12. k-means, hierarchical clustering and principal component analysis

Shabana K M
PhD Research Scholar
Computer Science and Engineering

IIT Palakkad

12 December 2021

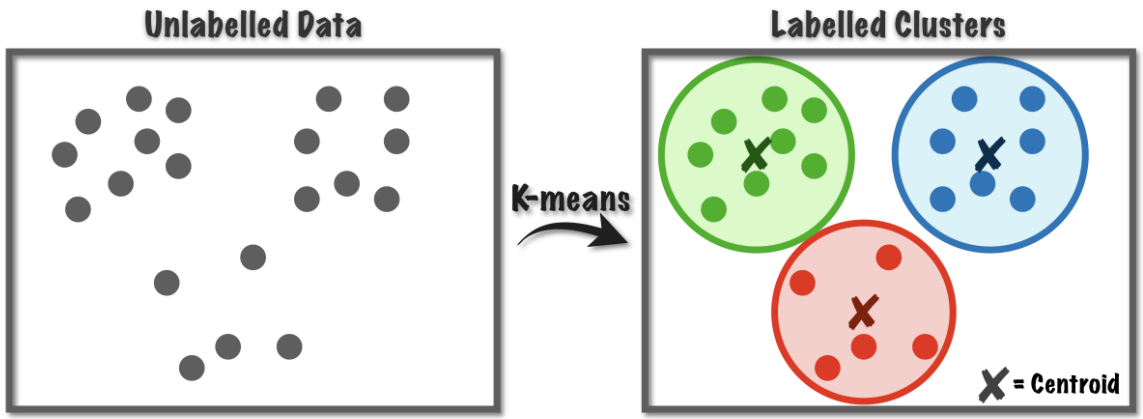
INDIAN INSTITUTE
OF TECHNOLOGY
PALAKKAD

Recap

- Artificial neural networks
  - feed-forward computation
  - back-propagation
  - limitations
- Clustering
  - types of clustering algorithms

k-means clustering algorithm

- one of the simplest and most popular clustering algorithm
- `partitional, centroid` based clustering technique
  - partitions the given data into k clusters
  - each cluster represented by a `cluster centroid`
  - k is specified by the user
- each data point belongs to exactly one cluster



k-means algorithm

Let the set of data points D be $\{x_1, x_2, ..., x_n\}$ where each $x_i \in \mathcal{R}^d$

Given **k**, the `k-means` algorithm works as follows:

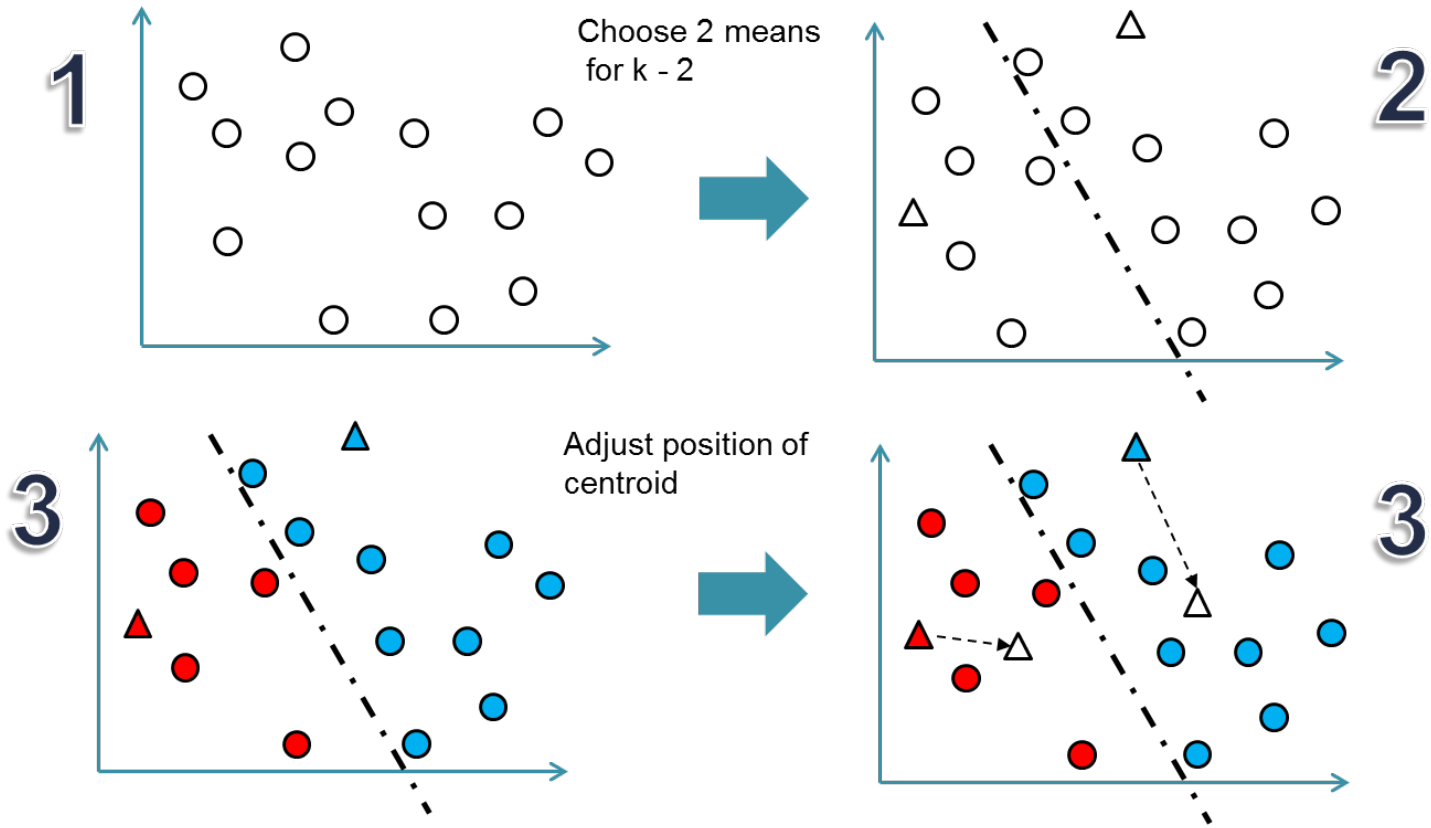
1. Choose k (random) data points (seeds) to be the initial centroids
2. Assign each data point to the closest centroid
3. Re-compute the centroids using the current cluster memberships

