Expectation Maximization
Lecture 22

David Sontag

New York University

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

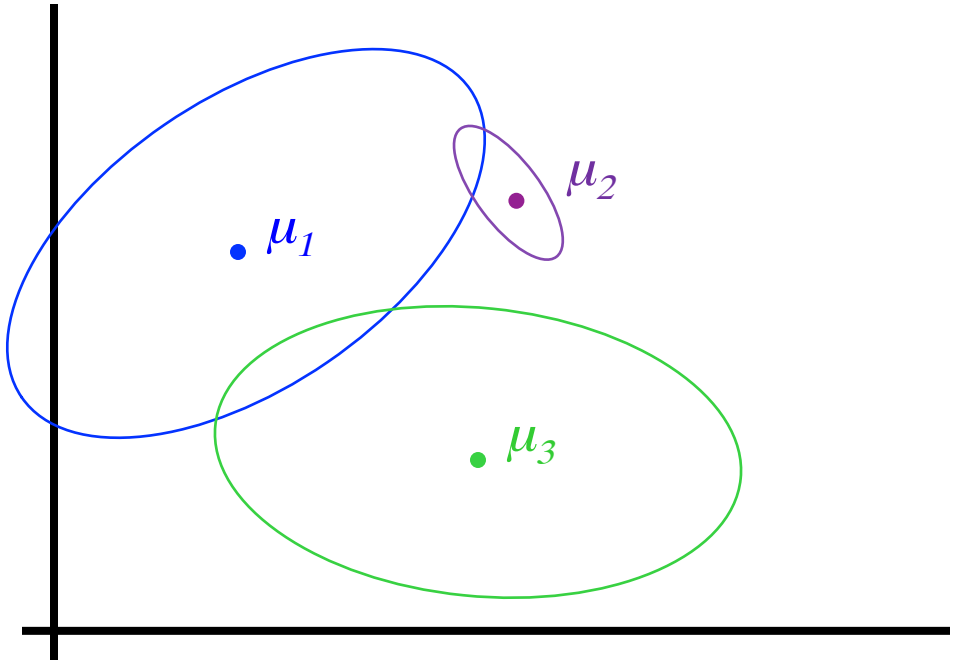
The General GMM assumption

- 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*:**

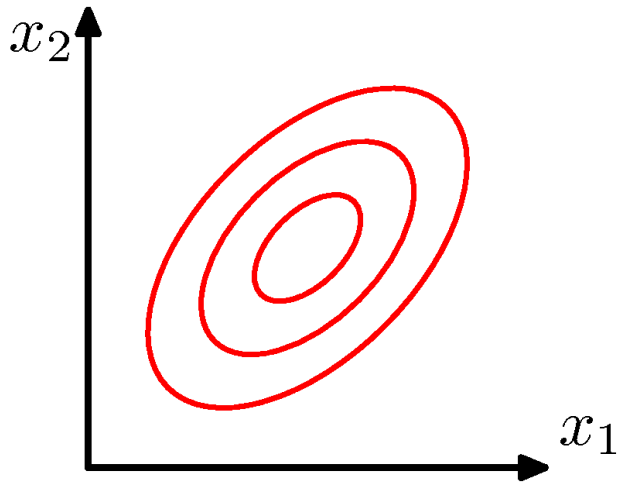
1. Choose component i with probability *P(y=i)*
2. Generate datapoint ~ N(m_i, Σ_i)

Gaussian mixture model (GMM)



Multivariate Gaussians

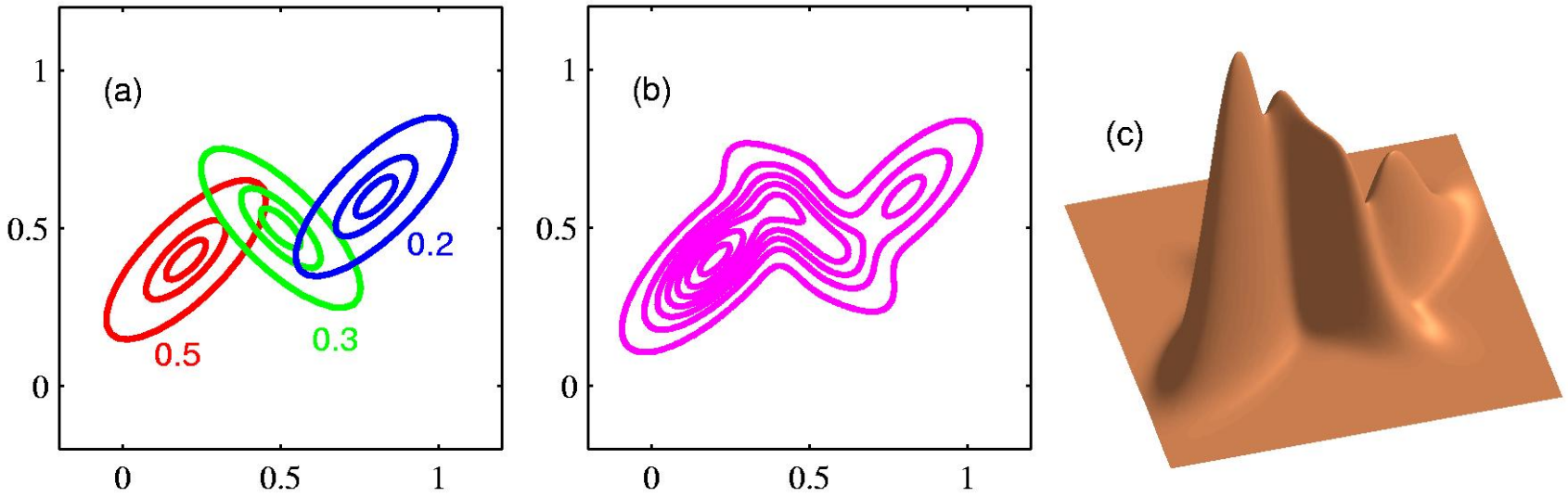
$$P(X=\mathbf{x}_j) = \frac{1}{(2\pi)^{m/2}\,\|\Sigma\|^{1/2}} \exp\left[-\frac{1}{2}\left(\mathbf{x}_j-\mu\right)^T \Sigma^{-1}\left(\mathbf{x}_j-\mu\right)\right]$$



