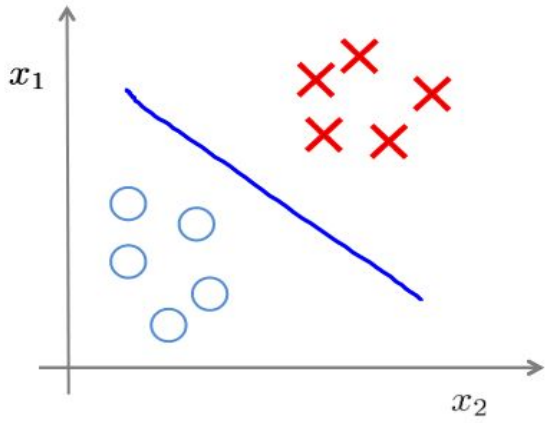
# Week : 8

# First part : Unsupervised learning

## Clustering :

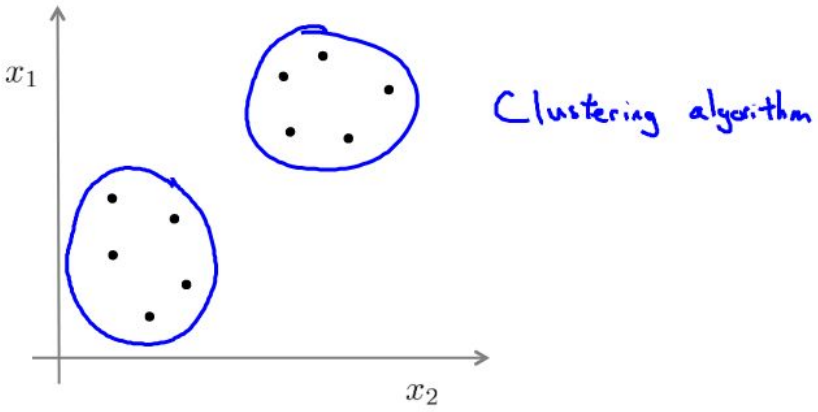
In supervised learning we had lebel with every training set. But now in unsupervised learning we have some unlabeled training sets and we need to classify them based on some logic.

**Supervised learning**




Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(3)}, y^{(3)}), \ldots, (x^{(m)}, y^{(m)})\}$

