CSE546: Clustering and EM
Winter 2012

Luke Zettlemoyer

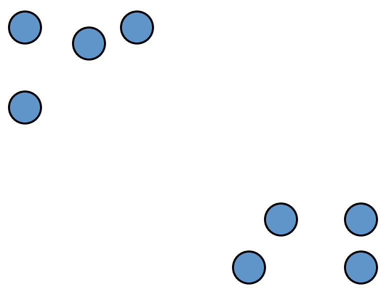
Slides adapted from Carlos Guestrin and Dan Klein

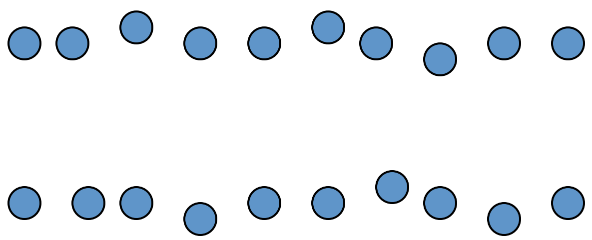
Clustering

- 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



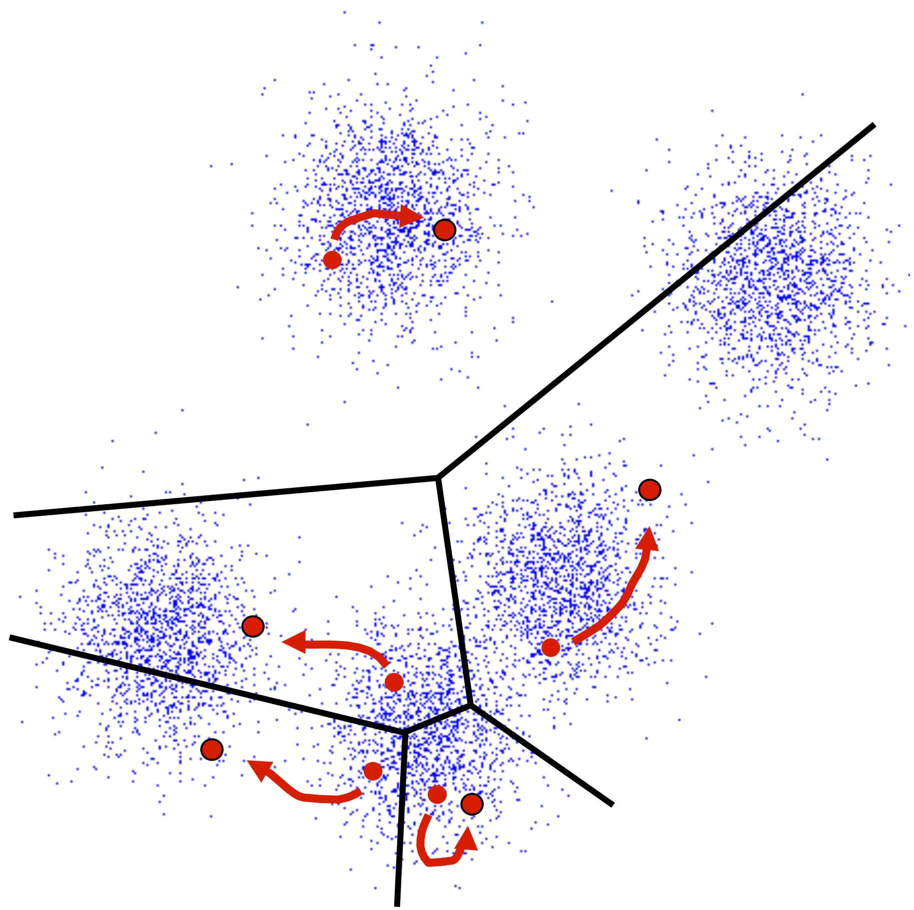


- What could “similar” mean?
  - One option: small (squared) Euclidean distance

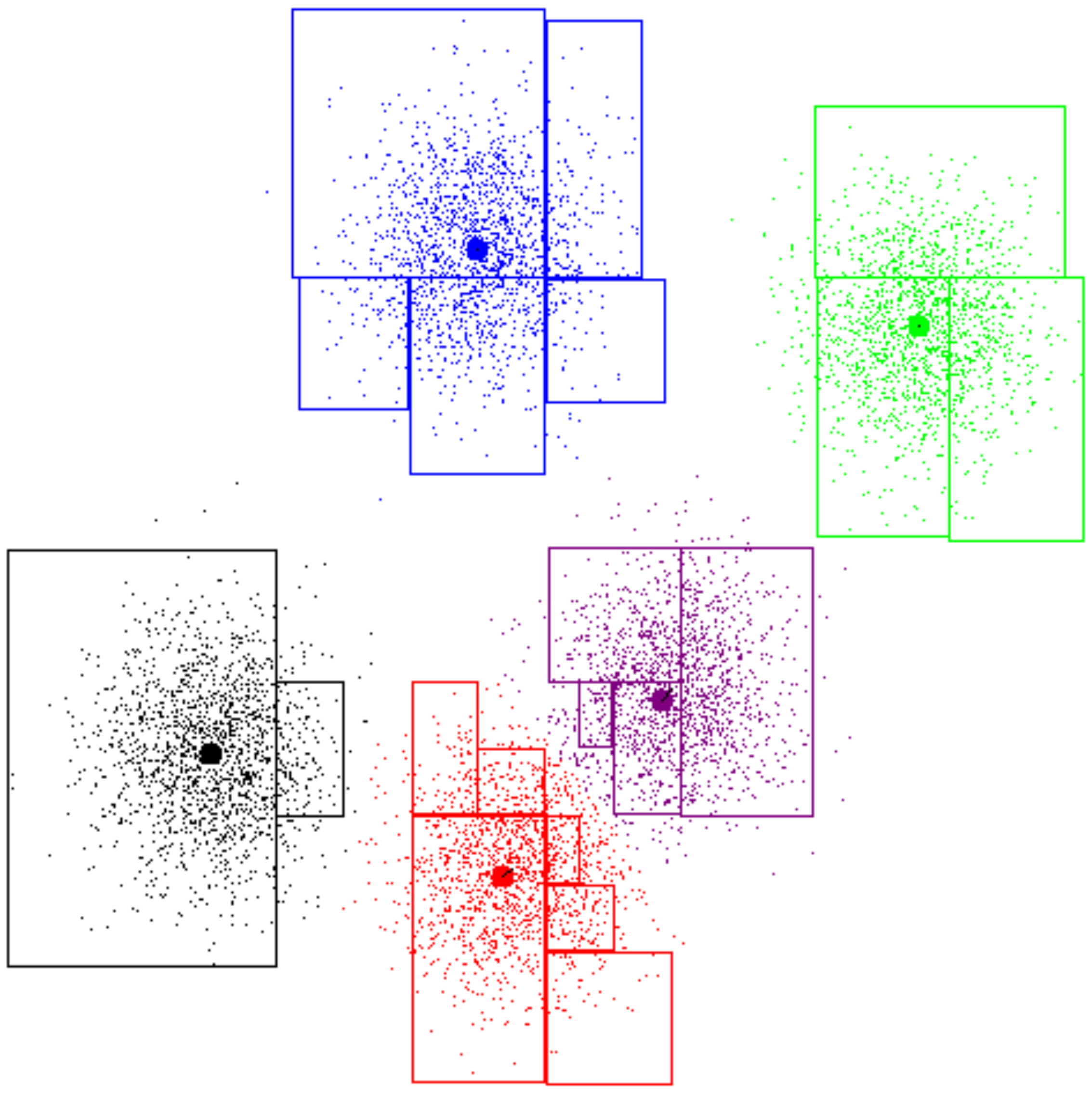
$$\text{dist}(x, y) = (x - y)^\top (x - y) = \sum_i (x_i - y_i)^2$$

K-Means

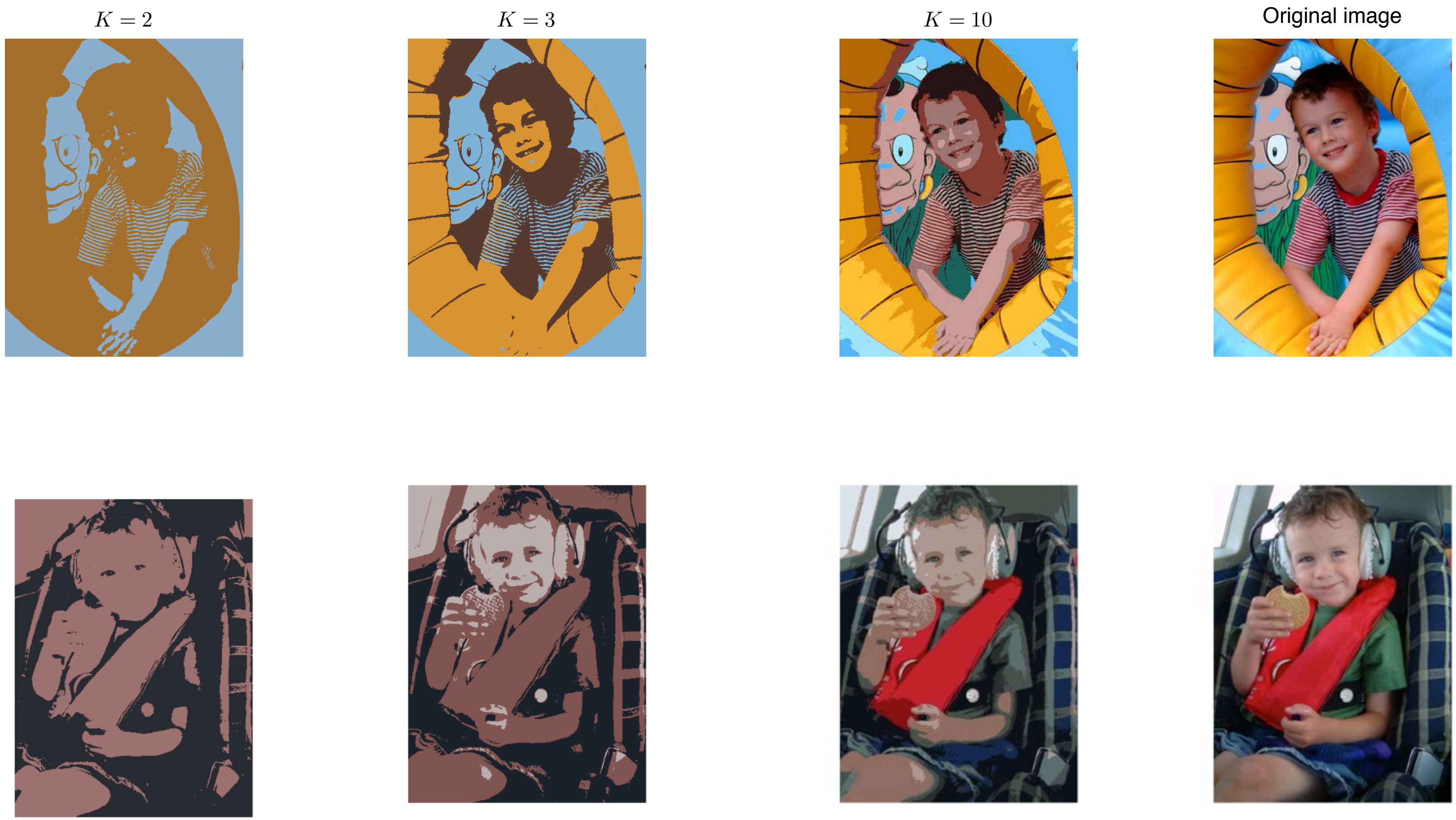
- An iterative clustering algorithm
  – Pick K random points as cluster centers (means)
  – Alternate:
    - Assign data instances to closest mean
    - Assign each mean to the average of its assigned points
  – Stop when no points' assignments change



