



Mixture Models & the EM Algorithm

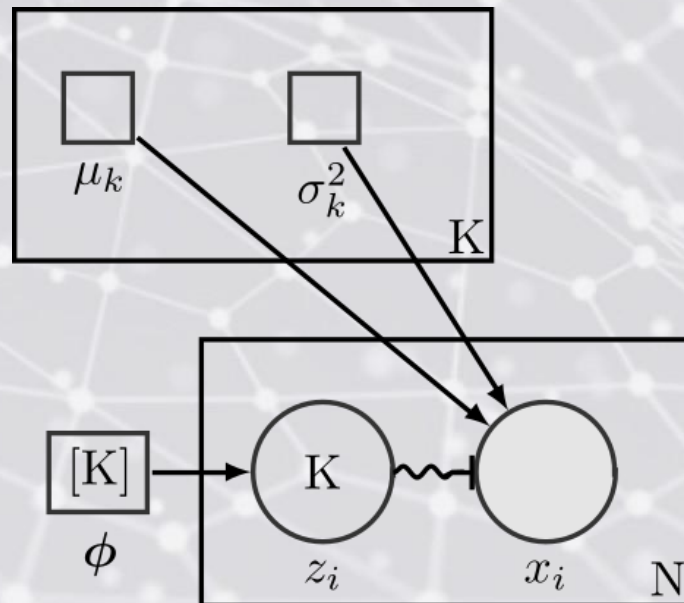
CS 446/546

Outline

- k-Means and Analysis of k-Means
- Fuzzy c-Means
- GMMs (Gaussian Mixture Models)
- EM Algorithm
- HMMs (Hidden Markov Models)

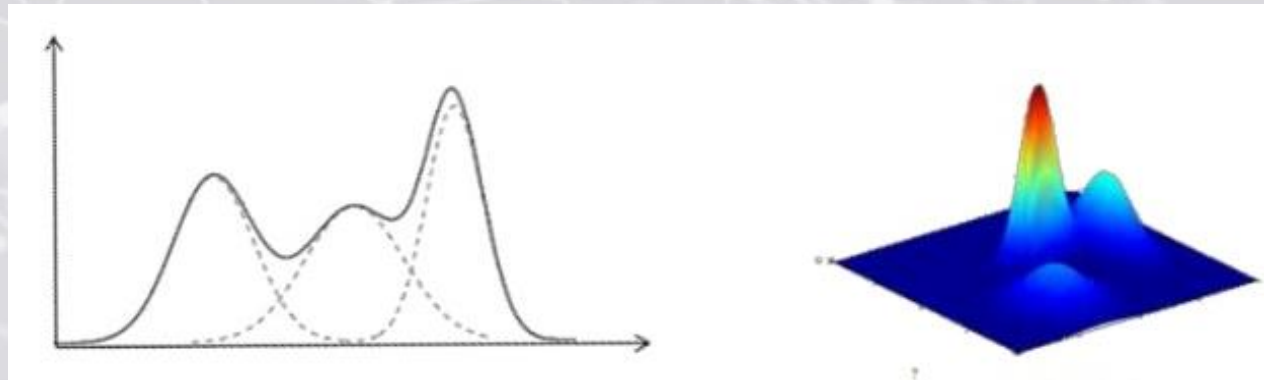
Overview

- The **EM algorithm (expectation-maximization)** is used to find maximum likelihood parameter estimates (i.e. *MLE*) for a model in cases where the equations cannot be solved directly.
- Most commonly, these models incorporate **latent/hidden variables** (i.e. unobserved data) in addition to **known/observed variables**.



Overview

- In some instances, the latent variables comprise *intrinsic* unknown quantities (e.g. the unknown parameters of a mixture model); conversely, however, it is also often useful to introduce latent variables *extrinsically* to model complex dependencies among variables using simpler components for increased tractability.
- In particular, *mixture models* (e.g. GMM, HMM) can be interpreted in terms of discrete latent variables.



Overview

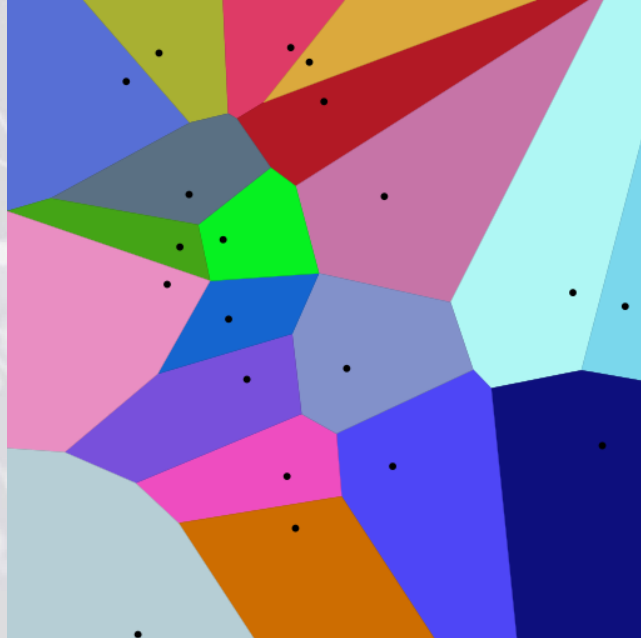
- In addition to providing a framework for building more complex probability distributions, mixture models can also be used to cluster data.
- For mixture models, the latent variables can be interpreted as defining assignments of data points to specific components of the mixture.
- Next we consider (3) classic applications of latent variable models commonly used in combination with the EM algorithm: *K-means*, *GMMs* and *HMMs*. Lastly, we discuss the EM algorithm in the general case.

Issues for clustering algorithms

- How to measure distance between pairs of instances?
- How many clusters to create?
- Should clusters be hierarchical? (i.e., clusters of clusters)
- Should clustering be “soft”? (i.e., an instance can belong to different clusters, with “weighted belonging”)

k-Means

- k-means is a very popular (and simple) clustering algorithm used in ML and data science.
- k -means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a *prototype of the cluster*. This results in a partitioning of the data space into **Voronoi cells**.



*Voronoi
Tessellation; 20
points and their
Voronoi cells.*



k-Means

- Given a set of observations $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$, where each observation is a d-dimensional real vector, k-means clustering endeavors to partition the n observations into k ($\leq n$) sets $\mathbf{S} = \{S_1, S_2, \dots, S_k\}$ so as to **minimize the within-cluster sum of squares (WCSS)** – which is to say we desire a reasonably “tight” clustering.

k-Means

- Given a set of observations $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$, where each observation is a d-dimensional real vector, k-means clustering endeavors to partition the n observations into k ($\leq n$) sets $\mathbf{S} = \{S_1, S_2, \dots, S_k\}$ so as to **minimize the within-cluster sum of squares** (WCSS) – which is to say we desire a reasonably “tight” clustering.
- Define **binary indicator variables**, $r_{nk} \in \{0,1\}$, where $k=1,\dots,K$ describing the cluster to which the \mathbf{x}_n datum is assigned.
- Formally, the objective is given by:

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

where $\boldsymbol{\mu}_i$ is the mean of cluster S_i .

k-Means

- Formally, the objective is given by:

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

where $\boldsymbol{\mu}_i$ is the mean of cluster S_i .

- Our goal is to find values for the $\{r_{nk}\}$ and the $\{\mu_k\}$ as to minimize J .
- We can do this through an iterative procedure in which each iteration involves two successive steps corresponding to successive optimizations with respect to the r_{nk} and μ_k .

k-Means

- Formally, the objective is given by:

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

where $\boldsymbol{\mu}_i$ is the mean of cluster S_i .

- Our goal is to find values for the $\{r_{nk}\}$ and the $\{\mu_k\}$ as to minimize J .
- We can do this through an iterative procedure in which each iteration involves two successive steps corresponding to successive optimizations with respect to the r_{nk} and μ_k .

Choose *initial values* for the μ_k .

- (1) Minimize J wrt r_{nk} keeping the μ_k fixed **(E-step)**
- (2) Minimize J wrt μ_k keeping the r_{nk} fixed **(M-step)**

Repeat until convergence

k-Means

- Formally, the objective is given by:

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

where $\boldsymbol{\mu}_i$ is the mean of cluster S_i .

- (1) Minimize J wrt r_{nk} keeping the μ_k fixed (**E-step**)

Let's consider the determination of r_{nk} .

- Notice that because J is a linear function of r_{nk} , this optimization is elementary and yields a closed form solution.

Q: What is the solution?

k-Means

- Formally, the objective is given by:

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

where $\boldsymbol{\mu}_i$ is the mean of cluster S_i .

- (1) Minimize J wrt r_{nk} keeping the μ_k fixed (**E-step**)

Let's consider the determination of r_{nk} .

- Notice that because J is a linear function of r_{nk} , this optimization is elementary and yields a closed form solution.

(*) Intuitively: J involves the sum of n different (independent) terms, so we can optimize for each n separately by choosing r_{nk} to be 1 for whichever value k minimizes – which is to say we simply assign the n th datum to the closest cluster center.

k-Means

- Formally, the objective is given by:

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

where $\boldsymbol{\mu}_i$ is the mean of cluster S_i .

- (1) Minimize J wrt r_{nk} keeping the $\boldsymbol{\mu}_k$ fixed (**E-step**)

- The preceding argument yields:

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

k-Means

- Formally, the objective is given by:

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

where $\boldsymbol{\mu}_i$ is the mean of cluster S_i .

(2) Minimize J wrt μ_k keeping the r_{nk} fixed (**M-step**)

Let's consider the determination of μ_k with the r_{nk} fixed.

Q: How do we proceed?

k-Means

- Formally, the objective is given by:

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

where $\boldsymbol{\mu}_i$ is the mean of cluster S_i .

(2) Minimize J wrt μ_k keeping the r_{nk} fixed (**M-step**)

Let's consider the determination of μ_k with the r_{nk} fixed.

(*) The objective function J is quadratic in μ_k and it can thus be minimized by setting the derivative of J wrt μ_k equal to zero:

$$2 \sum_{n=1}^N r_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k) = 0 \rightarrow \boldsymbol{\mu}_k = \frac{\sum_n r_{nk} \mathbf{x}_n}{\sum_n r_{nk}}$$

Where the denominator of the last expression is equal to the number of points assigned to cluster k , and so this result has a simple interpretation: namely, we set μ_k equal to the mean of all the data points \mathbf{x}_n assigned to cluster k .

k-Means: Summary

- Given an initial set of k means $\mu_1^{(1)}, \dots, \mu_k^{(1)}$ k-means alternates between the following (2) steps:

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

(I) Assignment Step (i.e., the expectation step):

Assign each observation to the cluster whose mean has the least squared Euclidean distance, this is intuitively the "nearest" mean. Mathematically, this means partitioning the observations according to the Voroni tessellation generated by the means.

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

(II) Update Step (i.e., the parameter maximization step):

- Calculate the new means to be the **centroids** of the observations in the new clusters.

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

- The algorithm has **converged** when the assignments no longer change. There is no guarantee that the optimum is found using this algorithm.

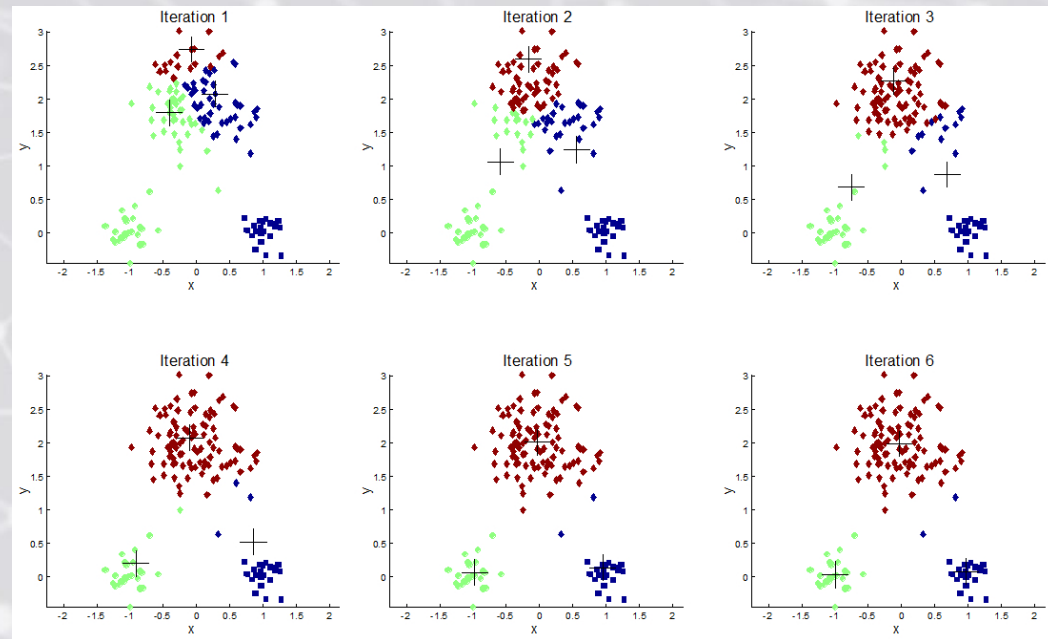
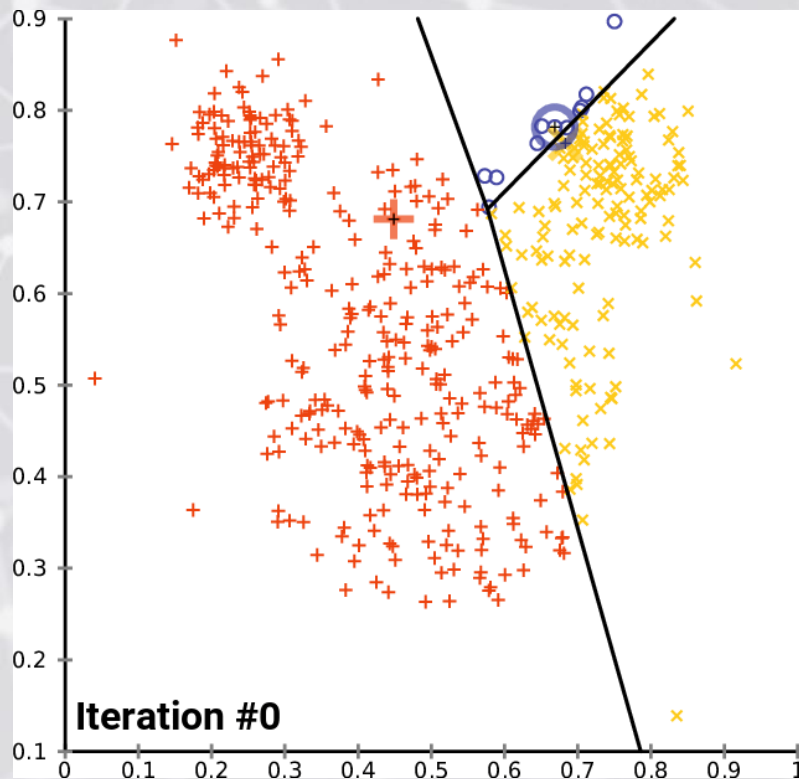
k-Means

(I) **Assignment Step** (i.e., the expectation step):

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

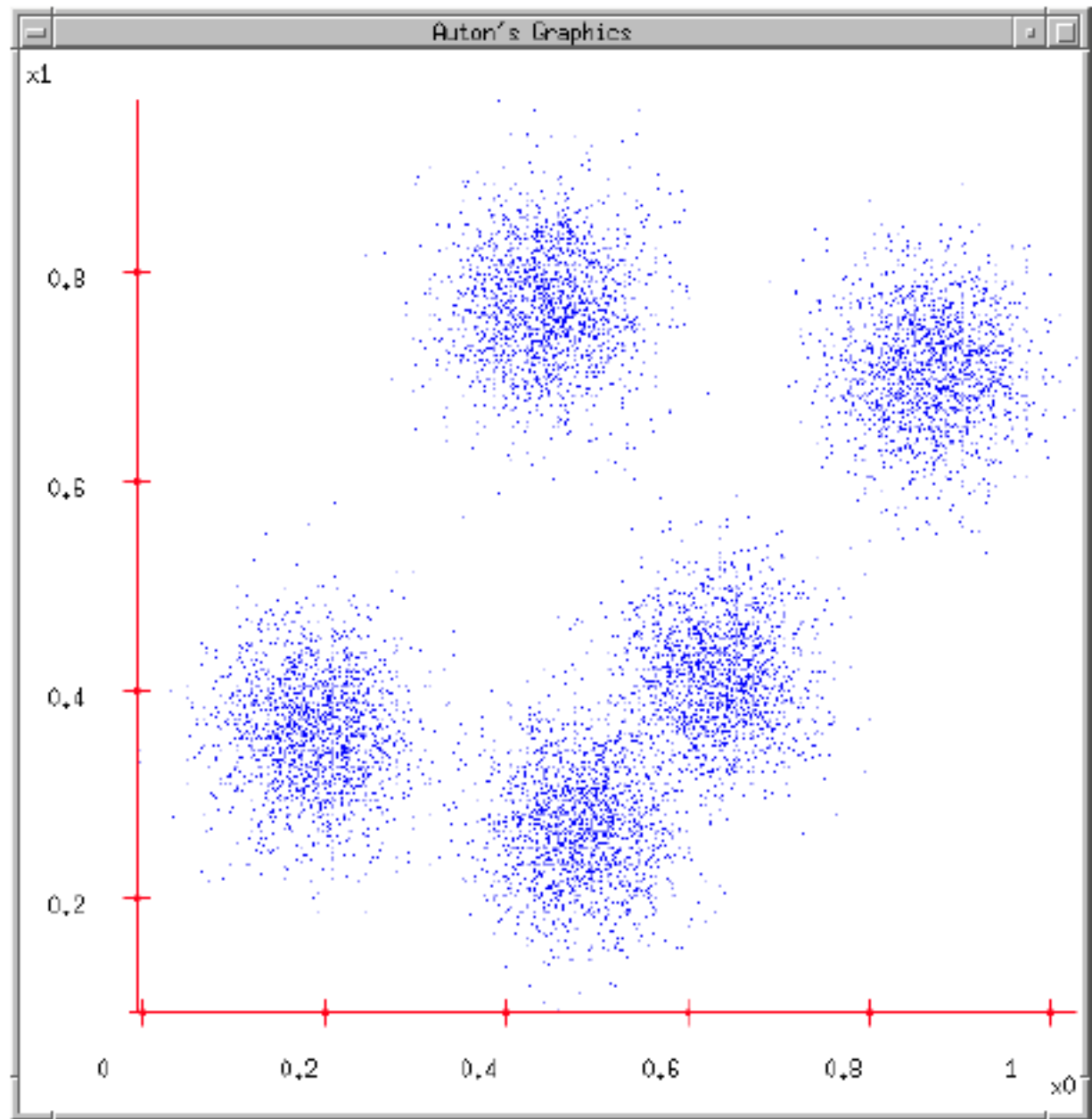
(II) **Update Step** (i.e., the parameter maximization step):

