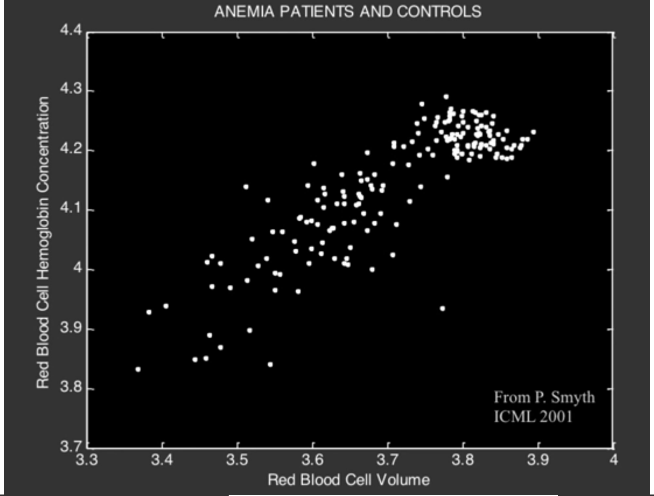
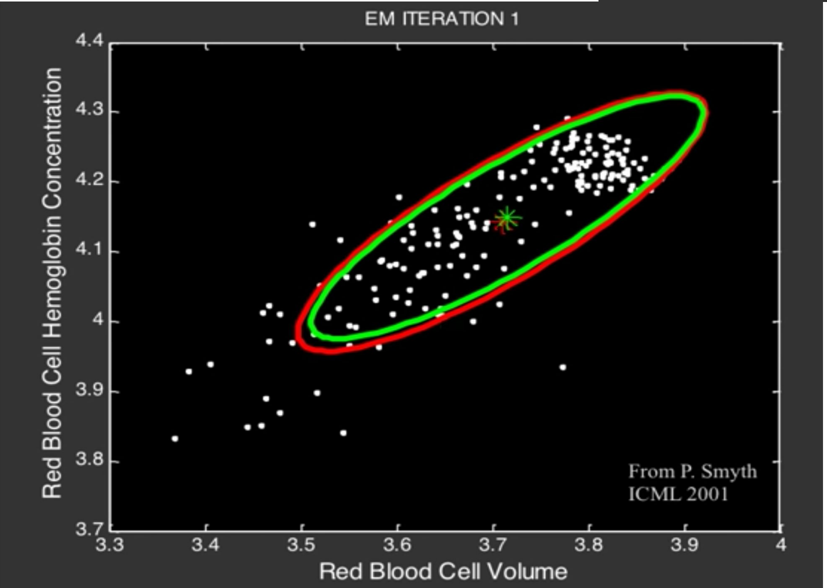
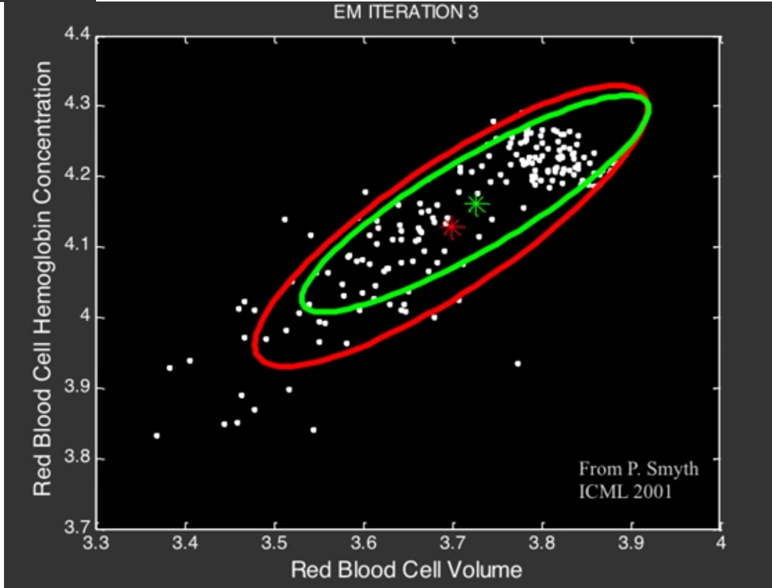
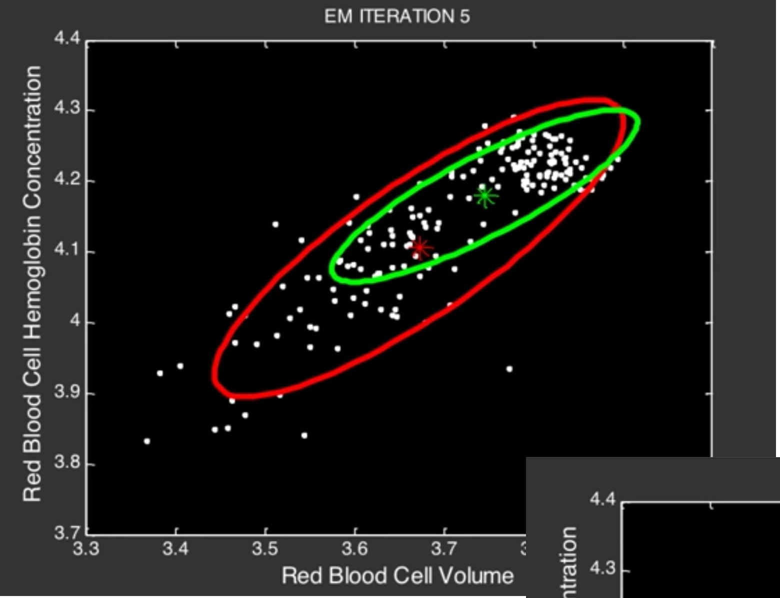
Probabilistic Learning
(Gaussian Mixture Models)

