CS60050
Machine Learning
GMM

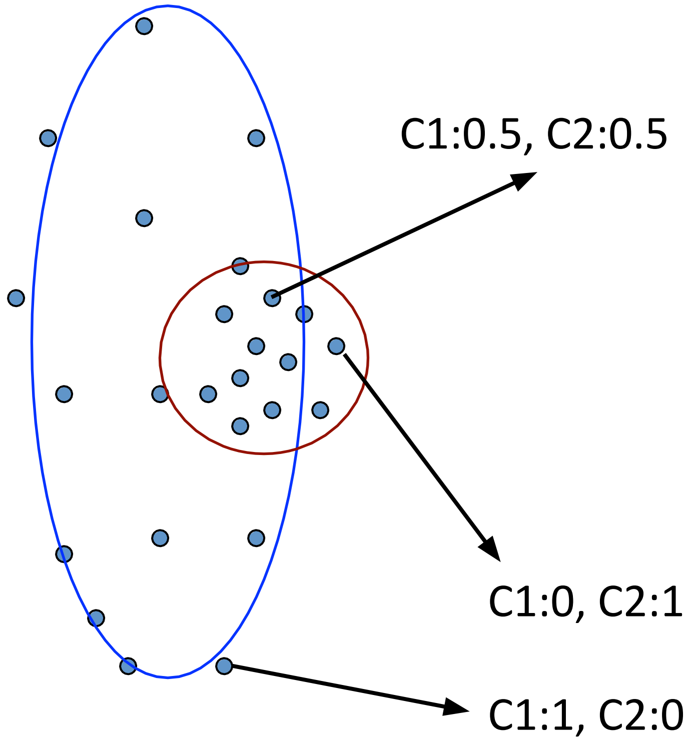
Somak Aditya Sudeshna Sarkar
Department of CSE, IIT Kharagpur

Slide sources

- David Sontag New York University
  - Slides adapted from Carlos Guestrin, Dan Klein, Luke Zettlemoyer, Dan Weld, Vibhav Gogate, and Andrew Moore
- Matt Gormley (CMU)
  - **10-601B Introduction to Machine Learning**

Issues with Clustering

- Clusters may overlap
- Some clusters may be "wider" than others
- Can we model this explicitly?
- With what **probability** is a point from a cluster? (soft clustering)



Probabilistic Clustering

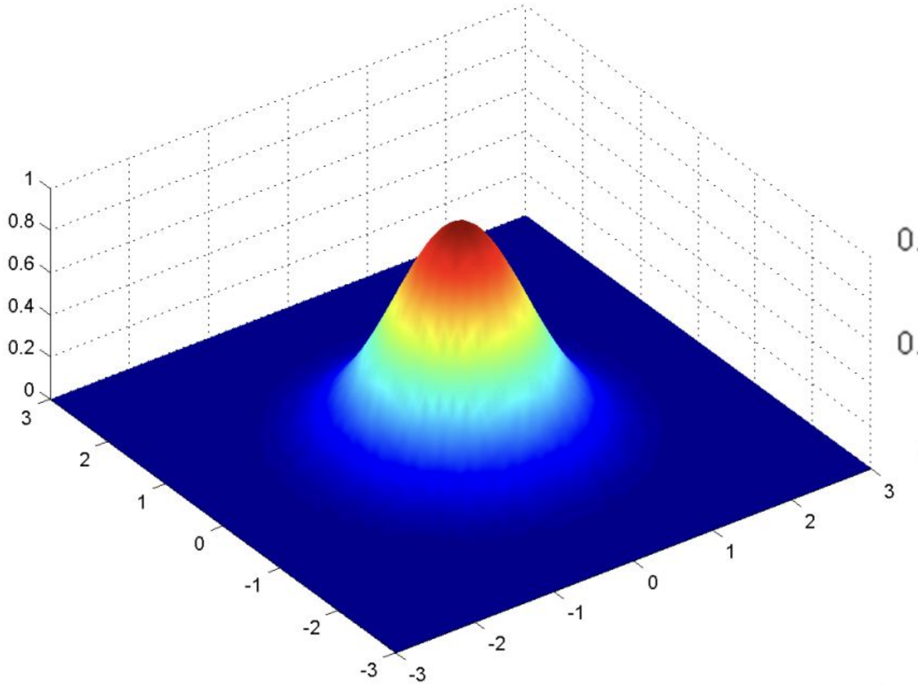
- Allow overlaps, clusters of different size, etc.
- Can tell a generative story for data
  - $P(Y)P(X|Y)$
- Challenge: we need to estimate model parameters without labeled Ys

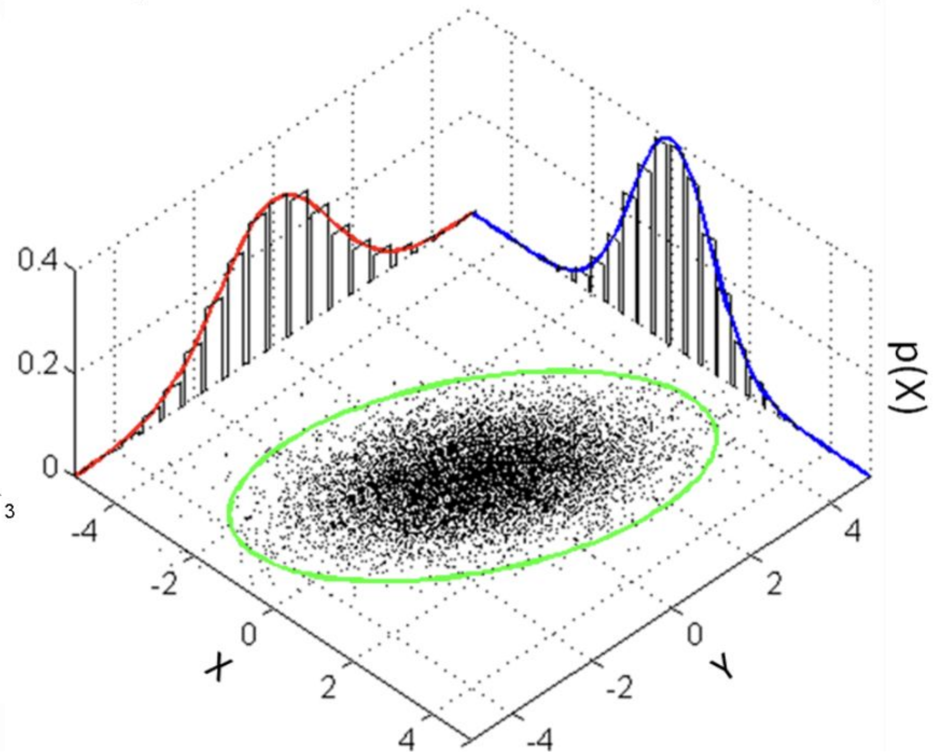
Slide adapted from David Sontag

Recall Gaussian Distribution

- Recall the Gaussian distribution:

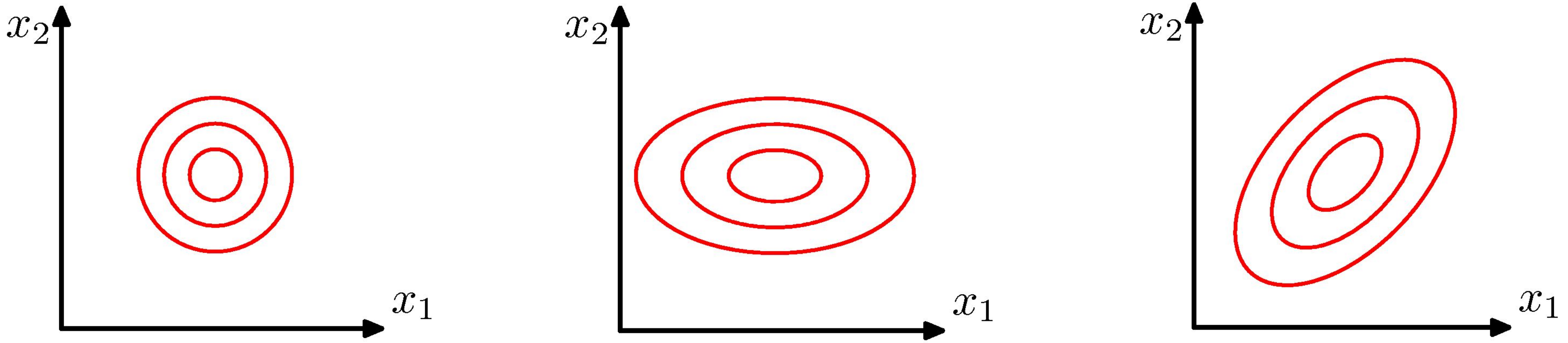
$$P(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\intercal \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$





Multivariate Gaussians

- $$P(X = x_j | \mu, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2} \|\boldsymbol{\Sigma}\|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(x_j - \mu)^T \boldsymbol{\Sigma}^{-1}(x_j - \mu)\right]$$