$$\mu_j = \frac{1}{N_j} \sum_{x \in C_j} x \quad \text{where}$$

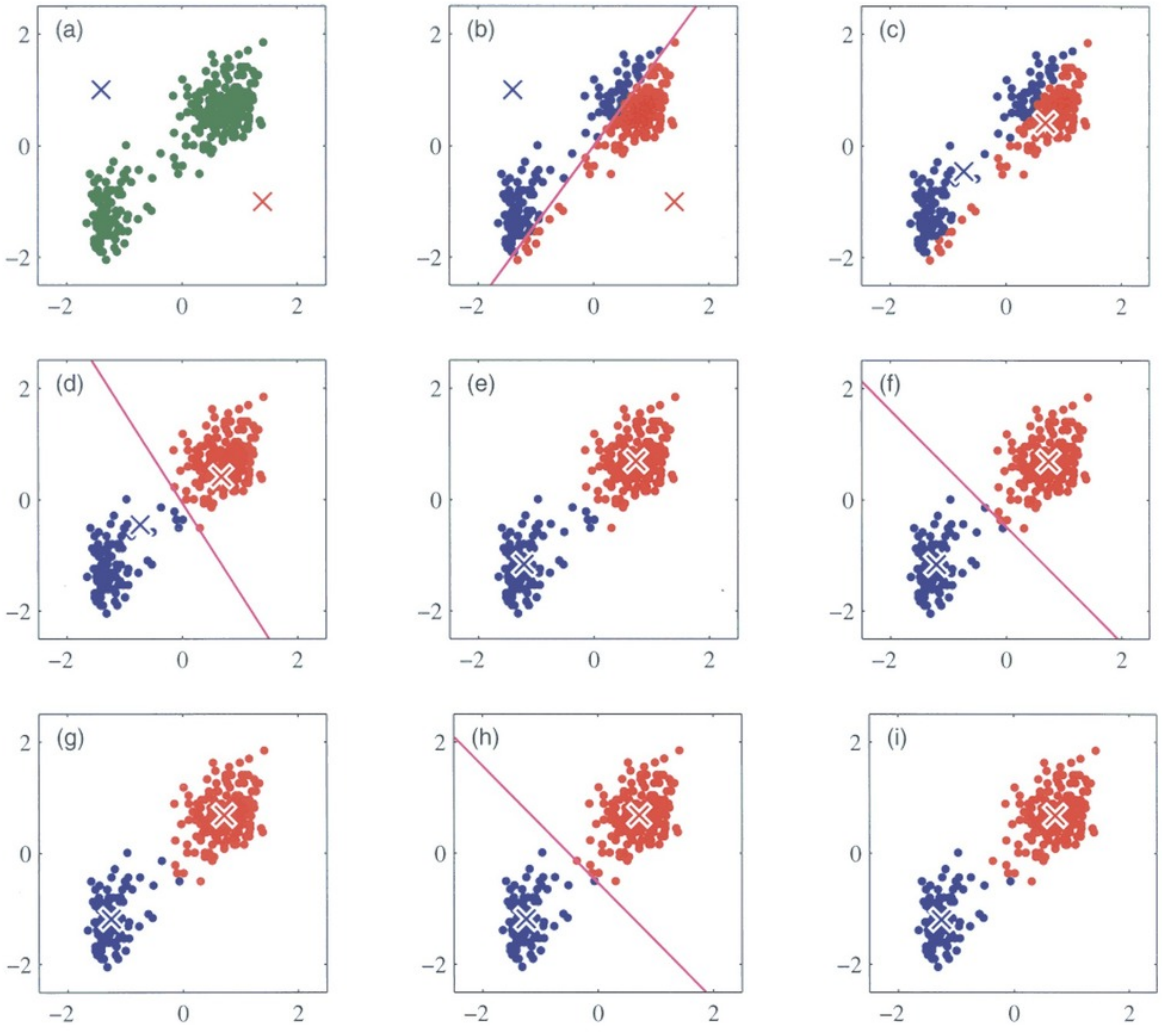
μ_j : centroid of cluster j
N_j : number of data points belonging to cluster j
C_j : set of points in cluster j

4. If a convergence criterion is not met, repeat steps 2 and 3

k-means clustering



k-means clustering: Example



k-means algorithm: Convergence criteria

- no (or minimum) re-assignments of data points to different clusters
- no (or minimum) change of centroids
- minimum decrease in the **sum of squared error(SSE)**

$$SSE = \sum_{i=1}^{k} \sum_{x \in C_j} d(x, \mu_j)^2$$

  - C_j is the j^{th} cluster
  - μ_j is the centroid of cluster j
  - $d(x, \mu_j)$ is the Euclidean distance between data point x and centroid μ_j

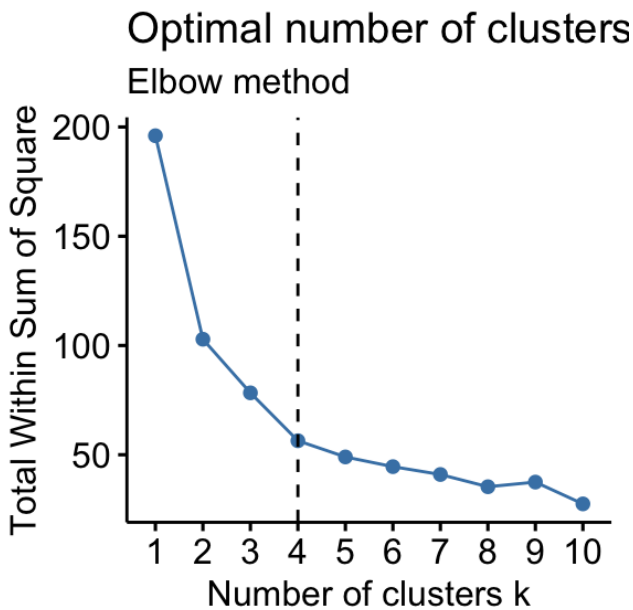
Why use k-means?

- **simple:** easy to understand and to implement
- **efficient:** Time complexity: $O(tkn)$, where
  n : number of data points
  k : number of clusters and
  t : number of iterations
- scales to large data sets
- guaranteed convergence
- easily adapts to new examples

k-means: Disadvantages

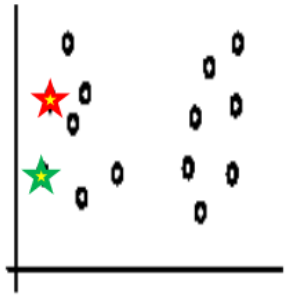
- need to choose k manually

  solution: Use the `elbow method` to select k

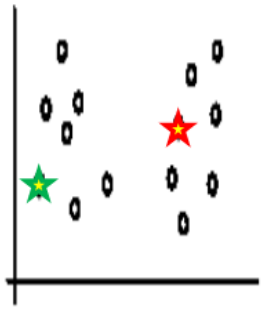


k-means: Disadvantages

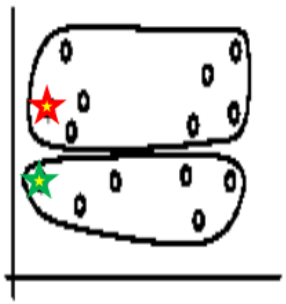
- final clustering solution dependent on the initial centroids chosen



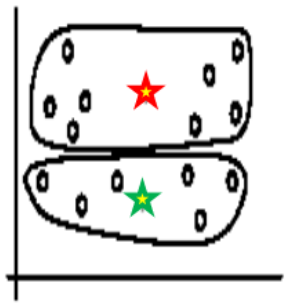
Random selection of seeds (centroids)



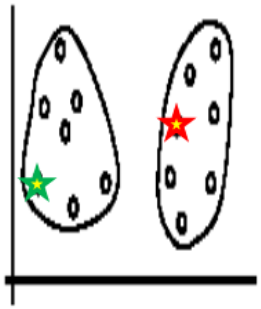
Random selection of seeds (centroids)