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



K-means

1. Ask user how many clusters they'd like.
(e.g. $k=5$)



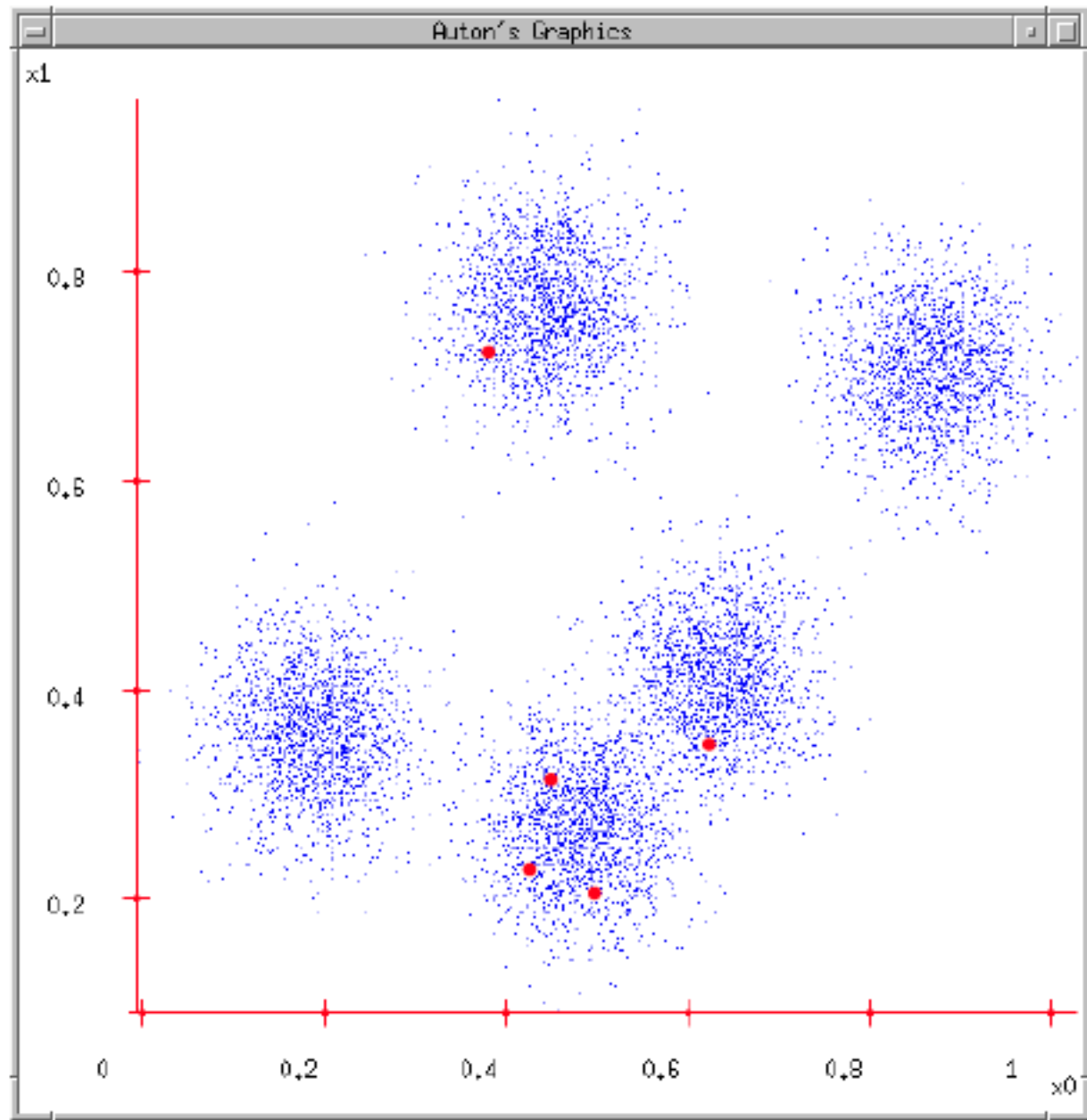
Adapted from Andrew Moore,
<http://www.cs.cmu.edu/~awm/tutorials>

Copyright © 2001, 2004, Andrew W. Moore

K-means and Hierarchical Clustering: Slide 6

K-means

1. Ask user how many clusters they'd like.
(e.g. $k=5$)
2. Randomly guess k cluster Center locations



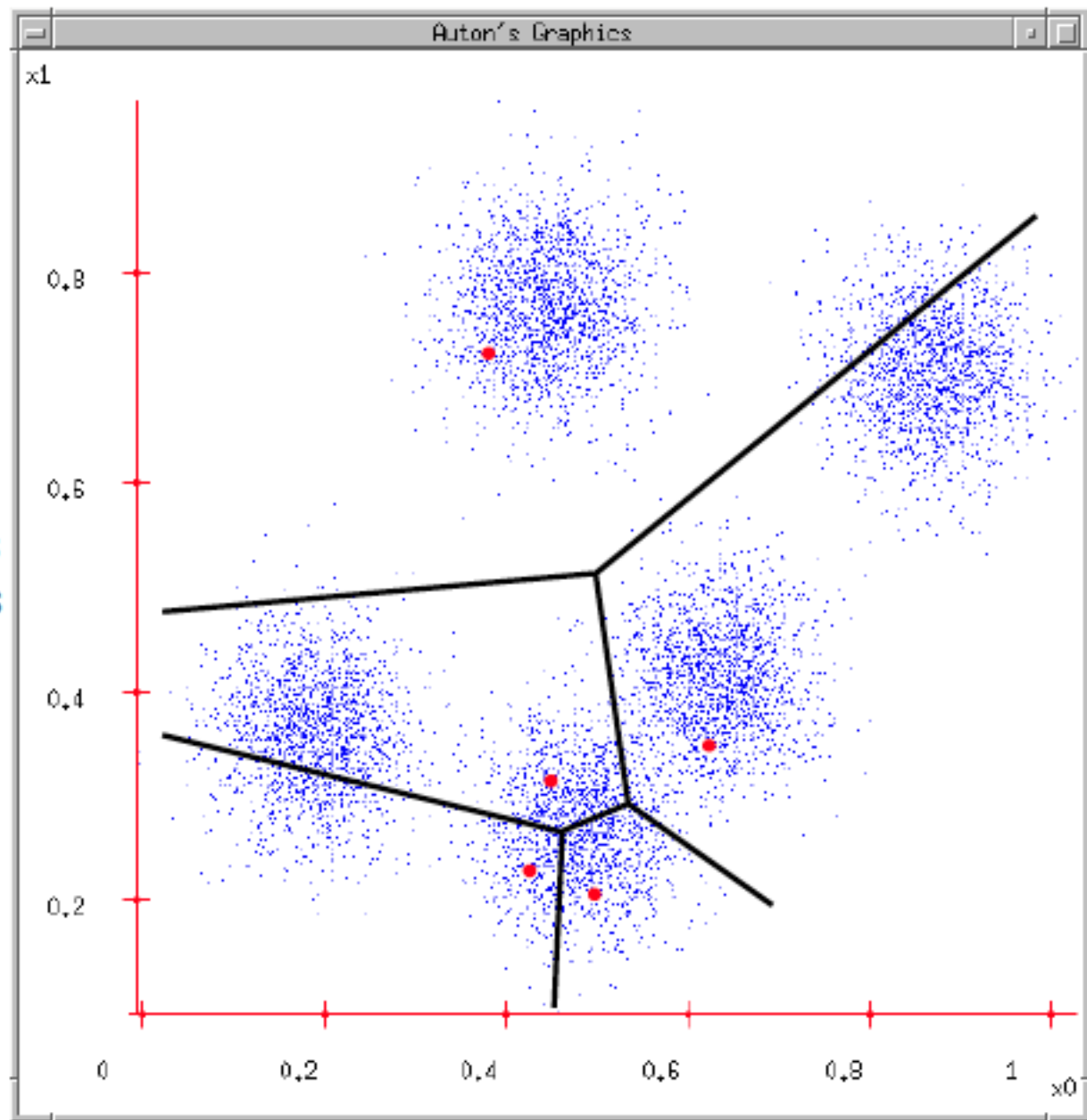
Adapted from Andrew Moore,
<http://www.cs.cmu.edu/~awm/tutorials>

Copyright © 2001, 2004, Andrew W. Moore

K-means and Hierarchical Clustering: Slide 7

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. (Thus each Center "owns" a set of datapoints)



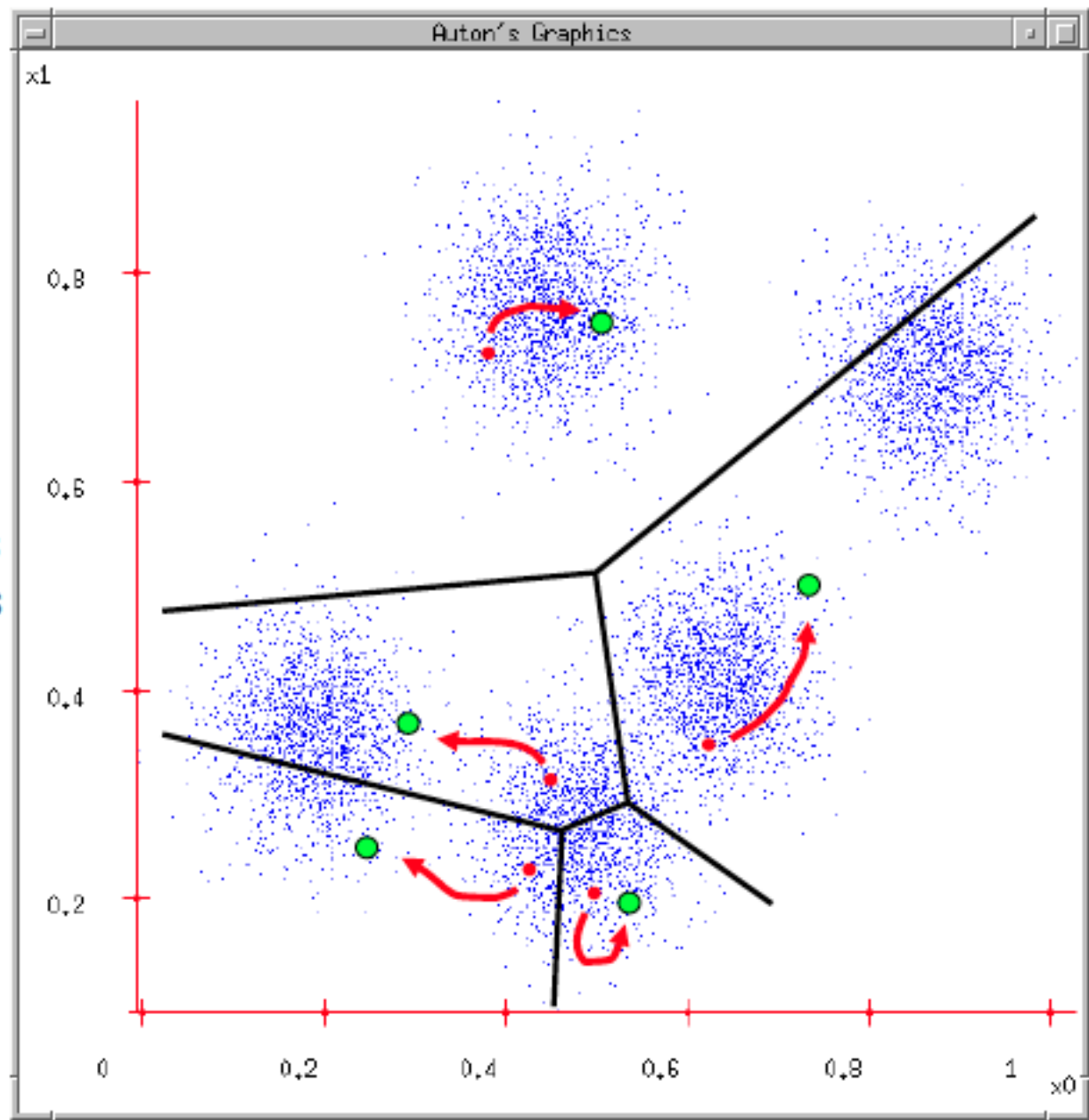
Adapted from Andrew Moore,
<http://www.cs.cmu.edu/~awm/tutorials>

Copyright © 2001, 2004, Andrew W. Moore

K-means and Hierarchical Clustering: Slide 8

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



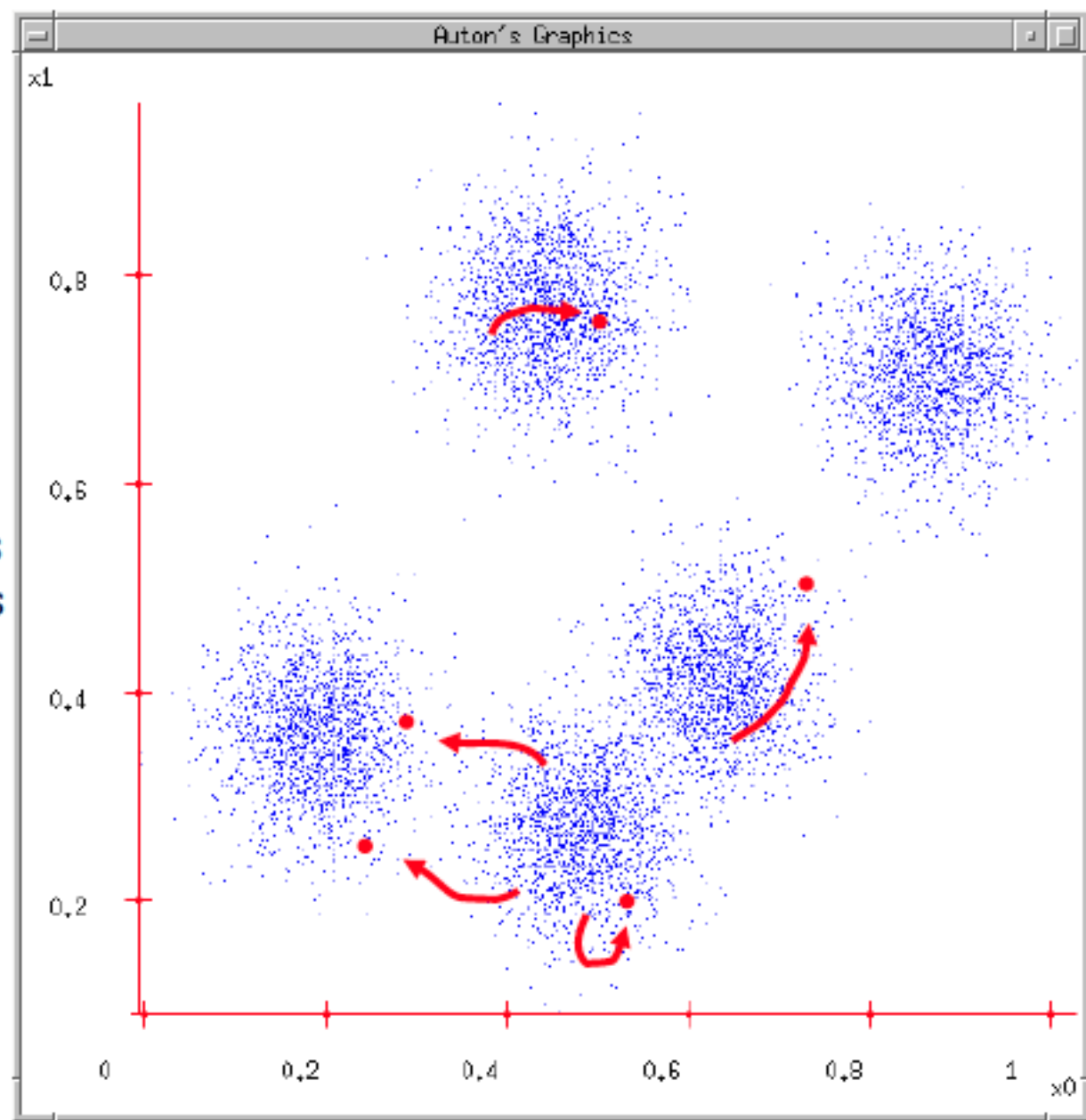
Adapted from Andrew Moore,
<http://www.cs.cmu.edu/~awm/tutorials>

Copyright © 2001, 2004, Andrew W. Moore

K-means and Hierarchical Clustering: Slide 9

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!



K-means Clustering Algorithm Pseudocode

Algorithm	Basic K-means algorithm.
------------------	--------------------------

- 1: Select K points as initial centroids.
 - 2: **repeat**
 - 3: Form K clusters by assigning each point to its closest centroid.
 - 4: Recompute the centroid of each cluster.
 - 5: **until** Centroids do not change.
-

Distance metric: Chosen by user.

For numerical attributes, often use L_2 (Euclidean) distance: $d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$

Centroid of a cluster here refers to the *mean* of the points in the cluster.

(*) NB: Using a different distance function other than (squared) Euclidean distance may stop the algorithm from converging. Various modifications of k-means such as *spherical k-means* have been proposed to allow using other distance measures.

