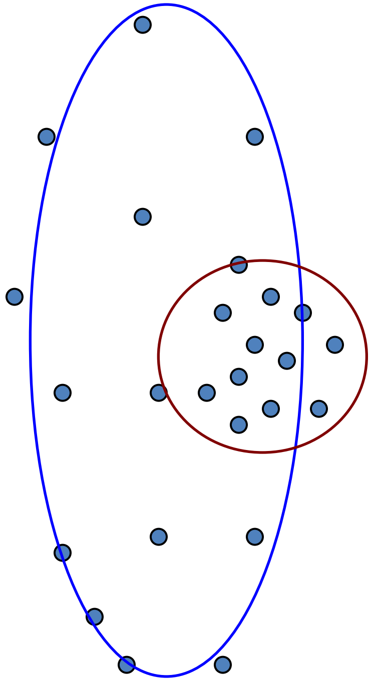
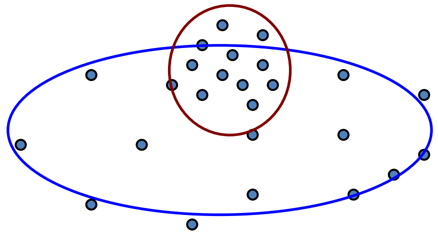
The Evils of "**Hard Assignments**"?



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

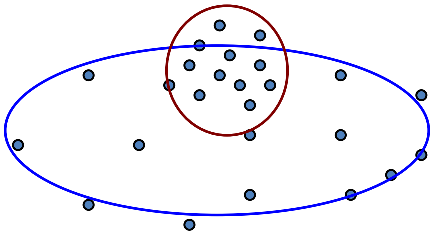
Probabilistic Clustering



- 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
…	…	…

What Model Should We Use?



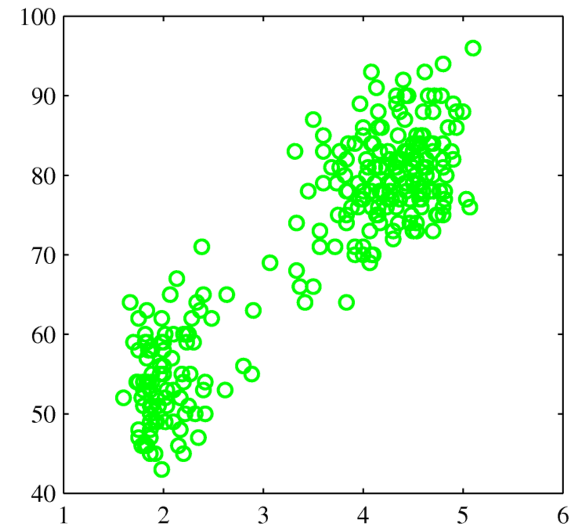
- Depends on X!
- Here, maybe Gaussian Naïve Bayes?
  - Multinomial over clusters Y
  - Gaussian over each X_i given Y

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?



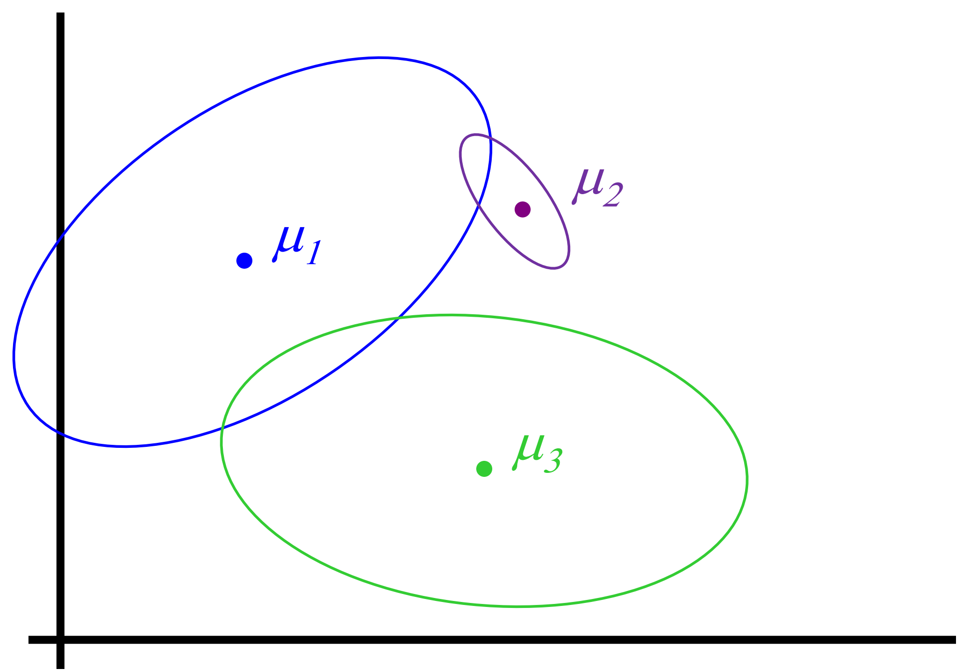
- What if the X_i co-vary?
- What if there are multiple peaks?
- Gaussian Mixture Models!
  - P(Y) still multinomial
  - P(**X**|Y) is a multivariate Gaussian dist'n

$$P(X = \mathbf{x}_j \mid Y = i) = \frac{1}{(2\quad)^{m/2} \,||\quad_i\,||^{1/2}} \exp \quad \frac{1}{2}\left(\mathbf{x}_j \qquad _i\right)^T \quad _i^{\;1}\left(\mathbf{x}_j \qquad _i\right)$$

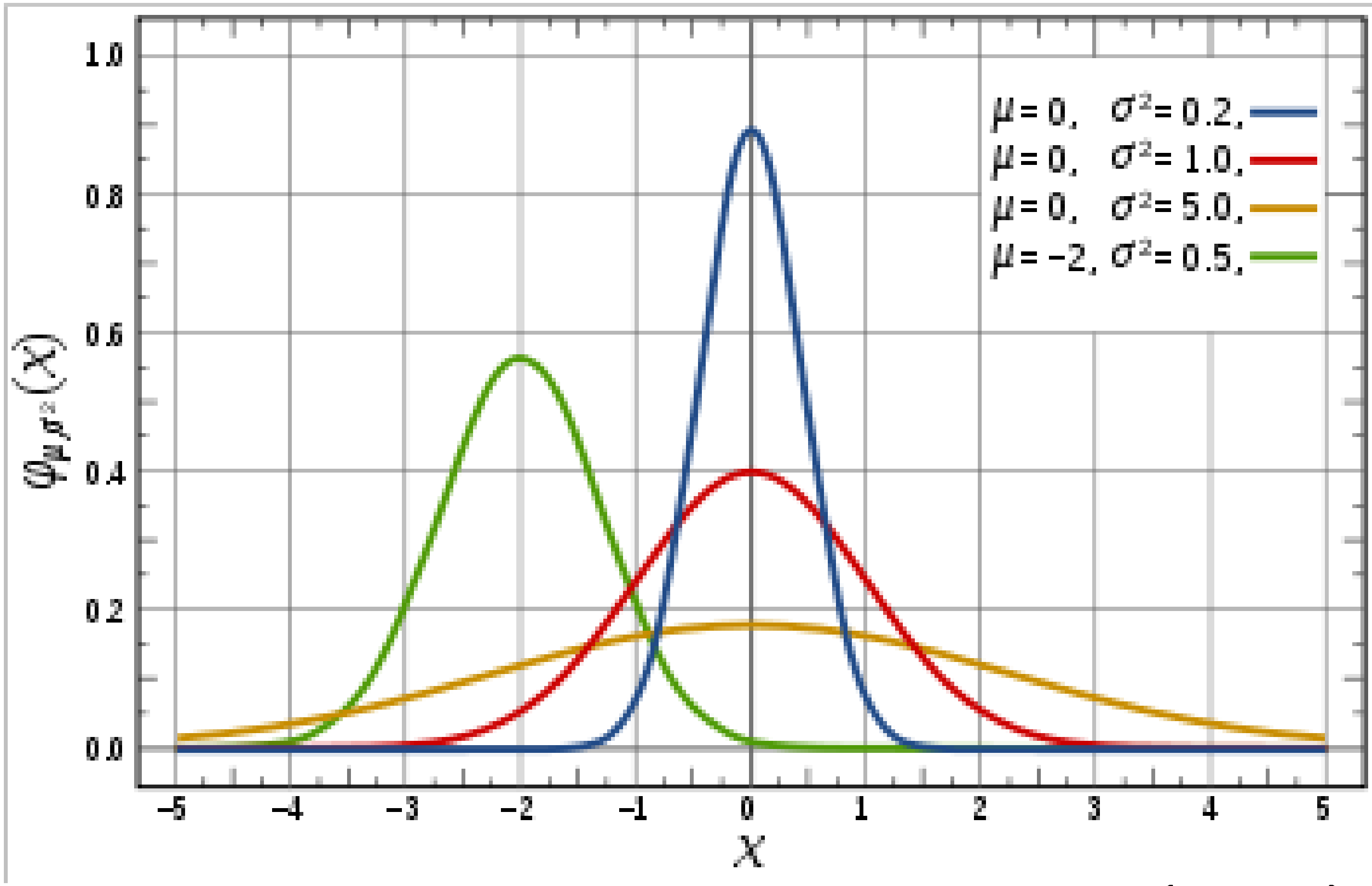
The General GMM assumption

1. What's a ***Multivariate*** Gaussian?

2. What's a ***Mixture Model***?



Review: Gaussians



$$P(x \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

Learning Gaussian Parameters (given fully-observable data)

