Unsupervised Learning (2)

Never Stand Still

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:

- 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

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.

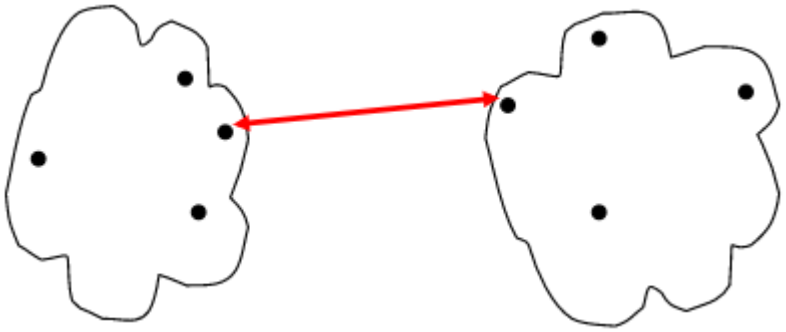
Hierarchical Clustering

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.

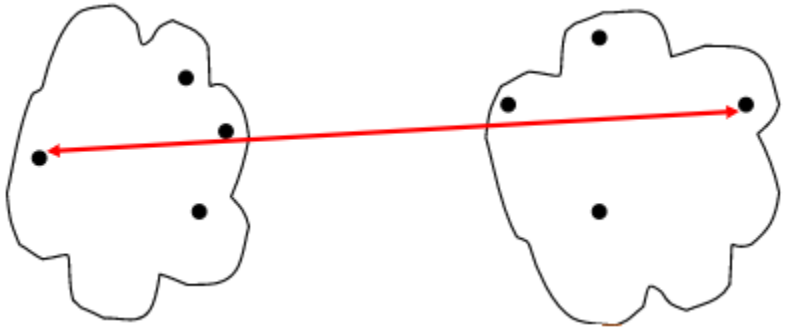
Single linkage $d_{k,i\cup j} = \min(d_{ki}, d_{kj})$ and $\alpha_i = \alpha_j = \frac{1}{2}$ and $\gamma = -\frac{1}{2}$.
Complete linkage $d_{k,i\cup j} = \max(d_{ki}, d_{kj})$ and $\alpha_i = \alpha_j = \frac{1}{2}$ and $\gamma = \frac{1}{2}$.
Average linkage $d_{k,i\cup j} = \frac{n_i d_{ki}}{n_i+n_j}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

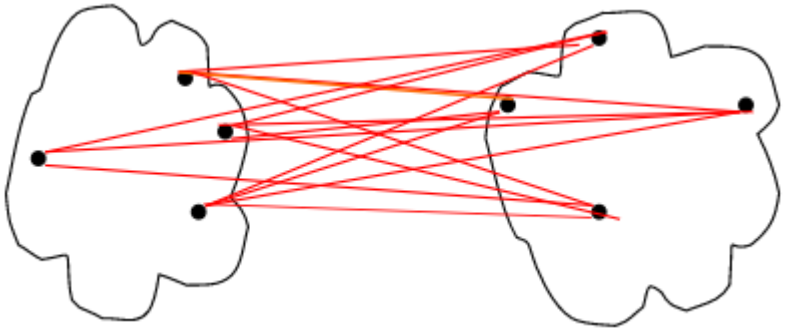
Hierarchical Clustering



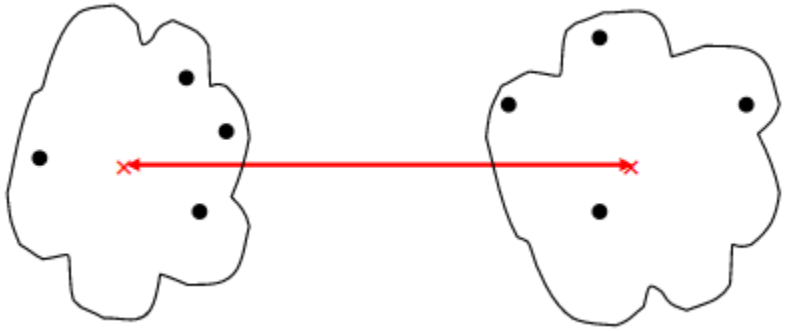
Single link



Complete link



Average link



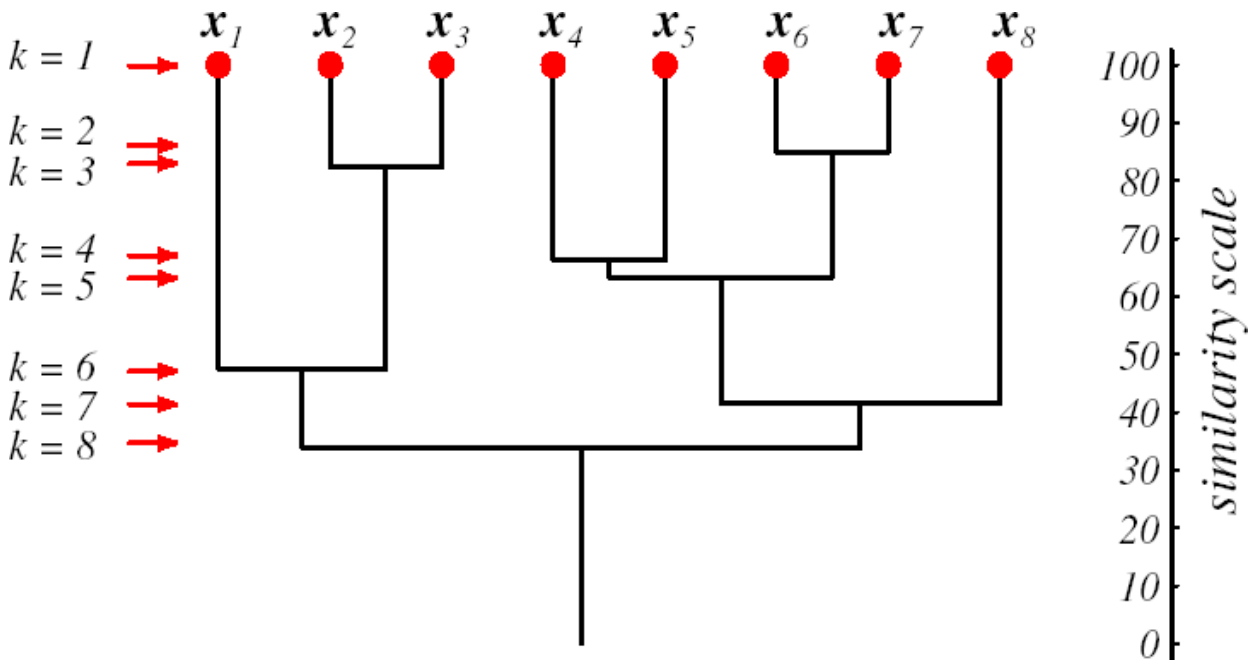
Centroid distance

Hierarchical Clustering

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

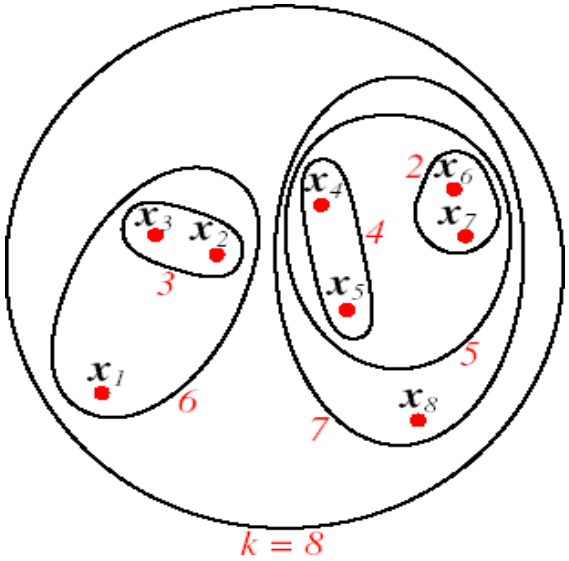
- at level 1 all points in individual clusters
- x_6 and x_7 are most similar and are merged at level 2
- dendrogram drawn to scale to show similarity between grouped clusters

