Departamento de Eletrónica, Telecomunicações e Informática

LECTURE 8

UNSUPERVISED LEARNING
(K-MEANS CLUSTERING AND PCA)

Petia Georgieva
(petia@ua.pt)

Outline

Unsupervised learning

1. K-means clustering

2. Data dimensionality reduction
- data compression /data visualization

3. Principal Component Analysis (PCA)

SUPERVISED vs. UNSUPERVISED LEARNING

Supervised Learning - (given DATA + LABELS):
ML method is trained with labeled data to predict the labels of new examples (learning by labeled examples)

Matrix X	feature x_0	feature x_1	.....	feature x_n	Vector y - output (label)
Example 1	1	$x^{(1)}$		$x_n^{(1)}$	$y^{(1)}$
Example 2	1	$x^{(2)}$		$x_n^{(2)}$	$y^{(2)}$
	1				
				$x_n^{(i)}$	
Example m	1	$x^{(m)}$		$x_n^{(m)}$	$y^{(m)}$

Unsupervised Learning - given UNLABELED DATA
ML method to discover the data internal (statistical) structure

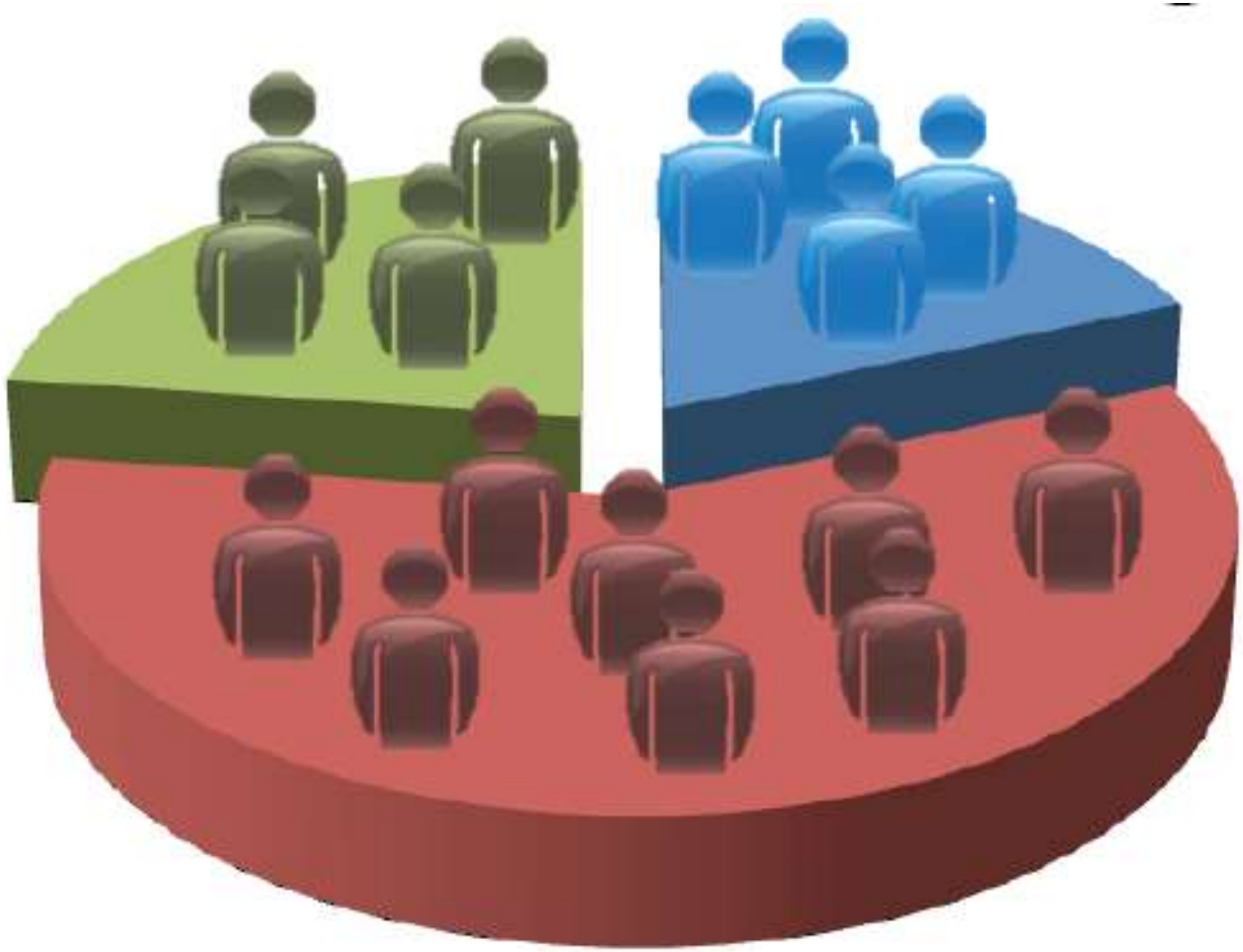
Matrix X	feature x_1	feature x_2	.....	feature x_n
Example 1	$x_1^{(1)}$	$x_2^{(1)}$		$x_n^{(1)}$
Example 2	$x_1^{(2)}$	$x_2^{(2)}$		$x_n^{(2)}$
...				
Example i	$x_1^{(i)}$	$x_2^{(i)}$		$x_n^{(i)}$
...				
...				
Example m	$x_1^{(m)}$	$x_2^{(m)}$		$x_n^{(m)}$

Semi-Supervised Learning – mixture of labeled and unlabeled data

Unsupervised learning -

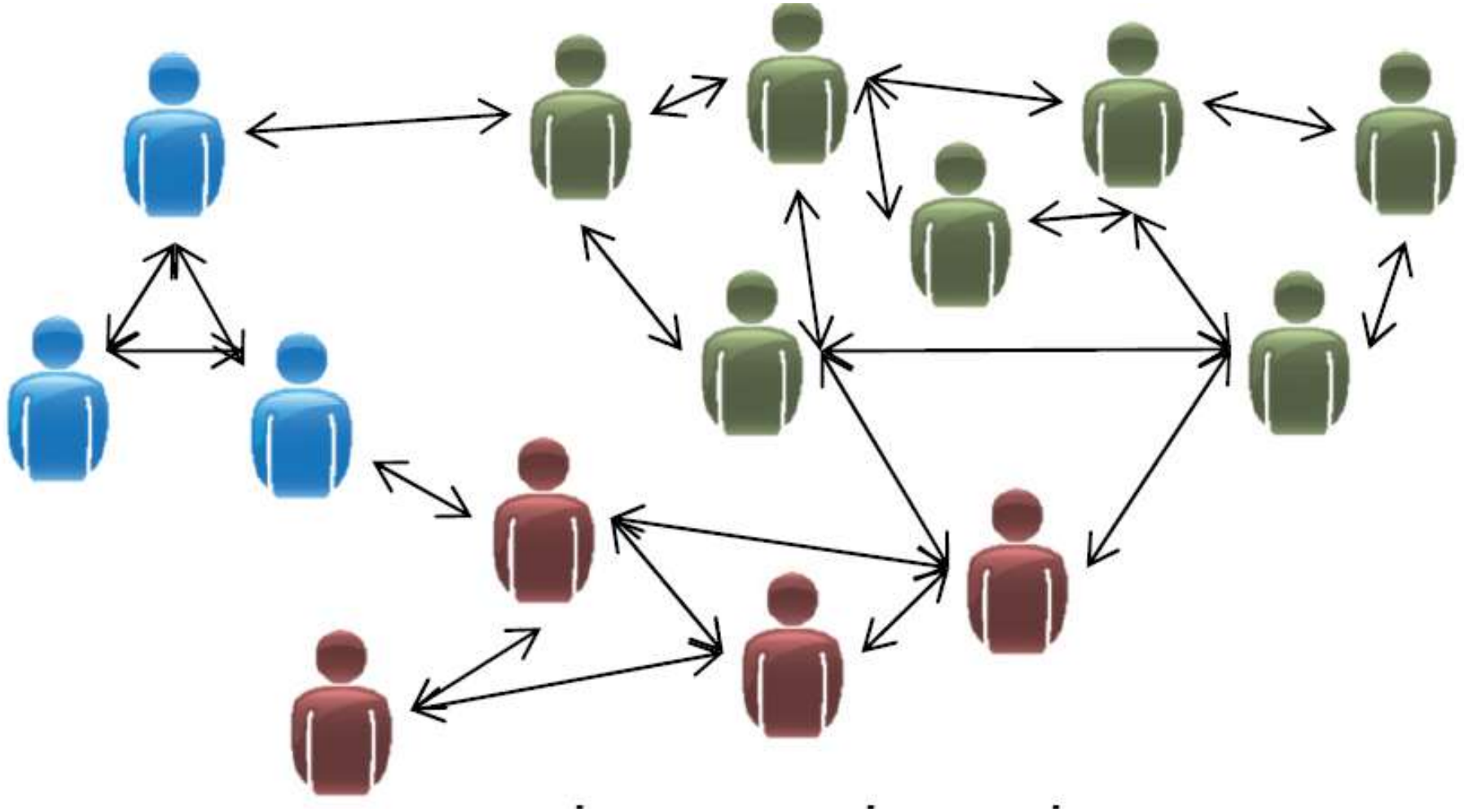
Market segmentation: data base of customers => division in target groups

Features: education, job, age, marital status, etc.

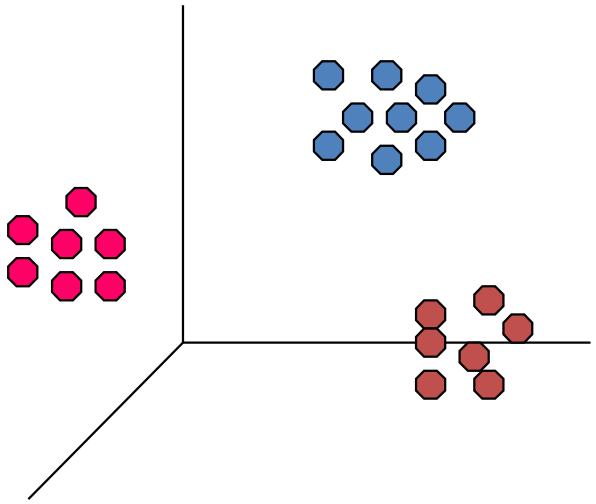


Unsupervised learning

Social network analysis: user grouping, group-specific advertising



Clustering intuition



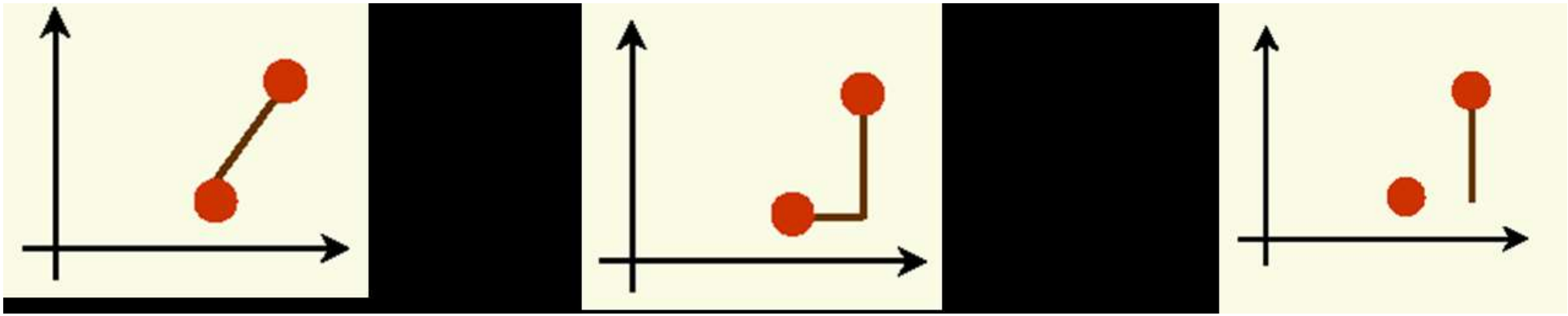
- Given a set of not labeled examples

- Find a relevant grouping of the examples into clusters such that:

  - Examples in the same cluster have **high similarity**
  - Examples from different clusters have **high dissimilarity**