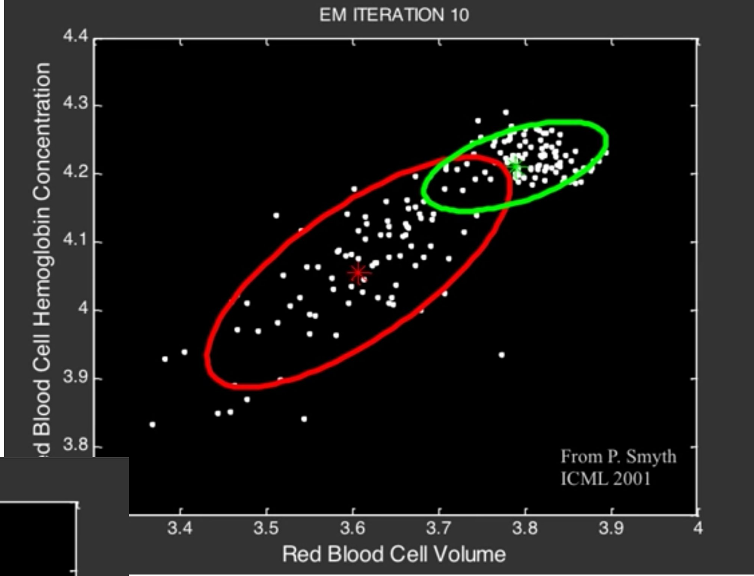
Expectation Maximization Algorithm - EM Algorithm

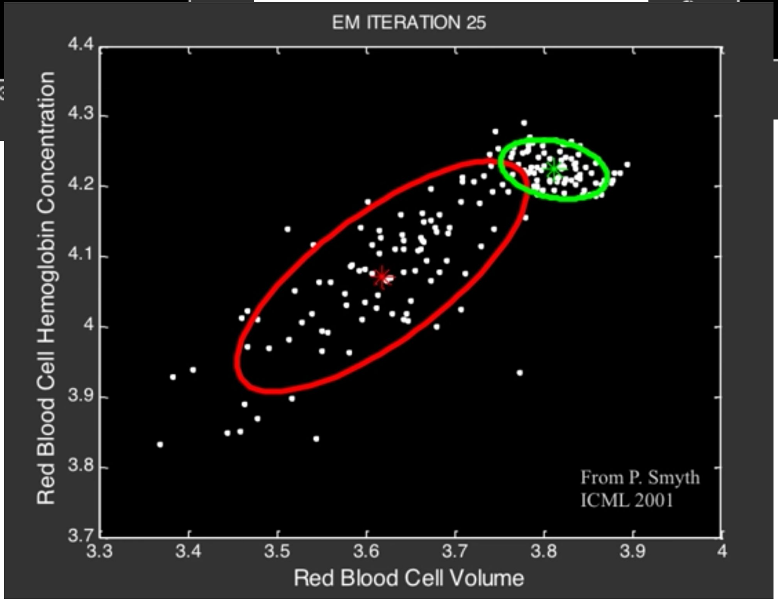

ANEMIA PATIENTS AND CONTROLS
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
From P. Smyth
ICML 2001


EM ITERATION 1
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
From P. Smyth
ICML 2001


EM ITERATION 3
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
From P. Smyth
ICML 2001


EM ITERATION 5
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume

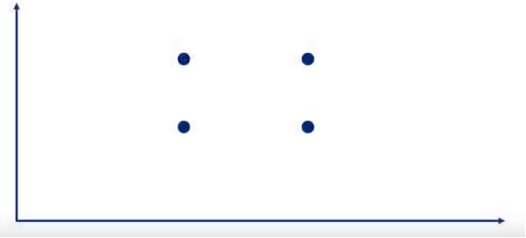

EM ITERATION 10
Red Blood Cell Volume
From P. Smyth
ICML 2001


EM ITERATION 25
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
From P. Smyth
ICML 2001

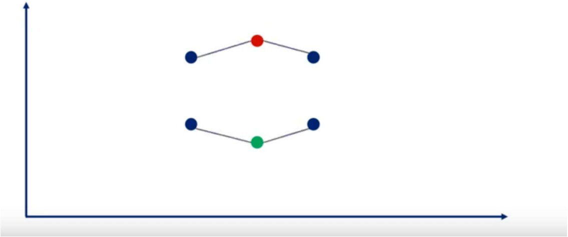
What is the difference between Clustering (K-Means) and Gaussian Mixture Models?

Problem with K-Means: Hard Assignment of Centroid Points during data initialization

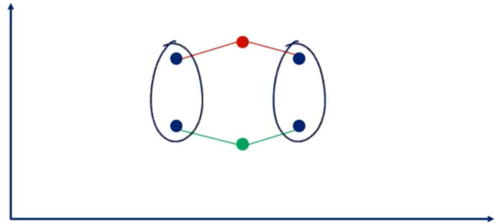
Initial Set of data points



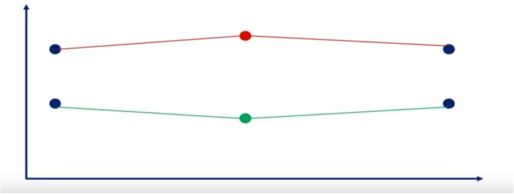
Red & **Green** are the initial centroids chosen for these data points. The centroids are aligned horizontally.

