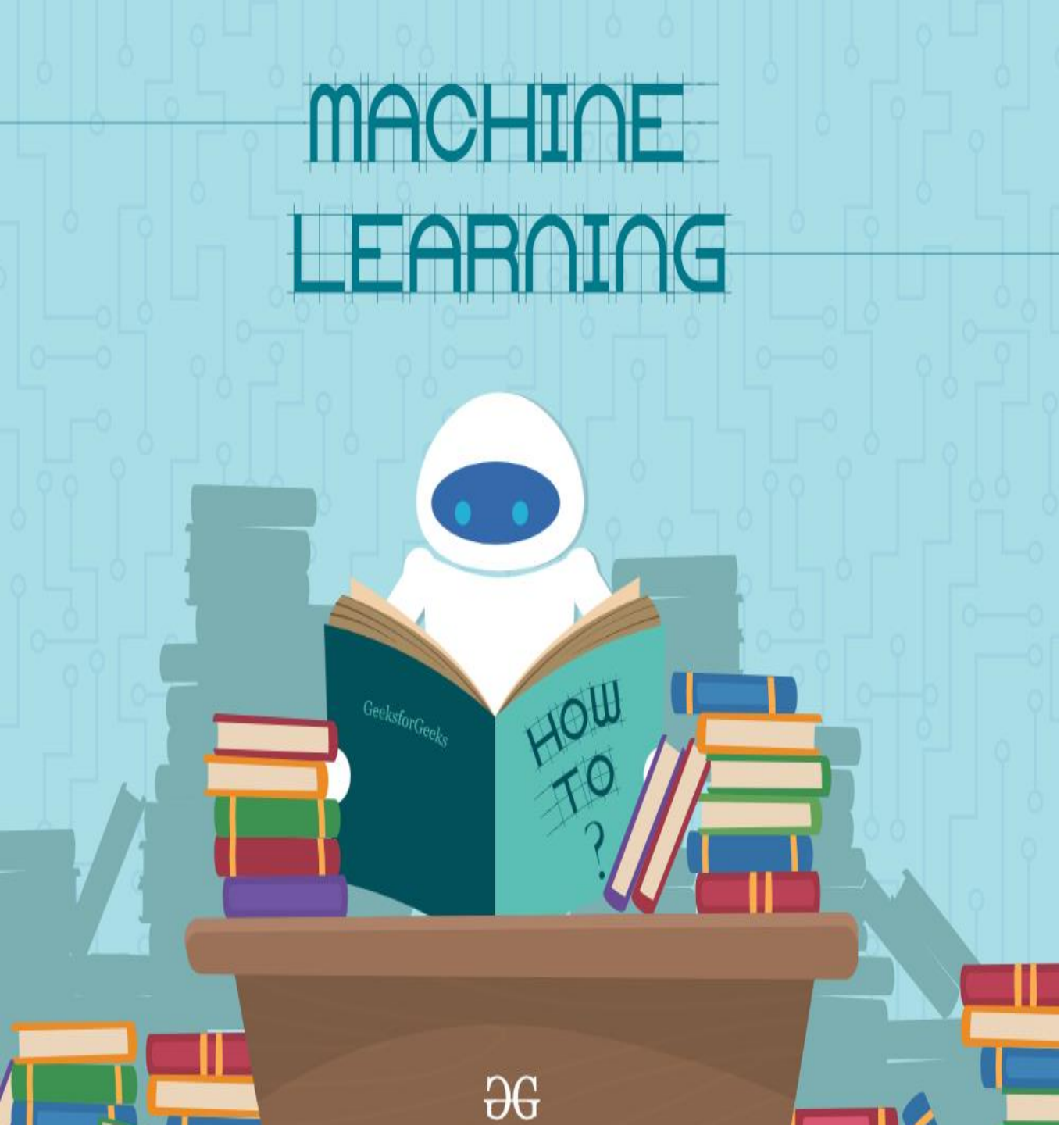


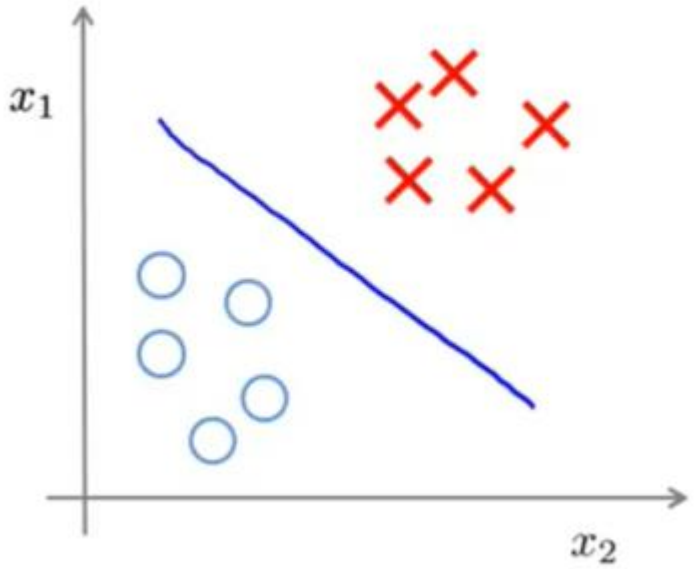
>TCC
THE CODERS' CLUB

Unsupervised Learning
Dimensionality Reduction

WEEK 8

Supervised Learning:

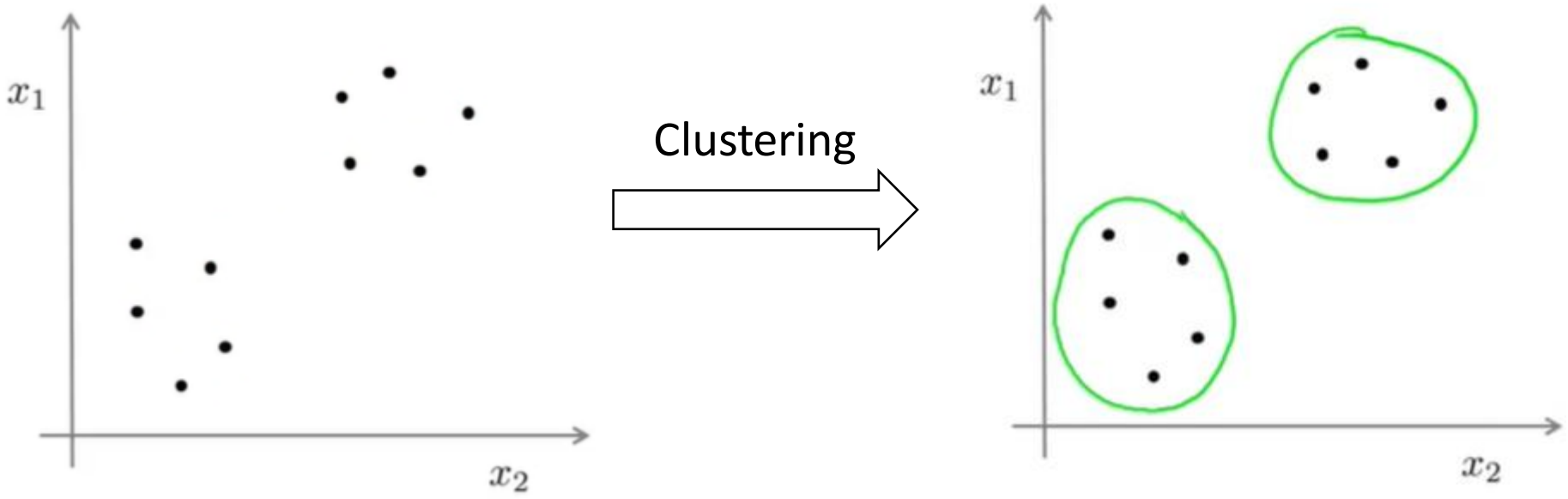
In supervised learning, we are given with a labeled training set and our goal is to fit a hypothesis to it to separate the positive and negative training examples.



Training Set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$

Unsupervised Learning

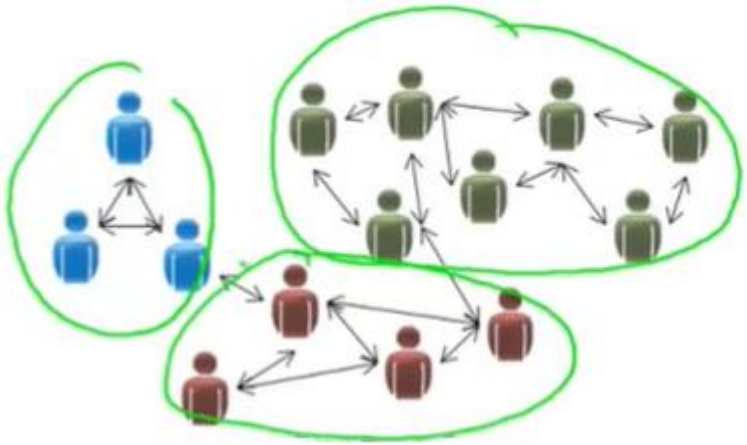
- In contrast to supervised learning, in the unsupervised learning problem, we're given data that does not have any labels associated with it.
- Training Set: $\{x^{(1)}, x^{(2)}, x^{(3)}, \dots, x^{(m)}\}$
- We feed the dataset to an algorithm to find some "structures" in the data.
- An example of one such algorithm is "Clustering"



Applications of Unsupervised Learning:

1) Market Segmentation

2) Social Network Analysis



3) Organize Computing Clusters

4) Astronomical Data Analysis

Which of the following statements are true? Check all that apply.

- In unsupervised learning, the training set is of the form $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$ without labels $y^{(i)}$

- Clustering is an example of unsupervised learning.

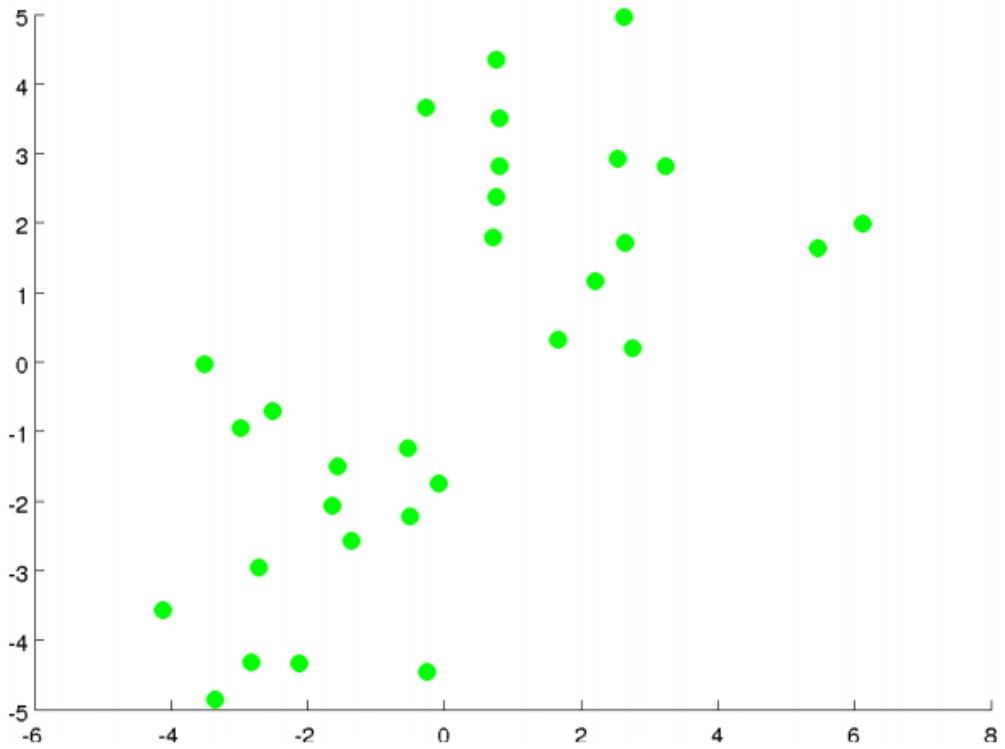
- In unsupervised learning, you are given an unlabeled dataset and are asked to find "structure" in the data.