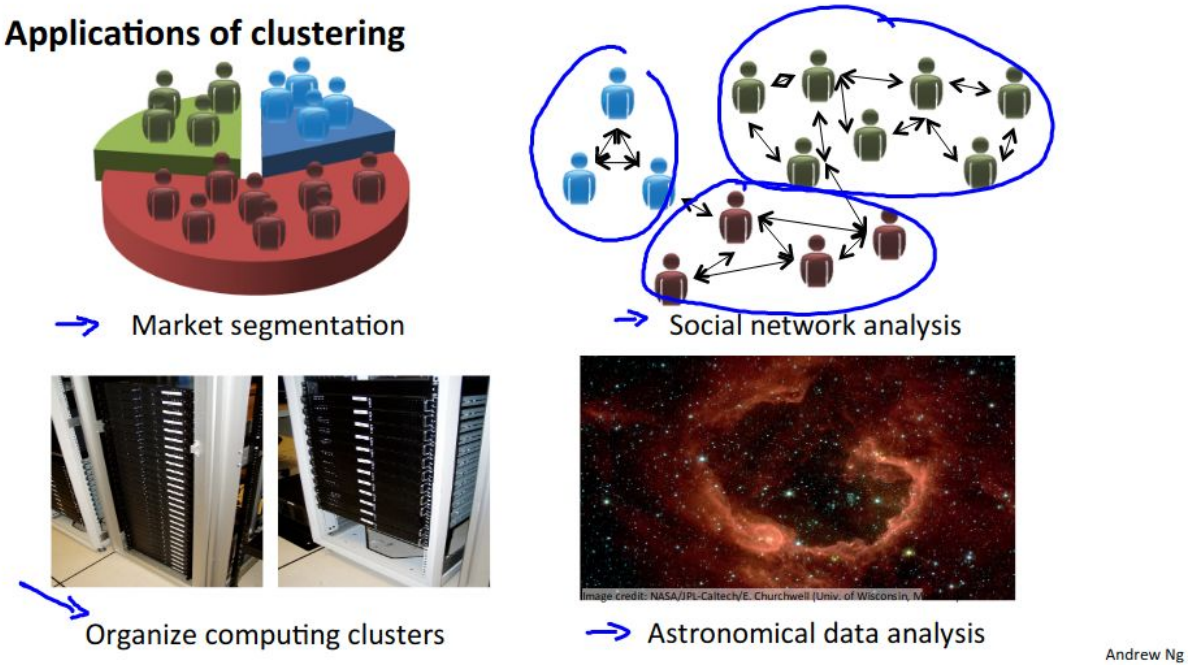
**Unsupervised learning**




Training set: $\{x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(m)}\}$

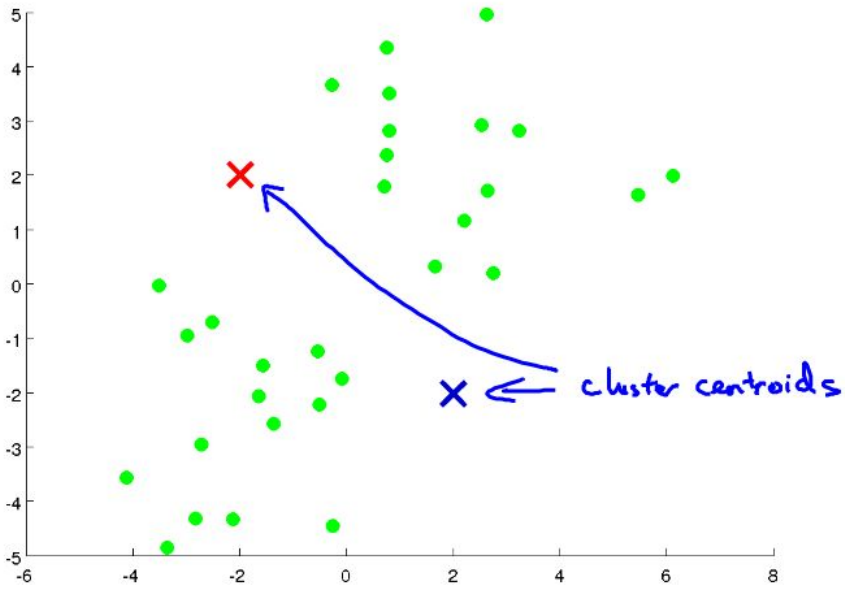
Andrew Ng

There are many uses of unsupervised learning in the field of social network, online marketing, server system and so on.