Hierarchical Clustering



Hierarchical Clustering

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



Dendrograms

Two things to beware of:

- tree structure is not unique for given clustering - for each bottom-up merge the sub-tree to the right or left must be specified - 2^{n-1} ways to permute the n leaves in a dendrogram
- hierarchical clustering imposes a bias - the clustering forms a dendrogram despite the possible lack of a implicit hierarchical structuring in the data

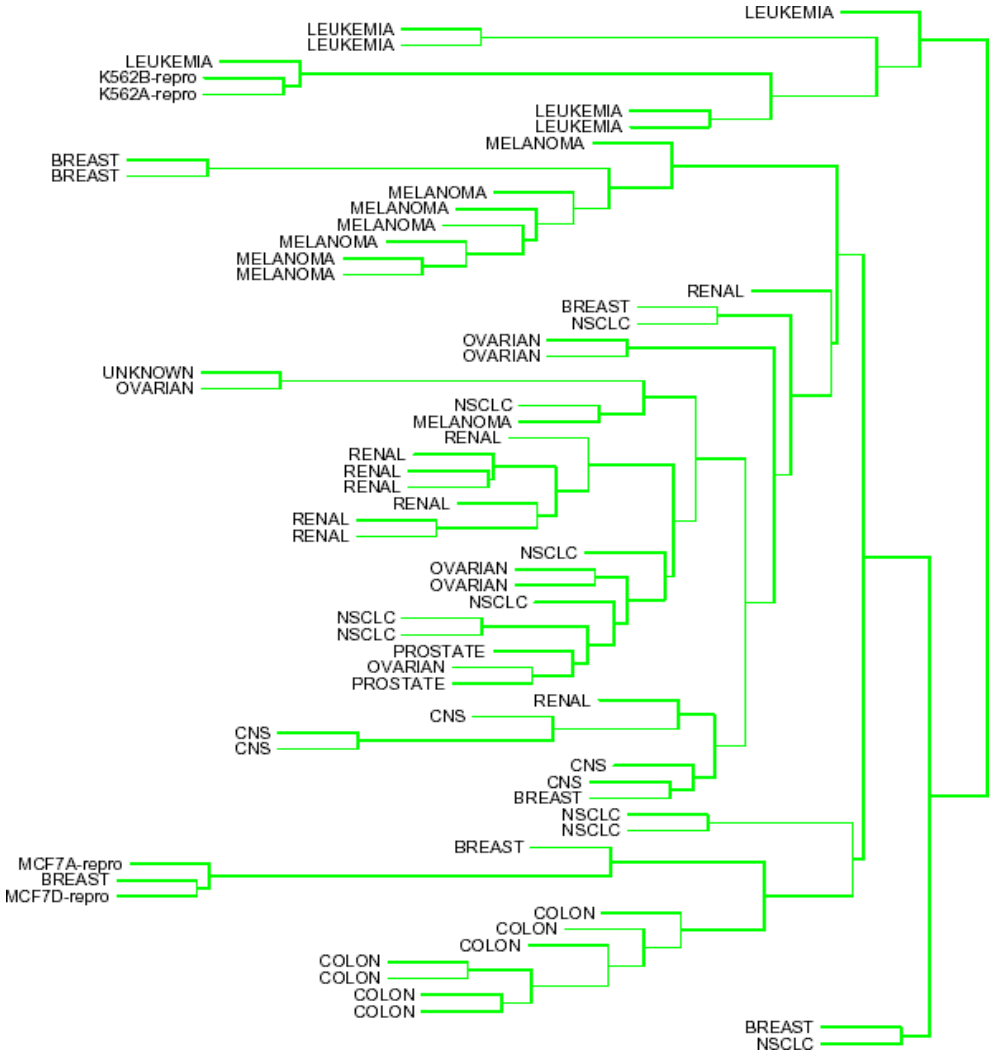
Dendrograms

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.

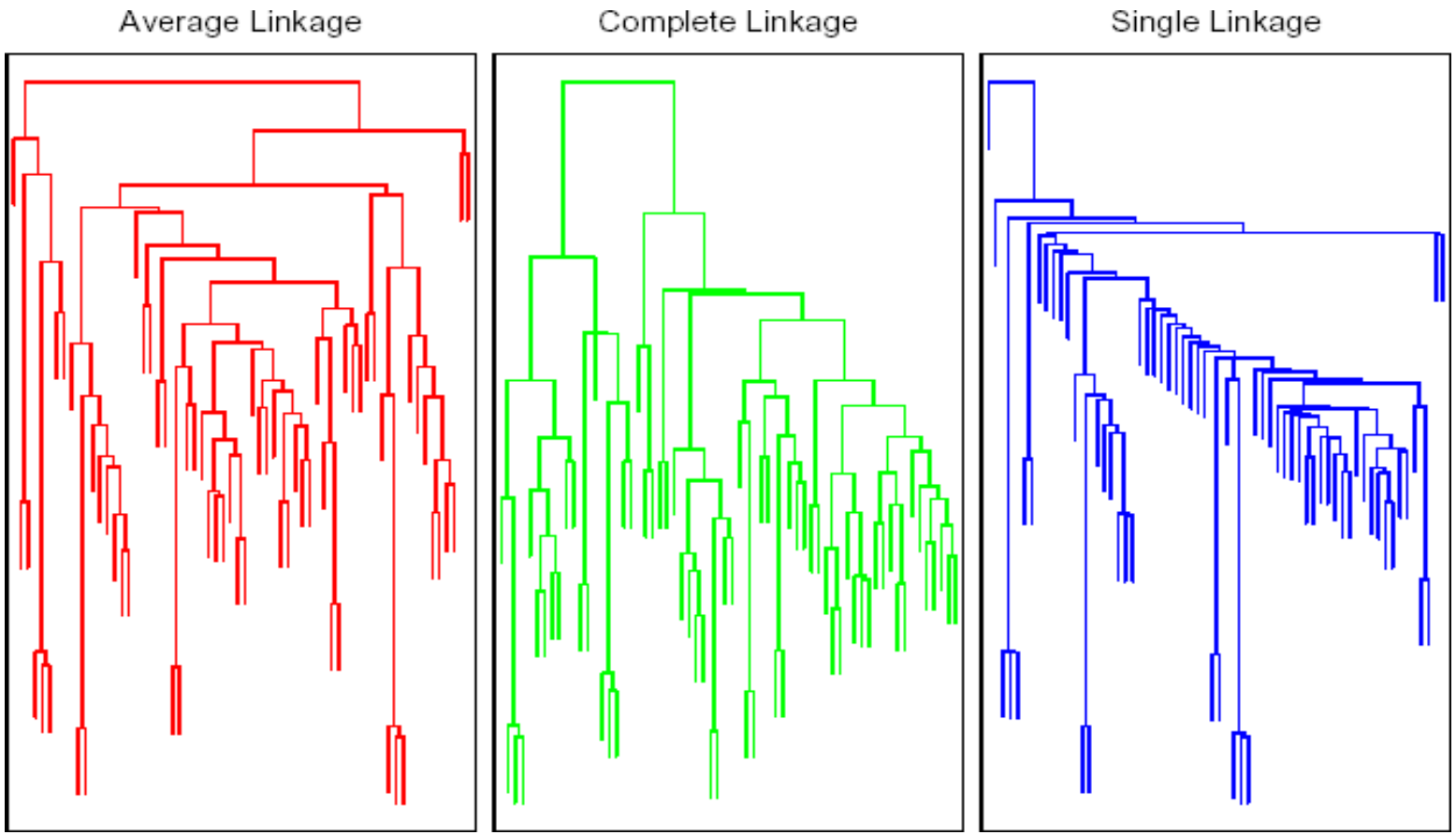
Followed on next slide by diagram showing:

- 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

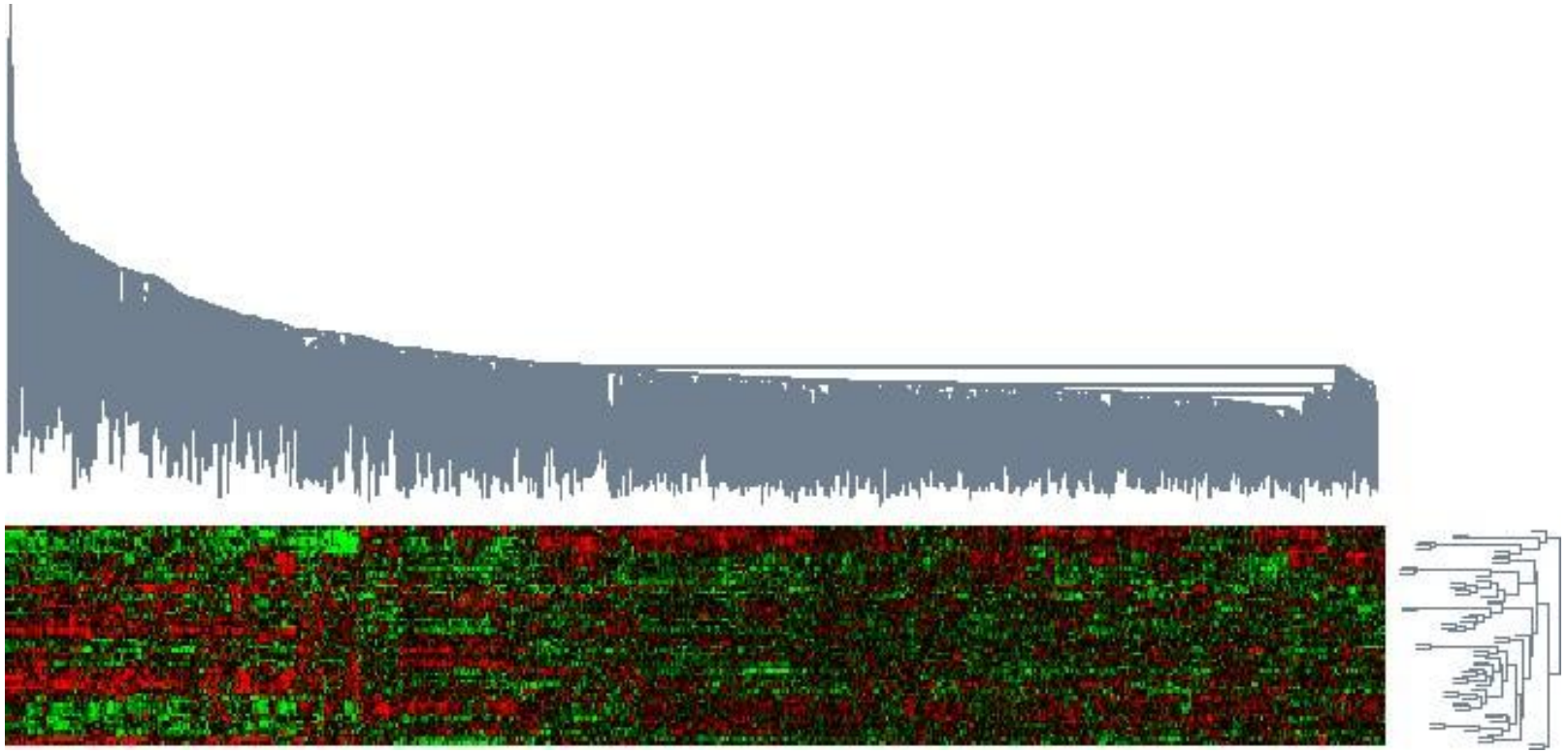
Dendrograms



Dendrograms



Dendrograms

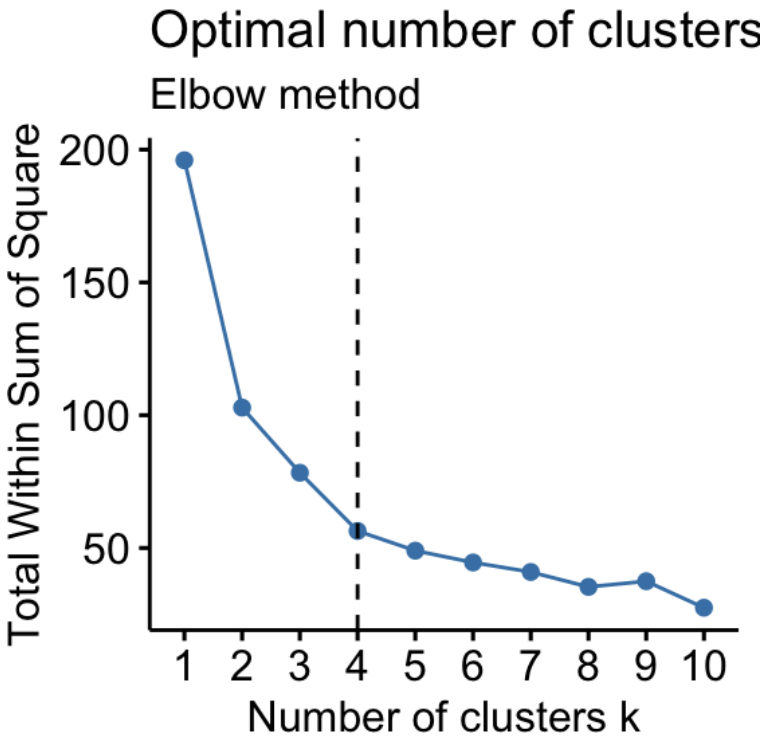


Number of Clusters

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

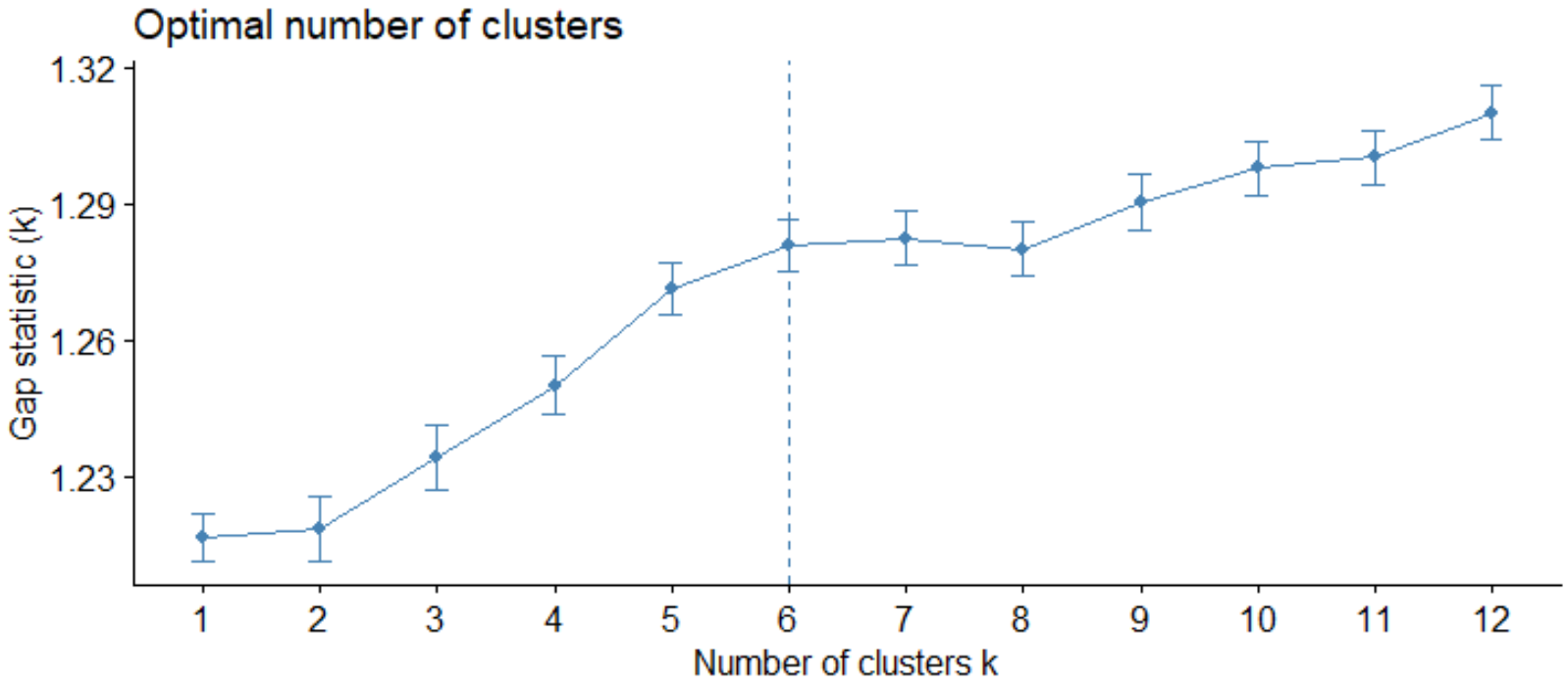


Number of Clusters

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} : $Gap(k)=\sum_b log(W_{kb})-log(W_k)$. 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, define its silhouette width $s(i) \in [-1, 1]$:

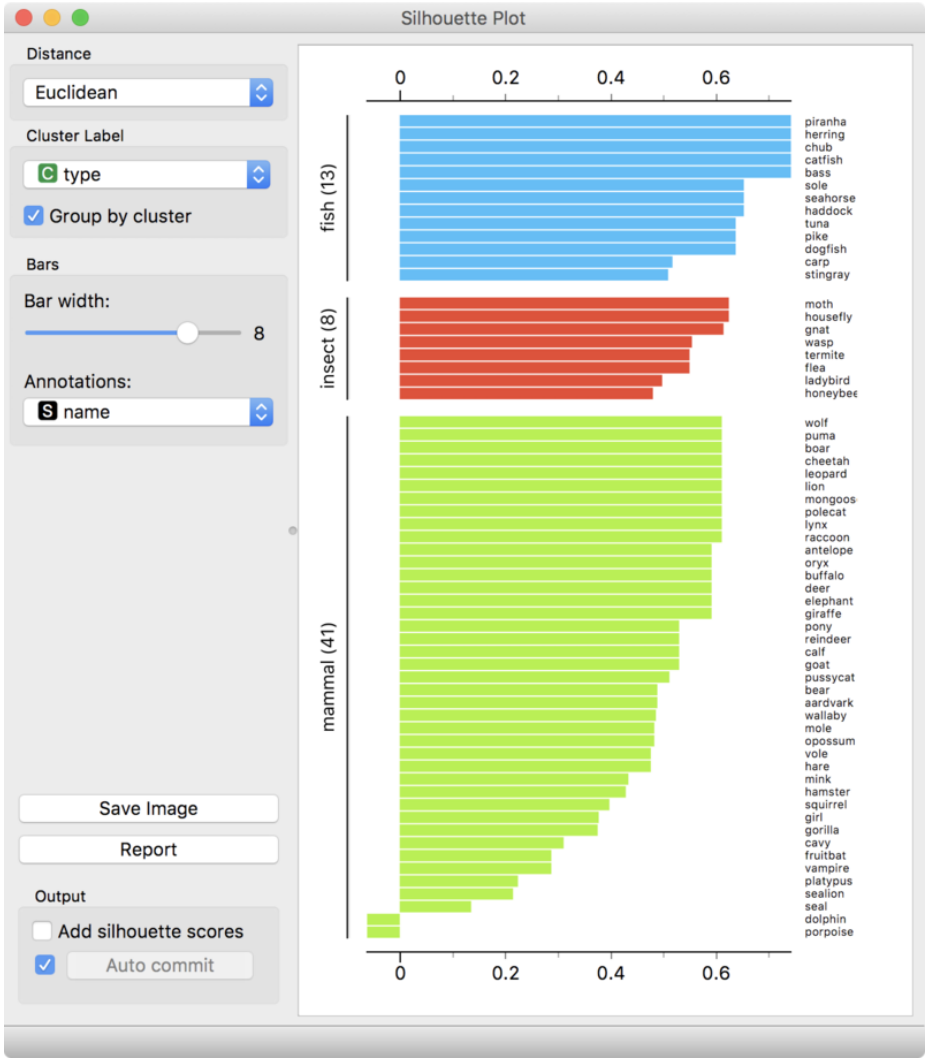
Let $a(i)$ be the average dissimilarity between i and elements of $P(i)$, i.e., cluster to which i belongs,

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

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

Cluster quality – Silhouette plot



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

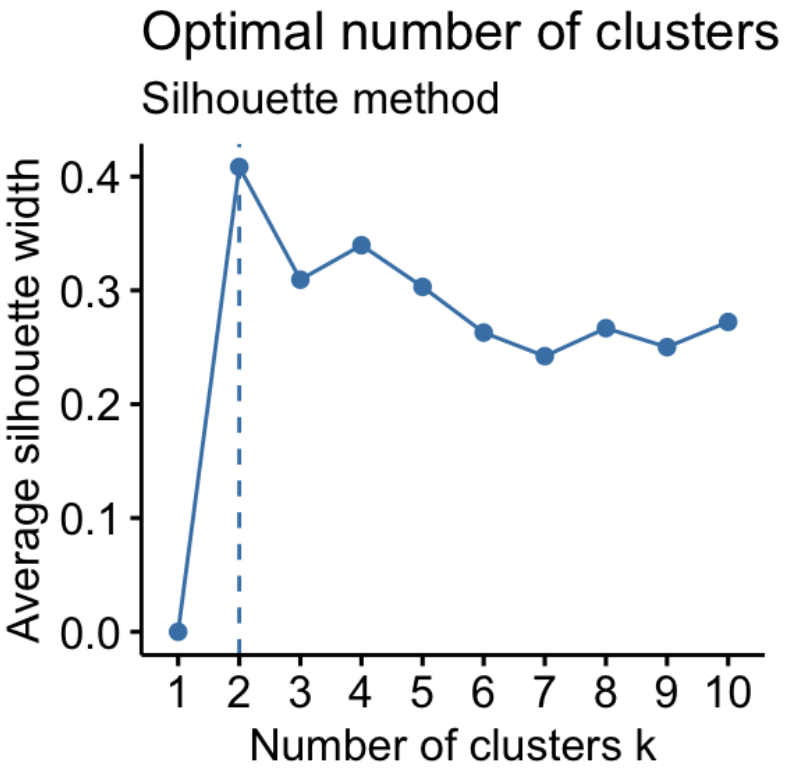
https://en.wikipedia.org/wiki/Silhouette_(clustering)