- Clustering is the only unsupervised learning algorithm.

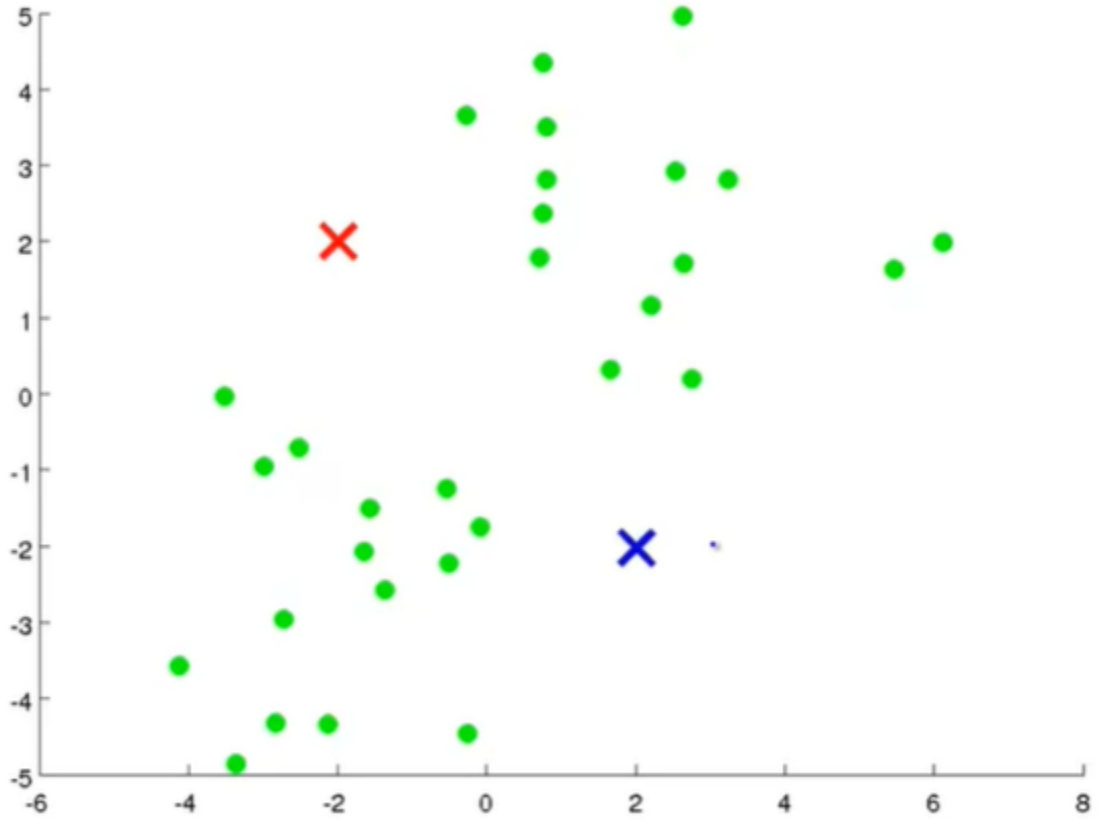
K-means Clustering

The K-means Clustering algorithm is best illustrated in pictures.

Let’s say we are given with the following dataset and want to group the data into two clusters.



To run the K-means Clustering algorithm on this data, the first step is to randomly initialize K data points called “cluster centroids”. Here, K=2 (for two clusters)

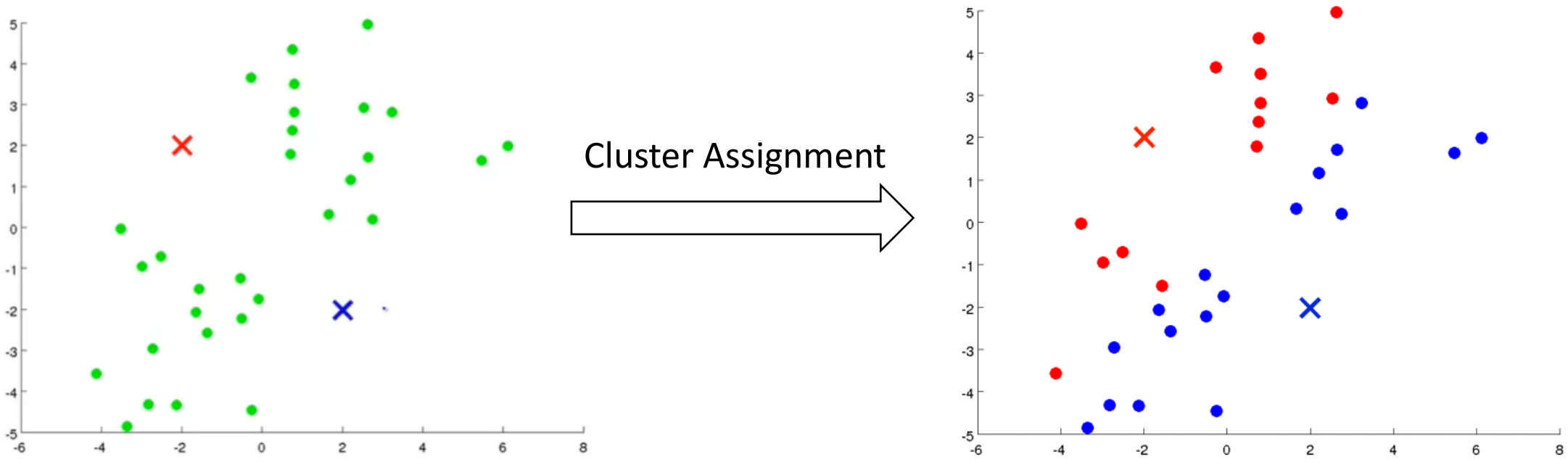


K-means Clustering is an iterative algorithm and it does two things:

- Cluster Assignment
- Move Centroid

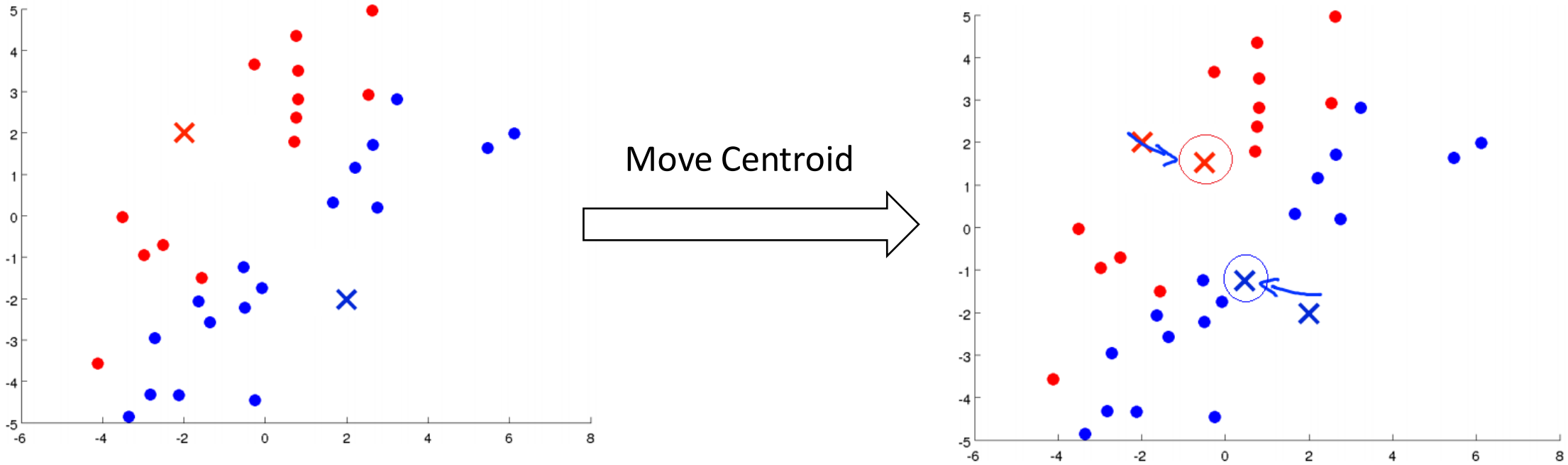
Cluster Assignment:

The algorithm goes through each of these examples shown in **green dots**, and depending on whether it is closer to **red cluster centroid** or the **blue cluster centroid,** it is going to assign each of the data points to one of the two clusters.

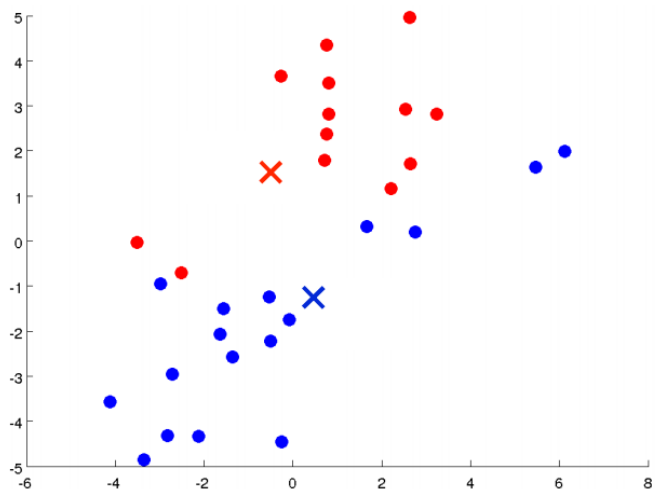


Move Centroid:

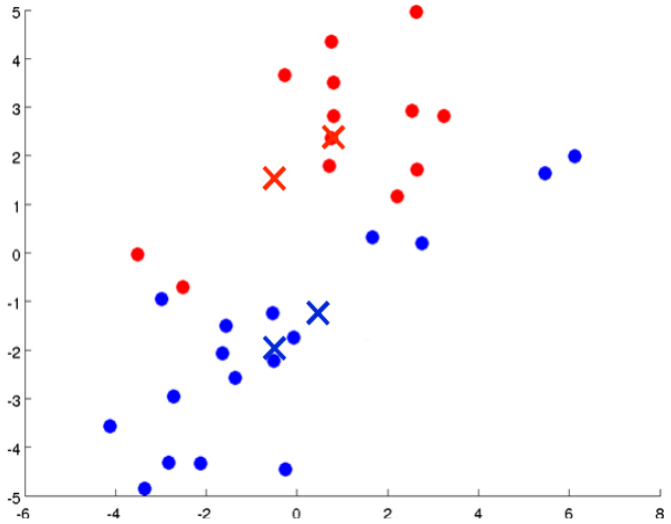
- We take the two cluster centroids and move them to the average of the points colored the same color.
- In other words, take all **red points** and compute the average and move the red cluster centroid to this new location.
- Similarly, take all **blue points** and compute the average and move the blue cluster centroid to this new location.



