AARHUS
UNIVERSITET

Optimization and Data Analytics

Alexandros Iosifidis
@
Aarhus University, Department of Engineering

Similarity measures

Similarity measures defined on sample pairs are important for two reasons:

- It is natural to assume that two similar samples share the same properties,
  e.g. they belong to the same class, the same cluster, etc

- They can be used in order to define optimization criteria for defining the parameters of a model

Similarity measures

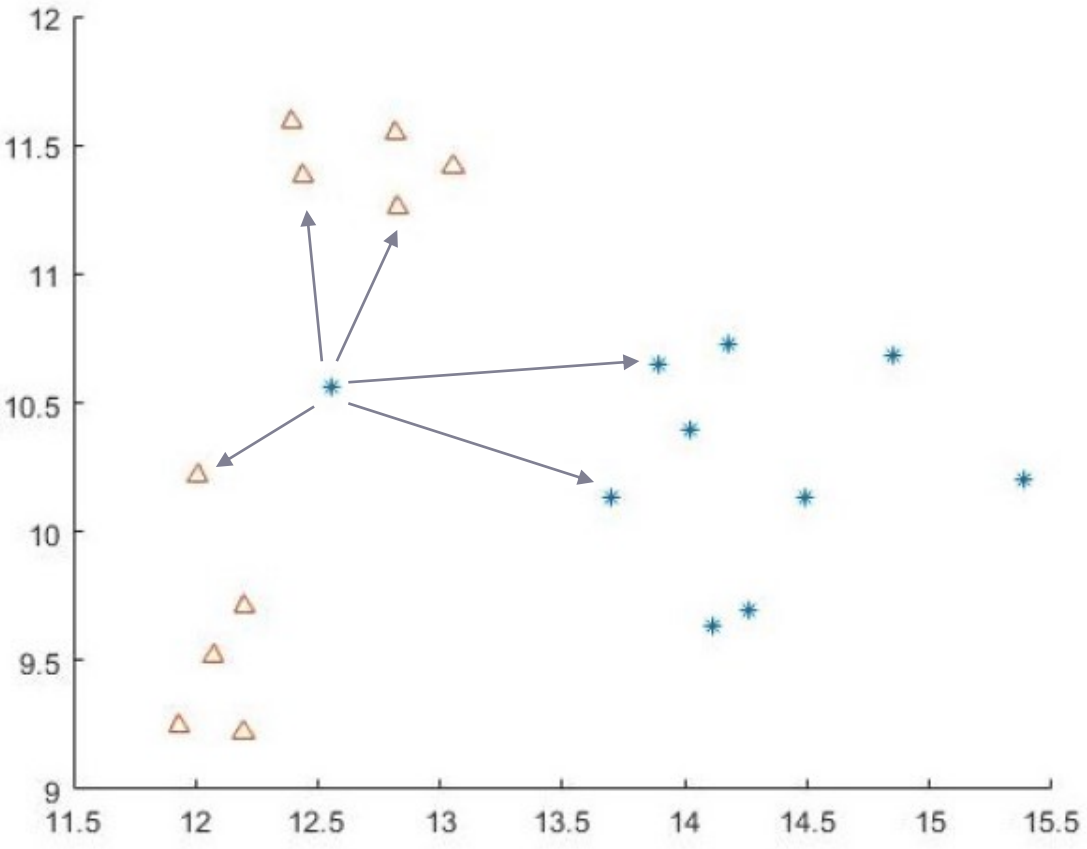
Similarity measures defined on sample pairs are important for two reasons:

- It is natural to assume that two similar samples share the same properties,
  e.g. they belong to the same class, the same cluster, etc

- They can be used in order to define optimization criteria for defining the parameters of a model

However, we should be careful on how to use them!

Similarity measures



Distance-based similarity measures

A natural way to define a similarity between two vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ is based on their distance $d(\mathbf{x}_i,\mathbf{x}_j)$:

$$s(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sigma}{d(\mathbf{x}_i, \mathbf{x}_j)}$$

$$s(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{d(\mathbf{x}_i, \mathbf{x}_j)}{\sigma}}$$

In the above, any distance function between vector-pairs can be used.