Cluster quality – Silhouette plot

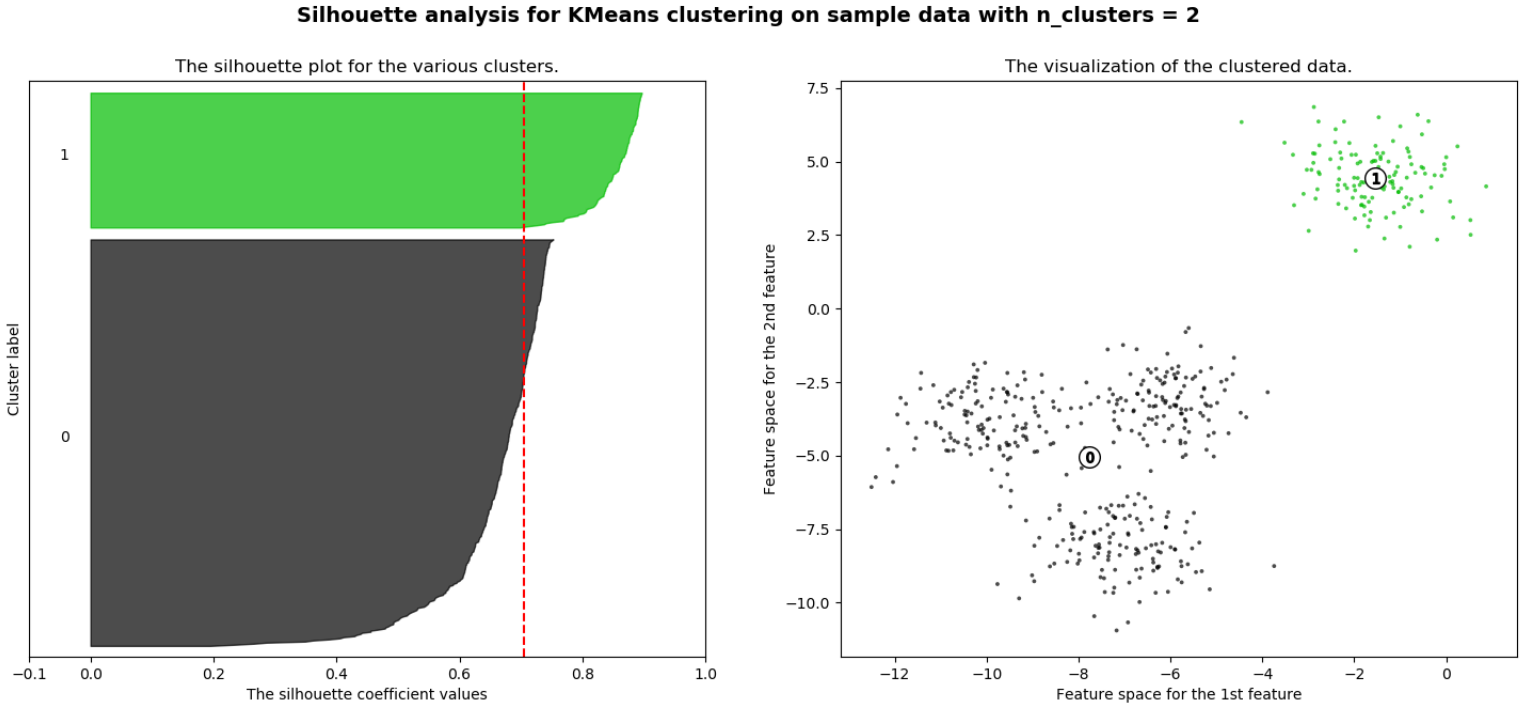
How can we intepret a Silhouette plot ?