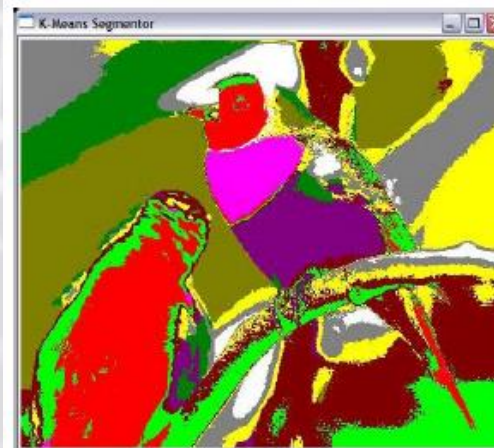
Example: Image segmentation by K -means clustering by color

From <http://vitroz.com/Documents/Image%20Segmentation.pdf>

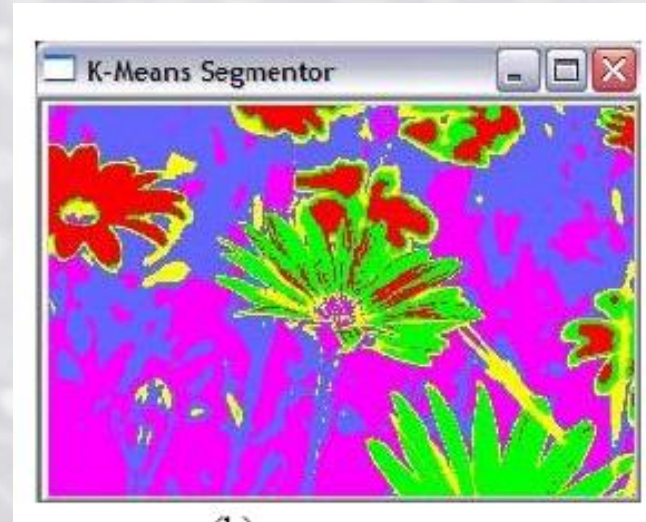
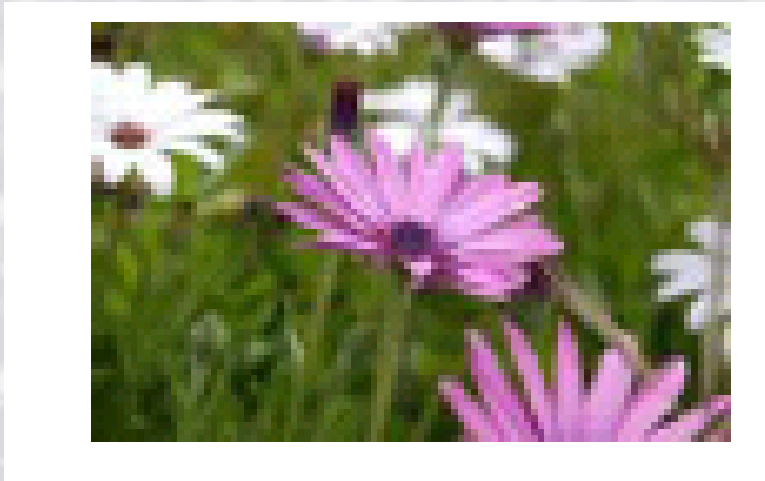
$K=5$, RGB space



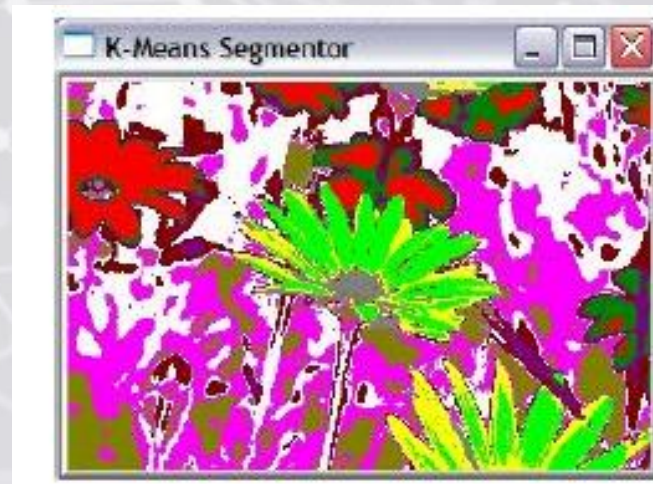
$K=10$, RGB space



$K=5$, RGB space

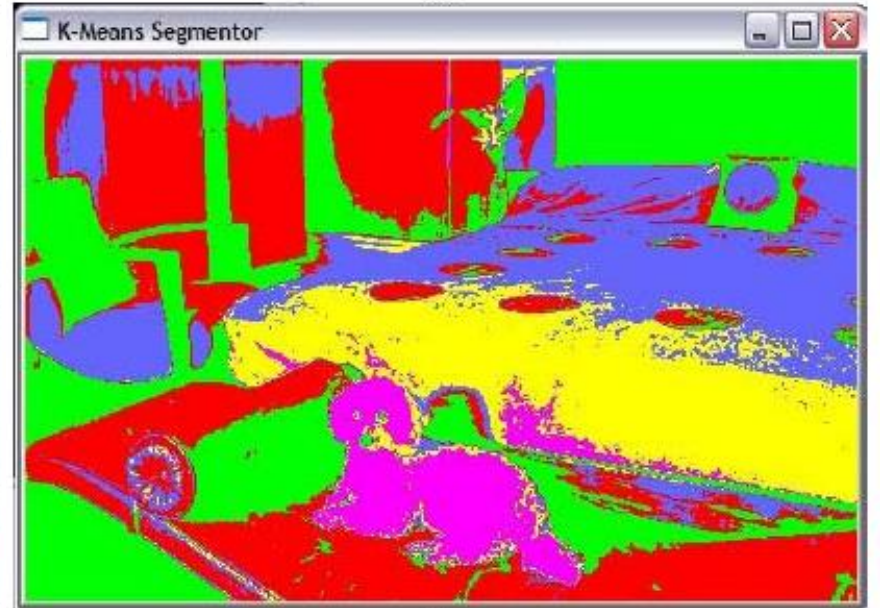


$K=10$, RGB space

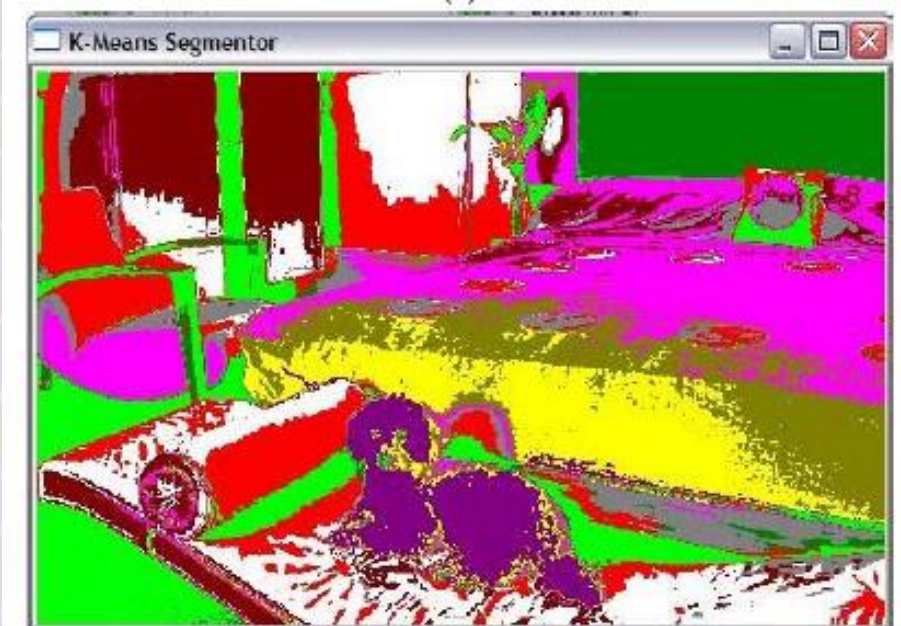


$K=5$, RGB space

(c)



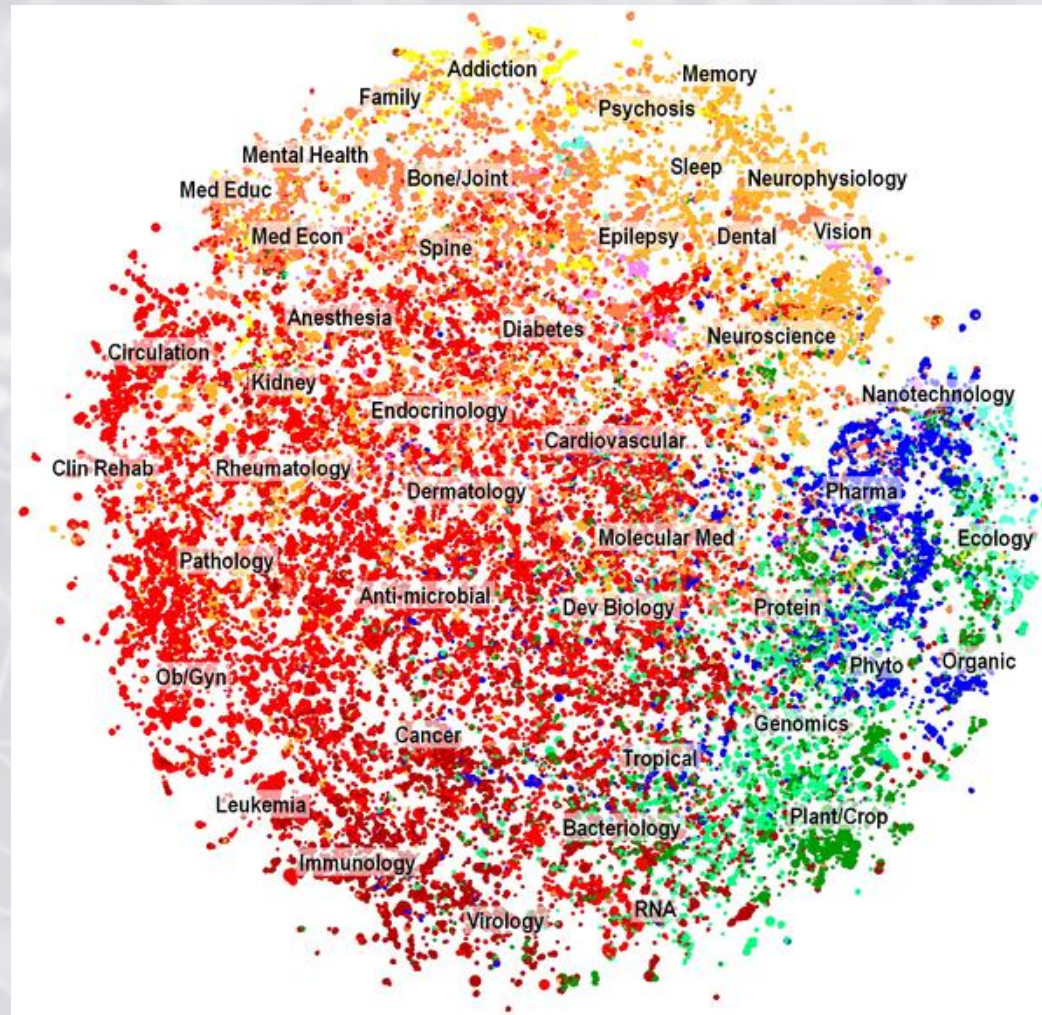
$K=10$, RGB space



Example: Clustering text documents

- A text document is represented as a feature vector of word frequencies (see: *Word2vec*).
- Distance between two documents is the cosine of the angle between their corresponding feature vectors.

Figure 4. Two-dimensional map of the PMRA cluster solution, representing nearly 29,000 clusters and over two million articles.



Boyack KW, Newman D, Duhon RJ, Klavans R, et al. (2011) Clustering More than Two Million Biomedical Publications: Comparing the Accuracies of Nine Text-Based Similarity Approaches. PLoS ONE 6(3): e18029. doi:10.1371/journal.pone.0018029
<http://www.plosone.org/article/info:doi/10.1371/journal.pone.0018029>

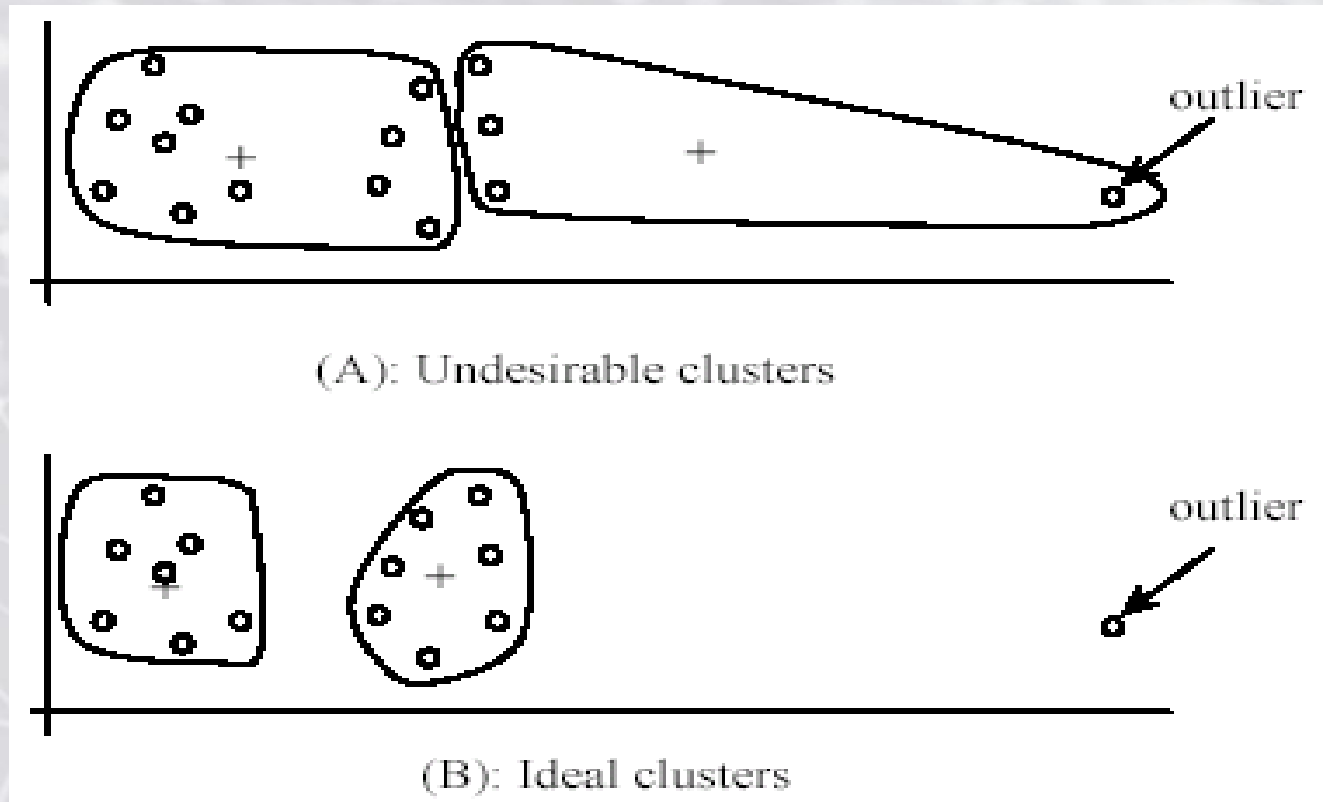
K-means Analysis

- Convergence: k-means is guaranteed to converge in a finite number of steps (irrespective of the initial centroid assignment). Why? In short: there are only a finite number of ways to cluster n data points into k clusters (although note that this number can be large). Usually convergence is relatively fast in practice.
- *NB*: The algorithm is nevertheless not guaranteed to generate a (globally) optimal clustering.
- Complexity: In general, finding the optimal solution to k-means for observations in d dimensions is **NP-hard** (even in the 2-class case).
- The run-time of k-means is $O(nkdi)$, where n is the size of the data set, d is the dimension, k is the number of clusters and i is the number of clusters needed until convergence. k-means is therefore oftentimes considered a linear run-time algorithm; in the worst-case it is more aptly described as superpolynomial.

Potential Issues for K -means

- The algorithm is only applicable if the mean is defined.
 - For categorical data, use **K -modes**: The centroid is represented by the most frequent values.
- The user needs to specify K .
- Cluster morphology can be severely limited (epsilon-balls, etc.)
- Algorithms makes hard cluster assignments (either element belongs to a particular cluster or it does not).
- The algorithm is sensitive to outliers
 - Outliers are data points that are very far away from expected range/other data points.
 - Outliers could be errors in the data recording or some special data points with very different values.
 - Note the *mode* is a more **robust measure of center** (than, say the mean), meaning it is less susceptible to outliers; thus, we can potentially use K -modes to safeguard against influence of outliers.

Issues for K -means: Problems with outliers

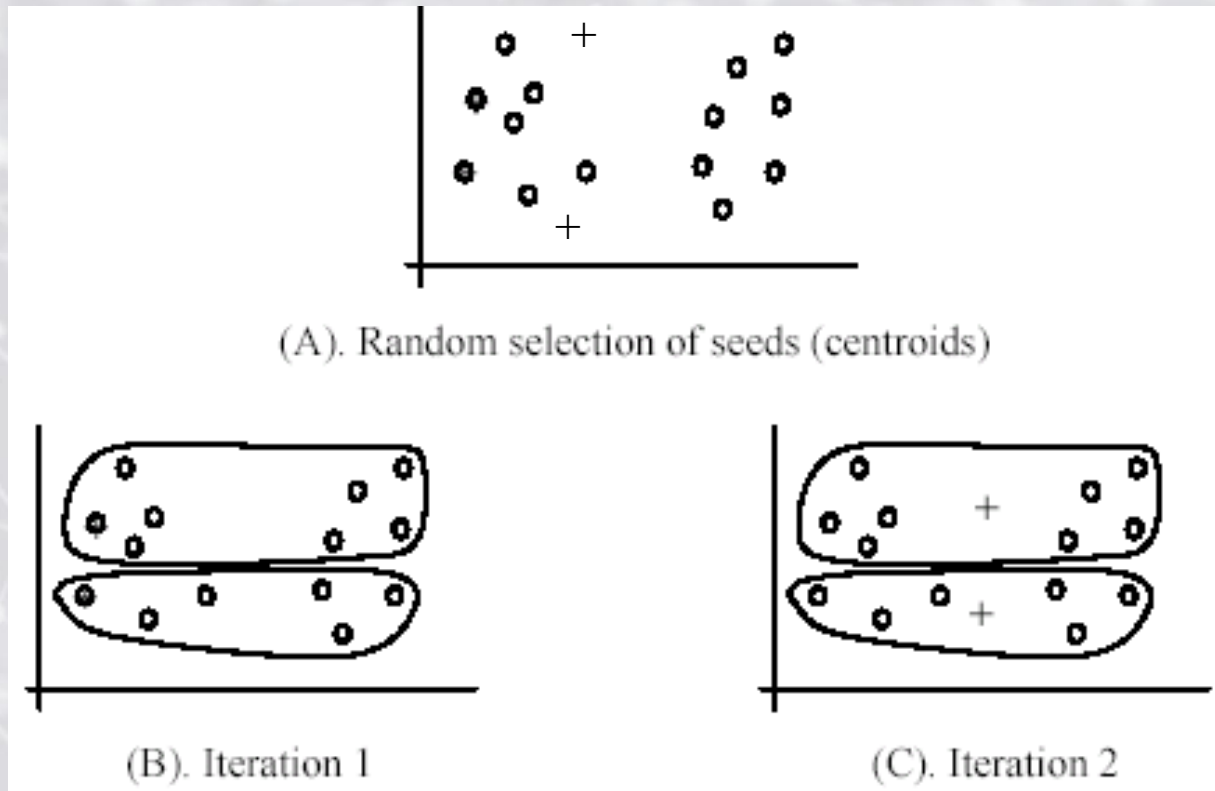


Dealing with outliers

- One method is to remove some data points in the clustering process that are much further away from the centroids than other data points.
 - Expensive
 - Not always a good idea!
- Another method is to perform random sampling. Since in sampling we only choose a small subset of the data points, the chance of selecting an outlier is very small.
 - Assign the rest of the data points to the clusters by distance or similarity comparison, or classification