$\Sigma \propto$ identity matrix

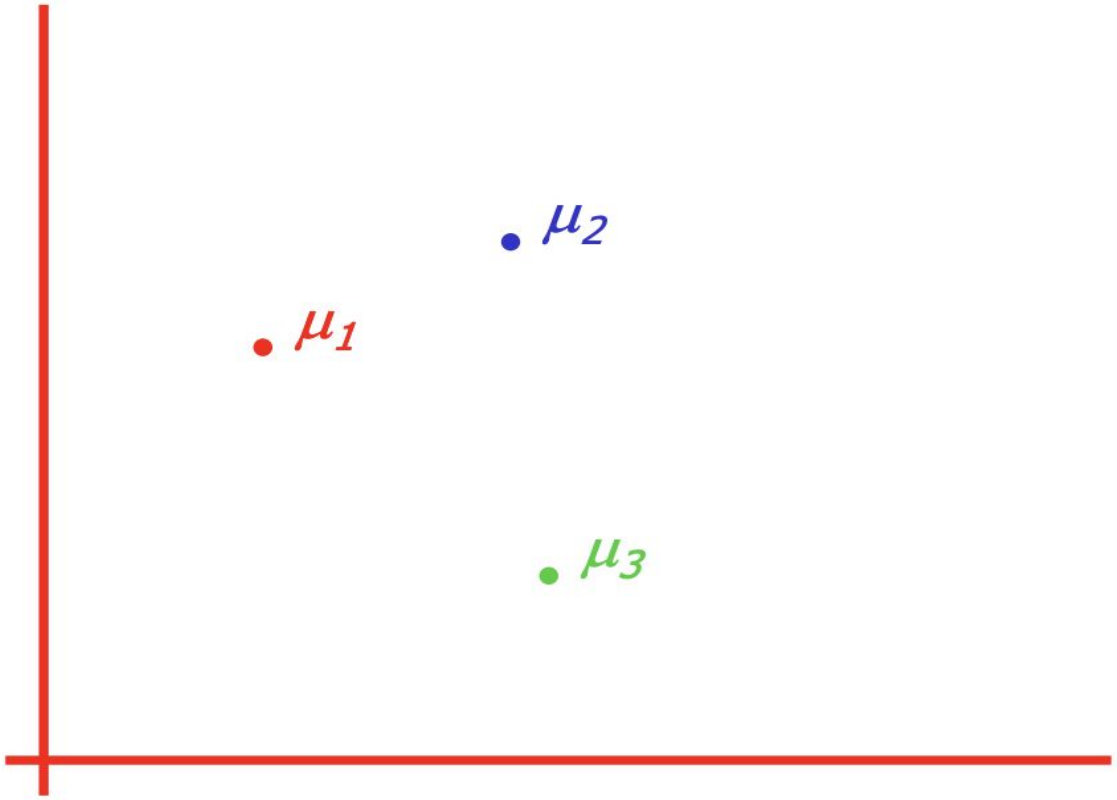
$\Sigma =$ diagonal matrix
X_i are independent

$\Sigma =$ arbitrary (semidefinite) matrix
X_i are independent

Slide adapted from David Sontag

The GMM Assumption

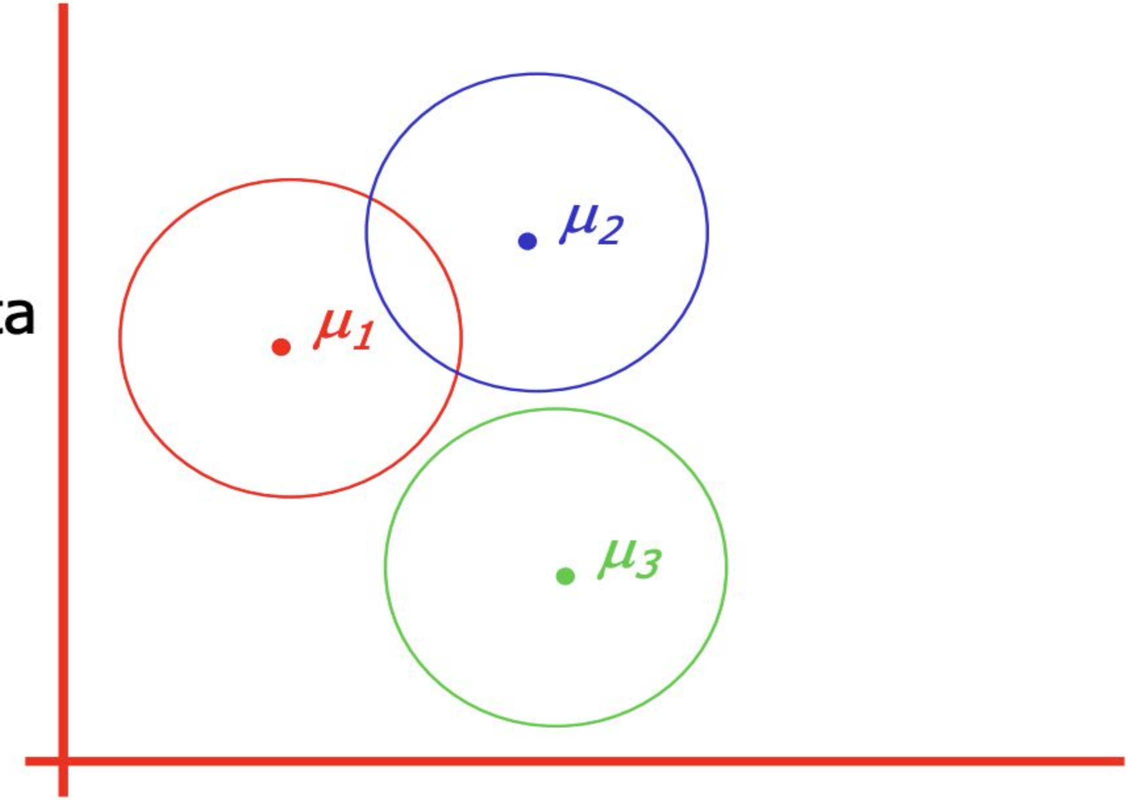
- There are k components. The i'th component is called ω_i
- Component ω_i has an associated mean vector μ_i



The GMM Assumption

- There are k components. The i'th component is called ω_i
- Component ω_i has an associated mean vector μ_i
- Each component generates data from a Gaussian with mean μ_i and covariance matrix $\sigma^2 \boldsymbol{I}$

Assume that each datapoint is generated according to the following recipe:

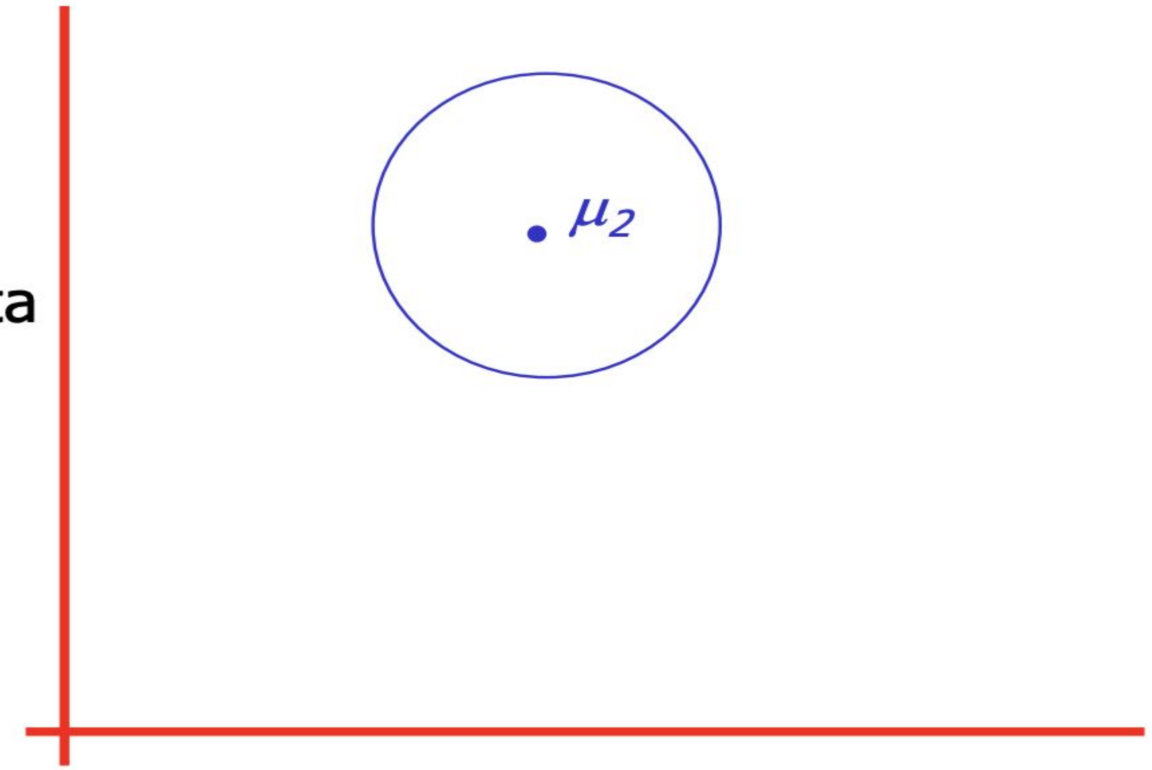


The GMM Assumption

- There are k components. The i'th component is called ω_i
- Component ω_i has an associated mean vector μ_i
- Each component generates data from a Gaussian with mean μ_i and covariance matrix $\sigma^2 \boldsymbol{I}$

Assume that each datapoint is generated according to the following recipe:

1. Pick a component at random: choose component i with probability $P(\omega_i)$.

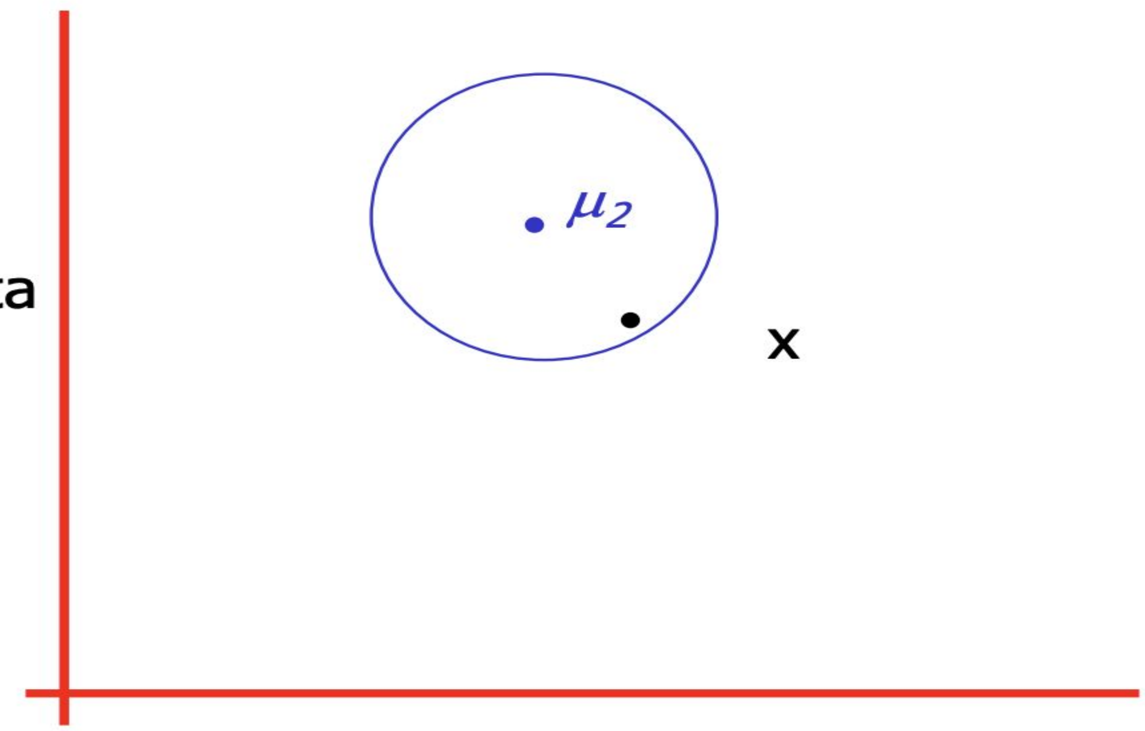


The GMM Assumption

- There are k components. The i'th component is called ω_i
- Component ω_i has an associated mean vector μ_i
- Each component generates data from a Gaussian with mean μ_i and covariance matrix $\sigma^2 \boldsymbol{I}$

Assume that each datapoint is generated according to the following recipe:

1. Pick a component at random: choose component i with probability $P(\omega_i)$.
2. Datapoint ~ $N(\mu_i, \sigma^2 \boldsymbol{I})$

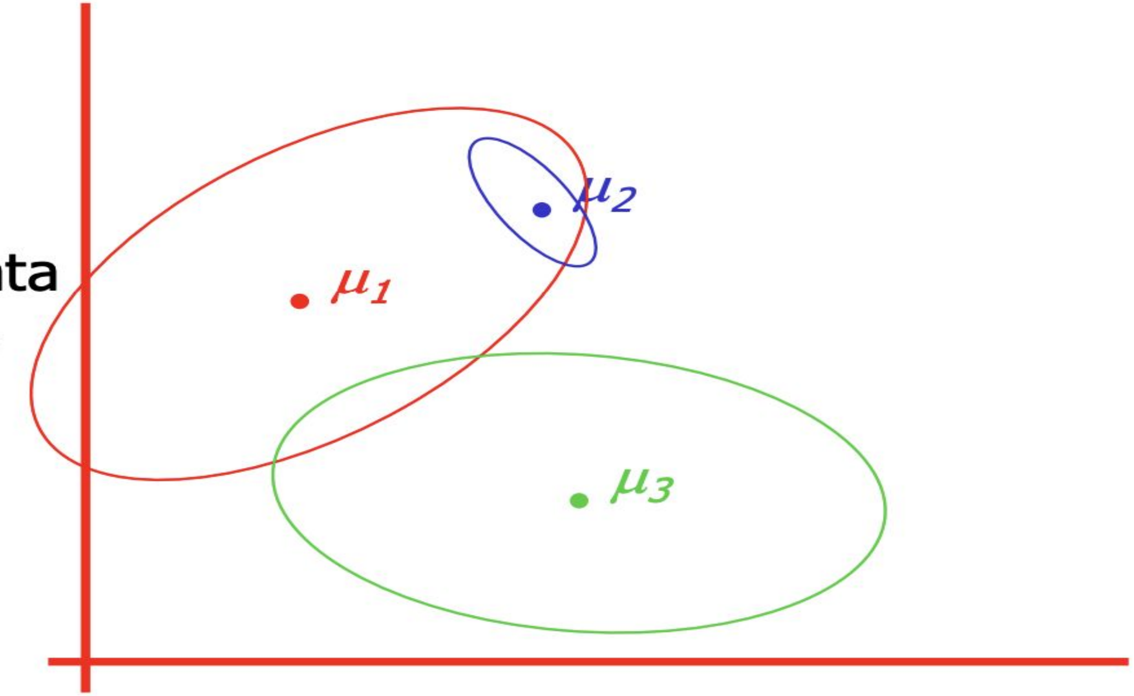


The General GMM assumption

- There are k components. The i'th component is called ω_i
- Component ω_i has an associated mean vector μ_i
- Each component generates data from a Gaussian with mean μ_i and covariance matrix Σ_i

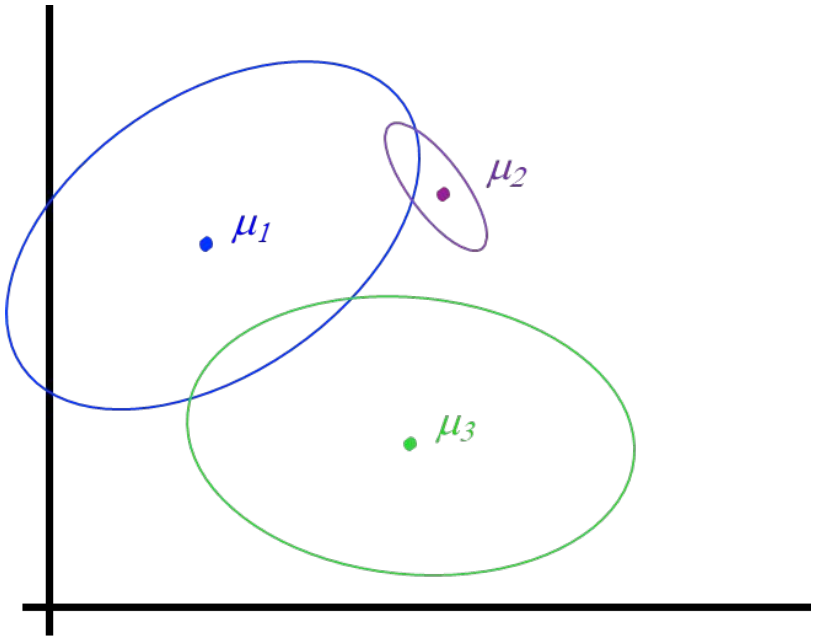
Assume that each datapoint is generated according to the following recipe:

1. Pick a component at random: choose component i with probability $P(\omega_i)$.
2. Datapoint $\sim N(\mu_i, \Sigma_i)$



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 assumed to have been sampled from a ***generative process***:

1. Choose component i with probability $P(y = i)$ [Multinomial]
2. Generate datapoint $\sim N(\mu_i, \Sigma_i)$

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

By fitting this model (unsupervised learning), we can learn new insights about the data

Slide adapted from David Sontag

Mixture Models

- Formally a Mixture Model is the weighted sum of a number of pdfs where the weights are determined by a distribution π

$$p(x) = \pi_0 f_0(x) + \pi_1 f_1(x) + \pi_2 f_2(x) + \ldots + \pi_k f_k(x)$$

$$\text{where } \sum_{i=0}^{k} \pi_i = 1$$

$$\boxed{p(x) = \sum_{i=0}^{k} \pi_i f_i(x)}$$

Gaussian Mixture Models

- GMM: the weighted sum of a number of Gaussians where the weights are determined by a distribution π