- 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

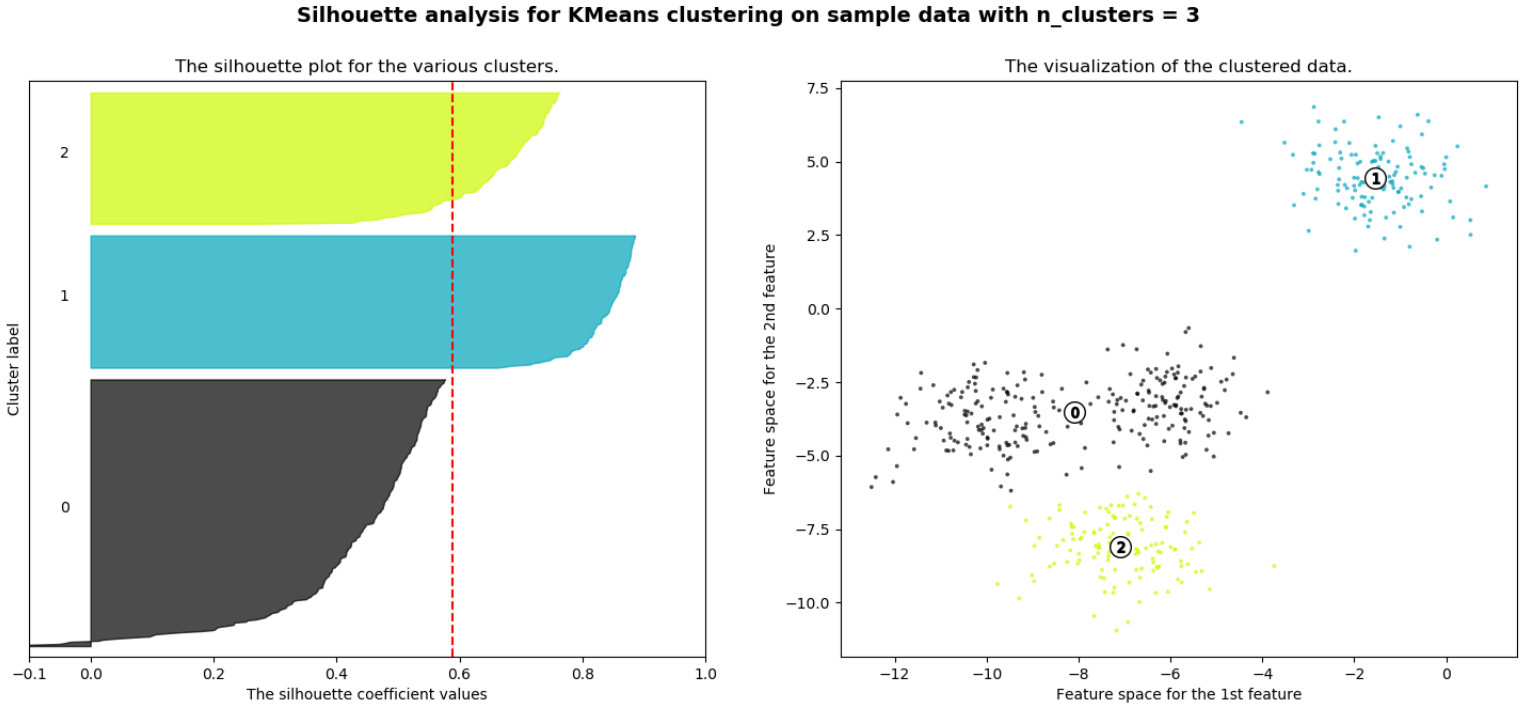


Cluster quality – Silhouette plot



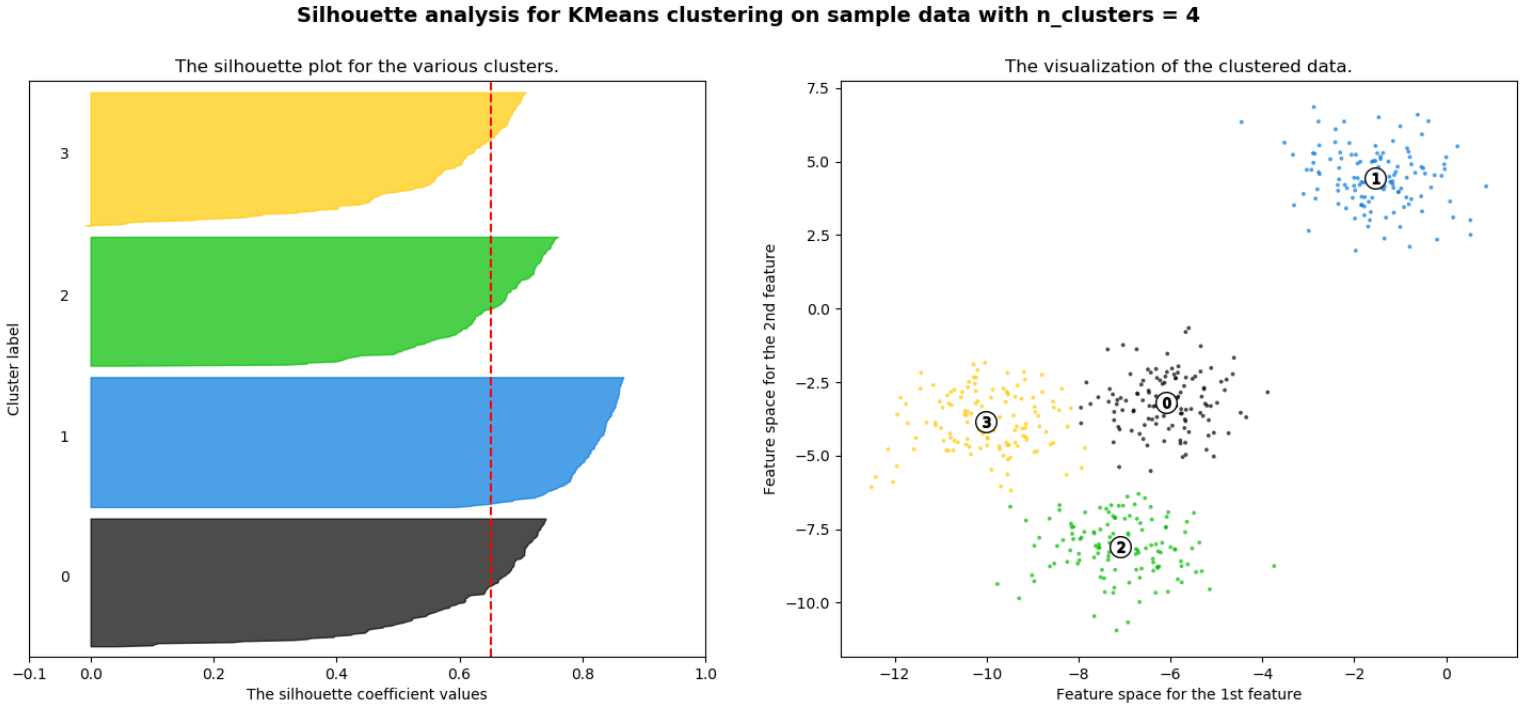
For n_clusters = 2 The average silhouette_score is : 0.7049787496083262

Cluster quality – Silhouette plot



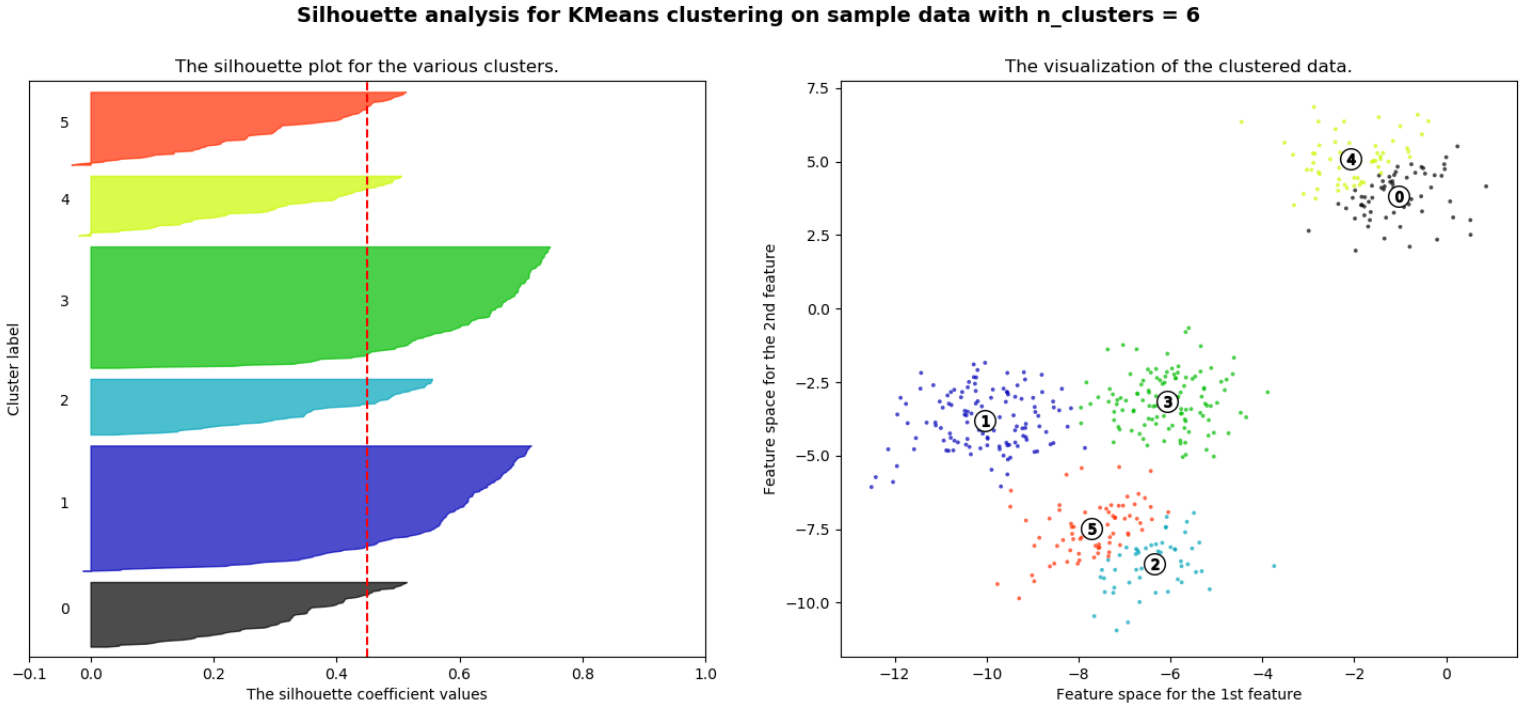
For n_clusters = 3 The average silhouette_score is : 0.5882004012129721

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
- Evaluation 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
- Higher nodes can be very heterogeneous
- Problem may not have a “real” hierarchical structure

Clustering summary

- k-means avoids some of these problems, but also has drawbacks
- 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 (average similarity between feature vectors in a cluster and the centroid) and separation (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

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)