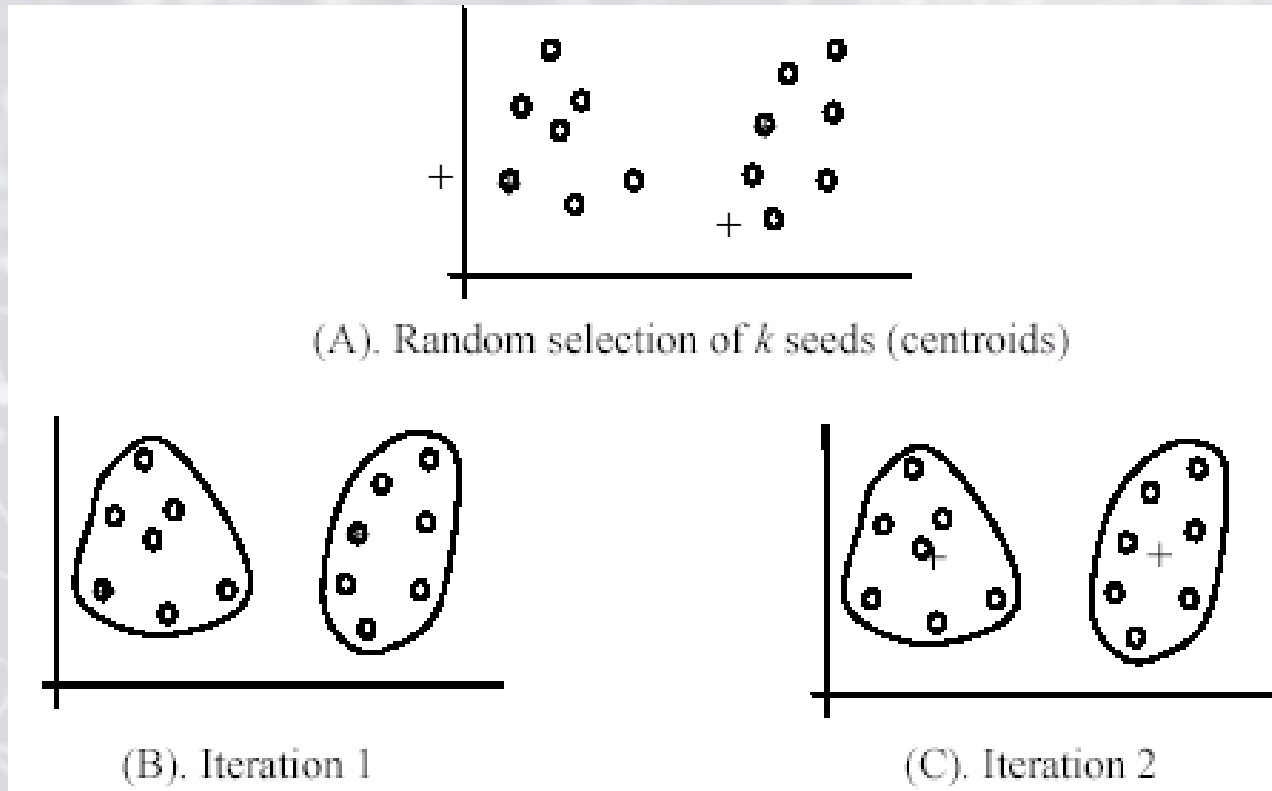
Issues for K -means (cont ...)

- The algorithm is sensitive to **initial seeds**.



Issues for K -means (cont ...)

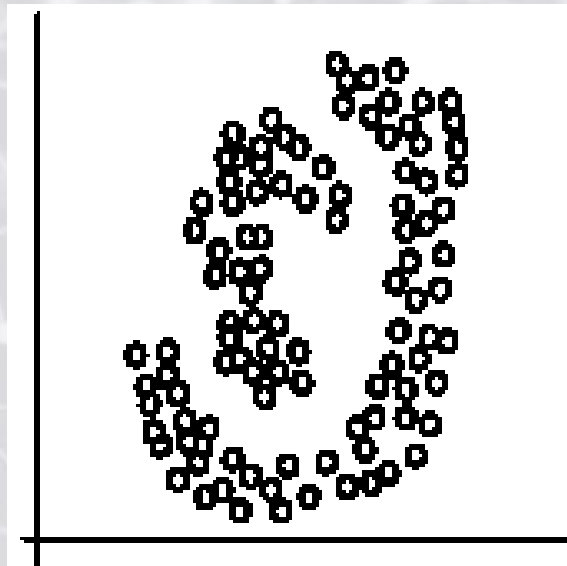
- If we use **different seeds**: good results



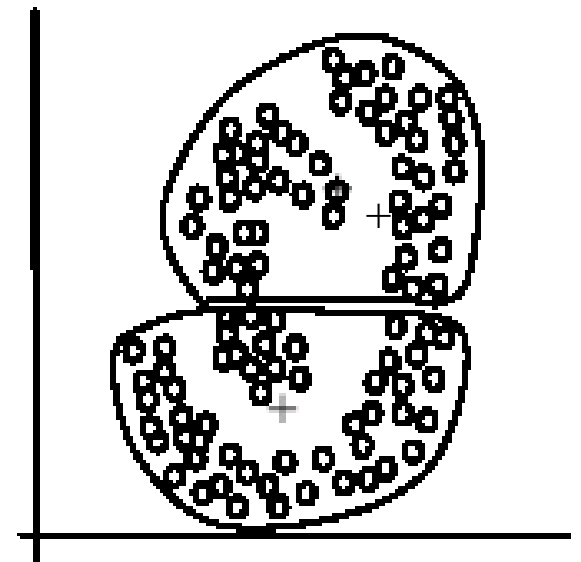
- Often we can improve k -means results by performing **several random restarts**.
- It is commonly helpful to use actual data values for the initial seeds.

Issues for K -means (cont ...)

- The K -means algorithm is not suitable for discovering clusters that are not hyper-ellipsoids (or hyper-spheres).



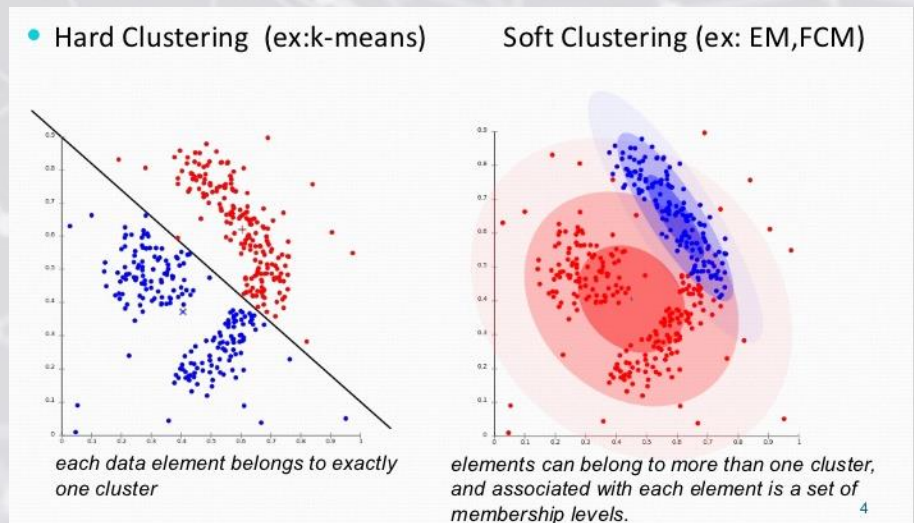
(A): Two natural clusters



(B): k -means clusters

Fuzzy c-means

- In non-fuzzy clustering (also known as hard clustering), data is divided into distinct clusters, where each data point can only belong to exactly one cluster. (cf., k-means from previous slides).
- In *fuzzy clustering* (also: soft clustering), data points can potentially belong to multiple clusters.
- Commonly, “**membership grades**” (i.e. class probabilities) are assigned to each of the data points. These membership grades indicate the degree to which data points belong to each cluster. Thus, points on the edge of a cluster, with lower membership grades, may be *in the cluster* to a lesser degree than points in the center of cluster.



Fuzzy c-means

- The **FCM** (*fuzzy c-means* (1973), as it is usually called) algorithm is very similar to the k-means algorithm:


Here is the basic idea:

- Choose a number of clusters: c (a hyperparameter).
- Initially assign coefficients randomly to each data point for being in the clusters (these are the *initial membership grades*).
- Repeat until the algorithm has converged/stopping condition:
 - (I) Compute the centroid for each cluster (m-step).
 - (II) For each data point, compute its coefficients/membership grades for being in the clusters (e-step).

Fuzzy c-means

Here is the basic idea:

- Choose a number of clusters: c (a hyperparameter).
- Initially assign coefficients randomly to each data point for being in the clusters (these are the *initial membership grades*).
- Repeat until the algorithm has converged/stopping condition:
 - (I) Compute the centroid for each cluster (m-step).
 - (II) For each data point, compute its coefficients/membership grades for being in the clusters (e-step).



$$\mathbf{c}_k = \frac{\sum_x w_k(\mathbf{x})^m \mathbf{x}}{\sum_x w_k(\mathbf{x})^m}$$


where $\mathbf{C} = \{\mathbf{c}_1, \dots, \mathbf{c}_c\}$ are the cluster centers, $W = w_{i,j} \in [0,1], i=1, \dots, n, j=1, \dots, c$, and each element w_{ij} tells the degree to which element \mathbf{x}_i belongs to cluster \mathbf{c}_j (i.e. the w 's are the *membership grades*); $m > 1$ is a hyperparameter known as the **fuzzifier parameter** which controls the amount of “fuzziness” in the partition.

Fuzzy c-means

Here is the basic idea:

- Choose a number of clusters: c (a hyperparameter).
- Initially assign coefficients randomly to each data point for being in the clusters (these are the *initial membership grades*).
- Repeat until the algorithm has converged/stopping condition:
 - (I) Compute the centroid for each cluster (m-step).
 - (II) For each data point, compute its coefficients/membership grades for being in the clusters (e-step).



$$\mathbf{c}_k = \frac{\sum_x w_k (\mathbf{x})^m \mathbf{x}}{\sum_x w_k (\mathbf{x})^m}$$



$$w_{ij} = \frac{1}{\sum_{k=1}^c \left(\frac{\|\mathbf{x}_i - \mathbf{c}_j\|}{\|\mathbf{x}_i - \mathbf{c}_k\|} \right)^{\frac{2}{m-1}}}$$

Fuzzy c-means

Here is the basic idea:

- Choose a number of clusters: c (a hyperparameter).
- Initially assign coefficients randomly to each data point for being in the clusters (these are the *initial membership grades*).
- Repeat until the algorithm has converged/stopping condition:
 - (I) Compute the centroid for each cluster (m-step).
 - (II) For each data point, compute its coefficients/membership grades for being in the clusters (e-step).


$$\mathbf{c}_k = \frac{\sum_x w_k (\mathbf{x})^m \mathbf{x}}{\sum_x w_k (\mathbf{x})^m}$$


$$w_{ij} = \frac{1}{\sum_{k=1}^c \left(\frac{\|\mathbf{x}_i - \mathbf{c}_j\|}{\|\mathbf{x}_i - \mathbf{c}_k\|} \right)^{\frac{2}{m-1}}}$$

(*) FCM aims to minimize the an objective function:

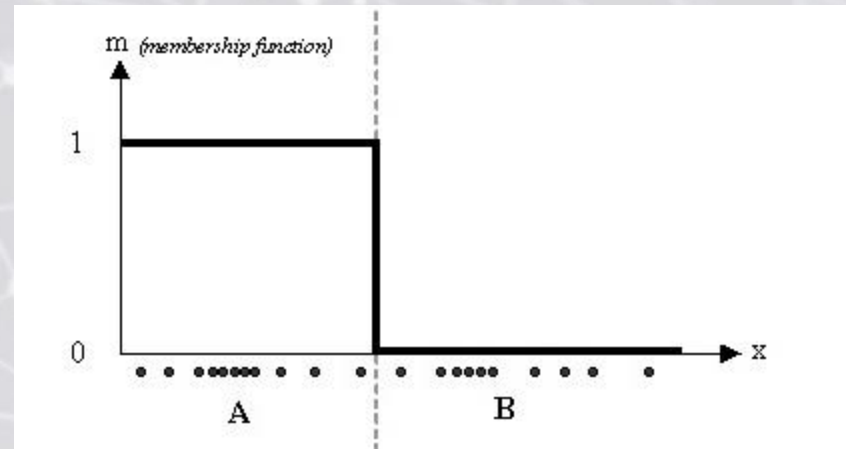
$$\arg \min_C \sum_{i=1}^n \sum_{j=1}^c w_{ij}^m \|\mathbf{x}_i - \mathbf{c}_j\|^2$$

Fuzzy c-means

- For example, consider a simple 1-d data set, where we want to determine a plausible clustering for 2 classes (A and B).



- Using “**hard k-means**” we associate each datum to a specific centroid; therefore the membership function looks like this:

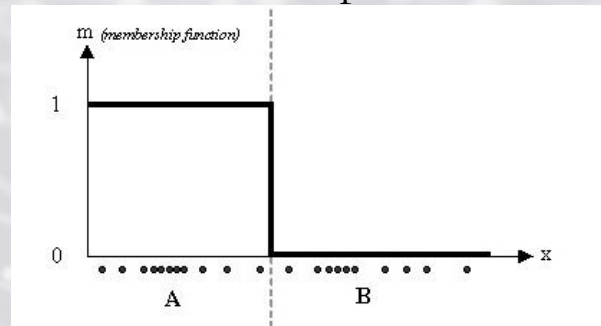


Fuzzy c-means

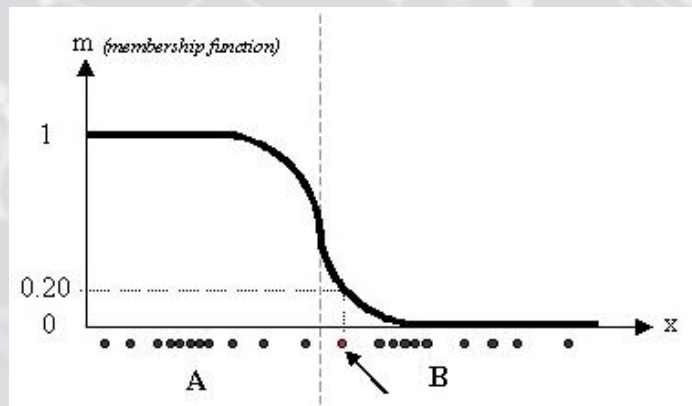
- For example, consider a simple 1-d data set, where we want to determine a plausible clustering for 2 classes (A and B).



- Using “**hard k-means**” we associate each datum to a specific centroid; therefore the membership function looks like this:

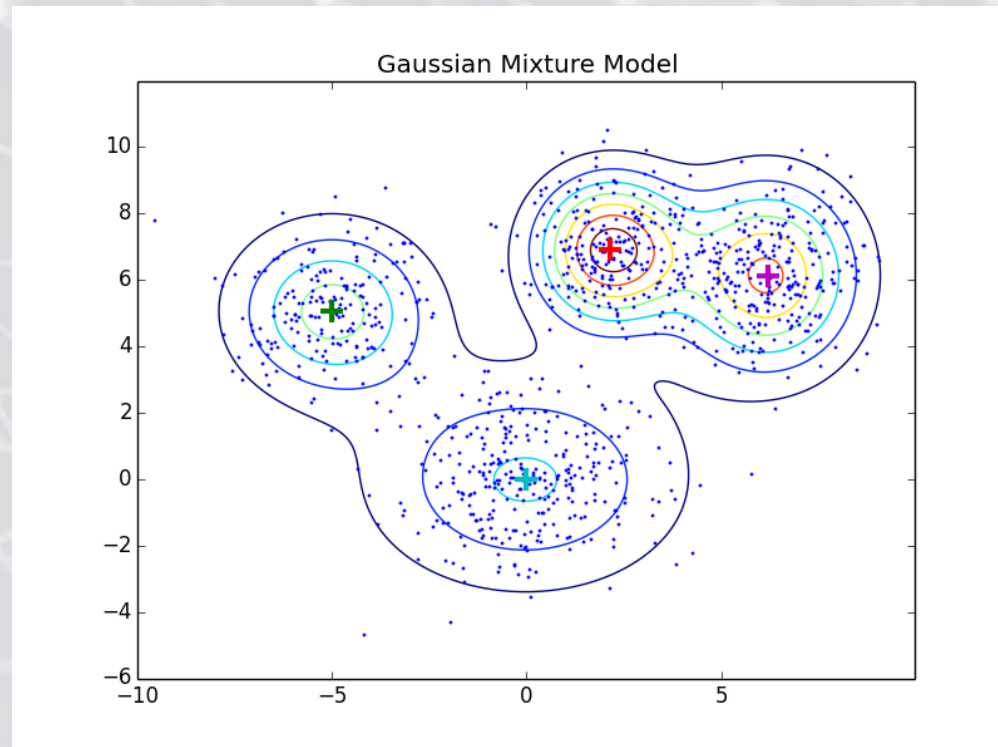


- In the **FCM approach**, instead, the same given datum does not belong exclusively to a well defined cluster. In this case, the membership function follows a smoother line to indicate that every datum may belong to several clusters with different values of the membership coefficient.



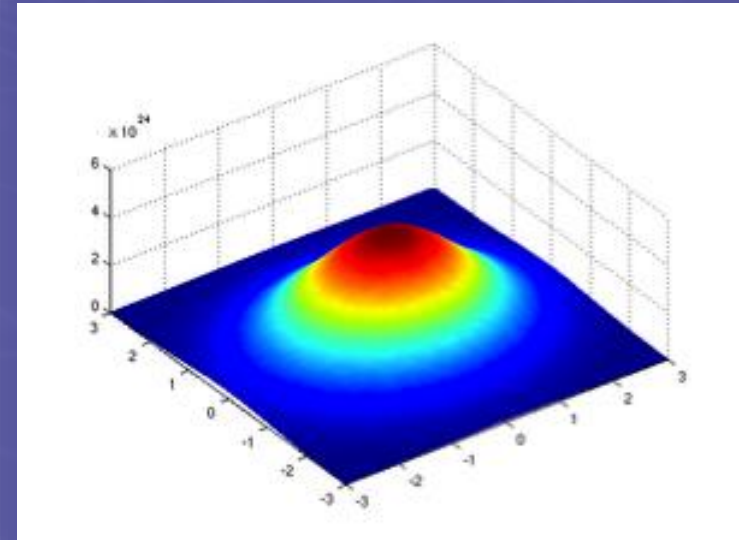
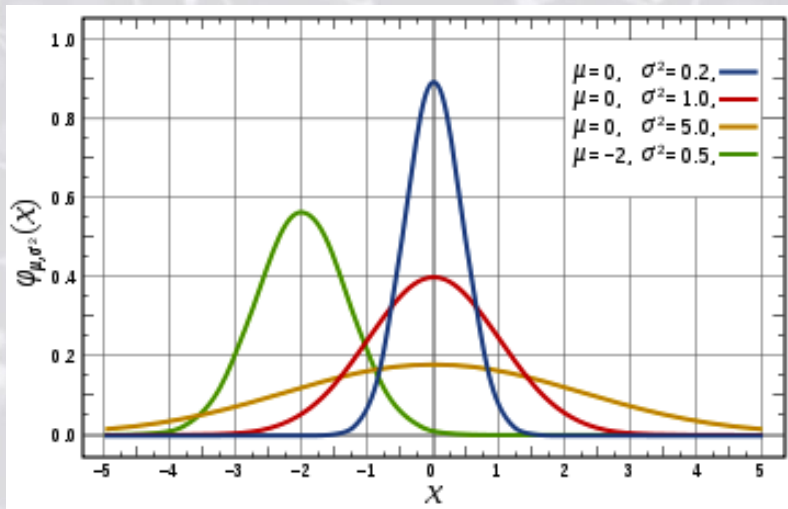
GMMs

- FCM was essentially, again, a “fuzzy” version of the k-means algorithm, where data points are assigned to each cluster with an associated probability/membership grade.
- An additional, commonly used soft clustering model is the GMM (**Gaussian mixture model**); with GMMs, we assume (*a priori*) that the clusters resemble tightly-packed balls (i.e. Gaussian distributions).