$$\widehat{\mu}_{MLE} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

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

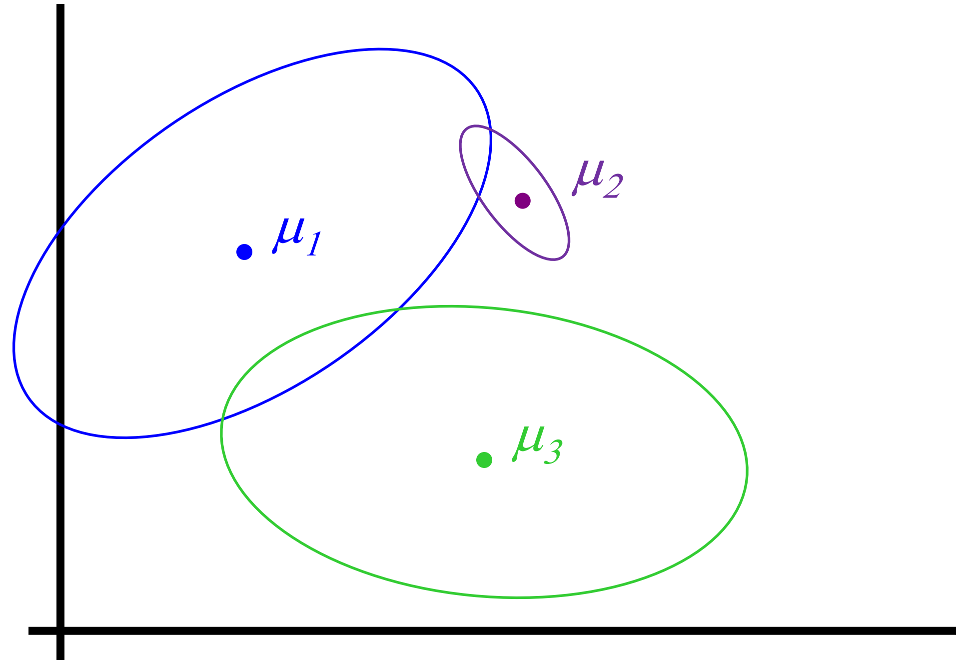
The General GMM assumption

- 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 ~ $N(m_i, \Sigma_i)$

Gaussian mixture model (GMM)



Geometry of the Multivariate Gaussian

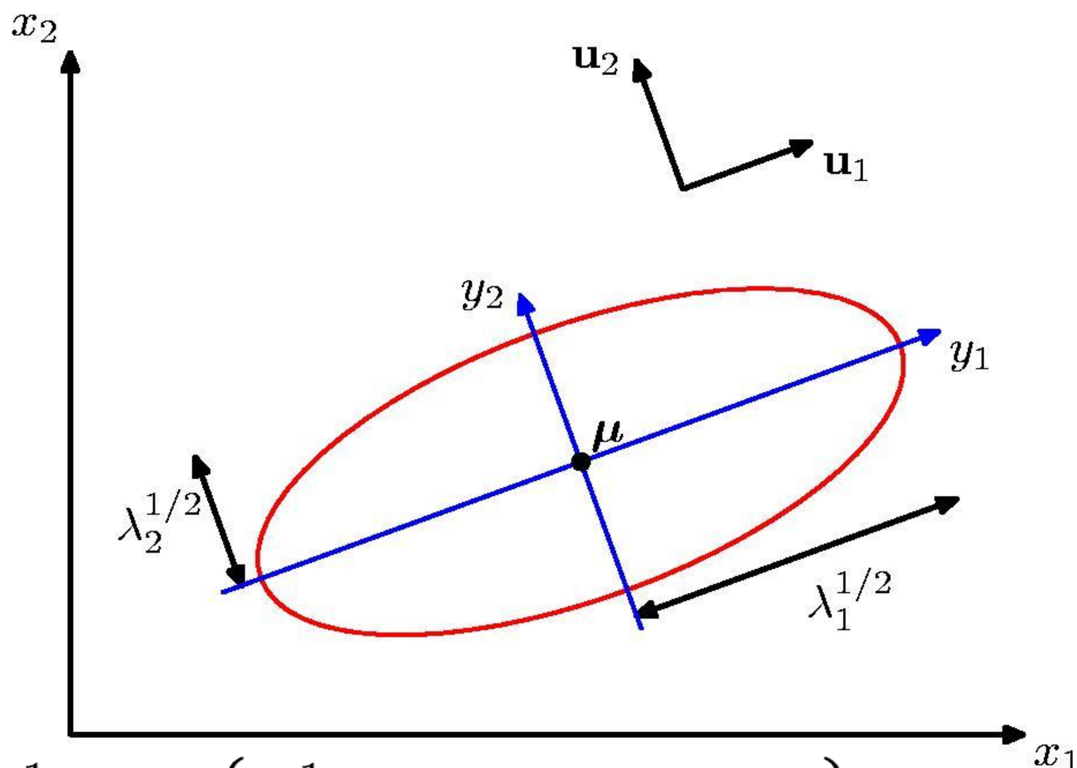
$$\Delta^2 = (\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})$$

$$\boldsymbol{\Sigma}^{-1} = \sum_{i=1}^{D} \frac{1}{\lambda_i} \mathbf{u}_i \mathbf{u}_i^{\mathrm{T}}$$

$$\Delta^2 = \sum_{i=1}^{D} \frac{y_i^2}{\lambda_i}$$

$$y_i = \mathbf{u}_i^{\mathrm{T}} (\mathbf{x} - \boldsymbol{\mu})$$

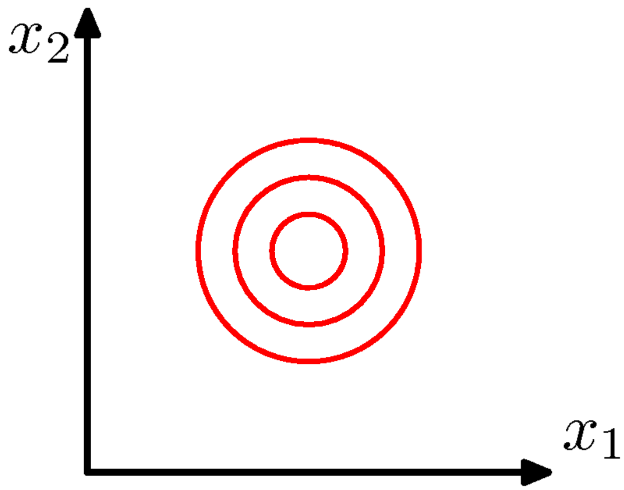
Eigenvalue, λ
Eigenvector **u**$_i$



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

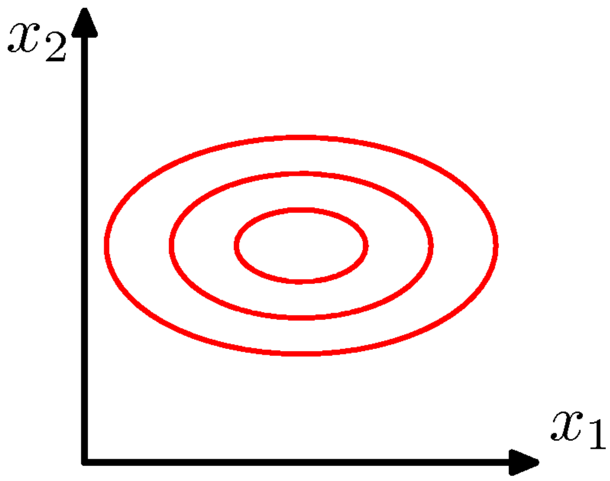
$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right\}$$