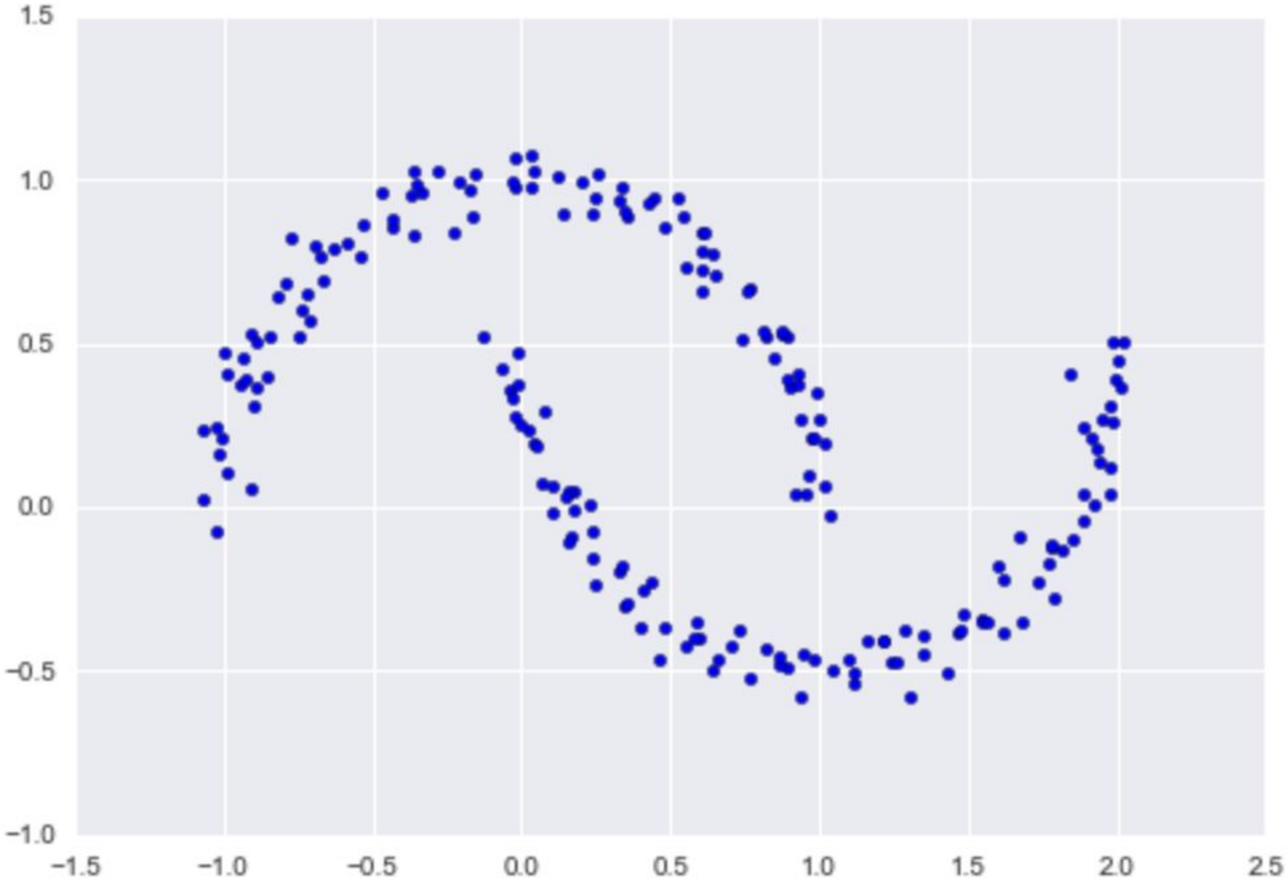
$$p(x) = \pi_0 N(x|\mu_0, \Sigma_0) + \pi_1 N(x|\mu_1, \Sigma_1) + \ldots + \pi_k N(x|\mu_k, \Sigma_k)$$

$$\text{where } \sum_{i=0}^{k} \pi_i = 1$$

$$\boxed{p(x) = \sum_{i=0}^{k} \pi_i N(x|\mu_k, \Sigma_k)}$$

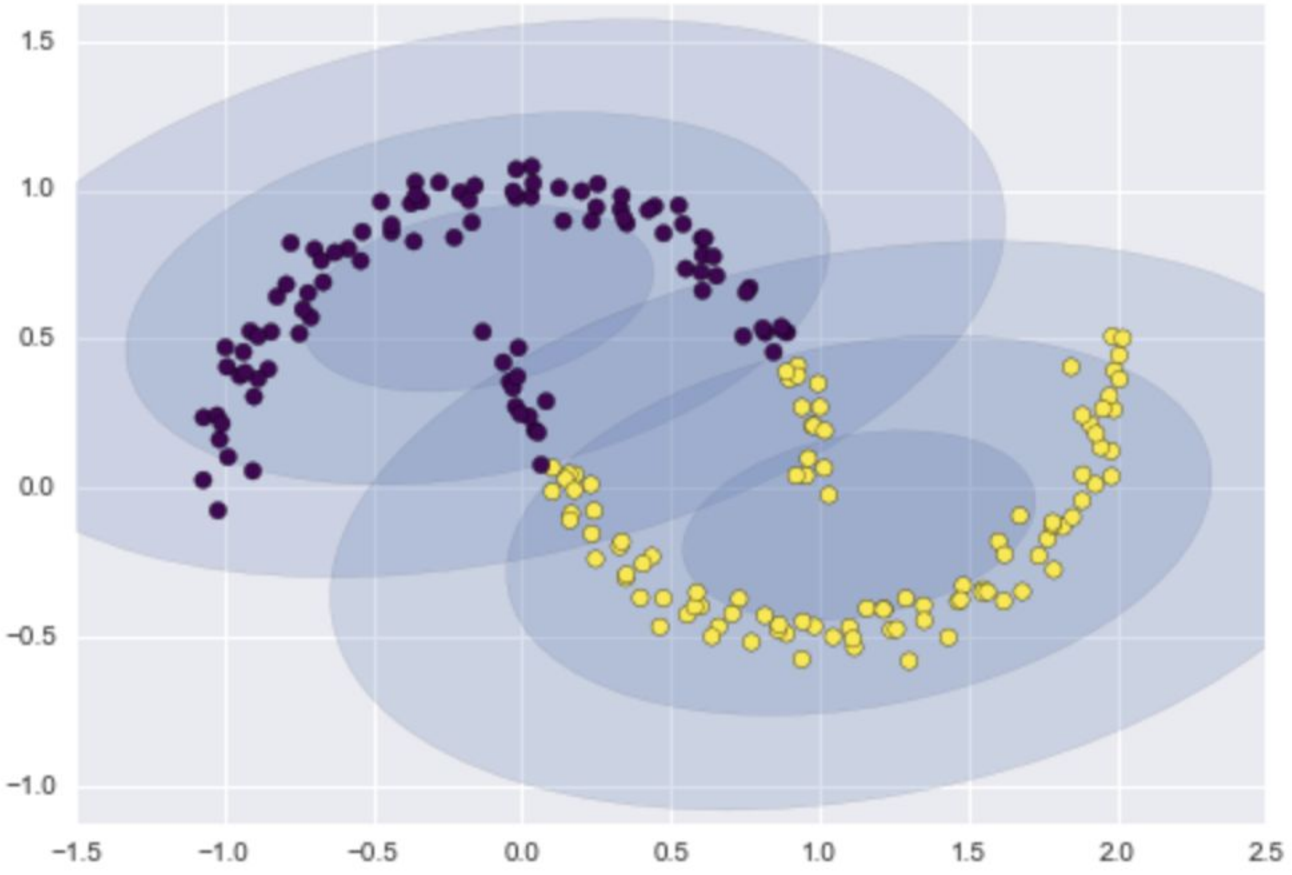
Example Visualization

```python
from sklearn.datasets import make_moons
Xmoon, ymoon = make_moons(200, noise=.05, random_state=0)
plt.scatter(Xmoon[:, 0], Xmoon[:, 1]);
```



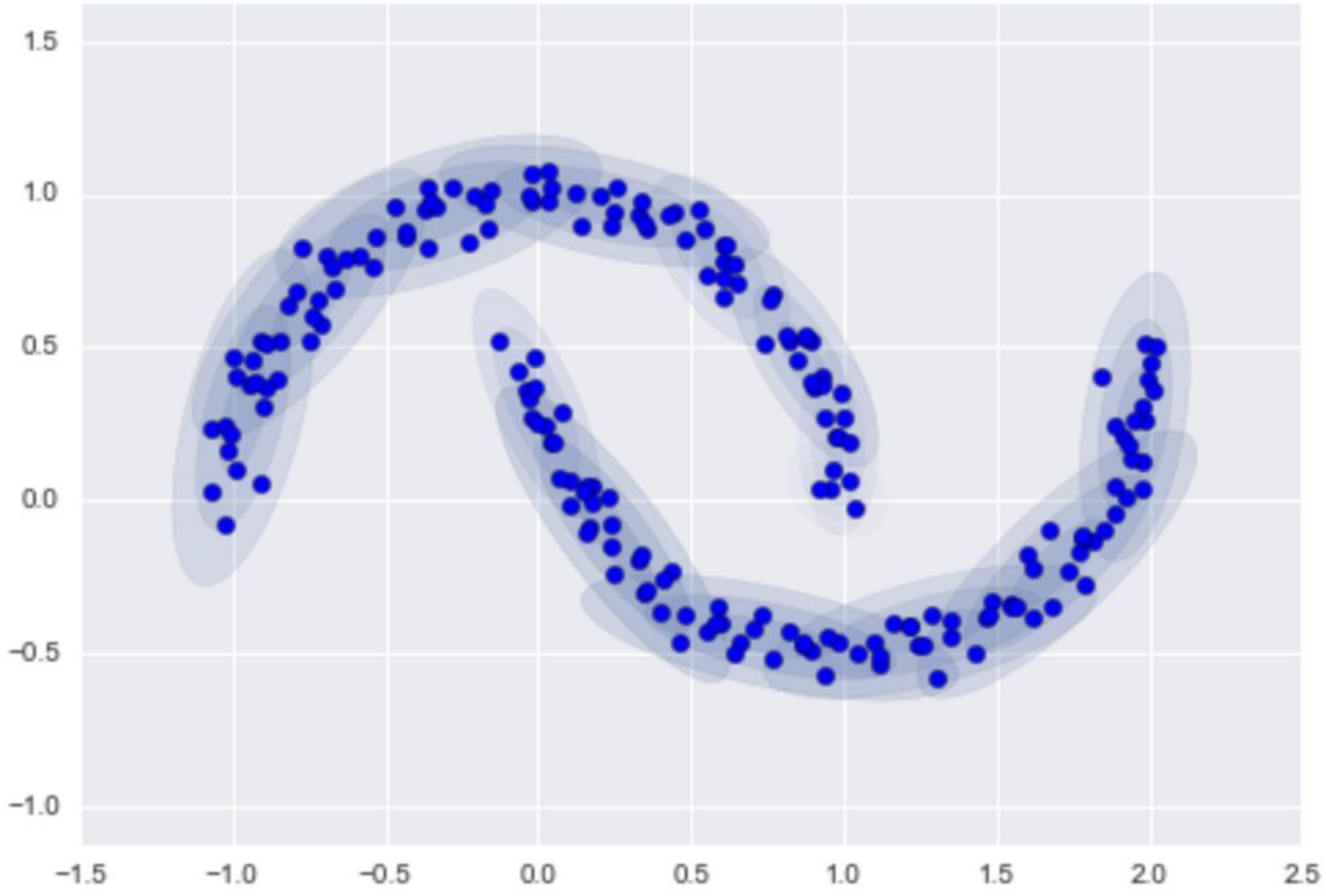
With 2 Components

```
gmm2 = GMM(n_components=2, covariance_type='full', random_state=0)
plot_gmm(gmm2, Xmoon)
```



