Machine Learning
DSECL ZG565

Unsupervised Learning

Dr. Monali Mavani

BITS Pilani
Pilani Campus

These slides are prepared by the instructor, with grateful acknowledgement of Prof. Tom Mitchell, Prof.. Burges, Prof. Andrew Moore and many others who made their course materials freely available online

Topics to be covered

Ref: Christopher Bishop: Chapter 9

- Unsupervised learning
- Clustering
- K-means Clustering
- Gaussian Mixture Models
- EM algorithm

Unsupervised Learning

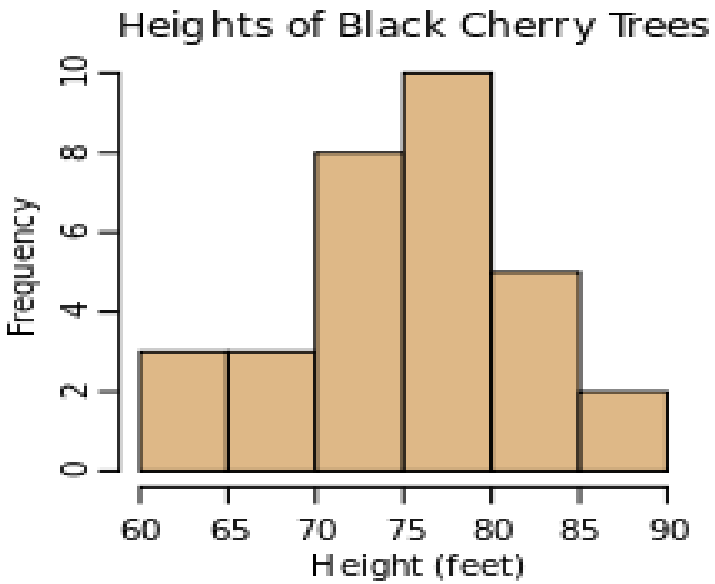
- We only use the features X, not the labels Y
- This is useful because we may not have any labels but we can still detect patterns
- For example:
  - We can detect that news articles revolve around certain topics, and group them accordingly
  - Discover a distinct set of objects appear in a given environment, even if we don't know their names, then ask humans to label each group
  - Identify health factors that correlate with a disease

What is clustering?

- Grouping items that “belong together” (i.e. have similar features)

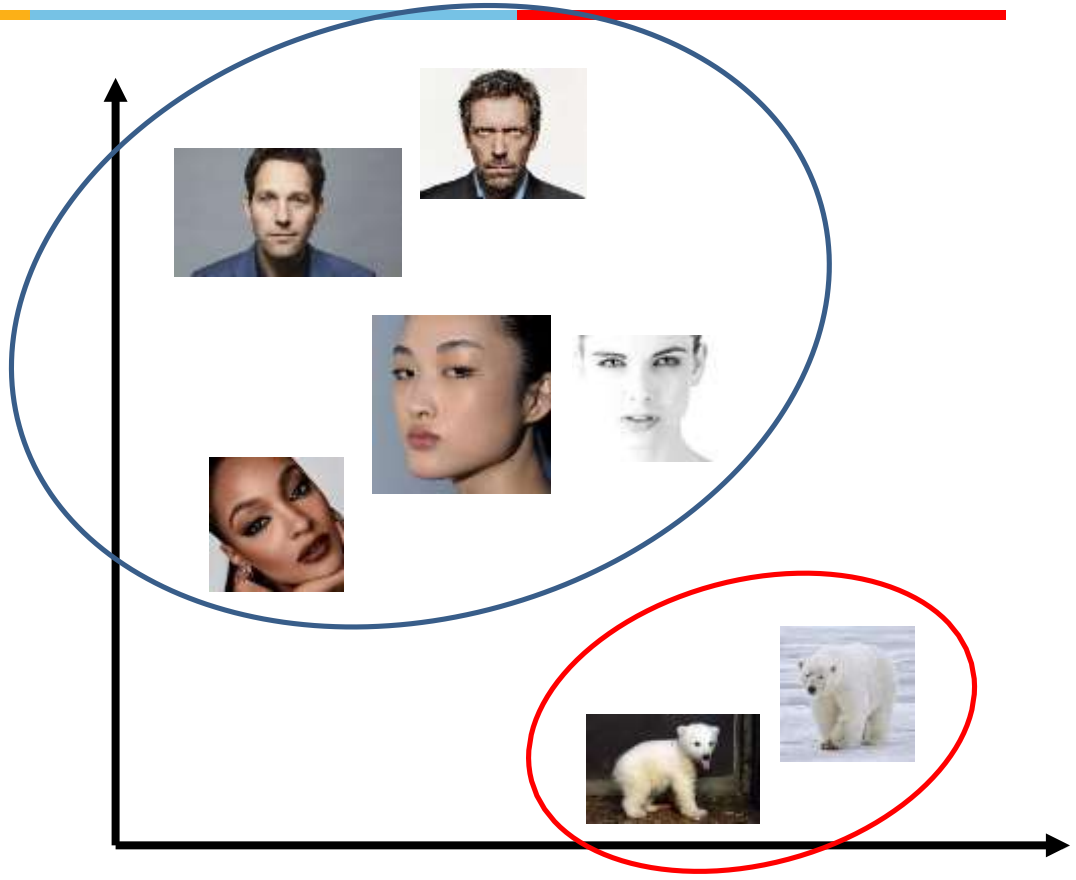
Why do we cluster?

- Counting
  - Feature histograms: by grouping similar features and counting how many of each a data sample has
- Summarizing data
  - Look at large amounts of data
  - Represent a large continuous vector with the cluster number
- Prediction
  - Data points in the same cluster may have the same labels
  - Ask a human to label the clusters



Slide credit: J. Hays, D. Hoiem

Unsupervised discovery



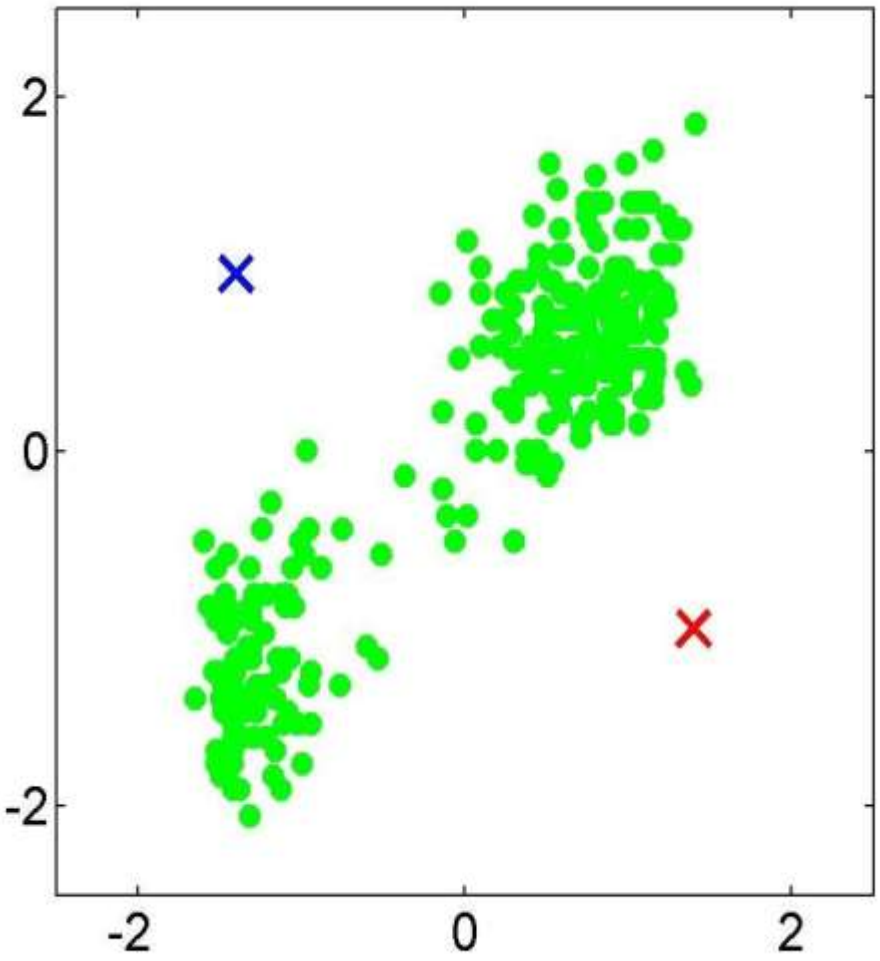
Clustering algorithms

- In depth
  - K-means (iterate between finding centers and assigning points)
  - Gaussian Mixture Models (GMMs)

K-means Algorithm

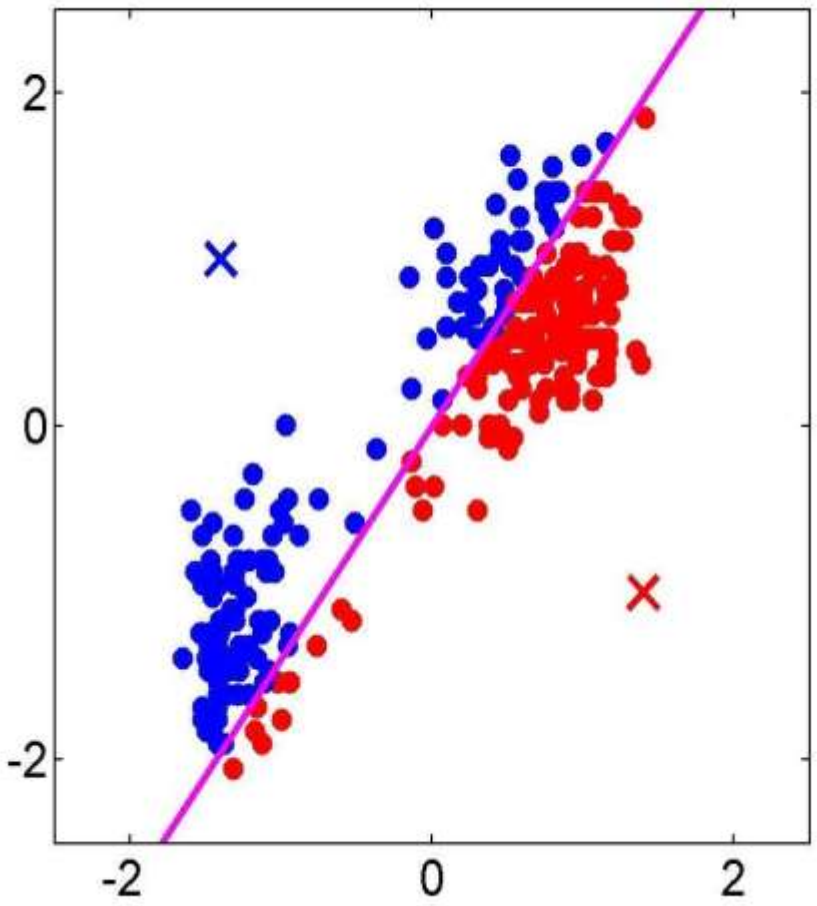
- Goal: represent a data set in terms of K clusters each of which is summarized by a prototype μ_k
- Initialize prototypes, then iterate between two phases:
  - E-step: assign each data point to nearest prototype
  - M-step: update prototypes to be the cluster means
- Simplest version is based on Euclidean distance

Example



- Pick K random points as cluster centers (means)
- Shown here for K=2
- The initial choices for centres μ_1 and μ_2 are shown by the red and blue crosses, respectively

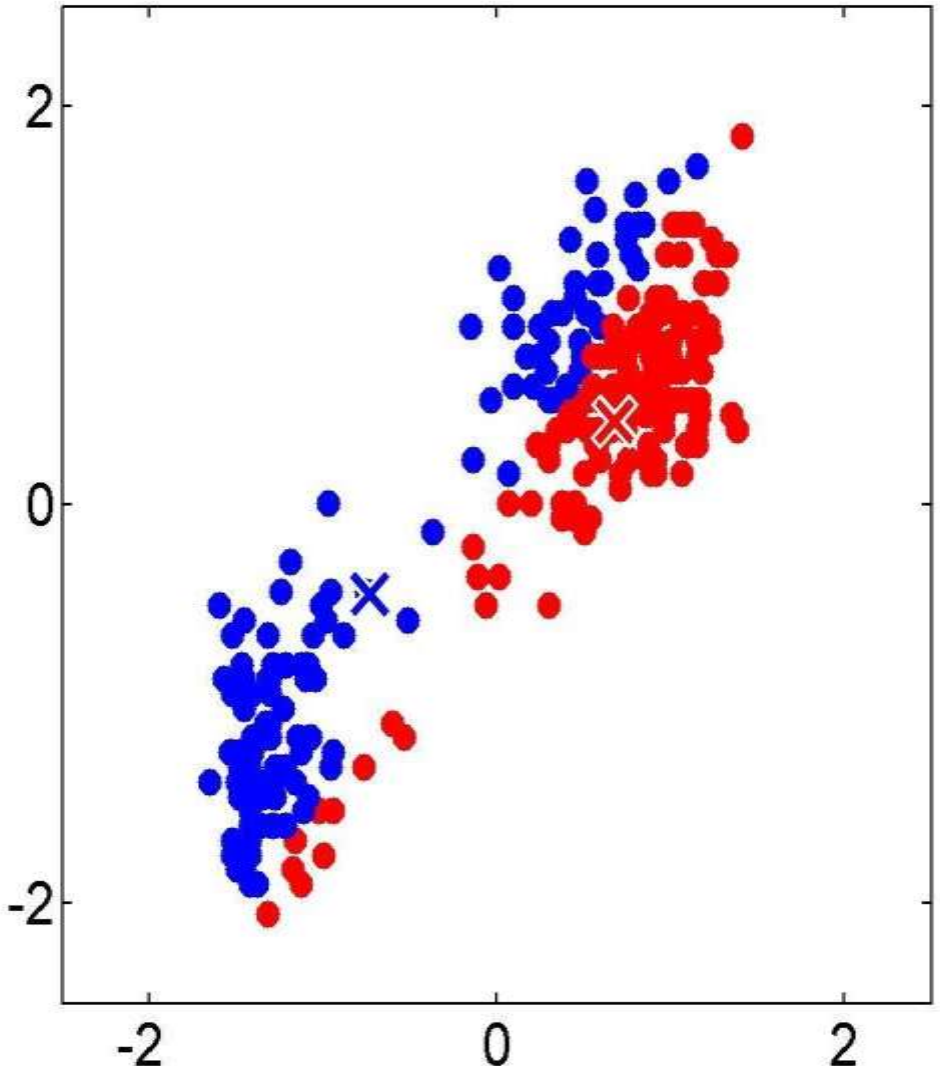
Iterative Step 1
Assign data points to closest cluster center



- In the initial E step, each data point is assigned either to the red cluster or to the blue cluster, according to which cluster centre is nearer.
- This is equivalent to classifying the points according to which side of the perpendicular bisector of the two cluster centres, shown by the magenta line, they lie on.