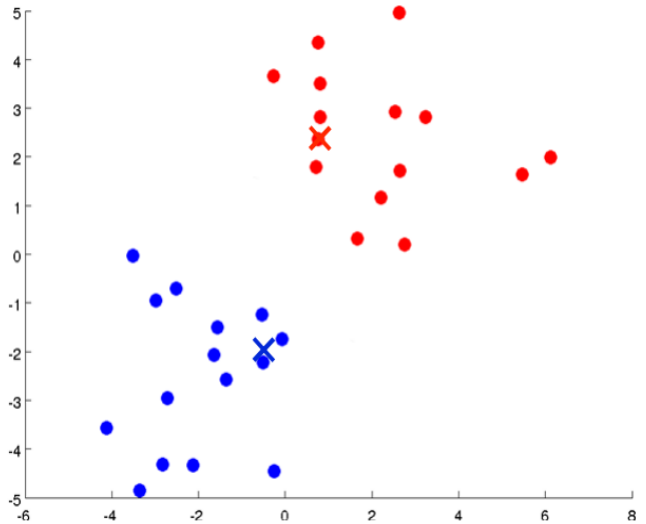
Repeat the steps until the cluster centroids can no longer be moved.



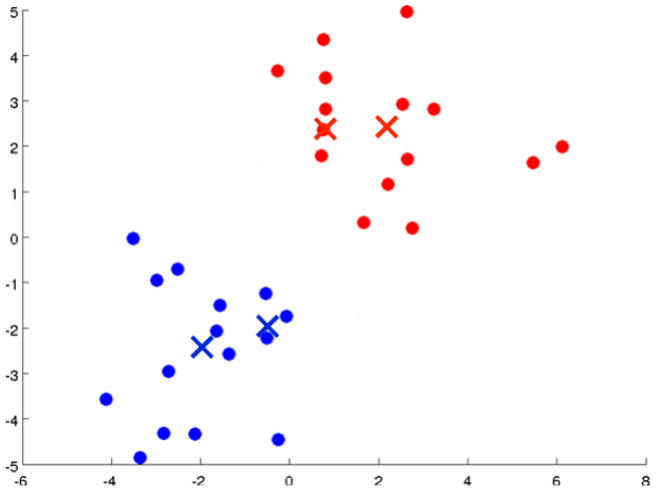
1st Iteration



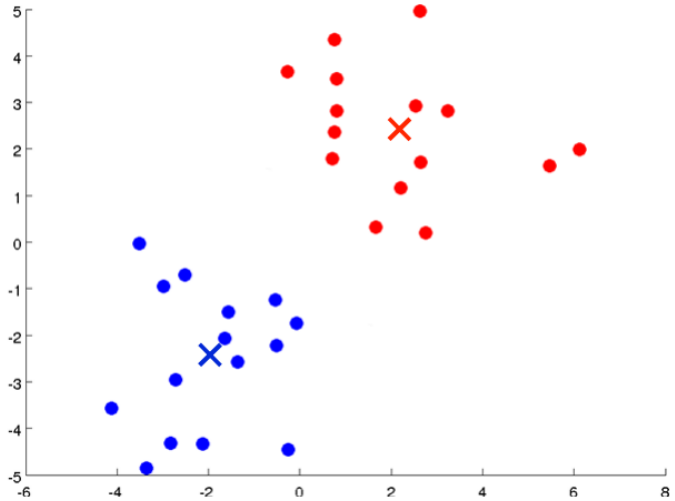
2nd Iteration



3rd Iteration



4th Iteration



5th Iteration

K-means Algorithm:

Input:

- K (no. of clusters)
- Training Set: $\{x^{(1)}, x^{(2)}, x^{(3)}, \dots, x^{(m)}\}$

Randomly initialize K cluster centroids $\mu_1, \mu_2, \dots, \mu_k \in R^n$

Repeat {

for i = 1 to m
c(i) := index (from 1 to K) of cluster centroid closest to x(i)

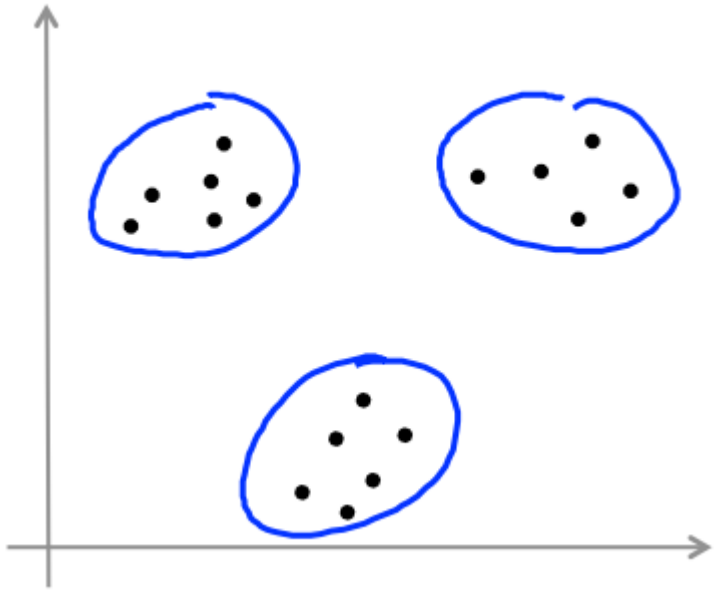
} Cluster Assignment

for k = 1 to K
μ_k := average (mean) of points assigned to cluster k

} Move Centroid

K-means for non-separated clusters

By looking at the following data, it is easy to predict that there are three clusters in the data.



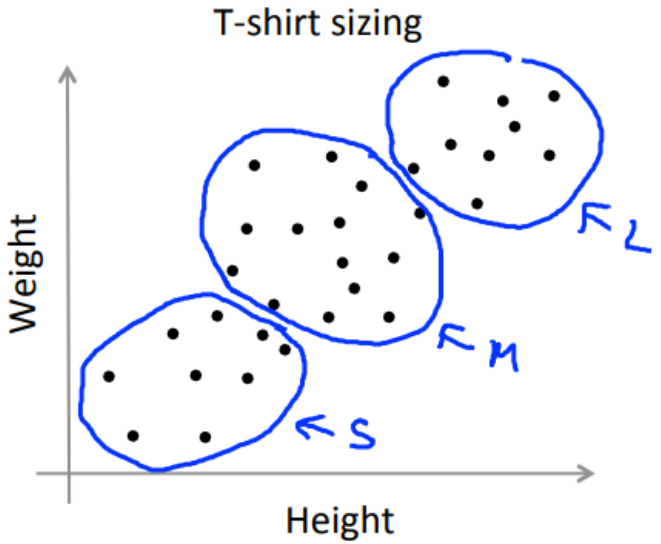
But what if the clusters are not well separated?

Consider an example of T-shirt sizing:

For this example, the clusters could be the T-shirt size.

- Small
- Medium
- Large

So, depending upon the height and weight, the K-means algorithm will separate the population into three clusters of small, medium and large T-shirt sizes.



Suppose you run k-means and after the algorithm converges, you have: $c^{(1)} = 3$, $c^{(2)} = 3$, $c^{(3)} = 5$ …

Which of the following statements are true? Check all that apply.

- The third example $x^{(3)}$ has been assigned to cluster 5.
- The first and second training examples $x^{(1)}$, $x^{(2)}$ have been assigned to the same cluster.
- The second and third training examples have been assigned to the same cluster.
- Out of all the possible values of k ∈ {1, 2, … , K}, the value k=3 minimizes $|| x^{(2)} - \mu_k ||^2$

K-means Clustering: Optimization Objective

$c^{(i)}$ = index of cluster (1, 2, … , K) to which example x(i) is currently assigned.

μ_k = cluster centroid k ($\mu_k \in R^n$)

$\mu_c^{(i)}$ = cluster centroid of cluster to which example $x^{(i)}$ has been assigned.

Optimization Objective:

The cost function a.k.a. "distortion" of K-means clustering is given by,

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

Objective is to find the minimum of this cost function.

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

Hence, we try to find the value of c and μ such that the cost function is minimized.

What does the algorithm actually do?

K cluster centroids $\mu_1, \mu_2, \dots, \mu_k \in R^n$

Repeat {

for i = 1 to m
c(i) := index (from 1 to K) of cluster centroid closest to x(i)

Minimizes J(...) w.r.t $c^{(1)}, c^{(2)}, \dots, c^{(m)}$ with fixed $\mu_1, \mu_2, \dots, \mu_k$

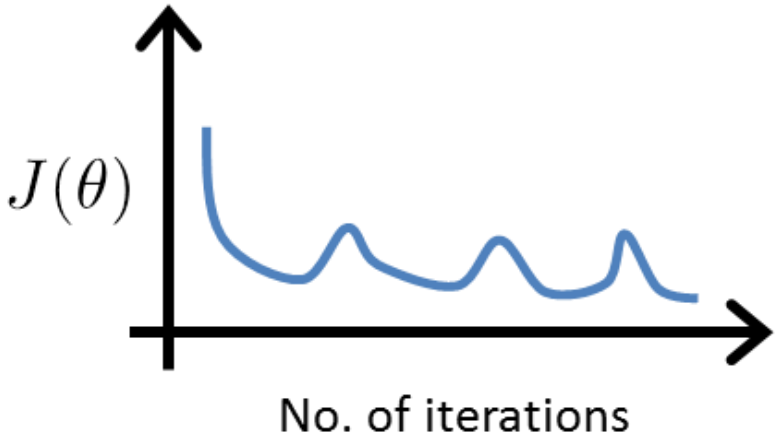
for k = 1 to K
μ_k := average (mean) of points assigned to cluster k

Minimizes J(...) w.r.t $\mu_1, \mu_2, \dots, \mu_k$

- **Cluster Assignment:**
  Minimizes cost function w.r.t $c^{(1)}, c^{(2)}, \dots c^{(m)}$ with fixed $\mu_1, \mu_2, \dots, \mu_k$

- **Move Centroid:**
  Minimizes cost function w.r.t $\mu_1, \mu_2, \dots, \mu_k$

Suppose you have implemented k-means and to check that it is running correctly, you plot the cost function $J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_k)$ as a function of the number of iterations. Your plot looks like this:



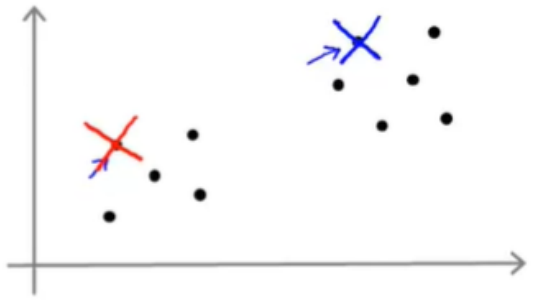
What does this mean?

