Never Stand Still

Unsupervised Learning

COMP9417 Machine Learning & Data Mining

Aims

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

- describe the problem of unsupervised learning
- describe *k*-means clustering
- describe Gaussian Mixture Models (GMM)
- Outline Expectation Maximization (EM) algorithm for k-means which is a specific case of EM
- describe hierarchical clustering
- describe evaluation methods for clustering
- describe the problem of dimensionality reduction
- outline the method of Principal Component Analysis
- outline semi-supervised learning

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

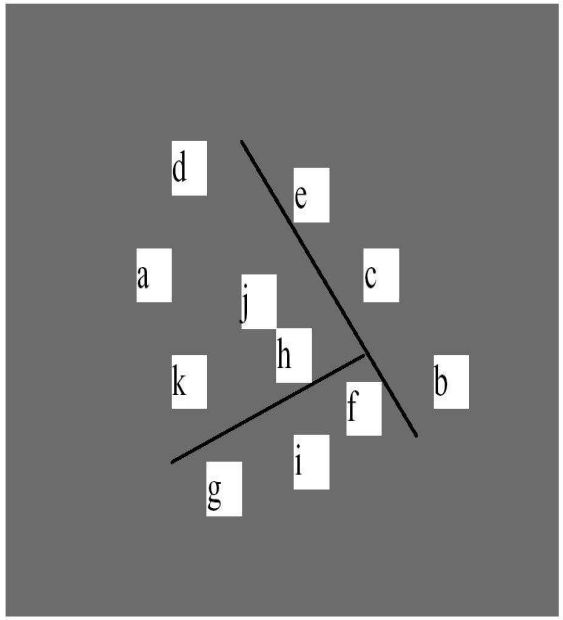
Success of clustering often measured subjectively

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

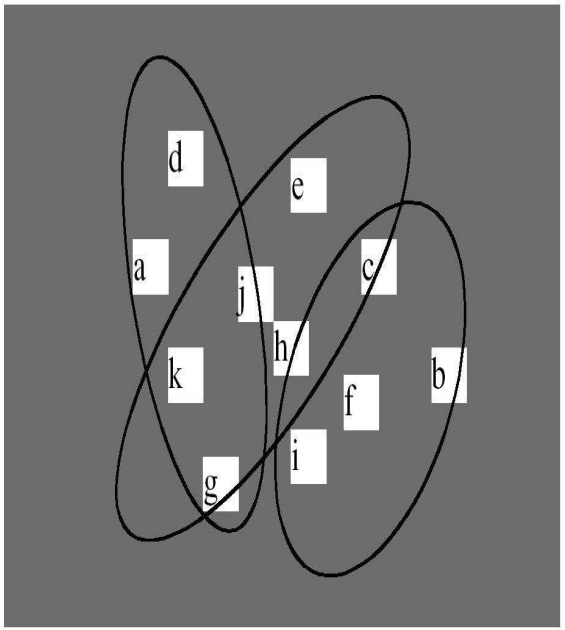
A dataset for clustering is just like a dataset for classification, but without the class labels

Simple 2D representations of clustering

Clusters form a partition



Venn diagram (overlapping clusters)

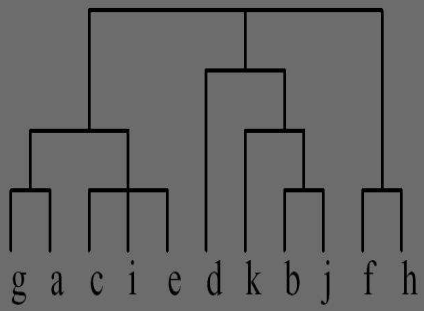


Other representations of clustering

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:

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

- Partitioning methods usually require specification of the number of clusters, then try to construct the clusters and fit objects to them.
- Hierarchical algorithms are either **agglomerative** i.e., bottom-up or

divisive i.e., top-down.

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

Representation

Let $X = \{x_1, \ldots, x_m\}$ be a set of instances.

Let $C = (C_1, \ldots, C_k)$ be a *partition* of m elements into k 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 n features e.g. measurement levels for different features
- pairwise similarity data between elements, e.g., correlation, distance (dissimilarity)

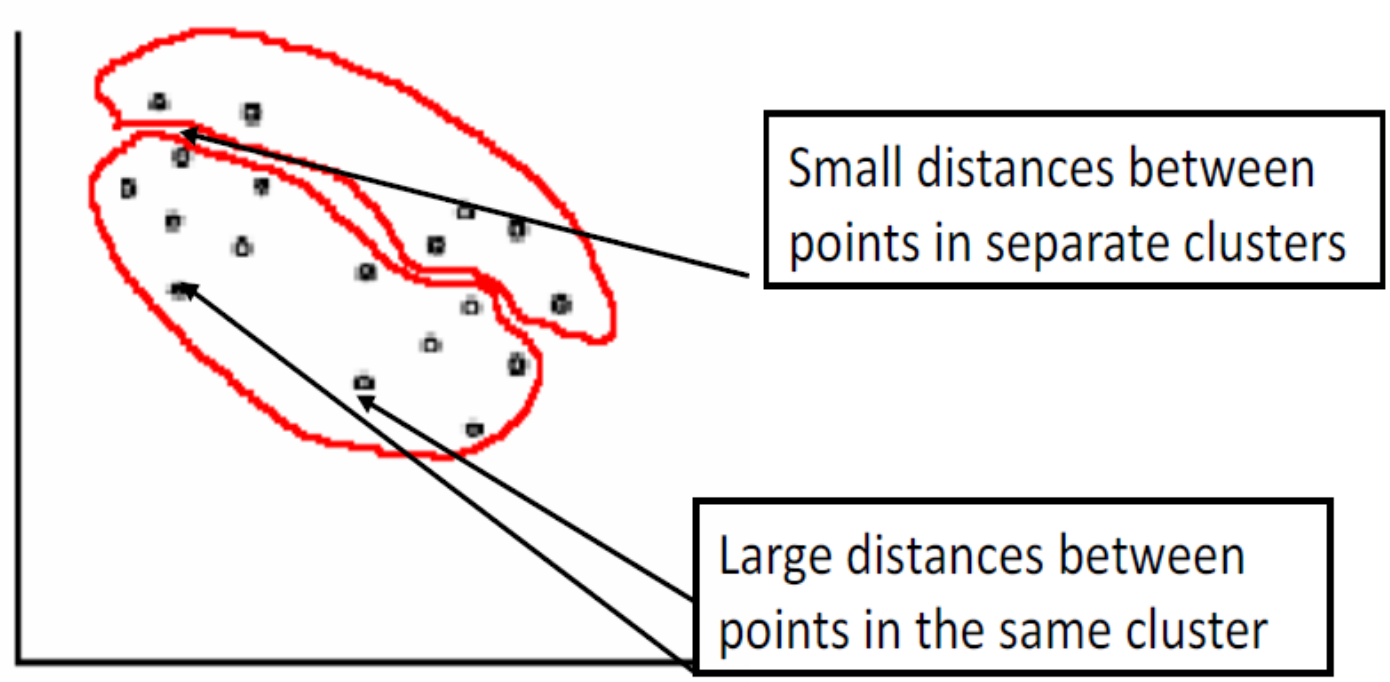
Feature-vectors have more information, but similarity is generic (given the appropriate function). Feature-vector matrix: $m \times n$, similarity matrix $m \times m$. In general, often $m \gg n$.

Clustering Framework

- Goal of clustering: find a partition of m 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

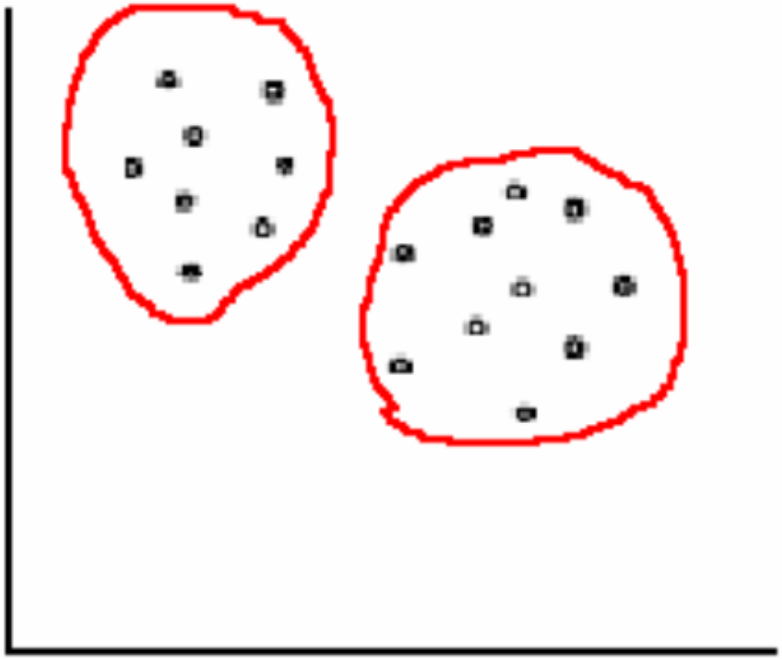
A bad clustering

This clustering violates both homogeneity and separation principles



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

k-means Clustering

P_i is the cluster assigned to element i (or x_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:

$$E_P = \sum_{i=1}^{m} d(\mathrm{x}_i, c_{p_i})$$

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

- k-means minimizes the within-cluster sum of squares.
- 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

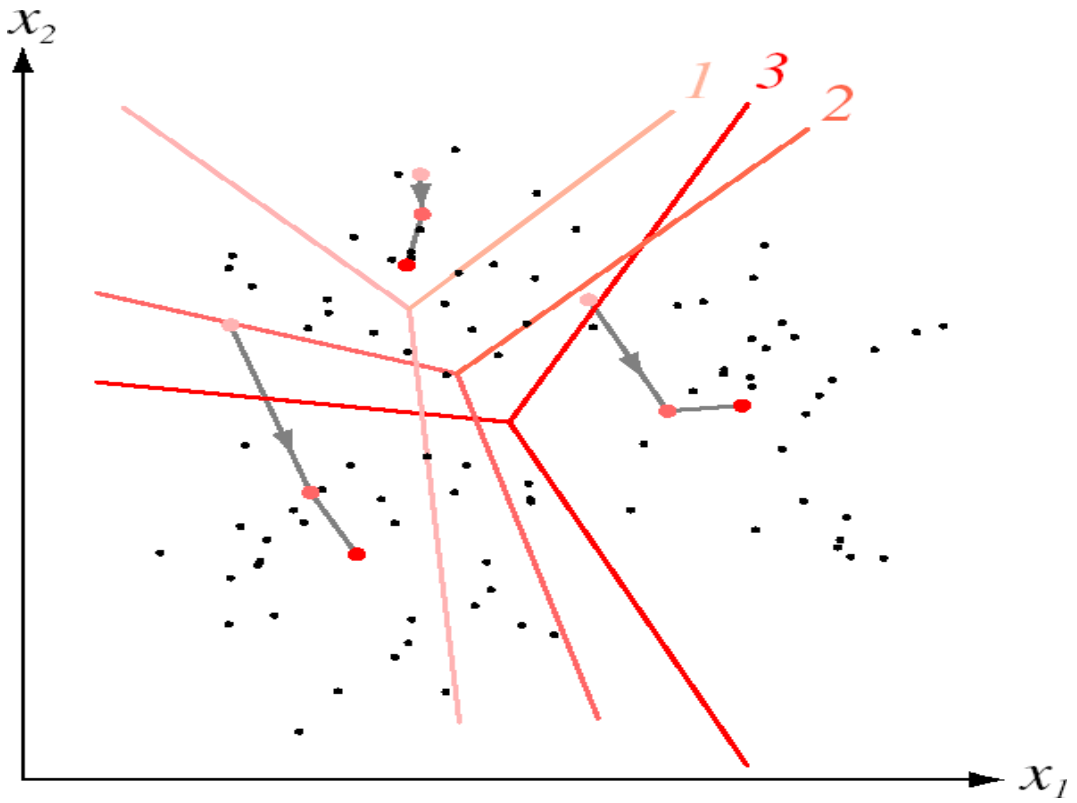
Here is another view of k-means clustering algorithm:

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

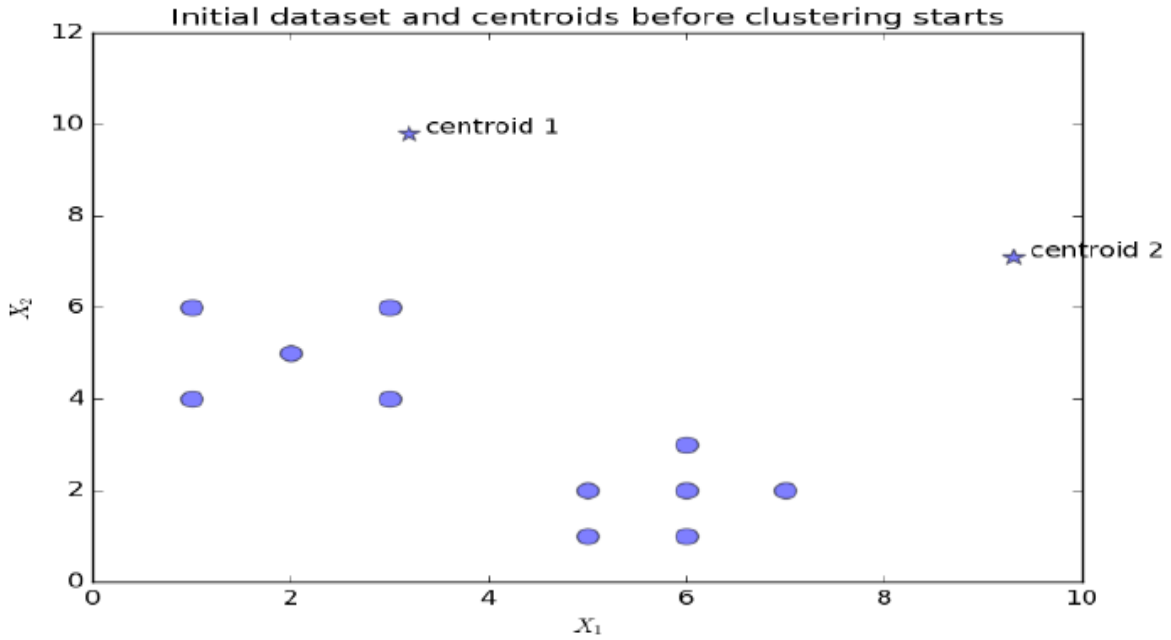
Previous diagram shows three steps to convergence in k-means with $k = 3$

- 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	-

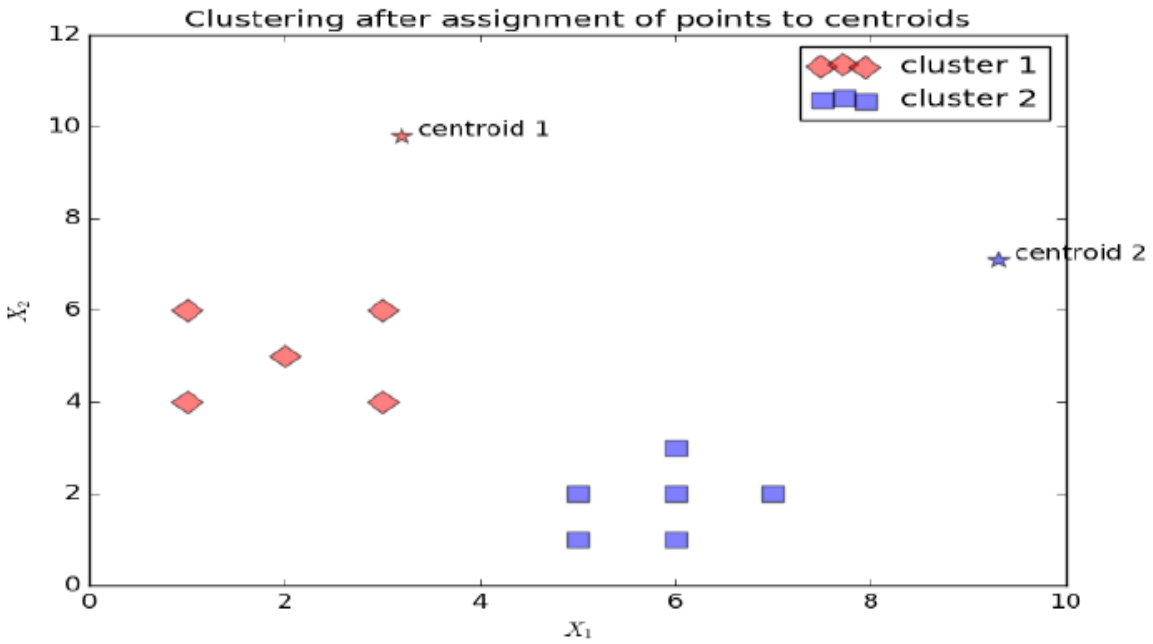
Centroid locations
centroid 1: $(3.2, 9.8)$
centroid 2: $(9.3, 7.1)$



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

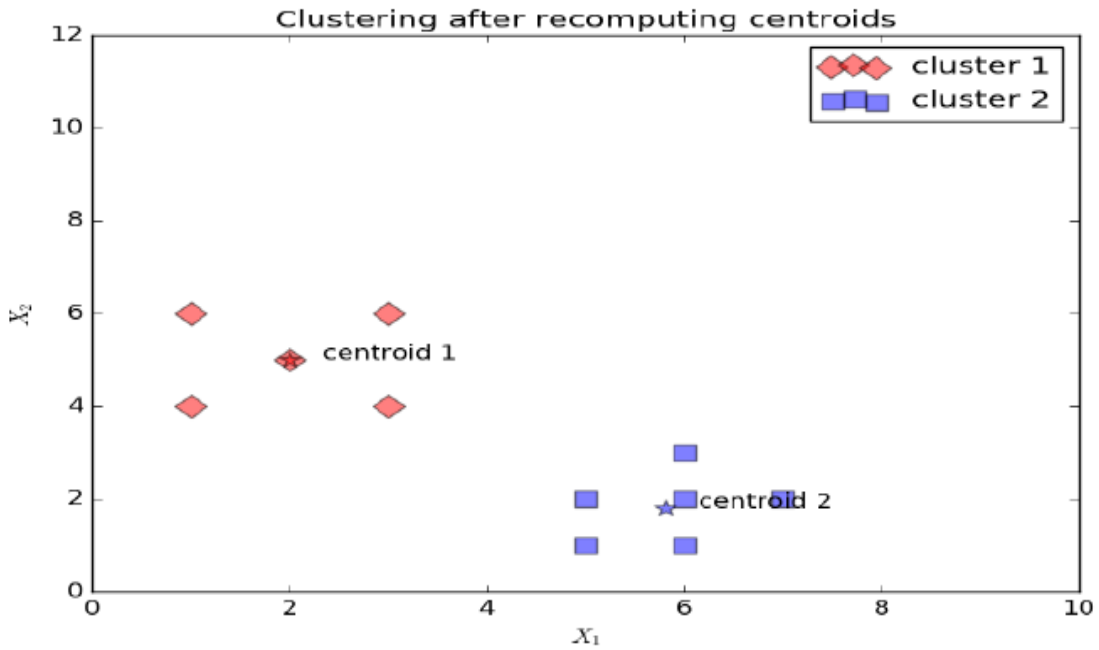
Centroid locations
centroid 1: $(3.2, 9.8)$
centroid 2: $(9.3, 7.1)$