With 16 components

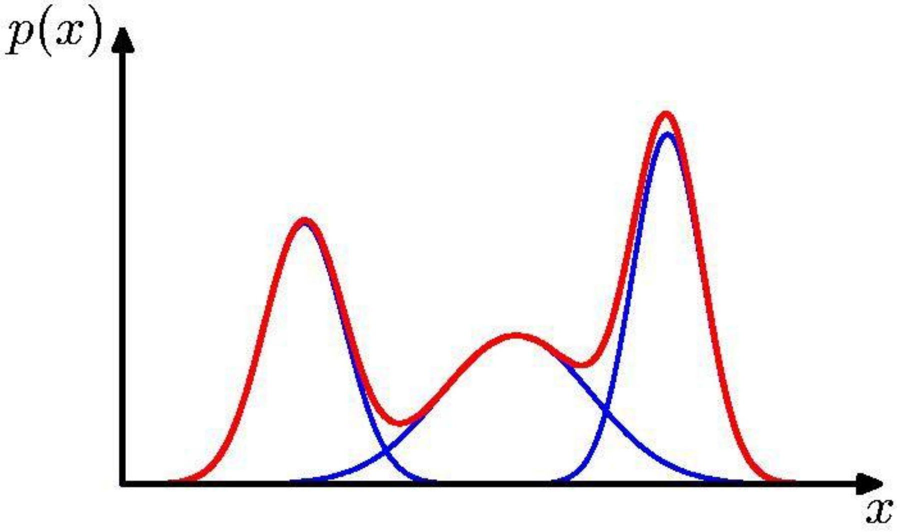
```
gmm16 = GMM(n_components=16, covariance_type='full', random_state=0)
plot_gmm(gmm16, Xmoon, label=False)
```



Marginal distribution for mixtures of Gaussians

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

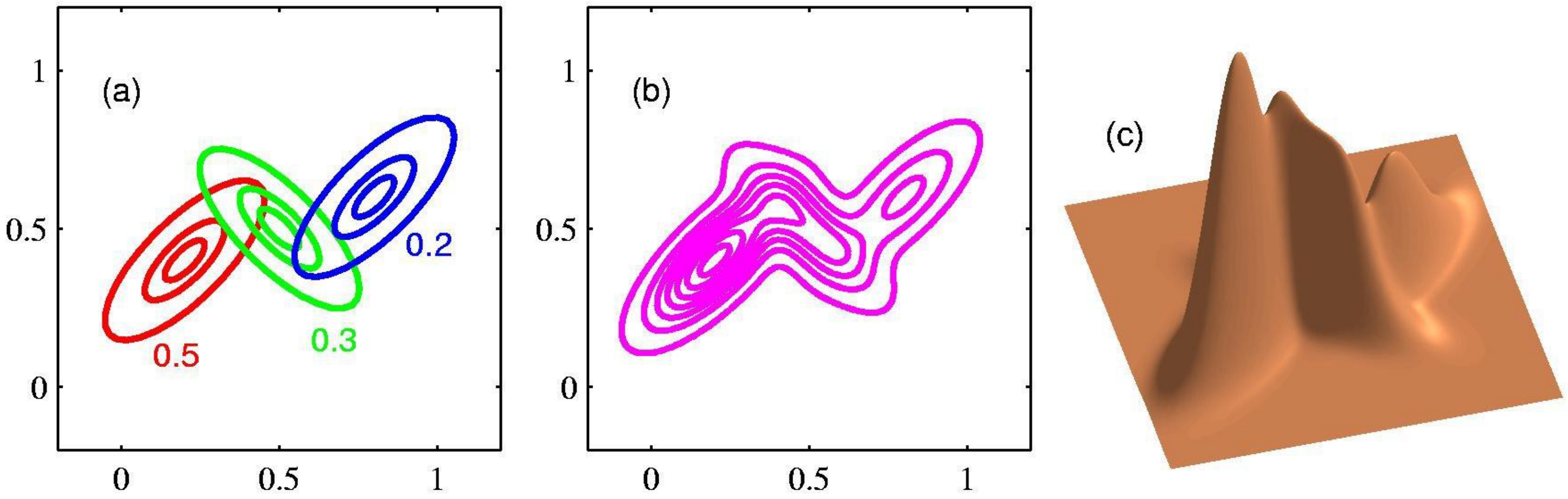
Mixing coefficient



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

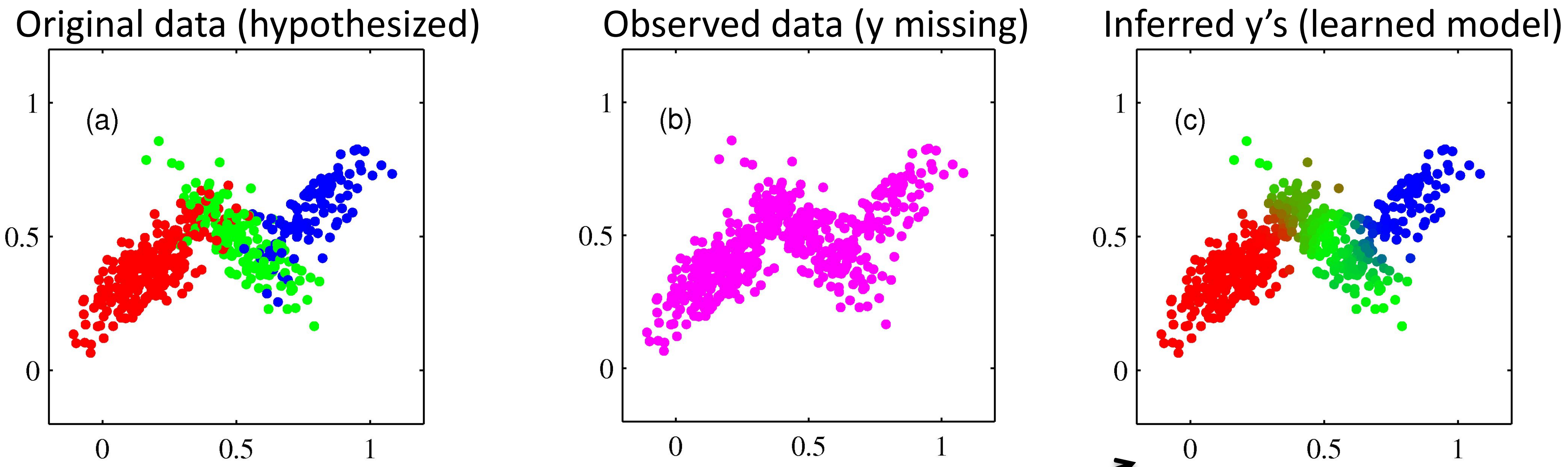
Slide adapted from David Sontag

Marginal distribution for mixtures of Gaussians



Slide adapted from David Sontag

Learning mixtures of Gaussians



Shown is the *posterior probability* that a point was generated from i^{th} Gaussian: $\Pr(Y = i \mid x)$

Slide adapted from David Sontag

ML estimation in supervised setting

- Univariate Gaussian

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

- Mixture of Multivariate Gaussians
  - ML estimate for each of the Multivariate Gaussians is given by

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

Just sums over x generated from the k’th Gaussian

Slide adapted from David Sontag

What about with unobserved data?

- Maximize ***marginal likelihood***:

$$\underset{\theta}{\operatorname{argmax}} \prod_j P(x_j) = \underset{\theta}{\operatorname{argmax}} \prod_j \sum_{k=1}^{K} P(Y_j = k, x_j)$$

Almost always a hard problem! – Usually no closed form solution

Slide adapted from David Sontag

Expectation Maximization

The EM Algorithm

- A clever method for maximizing ***marginal likelihood***:

$$\underset{\theta}{\operatorname{argmax}} \prod_j P(x_j) = \underset{\theta}{\operatorname{argmax}} \prod_j \sum_{k=1}^{K} P(Y_j = k, x_j)$$

- Based on coordinate descent.
- Alternate between two steps:
  - Compute an expectation
  - Compute a maximization

The EM Algorithm: Two steps

- Objective: $\underset{\theta}{\operatorname{argmax}} \prod_j P(x_j) = \underset{\theta}{\operatorname{argmax}} \prod_j \sum_{k=1}^{K} P(Y_j = k, x_j)$
- Data: $\{x_j\}$
  - **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: $P(Y_j = k | x_j; \theta)$
  - **M-step**: Re-estimate the parameters with "weighted" MLE estimates.

$$\theta^{new} = \underset{\theta}{\operatorname{argmax}} \sum_j \sum_k P(Y_j = k | x_j;\ \theta) \log P(Y_j = k, x_j; \theta)$$

Weight each example according to model's confidence

Assignment!!