Distance-based similarity measures

Distance functions:

Euclidean distance:

$$d_E(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2 = \sqrt{\mathbf{x}_i^T\mathbf{x}_i + \mathbf{x}_j^T\mathbf{x}_j - 2\mathbf{x}_i^T\mathbf{x}_j} = \sqrt{\sum_{d=1}^{D}(x_{id} - x_{jd})^2}$$

Manhatan distance:

$$d_M(\mathbf{x}_i, \mathbf{x}_j) = \sum_{d=1}^{D} |x_{id} - x_{jd}|$$

Minkowski distance:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \left(\sum_{d=1}^{D} |x_{id} - x_{jd}|^q\right)^{\frac{1}{q}}$$

What happens if q = 2 and if q = 1?

Similarity measures

There are also similarity measures that are not defined using distance functions.

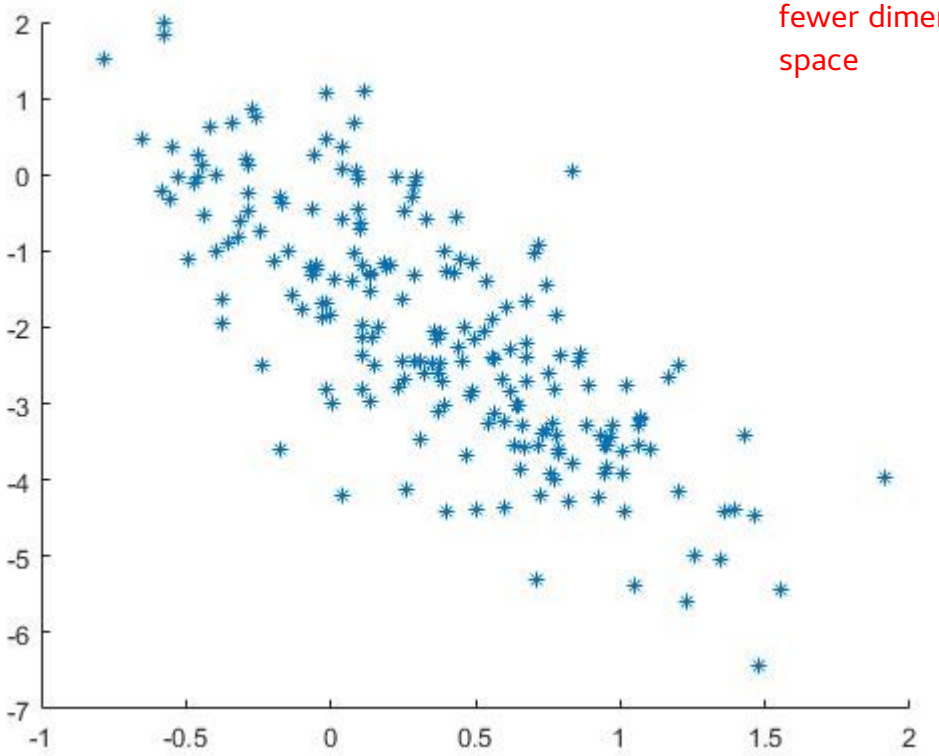
For example a similarity function based on the angle between the two vectors is given by:

$$s(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{x}_i^T \mathbf{x}_j}{\|\mathbf{x}_i\|_2 \|\mathbf{x}_j\|_2}$$

Principal Component Analysis

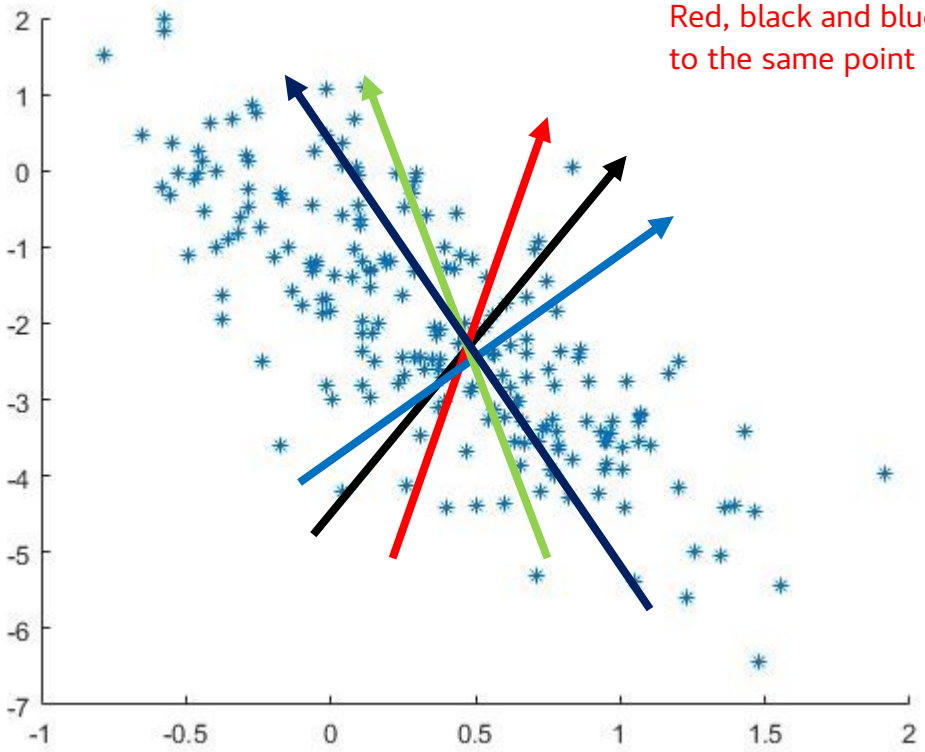
If we want to map the samples in the plot in one dimension, which one would you choose?

