Unsupervised Learning

Instructor: Alan Ritter

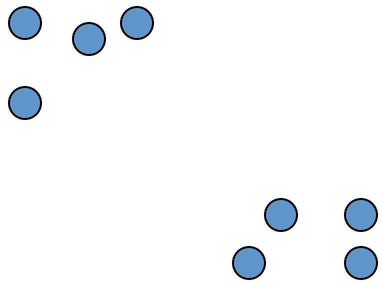
Slides Adapted from Carlos Guestrin Dan Klein and Luke Zettlemoyer

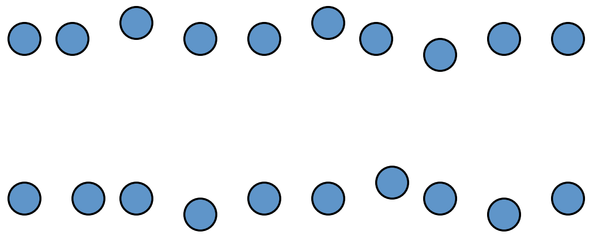
Clustering

- Clustering systems:
  - Unsupervised learning
  - Detect patterns in unlabeled data
    - E.g. group emails or search results
    - E.g. find categories of customers
    - E.g. detect anomalous program executions
  - Useful when don't know what you're looking for
  - Requires data, but no labels
  - Often get gibberish

Clustering

- Basic idea: group together similar instances
- Example: 2D point patterns



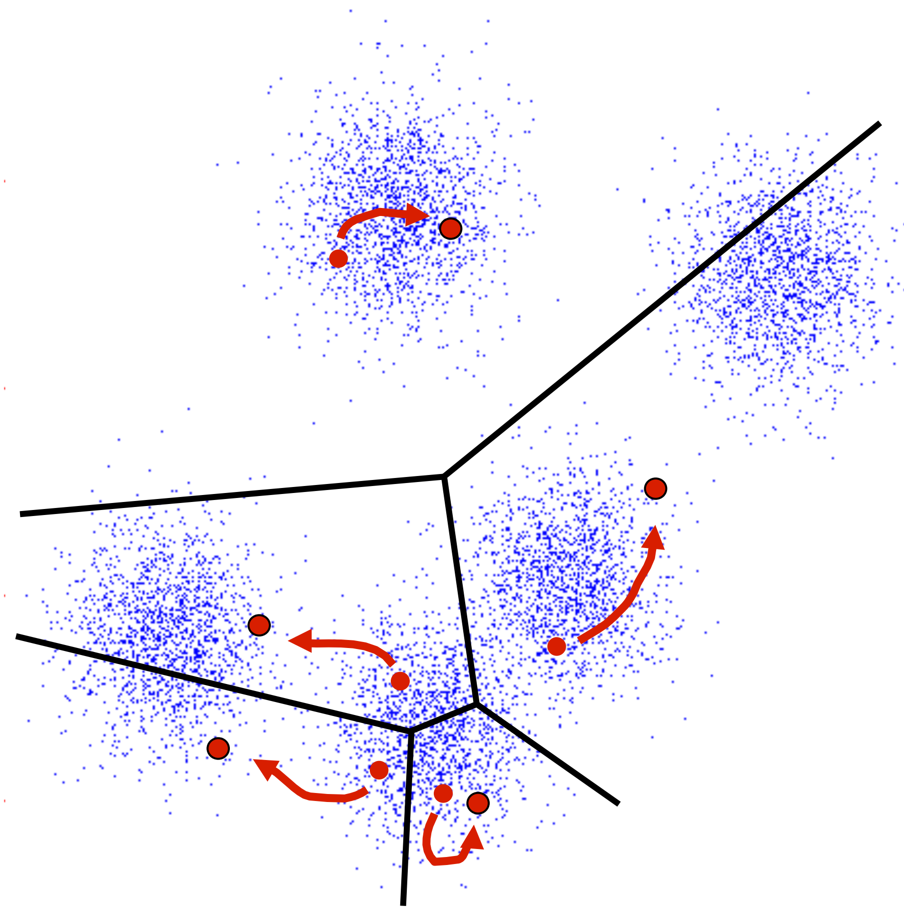


- What could "similar" mean?
  - One option: small (squared) Euclidean distance

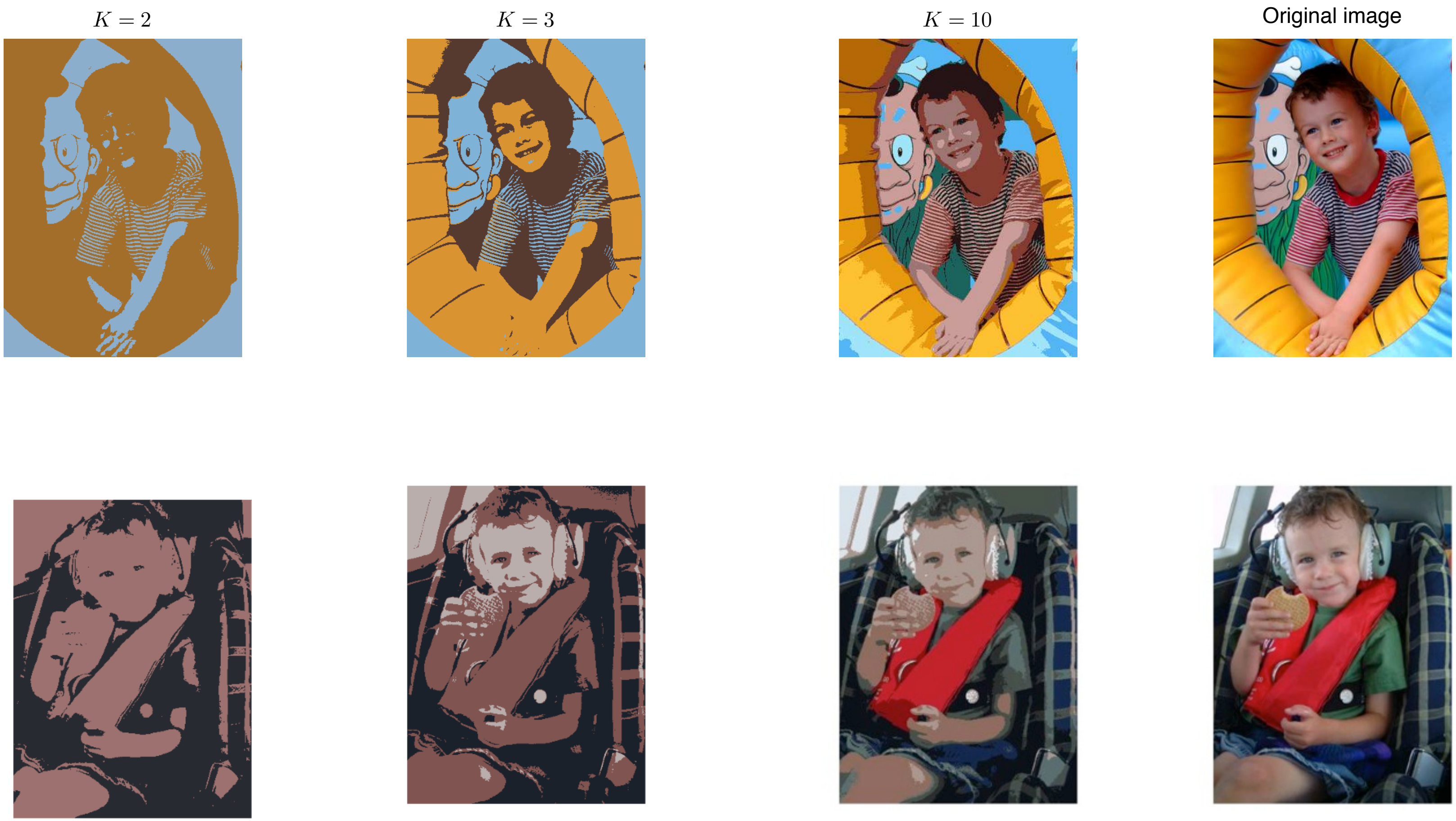
$$dist(x, x') = \|x - x'\|_2^2 = (x - x')^T (x - x') = \sum_i (x_i - x'_i)^2$$

K-Means

- An iterative clustering algorithm
  - Pick K random points as cluster centers (means), $c^1 \ldots c^k$
  - Alternate:
    - Assign each example x^i to the mean c^j that is closest to it
    - Set each mean c^j to the average of its assigned points
  - Stop when no points' assignments change



Example: K-Means for Segmentation



K-Means

- Data: $\{x^j \mid j=1..n\}$
- An iterative clustering algorithm
  - Pick K random cluster centers, $c^1 \ldots c^k$
  - For t=1..T: [or, stop if assignments don't change]
    - for j = 1.. n: [recompute cluster assignments]

$$a^j = \arg\min_i dist(x^j, c^i)$$

    - for j= 1…k: [recompute cluster centers]

$$c^j = \frac{1}{|\{i|a^i = j\}|} \sum_{\{i|a^i=j\}} x^i$$

Pick K random cluster centers, $c^1 \ldots c^k$
For t=1..T:

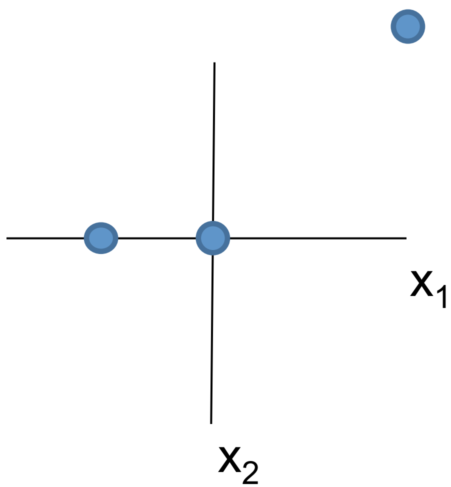
- for j = 1.. n: [recompute assignments]

$$a^j = \arg\min_i dist(x^j, c^i)$$

- for j= 1…k: [recompute cluster centers]

$$c^j = \frac{1}{|\{i|a^i = j\}|} \sum_{\{i|a^i=j\}} x^i$$

x_1	x_2
-1	0
0	0
2	2



$$dist(x, x') = \sum_i (x_i - x'_i)^2$$

Random cluster means:

- $c^1=[-1,0]$, $c^2=[0,0]$

t=0:

$d(x^j,c^i)$	x^1	x^2	x^3
c^1	0	1	13
c^2	1	0	8

- $a^1 = \text{argmin}_i\, dist(x^1,c^i) = 1$
- $a^2 = \text{argmin}_i\, dist(x^2,c^i) = 2$
- $a^3 = \text{argmin}_i\, dist(x^3,c^i) = 2$
- $c^1 = (1/1) * [-1,0] = [-1,0]$
- $c^2 = (1/2) * ([0,0]+[2,2]) = [1,1]$

t=1:

$d(x^j,c^i)$	x^1	x^2	x^3
c^1	0	1	13
c^2	4	4	18

- $a^1 = \text{argmin}_i\, dist(x^1,c^i) = 1$
- $a^2 = \text{argmin}_i\, dist(x^2,c^i) = 1$
- $a^3 = \text{argmin}_i\, dist(x^3,c^i) = 2$
- $c^1 = (1/2) * ([-1,0]+[0,0]) = [-0.5,0]$
- $c^2 = (1/1) * ([2,2]) = [2,1]$

t=2:

- Stop!! (cluster assignments a^i won't change in next round; you can verify!)

K-Means as Optimization

- Consider the total distance to the means:

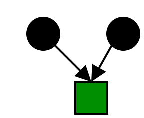
$$L(\{x^i\},\{a^j\},\{c^k\}) = \sum_i dist(x^i, c^{a^i})$$

points

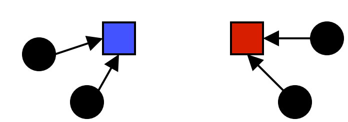
assignments

means

- Two stages each iteration:
  – Update assignments: fix means c,
    change assignments a



  – Update means: fix assignments a,
    change means c



- *Coordinate gradient descent* on L
- Will it converge?
  – Yes!, if you can argue that each update can't increase L

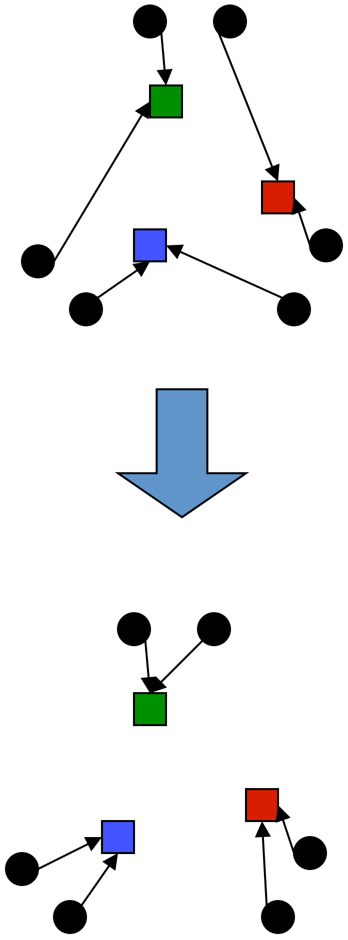
Phase I: Update Assignments

- For each point, re-assign to closest mean:

$$a^j = \arg\min_i dist(x^j, c^i)$$

- Can only decrease total distance L!

$$L(\{x^i\}, \{a^j\}, \{c^k\}) = \sum_i dist(x^i, c^{a^i})$$



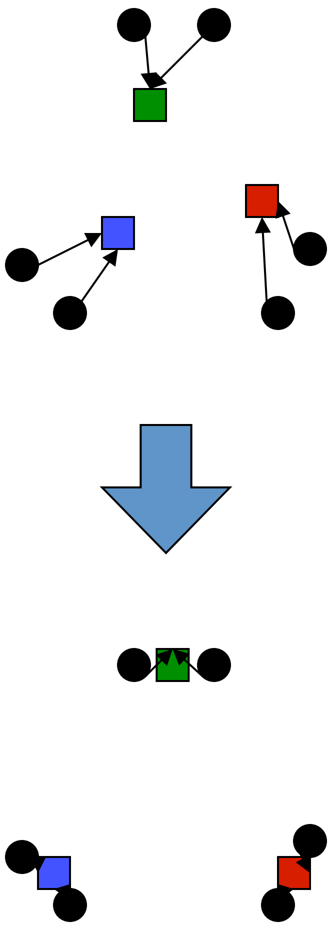
Phase II: Update Means

- Move each mean to the average of its assigned points:

$$c^j = \frac{1}{|\{i|a^i = j\}|} \sum_{\{i|a^i=j\}} x^i$$

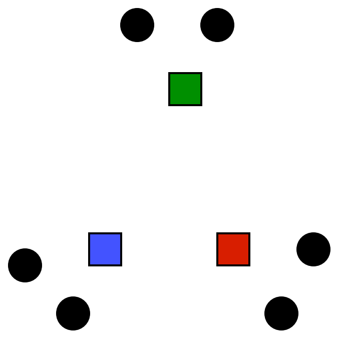
- Also can only decrease total distance… (Why?)

- Fun fact: the point y with minimum squared Euclidean distance to a set of points {x} is their mean

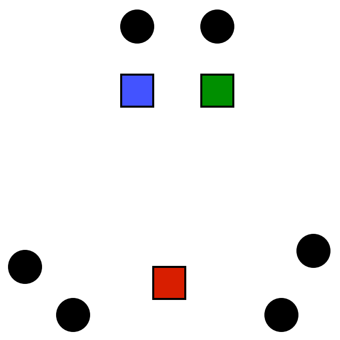


Initialization

- K-means is non-deterministic
  - Requires initial means
  - It does matter what you pick!

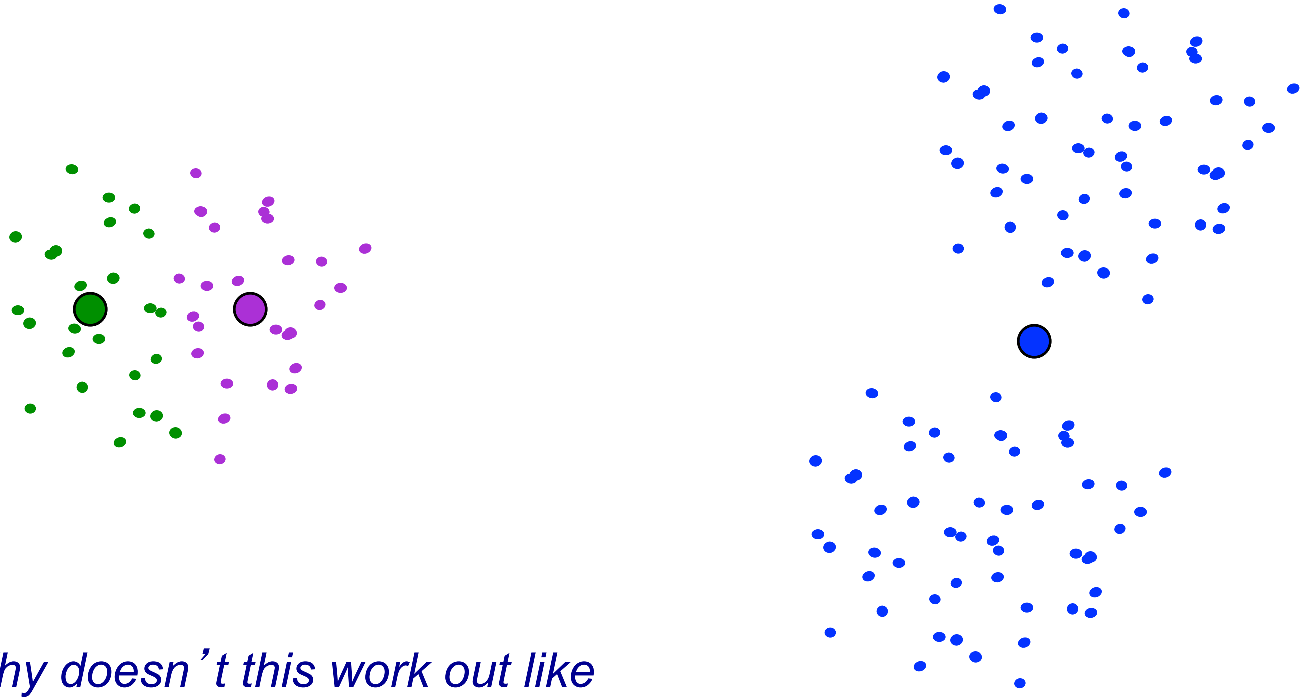


  - What can go wrong?
  - Various schemes for preventing this kind of thing: variance-based split / merge, initialization heuristics