K-Means Example



Example: K-Means for Segmentation



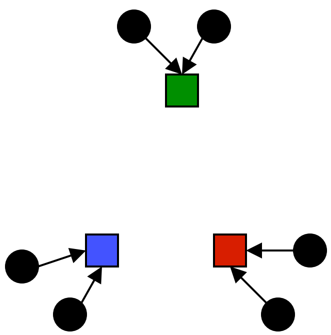
K-Means as Optimization

- Consider the total distance to the means:

$$\phi(\{x_i\}, \{a_i\}, \{c_k\}) = \sum_i \text{dist}(x_i, c_{a_i})$$

points → $\{x_i\}$, assignments → $\{a_i\}$, means → $\{c_k\}$

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



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

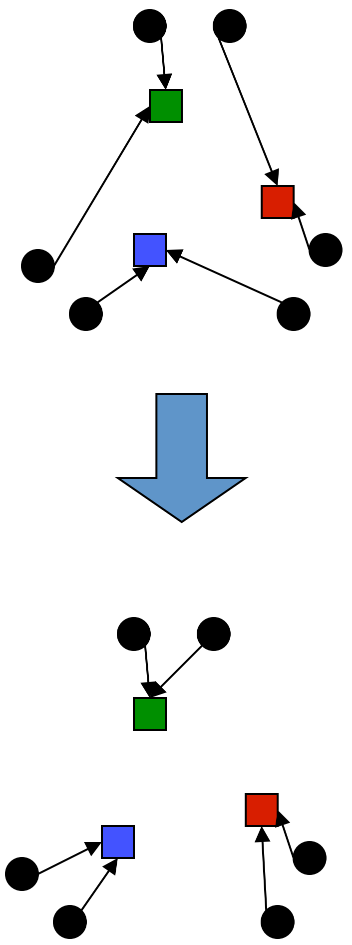
Phase I: Update Assignments

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

$$a_i = \underset{k}{\text{argmin}}\ \text{dist}(x_i, c_k)$$

- Can only decrease total distance phi!

$$\phi(\{x_i\}, \{a_i\}, \{c_k\}) = \sum_i \text{dist}(x_i, c_{a_i})$$

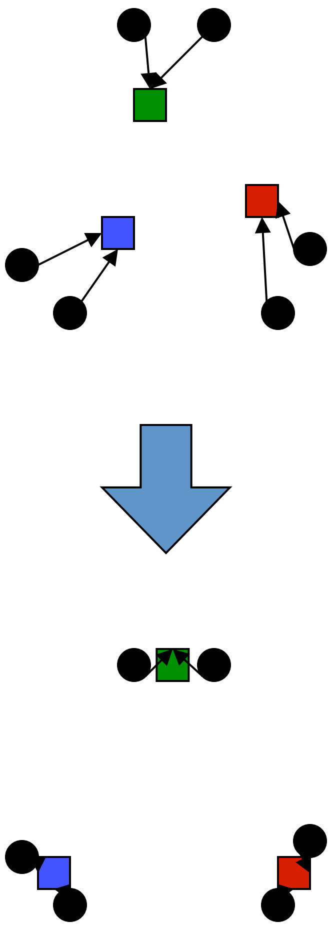


Phase II: Update Means

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

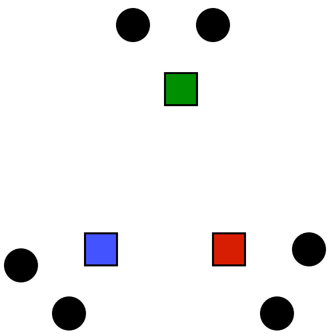
$$c_k = \frac{1}{|\{i : a_i = k\}|} \sum_{i:a_i=k} x_i$$

- 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

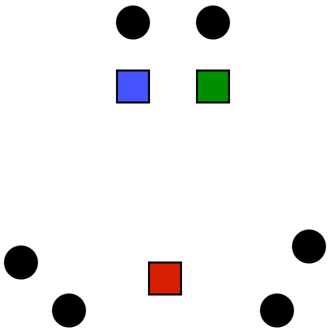


Initialization

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



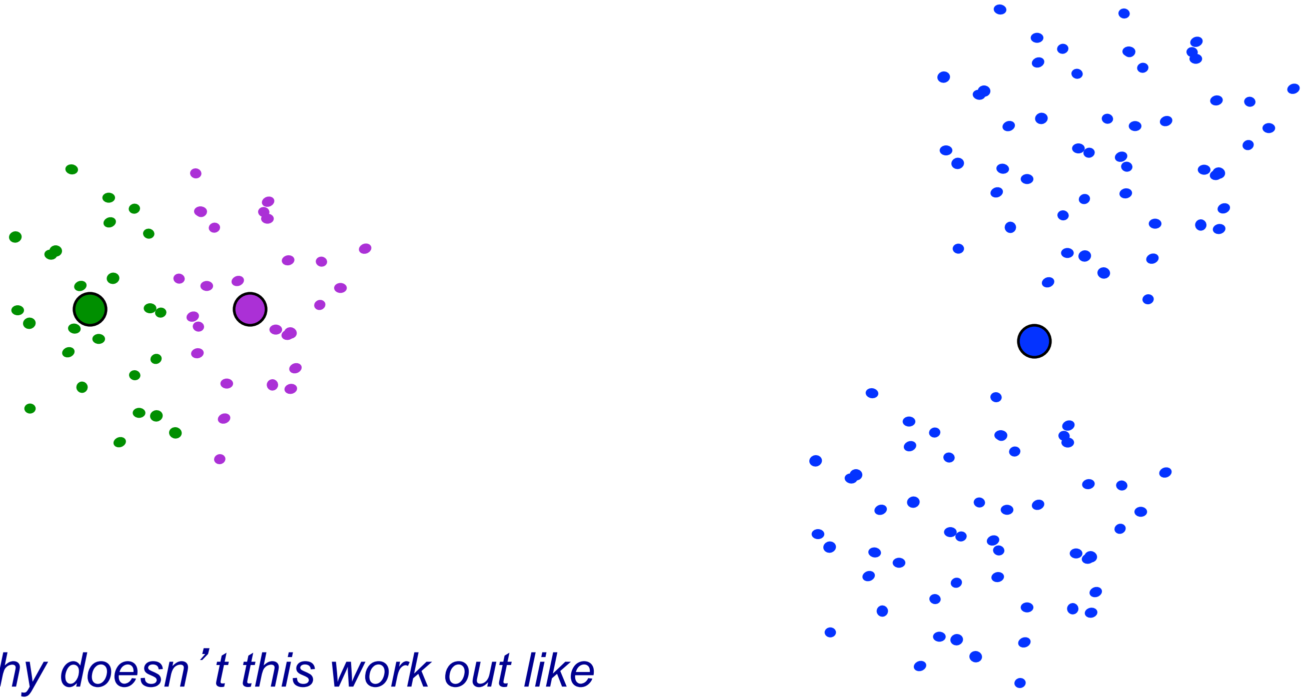
  - What can go wrong?



  - Various schemes for preventing this kind of thing: variance-based split / merge, initialization heuristics

K-Means Getting Stuck

- A local optimum:



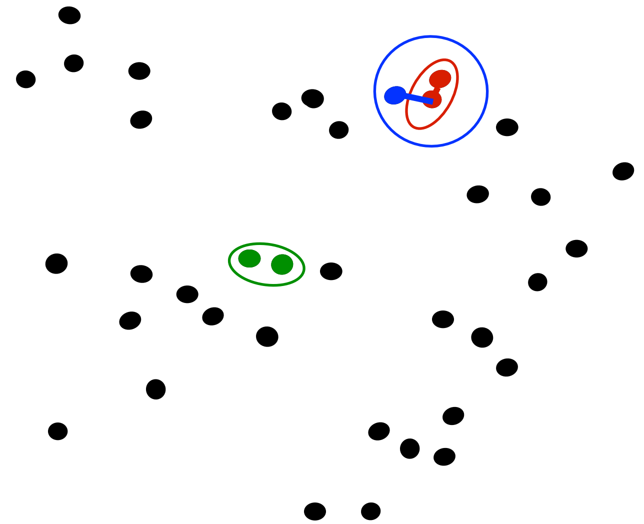
Why doesn't this work out like the earlier example, with the purple taking over half the blue?

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?

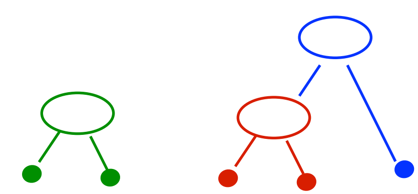
Agglomerative Clustering

- 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

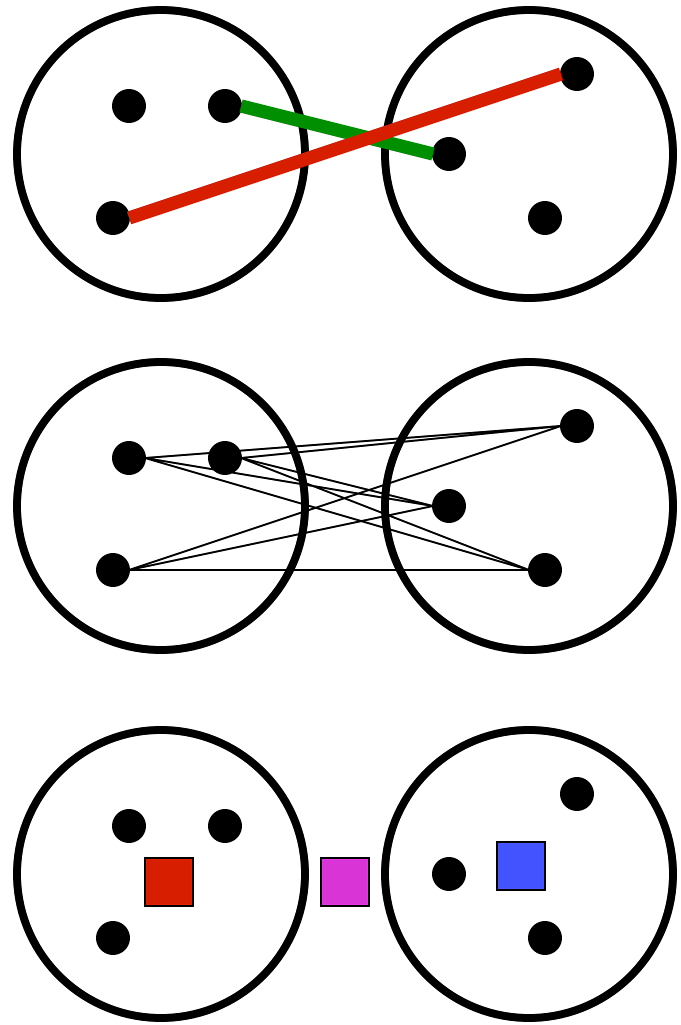


Agglomerative Clustering

- 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
  - Ward's method (min variance, like k-means)

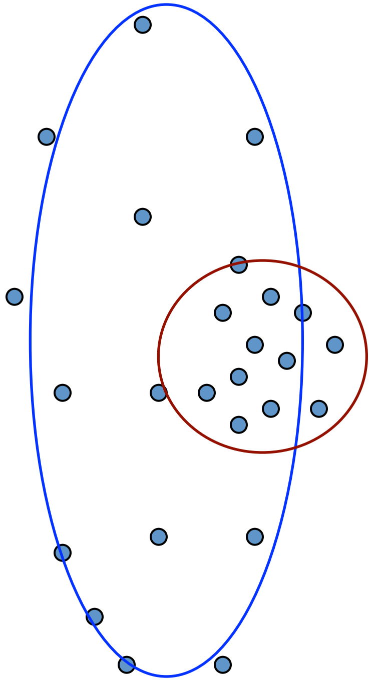
- Different choices create different clustering behaviors



Agglomerative Clustering Questions

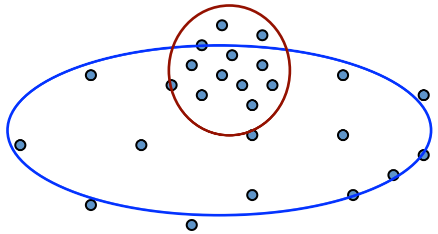
- Will agglomerative clustering converge?
  - To a global optimum?
- Will it always find the true patterns in the data?
  - If the patterns are very very clear?
- Will it find something interesting?
- Do people ever use it?
- How many clusters to pick?