Iterative Step 2

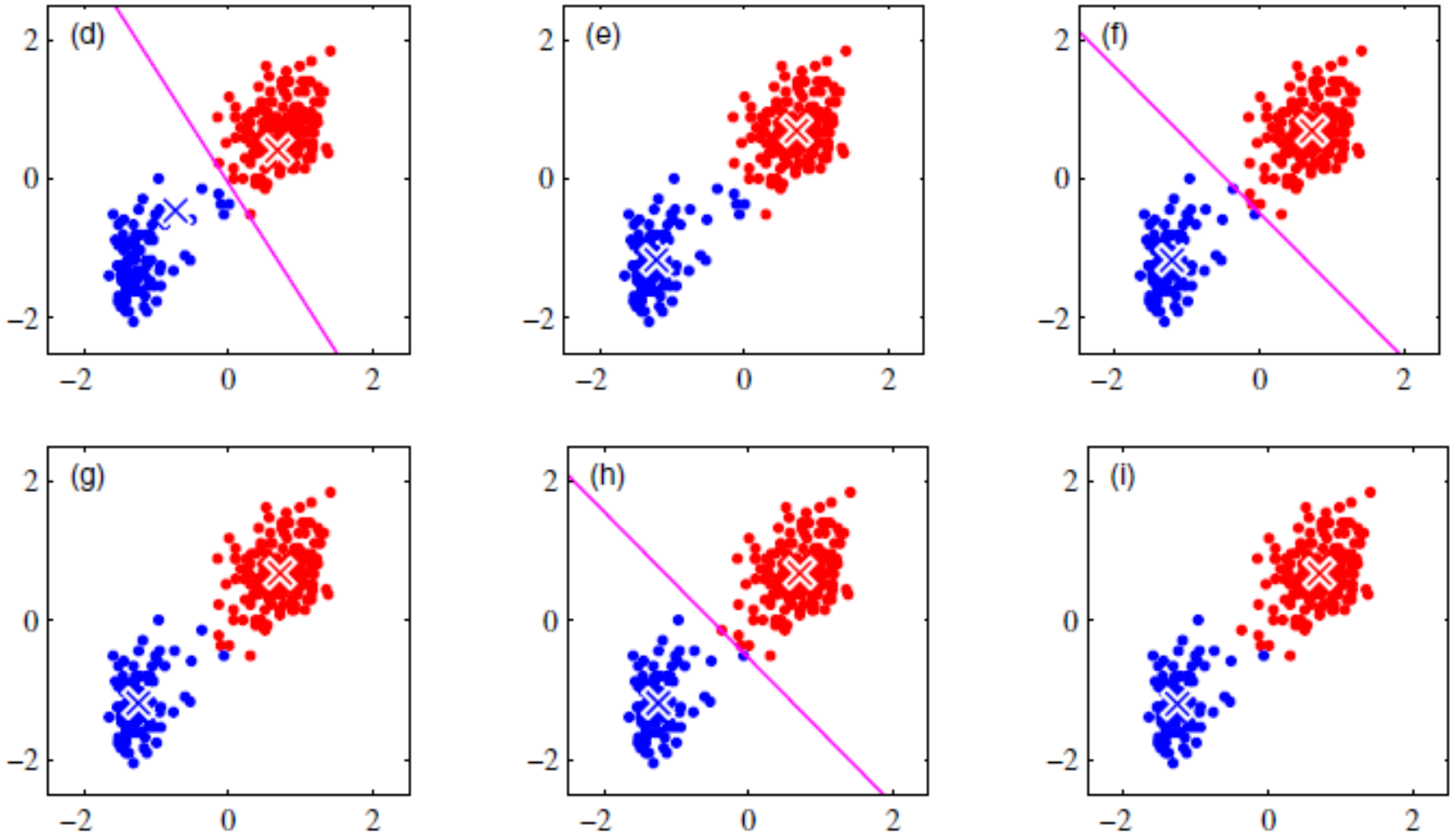
Change the cluster center to the average of the assigned points



In the M step, each cluster centre is re-computed to be the mean of the points assigned to the corresponding cluster.

Repeat until convergence

successive E and M steps through to final convergence of the algorithm.



1 of K coding mechanism

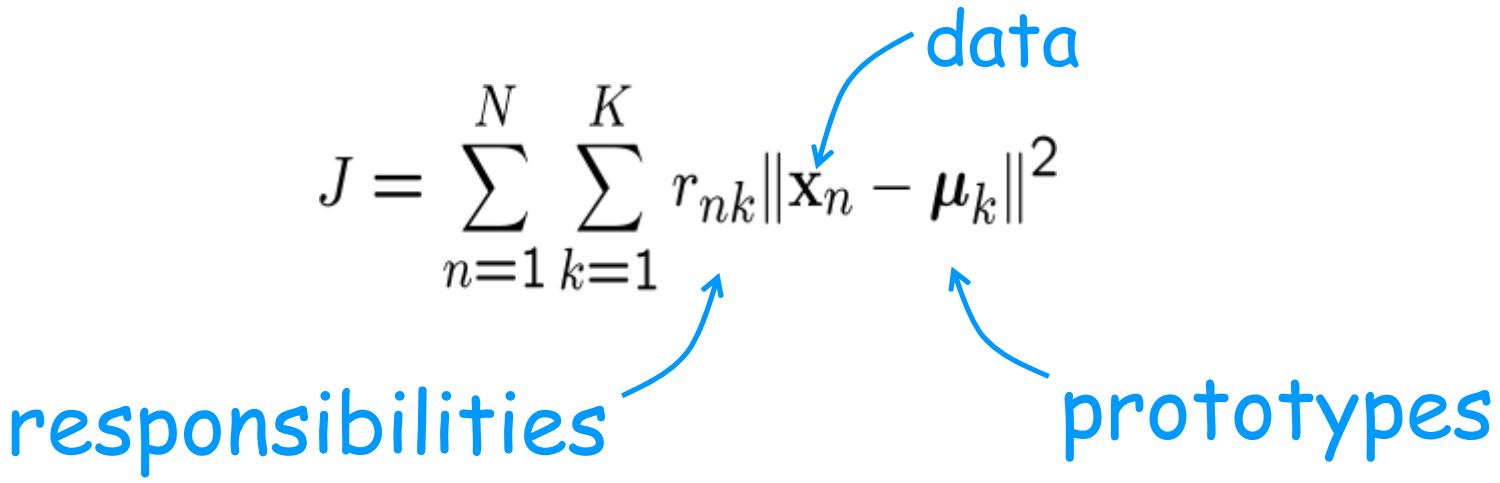
- *For each data point x_n, we introduce a set of binary indicator variables* $r_{nk} \in \{0, 1\}$
  such that $\sum_k r_{nk} = 1$

  where k =1,...,K describing which of the K clusters the data point x_n is assigned to, so that if data point x_n is assigned to cluster k then r_{nk} =1, and r_{nj} =0 for j not equal to k.

- Example: 5 data points and 3 clusters

$$(r_{nk}) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

K-means Cost Function



- **goal is to find values for the $\{r_{nk}\}$ and the $\{\mu_k\}$ so as to minimize J** *(Distortion measure)*
- relatively slow, because in each E step it is necessary to compute the Euclidean distance between every prototype vector and every data point

Minimizing the Cost Function

E-step: minimize J w.r.t r_{nk}

- choose some initial values for the $\boldsymbol{\mu}_k$. minimize J with respect to the r_{nk}, keeping the $\boldsymbol{\mu}_k$ fixed. Assign the nth data point to the closest cluster centre

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_n - \boldsymbol{\mu}_j\|^2 \\ 0 & \text{otherwise.} \end{cases}$$

M-step: minimize J w.r.t μ_k

- minimize J with respect to the $\boldsymbol{\mu}_k$, keeping r_{nk} fixed. The objective function J is a quadratic function of $\boldsymbol{\mu}_k$, and it can be minimized by setting its derivative with respect to $\boldsymbol{\mu}_k$ to zero giving

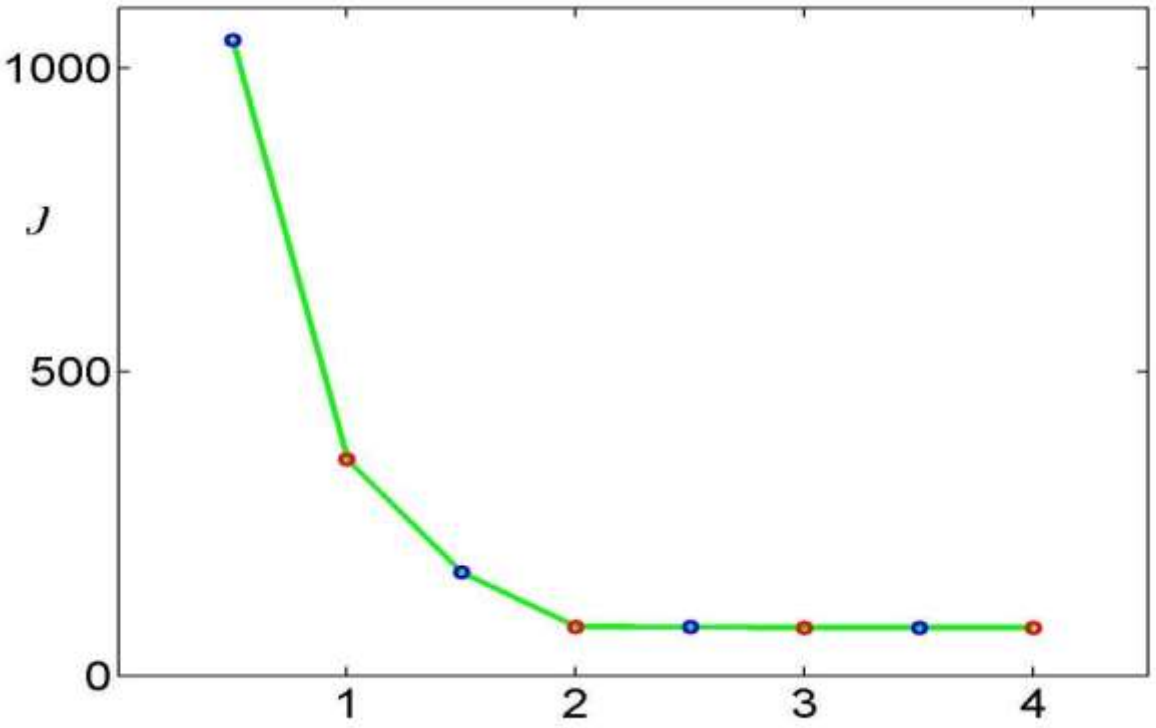
$$\boldsymbol{\mu}_k = \frac{\sum_n r_{nk}\mathbf{x}_n}{\sum_n r_{nk}}.$$

Stopping Criteria for K-Mean Clustering

- Centroids of newly formed clusters do not change
- Points remain in the same cluster
- Maximum number of iterations are reached

