Distributed Clustering

Barnabás Póczos

Contents

- Clustering
  - K-means
  - Mixture of Gaussians

Clustering

What is clustering?

Clustering:

The process of grouping a set of objects into classes of similar objects

–high intra-class similarity

–low inter-class similarity

–It is the most common form of unsupervised learning

Clustering is Subjective

What is clustering?

Clustering:

The process of grouping a set of objects into classes of similar objects

–high intra-class similarity

–low inter-class similarity

–It is the most common form of unsupervised learning

Clustering is subjective

Simpson's Family | School Employees | Females | Males

What is Similarity?

Hard to define! *…but we know it when we see it*

The K- means Clustering Problem

K-means Clustering Problem

Given a set of observations $(x_1, x_2, \ldots, x_n)$, where $x_i \in \mathbb{R}^d$

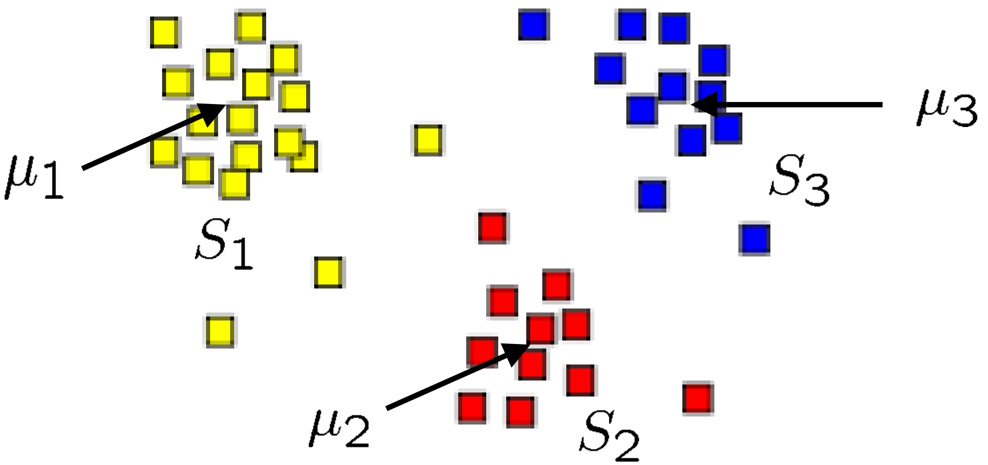
***K*-means clustering problem:**

Partition the *n* observations into *K* sets ($K \le n$) $\mathbf{S} = \{S_1, S_2, \ldots, S_K\}$ such that the sets minimize the within-cluster sum of squares:

$$\underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{K} \sum_{\mathbf{x}_j \in S_i} \left\| \mathbf{x}_j - \boldsymbol{\mu}_i \right\|^2$$

where μ_i is the mean of points in set S_i.

K=3



K-means Clustering Problem

Given a set of observations $(x_1, x_2, \ldots, x_n)$, where $x_i \in \mathbb{R}^d$

***K*-means clustering problem:**

Partition the *n* observations into *K* sets ($K \le n$) $\mathbf{S} = \{S_1, S_2, \ldots, S_K\}$ such that the sets minimize the within-cluster sum of squares:

$$\underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{K} \sum_{\mathbf{x}_j \in S_i} \left\| \mathbf{x}_j - \boldsymbol{\mu}_i \right\|^2$$

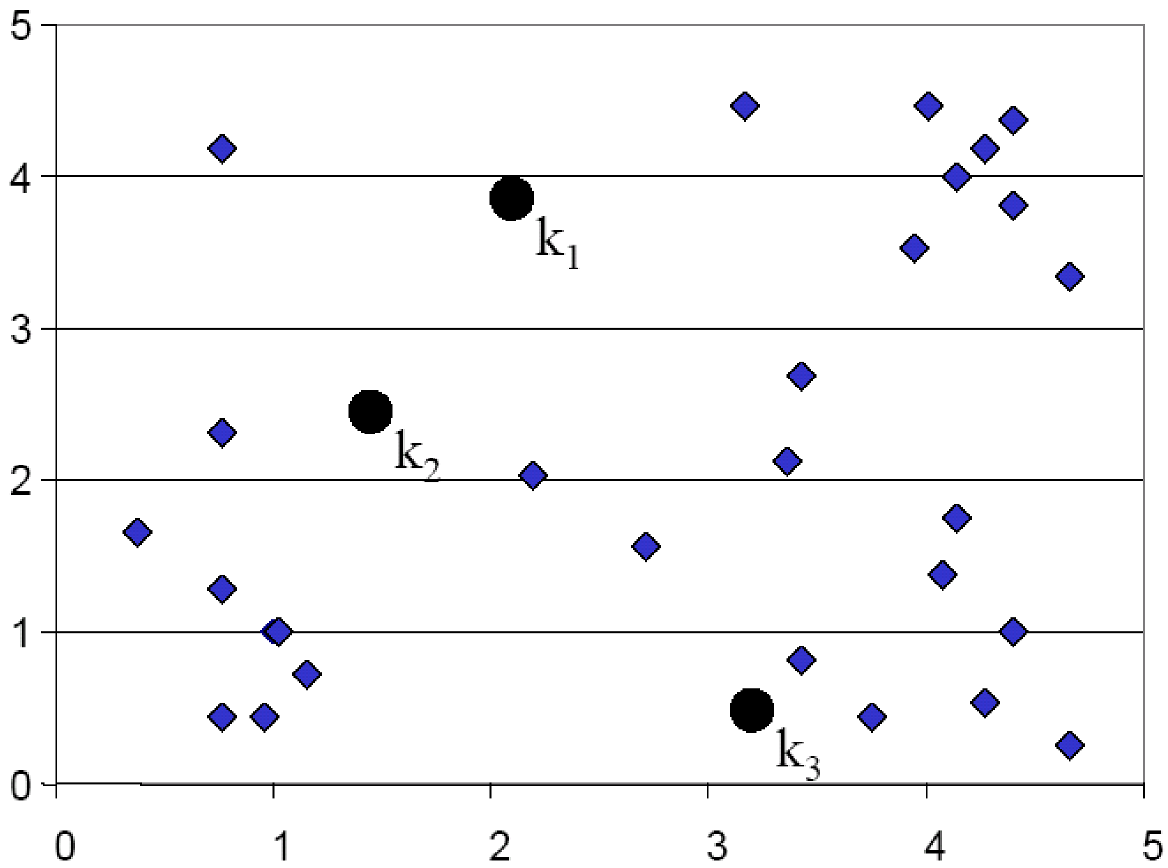
where μ_i is the mean of points in set S_i.

How hard is this problem?

The problem is NP hard, but there are good heuristic algorithms that seem to work well in practice:

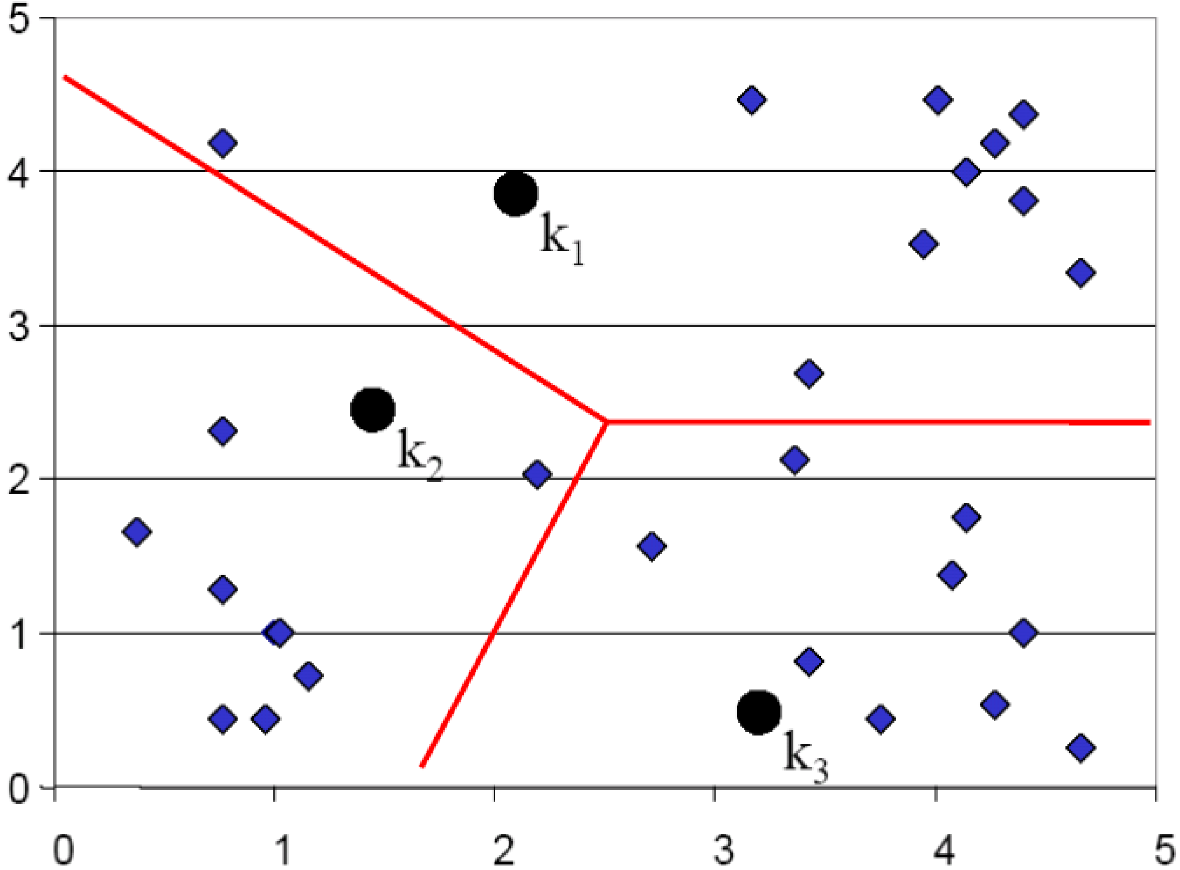
- K–means algorithm
- mixture of Gaussians

K-means Clustering Alg: Step 1



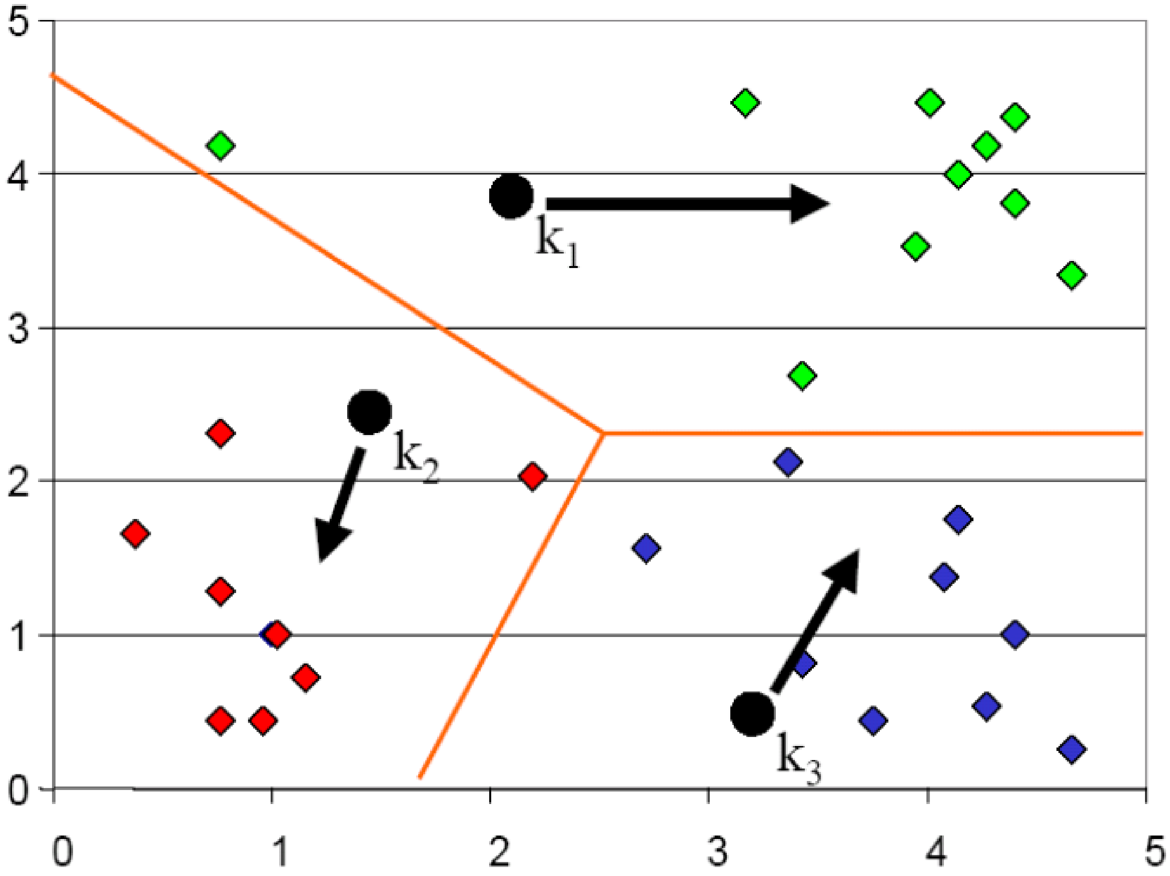
- Given n objects.
- Guess the cluster centers (k_1, k_2, k_3. They were μ_1, μ_2,μ_3 in the previous slide)

K-means Clustering Alg: Step 2



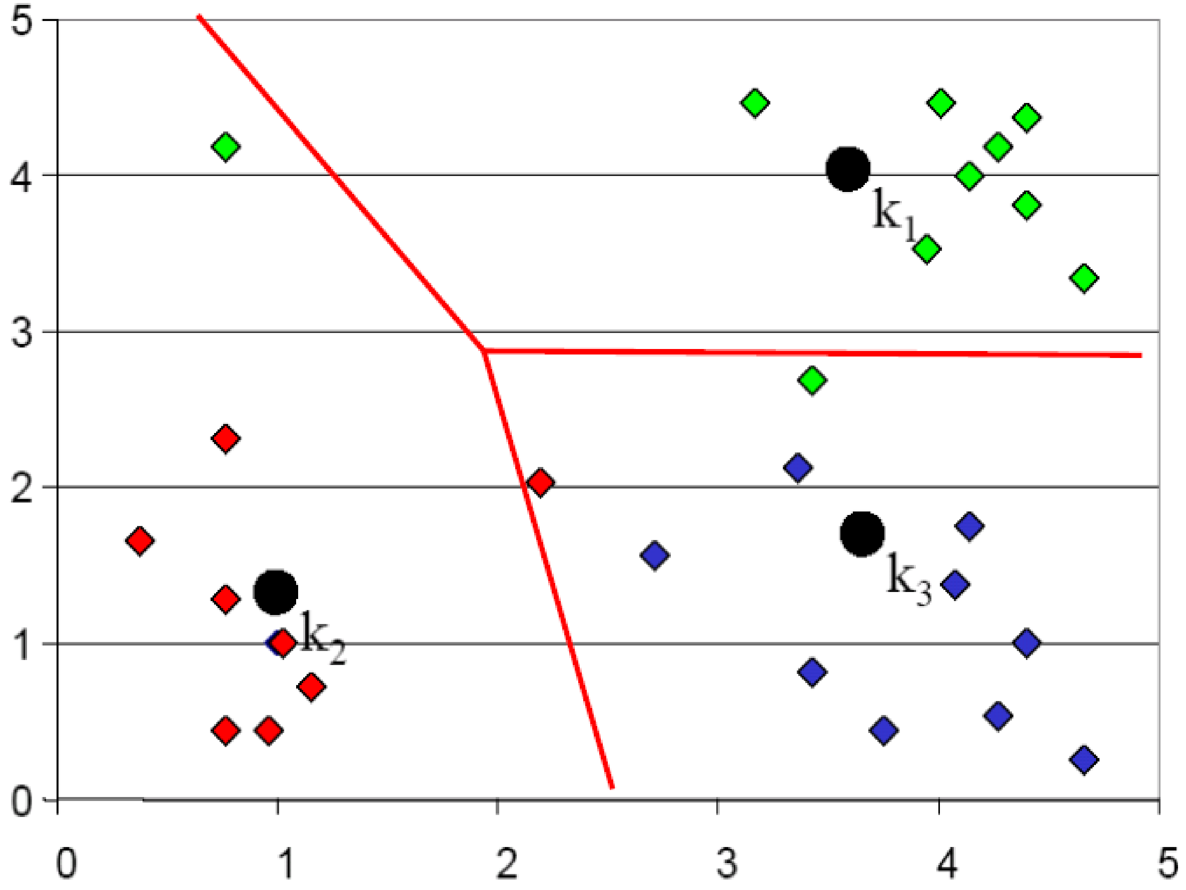
- Build a Voronoi diagram based on the cluster centers k_1, k_2, k_3.
- Decide the class memberships of the n objects by assigning them to the nearest cluster centers k_1, k_2, k_3.

