UNSW AUSTRALIA

Never Stand Still

Unsupervised Learning (1)

COMP9417 Machine Learning & Data Mining

Term 3, 2019

Adapted from slides by Dr Michael Bain

Aims

This lecture will develop your understanding of unsupervised learning methods. Following it you should be able to:

- compare supervised with unsupervised learning
- describe the problem of unsupervised learning
- describe *k*-means clustering
- outline the role of the EM algorithm in *k*-means clustering
- understand the use of clustering in several applications

Supervised vs. Unsupervised Learning

Supervised learning — classes are *known* and need a "definition", in terms of the data. Methods are known as: classification, discriminant analysis, class prediction, supervised pattern recognition.

Unsupervised learning — classes are initially *unknown* and need to be "discovered" with their definitions from the data. Methods are known as: cluster analysis, class discovery, unsupervised pattern recognition.

So: *unsupervised learning* methods, such as *clustering*, address the problem of assigning instances to classes *given only observations about the instances*, i.e., without being given class "labels" for instances by a "teacher".

Unsupervised Learning

Why do we need unsupervised learning ?

- most of the world's data is *unlabelled*
- getting a human to label data is often
  - difficult (what are the classes?)
  - time-consuming (labelling requires thinking)
  - expensive (see above)
  - error-prone (mistakes, ambiguity)
- in principle, can use any feature as the "label"
- unfortunately, often the class is not a known feature

Unsupervised Learning

What is unsupervised learning good for ?

- simplifying a problem, e.g., by dimensionality reduction
- exploratory data analysis, e.g., with visualization
- data transformation to simplify a classification problem
- to group data instances into subsets
- to discover structure, like hierarchies of subconcepts
- to learn new "features" for later use in classification
- to track "concept drift" over time
- to learn generative models for images, text, video, speech, etc.

Clustering

Finding groups of items that are similar

Clustering is unsupervised

- the class of any data instance is not known

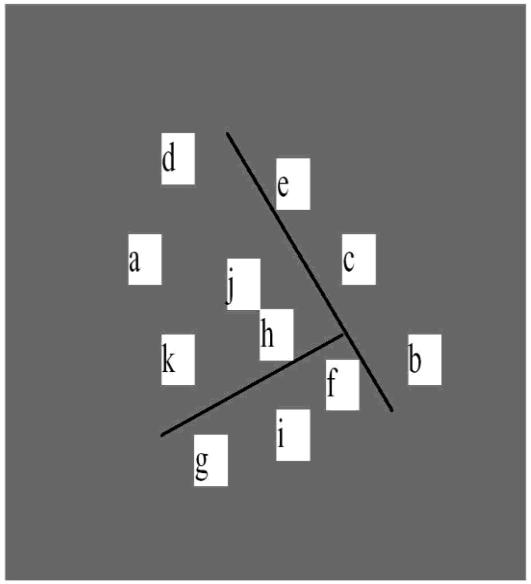
Success of clustering often measured subjectively

- OK for *exploratory data analysis* (EDA) *. . .*
- but problematic if you need quantitive results *. . .*
- some visual and statistical approaches

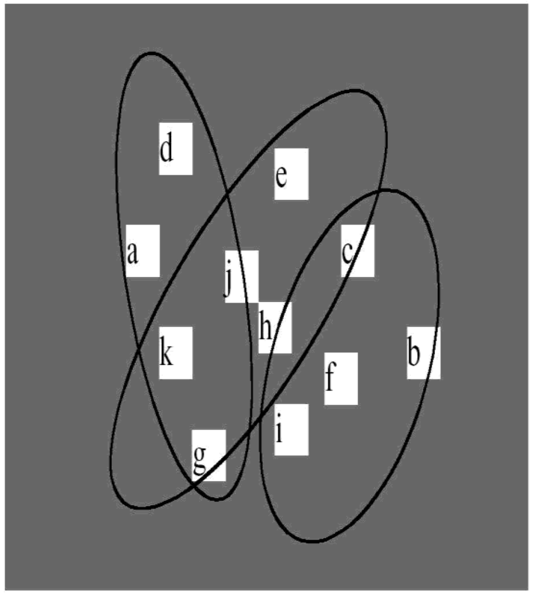
A dataset for clustering is just like a dataset for classification, but without the class labels

Simple 2D representations of clustering

Clusters form a partition

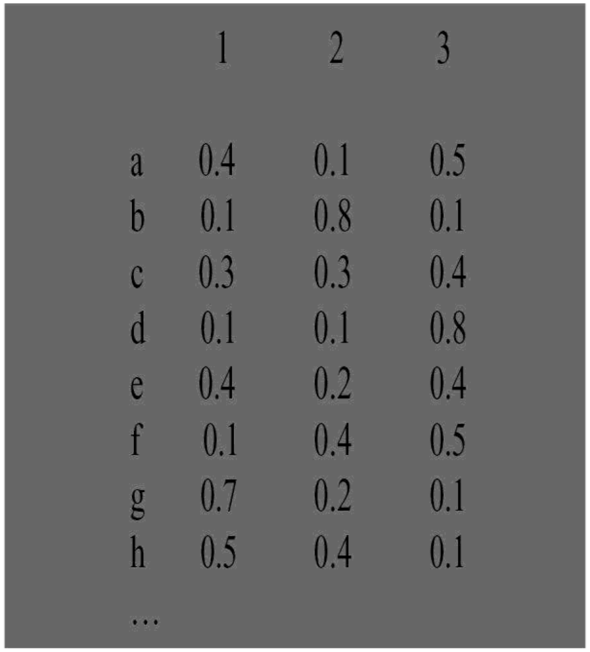


Venn diagram (overlapping clusters)



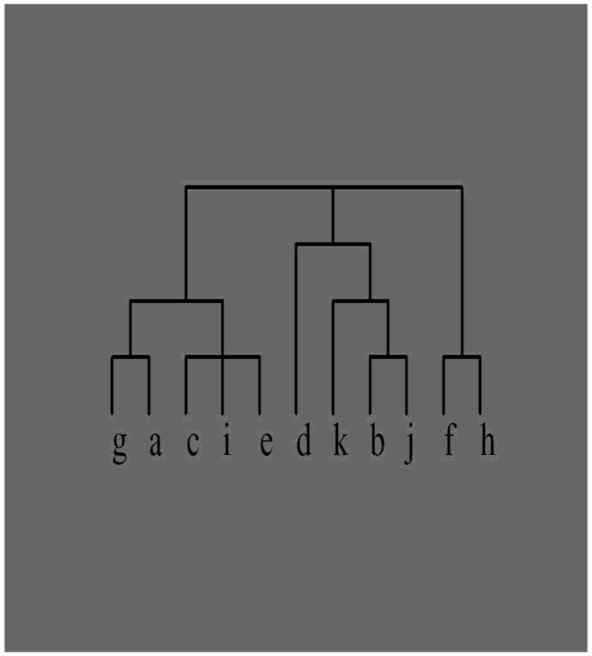
Other representations of clustering

Probabilistic assignment



	1	2	3
a	0.4	0.1	0.5
b	0.1	0.8	0.1
c	0.3	0.3	0.4
d	0.1	0.1	0.8
e	0.4	0.2	0.4
f	0.1	0.4	0.5
g	0.7	0.2	0.1
h	0.5	0.4	0.1
…			

Dendrogram



Cluster Analysis

Clustering algorithms form two broad categories:

- **hierarchical methods** and **partitioning methods**

Hierarchical algorithms are either **agglomerative** i.e. bottom-up or **divisive** i.e. top-down.

In practice, hierarchical agglomerative methods often used - efficient exact algorithms available, but more importantly to users the *dendrogram*, or tree, can be visualized.

Partitioning methods usually require specification of the number of clusters, then try to construct the clusters and fit objects to them.

Representation

Let $N = \{e_1, . . . , e_n\}$ be a set of elements, i.e. instances.

Let C = $(C_1, . . . , C_l)$ be a *partition* of N into subsets.

Each subset is called a *cluster*, and C is called a *clustering*.

Input data can have two forms:

- each element is associated with a real-valued vector of p features e.g. measurement levels for different features
- pairwise similarity data between elements, e.g. correlation, distance (dissimilarity)

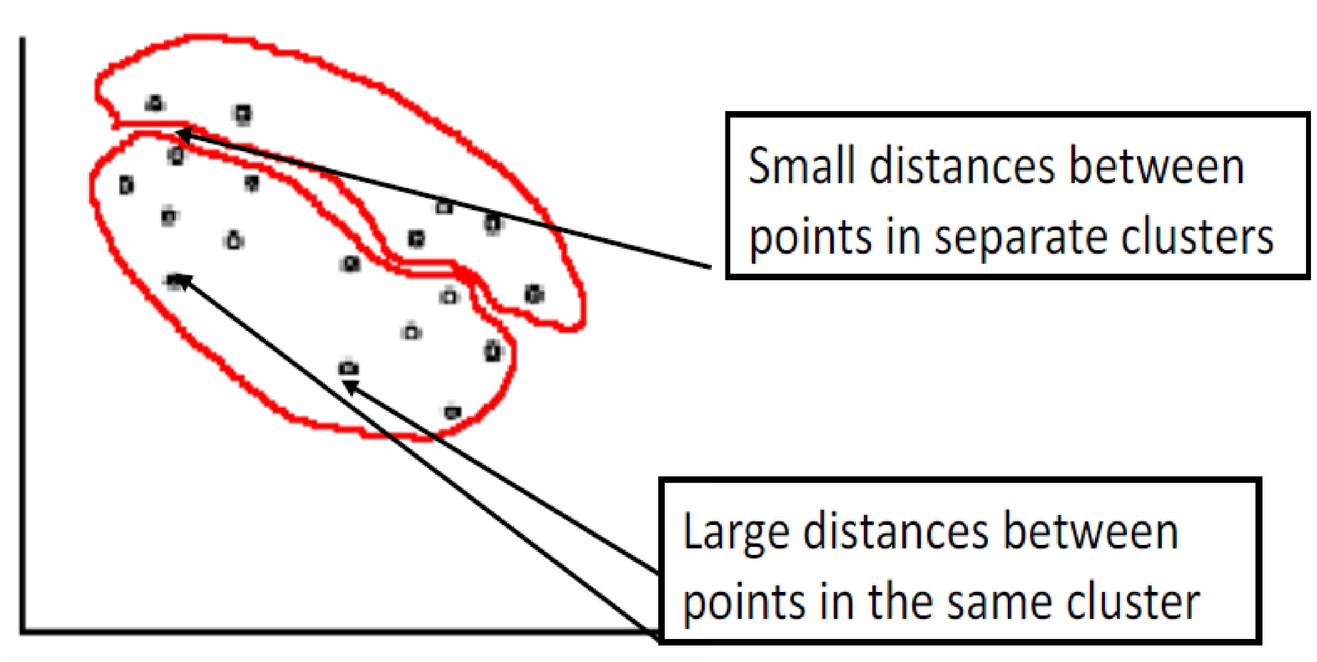
Feature-vectors have more information, but similarity is generic (given the appropriate function). Feature-vector matrix: $N \times p$, similarity matrix $N \times N$. In general, often $N \gg p$.