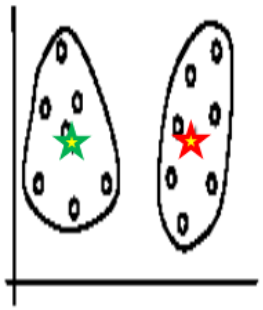
Iteration 1



Iteration 2



Iteration 1



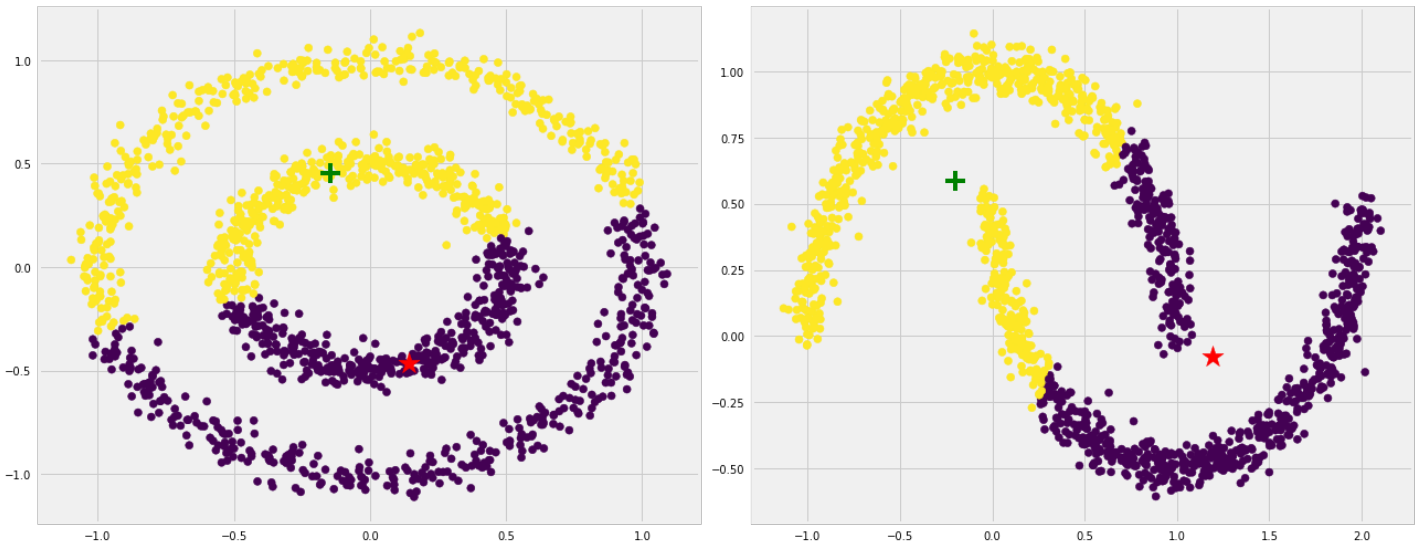
Iteration 2

k-means: Disadvantages

- algorithm only applicable if `mean` is defined

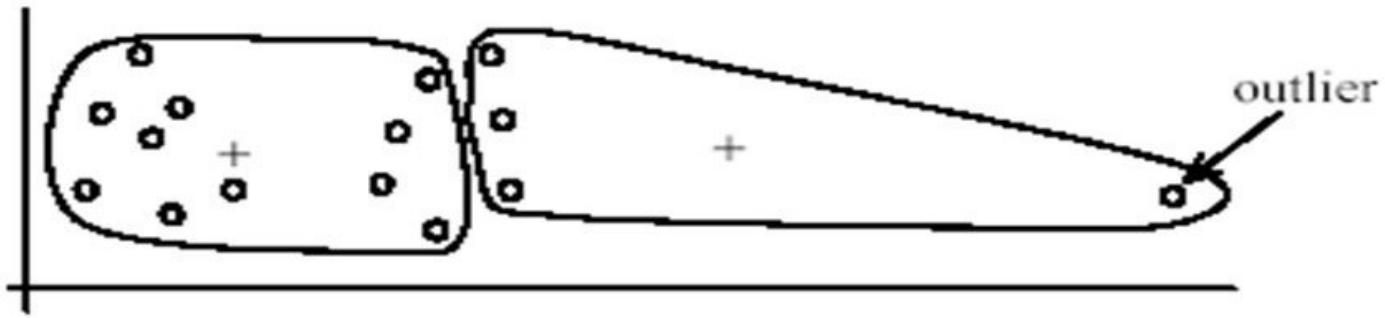
  for categorical data, `k-mode` - centroid represented by the most frequent values

- not suitable for discovering clusters that are not spherical

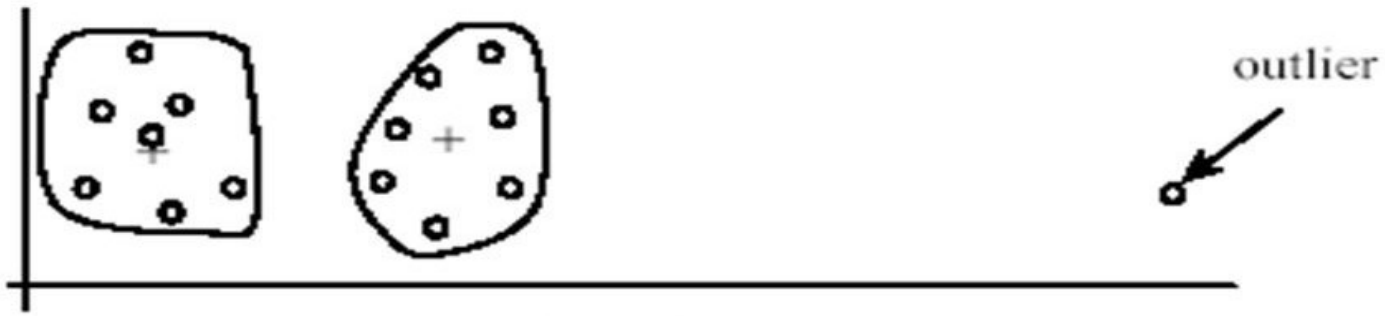


k-means: Disadvantages

- sensitive to **outliers**



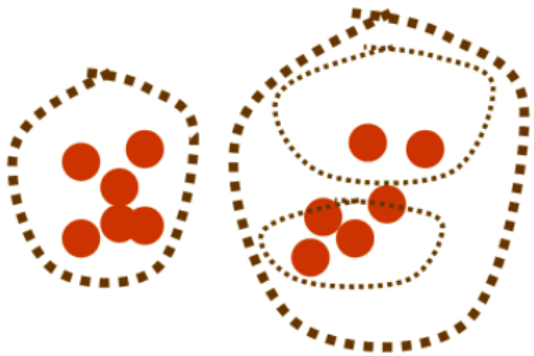
(A): Undesirable clusters



(B): Ideal clusters

Hierarchical clustering

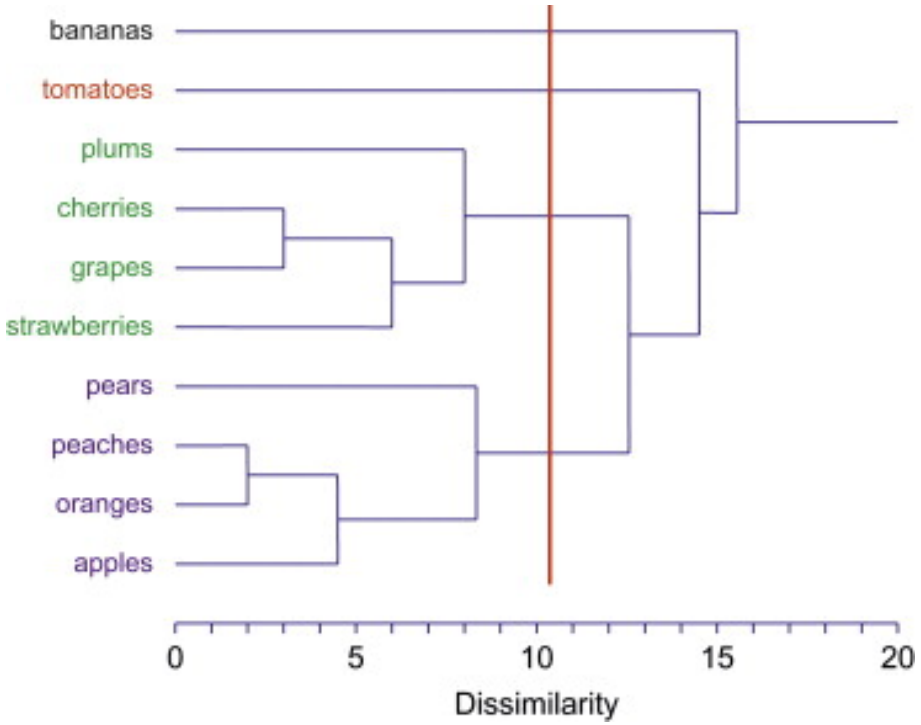
- clustering algorithm that seeks to build a hierarchy of clusters
- gives a hierarchical decomposition of the data based on group similarities
- produce a nested sequence of clusters usually represented as a `dendrogram`