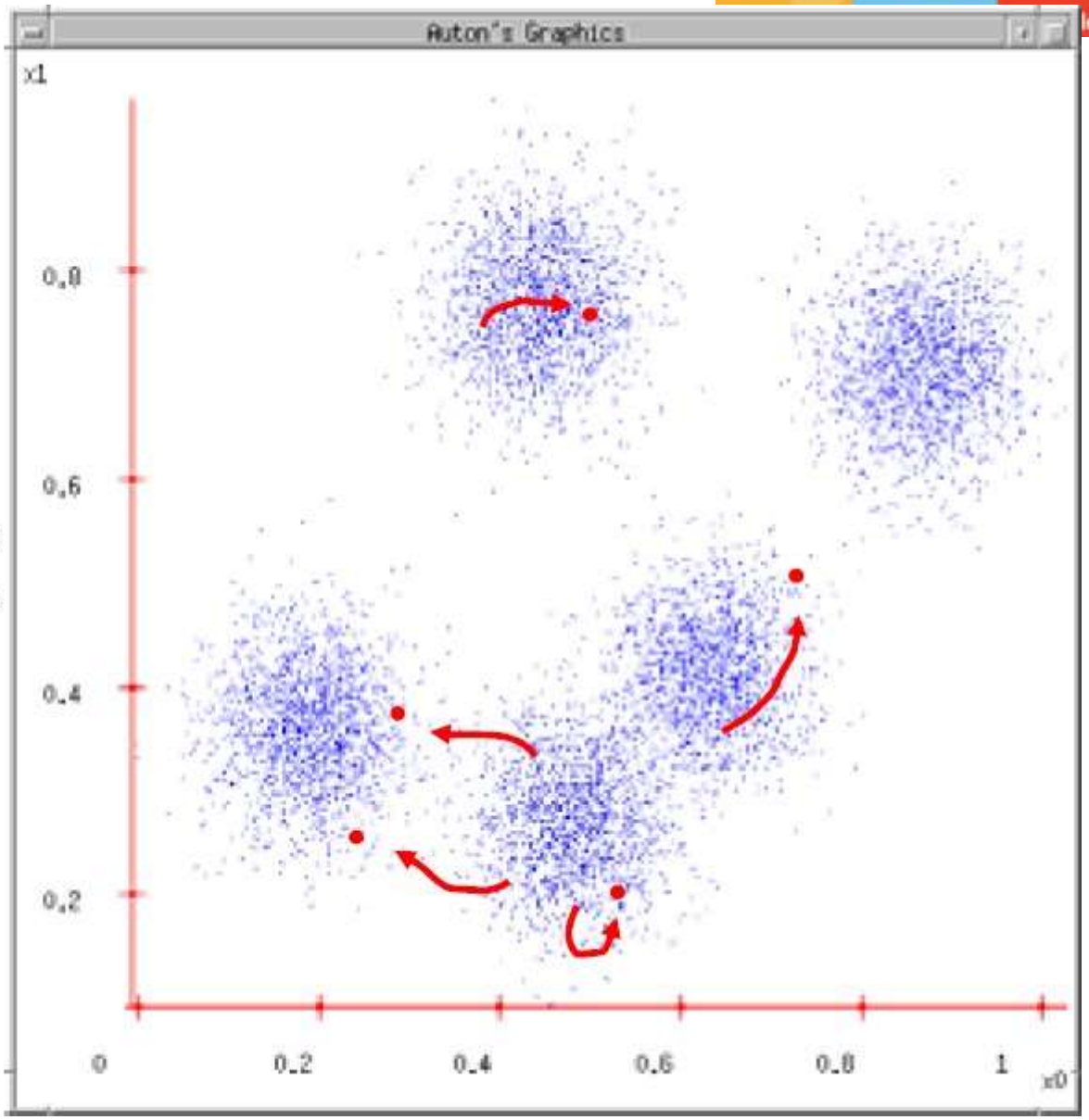
Convergence

- Each E step (blue points) and M step (red points) . The algorithm has converged after the third M step, and the final EM cycle produces no changes in either the assignments or the prototype vectors.
- Because each phase reduces the value of the objective function *J*, convergence of the algorithm is assured. However, it may converge to a local rather than global minimum of *J*



K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns…
5. …and jumps there
6. …Repeat until terminated!



Example

A_1	A_2
6.8	12.6
0.8	9.8
1.2	11.6
2.8	9.6
3.8	9.9
4.4	6.5
4.8	1.1
6.0	19.9
6.2	18.5
7.6	17.4
7.8	12.2
6.6	7.7
8.2	4.5
8.4	6.9
9.0	3.4
9.6	11.1

objects with two attributes A_1 and A_2.

k=3

Three objects are chosen at random

Example

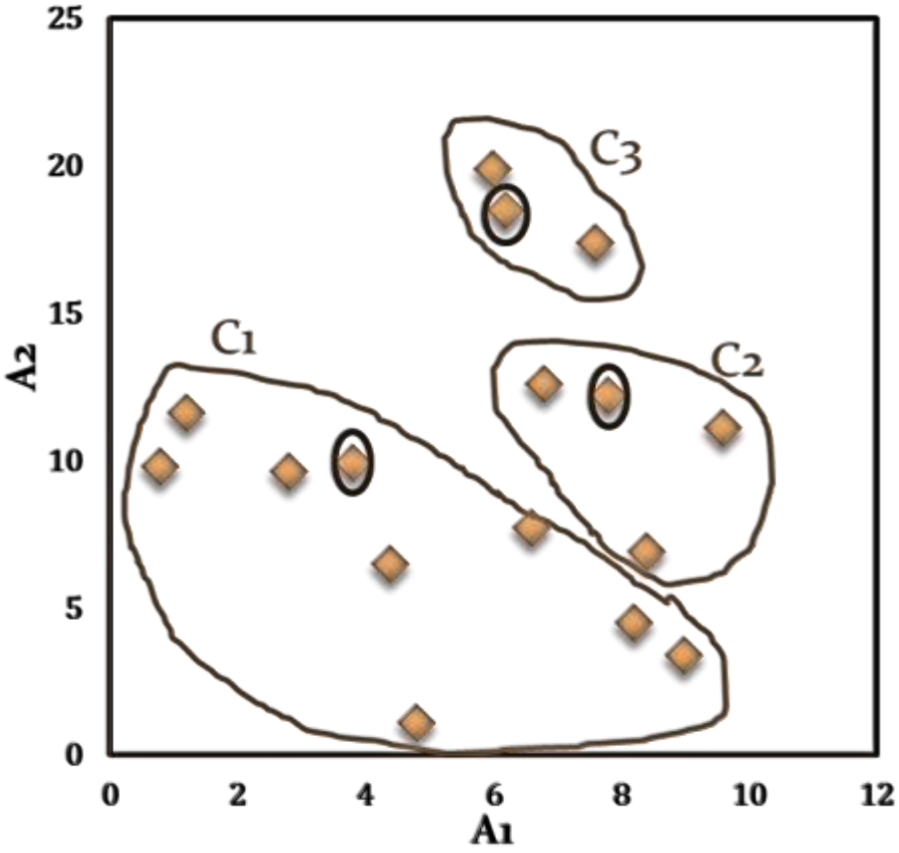
Coordinates of 3 randomly chosen Centroids

Centroid		
	A1	A2
c_1	3.8	9.9
c_2	7.8	12.2
c_3	6.2	18.5

Let d1, d2 and d3 : Euclidean distance to c1, c2 and c3 respectively..

Example

A_1	A_2	d_1	d_2	d_3	cluster
6.8	12.6	4.0	1.1	5.9	2
0.8	9.8	3.0	7.4	10.2	1
1.2	11.6	3.1	6.6	8.5	1
2.8	9.6	1.0	5.6	9.5	1
3.8	9.9	0.0	4.6	8.9	1
4.4	6.5	3.5	6.6	12.1	1
4.8	1.1	8.9	11.5	17.5	1
6.0	19.9	10.2	7.9	1.4	3
6.2	18.5	8.9	6.5	0.0	3
7.6	17.4	8.4	5.2	1.8	3
7.8	12.2	4.6	0.0	6.5	2
6.6	7.7	3.6	4.7	10.8	1
8.2	4.5	7.0	7.7	14.1	1
8.4	6.9	5.5	5.3	11.8	2
9.0	3.4	8.3	8.9	15.4	1
9.6	11.1	5.9	2.1	8.1	2



Example

new centroids

New Centroid	Revised Centroids	
	A1	A2
c_1	4.6	7.1
c_2	8.2	10.7
c_3	6.6	18.6

Initial cluster with new centroids

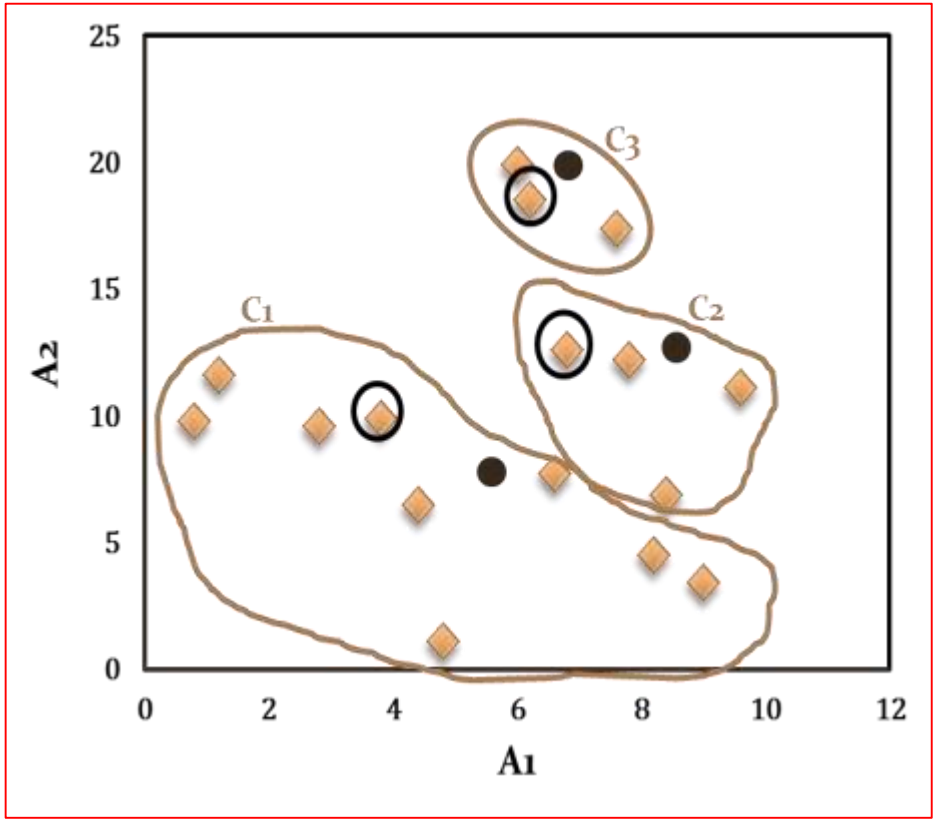
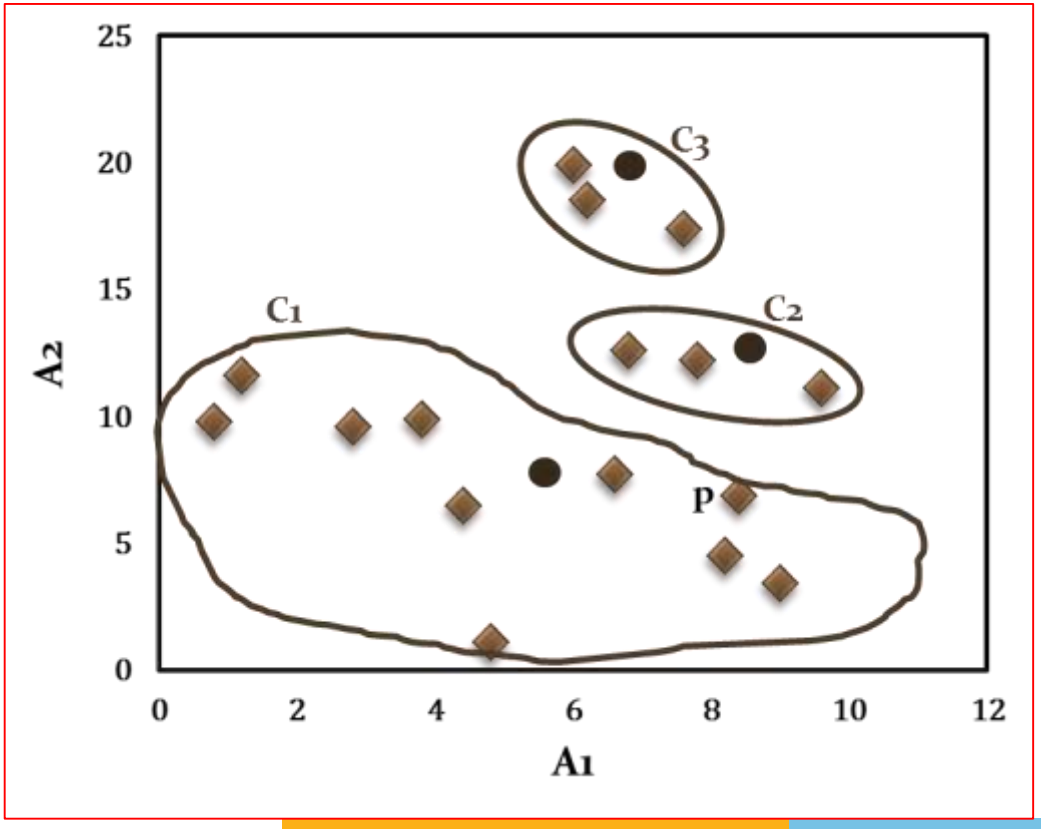


Illustration of k-Means clustering algorithms

reassign the 16 data points
Note that point p moves from cluster C2 to cluster C1.