Clustering Framework

- Goal of clustering: find a partition of N elements (instances) into *homogeneous* and *well-separated* clusters
- Elements from same cluster should have high similarity, i.e, form a homogeneous cluster, while elements from different clusters should have low similarity, i.e., be well-separated
- Note: homogeneity and separation need to be defined
- In practice, use a distance measure appropriate to the problem
- Also note: typically there are interactions between homogeneity and separation – usually, high homogeneity is linked with low separation, and vice versa, unless there is clear *structure* in the data

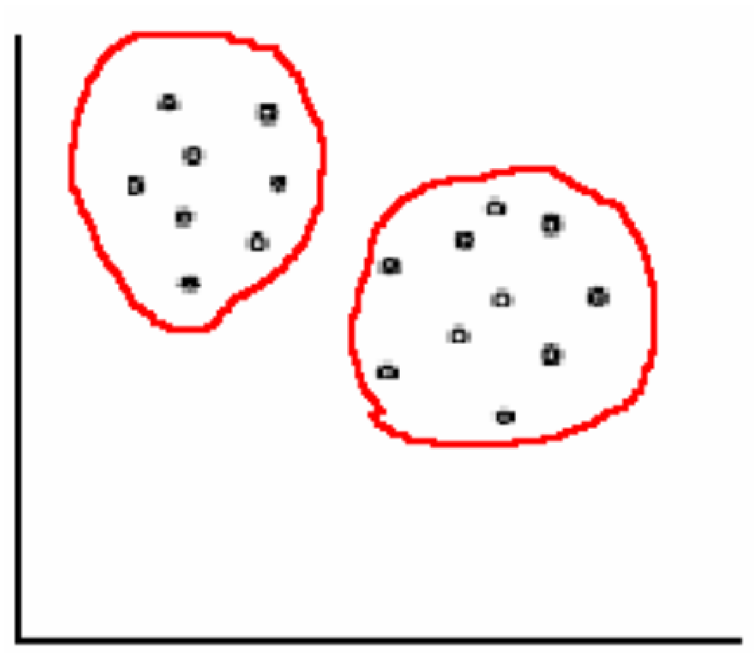
A bad clustering

This clustering violates both homogeneity and separation principles


Small distances between points in separate clusters
Large distances between points in the same cluster

A good clustering

This clustering satisfies both homogeneity and separation principles



k-means Clustering

Set value for k, the number of clusters (by prior knowledge or via search)

Initialise: choose points for centres (means) of k clusters (at random)

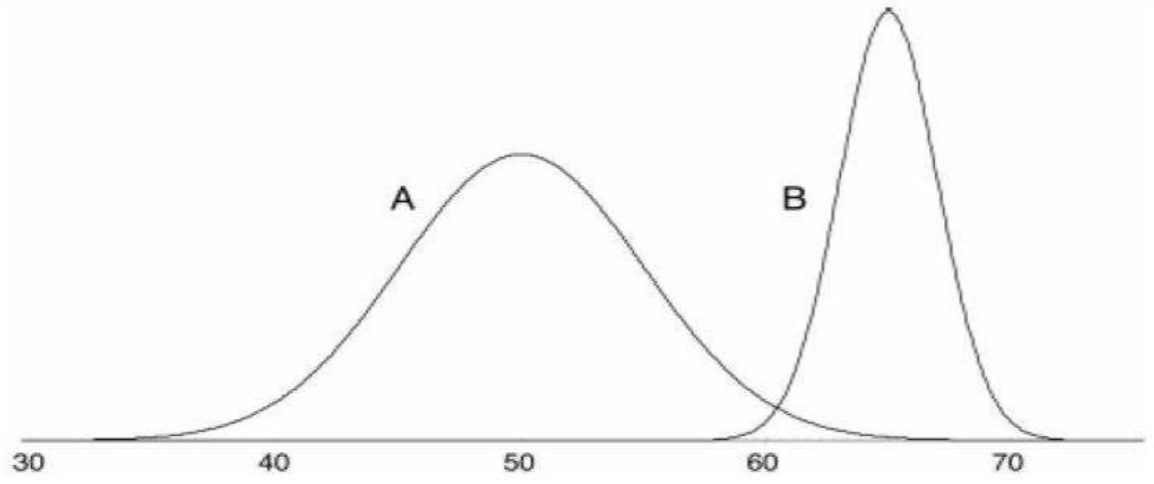
Procedure:

1) assign each instance x to the closest of the k points to form k clusters
2) re-assign the k points to be the means of each of the k clusters
3) repeat 1 and 2 until convergence to a reasonably stable clustering

Example: one variable 2-means

A	51	B	62	B	64	A	48	A	39	A	51
A	43	A	47	A	51	B	64	B	62	A	48
B	62	A	52	A	52	A	51	B	64	B	64
B	64	B	64	B	62	B	63	A	52	A	42
A	45	A	51	A	49	A	43	B	63	A	48
A	42	B	65	A	48	B	65	B	64	A	41
A	46	A	48	B	62	B	66	A	48		
A	45	A	49	A	43	B	65	B	64		
A	45	A	46	A	40	A	46	A	48		

model



$\mu_A=50,\ \sigma_A=5,\ p_A=0.6 \qquad \mu_B=65,\ \sigma_B=2,\ p_B=0.4$

k-means Clustering

$P(i)$ is the cluster assigned to element i, $c(j)$ is the centroid of cluster j, $d(v_1,v_2)$ is the Euclidean distance between feature vectors v_1 and v_2.

The goal is to find a partition P for which the error (distance) function is minimum:

$$E_P = \sum_{i=1}^{n} d(i, c(P(i))$$

Centroid is the mean or weighted average of the points in the cluster.

k-means is an important clustering method, widely-used in many different areas, that can be viewed in terms of the EM (Expectation-Maximization) algorithm.

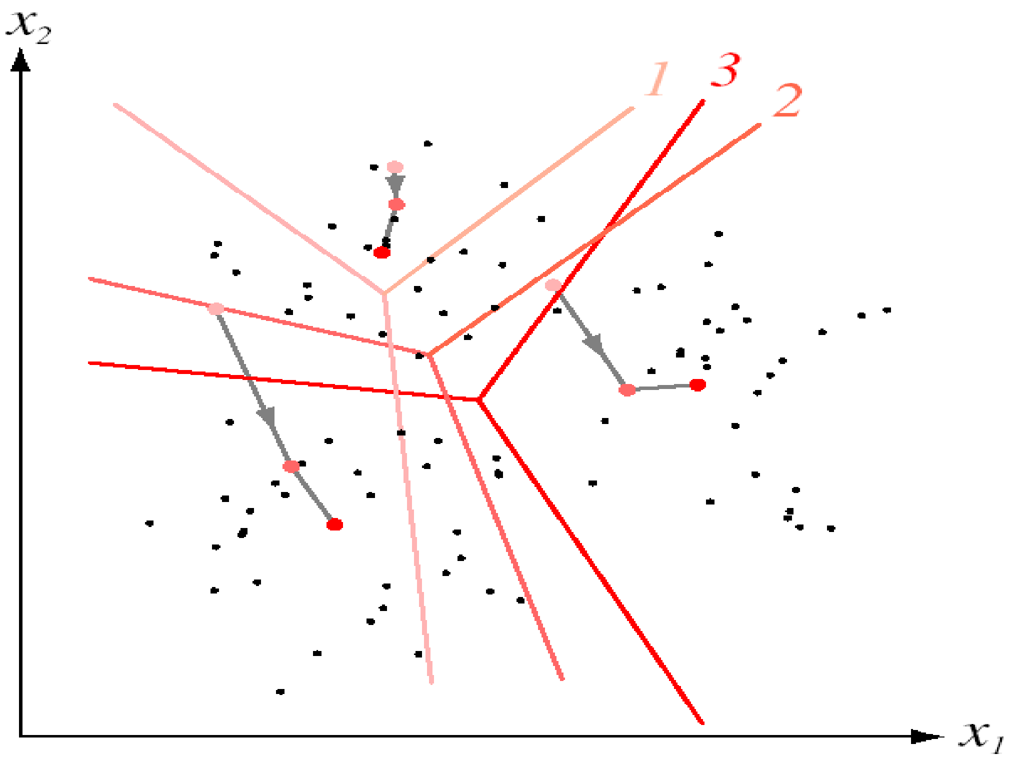
k-means Clustering

Algorithm k-means

```
/* feature-vector matrix $M(ij)$ is given */

1 Start with an arbitrary partition $P$ of $N$ into $k$
  clusters
2 for each element $i$ and cluster $j \neq P(i)$ let $E_P^{ij}$ be the
  cost of a solution in which $i$ is moved to $j$:
    1 if $E_P^{i^*j^*} = min_{ij} E_P^{ij} < E_P$ then move $i^*$ to cluster $j^*$ and
      repeat step 2 else halt.
```

k-means Clustering



k-means Clustering

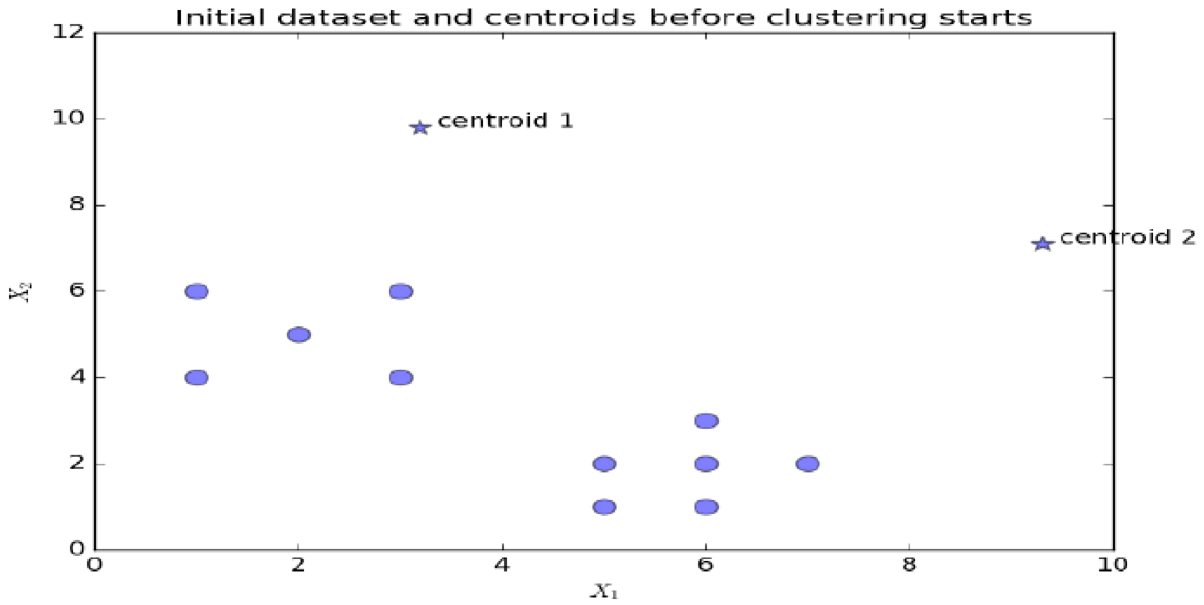
Previous diagram shows three steps to convergence in k-means with $k=3$

- means move to minimize squared-error criterion
- approximate method of obtaining maximum-likelihood estimates for means
- each point assumed to be in exactly one cluster
- if clusters “blend”, fuzzy k-means (i.e., overlapping clusters)

k-means Clustering: initialisation

X_1	X_2	Centroid
1	4	-
1	6	-
2	5	-
3	4	-
3	6	-
5	1	-
5	2	-
6	1	-
6	2	-
6	3	-
7	2	-

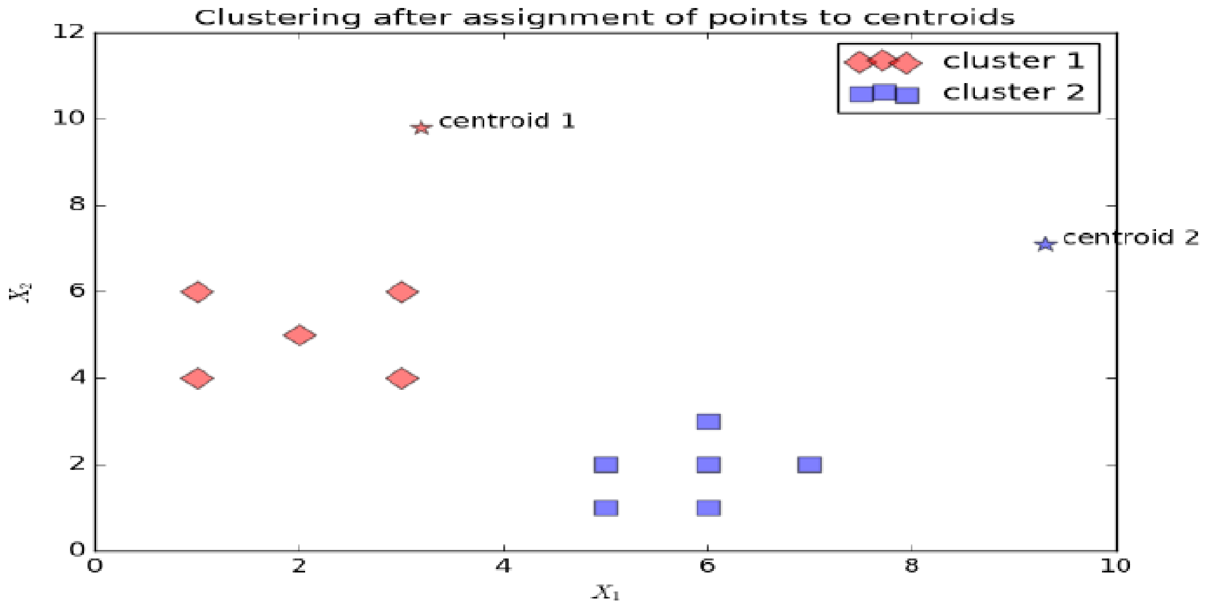
Centroid locations
centroid 1: $(3.2, 9.8)$
centroid 2: $(9.3, 7.1)$



k-means Clustering: assign to centroids

X_1	X_2	Centroid
1	4	1
1	6	1
2	5	1
3	4	1
3	6	1
5	1	2
5	2	2
6	1	2
6	2	2
6	3	2
7	2	2