K-Means Getting Stuck

- A local optimum:



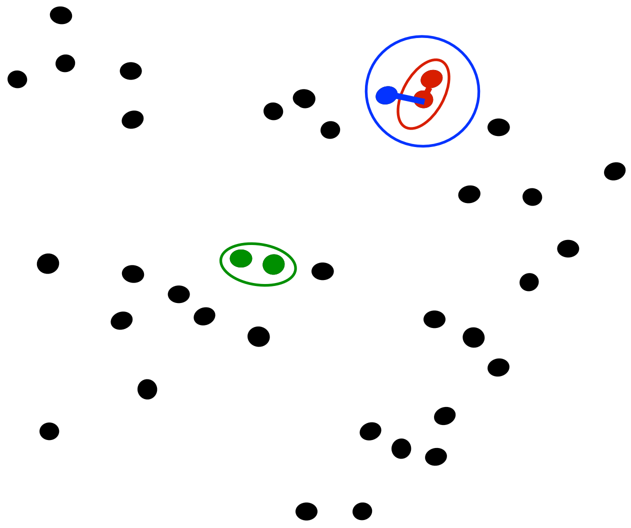
Why doesn't this work out like the earlier example, with the purple taking over half the blue?

K-Means Questions

- Will K-means converge?
  - To a global optimum?
- Will it always find the true patterns in the data?
  - If the patterns are very very clear?
- Will it find something interesting?
- Do people ever use it?
- How many clusters to pick?

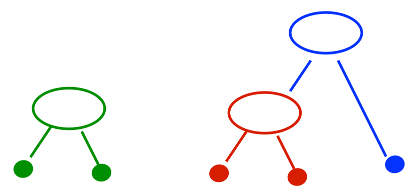
Agglomerative Clustering

- Agglomerative clustering:
  - First merge very similar instances
  - Incrementally build larger clusters out of smaller clusters



- Algorithm:
  - Maintain a set of clusters
  - Initially, each instance in its own cluster
  - Repeat:
    - Pick the two closest clusters
    - Merge them into a new cluster
    - Stop when there's only one cluster left

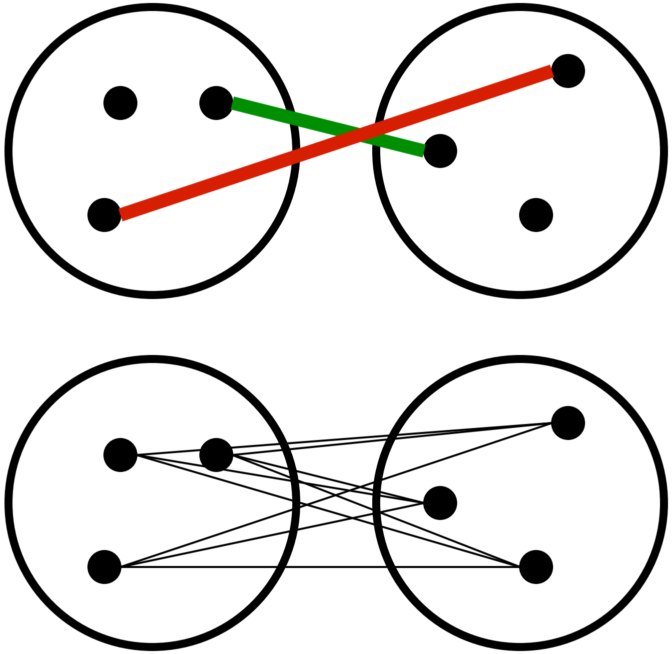
- Produces not one clustering, but a family of clusterings represented by a dendrogram



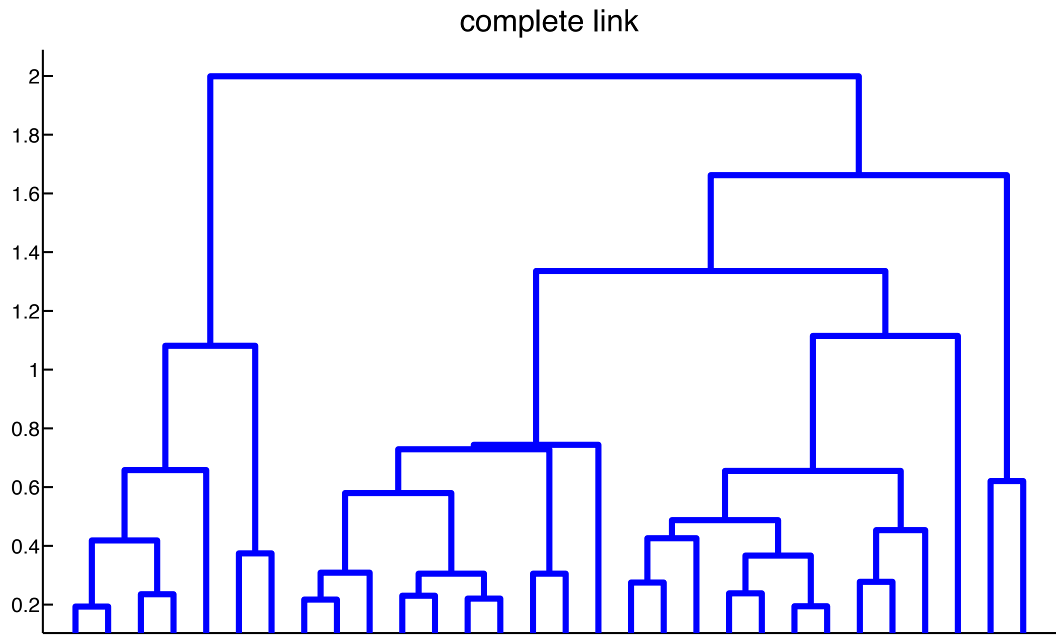
Agglomerative Clustering

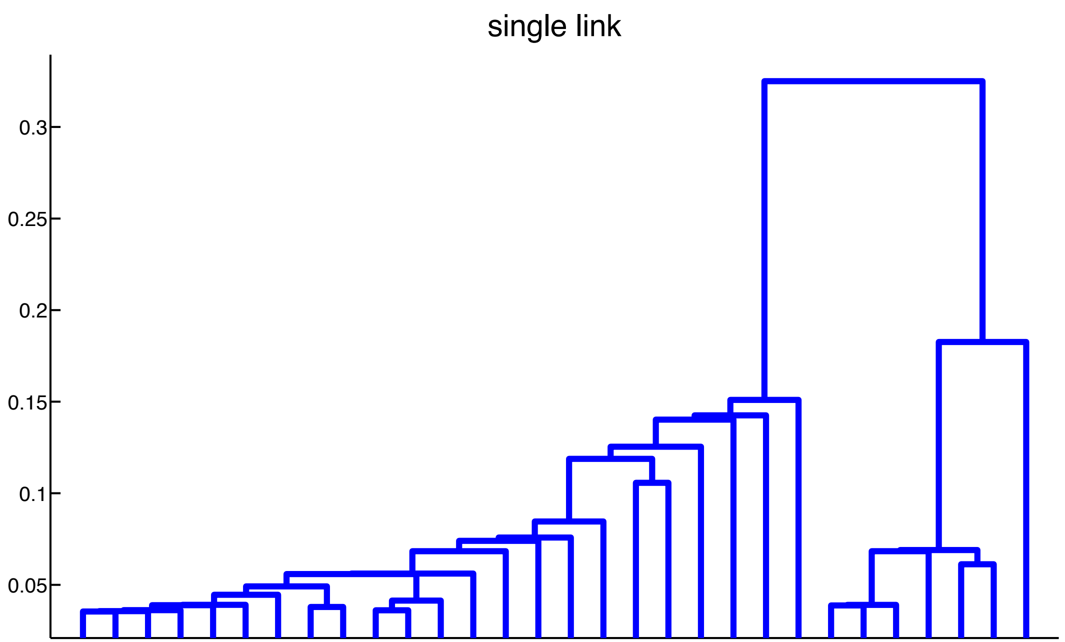
- How should we define "closest" for clusters with multiple elements?

