Unsupervised learning
Lecture 13

David Sontag
New York University

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

Gaussian Mixture Models

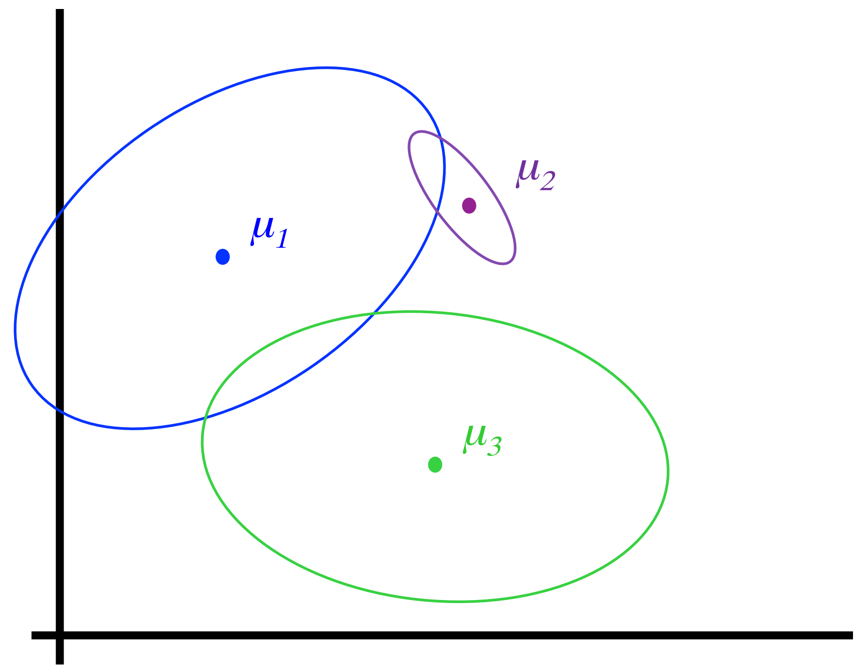
- P(Y): There are k components
- P(X|Y): Each component generates data from a **multivariate Gaussian** with mean μ_i and covariance matrix Σ_i

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

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

$$P(X = \mathbf{x}_j \mid Y = i) =$$

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



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

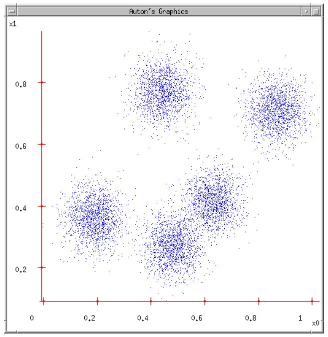
ML estimate for each of the Multivariate Gaussians is given by:

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

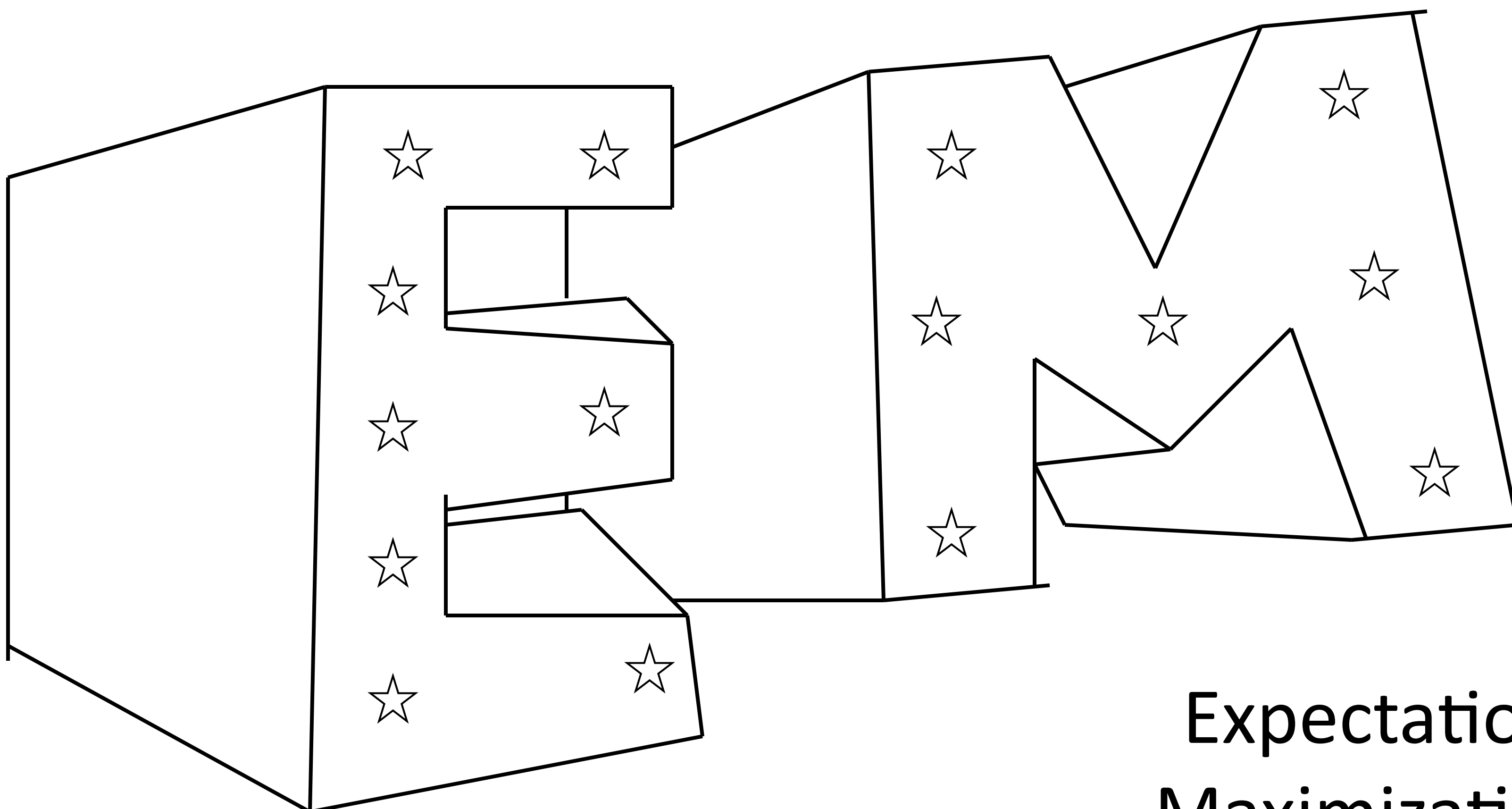
Just sums over *x* generated from the *k*'th Gaussian

What about with unobserved data?

- 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…



Expectation
Maximization

1977: Dempster, Laird, & Rubin

The EM Algorithm

- A clever method for maximizing marginal likelihood:
  - $\text{argmax}_\theta \prod_j P(x_j) = \text{argmax}_\theta \prod_j \sum_{k=1}^{K} P(Y_j=k, x_j)$
  - Based on coordinate descent. 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: $\text{argmax}_\theta \lg \prod_j \sum_{k=1}^K P(Y_j=k, x_j ; \theta) = \sum_j \lg \sum_{k=1}^K P(Y_j=k, x_j; \theta)$