Similarity measures –
Euclidian distance; Chebyshev distance; Manhattan distance

Distance (similarity) measures



Euclidian Distance (L2 norm)

$$d(p,q)=\sqrt{(x_p-x_q)^2+(y_p-y_q)^2}$$

Manhattan Distance (L1 norm)

$$d(p,q)=|x_p-x_q|+|y_p-y_q|$$

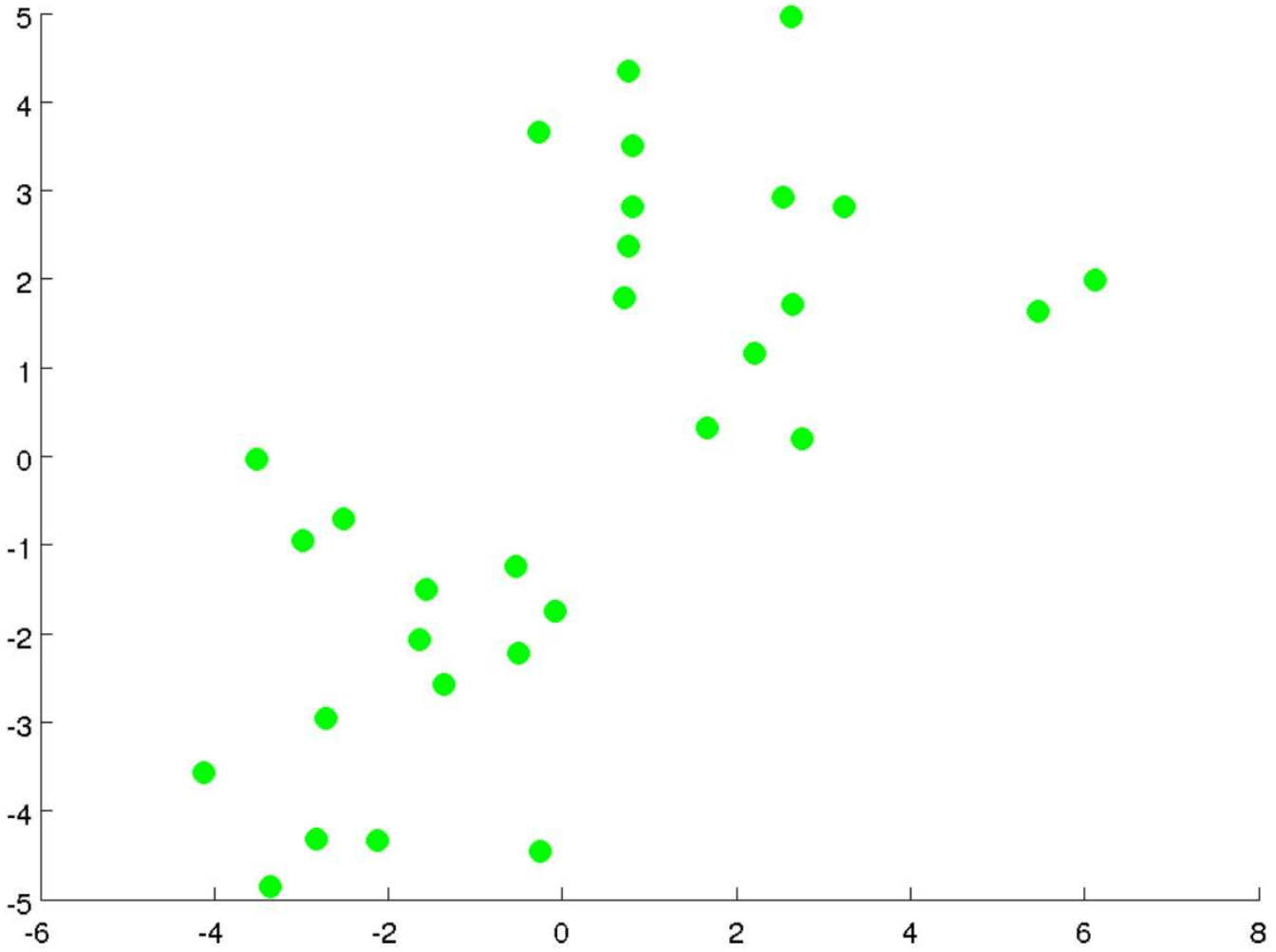
Chebyshev distance

$$d(p,q)=\max|(x_p-x_q),(y_p-y_q)|$$

K-means algorithm

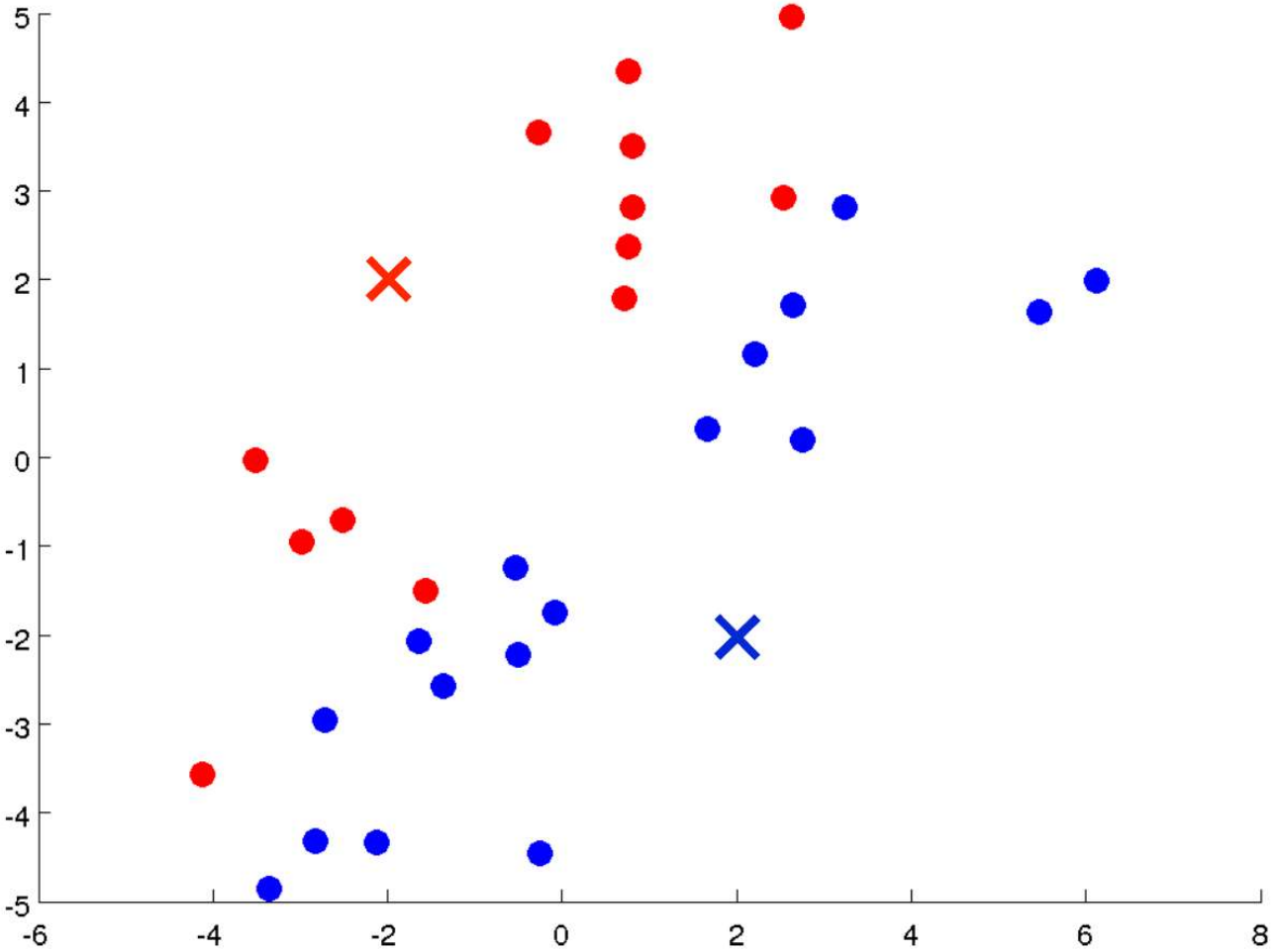
Given:

- **K – the number of clusters**
- **Training set - no labels**



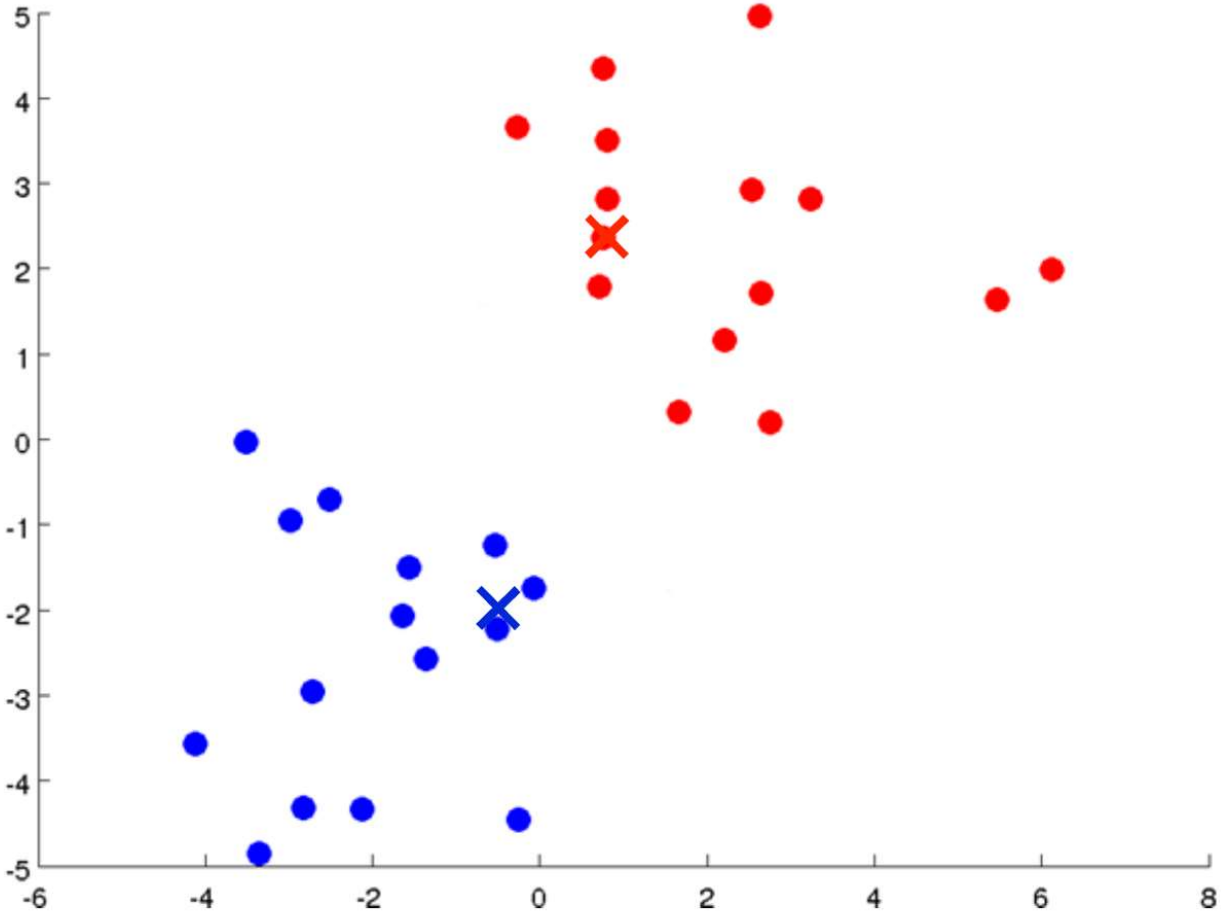
K-means algorithm

Randomly initialize K cluster centroids (e.g. K=2)
Assign data points to their closest centroid (Euclidian distance)