- Many options:
  - Closest pair (single-link clustering)
  - Farthest pair (complete-link clustering)
  - Average of all pairs

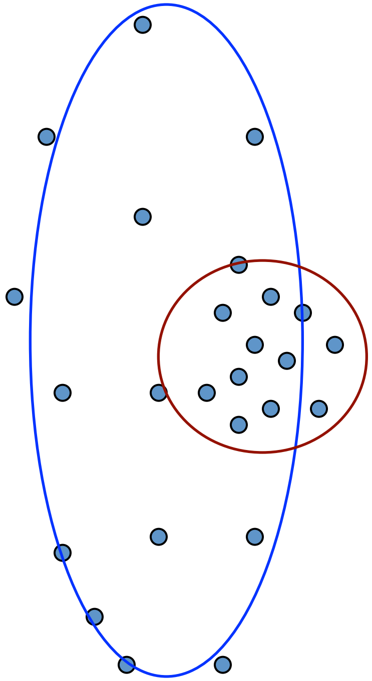


- Different choices create different clustering behaviors


complete link
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2

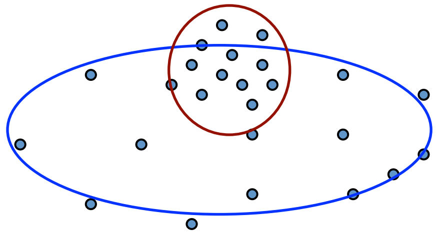

single link
0.3
0.25
0.2
0.15
0.1
0.05

(One) bad case for “hard assignments”?



- Clusters may overlap
- Some clusters may be “wider” than others

Probabilistic Clustering



- We can use a probabilistic model!
  - allows overlaps, clusters of different size, etc.
- Can tell a *generative story* for data
  - P(X|Y) P(Y) is common
- 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

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

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

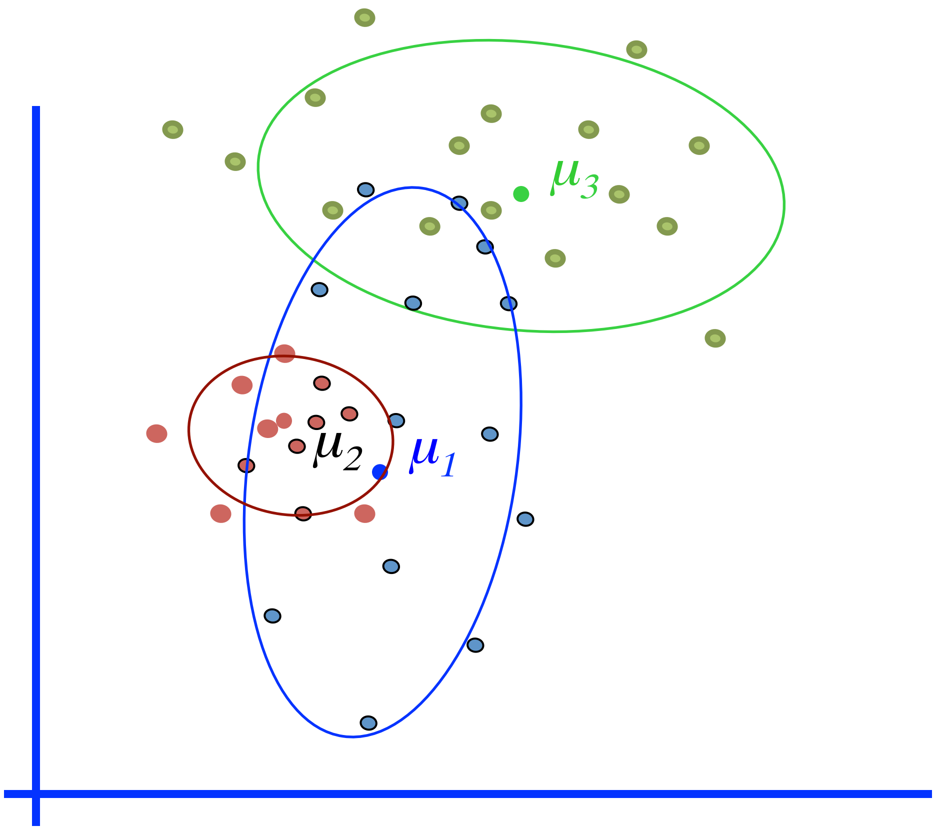
Y	X_1	X_2
??	0.1	2.1
??	0.5	-1.1
??	0.0	3.0
??	-0.1	-2.0
??	0.2	1.5
…	…	…

The General GMM assumption

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

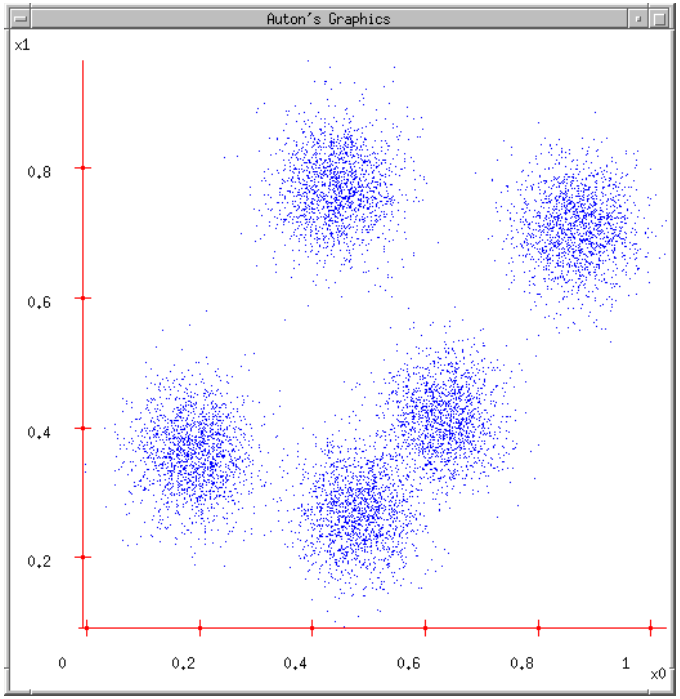
Each data point is sampled from a *generative process*:

1. Pick a component at random: Choose component i with probability *P(y=i)*
2. Datapoint ~ N(μ_i, Σ_i)

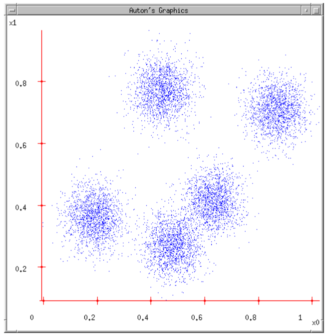


That was easy! Now, lets estimate parameters!

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



How do we optimize? Closed Form?

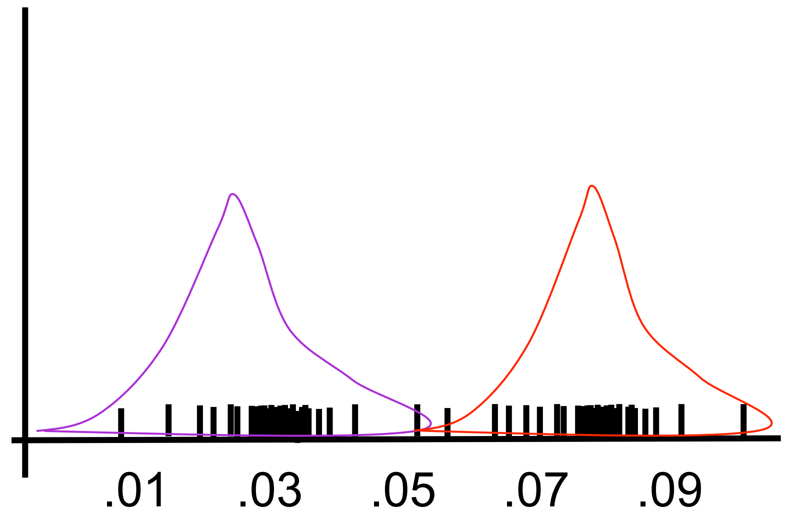


- Maximize *marginal likelihood*:
  - $\text{argmax}_\theta \prod_j P(x^j;\theta) = \text{argmax} \prod_j \sum_{i=1}^{k} P(y^j=i,x^j;\theta)$
- Almost always a hard problem!
  - Usually no closed form solution
  - Even when $P(X,Y;\theta)$ is convex, $P(X;\theta)$ generally isn't…
  - For all but the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

Simple example: learn means only!