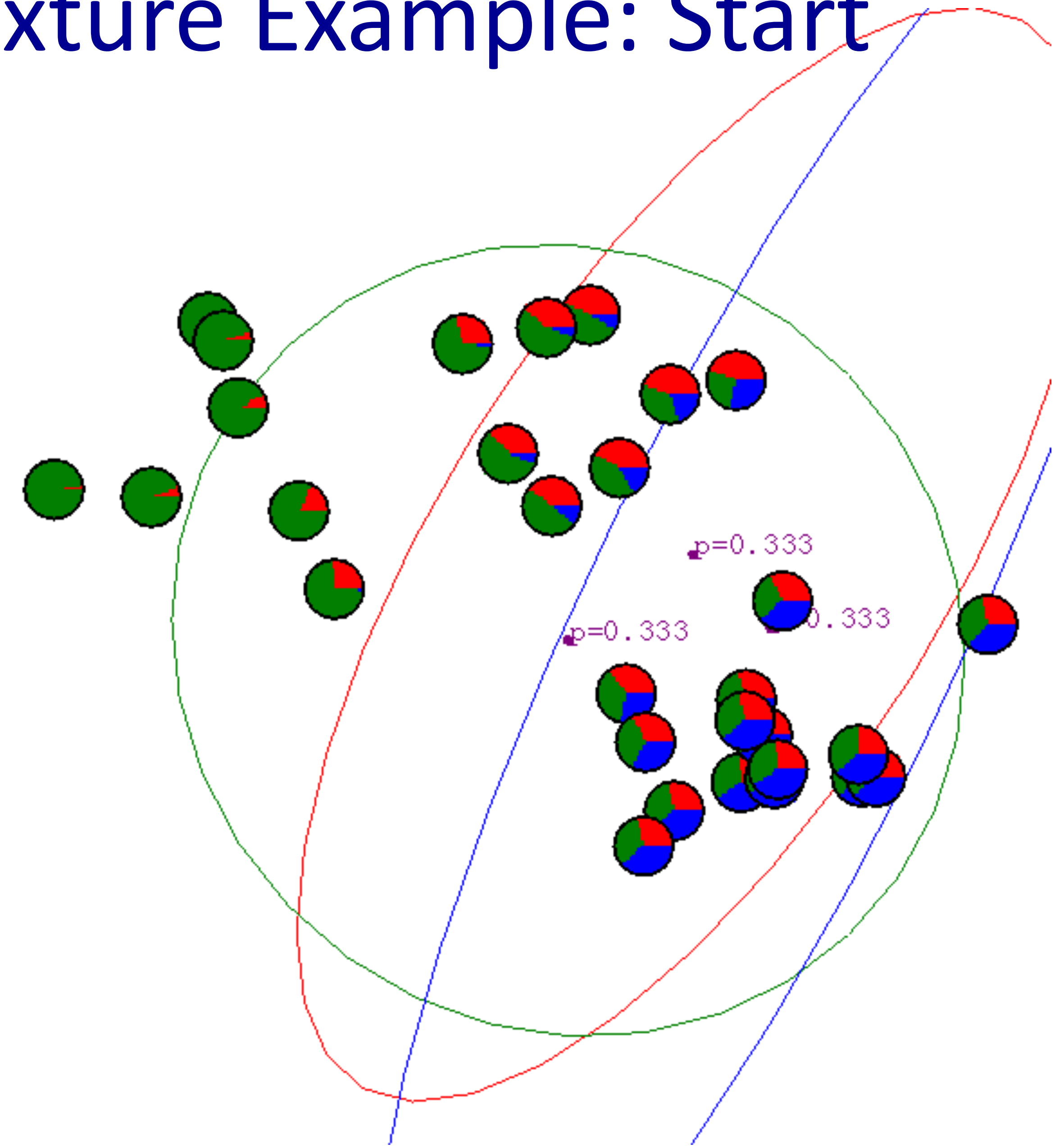
Data: $\{x_j \mid j=1 .. n\}$

- **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: $P(Y_j=k \mid x_j; \theta)$