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

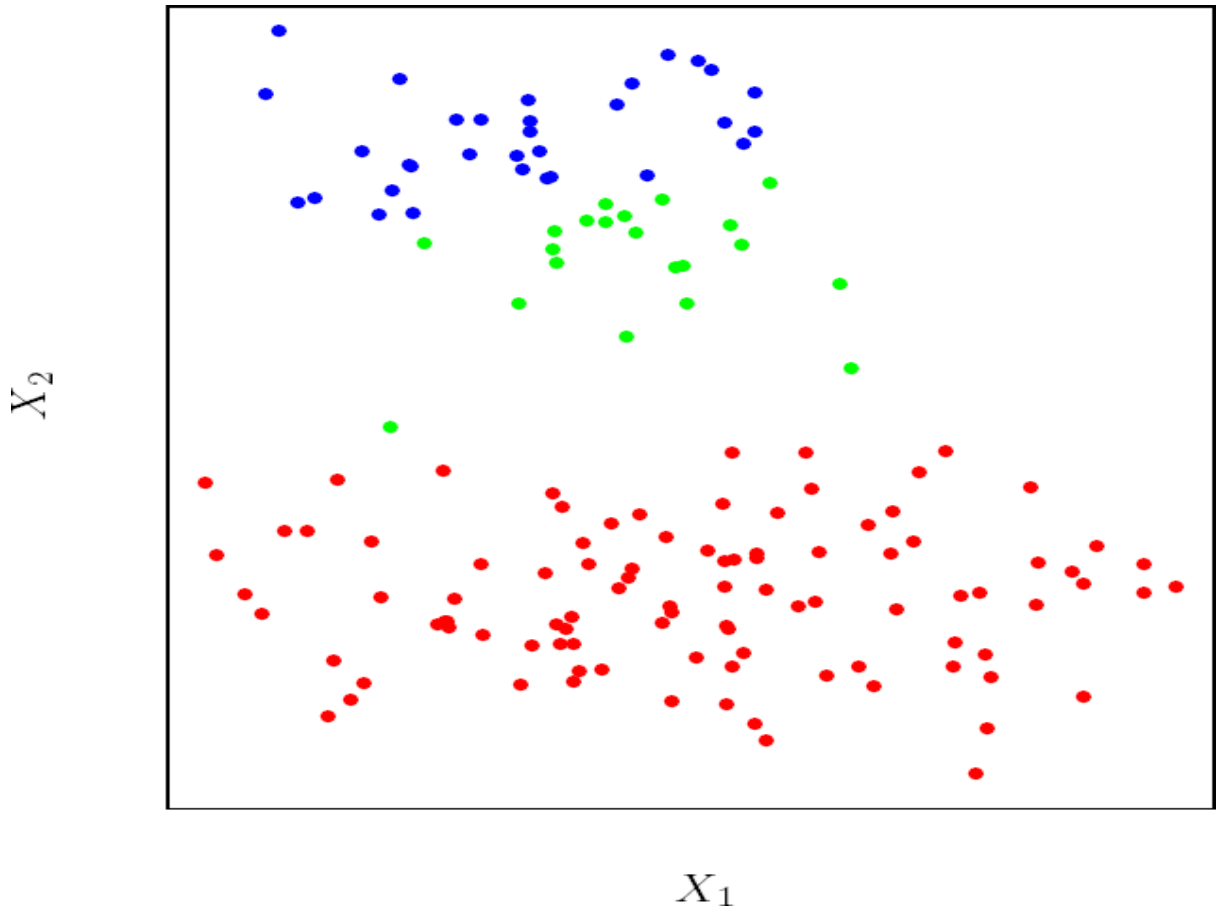
- the distribution of the points in the dataset
- the choice of k
- outliers
- the choice of the location to initialise the centroids.

k-means Clustering: parameter

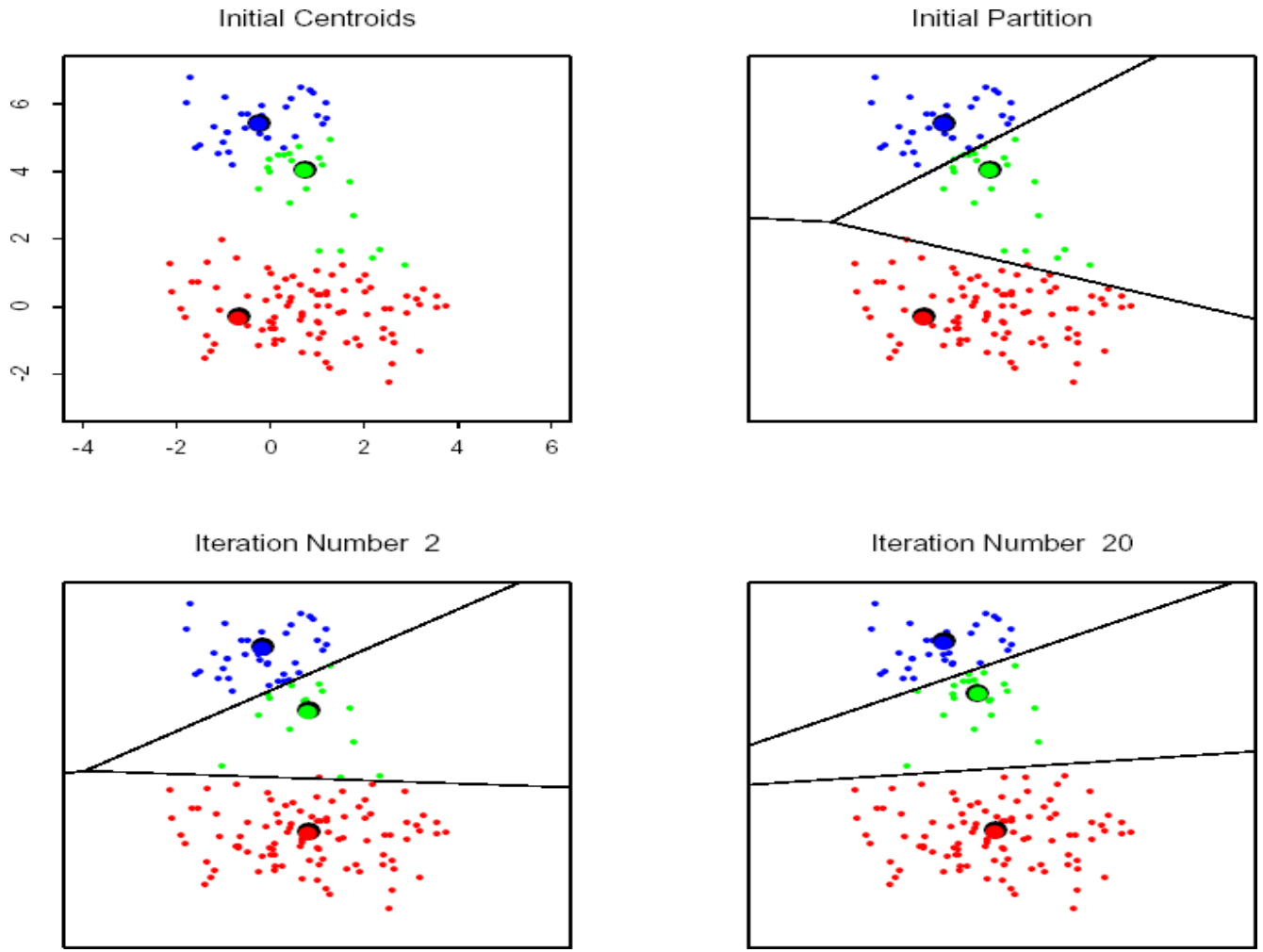
What about the number of clusters k ?

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

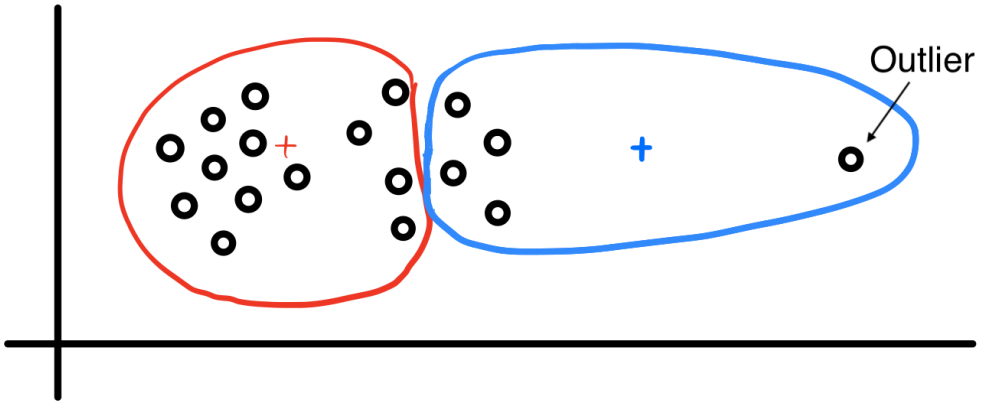
k-means Clustering: parameter



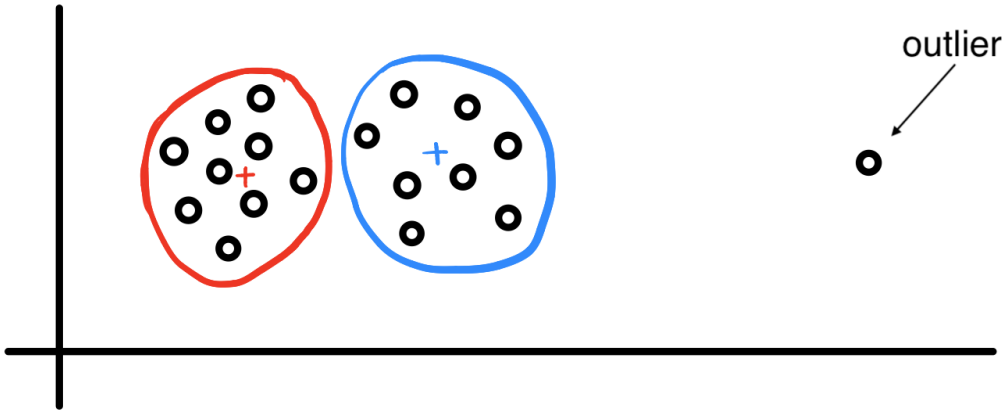
k-means Clustering: parameter



k-means Clustering: outliers



Undesirable clusters



Desirable clusters

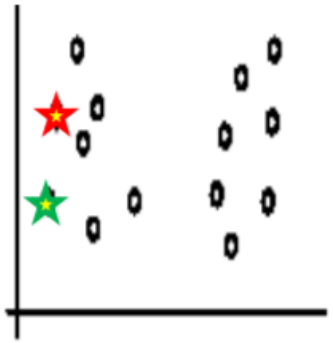
k-means Clustering: outliers

Deal with outliers:

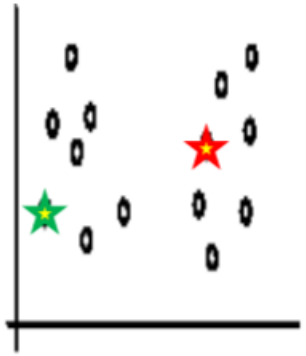
- Remove some data points that are much further away from the centroids than other data points
  - To be safe, we may want to monitor these possible outliers over a few iterations and then decide to remove them.

- 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

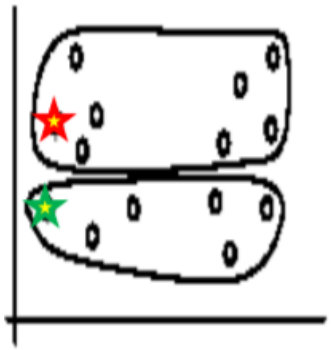
k-means Clustering: initial seeds



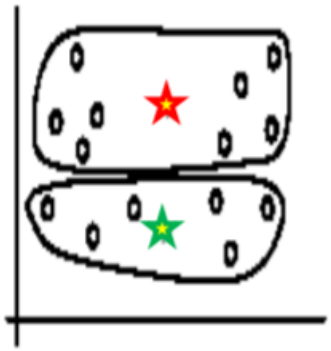
Random selection of seeds (centroids)



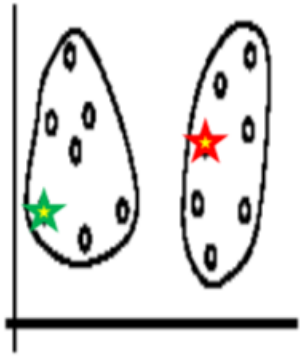
Random selection of seeds (centroids)



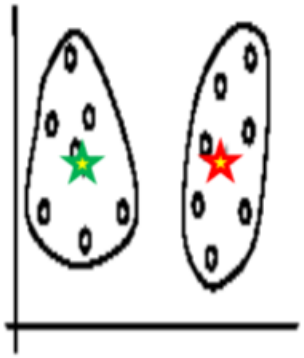
Iteration 1



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

K=2

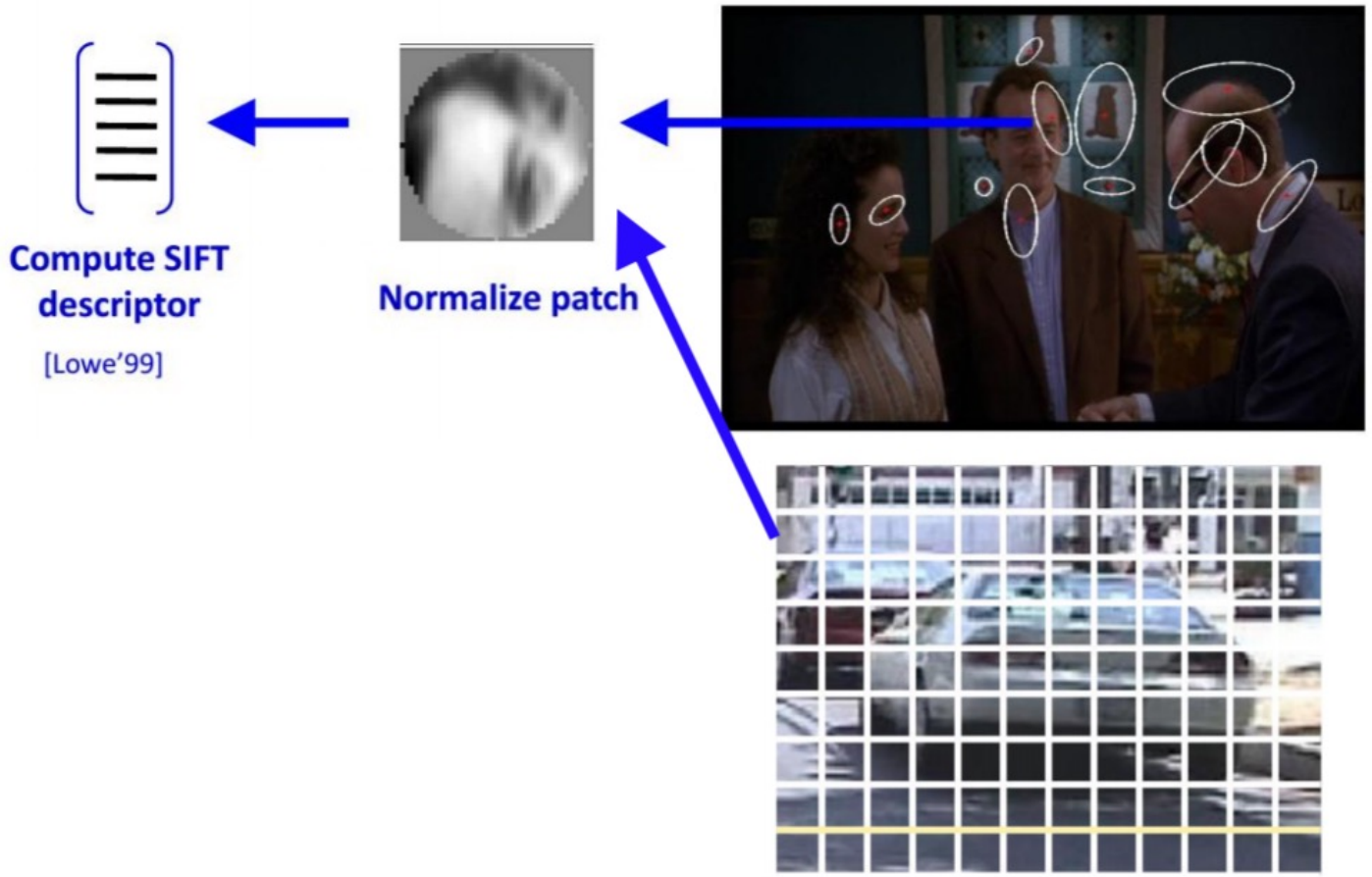
Goal of Segmentation is to partition an image into regions each of which has reasonably homogenous visual appearance.