K-means Clustering Alg: Step 3



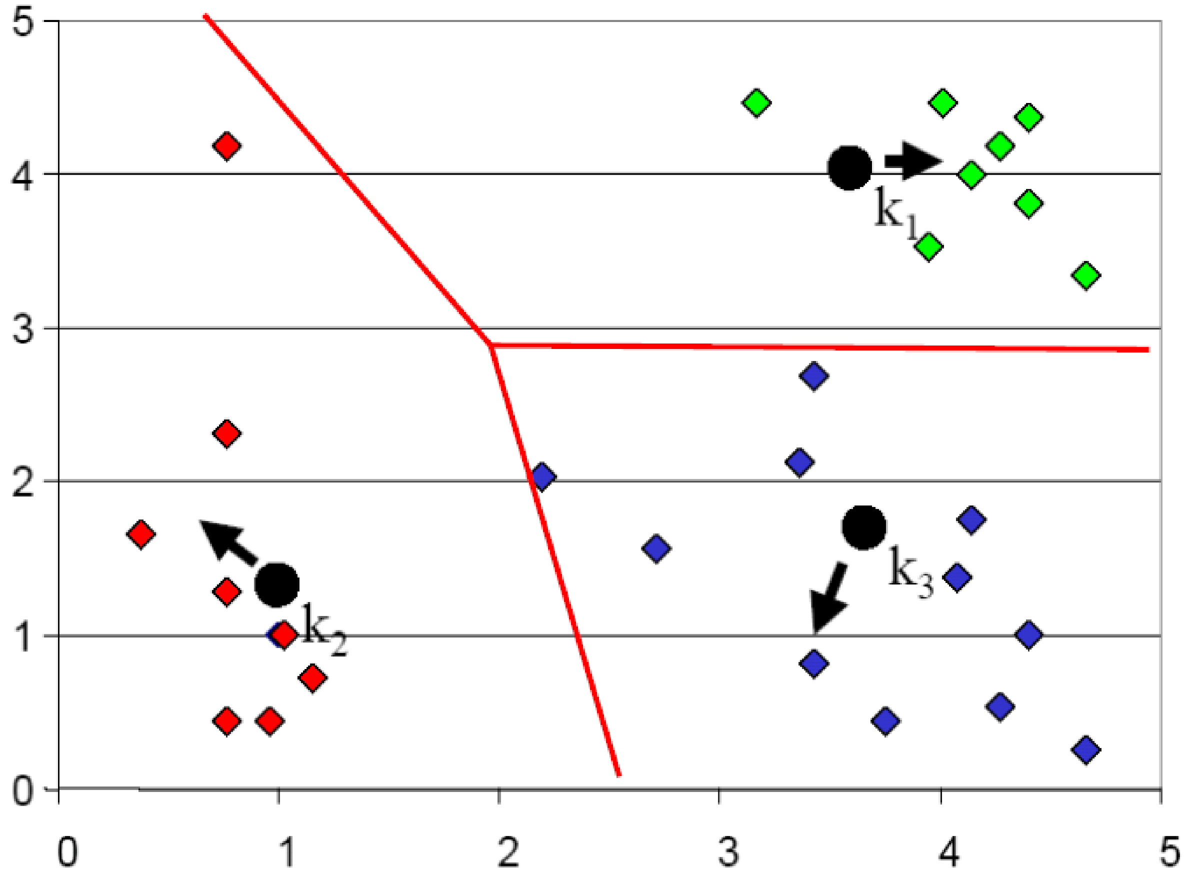
- Re-estimate the cluster centers (aka the centroid or mean), by assuming the memberships found above are correct.

K-means Clustering Alg: Step 4



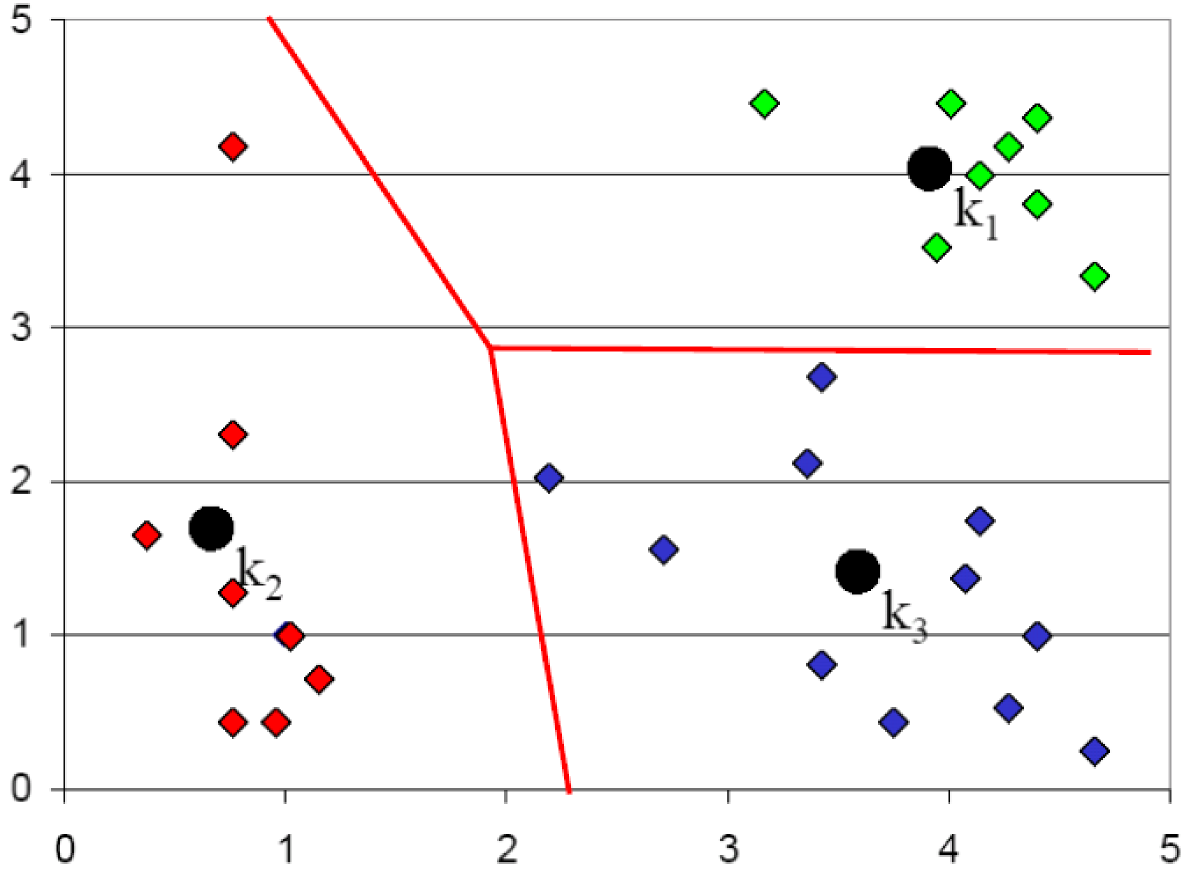
- Build a new Voronoi diagram based on the new cluster centers.
- Decide the class memberships of the n objects based on this diagram

K-means Clustering Alg: Step 5



- Re-estimate the cluster centers.

K-means Clustering Alg: Step 6



- Stop when everything is settled.
  (The Voronoi diagrams don’t change anymore)

K- means Clustering Algorithm

Algorithm

Input

– Data + Desired number of clusters, K

Initialize

– the K cluster centers (randomly if necessary)

Iterate

1. Decide the class memberships of the n objects by assigning them to the nearest cluster centers

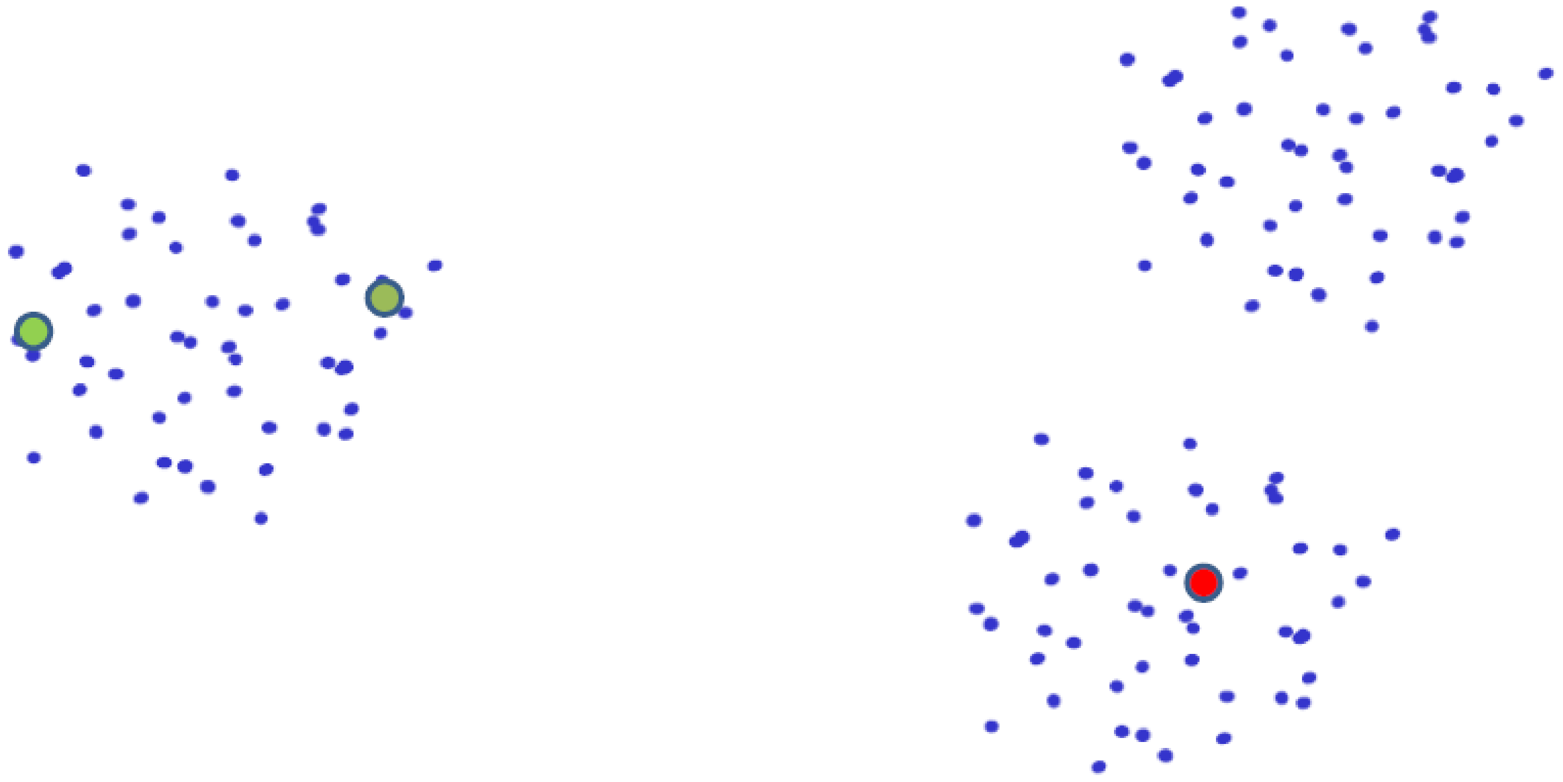
2. Re-estimate the K cluster centers (aka the centroid or mean), by assuming the memberships found above are correct.

Termination

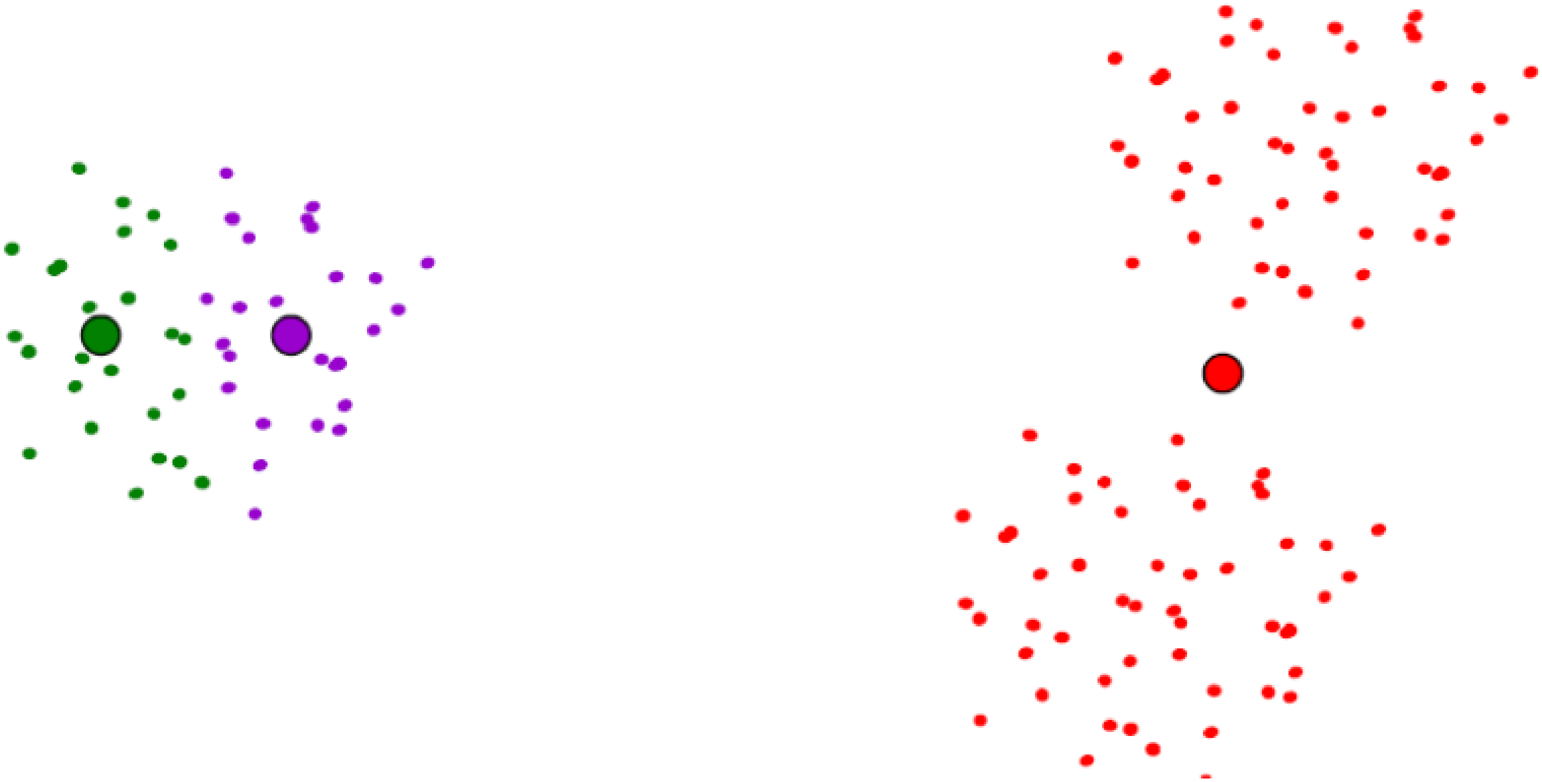
– If none of the n objects changed membership in the last iteration, exit.