subspace= a space that has fewer dimensions than the original space



Principal Component Analysis

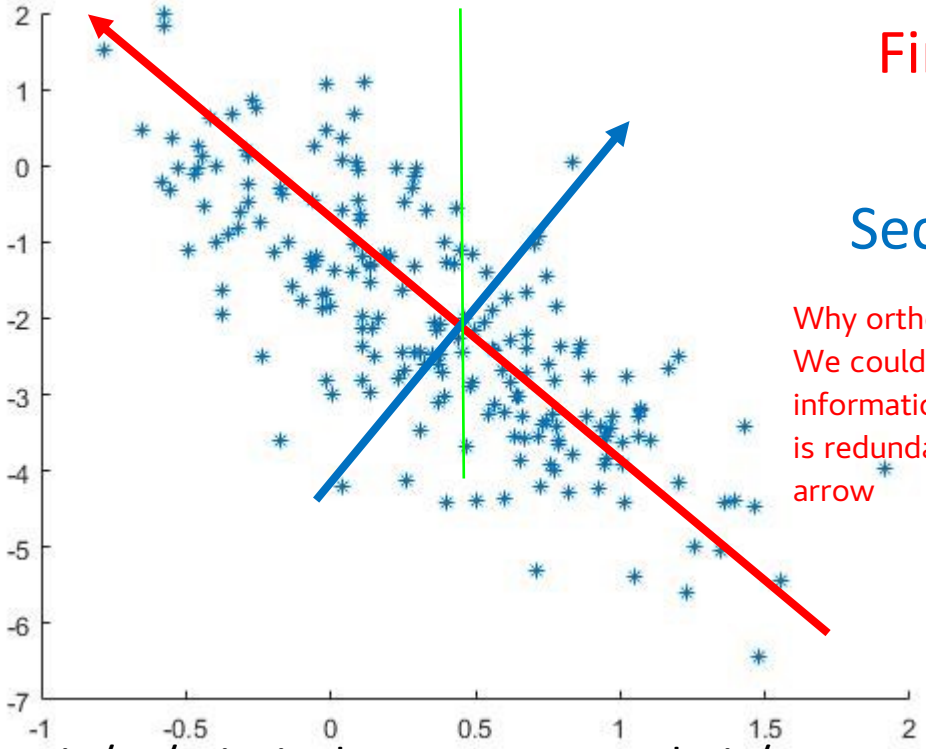
If we want to map the samples in the plot in one dimension, which one would you choose?



Red, black and blue maps many points in R^D to the same point in R^d

Principal Component Analysis

If we want to map the samples in the plot in one dimension, which one would you choose?



First principal axis

Second principal axis

Why orthogonality constraint?
We could use the green line but some of the information regarding the first dimension is redundant, so it is better to stick with the blue arrow

Demo: http://setosa.io/ev/principal-component-analysis/

Principal Component Analysis

How to determine (calculate from data) the two principal axes?

1. We define the data representations

W can select with random values

$$\mathbf{y}_i = \mathbf{W}^T \mathbf{x}_i$$

Principal Component Analysis

How to determine (calculate from data) the two principal axes?

1. We define the data representations

$$\mathbf{y}_i = \mathbf{W}^T \mathbf{x}_i$$

2. We define the variance using the representations $\mathbf{y}_i$, i=1,…,N

This is expresses our objective i.e., maximise the variance

$$\mathbf{S}_T = \sum_{i=1}^{N} (\mathbf{y}_i - \mathbf{m})(\mathbf{y}_i - \mathbf{m})^T = \sum_{i=1}^{N} \bar{\mathbf{y}}_i \bar{\mathbf{y}}_i^T = \bar{\mathbf{Y}} \bar{\mathbf{Y}}^T$$

The expression cannot be optimised because we don't know what y_i are

Principal Component Analysis

How to determine (calculate from data) the two principal axes?