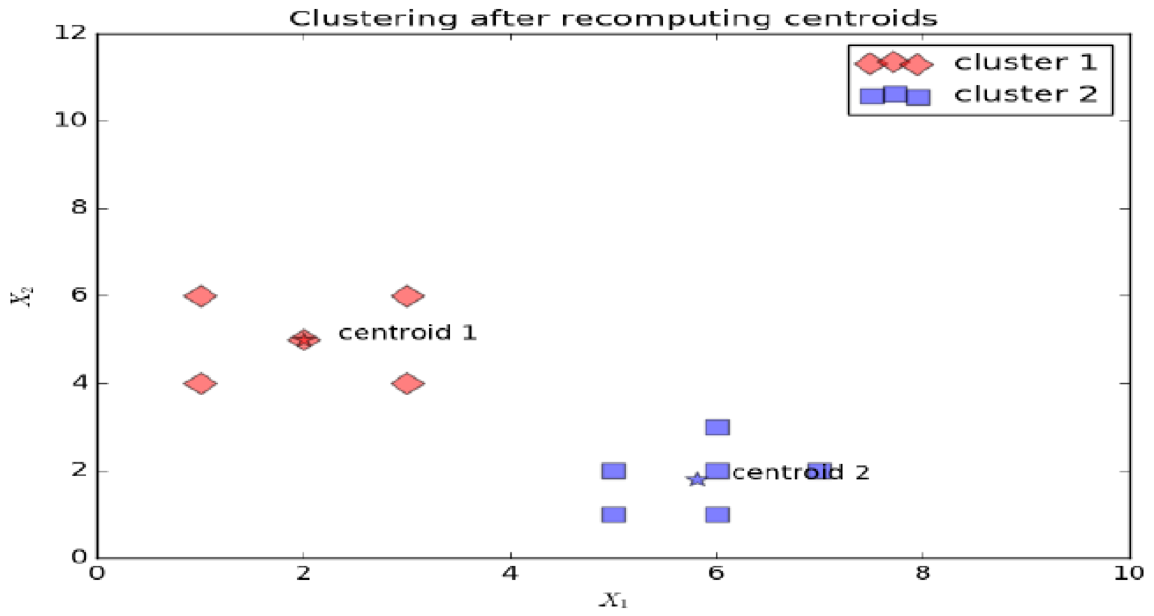
Centroid locations
centroid 1: $(3.2, 9.8)$
centroid 2: $(9.3, 7.1)$



k-means Clustering: recompute centroids

X_1	X_2	Centroid
1	4	1
1	6	1
2	5	1
3	4	1
3	6	1
5	1	2
5	2	2
6	1	2
6	2	2
6	3	2
7	2	2

Centroid locations
centroid 1: $(2.0, 5.0)$
centroid 2: $(5.8, 1.8)$



k-means Clustering: solution found

Shown on the 3 previous slides are the initialization and the two main steps of the k-means algorithm on the given dataset.

In this simple example k-means clustering has found a solution (the two centroids) after a single iteration, and the algorithm will not change it on further iterations.

By inspection, we can see the solution is a “good clustering”, in the sense that the two “natural” clusters in the dataset have been identified.

In general, the quality of the solution will depend on

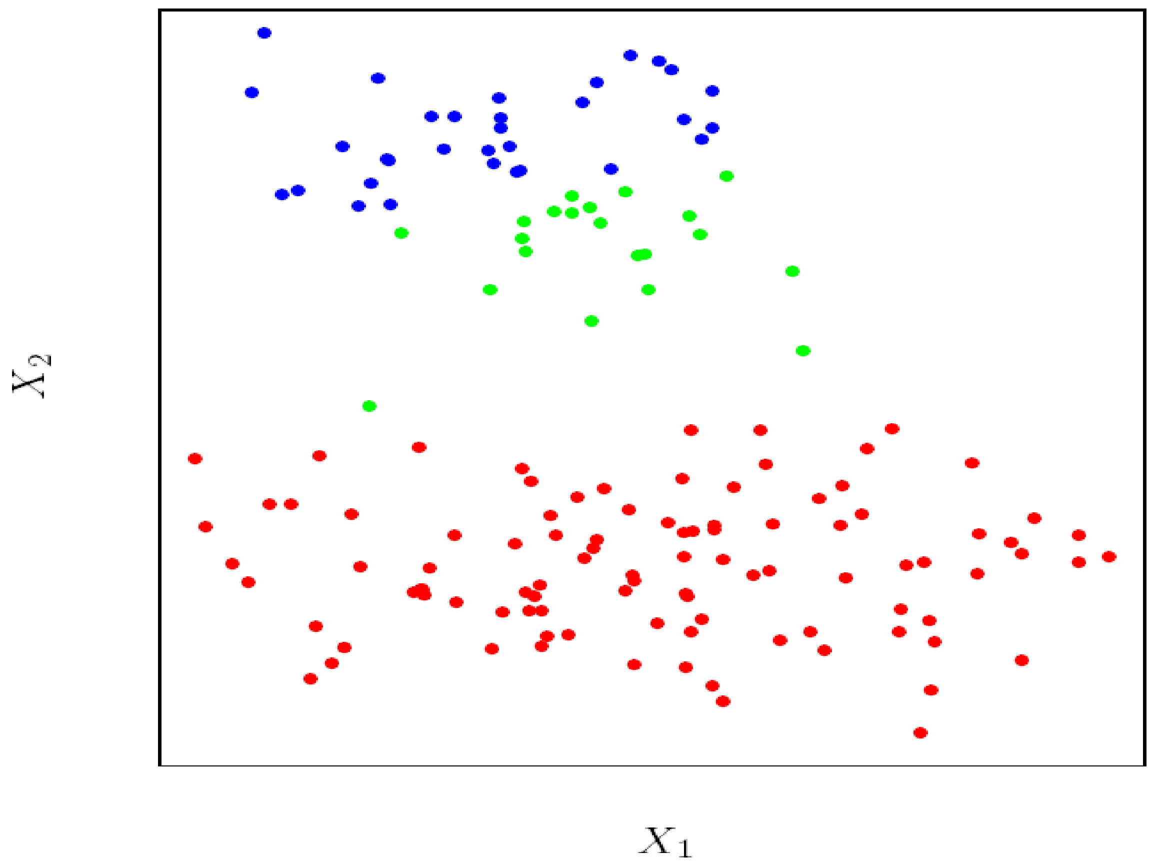
- the distribution of the points in the dataset
- the choice of k
- the choice of the location to initialise the centroids.

k-means Clustering: parameter

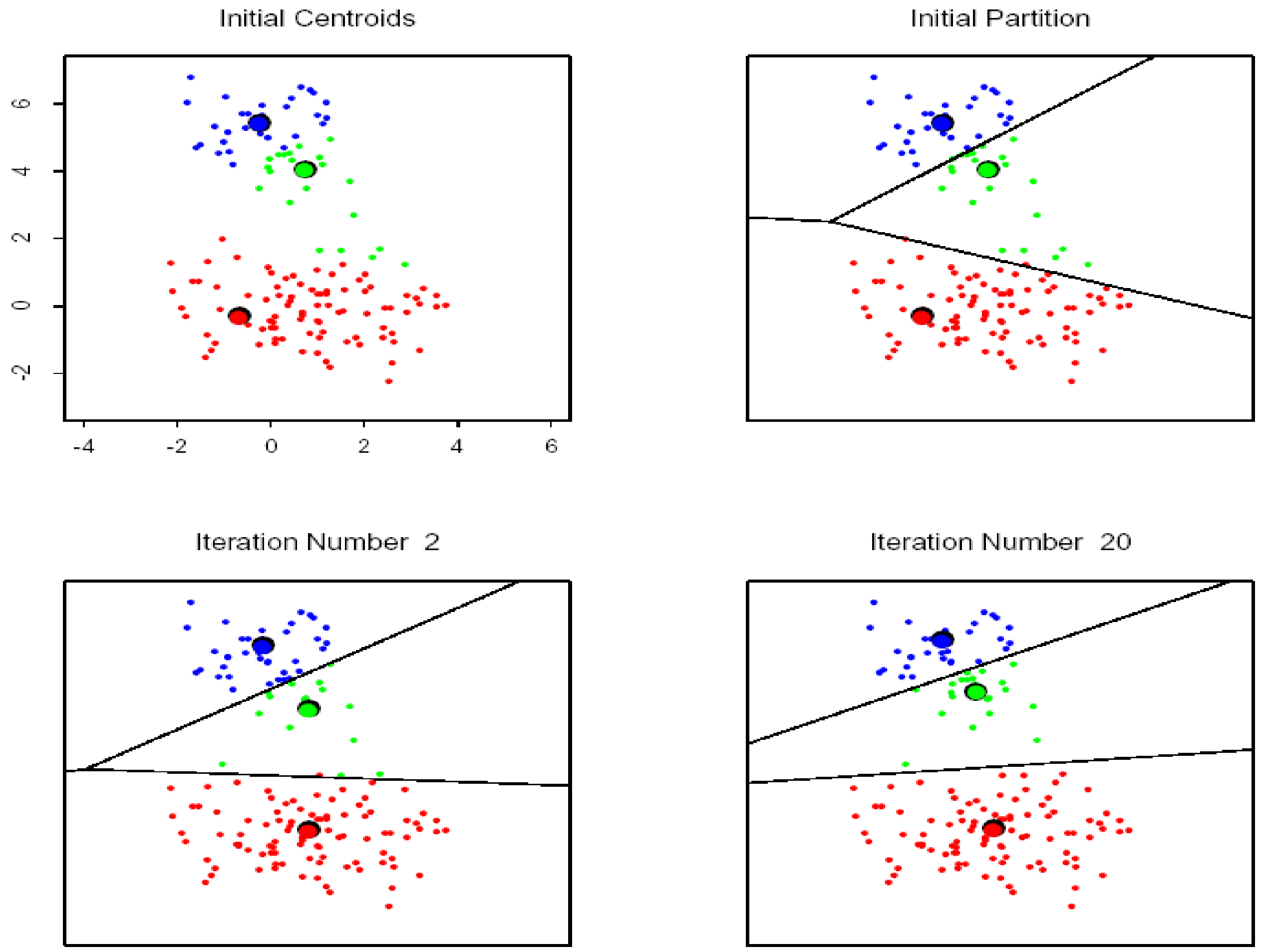
What about the number of clusters k?

Next diagrams show convergence in k-means clustering with $k = 3$ for data with two clusters not well separated.