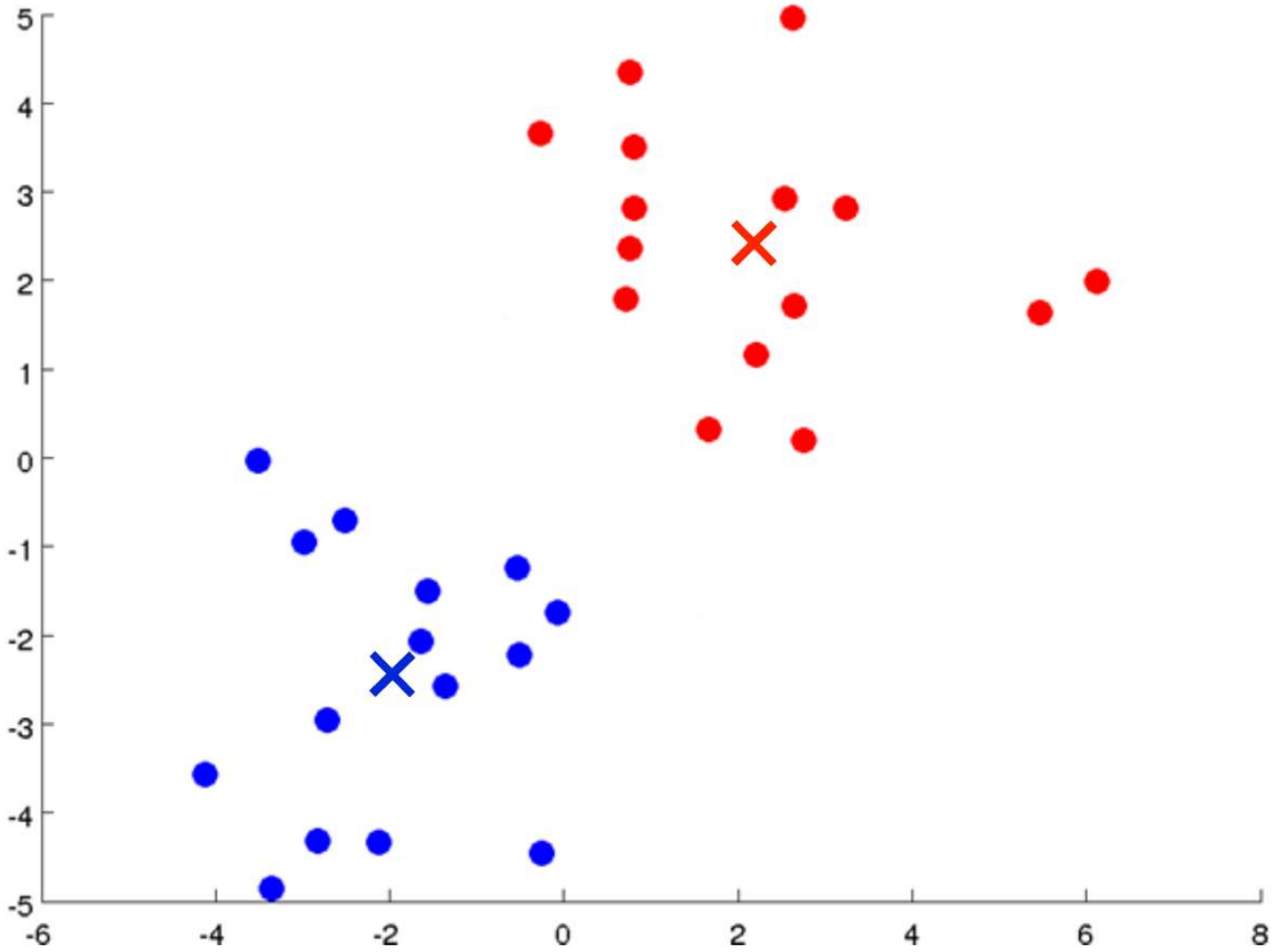
K-means algorithm

Compute new centroids = mean of the points assigned to that cluster.
Assign data points to the new closest centroid.



K-means algorithm

Repeat until convergence



K-means algorithms

Input:
- K (number of clusters)
- Training set (no labels)

Randomly initialize K cluster centroids $\mu_1, \mu_2, \ldots, \mu_K \in \mathbb{R}^n$

Repeat {

Cluster assignment => step

for i = 1 to m

$c^{(i)}$:= index (from 1 to K) of cluster centroid closest to $x^{(i)}$

Move centroid => step

for k = 1 to K

μ_k := average (mean) of points assigned to cluster k

}

K-means optimization objective (distortion = average distance)

$$J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K) = \frac{1}{m} \sum_{i=1}^{m} ||x^{(i)} - \mu_{c^{(i)}}||^2$$

$$\min_{\substack{c^{(1)},\ldots,c^{(m)}, \\ \mu_1, \ldots, \mu_K}} J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K)$$

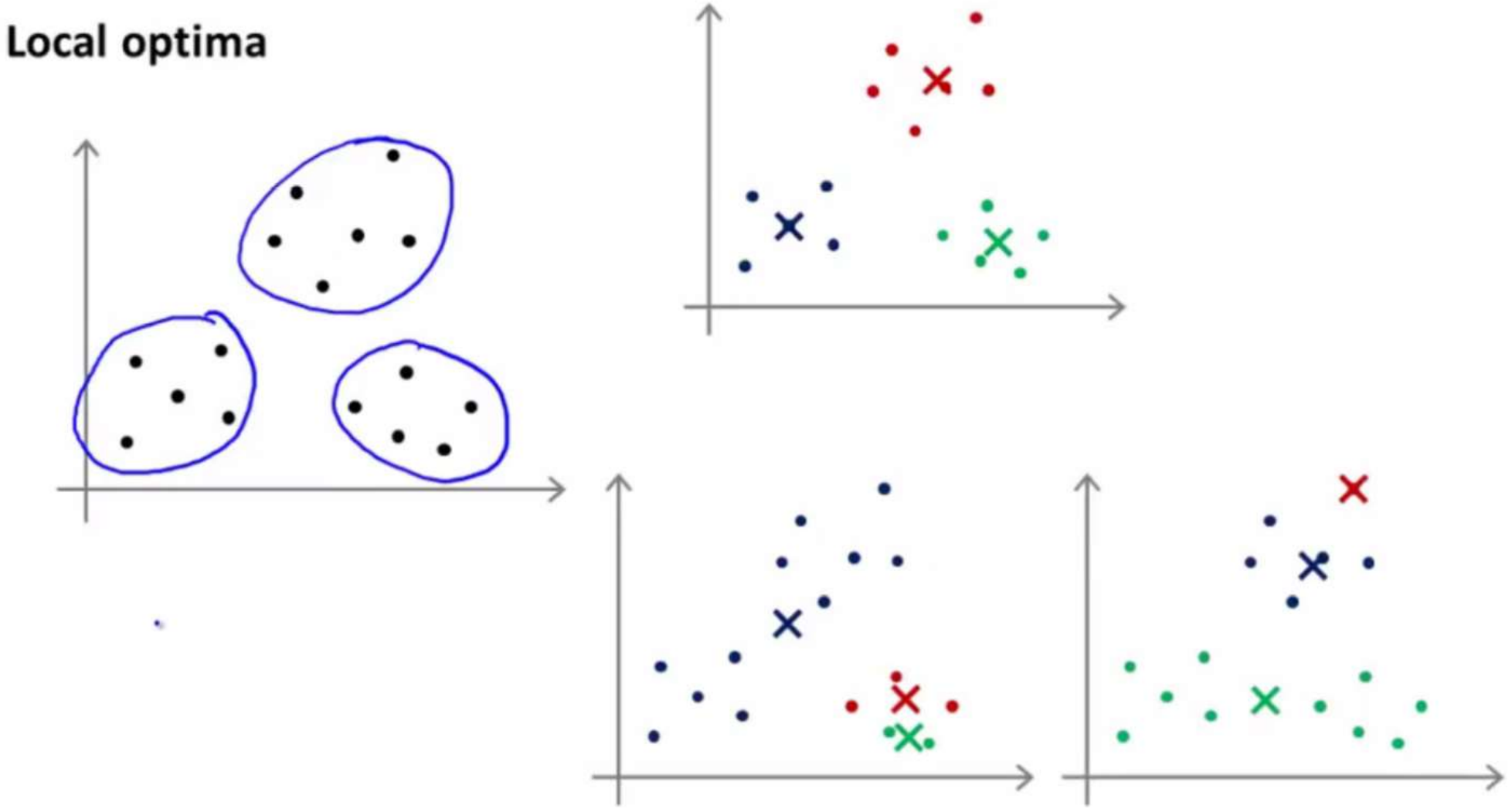
Stop K-means learning (different criteria):

- Achieved Max number of iterations
- J < some threshold
- No improvement of *J* between subsequent iterations

Single (Random) Initialization

Choose number of clusters K

Inicialize K cluster centroids = randomly picked K training examples



Repeat Random Initializations

For i = 1 to 100 {

Randomly initialize K-means.

Run K-means. Get $c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K$.

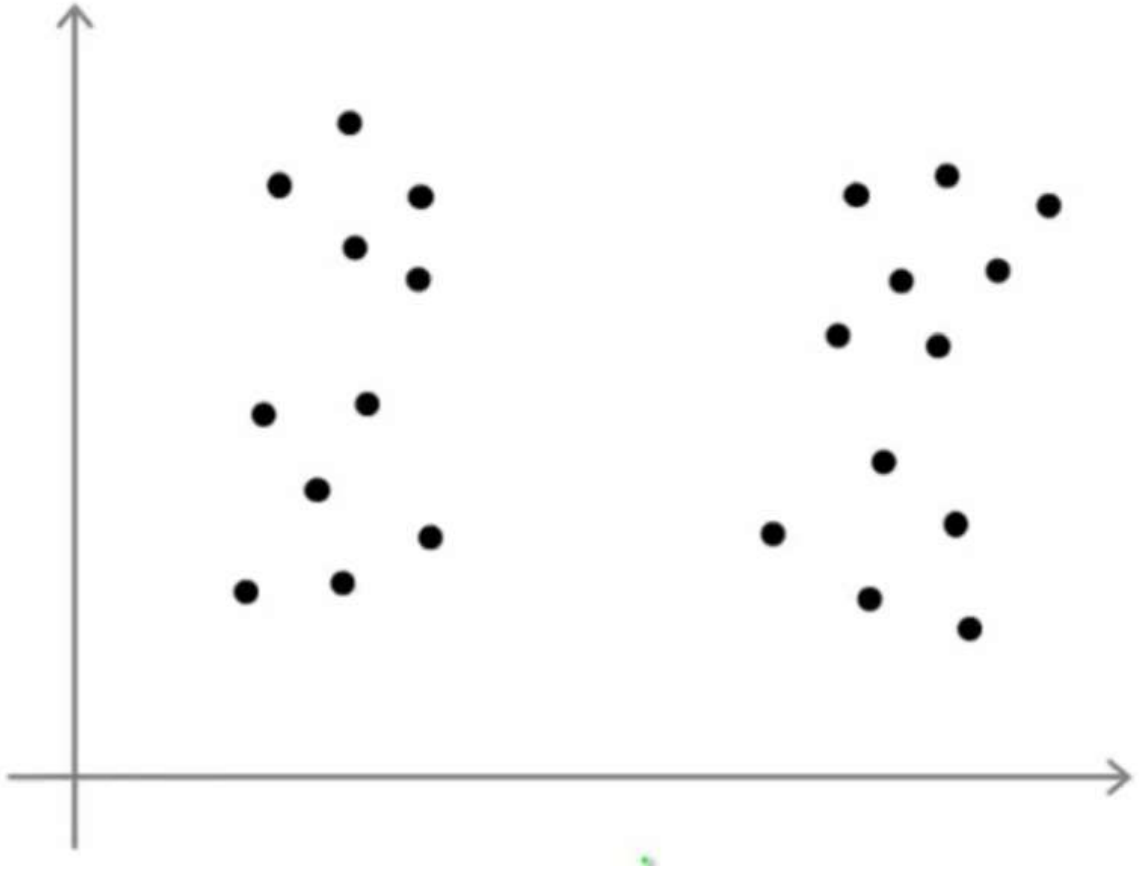
Compute cost function (distortion)