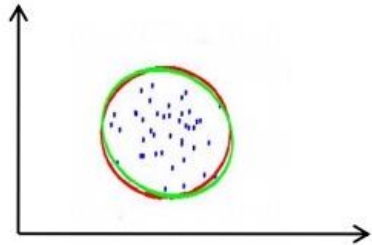


GMMs: Gaussian Distribution Review

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

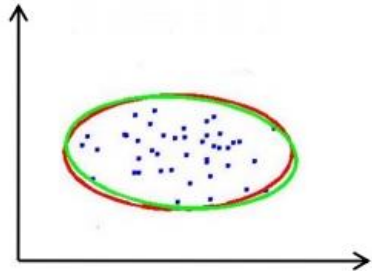


GMMs: Gaussian Distribution Review



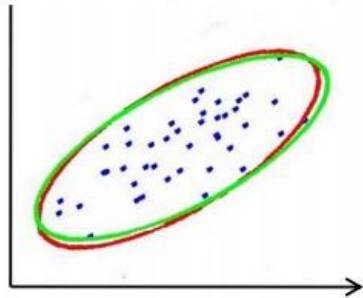
Covariance matrix $\Sigma =$

$$\begin{matrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{matrix}$$



Covariance matrix $\Sigma =$

$$\begin{matrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{matrix}$$



Covariance matrix $\Sigma =$

$$\begin{matrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{matrix}$$

Using different forms of covariance matrix allows for clusters of different shapes

GMMs

Main ideas for clustering using GMM:

(*) *Initialization*: given a data set, fix k , the number of clusters; initialize the mean (μ) and covariance matrices (Σ) for the k Gaussian clusters.

(*) Assign the data points to the k clusters (using a soft clustering) (**assignment step/E-step**)

(*) Update the parameters (i.e. μ , Σ) for each of the clusters. (**update step/M-step**)

...repeat until stopping condition/convergence

GMMs

Main ideas for clustering using GMM:

- (*) *Initialization*: given a data set, fix k , the number of clusters; initialize the mean (μ) and covariance matrices (Σ) for the k Gaussian clusters.
 - (*) Assign the data points to the k clusters (using a soft clustering) (**assignment step/E-step**)
 - (*) Update the parameters (i.e. μ , Σ) and prior class estimates ($P(C_i | x)$) (for each of the clusters). (**update step/M-step**)
- ...repeat until stopping condition/convergence

What makes this problem challenging? There are, ostensibly, **many unknowns!**

- (*) Strictly speaking, we don't know the cluster assignments nor any of the Gaussian distribution parameters.

GMMs

What makes this problem challenging? There are, ostensibly, **many unknowns!**

(*) Strictly speaking, we don't know the cluster assignments nor any of the Gaussian distribution parameters.

How can we simplify things?

A nice trick...Solve each subproblem separately!

- (1) For instance, to find the optimal class assignments for each datum, use the current approximations for the Gaussian parameters distributions (i.e. treat μ and Σ as known for each cluster, as well as each class prior) and compute the class posterior: $P(C_i | x)$ using Bayes' Rule.
- (2) Conversely, to find the optimal estimates for μ and Σ for each cluster, in addition to the class priors, use the current (soft) class posterior assignments and compute the MLE.

GMMs

- The GMM distribution can be written as a *linear superposition* of Gaussians in the form:

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

Each Gaussian density $N(\mathbf{x} | \boldsymbol{\mu}_k, \Sigma_k)$ is called a *component* of the mixture and has its own mean and covariance; π_k are called *mixing coefficients* and satisfy: $\sum_k \pi_k = 1$.

GMMs

- The GMM distribution can be written as a *linear superposition* of Gaussians in the form:

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

Each Gaussian density $N(\mathbf{x} | \boldsymbol{\mu}_k, \Sigma_k)$ is called a *component* of the mixture and has its own mean and covariance; π_k are called *mixing coefficients* and satisfy: $\sum_k \pi_k = 1$.

- Let us introduce a K-dimensional binary random variable \mathbf{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 (i.e. this is a **one-hot encoding**).

(*) The values of z_k therefore satisfy $z_k \in \{0,1\}$ and $\sum_k z_k = 1$.

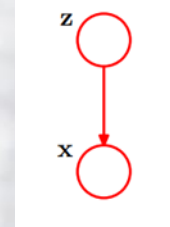
- We shall define the joint distribution $p(\mathbf{x}, \mathbf{z})$ in terms of a *marginal distribution* $p(\mathbf{z})$ and a *conditional distribution* $p(\mathbf{x} | \mathbf{z})$, corresponding to the following *graphical model*:

Graphical representation of a mixture model, in which the joint distribution is expressed in the form $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$.



GMMs

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



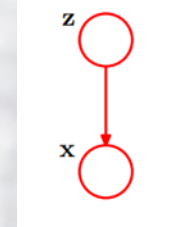
- The marginal distribution over \mathbf{z} is specified in terms of the mixing coefficients π_k such that: $p(z_k = 1) = \pi_k$, where the parameters $\{\pi_k\}$ satisfy: $0 \leq \pi_k \leq 1$ & $\sum_k \pi_k = 1$.

Because \mathbf{z} uses a 1-of- K representation, we can also write this distribution in the form:

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

GMMs

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



- The marginal distribution over z is specified in terms of the mixing coefficients π_k such that: $p(z_k = 1) = \pi_k$, where the parameters $\{\pi_k\}$ satisfy: $0 \leq \pi_k \leq 1$ & $\sum_k \pi_k = 1$.

Because \mathbf{z} uses a 1-of- K representation, we can also write this distribution in the form:

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

- Similarly, the conditional distribution of \mathbf{x} given a particular value for \mathbf{z} is a Gaussian:

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

Which can also be written in the form:

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

GMMs

- The joint distribution is given by $p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$ and the marginal distribution of \mathbf{x} is then obtained by summing the joint distribution over all possible states of \mathbf{z} to give:

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




marginalization

(*) Note that instead of using the marginal distribution $p(\mathbf{x})$ directly, the formula above allows us to work with the joint $p(\mathbf{x}, \mathbf{z})$, which will lead to significant simplifications, most notably through the introduction of the EM algorithm.

GMMs

- The joint distribution is given by $p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$ and the marginal distribution of \mathbf{x} is then obtained by summing the joint distribution over all possible states of \mathbf{z} to give:

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


marginalization

(*) Note that instead of using the marginal distribution $p(\mathbf{x})$ directly, the formula above allows us to work with the joint $p(\mathbf{x}, \mathbf{z})$, which will lead to significant simplifications, most notably through the introduction of the EM algorithm.

- An additional quantity of interest is the conditional probability of \mathbf{z} given \mathbf{x} . We shall use $\gamma(z_k)$ to denote $p(z_k=1|\mathbf{x})$, whose value can be ascertained via Bayes' Theorem as follows:

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

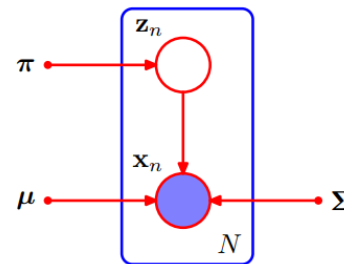
(*) We view π_k as the prior probability of $z_k = 1$, and the quantity $\gamma(z_k)$ as the corresponding posterior probability once we have observed \mathbf{x} ; note that $\gamma(z_k)$ is often referred to as the **responsibility** that component k takes for “explaining” the observation \mathbf{x} .

GMMs: MLE

- Suppose we have a data set of observations $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, and we wish to model this data using a mixture of Gaussians. We can represent this data set as an $N \times D$ matrix X in which the n th row is given by \mathbf{x}_n^T .
- If we assume the data points are drawn independently from the distribution, then we can express the Gaussian mixture model for this IID data set using the graphical representation shown; the *log likelihood* function is given by:

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

Graphical representation of a Gaussian mixture model for a set of N i.i.d. data points $\{\mathbf{x}_n\}$, with corresponding latent points $\{\mathbf{z}_n\}$, where $n = 1, \dots, N$.

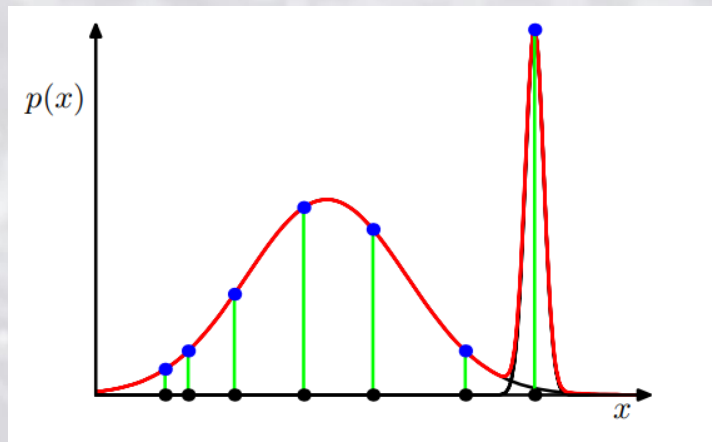


Q: Why do we use the *log likelihood* as opposed to simply the likelihood in practice for MLE calculations?

GMMs: MLE

- It is important to be aware that when using the MLE to render a GMM, (2) significant pathologies can frequently arise.

(1) The presence of *singularities*



(*) It is not uncommon for GMMs to yield singularities in which one (or more) of the Gaussian components collapses onto a single data point.

Why does this occur? The “collapsed” component will contribute an ever-increasing additive value to the log likelihood expression – the maximization of the log likelihood function for GMMs is therefore not a well-posed problem.

- Singularities provide another example of extreme over-fitting that can occur with the MLE approach. How can one remedy this situation? Be Bayesian! (See “*variational Bayes*” methods).

GMMs: MLE

- It is important to be aware that when using the MLE to render a GMM, (2) significant pathologies can frequently arise.

(2) *Identifiability*

(*) A further issue in finding MLE solutions occurs due to the fact that for any given MLE solution, a K -component mixture will have a total of $K!$ equivalent solutions corresponding to the $K!$ ways of assigning K sets of parameters to K components.

- This means that for any given point in the space of parameters values there will be a further $K!-1$ additional points all of which give rise to exactly the same distribution.
- Identifiability can be a serious issue when we wish to interpret the parameter values discovered by a model.

EM for GMMs

- The EM algorithm has broad applicability across ML. Subsequently we will give a more general treatment of the EM algorithm – for now we start with a relatively informal application in relation to GMMs.
- To begin, we take the derivatives of $\ln p(\mathbf{X} | \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma)$ with respect to the means $\boldsymbol{\mu}_k$ of the Gaussians components and set this equal to zero:

$$\frac{\partial}{\partial \boldsymbol{\mu}_k} \ln p(\mathbf{X} | \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) = \frac{\partial}{\partial \boldsymbol{\mu}_k} \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k N(\mathbf{x}_n | \boldsymbol{\mu}_k, \Sigma_k) \right\} = \sum_{n=1}^N \underbrace{\frac{\pi_k N(\mathbf{x}_n | \boldsymbol{\mu}_k, \Sigma_k)}{\sum_j \pi_j N(\mathbf{x}_n | \boldsymbol{\mu}_j, \Sigma_j)}}_{\gamma(z_{nk})} \Sigma_k^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_k) = 0$$

EM for GMMs

- The EM algorithm has broad applicability across ML. Subsequently we will give a more general treatment of the EM algorithm – for now we start with a relatively informal application in relation to GMMs.

(1) To begin, we take the derivatives of $\ln p(\mathbf{X} | \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma)$ with respect to the means $\boldsymbol{\mu}_k$ of the Gaussians components and set this equal to zero:

$$\frac{\partial}{\partial \boldsymbol{\mu}_k} \ln p(\mathbf{X} | \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) = \frac{\partial}{\partial \boldsymbol{\mu}_k} \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k N(\mathbf{x}_n | \boldsymbol{\mu}_k, \Sigma_k) \right\} = \sum_{n=1}^N \underbrace{\frac{\pi_k N(\mathbf{x}_n | \boldsymbol{\mu}_k, \Sigma_k)}{\sum_j \pi_j N(\mathbf{x}_n | \boldsymbol{\mu}_j, \Sigma_j)}}_{\gamma(z_{nk})} \Sigma_k^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_k) = 0$$

Solving yields:

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

(*) One can interpret the quantity N_k as the “effective number of points assigned to cluster k ”. Notice that $\boldsymbol{\mu}_k$ for the k th Gaussian component is obtained by taking a weighted mean of all the points in the data set, in which the weighting factor for data point \mathbf{x}_n is given by the posterior probability $\gamma(z_{nk})$ that component k was responsible for generating \mathbf{x}_n .

GMMs

(2) We set the derivative of $\ln p(\mathbf{X} | \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ with respect to $\boldsymbol{\Sigma}_k$ to zero, and solve, producing:

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

which is merely the covariance formula with each datum weighted by the corresponding posterior probability and with the denominator given by the effective number of points associated with the corresponding component.

GMMs

(2) We set the derivative of $\ln p(\mathbf{X} | \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ with respect to $\boldsymbol{\Sigma}_k$ to zero, and solve, producing:

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

which is merely the covariance formula with each datum weighted by the corresponding posterior probability and with the denominator given by the effective number of points associated with the corresponding component.

(3) Lastly, we maximize $\ln p(\mathbf{X} | \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ wrt to the mixing coefficients π_k . Here we must take into account the constraint: $\sum_k \pi_k = 1$.

Q: How do we conventionally solve a constrained optimization question?

GMMs

(2) We set the derivative of $\ln p(\mathbf{X} | \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ with respect to $\boldsymbol{\Sigma}_k$ to zero, and solve, producing:

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

which is merely the covariance formula with each datum weighted by the corresponding posterior probability and with the denominator given by the effective number of points associated with the corresponding component.

(3) Lastly, we maximize $\ln p(\mathbf{X} | \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ wrt to the mixing coefficients π_k . Here we must take into account the constraint: $\sum_k \pi_k = 1$.

Q: How do we conventionally solve a constrained optimization question?

A: *Langrange multipliers.*

GMMs

(2) We set the derivative of $\ln p(\mathbf{X} | \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma)$ with respect to Σ_k to zero, and solve, producing:

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

which is merely the covariance formula with each datum weighted by the corresponding posterior probability and with the denominator given by the effective number of points associated with the corresponding component.

(3) Lastly, we maximize $\ln p(\mathbf{X} | \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma)$ wrt to the mixing coefficients π_k . Here we must take into account the constraint: $\sum_k \pi_k = 1$.

According to the method of *Langrange multipliers*, we maximize the following quantity:

$$\ln p(\mathbf{X} | \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) + \lambda \left(\sum_{k=1}^K \pi_k - 1 \right) \rightarrow \sum_{n=1}^N \frac{\pi_k N(\mathbf{x}_n | \boldsymbol{\mu}_k, \Sigma_k)}{\sum_{j=1}^K \pi_j N(\mathbf{x}_n | \boldsymbol{\mu}_j, \Sigma_j)} + \lambda = 0$$

GMMs

(3) Lastly, we maximize $\ln p(X | \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma)$ wrt to the mixing coefficients π_k . Here we must take into account the constraint: $\sum_k \pi_k = 1$.

According to the method of *Langrange multipliers*, we maximize the following quantity:

$$\ln p(X | \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) + \lambda \left(\sum_{k=1}^K \pi_k - 1 \right) \rightarrow \sum_{n=1}^N \frac{\pi_k N(\mathbf{x}_n | \boldsymbol{\mu}_k, \Sigma_k)}{\sum_{j=1}^K \pi_j N(\mathbf{x}_n | \boldsymbol{\mu}_j, \Sigma_j)} + \lambda = 0$$

(*) Multiplying both sides by π_k and summing over k gives $\lambda = -N$; we then eliminate λ , rearrange and obtain:

$$\pi_k = \frac{N_k}{N}$$

So that the mixing coefficient of the k th component is given by the average responsibility which that component takes for explaining the data points.

GMMs

In summary, we have the GMM MLE parameter formulae:

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n \text{ with } N_k = \sum_{n=1}^N \gamma(z_{nk})$$

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

$$\pi_k = \frac{N_k}{N}$$

Each parameter estimate is used as an iterative update in the EM algorithm. First we choose some initial values for the means, covariances and mixing coefficients.

Then we alternate between the two update steps:

(1) E-step: (expectation) here we use the current values for the parameters to evaluate the posterior probabilities/responsibilities.

(2) M-step: (maximization) here we update the parameter values for the means and

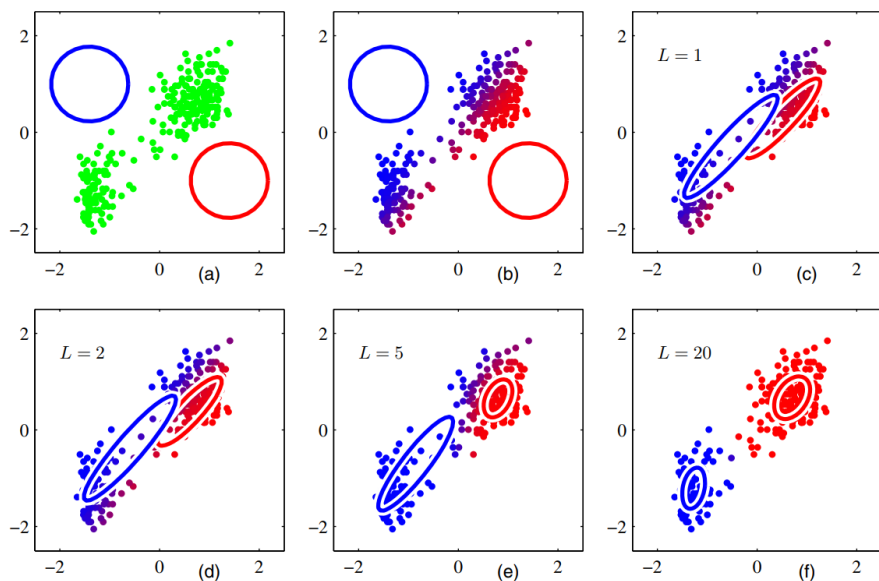
GMMs: Summary

- Each parameter estimate is used as an iterative update in the EM algorithm. First we choose some initial values for the means, covariances and mixing coefficients.