Otherwise go to 1.

Seed Choice



Seed Choice



Seed Choice

The results of the K- means Algorithm can vary based on random seed selection.

- Some seeds can result in **poor convergence rate**, or convergence to **sub-optimal** clustering.
- K-means algorithm can get stuck easily in **local minima.**
  - Select good seeds using a heuristic (e.g., object least similar to any existing mean)
  - Try out **multiple** starting points (very important!!!)
  - Initialize with the results of another method.

How to Scale it Up to Large Datasets?

Distributed K- means Clustering Algorithm

Algorithm

Input: Data + Desired number of clusters, K

Initialize: the K cluster centers (randomly if necessary)

Preprocess: Partition the data and place them on the workers
(e.g. worker_1 one will process n_1, … ,worker_m will process n_m datapoints

Iterate

1. Server sends **all** the K cluster centers to the m workers

2. Decide the class memberships of the $n_1+\ldots+n_m$ objects on the workers by assigning each object to the nearest cluster centers. Do this parallel on the m workers.

3. After step 2 is done on all workers, the server collects parallel some information from the m workers (new center vectors on each worker) and re-estimate the K cluster centers on the server as the weighted mean of these vectors

Termination

– If none of the n objects changed membership in the last iteration, exit.

Otherwise go to 1.

Distributed K- means Clustering Algorithm

Distribute the data to the M workers

The m^{th} worker stores n_m d-dimensional data points.

$(n_1 + n_2 + \ldots + n_M = n)$ We have n_m training data on the m^{th} worker:

$$D_m = \{x_1^m, \ldots, x_{n_m}^m\} \quad \text{where } x_j^m \in \mathbb{R}^d \text{ for all } 1 \leq j \leq n_m,\ 1 \leq m \leq M$$

All training data: $D = \{D_1, \ldots, D_M\}$

$$C^0 = [\mu_1^0, \ldots, \mu_K^0] \in \mathbb{R}^{d \times K}$$
$$t = 0$$

The server sends $C^t \in \mathbb{R}^{d \times K}$ to the workers

Worker 1
Dataset D_1
Cluster centers C^0

…

Worker m
Dataset D_m
Cluster centers C^0

…

Worker M
Dataset D_M
Cluster centers C^0

E Step Starts

Distributed K- means Clustering Algorithm

Initial cluster centers: $C^0 = [\mu_1^0, \ldots, \mu_K^0] \in \mathbb{R}^{d \times K}$

$t = 0$

E-step. Parallel on the M worker machines

for $j = 1 \ldots n_m$: # For all data points on the m^{th} machine

for $i = 1 \ldots K$: # For all cluster centers

$Dist^m(i) = \|x_j^m - \mu_i^t\|^2$ # Distance between the j^{th} data point
and the i^{th}cluster center.

$x_j^m[\text{``}label\text{``}] = \arg\min_i Dist^m(i)$ # Label of the j^{th} data point

M-step. Parallel on the M worker machines

for $i = 1 \ldots K$: # For all cluster centers

$points^i = [x_j^m \text{ for } j \text{ in range}(n_m) \text{ if } x_j^m[\text{``}label\text{``}] == i]$

$sumPoints^m[i] = \sum_j points^i[j] \in \mathbb{R}^d$

$numPoints^m[i] = \text{len}(points^i)$

The m^{th} worker sends $numPoints^m \in \mathbb{R}^K$ and $sumPoints^m \in \mathbb{R}^{d \times K}$ to the server.

Distributed K- means Clustering Algorithm

Data aggregation on the server.

for $i = 1 \ldots K$:

$$\mu_i^{t+1} = \frac{\sum_{m=1}^{M} sumPoints^m[i]}{\sum_{m=1}^{M} numPoints^m[i]}$$

The updated cluster centers

Set this to a random vector if we divide by zero

$$C^{t+1} = [\mu_1^{t+1}, \ldots, \mu_K^{t+1}] \in \mathbb{R}^{d \times K}$$

We will resend these updated cluster centers to the workers

t=t+1, Go back to the E-Step and Repeat.

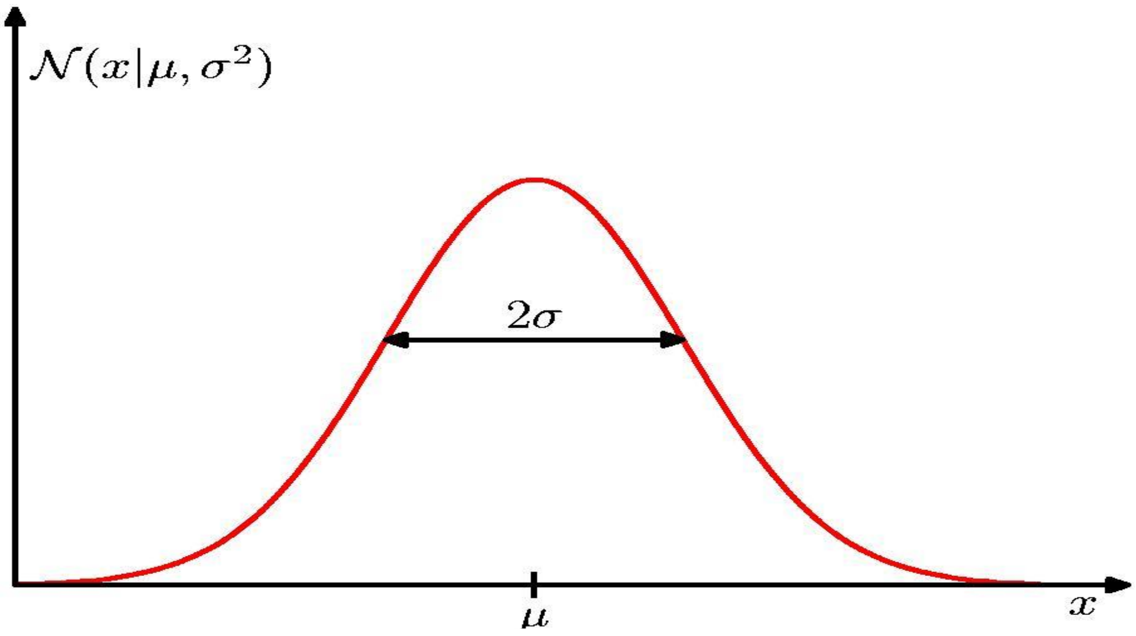
Gaussian Mixture Model

Properties of
Multivariate Gaussian Distributions

1-Dimensional Gaussian

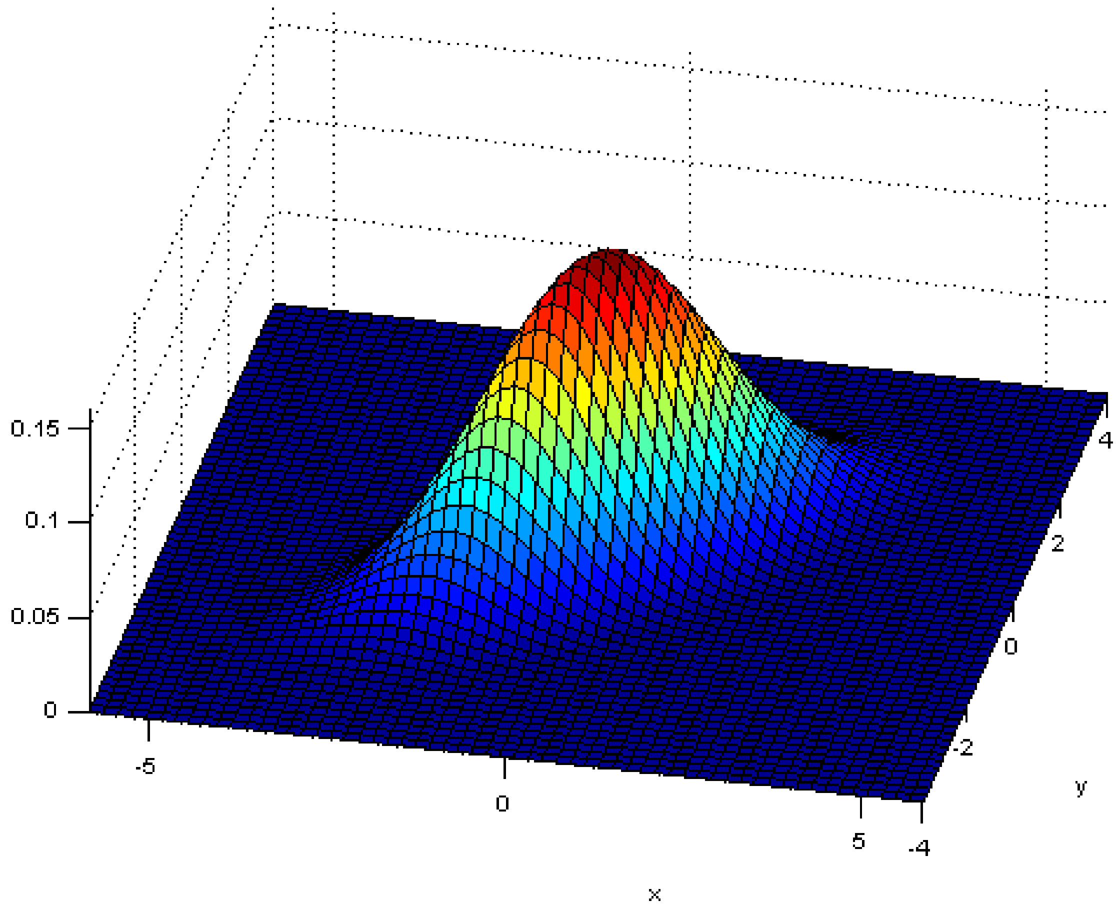
Parameters

- Mean, μ
- Variance, σ^2



$$P(x \mid \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

Multivariate Gaussian



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

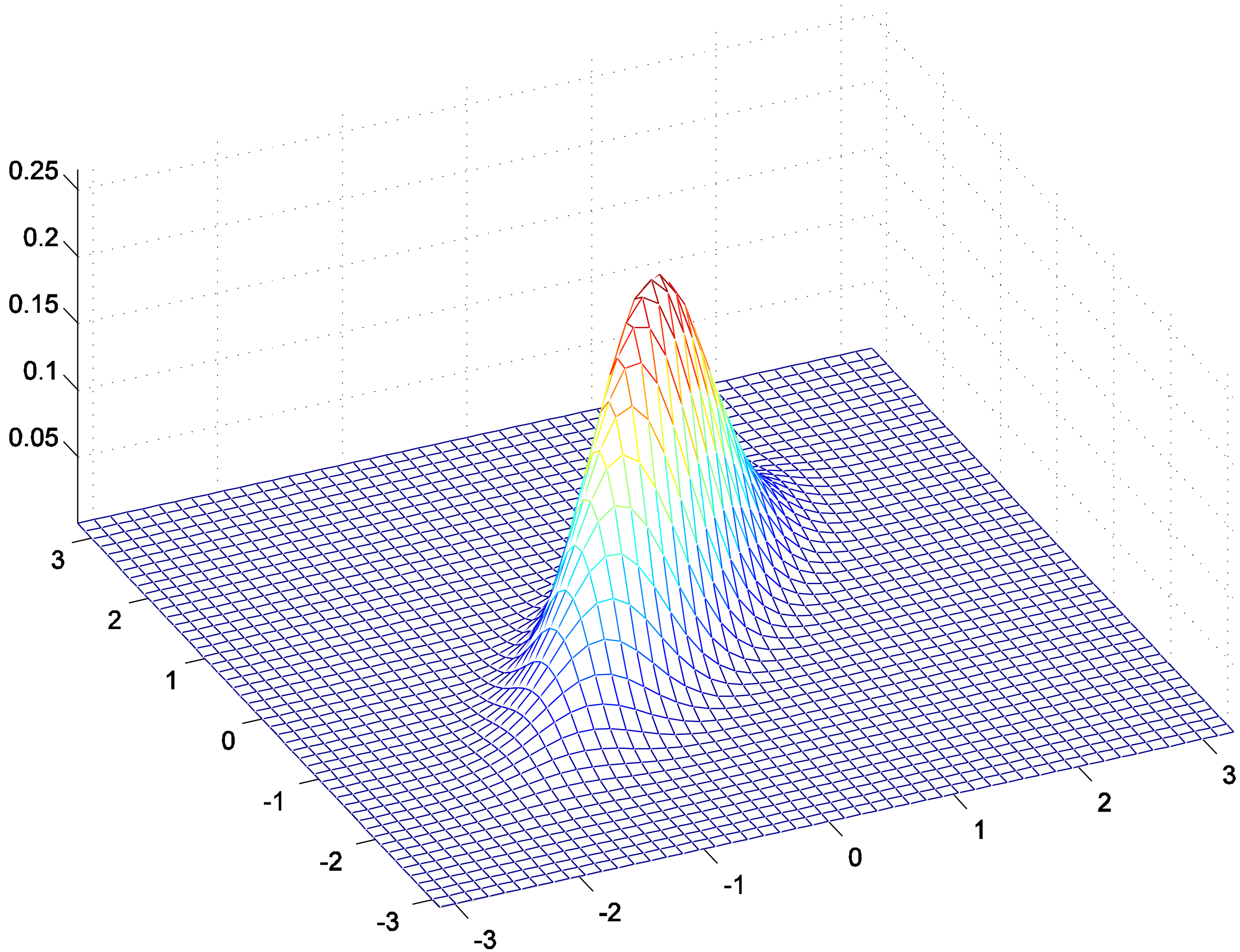
Multivariate Gaussian

- A 2-dimensional Gaussian is defined by
  - a mean vector $\mu = [\ \mu_1, \mu_2\ ]$
  - a covariance matrix: $\Sigma = \begin{bmatrix} \sigma_{1,1} & \sigma_{2,1} \\ \sigma_{1,2} & \sigma_{2,2} \end{bmatrix}$

  where $\sigma_{i,j} = E[\ (x_i - \mu_i)\ (x_j - \mu_j)\ ]$
  is the (co)variance

- Note: Σ is symmetric,
  "positive semi-definite": $\forall x$: $x^T \Sigma x \geq 0$

Multivariate Gaussian examples



$\mu = (0,0)$ $\Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$

Useful Properties of Gaussians

- Marginal distributions of Gaussians are Gaussian
- Given:

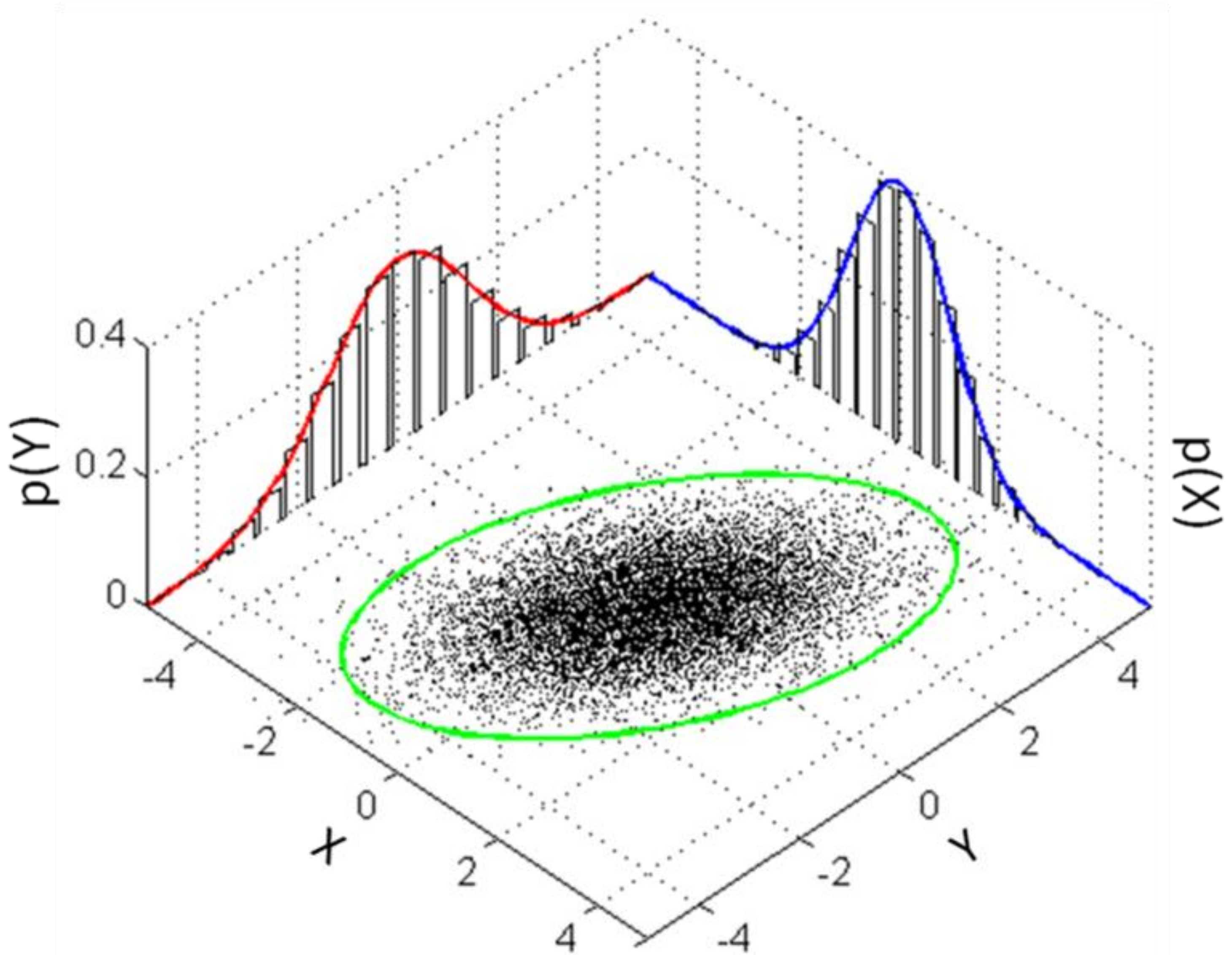
$$x = (x_a, x_b), \mu = (\mu_a, \mu_b)$$

$$\Sigma = \begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix}$$

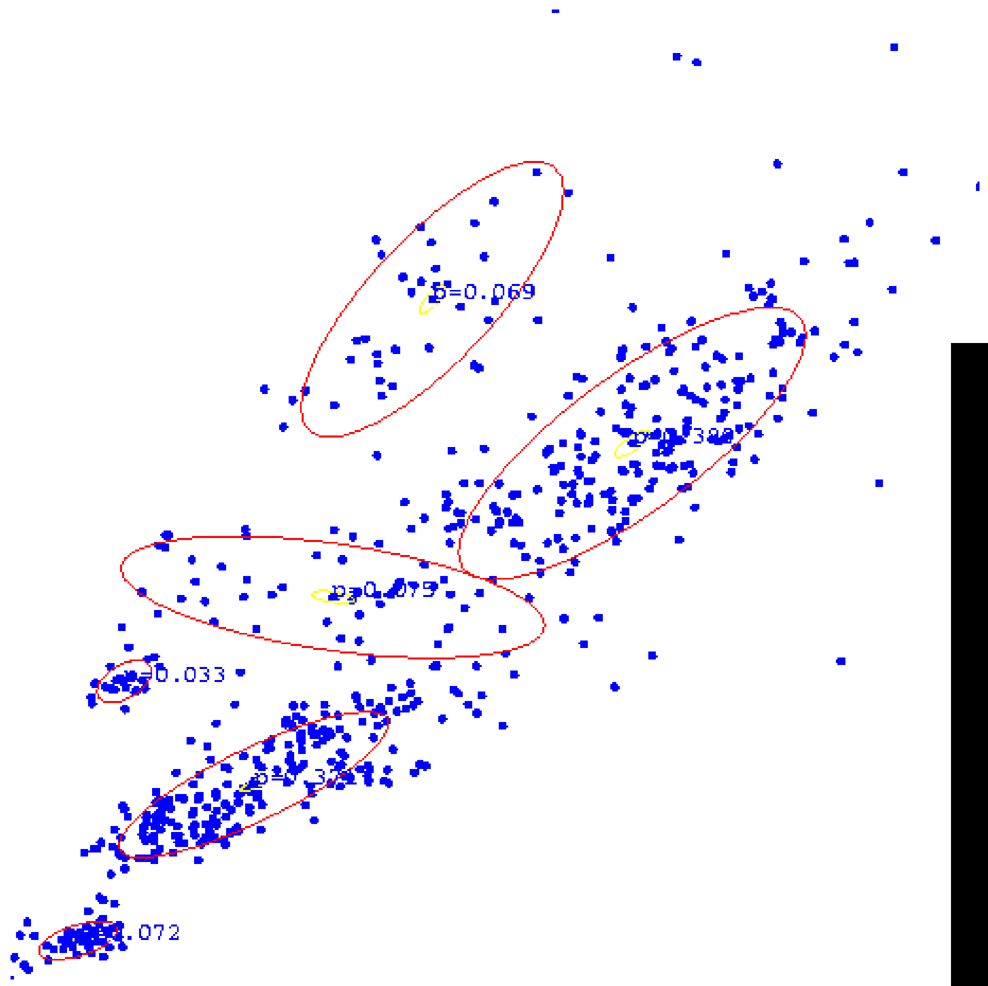
- Marginal Distribution:

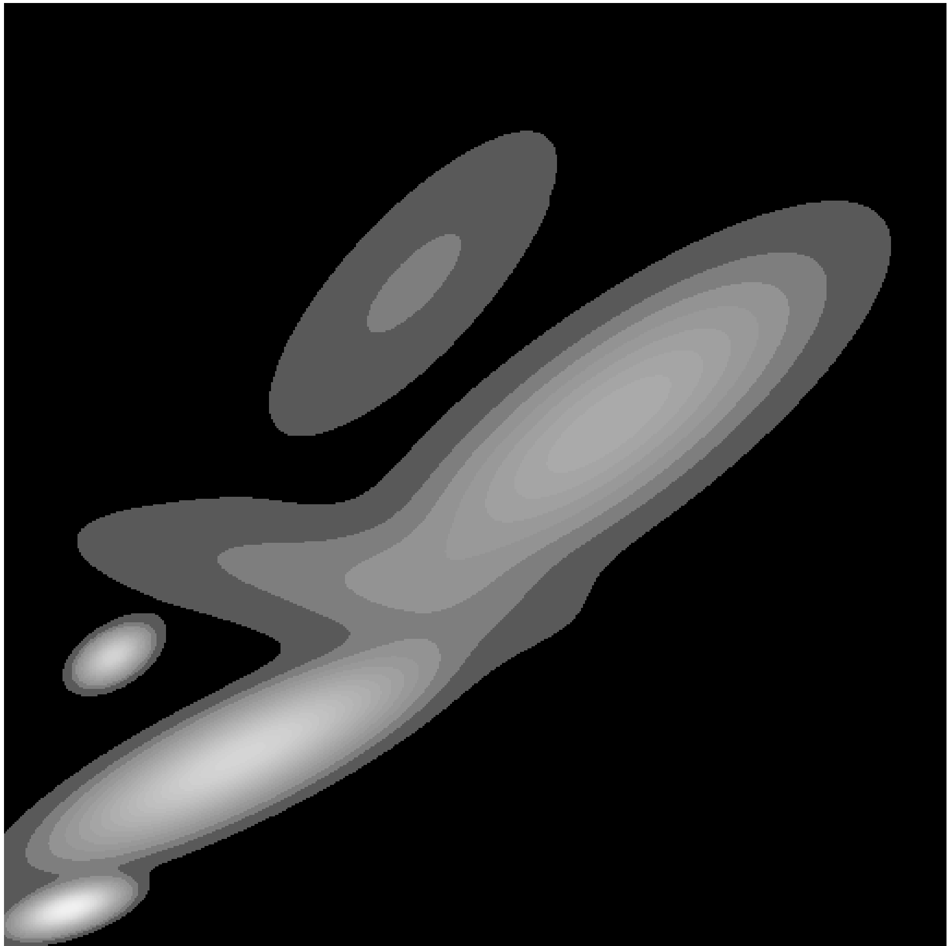
$$p(X_a) = \mathcal{N}(x_a \mid \mu_a, \Sigma_{aa})$$

Marginal distributions of Gaussians are Gaussian



Goal: GMM for Density Estimation





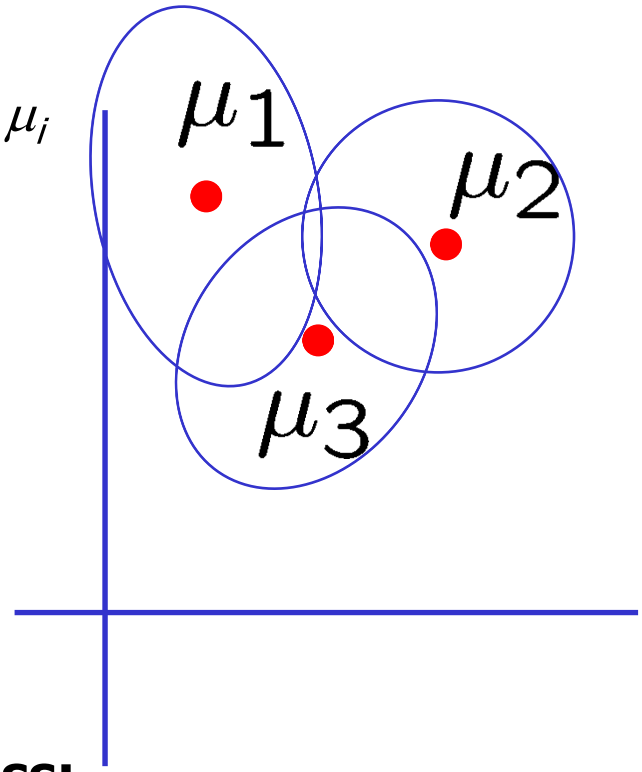
GMM as a Generative Model

The Generative Gaussian Mixture Model

Mixture of K Gaussians distributions: (Multi-modal distribution)

- There are K components
- Component *i* has an associated mean vector μ_i

Component *i* generates data from $N(\mu_i, \Sigma_i)$



Each data point is generated using this process:

1) Choose component i with probability $\pi_i = P(y = i)$
2) Datapoint $x \sim N(\mu_i, \Sigma_i)$

Gaussian Mixture Model

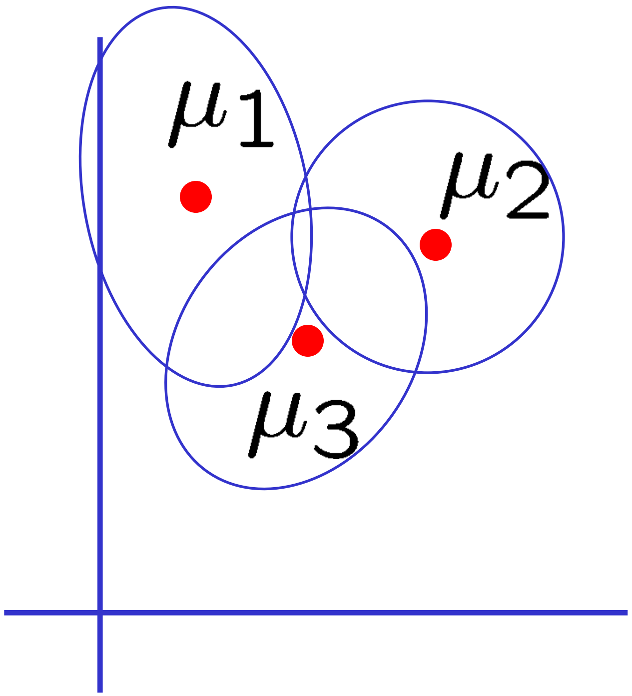
Mixture of K Gaussians distributions: (Multi-modal distribution)

Hidden variable

$$p(x|y=i) = N(\mu_i, \Sigma_i)$$

$$p(x) = \sum_{i=1}^{K} p(x|y=i) P(y=i)$$

Observed data · **Mixture component** · **Mixture proportion**



Mixture of Gaussians Clustering

Assume that

$\Sigma_i = \sigma^2 \mathrm{I}$, for simplicity.

$p(x|y=i) = N(\mu_i, \sigma^2 \mathbf{I})$

$p(y=i) = \pi_i$

All prameters $\mu_1, \ldots, \mu_K, \sigma^2, \pi_1, \ldots, \pi_K$ are known.