Σ = arbitrary (semidefinite) matrix:

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

Mixtures of Gaussians



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

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

Just evaluate a Gaussian at x_j

M-step

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

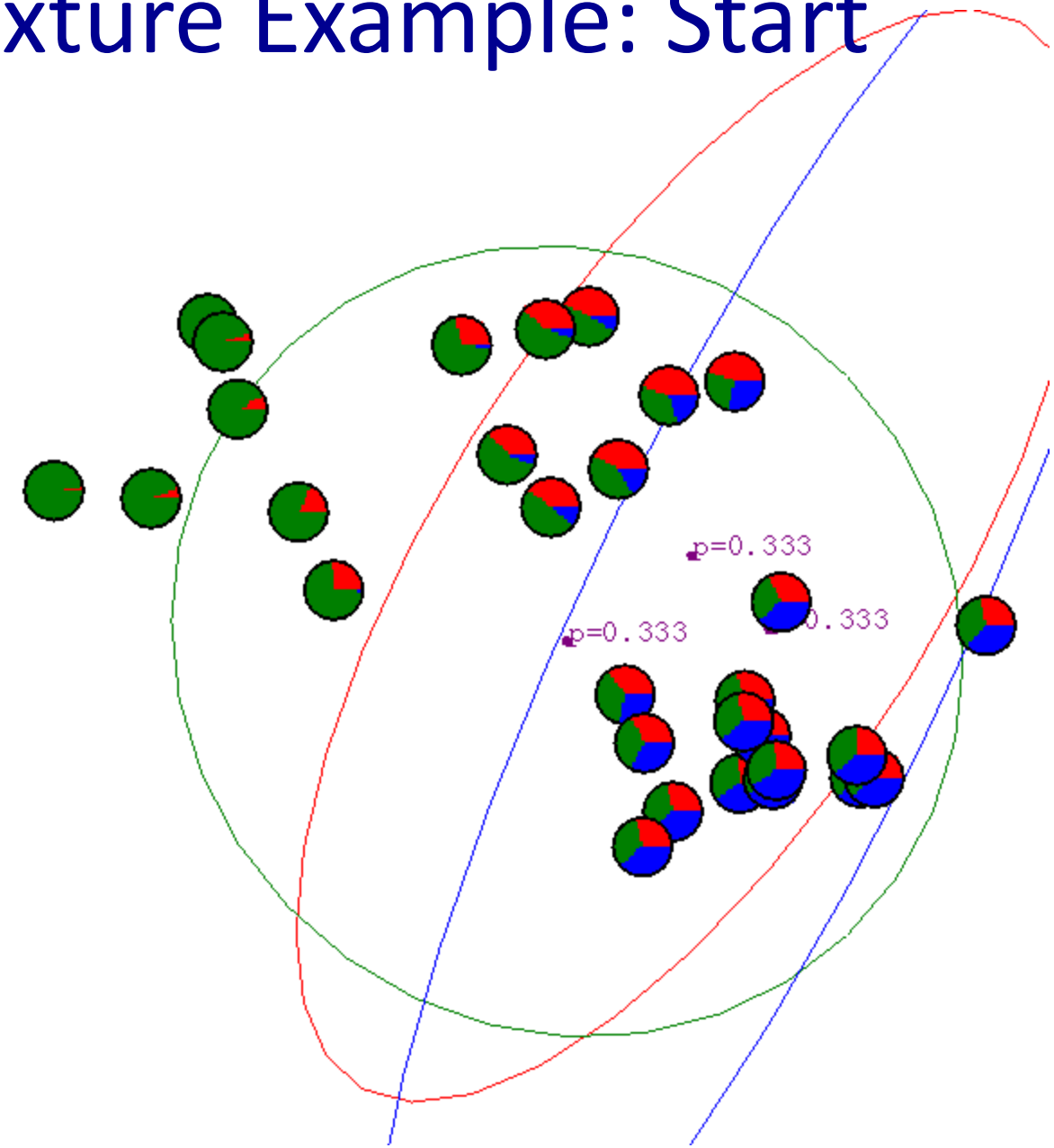
$$\mu_k^{(t+1)} = \frac{\sum_j P(Y_j = k | x_j, \lambda_t) x_j}{\sum_j P(Y_j = k | x_j, \lambda_t)}$$