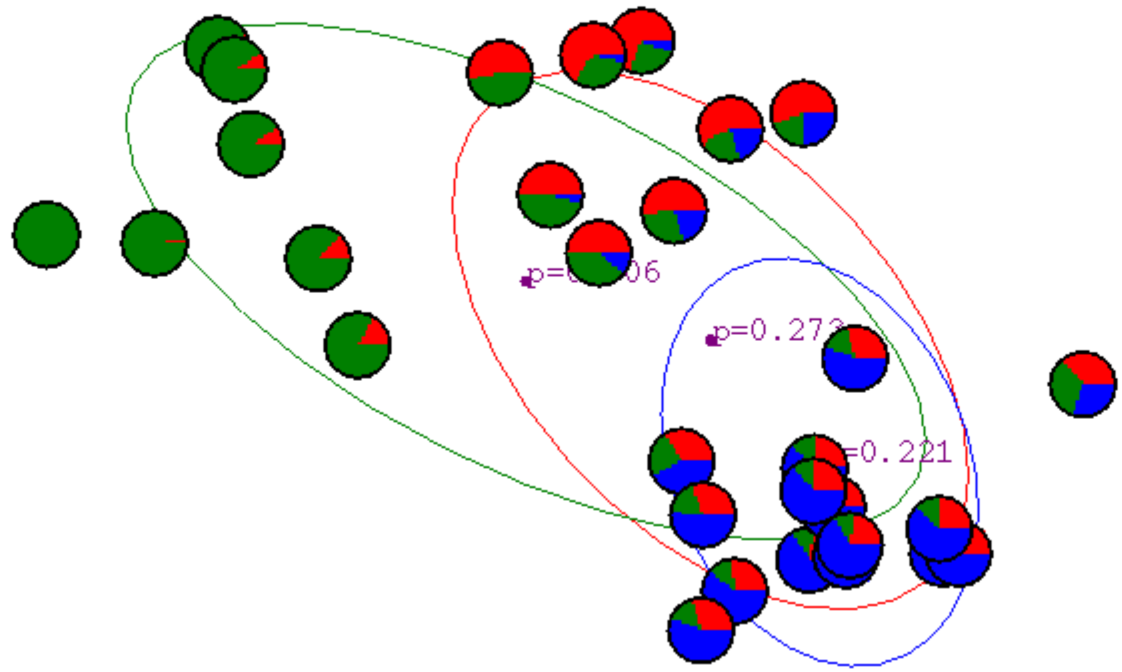
- **M-step**: Re-estimate the parameters with "weighted" MLE estimates
  – Set $\theta^{new} = \text{argmax}_\theta \sum_j \sum_k P(Y_j=k \mid x_j ;\theta^{old}) \log P(Y_j=k, x_j ; \theta)$

