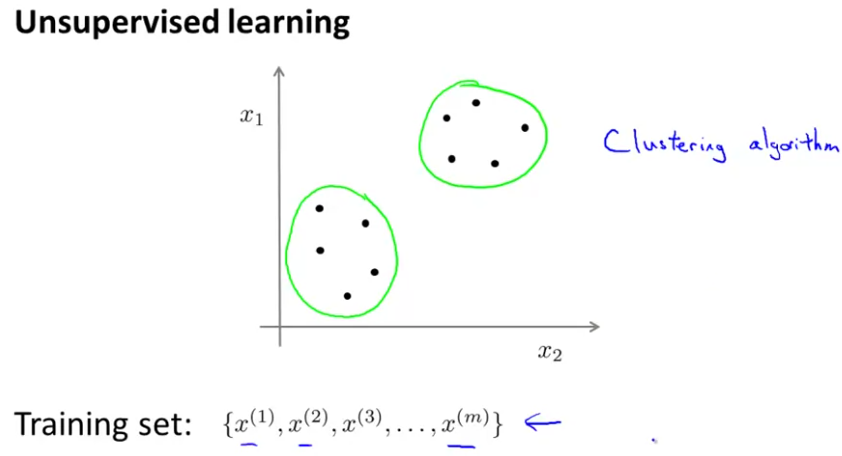
**Clustering**



We don’t label the input data (we don’t have y labels associated to the training data).

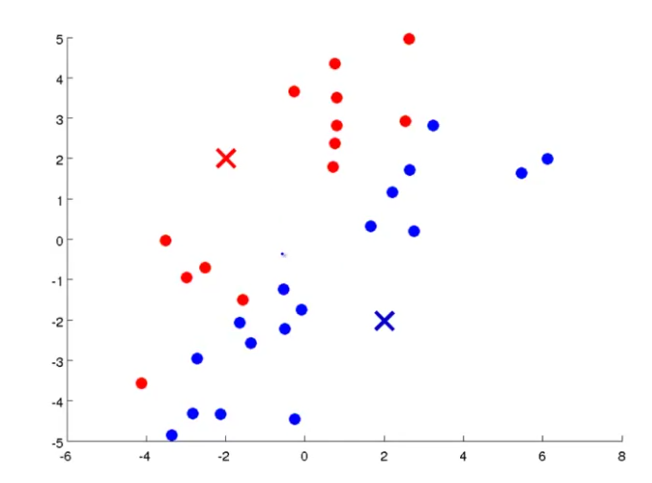
**K-Means Algorithm**

It’s an iterative algorithm.

First, we initialize some points, called the cluster centroids. We have as many points as clusters we wanna group the data on.

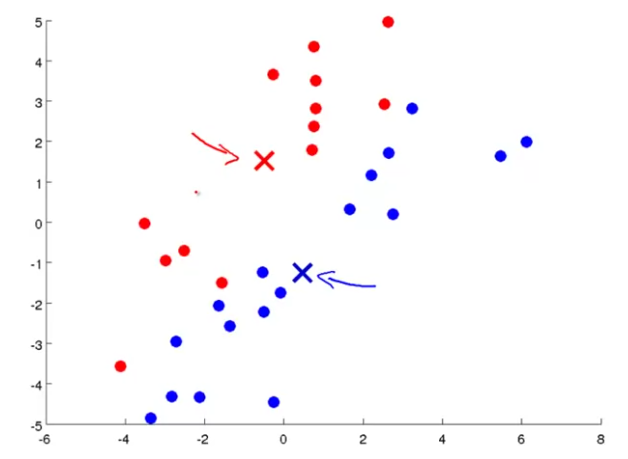
Then, the algorithm consists in two steps: the cluster assignment step and the move centroid step.

1º) The cluster assignment step consists in assigning each of the data points to one of the cluster centroids, based on the proximity of a point to a cluster. If a cluster centroid has 0 points assigned, we eliminate it.