Cluster after first iteration



Adopted from Debasis Samanta

Illustration of k-Means clustering algorithms

Cluster centres after second iteration

Centroid	Revised Centroids	
	A1	A2
c_1	5.0	7.1
c_2	8.1	12.0
c_3	6.6	18.6

Cluster after Second iteration

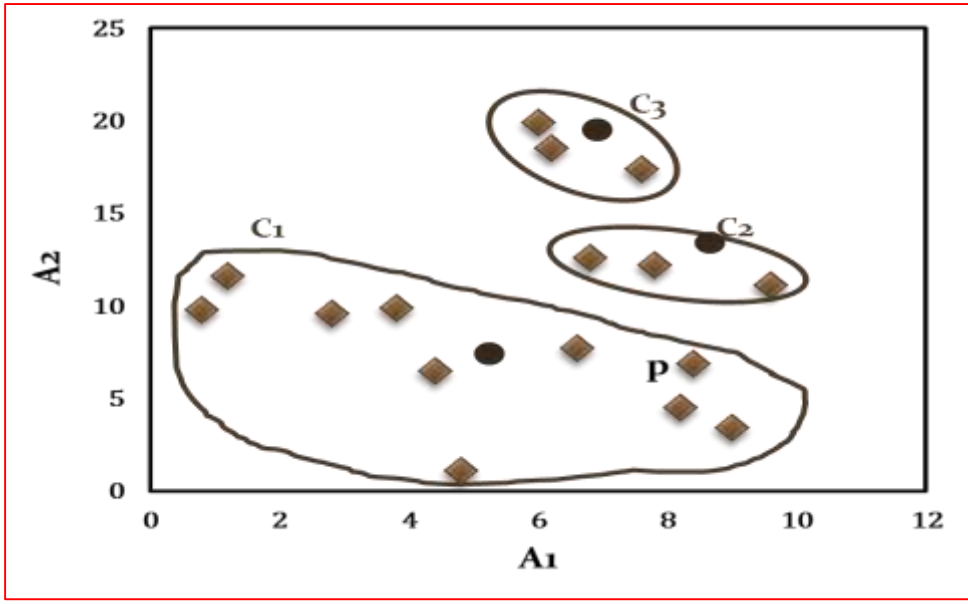
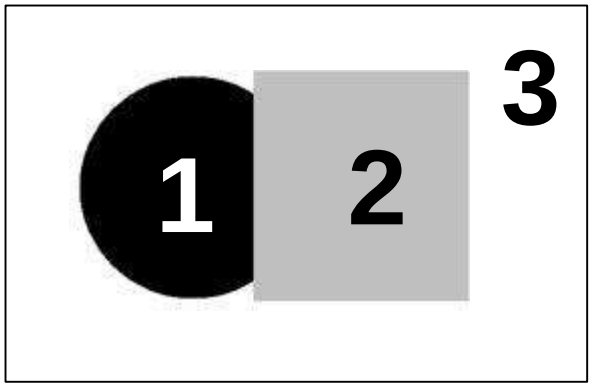
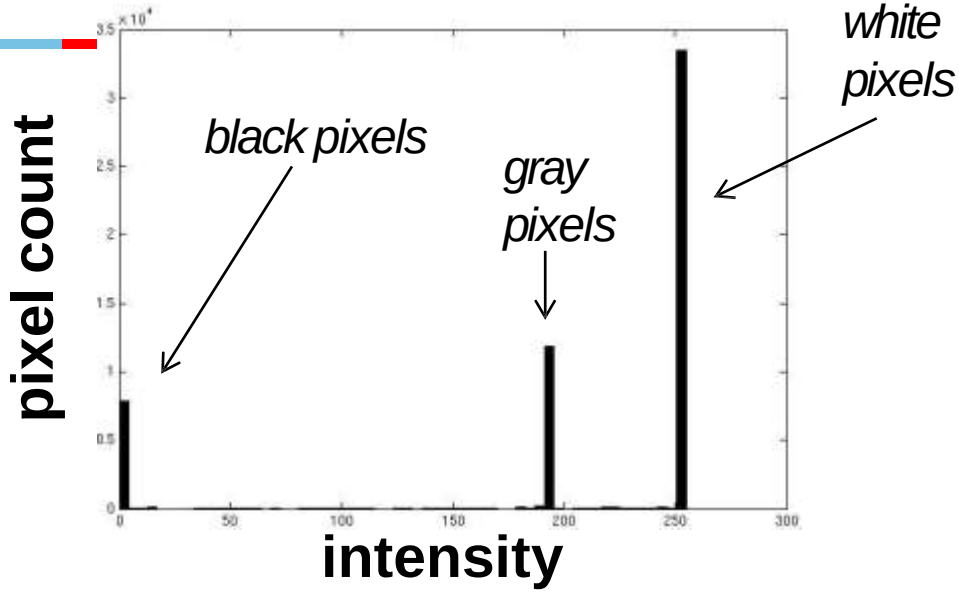


Image segmentation: toy example



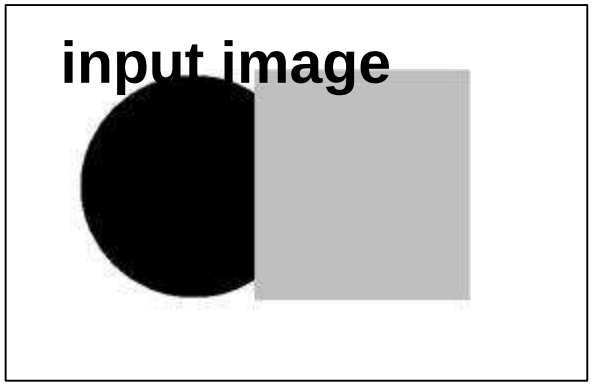
input image

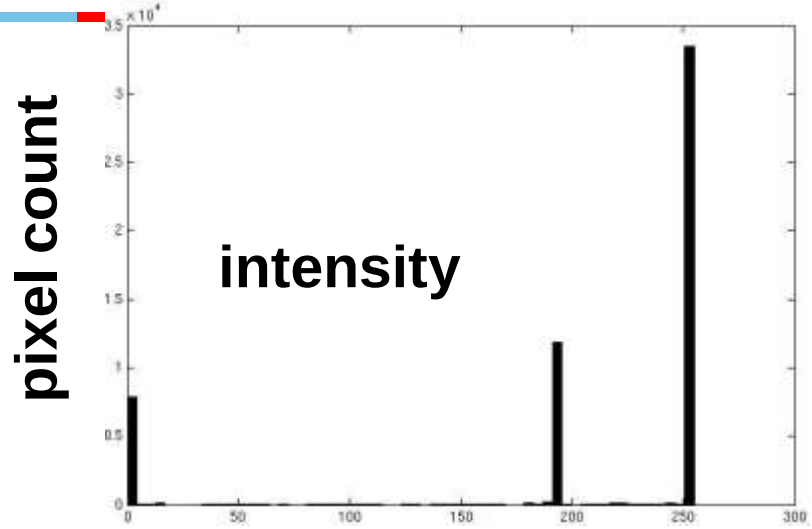


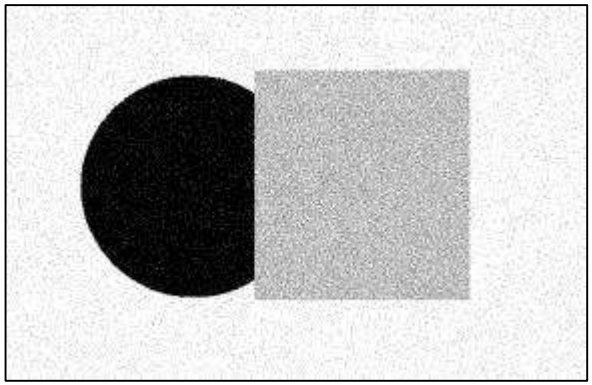
- These intensities define the three groups.
- We could label every pixel in the image according to which of these primary intensities it is.
  - i.e., *segment* the image based on the intensity feature.
- What if the image isn't quite so simple?

Source: K. Grauman

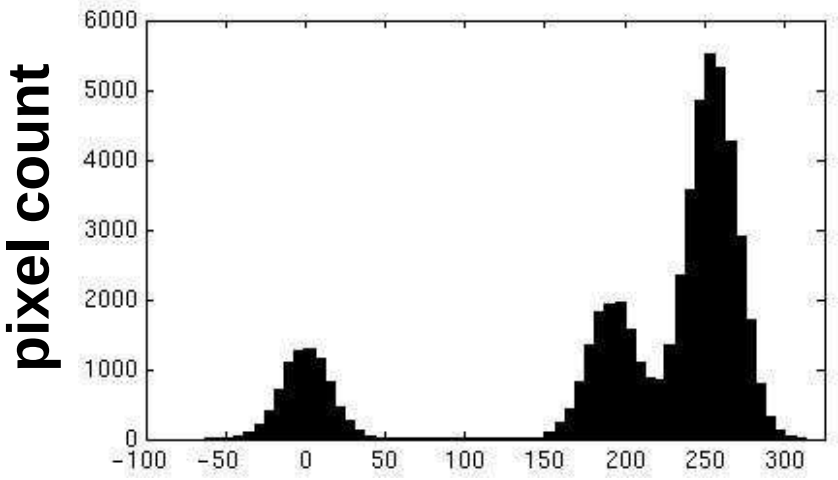
- Now how to determine the three main intensities that define our groups?
- We need to ***cluster.***



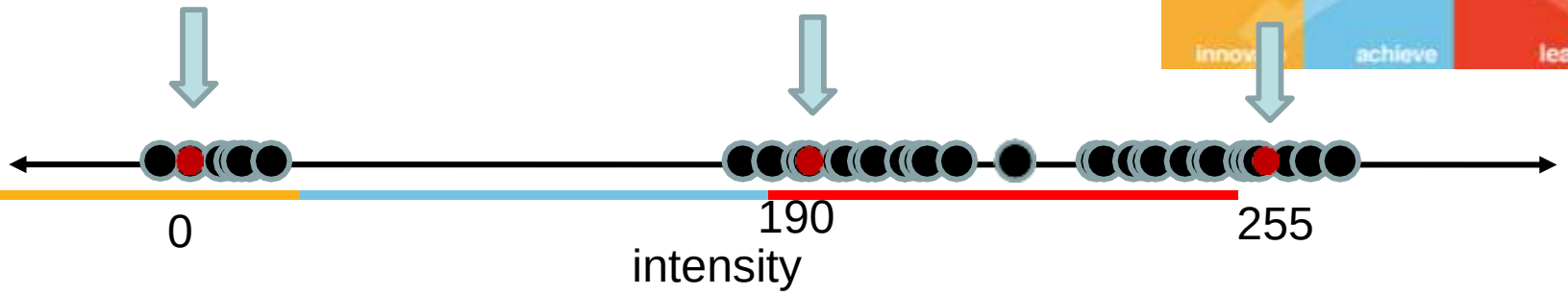


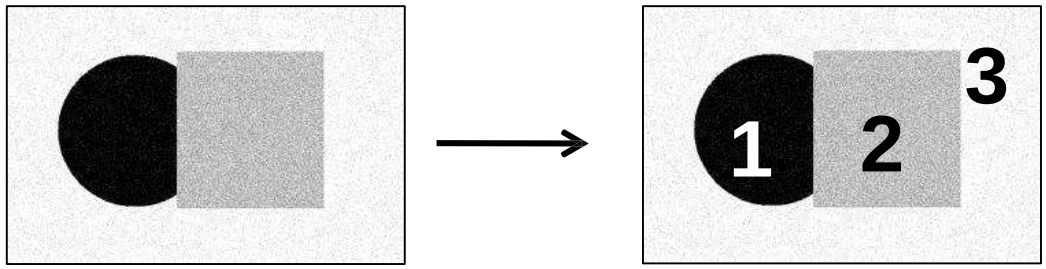


input image



Source: K. Grauman





- Goal: choose three "centers" as the representative intensities, and label every pixel according to which of these centers it is nearest to.
- Best cluster centers are those that minimize SSD between all points and their nearest cluster center c_i:

$$\sum_{\text{clusters } i} \sum_{\text{points p in cluster } i} \|p - c_i\|^2$$

Source: K. Grauman

Image Compression using Segmentation

smaller values of *K* give higher compression at the expense of poorer image quality.