Then we alternate between the two update steps:

(1) E-step: (expectation) here we use the current values for the parameters to evaluate the posterior probabilities/responsibilities.

(2) M-step: (maximization) here we update the parameter values for the means and covariances with the responsibilities kept constant.

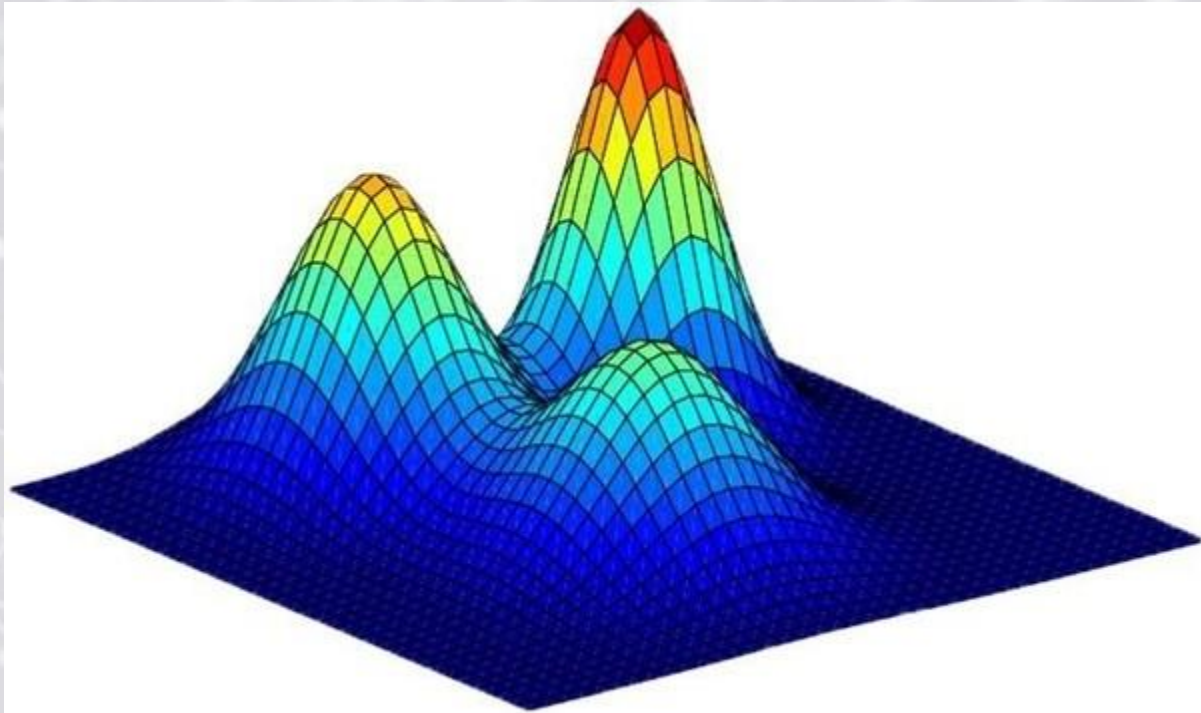


(*) Note that the EM algorithm takes many more iterations to reach (approximate) convergence than k-means in general.

It is consequently common practice to run the k-means algorithm in order to find a suitable initialization for a Gaussian mixture model that is subsequently adapted using EM; in addition, initial covariance matrices are often based on sample covariances of cluster generated by k-means; likewise mixing coefficients can be initialized similarly.

GMMs

- Demo: <https://lukapopijac.github.io/gaussian-mixture-model/>



GMM: Summary

EM for Gaussian Mixtures

Given a Gaussian mixture model, the goal is to maximize the likelihood function with respect to the parameters (comprising the means and covariances of the components and the mixing coefficients).

1. 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.** Evaluate the responsibilities using the current parameter values

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}_n | \mu_j, \Sigma_j)}. \quad (9.23)$$

3. **M step.** Re-estimate the parameters using the current responsibilities

$$\mu_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n \quad (9.24)$$

$$\Sigma_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) (\mathbf{x}_n - \mu_k^{\text{new}}) (\mathbf{x}_n - \mu_k^{\text{new}})^T \quad (9.25)$$

$$\pi_k^{\text{new}} = \frac{N_k}{N} \quad (9.26)$$

where

$$N_k = \sum_{n=1}^N \gamma(z_{nk}). \quad (9.27)$$

4. Evaluate the log likelihood

$$\ln p(\mathbf{X} | \mu, \Sigma, \pi) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \mu_k, \Sigma_k) \right\} \quad (9.28)$$

and check for convergence of either the parameters or the log likelihood. If the convergence criterion is not satisfied return to step 2.

- From the *Bishop* text, pp.438-439.

EM: An Alternative View

- Conventionally, the EM algorithm is considered in relation to **latent variables** (i.e. *hidden variables*). We now review the EM algorithm in this slightly more abstract setting, and then relate it concretely to the k-means algorithm for the purposes of review.

(*) The goal of the EM algorithm is to find the MLE solutions for models with latent variables.

- We denote the set of all observed data by \mathbf{X} , in which the n th row represents \mathbf{x}_n^T , and similarly we denote the set of all latent variables by \mathbf{Z} corresponding with row \mathbf{z}_n^T .

EM: An Alternative View

- Conventionally, the EM algorithm is considered in relation to **latent variables** (i.e. *hidden variables*). We now review the EM algorithm in this slightly more abstract setting, and then relate it concretely to the k-means algorithm for the purposes of review.

(*) The goal of the EM algorithm is to find the MLE solutions for models with latent variables.

- We denote the set of all observed data by \mathbf{X} , in which the n th row represents \mathbf{x}_n^T , and similarly we denote the set of all latent variables by \mathbf{Z} corresponding with row \mathbf{z}_n^T .

The set of all model parameters is denoted by $\boldsymbol{\theta}$, and so the *log likelihood* function is:

$$\ln p(\mathbf{X} | \boldsymbol{\theta}) = \ln \left\{ \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta}) \right\}$$

EM: An Alternative View

- The *log likelihood* function is:

$$\ln p(\mathbf{X} | \boldsymbol{\theta}) = \ln \left\{ \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta}) \right\}$$

A key observation is that the summation over the latent variables appears inside the log expression. We'll call the set $\{\mathbf{X}, \mathbf{Z}\}$ the complete data set, and we will likewise refer to the actual observed data \mathbf{X} as incomplete.

- In practice we are not given the complete data set $\{\mathbf{X}, \mathbf{Z}\}$ – but only the incomplete data \mathbf{X} . Our state of knowledge of the values of the latent variables in \mathbf{Z} is given only by the posterior distribution $p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})$.

(*) Because we cannot use the complete-data log likelihood, we consider instead its expected value under the posterior distribution of the latent variable (this corresponds with the E-step); in the subsequent M-step, we maximize this expectation.

EM: An Alternative View

- In the E-step, we use the current parameter values θ^{old} to find the posterior distribution of the latent variables given by $p(\mathbf{Z} | \mathbf{X}, \theta^{\text{old}})$. We then use this posterior distribution to find the expectation of the complete-data log likelihood evaluated for some general parameter value θ .

Define $Q(\theta, \theta^{\text{old}})$:

$$Q(\theta, \theta^{\text{old}}) = \sum_{\mathbf{Z}} p(\mathbf{Z} | \mathbf{X}, \theta^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z} | \theta).$$

EM: An Alternative View

- In the E-step, we use the current parameter values θ^{old} to find the posterior distribution of the latent variables given by $p(\mathbf{Z} | \mathbf{X}, \theta^{\text{old}})$. We then use this posterior distribution to find the expectation of the complete-data log likelihood evaluated for some general parameter value θ .

Define $Q(\theta, \theta^{\text{old}})$:

$$Q(\theta, \theta^{\text{old}}) = \sum_{\mathbf{Z}} p(\mathbf{Z} | \mathbf{X}, \theta^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z} | \theta).$$

In the M-step, we determine the revised parameter estimate θ^{new} by maximizing this function:

$$\theta^{\text{new}} = \arg \max_{\theta} Q(\theta, \theta^{\text{old}})$$

EM: An Alternative View

The General EM Algorithm

Given a joint distribution $p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$ over observed variables \mathbf{X} and latent variables \mathbf{Z} , governed by parameters $\boldsymbol{\theta}$, the goal is to maximize the likelihood function $p(\mathbf{X}|\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$.

1. Choose an initial setting for the parameters $\boldsymbol{\theta}^{\text{old}}$.

2. **E step** Evaluate $p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}})$.

3. **M step** Evaluate $\boldsymbol{\theta}^{\text{new}}$ given by

$$\boldsymbol{\theta}^{\text{new}} = \arg \max_{\boldsymbol{\theta}} \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) \quad (9.32)$$

where

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}). \quad (9.33)$$

4. Check for convergence of either the log likelihood or the parameter values. If the convergence criterion is not satisfied, then let

$$\boldsymbol{\theta}^{\text{old}} \leftarrow \boldsymbol{\theta}^{\text{new}} \quad (9.34)$$

and return to step 2.

EM: k-means Revisited

- Comparison of the k-means algorithm with the EM algorithm for Gaussian mixtures shows that there is a close similarity.
- Whereas the k-means algorithm performs a *hard assignment* of data points to clusters, in which each data point is associated uniquely with one cluster, the EM algorithm makes a *soft assignment* based on the posterior probabilities.

(*) In fact, we can derive the k-means algorithm as a particular limit of EM for Gaussian mixtures.

EM: k-means Revisited

- Consider a Gaussian mixture model in which the covariance matrices of the mixture components are given by an *isotropic spherical covariance matrix*: $\varepsilon \mathbf{I}$, where ε is a variance parameter that is shared by all of the components, so that:

$$p(\mathbf{x} | \boldsymbol{\mu}_k, \Sigma_k) = \frac{1}{(2\pi\varepsilon)^{M/2}} \exp \left\{ -\frac{1}{2} \|\mathbf{x} - \boldsymbol{\mu}_k\|^2 \right\}$$

- Consider the EM algorithm for a mixture of K Gaussians of this form where we treat ε as a fixed constant, instead of a parameter to be re-estimated.
- Recall the previous definition for the posterior probabilities (i.e. *responsibilities*) for a particular data point is given by:

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




$$\gamma(z_{nk}) = \frac{\pi_k \exp \left\{ -\|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2 / 2\varepsilon \right\}}{\sum_j \pi_j \exp \left\{ -\|\mathbf{x}_n - \boldsymbol{\mu}_j\|^2 / 2\varepsilon \right\}}$$

EM: k-means Revisited

$$\gamma(z_{nk}) = \frac{\pi_k \exp\left\{-\|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2 / 2\varepsilon\right\}}{\sum_j \pi_j \exp\left\{-\|\mathbf{x}_n - \boldsymbol{\mu}_j\|^2 / 2\varepsilon\right\}}$$

- If we consider the limit $\varepsilon \rightarrow 0$, we see that in the denominator, the term for which $\|\mathbf{x}_n - \boldsymbol{\mu}_j\|^2$ is smallest will go to zero more slowly, and hence the responsibilities $\gamma(z_{nk})$ for the data point \mathbf{x}_n all go to zero except for the j term, for which the responsibility $\gamma(z_{nj})$ will go to one (this holds independently of the π_k , so long as none are zero).
- Thus, in the limit, we obtain a hard assignment of data points to clusters, just as in the k-means algorithm so that $\gamma(z_{nk}) \rightarrow r_{nk}$ (where r_{nk} is the binary hard assignment of the cluster with closest centroid to the given datum).


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

EM: k-means Revisited

$$\gamma(z_{nk}) = \frac{\pi_k \exp\left\{-\|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2 / 2\varepsilon\right\}}{\sum_j \pi_j \exp\left\{-\|\mathbf{x}_n - \boldsymbol{\mu}_j\|^2 / 2\varepsilon\right\}}$$

$$\lim_{\varepsilon \rightarrow 0} \gamma(z_{nk}) = r_{nk}$$

- In the limit $\varepsilon \rightarrow 0$, the EM estimation for μ_k :

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n \text{ with } N_k = \sum_{n=1}^N \gamma(z_{nk})$$

reduces to the familiar k-means estimate:

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



EM: k-means Revisited

$$\gamma(z_{nk}) = \frac{\pi_k \exp\left\{-\|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2 / 2\varepsilon\right\}}{\sum_j \pi_j \exp\left\{-\|\mathbf{x}_n - \boldsymbol{\mu}_j\|^2 / 2\varepsilon\right\}}$$

$$\lim_{\varepsilon \rightarrow 0} \gamma(z_{nk}) = r_{nk}$$

- In the limit $\varepsilon \rightarrow 0$, the EM estimation for μ_k :

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n \text{ with } N_k = \sum_{n=1}^N \gamma(z_{nk})$$

reduces to the familiar k-means estimate:

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



- Similarly, the re-estimation of the *mixing coefficients*, π_k , simply resets the value of π_k to be equal to the fraction of data points assigned to cluster k .

$$\pi_k = \frac{N_k}{N}$$

EM: k-means Revisited

- Lastly, in the limit $\varepsilon \rightarrow 0$, the expected complete-data log likelihood, becomes:

$$E_Z \left[\ln p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\mu}, \Sigma, \boldsymbol{\pi}) \right] = \sum_{n=1}^N \sum_{k=1}^K \gamma(z_{nk}) \left\{ \ln \pi_k + \ln N(\mathbf{x}_n | \boldsymbol{\mu}_k, \Sigma_k) \right\} \rightarrow -\frac{1}{2} \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2 + \text{const.}$$

Thus we can see that in this limit, maximizing the expected complete-data log likelihood is equivalent to minimizing J for the k-means algorithm.



$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

In summary: we have shown that k-means represents a special case of the EM algorithm applied to GMMs in the case of an isotropic spherical covariance matrix $\varepsilon \mathbf{I}$ taken in limit $\varepsilon \rightarrow 0$.

EM: General Case

- EM is a general technique for finding MLE solutions for probabilistic models with latent variables. We now take up the EM algorithm in the most general case.
- Consider a probabilistic model with observed variables \mathbf{X} and hidden variables given by \mathbf{Z} (for simplicity we assume \mathbf{Z} is discrete). The joint distribution $p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})$ is governed by a set of parameters denoted $\boldsymbol{\theta}$. Our goal is to maximize the likelihood function given by:

$$\ln p(\mathbf{X} | \boldsymbol{\theta}) = \ln \left\{ \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta}) \right\}$$

EM: General Case

- EM is a general technique for finding MLE solutions for probabilistic models with latent variables. We now take up the EM algorithm in the most general case.

Consider a probabilistic model with observed variables \mathbf{X} and hidden variables given by \mathbf{Z} (for simplicity we assume \mathbf{Z} is discrete). The joint distribution $p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})$ is governed by a set of parameters denoted $\boldsymbol{\theta}$. Our goal is to maximize the likelihood function given by:

$$\ln p(\mathbf{X} | \boldsymbol{\theta}) = \ln \left\{ \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta}) \right\}$$

We shall suppose, naturally, that the direct optimization of $p(\mathbf{X} | \boldsymbol{\theta})$ is difficult (e.g. the case in which class labels are unknown), but that the optimization of the complete-data likelihood function $p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})$ is significantly easier.

EM: General Case

Goal of EM: Maximize Likelihood $\longrightarrow \ln p(\mathbf{X} | \boldsymbol{\theta}) = \ln \left\{ \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta}) \right\}$

- Next we introduce a distribution $q(\mathbf{Z})$ defined over the latent variables, and we observe that, for any choice of $q(\mathbf{Z})$, the following decomposition holds:

$$\ln p(\mathbf{X} | \boldsymbol{\theta}) = L(q, \theta) + KL(q \| p)$$

Where we define:

$$L(q, \theta) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})}{q(\mathbf{Z})} \right\}$$

$$KL(q \| p) = - \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})} \right\}$$

EM: General Case

- To verify the decomposition: $\ln p(\mathbf{X} | \boldsymbol{\theta}) = L(q, \boldsymbol{\theta}) + KL(q \| p)$,

we first utilize the **product rule of probability**
to get:

$$L(q, \boldsymbol{\theta}) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})}{q(\mathbf{Z})} \right\}$$

$$KL(q \| p) = - \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})} \right\}$$

$$\ln p(\mathbf{Z}, \mathbf{X} | \boldsymbol{\theta}) = \ln p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}) + \ln p(\mathbf{X} | \boldsymbol{\theta})$$

which we then substitute into the expression for $L(q, \boldsymbol{\theta})$. This gives rise to (2) terms, one of which cancels $KL(q \| p)$ while the other gives the required log likelihood $\ln p(\mathbf{X} | \boldsymbol{\theta})$ after noting that $q(\mathbf{Z})$ is a normalized distribution.

Let's show this in more detail...

EM: General Case

- To verify the decomposition: $\ln p(\mathbf{X} | \boldsymbol{\theta}) = L(q, \theta) + KL(q \| p)$,

we first utilize the **product rule of probability**

to get:

$$\ln p(\mathbf{Z}, \mathbf{X} | \boldsymbol{\theta}) = \ln p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}) + \ln p(\mathbf{X} | \boldsymbol{\theta})$$

$$L(q, \theta) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})}{q(\mathbf{Z})} \right\}$$

$$KL(q \| p) = - \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})} \right\}$$

$$L(q, \theta) + KL(q \| p) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})}{q(\mathbf{Z})} \right\} - \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})} \right\}$$

EM: General Case

- To verify the decomposition: $\ln p(\mathbf{X}|\boldsymbol{\theta}) = L(q, \theta) + KL(q \| p)$,

we first utilize the **product rule of probability**

to get:

$$\ln p(\mathbf{Z}, \mathbf{X} | \boldsymbol{\theta}) = \ln p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}) + \ln p(\mathbf{X} | \boldsymbol{\theta})$$

$$L(q, \theta) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})}{q(\mathbf{Z})} \right\}$$

$$KL(q \| p) = - \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})} \right\}$$

$$L(q, \theta) + KL(q \| p) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})}{q(\mathbf{Z})} \right\} - \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})} \right\}$$

$$= \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}) p(\mathbf{X} | \boldsymbol{\theta})}{q(\mathbf{Z})} \right\} - \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})} \right\} \quad \text{Why?}$$

EM: General Case

- To verify the decomposition: $\ln p(\mathbf{X} | \boldsymbol{\theta}) = L(q, \boldsymbol{\theta}) + KL(q \| p)$,

we first utilize the **product rule of probability**

to get:

$$\ln p(\mathbf{Z}, \mathbf{X} | \boldsymbol{\theta}) = \ln p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}) + \ln p(\mathbf{X} | \boldsymbol{\theta})$$

$$L(q, \boldsymbol{\theta}) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})}{q(\mathbf{Z})} \right\}$$

$$KL(q \| p) = - \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})} \right\}$$

$$L(q, \boldsymbol{\theta}) + KL(q \| p) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})}{q(\mathbf{Z})} \right\} - \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})} \right\}$$

$$= \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}) p(\mathbf{X} | \boldsymbol{\theta})}{q(\mathbf{Z})} \right\} - \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})} \right\}$$

By product rule

$$= \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})} \right\} + \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln p(\mathbf{X} | \boldsymbol{\theta}) - \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})} \right\}$$

EM: General Case

- To verify the decomposition: $\ln p(\mathbf{X}|\boldsymbol{\theta}) = L(q, \theta) + KL(q \parallel p)$,

we first utilize the **product rule of probability**

to get:

$$L(q, \theta) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})}{q(\mathbf{Z})} \right\}$$

$$KL(q \parallel p) = - \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})} \right\}$$

$$\ln p(\mathbf{Z}, \mathbf{X} | \boldsymbol{\theta}) = \ln p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}) + \ln p(\mathbf{X} | \boldsymbol{\theta})$$

$$L(q, \theta) + KL(q \parallel p) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})}{q(\mathbf{Z})} \right\} - \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})} \right\}$$

$$= \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}) p(\mathbf{X} | \boldsymbol{\theta})}{q(\mathbf{Z})} \right\} - \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})} \right\}$$

By product rule

$$= \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{\cancel{p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})}}{q(\mathbf{Z})} \right\} + \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln p(\mathbf{X} | \boldsymbol{\theta}) - \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{\cancel{p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})}}{q(\mathbf{Z})} \right\}$$

$$= \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln p(\mathbf{X} | \boldsymbol{\theta}) = \ln p(\mathbf{X} | \boldsymbol{\theta})$$

Why?

