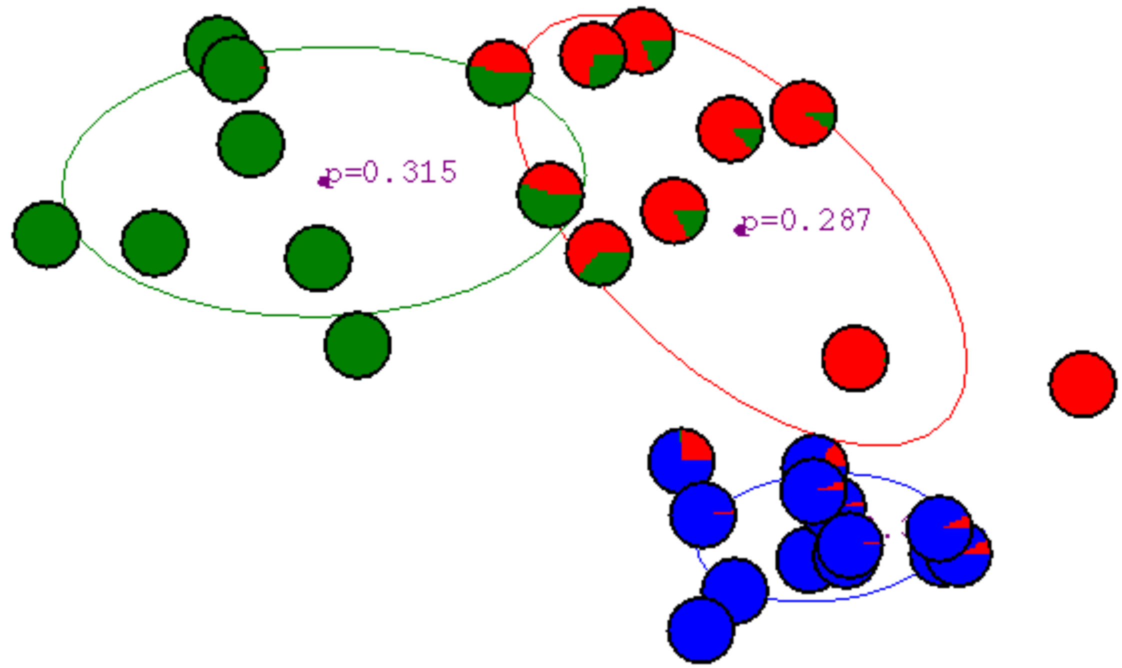
After 4th iteration



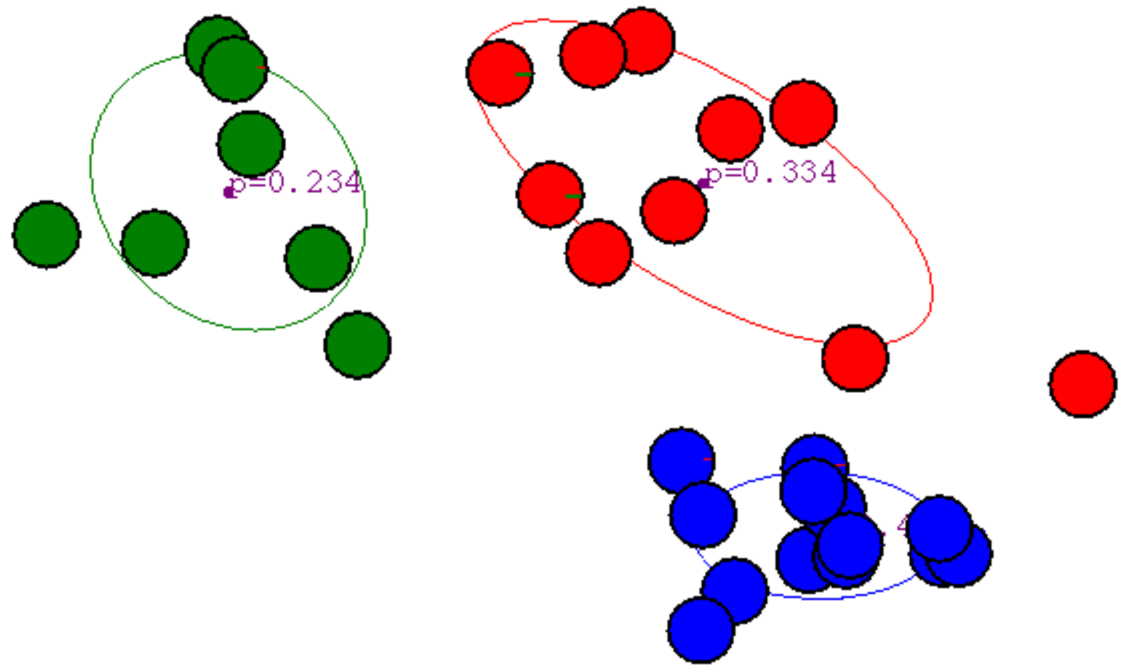
After 5th iteration



After 6th iteration



After 20th iteration



What if we do hard assignments?