For a given x we want to decide if it belongs to cluster i or cluster j

Cluster x based on the ratio of posteriors:

$$\log \frac{P(y=i|x)}{P(y=j|x)}$$

$$= \log \frac{p(x|y=i)P(y=i)/p(x)}{p(x|y=j)P(y=j)/p(x)}$$

$$= \log \frac{p(x|y=i)\pi_i}{p(x|y=j)\pi_j} = \log \frac{\pi_i \exp(\frac{-1}{2\sigma^2}\|x-\mu_i\|^2)}{\pi_j \exp(\frac{-1}{2\sigma^2}\|x-\mu_j\|^2)}$$

Mixture of Gaussians Clustering

Assume that

$\Sigma_i = \sigma^2 \mathbf{I}$, for simplicity. $\quad p(x|y=i) = N(\mu_i, \sigma^2 \mathbf{I})$

$p(y=i) = \pi_i \quad \mu_1, \ldots, \mu_K, \sigma^2, \pi_1, \ldots, \pi_K$ are known.

$$\log \frac{P(y=i|x)}{P(y=j|x)} = \log \frac{p(x|y=i)\pi_i}{p(x|y=j)\pi_j} = \log \frac{\pi_i \exp(\frac{-1}{2\sigma^2}\|x-\mu_i\|^2)}{\pi_j \exp(\frac{-1}{2\sigma^2}\|x-\mu_j\|^2)}$$

$$= \log \frac{\pi_i}{\pi_j} - \frac{1}{2\sigma^2}\underbrace{\|x-\mu_i\|^2}_{x^T x + \mu_i^T \mu_i - 2x^T \mu_i} + \frac{1}{2\sigma^2}\underbrace{\|x-\mu_j\|^2}_{x^T x + \mu_j^T \mu_j - 2x^T \mu_j}$$

$$= \log \frac{\pi_i}{\pi_j} - \frac{1}{2\sigma^2}\left[\mu_i^T \mu_i - \mu_j^T \mu_j + 2x^T(\mu_j - \mu_i)\right]$$

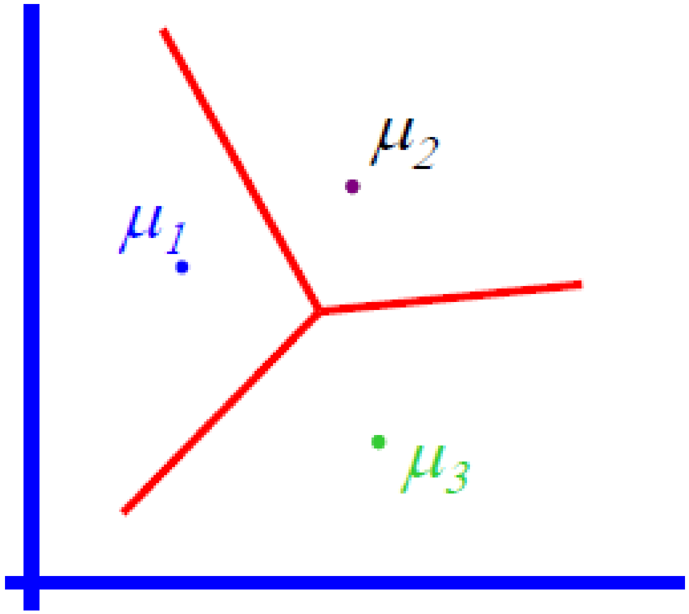
$$= c + w^T x$$

$c + w^T x > 0 \Rightarrow x$ belongs to cluster i

$c + w^T x < 0 \Rightarrow x$ belongs to cluster j

Linear decision boundary

Piecewise linear decision boundary

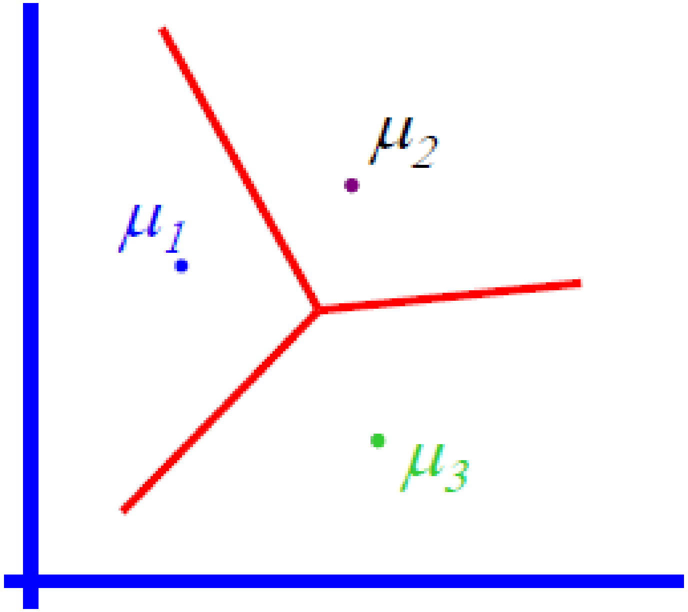


MLE for GMM

What if we don't know the parameters? $\mu_1, \ldots, \mu_K, \sigma^2, \pi_1, \ldots, \pi_K?$

Maximum Likelihood Estimate (MLE)

$$\theta = [\mu_1, \ldots, \mu_K, \sigma^2, \pi_1, \ldots, \pi_K]$$



$$\arg\max_\theta \prod_{j=1}^n P(x_j|\theta)$$

$$= \arg\max_\theta \prod_{j=1}^n \sum_{i=1}^K P(y_j = i, x_j|\theta)$$

$$= \arg\max_\theta \prod_{j=1}^n \sum_{i=1}^K P(y_j = i|\theta) p(x_j|y_j = i, \theta)$$

$$= \arg\max_\theta \prod_{j=1}^n \sum_{i=1}^K \pi_i \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-1}{2\sigma^2}\|x_j - \mu_i\|^2\right)$$

General GMM

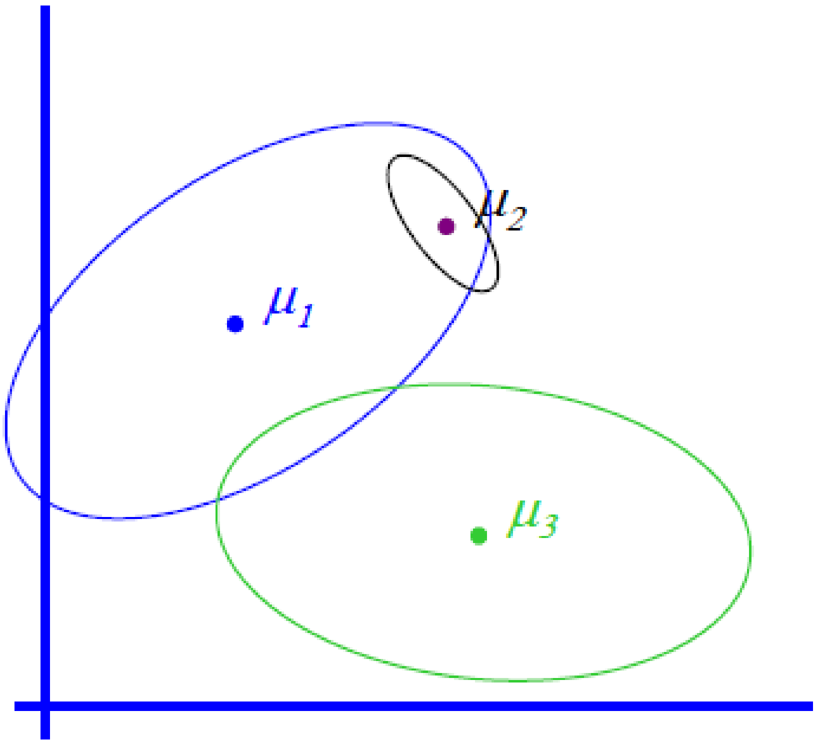
General GMM

General GMM –Gaussian Mixture Model

- There are k components

- Component *i* has an associated mean vector μ_i

- Each component generates data from a Gaussian with mean μ_i and covariance matrix Σ_i. Each data point is generated according to the following recipe:



1) Pick a component at random: Choose component i with probability *P(y=i)*

2) Datapoint x~ N(μ_i ,Σ_i)

General GMM

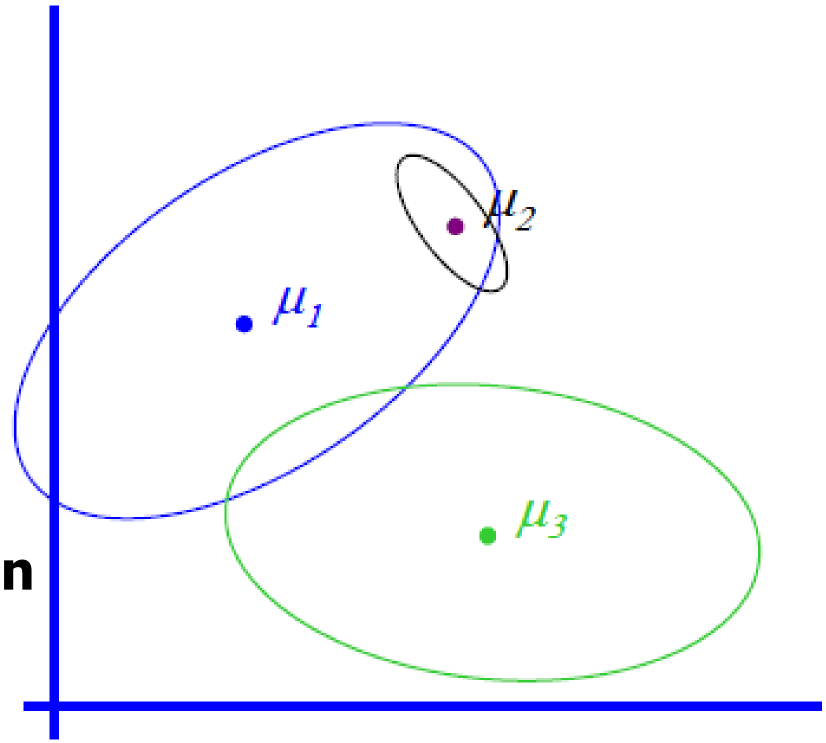
GMM –Gaussian Mixture Model

$$p(x|y=i) = N(\mu_i, \Sigma_i)$$

$$p(x) = \sum_{i=1}^{K} \underbrace{p(x|y=i)}_{\text{Mixture component}} \underbrace{P(y=i)}_{\text{Mixture proportion}}$$

Mixture component

Mixture proportion



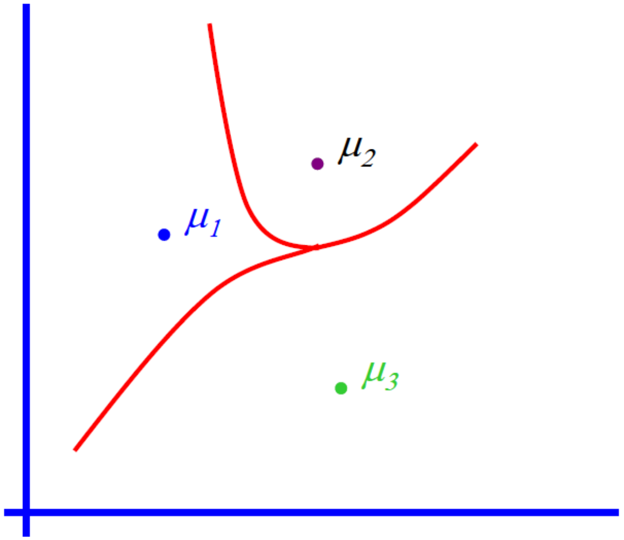
Clustering with General GMM

Assume that

$\theta = [\mu_1, \ldots, \mu_K, \Sigma_1, \ldots, \Sigma_K, \pi_1, \ldots, \pi_K]$ are known.

$$p(x|y=i) = N(\mu_i, \Sigma_i)$$

$$p(y=i) = \pi_i$$



Clustering based on ratios of posteriors:

$$\log \frac{P(y=i|x)}{P(y=j|x)}$$

$$= \log \frac{p(x|y=i)P(y=i)/p(x)}{p(x|y=j)P(y=j)/p(x)}$$

$$= \log \frac{p(x|y=i)\pi_i}{p(x|y=j)\pi_j} = \log \frac{\pi_i \frac{1}{\sqrt{|2\pi\Sigma_i|}} \exp\left[-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)\right]}{\pi_j \frac{1}{\sqrt{|2\pi\Sigma_j|}} \exp\left[-\frac{1}{2}(x-\mu_j)^T \Sigma_j^{-1}(x-\mu_j)\right]}$$

$$= x^T W x + w^T x + c$$

Depends on $\mu_1, \ldots, \mu_K, \Sigma_1, \ldots, \Sigma_K, \pi_1, \ldots, \pi_K$

"Quadratic Decision boundary" – second-order terms don't cancel out

General GMM MLE Estimation

What if we don't know $\theta = [\mu_1, \ldots, \mu_K, \Sigma_1, \ldots, \Sigma_K, \pi_1, \ldots, \pi_K]$?

Maximize marginal likelihood (MLE):

$$\arg\max_\theta \prod_{j=1}^n P(x_j|\theta) = \arg\max_\theta \prod_{j=1}^n \sum_{i=1}^K P(y_j = i, x_j|\theta)$$

$$= \arg\max_\theta \prod_{j=1}^n \sum_{i=1}^K P(y_j = i|\theta) p(x_j|y_j = i|\theta)$$

$$= \arg\max_\theta \prod_{j=1}^n \sum_{i=1}^K \pi_i \frac{1}{\sqrt{|2\pi\Sigma_i|}} \exp\left[-\frac{1}{2}(x_j - \mu_i)^T \Sigma_i^{-1} (x_j - \mu_i)\right]$$

* Set $\frac{\partial}{\partial \mu_i} \log \text{Prob}(\ldots) = 0$, and solve for μ_i.

Non-linear, non-analytically solvable

* Use gradient descent. Doable, but often slow
* Use EM.

The EM Algorithm

The Training Goal

Notation

Observed data: $D = \{x_1, \ldots, x_n\}$

Unknown variables: (Cluster assignments) $y = (y_1, \ldots, y_n)$

Paramaters: $\theta = [\mu_1, \ldots, \mu_K, \pi_1, \ldots, \pi_K, \Sigma_1, \ldots, \Sigma_K]$

Goal: $\widehat{\theta}_n = \arg\max_{\theta} \log P(D|\theta)$

General EM algorithm

Goal: $\arg\max_{\theta} \log P(D|\theta)$

E Step:

$$Q(\theta|\theta^{t-1}) = \mathbb{E}_y[\log P(y, D|\theta)|D, \theta^{t-1}]$$

$$= \int dy\, P(y|D, \theta^{t-1}) \log P(y, D|\theta)$$

M Step:

$$\theta^t = \arg\max_{\theta} Q(\theta|\theta^{t-1})$$

During the EM algorithm the marginal likelihood is not decreasing!

$$P(D|\theta^t) \leq P(D|\theta^{t+1})$$

EM for general GMMs

The more general case:

- We have unlabeled data $x_1, x_2, \ldots, x_m$
- We know there are K classes
- We **don't** know $P(y=1)=\pi_1$, $P(y=2)=\pi_2$ $P(y=3)$ … $P(y=K)=\pi_K$
- We **don't** know $\Sigma_1, \ldots \Sigma_K$
- We **don't** know $\mu_1, \mu_2, \ldots \mu_K$

We want to learn: $\theta = [\mu_1, \ldots, \mu_K, \pi_1, \ldots, \pi_K, \Sigma_1, \ldots, \Sigma_K]$

Our estimator at the end of iteration t-1:

$$\theta^{t-1} = [\mu_1^{t-1}, \ldots, \mu_K^{t-1}, \pi_1^{t-1}, \ldots, \pi_K^{t-1}, \Sigma_1^{t-1}, \ldots, \Sigma_K^{t-1}]$$

The idea is the same:

At iteration t, construct function Q (E step) and maximize it in θ^t (M step)

$$Q(\theta^t|\theta^{t-1}) = \sum_{j=1}^{n} \sum_{i=1}^{K} P(y_j = i|x_j, \theta^{t-1}) \log P(x_j, y_j = i|\theta^t)$$

EM for general GMM

E-step

Compute "expected" classes of all datapoints for each class

$$R_{i,j}^{t-1} = P(y_j = i | x_j, \theta^{t-1}) = \frac{\exp\left\{-\frac{1}{2}(x_j - \mu_i^{t-1})^T (\Sigma_i^{t-1})^{-1} (x_j - \mu_i^{t-1})\right\} \pi_i^{t-1}}{\sum_{i=1}^{K} \exp\left\{-\frac{1}{2}(x_j - \mu_i^{t-1})^T (\Sigma_i^{t-1})^{-1} (x_j - \mu_i^{t-1})\right\} \pi_i^{t-1}}$$

M-step $\frac{\partial}{\partial \theta^t} Q(\theta^t | \theta^{t-1}) = 0$

Compute MLEs given our data's class membership distributions (weights)