(One) bad case for "hard assignments"?



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

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?

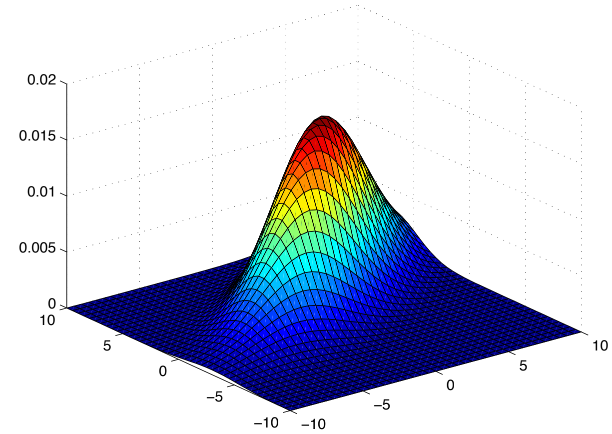
- Depends on X!
- Here, maybe Gaussian Naïve Bayes?
  - Multinomial over clusters Y, Gaussian over each X_i given Y

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

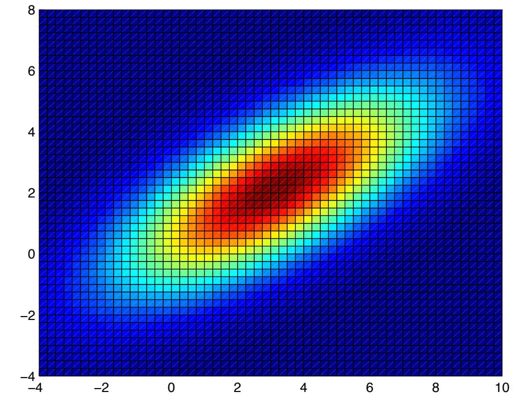
$$P(X_i = x \mid Y = y_k) = \frac{1}{\sigma_{ik}\sqrt{2\pi}} \quad e^{\frac{-(x-\mu_{ik})^2}{2\sigma_{ik}^2}}$$

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
…	…	…

Could we make fewer assumptions?



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



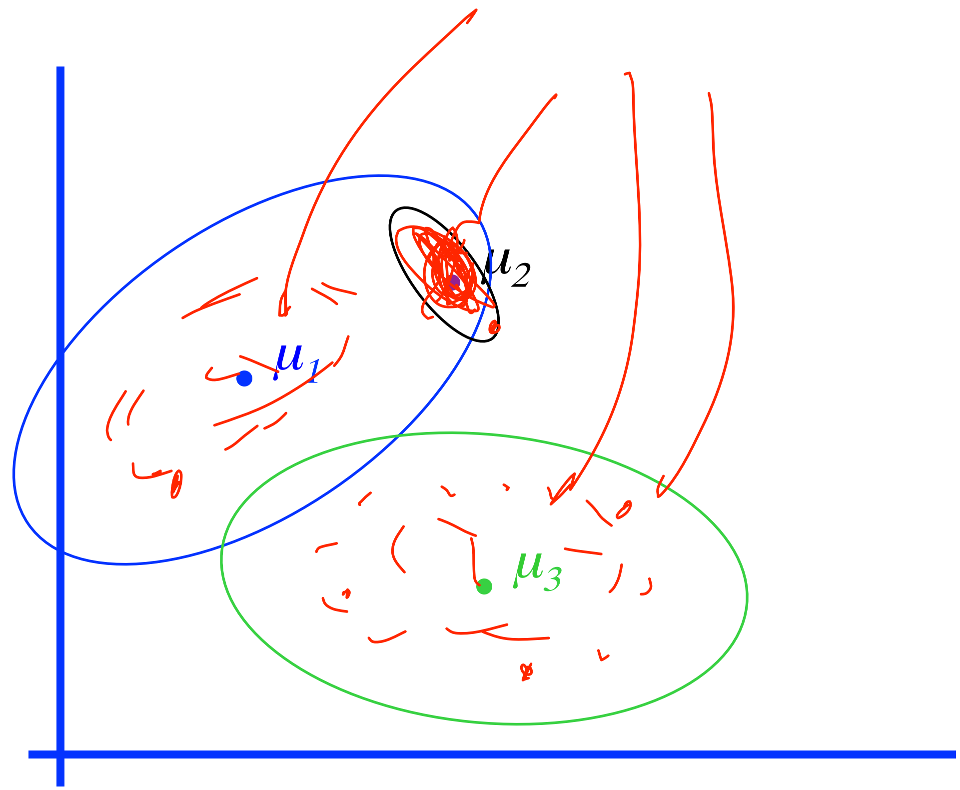
$$P(X = \mathbf{x}_j \mid Y = i) = \frac{1}{(2\pi)^{m/2} \| \Sigma_i \|^{1/2}} \exp\left[-\frac{1}{2} \left(\mathbf{x}_j - \mu_i \right)^T \Sigma_i^{-1} \left(\mathbf{x}_j - \mu_i \right) \right]$$

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(m_i, Σ_i)



Detour/Review: Supervised MLE for GMM

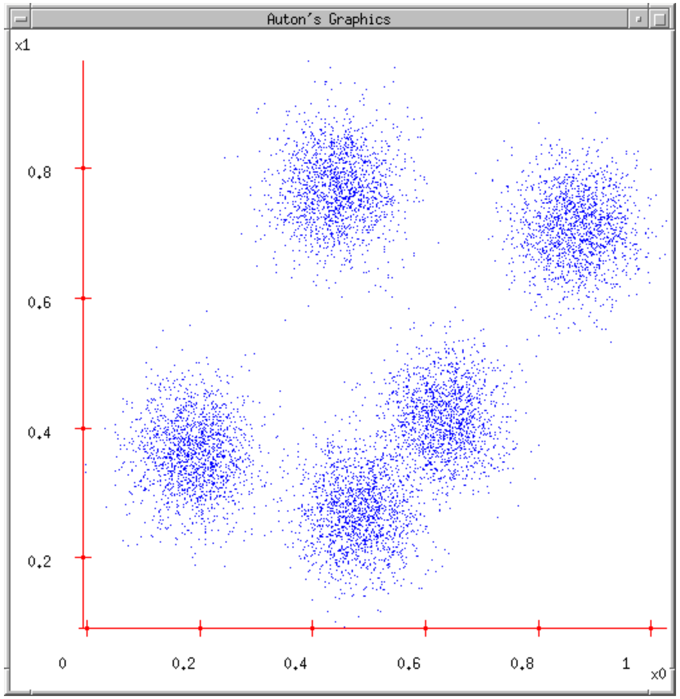
- How do we estimate parameters for Gaussian Mixtures with fully supervised data?
- Have to define objective and solve optimization problem.

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

- For example, MLE estimate has closed form solution:

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

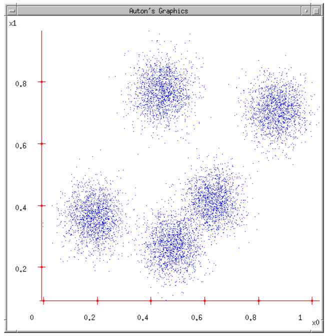
That was easy! Now, lets estimate parameters!



- 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…



One simple example: spherical Gaussians, known variance

- 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:

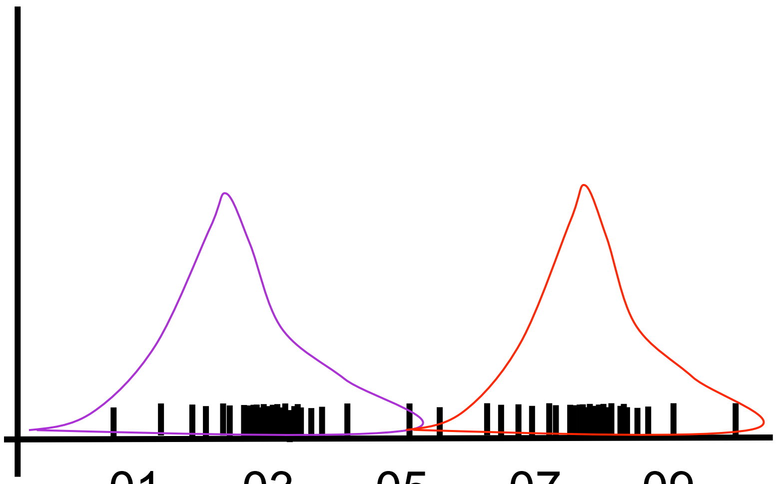
$$\prod_{j=1}^{m}\sum_{i=1}^{k} P(\mathbf{x}_j, y = i) \propto \prod_{j=1}^{m}\sum_{i=1}^{k} \exp\left[-\frac{1}{2\sigma^2}\left\|\mathbf{x}_j - \mu_i\right\|^2\right] P(y = i)$$

- What is the difference between this and Naïve Bayes?

Simple example: learn means only!

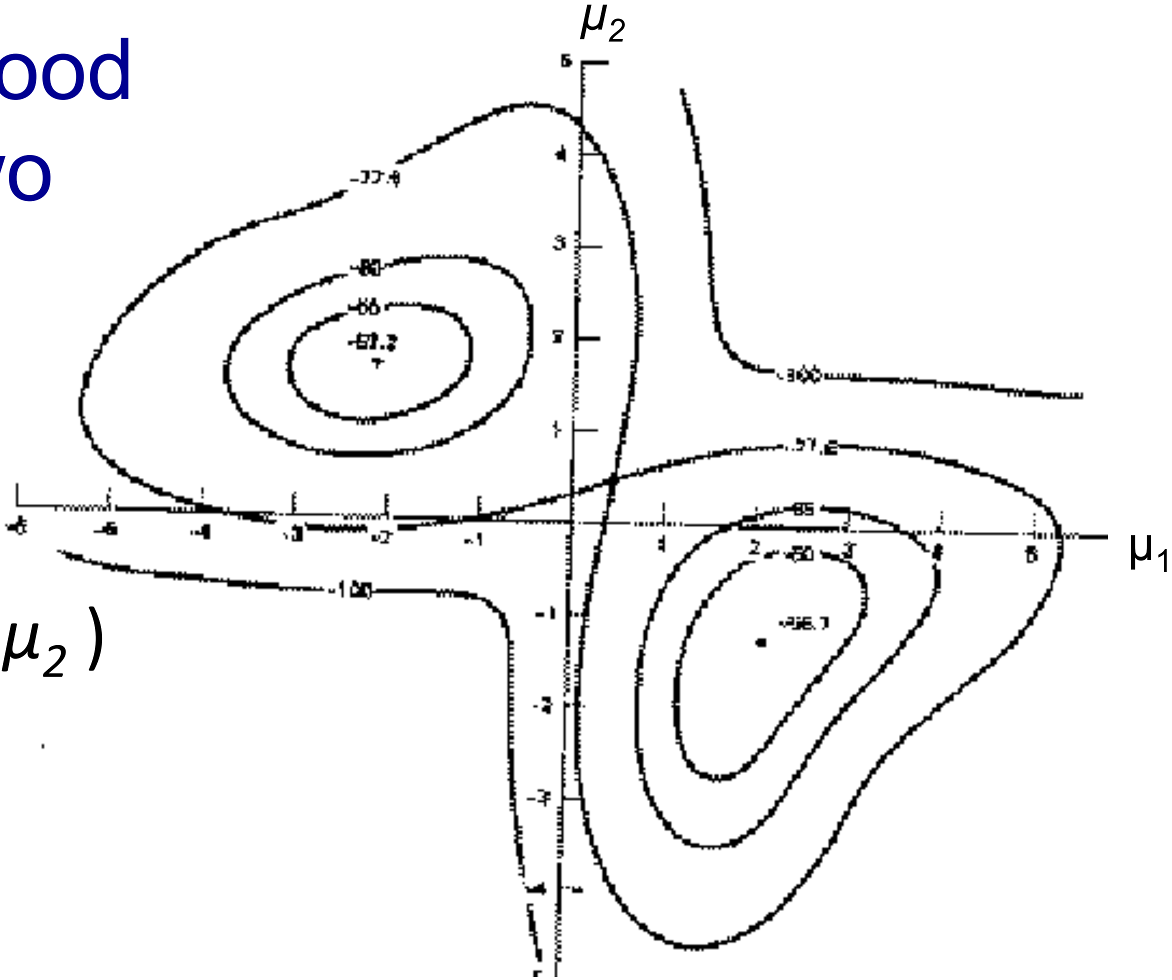
Consider:

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



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

Marginal Likelihood for Mixture of two Gaussians



Graph of

$\log P(x_1, x_2 .. x_n \mid \mu_1, \mu_2)$

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.