- The learning rate is too large.
- The algorithm is working correctly.
- The algorithm is working, but k is too large.
- It is not possible for the cost function to sometimes increase. There must be a bug in the code.

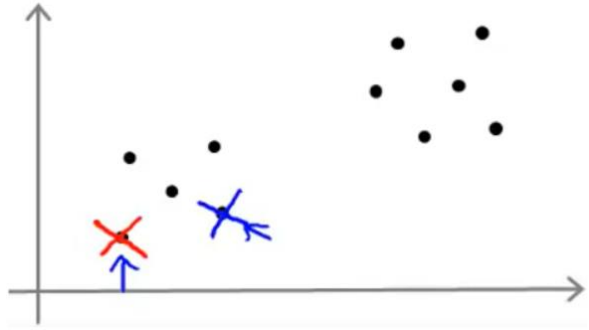
K-means Clustering: Random Initialization

Random Initialization:

- The no. of clusters (K) should be less than no. of training examples (m).
- Randomly pick K training examples.



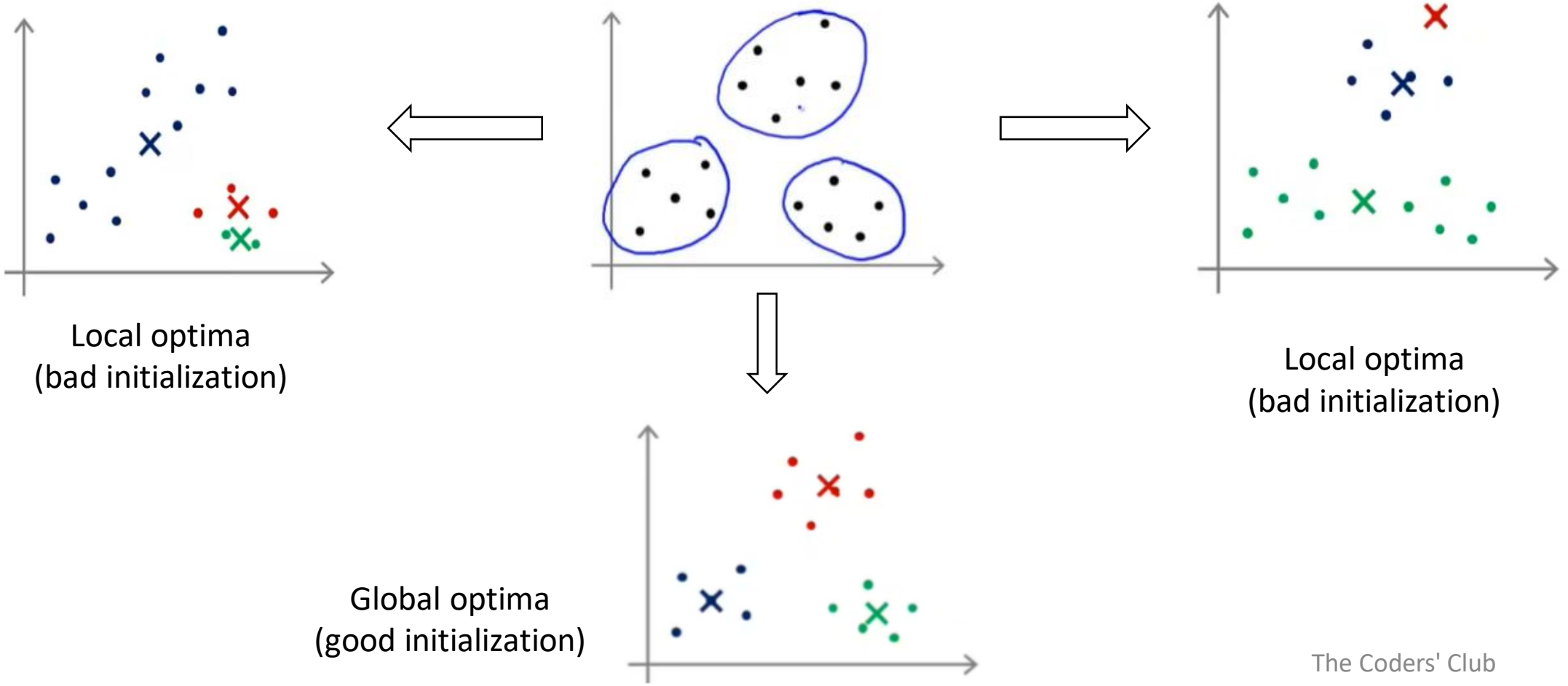
- Set $\mu_1, \dots, \mu_k$ equal to these K examples.



Local Optima:

K-means can end up converging to different solutions depending on how the clusters were initialized.

Example: Consider the dataset below.



Local optima
(bad initialization)

Local optima
(bad initialization)

Global optima
(good initialization)

- To avoid the possibility of local optima, we should initialize and run K-means lots of times.
- With each initialization, K-means would end up converging to a different solution.
- Compute the cost of each solution obtained and choose the clustering having minimum distortion.

Random Initialization:

for i = 1 to 100 {

Randomly initialize K-means
Run K-means. Get $c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_k$
Compute cost function (distortion)
$J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_k)$

}

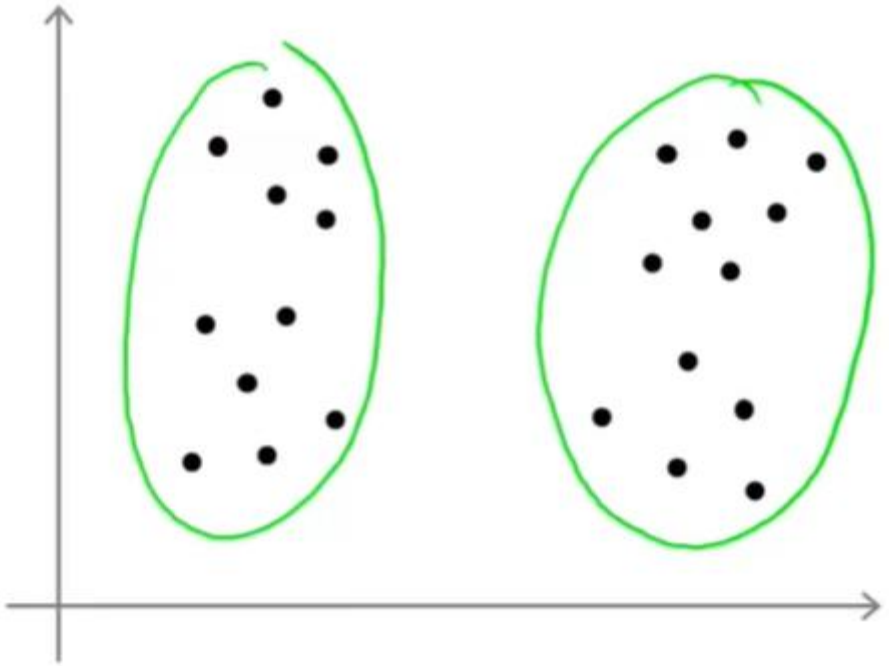
Pick clustering that gave lowest $J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_k)$

Which of the following is the recommended way to initialize k-means?

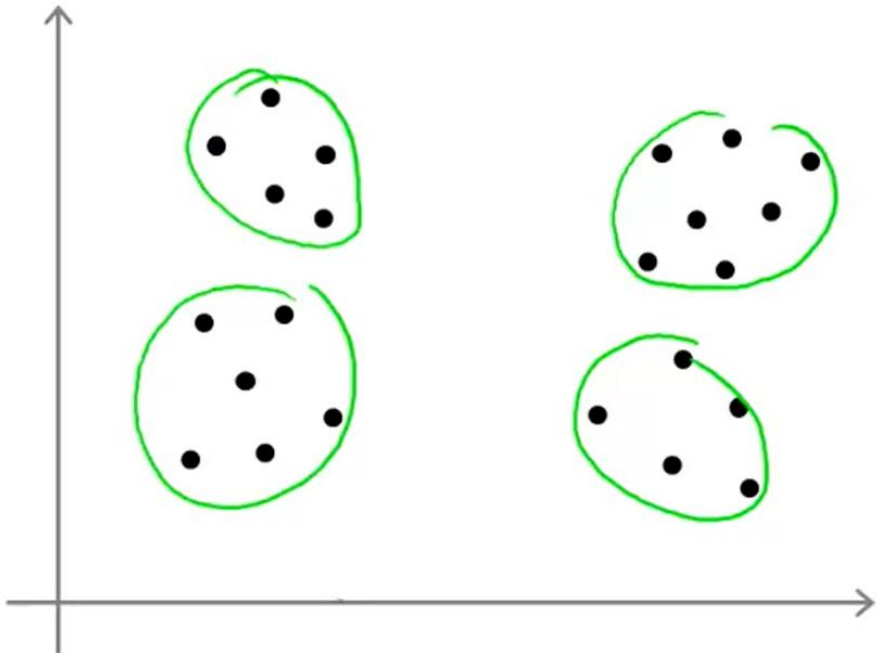
- Pick a random integer i from {1, … k}. Set $\mu_1 = \mu_2 = \ldots = \mu_k = x^{(i)}$
- Pick k distinct random integers $i_1, \ldots i_k$ from {1, … k}.
  Set $\mu_1 = x^{(i1)}, \mu_2 = x^{(i2)}, \ldots, \mu_k = x^{(ik)}$
- Pick k distinct random integers $i_1, \ldots i_k$ from {1, … m}.
  Set $\mu_1 = x^{(i1)}, \mu_2 = x^{(i2)}, \ldots, \mu_k = x^{(ik)}$
- Set every element of $\mu_1 \in R^n$ to a random value between $-\varepsilon$ and ε, for some small ε

K-means Clustering: Choosing the no. of clusters

What is the right value of K?

