

# CS60050

# Machine Learning

# GMM

Somak Aditya

Sudeshna Sarkar

Department of CSE, IIT Kharagpur

# Slide sources

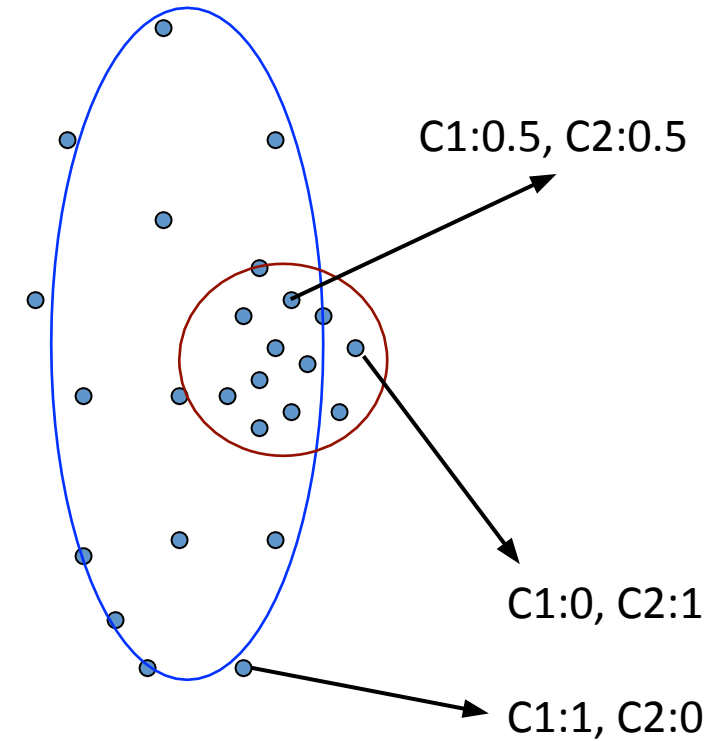
- David Sontag New York University
  - Slides adapted from Carlos Guestrin, Dan Klein, Luke Zettlemoyer, Dan Weld, Vibhav Gogate, and Andrew Moore
- Matt Gormley (CMU)
  - **10-601B Introduction to Machine Learning**

# 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

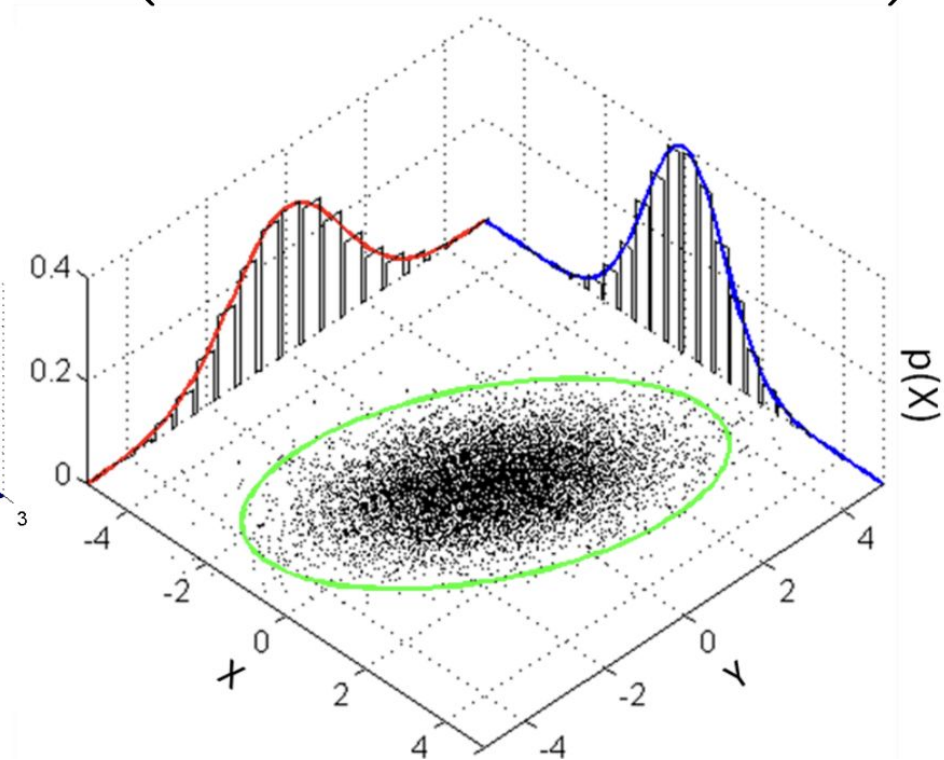
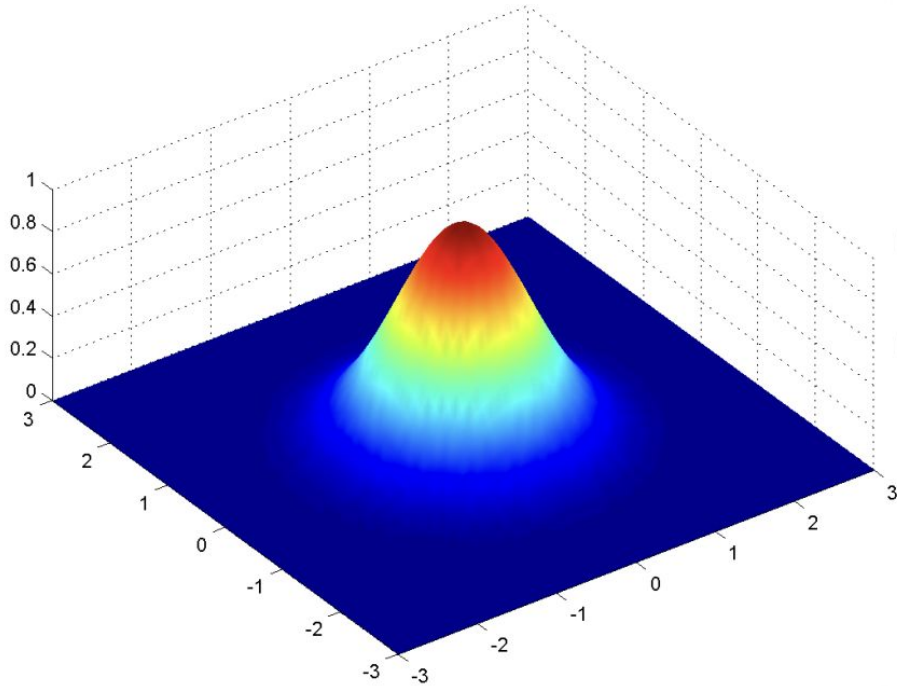
- Allow overlaps, clusters of different size, etc.
- Can tell a generative story for data
  - $P(Y)P(X|Y)$
- Challenge: we need to estimate model parameters without labeled  $Y$ s



# Recall Gaussian Distribution

- Recall the Gaussian distribution:

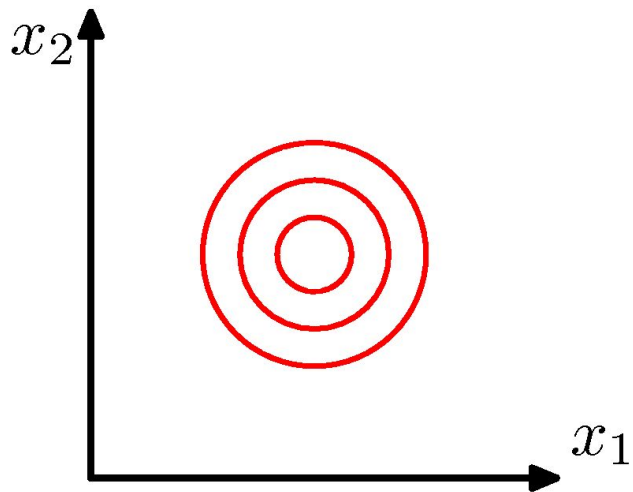
$$P(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} \exp \left( -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right)$$