Original

Example: Image Segmentation

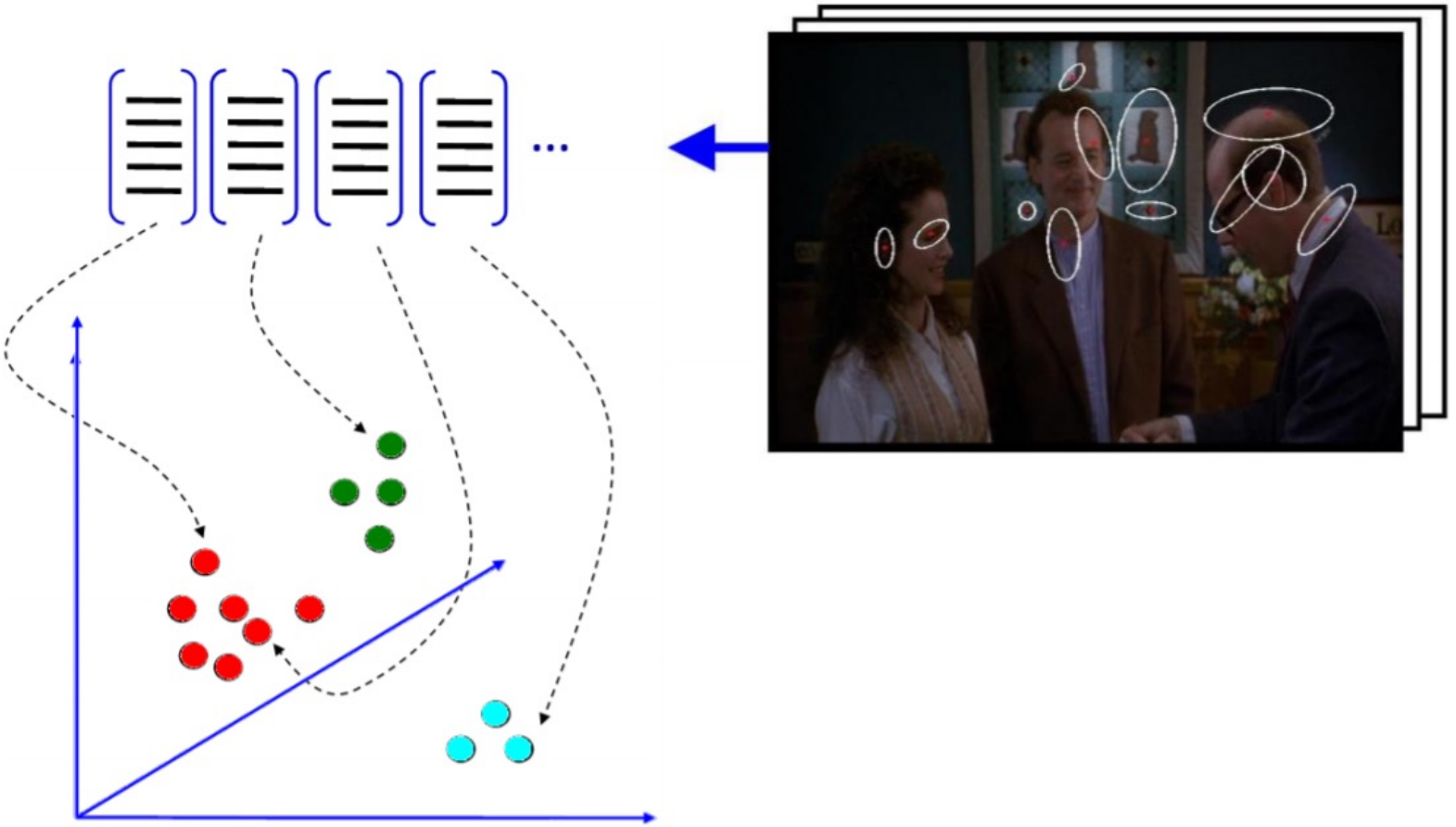
Example: Bag of Words

Feature extraction



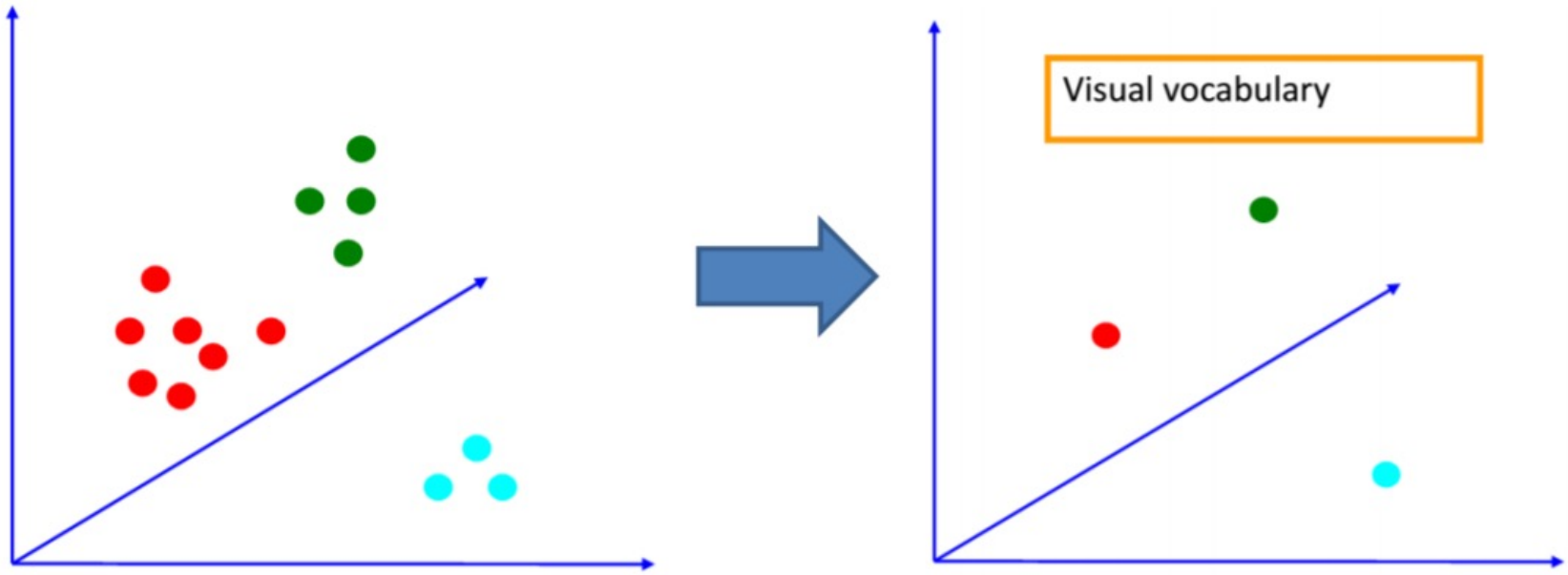
Example: Bag of Words

Dictionary learning



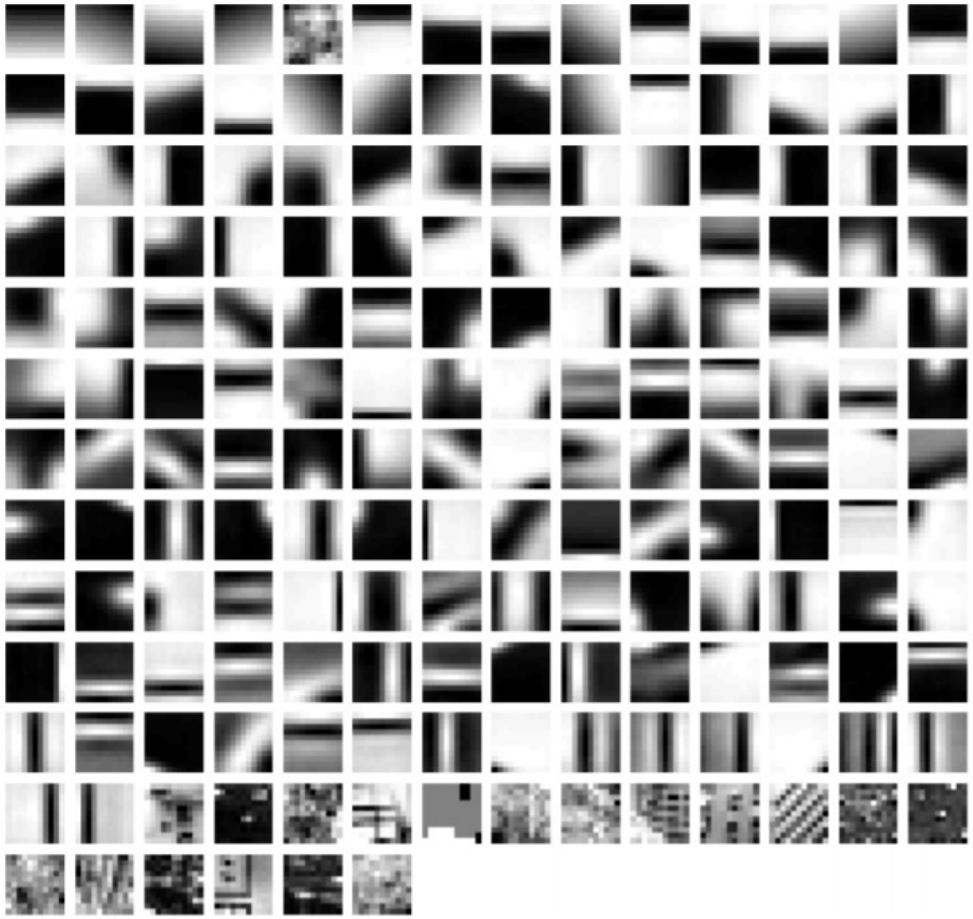
Example: Bag of Words

Dictionary learning



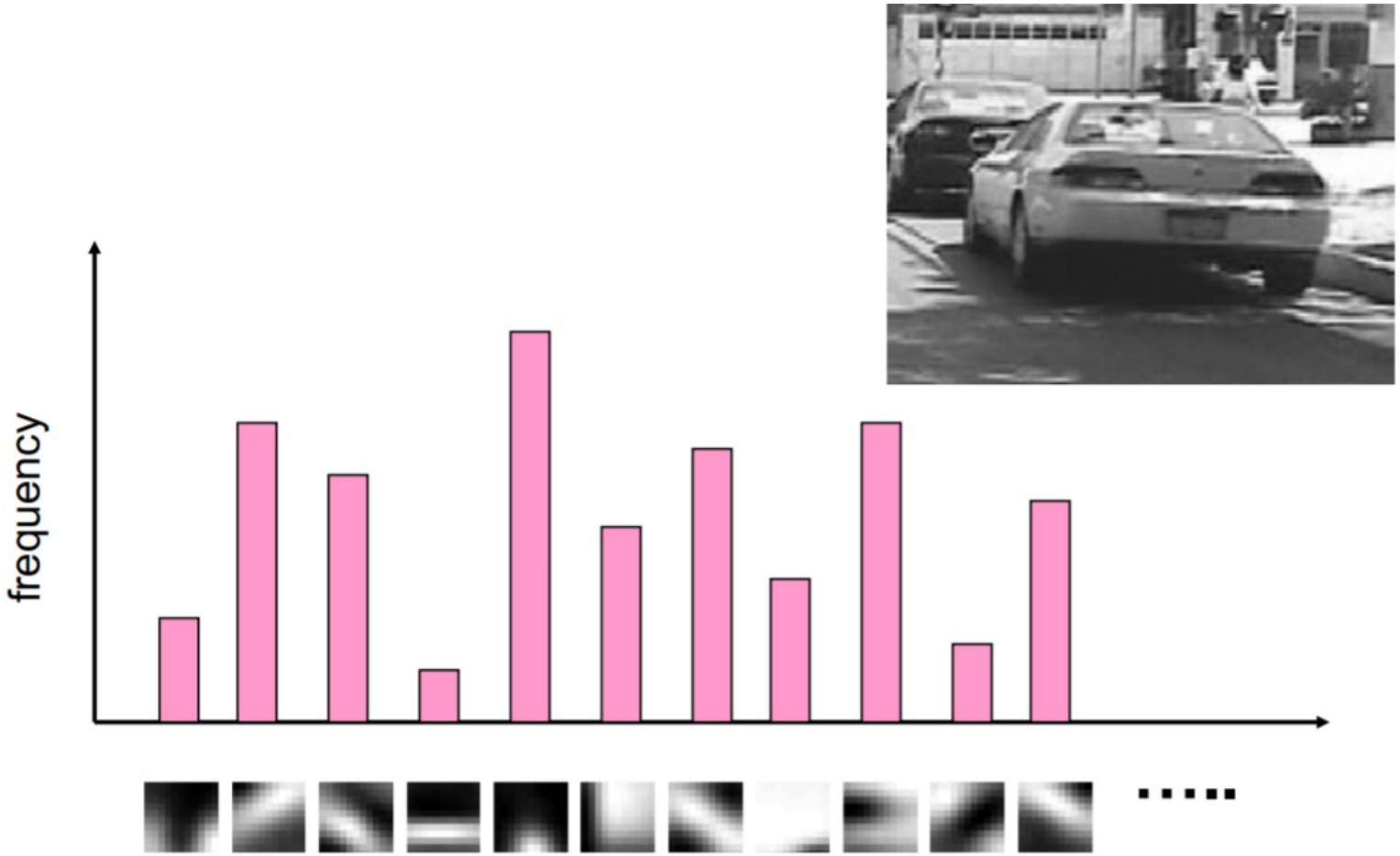
Example: Bag of Words

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)

Here we only focus on how to use EM for k-means which is a specific case of EM for Gaussian Mixture Models.

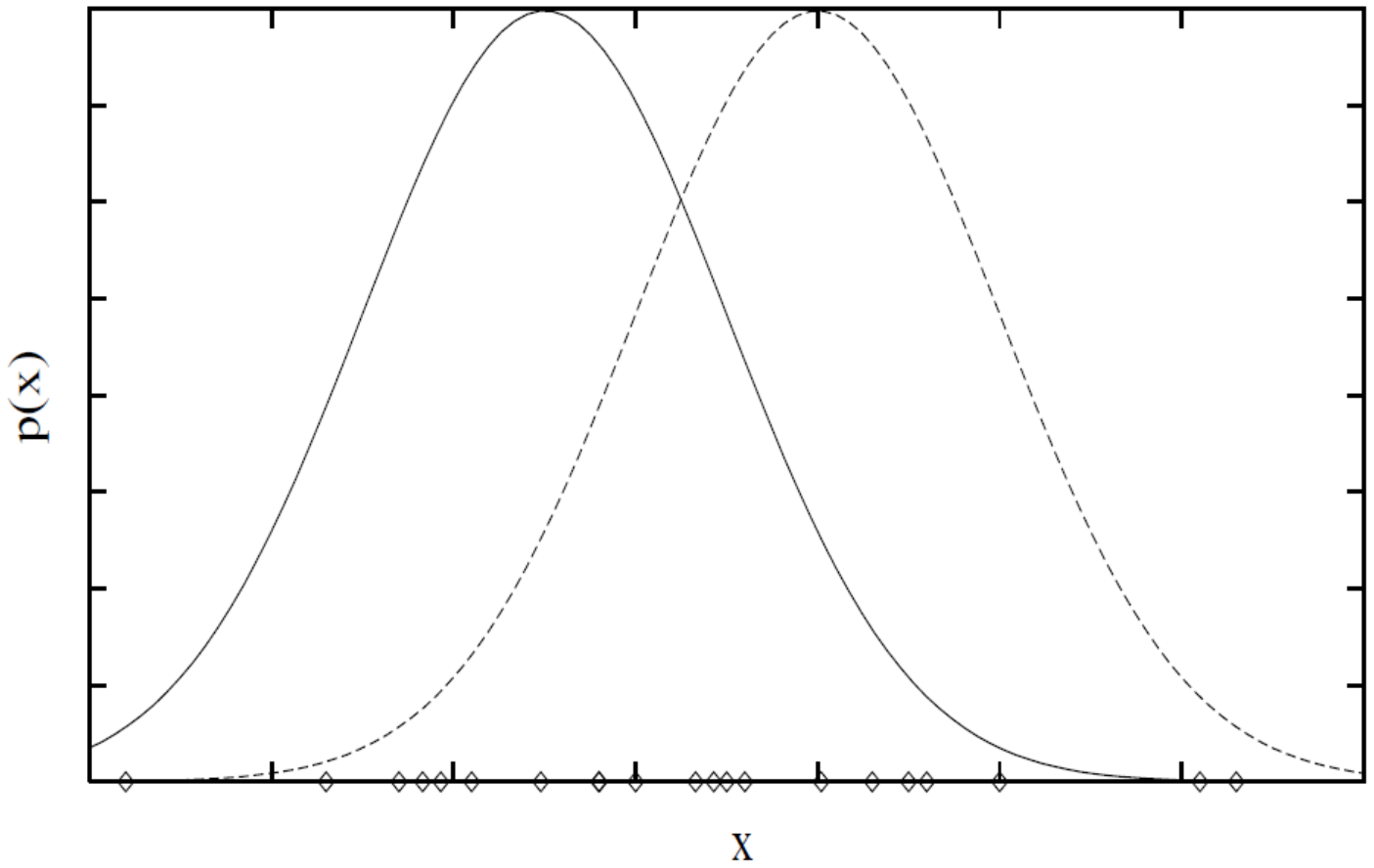
Finite Mixtures

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.