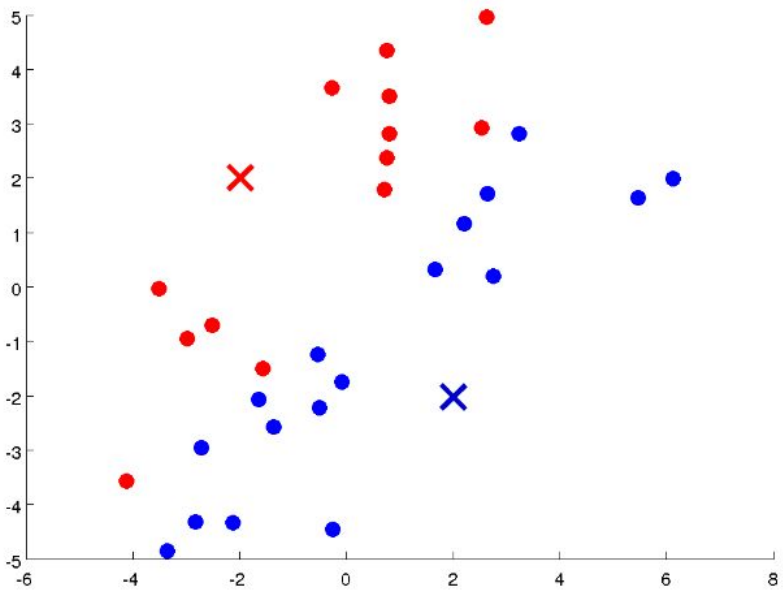


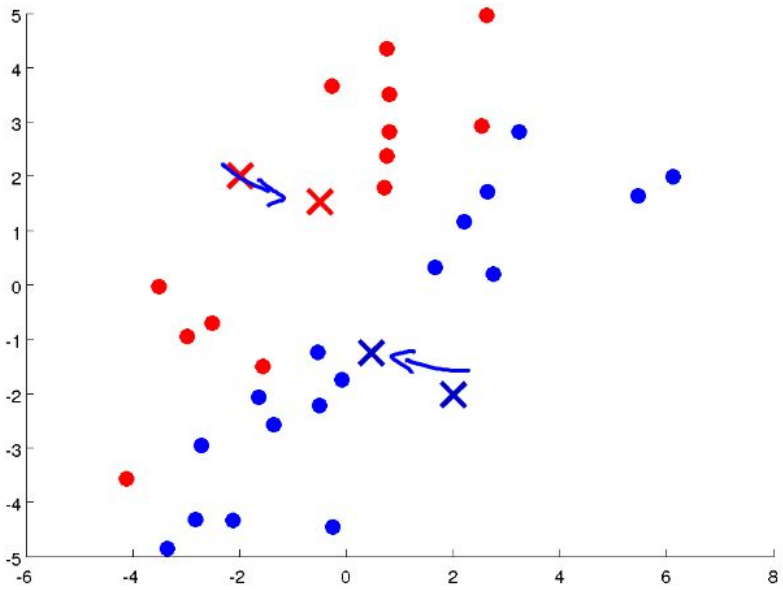
# K-means algorithm :

In K-means algorithm there are two steps : first one is cluster assignment step and the second one is move centroid step. We need to do these two steps repeatedly to get perfect clusters. For the cluster assignment step we need to assign K centroid randomly where K is the number of clusters. Then we need to assign training sets to a centroid. Afterwarth for every centroid we calculate the mean of all training sets in one centroid and move it to that mean position. We need to repeat these task for every K centroid and whole steps for a certain time until we get a perfect result of clustering .