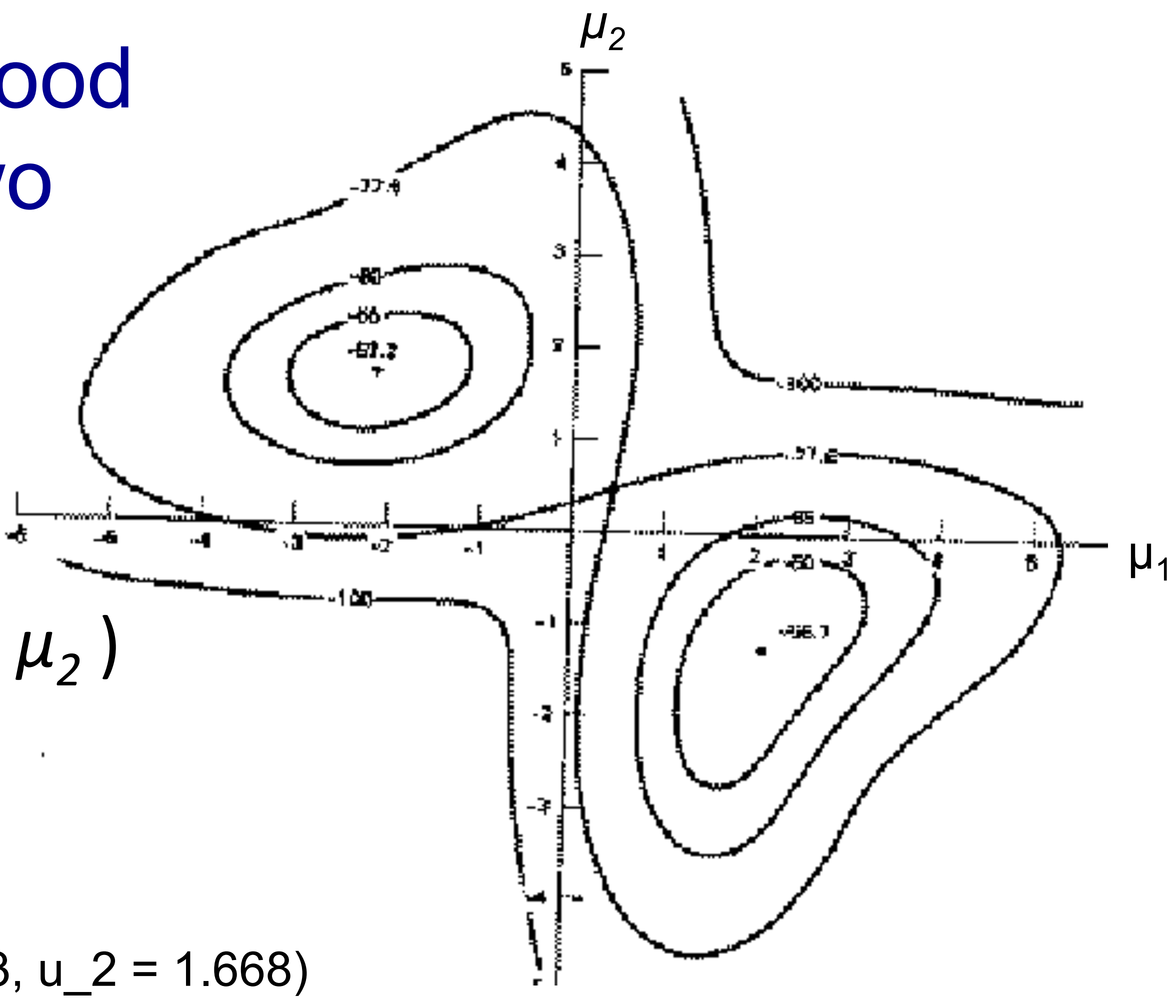
Consider:

- 1D data, m points
- Mixture of k=2 Gaussians
- Variances fixed to σ=1
- Dist'n over classes is uniform
- Need to estimate μ_1 and μ_2



$$\prod_{j=1}^{n}\sum_{i=1}^{k} P(X = x^j, Y = i) \propto \prod_{j=1}^{n}\sum_{i=1}^{k} \exp\left(-\frac{1}{2\sigma^2}(x^j - \mu_i)^2\right)$$

Marginal Likelihood for Mixture of two Gaussians



Graph of

$\log P(x^1, x^2 .. x^m \mid \mu_1, \mu_2)$

against μ_1 and μ_2

Max likelihood = (u_1 = 2.13, u_2 = 1.668)

Local minimum, but very close to global at (u_1 =1.668, u_2 = 2.13) *

* corresponds to switching y_1 with y_2.

Learning general mixtures of Gaussian

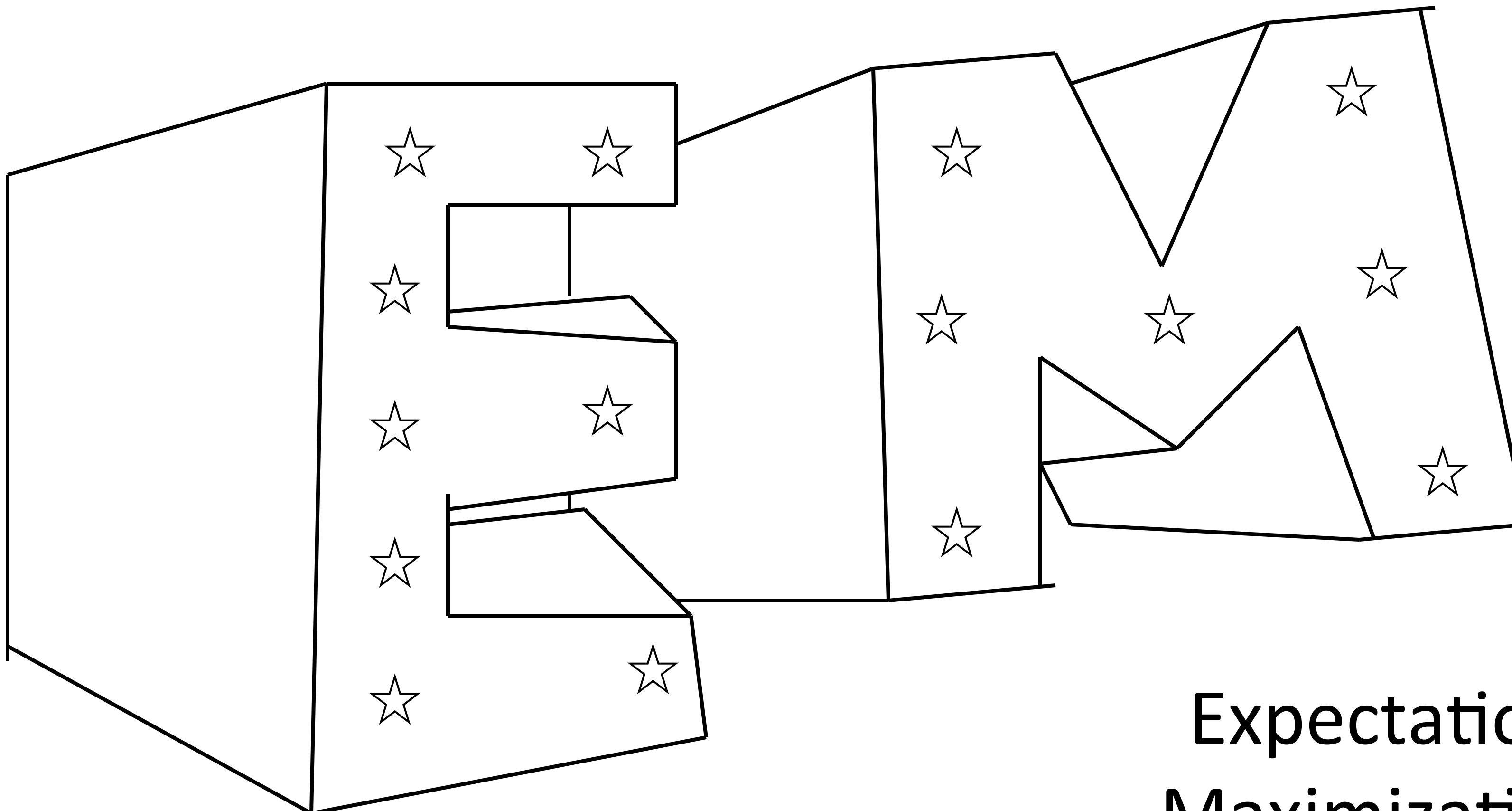
$$P(X = x|Y = i) = \frac{1}{\sqrt{(2\pi)^m|\Sigma_i|}} \exp\left(-\frac{1}{2}(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i)\right)$$

- Marginal likelihood, for data $\{x^j \mid j = 1..n\}$:

$$\prod_{j=1}^{n} P(x^j) = \prod_{j=1}^{n}\sum_i P(X = x^j, Y = i) = \prod_{j=1}^{n}\sum_i P(X = x^j|Y = i)P(Y = i)$$

$$= \prod_{j=1}^{n}\sum_i \frac{1}{\sqrt{(2\pi)^m|\Sigma_i|}} \exp\left(-\frac{1}{2}(x^j-\mu_i)^T\Sigma_i^{-1}(x^j-\mu_i)\right) P(Y = i)$$

- Need to differentiate and solve for μ_i, Σ_i, and P(Y=i) for i=1..k
- There will be no closed for 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: 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: $\{x^j \mid j=1 \,..\, n\}$

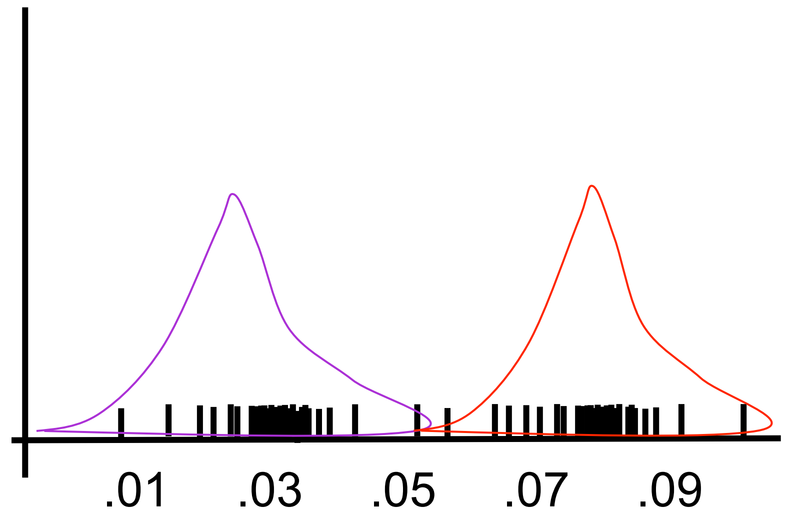
- **E-step**: Compute expectations to "fill in" missing y values according to current parameters
  - For all examples j and values i for y, compute: $P(y^j=i \mid x^j, \theta)$
- **M-step**: Re-estimate the parameters with "weighted" MLE estimates
  - Set $\theta = \text{argmax}_\theta \sum_j \sum_{i=1}^k P(y^j=i \mid x^j, \theta) \log P(y^j=i,x^j|\theta)$

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

Simple example: learn means only!