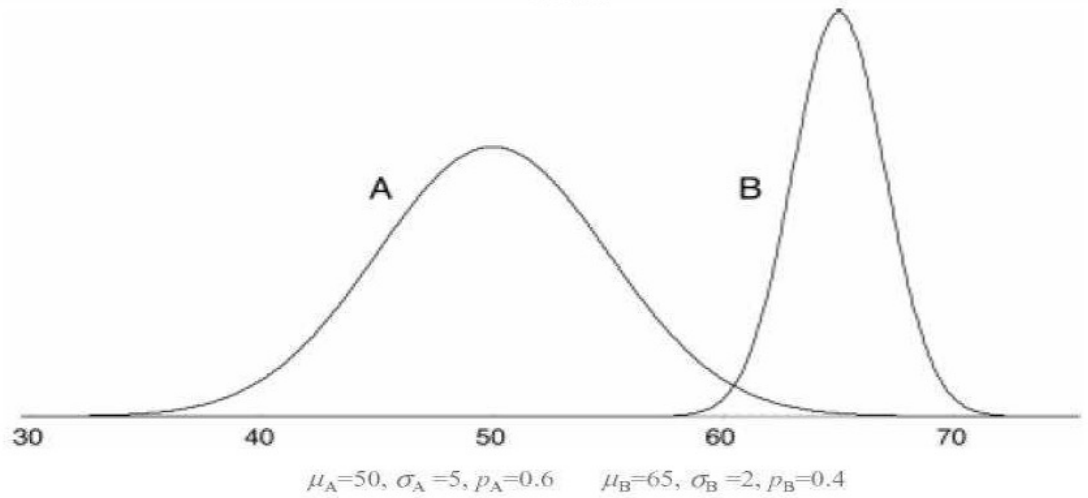
Generate Data from Mixture of Gaussians



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



μ_A=50, σ_A =5, p_A=0.6 μ_B=65, σ_B =2, p_B=0.4

EM for Estimating k means

Given:

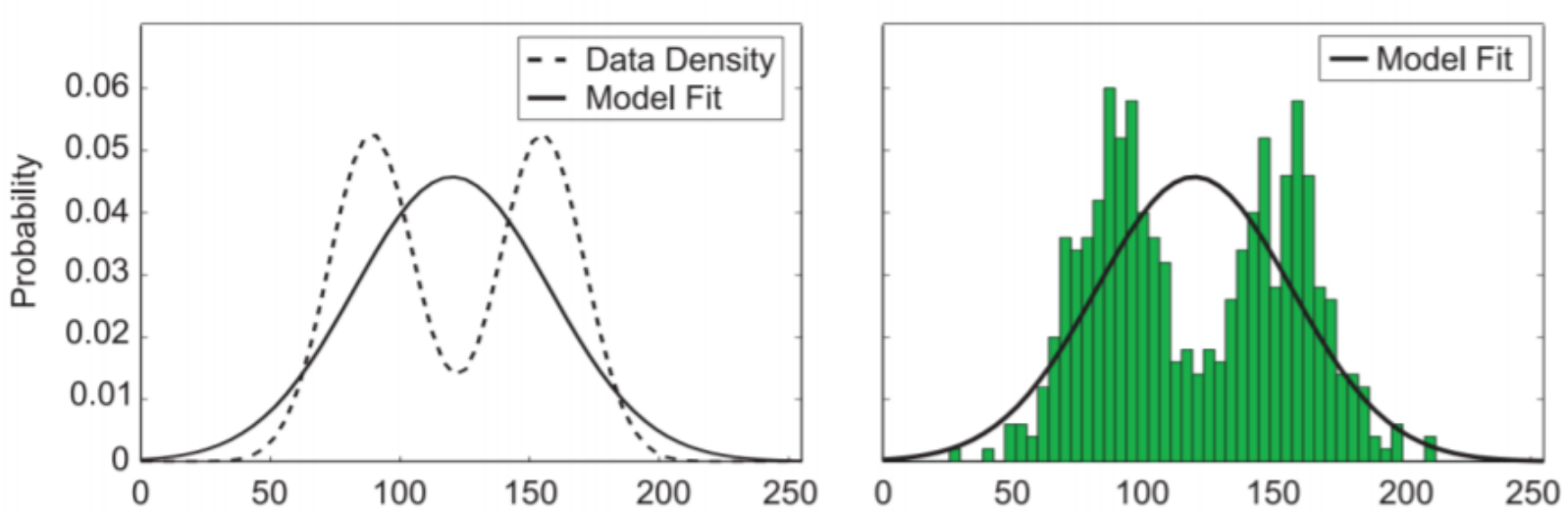
- Instances from X generated by mixture of k Gaussian distributions
- Unknown means of the k Gaussians
- Assuming identity covariance matrix
- Don't know which instance x_i was generated by which Gaussian

Determine Maximum likelihood estimates of:

- means $(\mu_1, \ldots, \mu_k)$

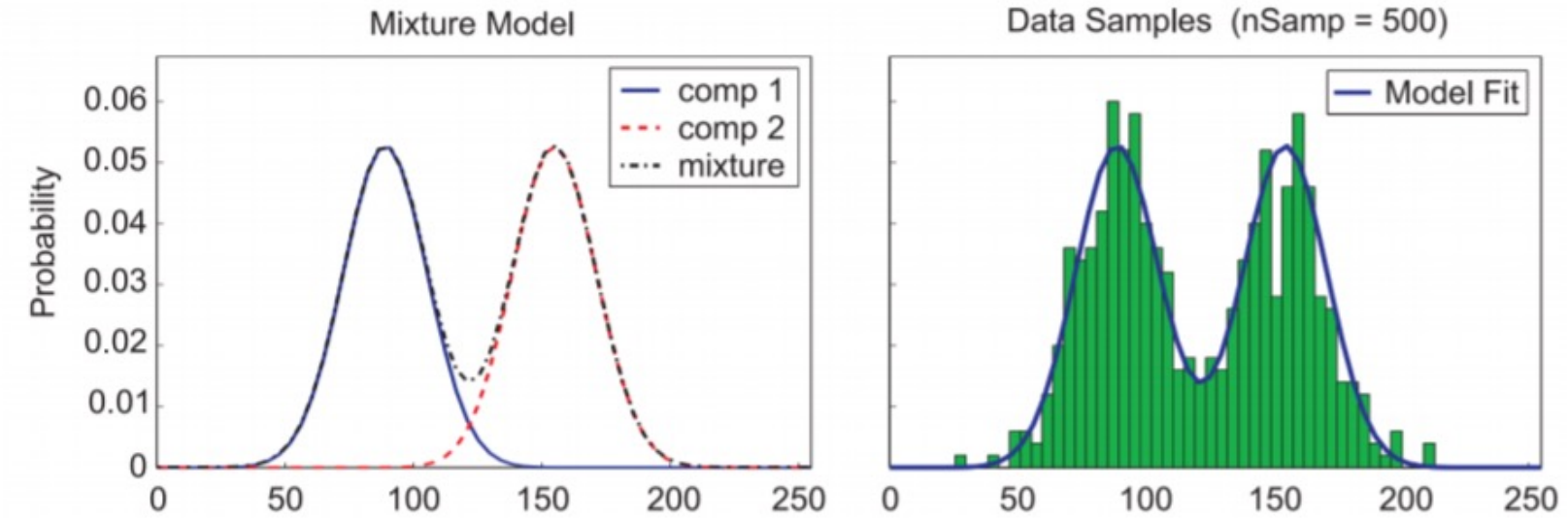
EM for Estimating k-means

- 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

For $k = 2$, think of the full description of each instance as $y_i = \langle x_i, z_{i1}, z_{i2} \rangle$, where

- z_{ij} is 1 if x_i generated by jth Gaussian, otherwise zero in k-means (for a general case, z_{ij} represents the probability of belonging to jth Gaussian)
- 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 $h' = \langle \mu'_1, \mu'_2 \rangle$.

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

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.

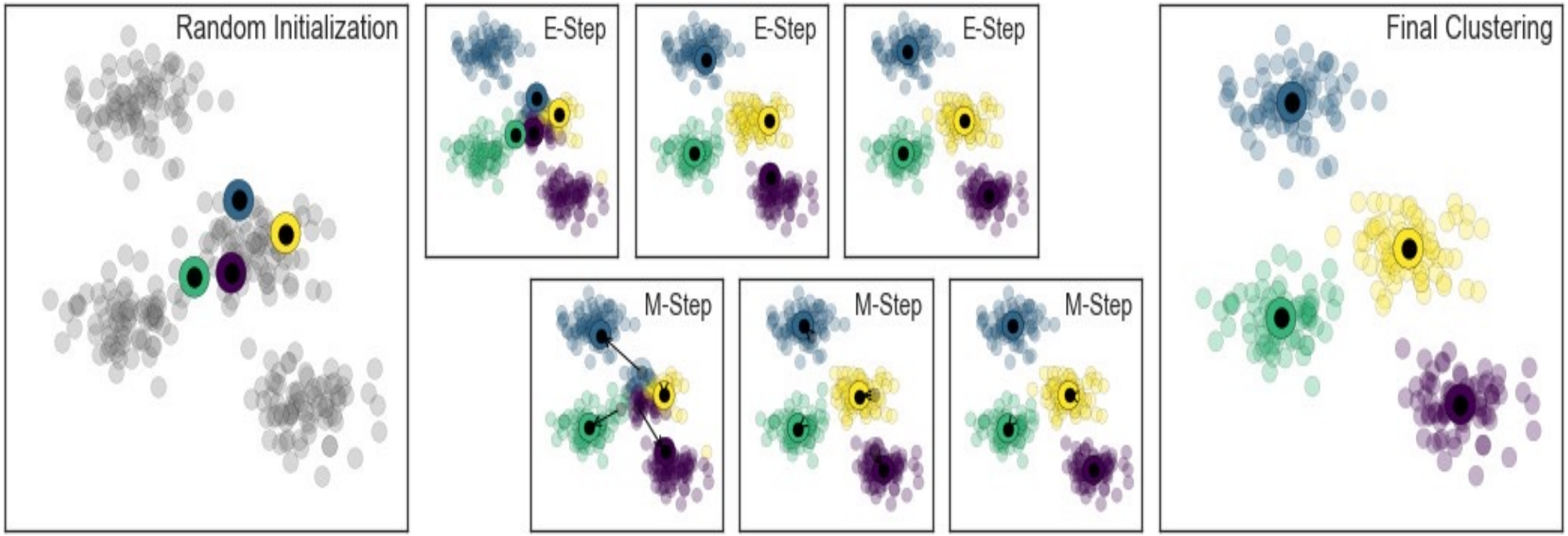
EM Algorithm

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

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

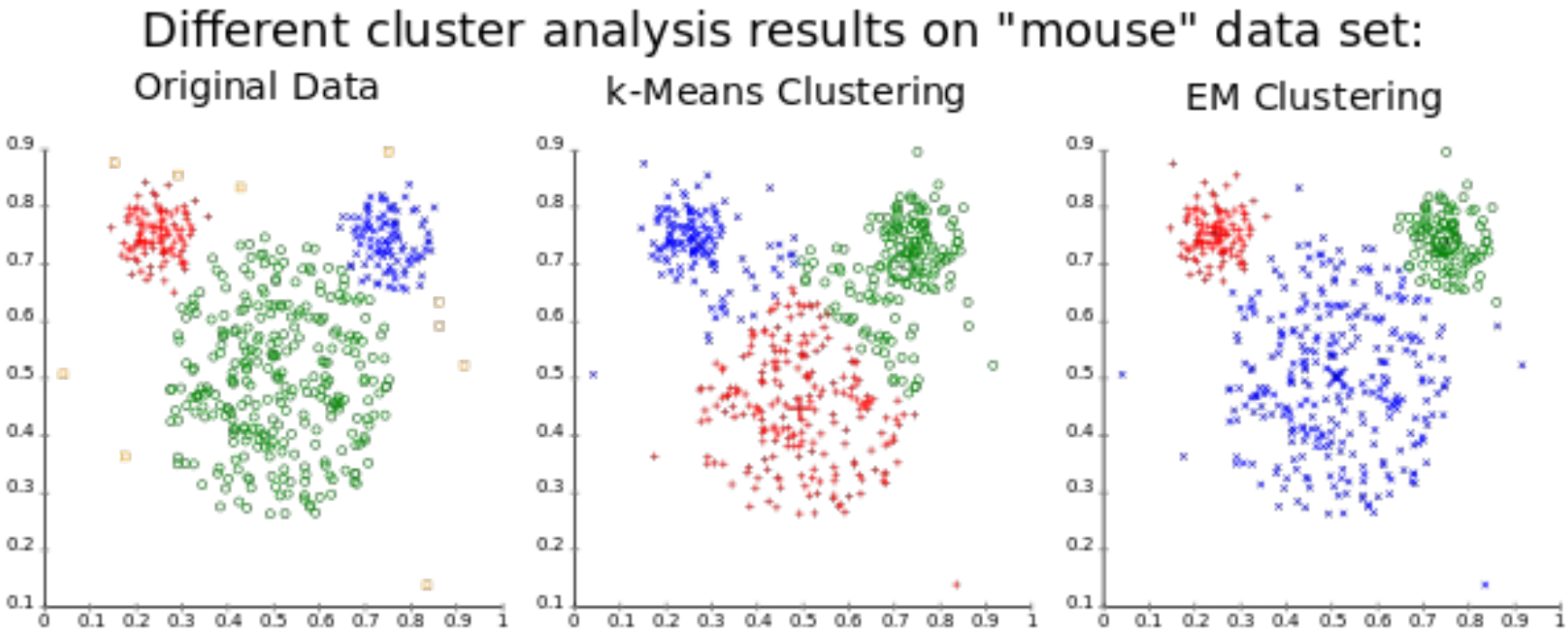
- Y is complete (observable plus unobservable variables) data
- Expected value taken over possible values of unobserved variables in Y

k-means Clustering with 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/multivariate 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 *more* 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:

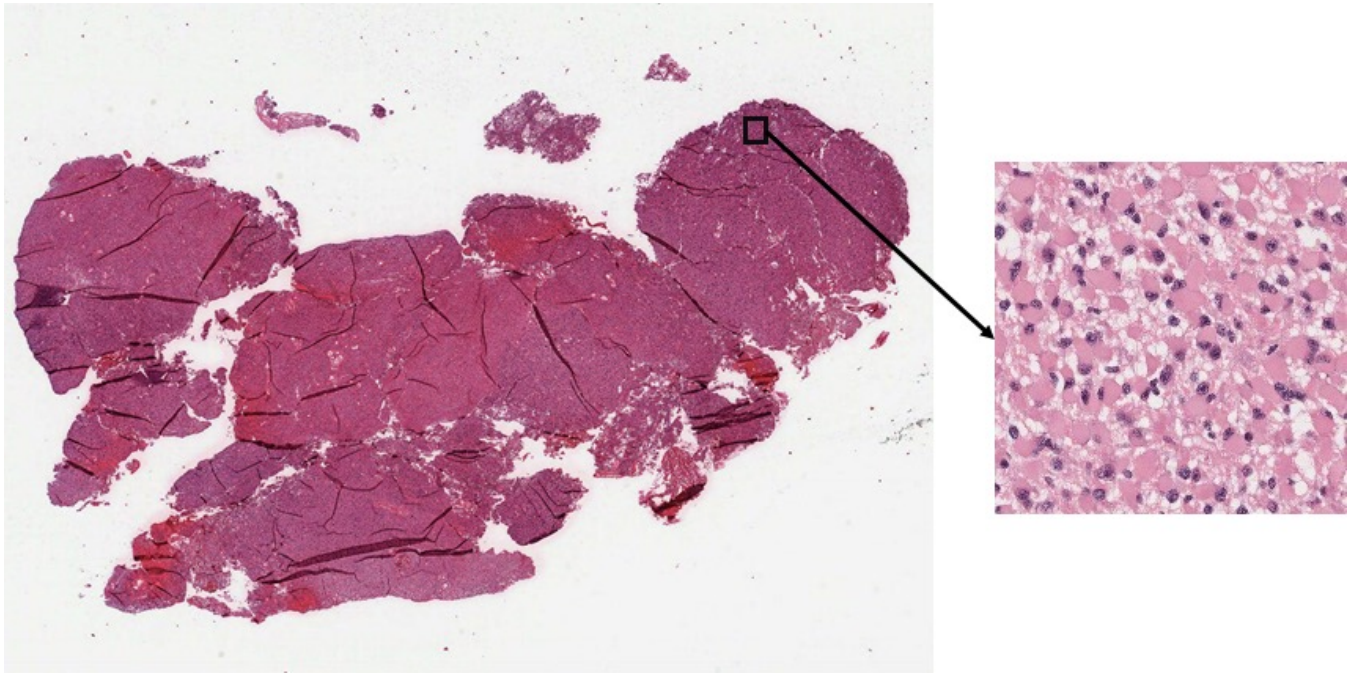
- Observed data $X = \{x_1, \ldots, x_m\}$
- Unobserved data (latent variables) $Z = \{z_1, \ldots, z_m\}$
- Parameterized probability distribution $P(Y|h)$, where
  - $Y = \{y_1, \ldots, y_m\}$ is the full data $y_i = x_i \cup z_i$
  - h are the parameters

Determine:

- h that (locally) maximizes $E[\ln P(Y|h)]$

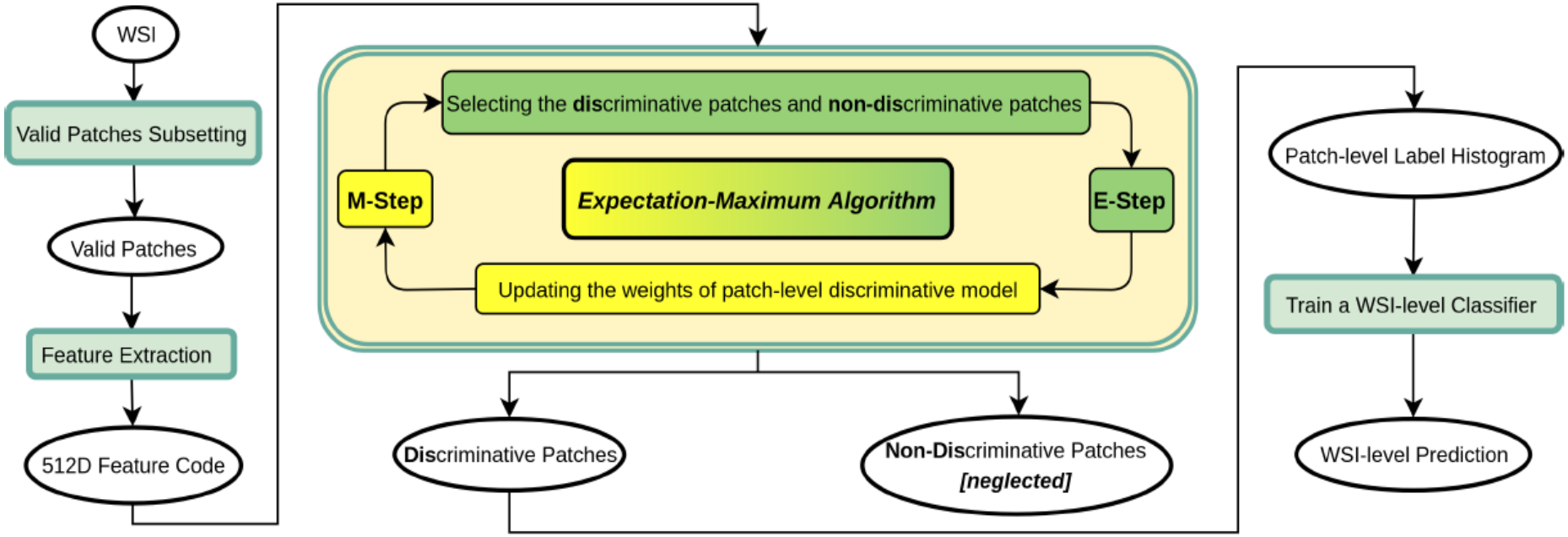
Example: WSI Analysis

Tumour classification in WSI



Example: WSI Analysis

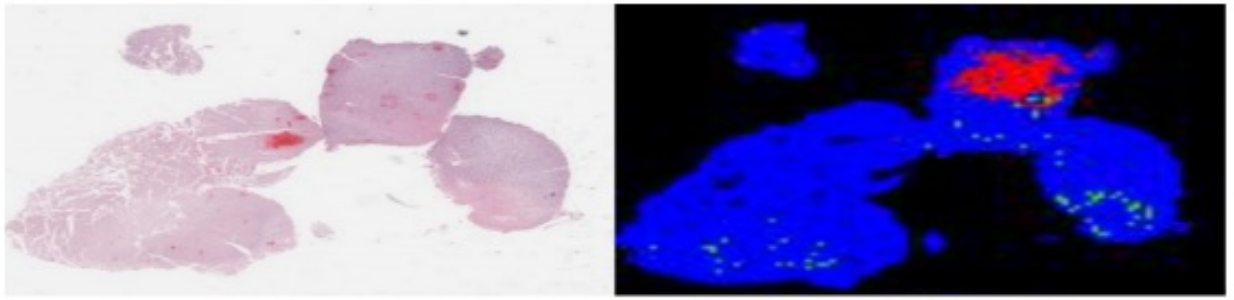
Discriminative patch-based CNN



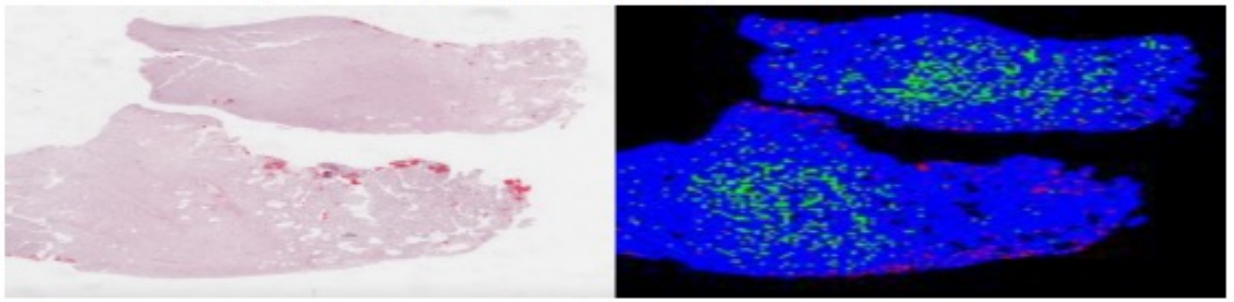
Source: C. Zhang et al. *Whole slide image classification via iterative patch labelling*. ICIP, 2018.

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.

Hierarchical Clustering

Hierarchical Clustering

- Bottom up: at each step join the two closest clusters (starting with single-instance clusters)
  - Design decision: distance between clusters
    E.g., two closest instances in clusters vs. distance between means
- Top down: find two clusters and then proceed recursively for the two subsets
  - Can be very fast
- Both methods produce a dendrogram (tree of “clusters”)

Hierarchical Clustering

Algorithm Hierarchical agglomerative

/* dissimilarity matrix $D(ij)$ is given */

1. Find minimal entry d_{ij} in D and merge clusters i and j
2. Update D by deleting column i and row j, and adding new row and column $i \cup j$
3. Revise entries using
   $d_{k,i\cup j} = d_{i\cup j,k} = \alpha_i d_{ki} + \alpha_j d_{kj} + \gamma |d_{ki} - d_{kj}|$
4. If there is more than one cluster then go to step 1.

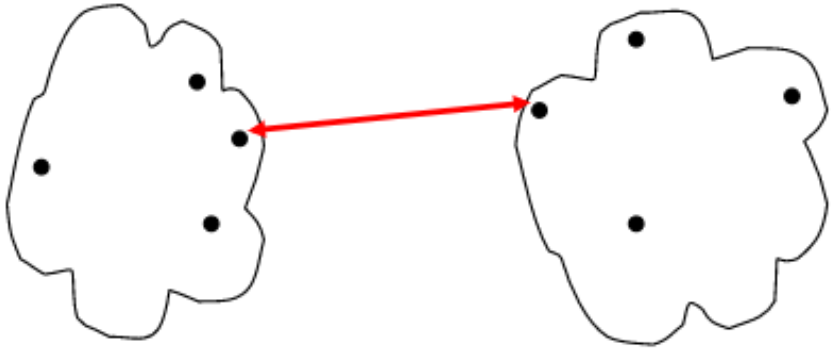
Hierarchical Clustering

The algorithm relies on a general updating formula. With different operations and coefficients, many different versions of the algorithm can be used to give variant clusterings.

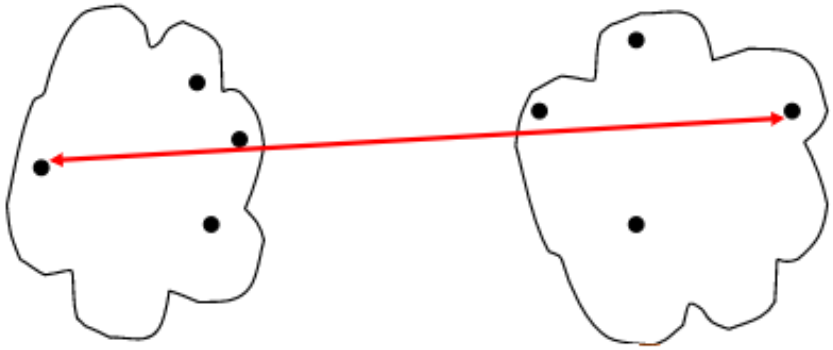
Single linkage $d_{k,i\cup j} = \min(d_{ki}, d_{kj})$ and $\alpha_i = \alpha_j = \frac{1}{2}$ and $\gamma = -\frac{1}{2}$.
Complete linkage $d_{k,i\cup j} = \max(d_{ki}, d_{kj})$ and $\alpha_i = \alpha_j = \frac{1}{2}$ and $\gamma = \frac{1}{2}$.
Average linkage $d_{k,i\cup j} = \frac{n_i d_{ki}}{n_i+n_j} + \frac{n_j d_{kj}}{n_i+n_j}$ and $\alpha_i = \frac{n_i}{n_i+n_j}, \alpha_j = \frac{n_j}{n_i+n_j}$ and $\gamma = 0$.

Note: dissimilarity computed for every pair of points with one point in the first cluster and the other in the second.

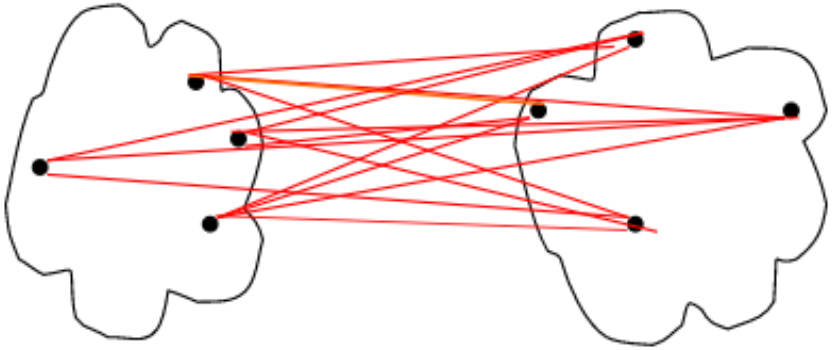
Hierarchical Clustering



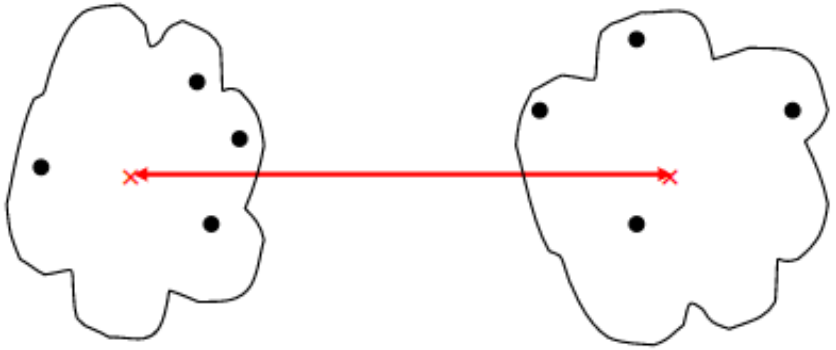
Single link



Complete link



Average link



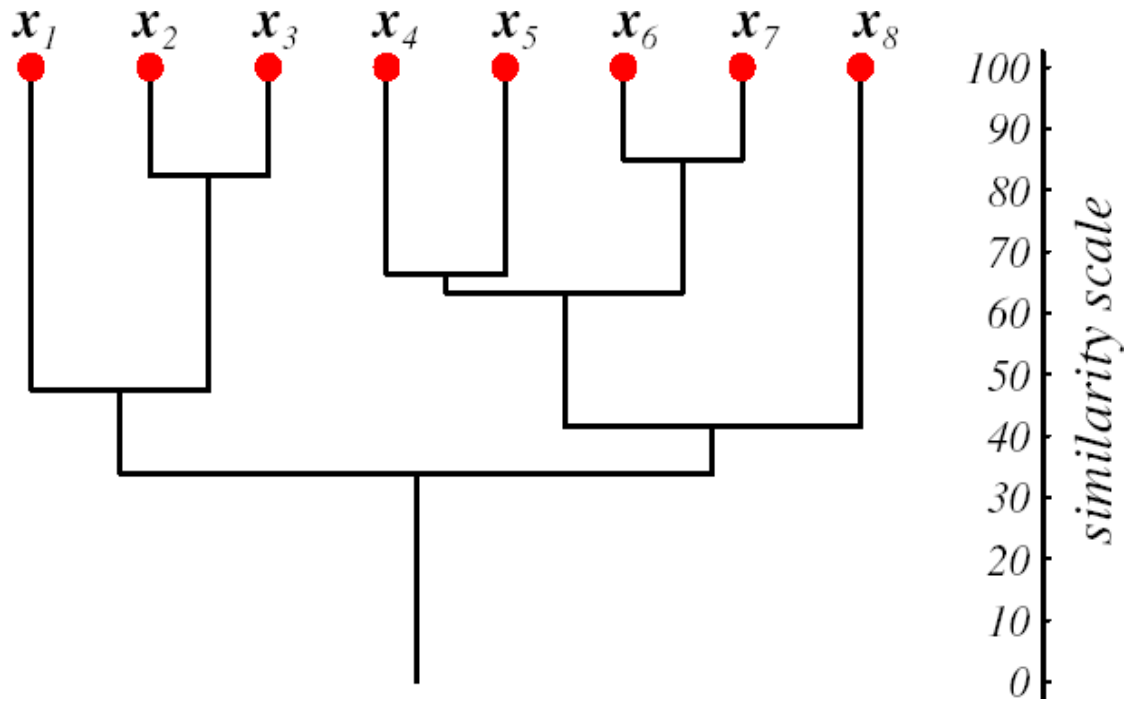
Centroid distance

Hierarchical Clustering

Represent results of hierarchical clustering with a *dendrogram*
See next diagram

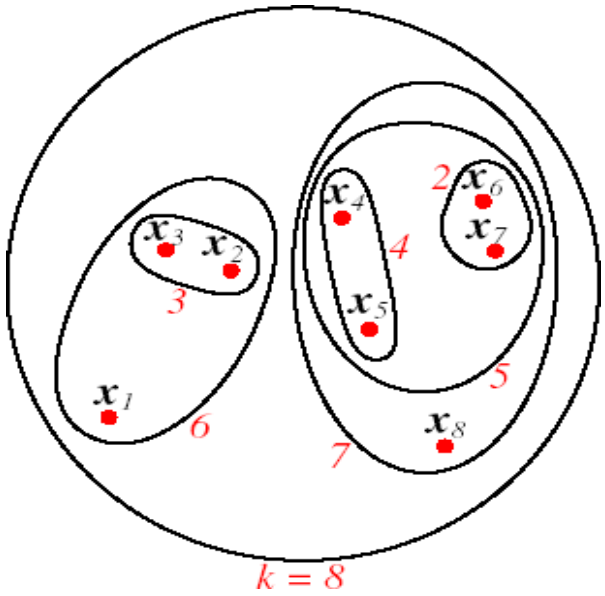
- at level 1 all points in individual clusters
- x_6 and x_7 are most similar and are merged at level 2
- dendrogram drawn to scale to show similarity between grouped clusters

Hierarchical Clustering



Hierarchical Clustering

An alternative representation of hierarchical clustering based on sets shows hierarchy (by set inclusion), but not distance.



Hierarchical Clustering

Limitation to beware of:

- hierarchical clustering imposes a bias - the clustering forms a dendrogram despite the possible lack of a implicit hierarchical structuring in the data

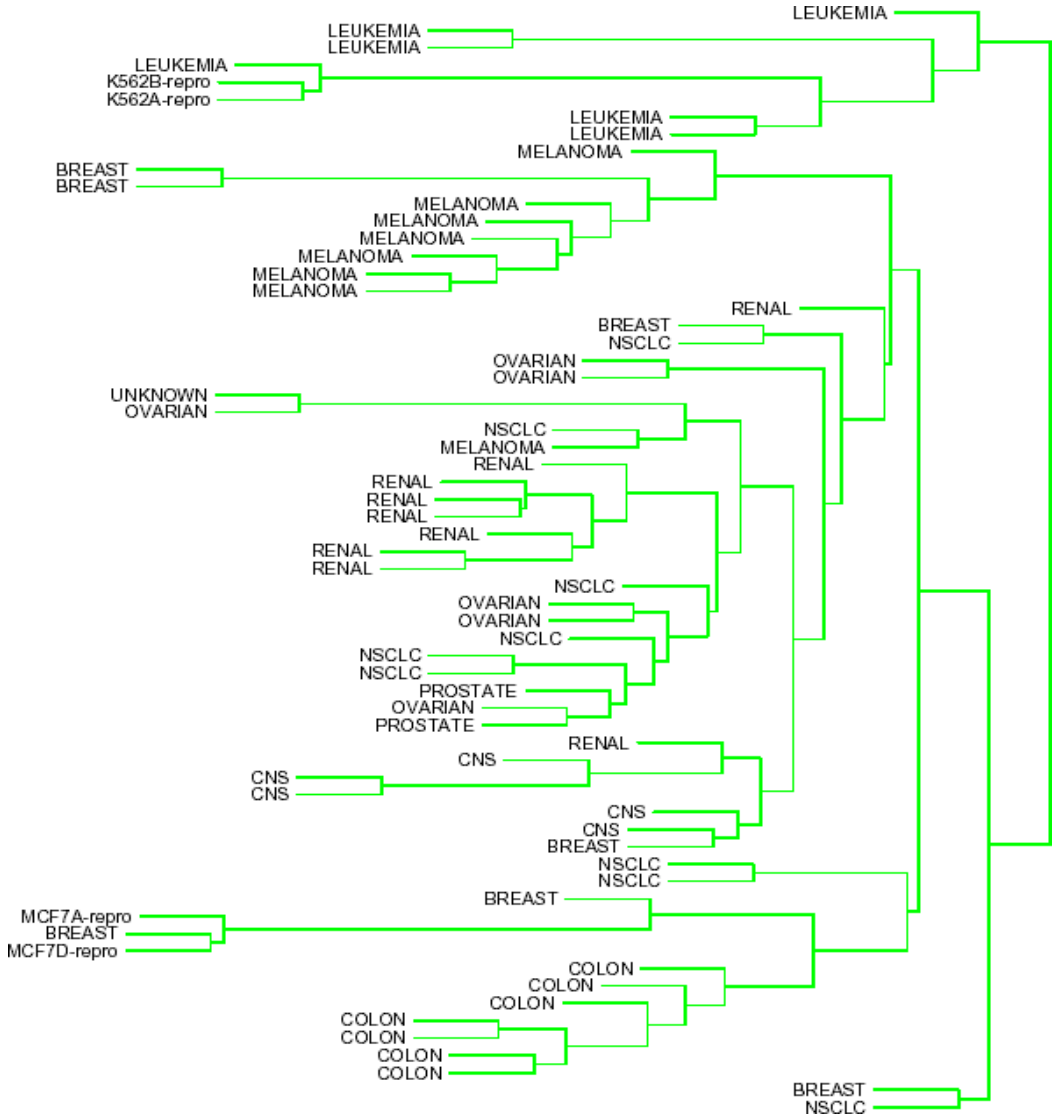
Dendrograms

Next diagram: average-linkage hierarchical clustering of microarray data. For this dataset the class of each instance is shown in each leaf of dendrogram to illustrate how clustering has grouped similar tissue samples coincides with the labelling of samples by cancer subtype.