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

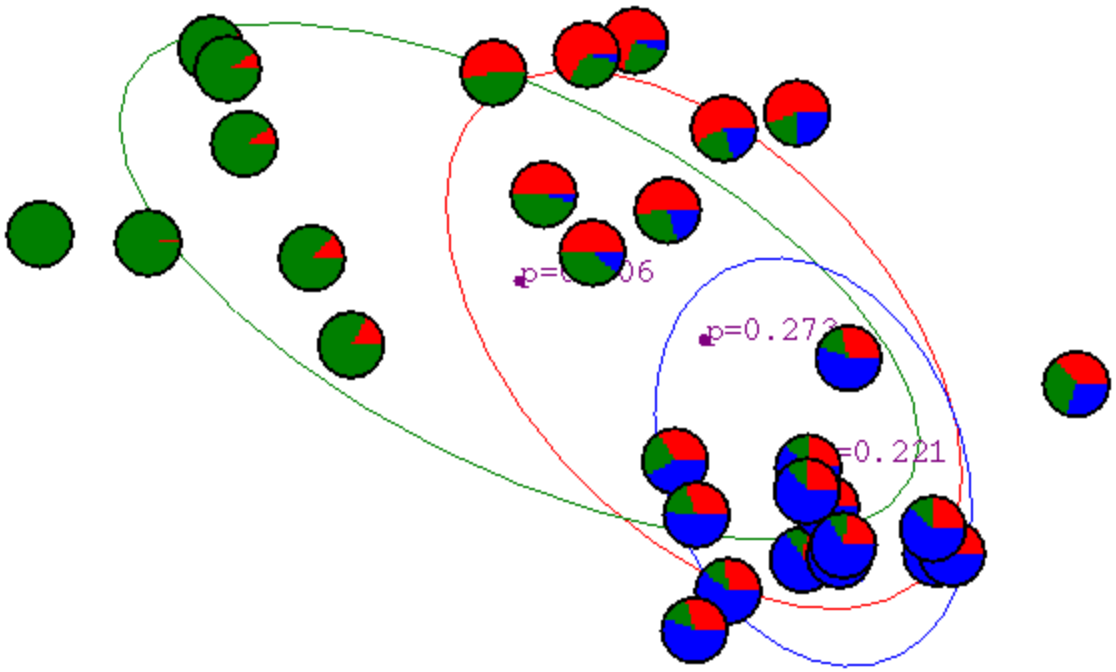
$$p_k^{(t+1)} = \frac{\sum_j P(Y_j = k | x_j, \lambda_t)}{m}$$