- the merges and splits are determined in a greedy manner

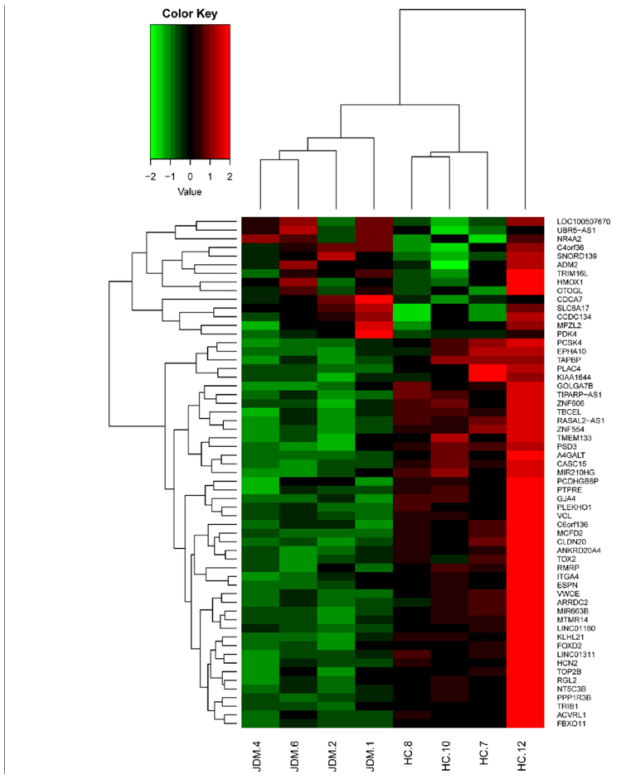
Hierarchical clustering: Applications

Biological taxonomy



Hierarchical clustering: Applications

Clustering gene expression data



Types of hierarchical clustering

- **Divisive (top down) clustering**

  Starts with all data points in one cluster, the root

  - splits the root into a set of child clusters. Each child cluster is recursively divided further
  - stops when only singleton clusters of individual data points remain, i.e., each cluster with only a single point

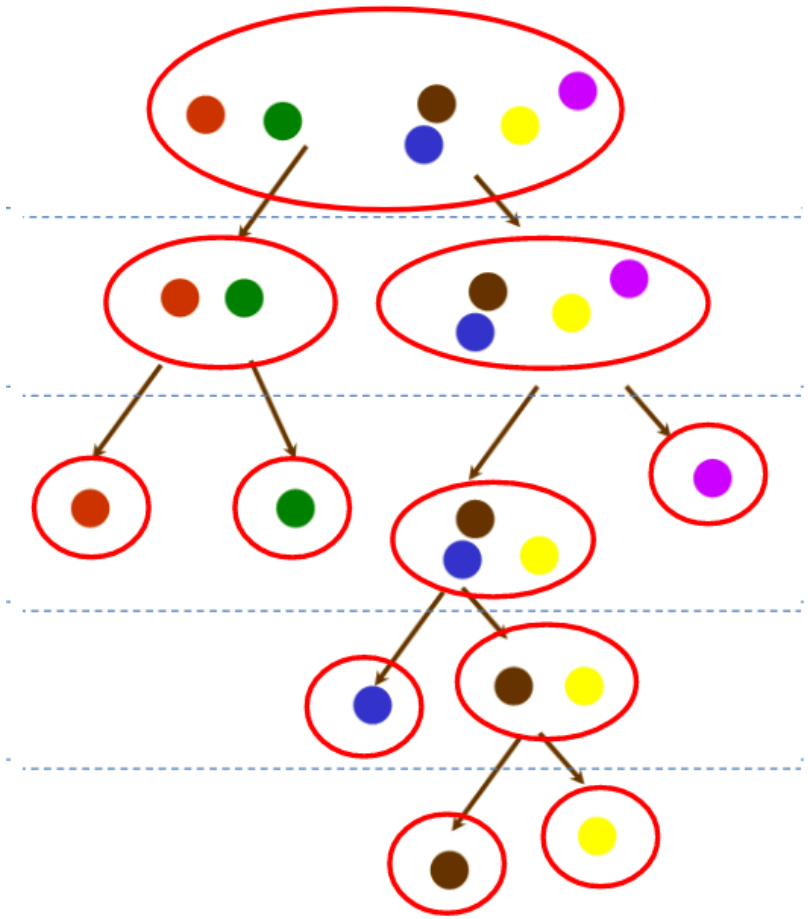
- **Agglomerative (bottom up) clustering**

  The dendrogram is built from the bottom level by

  - merging the most similar (or nearest) pair of clusters
  - stopping when all the data points are merged into a single cluster (i.e., the root cluster)

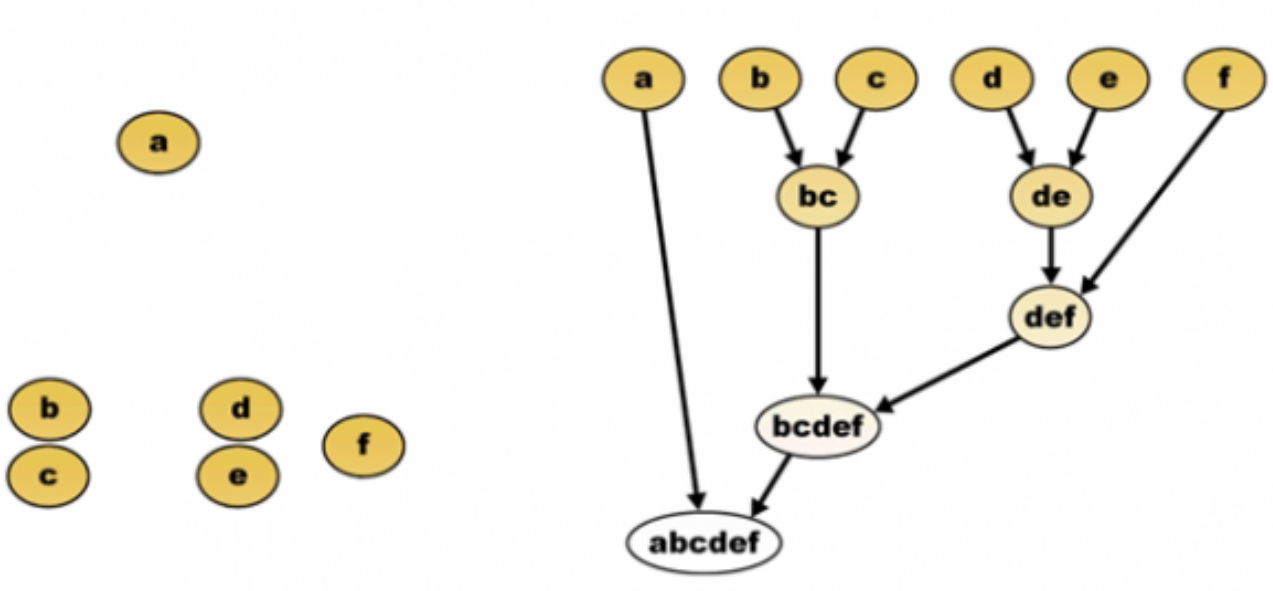
Divisive hierarchical clustering

- Any partitional clustering algorithm that produces a fixed number of clusters can be used



Agglomerative clustering

1. Make each data point a single-point cluster $\rightarrow$ N `singleton` clusters
2. **while** there is more than 1 cluster
   - find two nearest clusters
   - merge them