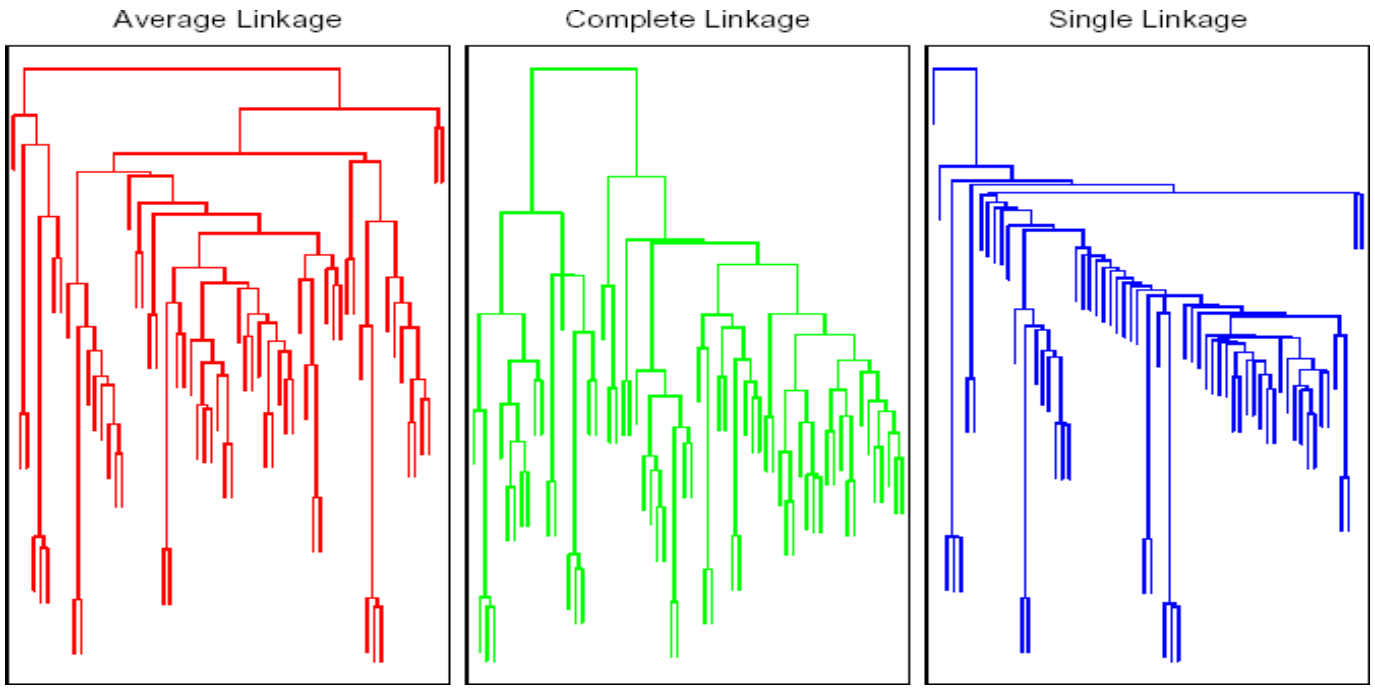
Followed on next slide by diagram showing:

- average-linkage based on average dissimilarity between groups
- complete-linkage based on dissimilarity of furthest pair between groups
- single-linkage based on dissimilarity of closest pair between groups

Dendrograms



Dendrograms

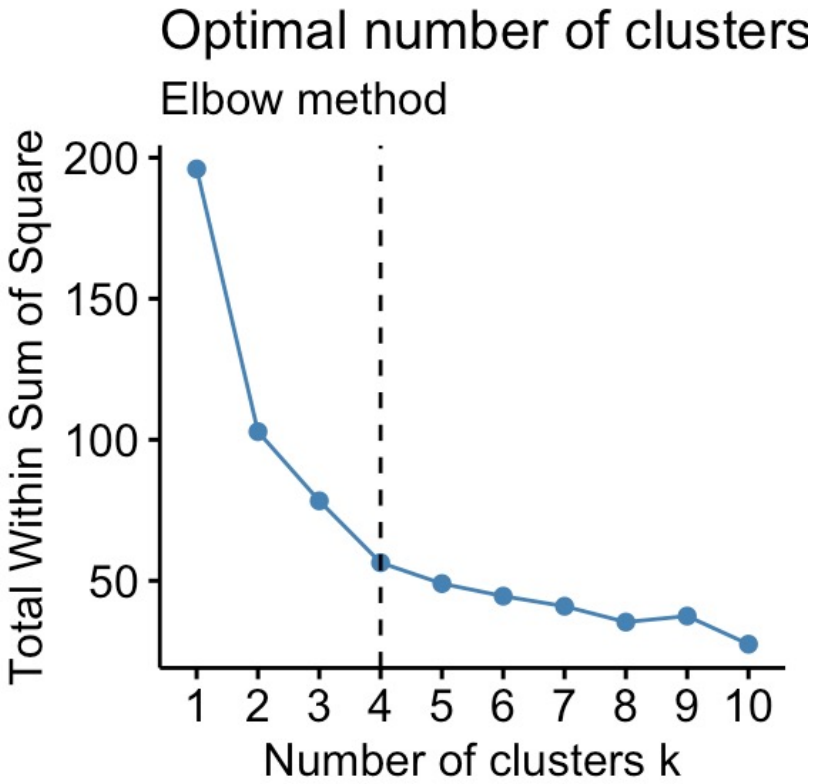


Number of Clusters

Many methods of estimating the "correct" number of clusters have been proposed, based on some clustering criterion:

- Elbow method:
  - measure the within-cluster dispersion (total sum of squared distances from each point to the cluster centroid)
  - compute this for various k choices
  - choose the k that doesn't improve the dispersion much

Number of Clusters



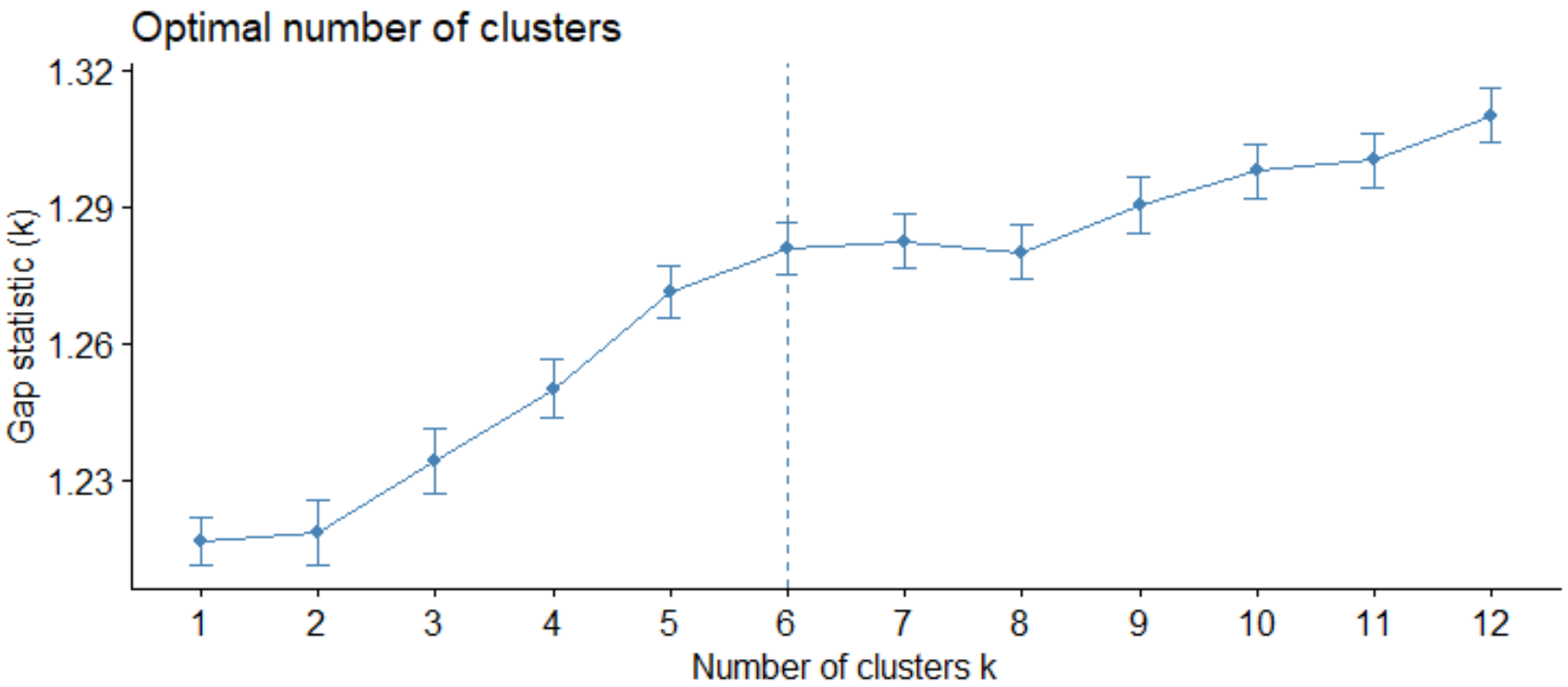
Number of Clusters

Another technique – Gap statistics:

- Cluster the observed data and compute the corresponding total within-cluster variation W_k.
- Generate B reference data sets with a random uniform distribution. Cluster each of these reference data sets and compute the corresponding total within-cluster variation W_{kb}.
- Compute the estimated gap statistic as the deviation of the observed W_k value from its expected value W_{kb} : $Gap(k) = \frac{1}{B}\sum_b log(W_{kb}) - \log(W_k)$. Compute also the standard deviation of the statistics.
- Choose the number of clusters as the smallest value of k such that the gap statistic is within one standard deviation of the gap at $k + 1$:

$$Gap(k) \geq Gap(k + 1) - s_{k+1}.$$

Number of Clusters



Cluster quality – Silhouette plot

Key idea: compare each object's *separation* from other clusters relative to the *homogeneity* of its cluster.

For each object i (e.g x_i), define its silhouette width $s(i) \in [-1, 1]$:

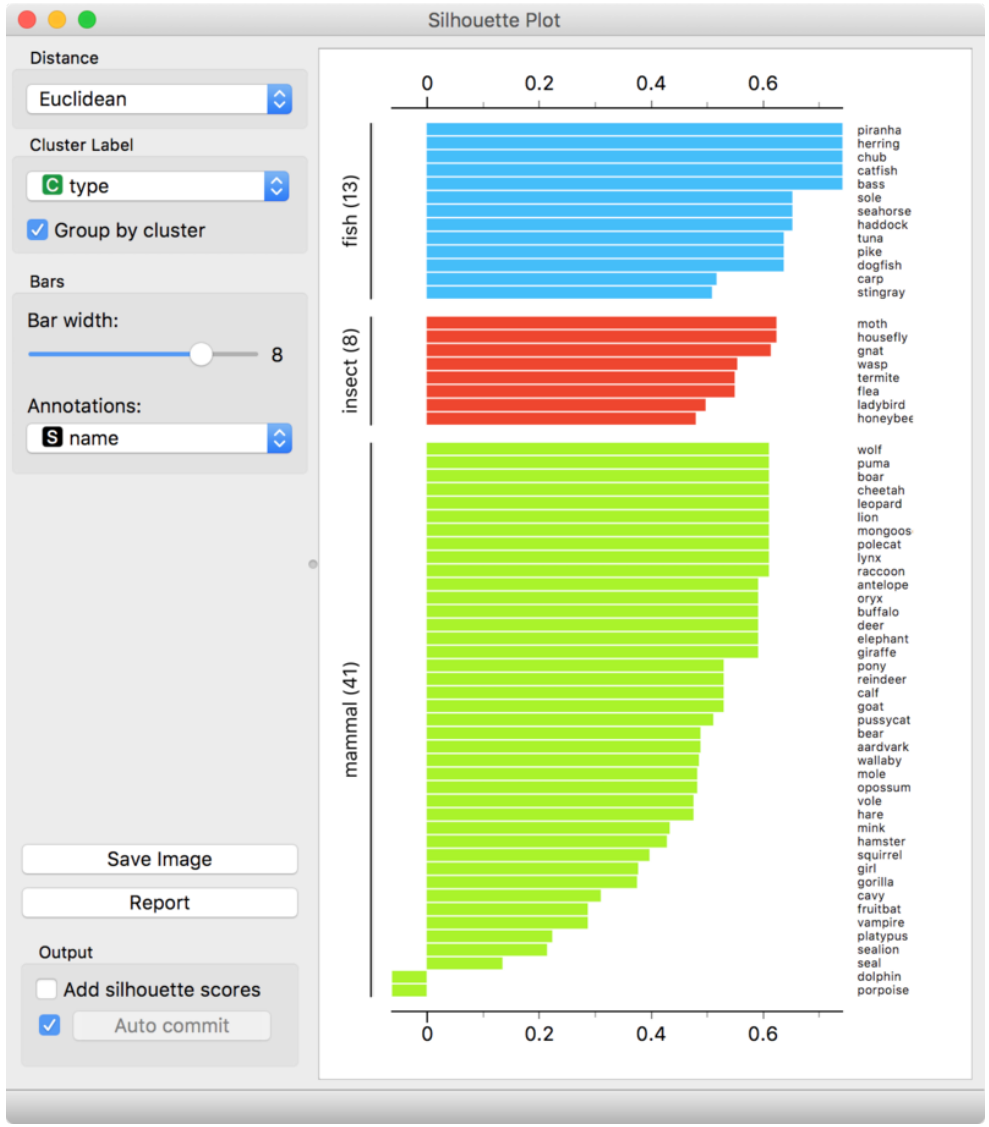
Let $a(i)$ be the average dissimilarity between x_i and elements of P_i, i.e., cluster to which x_i belongs,

Let $d(i, C)$ be the average dissimilarity of i to elements of some *other* cluster C.

Let $b(i) = min_C\, d(i, C)$. The *silhouette width (value)* is

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

Cluster quality – Silhouette plot



$$s(i) = \begin{cases} 1 - a(i)/b(i), & \text{if } a(i) < b(i) \\ 0, & \text{if } a(i) = b(i) \\ b(i)/a(i) - 1, & \text{if } a(i) > b(i) \end{cases}$$

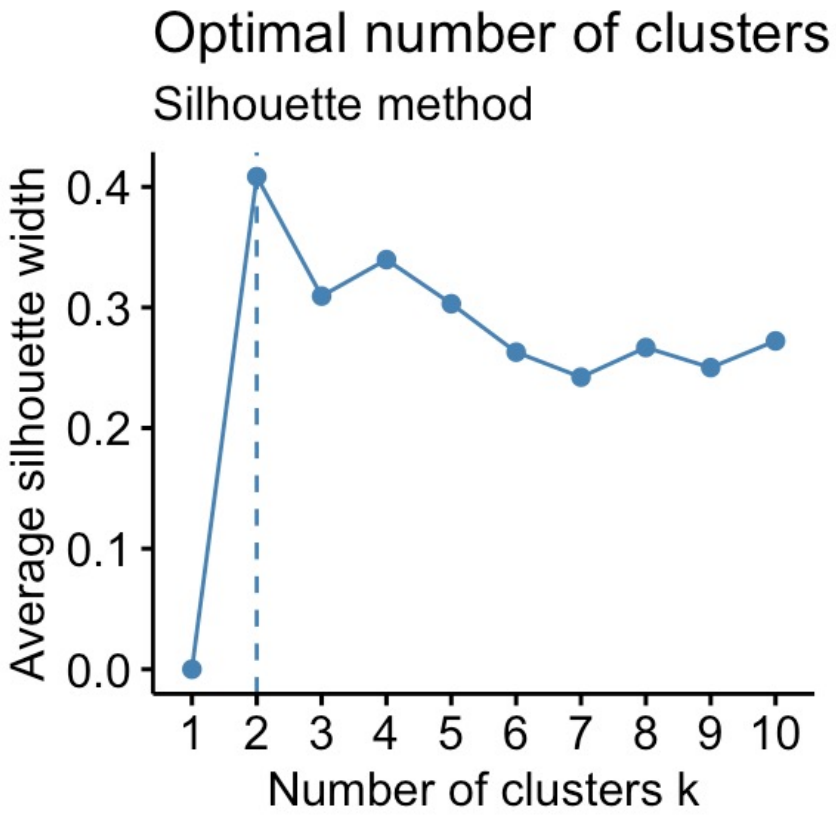
https://en.wikipedia.org/wiki/Silhouette_(clustering)

Cluster quality – Silhouette plot

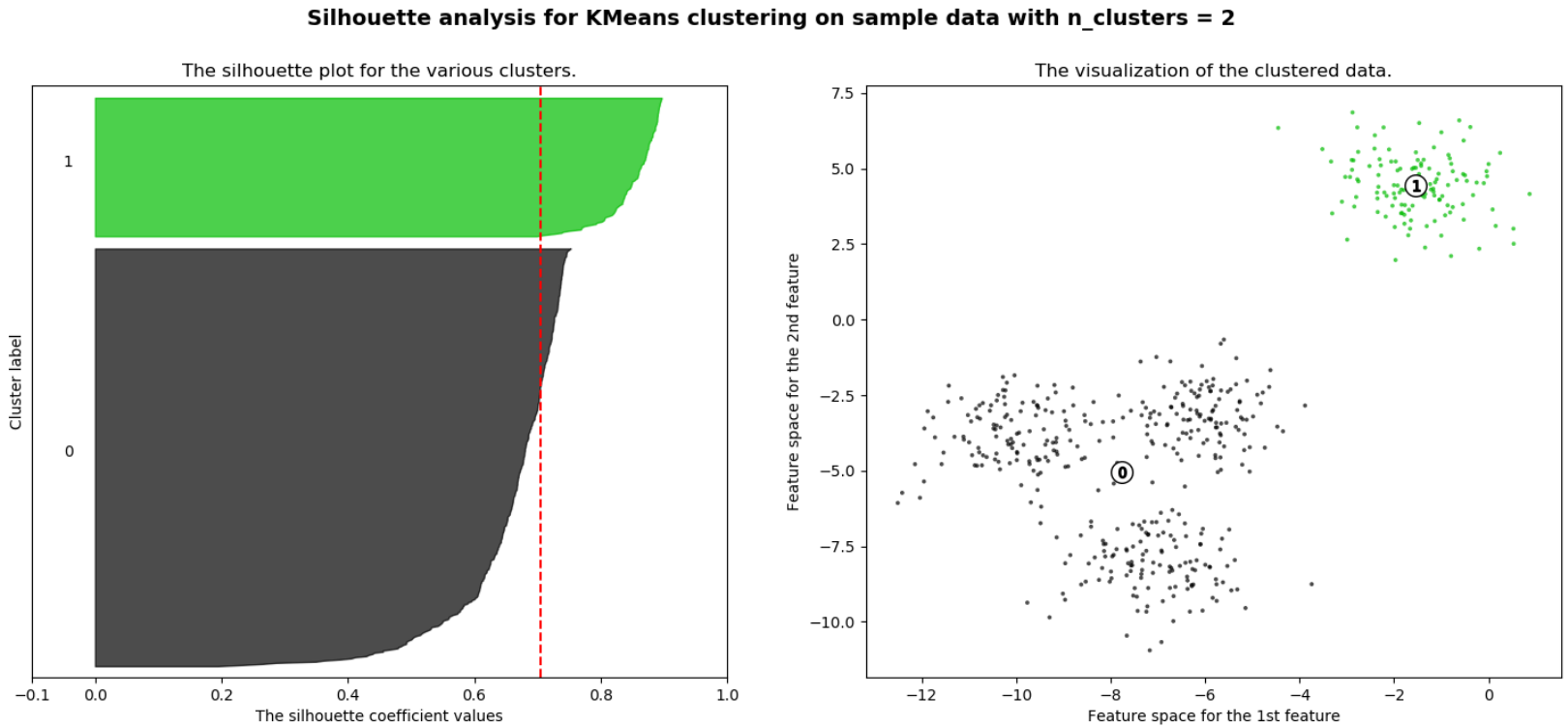
How can we intepret a Silhouette plot ?