m = #training examples

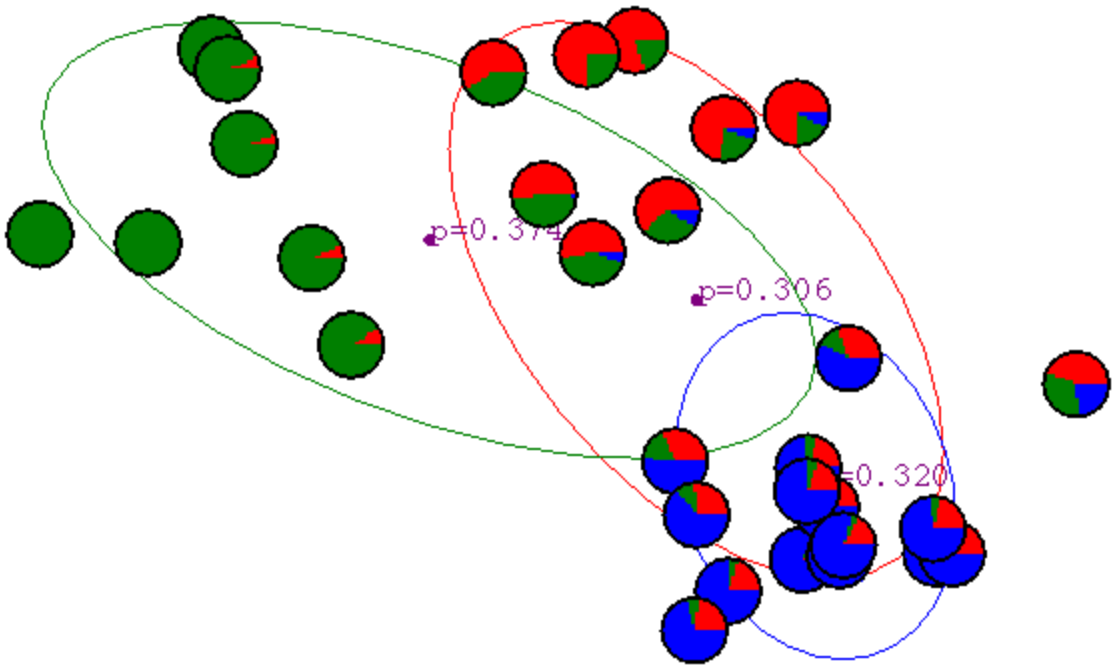
Gaussian Mixture Example: Start



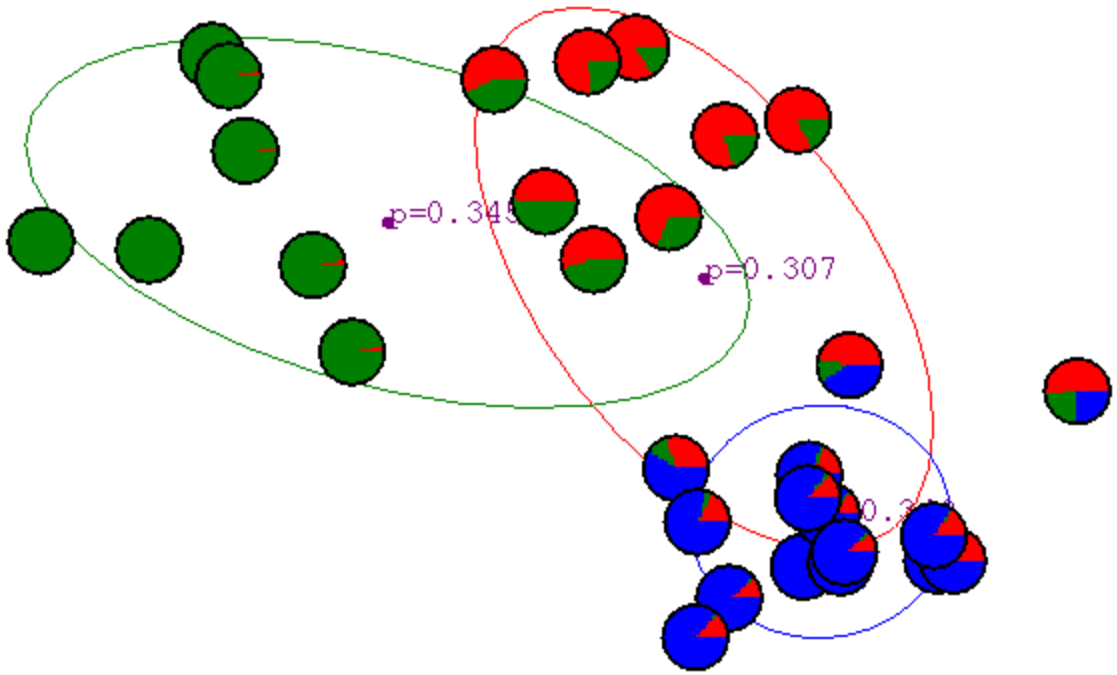
After first iteration



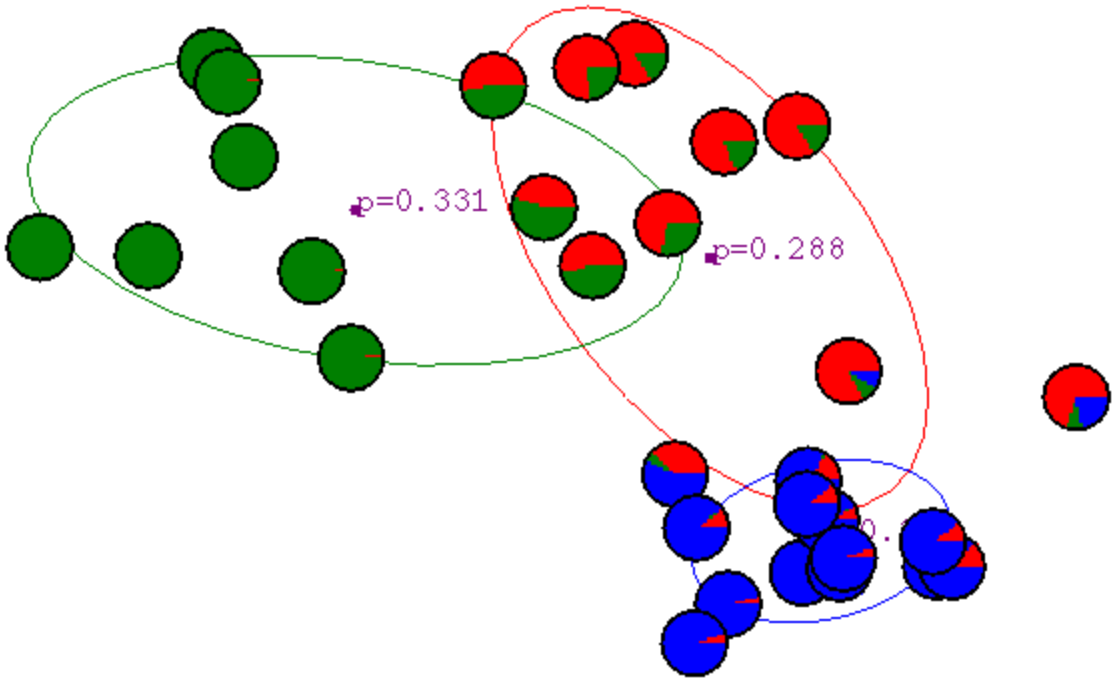