Measuring cluster distance

Four common ways to measure cluster distance

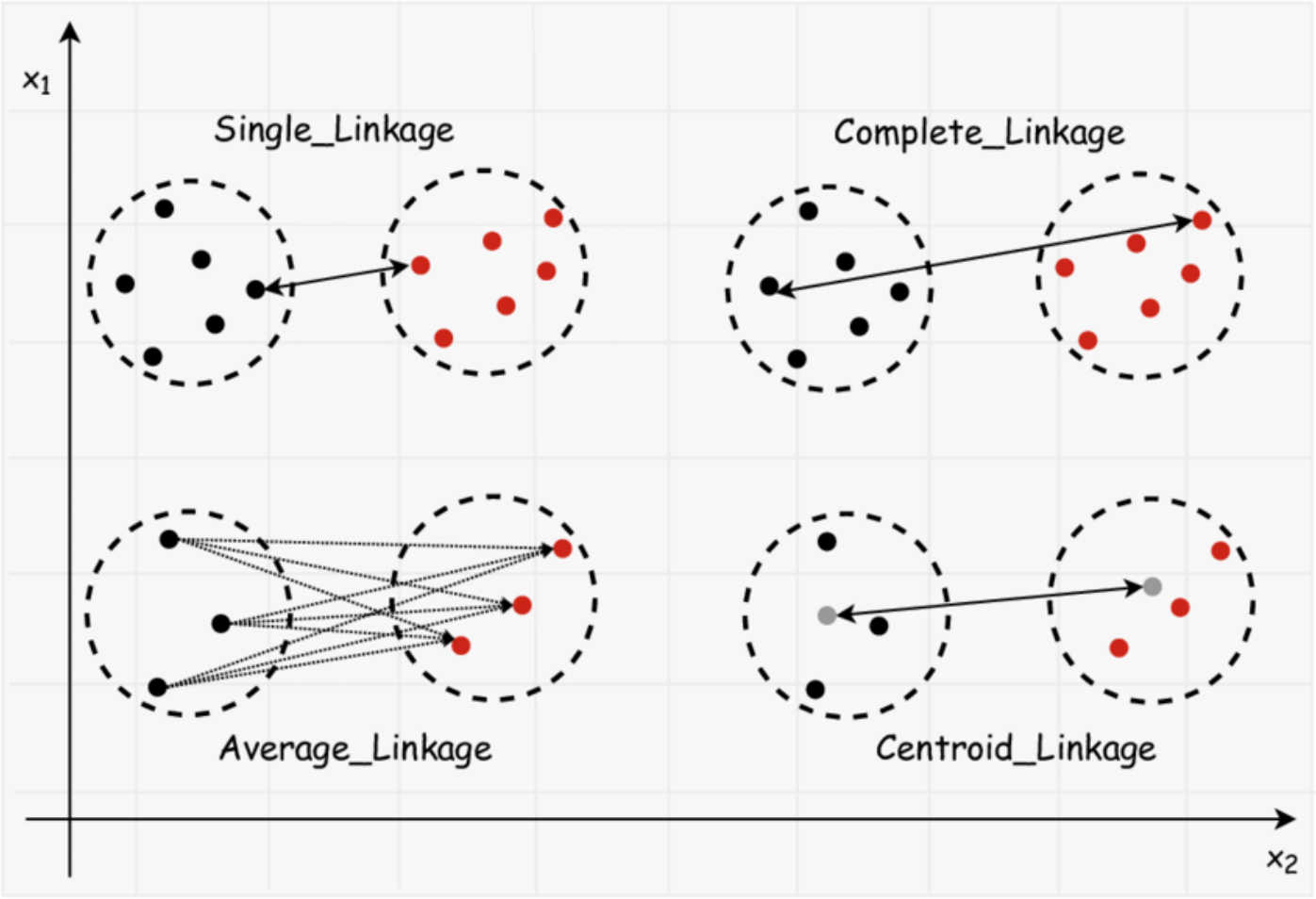
1. **Minimum distance or Single linkage**
   $d_{min}(C_i, C_j) = \min_{x \in C_i, y \in C_j} ||x - y||$

2. **Maximum distance or Complete linkage**
   $d_{max}(C_i, C_j) = \max_{x \in C_i, y \in C_j} ||x - y||$

3. **Average distance or Average linkage**
   $d_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{x \in C_i} \sum_{y \in C_j} ||x - y||$

4. **Mean distance or Centroid linkage**
   $d_{mean}(C_i, C_j) = ||\mu_i - \mu_j||$

Measuring cluster distance



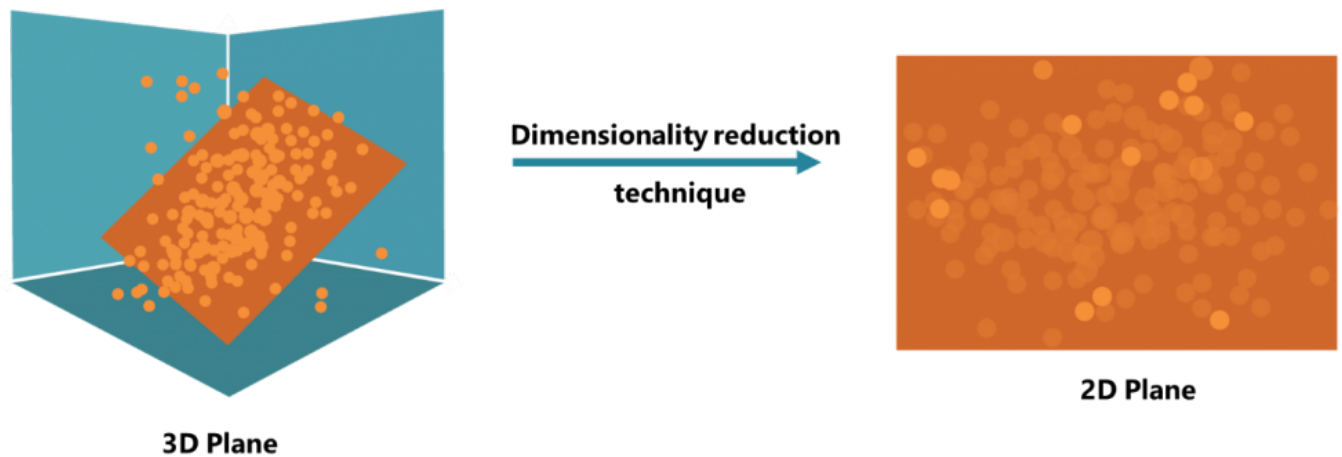
Stopping criteria

- Stop splitting when:
  - there are k clusters
  - the **cohesion** of the cluster resulting from the best merger falls above a threshold
    - `cohesion` measures how closely related objects are in a cluster
    - can be measured in different ways - maximum or average distance between points in a cluster, etc.
  - there is a sudden jump in the cohesion value

- The standard hierarchical agglomerative clustering algorithm has a time complexity of $O(n^3)$ and requires $\Omega(n^2)$ memory

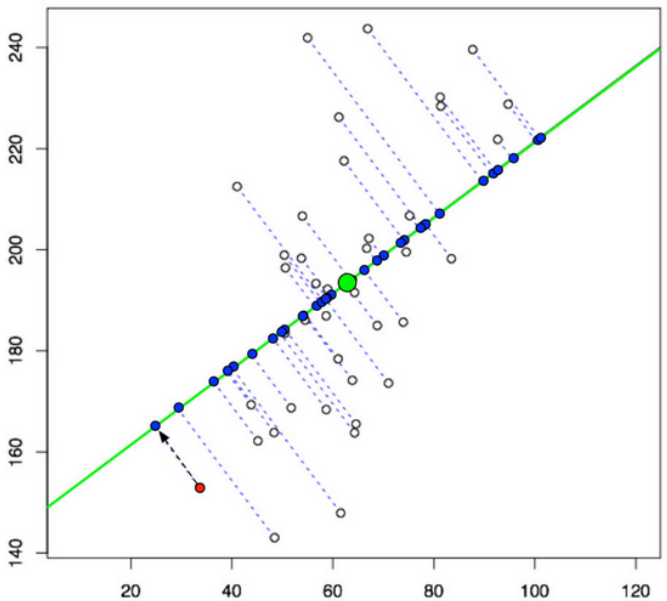
Dimensionality reduction techniques

- transforms data from a high-dimensional space into a low-dimensional space that retains meaningful properties of the original data
- smaller data sets easier to explore and visualize
- helps improve model performance by eliminating redundant features and reducing noise
- approaches divided into **feature selection** and **feature extraction**