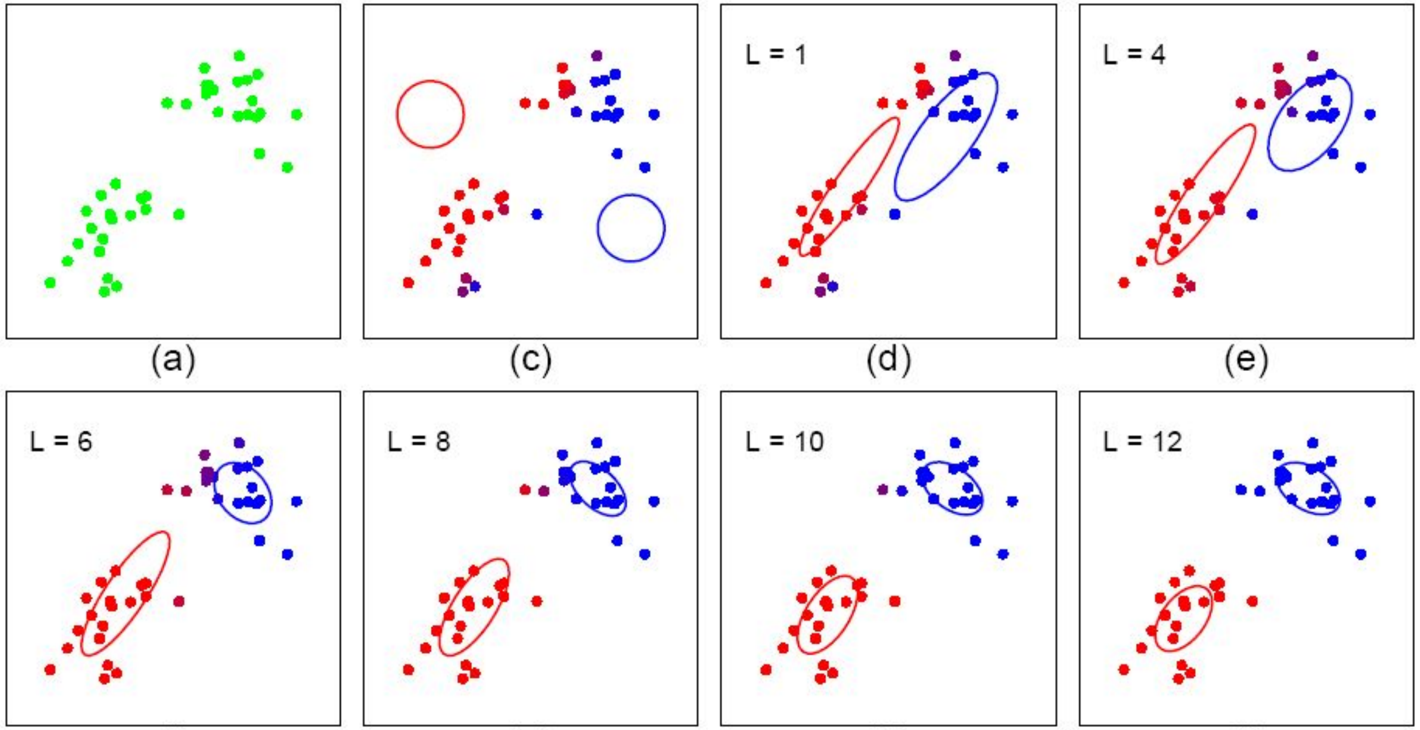
Set the parameters to the values that maximizes likelihood

Recalculate "centers"/MLEs!!

Slide adapted from David Sontag

(Soft) EM for GMM

- Start:
  - "Guess" the centroid μ_k and coveriance Σ_k of each of the K clusters
- Loop:



(a) (c) (d) (e)

(f) (g) (h) (i)

EM for GMMs: only learning μ (single feature)

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

$$\theta^{(t)} = \left\{\mu_1^{(t)}, \dots, \mu_K^{(t)}\right\}$$

- **E-Step**: Compute "expected" classes of all datapoints

$$P\left(Y_j = k \middle| x_j, \theta\right) \propto \exp\left(-\frac{1}{2\sigma^2}\left(x_j - \mu_k\right)^2\right) P(Y_j = k)$$

- **M-step:** Compute most likely new μs given class

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

Slide adapted from David Sontag

Hard Assignment = k-means

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

$$\theta^{(t)} = \left\{\mu_1^{(t)}, \dots, \mu_K^{(t)}\right\}$$

- **E-Step**: Compute "expected" classes of all datapoints

$$P\left(Y_j = k \middle| x_j, \theta\right) \propto \exp\left(-\frac{1}{2\sigma^2}\left(x_j - \mu_k\right)^2\right) \cancel{P\left(Y_j = k\right)}$$

- **M-step:** Compute most likely new μs given class

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

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

Equivalent to k-means clustering algorithm!!!

Slide adapted from David Sontag

EM for general GMMs

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

$$\theta^{(t)} = \left\{\mu_1^{(t)}, \ldots, \mu_K^{(t)}, \Sigma_1^{(t)}, \ldots, \Sigma_K^{(t)}, \pi_1^{(t)}, \ldots, \pi_K^{(t)}\right\}$$

- **E-Step**: Compute "expected" classes of all datapoints

$$c_k^{(j)} = P\left(Y_j = k \middle| x_j; \theta\right) \propto \pi_k^{(t)} p\left(x_j; \mu_k^{(t)}, \Sigma_k^{(t)}\right)$$

- **M-step:** Compute weighted MLE for μs given expected classes above

$$\pi_k^{(t+1)} = \frac{\sum_{j=1}^N c_k^{(j)}}{N} \qquad \mu_k^{(t+1)} = \frac{\sum_{j=1}^N c_k^{(j)} x_j}{\sum_{j=1}^N c_k^{(j)}}$$

$$\Sigma_k^{(t+1)} = \frac{\sum_{j=1}^m c_k^{(j)} \left[x_j - \mu_k^{(t+1)}\right]\left[x_j - \mu_k^{(t+1)}\right]^T}{\sum_{j=1}^m c_k^{(j)}}$$

$\pi_k^{(t)}$ estimate of P(Y=k) in iteration t

Evaluate probability of a multivariate a Gaussian at x

Slide adapted from David Sontag

General EM Formulation

$$L(\boldsymbol{\theta}; \mathbf{X}) = p(\mathbf{X} \mid \boldsymbol{\theta}) = \int p(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta})\, d\mathbf{Z} = \int p(\mathbf{X} \mid \mathbf{Z}, \boldsymbol{\theta}) p(\mathbf{Z} \mid \boldsymbol{\theta})\, d\mathbf{Z}$$

However, this quantity is often intractable since $\mathbf{Z}$ is unobserved and the distribution of $\mathbf{Z}$ is unknown before attaining $\boldsymbol{\theta}$.

Z is unobserved, hidden variable.
X depends on Z.

The EM algorithm [edit]

The EM algorithm seeks to find the maximum likelihood estimate of the marginal likelihood by iteratively applying these two steps:

Expectation step (E step): Define $Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)})$ as the expected value of the log likelihood function of $\boldsymbol{\theta}$, with respect to the current conditional distribution of $\mathbf{Z}$ given $\mathbf{X}$ and the current estimates of the parameters $\boldsymbol{\theta}^{(t)}$:

$$Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)}) = \mathrm{E}_{\mathbf{Z} \sim p(\cdot \mid \mathbf{X}, \boldsymbol{\theta}^{(t)})}[\log p(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta})]$$

Using a current value of $\theta^{(t)}$ estimate the Form of likelihood function. θ governs The conditional $P(X|Z,\theta)$ and prior $P(Z|\theta)$

Maximization step (M step): Find the parameters that maximize this quantity:

$$\boldsymbol{\theta}^{(t+1)} = \underset{\boldsymbol{\theta}}{\arg\max}\, Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)})$$

Now maximize the likelihood function
Find new $\theta^{(t+1)}$.

More succinctly, we can write it as one equation:

$$\boldsymbol{\theta}^{(t+1)} = \underset{\boldsymbol{\theta}}{\arg\max}\, \mathrm{E}_{\mathbf{Z} \sim p(\cdot \mid \mathbf{X}, \boldsymbol{\theta}^{(t)})}[\log p(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta})]$$