1. We define the data representations

$$\mathbf{y}_i = \mathbf{W}^T \mathbf{x}_i$$

2. We define the variance using the representations $\mathbf{y}_i$, i=1,…,N

Total scatter matrix: scatter matrix for the entire dataset.

$$\mathbf{S}_T = \sum_{i=1}^{N} (\mathbf{y}_i - \mathbf{m})(\mathbf{y}_i - \mathbf{m})^T = \sum_{i=1}^{N} \bar{\mathbf{y}}_i \bar{\mathbf{y}}_i^T = \bar{\mathbf{Y}} \bar{\mathbf{Y}}^T$$

3. We express ST as a function of **W**

$$\begin{aligned} \mathbf{S}_T &= \sum_{i=1}^{N} \left[\mathbf{W}^T(\mathbf{x}_i - \boldsymbol{\mu})\right] \left[\mathbf{W}^T(\mathbf{x}_i - \boldsymbol{\mu})\right]^T = \sum_{i=1}^{N} \mathbf{W}^T \bar{\mathbf{x}}_i \bar{\mathbf{x}}_i^T \mathbf{W} \\ &= \mathbf{W}^T \bar{\mathbf{X}} \bar{\mathbf{X}}^T \mathbf{W} = \mathbf{W}^T \tilde{\mathbf{S}}_T \mathbf{W}, \end{aligned}$$

\tilde S_T is defined on X whereas S_T is a function of y

The scatter matrix: S_T. The diagonal element contains the variance for each dimensions.
The non-diagonal elements are the co-variance; variance between the pairs of dimensions

Principal Component Analysis

We will obtain the data representations yi that correspond to the maximal possible variance, if we optimize for

We want to maximise the trace because this describe the variance.
The diagonal elements are the variances of each dimensions of the scatter matrix

$$\begin{aligned} \mathbf{W}^* &= \arg\max \; Tr\left(\mathbf{W}^T \tilde{\mathbf{S}}_T \mathbf{W}\right) \\ subject\ to &: \mathbf{W}^T\mathbf{W} = \mathbf{I}, \end{aligned}$$

Why do we need the constraint $\mathbf{W}^T\mathbf{W} = \mathbf{I}$?

Because we want the principle axes to be orthogonal

Principal Component Analysis

We will obtain the data representations yi that correspond to the maximal possible variance, if we optimize for

$$\begin{aligned} \mathbf{W}^* &= \arg\max \; Tr\left(\mathbf{W}^T \tilde{\mathbf{S}}_T \mathbf{W}\right) \\ subject\ to &: \mathbf{W}^T\mathbf{W} = \mathbf{I}, \end{aligned}$$

How can you automatically select the number dimensions?
- Sum all the eigenvalues
- Calculate lamda_i / sum i.e. divide each eigenvalue with the sum
- Construct a vector
- Compute the cumulative sum and define a threshold like 90%. We only keep eigenvalues that are above the threshold.

W is obtained by applying eigenanalysis to the matrix $\mathbf{S}_T$

The number of eigenvalues after applying is the same as the rank of the matrix.
The eigendecomposition is A = W Lambda W^T, lambda is a diagonal matrix, Sigma contains the eigenvectors.
If one of the eigenvalues is zero, then the corresponding eigenvector will be zero vector.
One of the properties of Eigenvectors is that their L2 norm is 1.

Principal Component Analysis

It can be shown (Exercises of the course) that the problem

$$\begin{aligned} \mathbf{W}^* &= \arg\max \; Tr\left(\mathbf{W}^T \tilde{\mathbf{S}}_T \mathbf{W}\right) \\ subject\ to &: \mathbf{W}^T\mathbf{W} = \mathbf{I}, \end{aligned}$$

WY is the reconstruction error. W is a diagonal matrix so it will always invert

Is equivalent to solving for

Minimising the reconstruction error.