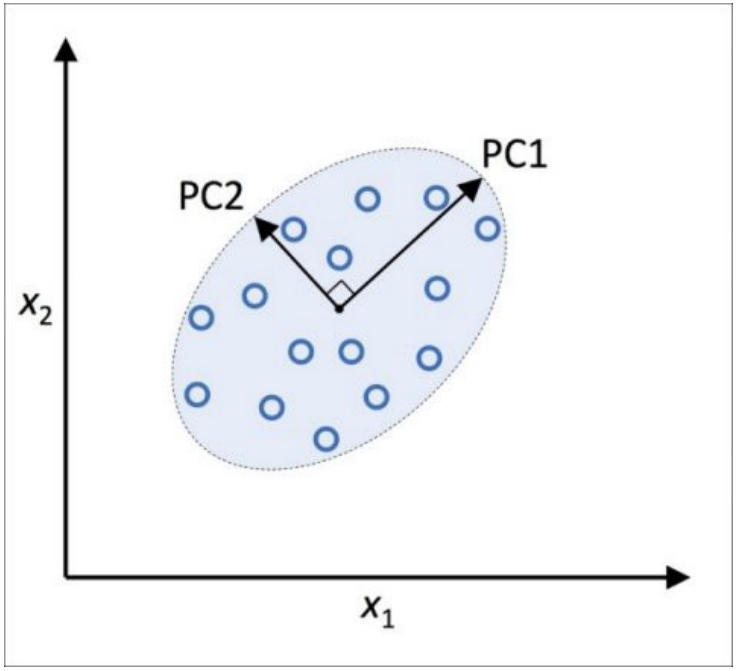
Principal Component Analysis (PCA)

- performs dimensionality reduction through feature extraction
- projects data onto a lower dimensional linear subspace - `linear dimensionality reduction technique`
- transforms features into a new set of variables called as `principal components` - linear combinations of original features
- mapping performed such that the variance of data is maximized



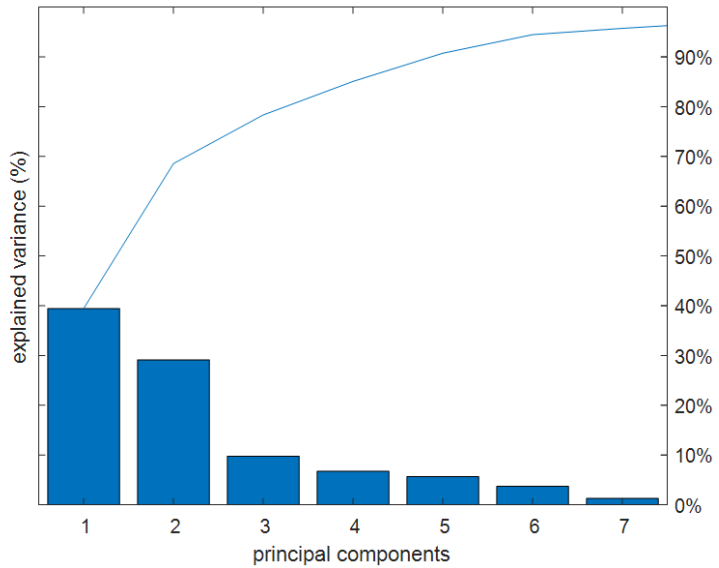
Principal components

- **First principal component:** direction along which projections have the largest variance
- **Second principal component:** orthogonal to the first principal component and captures the second-largest part of the variance



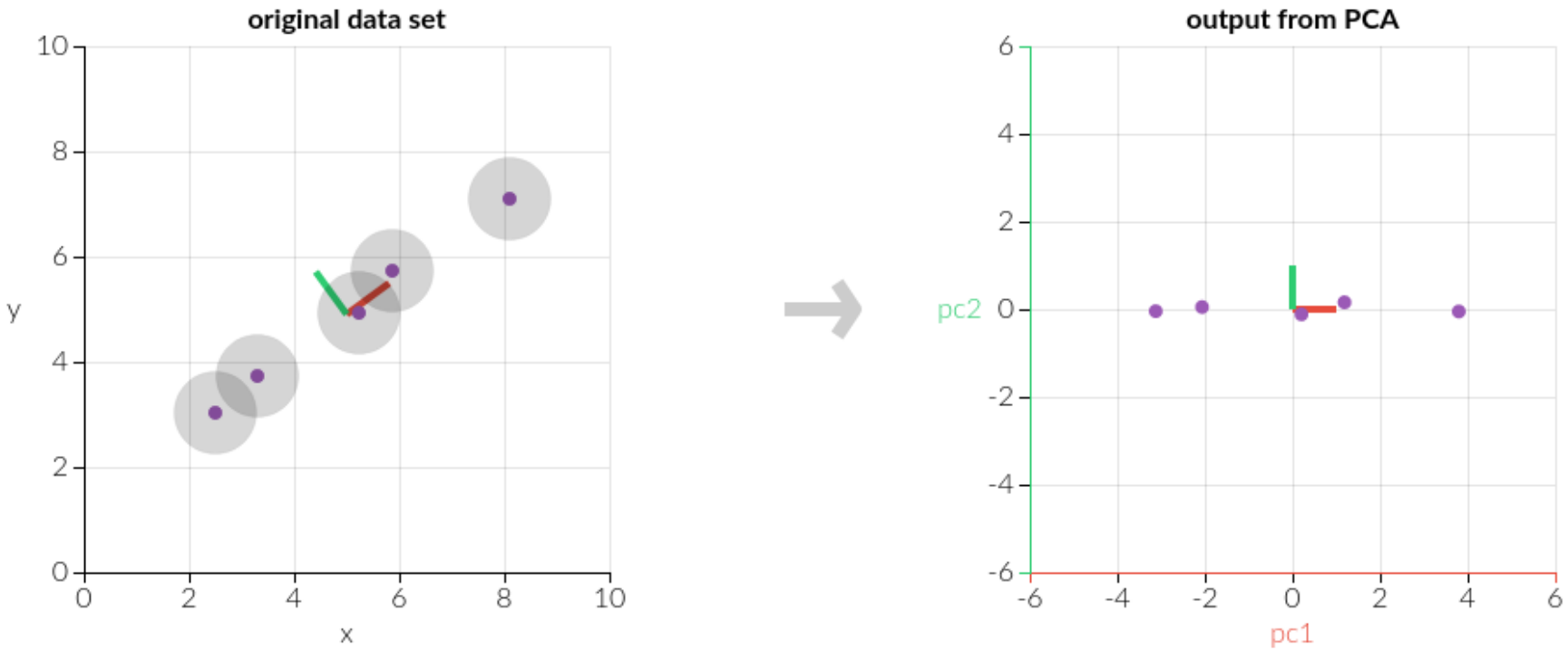
Dimensionality reduction using PCA

• the vast majority of variance in high dimensional data sets often captured by a small number of principal components



• dimensionality reduction achieved by using only a few of the principal components - retains most of the information in the original dataset

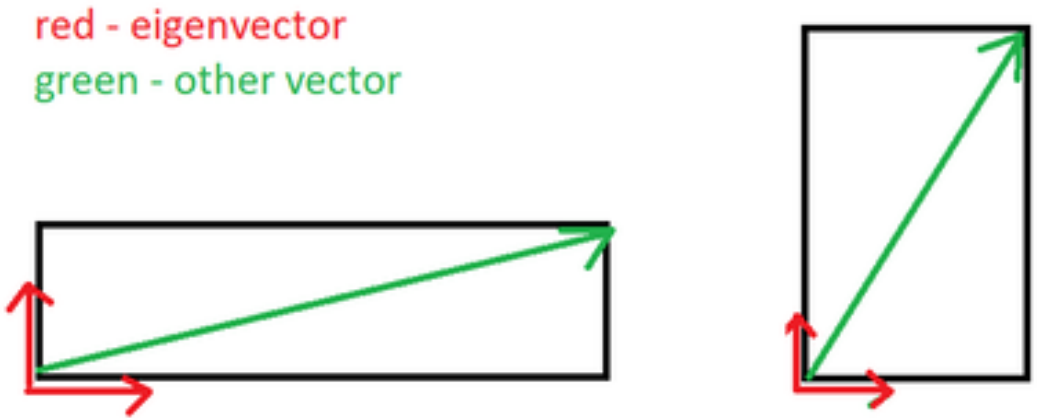
PCA: Example


original data set
y
x
0
2
4
6
8
10
output from PCA
pc2
pc1
-6
-4
-2
0
2
4
6

Eigenvalues and eigenvectors

Eigenvector: non-zero linearly independent vectors that do not change direction when a matrix transformation is applied