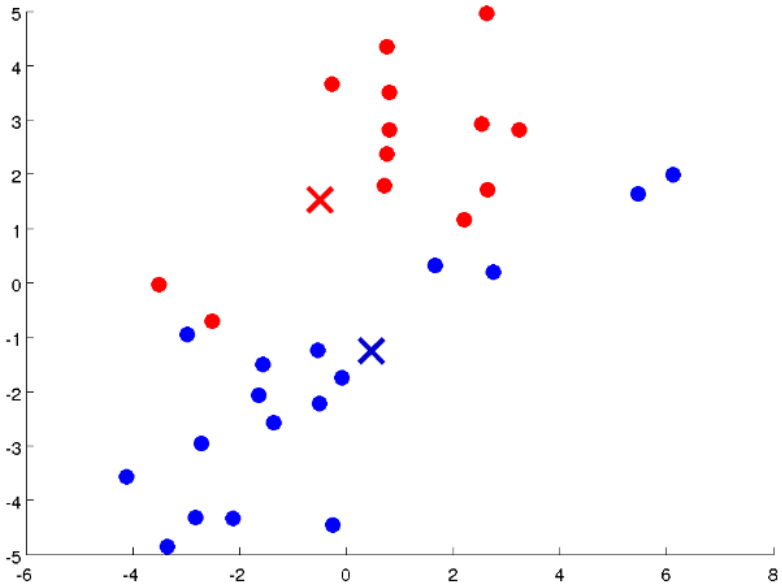


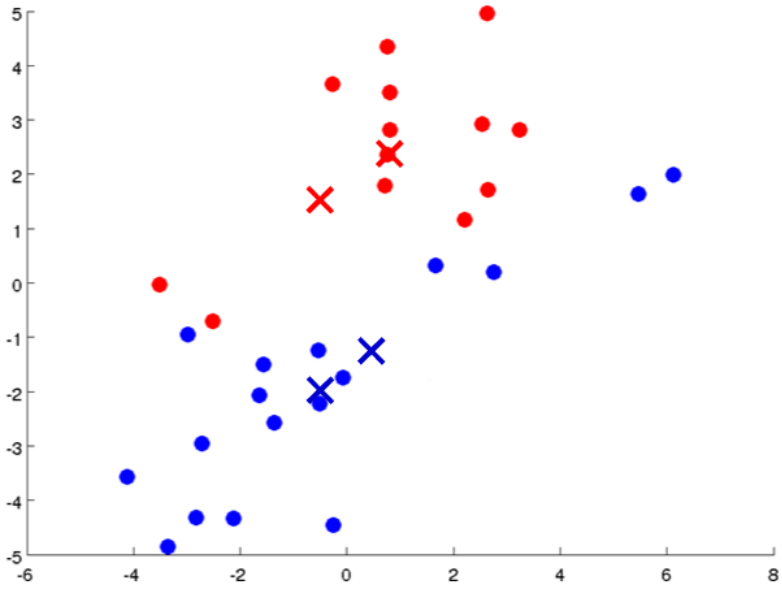

Andrew Ng



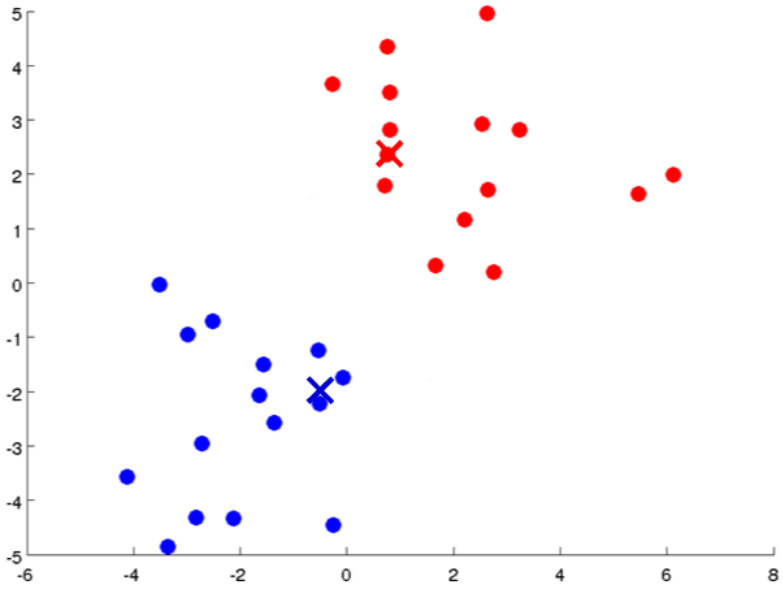



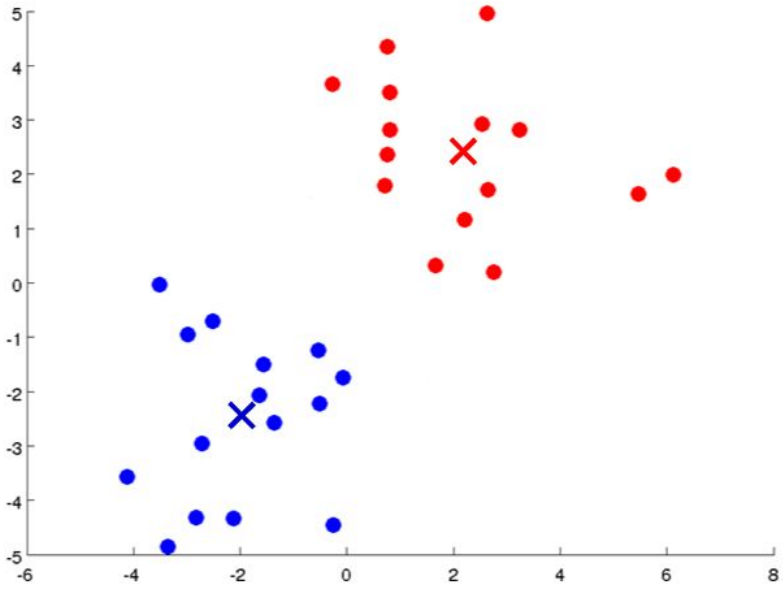

Andrew Ng




Andrew Ng


5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8


5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8

To perform K-mean we need two inputs : K(number of clusters) and the dataset.

## K-means algorithm

Input:

- $K$ (number of clusters)
- Training set $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$

$x^{(i)} \in \mathbb{R}^n$ (drop $x_0 = 1$ convention)