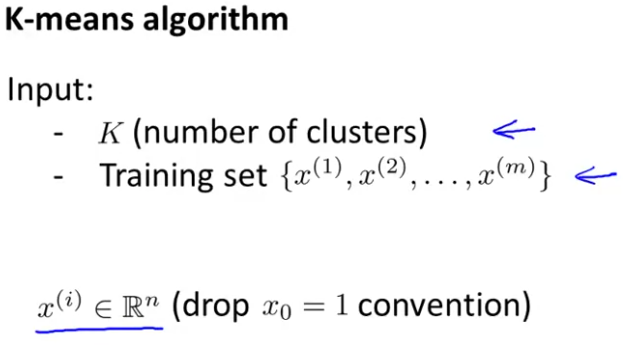


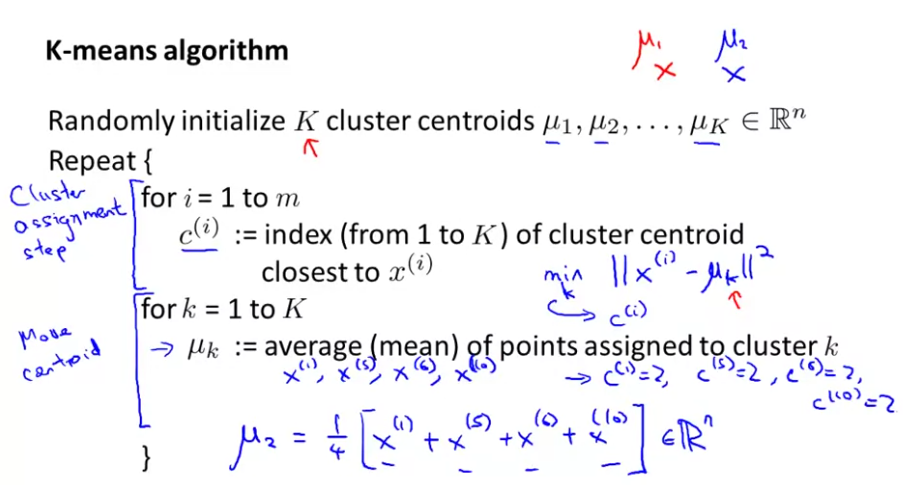
Cluster centroids

2º) The move centroid step consists in calculating the mean of the points of each cluster and in moving the cluster centroids to these positions.



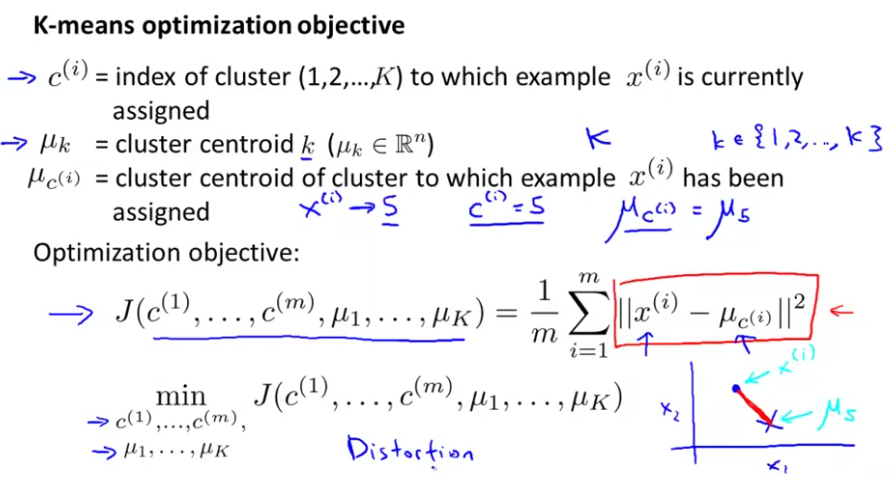
Then, we restart the algorithm and we repeat these two steps. Formally:





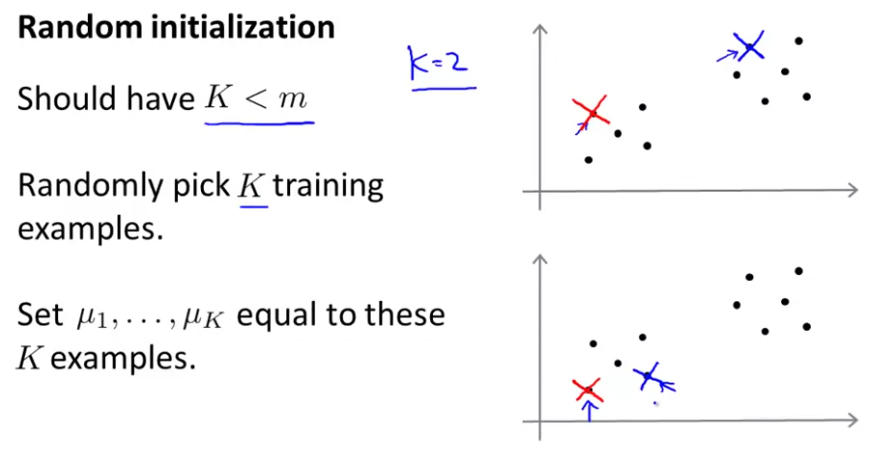
**Optimization objective**

It consists in minimizing the distortion J defined as:

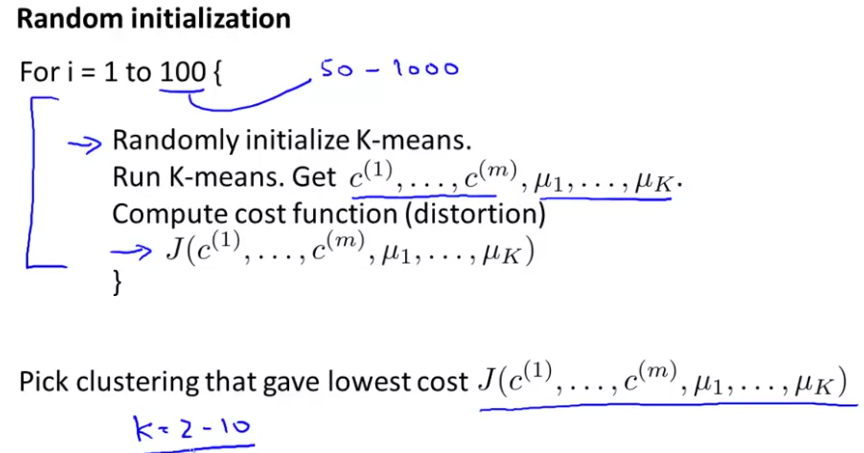


The K-means algorithm looks to minimize the distortion value by modifying the parameters C(i) and µk through the two steps mentioned previously.

**K-means algorithm random initialization**



In order to avoid the algorithm converging to a local optima, you have to randomly initialize the cluster centroids many times and choose the case with the lowest cost. This only works fine if the number of clusters K is small enough:



**Choosing the number of clusters (choosing K)**

Normally, you choose K in function of your application. However, if that’s not the case, you can plot the cost function J depending on the number of clusters and looking at the curve, choose the best value for K.

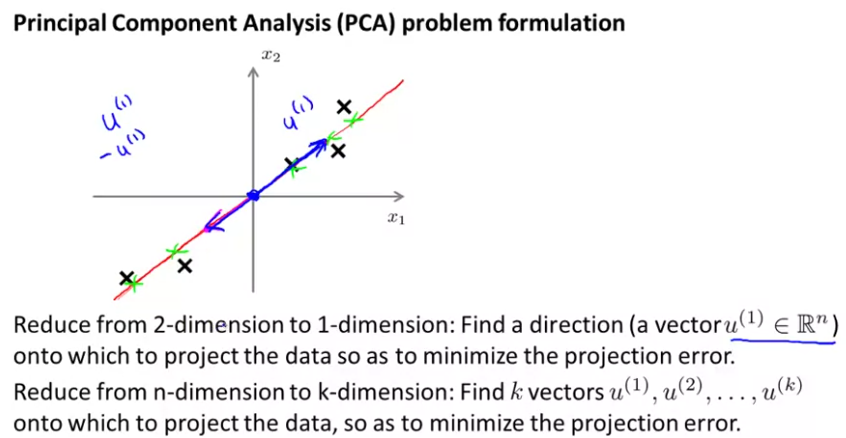
ONLY USE DIMENSIONALITY REDUCTION IF YOUR ALGORITHMS DON’T WORK WITH THE ORIGINAL DATA