Multivariate Gaussians



$\Sigma \propto$ identity matrix

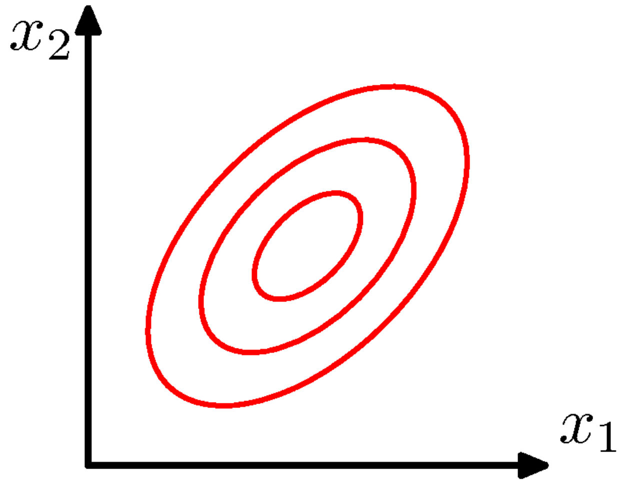
Multivariate Gaussians



Σ = diagonal matrix

X_i are independent *ala* Gaussian NB

Multivariate Gaussians

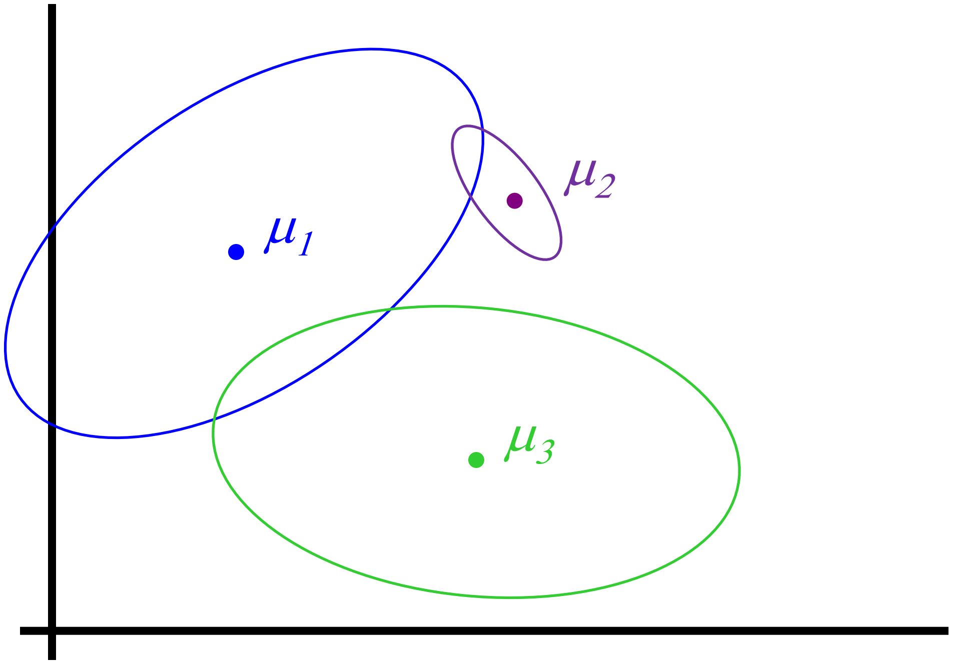


Σ = arbitrary (semidefinite) matrix
specifies rotation (change of basis)
eigenvalues specify relative elongation

The General GMM assumption

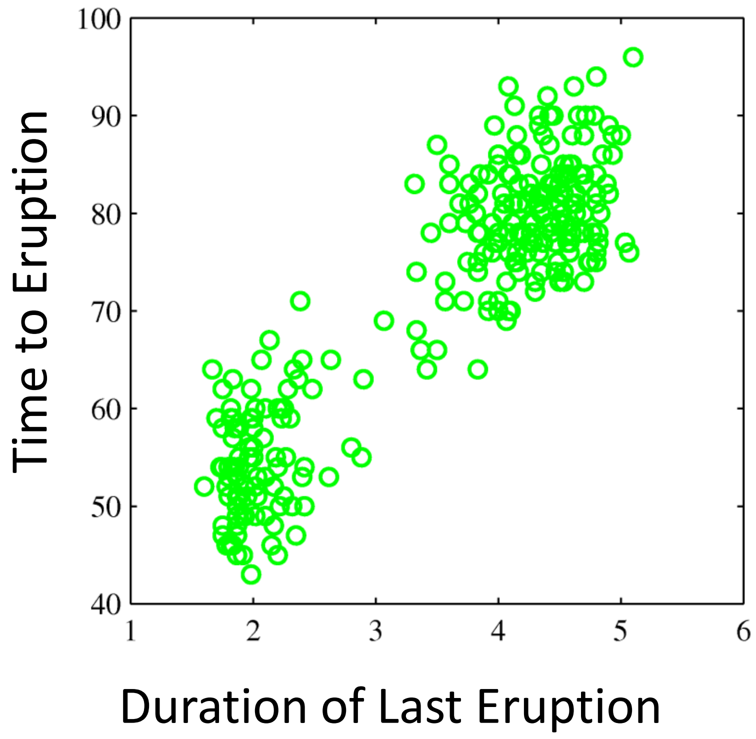
1. What's a Multivariate Gaussian?

2. What's a ***Mixture Model***?



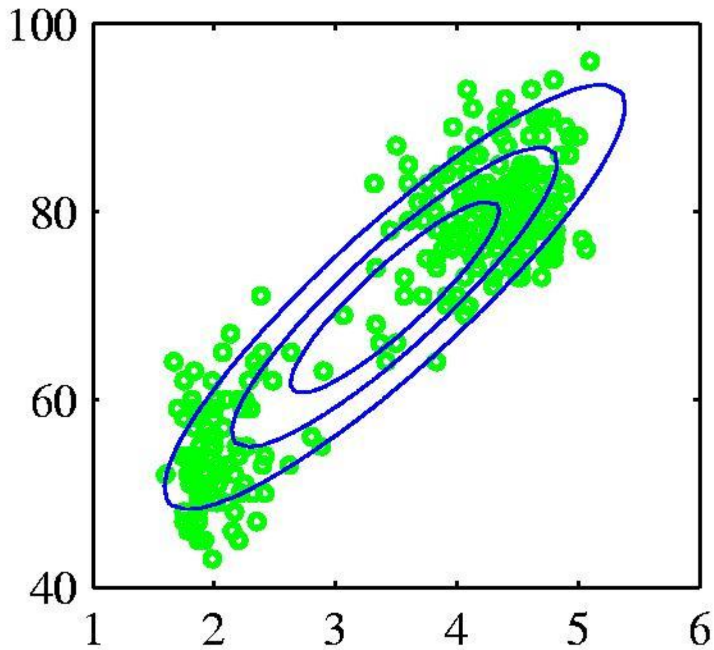
Mixtures of Gaussians (1)

Old Faithful Data Set

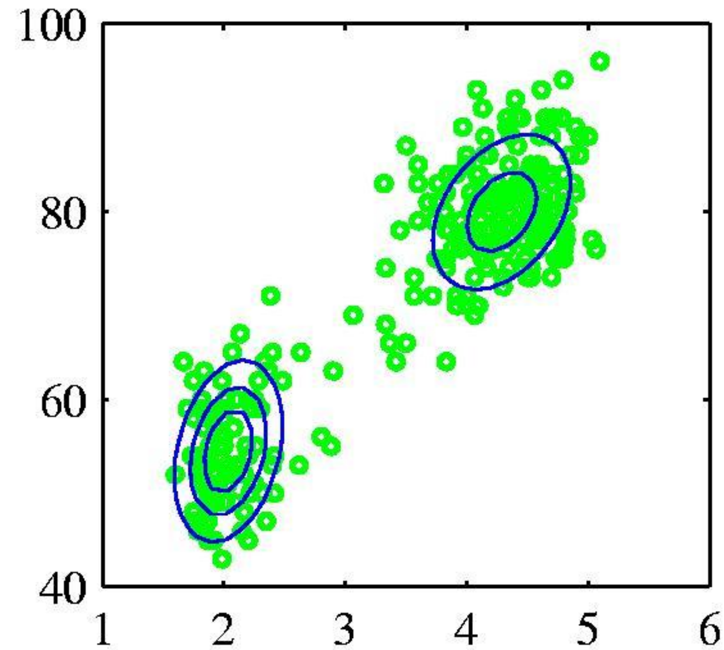


Mixtures of Gaussians (1)

Old Faithful Data Set



Single Gaussian



Mixture of two Gaussians

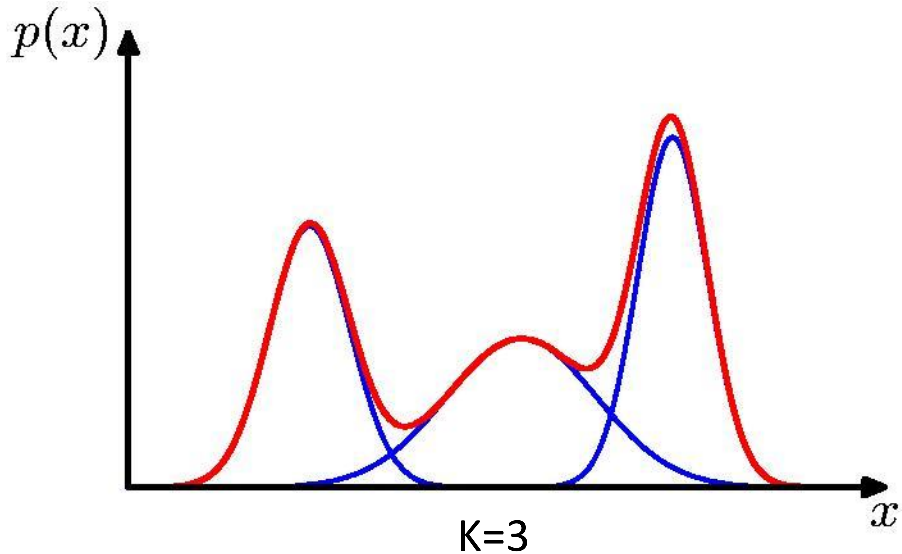
Mixtures of Gaussians (2)

Combine simple models into a complex model:

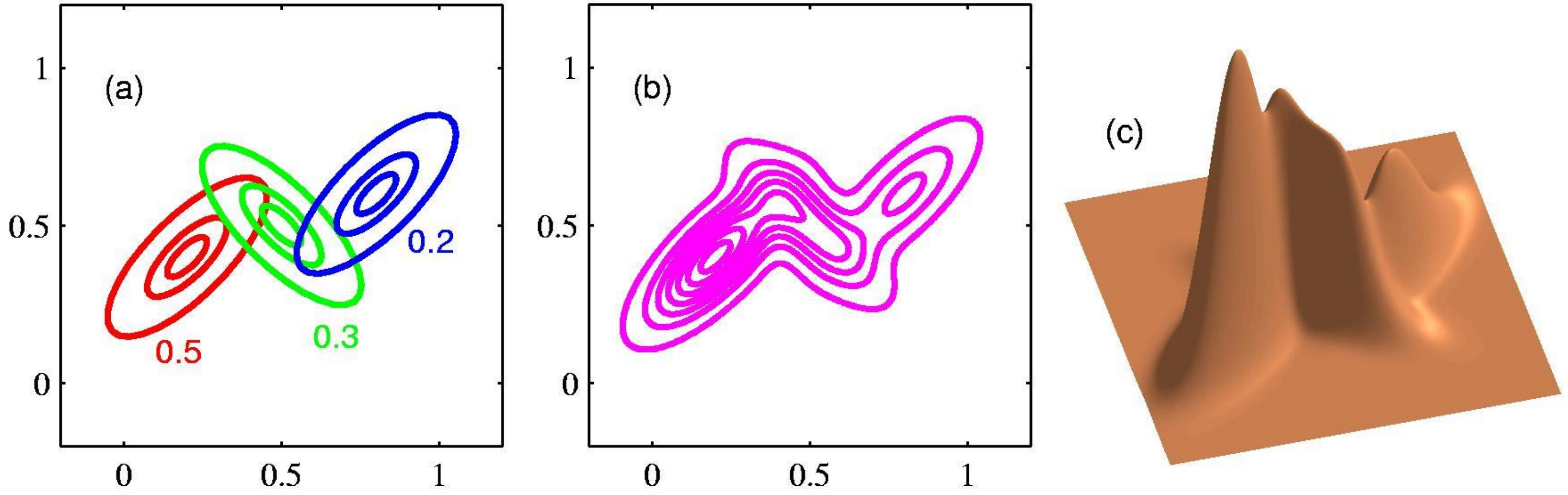
$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \underbrace{\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}_{\text{Component}}$$

Mixing coefficient

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

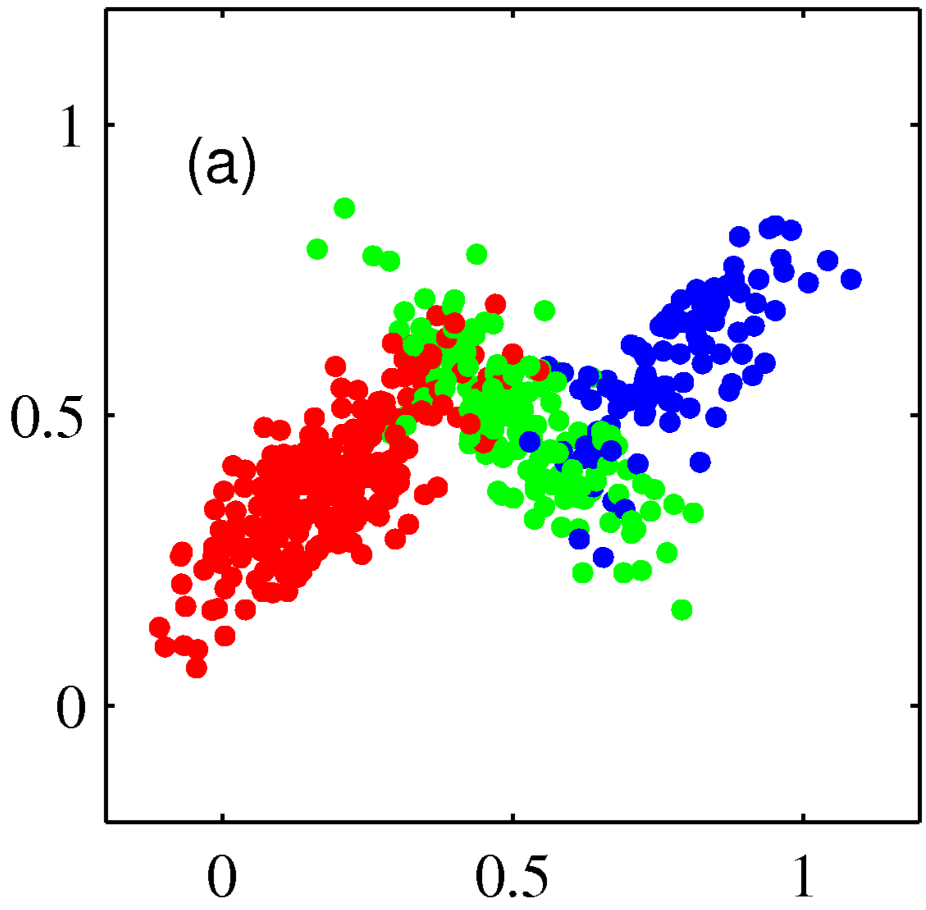


Mixtures of Gaussians (3)



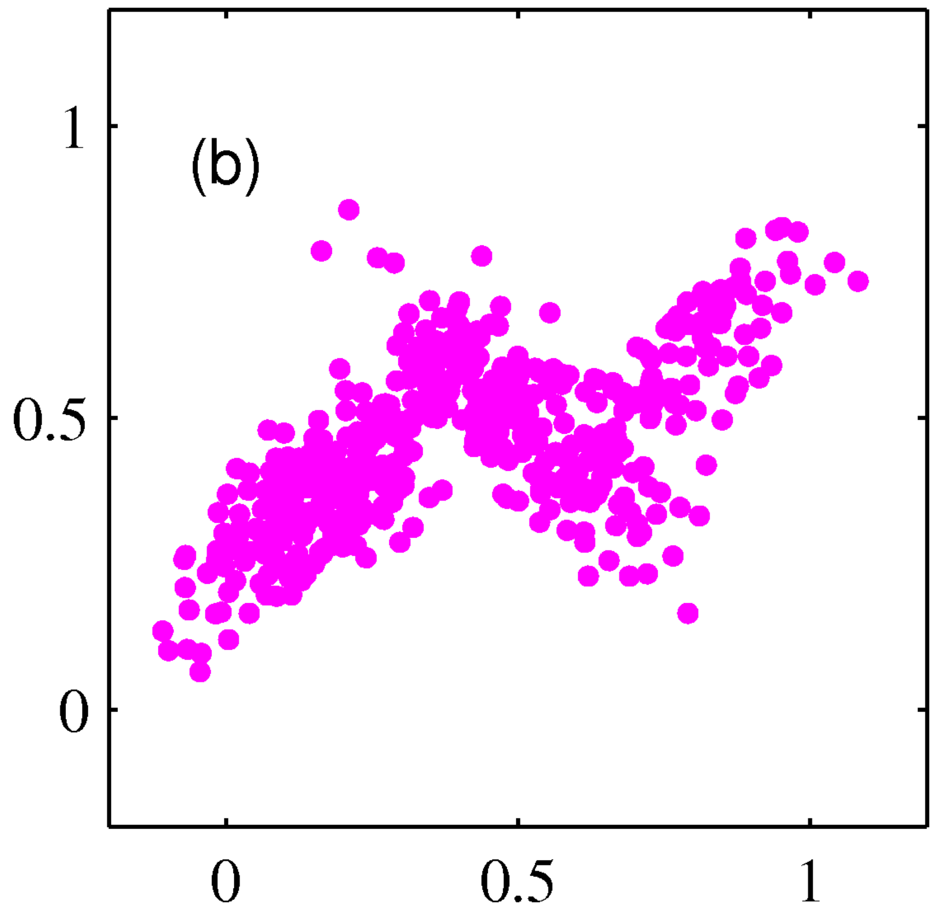
Eliminating Hard Assignments to Clusters

Model data as mixture of multivariate Gaussians



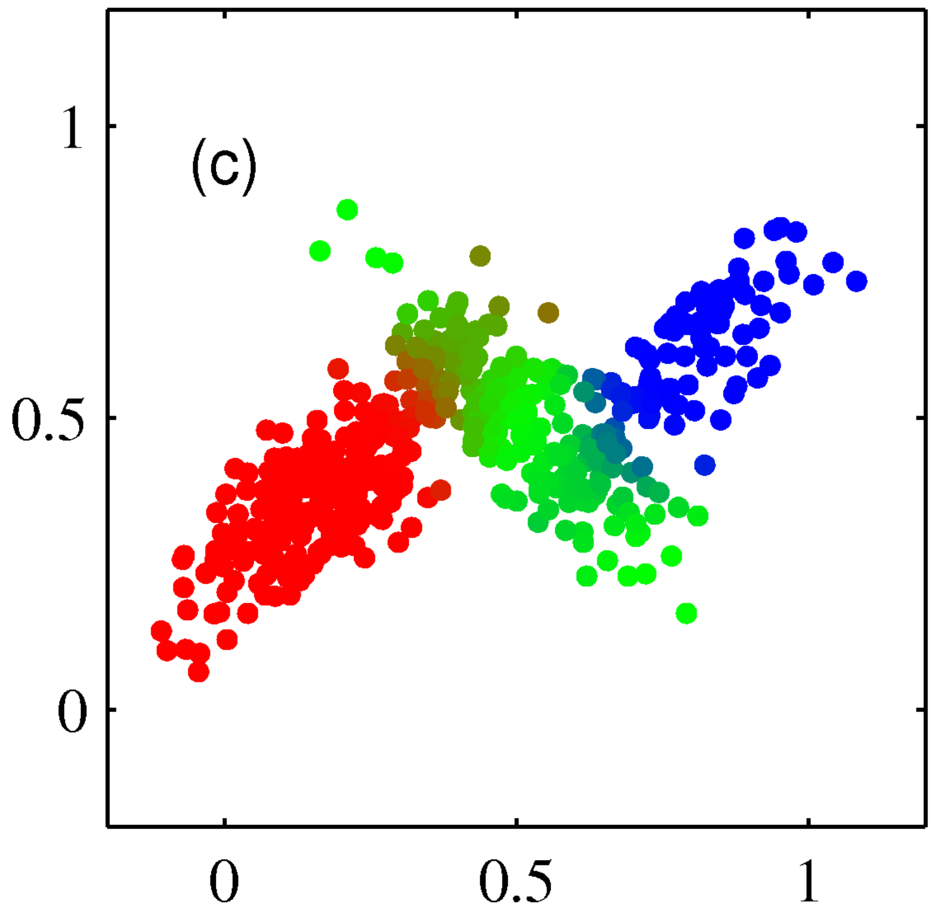
Eliminating Hard Assignments to Clusters