## K-means algorithm

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

for $k$ = 1 to $K$

$\mu_k$ := average (mean) of points assigned to cluster $k$

}

Andrew Ng

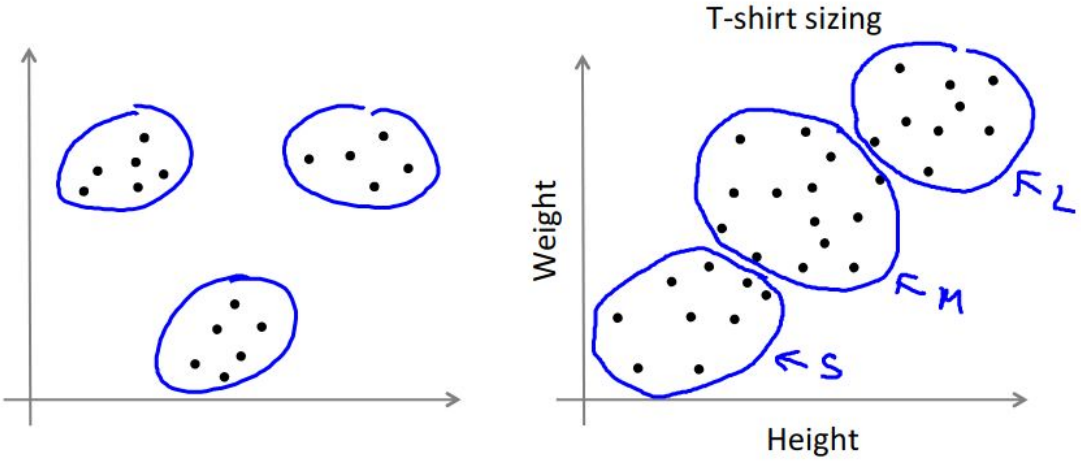
Rarely it happens that one centroid has no point assigned. In that case we have two ways to solve that problem.

1. We can remove this centroid and continue our algorithm.
2. We can assign that centroid randomly again.

We can run K-means for non-separated clusters and will get accurate results.

## K-means for non-separated clusters

S, M, L




Andrew Ng

# Optimisation objective :

### K-means optimization objective

- $c^{(i)}$ = index of cluster (1,2,…,$K$) to which example $x^{(i)}$ is currently assigned
- $\mu_k$ = cluster centroid $k$ ($\mu_k \in \mathbb{R}^n$)

K  $k \in \{1, 2, \ldots, K\}$

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

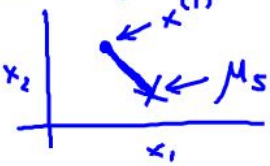
$x^{(i)} \rightarrow 5$  $c^{(i)} = 5$  $\mu_{c^{(i)}} = \mu_5$

Optimization objective:

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

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

Distortion




Andrew Ng

**K-means algorithm**

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

Repeat {

for $i$ = 1 to $m$

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

for $k$ = 1 to $K$

$\mu_k$ := average (mean) of points assigned to cluster $k$

}

Andrew Ng

# Random initialization :

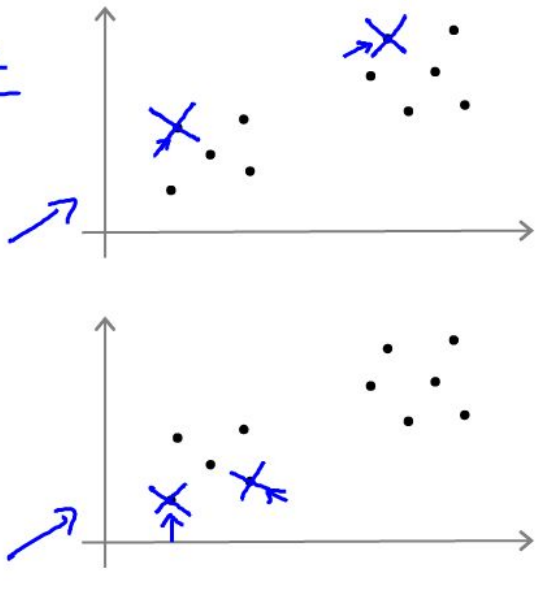
To initialize centroid we can follow random initialization technique. In this technique we will choose K points from m number of data randomly.