**Dimensionality Reduction**

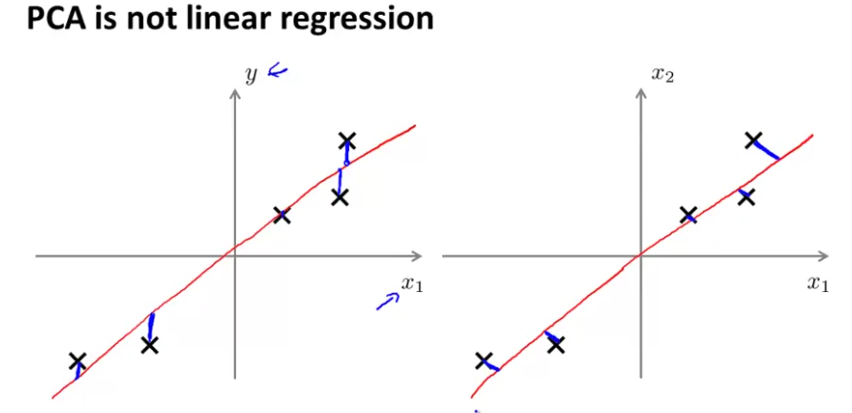
It is useful for data compressing, which can make our algorithms run faster and reduce the memory/disk needed for data. Also, it can help to obtain a better data visualization.

**PCA**

PCA is a dimensionality reduction method that tries to find the surface (or vector) which minimizes the projection error of our data.

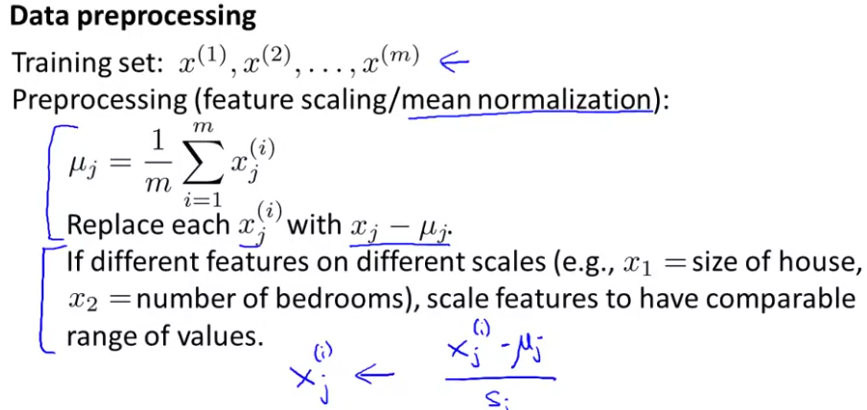


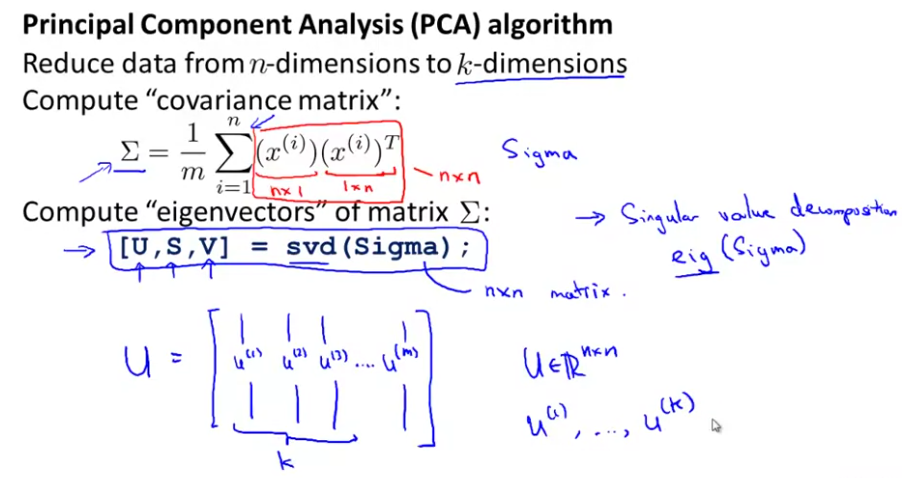
Linear regression tries to minimise the square error between the line and the data points, while PCA looks to reduce the orthogonal distance (squared projection error):