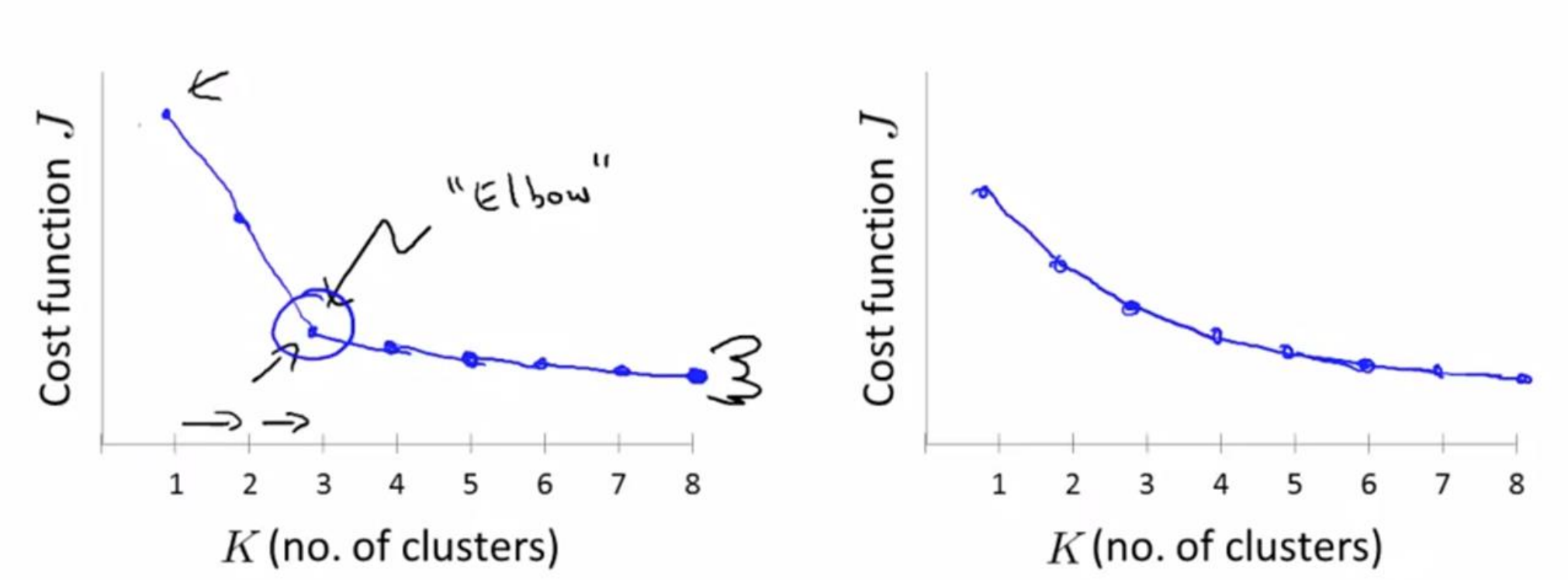


OR



Choosing the value of K

Elbow Method:

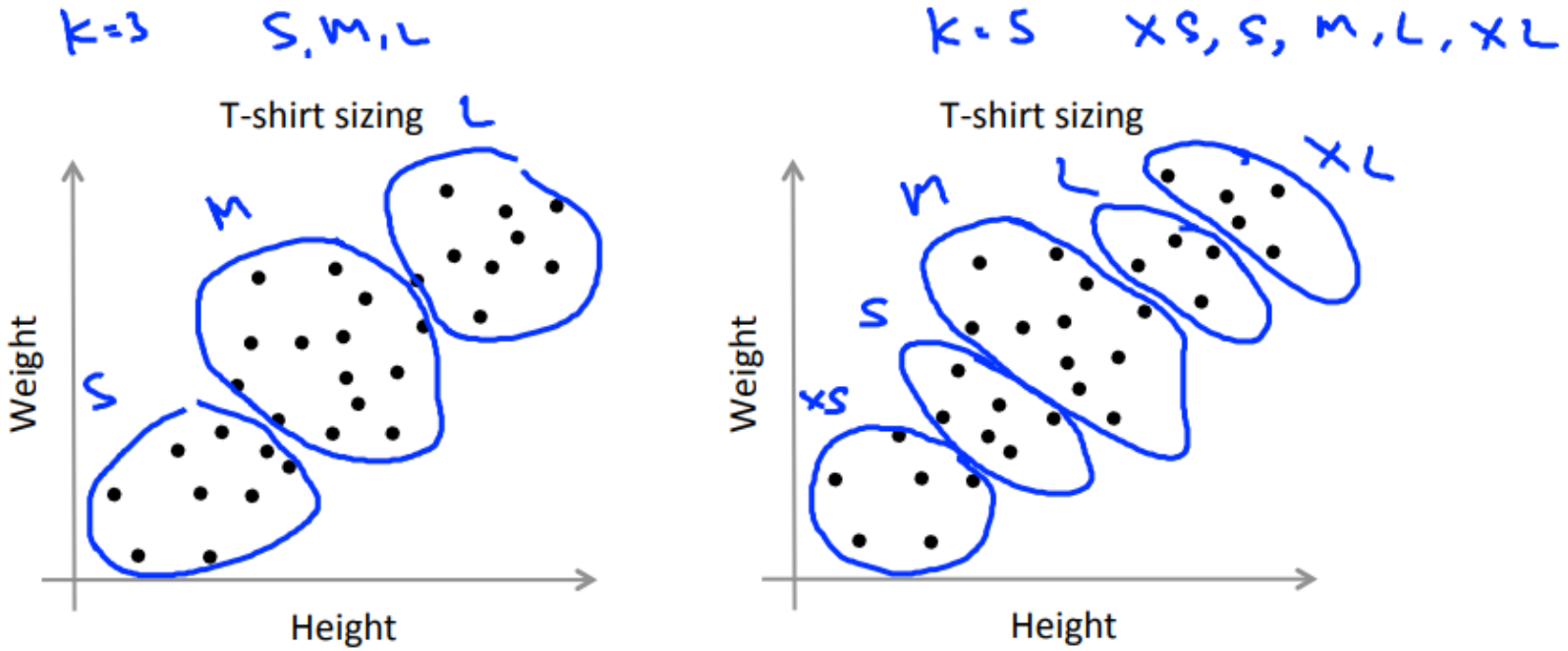


- The value of K that corresponds to the “elbow” of the curve should be chosen.
- However, Elbow Method is not used that often because sometimes, the curve obtained can be of the form as shown in the RHS. There is no specific “elbow” here that would decide the value of K to be chosen.

Choosing the value of K

Sometimes, you're running K-means to get clusters to use for some future purpose. Evaluate K-means based on a metric for how well it performs for that future purpose.

Example: Market segmentation of T-shirt sizing



Hence, the better way to think about how to choose the no. of clusters is to ask, for what purpose are you running K-means?

Suppose you run k-means using k = 3 and k = 5. You find that the cost function J is much higher for k = 5 than for k = 3. What can you conclude?

- This is mathematically impossible. There must be a bug in the code.
- The correct number of clusters is k = 3
- In the run with k = 5, k-means got stuck in a bad local minimum. You should try re-running k-means with multiple random initializations.
- In the run with k = 3, k-means got lucky. You should try re-running k-means with k = 3 and different random initializations until it performs no better than k = 5.

Dimensionality Reduction: Data Compression

Reduce data from 2D to 1D

Consider the dataset as shown.

The plot consists of two coordinates (features) –

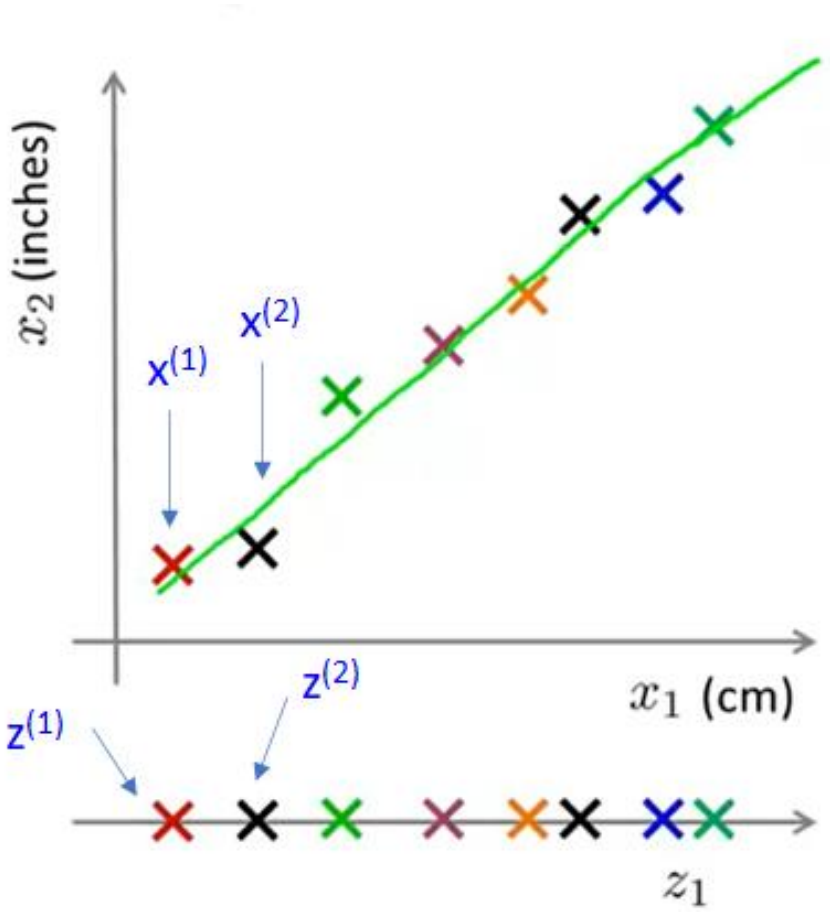
- x_1 (cm)
- x_2 (inches)

This can be reduced to one single feature by adding a new axis z_1 that is able to represent the examples in one dimension.