The general learning problem with missing data

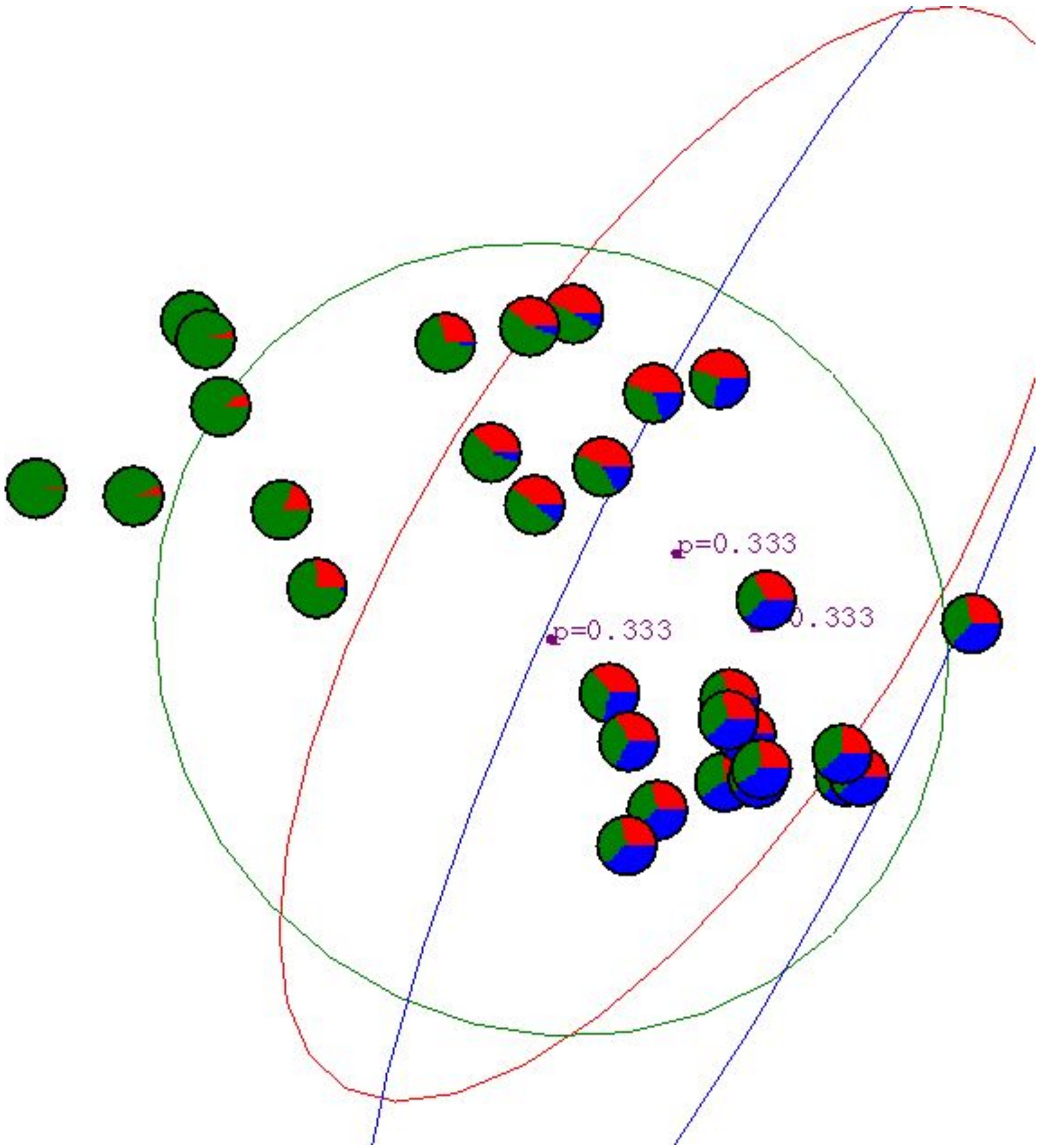
- Marginal likelihood: **X** is observed,
- **Z** (e.g. the class labels **Y**) is missing:

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

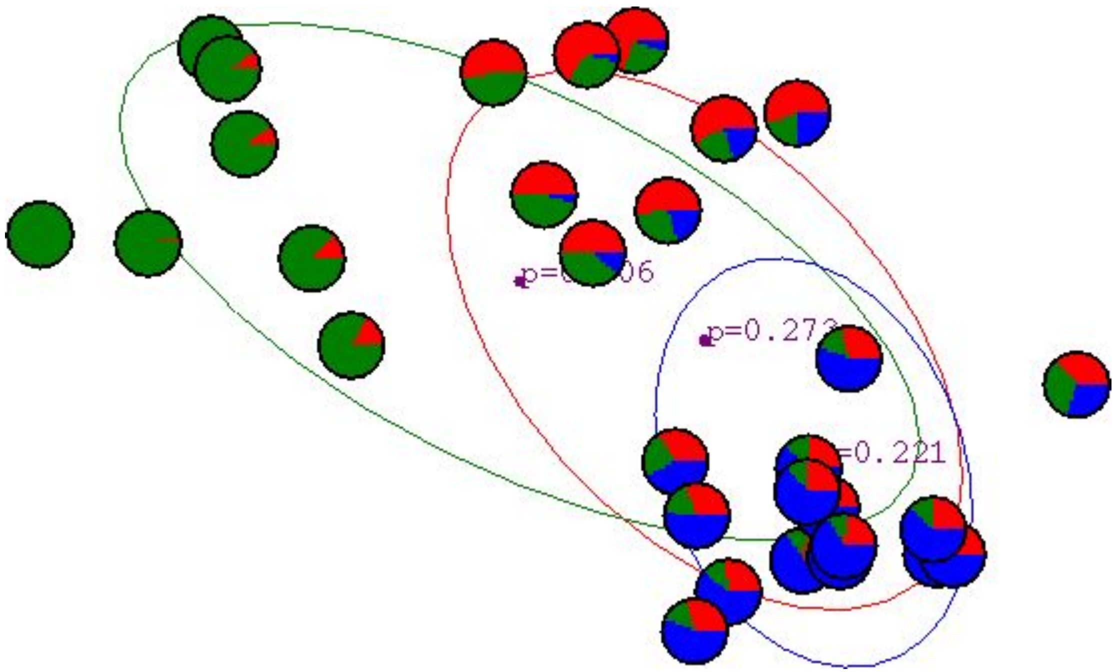
- Objective: Find $\text{argmax}_\theta\ l(\theta\text{:Data})$
- Assuming hidden variables are *missing completely at random* (otherwise, we should explicitly model *why* the values are missing)