After 2nd iteration



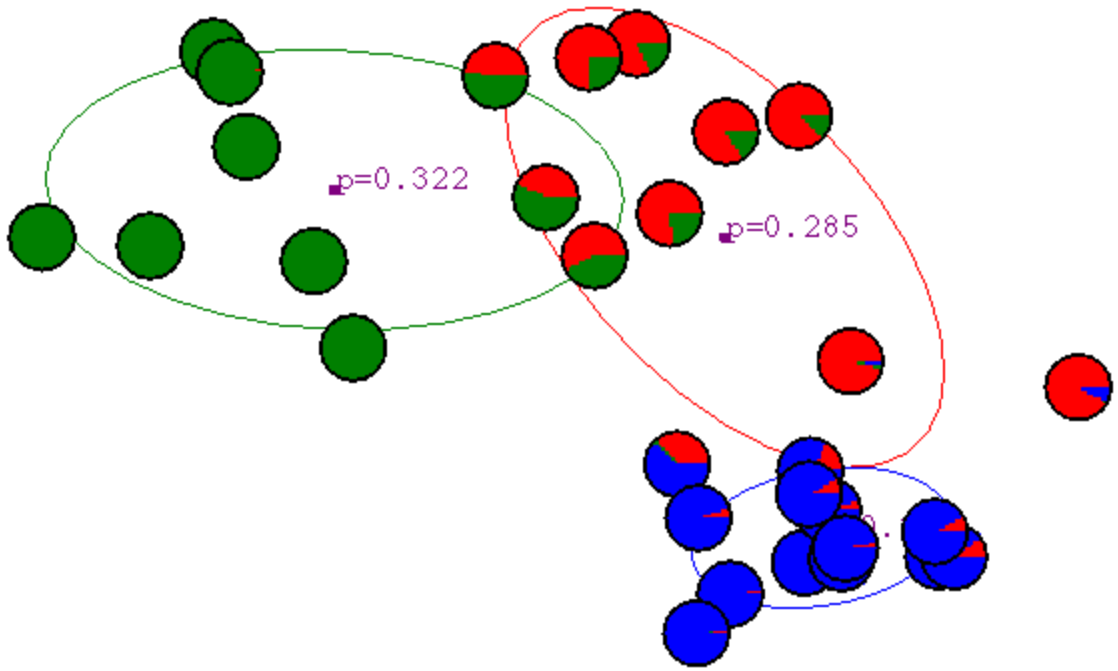
After 3rd iteration



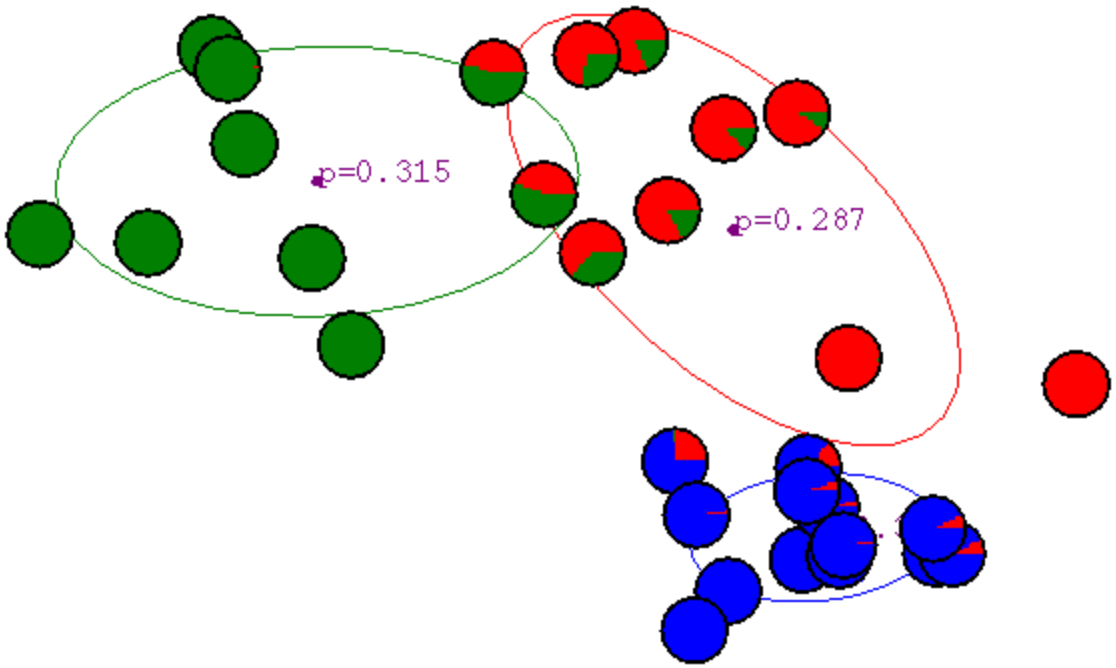
After 4th iteration



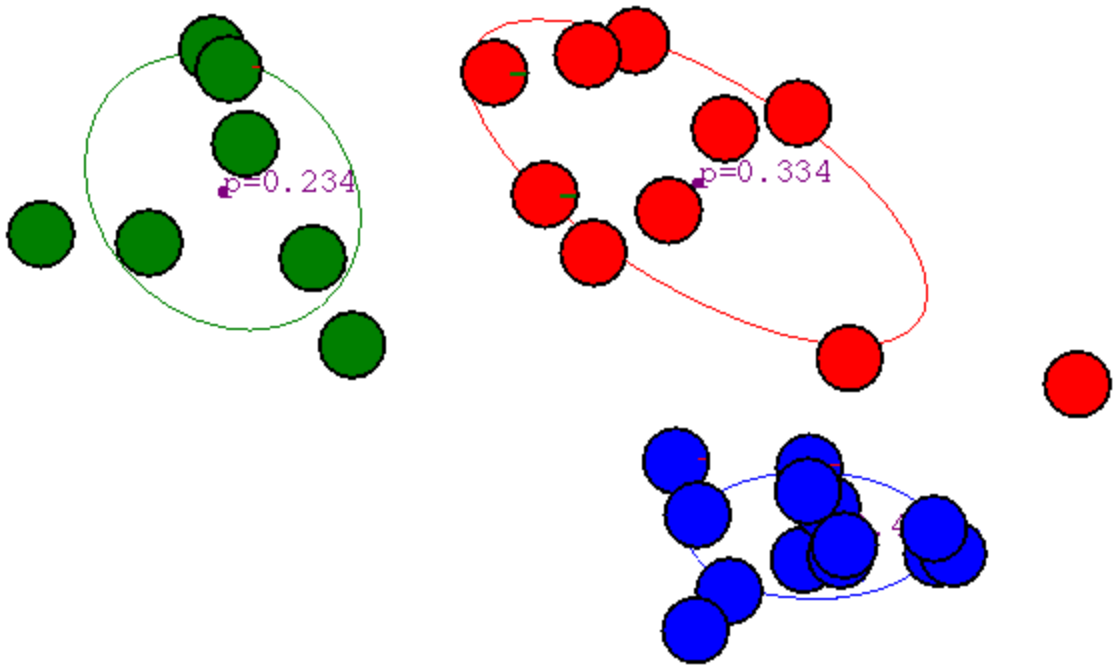
After 5th iteration