$J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K)$

}

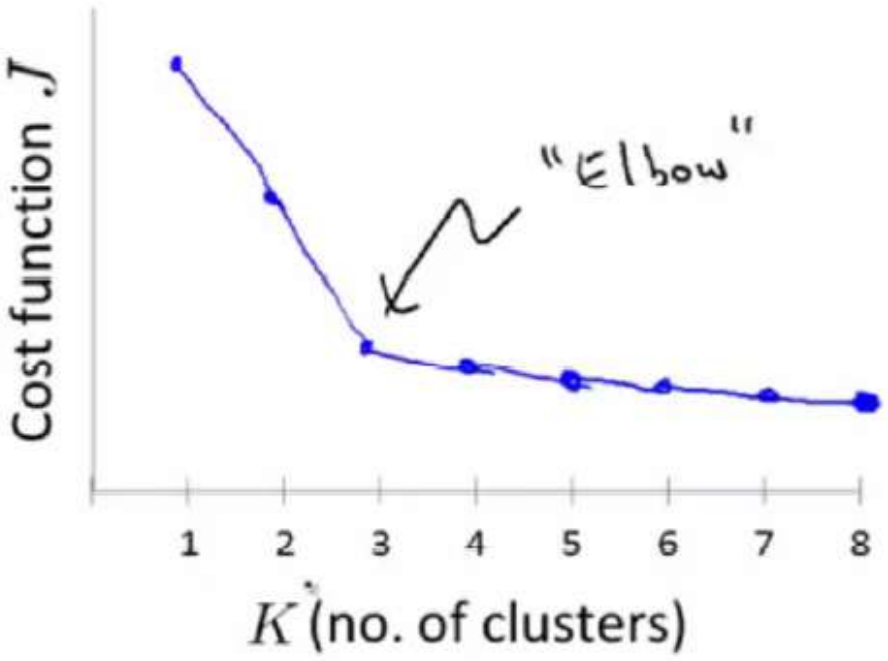
Pick clustering that gave lowest cost $J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K)$

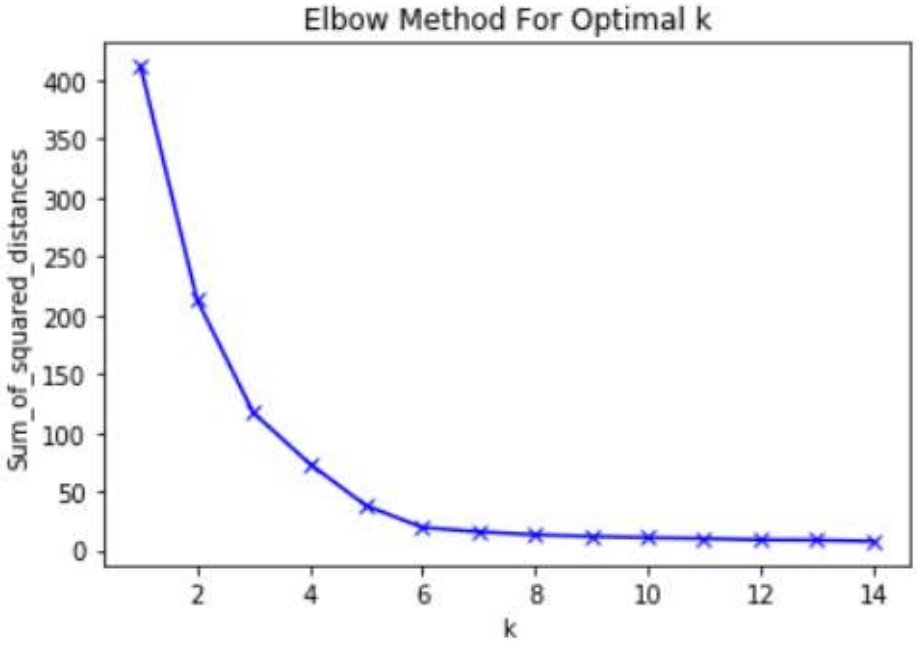
How to Choose # of clusters



- Choose K by data visualisation (if possible)
- Ask domain experts (highly recommendable) , e.g. anomaly detection (experts should know how many types of anomalies are expected)
- Choose K automatically (e.g. Elbow method)

Choosing the number of clusters (Elbow method)

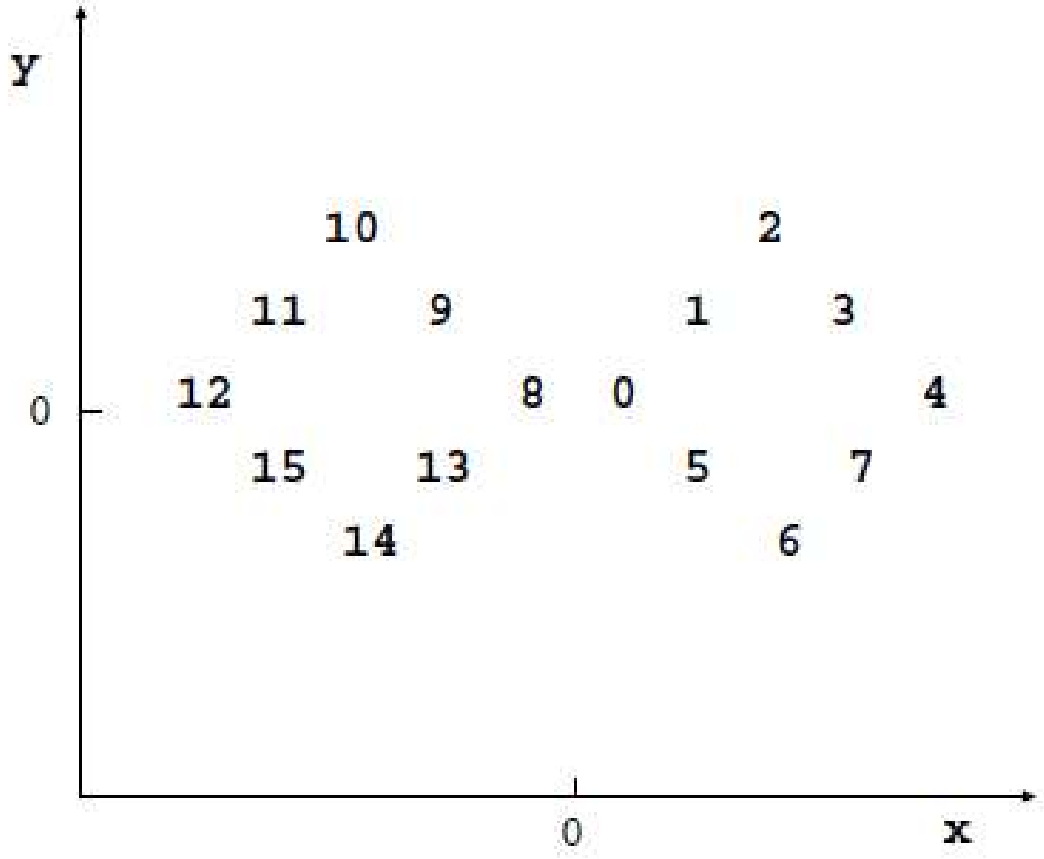




Tutorial: How to determine the optimal number of clusters for k-means clustering | by Tola Alade | Cambridge Spark
https://blog.cambridgespark.com/how-to-determine-the-optimal-number-of-clusters-for-k-means-clustering-14f27070048f

K-MEANS CLUSTERING – Example

Id	x	y
0:	1.0	0.0
1:	3.0	2.0
2:	5.0	4.0
3:	7.0	2.0
4:	9.0	0.0
5:	3.0	-2.0
6:	5.0	-4.0
7:	7.0	-2.0
8:	-1.0	0.0
9:	-3.0	2.0
10:	-5.0	4.0
11:	-7.0	2.0
12:	-9.0	0.0
13:	-3.0	-2.0
14:	-5.0	-4.0
15:	-7.0	-2.0

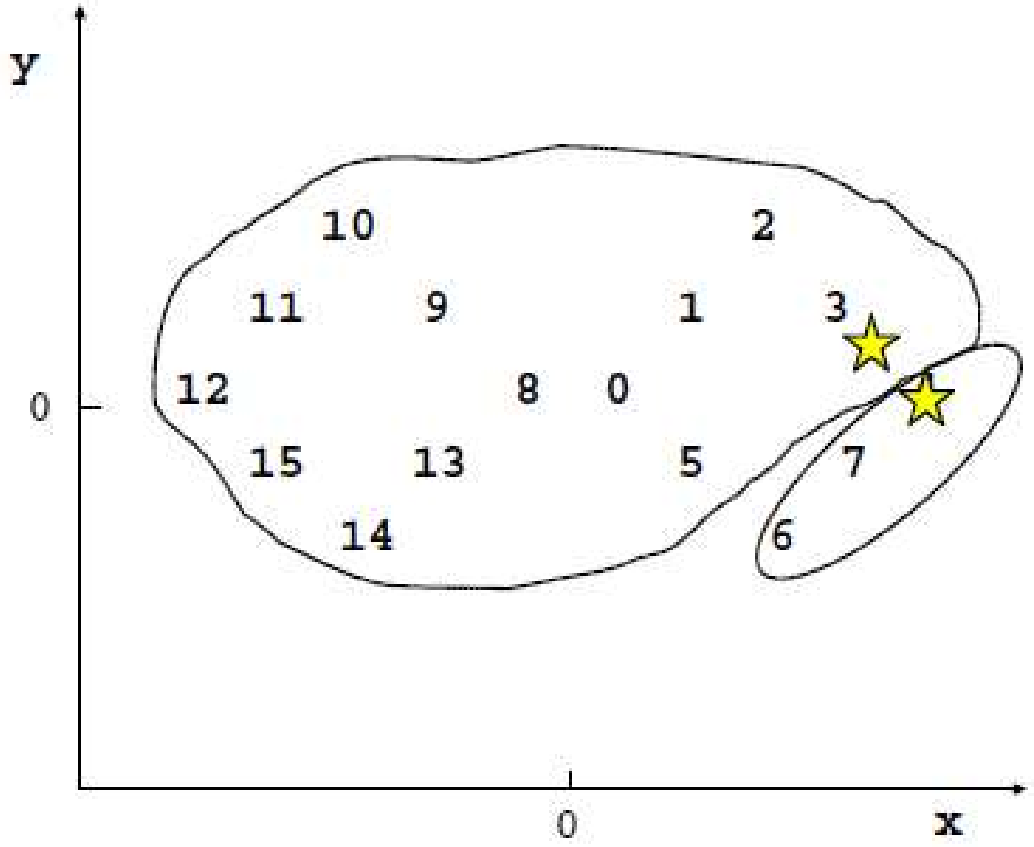


- find the best 2 clusters

K-MEANS CLUSTERING – Example

Seed: (9 0) (8 1)

Clustering: (4 6 7) (0 1 2 3 5 8 9 10 11 12 13 14 15)
Cluster Centers: (7.0 -2.0) (-1.61538 0.46153)
Average Distance: 4.35887

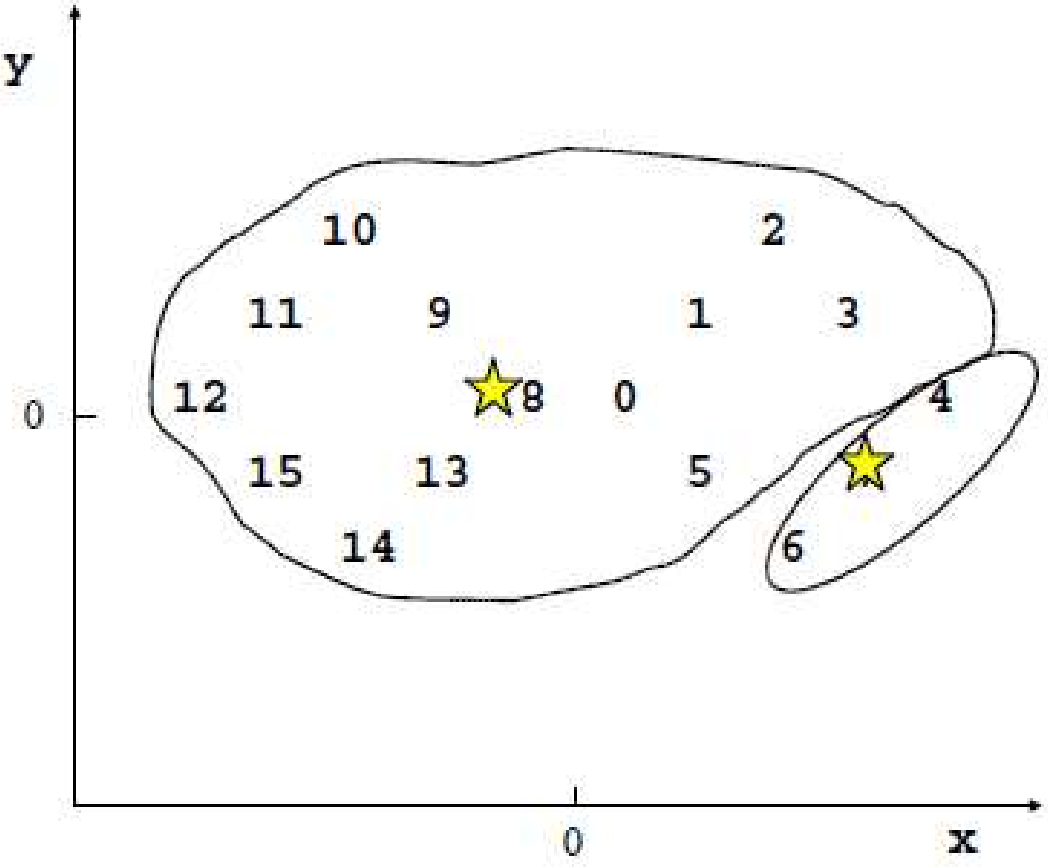


K-MEANS CLUSTERING – Example

Seed: (9 0) (8 1)