EM: General Case

- To verify the decomposition: $\ln p(\mathbf{X} | \boldsymbol{\theta}) = L(q, \boldsymbol{\theta}) + KL(q \| p)$,

we first utilize the **product rule of probability**

to get:

$$L(q, \boldsymbol{\theta}) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})}{q(\mathbf{Z})} \right\}$$

$$KL(q \| p) = - \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})} \right\}$$

$$\ln p(\mathbf{Z}, \mathbf{X} | \boldsymbol{\theta}) = \ln p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}) + \ln p(\mathbf{X} | \boldsymbol{\theta})$$

$$(L(q, \boldsymbol{\theta}) + KL(q \| p)) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})}{q(\mathbf{Z})} \right\} - \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})} \right\}$$

$$= \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}) p(\mathbf{X} | \boldsymbol{\theta})}{q(\mathbf{Z})} \right\} - \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})} \right\}$$

$$= \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{\cancel{p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})}}{q(\mathbf{Z})} \right\} + \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln p(\mathbf{X} | \boldsymbol{\theta}) - \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{\cancel{p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})}}{q(\mathbf{Z})} \right\}$$

$$= \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln p(\mathbf{X} | \boldsymbol{\theta}) = \ln p(\mathbf{X} | \boldsymbol{\theta})$$

This verifies the decomposition of the likelihood of the observed data

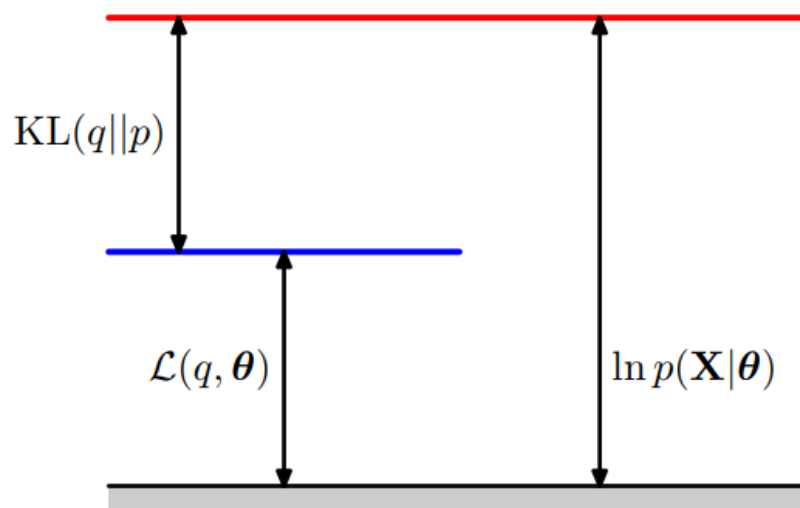
EM: General Case

$$\ln p(\mathbf{X} | \boldsymbol{\theta}) = L(q, \boldsymbol{\theta}) + KL(q || p)$$

- We note that $KL(q || p)$ is the KL-divergence between $q(\mathbf{Z})$ (the distribution over the hidden variables) and the posterior distribution $p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})$. Recall that the K-L divergence satisfies: $KL(q || p) \geq 0$ with equality *iff* $q(\mathbf{Z}) = p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})$.

(*) It therefore follows that $L(q, \boldsymbol{\theta}) \leq \ln p(\mathbf{X} | \boldsymbol{\theta})$ – which is to say that $L(q, \boldsymbol{\theta})$ forms a lower bound for $\ln p(\mathbf{X} | \boldsymbol{\theta})$.

Illustration of the decomposition given by (9.70), which holds for any choice of distribution $q(\mathbf{Z})$. Because the Kullback-Leibler divergence satisfies $KL(q || p) \geq 0$, we see that the quantity $\mathcal{L}(q, \boldsymbol{\theta})$ is a lower bound on the log likelihood function $\ln p(\mathbf{X} | \boldsymbol{\theta})$.



EM: General Case

$$\ln p(\mathbf{X}|\boldsymbol{\theta}) = L(q, \theta) + KL(q \| p)$$

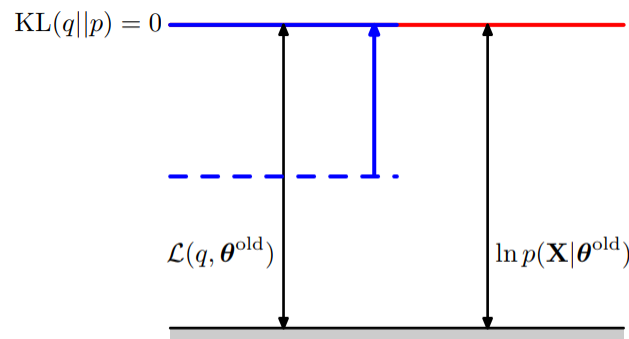
- The EM algorithm is a two-stage iterative optimization technique for finding MLE solutions. Let's demonstrate that decomposition above confirms that EM does in fact maximize the log likelihood.
- Suppose that the current value of the parameter vector is θ^{old} . In the E-step, the lower bound $L(q, \theta^{\text{old}})$ is maximized with respect to $q(\mathbf{Z})$ while holding θ^{old} fixed.

EM: General Case

$$\ln p(\mathbf{X} | \boldsymbol{\theta}) = L(q, \boldsymbol{\theta}) + KL(q \| p)$$

- The EM algorithm is a two-stage iterative optimization technique for finding MLE solutions. Let's demonstrate that decomposition above confirms that EM does in fact maximize the log likelihood.
- Suppose that the current value of the parameter vector is $\boldsymbol{\theta}^{\text{old}}$. In the E-step, the lower bound $L(q, \boldsymbol{\theta}^{\text{old}})$ is maximized with respect to $q(\mathbf{Z})$ while holding $\boldsymbol{\theta}^{\text{old}}$ fixed.
- The solution to this maximization problem is easily seen by noting that the value of $\ln p(\mathbf{X} | \boldsymbol{\theta}^{\text{old}})$ does not depend on $q(\mathbf{Z})$ and so the largest value of $L(q, \boldsymbol{\theta}^{\text{old}})$ will occur when the K-L divergence vanishes (i.e. when $q(\mathbf{Z}) = p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}^{\text{old}})$). In this case the lower bound will equal the log likelihood, as shown below.

Illustration of the E step of the EM algorithm. The q distribution is set equal to the posterior distribution for the current parameter values $\boldsymbol{\theta}^{\text{old}}$, causing the lower bound to move up to the same value as the log likelihood function, with the KL divergence vanishing.



EM: General Case

$$\ln p(\mathbf{X} | \boldsymbol{\theta}) = L(q, \theta) + KL(q \| p)$$

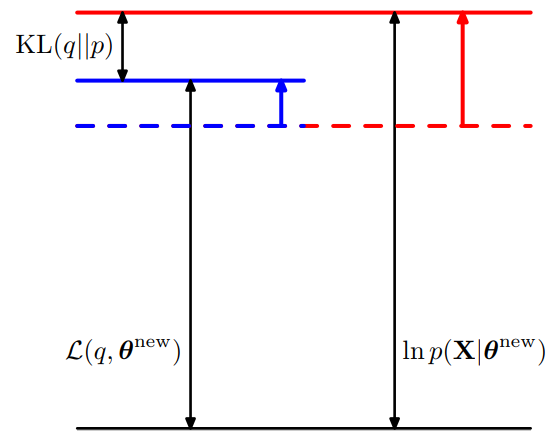
- In the M-step, the distribution $q(Z)$ is held fixed and the lower bound $L(q, \theta)$ is maximized with respect to θ to give some new value θ^{new} . This will cause the lower bound L to increase (unless it is already at a maximum), which will necessarily cause the corresponding log likelihood function to increase.
- Because the distribution q is determined using the old parameter values rather than the new values and is held fixed during the M-step, it will not equal the new posterior distribution $p(Z | X, \theta^{\text{new}})$, and hence there will be a non-zero KL divergence.

EM: General Case

$$\ln p(\mathbf{X} | \boldsymbol{\theta}) = L(q, \boldsymbol{\theta}) + KL(q \| p)$$

- In the M-step, the distribution $q(\mathbf{Z})$ is held fixed and the lower bound $L(q, \boldsymbol{\theta})$ is maximized with respect to $\boldsymbol{\theta}$ to give some new value $\boldsymbol{\theta}^{\text{new}}$. This will cause the lower bound L to increase (unless it is already at a maximum), which will necessarily cause the corresponding log likelihood function to increase.
- Because the distribution q is determined using the old parameter values rather than the new values and is held fixed during the M-step, it will not equal the new posterior distribution $p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}^{\text{new}})$, and hence there will be a non-zero KL divergence.
- The increase in the log likelihood function is therefore greater than the increase in the lower bound, as shown in the figure below.

Illustration of the M step of the EM algorithm. The distribution $q(\mathbf{Z})$ is held fixed and the lower bound $\mathcal{L}(q, \boldsymbol{\theta})$ is maximized with respect to the parameter vector $\boldsymbol{\theta}$ to give a revised value $\boldsymbol{\theta}^{\text{new}}$. Because the KL divergence is nonnegative, this causes the log likelihood $\ln p(\mathbf{X} | \boldsymbol{\theta})$ to increase by at least as much as the lower bound does.



EM: General Case

$$\ln p(\mathbf{X} | \boldsymbol{\theta}) = L(q, \theta) + KL(q \| p)$$

- If we substitute $q(\mathbf{Z}) = p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}^{old})$ into $L(q, \theta) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})}{q(\mathbf{Z})} \right\}$

We see that, after the E-step, the lower bound takes the form:

$$\begin{aligned} L(q, \theta) &= \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})}{q(\mathbf{Z})} \right\} = \sum_{\mathbf{Z}} p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}^{old}) \ln p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta}) - \sum_{\mathbf{Z}} p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}^{old}) \ln p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}^{old}) \\ &= Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}) + const \end{aligned}$$

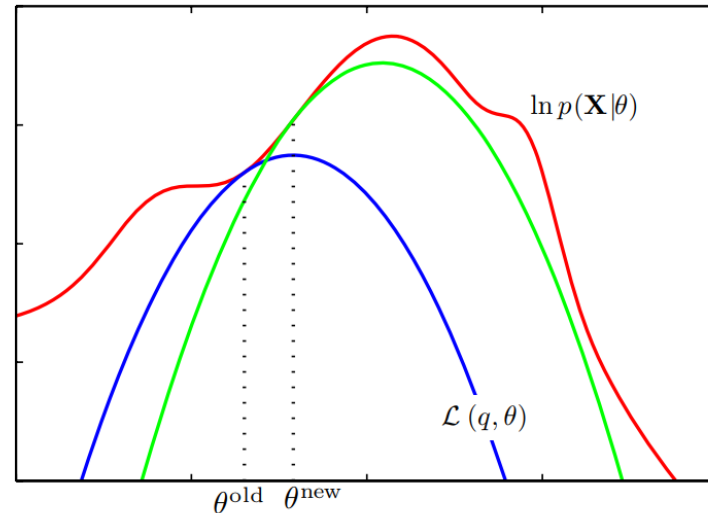
where the constant is simply the entropy of the q distribution and is therefore independent of θ .

(*) Thus in the M-step, the quantity being maximized is the expectation of the complete-data log likelihood, as we saw earlier in the case of the GMM model.

EM: General Case

- The operation of the EM algorithm can also be viewed in the space of parameters as shown below.

The EM algorithm involves alternately computing a lower bound on the log likelihood for the current parameter values and then maximizing this bound to obtain the new parameter values. See the text for a full discussion.

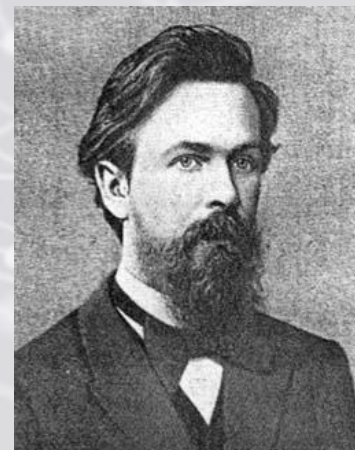
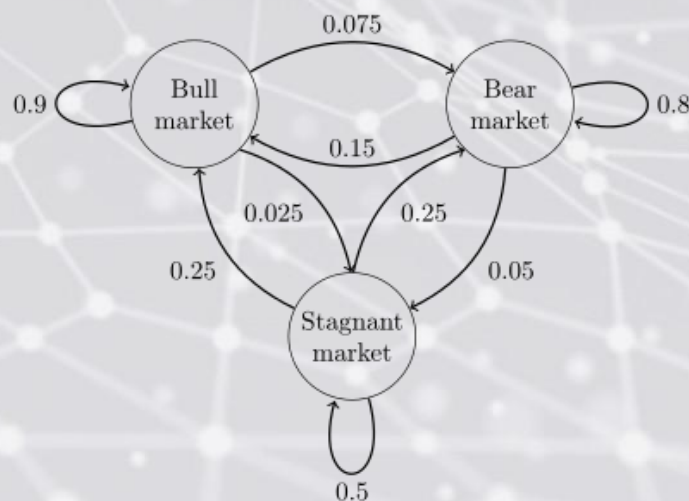
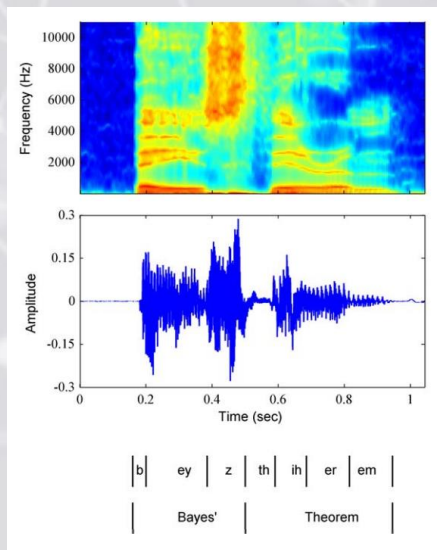


- Here the red curve depicts the (incomplete data) log likelihood function whose value we wish to maximize. We start with some initial parameter value θ^{old} , and in the first E-step we evaluate the posterior distribution over latent variables, which gives rise to a lower bound $\mathcal{L}(q, \theta^{\text{old}})$ whose value equals the log likelihood at θ^{old} , so that both curves have the same gradient (the bound is a convex function).
- In the M-step, the bound is maximized giving the value θ^{new} which yields a larger value of log likelihood than theta old.
- The subsequent E-step then constructs a bound that is tangential at θ^{new} , as shown by the green curve.

Markov Models

- **Markov models** represent a classic paradigm for modeling sequential data.
- Given a time series (say of observed weather from day to day), we would like to predict the current system state (e.g. the weather forecast today).
- Notice that if we treat the data as I.I.D., then the only information we can glean from the data is the relative frequency of various weather outcomes (i.e. how many times it was sunny, raining, etc.). Naturally, in practice weather and other time series data modalities, e.g., “stock market outlook”, are highly dependent on previous states. A Markov model encodes these dependencies.

Audio signals
represent a
natural
modality for
sequential data



Markov Models

- *WLOG* (without loss of generality) we can use the (generalized) **product rule of probability** to express *any* joint distribution for a sequence of observations in the form:


$$p(x_1, \dots, x_N) = \prod_{n=2}^N p(x_n \mid x_1, \dots, x_{n-1})$$

Markov Models

- *WLOG* (without loss of generality) we can use the (generalized) **product rule of probability** to express *any* joint distribution for a sequence of observations in the form:

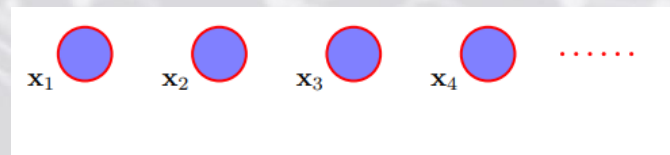
$$p(x_1, \dots, x_N) = \prod_{n=2}^N p(x_n | x_1, \dots, x_{n-1})$$

(*) Now if we impose the assumption that each conditional distribution on the RHS above is independent of all previous observations – *except the most recent* – we obtain a **first-order Markov chain**.