Consider:

- 1D data, m points
- Mixture of k=2 Gaussians
- Variances fixed to σ=1
- Dist'n over classes is uniform
- Need to estimate μ_1 and μ_2



$$\prod_{j=1}^{n} \sum_{i=1}^{k} P(X = x^j, Y = i) \propto \prod_{j=1}^{n} \sum_{i=1}^{k} \exp\left(-\frac{1}{2\sigma^2}(x^j - \mu_i)^2\right)$$

EM for GMMs: only learning means

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

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

E-step

Compute "expected" classes of all datapoints

$$p(y = i | x^j; \theta_t) \propto \exp\left(-\frac{1}{2\sigma^2} (x^j - \mu_i)^2 \right)$$

M-step

Compute most likely new **μ**s given class expectations, by doing weighted ML estimates:

$$\mu_i = \frac{\sum_{j=1}^{m} p(y = i | x^j; \theta_t) x^j}{\sum_{j=1}^{m} p(y = i | x^j; \theta_t)}$$

Pick K random cluster centers, $\mu_1 \ldots \mu_k$
For t=1..T:

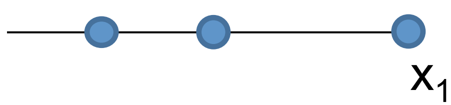
- E step:

$$p(y=i|x^j;\theta_t) \propto \exp\left(-\frac{1}{2\sigma^2}(x^j-\mu_i)^2\right)$$

- M step:

$$\mu_i = \frac{\sum_{j=1}^m p(y=i|x^j;\theta_t)x^j}{\sum_{j=1}^m p(y=i|x^j;\theta_t)}$$

x_1
-1
0
2



Initialization, random means and σ=1:

- $\mu_1=-1$, $\mu_2=0$

t=0:

- $P(y=1|x^1)\ \alpha\ \exp(-0.5\times(-1+1)^2) = 1$
- $P(y=2|x^1)\ \alpha\ \exp(-0.5\times(-1-0)^2) = 0.6$
  - $P(y=1|x^1) = 0.63$, $P(y=2|x^1)=0.37$
- $P(y=1|x^2)\ \alpha\ \exp(-0.5\times(0+1)^2) = 0.6$
- $P(y=2|x^2)\ \alpha\ \exp(-0.5\times(0-0)^2) = 1$
  - $P(y=1|x^2) = 0.37$, $P(y=2|x^2)=0.63$
- $P(y=1|x^3)\ \alpha\ \exp(-0.5\times(2+1)^2) = 0.07$
- $P(y=2|x^3)\ \alpha\ \exp(-0.5\times(2-0)^2) = 0.93$
  - $P(y=1|x^3) = 0.01$, $P(y=2|x^3)=0.93$
- $\mu^1 = (0.63\times-1 + 0.37\times0 + 0.07\times2) / (0.63 + 0.37 + 0.07) = -0.45$
- $\mu^2 = (0.37\times-1 + 0.67\times0 + 0.93\times2) / (0.37+0.67+0.93)=0.75$

t=1:

- learning continues, when do we stop?

E.M. for General GMMs

Iterate: On the *t*'th iteration let our estimates be, for y with k classes

$$\theta_t = \{ \mu_1 \ldots \mu_k, \Sigma_1 \ldots \Sigma_k, p_1, \ldots, p_k \}$$

E-step

Compute "expected" classes of all datapoints for each class

$$P(y=i|x^j;\theta_t) \propto \frac{1}{\sqrt{(2\pi)^m |\Sigma_i|}} \exp\left(-\frac{1}{2}(x^j-\mu_i)^T \Sigma_i^{-1} (x^j-\mu_i)\right) p_i$$