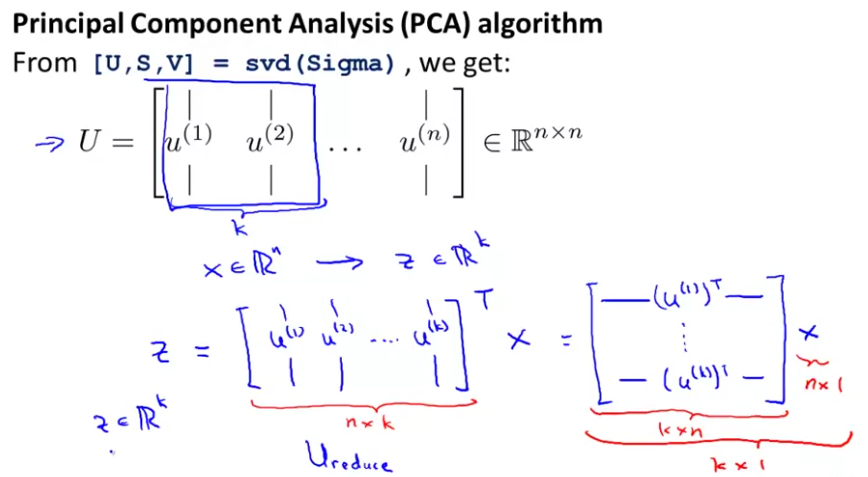
**PCA analysis algorithm**

PCA requires of a data pre-processing, consisting in feature scaling and mean normalization.

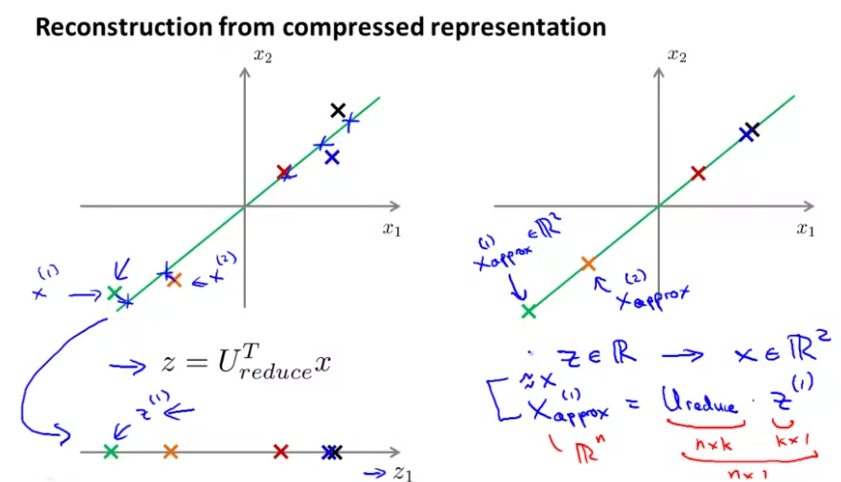




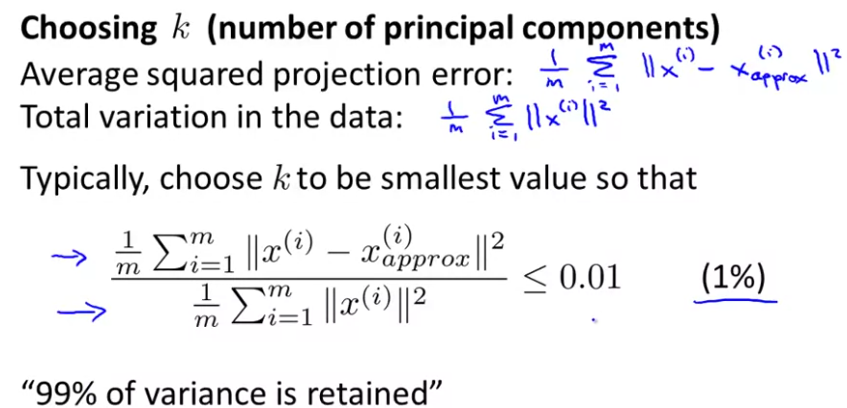
For reducing the data from n to k dimensions, we calculate the matrix U and we take the first k columns.