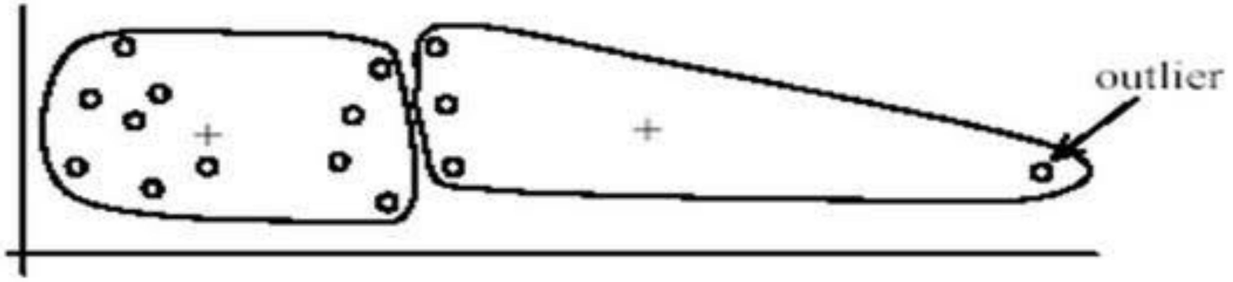
k-means Clustering: parameter



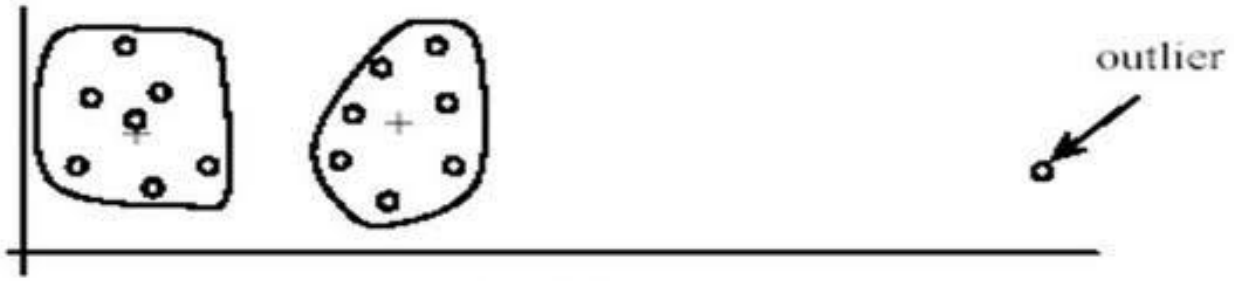
k-means Clustering: parameter



k-means Clustering: outliers



(A): Undesirable clusters



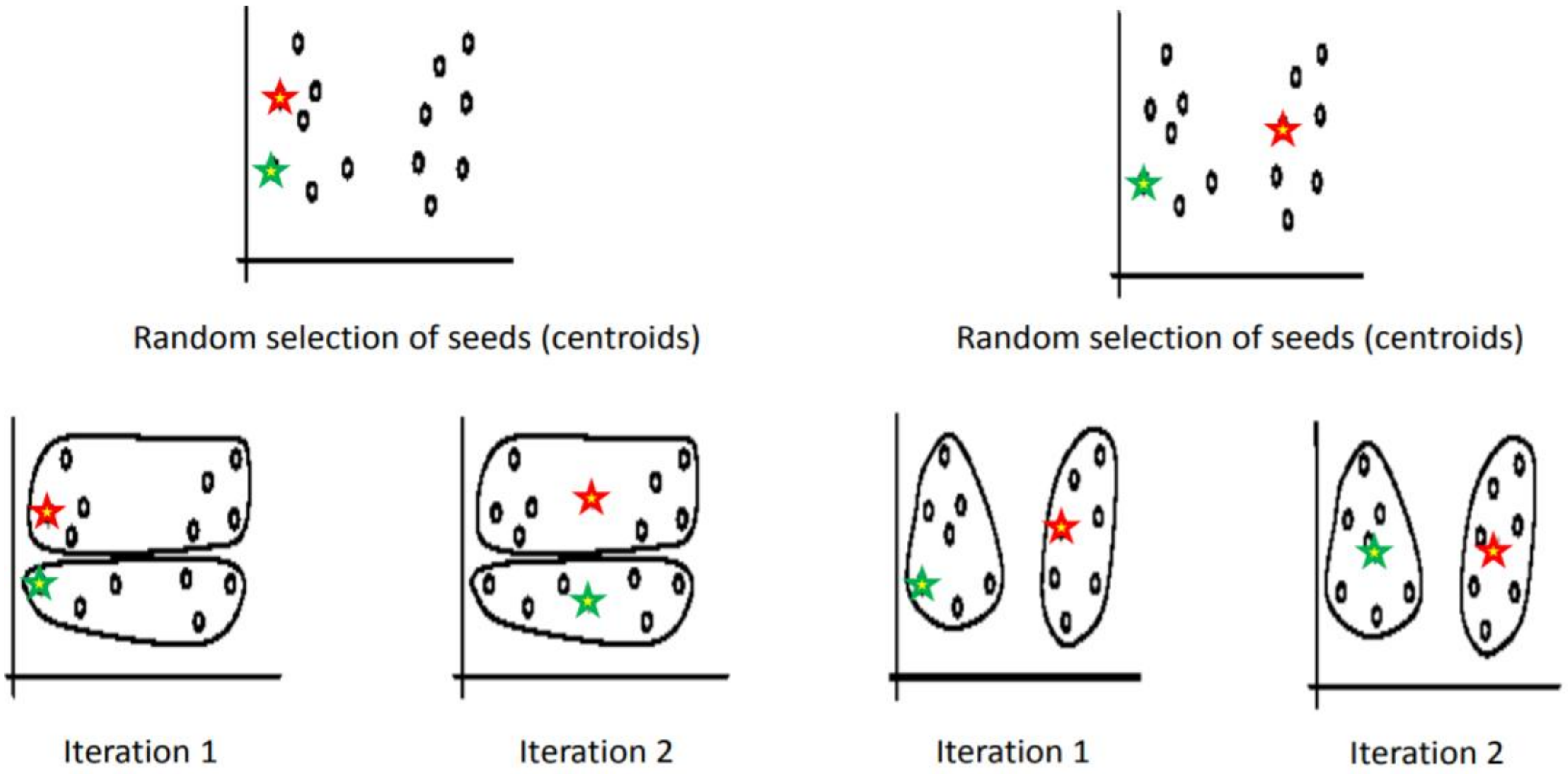
(B): Ideal clusters

k-means Clustering: outliers

Deal with outliers:

- Remove some data points that are much further away from the centroids than other data points
  - To be safe, we may want to monitor these possible outliers over a few iterations and then decide to remove them.
- Perform random sampling: by choosing a small subset of the data points, the chance of selecting an outlier is much smaller
  - Assign the rest of the data points to the clusters by distance or similarity comparison, or classification

k-means Clustering: initial seeds


Random selection of seeds (centroids)
Random selection of seeds (centroids)
Iteration 1
Iteration 2
Iteration 1
Iteration 2

In Practice

Algorithm can get trapped in a local minimum, toy example:

- Place four instances at the vertices of a two-dimensional rectangle
- Local minimum: two cluster centers at the midpoints of the rectangle's long sides

Result can vary significantly based on initial choice of seeds

Simple way to increase chance of finding a global optimum: restart with different random seeds

- can be time-consuming

Or use the k-means++ algorithm, which initialises k centroids to be maximally distant from each other

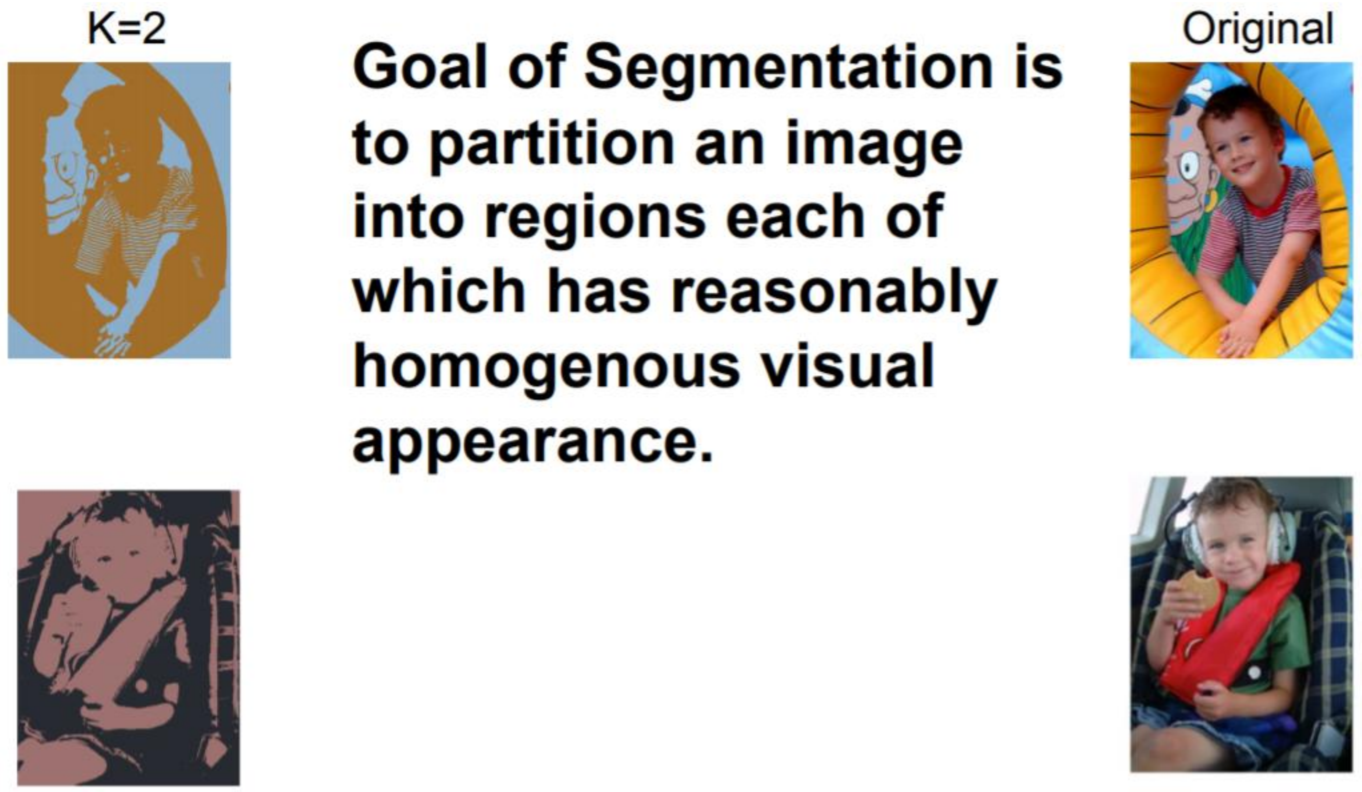
Remarks

Despite weaknesses, k-means is still the most popular algorithm due to its simplicity and efficiency

No clear evidence that any other clustering algorithm performs better in general

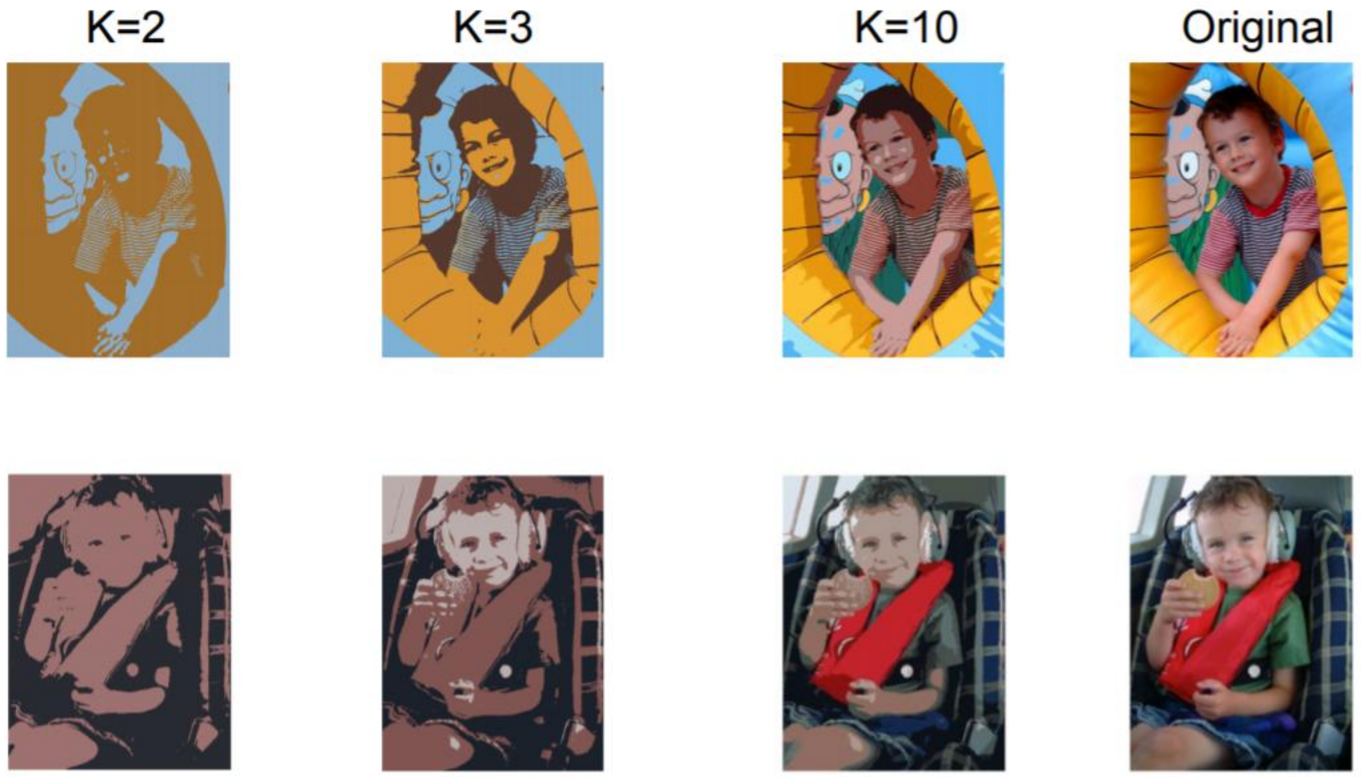
Comparing different clustering algorithms is a difficult task. No one knows the correct clusters!