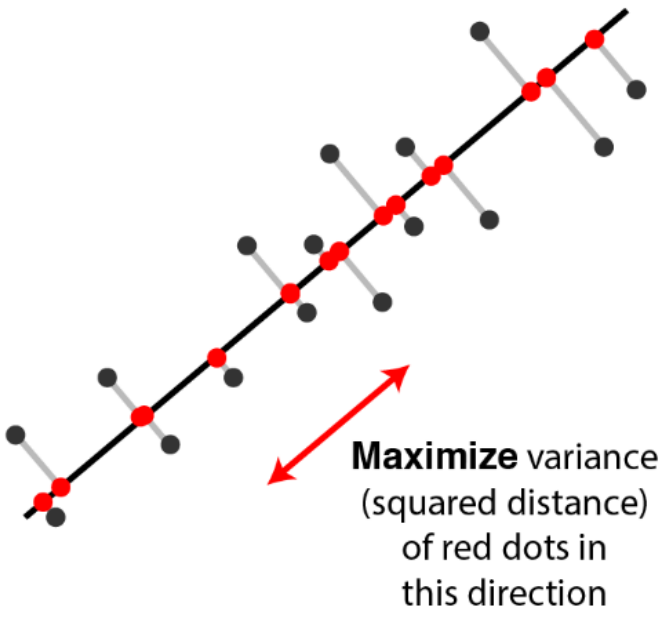
Eigenvalue: the factor by which the eigenvector is scaled during the transformation (denoted by λ)

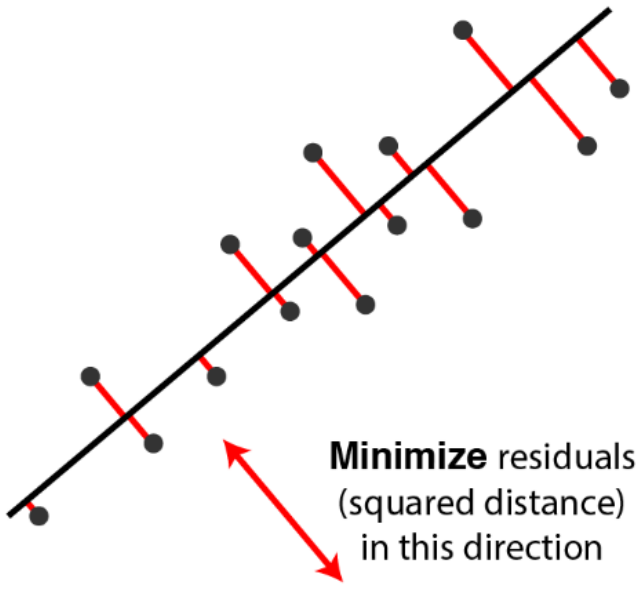


Eigenvectors and principal components

- the eigenvectors of the covariance matrix point in the direction of the largest variance
- the larger the eigenvalue, the more of the variance explained
- the eigenvector with the largest eigenvalue corresponds to the first principal component
- the eigenvector with the second-largest eigenvalue corresponds to the second principal component, etc

Two equivalent views of PCA





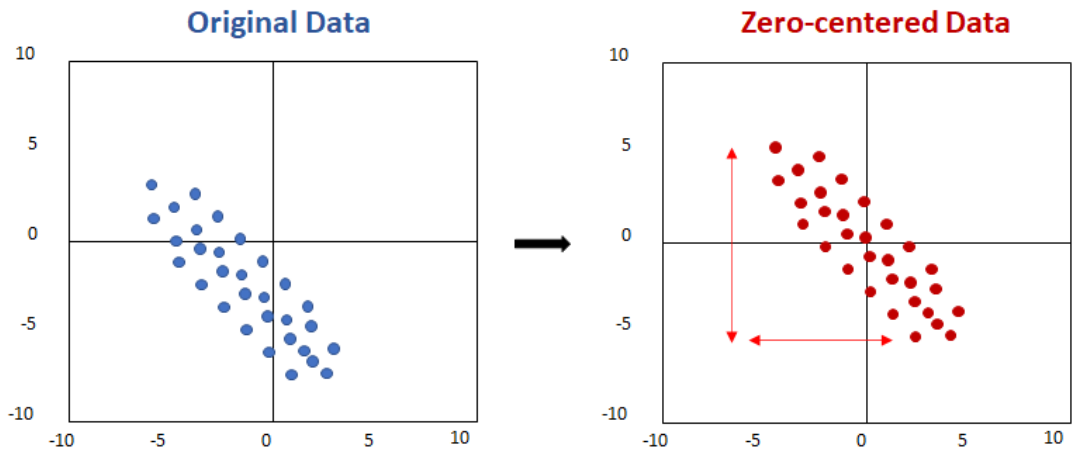
Steps in PCA

1. Center the data at zero mean

- achieved by subtracting the mean of each feature

$$x_{new} = x - \mu \quad \text{where}$$

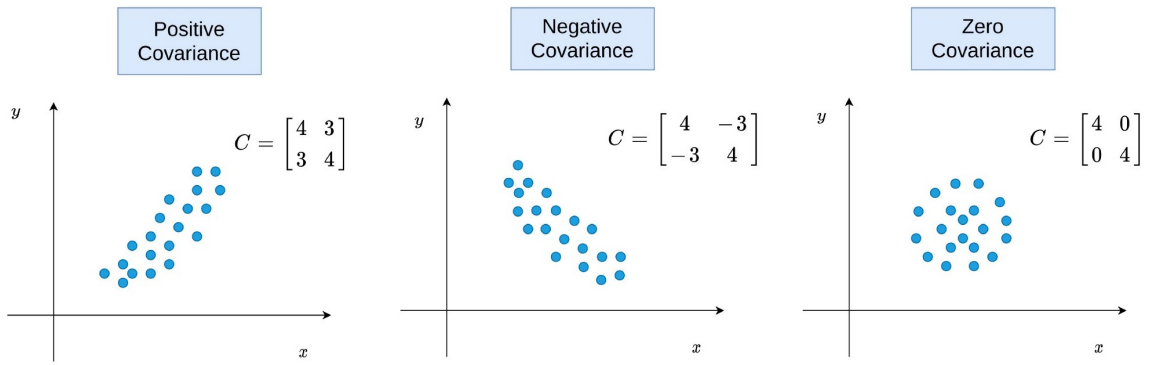
x_{new}: standardized value
x: original value
μ: mean value of feature



Steps in PCA

2. Compute the covariance matrix

- `covariance:` measure of the joint variability of two random variables: $cov(X, Y) = E[(X - E[X])(Y - E[Y])]$



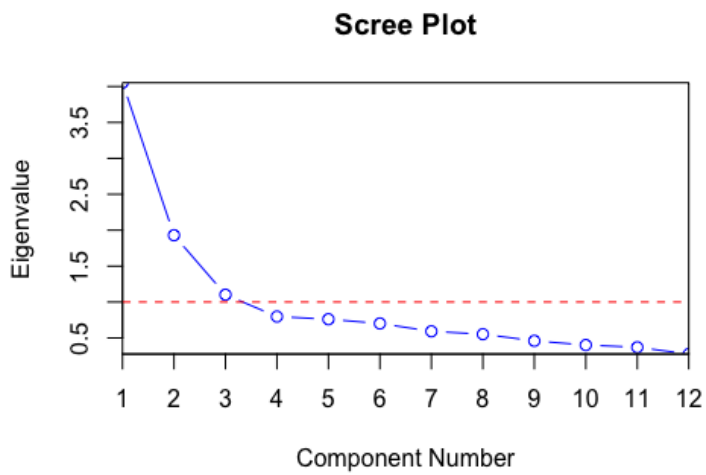
- `covariance matrix:` a square matrix giving the covariance between each pair of variables of a given set of observations

$$\begin{bmatrix} cov(x_1, x_2) & \cdots & cov(x_1, x_n) \\ \vdots & \ddots & \vdots \\ cov(x_n, x_1) & \cdots & cov(x_n, x_n) \end{bmatrix}$$