Particularly useful when the E and M steps have closed form solutions

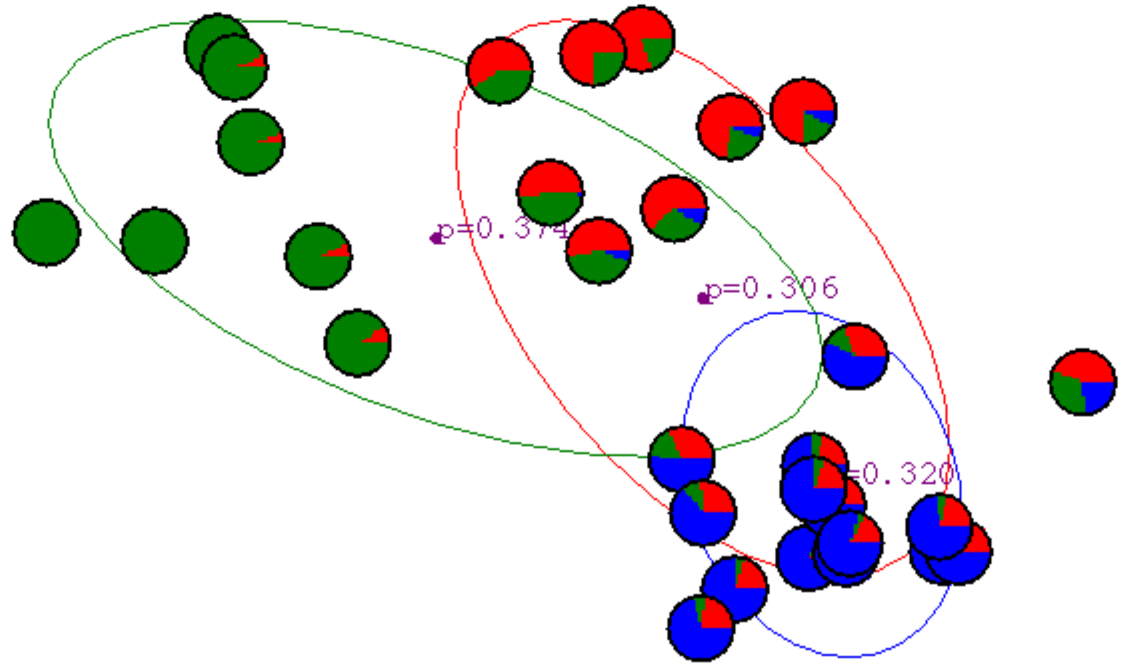
Gaussian Mixture Example: Start



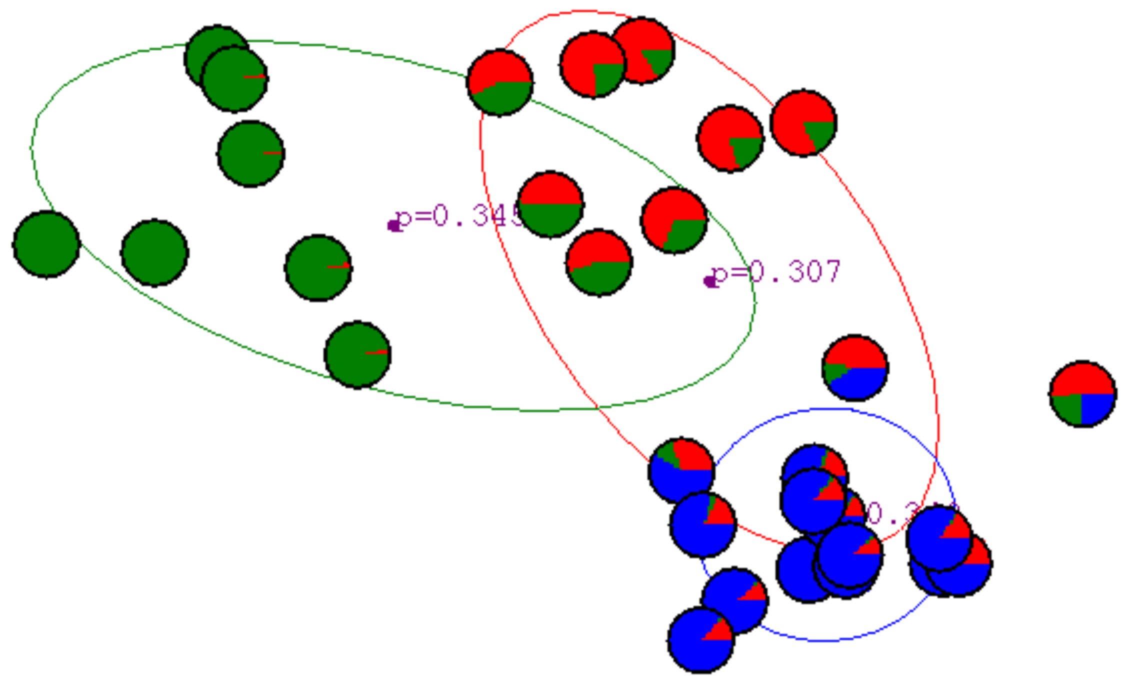
After first iteration



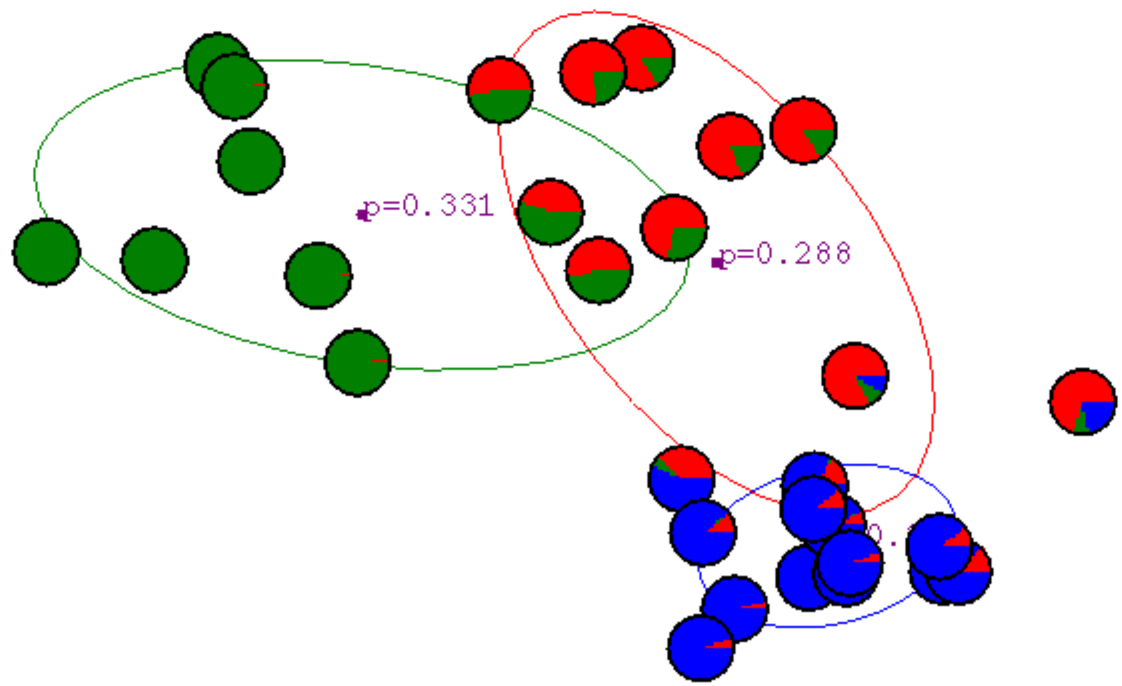
After 2nd iteration



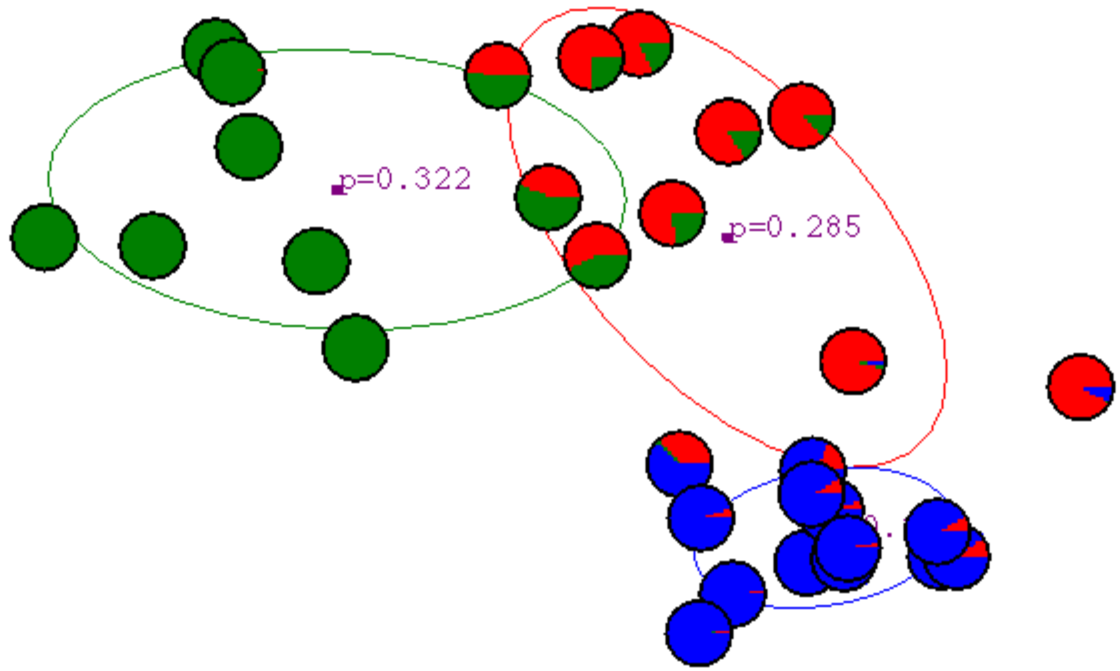
After 3rd iteration



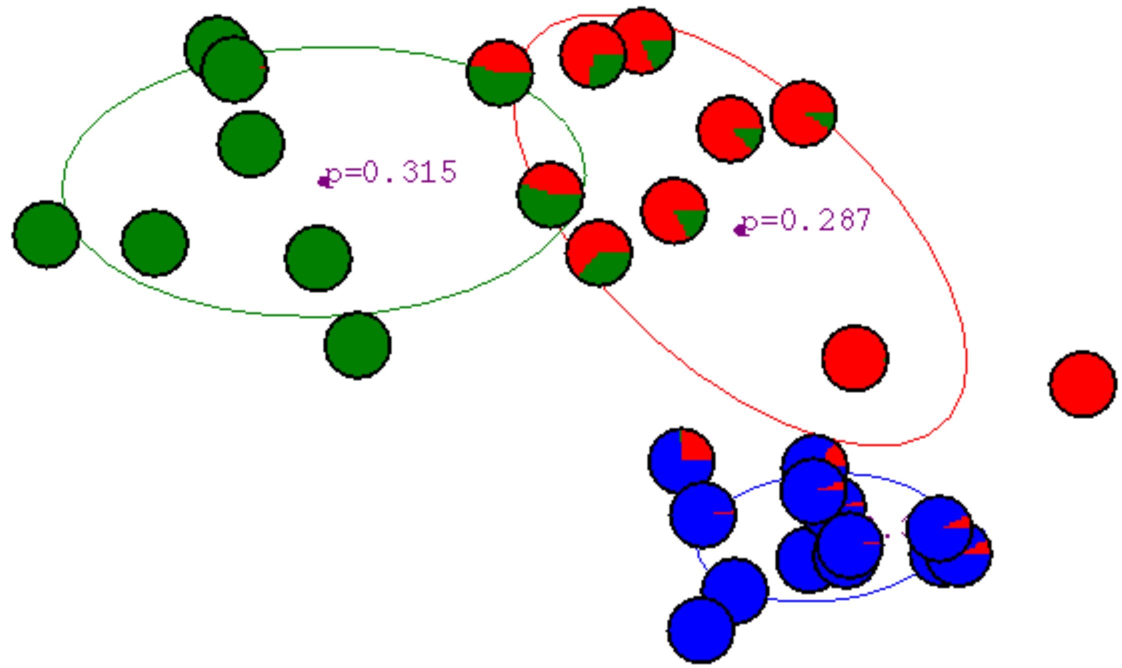
After 4th iteration



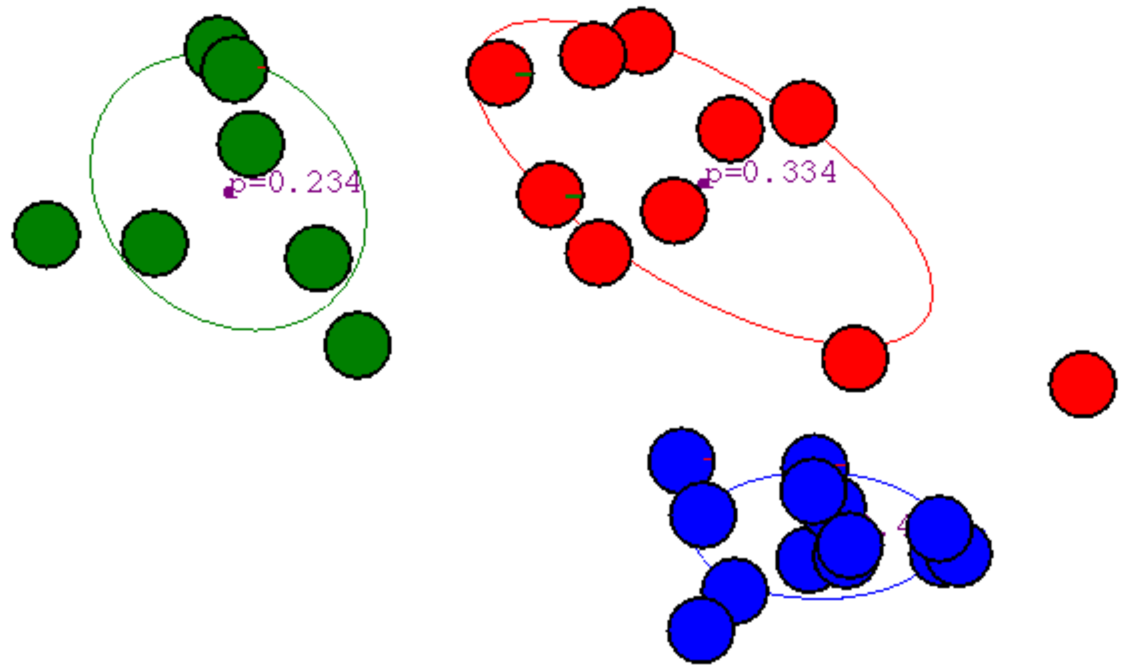
After 5th iteration



After 6th iteration



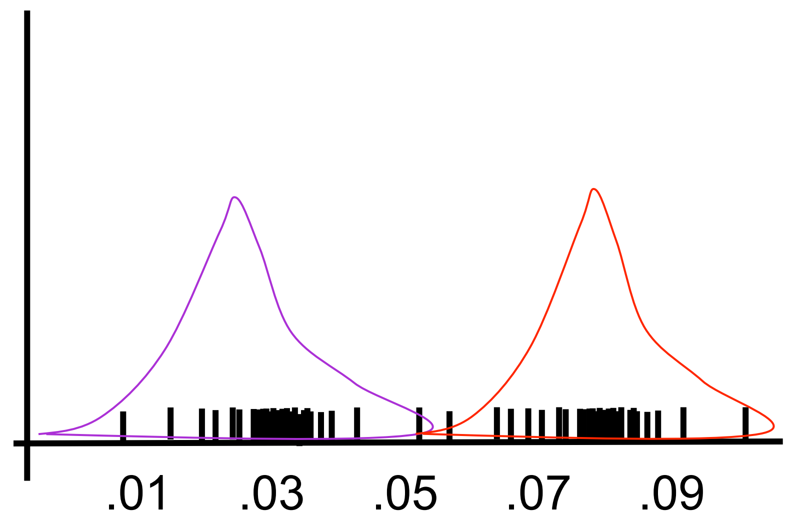
After 20th iteration



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



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

EM for GMMs: only learning means

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

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

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

$p_k^{(t)}$ is shorthand for estimate of $P(y=k)$ 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

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

Just evaluate a Gaussian at x_j

M-step

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

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

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

m = #training examples

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

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

M-step

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

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

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

Equivalent to k-means clustering algorithm!!!

Properties of EM

- We will prove that
  - EM converges to a local maxima
  - Each iteration improves the log-likelihood
- How? (Same as k-means)
  - E-step can never decrease likelihood
  - M-step can never decrease likelihood

The general learning problem with missing data

- Marginal likelihood: **X** is observed,

  Z (e.g. the class labels **Y**) 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 $l(\theta:\text{Data})$
- **Assuming hidden variables are *missing completely at random* (otherwise, we should explicitly model *why* the values are missing)**