Example: Image Segmentation



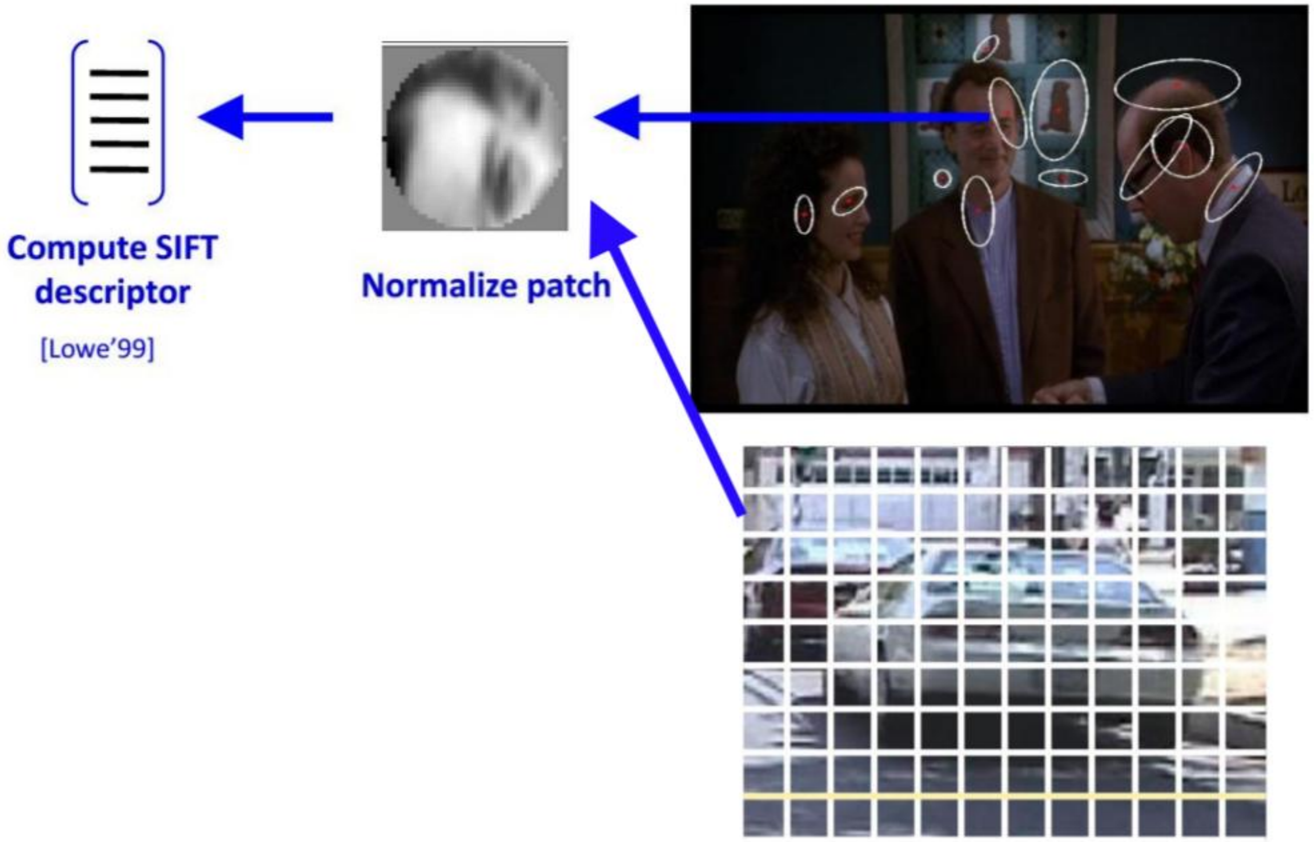
Goal of Segmentation is to partition an image into regions each of which has reasonably homogenous visual appearance.