Learning general mixtures of Gaussian

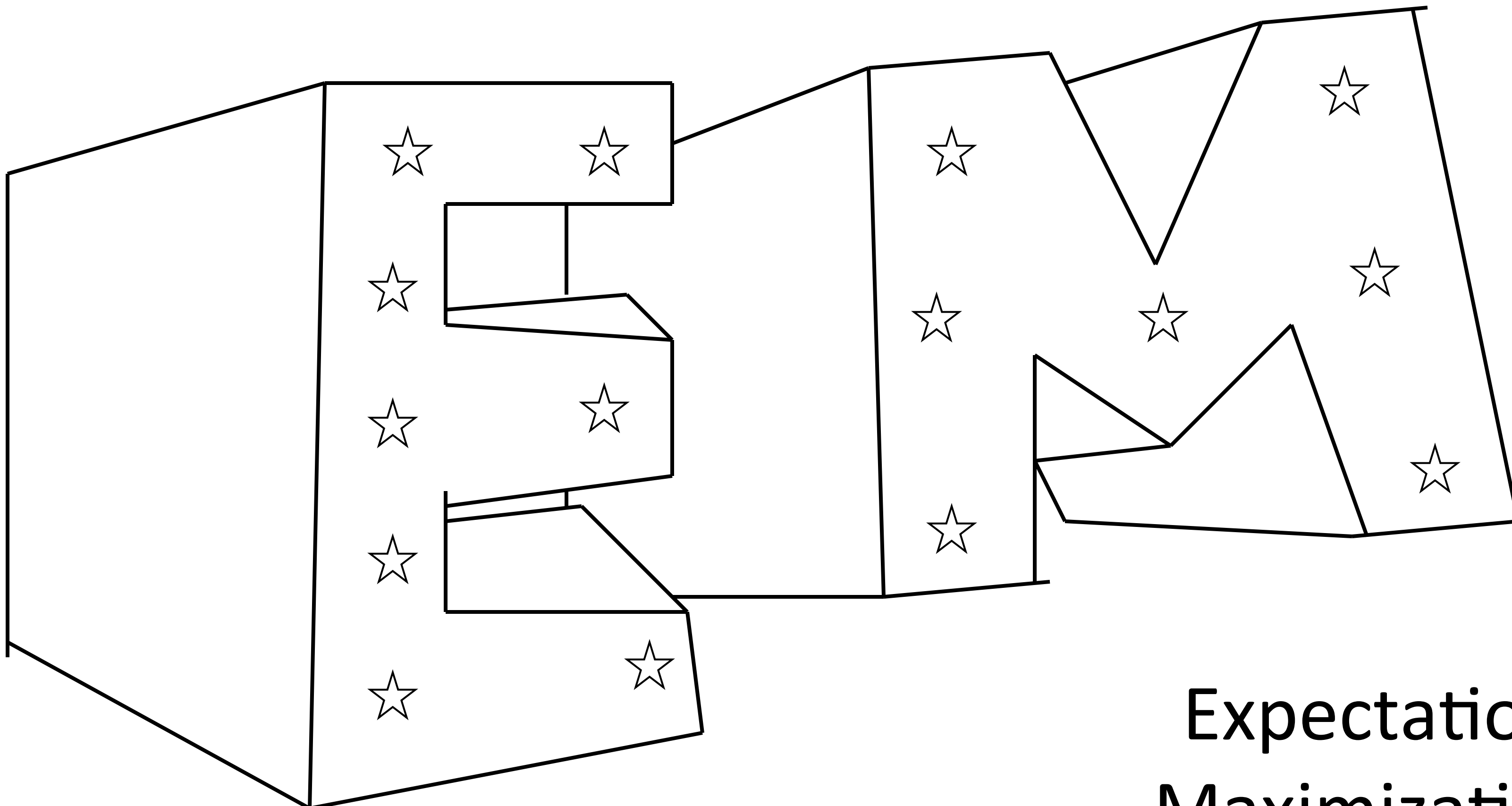
$$P(y=i \mid \mathbf{x}_j) \propto \frac{1}{(2\pi)^{m/2} \| \Sigma_i \|^{1/2}} \exp\left[-\frac{1}{2}\left(\mathbf{x}_j - \mu_i\right)^T \Sigma_i^{-1} \left(\mathbf{x}_j - \mu_i\right)\right] P(y=i)$$

- 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 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: $P(y_j=i \mid x_j, \theta)$

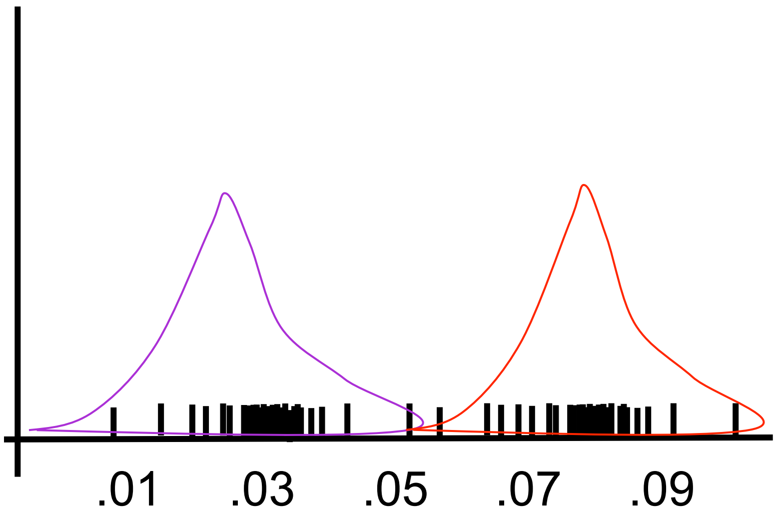
- **M-step**: Re-estimate the parameters with "weighted" MLE estimates
  - Set $\theta = \text{argmax}_\theta \sum_j \sum_{i=1}^k P(y_j=i \mid x_j, \theta) \log P(y_j=i,x_j|\theta)$

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

Simple example: learn means only!

Consider:

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



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

EM for GMMs: only learning means

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

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

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

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

E-step

Compute "expected" classes of all datapoints for each class

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

Just evaluate a Gaussian at x_j

M-step

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

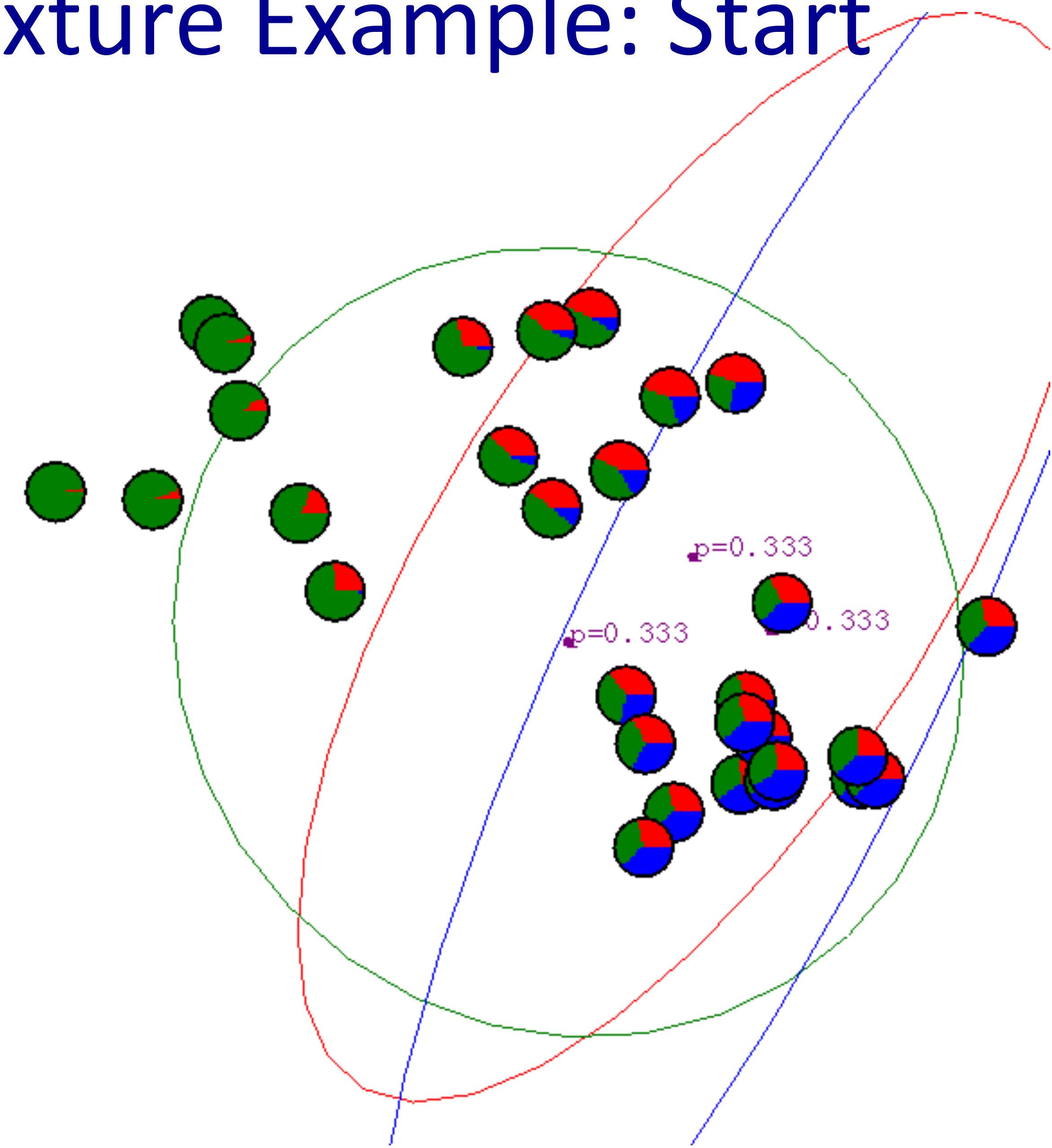
$$\mu_i^{(t+1)} = \frac{\sum_j P(y = i | x_j, \lambda_t) x_j}{\sum_j P(y = i | x_j, \lambda_t)}$$