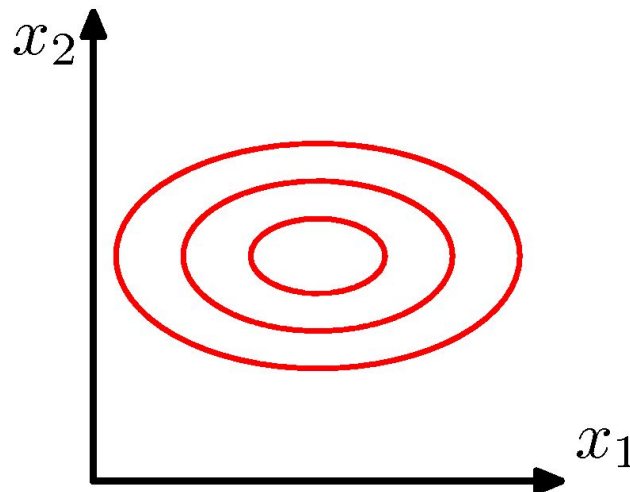
# Multivariate Gaussians

- 

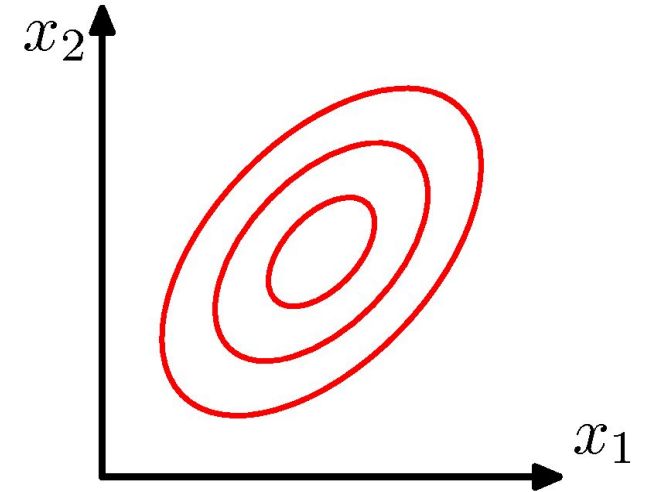
$$P(X = x_j | \mu, \Sigma) = \frac{1}{(2\pi)^{d/2} \|\Sigma\|^{\frac{1}{2}}} \exp \left[ -\frac{1}{2} (x_j - \mu)^T \Sigma^{-1} (x_j - \mu) \right]$$



$\Sigma \propto$  identity matrix



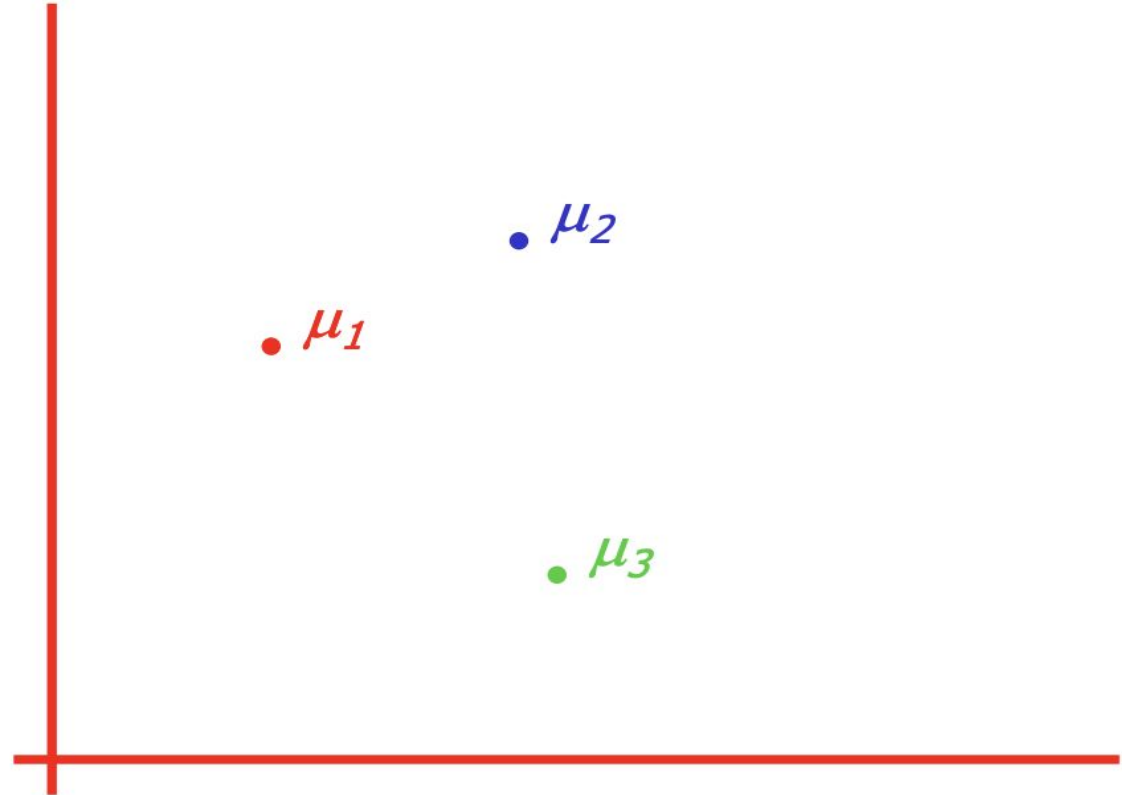
$\Sigma =$  diagonal matrix  
 $X_i$  are independent



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

# The GMM Assumption

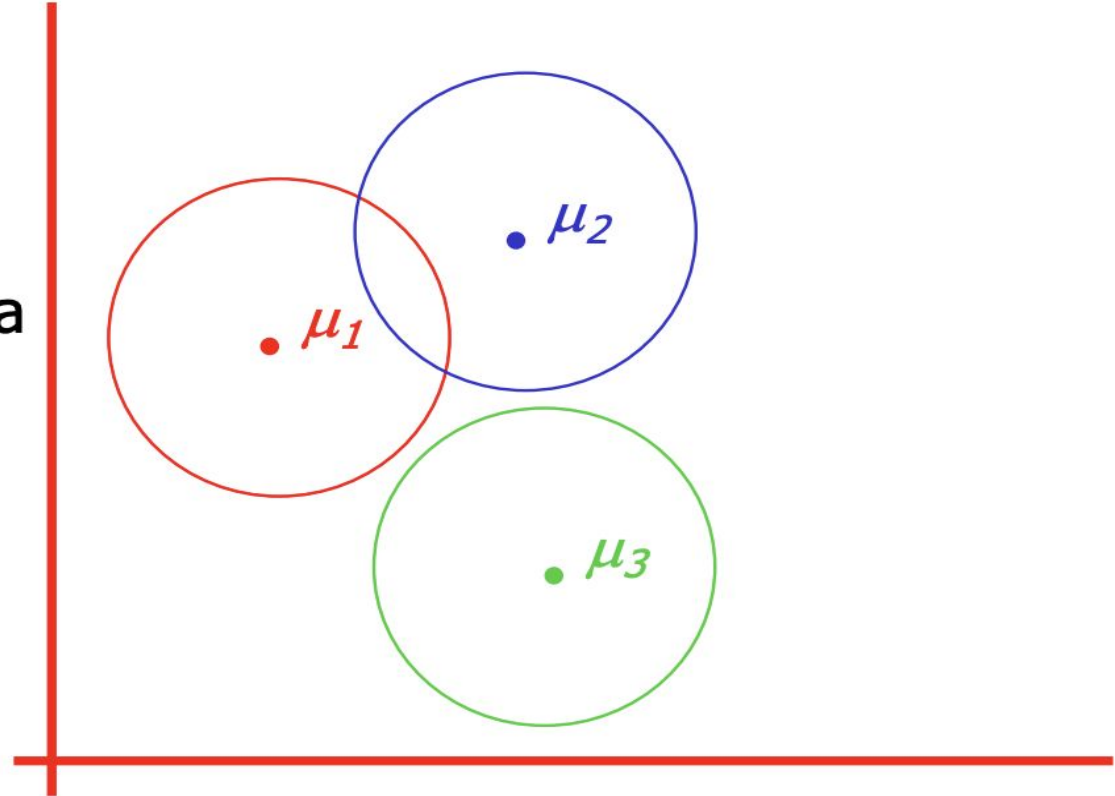
- There are  $k$  components. The  $i$ 'th component is called  $\omega_i$
- Component  $\omega_i$  has an associated mean vector  $\mu_i$