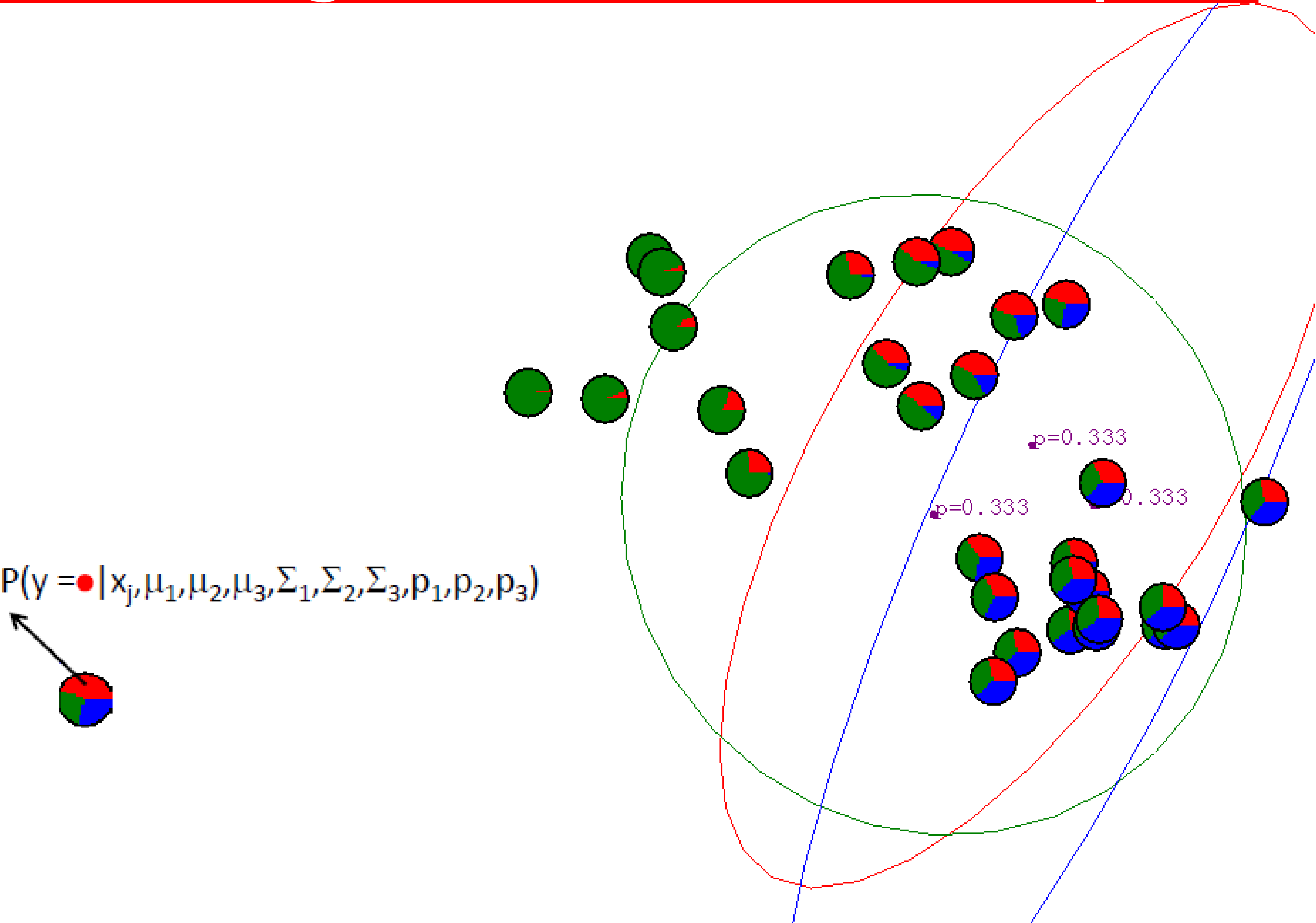
$$\mu_i^t = \sum_{j=1}^{n} w_j x_j \quad \text{where } w_j = \frac{R_{i,j}^{t-1}}{\sum_{j=1}^{n} R_{i,j}^{t-1}}$$

$$\Sigma_i^t = \sum_{j=1}^{n} w_j (x_j - \mu_i^t)^T (x_j - \mu_i^t)$$

$$\pi_i^t = \frac{1}{n} \sum_{j=1}^{n} R_{i,j}^{t-1}$$

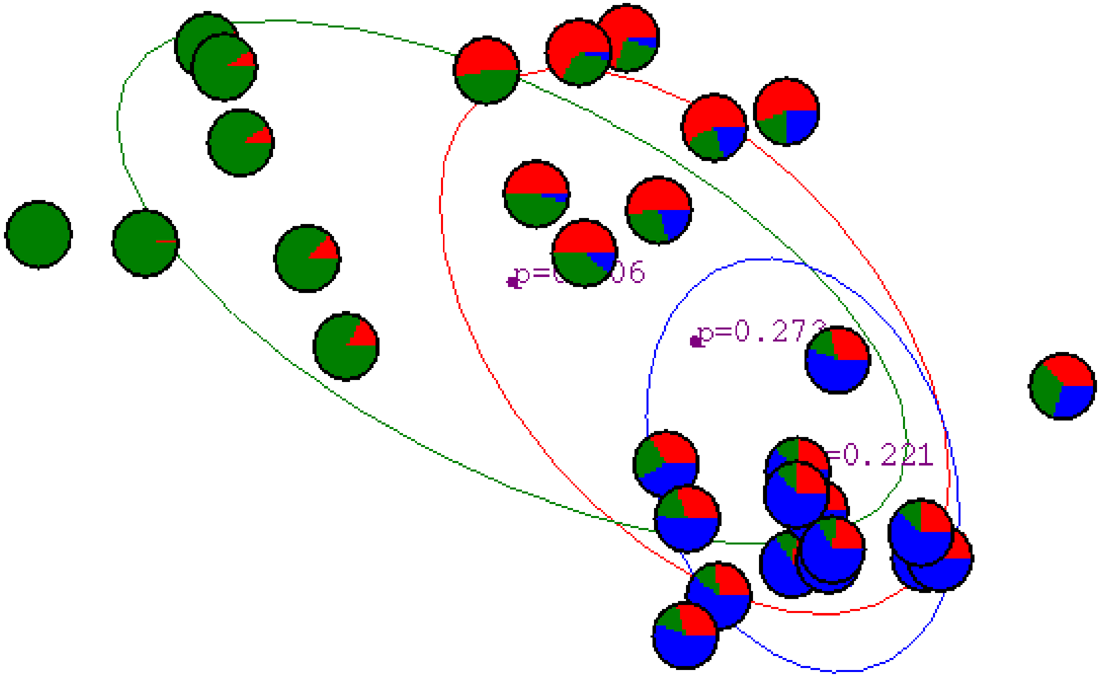
EM for general GMMs: Example

$P(y = \bullet \mid x_j, \mu_1, \mu_2, \mu_3, \Sigma_1, \Sigma_2, \Sigma_3, p_1, p_2, p_3)$