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

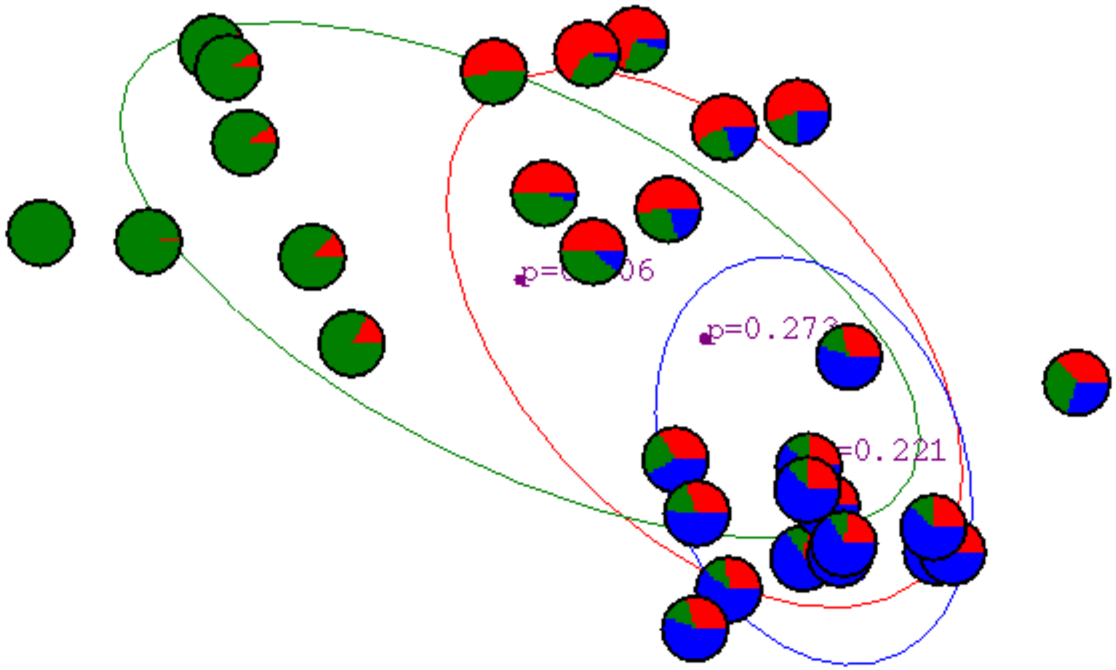
$$p_i^{(t+1)} = \frac{\sum_j P(y = i | x_j, \lambda_t)}{m}$$