Example: Image Segmentation



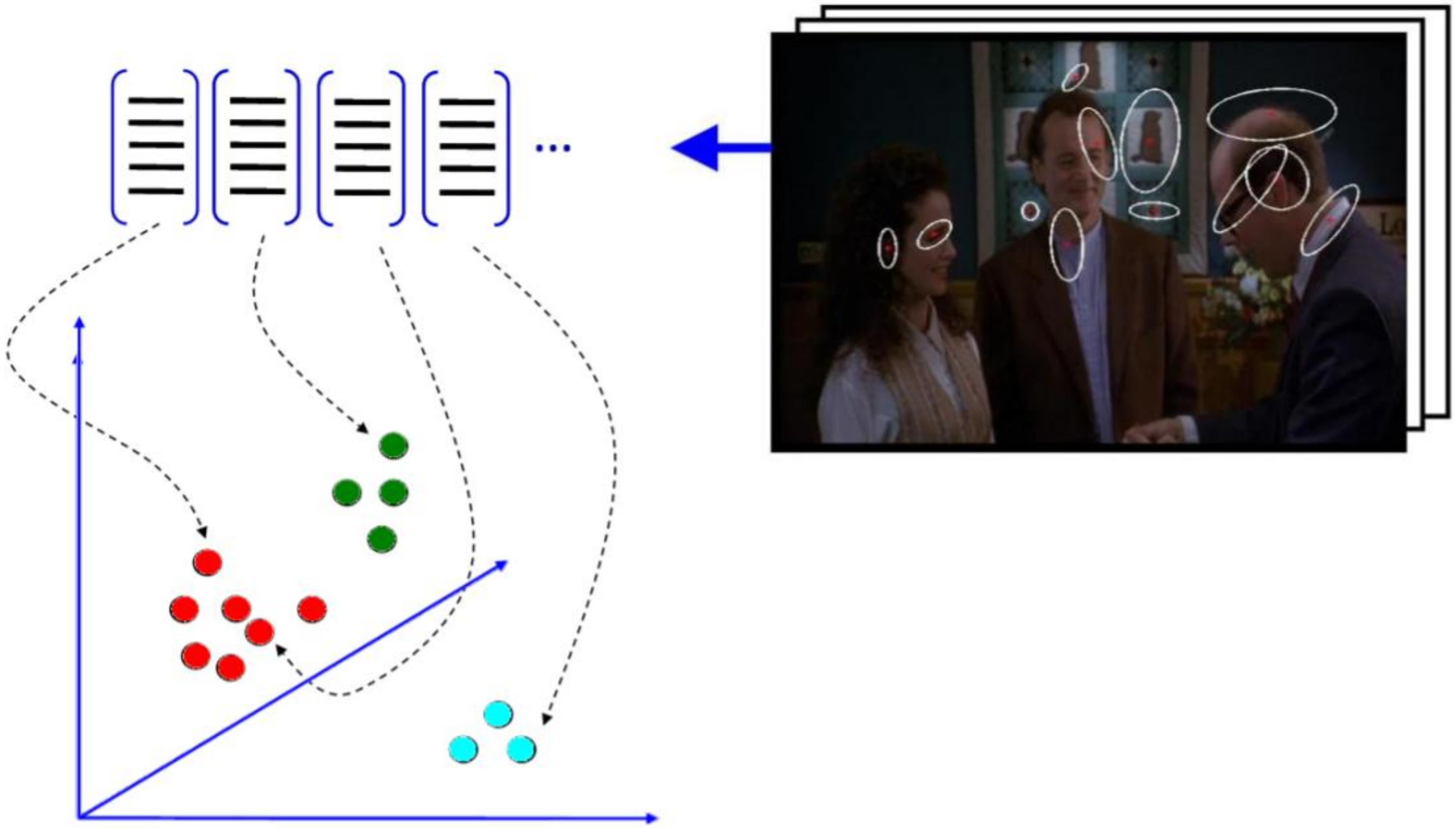
Example: Bag of Words

Feature extraction



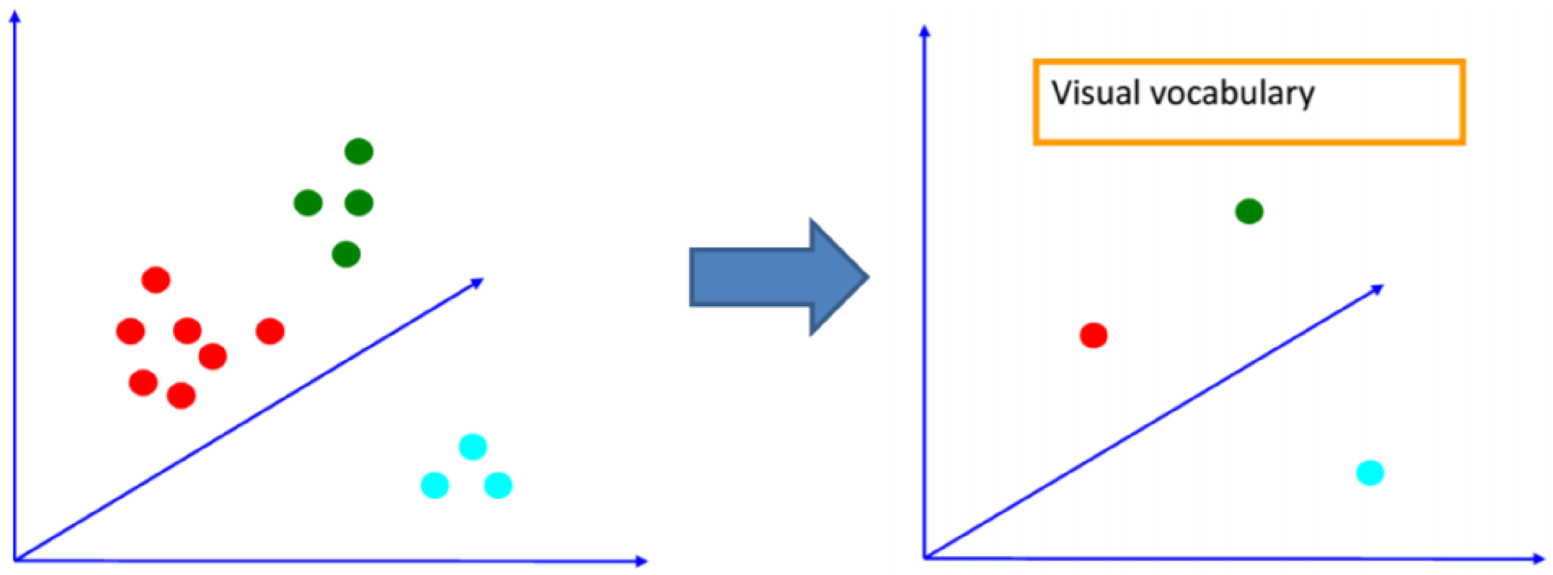
Example: Bag of Words

Dictionary learning



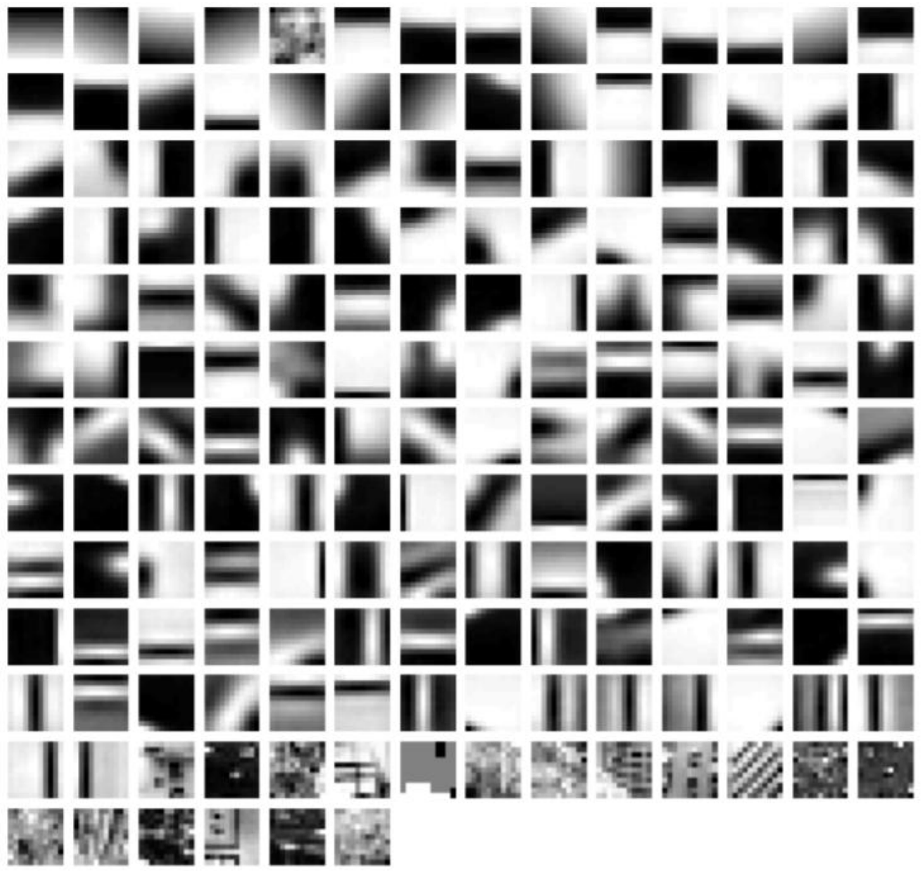
Example: Bag of Words

Dictionary learning



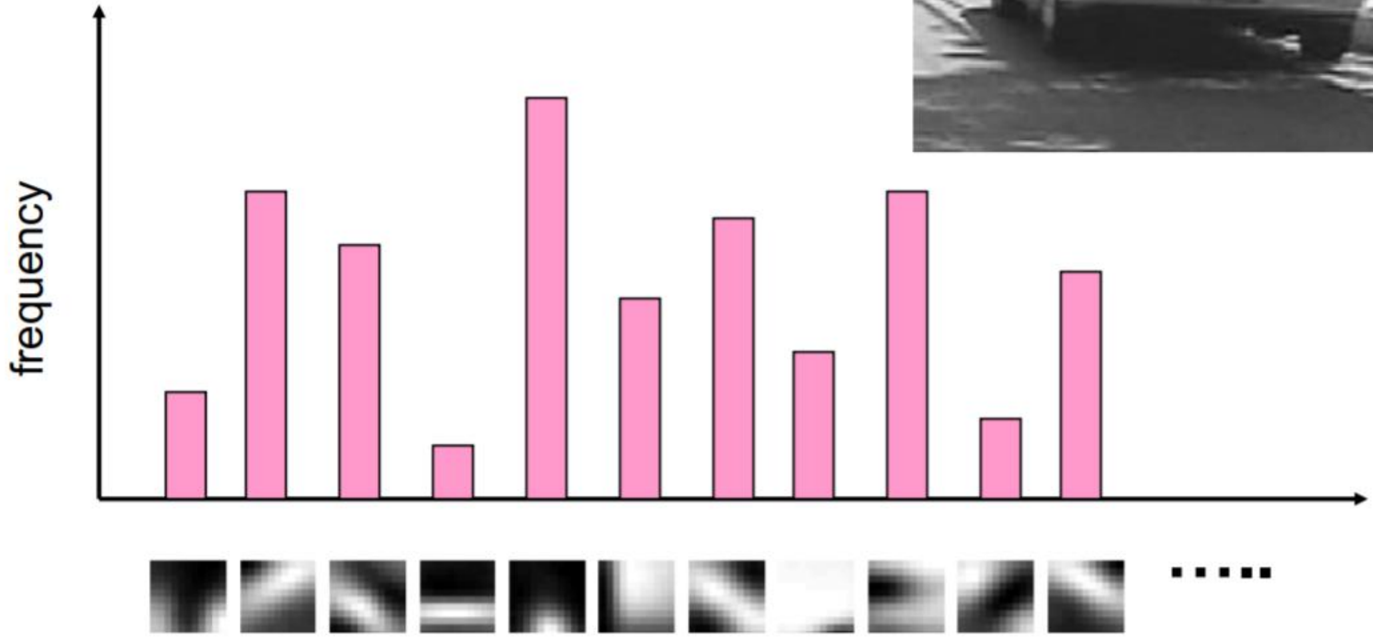
Example: Bag of Words

Example visual words



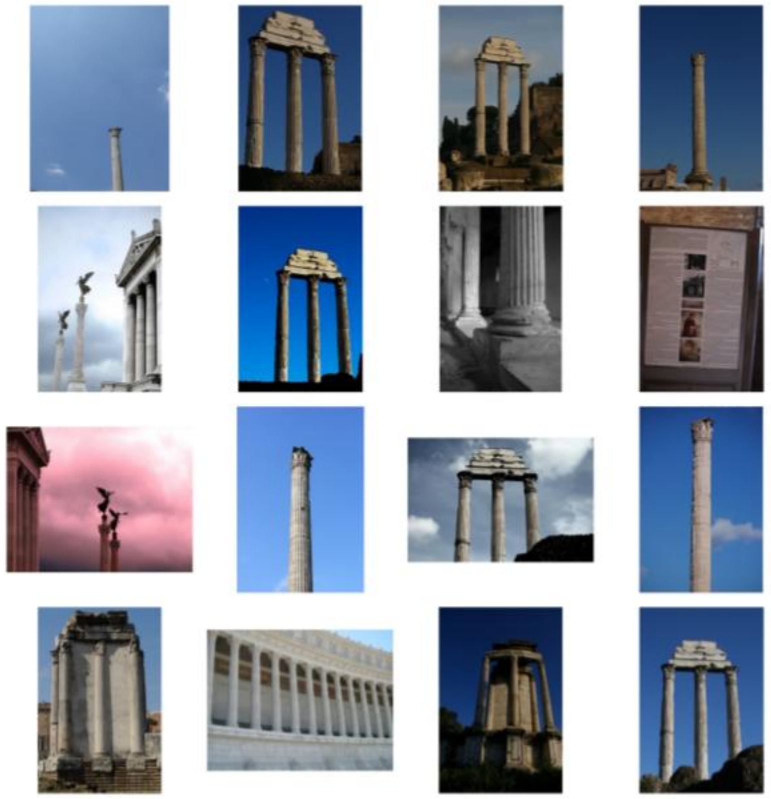
Example: Bag of Words

Image representation



Example: Bag of Words

Application – image retrieval



Expectation Maximization (EM)

When to use:

- Data is only partially observable
- Unsupervised learning, e.g., clustering (class value “unobservable”)
- Supervised learning (some instance attributes unobservable)

Some uses:

- Train Bayesian Belief Networks
- Unsupervised clustering (k-means, AUTOCLASS)
- Learning Hidden Markov Models (Baum-Welch algorithm)

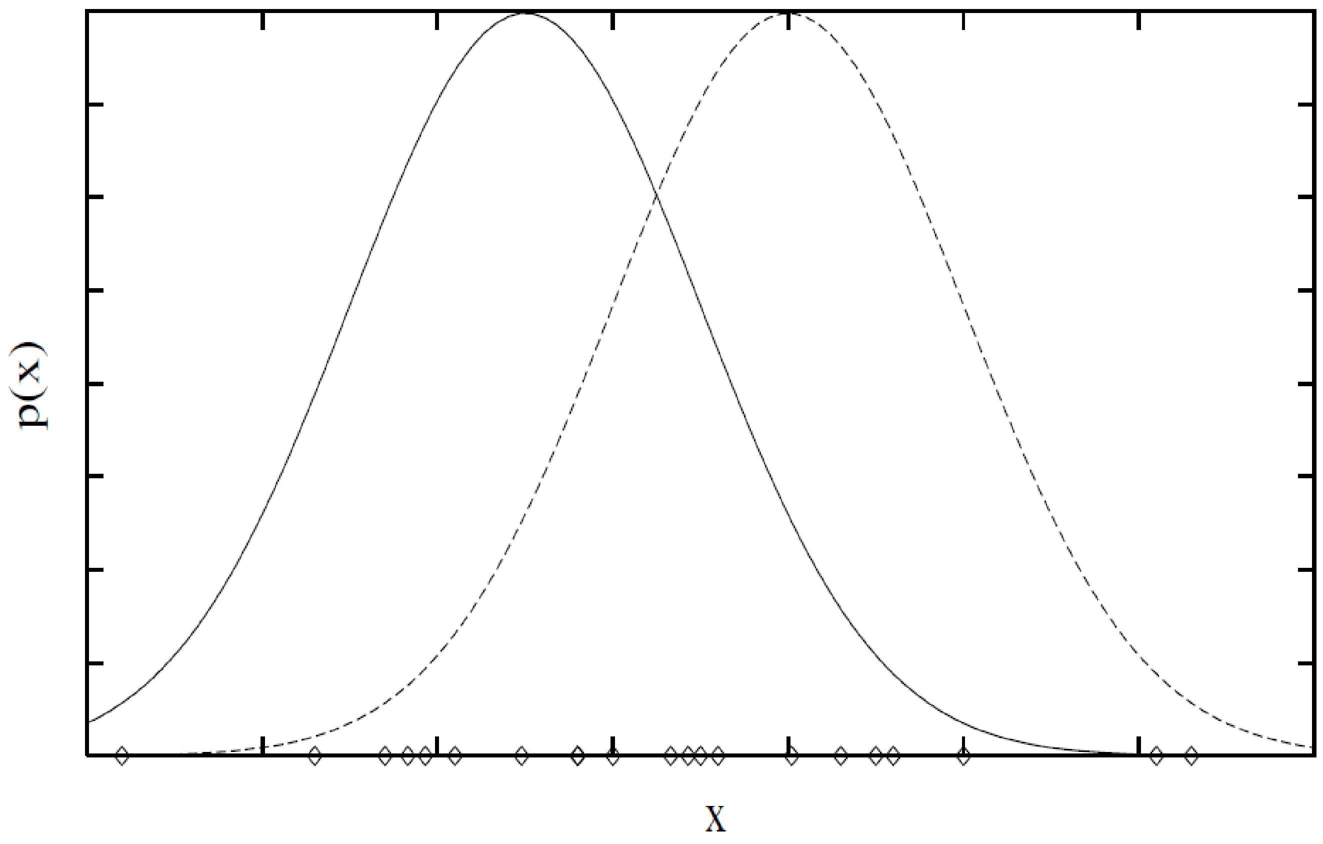
Finite Mixtures

Each instance x generated by

- Choosing one of the k Gaussians with uniform probability
- Generating an instance at random according to that Gaussian

Called *finite mixtures* because there is only a finite number of *generating distributions* being represented.