Model data as mixture of multivariate Gaussians



Eliminating Hard Assignments to Clusters

Model data as mixture of multivariate Gaussians



π_i = probability point was generated from i^{th} Gaussian

Detour/Review: Supervised MLE for GMM

- How do we estimate parameters for Gaussian Mixtures with fully supervised data?
- Have to define objective and solve optimization problem.

$$P(y=i,\mathbf{x}_j)=\frac{1}{(2\quad)^{m/2}\,\|\quad_i\|^{1/2}}\exp\quad\frac{1}{2}\left(\mathbf{x}_j\qquad_i\right)^T\quad_i^{\;1}\left(\mathbf{x}_j\qquad_i\right)\;P(y=i)$$

- For example, MLE estimate has closed form solution:

$${}_{ML}=\frac{1}{n}\quad{}^{n}_{j=1}\;x_n \qquad\qquad {}_{ML}=\frac{1}{n}\quad{}^{n}_{j=1}\left(\mathbf{x}_j\qquad{}_{ML}\right)\left(\mathbf{x}_j\qquad{}_{ML}\right)^T$$

Compare

- Univariate Gaussian

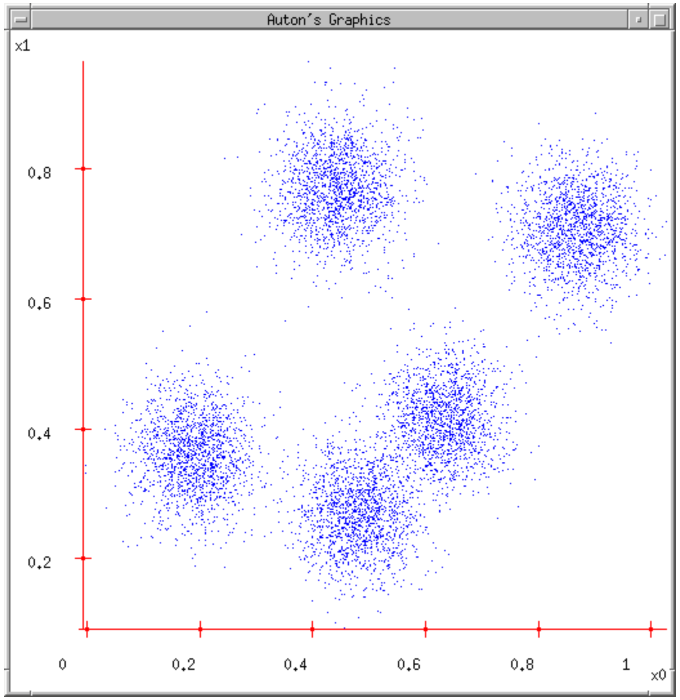
$$\mu_{MLE} = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad \sigma^2_{MLE} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \widehat{\mu})^2$$

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

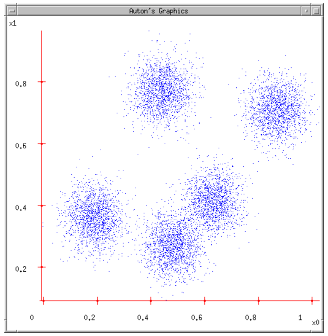
$$_{ML} = \frac{1}{n} \, {}_{j=1}^{n} \, x_n \qquad _{ML} = \frac{1}{n} \, {}_{j=1}^{n} \left(\mathbf{x}_j \quad _{ML}\right)\left(\mathbf{x}_j \quad _{ML}\right)^T$$

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

- MLE:
  - – $\text{argmax}_\theta \prod_j P(y_j, x_j)$
  - – θ: all model parameters
    - eg, class probs, means, and variance for naïve Bayes
- But we don't know y_j's!!!
- Maximize ***marginal likelihood***:
  - – $\text{argmax}_\theta \prod_j P(x_j) = \text{argmax} \prod_j \sum_{i=1}^{k} P(y_j=i, x_j)$



How do we optimize? Closed Form?



- Maximize ***marginal likelihood***:
  - $\text{argmax}_\theta \prod_j P(x_j) = \text{argmax} \prod_j \sum_{i=1}^{k} P(y_j=i,x_j)$
- Almost always a hard problem!
  - Usually no closed form solution
  - Even when P(X,Y) is convex, P(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

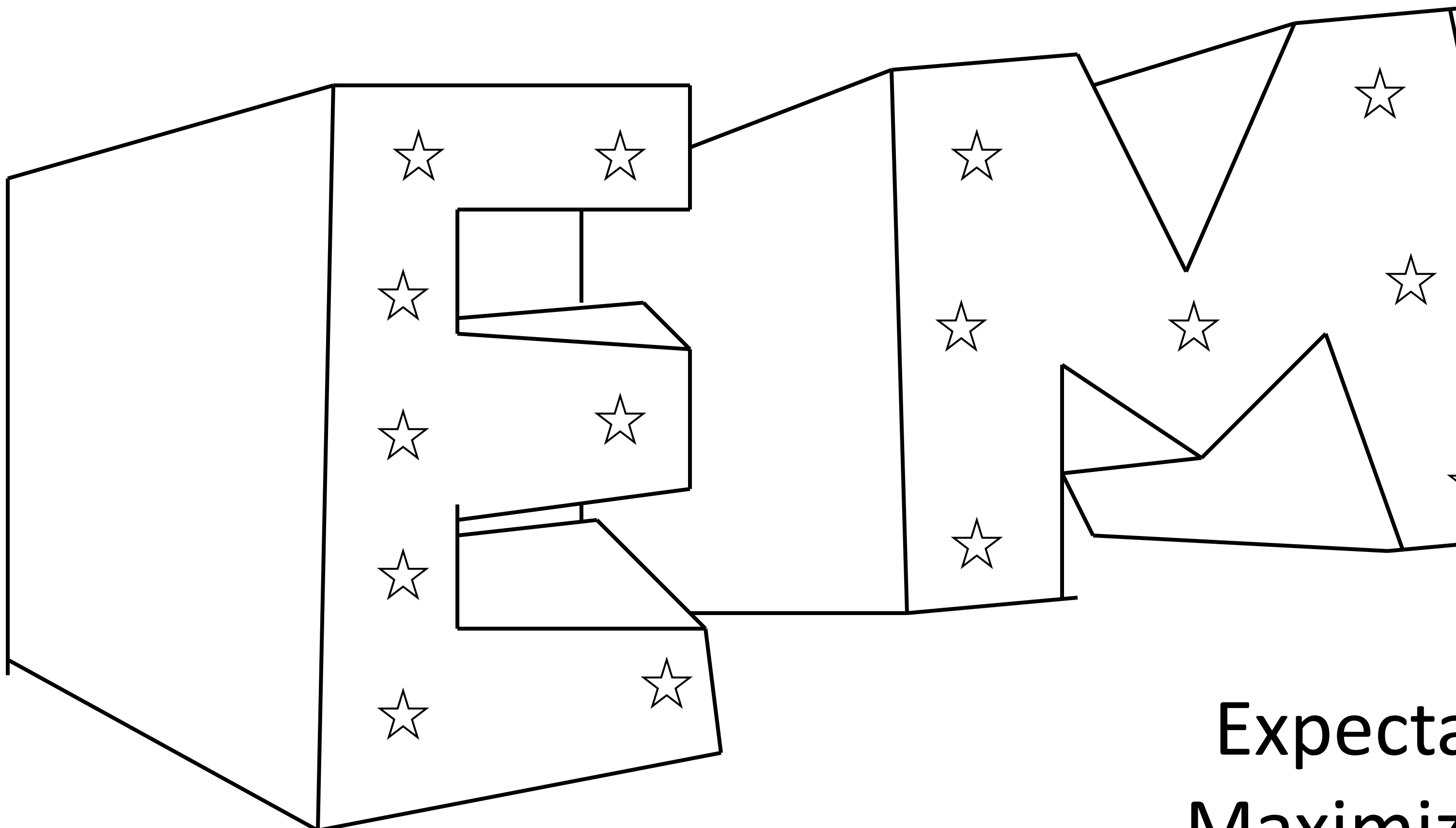
$$P(y=i \mid \mathbf{x}_j) \propto \frac{1}{(2\pi)^{m/2} \| \Sigma_i \|^{1/2}} \exp\left[-\frac{1}{2}\left(\mathbf{x}_j - \mu_i\right)^T \Sigma_i^{-1} \left(\mathbf{x}_j - \mu_i\right)\right] P(y=i)$$