- say for some object i we have $b(i) \gg a(i)$ then it will be well-separated from all the other clusters, and its cluster will probably be homogeneous
  - in such cases $s(i)$ will tend to be close to 1 for object i, and we can take it to be "well-classified" by its cluster
- conversely, if $s(i)$ is close to -1 we can view it as "misclassified" by its cluster
- can see which clusters are "good" or otherwise, and estimate number of clusters
- can take average $s(i)$ over different clusterings for comparison

Cluster quality – Silhouette plot

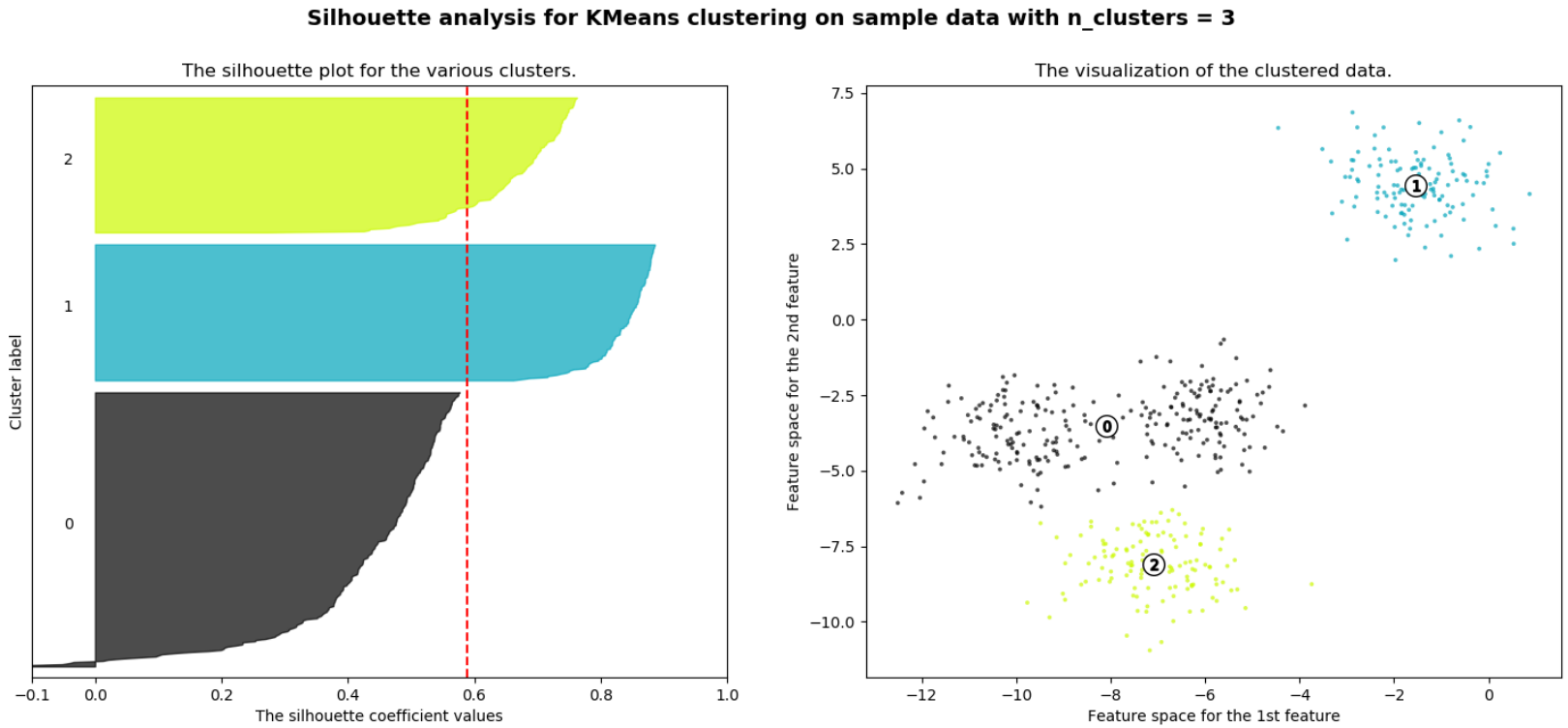


Cluster quality – Silhouette plot



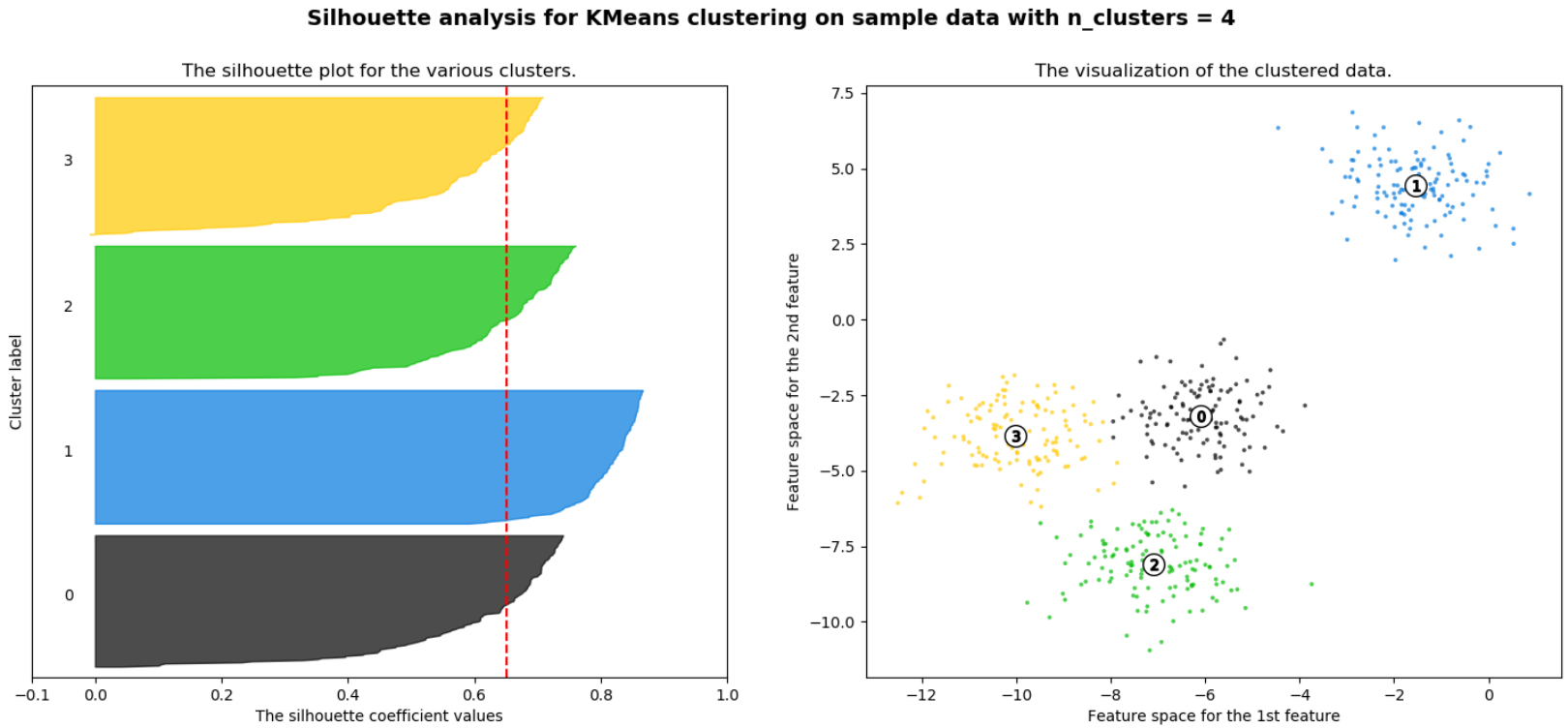
For n_clusters = 2 The average silhouette_score is : 0.7049787496083262

Cluster quality – Silhouette plot



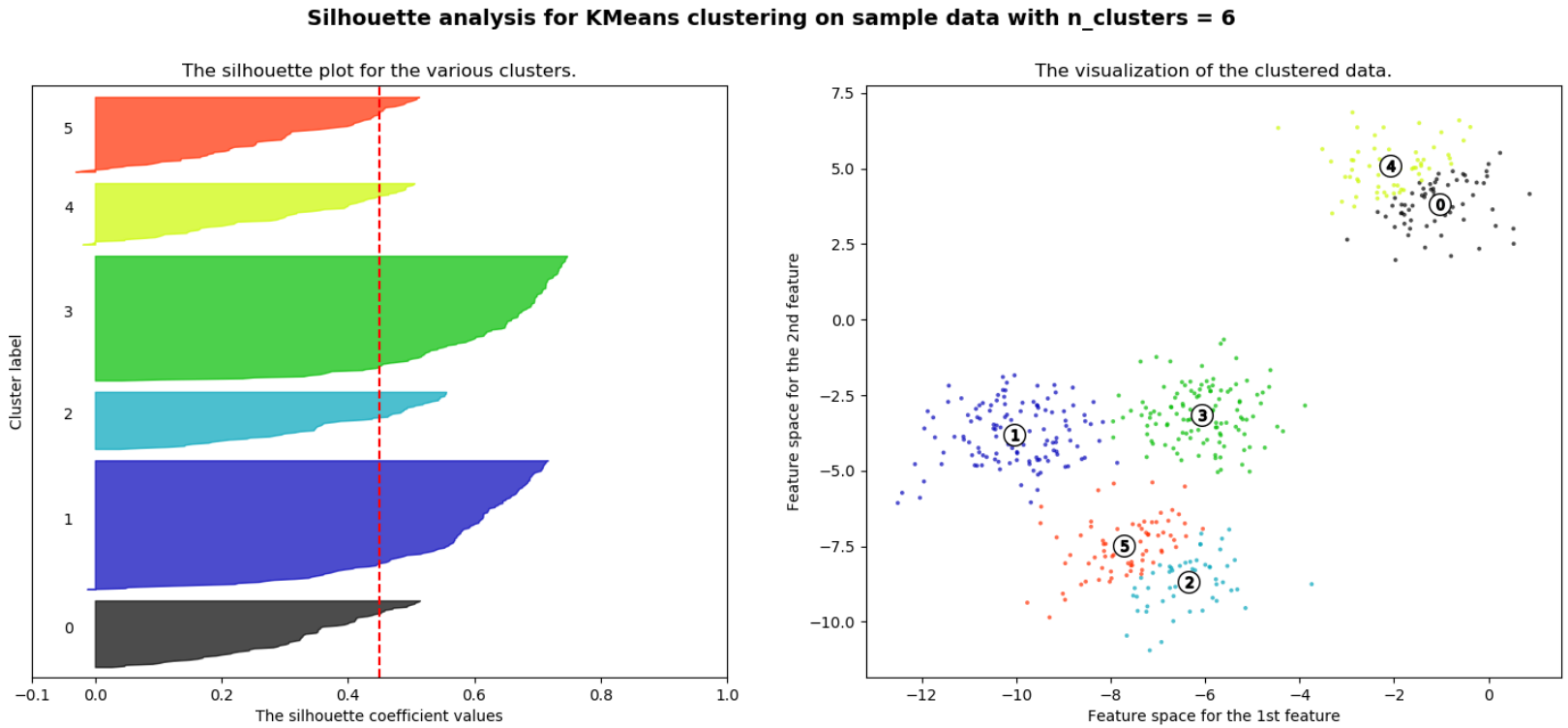
For n_clusters = 3 The average silhouette_score is : 0.5882004012129721

Cluster quality – Silhouette plot



For n_clusters = 4 The average silhouette_score is : 0.6505186632729437

Cluster quality – Silhouette plot



For n_clusters = 6 The average silhouette_score is : 0.4504666294372765

Cluster quality – external evaluation

Use another set of data with known labels for evaluation

- Perform clustering
- Evaluate classification performance from the clustering outputs
  - Rand index
  - F-measure
  - Jaccard index

Clustering summary

- Many techniques available – may not be single “magic bullet” rather different techniques useful for different aspects of data
- Hierarchical clustering gives a view of the complete structure found, without restricting the number of clusters, but can be computationally expensive
- Different linkage methods can produce very different dendrograms
- Problem may not have a “real” hierarchical structure

Clustering summary

- k-means avoids some of these problems, but also has drawbacks
- The value for k need to be set beforehand
- Cannot extract “intermediate features” e.g., a subset of features in which a subset of objects is co-expressed
- For all of these methods, can cluster objects or features, but not both together (coupled two-way clustering)
- Should all the points be clustered ? modify algorithms to allow points to be discarded
- Visualization is important: dendrograms and alternatives like Self-Organizing Maps (SOMs) are good but further improvements would help

Clustering summary

- How can the quality of clustering be estimated ?
  - if clusters known, measure proportion of disagreements to agreements
  - if unknown, measure homogeneity (e.g., average similarity between feature vectors in a cluster and the centroid) and separation (e.g., weighted average distances between cluster centroids) with aim of increasing homogeneity and separation
  - silhouette method, etc.
- Clustering is only the first step - mainly exploratory; classification, modelling, hypothesis formation, etc.

Dimensionality Reduction

Dimensionality Reduction

What is this and why would we need it ?

- each numeric feature in a dataset is a dimension
- in general, no restrictions on the number of dimensions
- however, many features could be related
- do we need them all in our dataset ?
  - including them all is unlikely to improve models
  - curse of dimensionality
  - feature selection may return arbitrary features
- so, what to do ?
- one solution would be to find a set of *new* features
  - should be fewer than the original set
  - should preserve information in original set (as much as possible)

Principal Component Analysis (PCA)

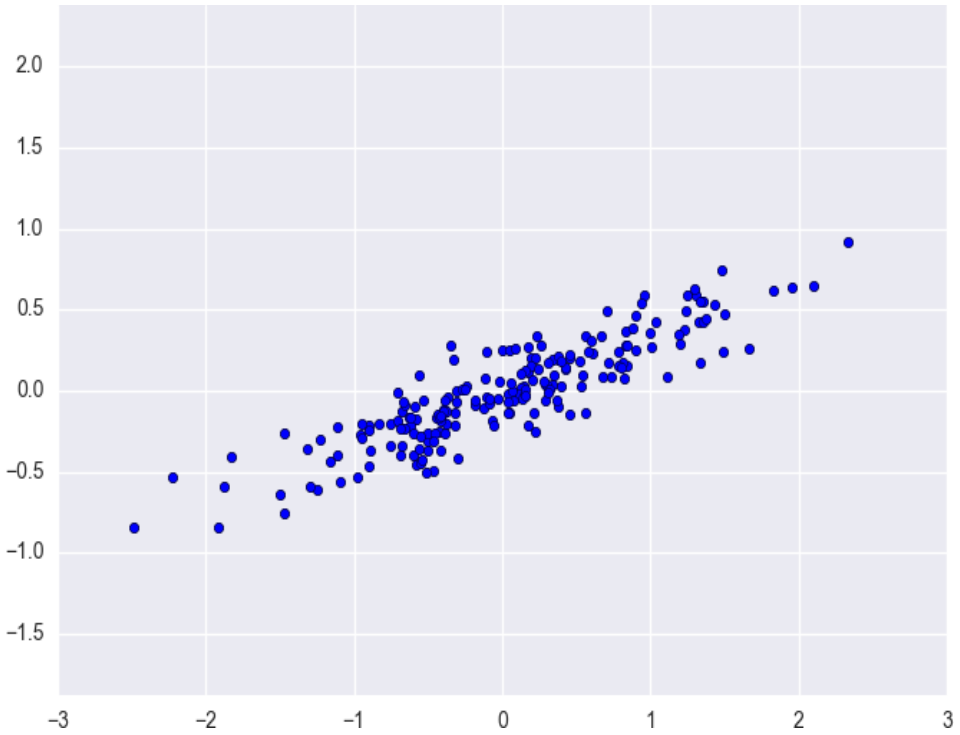
Key idea: look for features in a transformed space so that each dimension in the new space captures the most variation in the original data when it is projected onto that dimension.

Any new features should be highly correlated with (some of) the original features, but not with any of the other new features.

This suggests an approach: consider using the variance-covariance matrix we recall from correlation and regression

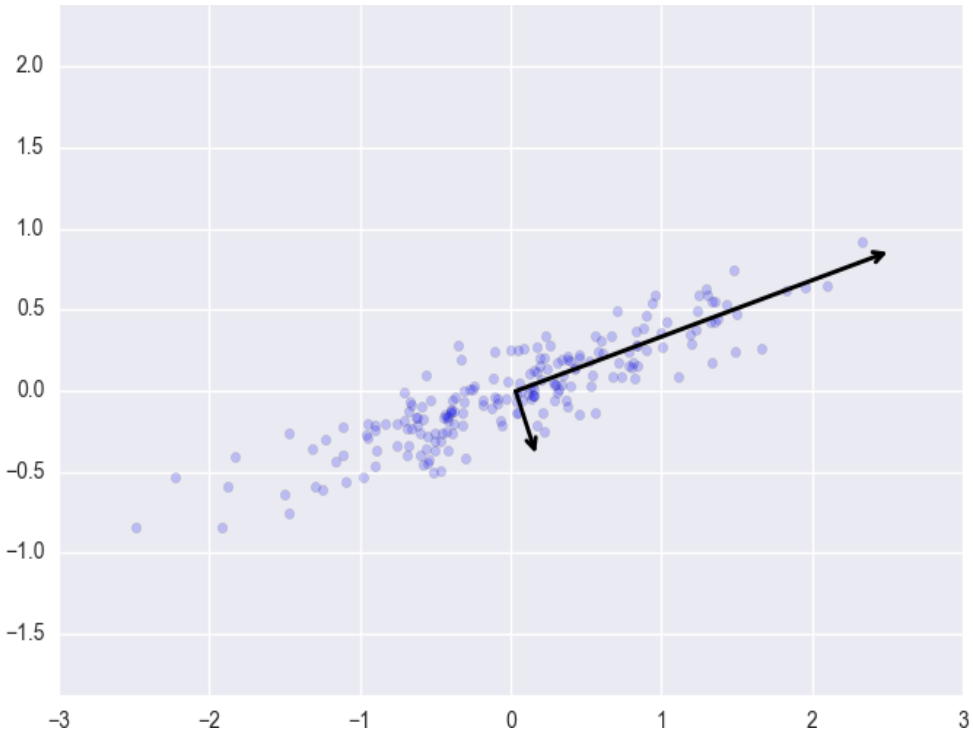
PCA Example

PCA looks for linear combinations of the original features. This dataset of 200 points seems to show such a relationship between two feature dimensions.