Another way of writing objective

K-means:

- The centre of a cluster is not necessarily one of the input data points (it is the average between the points in the cluster).
- Uses squared Euclidean distance as the measure of dissimilarity between a data point and a prototype vector not suitable where some or all of the variables represent categorical labels

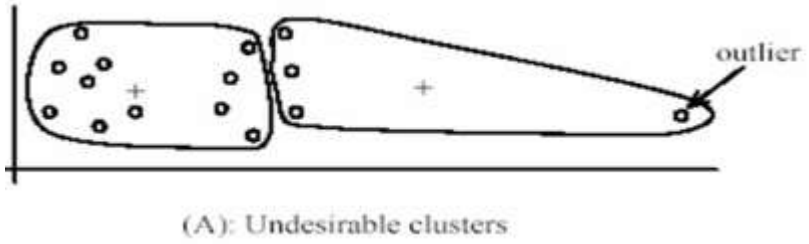
K-medoids (more general distances):

$$\widetilde{J} = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \mathcal{V}(\mathbf{x}_n, \boldsymbol{\mu}_k)$$

- It is common to restrict each cluster prototype to be equal to one of the data vectors assigned to that cluster and can be used with **arbitrary distances**
- k-medoids more robust to noise and outliers as compared to k-means because it minimizes a sum of pairwise dissimilarities instead of a sum of squared Euclidean distances.

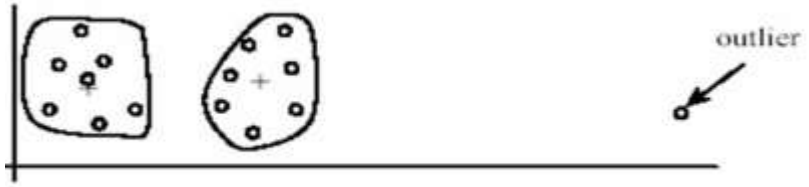
Limitations of K-means

- Setting k?



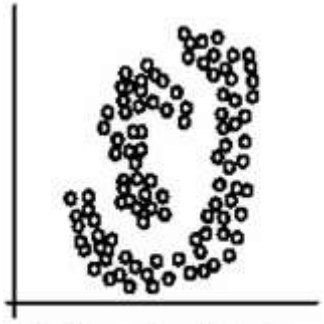
(A): Undesirable clusters

- Sensitive to outliers

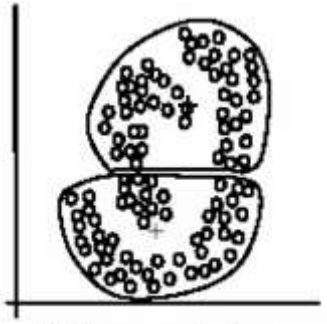


(B): Ideal clusters

- Detects spherical clusters



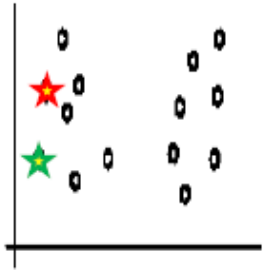
(A): Two natural clusters



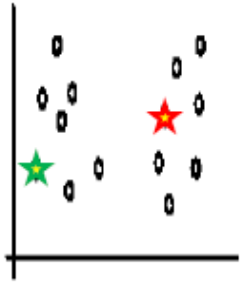
(B): *k*-means clusters

Limitations of K-means

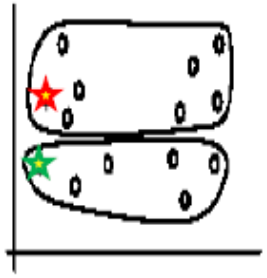
- Sensitive to initial centers
  - Use heuristics or output of another method



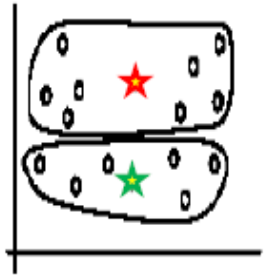
Random selection of seeds (centroids)



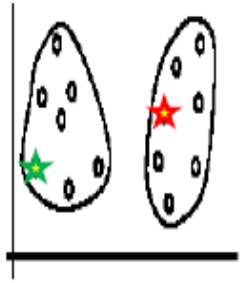
Random selection of seeds (centroids)



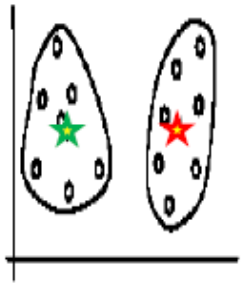
Iteration 1



Iteration 2



Iteration 1



Iteration 2

Hard Clustering VS soft Clustering

- Hard assignments of data points to clusters – small shift of a data point can flip it to a different cluster
- Solution: replace ‘hard’ clustering of K-means with ‘soft’ probabilistic assignments
- Represents the probability distribution of the data as a *Gaussian mixture model*
- GMMs give a probabilistic assignment of points to clusters. This lets us quantify uncertainty. For example, if a point is near the 'border' between two clusters, it's often better to know that it has near equal membership probabilities for these clusters, rather than blindly assigning it to the nearest one.

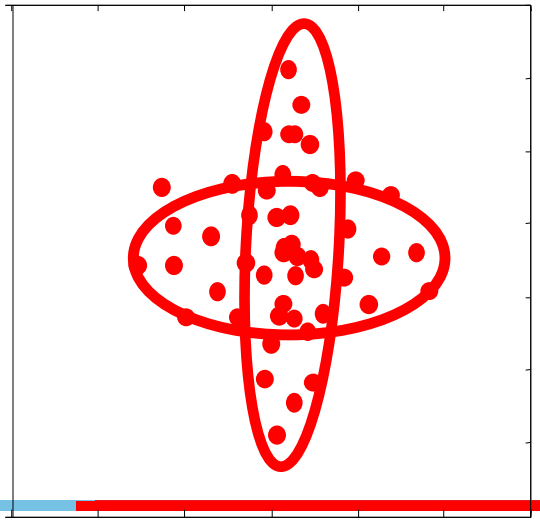
Mixtures of Gaussians

K-means algorithm

- Assigned each example to exactly one cluster
- What if clusters are overlapping?
  - Hard to tell which cluster is right
  - Maybe we should try to remain uncertain
- What if cluster has a non-circular shape?

Gaussian mixture models

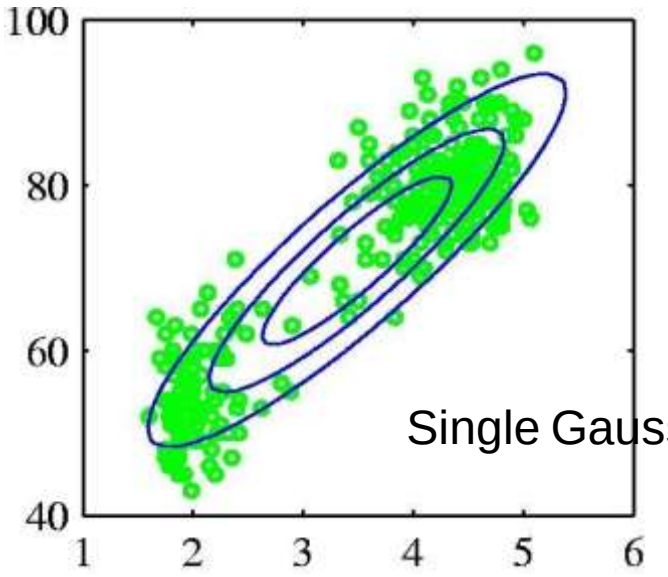
- Clusters modeled as Gaussians
- EM algorithm: assign data to cluster with some *probability*



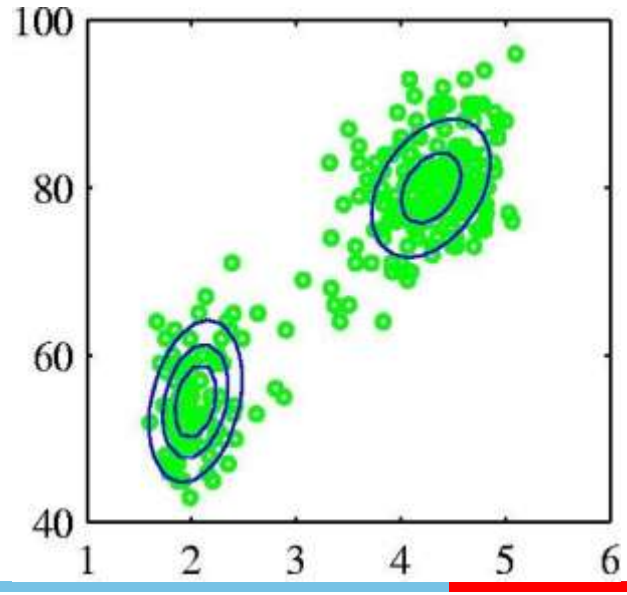
Slide Credit: Alexander Ihler

Probabilistic Clustering

- Represent the probability distribution of the data as a *mixture model*
  - captures uncertainty in cluster assignments
  - gives model for data distribution
  - *Bayesian* mixture model allows us to determine *K*
- Consider mixtures of *Gaussians*
- EM algorithm: assign data to cluster with some *probability*

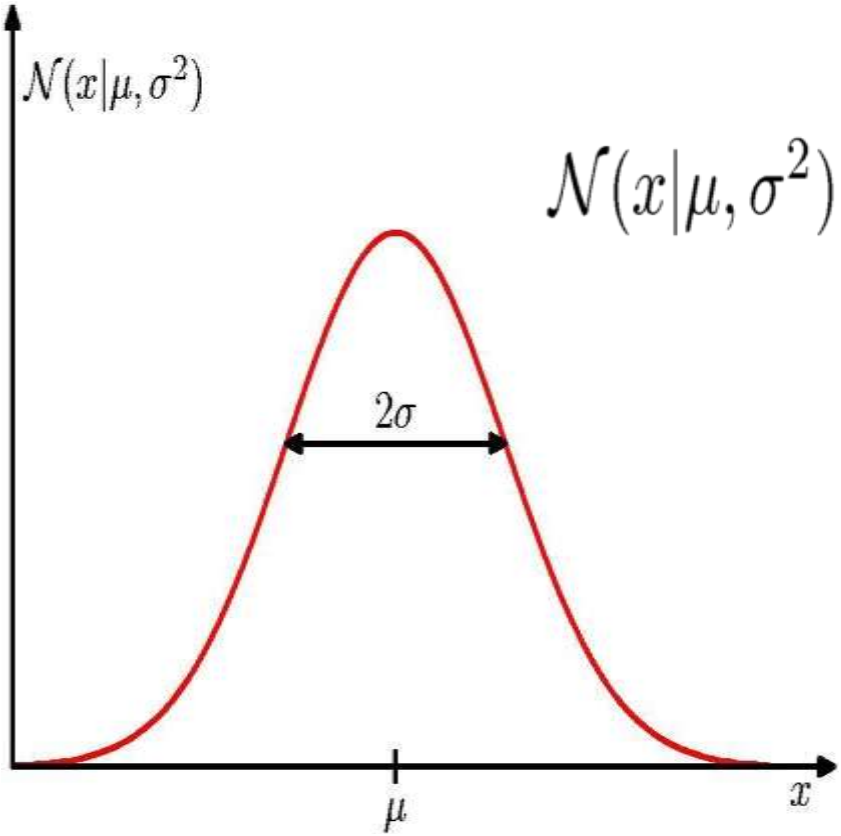


Single Gaussian



Mixture of two Gaussians

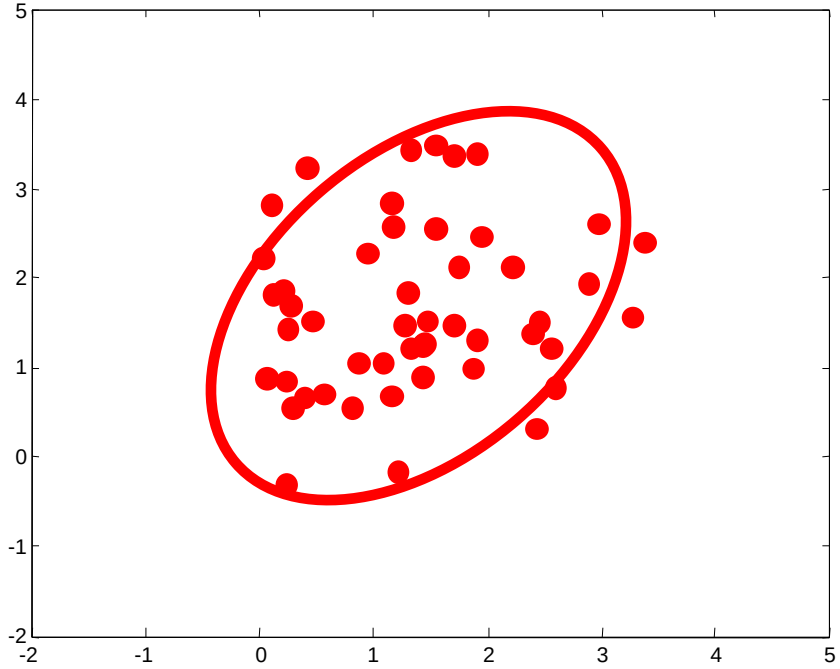
Review: Gaussian Distribution



$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$$

Multivariate Gaussian models

$$\mathcal{N}(\underline{x}\ ;\ \underline{\mu}, \Sigma) = \frac{1}{(2\pi)^{d/2}} |\Sigma|^{-1/2} \exp\left\{ -\frac{1}{2} (\underline{x} - \underline{\mu})^T \Sigma^{-1} (\underline{x} - \underline{\mu}) \right\}$$