Slide adapted from David Sontag

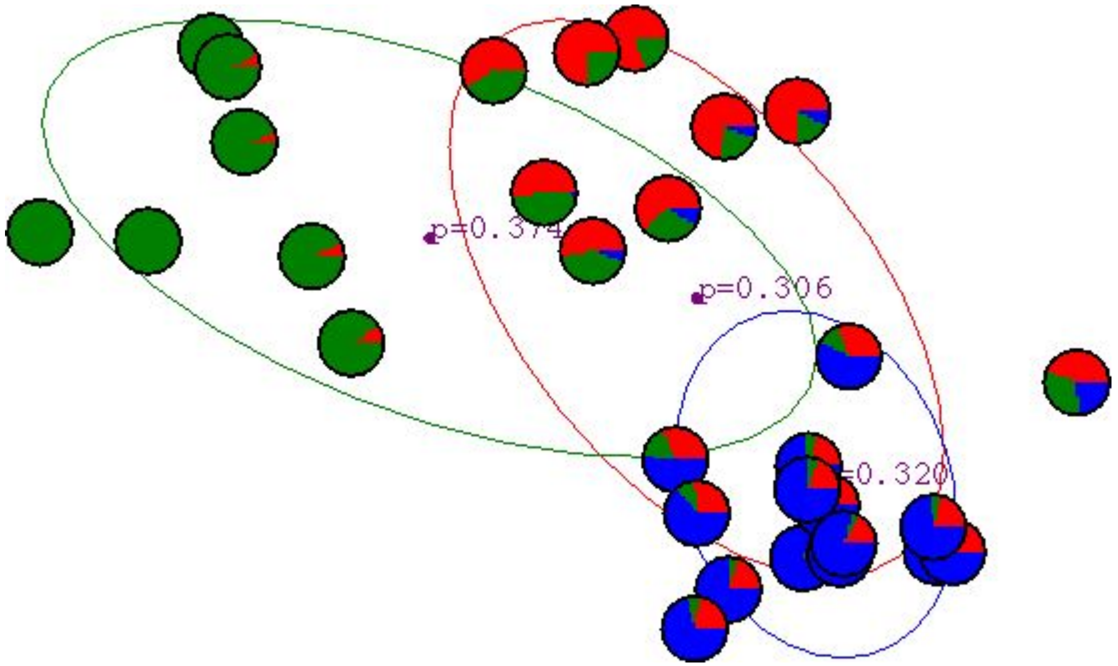
Gaussian Mixture Example: Start



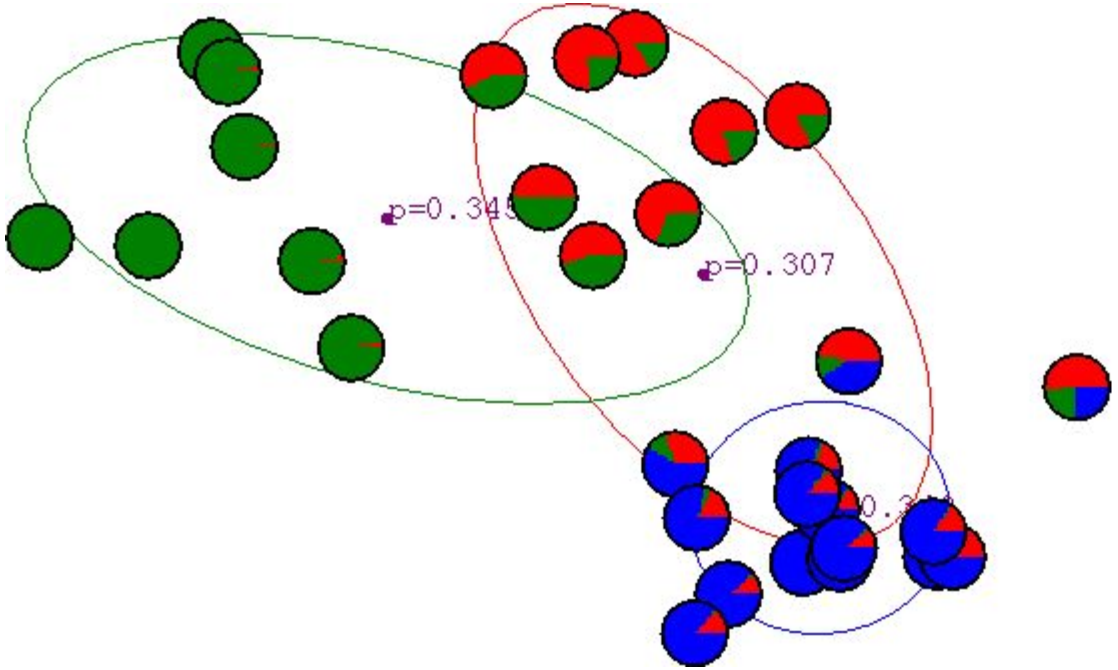
After first iteration



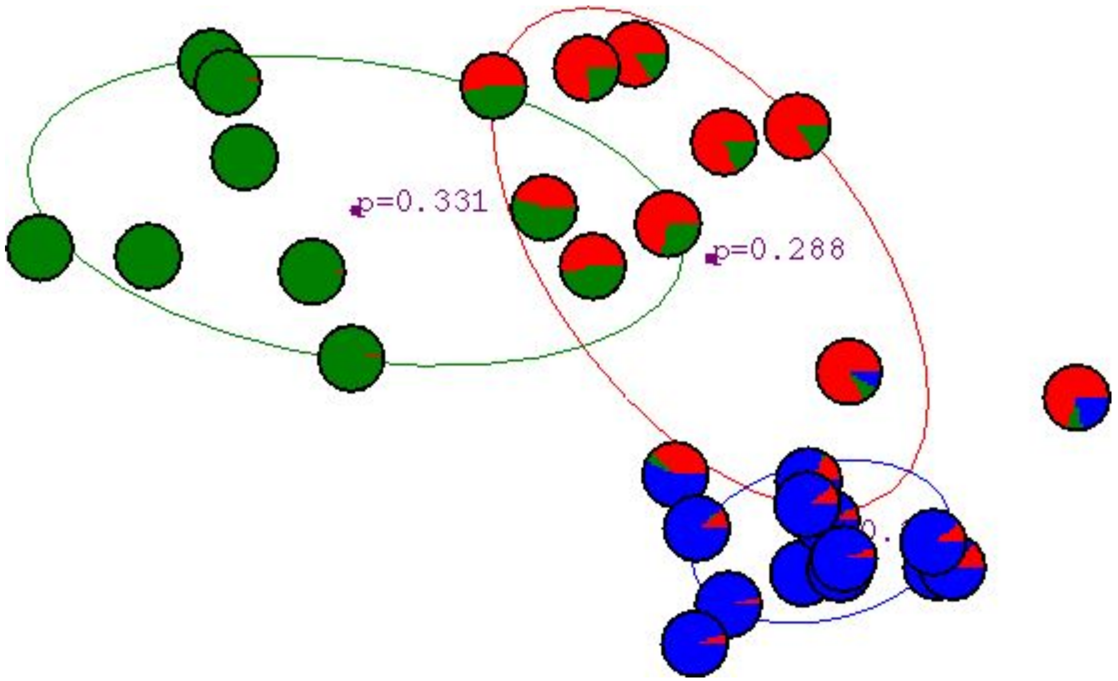
After 2nd iteration



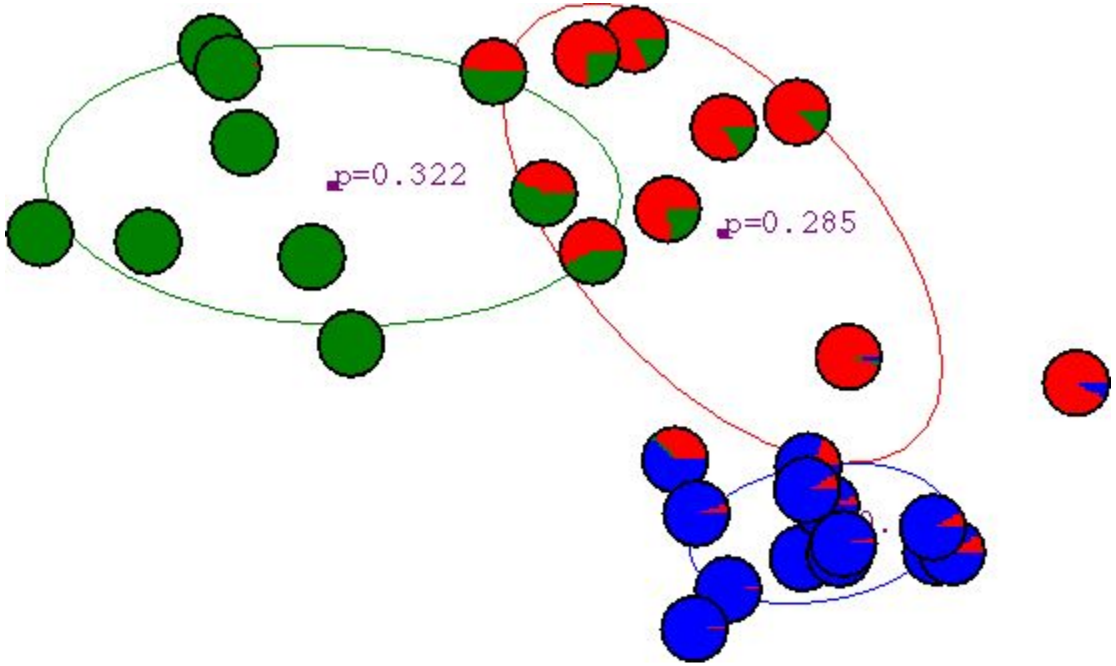
After 3rd iteration



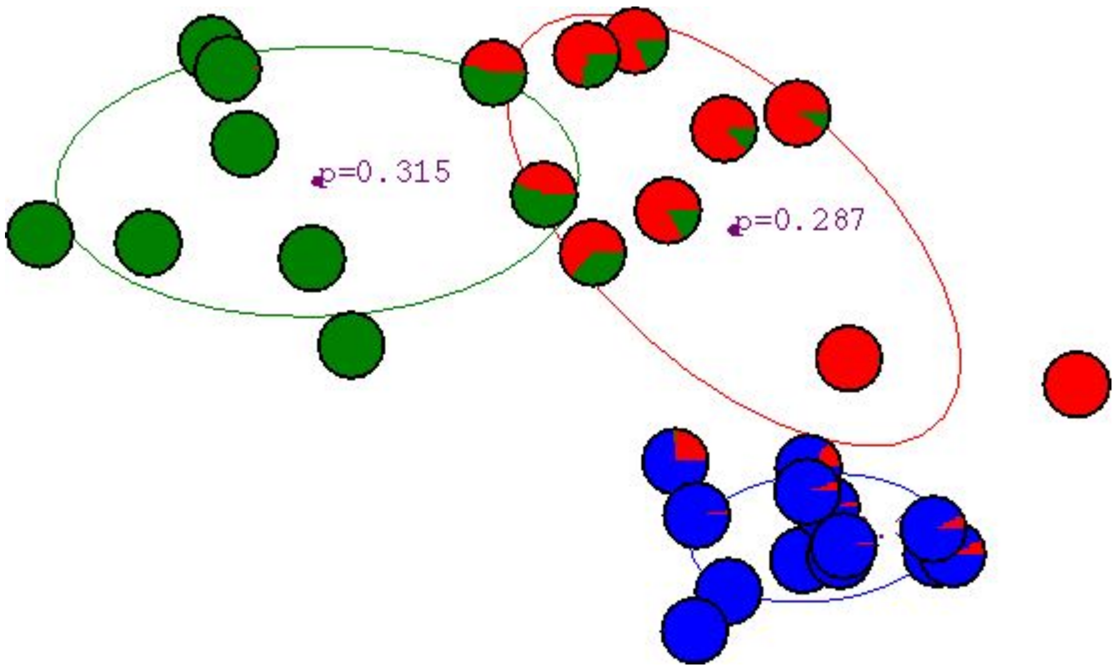
After 4th iteration



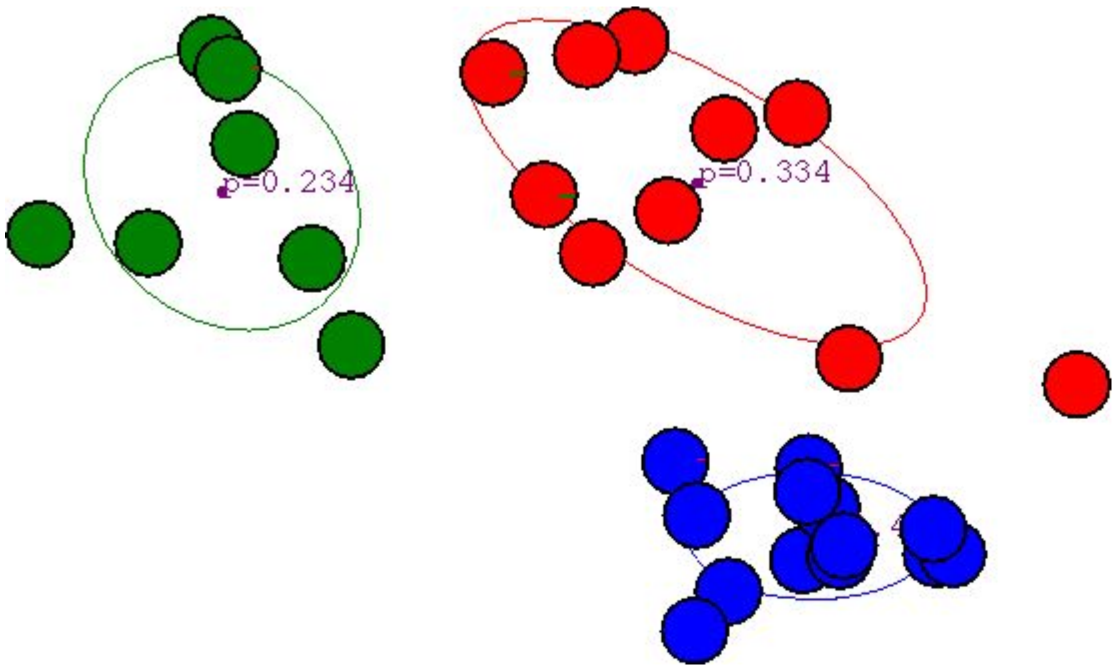
After 5th iteration



After 6th iteration



After 20th iteration



Properties of EM

- One can prove that:
  - EM converges to a local maxima
  - Each iteration improves the log-likelihood
- How?
  - Likelihood objective instead of k-means objective
  - M-step can never decrease likelihood

Slide adapted from David Sontag