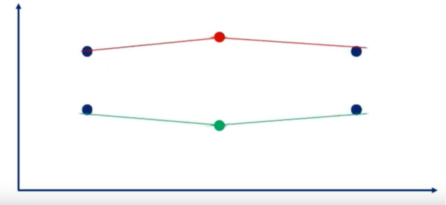


Centroids could have been chosen vertically too - resulting in the clusters shown.



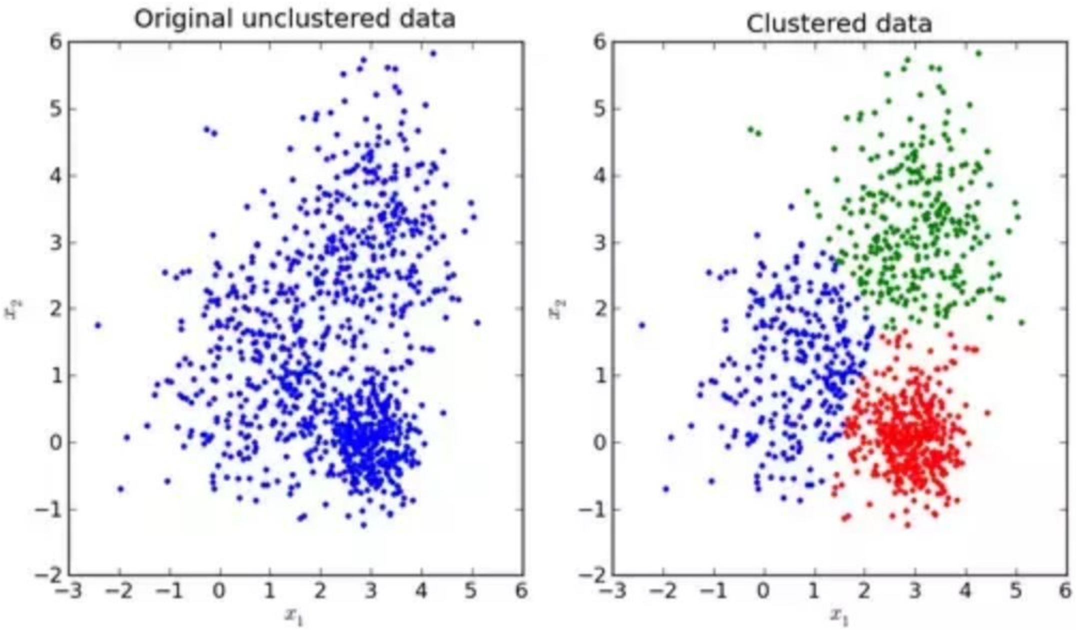
- This problem continues when the data points are more widely spread as shown below.
- This results in very useless clusters if the natural grouping was vertically all along.
- **Inference**: Choosing the centroid during data initialization is a very important step in K-means. If its inappropriately chosen the entire K-Means clustering will be useless.





Hard Assignment: We are certain that particular points belong to particular centroid, when the centroid itself could be inappropriately chosen.

Problem with K-Means: Hard Assignment of Data Points to a Centroid

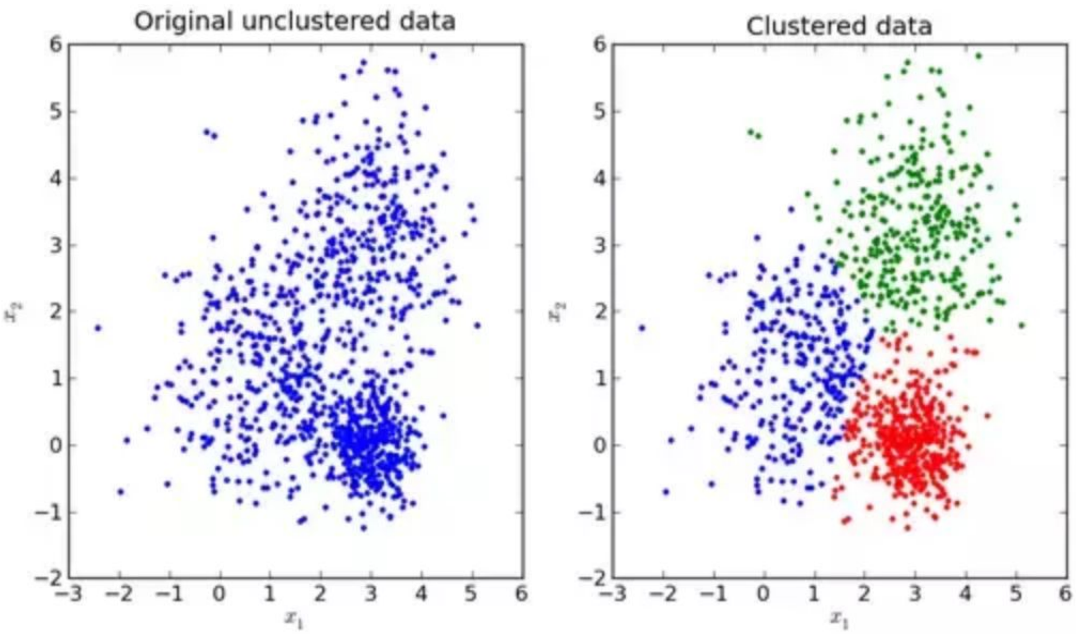


Hard Assignment: During K-Means iterations we will change the data points from the **'blue'** cluster to the **'red'** cluster or to the **'green'** cluster. But, we are **sure** it belongs to any of these three clusters.