# The GMM Assumption

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

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

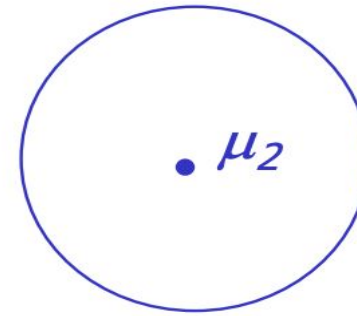


# The GMM Assumption

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

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

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



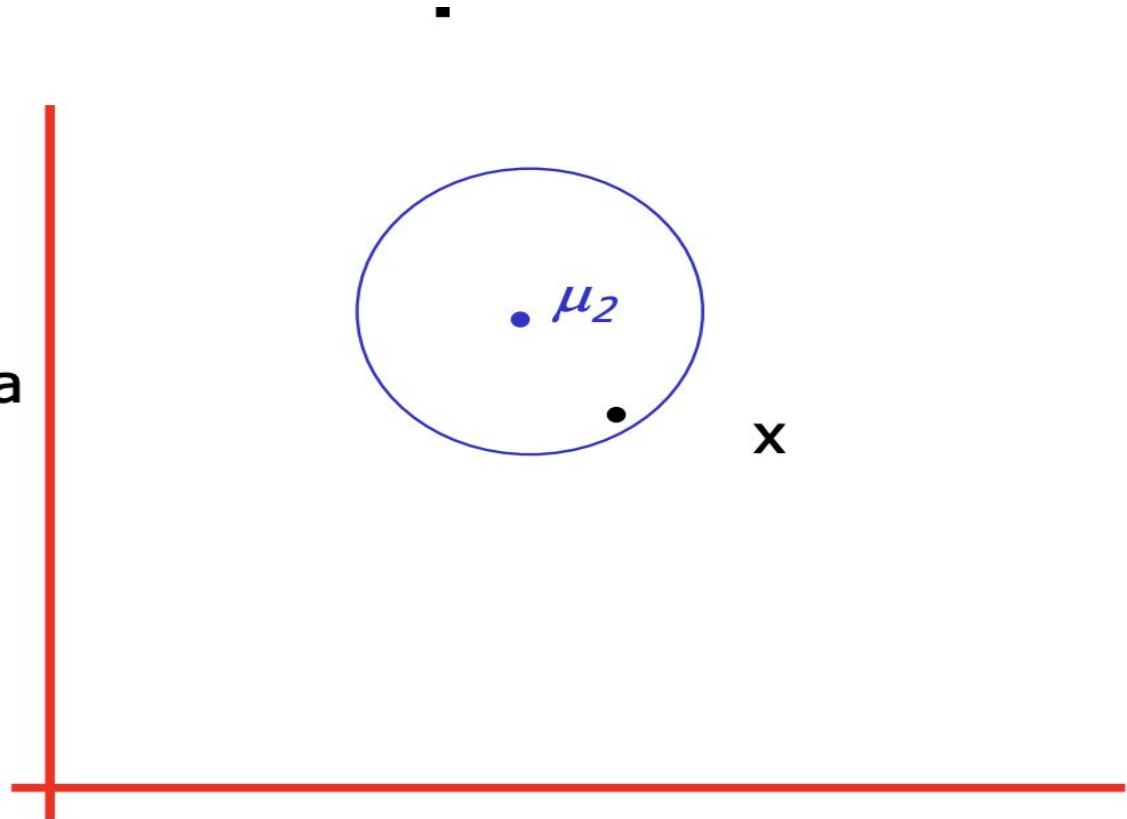


# The GMM Assumption

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

**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  $\sim N(\mu_i, \sigma^2 \mathbf{I})$

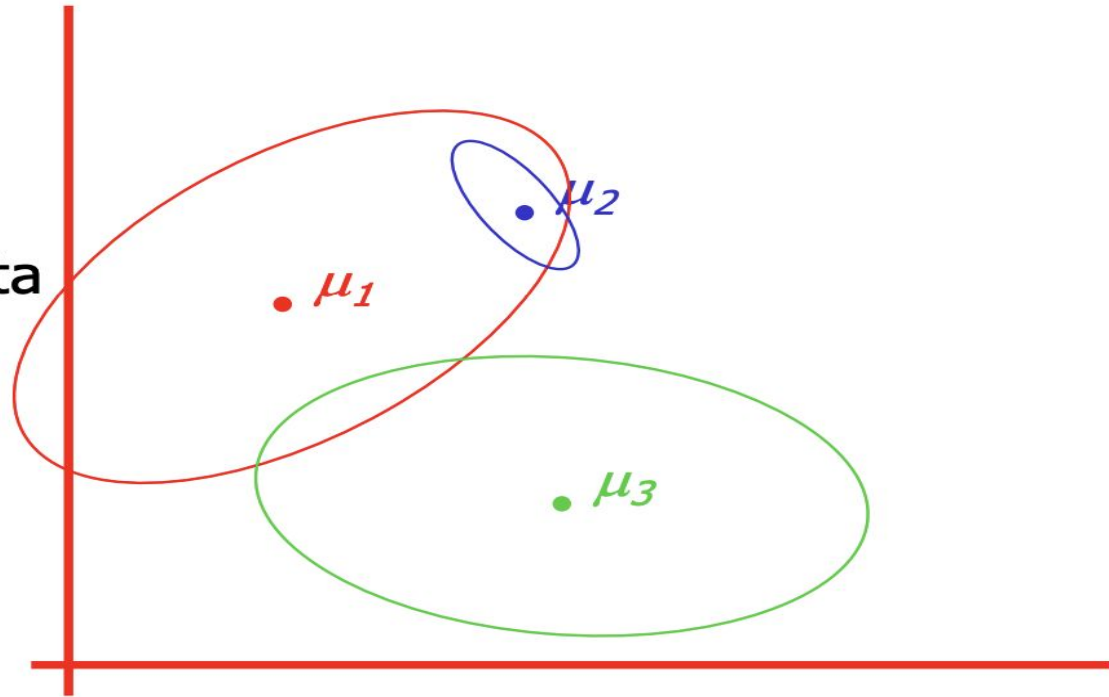


# The General GMM assumption

- There are  $k$  components. The  $i$ 'th component is called  $\omega_i$
- Component  $\omega_i$  has an associated mean vector  $\mu_i$
- Each component generates data from a Gaussian with mean  $\mu_i$  and covariance matrix  $\Sigma_i$

**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

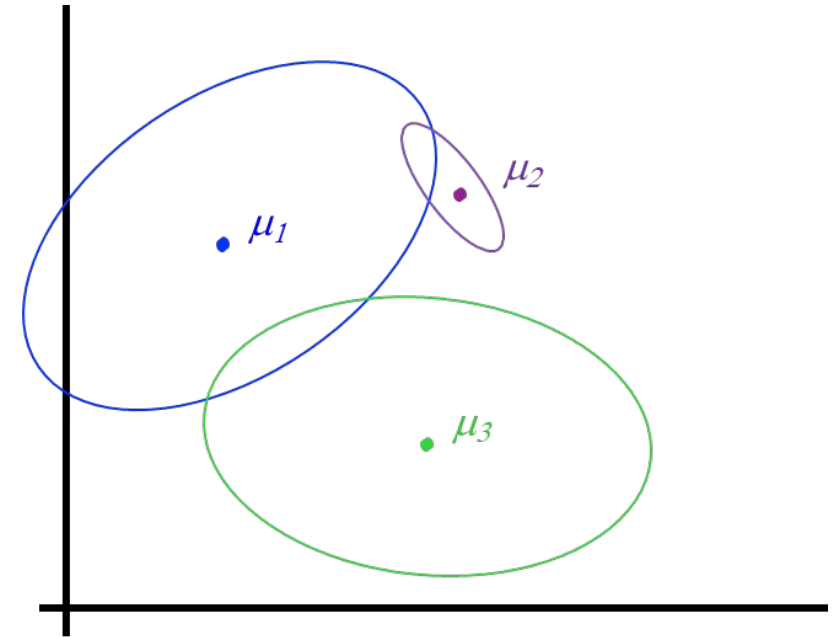
- $P(Y)$ : There are  $k$  components
- $P(X|Y)$ : Each component generates data from a **multivariate Gaussian** with mean  $\mu_i$  and covariance matrix  $\Sigma_i$