Maximum Likelihood estimates

$$\hat{\mu} = \frac{1}{N} \sum_i x^{(i)}$$

$$\hat{\Sigma} = \frac{1}{N} \sum_i (x^{(i)} - \hat{\mu})^T (x^{(i)} - \hat{\mu})$$

We' ll model each cluster using one of these Gaussian "bells"…

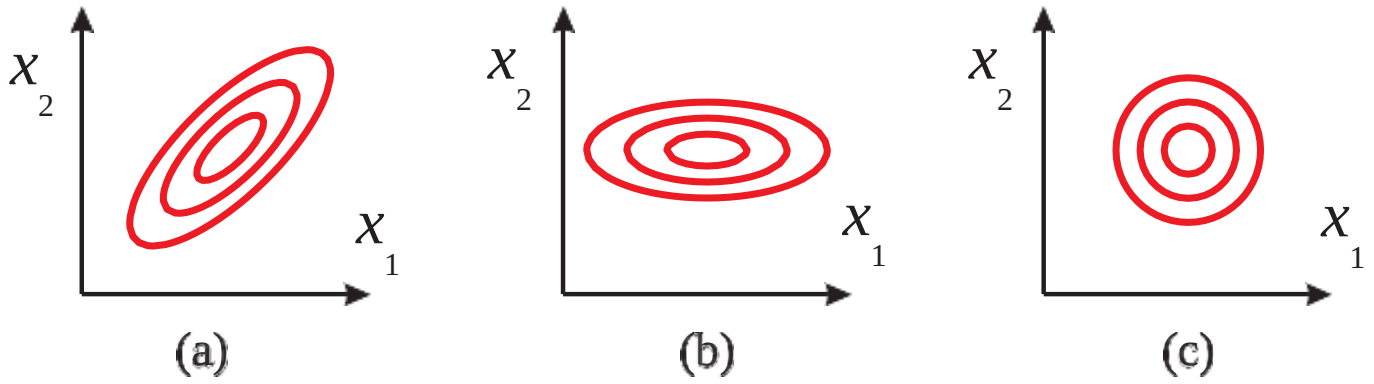
Slide Credit: Alexander Ihler

Example: Mixture of 3 Gaussians

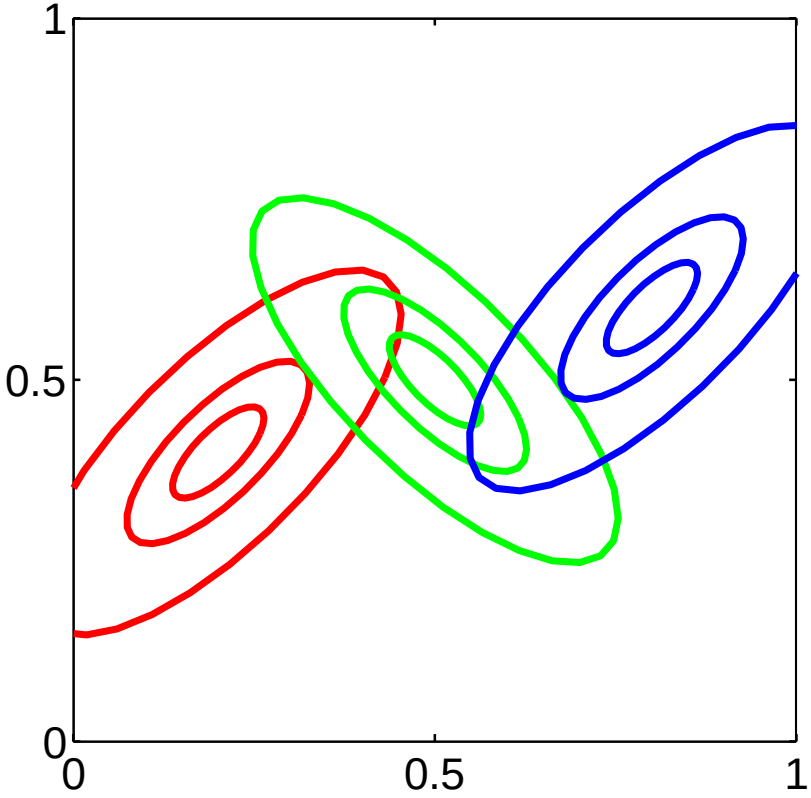
- Multivariate Gaussian

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

mean covariance



Example: Mixture of 3 Gaussians

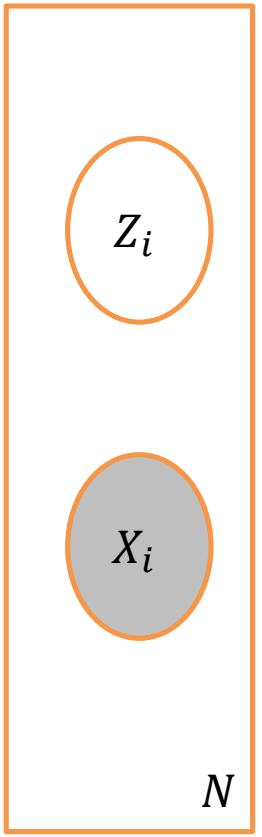


Gaussian Mixture Model - GMM

- The observed data come from a Gaussian mixture distribution which consists of K Gaussians with their own means and variances.
- K classes are latent
- The goal of mixture modeling is now to estimate the most likely class for each observation.
- Therefore, Gaussian mixture modeling can be viewed as a missing data problem.
- Estimation is usually done using the EM algorithm.

GMM as Latent Variable model

- . Each data point is associated with a latent variable(new binary random variable Z_i for each X_i) that indicates which cluster it belongs to.
- When fitting a GMM, we learn a distribution over these latent variables.
- This gives a probability that each data point is a member of each cluster.*Given a data point* **x**, *what is the probability it came from Gaussian k*
- Latent / hidden: not observed in the data
- Joint distribution p(x,z) in terms of a marginal distribution p(z) and a conditional distribution p(x|z)



Parameters of GMM

X	Assuming Latent Variable z for 2 mixture components.	
	$z(1)$	$z(2)$
$x(1)$	0	1
$x(2)$	1	0
...		
$x(N)$	1	0

$\pi : \{\pi_1, \ldots, \pi_k\}$

$\mu : \{\mu_1, \ldots, \mu_K\}$

$\Sigma : \{\Sigma_1, \ldots \Sigma_K\}$

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

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(k)p(\mathbf{x}|k)$$

- K-dimensional binary random variable z having a 1-of-K representation in which a particular element z_k is equal to 1 and all other elements are equal to 0.

GMM Model - p(z) Mixing weights

- Marginal distribution over z is specified in terms of the mixing coefficients π_k, such that $p(z_k = 1) = \pi_k$

- what is the **likelihood that the ith sample came from Gaussian *k***.

- $P(z_k=1) = \pi_k \rightarrow$ $$p(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k}$$

- $0 \leq \pi_k \leq 1$ and $\Sigma_k \pi_k = 1$

GMM Model- p(x|z) Component Density

- p(x|z) : Conditional distribution of x given a particular value for z is a Gaussian. Mixture densities

$$p(\mathbf{x}|z_k = 1) = N(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- z uses a 1-of-K representation, we can also write this distribution in the form

$$p(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k}$$

GMM Model - Marginal Distribution of x

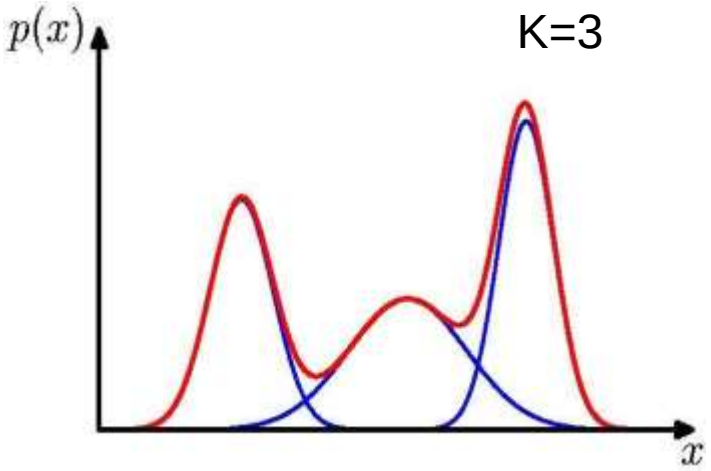
- Consider a superposition of K Gaussian densities of the form

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$$

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

Component

Mixing coefficient



- Normalization and positivity require

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

- Each Gaussian density $N(x|\mu_k,\Sigma_k)$ is called a component of the mixture and has its own mean μ_k and covariance Σ_k

GMM as Clustering algorithm

- For every observed data point $\mathbf{x}_i$ there is a corresponding latent variable $\mathbf{z}_i$
- Probabilistic model of the distribution, where we have represented the marginal distribution in the form $p(\mathbf{x}) = {}_{\mathbf{z}}p(\mathbf{x}, \mathbf{z})$
- posterior probability of a given instance x_i belonging to component k, referred to as the 'responsibility' of component k for producing x_i , denoted as $\boldsymbol{\gamma(z_k) = p(z_k = 1 \mid x)}$
- This gives us the probabilities of x_i belonging to the different components.
- That is precisely how a GMM can be used to cluster data.
- Use Bayes' theorem

GMM Model - the 'responsibility' of component k for producing x_i

using Bayes' theorem

$$\gamma(z_k) \equiv p(z_k = 1|\mathbf{x}) = \frac{p(z_k = 1)p(\mathbf{x}|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(\mathbf{x}|z_j = 1)}$$

$$= \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}.$$

π_k : prior probability of z_k =1, i.e prior probability of picking the *k*th component

Maximum Likelihood

data set of observations $\{\mathbf{x}1, \ldots, \mathbf{x}N\}$, and we wish to model this data using a mixture of Gaussians.

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

Summation inside the logarithm → Complicated expressions for ML solution → Cannot get closed form solutions to ML parameters.

Log of likelihood function:

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

Maximizing the log likelihood function

- Set $d/d\theta \{LL(\theta)\} = 0$ and solve for θ : Non-linear, non-analytically **solvable**
- Use gradient Descent: Doable, but often slow
- **Use EM**

Expectation-Maximization (EM)

- **A general algorithm to deal with hidden data**
- EM is an optimization strategy for objective functions that can be interpreted as likelihoods in the presence of missing data.
- Used to determine the parameters of a mixture with an *a priori* given number of components
- EM is “simpler” than gradient methods
  - No need to choose step size.
- EM is an iterative algorithm with two linked steps:
  - E-step: fill-in hidden values using inference
  - M-step: apply standard MLE/MAP method to completed data

General EM algorithm

Observed data: D={x_1, ……., x_n}

Unknown variables: y

In clustering: y=1….K clusters

Parameters: θ

In GMM : θ= $\pi : \{\pi_1, \ldots, \pi_k\}$

$\mu : \{\mu_1, \ldots, \mu_K\}$

$\Sigma : \{\Sigma_1, \ldots \Sigma_K\}$

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

In GMM : $\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$

EM algorithm for GMM

1. Start with parameters describing each cluster. Initialize the means μ_k, covariances Σ_k and mixing coefficients π_k, and evaluate the initial value of the log likelihood.
2. E Step :

–With initial guesses for the parameters of mixture model, "partial membership" of each data point in each constituent distribution is computed by calculating expectation values for the membership variables of each data point.