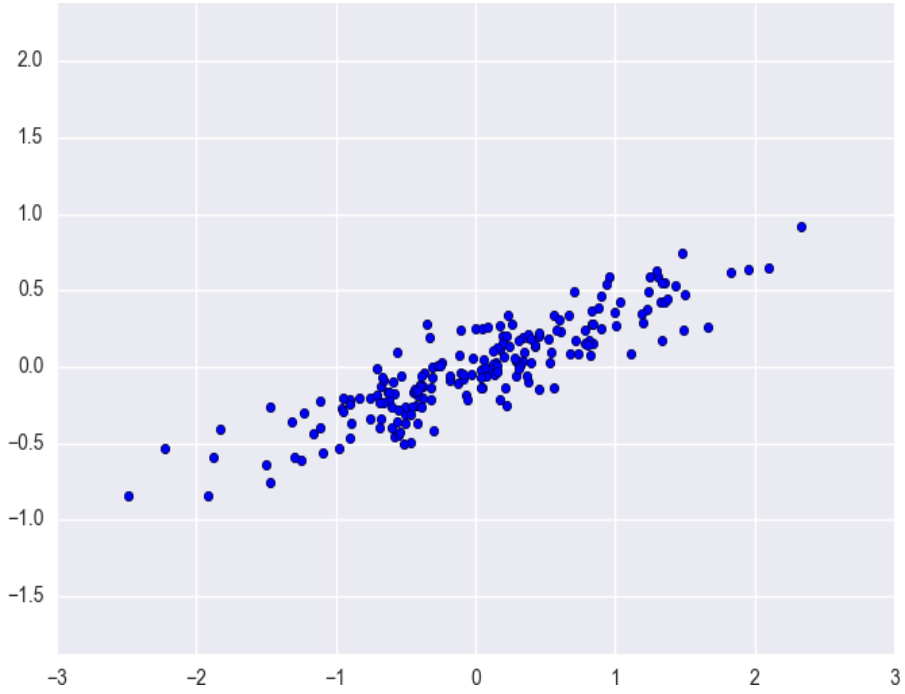
Key idea: look for features in a transformed space so that each dimension in the new space captures the most variation in the original data when it is projected onto that dimension.

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

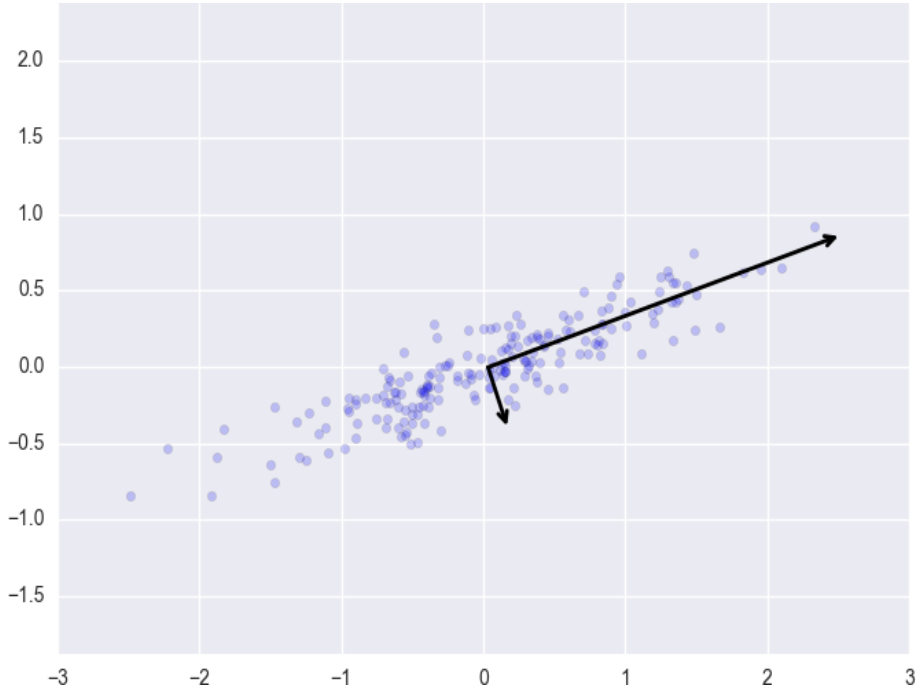
PCA Example

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.