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  $\sim N(\mu_i, \Sigma_i)$

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

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



# Mixture Models

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

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

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

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

# Gaussian Mixture Models

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

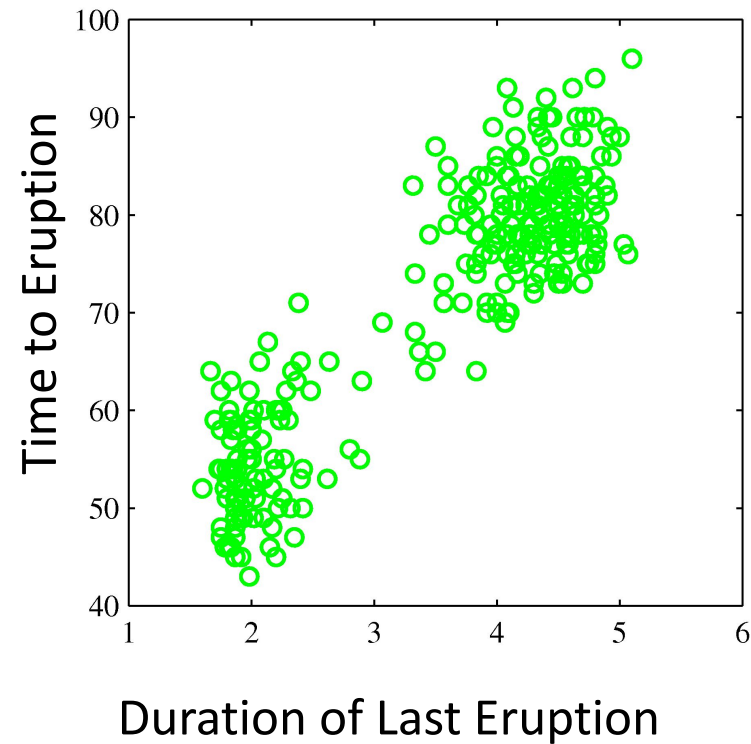
$$p(x) = \pi_0 N(x|\mu_0, \Sigma_0) + \pi_1 N(x|\mu_1, \Sigma_1) + \dots + \pi_k N(x|\mu_k, \Sigma_k)$$