**Random initialization**

Should have $K < m$

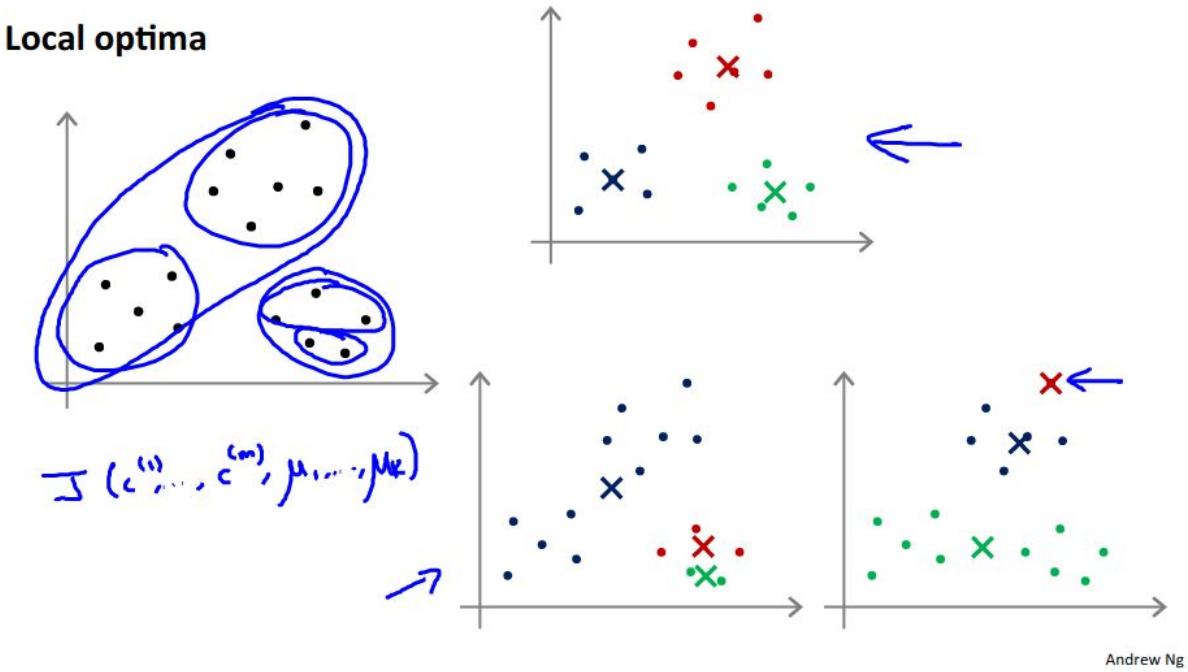
Randomly pick $K$ training examples.

Set $\mu_1, \ldots, \mu_K$ equal to these $K$ examples.



Andrew Ng

Sometimes after performing K-means we may get stuck in local optimum. Here are some example of local optimum .




## Random initialization

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

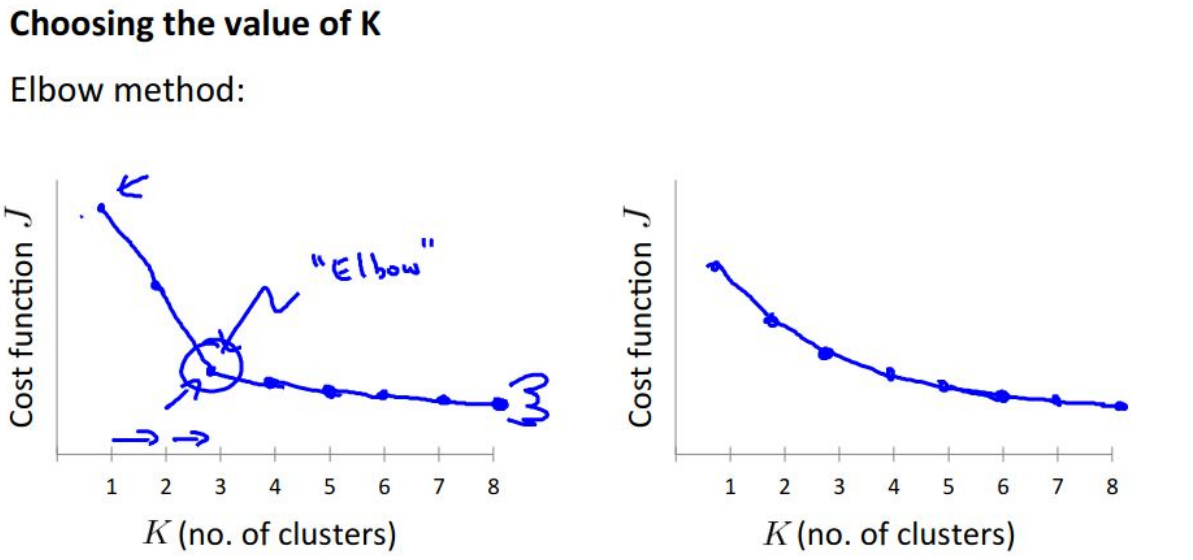
Pick clustering that gave lowest cost $J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K)$

Andrew Ng

# Choosing the number of clusters:

It is good to set the number of clusters manually. But we have an algorithm which can help you in some cases.