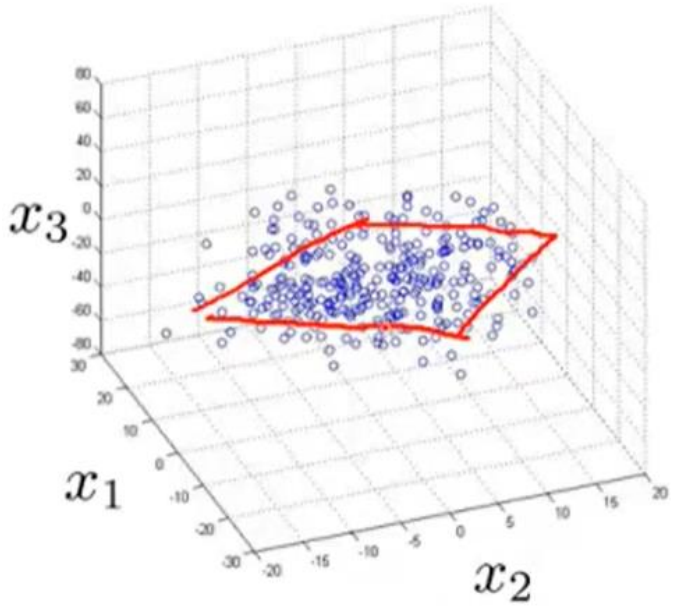
Thus, $x^{(1)} \in R^2 \longrightarrow z^{(1)} \in R$

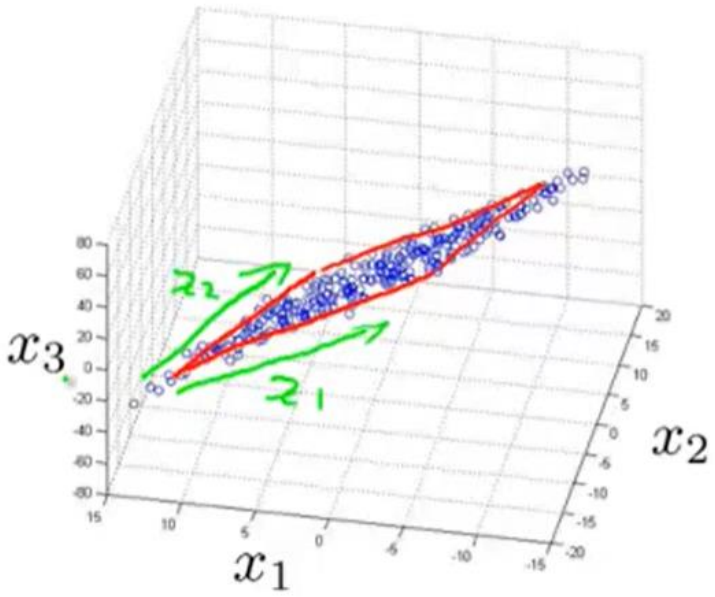
$x^{(2)} \in R^2 \longrightarrow z^{(2)} \in R$

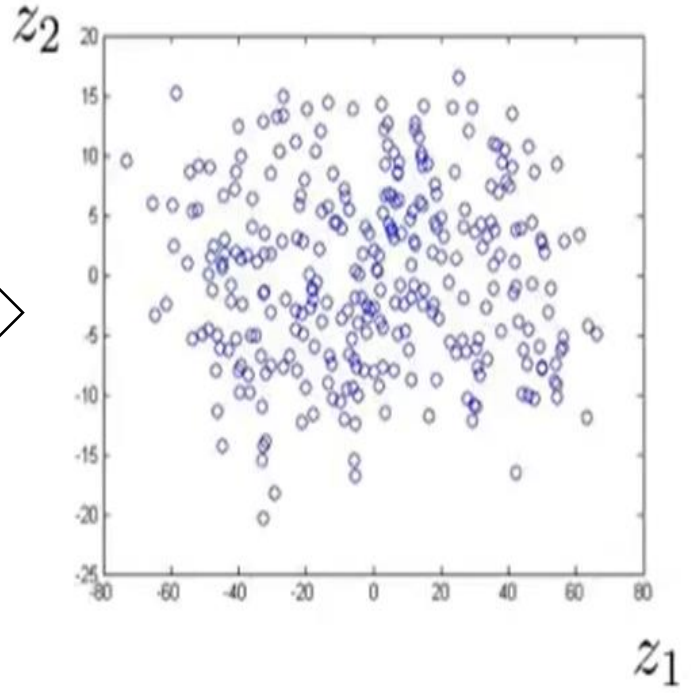
… and so on



Reduce data from 3D to 2D







Original Dataset in 3D

Projected Dataset

Projection of Dataset in 2D

Suppose we apply dimensionality reduction to a dataset of m examples $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$, where $x^{(i)} \in R^n$. As a result of this, we will get out:

- A lower dimensional dataset $\{z^{(1)}, z^{(2)}, \dots, z^{(k)}\}$ of k examples where $k \leq n$.
- A lower dimensional dataset $\{z^{(1)}, z^{(2)}, \dots, z^{(k)}\}$ of k examples where $k > n$.
- A lower dimensional dataset $\{z^{(1)}, z^{(2)}, \dots, z^{(m)}\}$ of m examples where $z^{(i)} \in R^k$ for some value of k and $k \leq n$.
- A lower dimensional dataset $\{z^{(1)}, z^{(2)}, \dots, z^{(m)}\}$ of m examples where $z^{(i)} \in R^k$ for some value of k and $k > n$.

Dimensionality Reduction: Data Visualization

Consider a dataset as shown:

Data Visualization

$x \in \mathbb{R}^{50}$ $x^{(i)} \in \mathbb{R}^{50}$

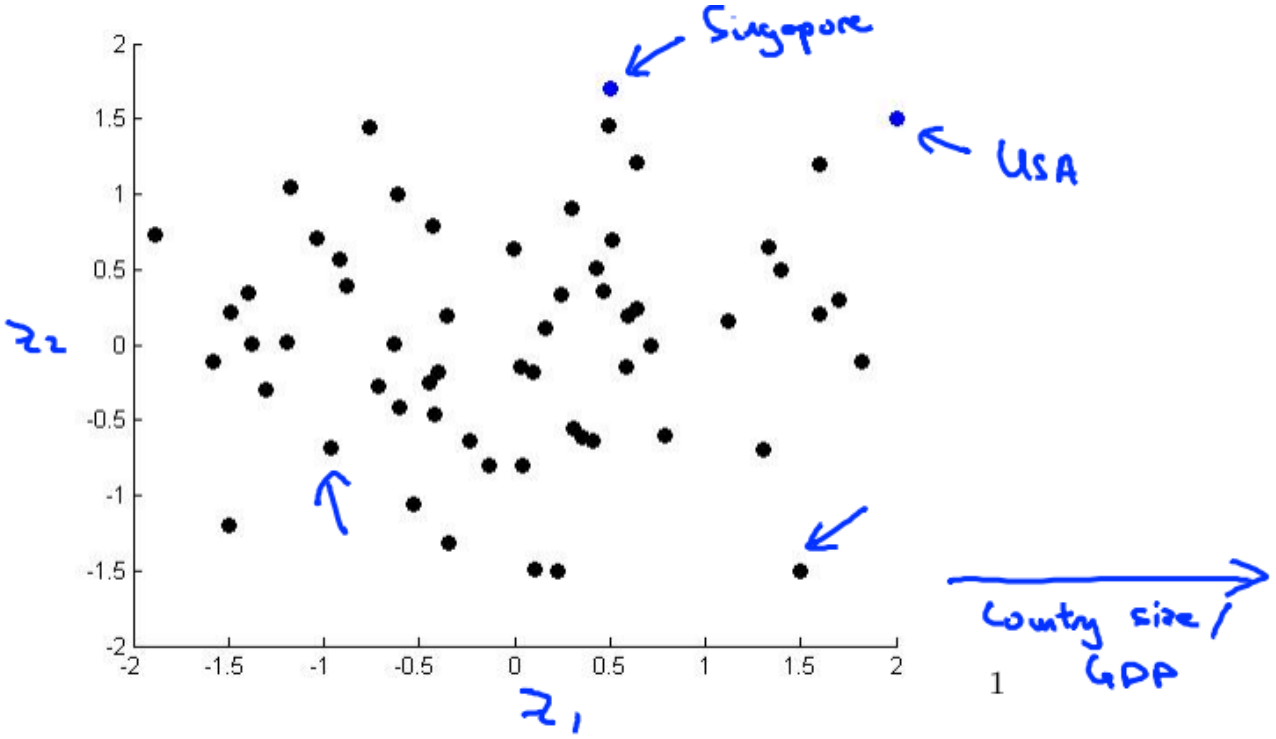
Country	x_1 GDP (trillions of US$)	x_2 Per capita GDP (thousands of intl. $)	x_3 Human Develop-ment Index	x_4 Life expectancy	x_5 Poverty Index (Gini as percentage)	x_6 Mean household income (thousands of US$)	...
Canada	1.577	39.17	0.908	80.7	32.6	67.293	...
China	5.878	7.54	0.687	73	46.9	10.22	...
India	1.632	3.41	0.547	64.7	36.8	0.735	...
Russia	1.48	19.84	0.755	65.5	39.9	0.72	...
Singapore	0.223	56.69	0.866	80	42.5	67.1	...
USA	14.527	46.86	0.91	78.3	40.8	84.3	...
...	...	...	...	...	...	...	

- There are about 50 features in this dataset.
- It is difficult to plot such dataset with so many features.

Hence, in order to visualize this data, we reduce it from 50D to 2D as follow:

Country	z_1	z_2
Canada	1.6	1.2
China	1.7	0.3
India	1.6	0.2
Russia	1.4	0.5
Singapore	0.5	1.7
USA	2	1.5
…	…	…

It is easier to plot this data now with only two axis – z_1 and z_2



- Each dot represents a country.
- The axis z_1 and z_2 may represent features such as GDP, life expectancy, poverty index, etc.
- Hence, data is better visualized by reducing the dimensions of dataset.

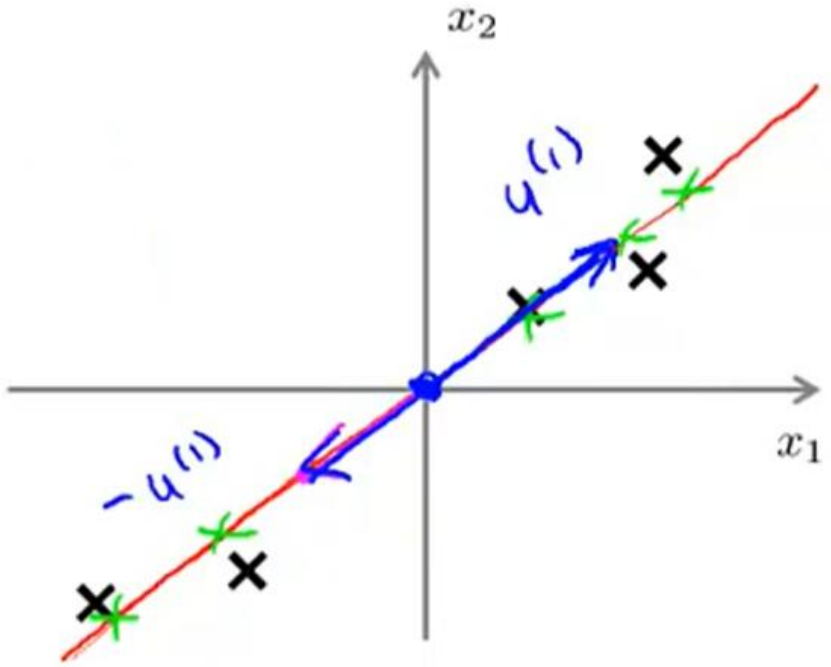
Suppose you have a dataset $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$ where $x^{(i)} \in R^n$. In order to visualize it, we apply dimensionality reduction and get $\{z^{(1)}, z^{(2)}, \dots, z^{(m)}\}$ where $z^{(i)} \in R^k$ is k-dimensional. In a typical setting, which of the following would you expect to be true? Check all that apply.