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

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

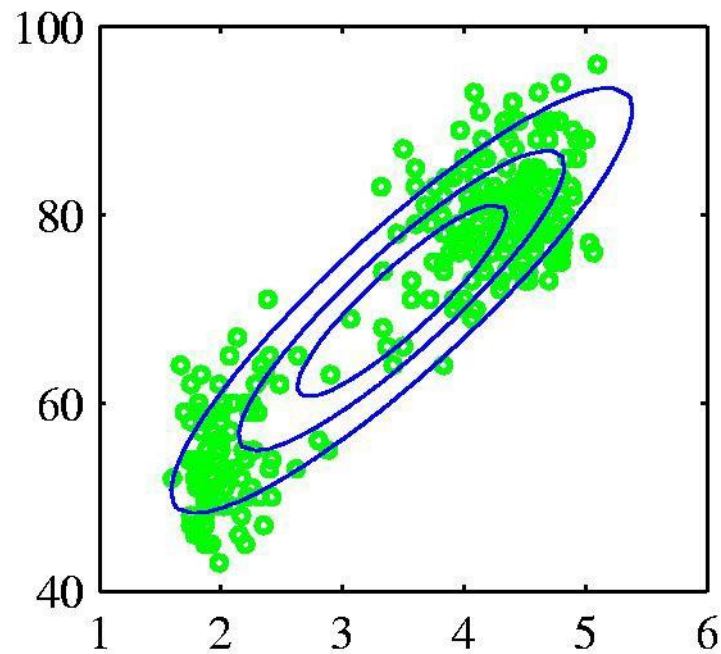
# Modelling eruption of geysers

## Old Faithful Data Set

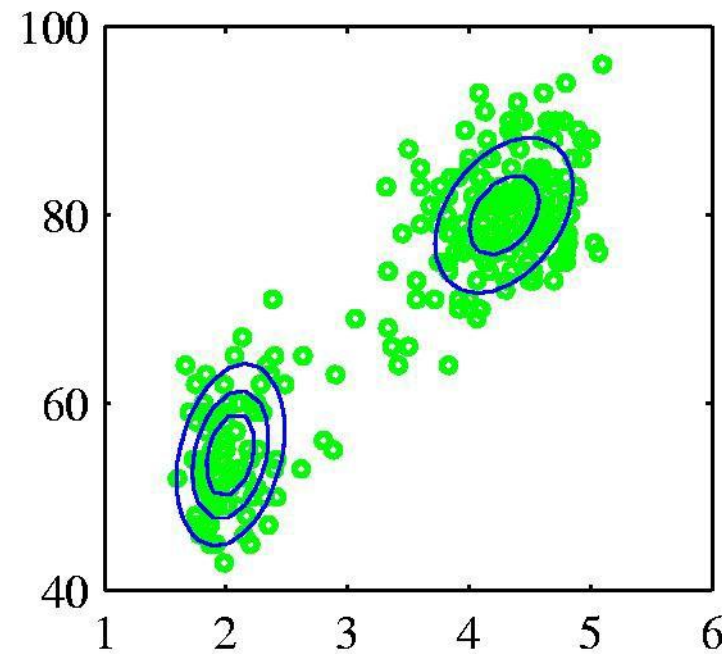


# Modelling eruption of geysers

## Old Faithful Data Set



Single Gaussian

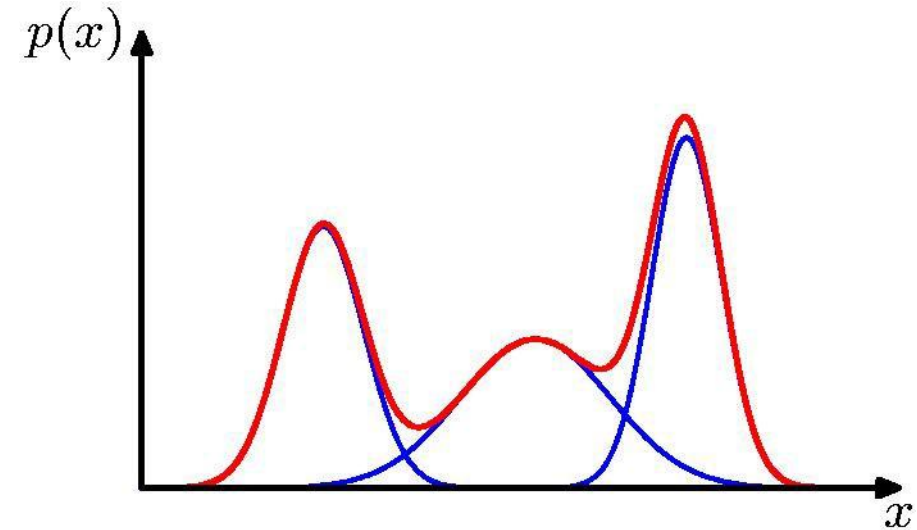


Mixture of two Gaussians

# Marginal distribution for mixtures of Gaussians

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

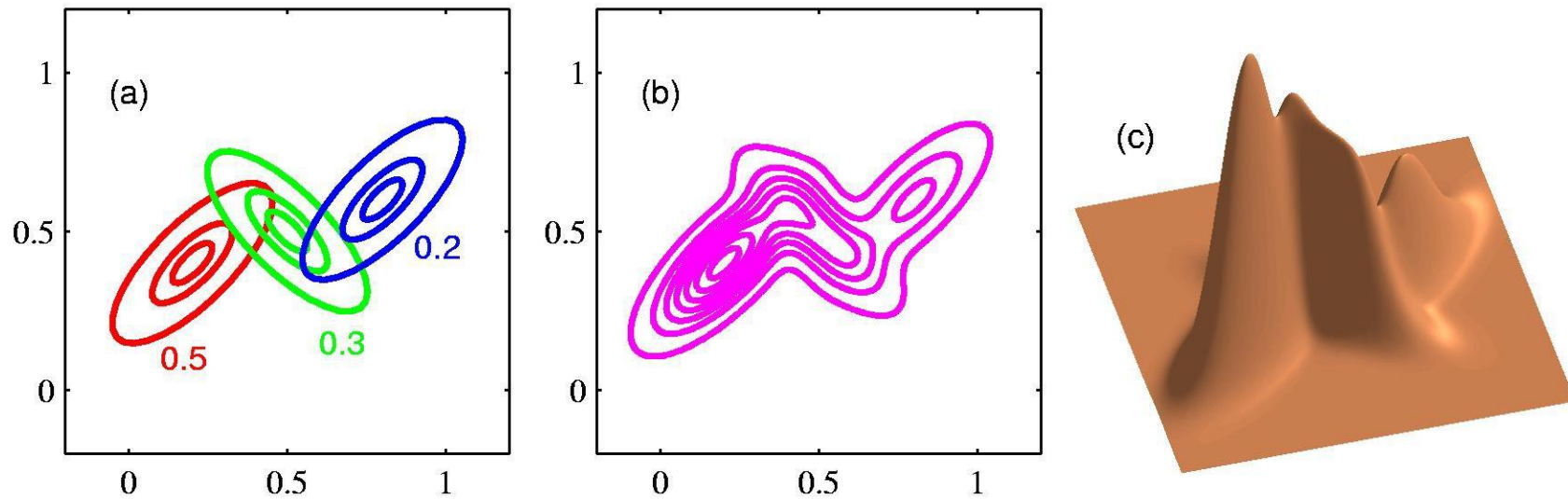
coefficient



$$\forall k : \pi_k \geq 0 \quad \sum_{k=1}^K \pi_k = 1$$

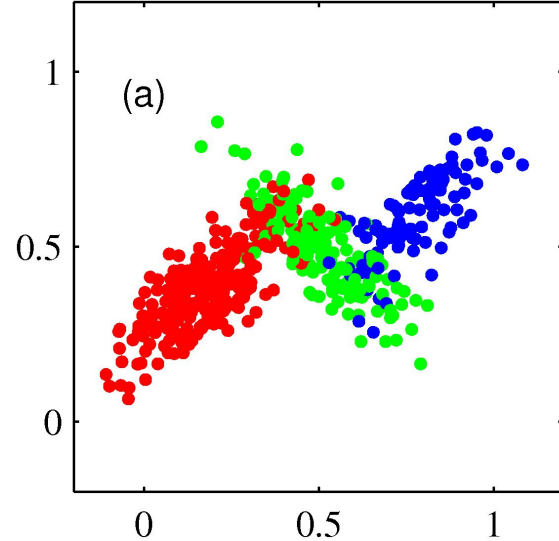


# Marginal distribution for mixtures of Gaussians

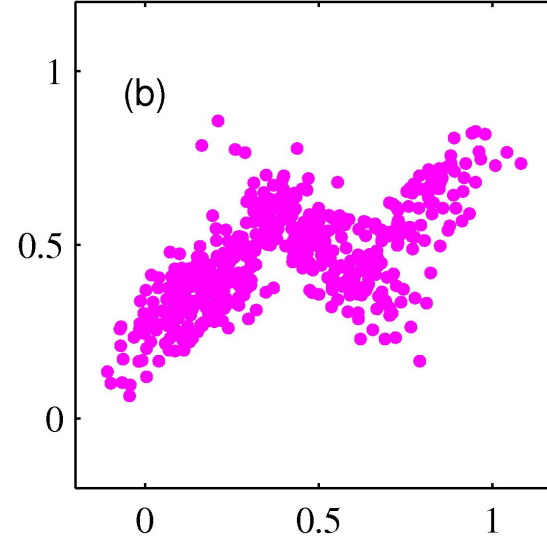


# Learning mixtures of Gaussians

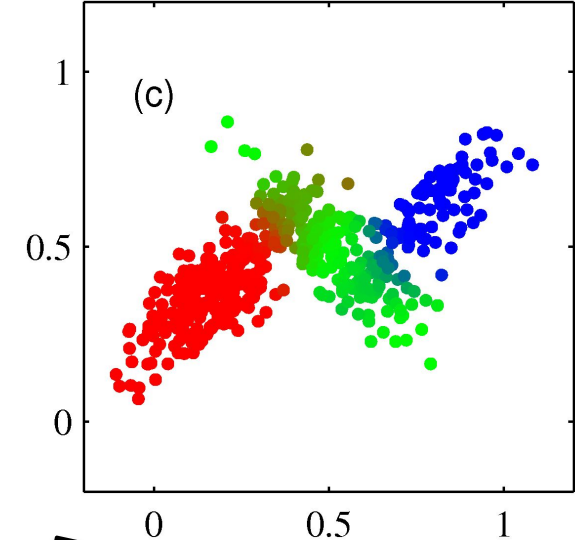
Original data (hypothesized)



Observed data (y missing)



Inferred y's (learned model)



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


# ML estimation in supervised setting

- Univariate Gaussian

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

- Mixture of Multivariate Gaussians

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

$$\mu_{MLE}^k = \frac{1}{n} \sum_{j=1}^n x_j^k \quad \Sigma_{MLE}^k = \frac{1}{n} \sum_{j=1}^n (x_j^k - \mu_{ML}^k)(x_j^k - \mu_{ML}^k)^T$$


Just sums over x generated from the k'th Gaussian

# What about with unobserved data?

- Maximize *marginal likelihood*:

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

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

# Expectation Maximization

# The EM Algorithm

- A clever method for maximizing *marginal likelihood*:

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

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

# The EM Algorithm: Two steps

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

- Data:  $\{x_j\}$

- **E-step:** Compute expectations to “fill in” missing  $y$  values according to current parameters,  $\theta$

- For all examples  $j$  and values  $k$  for  $Y_j$ , compute:  $P(Y_j = k | x_j; \theta)$

Weight each example according to model's confidence

Assignment!!

- **M-step:** Re-estimate the parameters with “weighted” MLE estimates.