The name of the method is elbow method. Here we will plot a graph cost vs number of clusters and hope to get an elbow in this graph. And the elbo is the number of our clusters.



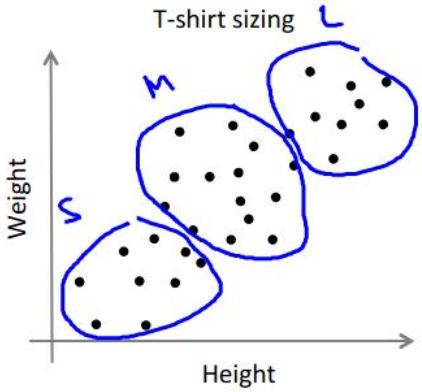

But in all cases it can't help us. In the picture above we can see in graph no 2 there is no elbow. So it is not always applicable.

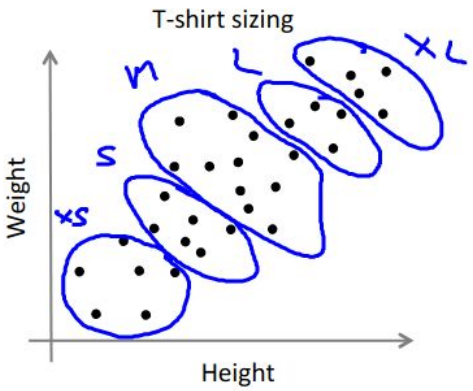
We can choose the number of clusters manually rather than by algorithm. It will be more practical and realistic.

**Choosing the value of K**

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

E.g.



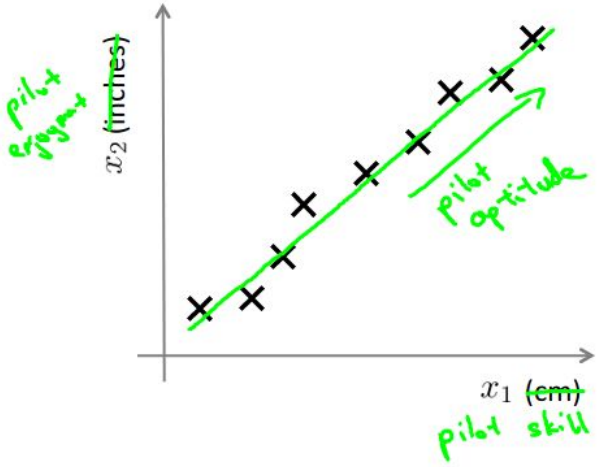




Andrew Ng

# Part two

Dimensionality reduction :
Sometimes we need to reduce our features to avoid redundancy.