Evaluate a Gaussian at x^j

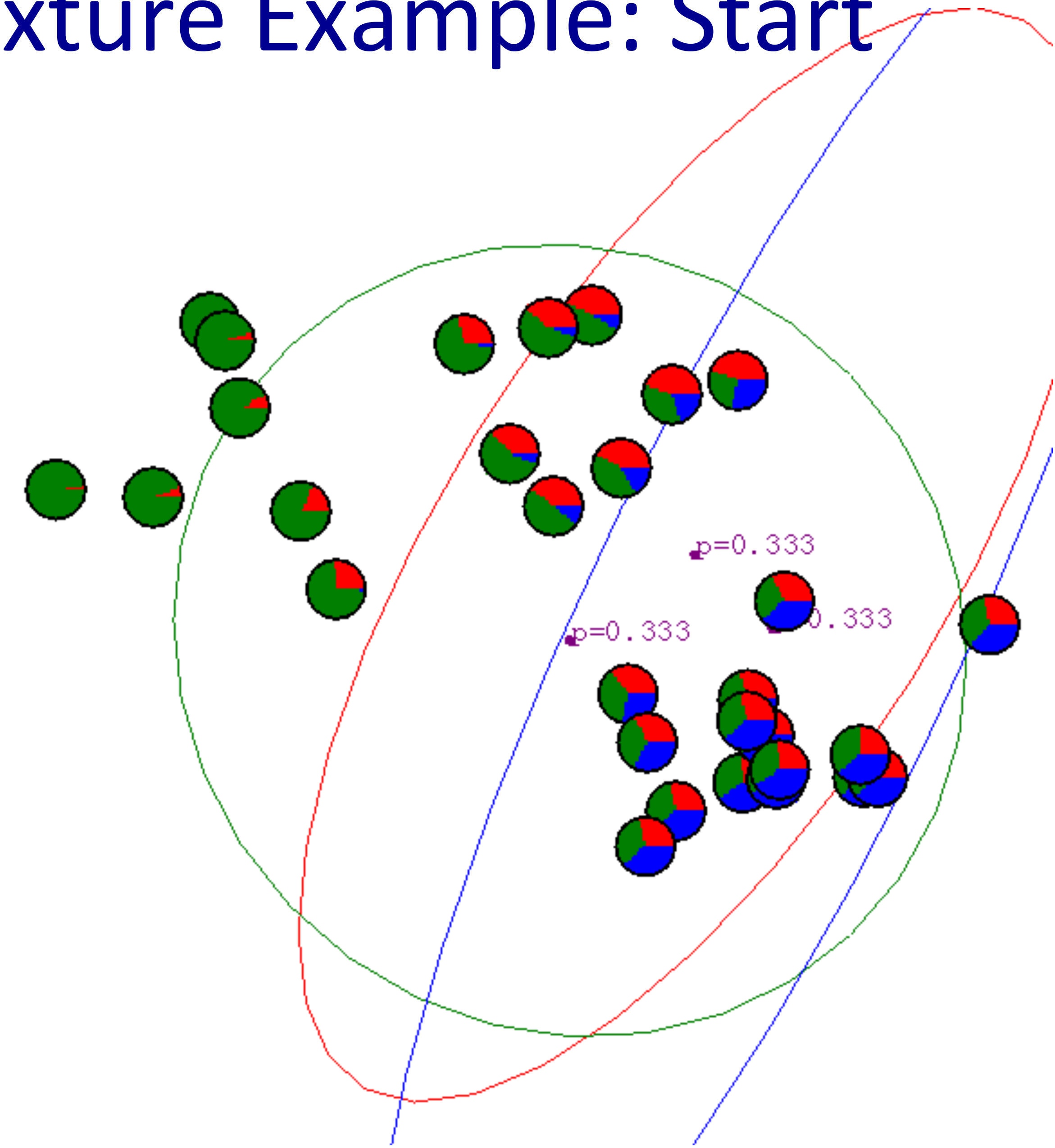
M-step

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

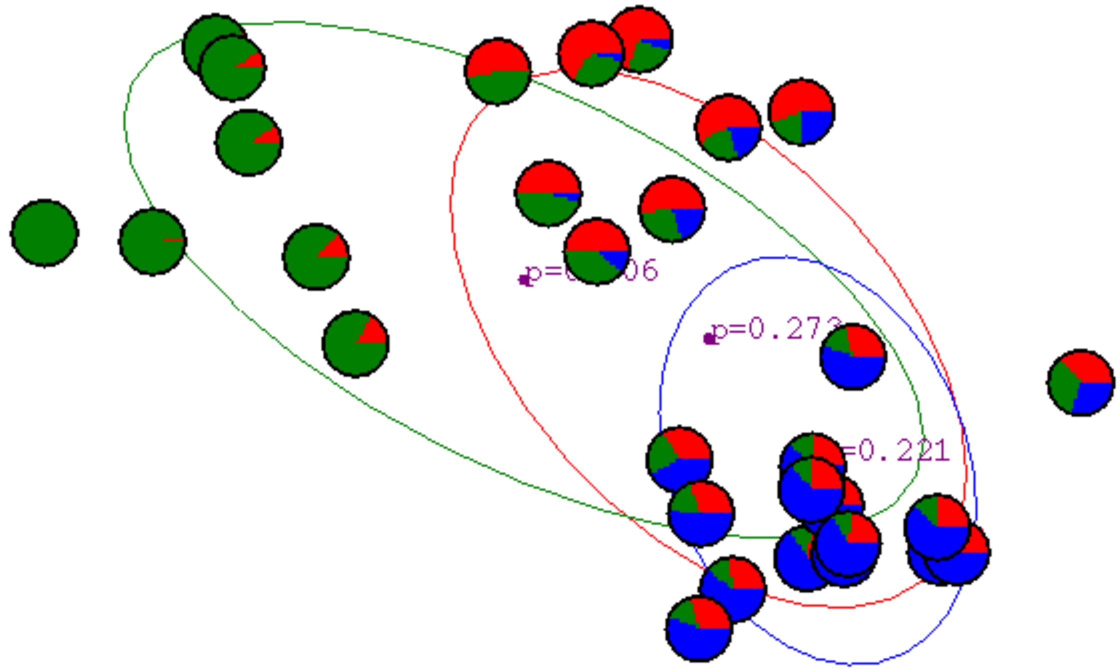
$$\mu_i = \frac{\sum_{j=1}^m p(y=i|x^j;\theta_t) x^j}{\sum_{j=1}^m p(y=i|x^j;\theta_t)} \qquad \Sigma_i = \frac{\sum_{j=1}^m p(y=i|x^j;\theta_t)(x^j-\mu_i)(x^j-\mu_i)^T}{\sum_{j=1}^m p(y=i|x^j;\theta_t)}$$