Clustering: (4 6 7) (0 1 2 3 5 8 9 10 11 12 13 14 15)
Cluster Centers: (7.0 -2.0) (-1.61538 0.46153)
Average Distance: 4.35887

Clustering: (2 3 4 5 6 7) (0 1 8 9 10 11 12 13 14 15)

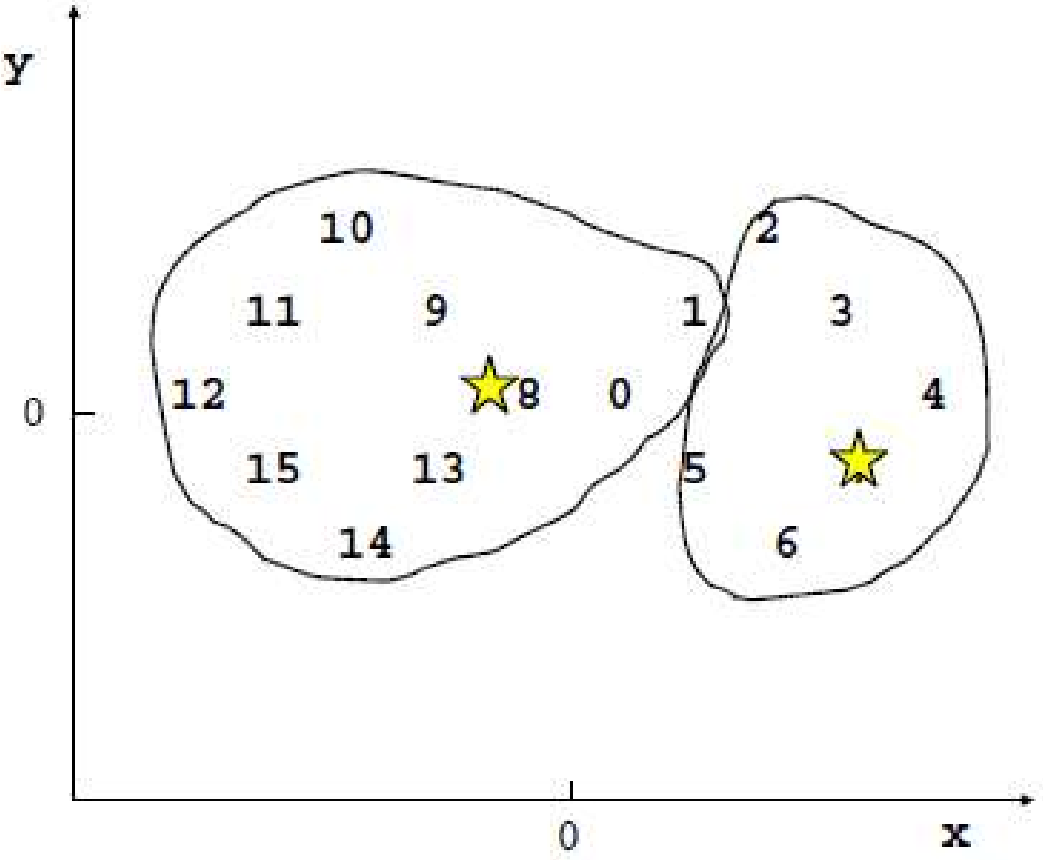


K-MEANS CLUSTERING – Example

Seed: (9 0) (8 1)

Clustering: (4 6 7) (0 1 2 3 5 8 9 10 11 12 13 14 15)
Cluster Centers: (7.0 -2.0) (-1.61538 0.46153)
Average Distance: 4.35887

Clustering: (2 3 4 5 6 7) (0 1 8 9 10 11 12 13 14 15)
Cluster Centers: (6.0 -0.33334) (-3.6 0.2)
Average Distance: 3.6928



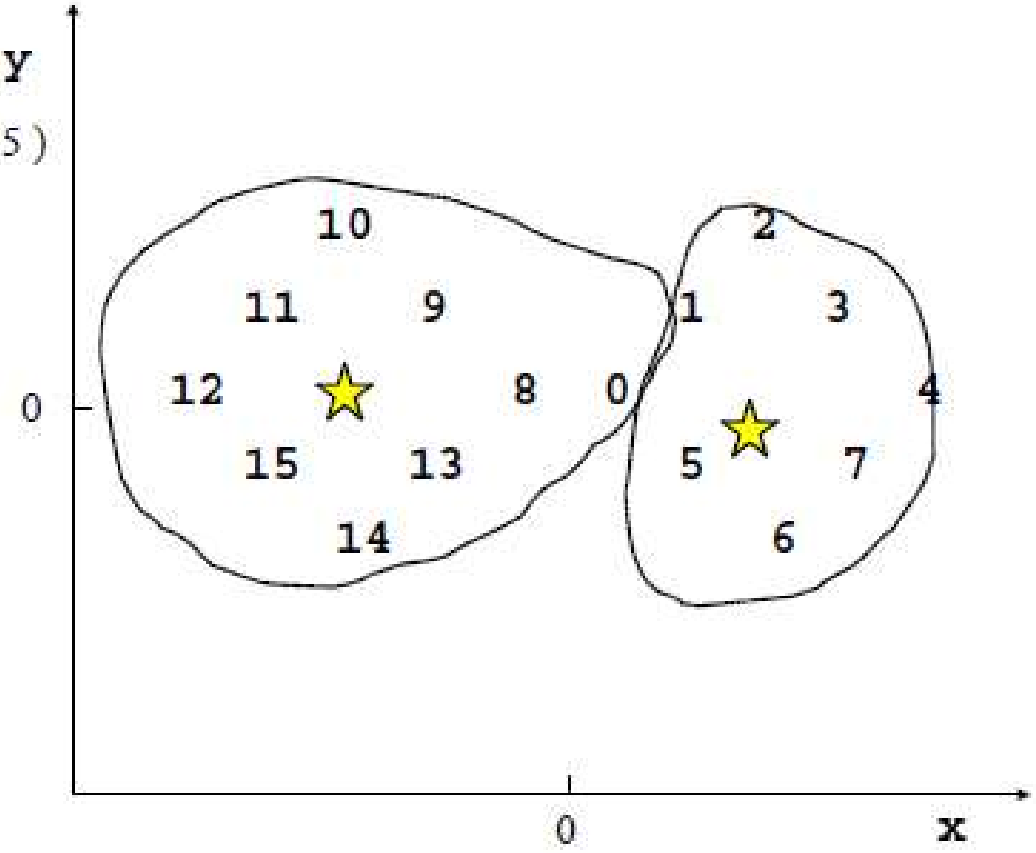
K-MEANS CLUSTERING – Example

Seed: (9 0) (8 1)

Clustering: (4 6 7) (0 1 2 3 5 8 9 10 11 12 13 14 15)
Cluster Centers: (7.0 -2.0) (-1.61538 0.46153)
Average Distance: 4.35887

Clustering: (2 3 4 5 6 7) (0 1 8 9 10 11 12 13 14 15)
Cluster Centers: (6.0 -0.33334) (-3.6 0.2)
Average Distance: 3.6928

Clustering: (1 2 3 4 5 6 7) (0 8 9 10 11 12 13 14 15)



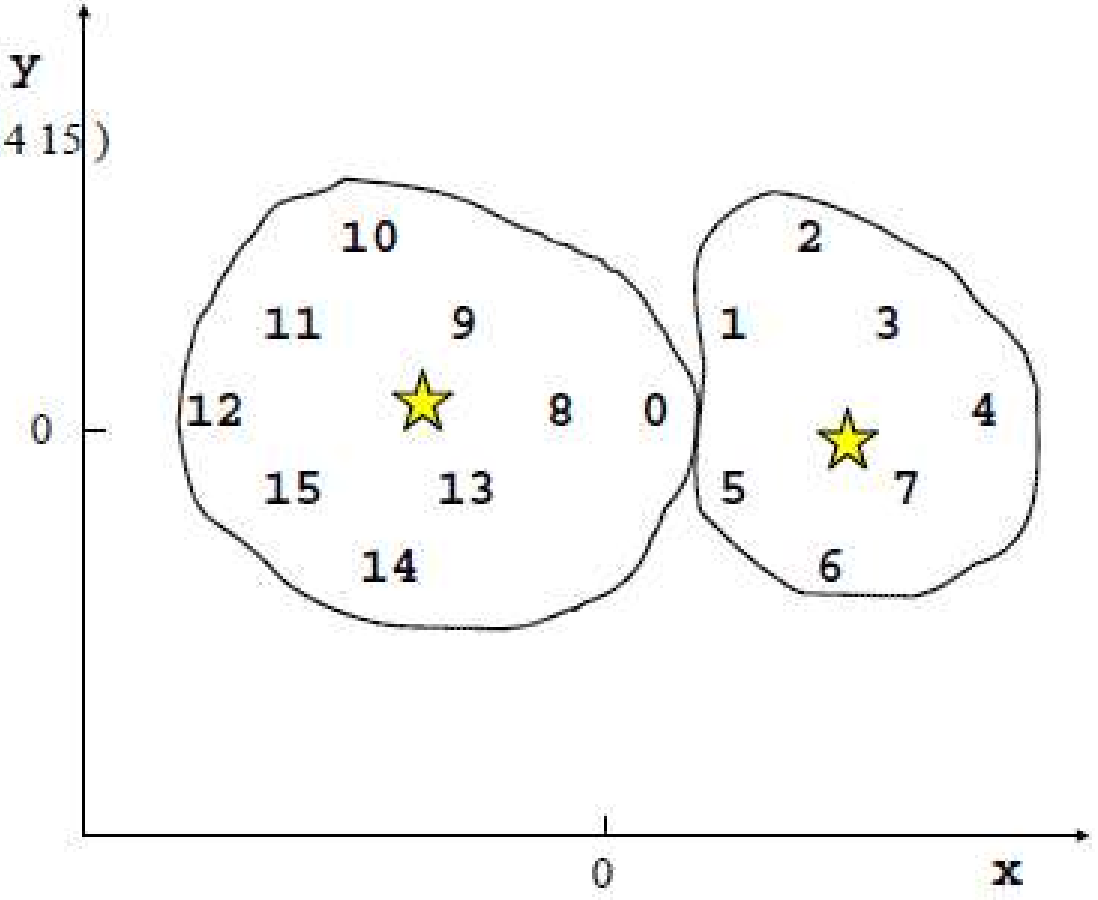
K-MEANS CLUSTERING – Example

Seed: (9 0) (8 1)

Clustering: (4 6 7) (0 1 2 3 5 8 9 10 11 12 13 14 15)
Cluster Centers: (7.0 -2.0) (-1.61538 0.46153)
Average Distance: 4.35887

Clustering: (2 3 4 5 6 7) (0 1 8 9 10 11 12 13 14 15)
Cluster Centers: (6.0 -0.33334) (-3.6 0.2)
Average Distance: 3.6928

Clustering: (1 2 3 4 5 6 7) (0 8 9 10 11 12 13 14 15)
Cluster Centers: (5.57143 0.0) (-4.33334 0.0)
Average Distance: 3.49115