What is we are not very sure?

Solution: Probabilistic Learning - Gaussian Mixture

Gaussian Mixture Model: Instead of Hard assigning data points to a cluster, if we are **uncertain** about the data points where they belong or to which group, we use this method. It uses **probability** of a sample to determine the feasibility of it belonging to a cluster.



Solution Soft Assignment: There is 70% chance that a data point belongs to the **'red'** cluster, but also 10% chance its in **'green'**, 20% chance it might be **'blue'**.

What is the diff between Clustering (K-Means) & Gaussian Mixture Models?

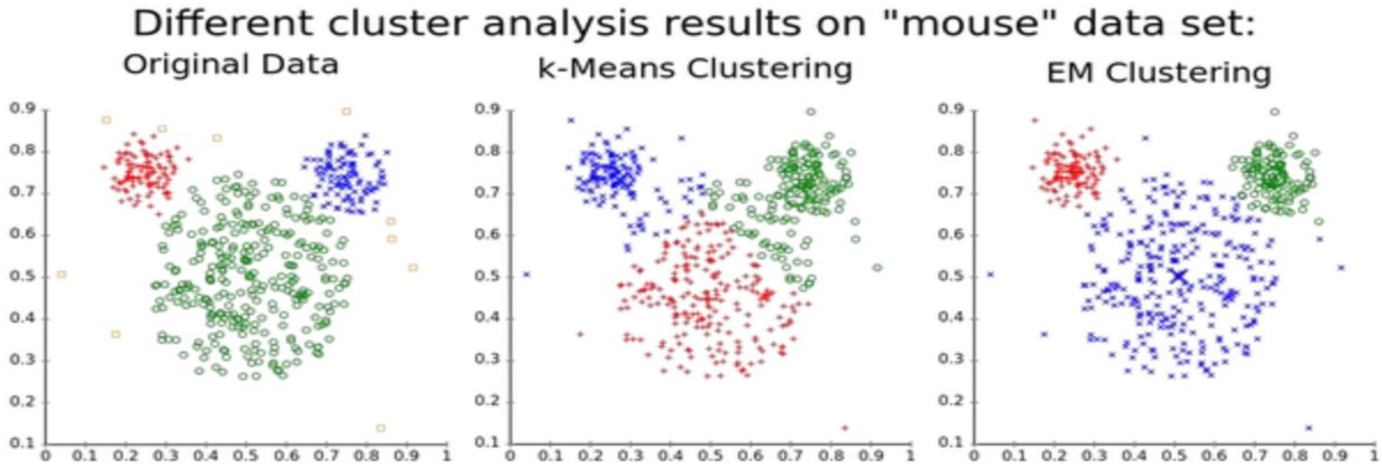
Kmeans: find k to minimize $(x - \mu_k)^2$

Gaussian Mixture (EM clustering) : find k to minimize $\frac{(x-\mu_k)^2}{\sigma^2}$

The difference (mathematically) is the denominator "σ^2", which means GM takes variance into consideration when it calculates the measurement.

Kmeans only calculates conventional Euclidean distance.

In other words, Kmeans calculate distance, while GM calculates "weighted" distance.



Why do we need to do Gaussian Mixture Models?

The reason why we need to do Gaussian Mixture Model:

In the K-mean Clustering, we assume hard-assignment. No probability of data sample for each cluster is provided, and the data distribution is unknown. So Now if we want to know:

- What is the probability that a point x is in cluster m?
- What is the shape of each cluster?

We should use Probabilistic clustering and Gaussian mixture model is the most popular one.