– Compute “expected” classes (conditional probabilities of the latent variables) of all datapoints for each class

–In K-means “E-step” we do hard assignment. EM does soft assignment

EM algorithm for GMM

2. E step: Evaluate the responsibilities (posterior probabilities) using the current parameter

$$\gamma(z_k) \equiv p(z_k = 1|\mathbf{x}) = \frac{p(z_k = 1)p(\mathbf{x}|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(\mathbf{x}|z_j = 1)}$$

$$= \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}.$$

For each example x_n,

- **Compute $\gamma(z_{nk})$ the probability that x_n is generated by component Z_k i.e it belongs to cluster k**
- If x_n is very likely under the k_{th} Gaussian, it gets high weight

EM algorithm for GMM

3. M-Step :

- maximize the expectation of the complete-data log-likelihood, computed with respect to the conditional probabilities found in the Expectation step. The result of the maximization is a new parameter vector μ^{new} , Σ^{new} and π^{new}
- Keep γ (z_{nk}) fixed, and apply MLE for maximizing of ln p(X|π,μ,Σ) for μ_k , Σ_k and π_k, to get μ^{new} , Σ^{new} and π^{new}

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

4. Repeat E & M until convergence
   - In practice, the algorithm is deemed to have converged when the change in the log likelihood function, or alternatively in the parameters, falls below some threshold

EM algorithm for GMM

To Estimate:

M-Step

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})\mathbf{x}_n$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})\left(\mathbf{x}_n-\boldsymbol{\mu}_k^{\text{new}}\right)\left(\mathbf{x}_n-\boldsymbol{\mu}_k^{\text{new}}\right)^{\mathrm{T}}$$

$$\pi_k^{\text{new}} = \frac{N_k}{N}$$

$$N_k = \sum_{n=1}^{N}\gamma(z_{nk})$$

Initialize $\boldsymbol{\pi}$, μ,Σ *and also evaluate the log likelihood*

Perform E-Step Given $\gamma(z_k)$

E-Step

$$\gamma(z_k) \equiv p(z_k=1|\mathbf{x}) = \frac{p(z_k=1)p(\mathbf{x}|z_k=1)}{\sum_{j=1}^{K}p(z_j=1)p(\mathbf{x}|z_j=1)} = \frac{\pi_k\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K}\pi_j\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)}$$

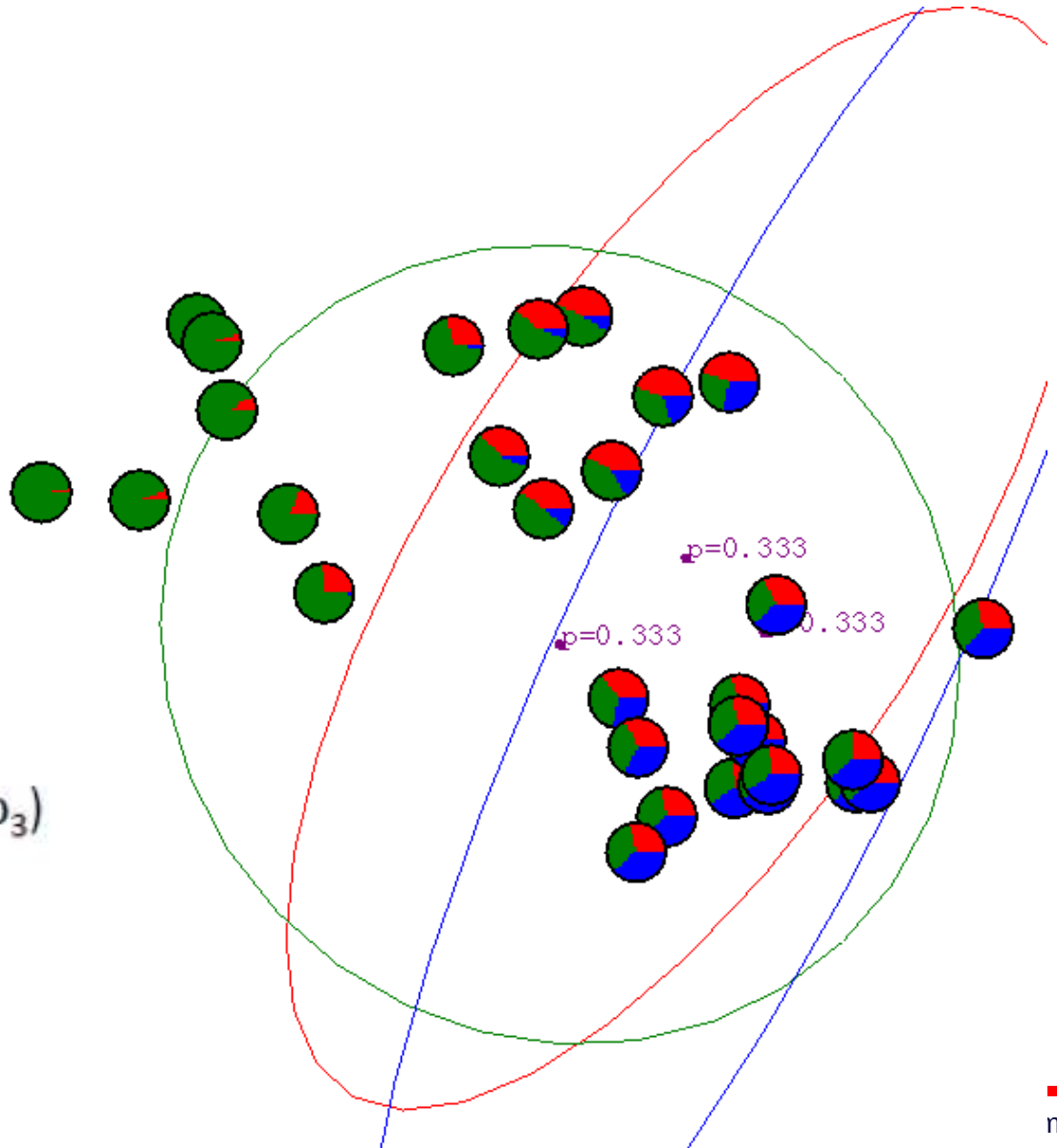
$$p(z_k=1)=\pi_k$$

Perform M-Step Given $\boldsymbol{\pi}$, μ,Σ

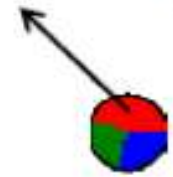
Repeat Until Convergence

N_k = the effective number of points assigned to cluster k

Gaussian Mixture Example: Start

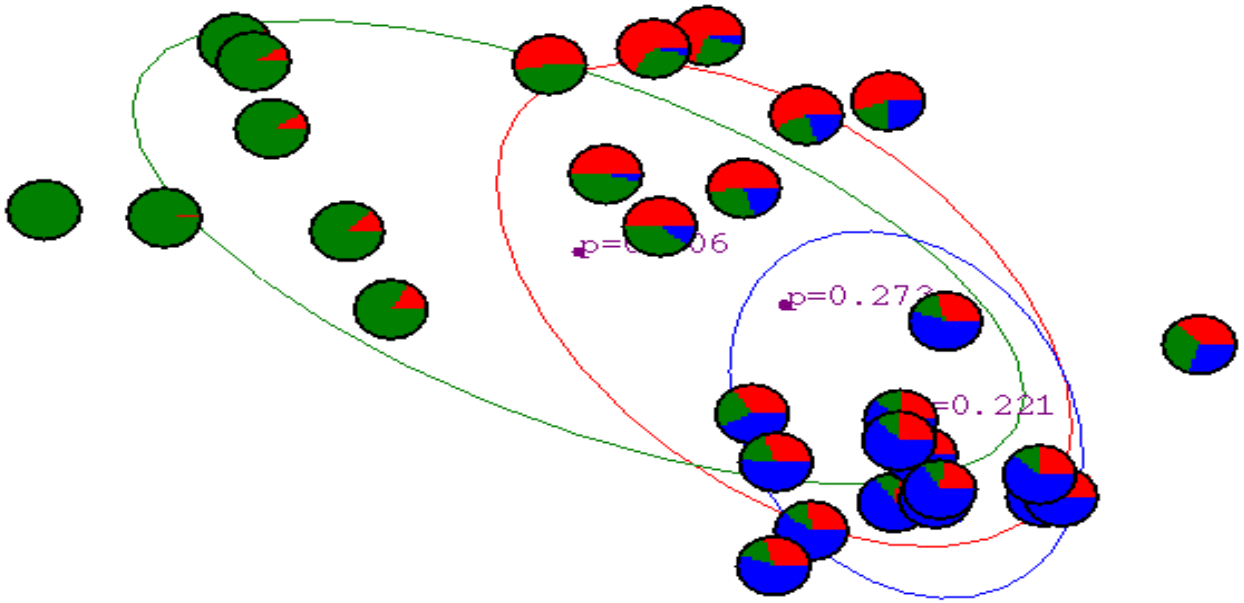


$P(y = \bullet \mid x_j, \mu_1, \mu_2, \mu_3, \Sigma_1, \Sigma_2, \Sigma_3, p_1, p_2, p_3)$