- Marginal likelihood:

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

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

- Need to differentiate and solve for μ_i, Σ_i, and P(Y=i) for i=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!?!***

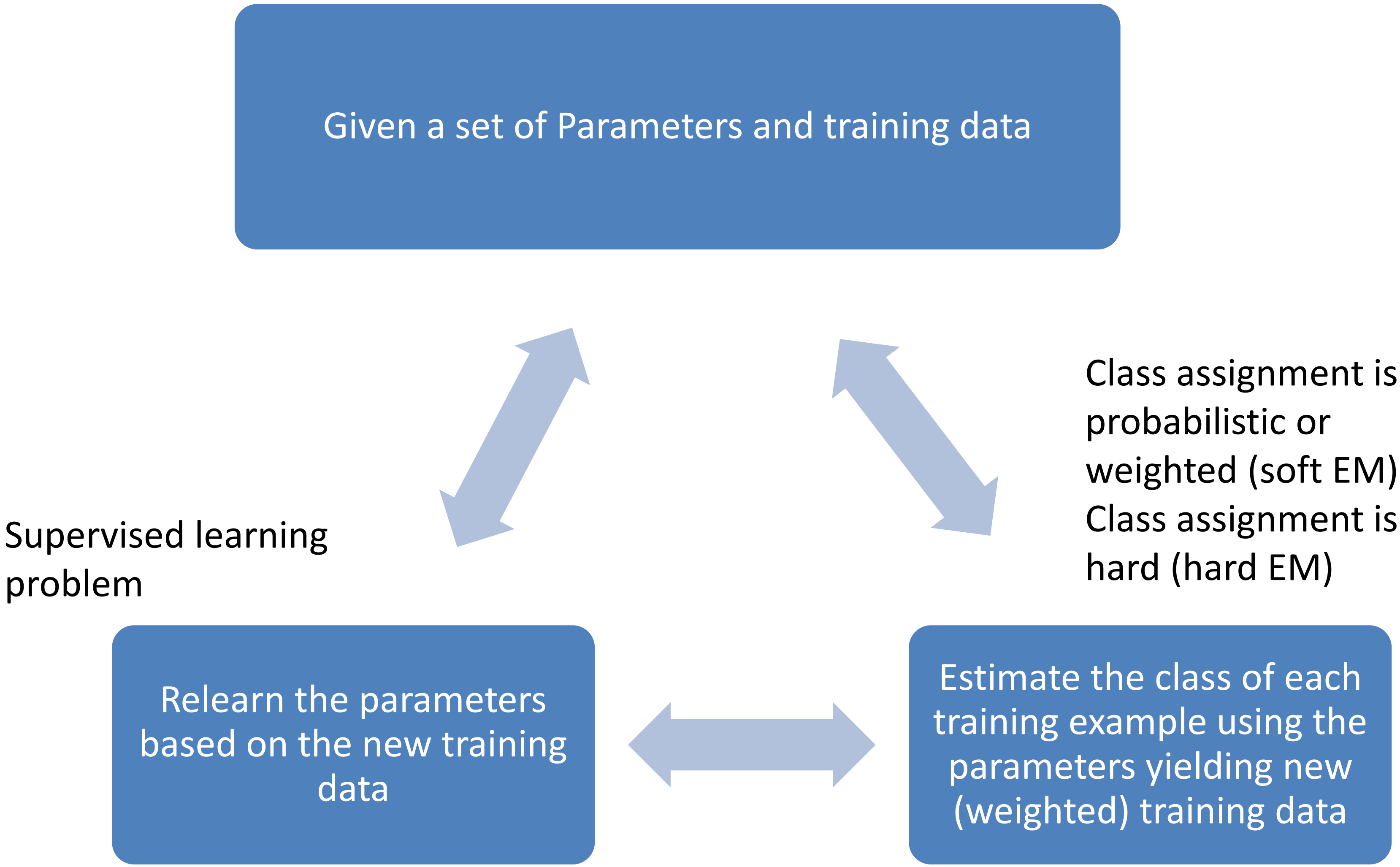


Expectation
Maximization

The EM Algorithm

- A clever method for maximizing marginal likelihood:
  - $\text{argmax}_\theta \prod_j P(x_j) = \text{argmax}_\theta \prod_j \sum_{i=1}^{k} P(y_j=i,x_j)$
  - 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 algorithm: Pictorial View



EM: Two Easy Steps

Objective: $\text{argmax}_\theta \prod_j \sum_{i=1}^{k} P(y_j=i,x_j|\theta) = \sum_j \log \sum_{i=1}^{k} P(y_j=i,x_j|\theta)$

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 i for y, compute: $P(y_j=i \mid x_j, \theta)$

- **M-step**: Re-estimate the parameters with "weighted" MLE estimates
  - Set $\theta = \text{argmax}_\theta \sum_j \sum_i P(y_j=i \mid x_j, \theta) \log P(y_j=i,x_j|\theta)$

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

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

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

M-step

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

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

E.M. for General GMMs

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

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

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

E-step

Compute "expected" classes of all datapoints for each class

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

Just evaluate a Gaussian at x_j

M-step

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

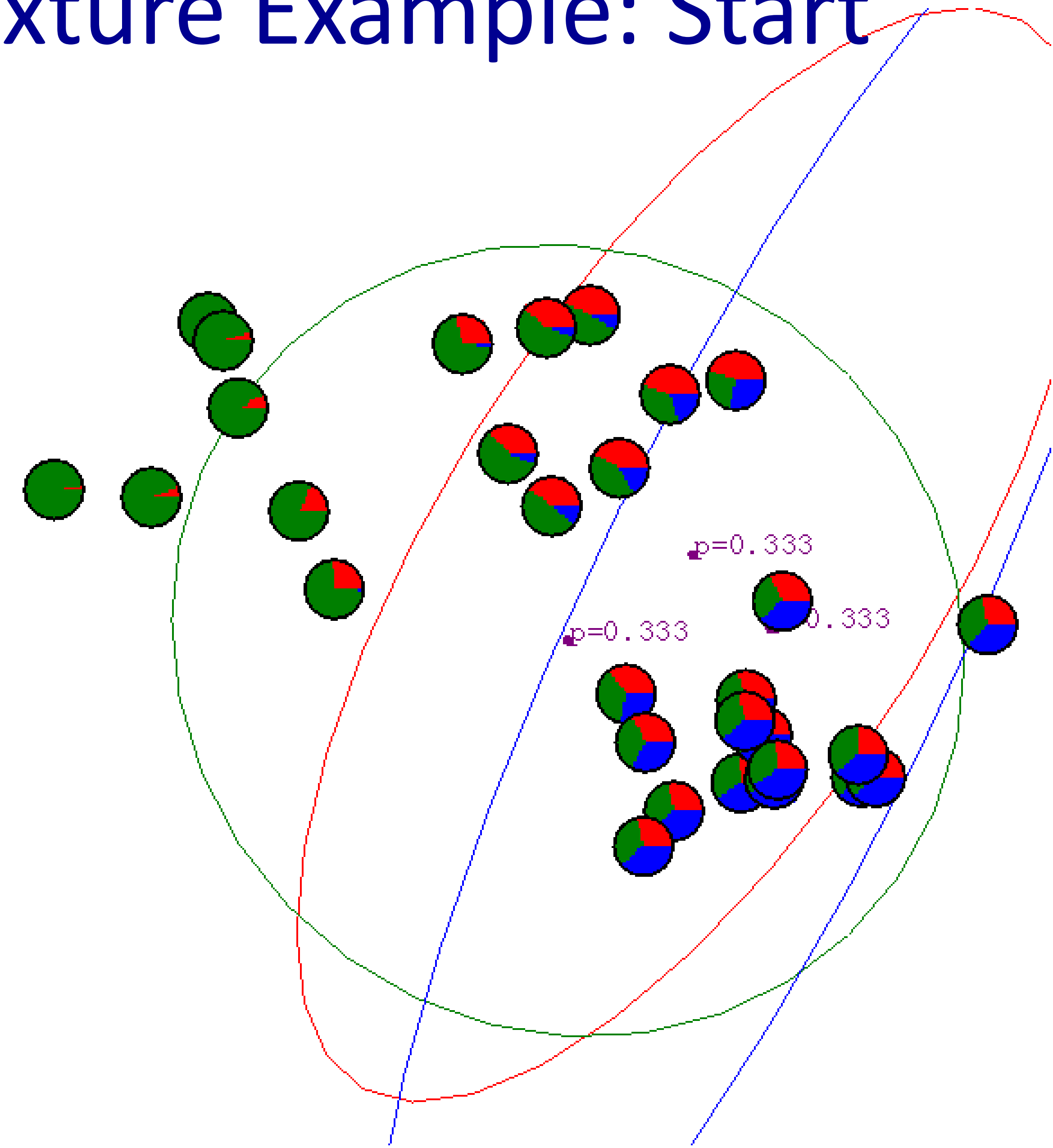
$$\mu_i^{(t+1)} = \frac{\sum_j \mathrm{P}\left(y=i \middle| x_j, \lambda_t\right) x_j}{\sum_j \mathrm{P}\left(y=i \middle| x_j, \lambda_t\right)}$$