After 6th iteration



After 20th iteration



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

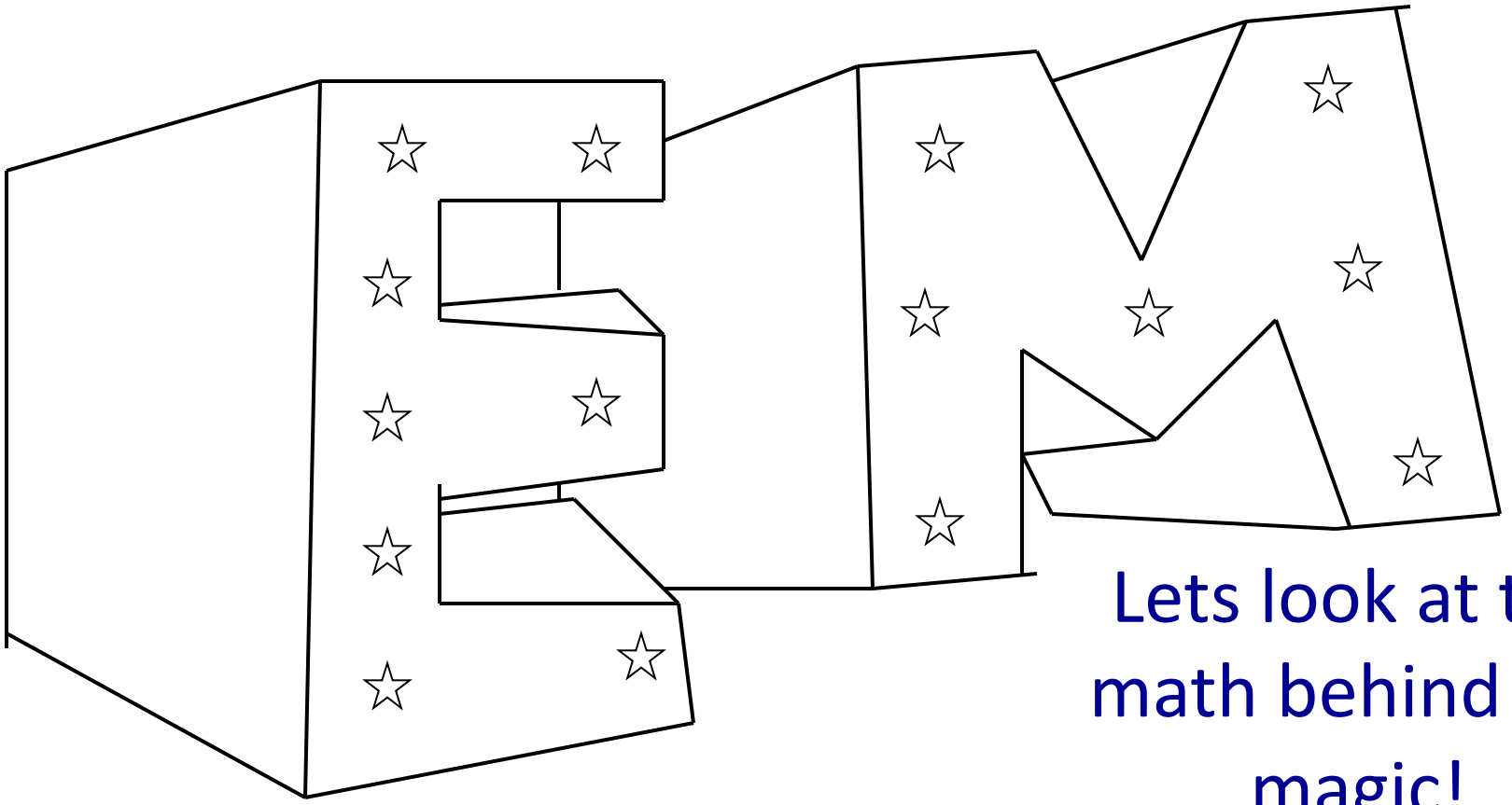
Compute most likely new **μ**s given class expectations