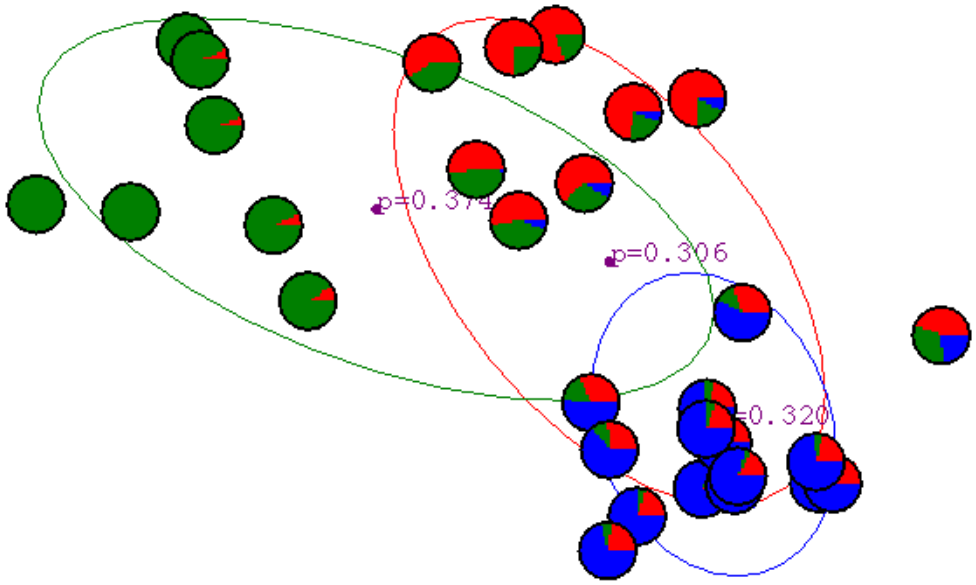
Slide Credit: Andrew W. Moore

After first iteration

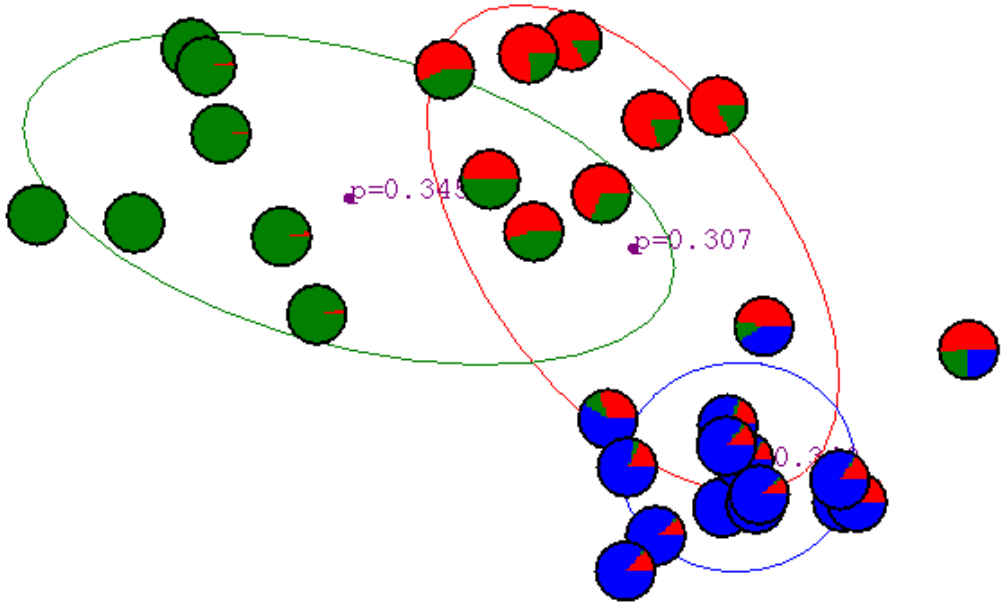


Slide Credit: Andrew W. Moore

After 2nd iteration

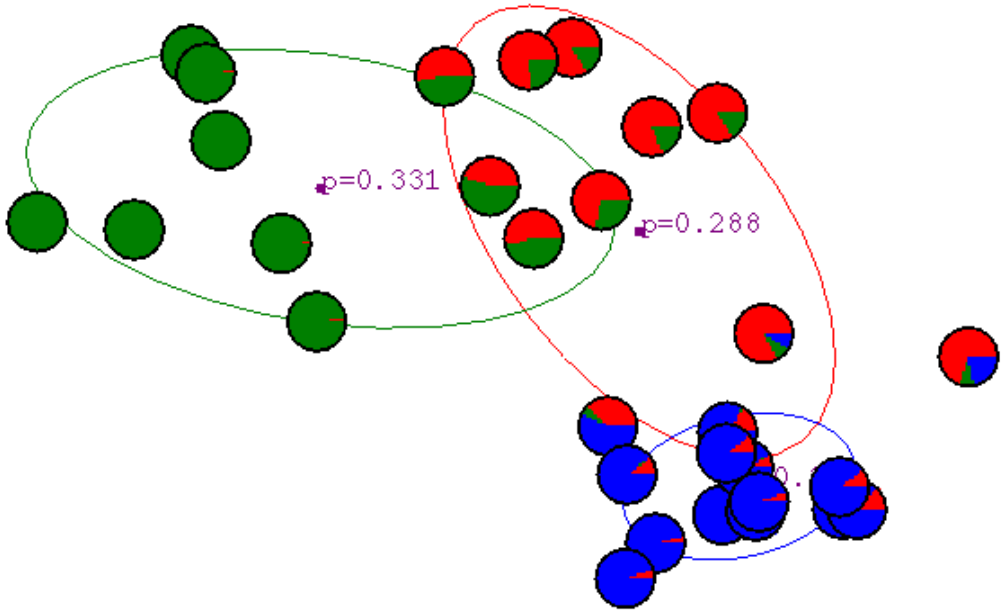


After 3rd iteration



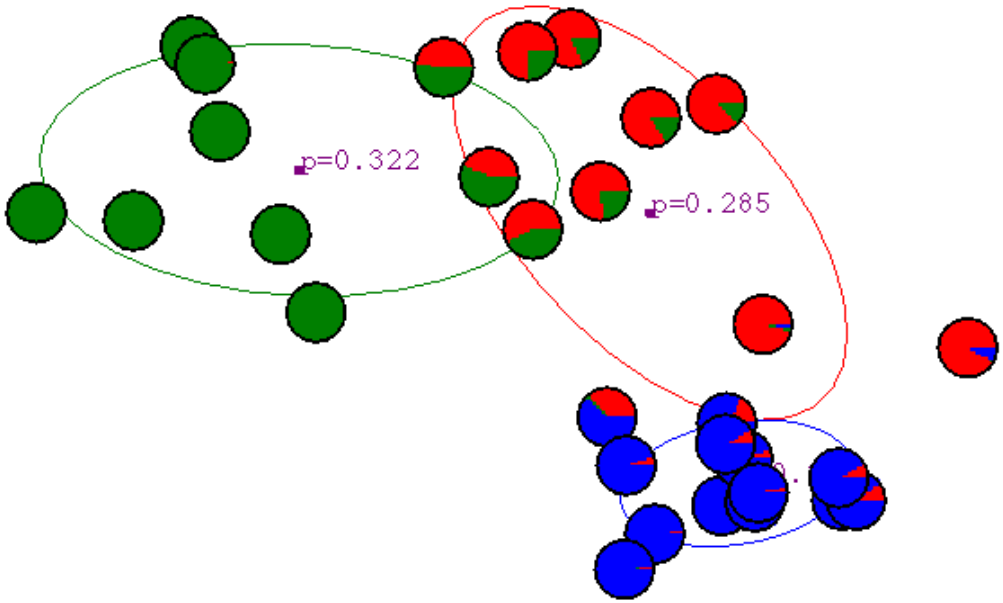
Slide Credit: Andrew W. Moore

After 4th iteration



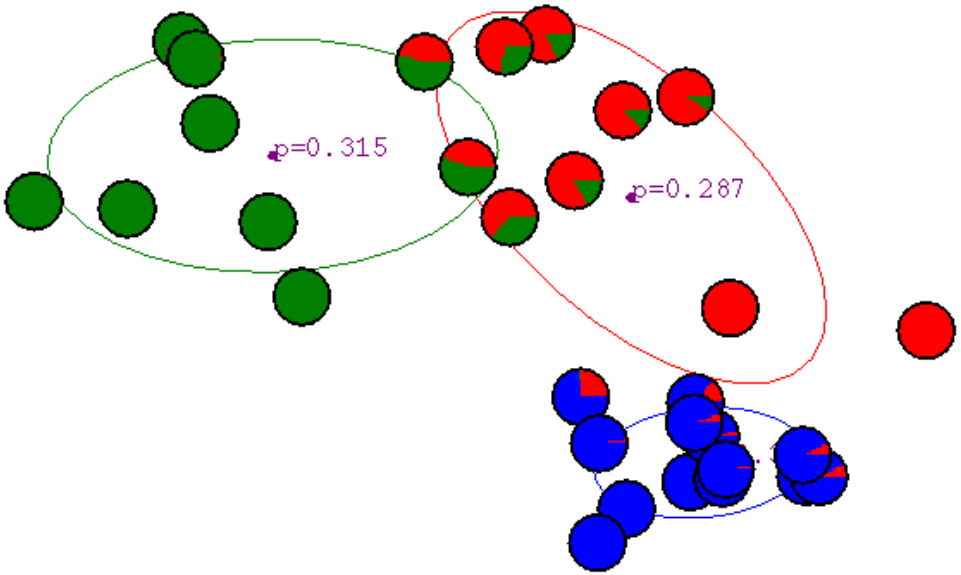
Slide Credit: Andrew W. Moore

After 5th iteration



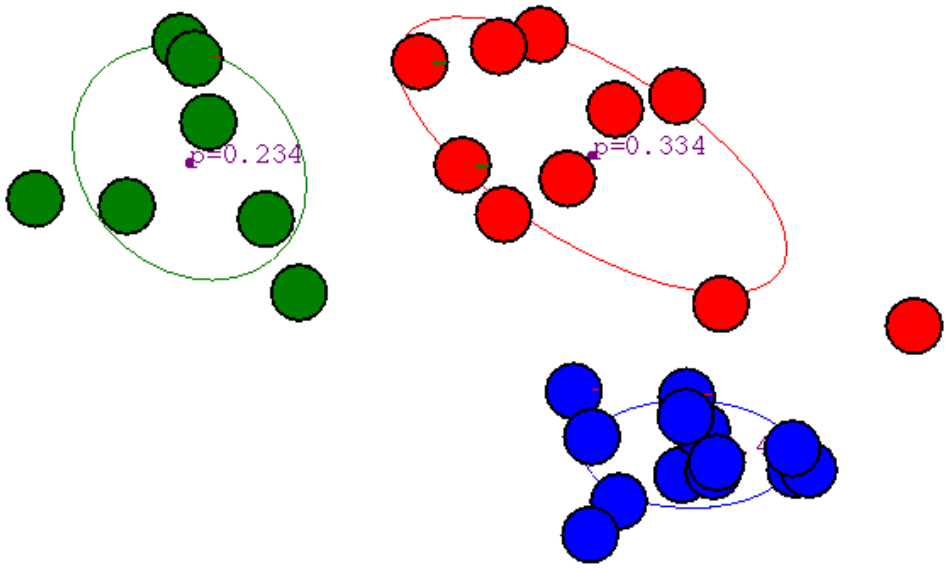
Slide Credit: Andrew W. Moore

After 6th iteration



Slide Credit: Andrew W. Moore

After 20th iteration



Slide Credit: Andrew W. Moore

Closing EM

- EM Algorithm breaks down the problem of estimating the ML parameters of a Mixture Model into E & M Steps and provides an iterative way of computing this
- EM Algorithm converges slowly
  - Use K-Means in the initialization
- EM Algorithm converges to local optimum
  - Sensitive to initialization
  - Each iteration increases the Log likelihood until convergence
  - Many restarts
- What is a good k?
- Issues due to singularity

Example

A simple case:

- We have unlabeled data x_1 , x_2 , …, x_n
- We know there are K classes
- We know P(z=1)=π_1 , P(z=1)=π_2 ,……., P(z=1)=π_k
- We know common variance σ^2
- We don't know μ_1 , μ_2 ,……, μ_k and we want to learn them

Example

Let $(x_1, x_2, x_3) = (2, 4, 7)$ be our three datapoints, presumed to have each been generated from one of two Gaussians.

The stdev of both Gaussians are given: $\sigma_1 = \sigma_2 = 1/\sqrt{2}$.
The prior over the two Gaussians in also given: $\lambda_1 = \lambda_2 = 0.5$

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

Let us initialize the Gaussian means to some reasonable values (inside the data range, and integer valued, to make calculation easy): $\mu_1^{[0]} = 3, \mu_2^{[0]} = 6$.

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right) \left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right)^{\mathrm{T}}$$

$$\pi_k^{\text{new}} = \frac{N_k}{N}$$

i	1	2	3
x_i	2.0	4.0	7.0
$\mathcal{N}(x_i \mid \mu_1)$	$\frac{1}{\sqrt{\pi}}e^{-1}$	$\frac{1}{\sqrt{\pi}}e^{-1}$	$\frac{1}{\sqrt{\pi}}e^{-16}$
$\mathcal{N}(x_i \mid \mu_2)$	$\frac{1}{\sqrt{\pi}}e^{-16}$	$\frac{1}{\sqrt{\pi}}e^{-4}$	$\frac{1}{\sqrt{\pi}}e^{-1}$
$\mathcal{N}(x_i \mid \mu_1)+\mathcal{N}(x_i \mid \mu_2)$	$\frac{1}{2\sqrt{\pi}}(e^{-1}+e^{-16})$	$\frac{1}{2\sqrt{\pi}}(e^{-1}+e^{-4})$	$\frac{1}{2\sqrt{\pi}}(e^{-16}+e^{-1})$
$\gamma(z_i,1)$	$\frac{e^{-1}}{e^{-1}+e^{-16}} \approx 1$	$\frac{e^{-1}}{e^{-1}+e^{-4}} \approx 0.953$	$\frac{e^{-16}}{e^{-1}+e^{-16}} \approx 0$
$\gamma(z_i,2)$	$\frac{e^{-16}}{e^{-1}+e^{-16}} \approx 0$	$\frac{e^{-4}}{e^{-1}+e^{-4}} \approx 0.047$	$\frac{e^{-1}}{e^{-1}+e^{-16}} \approx 1$

And therefore:

$$\mu_1^{[1]} \approx \frac{1 * 2.0 + 0.953 * 4.0 + 0 * 7.0}{1 + 0.953} \approx 2.976$$

and

$$\mu_2^{[1]} \approx \frac{0 * 2.0 + 0.047 * 4.0 + 1 * 7.0}{1 + 0.047} \approx 6.865$$

Segmentation as clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on **intensity** similarity

Feature space: intensity value (1d)

K=2

K=3

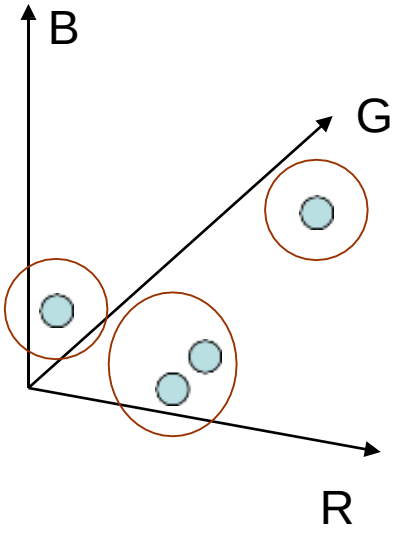
quantization of the feature space; segmentation label map

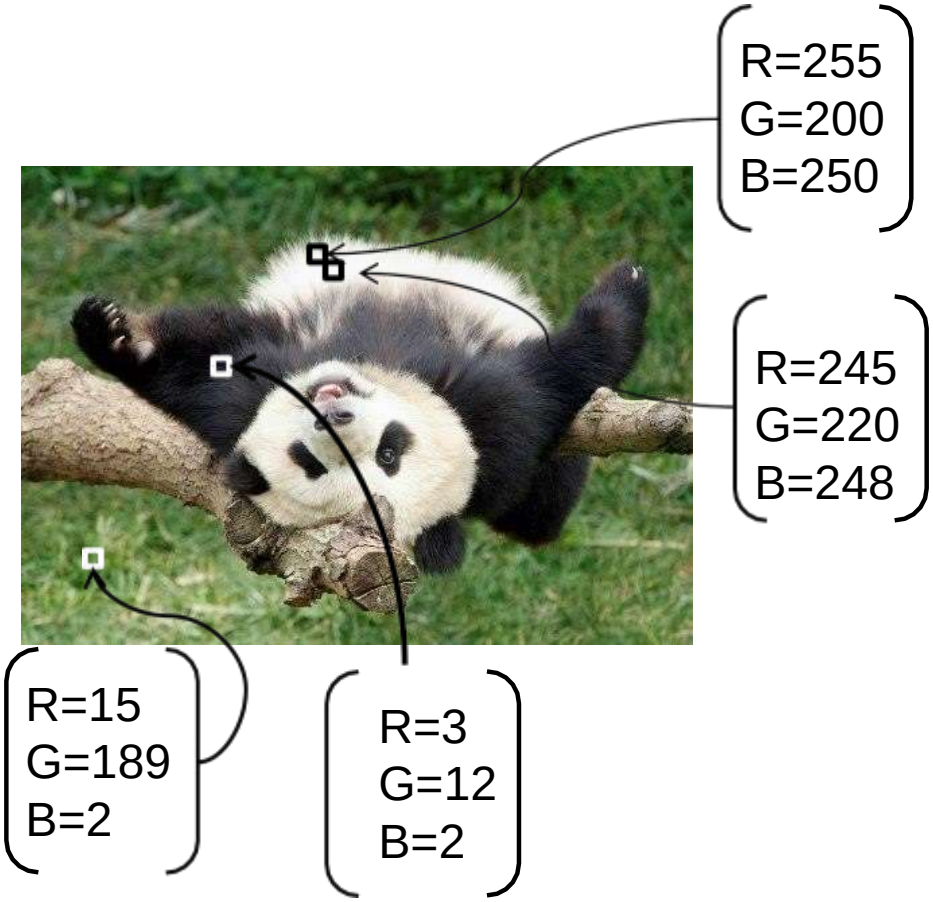
Source: K. Grauman

Segmentation as clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on **color** similarity





Feature space: color value (3-d)

References

Christopher Bishop: Pattern Recognition and Machine Learning, Springer International Edition

https://www.youtube.com/watch?v=TG6Bh-NFhA0

https://www.youtube.com/watch?v=qMTuMa86NzU