$\theta^{new}$

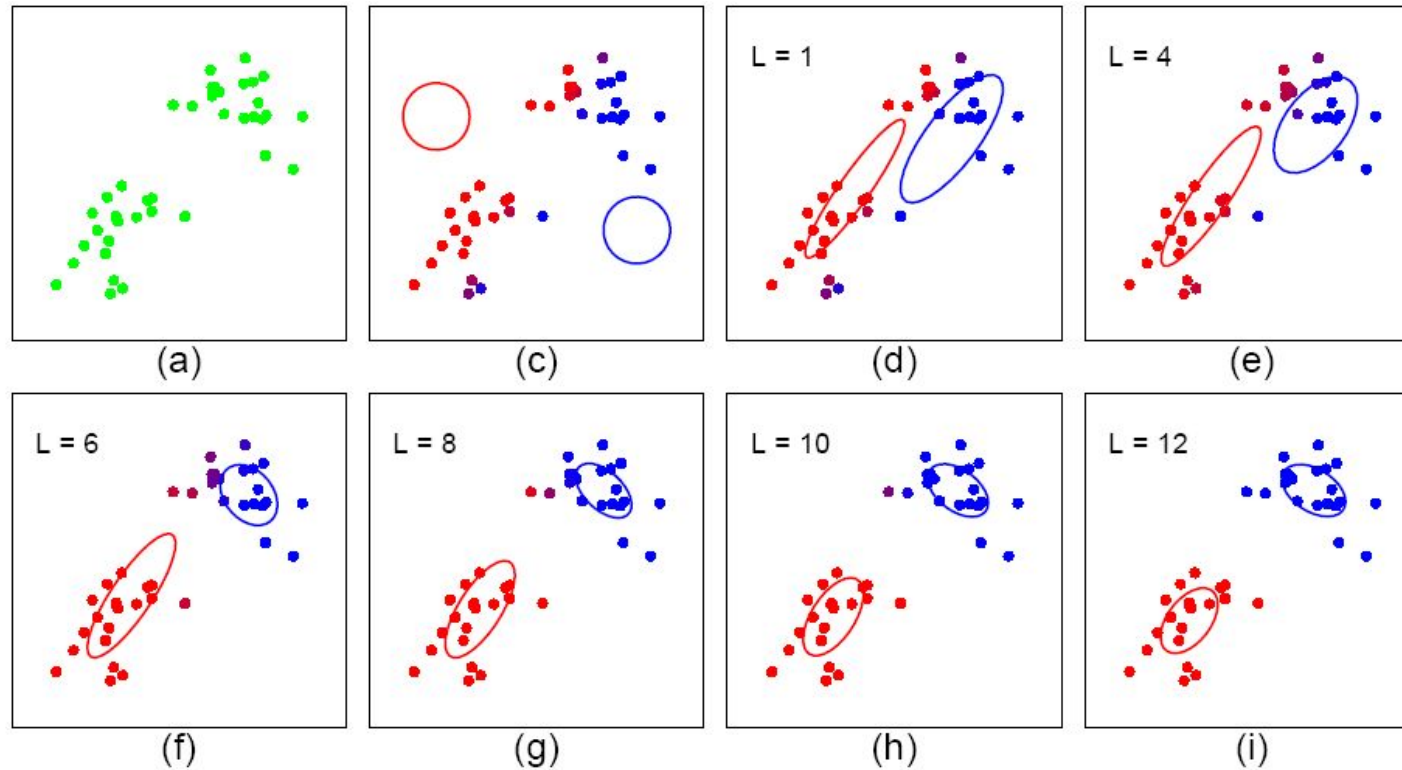
$$= \operatorname{argmax}_{\theta} \sum_j \sum_k P(Y_j = k | x_j; \theta) \log P(Y_j = k, x_j; \theta)$$

Set the **parameters** to the values that maximizes likelihood

Recalculate “centers”/MLEs!!

# (Soft) EM for GMM

- Start:
  - "Guess" the centroid  $\mu_k$  and covariance  $\Sigma_k$  of each of the K clusters
- Loop:





# EM for GMMs: only learning $\mu$ (single feature)

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

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

- **E-Step:** Compute “expected” classes of all datapoints

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

- **M-step:** Compute most likely new  $\mu$ s given class

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

# Hard Assignment = k-means

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

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

- **E-Step:** Compute “expected” classes of all datapoints

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

- **M-step:** Compute most likely new  $\mu$ s given class

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

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

$\delta$  represents hard assignment to “most likely” or nearest cluster

Equivalent to k-means clustering algorithm!!!

# EM for general GMMs

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

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

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

- **E-Step:** Compute “expected” classes of all datapoints

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

Evaluate probability  
of a multivariate a  
Gaussian at  $x$

- **M-step:** Compute weighted MLE for  $\mu$ s given expected classes above

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

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

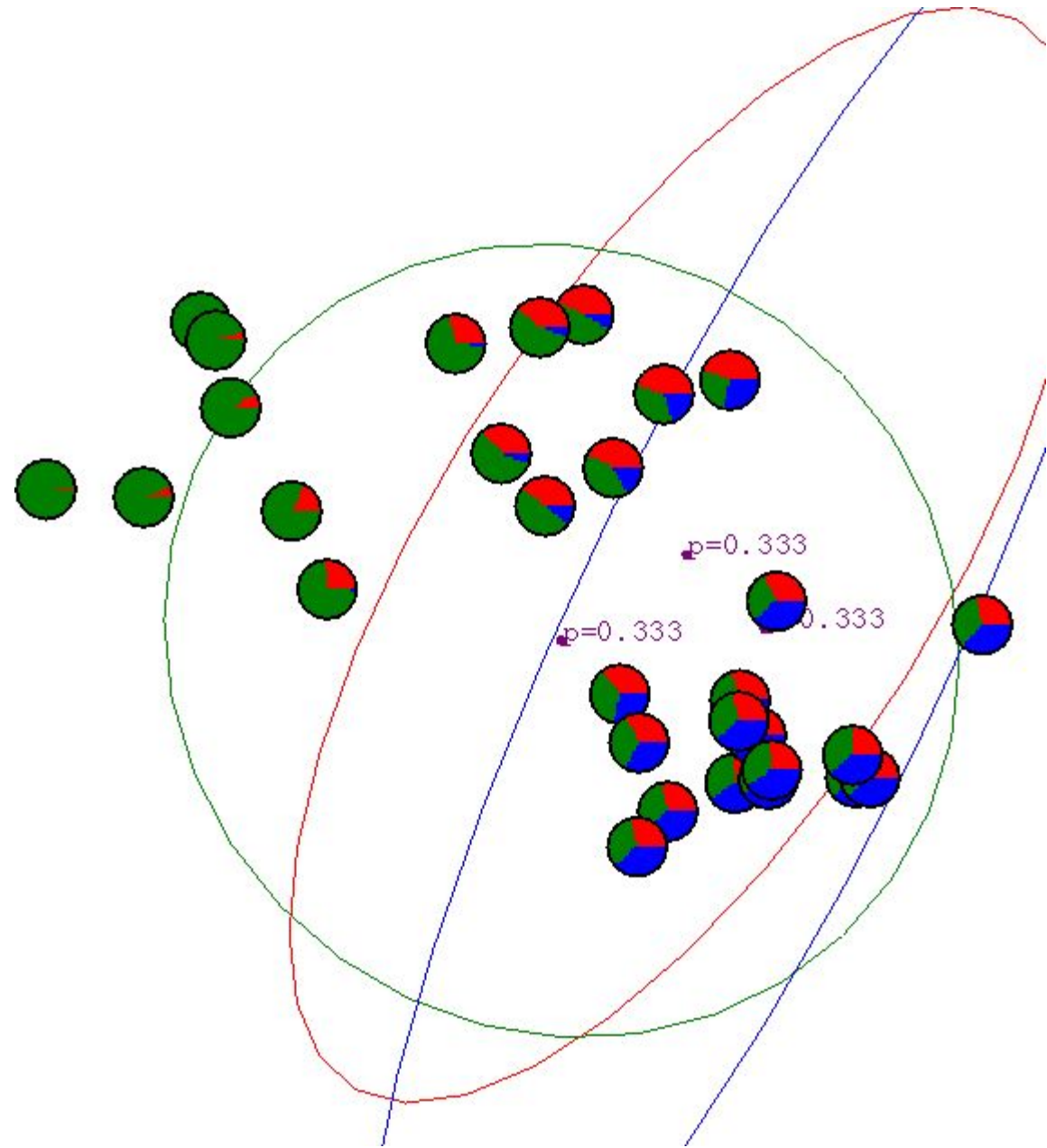
# The general learning problem with missing data