K-MEANS CLUSTERING – Example

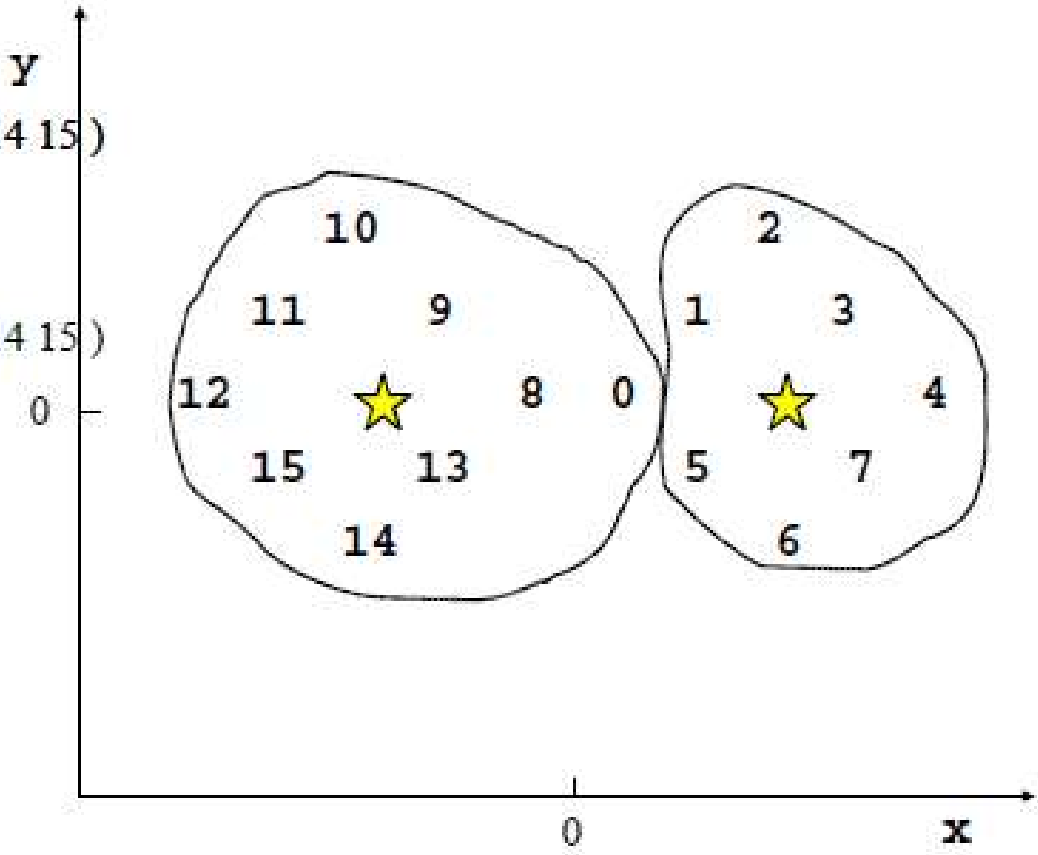
Seed: (9 0) (8 1)

Clustering: (4 6 7) (0 1 2 3 5 8 9 10 11 12 13 14 15)
Cluster Centers: (7.0 -2.0) (-1.61538 0.46153)
Average Distance: 4.35887

Clustering: (2 3 4 5 6 7) (0 1 8 9 10 11 12 13 14 15)
Cluster Centers: (6.0 -0.33334) (-3.6 0.2)
Average Distance: 3.6928

Clustering: (1 2 3 4 5 6 7) (0 8 9 10 11 12 13 14 15)
Cluster Centers: (5.57143 0.0) (-4.33334 0.0)
Average Distance: 3.49115

Clustering: (0 1 2 3 4 5 6 7) (8 9 10 11 12 13 14 15)



K-MEANS CLUSTERING – Example

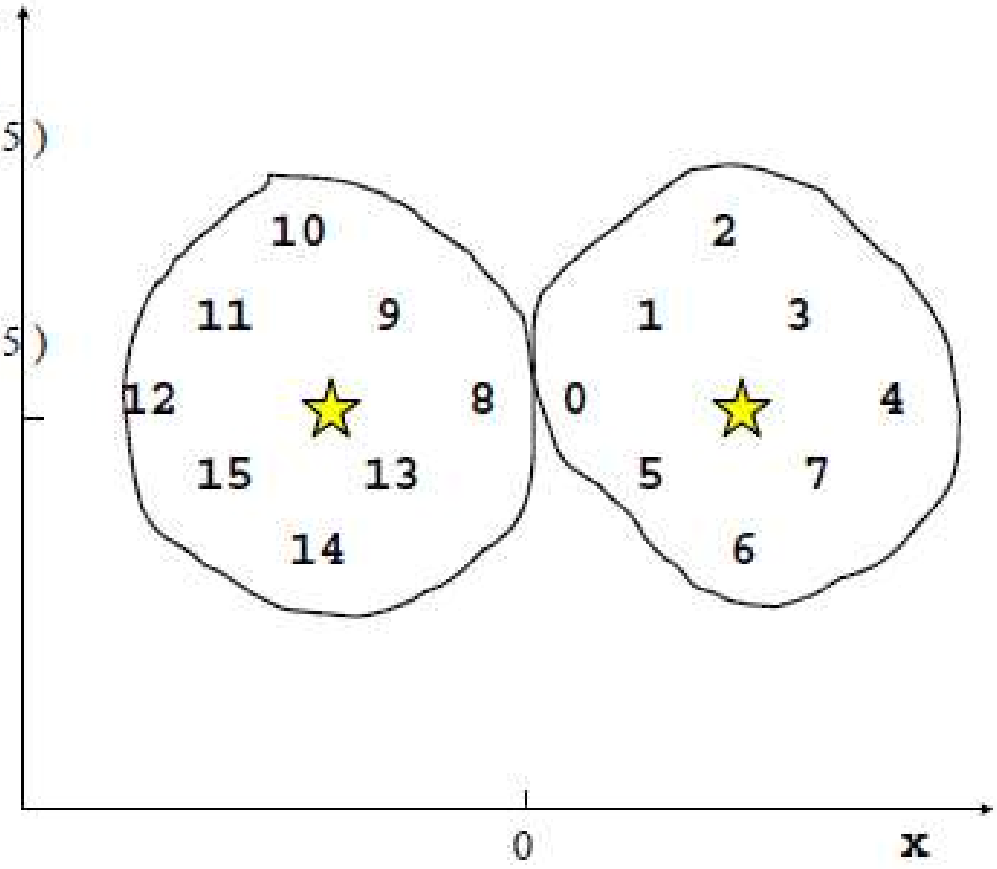
Seed: (9 0) (8 1)

Clustering: (4 6 7) (0 1 2 3 5 8 9 10 11 12 13 14 15)
Cluster Centers: (7.0 -2.0) (-1.61538 0.46153)
Average Distance: 4.35887

Clustering: (2 3 4 5 6 7) (0 1 8 9 10 11 12 13 14 15)
Cluster Centers: (6.0 -0.33334) (-3.6 0.2)
Average Distance: 3.6928

Clustering: (1 2 3 4 5 6 7) (0 8 9 10 11 12 13 14 15)
Cluster Centers: (5.57143 0.0) (-4.33334 0.0)
Average Distance: 3.49115

Clustering: (0 1 2 3 4 5 6 7) (8 9 10 11 12 13 14 15)
Cluster Centers: (5.0 0.0) (-5.0 0.0)
Average Distance: 3.41421



K-MEANS CLUSTERING – Example

Seed: (9 0) (8 1)

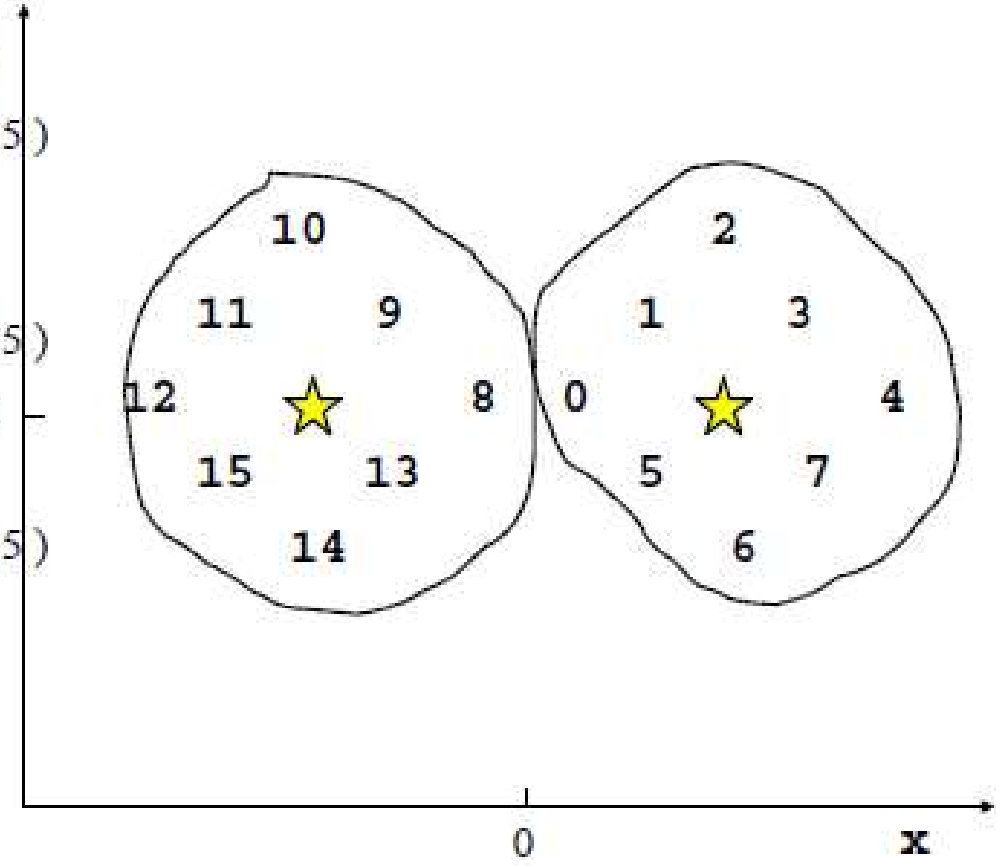
Clustering: (4 6 7) (0 1 2 3 5 8 9 10 11 12 13 14 15)
Cluster Centers: (7.0 -2.0) (-1.61538 0.46153)
Average Distance: 4.35887

Clustering: (2 3 4 5 6 7) (0 1 8 9 10 11 12 13 14 15)
Cluster Centers: (6.0 -0.33334) (-3.6 0.2)
Average Distance: 3.6928

Clustering: (1 2 3 4 5 6 7) (0 8 9 10 11 12 13 14 15)
Cluster Centers: (5.57143 0.0) (-4.33334 0.0)
Average Distance: 3.49115

Clustering: (0 1 2 3 4 5 6 7) (8 9 10 11 12 13 14 15)
Cluster Centers: (5.0 0.0) (-5.0 0.0)
Average Distance: 3.41421

Clustering: (0 1 2 3 4 5 6 7) (8 9 10 11 12 13 14 15)
No improvement.



K-MEANS -summary

- The most popular clustering method.
- Need to know K.
- May converge to a Local Minimum .
- High number of computations.