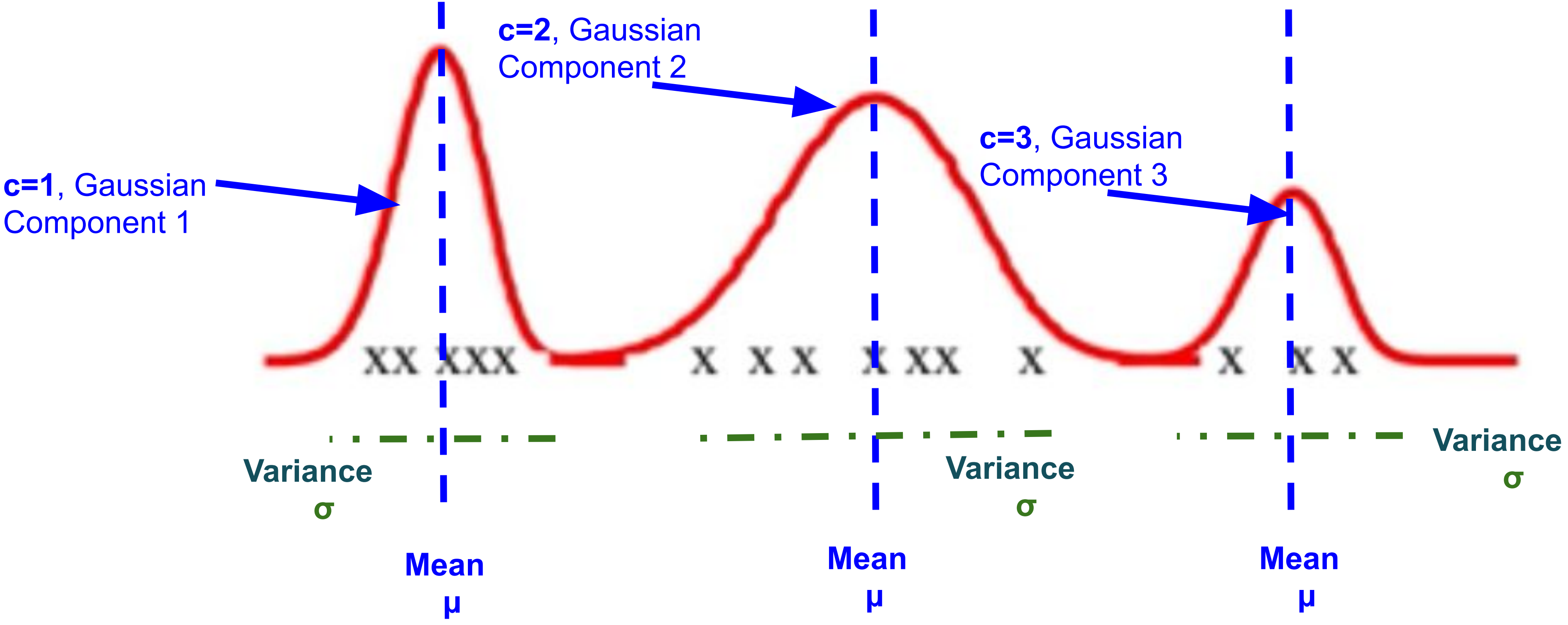
Gaussian Mixture Models : The Technique

The Idea: Instead of treating the data as a bunch of points assume that they are all generated by sampling a continuous function. This function is called a generative model. One generative model is a mixture of Gaussian (MOG)

Assumption

- Data distribution is usually normal distribution. So that, by combining several Gaussian models, we can approximate any continuous density distribution.

The Idea: For every cluster, we will try and form a Gaussian distribution that explains what that cluster looks like. Similar to K-means, we will calculate the **mean** of a distribution. Additionally, we will also calculate its **Variance**, so that we can figure out the **Gaussian distribution**.



A Gaussian Distribution is defined by **Mean** **μ** **Variance** **σ**

Probability of a data point belonging to a particular distribution of data points, is determined by finding the **Joint Probability** of the data point belonging to each distribution (Gaussian Component).

Mixtures of Gaussians

Symbol for variance σ_c or Σ_c

- Start with parameters describing each cluster
- Mean μ_c, variance σ_c, "size" π_c Mixing coefficient
- Probability distribution: $p(x) = \sum_c \pi_c \, \mathcal{N}(x \; ; \; \mu_c, \sigma_c)$

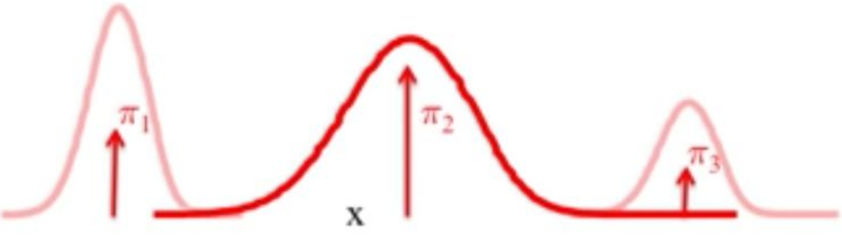
- Equivalent "latent variable" form:

$$p(z = c) = \pi_c$$ Select a mixture component with probability π

$$p(x|z = c) = \mathcal{N}(x \; ; \; \mu_c, \sigma_c)$$ Sample from that component's Gaussian

"Latent assignment" z:
we observe x, but z is hidden

p(x) = marginal over x



EM Algorithm: E-step

- Start with clusters: Mean μ_c, Covariance Σ_c, "size" π_c

- E-step ("Expectation")
  - For each datum (example) x_i,
  - Compute "r_{ic}", the probability that it belongs to cluster c
    - Compute its probability under model c
    - Normalize to sum to one (over clusters c)

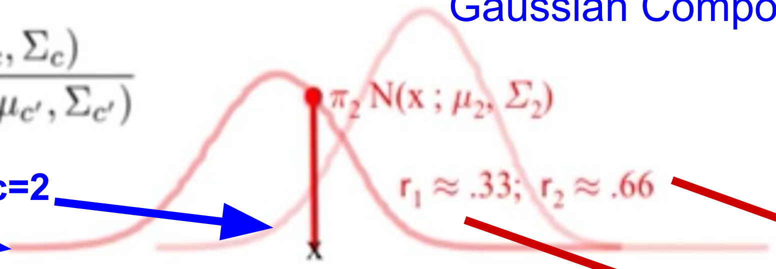
r_{ic} is a **Matrix** of number of data (**'m')** and number of clusters(**'k'**). That is **m x k** matrix.