- $k > n$
- $k \leq n$
- $k \geq 4$
- $k = 2$ or $k = 3$ (since we can plot 2D or 3D data but don't have ways to visualize higher dimensional data)

Principal Component Analysis

Problem Formulation:

Reduce data from 2D to 1D: Find a direction (a vector $u^{(1)} \in R^n$) onto which to project the data so as to minimize the projection error.

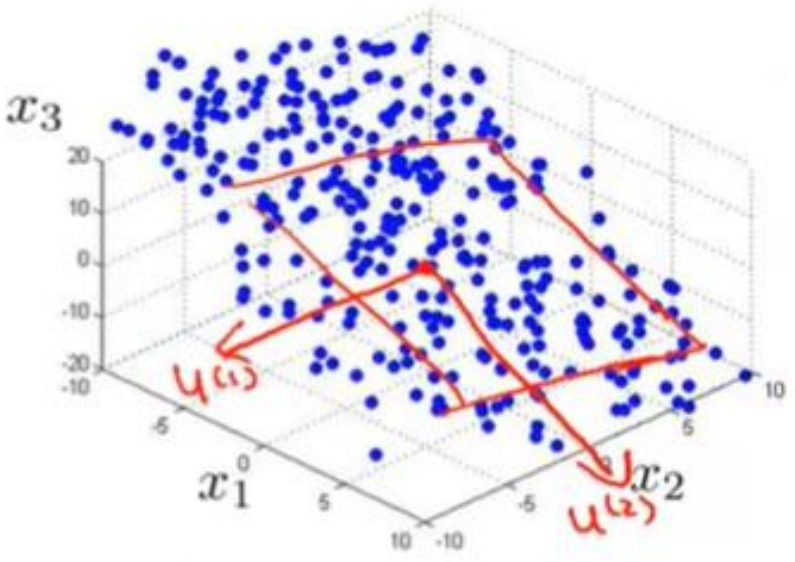


Reduce data from n-dimension to k-dimension:

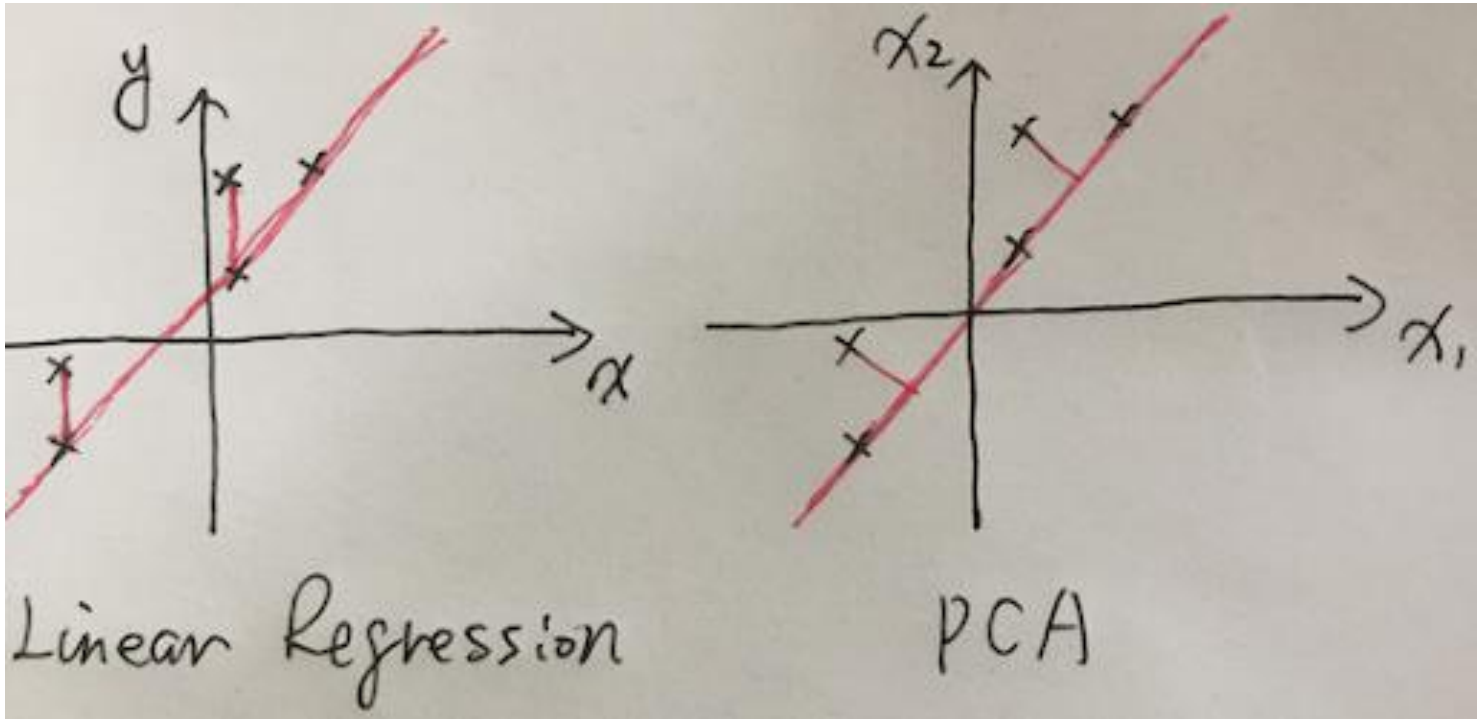
Find k vectors $u^{(1)}, u^{(2)}, \dots, u^{(k)}$ onto which to project the data so as to minimize the projection error.

Example: 3D to 2D

$$3D \rightarrow 2D$$
$$K = 2$$



PCA is not linear regression



- Linear Regression is used to fit a hypothesis in order to predict a dependent variable (y).
- PCA is used for dimensionality reduction.

Some References:
https://stats.stackexchange.com/questions/410516/using-pca-vs-linear-regression
https://shankarmsy.github.io/posts/pca-vs-lr.html

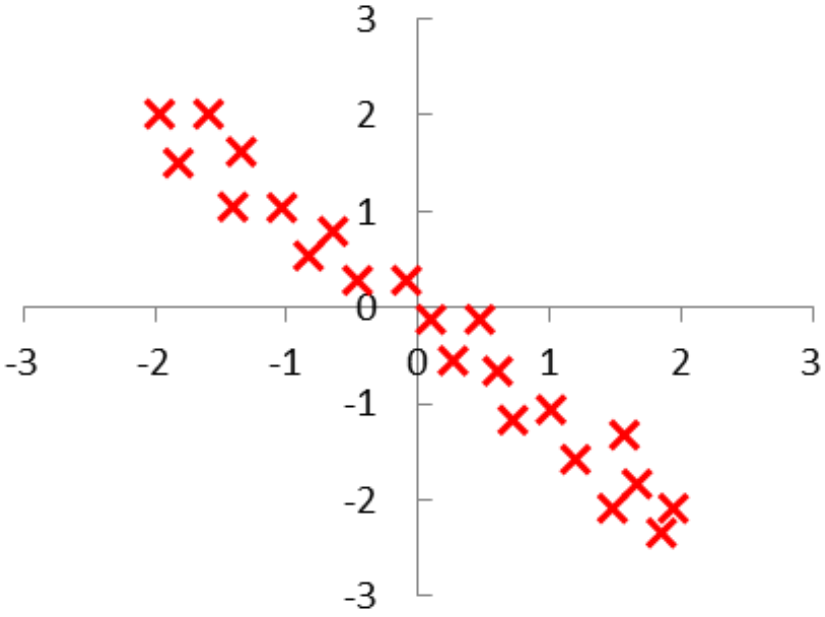
Suppose you run PCA on the dataset below. Which of the following would be a reasonable vector $u^{(1)}$ onto which to project the data? (By convention, we choose $u^{(1)}$ so that $||u^{(1)}|| = \sqrt{(u_1^{(1)})^2 + (u_2^{(1)})^2}$, the length of the vector $u^{(1)}$, equals 1).

$$u^{(1)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$u^{(1)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$u^{(1)} = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$$

$$u^{(1)} = \begin{bmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$$ ✓



Principal Component Analysis: Algorithm

Data Preprocessing:

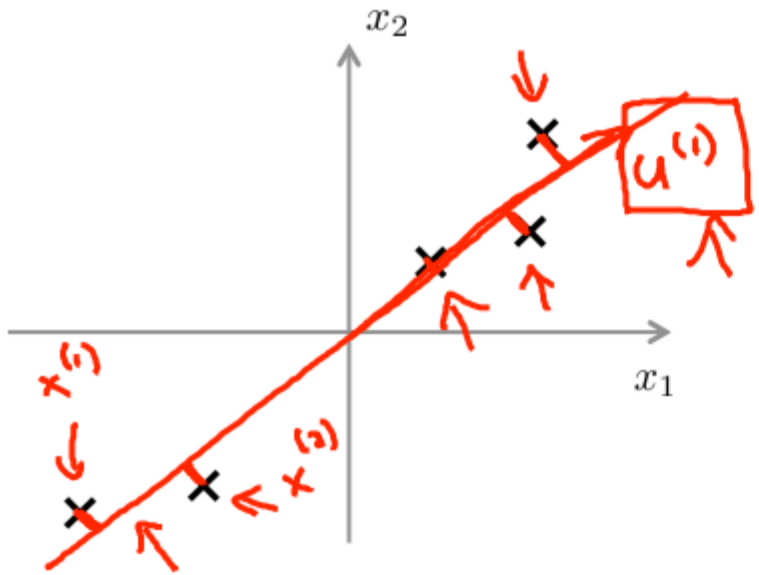
Training set: $x^{(1)}, x^{(2)}, \dots, x^{(m)}$

Preprocessing (feature scaling/mean normalization):

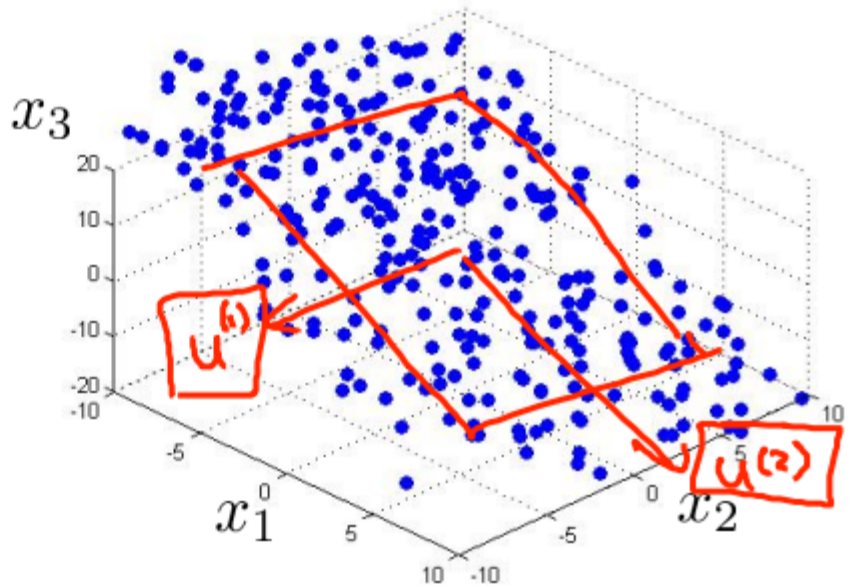
$$\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)}$$

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

If different features on different scales (e.g.: x_1 = size of house, x_2 = no. of bedrooms), scale features to have comparable range of values.