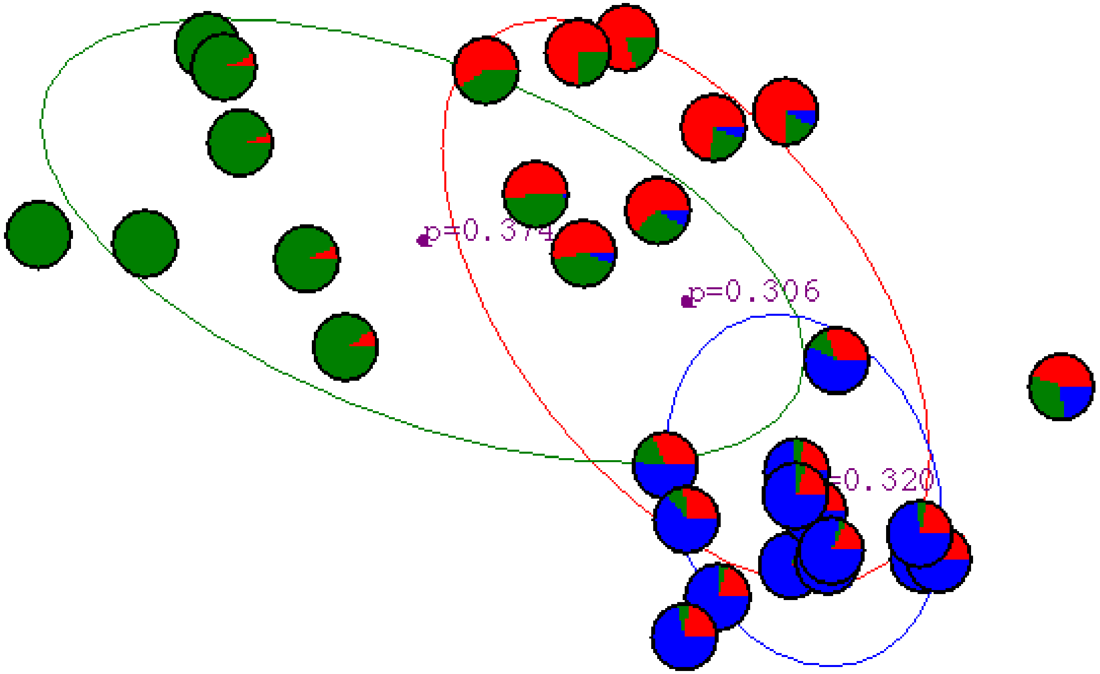
EM for general GMMs: Example

After 1st iteration



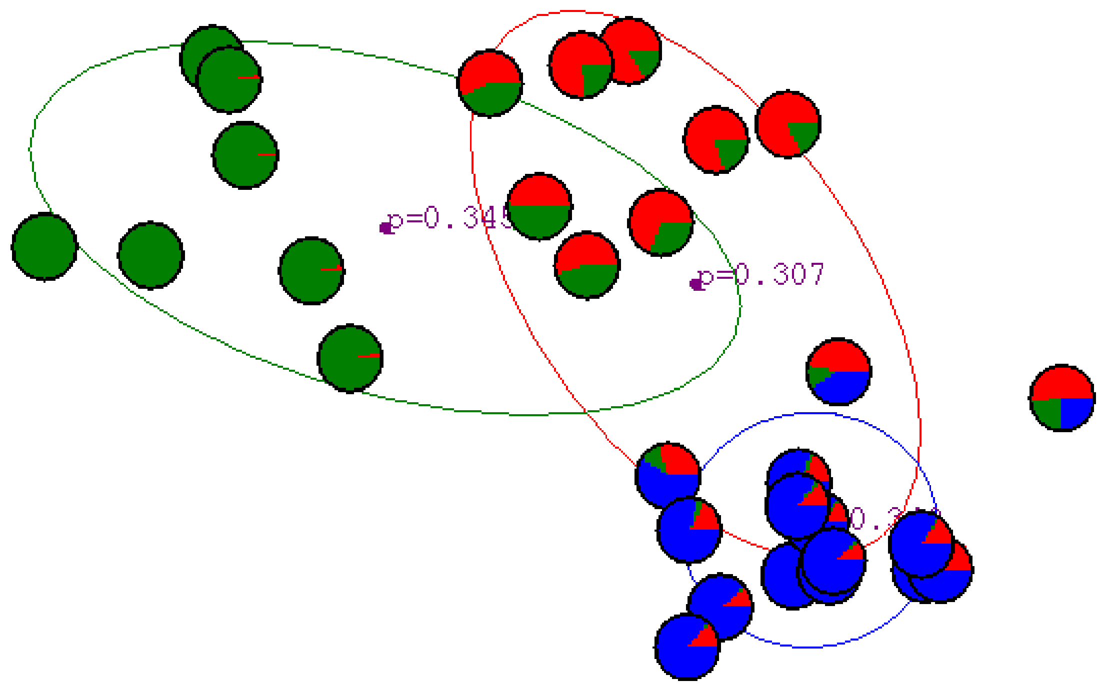
EM for general GMMs: Example

After 2nd iteration



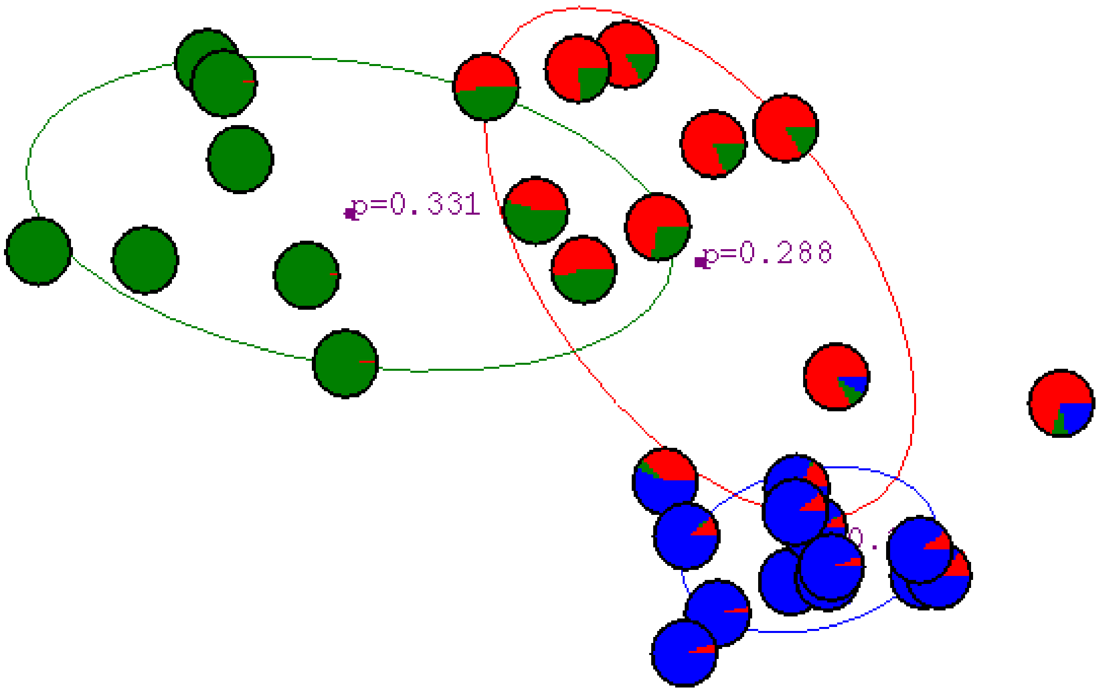
EM for general GMMs: Example

After 3rd iteration



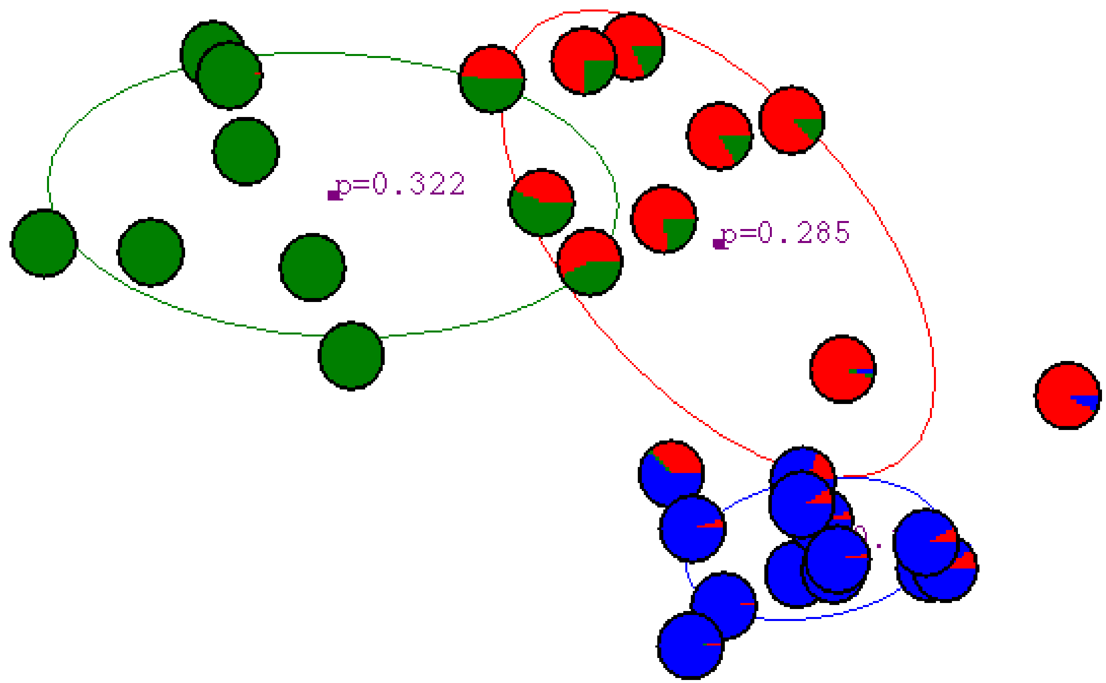
EM for general GMMs: Example

After 4th iteration



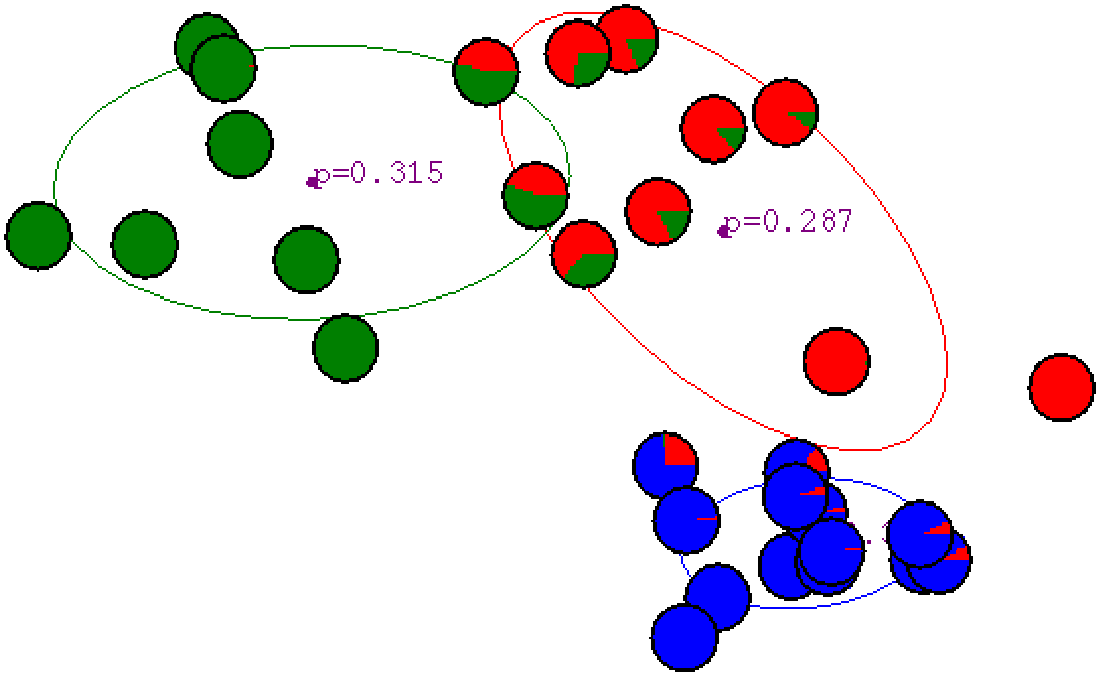
EM for general GMMs: Example

After 5th iteration



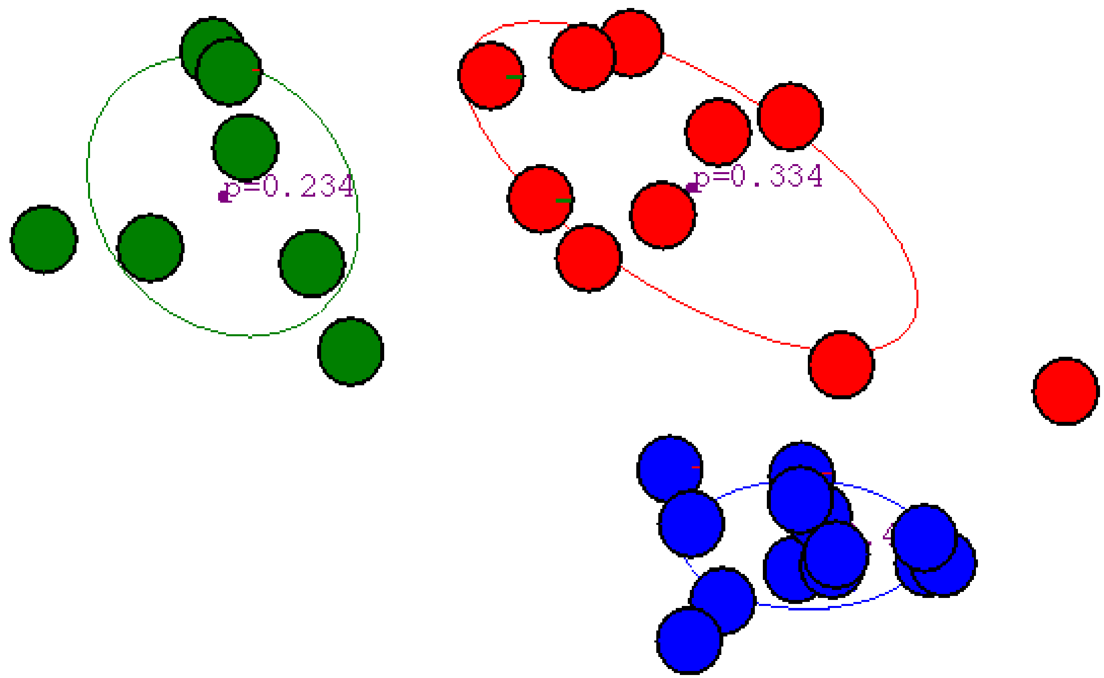
EM for general GMMs: Example

After 6th iteration

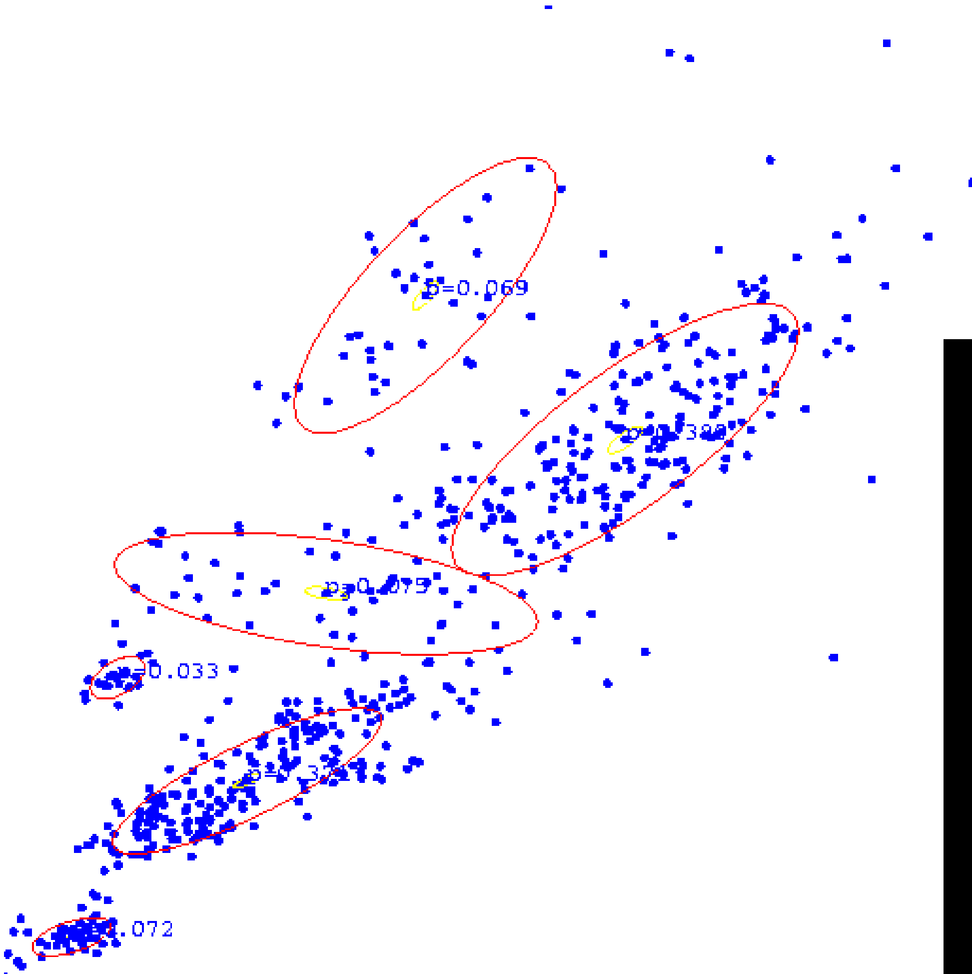


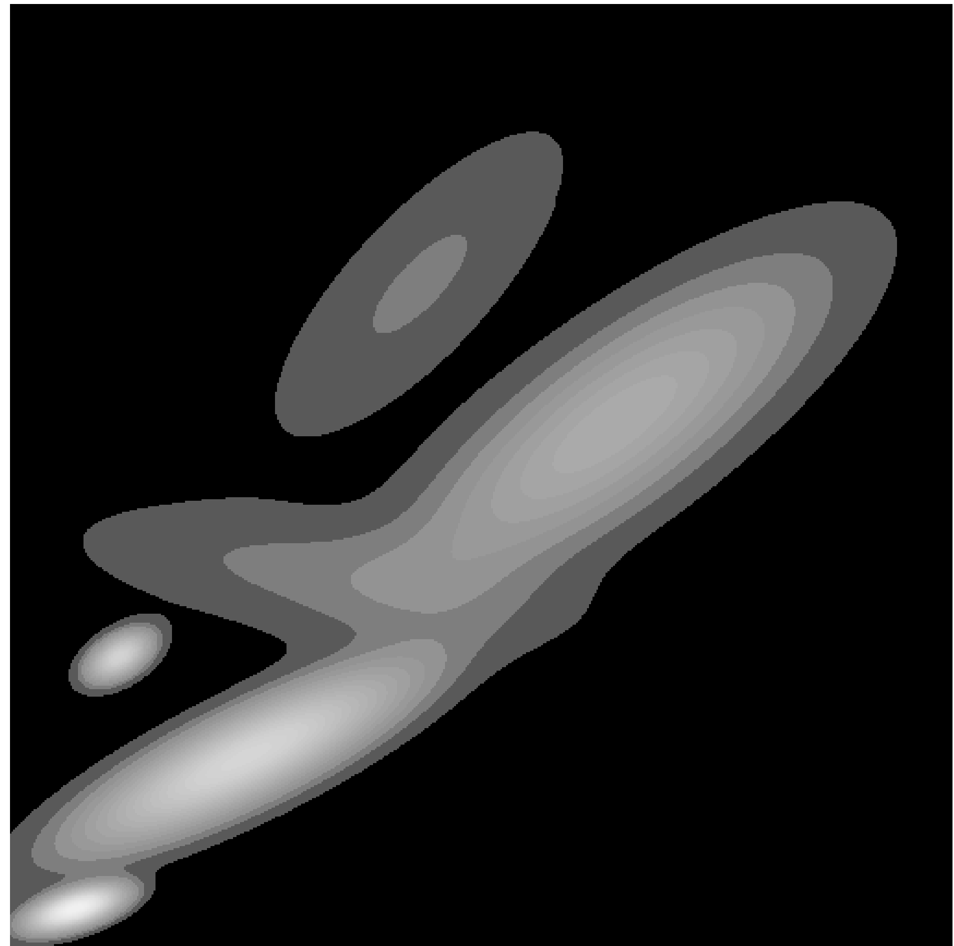
EM for general GMMs: Example

After 20th iteration



Goal: GMM for Density Estimation





Standard (nondistributed) EM

E-step

Compute "expected" classes of all datapoints for each class

$$R_{i,j}^{t-1} = P(y_j = i | x_j, \theta^{t-1}) = \frac{\exp\left\{-\frac{1}{2}(x_j - \mu_i^{t-1})^T (\Sigma_i^{t-1})^{-1} (x_j - \mu_i^{t-1})\right\} \pi_i^{t-1}}{\sum_{i=1}^{K} \exp\left\{-\frac{1}{2}(x_j - \mu_i^{t-1})^T (\Sigma_i^{t-1})^{-1} (x_j - \mu_i^{t-1})\right\} \pi_i^{t-1}}$$

M-step $\frac{\partial}{\partial \theta^t} Q(\theta^t | \theta^{t-1}) = 0$

Compute MLEs given our data's class membership distributions (weights)

$$\mu_i^t = \sum_{j=1}^{n} w_j x_j \quad \text{where } w_j = \frac{R_{i,j}^{t-1}}{\sum_{j=1}^{n} R_{i,j}^{t-1}}$$

$$\Sigma_i^t = \sum_{j=1}^{n} w_j (x_j - \mu_i^t)^T (x_j - \mu_i^t)$$

$$\pi_i^t = \frac{1}{n} \sum_{j=1}^{n} R_{i,j}^{t-1}$$

Distributed EM

Distributed the data to the M workers

$$D_m = \{x_1^m, \ldots, x_{n_m}^m\} \quad D = \{D_1 \ldots, D_M\}$$

E-step. Parallel on the M worker machines

On the m-th worker, calculate for all $i \in \{1, \ldots, K\}, j \in \{1, \ldots, n_m\}$:

$$R_{i,j,m}^{t-1} = P(y_j^m = i | x_j^m, \theta^{t-1})$$
$$= \frac{\exp\left\{-\frac{1}{2}(x_j^m - \mu_i^{t-1})^T (\Sigma_i^{t-1})^{-1} (x_j^m - \mu_i^{t-1})\right\} \pi_i^{t-1}}{\sum_{i=1}^K \exp\left\{-\frac{1}{2}(x_j^m - \mu_i^{t-1})^T (\Sigma_i^{t-1})^{-1} (x_j^m - \mu_i^{t-1})\right\} \pi_i^{t-1}}$$

After this is done, each worker can calculate and send the server:

$$s_i^m = \sum_{j=1}^{n_m} R_{i,j,m}^{t-1}$$

Then, the server can calculate $R = \sum_{m=1}^{M} s_i^m$ and send it to the workers.