m = #training examples

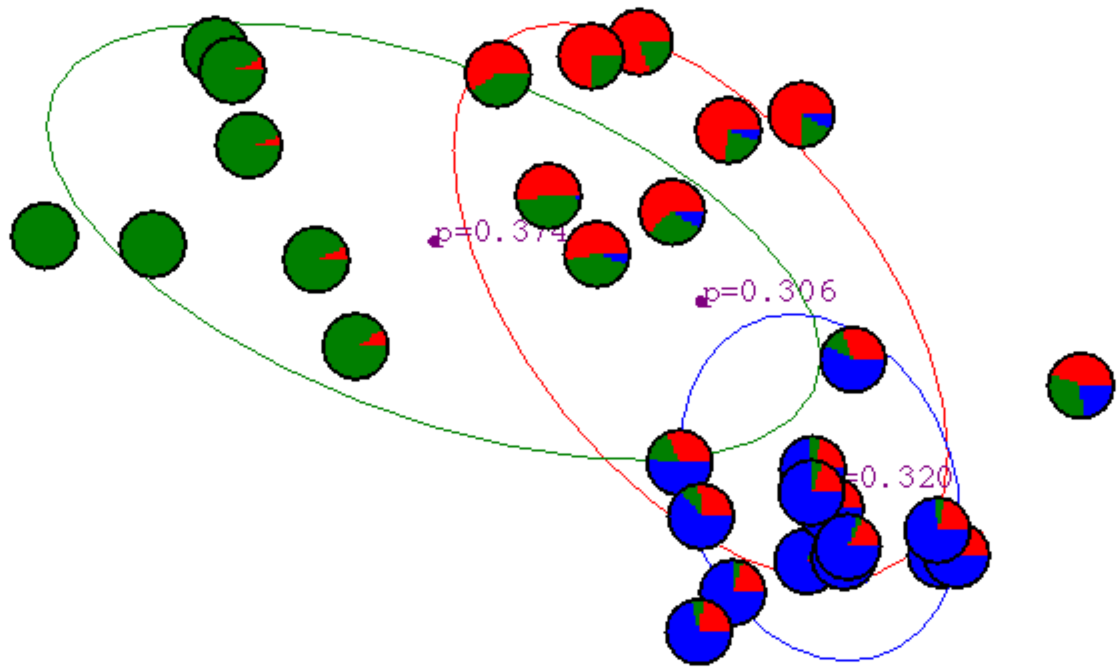
Gaussian Mixture Example: Start



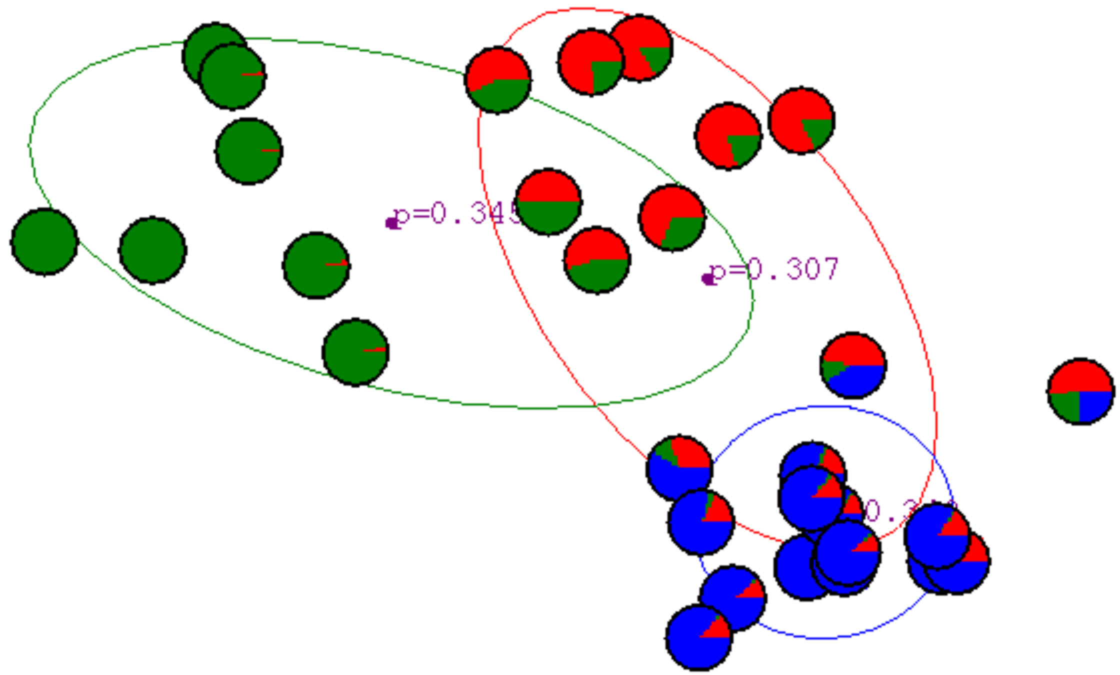
After first iteration



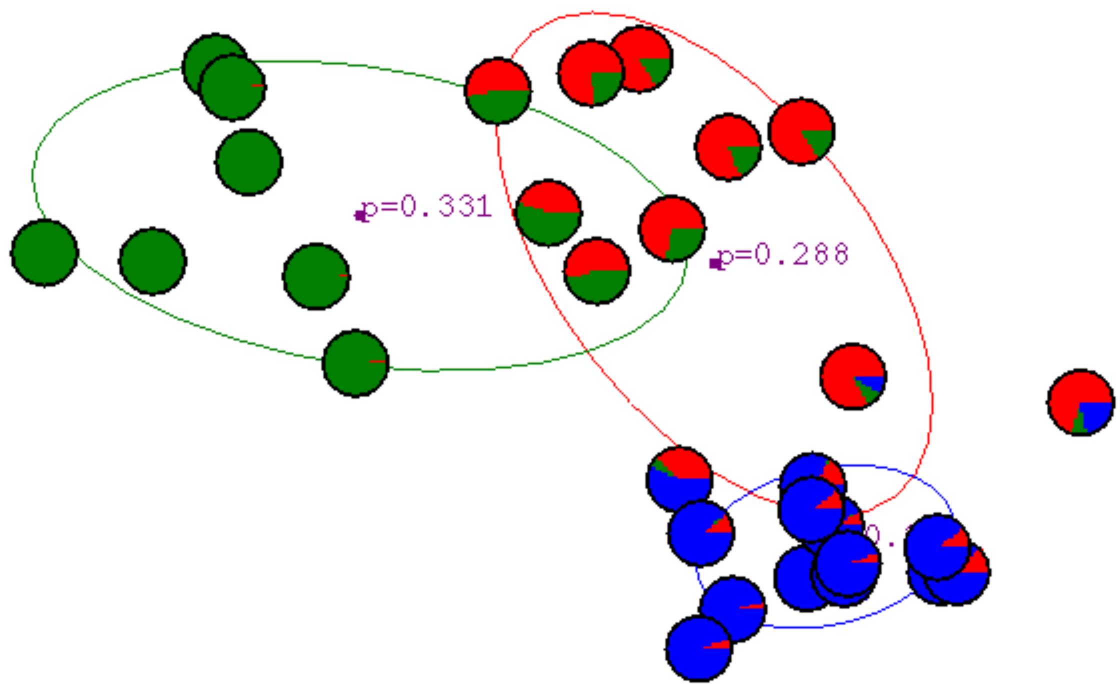
After 2nd iteration



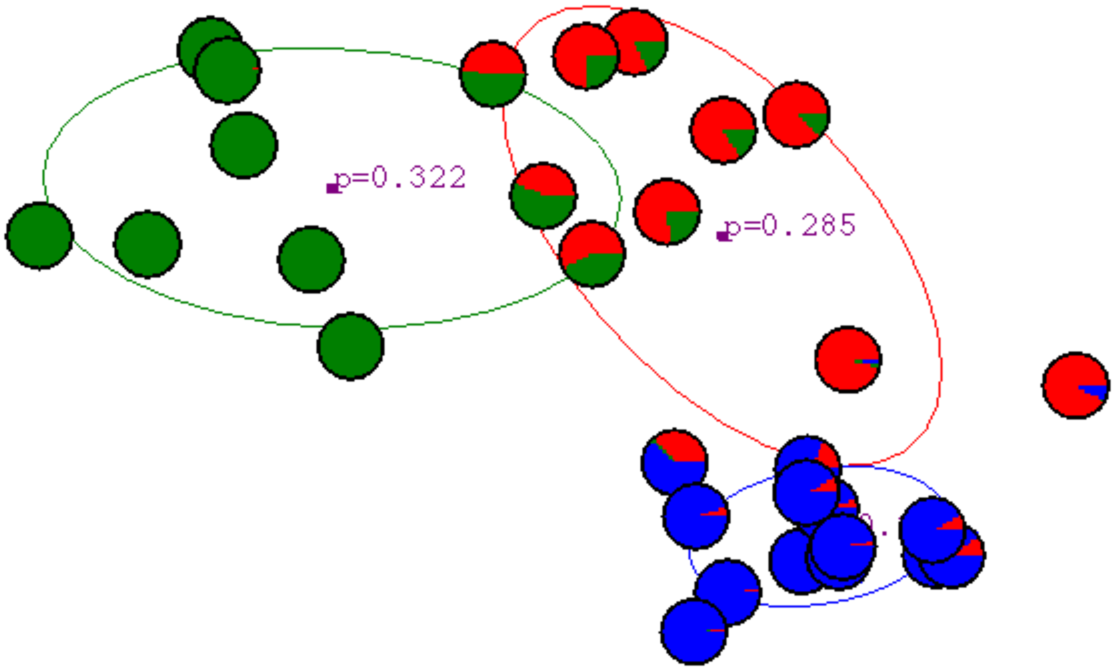
After 3rd iteration



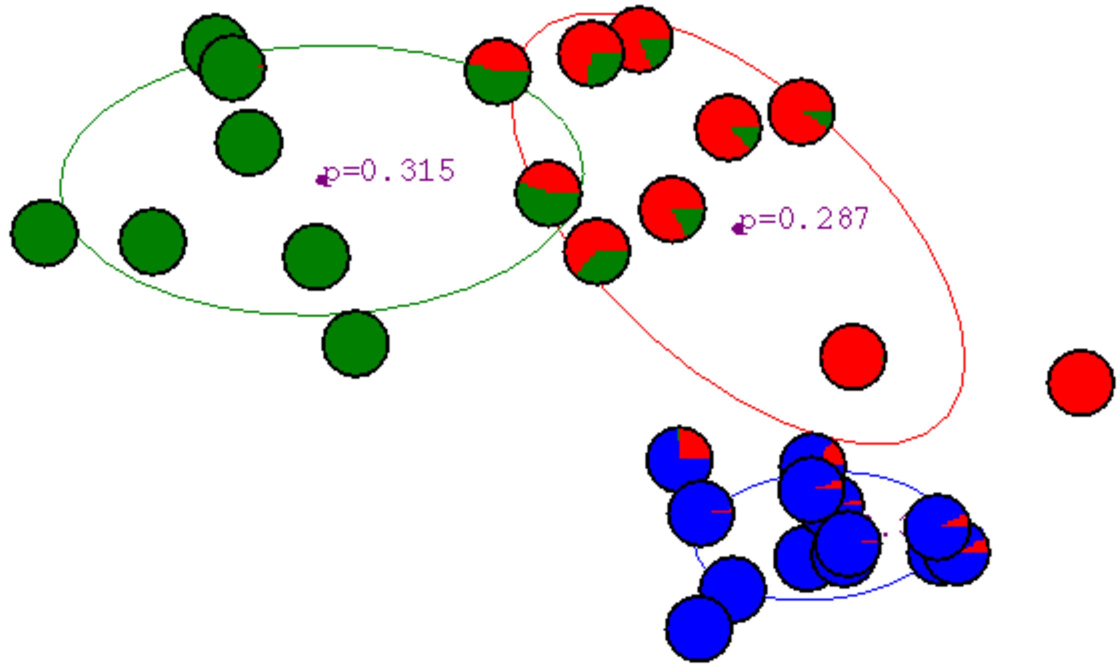
After 4th iteration



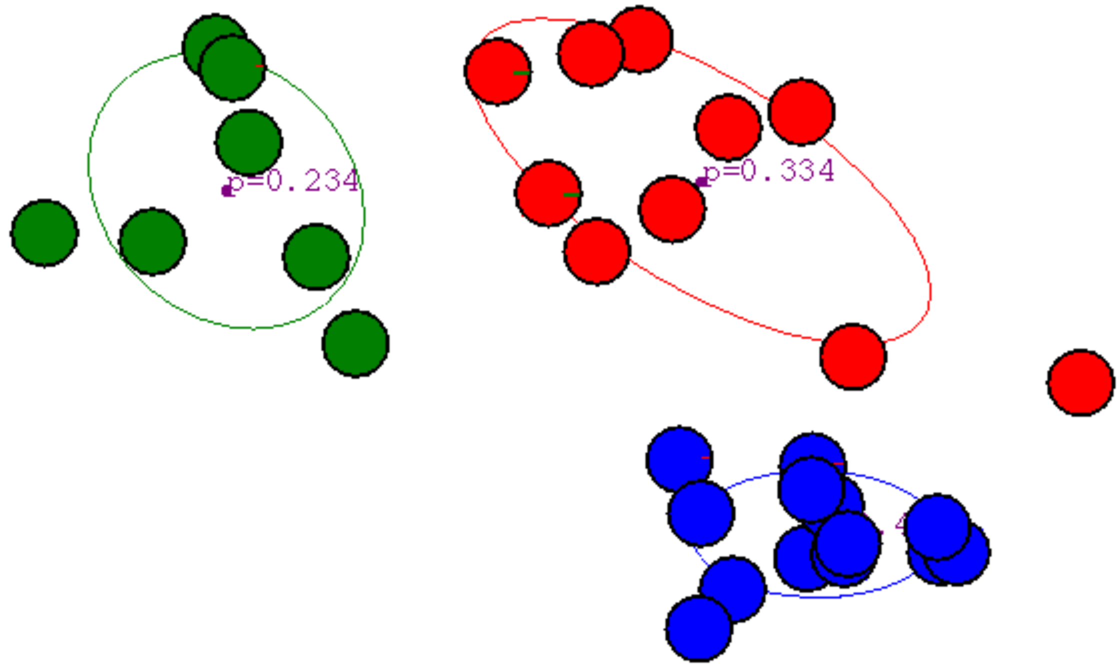
After 5th iteration



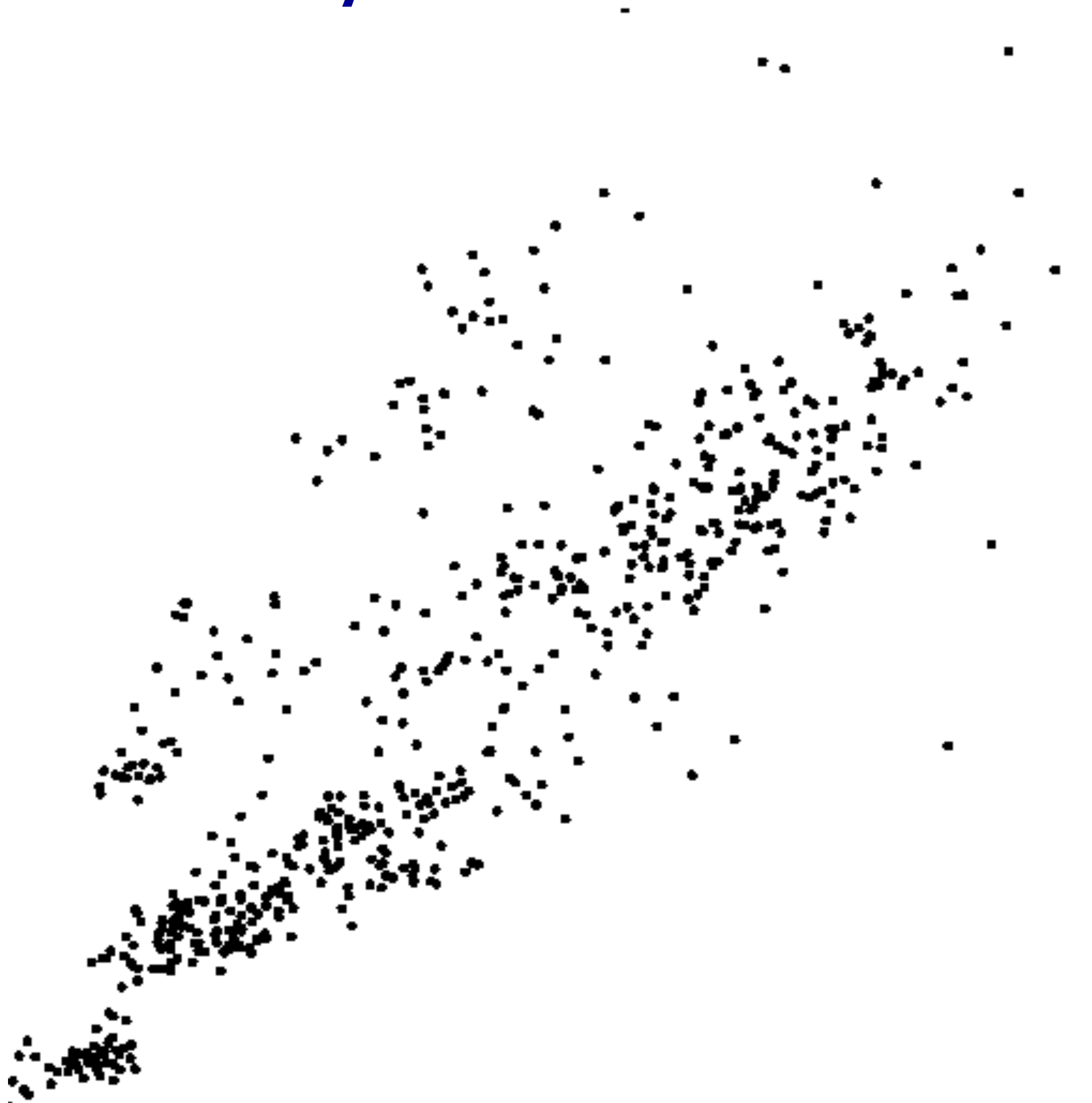
After 6th iteration



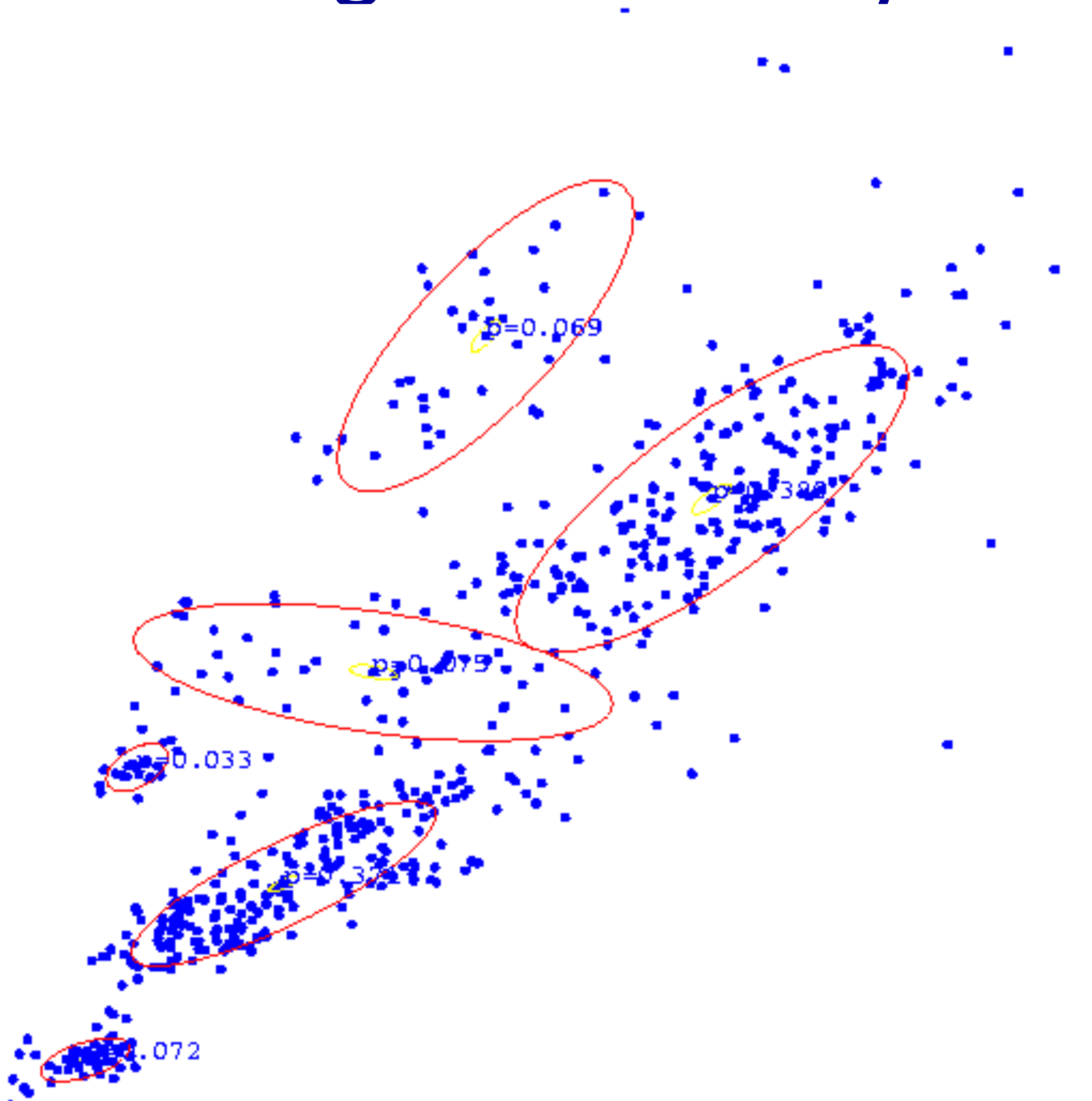
After 20th iteration



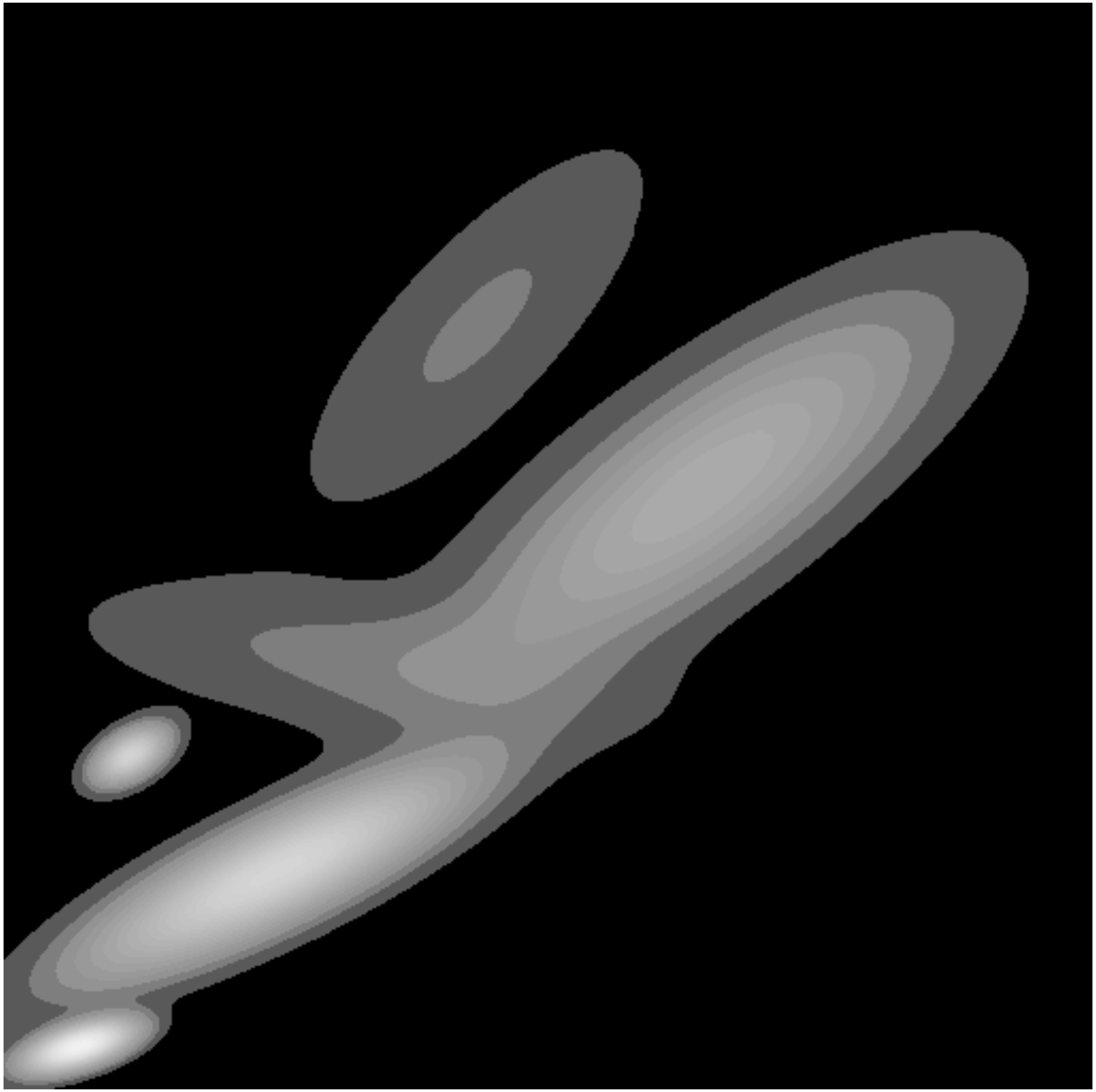
Some Bio Assay data



GMM clustering of the assay data