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

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

E-step

Compute "expected" classes of all datapoints

$$P\left(Y_j = k \middle| x_j, \mu_1 \ldots \mu_K\right) \propto \exp\left(-\frac{1}{2\sigma^2}\left\|x_j - \mu_k\right\|^2\right) \mathrm{P}\left(Y_j = k\right)$$

M-step

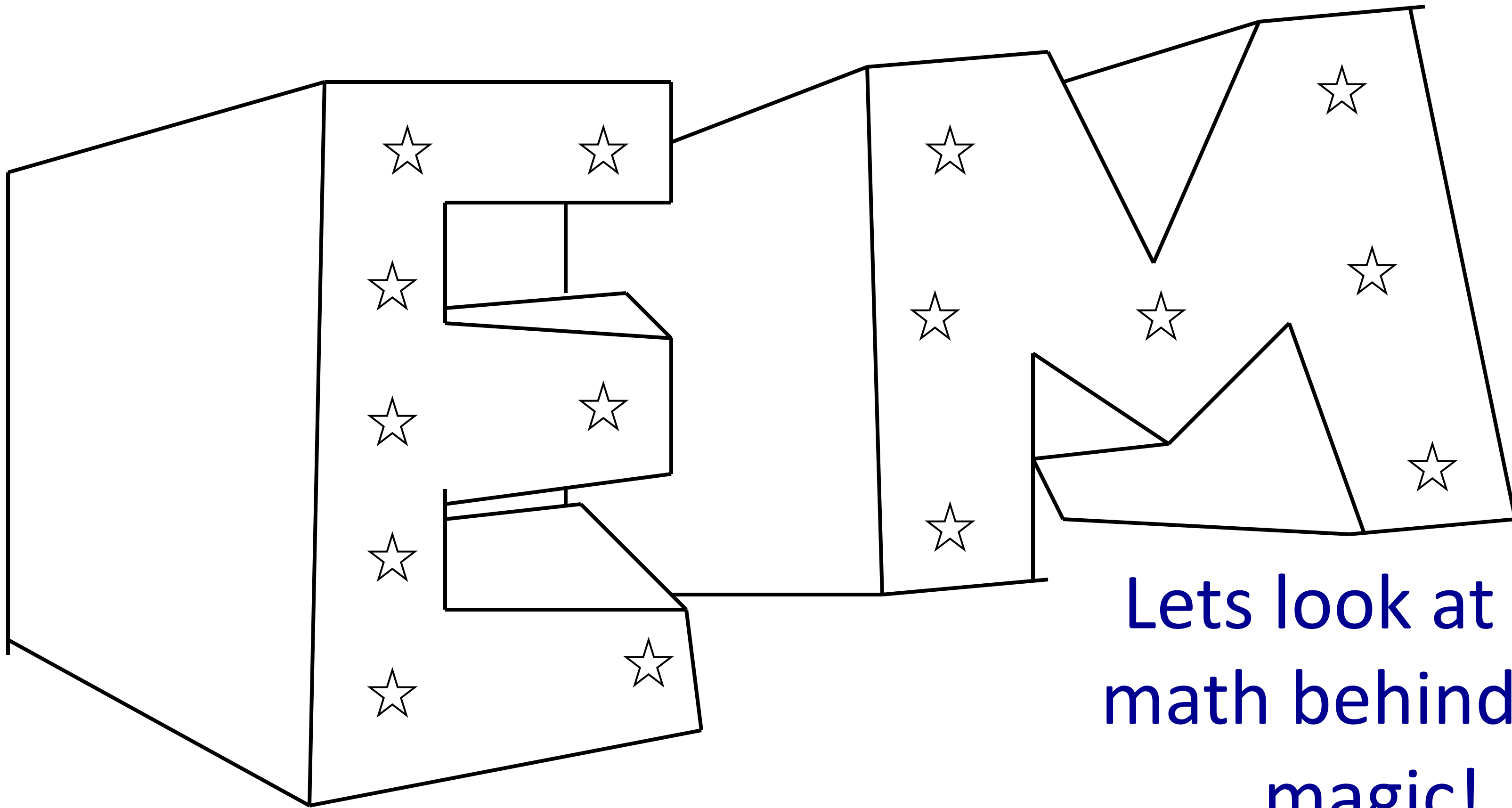
Compute most likely new **μ**s given class expectations

$$\mu_k = \frac{\sum_{j=1}^{m} P\left(Y_j = k \middle| x_j\right) x_j}{\sum_{j=1}^{m} P\left(Y_j = k \middle| x_j\right)}$$

$$\mu_k = \frac{\delta\left(Y_j = k, x_j\right) x_j}{\sum_{j=1}^{m} \delta\left(Y_j = k, x_j\right)}$$

δ represents hard assignment to "most likely" or nearest cluster

Equivalent to k-means clustering algorithm!!!



Lets look at the math behind the magic!

Next lecture, we will argue that EM:

- Optimizes a bound on the likelihood
- Is a type of coordinate ascent
- Is guaranteed to converge to a (often local) optima