The workers can calculate $w_{i,j}^m$:

$$w_{i,j}^m = \frac{R_{i,j,m}^{t-1}}{R} = \frac{R_{i,j,m}^{t-1}}{\sum_{m=1}^M \sum_{j=1}^n R_{i,j,m}^{t-1}}$$

Distributed EM

M-step 1. Calculate the updated means

Workers calculate and send to the server:

$$\tilde{\mu}_{i,m} = \sum_{j=1}^{n_m} w_{i,j}^m x_j^m$$

Server calculates and sends to the workers:

$$\mu_i^t = \sum_{m=1}^{M} \tilde{\mu}_{i,m} = \sum_{m=1}^{M} \sum_{j=1}^{n_m} w_{i,j}^m x_j^m$$

M-step 2. Calculate the updated π probability distribution

Server calculates and sends to the workers:

$$\pi_i^t = \frac{R}{\sum_{m=1}^{M} n_m} = \frac{1}{\sum_{m=1}^{M} n_m} \sum_{m=1}^{M} \sum_{j=1}^{n_m} R_{i,j,m}^{t-1}$$

Distributed EM

M-step 3. Calculate the updated covariance matrix

Workers calculate and send to the server:

$$\tilde{\Sigma}_{i,m} = \sum_{j=1}^{n_m} w_{i,j}^m (x_j^m - \mu_i^t)^T (x_j^m - \mu_i^t)$$

Server calculates and sends to the workers:

$$\Sigma_i^t = \sum_{m=1}^{M} \tilde{\Sigma}_{i,m} = \sum_{m=1}^{M} \sum_{j=1}^{n_m} w_j^m (x_j^m - \mu_i^t)^T (x_j^m - \mu_i^t)$$

t=t+1, Go back to the E-Step and Repeat.

Thanks for your attention! ☺

Mathematical Motivation for EM

Expectation-Maximization (EM)

A general algorithm to deal with hidden data, but we will study it in the context of unsupervised clustering with Gaussian mixture models.

- EM is an optimization strategy for objective functions that can be interpreted as likelihoods in the presence of missing data (e.g. cluster assignment).
- In the following examples EM is “simpler” than gradient methods: No need to choose step size.
- EM is an iterative algorithm with two linked steps:
    - o E-step: fill-in hidden values using inference
    - o M-step: apply standard MLE/MAP method to completed data
- We will prove that this procedure monotonically improves the likelihood (or leaves it unchanged).

General EM algorithm

Notation

Observed data: $D = \{x_1, \ldots, x_n\}$

Unknown variables: y

For example in clustering: $y = (y_1, \ldots, y_n)$

Parameters: θ

For example in MoG: $\theta = [\mu_1, \ldots, \mu_K, \pi_1, \ldots, \pi_K, \Sigma_1, \ldots, \Sigma_K]$

Goal: $\widehat{\theta}_n = \arg\max_{\theta} \log P(D|\theta)$

General EM algorithm

Goal: $\arg\max_\theta \log P(D|\theta)$

$$\log P(D|\theta^t) = \int dy\, q(y) log P(D|\theta^t)$$

$$= \int dy\, q(y) log \left[\frac{P(y,D|\theta^t)}{P(y|D,\theta^t)} \frac{q(y)}{q(y)}\right] \quad \text{since } P(y,D|\theta^t) = P(D|\theta^t)P(y|D,\theta^t)$$

$$= \underbrace{\int dy\, q(y) log P(y,D|\theta^t) \underbrace{- \int dy\, q(y) \log q(y)}_{H(q)}}_{\text{Free energy: } F_{\theta^t}(q(\cdot),D)} + \underbrace{\int dy\, q(y) \log \frac{q(y)}{P(y|D,\theta^t)}}_{KL(q(y)\|P(y|D,\theta^t))}$$

E Step: $Q(\theta^t|\theta^{t-1}) = \mathbb{E}_y[\log P(y,D|\theta^t)|D,\theta^{t-1}]$

$$= \int dy\, P(y|D,\theta^{t-1}) \log P(y,D|\theta^t)$$

M Step: $\theta^t = \arg\max_\theta Q(\theta|\theta^{t-1})$

We are going to discuss why this approach works

General EM algorithm

$$\log P(D|\theta^t) = \underbrace{\int dy\, q(y) log P(y, D|\theta^t) \underbrace{- \int dy\, q(y) \log q(y)}_{H(q)}}_{\text{Free energy: } F_{\theta^t}(q(\cdot), D)} + \underbrace{\int dy\, q(y) \log \frac{q(y)}{P(y|D,\theta^t)}}_{KL(q(y) \| P(y|D,\theta^t))}$$

E Step: $Q(\theta|\theta^t) = \int dy\, P(y|D,\theta^t) \log P(y, D|\theta)$

Let $q(y) = P(y|D,\theta^t)$

$\Rightarrow KL(q(y) \| P(y|D,\theta^t)) = 0$

$$\Rightarrow \log P(D|\theta^t) = F_{\theta^t}(P(y|D,\theta^t), D)$$

$$= \overbrace{\int dy\, P(y|D,\theta^t) log P(y, D|\theta^t)}^{Q(\theta^t|\theta^t)} - \int dy\, P(y|D,\theta^t) \log P(y|D,\theta^t)$$

M Step: $\leq \int dy\, P(y|D,\theta^t) log P(y, D|\theta^{t+1}) - \int dy\, P(y|D,\theta^t) \log P(y|D,\theta^t)$

$\theta^{t+1} = \arg\max_\theta Q(\theta|\theta^t)$

We maximize only here in θ!!!

General EM algorithm

$$\log P(D|\theta^t) = \underbrace{\int dy\, q(y) log P(y, D|\theta^t) - \underbrace{\int dy\, q(y) \log q(y)}_{H(q)}}_{\text{Free energy: } F_{\theta^t}(q(\cdot), D)} + \underbrace{\int dy\, q(y) \log \frac{q(y)}{P(y|D, \theta^t)}}_{KL(q(y) \| P(y|D, \theta^t))}$$

Theorem: During the EM algorithm the marginal likelihood is not decreasing!

$$P(D|\theta^t) \leq P(D|\theta^{t+1})$$

Proof:

$$\begin{aligned}
\log P(D|\theta^t) &= F_{\theta^t}(P(y|D, \theta^t), D) \\
&\leq \int dy\, P(y|D, \theta^t) log P(y, D|\theta^{t+1}) - \int dy\, P(y|D, \theta^t) \log P(y|D, \theta^t) \\
&= F_{\theta^{t+1}}(P(y|D, \theta^t), D) \\
&= \log P(D|\theta^{t+1}) - KL(P(y|D, \theta^t) \| P(y|D, \theta^{t+1})) \\
&\leq \log P(D|\theta^{t+1})
\end{aligned}$$

General EM algorithm

Goal: $\arg\max_{\theta} \log P(D|\theta)$

E Step:

$$Q(\theta|\theta^{t-1}) = \mathbb{E}_y[\log P(y, D|\theta)|D, \theta^{t-1}]$$

$$= \int dy\, P(y|D, \theta^{t-1}) \log P(y, D|\theta)$$

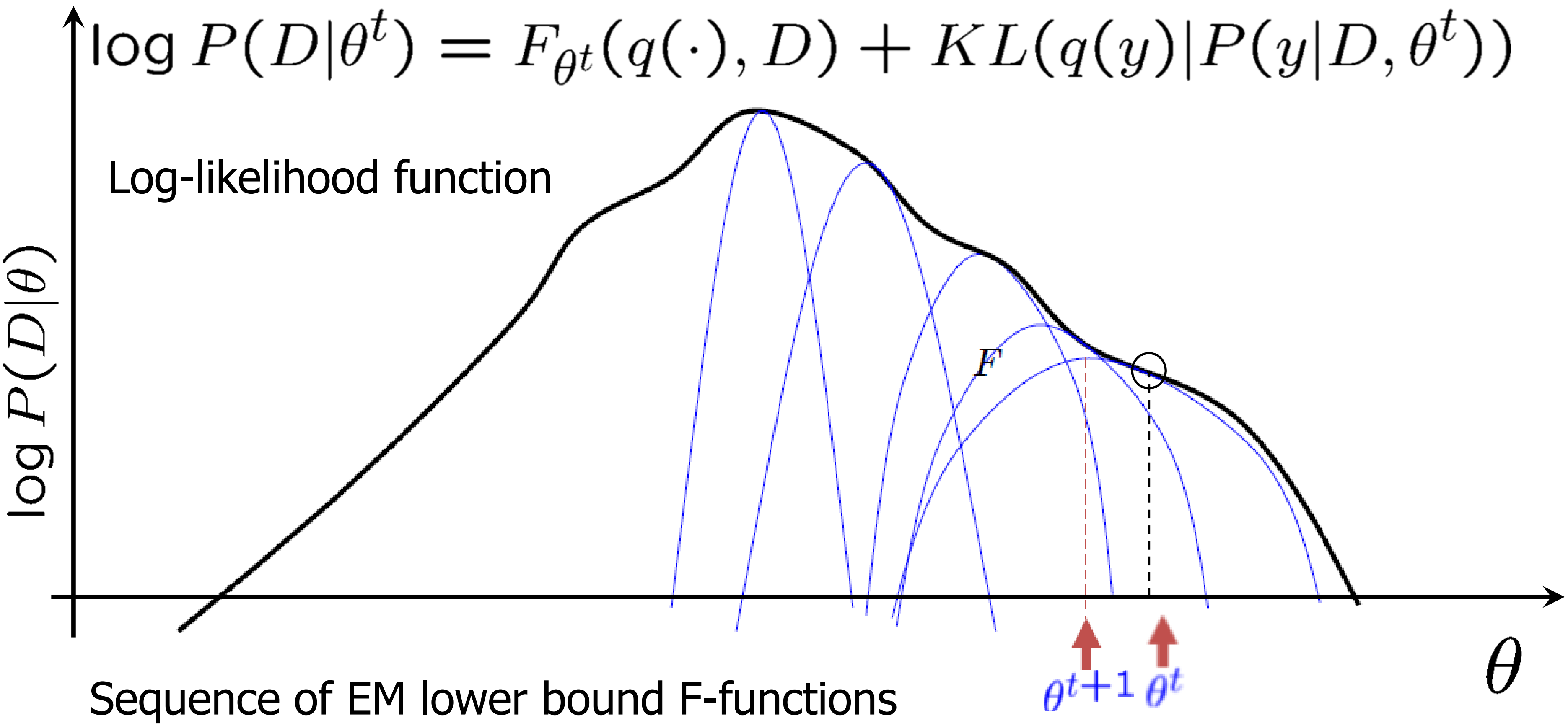
M Step:

$$\theta^t = \arg\max_{\theta} Q(\theta|\theta^{t-1})$$

During the EM algorithm the marginal likelihood is not decreasing!

$$P(D|\theta^t) \leq P(D|\theta^{t+1})$$

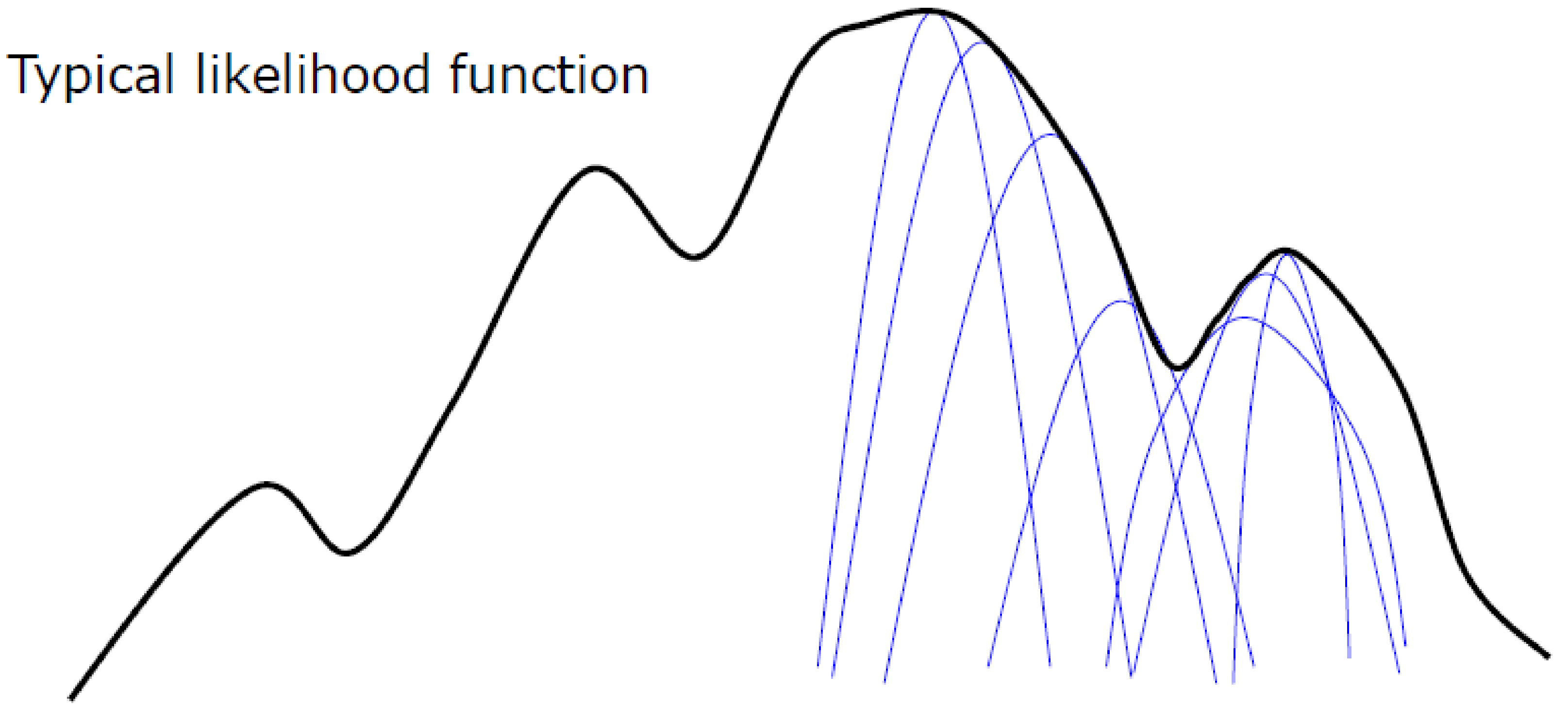
Convergence of EM



EM: (E) In a given θ_t set $q()$ such a way that the $KL = 0$ and F touches $\log P(D|\theta^t)$. (M) Maximise the lower bound F to get θ_{t+1}.

EM monotonically converges to a local maximum of likelihood !

Convergence of EM



Different sequence of EM lower bound F-functions depending on initialization

Use multiple, randomized initializations in practice

EM for spherical, same variance GMMs

For simplicity, assume that $\theta = [\mu_1, \ldots, \mu_K, \sigma^2, \pi_1, \ldots, \pi_K]$, and σ^2 and $\pi_1, \ldots, \pi_K$ are known, so we don't need to estimate them.
We only need to estimate $\mu_1, \ldots \mu_K$

E-step

Compute "expected" classes of all datapoints for each class

$$P(y_j = i | x_j, \theta^{t-1}) = \frac{\exp(-\frac{1}{2\sigma^2}\|x_j - \mu_i^{t-1}\|^2)\pi_i}{\sum_{i=1}^{K} \exp(-\frac{1}{2\sigma^2}\|x_j - \mu_i^{t-1}\|^2)\pi_i}$$

In K-means "E-step" we do hard assignment. EM does soft assignment

M-step

Update $\boldsymbol{\mu}$ given our data's class membership distributions (weights)

$$\mu_i^t = \sum_{j=1}^{n} w_j x_j \quad \text{where } w_j = \frac{P(y_j{=}i|x_j,\theta^{t-1})}{\sum_{l=1}^{n} P(y_l{=}i|x_l,\theta^{t-1})}$$

Iterate.

ATTIC

K- means Algorithm Computation Complexity

- At each iteration,
  - – Computing distance between each of the n objects and the K cluster centers is $O(Kn)$.
  - – Computing cluster centers: Each object gets added once to some cluster: $O(n)$.
- Assume these two steps are each done once for ℓ iterations: $O(\ell Kn)$.