Its value sums to one, over the index **'c',** the Gaussian Component.

$$r_{ic} = \frac{\pi_c \mathcal{N}(x_i \; ; \; \mu_c, \Sigma_c)}{\sum_{c'} \pi_{c'} \mathcal{N}(x_i \; ; \; \mu_{c'}, \Sigma_{c'})}$$

c=1, Gaussian Component 1

c=2



Gaussian Distribution Component 2 (c=2) has **66%** responsibility for the datapoint 'x'

Gaussian Distribution Component 1 (c=1) has **33%** responsibility for the datapoint 'x'

  - If x_i is very likely under the c^{th} Gaussian, it gets high weight
  - Denominator just makes r's sum to one

EM Algorithm: M-step

- Start with assignment probabilities r_{ic}
- Update parameters: mean μ_c, Covariance Σ_c, "size" π_c

- M-step ("Maximization")
  - For each cluster (Gaussian) z = c,
  - Update its parameters using the (weighted) data points

$$m_c = \sum_i r_{ic}$$

Total responsibility allocated to cluster c

Mixing coefficient

$$\pi_c = \frac{m_c}{m}$$

Fraction of total assigned to cluster c

number of data (**'m'**)

$$\mu_c = \frac{1}{m_c} \sum_i r_{ic} x^{(i)}$$

Weighted mean of assigned data

$$\Sigma_c = \frac{1}{m_c} \sum_i r_{ic} (x^{(i)} - \mu_c)^T (x^{(i)} - \mu_c)$$

Weighted covariance of assigned data
(use new weighted means here)

EM Algorithm: M-step

- Start with assignment probabilities r_{ic}
- Update parameters: mean μ_c, Covariance Σ_c, "size" π_c

- M-step ("Maximization")
  - For each cluster (Gaussian) z = c,
  - Update its parameters using the (weighted) data points

$$m_c = \sum_i r_{ic}$$ Total responsibility allocated to cluster c

$$\pi_c = \frac{m_c}{m}$$ Fraction of total assigned to cluster c

$$\mu_c = \frac{1}{m_c} \sum_i r_{ic} x^{(i)} \qquad \Sigma_c = \frac{1}{m_c} \sum_i r_{ic} (x^{(i)} - \mu_c)^T (x^{(i)} - \mu_c)$$

Weighted mean of assigned data

Weighted covariance of assigned data (use new weighted means here)

The General Expectation-Maximisation (EM) Algorithm

- Initialisation
  - guess parameters
- Repeat until convergence:
  - (E-step) compute the expectation
  - (M-step) estimate the new parameters

General EM Algorithm : What to calculate?

- Initialization of Guess Parameters π_c , Σ_c , μ_c
- E-Step : Calculate r_{ic}
- M-Step: Estimate new values of π_c , Σ_c , μ_c
- Substitute in E-Step
- Repeat until convergence

General EM Algorithm : How to calculate?

$$r_{ic} = \frac{\pi_c \mathcal{N}(x_i \; ; \; \mu_c, \Sigma_c)}{\sum_{c'} \pi_{c'} \mathcal{N}(x_i \; ; \; \mu_{c'}, \Sigma_{c'})}$$

$$m_c = \sum_i r_{ic}$$

$$\pi_c = \frac{m_c}{m}$$

$$\mu_c = \frac{1}{m_c} \sum_i r_{ic} x^{(i)}$$

$$\Sigma_c = \frac{1}{m_c} \sum_i r_{ic} (x^{(i)} - \mu_c)^T (x^{(i)} - \mu_c)$$

Expectation-Maximization

- Each step increases the log-likelihood of our model

$$\log p(\underline{X}) = \sum_i \log \left[\sum_c \pi_c \, \mathcal{N}(x_i \; ; \; \mu_c, \Sigma_c) \right]$$

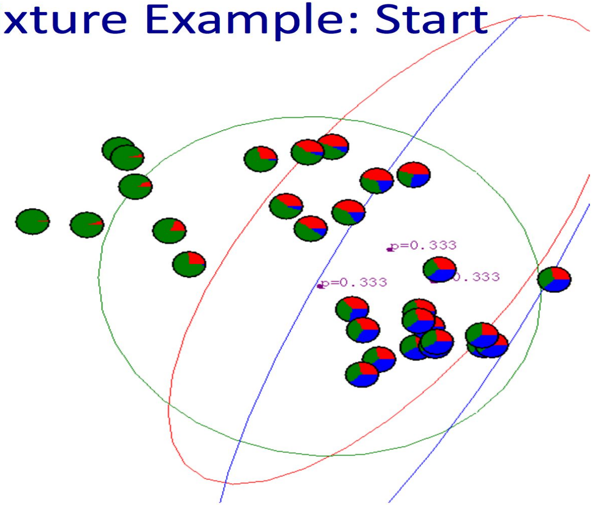
(we won't derive this here, though)

- Iterate until convergence
  - Convergence guaranteed – another ascent method
  - Local optima: initialization often important

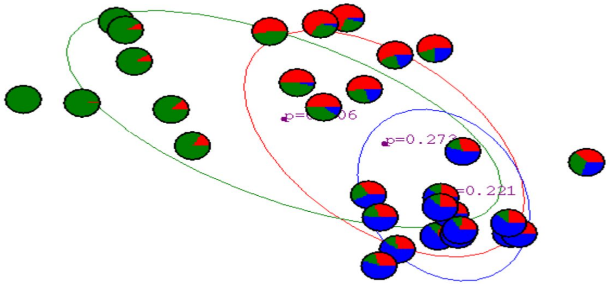
- What should we do
  - If we want to choose a single cluster for an "answer"?
  - With new data we didn't see during training?