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

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

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

Equivalent to k-means clustering algorithm!!!



Lets look at the math behind the magic!

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(θ:Data)

A Key Computation: E-step

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

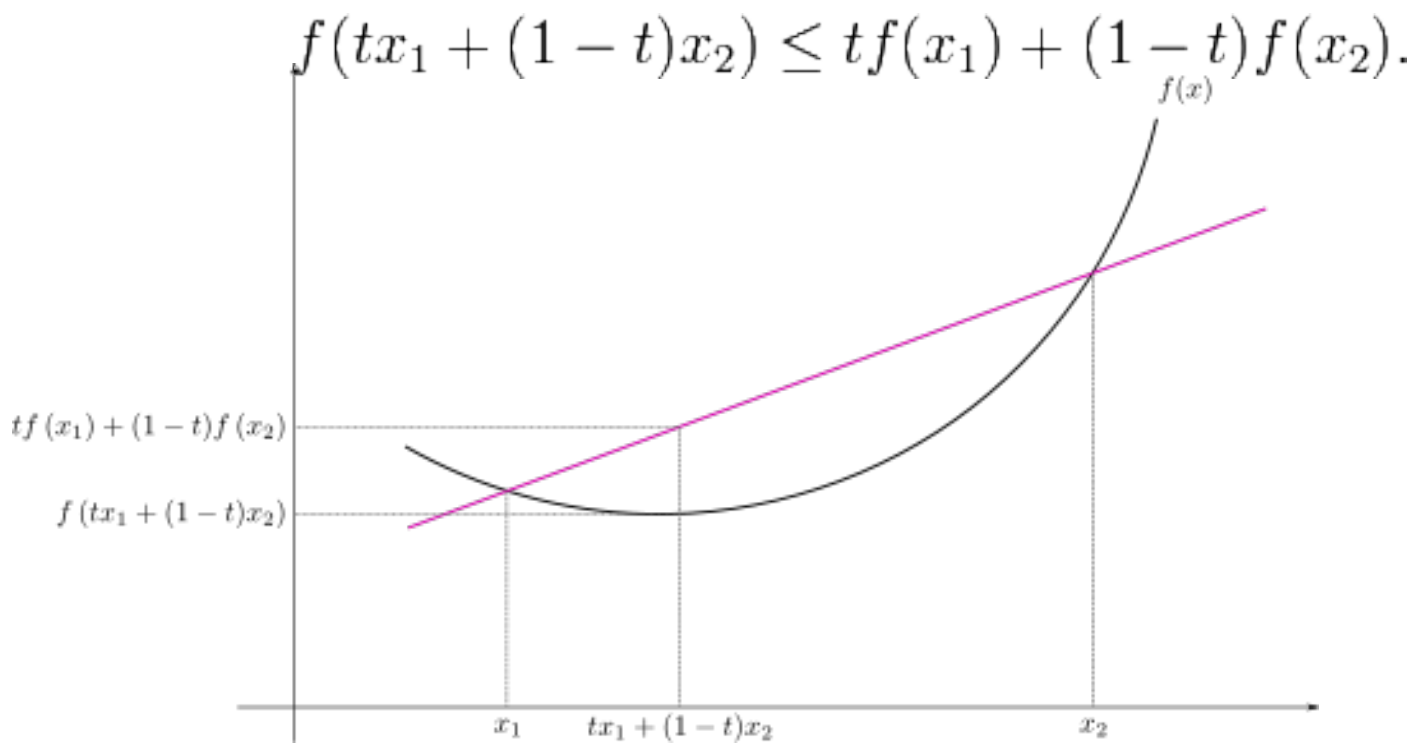
$$Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}_j) \;=\; P(\mathbf{z} \mid \mathbf{x}_j, \theta^{(t)})$$