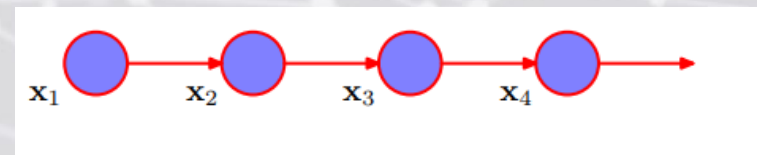

$$p(x_1, \dots, x_N) = p(x_1) \prod_{n=2}^N p(x_n | x_{n-1})$$

For a first-order Markov chain, the conditional distribution for x_n given all previous observations is given by:

$$p(x_n | x_1, \dots, x_{n-1}) = p(x_n | x_{n-1})$$



Graphical Model representing I.I.D. sequential data

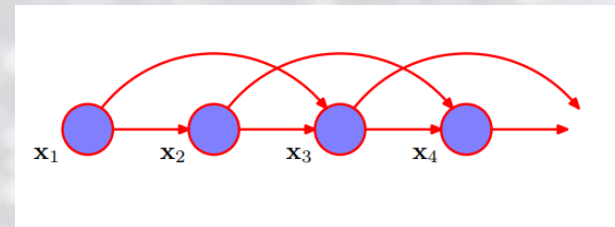


Graphical Model representing a first-order Markov chain

Markov Models

- In order to encode higher-order data dependencies, we can use a **higher-order Markov chain**. If, for example, we allow the predictions of the current state to depend also on the previous two states, we obtain a *second-order Markov chain*; this idea generalizes in the natural way.

$$p(x_1, \dots, x_n) = p(x_1) p(x_2 | x_1) \prod_{n=3}^N p(x_n | x_{n-1}, x_{n-2})$$

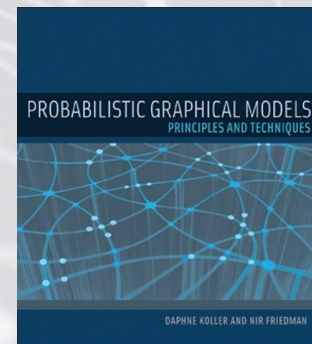


Graphical Model depicting a second-order Markov chain

- As before, one can easily show (using *d-separation**) that for a second-order Markov chain, the conditional distribution for x_n given all previous observations reduces to:

$$p(x_n | x_1, \dots, x_{n-1}) = p(x_n | x_{n-1}, x_{n-2})$$

(*) See: Daphne Koller's text for a thorough treatment of topics on graphical models, including d-separation.



Markov Models

- First-order Markov chain:

$$p(x_1, \dots, x_N) = p(x_1) \prod_{n=2}^N p(x_n | x_{n-1})$$

- Second-order *Markov chain*:

$$p(x_1, \dots, x_n) = p(x_1) p(x_2 | x_1) \prod_{n=3}^N p(x_n | x_{n-1}, x_{n-2})$$

- One can similarly consider extensions to an **Mth order Markov chain**, in which the conditional distribution for a particular variable depends on the previous M variables.

Q: What is the price we pay for this increased flexibility?

Markov Models

- First-order Markov chain:

$$p(x_1, \dots, x_N) = p(x_1) \prod_{n=2}^N p(x_n | x_{n-1})$$

- Second-order *Markov chain*:

$$p(x_1, \dots, x_n) = p(x_1) p(x_2 | x_1) \prod_{n=3}^N p(x_n | x_{n-1}, x_{n-2})$$

- One can similarly consider extensions to an **Mth order Markov chain**, in which the conditional distribution for a particular variable depends on the previous M variables.

Q: What is the price we pay for this increased flexibility?

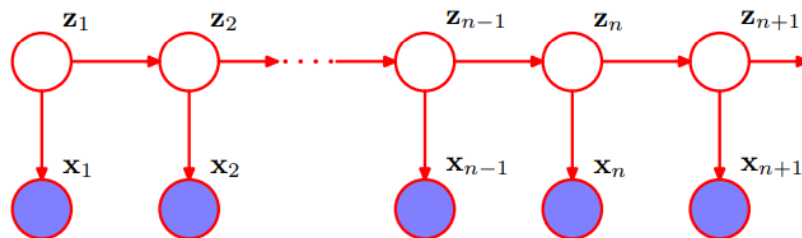
A: In a first-order Markov chain – where we assume K distinct possible discrete states – the conditional probability $p(x_n | x_{n-1})$ expressed as, say, a look-up table consists of $K(K-1)$ parameters. Why?

(*) In general, an M^{th} order Markov chain contains **$K^M(K-1)$ parameters**. Clearly, an arbitrarily large order Markov chain is impractical in general (in practice $M \leq 4$ is common)

Markov Models: HMMs

- Many ML applications naturally admit of latent/hidden variables or information (e.g. NLP); furthermore, it is often useful to simply introduce latent variables into a model in order to permit a rich class of models to be constructed out of simple components.
- For each observation x_n , we introduce a corresponding latent variable z_n (which may be of different type or dimensionality to the observed variable).
- We now assume that it is the hidden variables that form a Markov chain, giving rise to a graphical structure known as a **state space model**, as shown in the figure.

Figure 13.5 We can represent sequential data using a Markov chain of latent variables, with each observation conditioned on the state of the corresponding latent variable. This important graphical structure forms the foundation both for the hidden Markov model and for linear dynamical systems.

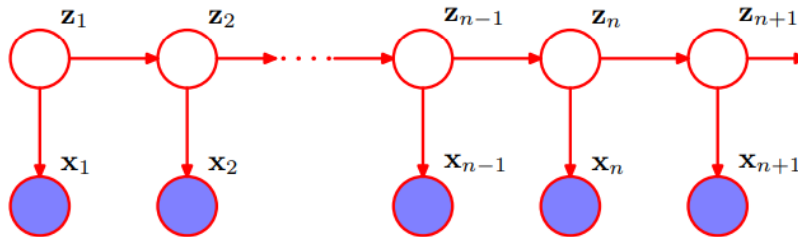


This model satisfies the key conditional independent property that z_{n-1} and z_{n+1} are independent given z_n , so that:

$$z_{n+1} \perp z_{n-1} \mid z_n$$

Markov Models: HMMs

Figure 13.5 We can represent sequential data using a Markov chain of latent variables, with each observation conditioned on the state of the corresponding latent variable. This important graphical structure forms the foundation both for the hidden Markov model and for linear dynamical systems.



- This model satisfies the key conditional independent property that z_{n-1} and z_{n+1} are independent given z_n , so that: $z_{n+1} \perp z_{n-1} \mid z_n$

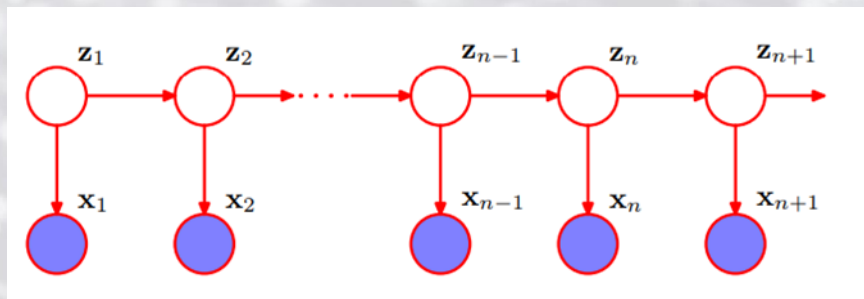
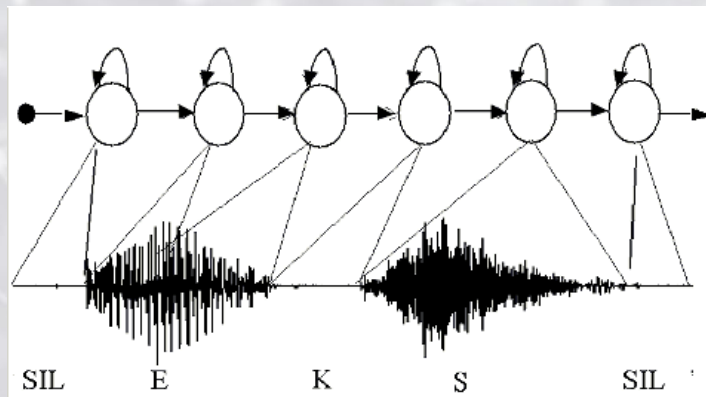
The joint distribution of this model is given by:

$$p(x_1, \dots, x_n, z_1, \dots, z_N) = p(z_1) \left[\prod_{n=2}^N p(z_n \mid z_{n-1}) \right] \prod_{n=1}^N p(x_n \mid z_n)$$

(*) Notice that the predictive distribution for observation x_{n+1} given all previous observations: $p(x_{n+1} \mid x_1, \dots, x_n)$ does not exhibit any conditional independent properties, and so our predictions for x_{n+1} depend on all previous observations. The observed variables. Typically the latent variables are discrete and the observed variables are either discrete or continuous.

Markov Models: HMMs

- HMMs are widely used in NLP applications (e.g., speech recognition, on-line handwriting recognition, parts-of-speech tagging), gesture recognition, and bioinformatics (DNA sequencing)



- Consider the observed variables denoted by x and latent variables are K -dimensional binary variables denoted by z . We assume that the hidden variables that form a first-order Markov chain, giving rise to a graphical structure known as a **state space model**.
- The conditional distribution over the latent variables corresponds to a table that we denote by \mathbf{A} , the elements of which are known as *transition probabilities*.

Markov Models: HMMs

- Consider the observed variables denoted by \mathbf{x} and latent variables are K -dimensional binary variables denoted by \mathbf{z} . We assume that the hidden variables that form a first-order Markov chain, giving rise to a graphical structure known as a **state space model**.
- The conditional distribution over the latent variables corresponds to a table that we denote by \mathbf{A} , the elements of which are known as *transition probabilities*.
- They are given by $A_{jk} = p(z_{n,k}=1 | z_{n-1,j})$, and because they are probabilities, they satisfy $0 \leq A_{jk} \leq 1$ with $\sum_k A_{jk} = 1$, so that the matrix A has $K(K-1)$ independent parameters. We can then write the conditional distribution explicitly in the form:

$$p(\mathbf{z}_n | \mathbf{z}_{n-1}, \mathbf{A}) = \prod_{k=1}^K \prod_{j=1}^K A_{jk}^{z_{n-1,j} z_{n,k}}$$

- The initial latent node \mathbf{z}_1 is special in that it does not have a parent node, and so it has a marginal distribution $p(\mathbf{z}_1)$ represented by a vector of probabilities $\boldsymbol{\pi}$ with elements $\pi_k = p(z_{1k}=1)$ so that

$$p(\mathbf{z}_1 | \boldsymbol{\pi}) = \prod_{k=1}^K \pi_k^{z_{1k}} \quad \text{where} \quad \sum_k \pi_k = 1$$

Markov Models: HMMs

- Recall that the latent variables are usually discrete and that the state variables are either discrete or continuous. The **emission probability** for an HMM is the distribution of the visible variable, given the latent variable:

$$p(x_n | \mathbf{z}_n, \phi) = \prod_{k=1}^K p(x_n | \phi_k)^{z_{nk}}$$

where φ represent the parameters of the emission distributions (e.g. μ, σ).

- The joint probability distribution over both latent and observed variables is then given by:

$$p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta}) = p(\mathbf{z}_1 | \boldsymbol{\pi}) \left[\prod_{n=2}^N p(\mathbf{z}_n | \mathbf{z}_{n-1}, \mathbf{A}) \right] \prod_{m=1}^N p(x_m | \mathbf{z}_m, \phi)$$

where: $\mathbf{X} = \{x_1, \dots, x_N\}$, $\mathbf{Z} = \{\mathbf{z}_1, \dots, \mathbf{z}_N\}$ and $\boldsymbol{\theta} = \{\boldsymbol{\pi}, \mathbf{A}, \varphi\}$ are the parameters governing the model.

Markov Models: MLE for HMMs

- If we have observed a data set $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, we can determine the parameters of an HMM using maximum likelihood.

The likelihood function is obtained from the joint via *marginalization*:

$$p(\mathbf{X} | \boldsymbol{\theta}) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})$$

Due to the presence of both observed and hidden variables, we can turn to the EM framework to generate the MLE for HMMs.

Recall:

The EM algorithm starts with some initial selection for the model parameters: $\boldsymbol{\theta}^{\text{old}}$

In the E-step, we take these parameter values and find the posterior distribution of the latent variables $p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}^{\text{old}})$. We then use this posterior distribution to evaluate the expectation of the logarithm of the complete-data likelihood function, as a function of the parameters $\boldsymbol{\theta}$, to give the function $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}})$ defined by:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta}^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})$$

Markov Models: MLE for HMMs

Recall:

The EM algorithm starts with some initial selection for the model parameters: θ^{old}

In the E-step, we take these parameter values and find the posterior distribution of the latent variables $p(\mathbf{Z} | \mathbf{X}, \theta^{\text{old}})$. We then use this posterior distribution to evaluate the expectation of the logarithm of the complete-data likelihood function, as a function of the parameters θ , to give the function $Q(\theta, \theta^{\text{old}})$ defined by:

$$Q(\theta, \theta^{\text{old}}) = \sum_{\mathbf{Z}} p(\mathbf{Z} | \mathbf{X}, \theta^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z} | \theta)$$

(*) Introducing some convenient notation: we shall use $\gamma(\mathbf{z}_n)$ to denote the marginal posterior distribution of a latent variable \mathbf{z}_n , and $\xi(\mathbf{z}_{n-1}, \mathbf{z}_n)$ to denote the joint posterior distribution of two successive latent variables so that:

$$\gamma(\mathbf{z}_n) = p(\mathbf{z}_n | \mathbf{X}, \theta^{\text{old}})$$

$$\xi(\mathbf{z}_{n-1}, \mathbf{z}_n) = p(\mathbf{z}_{n-1}, \mathbf{z}_n | \mathbf{X}, \theta^{\text{old}})$$

Markov Models: MLE for HMMs

(*) Introducing some convenient notation: we shall use $\gamma(\mathbf{z}_n)$ to denote the marginal posterior distribution of a latent variable \mathbf{z}_n , and $\xi(\mathbf{z}_{n-1}, \mathbf{z}_n)$ to denote the joint posterior distribution of two successive latent variables so that:

$$\gamma(\mathbf{z}_n) = p(\mathbf{z}_n | \mathbf{X}, \boldsymbol{\theta}^{old})$$

$$\xi(\mathbf{z}_{n-1}, \mathbf{z}_n) = p(\mathbf{z}_{n-1}, \mathbf{z}_n | \mathbf{X}, \boldsymbol{\theta}^{old})$$

- For each value of n , we can store $\gamma(\mathbf{z}_n)$ using a set of K non-negative numbers that sum to unity, and similarly we can store $\xi(\mathbf{z}_{n-1}, \mathbf{z}_n)$ using a $K \times K$ matrix of non-negative numbers that again sum to unity. We shall also use $\gamma(z_{nk})$ to connote the conditional probability of $z_{nk}=1$, with a similar use of notation for $\xi(z_{n-1,j}, z_{nk})$.

Note that the expected value of a binary random variable is just the probability that it takes value 1, giving us:

$$\gamma(z_{nk}) = E[z_{nk}] = \sum_{\mathbf{z}_n} \gamma(\mathbf{z}) z_{nk}$$

$$\xi(z_{n-1,j}, z_{nk}) = E[z_{n-1,j}, z_{nk}] = \sum_{\mathbf{z}_{n-1}, \mathbf{z}_n} \gamma(\mathbf{z}) z_{n-1,j} z_{nk}$$

Markov Models: MLE for HMMs

(*) We can now re-write: $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}) = \sum_{\mathbf{Z}} p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}^{old}) \ln p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})$



$$\gamma(z_{nk}) = E[z_{nk}] = \sum_{\mathbf{z}_n} \gamma(\mathbf{z}) z_{nk}$$

$$\xi(z_{n-1,j}, z_{nk}) = E[z_{n-1,j}, z_{nk}] = \sum_{\mathbf{z}_{n-1}, \mathbf{z}_n} \gamma(\mathbf{z}) z_{n-1,j} z_{nk}$$

$$\begin{aligned} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}) = & \sum_{k=1}^K \gamma(z_{1k}) \ln \pi_k + \sum_{n=2}^N \sum_{j=1}^K \sum_{k=1}^K \xi(z_{n-1,j}, z_{nk}) \ln A_{jk} \\ & + \sum_{n=1}^N \sum_{k=1}^K \gamma(z_{nk}) \ln p(\mathbf{x}_n | \phi_k). \end{aligned} \quad (13.17)$$

• In the M-step, we maximize $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old})$ with respect to the parameter set $\boldsymbol{\theta} = \{\boldsymbol{\pi}, \mathbf{A}, \boldsymbol{\varphi}\}$ in which we treat $\gamma(\mathbf{z}_n)$ and $\xi(\mathbf{z}_{n-1}, \mathbf{z}_n)$ as a constant. Maximization with respect to $\boldsymbol{\pi}$ and \mathbf{A} is achieved using Lagrange multipliers:

$$\begin{aligned} \pi_k &= \frac{\gamma(z_{1k})}{\sum_{j=1}^K \gamma(z_{1j})} \\ A_{jk} &= \frac{\sum_{n=2}^N \xi(z_{n-1,j}, z_{nk})}{\sum_{l=1}^K \sum_{n=2}^N \xi(z_{n-1,j}, z_{nl})}. \end{aligned}$$

Markov Models: MLE for HMMs

(*) We can now re-write: $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}) = \sum_{\mathbf{Z}} p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}^{old}) \ln p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})$

$$\begin{aligned} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}) = & \sum_{k=1}^K \gamma(z_{1k}) \ln \pi_k + \sum_{n=2}^N \sum_{j=1}^K \sum_{k=1}^K \xi(z_{n-1,j}, z_{nk}) \ln A_{jk} \\ & + \sum_{n=1}^N \sum_{k=1}^K \gamma(z_{nk}) \ln p(\mathbf{x}_n | \phi_k). \end{aligned} \quad (13.17)$$

• In the M-step, we maximize $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old})$ with respect to the parameter set $\boldsymbol{\theta} = \{\boldsymbol{\pi}, \mathbf{A}, \boldsymbol{\varphi}\}$ in which we treat $\gamma(\mathbf{z}_n)$ and $\xi(\mathbf{z}_{n-1}, \mathbf{z}_n)$ as a constant. Maximization with respect to $\boldsymbol{\pi}$ and \mathbf{A} is achieved using Lagrange multipliers:

$$\begin{aligned} \pi_k &= \frac{\gamma(z_{1k})}{\sum_{j=1}^K \gamma(z_{1j})} \\ A_{jk} &= \frac{\sum_{n=2}^N \xi(z_{n-1,j}, z_{nk})}{\sum_{l=1}^K \sum_{n=2}^N \xi(z_{n-1,j}, z_{nl})}. \end{aligned}$$

If the diffusion densities are Gaussians, then the maximization yields the GMM parameter formulae:

$$\begin{aligned} \boldsymbol{\mu}_k &= \frac{\sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n}{\sum_{n=1}^N \gamma(z_{nk})} \\ \boldsymbol{\Sigma}_k &= \frac{\sum_{n=1}^N \gamma(z_{nk}) (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T}{\sum_{n=1}^N \gamma(z_{nk})}. \end{aligned}$$

Markov Models: MLE for HMMs

(*) Note that it still remains to devise an efficient procedure to evaluate the quantities: $\gamma(\mathbf{z}_n)$ and $\xi(\mathbf{z}_{n-1}, \mathbf{z}_n)$, corresponding to the E-step in the EM algorithm.

(*) Because the HMM is a tree, one can use a “message passing” algorithm to estimate the posterior distribution of the latent variables. For HMMs in particular this message passing algorithm is known as the **forward-backward algorithm**. For brevity, we omit the details here.*

(*) An additional useful algorithm for HMMs that finds the most probable sequence of hidden states is called the **Viterbi algorithm**.**

Fin