- Choosing the number of clusters
  - Can use penalized likelihood of training data (like k-means
  - True probability model: can use log-likelihood of test data, log p(x')

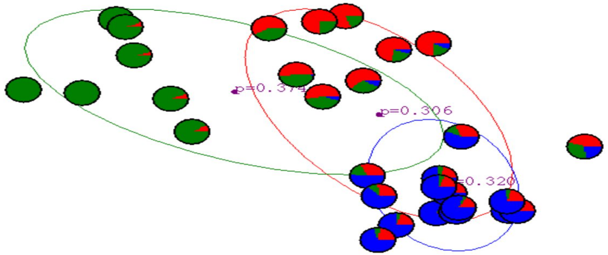
Gaussian Mixture Example: Start



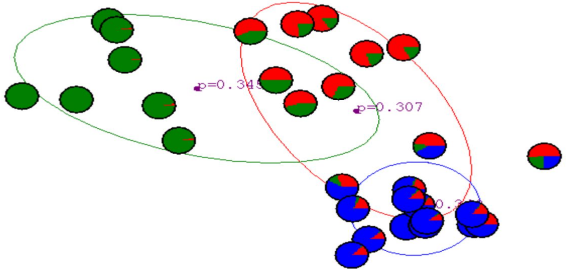
After first iteration



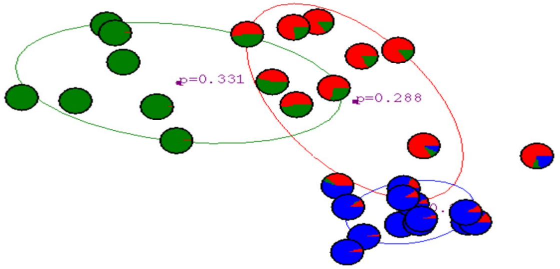
After 2nd iteration



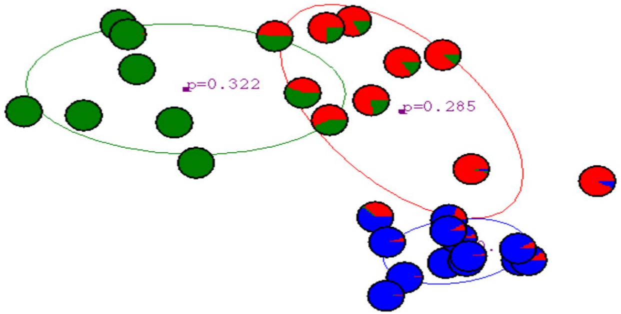
After 3rd iteration



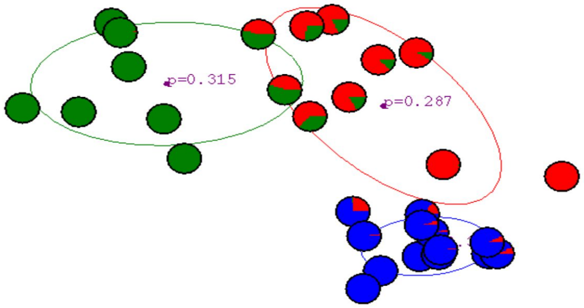
After 4th iteration



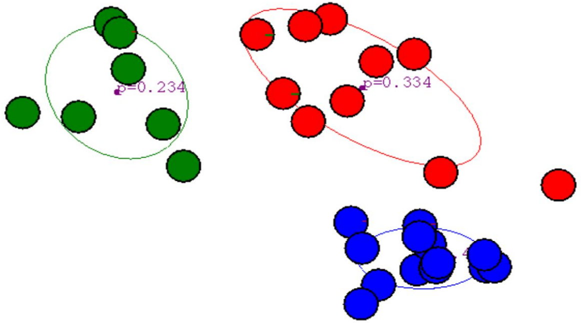
After 5th iteration



After 6th iteration



After 20th iteration



WHAT IS HAPPENING TO THE GAUSSIAN DISTRIBUTION?

EM: 1-d example

Actual Equations for Calculations:

$$P(x_i \mid b) = \frac{1}{\sqrt{2\pi\sigma_b^2}} \exp\left(-\frac{(x_i - \mu_b)^2}{2\sigma_b^2}\right)$$

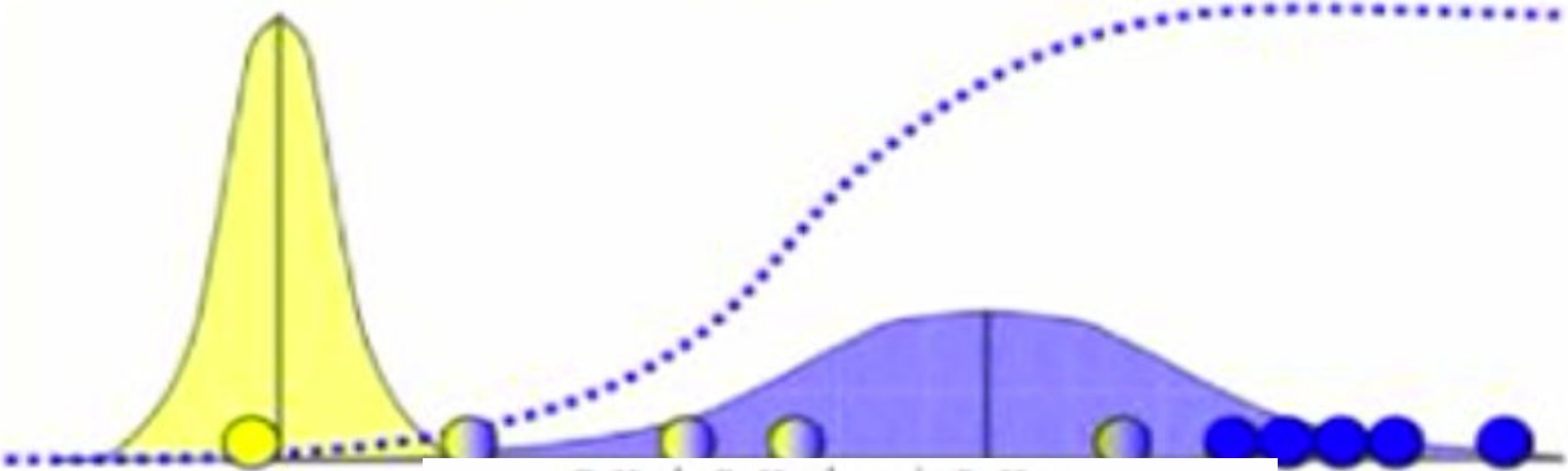
$$b_i = P(b \mid x_i) = \frac{P(x_i \mid b)P(b)}{P(x_i \mid b)P(b) + P(x_i \mid a)P(a)}$$