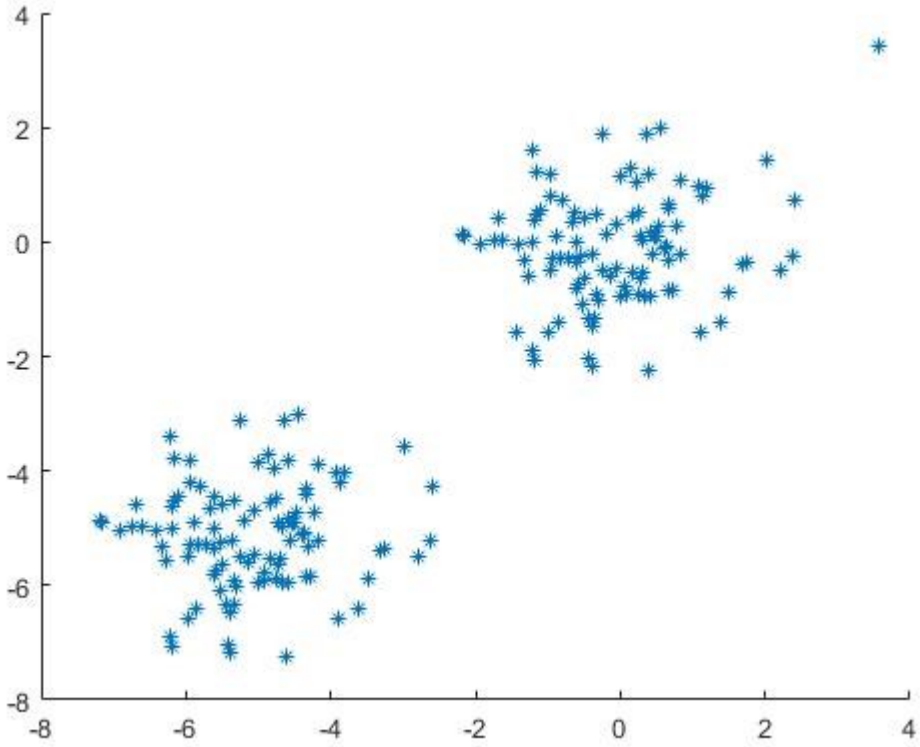
$$\begin{aligned} \mathbf{W}^* &= \arg\min \|\mathbf{W}\bar{\mathbf{Y}} - \bar{\mathbf{X}}\|_F^2 \\ subject\ to &: \mathbf{W}^T\mathbf{W} = \mathbf{I}, \end{aligned}$$

What does the second optimization problem indicate?

Data clustering

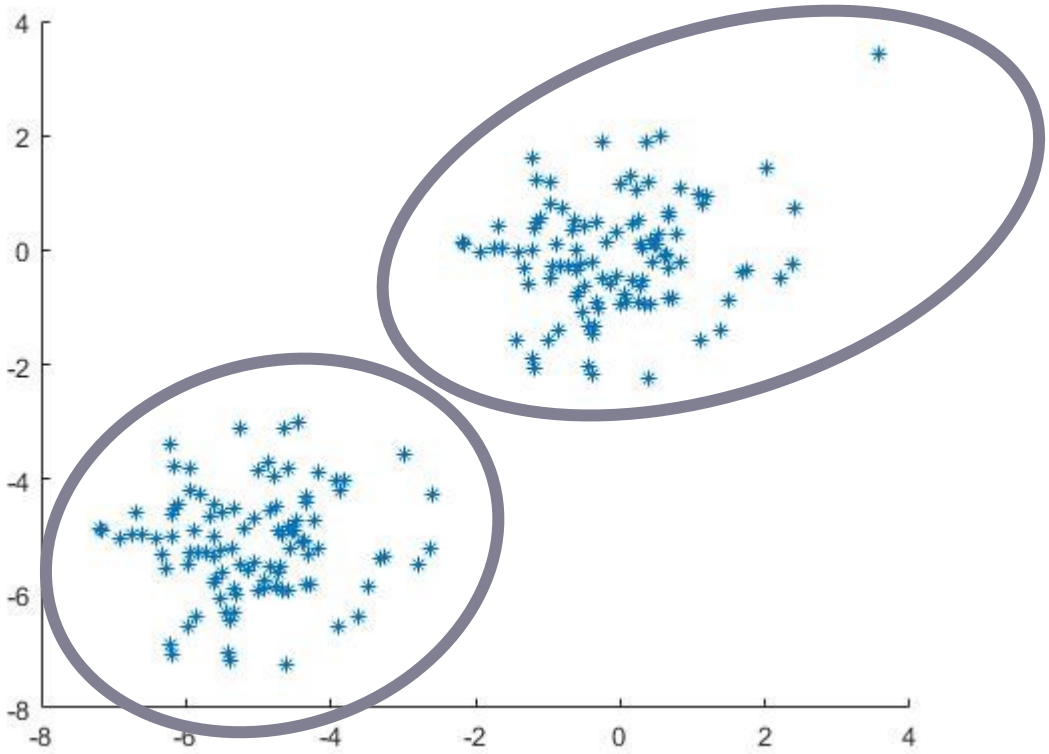
When the centroids do not change then the algorithm has converged