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

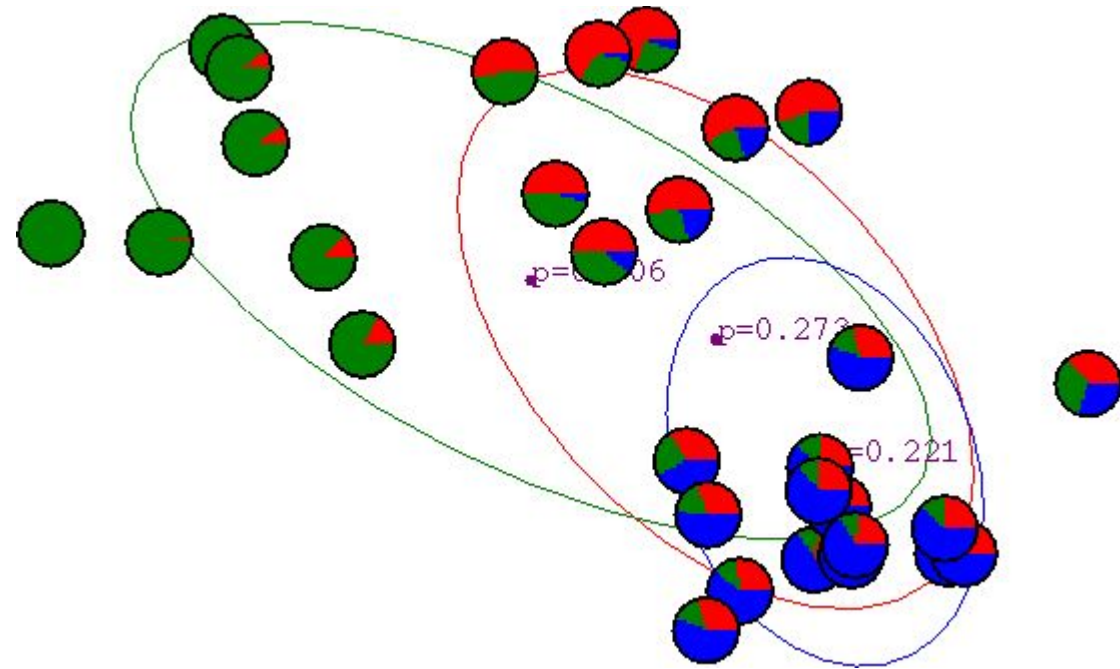
$$\begin{aligned}\ell(\theta : \mathcal{D}) &= \log \prod_{j=1}^m P(\mathbf{x}_j \mid \theta) \\ &= \sum_{j=1}^m \log P(\mathbf{x}_j \mid \theta) \\ &= \sum_{j=1}^m \log \sum_{\mathbf{z}} P(\mathbf{x}_j, \mathbf{z} \mid \theta)\end{aligned}$$