Generate data from mixture of k Gaussians



EM for estimating k means

Given:

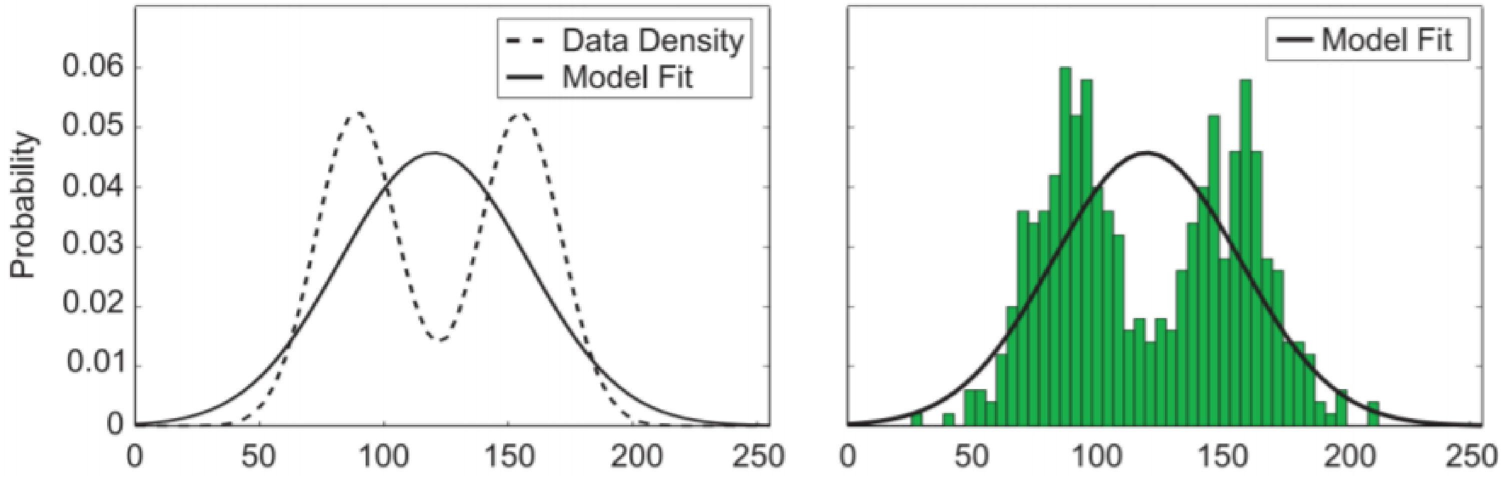
- Instances from X generated by mixture of k Gaussian distributions
- Unknown means $(\mu_1, \ldots, \mu_k)$ of the k Gaussians
- Don’t know which instance x_i was generated by which Gaussian

Determine:

- Maximum likelihood estimates of $(\mu_1, \ldots, \mu_k)$

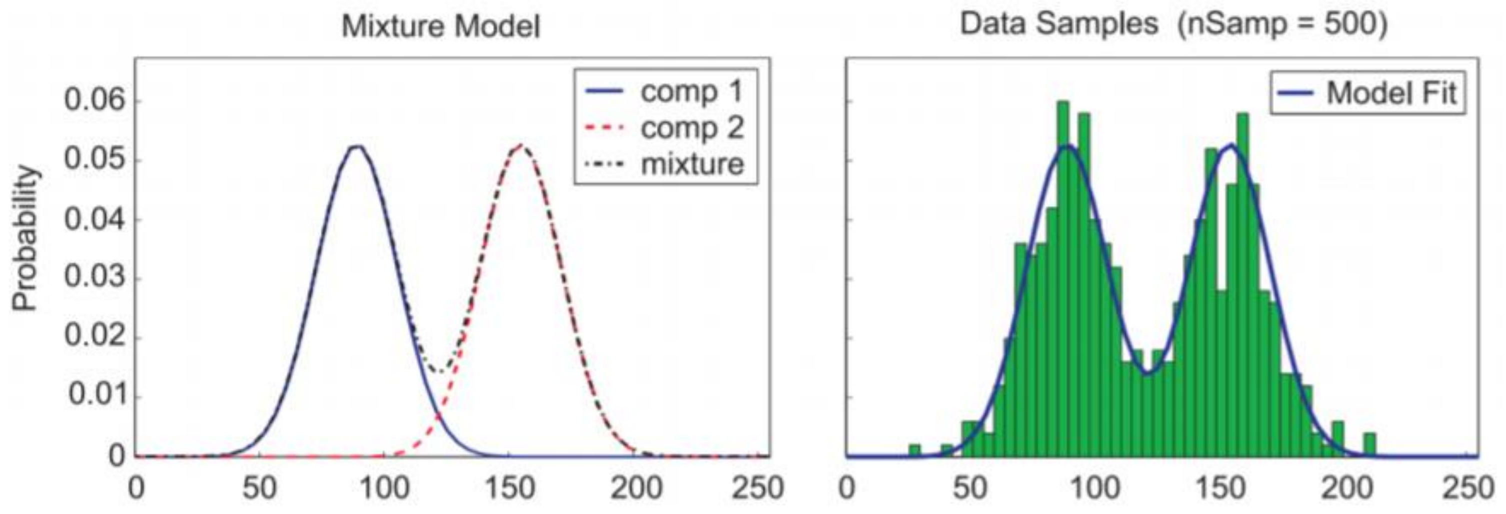
EM for estimating k means

- If you fit a Gaussian to data:



EM for estimating k means

- Now, we are trying to fit a GMM (with $K = 2$ in this example):



EM for estimating k means

Think of full description of each instance as $y_i = (x_i, z_{i1}, z_{i2})$, where

- z_{ij} is 1 if x_i generated by jth Gaussian, otherwise zero
- x_i is observable, from instance set $x_1, x_2, \ldots, x_m$
- z_{ij} is unobservable

EM for estimating k means

Initialise: Pick random initial $h = \langle \mu_1, \mu_2 \rangle$

Iterate:

E step:

Calculate expected value $E[z_{ij}]$ of each hidden variable z_{ij}, *assuming* current hypothesis $h = \langle \mu_1, \mu_2 \rangle$ holds:

$$\begin{aligned} E[z_{ij}] &= \frac{p(x = x_i | \mu = \mu_j)}{\sum_{n=1}^{2} p(x = x_i | \mu = \mu_n)} \\ &= \frac{e^{-\frac{1}{2\sigma^2}(x_i - \mu_j)^2}}{\sum_{n=1}^{2} e^{-\frac{1}{2\sigma^2}(x_i - \mu_n)^2}} \end{aligned}$$

EM for estimating k means

M step:

Calculate new maximum likelihood hypothesis $h' = \langle \mu'_1, \mu'_2 \rangle$, *assuming* value taken on by each hidden variable z_{ij} is the expected value $E[z_{ij}]$ calculated before.

Replace $h = \langle \mu_1, \mu_2 \rangle$ by $h' = \langle \mu'_1, \mu'_2 \rangle$.

$$\mu_j \leftarrow \frac{\sum_{i=1}^{m} E[z_{ij}] \;\; x_i}{\sum_{i=1}^{m} E[z_{ij}]}$$

i.e.

$$\mu_j \leftarrow \frac{1}{m} \sum_{i=1}^{m} E[z_{ij}] x_i$$

EM for estimating k means

E step: Calculate probabilities for unknown parameters for each instance

M step: Estimate parameters based on the probabilities

In k-means the probabilities are stored as instance weights.
EM produces soft assignments (probabilities) of data points into clusters.

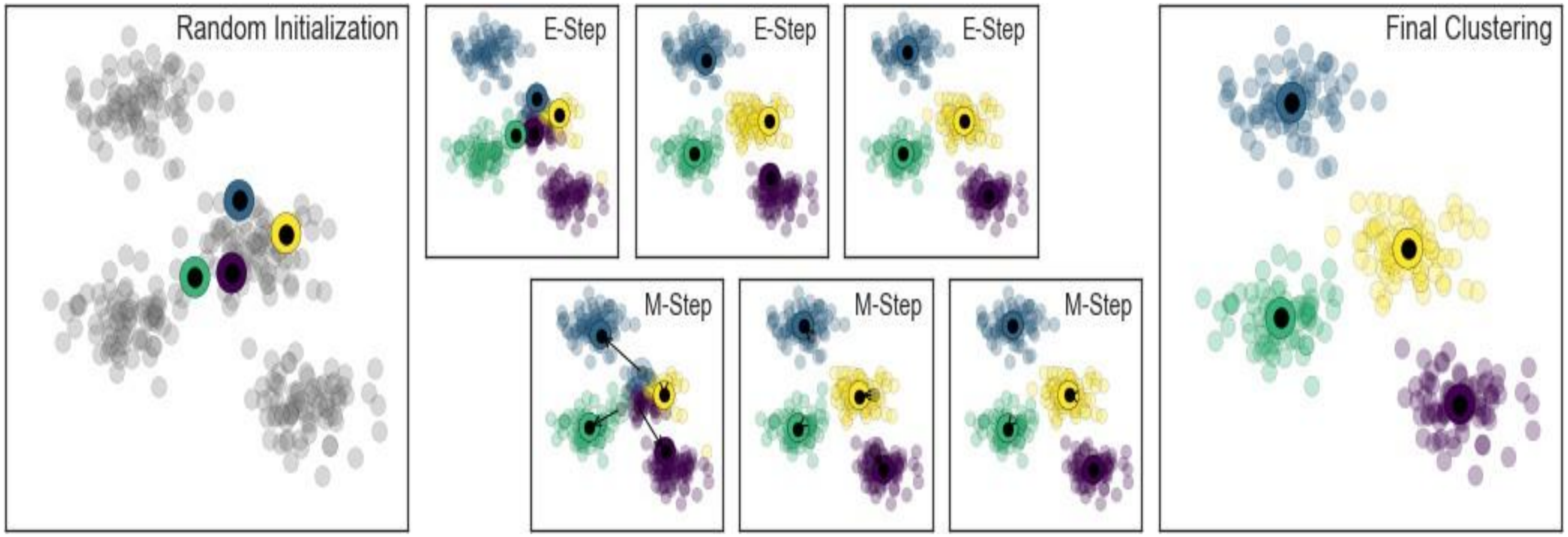
EM Algorithm

Converges to local maximum likelihood h
and provides estimates of hidden variables z_{ij}

In fact, local maximum in $E[\ln P(Y|h)]$

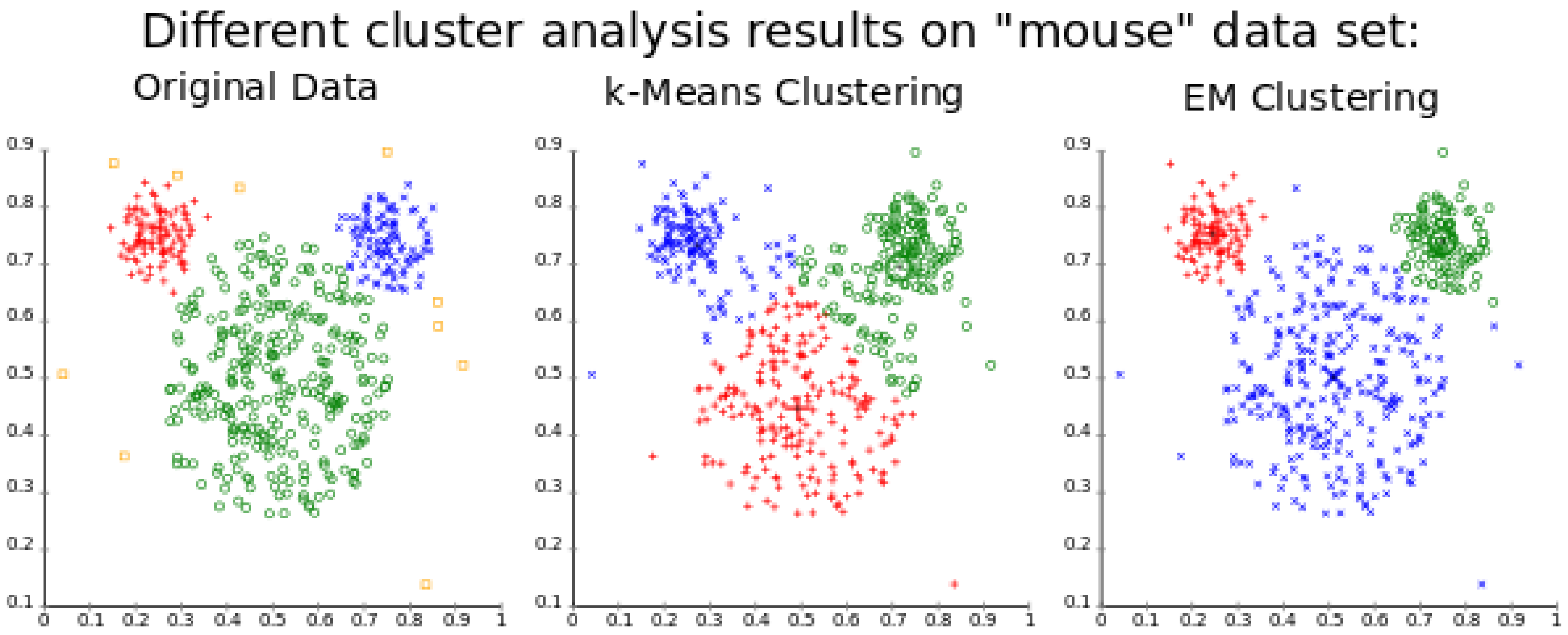
- Y is complete (observable plus unobservable variables) data
- Expected value taken over possible values of unobserved variables in Y

Clustering in EM



https://jakevdp.github.io/PythonDataScienceHandbook/05.11-k-means.html.

k-means clustering vs. EM



https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm

Extending the mixture model

- Using more than two distributions
- Several attributes: easy if independence assumed
- Correlated attributes: difficult
  - Modeled jointly using a bivariate normal distribution with a (symmetric) covariance matrix
  - With n attributes this requires estimating $n + n(n + 1)/2$ parameters

Extending the mixture model

- Nominal attributes: easy if independence assumed

- Correlated nominal attributes: difficult
  - Two correlated attributes result in $v_1 \times v_2$ parameters

- Missing values: easy

- Distributions other than the normal distribution can be used:
  - "log-normal" if predetermined minimum is given
  - "log-odds" if bounded from above and below
  - Poisson for attributes that are integer counts

- Cross-validation can be used to estimate k – time consuming !