K-means for dimensionality reduction

dimensionality reduction is useful for:

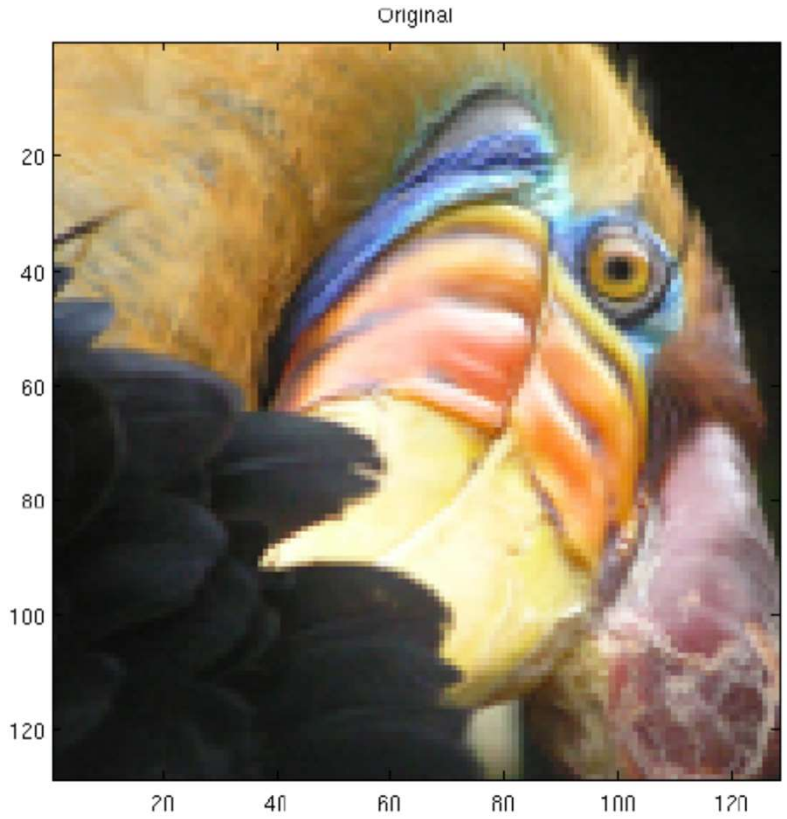
Data compression (from 10000-1000 D to) 100 D

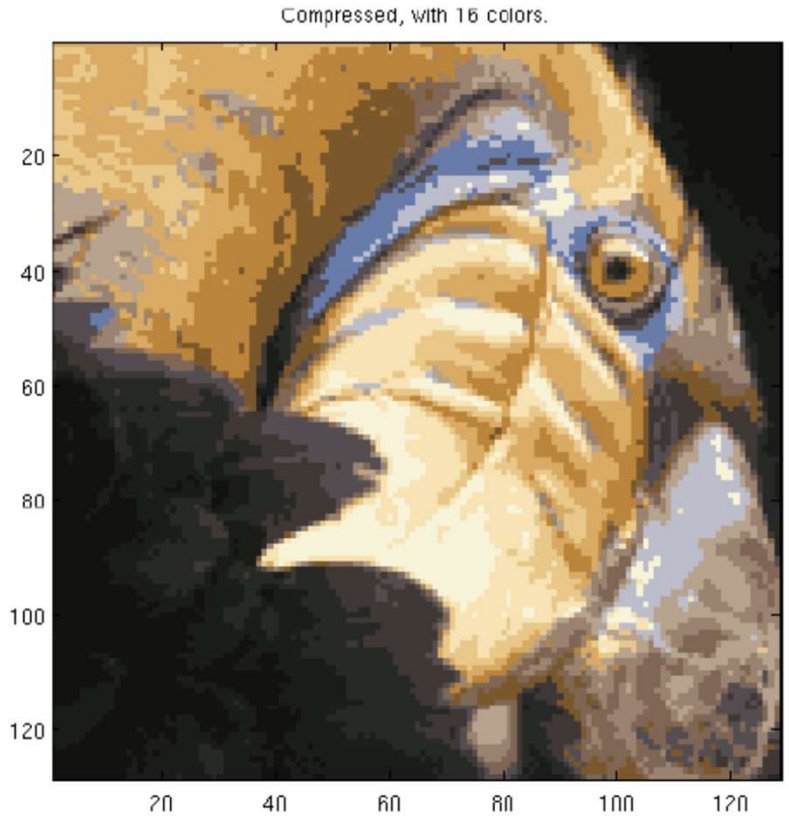
Reduce memory/ disk needed to store data
Speed up learning algorithm

Data visualization (from 100-50D to 2-3D)

Image compression with K-means

RGB image: 3*8 bits/pixel Compressed image:16 colors(clusters) => 4 bits

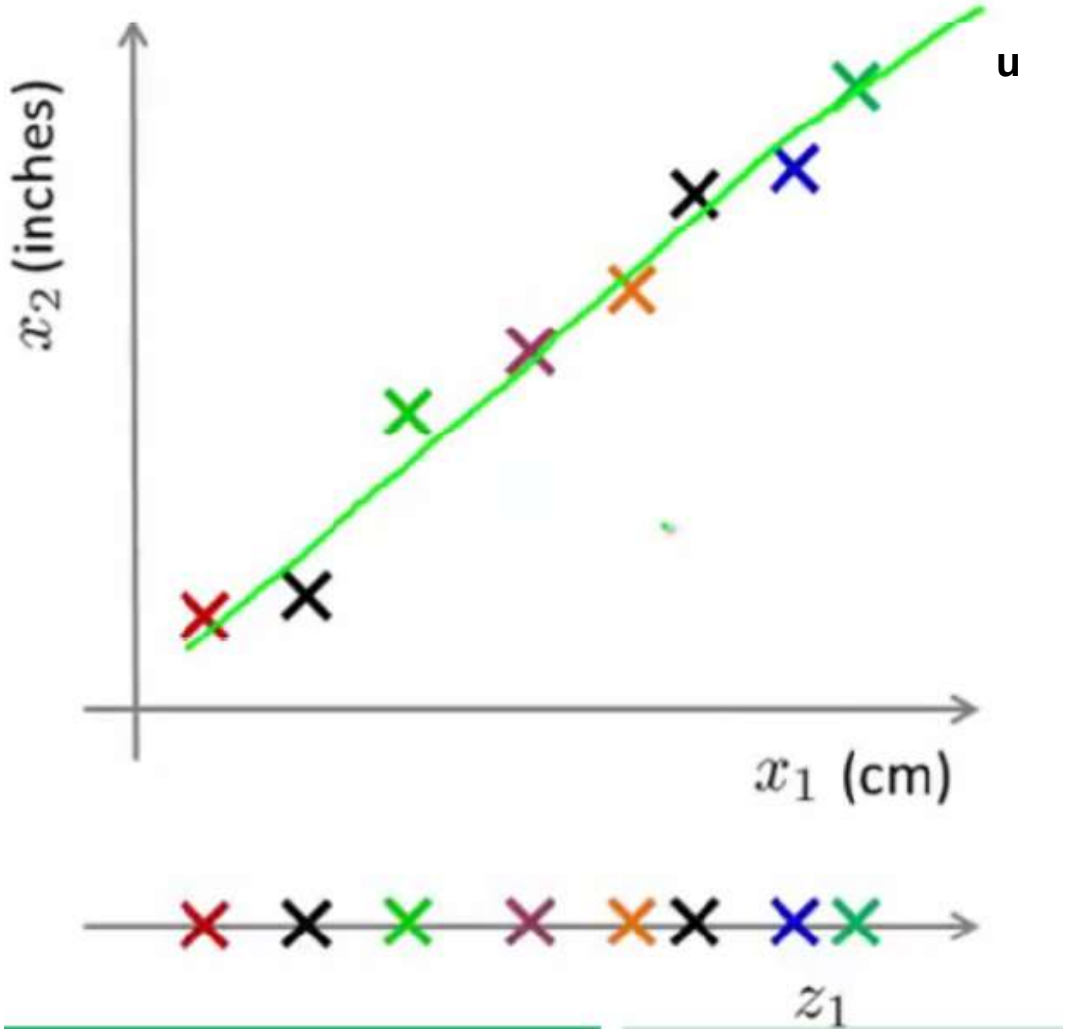




DATA COMPRESSION

Example: reduce data from 2D to 1D

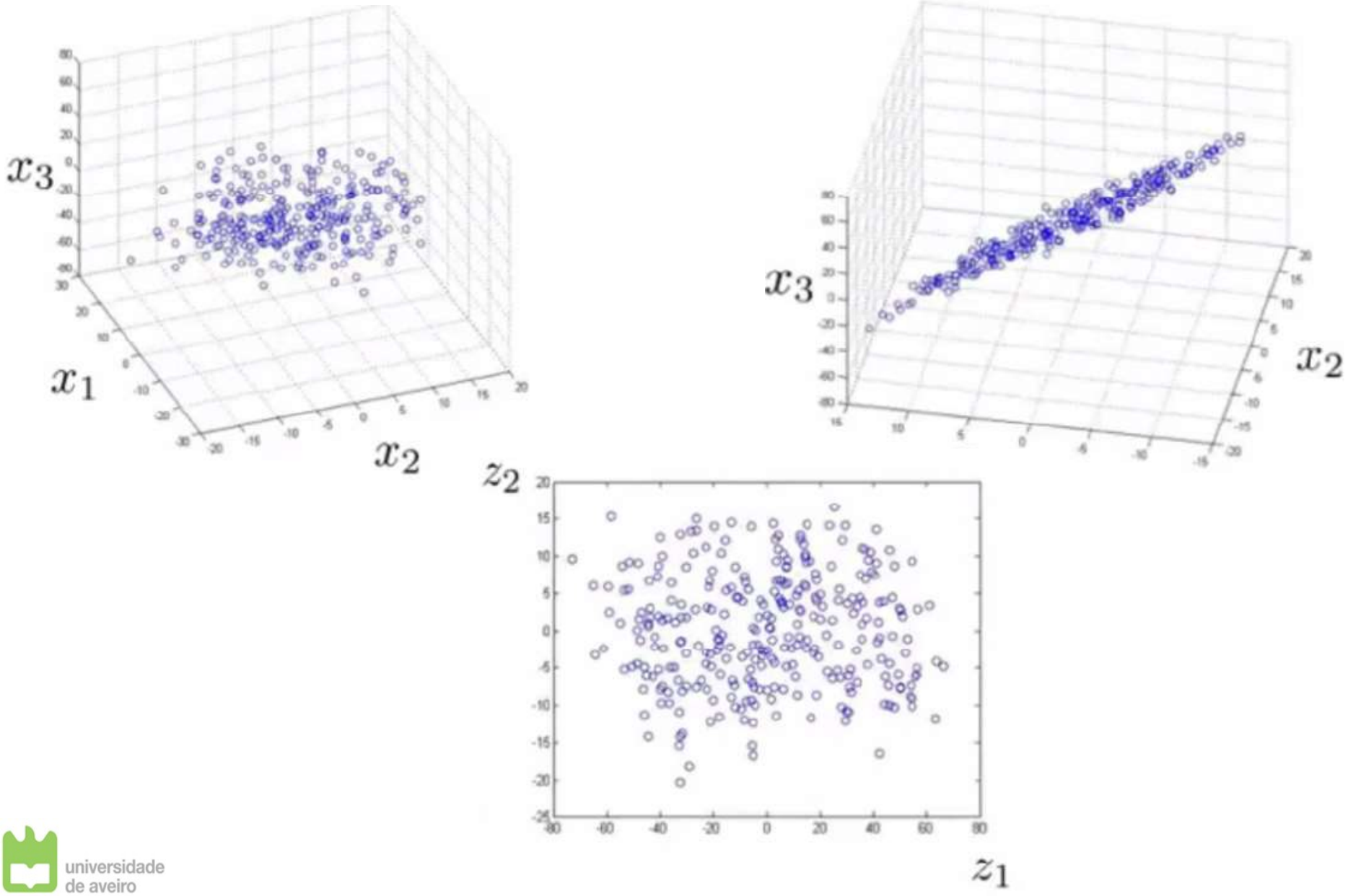
If 2D data is located along a line, the second dimension is redundant



DATA COMPRESSION

Example: reduce data from 3D to 2D

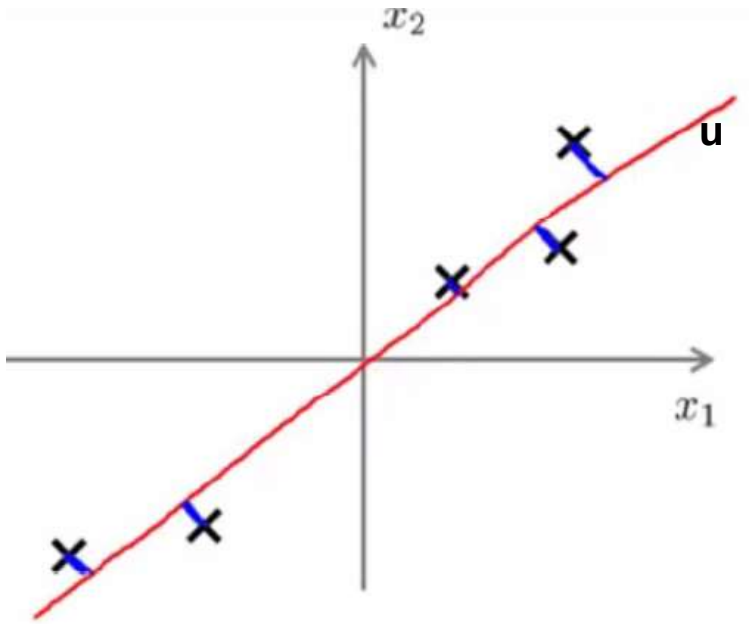
If 3D data is located along a plane, the 3rd dimension is redundant



PRINCIPAL COMPONENT ANALYSIS (PCA)