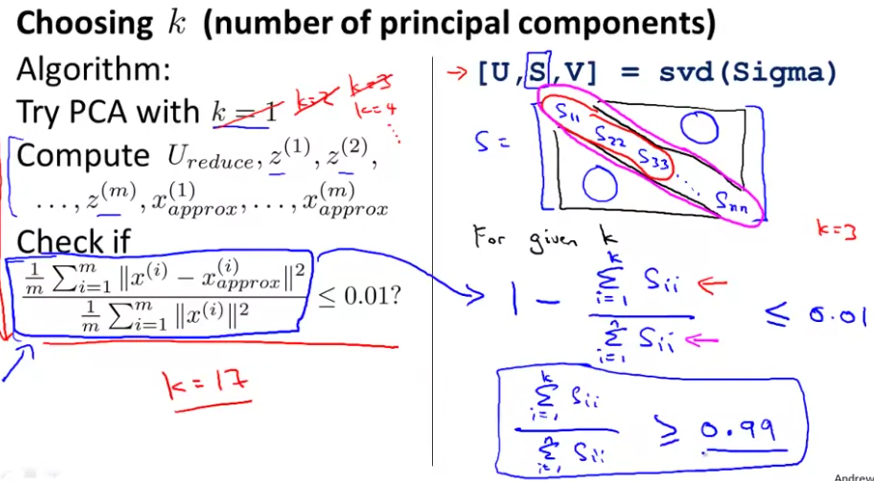


**Reconstruction from Compressed Representation**

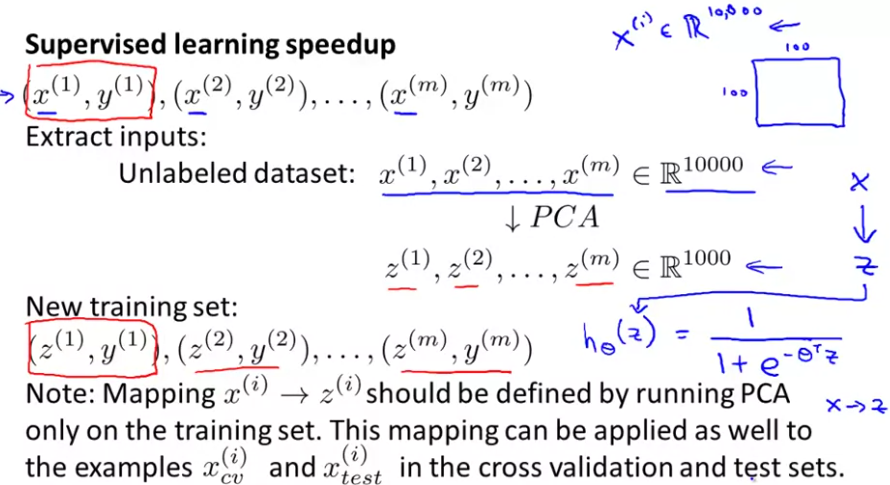


**Choosing the number of principal components (k)**

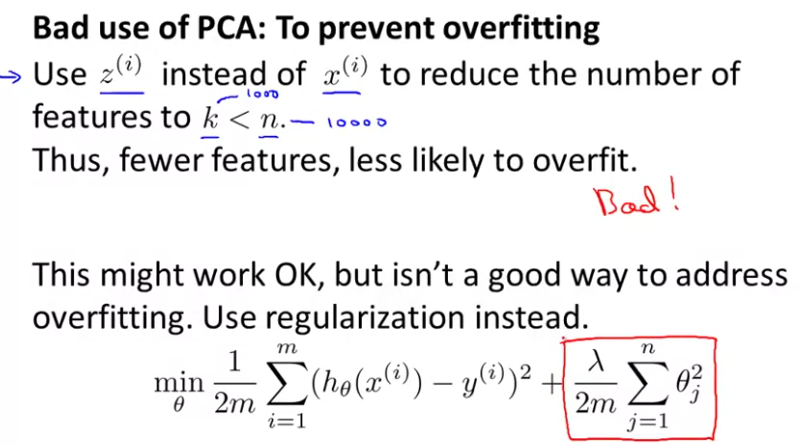




**How to speed up a supervised learning algorithm**



We run the PCA only on the training set. Then, we can apply the Ureduce matrix for the rest of the sets.



It’s bad because PCA doesn’t makes the difference between x and y, so by applying it, we can lose some valuable information.