PCA Example

PCA finds two new features on which the original data can be projected, rotated and scaled. These explain respectively 0.75 and 0.02 of the variance.

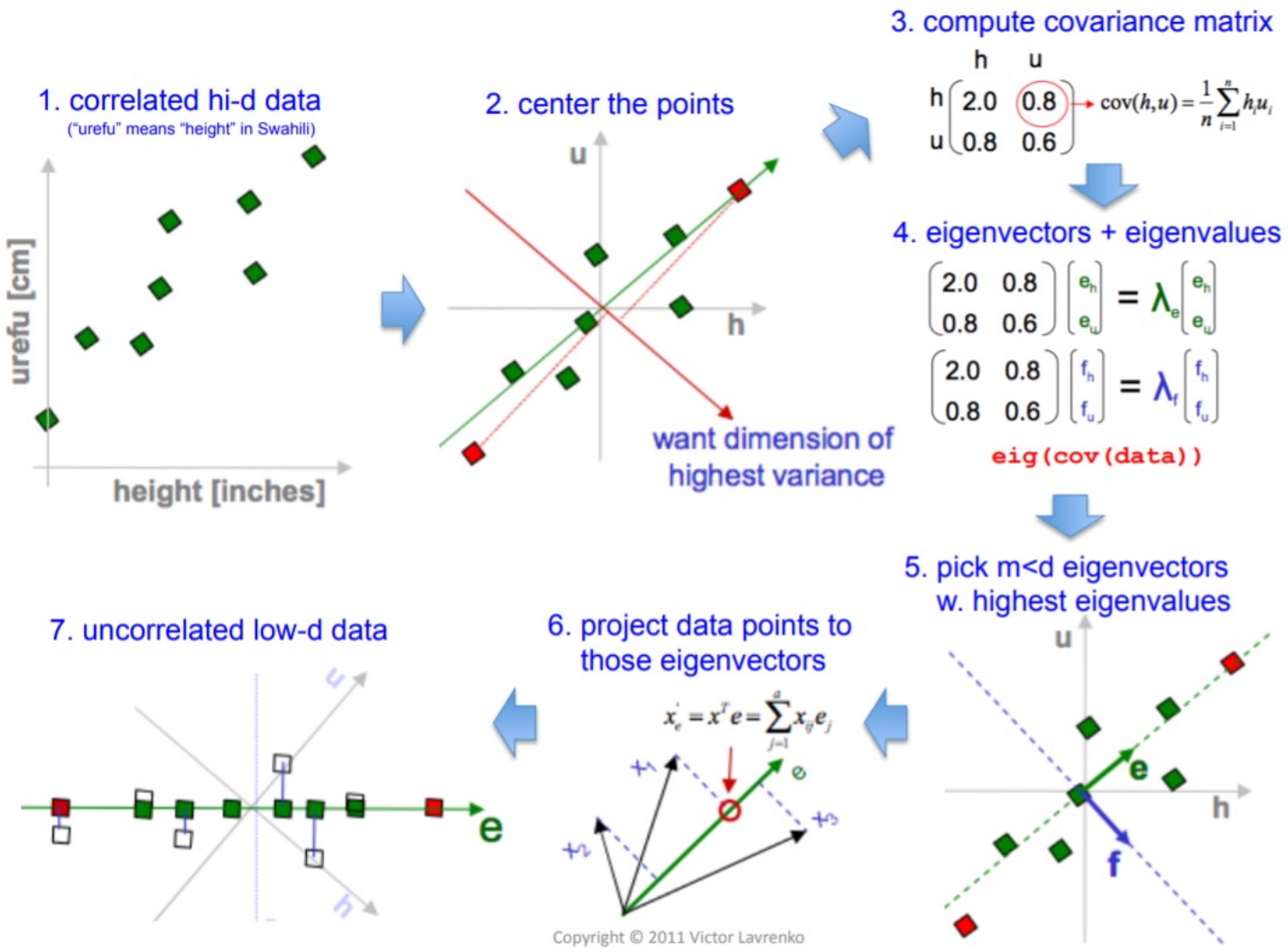


PCA Algorithm

This algorithm can be presented in several ways. Here are the basic steps in terms of the variance reduction idea:

1) take the data as an $m \times n$ matrix $\boldsymbol{X}$
2) "centre" the data by subtracting the mean of each column
3) construct covariance matrix $\boldsymbol{C}$ from centred matrix
4) compute eigenvector matrix V (rotation) and eigenvalue matrix S (scaling) such that $V^{-1}C\,V = S$, and S is a diagonal $n \times n$ matrix
5) sort columns of S in decreasing order (decreasing variance) and their corresponding eigenvectors
6) remove columns of S and V that the eigenvalues are below some minimum threshold
7) Transform the original data using the remaining eigenvectors

PCA algorithm



PCA Example

By rejecting the second component we reduce the dimensionality by 50% while preserving much of the original variance, seen by plotting the inverse transform of this component along with the original data.

PCA example: Eigenfaces

- Application – face recognition
- Assume that most face images lie on a low-dimensional subspace determined by the first k $(k << d)$ directions of maximum variance
- Use PCA to determine the vectors or “eigenfaces” that span that subspace
- Represent all face images in the dataset as linear combinations of eigenfaces

PCA example: Eigenfaces

Training

1. Align training images $x_1, x_2, \ldots, x_N$

Note that each image is formulated into a long vector!

2. Compute average face $\mu = \frac{1}{N}\sum x_i$

3. Compute the difference image (the centered data matrix)

$$X_c = \begin{bmatrix} | & & | \\ x_1 & \cdots & x_n \\ | & & | \end{bmatrix} - \begin{bmatrix} | & & | \\ \mu & \cdots & \mu \\ | & & | \end{bmatrix}$$

$$= X - \mu 1^T = X - \frac{1}{n} X 1 1^T = X\left(I - \frac{1}{n} 1 1^T\right)$$

PCA example: Eigenfaces

4. Compute the covariance matrix

$$\Sigma = \frac{1}{n}\begin{bmatrix} | & & | \\ x_1^c & \dots & x_n^c \\ | & & | \end{bmatrix}\begin{bmatrix} - & x_1^c & - \\ & \vdots & \\ - & x_n^c & - \end{bmatrix} = \frac{1}{n} X_c X_c^T$$

5. Compute the eigenvectors of the covariance matrix Σ
6. Compute each training image x_i 's projections as

$$x_i \rightarrow \left(x_i^c \cdot \phi_1, x_i^c \cdot \phi_2, \ldots, x_i^c \cdot \phi_K\right) \equiv \left(a_1, a_2, \ldots, a_K\right)$$

7. Visualize the estimated training face x_i

$$x_i \approx \mu + a_1\phi_1 + a_2\phi_2 + \ldots + a_K\phi_K$$

PCA example: Eigenfaces

6. Compute each training image x_i 's projections as

$$x_i \rightarrow \left(x_i^c \cdot \phi_1, x_i^c \cdot \phi_2, \ldots, x_i^c \cdot \phi_K\right) \equiv \left(a_1, a_2, \ldots, a_K\right)$$

7. Visualize the estimated training face x_i

$$x_i \approx \mu + a_1\phi_1 + a_2\phi_2 + \ldots + a_K\phi_K$$

PCA example: Eigenfaces

Testing

1. Take query image t
2. Project y into eigenface space and compute projection

$$t \rightarrow \left((t-\mu)\cdot\phi_1, (t-\mu)\cdot\phi_2, \ldots, (t-\mu)\cdot\phi_K\right) \equiv \left(w_1, w_2, \ldots, w_K\right)$$

3. Compare projection w with all N training projections
   - Simple comparison metric: Euclidean
   - Simple decision: K-Nearest Neighbor

   (note: this "K" refers to the k-NN algorithm, is different from the previous K's referring to the # of principal components)

PCA example: Eigenfaces

K = 4

K = 200

K = 400

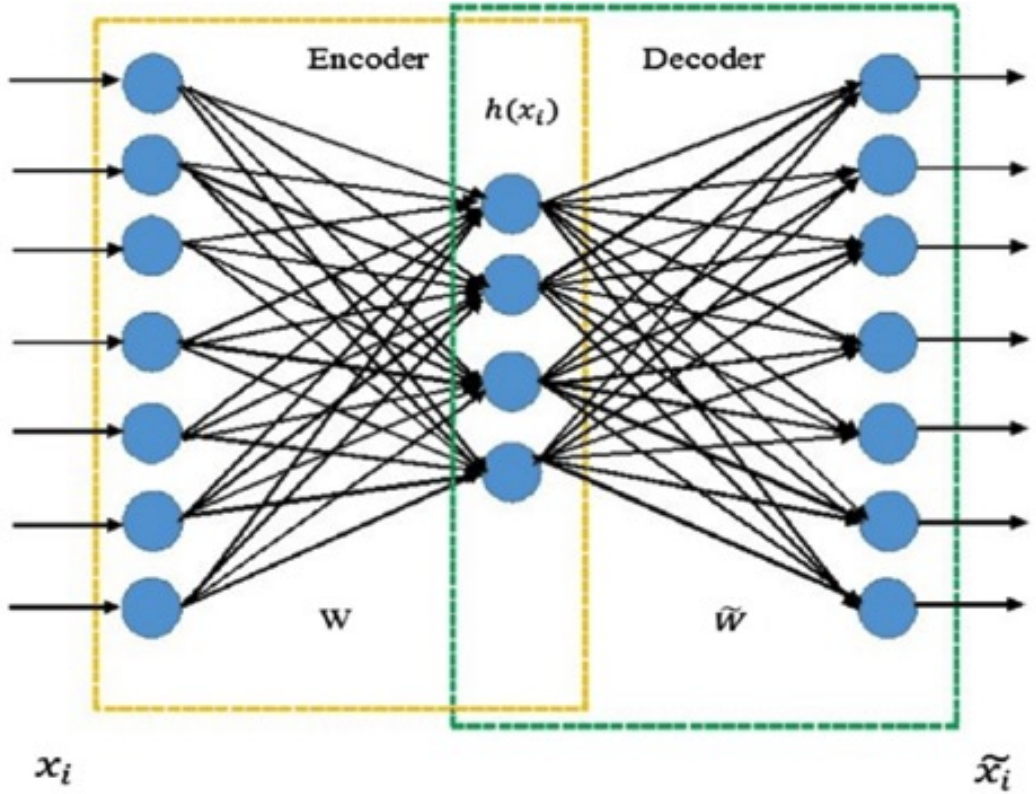
- Only selecting the top K eigenfaces → reduces the dimensionality.
- Fewer eigenfaces result in more information loss, and hence less discrimination between faces.

PCA and friends

- PCA complexity is cubic in number of original features
- this is not feasible for high-dimensional datasets
- alternatively, approximate the sort of projection found by PCA
- for example, can use Random Projections
- more scalable, but what about quality of components ?
- can be shown to preserve distance relations from the original data
- many other methods use essentially the same matrix decomposition idea, such as finding “topics” in text using Latent Semantic, finding hidden “sub-groups” for recommender systems, and so on

Autoencoders

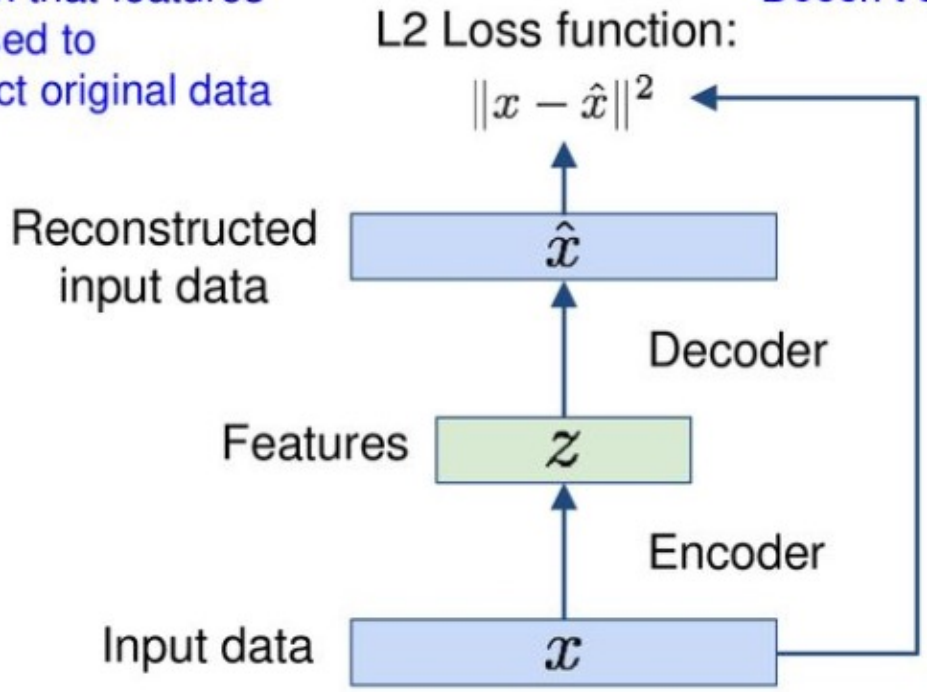
A neural network model – the encoder transforms the data in a way the decoder can then interpret and reconstruct with minimum error.

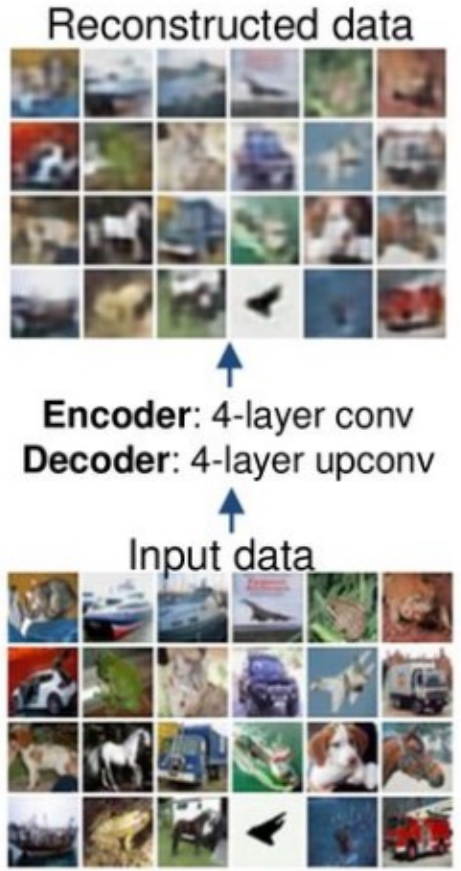


Autoencoders

Train such that features can be used to reconstruct original data

Doesn't use labels!





Dimensionality reduction summary

- PCA will transform original features to new space
- Every new feature is a linear combination of original features
- Aim for new dimensions to maximise variance
- Order by decreasing variance and remove those below a threshold
- Algorithm applies matrix operations to translate, rotate and scale
- Based on covariance matrix
  - can be “kernelised” (KernelPCA)
  - new feature space with non-linear axes
- Many alternatives, e.g., Random Projections, Independent Component Analysis, Multi-dimensional Scaling, Word2Vec,etc.
- Autoencoders and variants – sparse autoencoders, variational autoencoders, etc

Semi-supervised Learning

- Learn initial classifier using labelled set
- Apply classifier to unlabelled set
- The most confident predictions of each classifier on the unlabelled data are retained
- Learn new classifier from now-labelled data
- Repeat until convergence

- Useful when there are not sufficient training data with ground truth labels
- Can be generally applied with any supervised learning techniques

Self-training algorithm

Given: labelled data (x, y) and unlabelled data (x)

Repeat:

Train classifier h from labelled data using supervised learning

Label unlabelled data using classifier h

Assumes: classifications by h will tend to be correct (especially high probability ones)

Co-training

Blum & Mitchell (1998)
Key idea: two views of an instance, f_1 and f_2

- assume f_1 and f_2 independent and compatible
- if we have a good attribute set, leverage similarity between attribute values in each view, assuming they predict the class, to classify the unlabelled data

Co-training

Multi-view learning

- Given two (or more) perspectives on data, e.g., different attribute sets
- Train separate models for each perspective on small set of labelled data
- Use models to label a subset of the unlabelled data
- Repeat until no more unlabelled examples

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
  - cannot extract “intermediate features” e.g., a subset of features in which a subset of objects is co-expressed
  - The EM algorithm is much more robust than K-Means
- hierarchical clustering gives a view of the complete structure found without restricting the no. of clusters but can be computationally expensive
  - different linkage methods can produce very different dendrograms
  - higher nodes can be very heterogeneous
  - problem may not have a “real” hierarchical structure
- clustering is only the first step - mainly exploratory; classification, modelling, hypothesis formation, etc.
- many techniques available – may not be a single “magic bullet” but rather different techniques useful for different aspects of data

Summary

Unsupervised and supervised learning are at different ends of a continuum of "degrees of supervision".

Between these extremes many other approaches are possible:

- Semi-supervised learning, e.g.,
  - train with small labelled sample then improve with large unlabelled sample
  - train with large unlabelled sample then learn classes with small labelled sample
- Active learning
  - learning system acts to generate its own examples

Note: unsupervised learing an increasingly active research area, particularly in neural nets, e.g., Yann LeCun: "How Could Machines Learn as Efficiently as Animals and Humans?"
http://www.youtube.com/watch?v=uYwH4TSdVYs

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://www.inf.ed.ac.uk/teaching/courses/iaml/2011/slides/pca.pdf
http://vision.stanford.edu/teaching/cs131_fall1415/lectures/lecture17_face_recognition_cs131.pdf

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