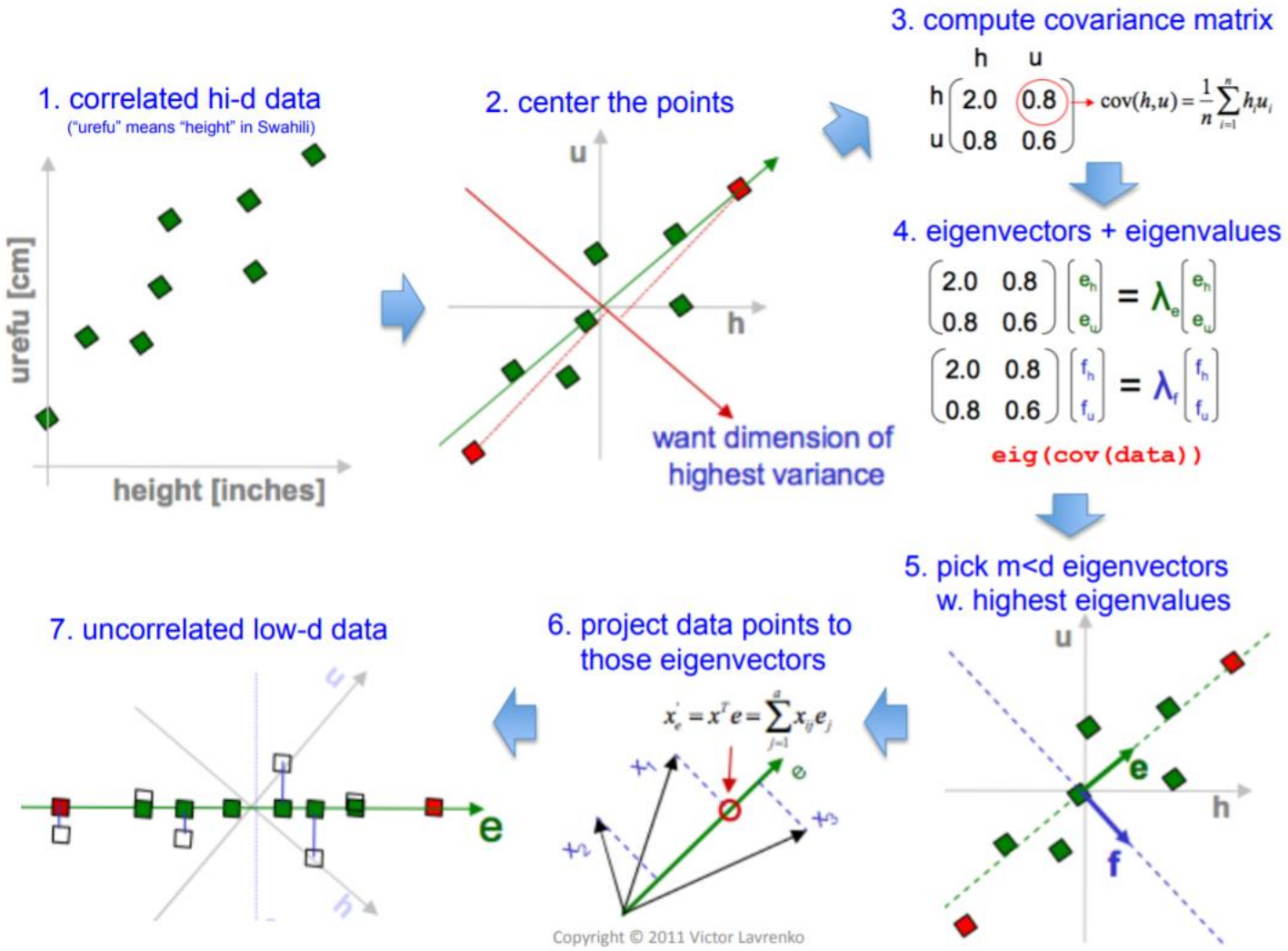


PCA Algorithm

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

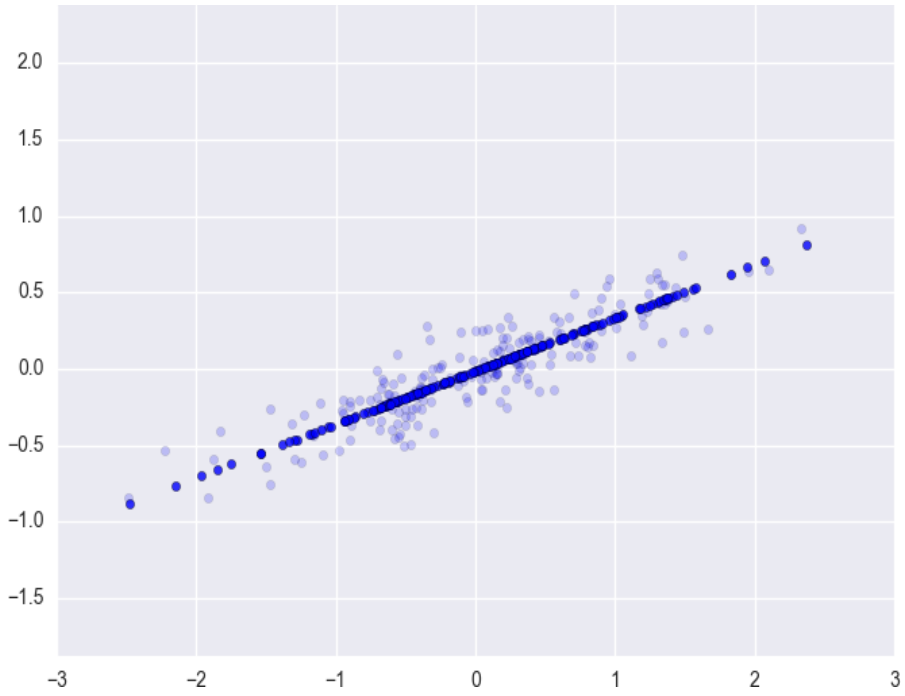
1) take the data as an $n \times m$ matrix $\mathbf{X}$
2) "centre" the data by subtracting the mean of each column
3) construct covariance matrix $\mathbf{C}$ from centred matrix
4) compute eigenvector matrix $\mathbf{V}$ (rotation) and eigenvalue matrix $\mathbf{S}$ (scaling) such that $\mathbf{V}^{-1}\mathbf{C}\mathbf{V}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

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

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

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

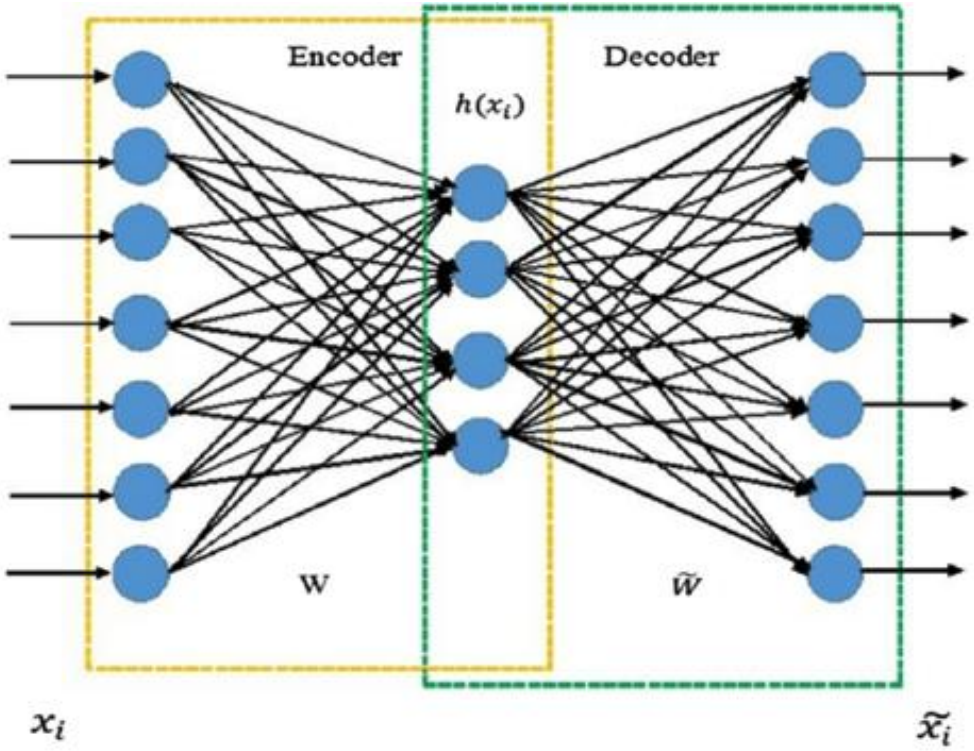
- Only selecting the top K eigenfaces → reduces the dimensionality.
- Fewer eigenfaces result in more information loss, and hence less discrimination between faces.

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 Analysis (next slide), finding hidden “sub-groups” for recommender systems, and so on

Autoencoders

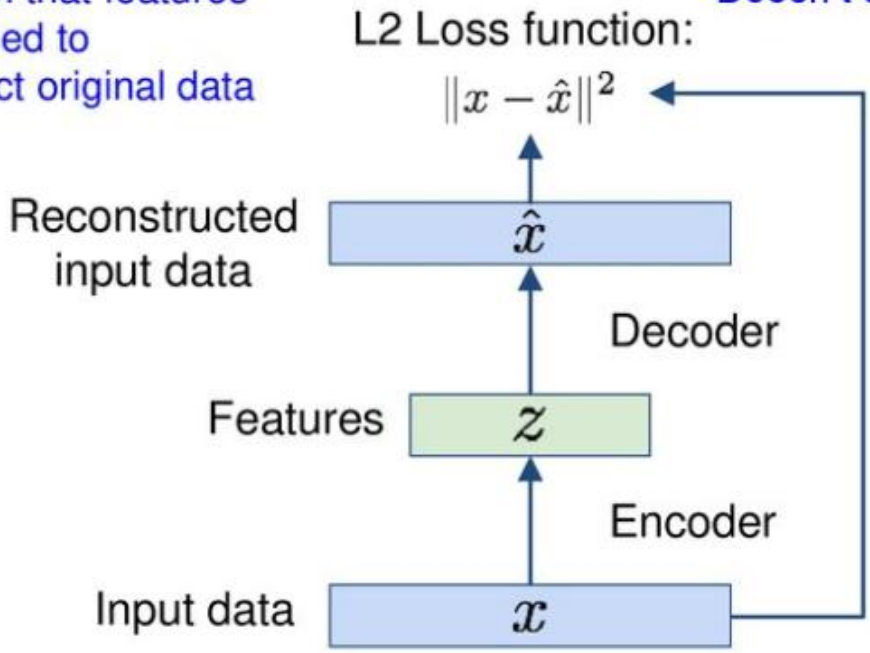
A neural network model – the encoder transforms the data in a way the decoder can then interpret and reconstruct with minimum error.



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
- 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

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

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