Steps in PCA

3. **Calculate the eigenvalues and eigenvectors of covariance matrix**

4. **Choosing the principal components**
   - sort the eigenvectors corresponding to their respective eigenvalues
   - the eigenvector with the largest eigenvalue corresponds to the first principal component and so on
   - pick the k eigenvectors corresponding to the largest k eigenvalues, with $k < m$ where m - number of features in the original dataset



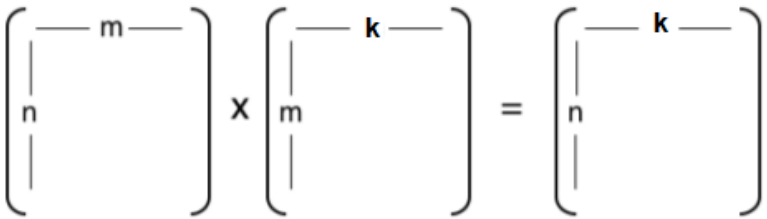
Steps in PCA

5. Deriving the new dataset

- create a **feature matrix** with the chosen eigenvectors as columns

$$\text{feature-matrix} = \begin{pmatrix} e_1 & e_2 & \cdots e_k \end{pmatrix}$$

- transformed-data = original-dataset $*$ feature-matrix

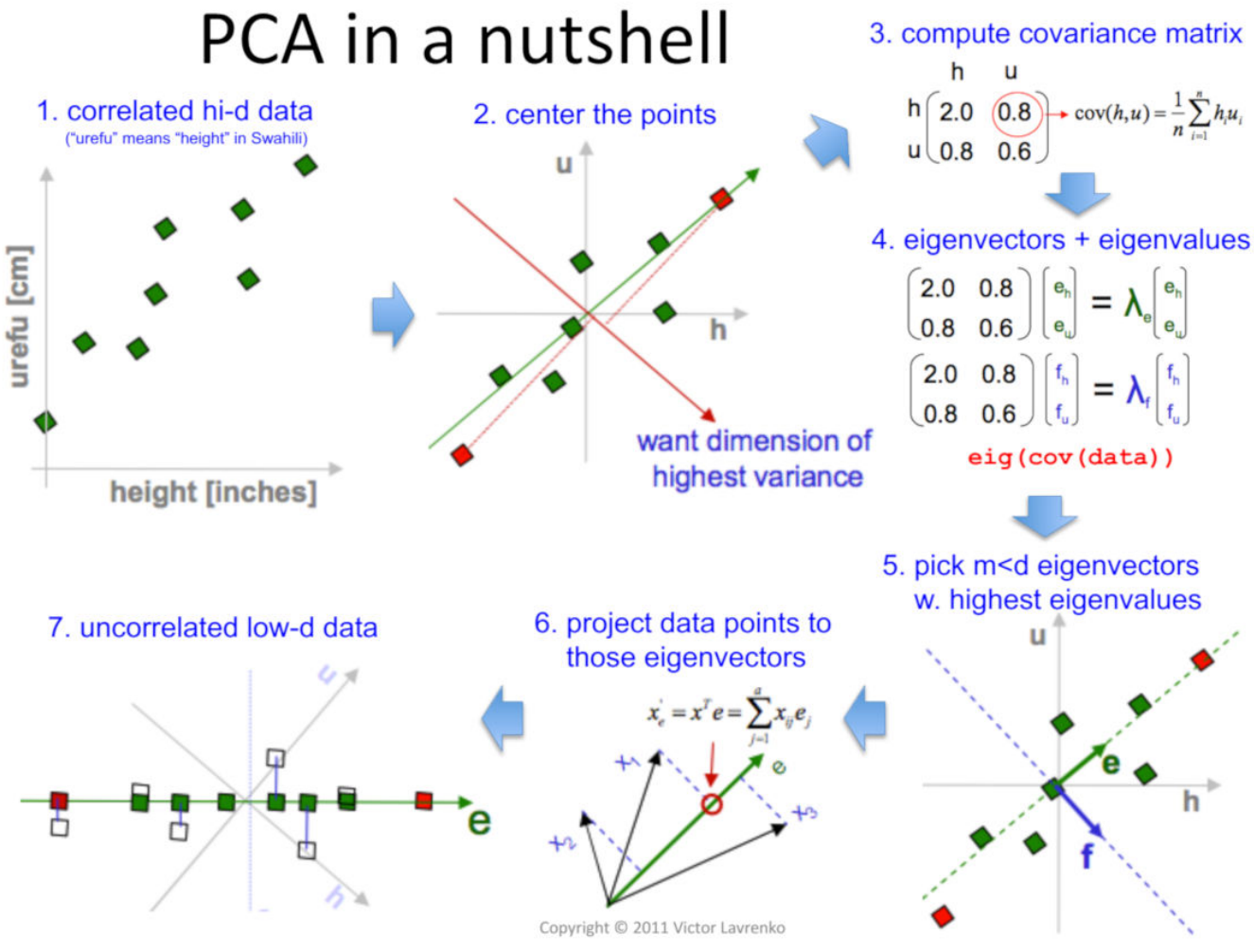


- The transformed data has dimension $n \; x \; k$

Reconstructing the original data

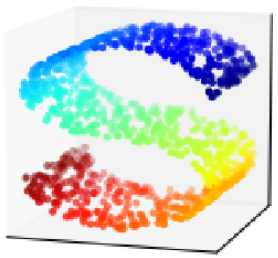
$$\text{reconstructed-data} = (\text{transformed-data} * \text{feature-matrix}^T) + \text{mean-vector}$$

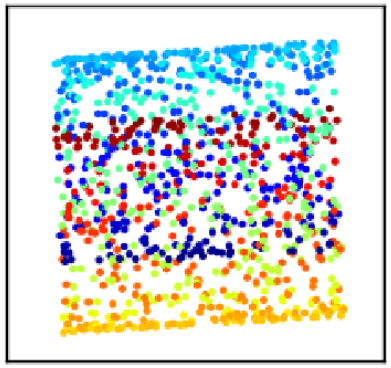
PCA in a nutshell

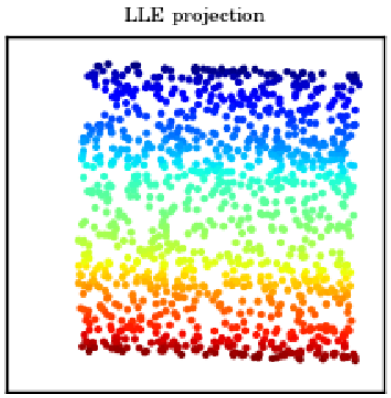


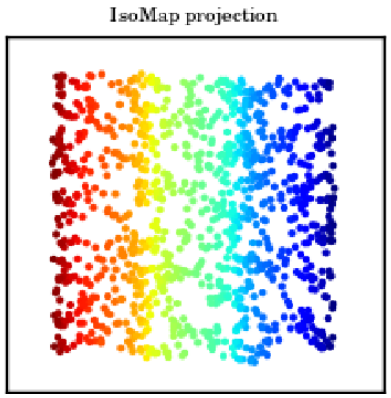
Limitations of PCA

- cannot capture nonlinear relationships in data









References

1. http://www.mit.edu/~9.54/fall14/slides/Class13.pdf
2. https://developers.google.com/machine-learning/clustering/algorithm/advantages-disadvantages
3. https://programmathically.com/principal-components-analysis-explained-for-dummies/
4. http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf

Thanks Google for the pictures!