Given a set of (unlabeled) data $\mathbf{x}_i$, i=1,…,N we usually want to define groups of data, we usually want to define groups (or clusters)



Data clustering

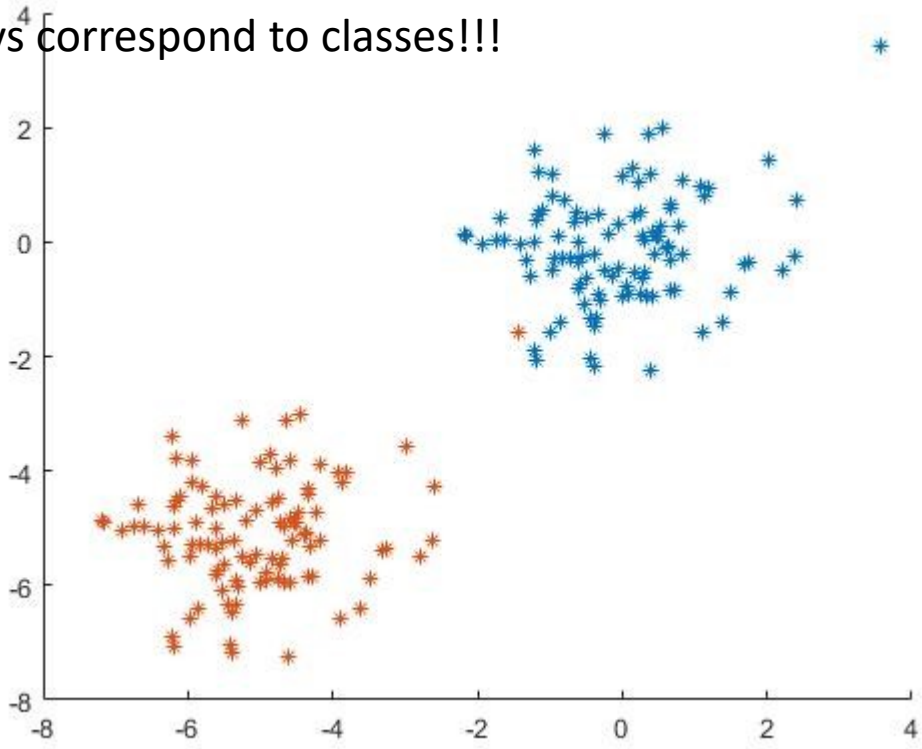
Given a set of (unlabeled) data $\mathbf{x}_i$, i=1,…,N we usually want to define groups of data, we usually want to define groups (or clusters)



Data clustering

Given a set of (unlabeled) data $\mathbf{x}_i$, i=1,…,N we usually want to define groups of data, we usually want to define groups (or clusters)

Clusters don’t always correspond to classes!!!



Data clustering

Given the hyperparameter, k, we want to minimise the distance between each point in the cluster and its centroid

In order to define groups, we solve the following optimization problem

x_i^(k) is the i'th vector in the k'th cluster

$$\mathcal{J}_{SSE} = \sum_{k=1}^{K} \sum_{i=1}^{N_k} \|\mathbf{x}_i^{(k)} - \boldsymbol{\mu}_k\|_2^2 = \sum_{k=1}^{K} \mathcal{J}_k$$

where

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i=1}^{N_k} \mathbf{x}_i^{(k)}$$

x^T x = Tr(x x^T)

Since optimisation J_SSE consists of two steps, we cannot optimise it at once, so we have to apply an iterative approach.

Here we look at if we move some the data point from one cluster to another affect the optimisation function.
Once we move a point from one cluster to another, then we need to update the centroids according the added/removed datapoint since the centroid describe the cluster

K-Means algorithm

Let us assume that (at some iteration) we want to check if we need to change the cluster for a sample, i.e. to move it from cluster k to cluster m.