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

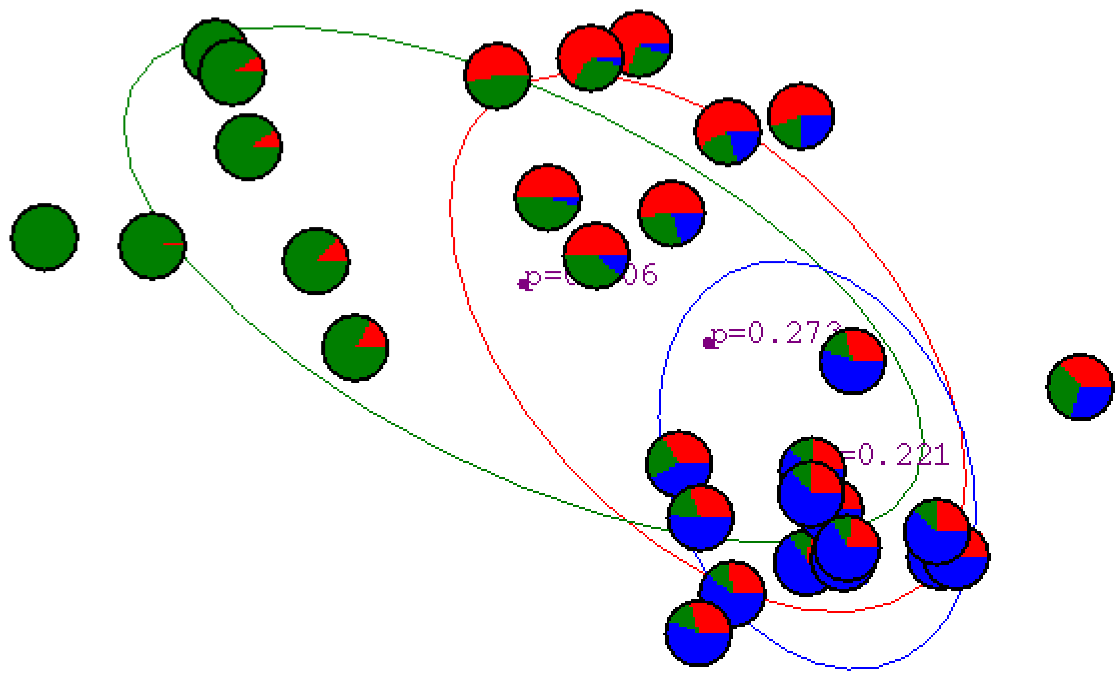
$$p_i^{(t+1)} = \frac{\sum_j \mathrm{P}\left(y=i \middle| x_j, \lambda_t\right)}{m}$$