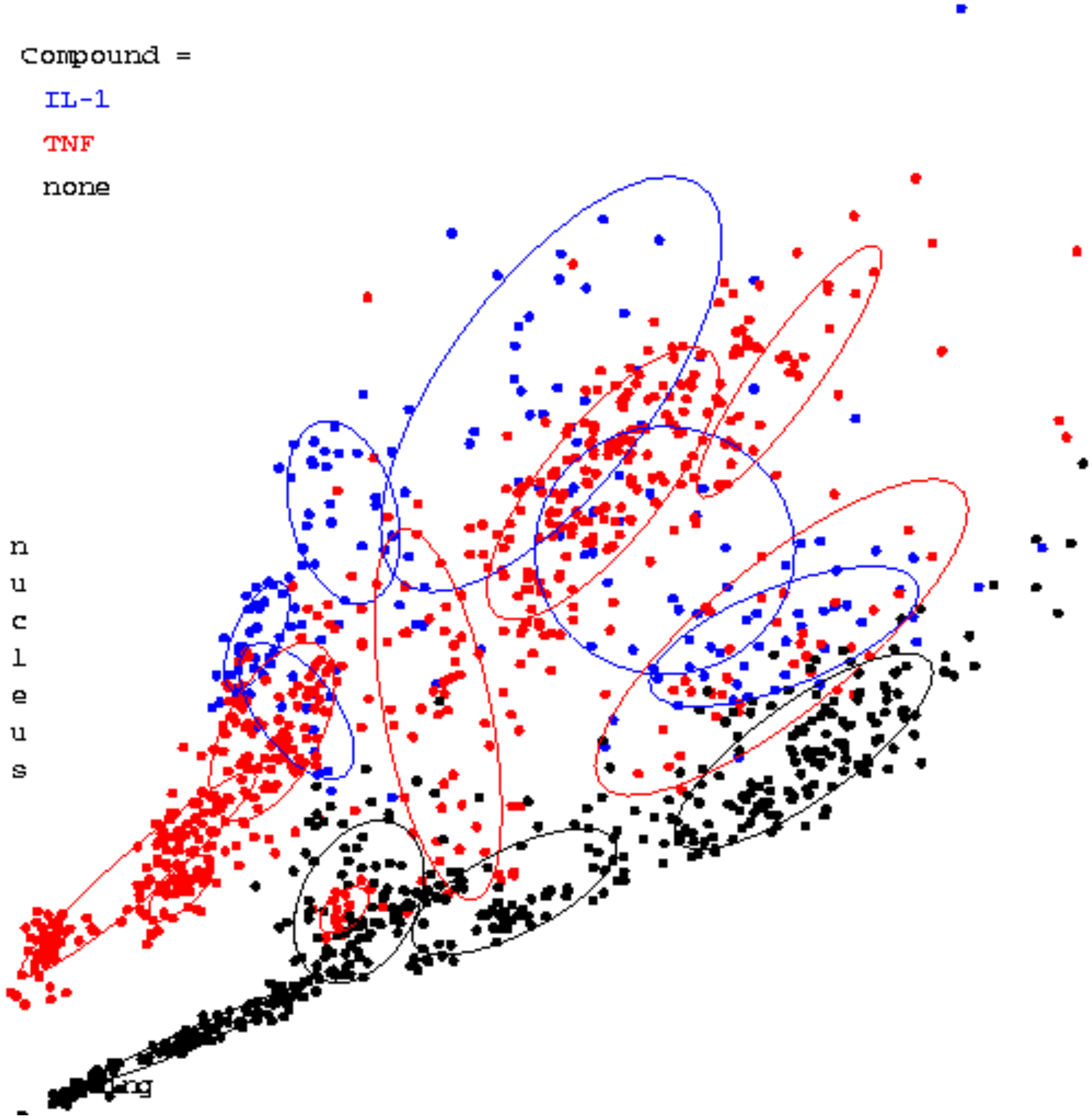


Resulting Density Estimator



Three classes of assay

(each learned with it's own mixture model)



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

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

M-step

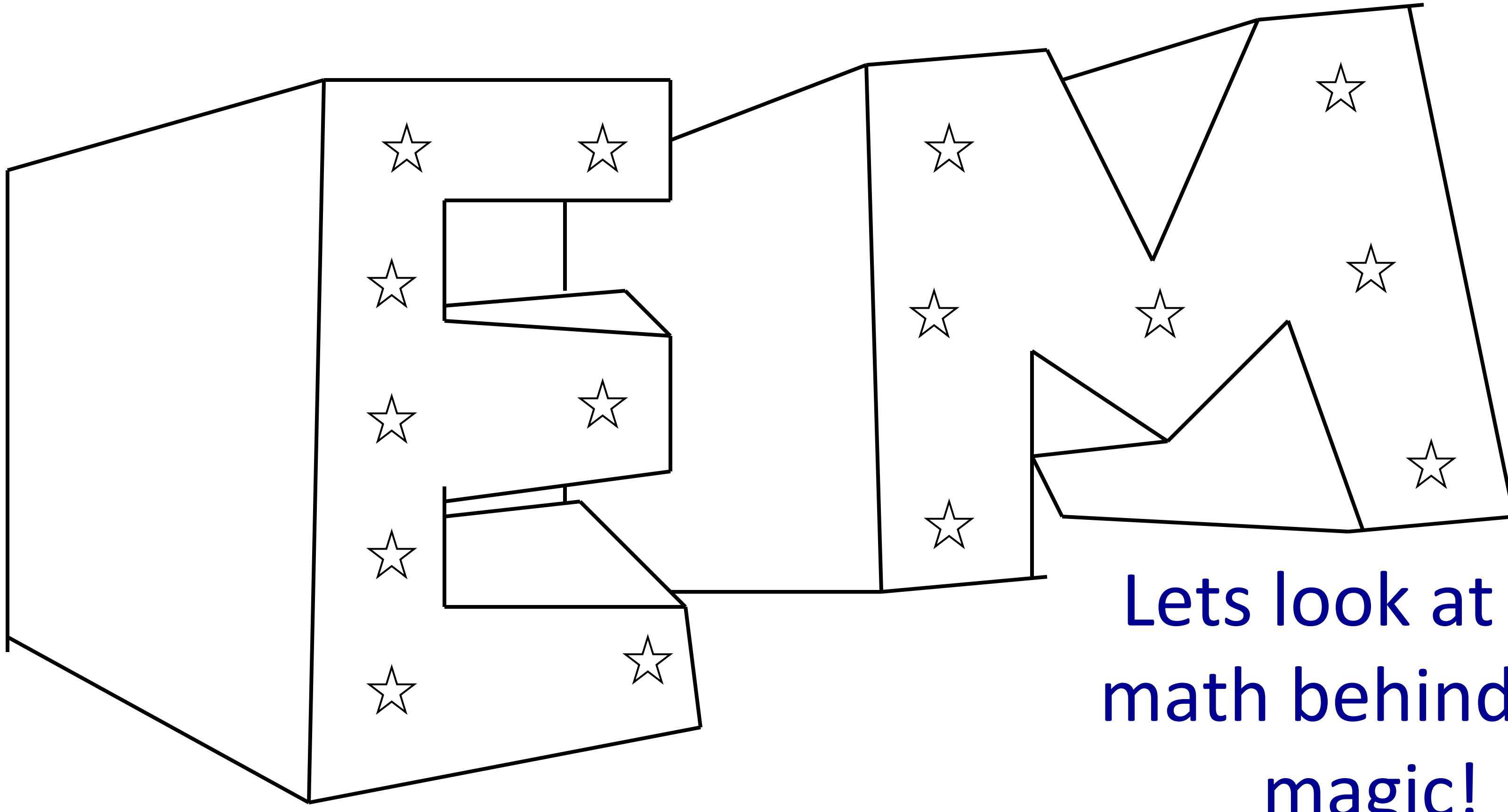
Compute most likely new **μ**s given class expectations

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

$$\mu_i = \frac{\delta\left(y = i, x_j\right) x_j}{\sum_{j=1}^{m} \delta\left(y = i, x_j\right)}$$

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

Equivalent to k-means clustering algorithm!!!