After the change, the mean vectors of clusters k and m will change as follows

$$\boldsymbol{\mu}_m^* = \boldsymbol{\mu}_m + \frac{N_m}{N_m + 1}(\mathbf{x}_i^{(k)} - \boldsymbol{\mu}_m)$$

$$\boldsymbol{\mu}_k^* = \boldsymbol{\mu}_k - \frac{N_k}{N_k - 1}(\mathbf{x}_i^{(k)} - \boldsymbol{\mu}_k)$$

K-Means algorithm

Let us assume that (at some iteration) we want to check if we need to change the cluster for a sample, i.e. to move it from cluster k to cluster m.

The change will affect the variance of clusters m and k as follows

$$\begin{aligned}
\mathcal{J}_m^* &= \sum_{j=1}^{N_m} \|\mathbf{x}_j^{(m)} - \boldsymbol{\mu}_m^*\|_2^2 + \|\mathbf{x}_i^{(k)} - \boldsymbol{\mu}_m^*\|_2^2 \\
&= \left(\sum_{j=1}^{N_m} \|\mathbf{x}_j^{(m)} - \boldsymbol{\mu}_m - \frac{1}{N_m+1}(\mathbf{x}_i^{(k)} - \boldsymbol{\mu}_m)\|_2^2 \right) + \|\frac{N_m}{N_m+1}(\mathbf{x}_i^{(k)} - \boldsymbol{\mu}_m)\|_2^2 \\
&= \mathcal{J}_m + \frac{N_m}{N_m+1}\|(\mathbf{x}_i^{(k)} - \boldsymbol{\mu}_m)\|_2^2
\end{aligned}$$

$$\mathcal{J}_k^* = \mathcal{J}_k - \frac{N_k}{N_k-1}\|(\mathbf{x}_i^{(k)} - \boldsymbol{\mu}_k)\|_2^2.$$

K-Means algorithm

Let us assume that (at some iteration) we want to check if we need to change the cluster for a sample, i.e. to move it from cluster k to cluster m.

Thus, if:

$$\frac{N_k}{N_k - 1}\|(\mathbf{x}_i^{(k)} - \boldsymbol{\mu}_k)\|_2^2 > \frac{N_m}{N_m + 1}\|(\mathbf{x}_i^{(k)} - \boldsymbol{\mu}_m)\|_2^2,$$

which (for the case where cluster are balanced, i.e. they are formed by the same/similar number of vectors) is the case when $\mathbf{x}_i^{(k)}$ is closer to $\boldsymbol{\mu}_m$ than to $\boldsymbol{\mu}_k$, i.e. when:

$$\|\mathbf{x}_i^{(k)} - \boldsymbol{\mu}_m\|_2^2 < \|\mathbf{x}_i^{(k)} - \boldsymbol{\mu}_k\|_2^2,$$

This is because we have take out the ratio in front because here we know that N_m = N_k

making the change will lead to a lower value for $\mathcal{J}_{SSE}$. Since our objective is to minimize $\mathcal{J}_{SSE}$, we choose to make the change.

K-Means algorithm

Algorithm 1: Iterative minimization of SSE for clustering

Initialize $\boldsymbol{\mu}_k,\ k = 1, \ldots, K$
Assign each vector $\mathbf{x}_i,\ i = 1, \ldots, N$ to a cluster by:
$$l^* = \arg\min_l \|\mathbf{x}_i - \boldsymbol{\mu}_l\|_2^2$$

% Start iterative optimization
do randomly select a samples $\mathbf{x}_i^{(k)}$
  if $N_k \neq 1$ **then** calculate
    $r_k = \frac{N_k}{N_k - 1} \|(\mathbf{x}_i^{(k)} - \boldsymbol{\mu}_k)\|_2^2$
    $r_m = \frac{N_m}{N_m + 1} \|(\mathbf{x}_i^{(k)} - \boldsymbol{\mu}_m)\|_2^2,\ \ m \neq k$
    Calculate the minimum value $r_q = min\, r_m,\ m \neq k$
    if $r_q < r_k$ **then** transfer $\mathbf{x}_i^{(k)}$ to cluster q
    recompute $\boldsymbol{\mu}_k$, $\boldsymbol{\mu}_q$ and $\mathcal{J}_{SSE}$
until no change in $\mathcal{J}_{SSE}$ in is obtained after checking all N samples

K-Means algorithm

Usually, we apply the process in a 'batch' mode, i.e. we apply the following process