m = #training examples

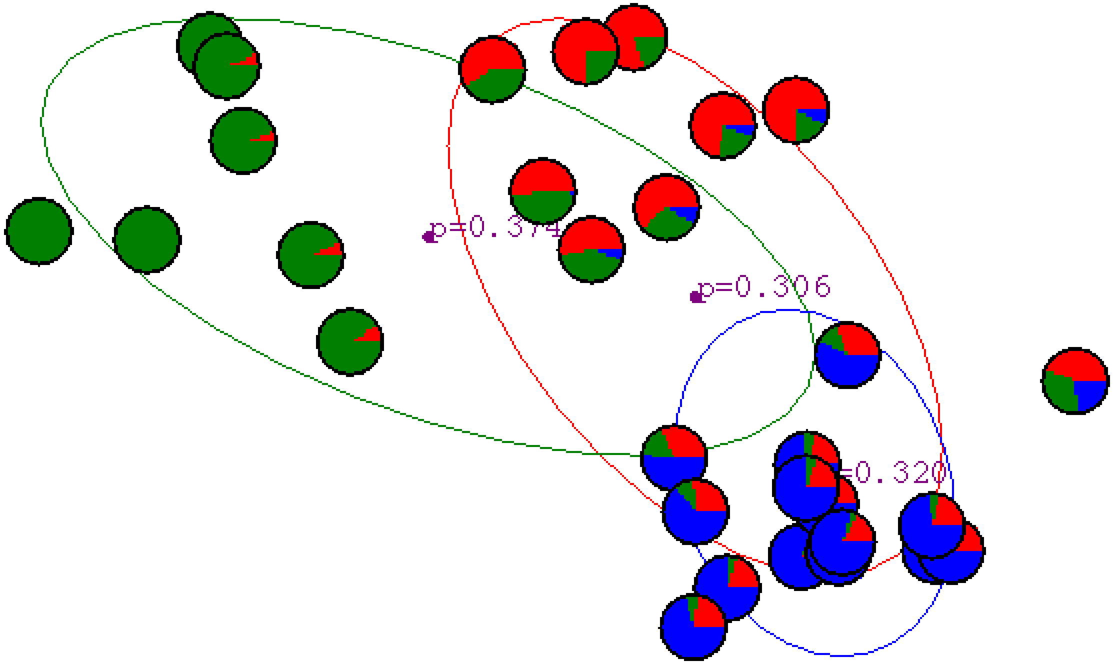
Gaussian Mixture Example: Start



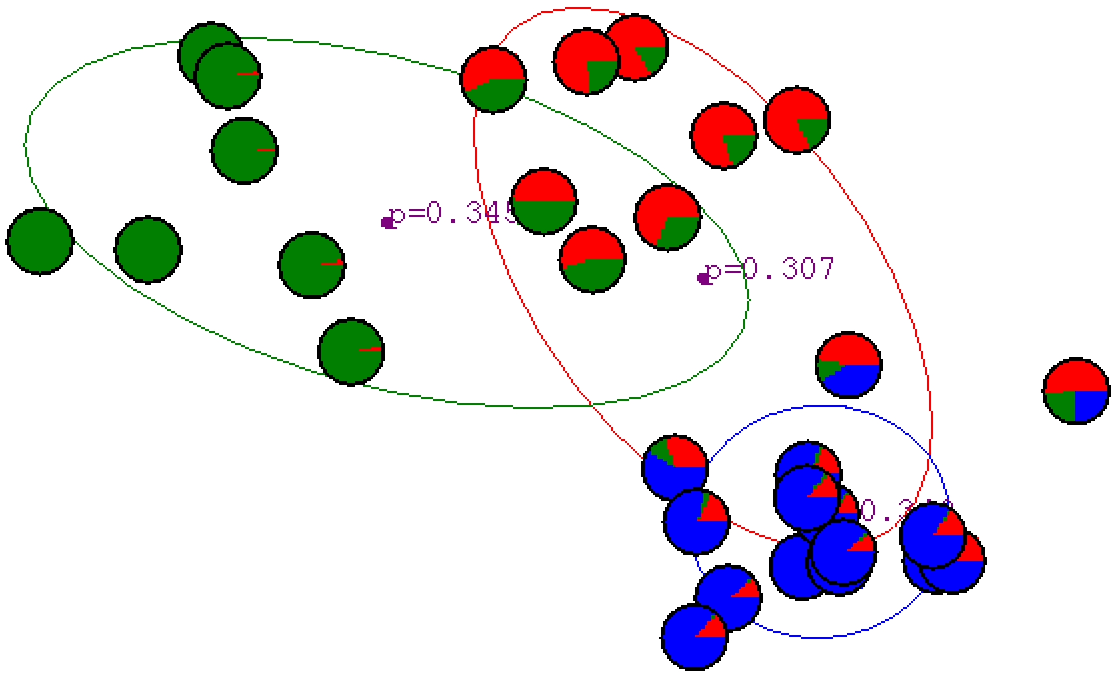
After first iteration



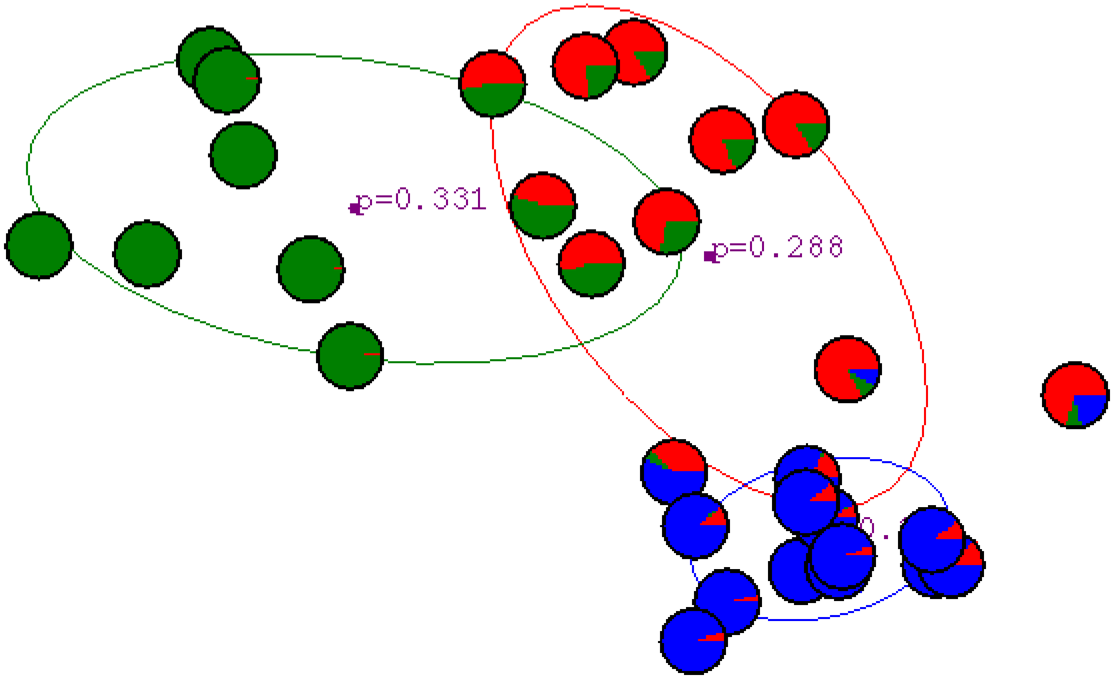
After 2nd iteration



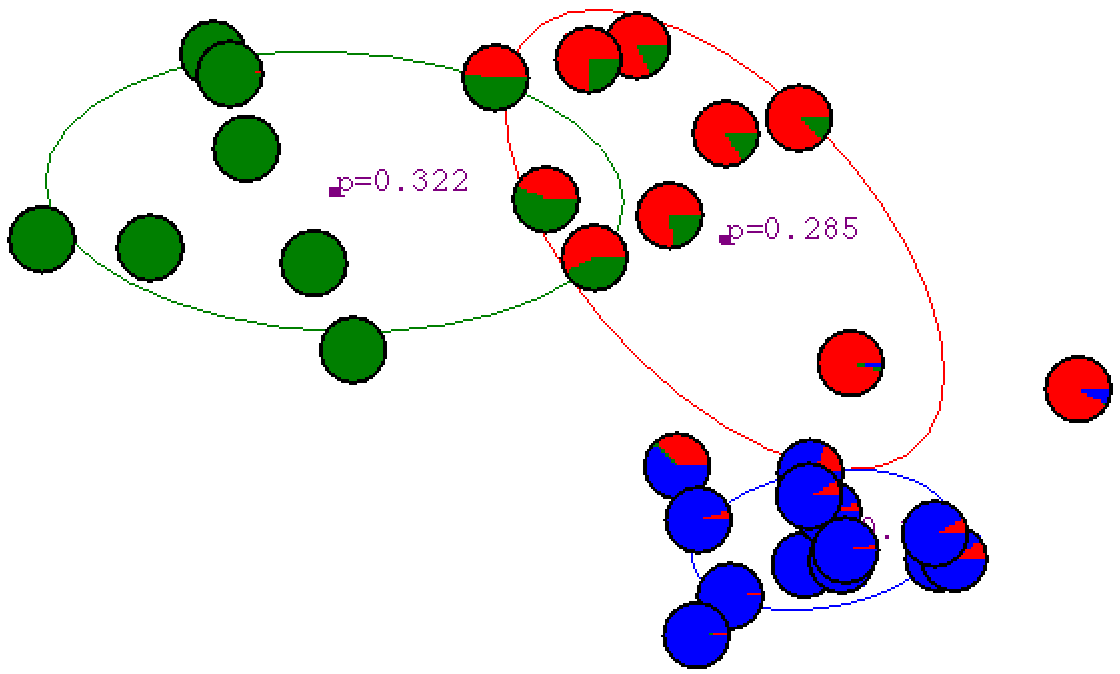
After 3rd iteration



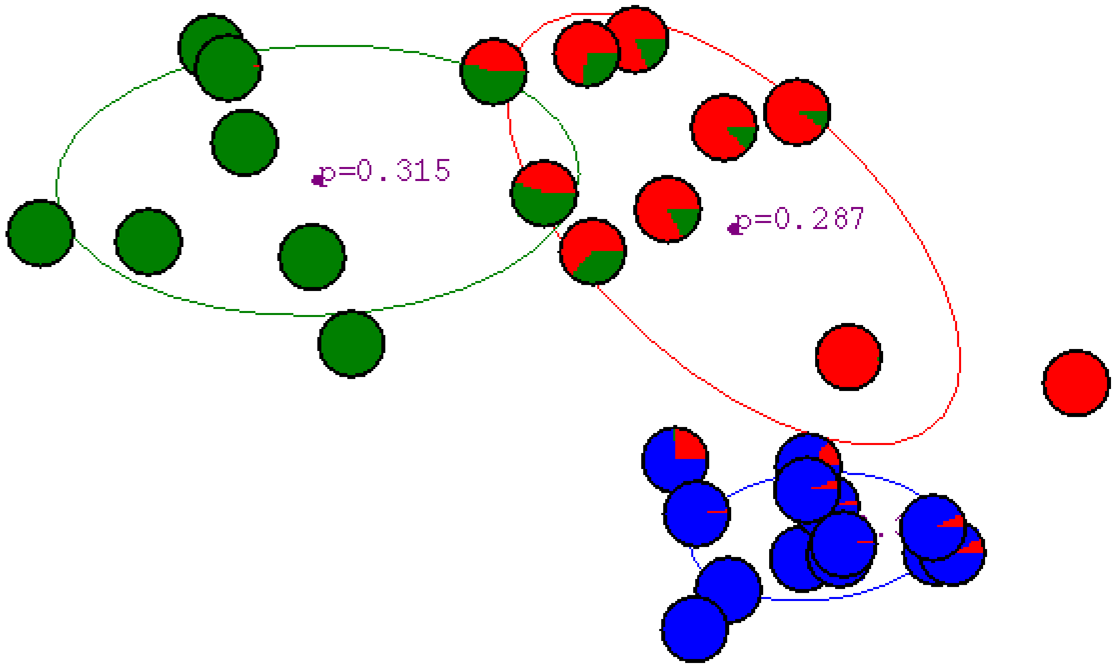
After 4th iteration



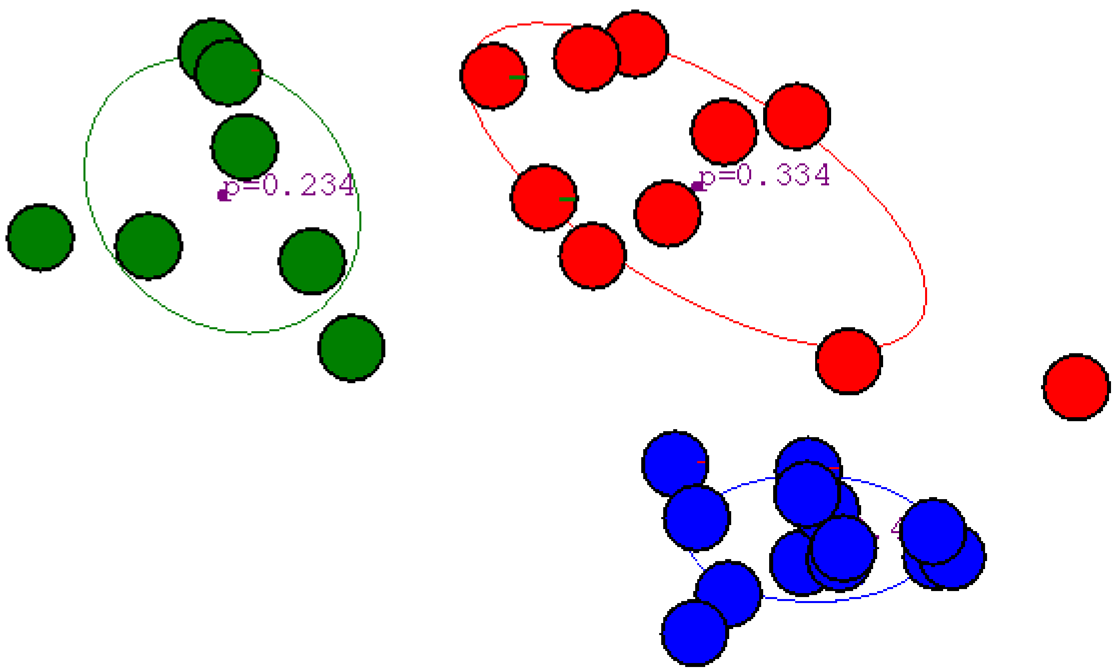
After 5th iteration



After 6th iteration



After 20th iteration



What if we do hard assignments?

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

$$\theta_t = \{ \mu_1^{(t)}, \mu_2^{(t)} \ldots \mu_k^{(t)} \}$$

E-step

Compute "expected" classes of all datapoints

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

M-step

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

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

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

$$_i = \frac{\left(y=i, x_j\right) x_j}{{}^{m}_{j=1} \left(y=i, x_j\right)}$$

Equivalent to k-means clustering algorithm!!!

What you should know

- K-means for clustering:
  - algorithm
  - converges because it's coordinate ascent
- Know what agglomerative clustering is
- EM for mixture of Gaussians:
  - Also coordinate ascent
  - How to "learn" maximum likelihood parameters (locally max. like.) in the case of unlabeled data
  - Relation to K-means
    - Hard / soft clustering
    - Probabilistic model
- Remember, E.M. can get stuck in local minima,
  - And empirically it ***DOES***

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