$$p_i = \frac{1}{m} \sum_{j=1}^m p(y=i|x^j;\theta_t)$$

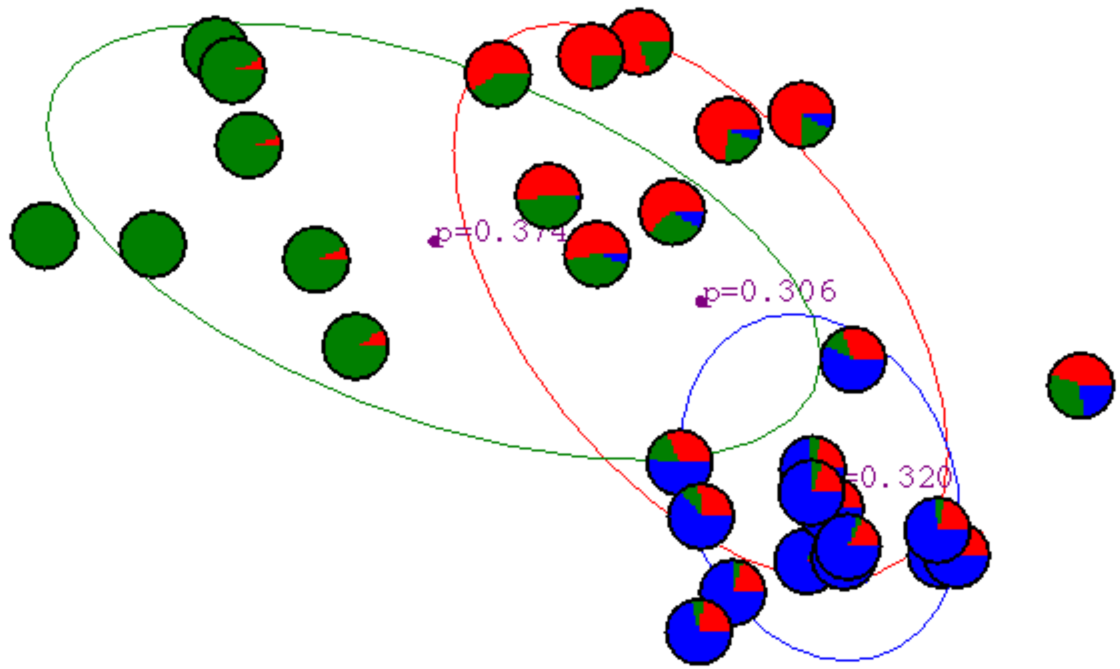
Gaussian Mixture Example: Start



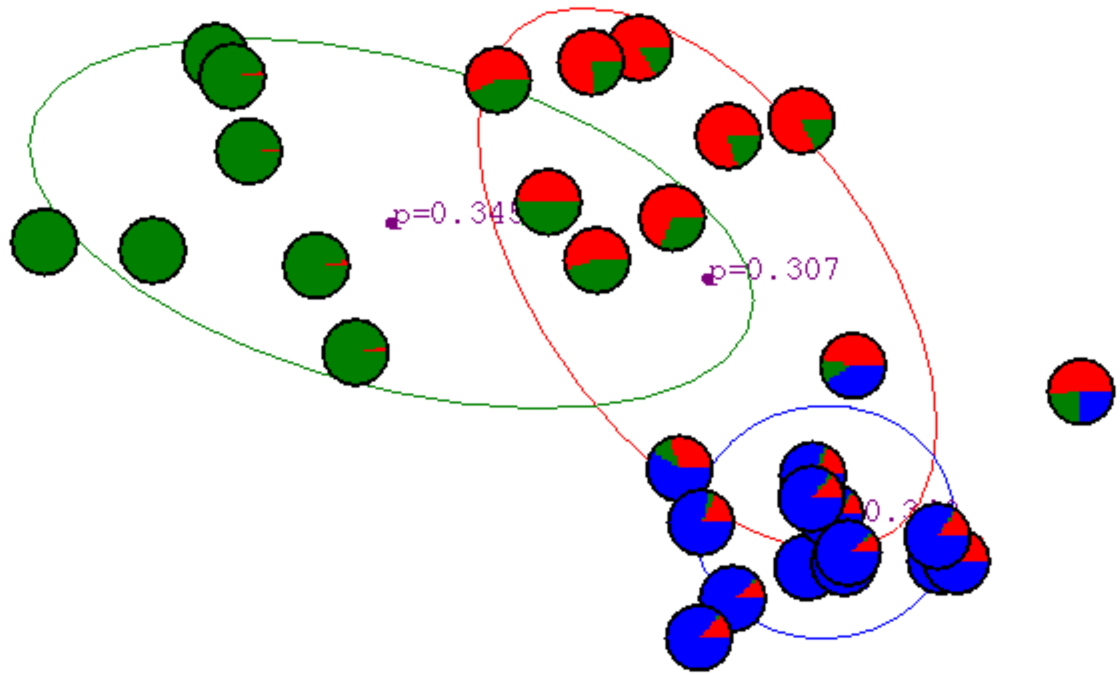
After first iteration



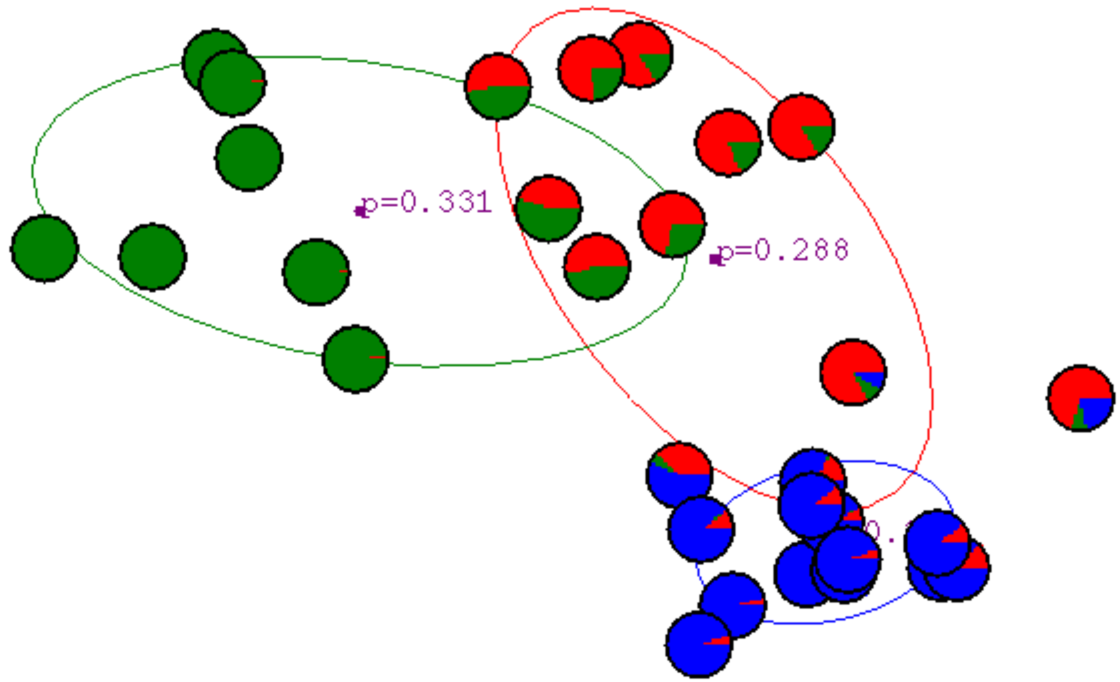
After 2nd iteration



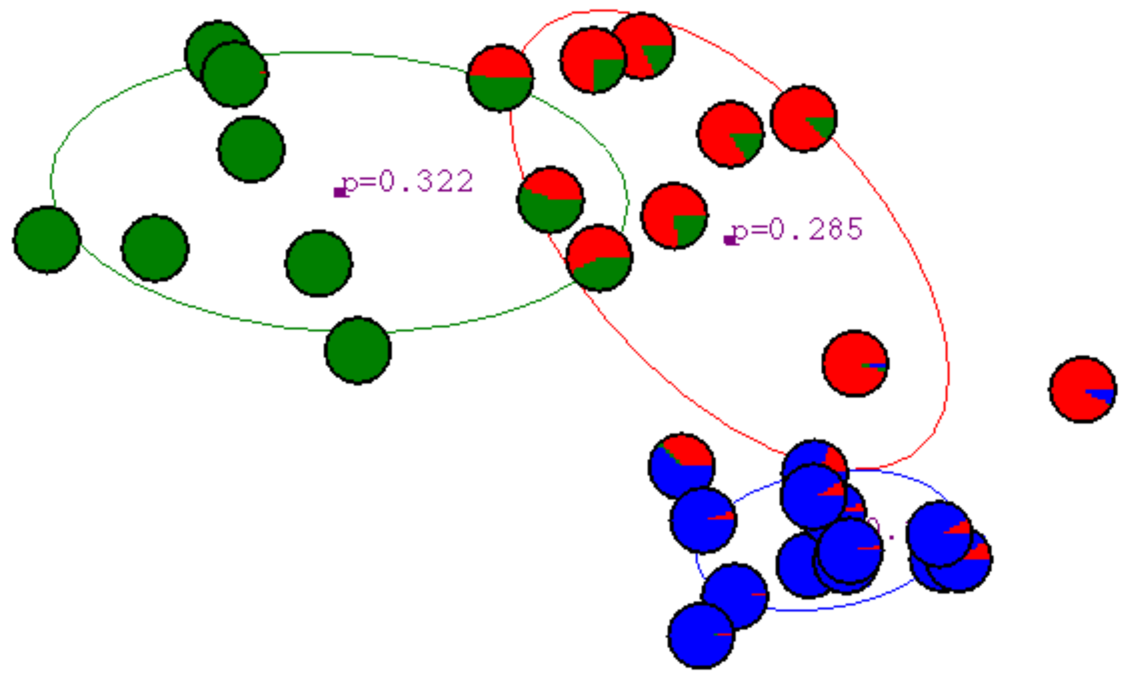
After 3rd iteration



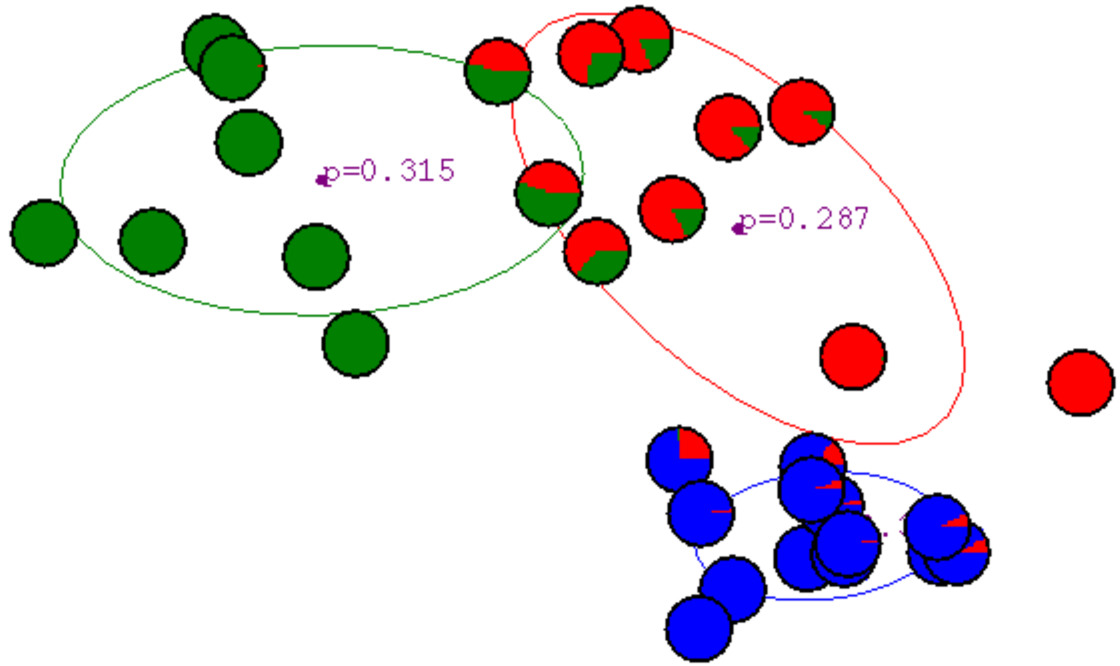
After 4th iteration



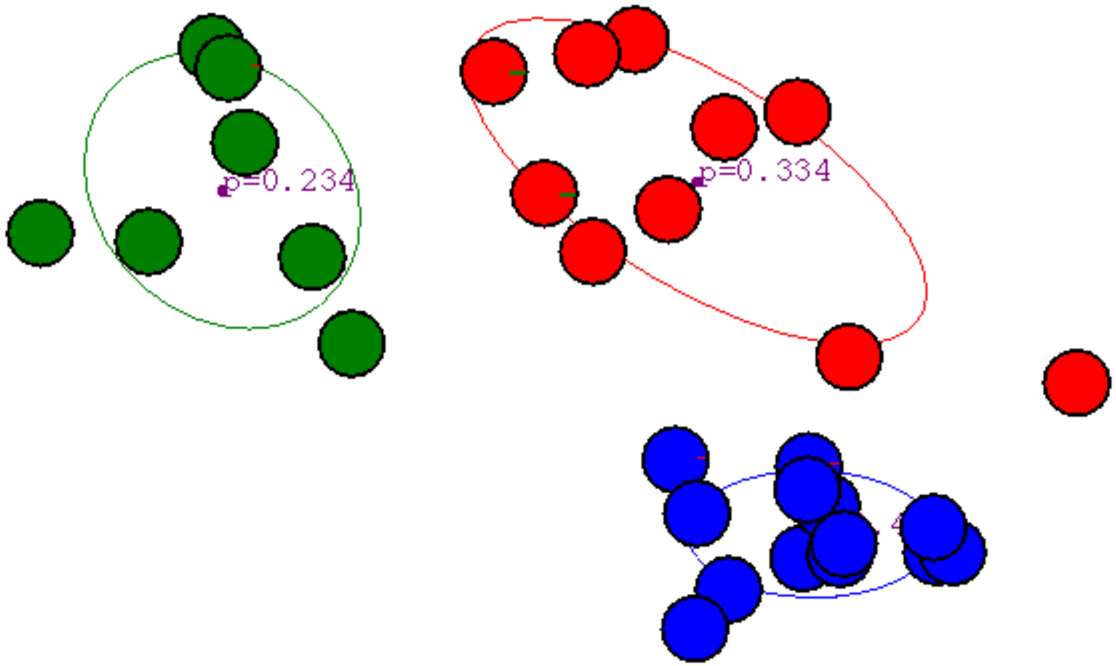
After 5th iteration



After 6th iteration



After 20th iteration



What if we do hard assignments, and learn means only?

E-step / Compute cluster assignment

Compute "expected" classes → set most likely class

$$p(y=i|x^j;\theta_t) = \exp\left(-\frac{1}{2\sigma^2}\|x^j-\mu_i\|_2^2\right) \quad \Rightarrow \quad a^i = \arg\min_j dist(x^i, c^j)$$

$$dist(x,x') = \|x-x'\|_2^2$$

M-step / Recompute cluster mean

Compute most likely new **μs** → averages over hard assignments

$$\mu_i = \frac{\sum_{j=1}^m p(y=i|x^j;\theta_t)x^j}{\sum_{j=1}^m p(y=i|x^j;\theta_t)} \quad \Rightarrow \quad c^i = \frac{1}{|\{j|a^j=i\}|}\sum_{\{j|a^j=i\}} x^j$$

With hard assignments and unit variance, EM is 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:
  - How to "learn" maximum likelihood parameters (locally max. like.) in the case of unlabeled data
- Be happy with this kind of probabilistic analysis
- Remember, E.M. can get stuck in local minima, and empirically it <u>DOES</u>
- EM is coordinate ascent
- General case for EM

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