General EM Problem

Given:

- Observed data $X = \{x_1, \ldots, x_m\}$
- Unobserved data $Z = \{z_1, \ldots, z_m\}$
- Parameterized probability distribution $P(Y \mid h)$, where
  - $Y = \{y_1, \ldots, y_m\}$ is the full data $y_i = x_i \cup z_i$
  - h are the parameters

Determine:

- h that (locally) maximises $E[\ln P(Y \mid h)]$

General GM Method

Define likelihood function $Q(h' \mid h)$ which calculates $Y = X \cup Z$ using observed X and current parameters h to estimate Z

$$Q(h' \mid h) \leftarrow E[\ln P(Y \mid h') \mid h, X]$$

General GM Method

- *Estimation (E) step:* Calculate $Q(h'|h)$ using the current hypothesis h and the observed data X to estimate the probability distribution over Y.

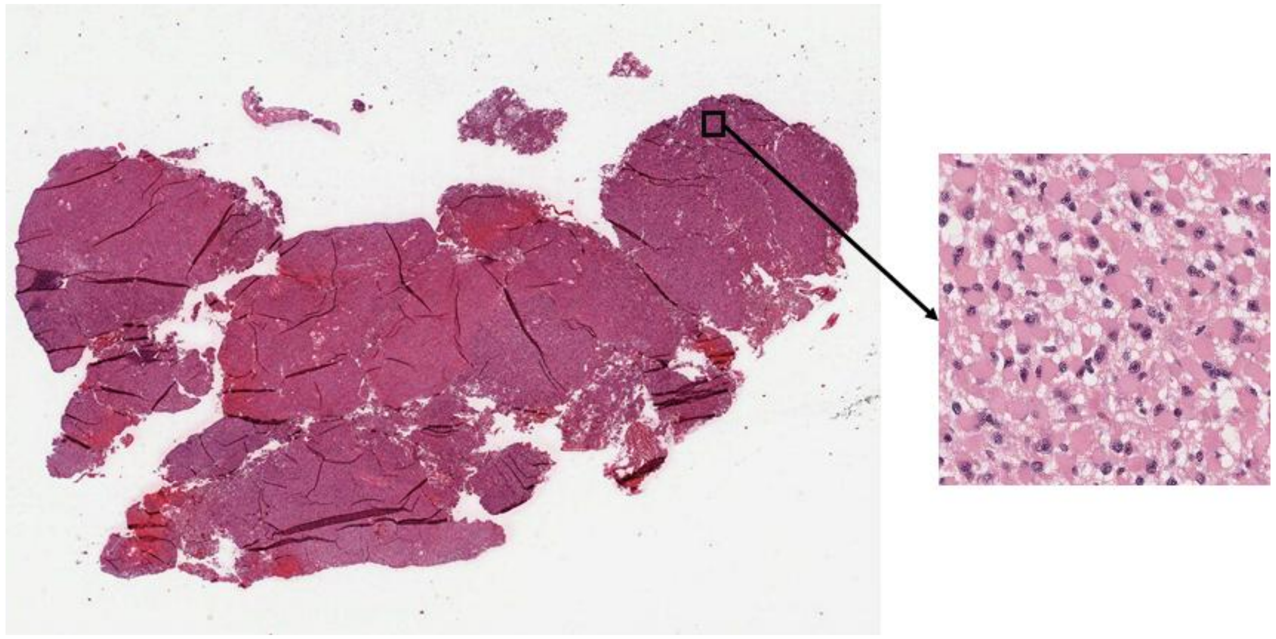
$$Q(h'|h) \leftarrow E[\ln P(Y|h')|h, X]$$

- *Maximization (M) step:* Replace hypothesis h by the hypothesis h' that maximises this Q function.

$$h \leftarrow \arg\max_{h'} Q(h'|h)$$

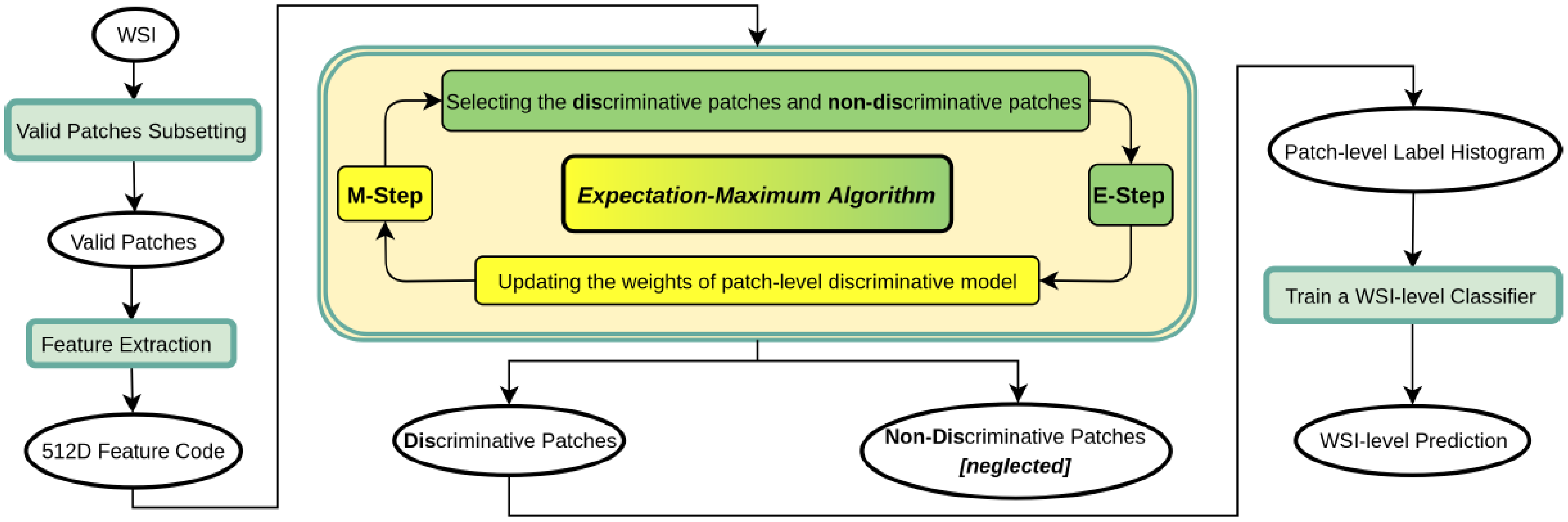
Example: WSI Analysis

Tumour classification in WSI



Example: WSI Analysis

Discriminative patch-based CNN

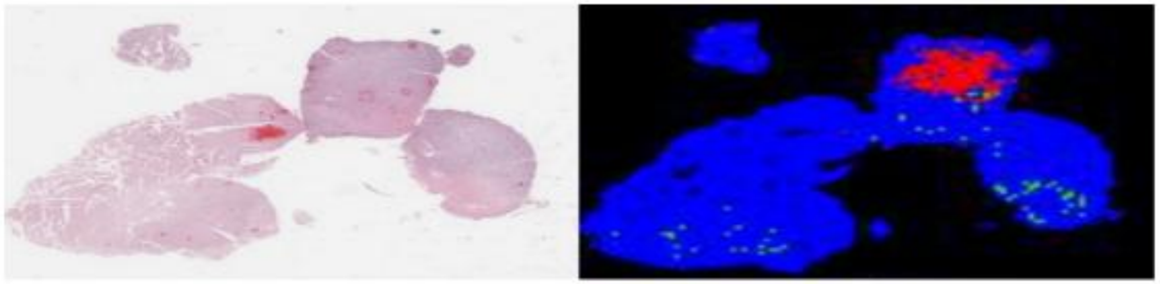


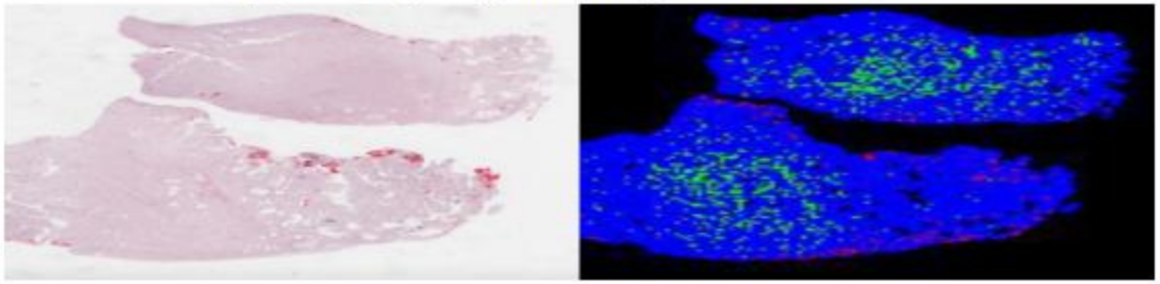
WSI
Valid Patches Subsetting
Valid Patches
Feature Extraction
512D Feature Code
Selecting the **dis**criminative patches and **non-dis**criminative patches
M-Step
Expectation-Maximum Algorithm
E-Step
Updating the weights of patch-level discriminative model
Discriminative Patches
Non-Discriminative Patches *[neglected]*
Patch-level Label Histogram
Train a WSI-level Classifier
WSI-level Prediction

Source: C. Zhang et al. *Whole slide image classification via iterative patch labelling*. ICIP, 2018.

Example: WSI Analysis

Discriminative patch-based CNN


(a) Testing oligodendroglioma instance


(b) Testing astrocytoma instance

Source: C. Zhang et al. *Whole slide image classification via iterative patch labelling*. ICIP, 2018.

Summary

Clustering is a typical unsupervised learning method

k-means clustering is one of the most well-known clustering techniques

EM algorithm can be used to estimate k-means

Next lecture:

– hierarchical clustering, dimensionality reduction, semi-supervised learning

Acknowledgement

Material derived from slides for the book
"Elements of Statistical Learning (2nd Ed.)" by T. Hastie,
R. Tibshirani & J. Friedman. Springer (2009)
http://statweb.stanford.edu/~tibs/ElemStatLearn/

Material derived from slides for the book
"Machine Learning: A Probabilistic Perspective" by P. Murphy MIT Press (2012)
http://www.cs.ubc.ca/~murphyk/MLbook

Material derived from slides for the book "Machine Learning" by P. Flach Cambridge University Press (2012)
http://cs.bris.ac.uk/~flach/mlbook

Material derived from slides for the book
"Bayesian Reasoning and Machine Learning" by D. Barber Cambridge University Press (2012)
http://www.cs.ucl.ac.uk/staff/d.barber/brml

Material derived from figures for the book
"Python Data Science Handbook" by J. VanderPlas O'Reilly Media (2017)
http://shop.oreilly.com/product/0636920034919.do

Material derived from slides for the course "Machine Learning" by A. Srinivasan BITS Pilani, Goa, India (2016)

http://people.csail.mit.edu/dsontag/courses/ml12/slides/lecture14.pdf

http://www.mit.edu/~9.54/fall14/slides/Class13.pdf

http://vision.stanford.edu/teaching/cs131_fall1718/files/14_BoW_bayes.pdf

https://www.cs.toronto.edu/~jlucas/teaching/csc411/lectures/lec15_16_handout.pdf