Reduce data from 2D to 1D



Reduce data from 3D to 2D

Algorithm:

Reduce data from n-dimensions to k-dimensions

Compute “covariance matrix”:

$$\Sigma = \frac{1}{m}\sum_{i=1}^{n}(x^{(i)})(x^{(i)})^T$$

Compute “eigen vectors” of matrix Σ

```
[U, S, V] = svd(Sigma);
```

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

```
Ureduce = U(: , 1 : k);
z = Ureduce’ * x;
```

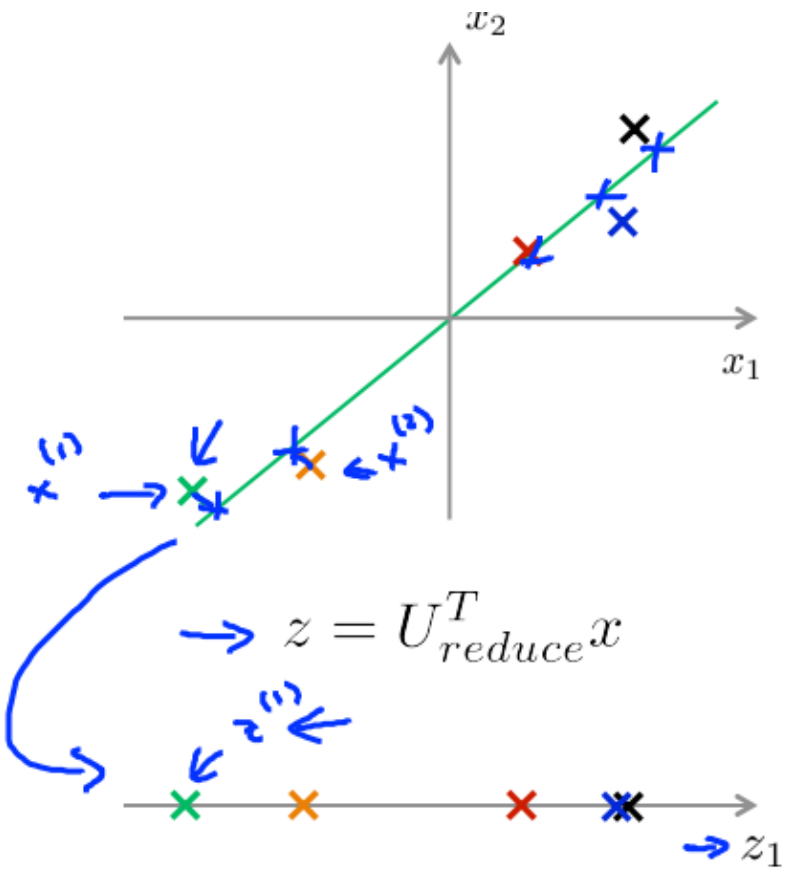
In PCA, we obtain $z \in R^k$ from $x \in R^n$ as follows:

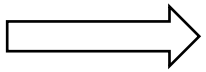
$$z = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(k)} \\ | & | & & | \end{bmatrix}^T x = \begin{bmatrix} --- & (u^{(1)})^T & --- \\ --- & (u^{(2)})^T & --- \\ & \vdots & \\ --- & (u^{(k)})^T & --- \end{bmatrix} x$$

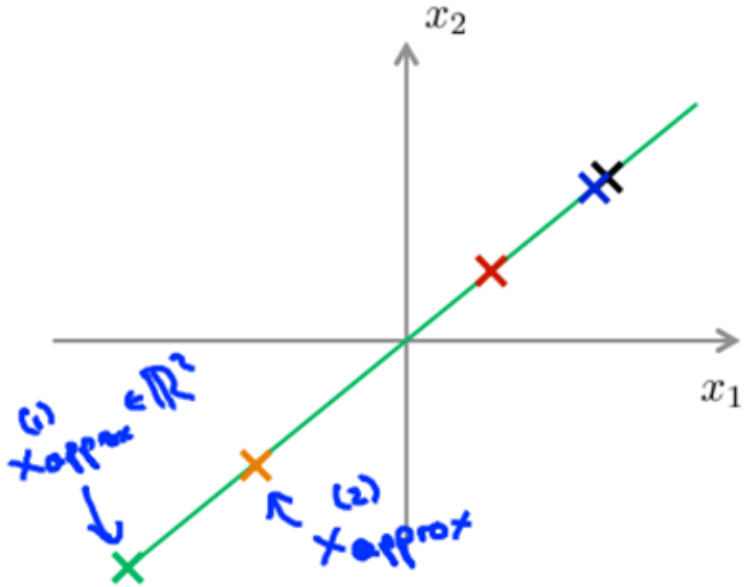
Which of the following is a correct expression for z_j ?

- $z_j = (u^{(k)})^T x$
- $z_j = (u^{(j)})^T x_j$
- $z_j = (u^{(j)})^T x_k$
- $z_j = (u^{(j)})^T x$

PCA: Reconstruction from Compressed Representation







Compressed Representation

$x \in R^2 \longrightarrow z \in R$

Reconstruction of original representation

$z \in R \longrightarrow x \in R^2$

Suppose we run PCA with k = n, so that the dimension of the data is not reduced at all. (This is not useful in practice but is a good thought exercise.) Recall that the percent / fraction of variance retained is given by:

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

Which of the following will be true? Check all that apply.

- U_{reduce} will be a n x n matrix
- x_{approx} = x for every example x
- The percentage of variance retained will be 100%
- We have that $\frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} > 1$

PCA: Choosing the Number of Principal Components

Choosing k (No. of Principal Components):

- Average squared projection error: $\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)} - x_{approx}^{(i)}\|^2$
- Total variation in the data: $\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)}\|^2$

Typically, choose k to be smallest value so that:

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

Value	Value in %	Variance retained in %
0.01	1	99
0.05	5	95
0.10	10	90

Algorithm:

Try PCA with k=1, k=2, k=3, … and so on, so that 99% of variance is retained.

Compute U_{reduce} , $z^{(1)}$, $z^{(2)}$, … , $z^{(m)}$, $x_{approx}^{(1)}$, … , $x_{approx}^{(m)}$

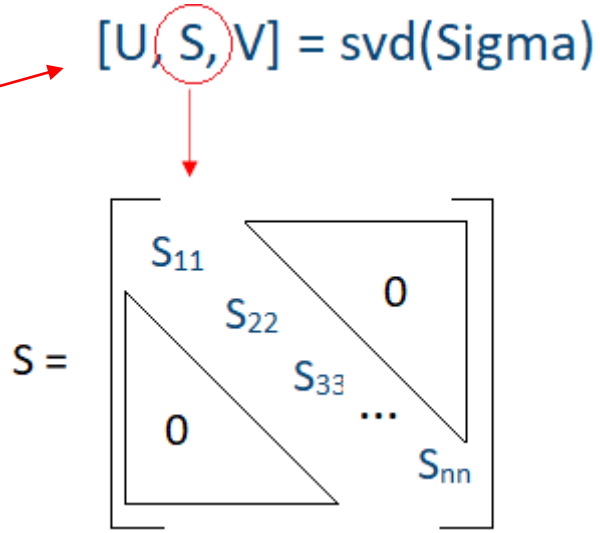
Check if $\frac{\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)} - x_{approx}^{(i)}\|^2}{\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)}\|^2} \leq 0.01$?

Alternate Method:

[U, S, V] = svd(Sigma)

Pick smallest value of k for which $\frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{m} S_{ii}} \geq 0.99$

(99% of variance retained)



Previously, we said that PCA chooses a direction $u^{(1)}$ (or k directions $u^{(1)}, \dots, u^{(k)}$) onto which to project the data so as to minimize the (squared) projection error. Another way to say the same is that PCA tries to minimize:

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

Advice for Applying PCA

PCA can be used to speed up a supervised learning algorithm.

Consider an example of a Computer Vision problem, where you have a 100x100 image. This means, you have 10,000 features which is very large.

- The training set is given by input X and label Y as follows:
  $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$
- We extract the inputs from this training set to form an unlabeled dataset and apply PCA to reduce the dimensions.

$x^{(1)}, x^{(2)}, \dots, x^{(m)} \in R^{10000}$

PCA

$z^{(1)}, z^{(2)}, \dots, z^{(m)} \in R^{1000}$

Note:
Mapping $x^{(i)} \longrightarrow z^{(i)}$ should be defined by running PCA only on the training set. This mapping can be applied as well to the examples $x_{cv}^{(i)}$ and $x_{test}^{(i)}$ in the cross-validation and test sets.

- This reduced dimension training set can now be fed into a learning algorithm (logistic regression/neural network) to form a hypothesis that would be able to make predictions.

Applications of PCA

Compression: Choose the value of k by % of variance retained.

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

Visualization: Choose k=2 or k=3

- To visualize a high-dimensional data by reducing its dimensions.
  Example: 2D plot of a 50-D data.
  3D plot of a 100-D data

Bad use of PCA:

PCA should not be used to prevent overfitting.

- It is possible to use PCA and make use of $z^{(i)}$ instead of $x^{(i)}$ in order to reduce the number of features to k<n.
- Thus, fewer features are less likely to overfit.
- However, this might work OK but isn't a good way to address overfitting.
- Use regularization instead.

$$\min_{\theta} \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

PCA is sometimes used where it shouldn't be:

Design of ML system:

- Get training set $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$
- Run PCA to reduce $x^{(i)}$ in dimension to get $z^{(i)}$
- Train logistic regression on $\{(z^{(1)}, y^{(1)}), (z^{(2)}, y^{(2)}), \dots, (z^{(m)}, y^{(m)})\}$
- Test on test set: Map $x_{test}^{(i)}$ to $z_{test}^{(i)}$. Run $h_\theta(z)$ on $\{(z_{test}^{(1)}, y_{test}^{(1)}), \dots, (z_{test}^{(m)}, y_{test}^{(m)})\}$

How about doing the whole thing without using PCA?

Before implementing PCA, first try running whatever you want to do with the original/raw data $x^{(i)}$. Only if that doesn't do what you want, then implement PCA and consider using $z^{(i)}$.

Which of the following are good / recommended applications of PCA? Select all that apply.

- To compress the data so it takes up less computer memory/disk space.
- To reduce the dimension of the input data so as to speed up a learning algorithm.
- Instead of using regularization, use PCA to reduce the number of features to reduce overfitting.
- To visualize high-dimensional data (by choosing k=2 or k=3)