Properties of EM

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

Jensen's inequality

- **Theorem**: log $\sum_z$ P(**z**) f(**z**) $\geq$ $\sum_z$ P(**z**) log f(**z**)
  - **e.g., Binary case for convex function f:**



Applying Jensen's inequality

- Use: $\log \sum_z P(z)\, f(z) \geq \sum_z P(z) \log f(z)$

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

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

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

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

The M-step

Lower bound:

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

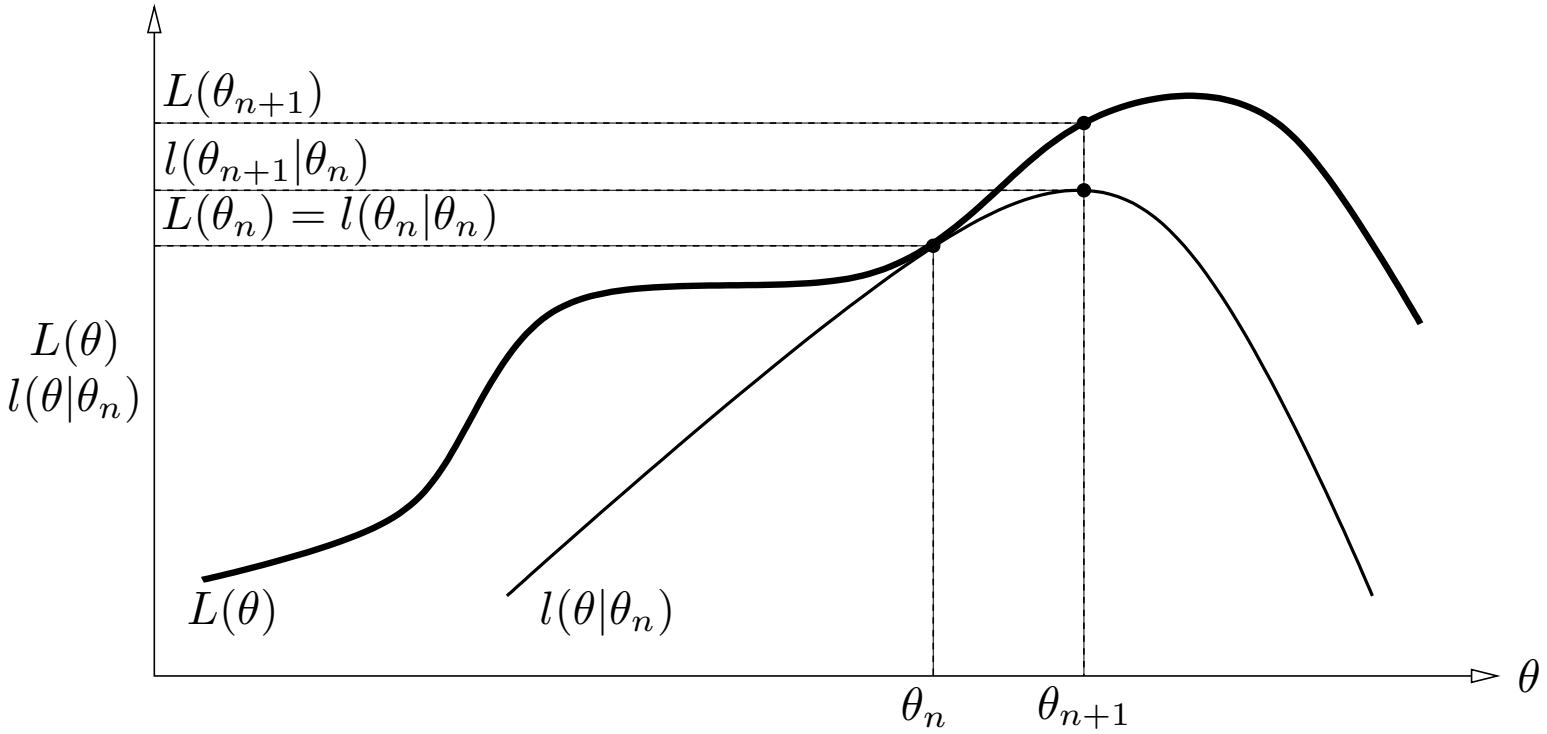
This term is a constant with respect to θ

- 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!

EM pictorially



(Figure from tutorial by Sean Borman)

What you should know

- Mixture of Gaussians
- EM for mixture of Gaussians:
  - Coordinate ascent, just like k-means
  - 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***