Lets look at the math behind the magic!

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

The general learning problem with missing data

- Marginal likelihood: **x** is observed, **z** is missing:

$$\begin{aligned}\ell(\theta : \mathcal{D}) &= \log \prod_{j=1}^{m} P(\mathbf{x}_j \mid \theta) \\ &= \sum_{j=1}^{m} \log P(\mathbf{x}_j \mid \theta) \\ &= \sum_{j=1}^{m} \log \sum_{\mathbf{z}} P(\mathbf{x}_j, \mathbf{z} \mid \theta)\end{aligned}$$

- Objective: Find argmax_θ l(θ:D)

A Key Computation: E-step

- **x** is observed, **z** is missing
- Compute probability of missing data given current choice of θ
  - $Q(\mathbf{z}|\mathbf{x}_j)$ for each $\mathbf{x}_j$
    - e.g., probability computed during classification step
    - corresponds to "classification step" in K-means

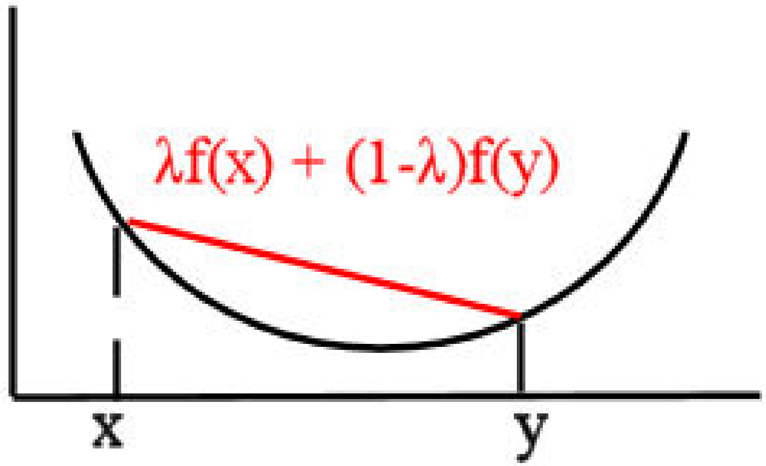
$$Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}_j) \;=\; P(\mathbf{z} \mid \mathbf{x}_j, \theta^{(t)})$$

Jensen's inequality

$$\ell(\theta : \mathcal{D}) \;=\; \sum_{j=1}^{m} \log \sum_{\mathbf{z}} P(\mathbf{z} \mid \mathbf{x}_j) P(\mathbf{x}_j \mid \theta)$$

- **Theorem:**
  - $\log \sum_{\mathbf{z}} P(\mathbf{z})\, f(\mathbf{z}) \;\geq\; \sum_{\mathbf{z}} P(\mathbf{z}) \log f(\mathbf{z})$
  - **e.g., Binary case for convex function f:**

$$\lambda f(x) + (1-\lambda) f(y) \geq f(\lambda x + (1-\lambda) y)$$



  - actually, holds for any concave (convex) function applied to an expectation!

Applying Jensen's inequality

- Use: $\log \sum_{\mathbf{z}} P(\mathbf{z})\, f(\mathbf{z}) \geq \sum_{\mathbf{z}} P(\mathbf{z}) \log f(\mathbf{z})$

$$\ell(\theta^{(t)} : \mathcal{D}) \; = \; \sum_{j=1}^{m} \log \sum_{\mathbf{z}} Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}_j) \frac{P(\mathbf{z}, \mathbf{x}_j \mid \theta^{(t)})}{Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}_j)}$$

$$\geq \sum_{j=1}^{m} \sum_{z} Q^{(t+1)}(z \mid x_j) \log \left(\frac{p(z, x_j \mid \theta^{(t)})}{Q^{(t+1)}(z \mid x_j)} \right)$$

$$= \sum_{j=1}^{m} \sum_{z} Q^{(t+1)}(z \mid x_j) \log \left(p(z, x_j \mid \theta^{(t)}) \right) - \sum_{j=1}^{m} \sum_{z} Q^{(t+1)}(z \mid x_j) \log \left(Q^{(t+1)}(z \mid x_j) \right)$$

$$\ell(\theta^{(t)} : \mathcal{D}) \; \geq \; \sum_{j=1}^{m} \sum_{\mathbf{z}} Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}_j) \log P(\mathbf{z}, \mathbf{x}_j \mid \theta^{(t)}) + m.H(Q^{(t+1)})$$

The M-step

Lower bound:

$$\ell(\theta^{(t)} : \mathcal{D}) \geq \sum_{j=1}^{m} \sum_{\mathbf{z}} Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}_j) \log P(\mathbf{z}, \mathbf{x}_j \mid \theta^{(t)}) + m.H(Q^{(t+1)})$$

- Maximization step:

$$\theta^{(t+1)} \leftarrow \arg\max_{\theta} \sum_{j=1}^{m} \sum_{\mathbf{z}} Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}_j) \log P(\mathbf{z}, \mathbf{x}_j \mid \theta)$$

- We are optimizing a lower bound!
- Use expected counts to do weighted learning:
  - If learning requires Count(**x**,**z**)
  - Use $E_{Q(t+1)}$[Count(**x**,**z**)]
  - *Looks a bit like boosting!!!*

Convergence of EM

- Define: potential function F(θ,Q):

$$\ell(\theta : \mathcal{D}) \;\; \geq \;\; F(\theta, Q) = \sum_{j=1}^{m} \sum_{\mathbf{z}} Q(\mathbf{z} \mid \mathbf{x}_j) \log \frac{P(\mathbf{z}, \mathbf{x}_j \mid \theta)}{Q(\mathbf{z} \mid \mathbf{x}_j)}$$

  – lower bound from Jensen's inequality

- EM is coordinate ascent on F!
  – Thus, maximizes lower bound on marginal log likelihood

M-step can't decrease F(θ,Q): by definition!

$$\theta^{(t+1)} \leftarrow \arg\max_{\theta} \sum_{j=1}^{m} \sum_{\mathbf{z}} Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}_j) \log P(\mathbf{z}, \mathbf{x}_j \mid \theta)$$

- We are maximizing F directly, by ignoring a constant!

$$F(\theta, Q^{(t+1)}) \;=\; \sum_{j=1}^{m} \sum_{\mathbf{z}} Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}_j) \log P(\mathbf{z}, \mathbf{x}_j \mid \theta) + m.H(Q^{(t+1)})$$

E-step: more work to show that F(θ,Q) doesn't decrease

- KL-divergence: measures distance between distributions

$$KL(Q||P) = \sum_z Q(z) \log \frac{Q(z)}{P(z)}$$

- KL=zero if and only if Q=P

E-step also doesn't decrease F: Step 1

- Fix θ to $\theta^{(t)}$, take a max over Q:

$$\ell(\theta^{(t)} : \mathcal{D}) \;\geq\; F(\theta^{(t)}, Q) = \sum_{j=1}^{m} \sum_{\mathbf{z}} Q(\mathbf{z} \mid \mathbf{x}_j) \log \frac{P(\mathbf{z}, \mathbf{x}_j \mid \theta^{(t)})}{Q(\mathbf{z} \mid \mathbf{x}_j)}$$

$$= \sum_{j=1}^{m} \sum_{z} Q(z \mid x_j) \log \left(\frac{P(z \mid x_j, \theta^{(t)}) P(x_j \mid \theta^{(t)})}{Q(z \mid x_j)} \right)$$

$$= \sum_{j=1}^{m} \sum_{z} Q(z \mid x_j) \log \left(P(x_j \mid \theta^{(t)}) \right) - \sum_{j=1}^{m} \sum_{z} Q(z \mid x_j) \log \left(\frac{Q(z \mid x_j)}{P(z \mid x_j, \theta^{(t)})} \right)$$

$$= \;\ell(\theta^{(t)} : \mathcal{D}) - \sum_{j=1}^{m} KL\left(Q(\mathbf{z} \mid \mathbf{x}_j) || P(\mathbf{z} \mid \mathbf{x}_j, \theta^{(t)}) \right)$$

E-step also doesn't decrease F: Step 2

- Fixing θ to $\theta^{(t)}$:

$$\ell(\theta^{(t)} : \mathcal{D}) \geq F(\theta^{(t)}, Q) \;=\; \ell(\theta^{(t)} : \mathcal{D}) + \sum_{j=1}^{m} \sum_{\mathbf{z}} Q(\mathbf{z} \mid \mathbf{x}_j) \log \frac{P(\mathbf{z} \mid \mathbf{x}_j, \theta^{(t)})}{Q(\mathbf{z} \mid \mathbf{x}_j)}$$

$$= \;\ell(\theta^{(t)} : \mathcal{D}) - \sum_{j=1}^{m} KL\left(Q(\mathbf{z} \mid \mathbf{x}_j) || P(\mathbf{z} \mid \mathbf{x}_j, \theta^{(t)})\right)$$

- Now, the max over Q yields:
  - $Q(z|x_j) \leftarrow P(z|x_j, \theta^{(t)})$
  - Why? The likelihood term is a constant; the KL term is zero iff the arguments are the same distribution!!
  - So, the E-step is actually a maximization / tightening of the bound. It ensures that:

$$F(\theta^{(t)}, Q^{(t+1)}) \;=\; \ell(\theta^{(t)} : \mathcal{D})$$

EM is coordinate ascent

$$\ell(\theta : \mathcal{D}) \;\; \geq \;\; F(\theta, Q) = \sum_{j=1}^{m} \sum_{\mathbf{z}} Q(\mathbf{z} \mid \mathbf{x}_j) \log \frac{P(\mathbf{z}, \mathbf{x}_j \mid \theta)}{Q(\mathbf{z} \mid \mathbf{x}_j)}$$

- **M-step**: Fix Q, maximize F over θ (a lower bound on $\ell(\theta : \mathcal{D})$):

$$\ell(\theta : \mathcal{D}) \;\; \geq \;\; F(\theta, Q^{(t)}) = \sum_{j=1}^{m} \sum_{\mathbf{z}} Q^{(t)}(\mathbf{z} \mid \mathbf{x}_j) \log P(\mathbf{z}, \mathbf{x}_j \mid \theta) + m.H(Q^{(t)})$$

- **E-step**: Fix θ, maximize F over Q:

$$\ell(\theta^{(t)} : \mathcal{D}) \geq F(\theta^{(t)}, Q) \;\; = \;\; \ell(\theta^{(t)} : \mathcal{D}) - m \sum_{j=1}^{m} KL\left(Q(\mathbf{z} \mid \mathbf{x}_j) || P(\mathbf{z} \mid \mathbf{x}_j, \theta^{(t)})\right)$$

  - "Realigns" F with likelihood:

$$F(\theta^{(t)}, Q^{(t+1)}) \;\; = \;\; \ell(\theta^{(t)} : \mathcal{D})$$

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