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

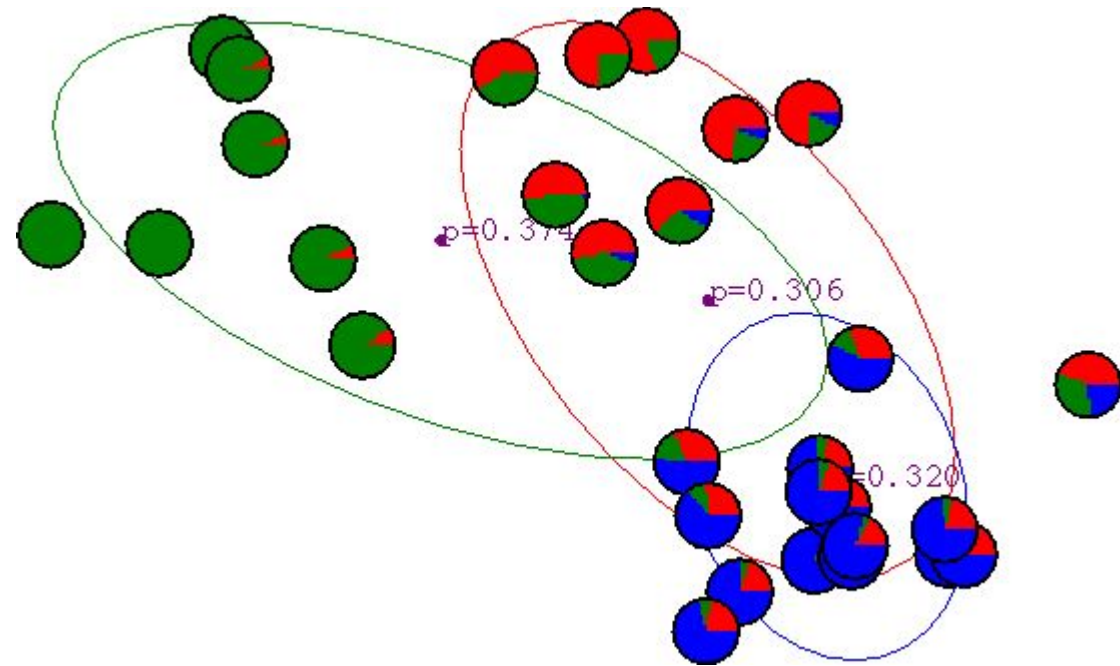
# Gaussian Mixture Example: Start



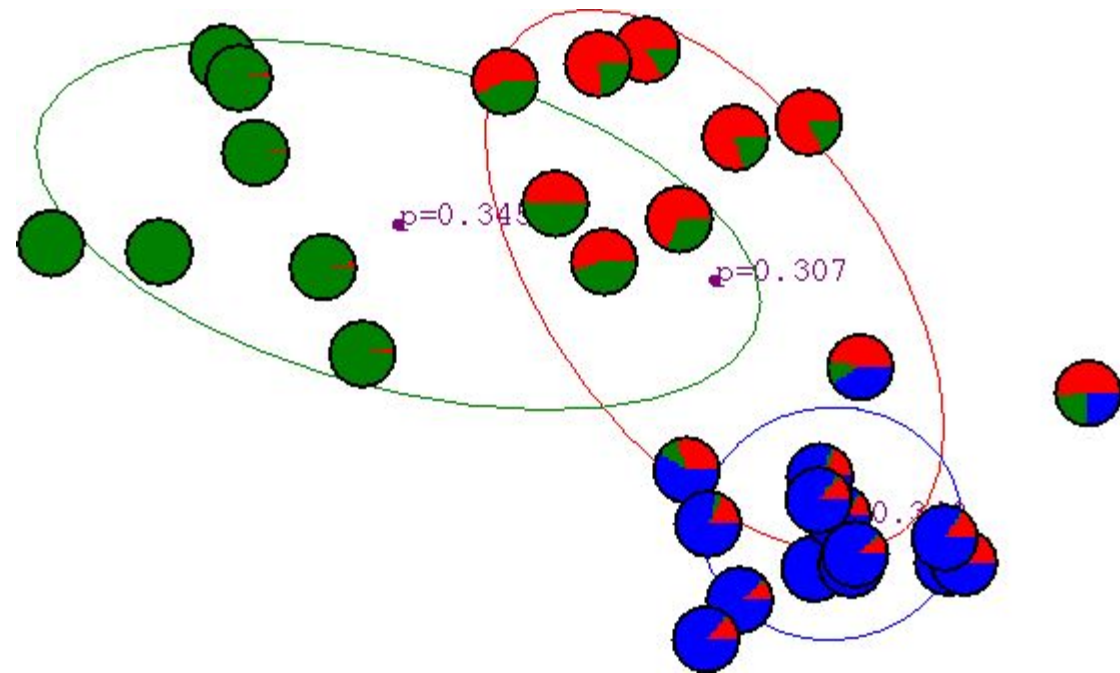
# After first iteration



# After 2nd iteration

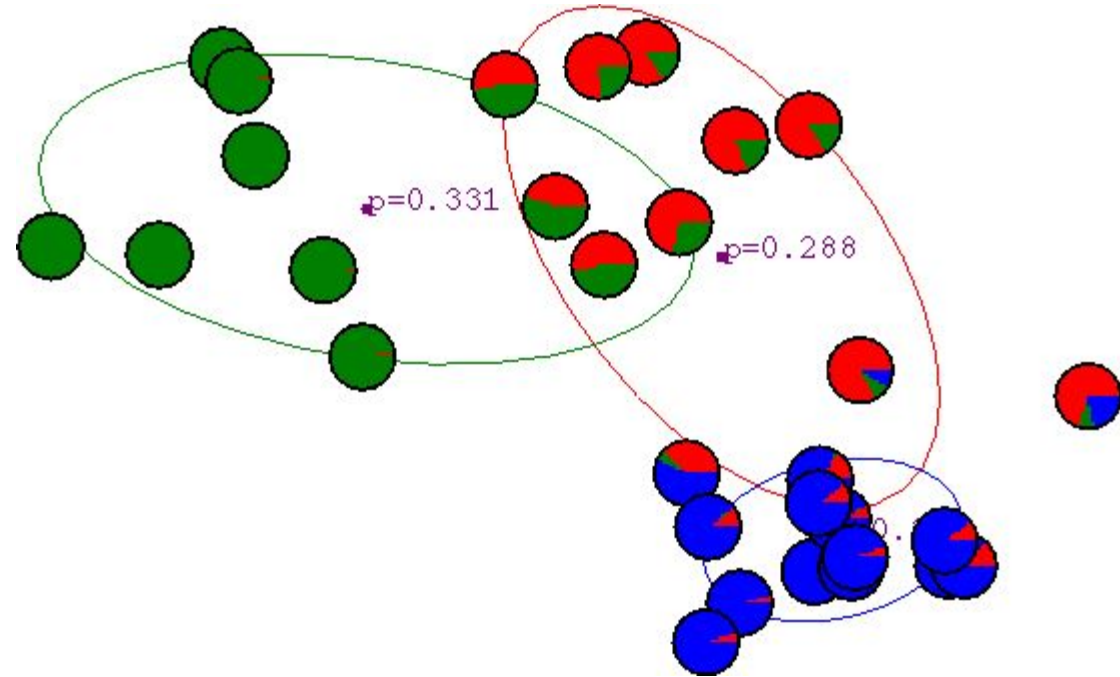


# After 3rd iteration

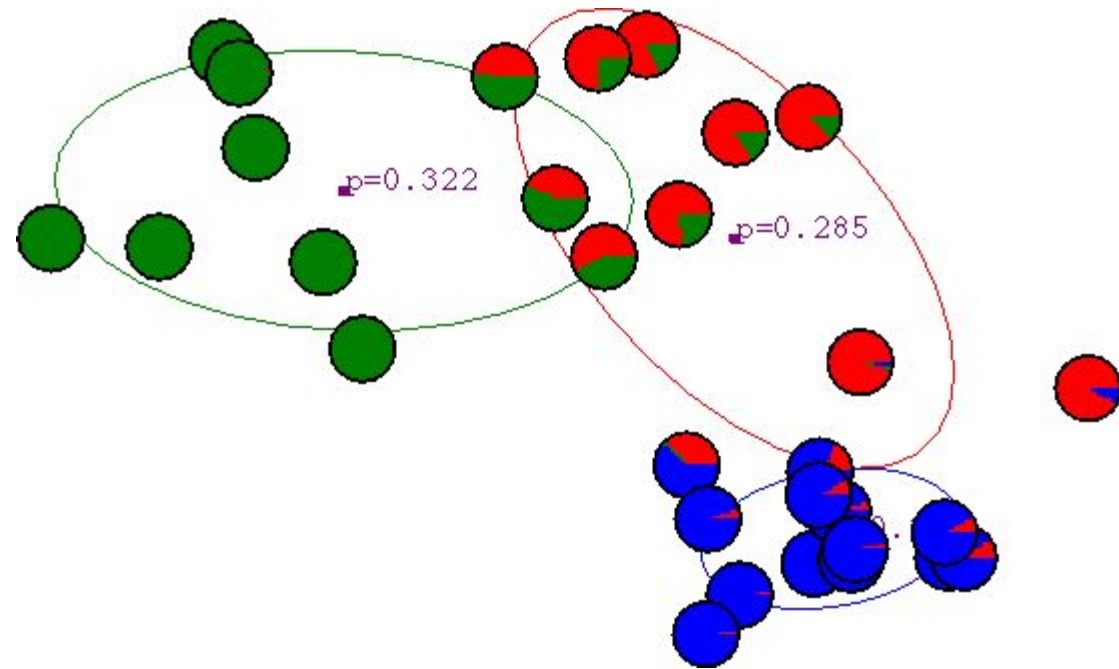




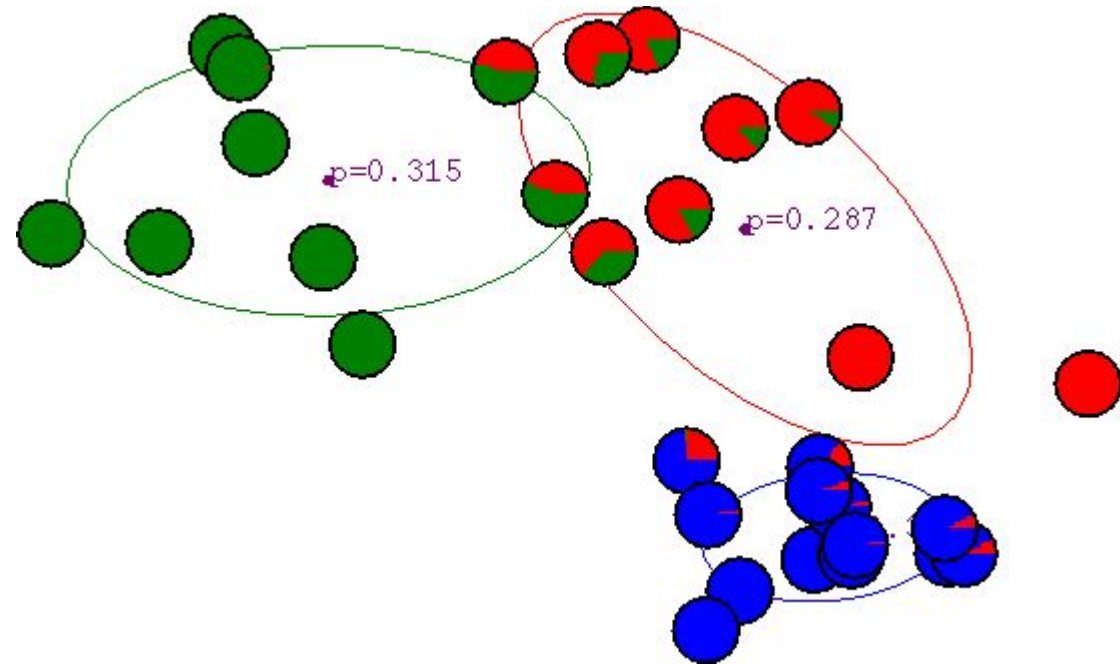
# After 4th iteration



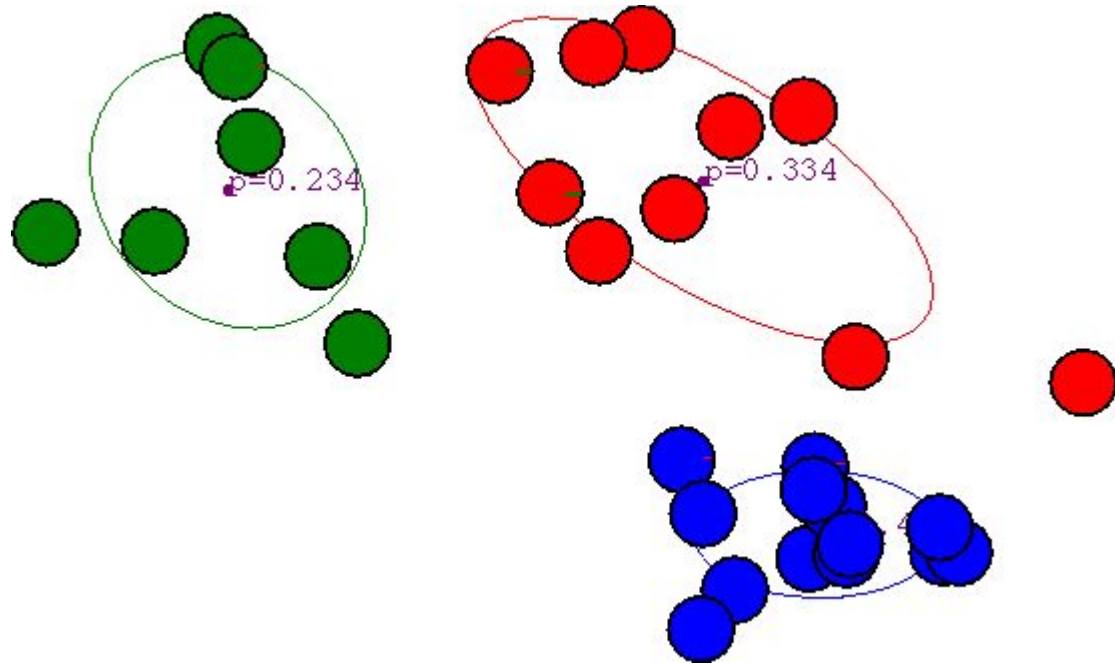
After 5th  
iteration



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