$$a_i = P(a \mid x_i) = 1 - b_i$$

$$\mu_b = \frac{b_1 x_1 + b_2 x_2 + \ldots + b_n x_n}{b_1 + b_2 + \ldots + b_n}$$

$$\sigma_b^2 = \frac{b_1 (x_1 - \mu_1)^2 + \ldots + b_n (x_n - \mu_n)^2}{b_1 + b_2 + \ldots + b_n}$$

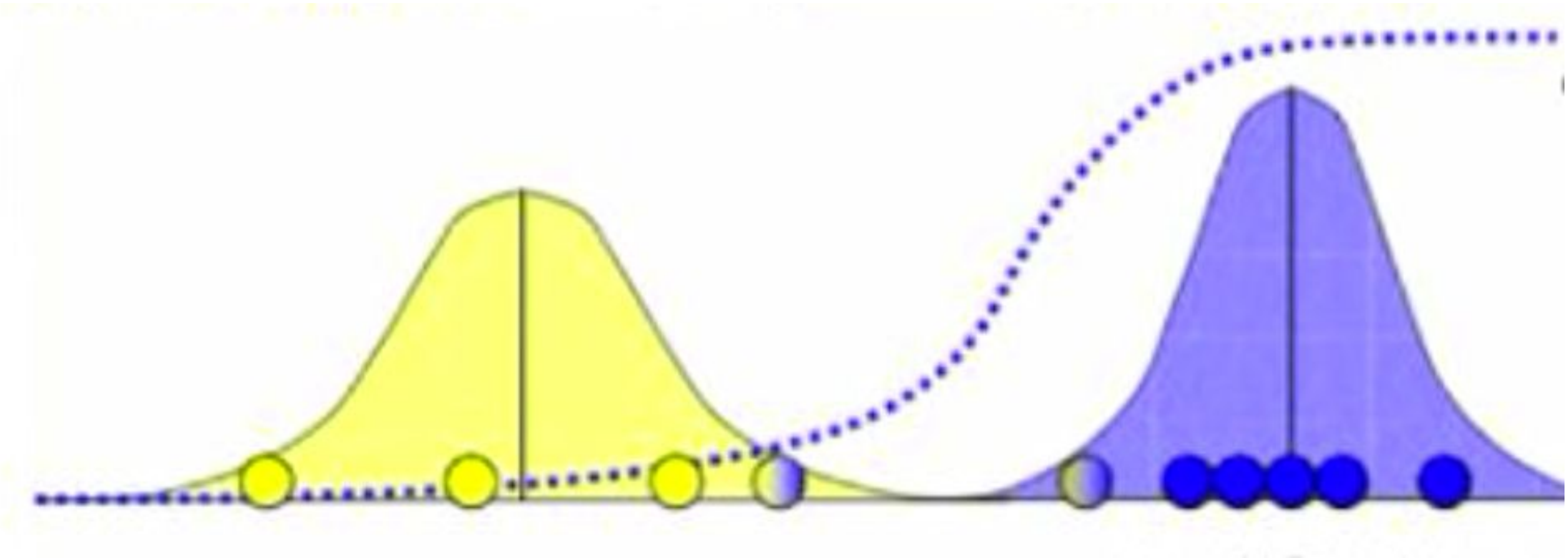
WHAT IS HAPPENING TO THE GAUSSIAN DISTRIBUTION?



$$\mu_a = \frac{a_1x_1 + a_2x_2 + \ldots + a_nx_n}{a_1 + a_2 + \ldots + a_n}$$

$$\sigma_a^2 = \frac{a_1(x_1 - \mu_1)^2 + \ldots + a_n(x_n - \mu_n)^2}{a_1 + a_2 + \ldots + a_n}$$

Final Gaussian Distribution



could also estimate priors:

$P(b) = (b_1 + b_2 + \ldots b_n) / n$

$P(a) = 1 - P(b)$

Pros and Cons of Gaussian Mixture Models

Advantages:

1. Does not assume clusters to be of any geometry. Works well with non-linear geometric distributions as well.

2. Does not bias the cluster sizes to have specific structures as does by K-Means (Circular).

Disadvantages:

1. Uses all the components it has access to, so initialization of clusters will be difficult when dimensionality of data is high.

2. Difficult to interpret.