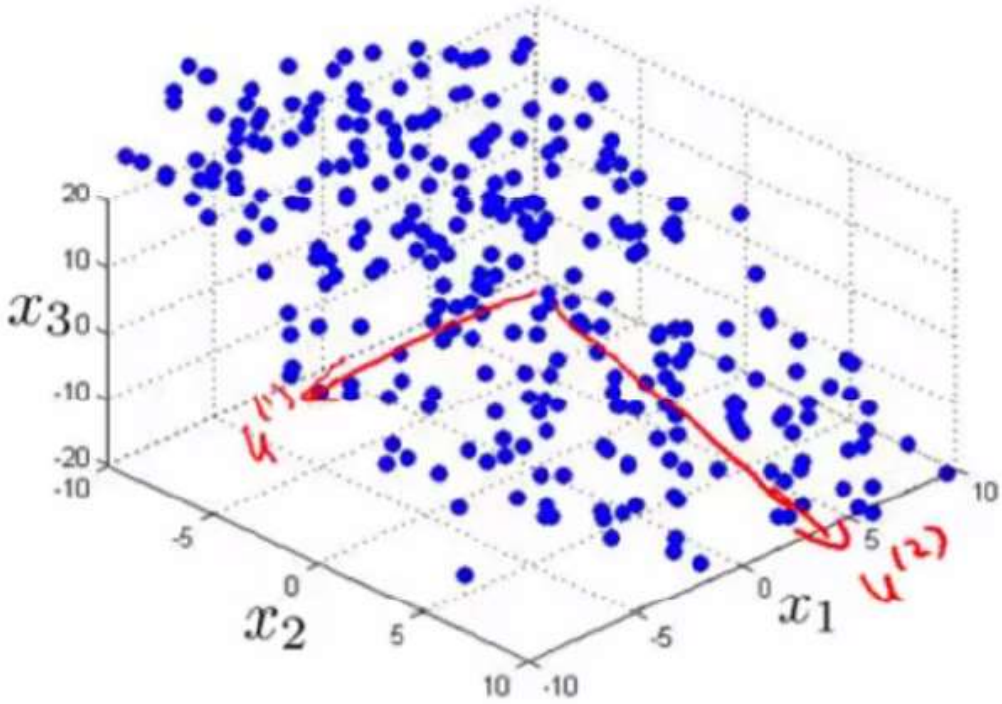
Reduce from 2D to 1D:

find the best direction (vector u) onto which to project data such that to minimize the projection error



Reduce from 3D to 2D:

find the orientation of the best plane (vectors u1, u2) onto which to project data such that to minimize the projection error

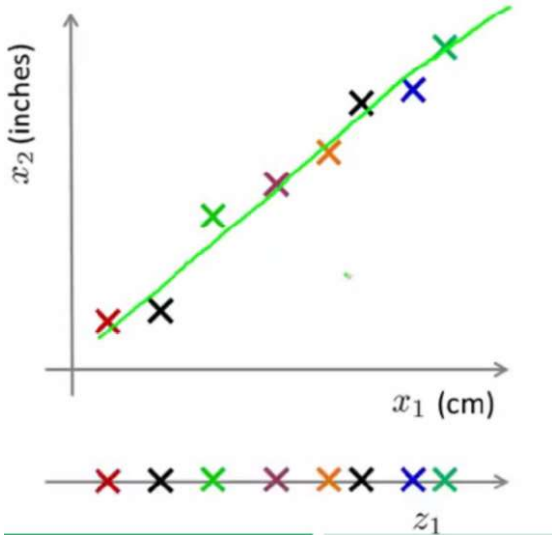


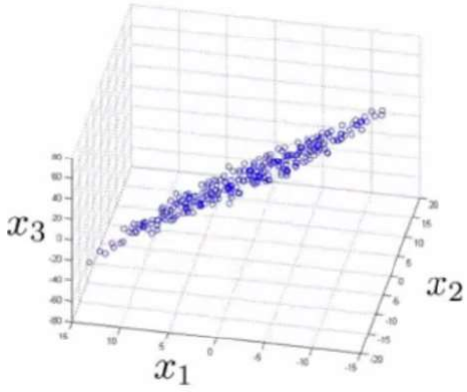
PRINCIPAL COMPONENT ANALYSIS (PCA)

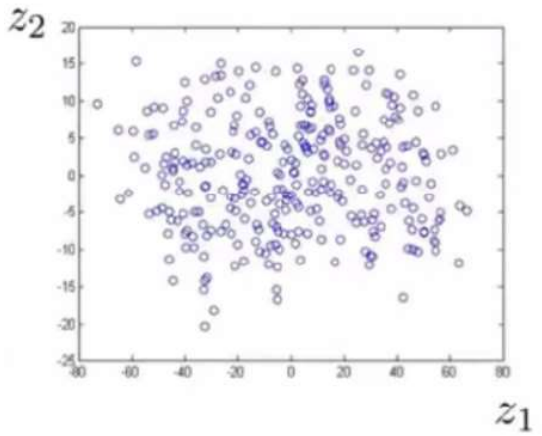
Reduce from n-dimension to k-dimension (k<n):

Two tasks:

- ***Task 1:*** Compute the coordinate system of the best lower k-dimensional surface (represented by vectors u1,..uk) onto which to project data such that to minimize the projection error.
- ***Task 2:*** Compute the values of the projected data in the lower dimensional space (z values)







PCA – DATA PREPROCESSING (step 1)

Training set: $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$

Preprocessing (feature scaling/mean normalization):

$$\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)}$$

Replace each $x_j^{(i)}$ with $x_j - \mu_j$.

Thus, all features have zero mean !

If the features have significantly different range of values, normalize them., e.g. in the interval [0,1] or [-1,1] or mean=0 & std=1

matrix X (mxn)	feature x_1	feature x_2	.....	feature x_n
Example 1	$x_1^{(1)}$	$x_2^{(1)}$		$x_n^{(1)}$
Example 2	$x_1^{(2)}$	$x_2^{(2)}$		$x_n^{(2)}$
...				
Example i	$x_1^{(i)}$	$x_2^{(i)}$		$x_n^{(i)}$
...				
...				
Example m	$x_1^{(m)}$	$x_2^{(m)}$		$x_n^{(m)}$

DATA NORMALIZATION

from sklearn.preprocessing import MinMaxScaler

```
mms = MinMaxScaler()
mms.fit(data)
data_transformed = mms.transform(data)
```

MinMaxScaler?

MinMaxScaler(feature_range=(0, 1), copy=True)

Transforms features by scaling each feature to a given range.

from sklearn.preprocessing import StandardScaler

```
sc=StandardScaler()
sc.fit(data)
data_transformed =sc.transform(data)
```

Standardize features by removing the mean and scaling to unit variance.

PCA – Singular Value Decomposition (step 2)

- Compute Covariance matrix of the mean normalized data matrix *X* (dimension *mxn* - m examples, n features):

$$\mathbf{Cov = X^T * X \,/ m}$$

- Compute Singular Value Decomposition(SVD) of Covariance matrix:

$$\mathbf{Cov = U * S * V}$$

U (*nxn*) - matrix of eigenvectors:

$$U = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \cdots & u^{(n)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times n}$$

S (*nxn*)– diagonal matrix of singular values in decreasing order:

$$S_{nxn} = \begin{bmatrix} S_{11} & 0 & \cdots & 0 \\ 0 & S_{22} & \vdots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ 0 & \cdots & 0 & S_{nn} \end{bmatrix}$$

SVD is equivalent to eigen-values/eigen-vector decomposition.

PCA

The projection vectors are the first k columns of U (k<n): -***Task 1***

$$U = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \cdots & u^{(n)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times n} \qquad Ureduce_{nxk} = U(:,1:k)$$

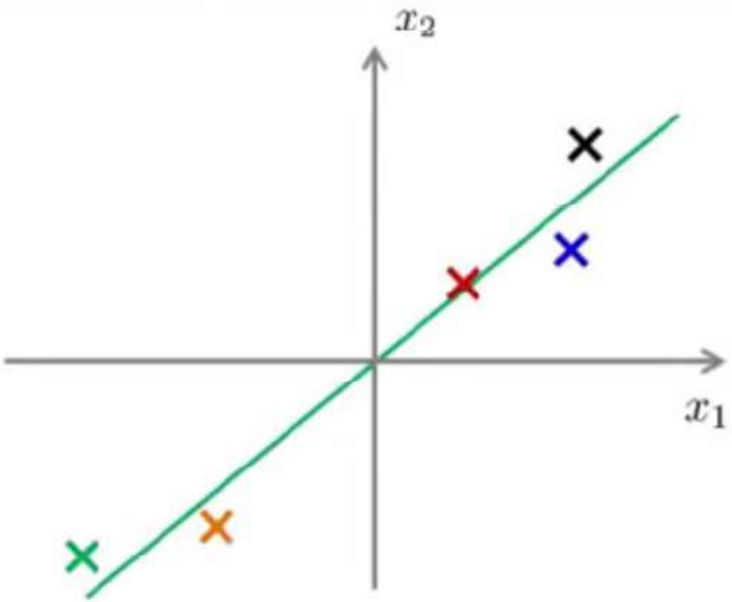
Step 3: Compute the new (projected) data matrix Z (m examples, k features): - ***Task 2***

$$Z_{mxk} = X_{mxn} * Ureduce_{nxk}$$

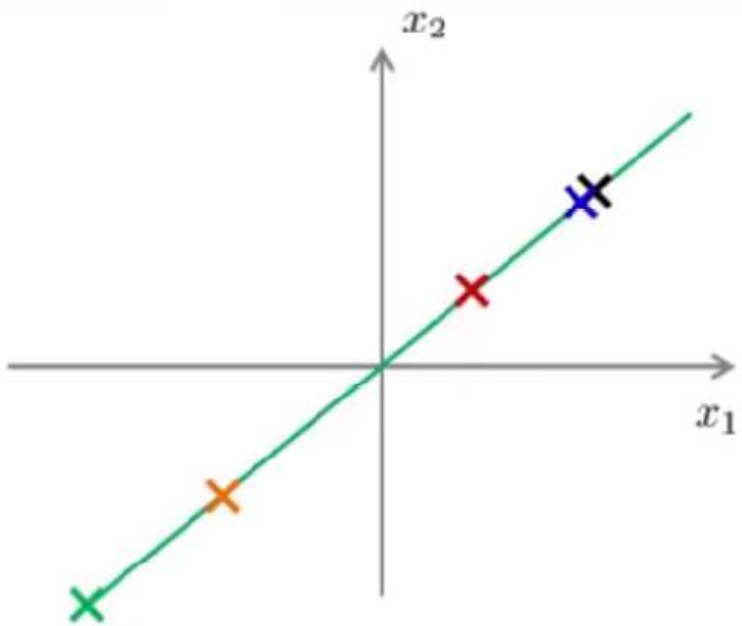
Step 4: Reconstruct data matrix X from the projected Z matrix :

$$X_{approx(mxn)} = Z_{mxk} * Ureduce^{T}_{kxn}$$

Reconstruction from compressed representation



$$z = U_{reduce}^{T} x$$



$$x_{approx} = U_{reduce} z$$

Choosing k (number of principal components)

Average squared approximation error:

$$\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)} - x_{approx}^{(i)}\|^2$$

Total data variation:

$$\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}\|^2$$

Typically, choose k to be smallest value so that

$$\frac{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)} - x_{approx}^{(i)}\|^2}{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}\|^2} \leq 0.01 \qquad (1\%)$$

"99% of variance is retained"

(typically the desired retained variance is between 90-99%)

Choosing k (number of principal components)

Compute SVD once: [U,S,V]=svd(Cov)

Choose k<n such that you get the desired retained variance (e.g. 99 %):

$$\frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \geq 0.99$$

$$S_{nxn} = \begin{bmatrix} S_{11} & 0 & \cdots & 0 \\ 0 & S_{22} & \vdots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ 0 & \cdots & 0 & S_{nn} \end{bmatrix}$$

PCA – summary

PCA is used to reduce the dimensionality of the original feature space and, at the same time, to maximise the orthogonality between the features in the transformed feature space.

The new set of features obtained through the PCA process are the principal components, which are computed by applying a linear transform to the original features.

Such principal components correspond to largest eigenvalues of the co-variance matrix of features. Then a reduced set of principal components can be used to reconstruct most of the original data with maximum variance, thus keeping most of its information.

The orthogonality between components ensures decorrelation in the transformed feature space.