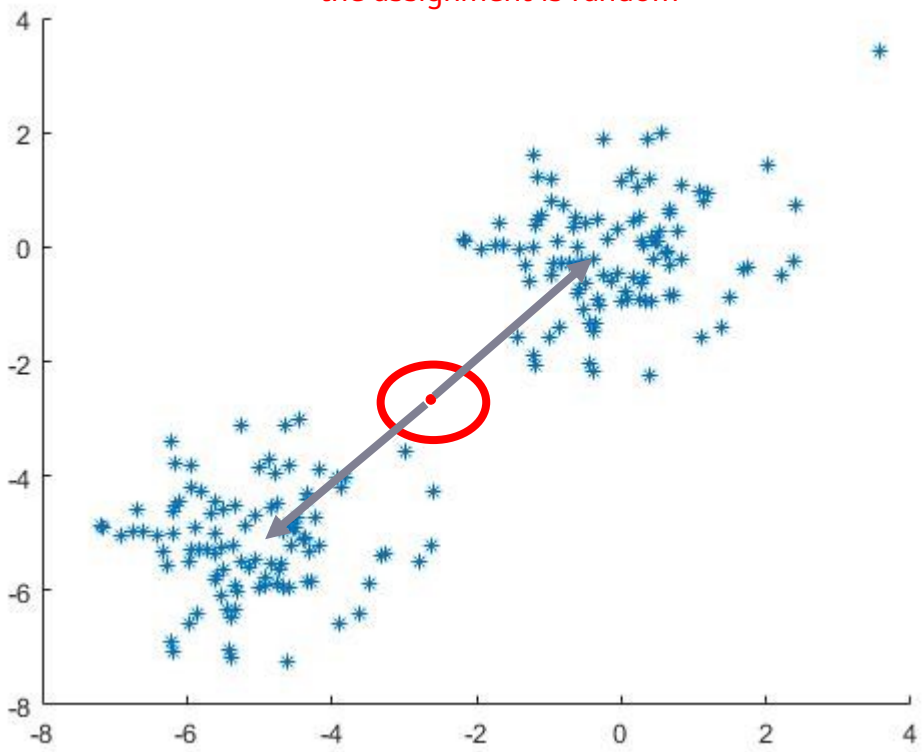
Algorithm 2: K-Means clustering

1: Initialize $\boldsymbol{\mu}_k,\ k = 1, \ldots, K$
2: **Do**
3: Assign all vectors $\mathbf{x}_i,\ i = 1, \ldots, N$ to a cluster by:
4: $l^* = \arg\min_l \|\mathbf{x}_i - \boldsymbol{\mu}_l\|_2^2$
5: Update the cluster mean vectors by:
6: $\boldsymbol{\mu}_k^* = \frac{1}{N_k} \sum_{\mathbf{x}_i \in D_k} \mathbf{x}_i,\ \ k = 1, \ldots, K$
7: **until** no change in $\boldsymbol{\mu}_k^*,\ k = 1, \ldots, K$

Demo: http://util.io/k-means

Fuzzy K-Means algorithm

In k-means we have hard assignmenet, each sample belongs to a specific cluster. The problem is when a sample is in the middle the assignment is random



K-Means algorithm

We define 'membership values' a_{ik} for each sample $\mathbf{x}_i$, so that $\sum_{k=1}^{K} a_{ik} = 1$

For example, we can use memberships in the form $a_{ik} = \frac{\|\mathbf{x}_i - \boldsymbol{\mu}_k\|_2^{-\gamma}}{\sum_{l=1}^{K} \|\mathbf{x}_i - \boldsymbol{\mu}_l\|_2^{-\gamma}}$

Gamma scales the distance between the sample and mean. If we set gamme to infinite number then we get the normal hard-assigning k-means. If gamma is low, then we get a more relaxed fuzzy k-means.

Then, the cluster vectors $\boldsymbol{\mu}_k$ are given by $\boldsymbol{\mu}_k = \sum_{i=1}^{N} a_{ik}\mathbf{x}_i, \;\; k = 1, \ldots, K$

and we optimize the following criterion

$$\mathcal{J}_{FKMeans} = \sum_{k=1}^{K} \sum_{i=1}^{N_k} a_{ik} \|\mathbf{x}_i^{(k)} - \boldsymbol{\mu}_k\|_2^2$$

K-Means algorithm

Algorithm 3: Fuzzy K-Means clustering

1: Initialize $\boldsymbol{\mu}_k,\ k = 1, \ldots, K$
2: **Do**
3: Calculate the membership vectors $\mathbf{a}_i,\ i = 1, \ldots, N$
4: Update the new cluster mean vectors $\boldsymbol{\mu}_k^*,\ k = 1, \ldots, K$
5: **until** no change in $\boldsymbol{\mu}_k^*,\ k = 1, \ldots, K$