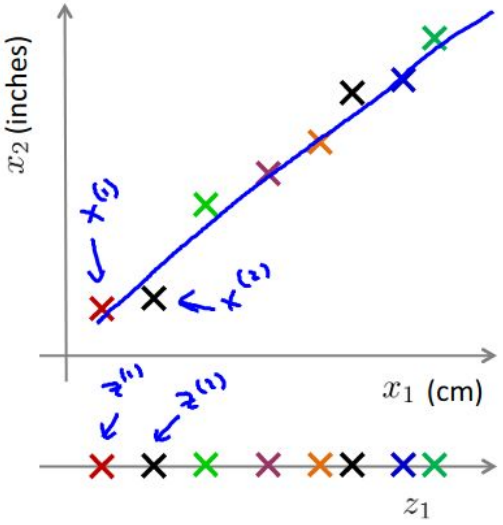
## Data Compression




Reduce data from 2D to 1D

Andrew Ng

## Data Compression




Reduce data from 2D to 1D

$x^{(1)} \in \mathbb{R}^2 \rightarrow z^{(1)} \in \mathbb{R}$

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

$\vdots$

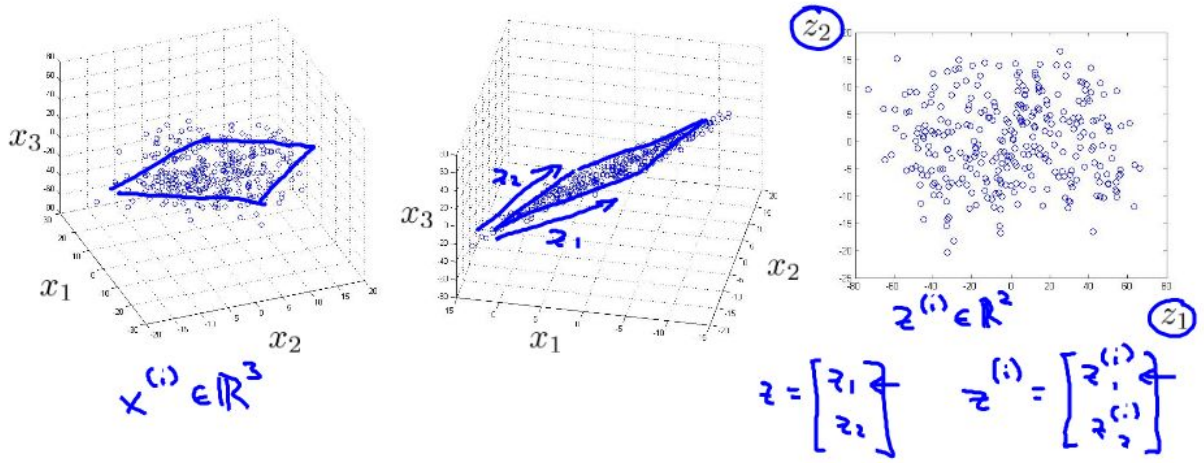
$x^{(m)} \in \mathbb{R}^2 \rightarrow z^{(m)} \in \mathbb{R}$

Andrew Ng

## Data Compression

10000 → 100D

Reduce data from 3D to 2D




Andrew Ng

Our target is to reduce our features into 2 or 3 dimensions. It will help us to plot and visualize data more clearly.

## 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 | ... |
| ... | ... | ... | ... | ... | ... | ... | |

[resources from en.wikipedia.org]

Andrew Ng

## Data Visualization

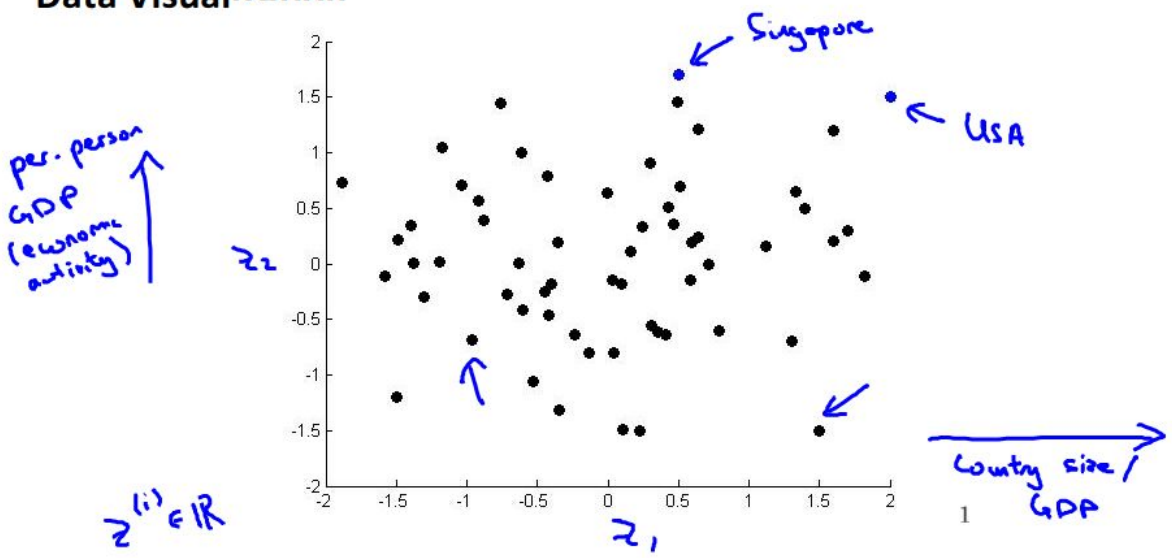
| Country | $z_1$ | $z_2$ |
|---|---|---|
| Canada | 1.6 | 1.2 |
| China | 1.7 | 0.3 |
| India | 1.6 | 0.2 |
| Russia | 1.4 | 0.5 |
| Singapore | 0.5 | 1.7 |
| USA | 2 | 1.5 |
| ... | ... | ... |

$z^{(i)} \in \mathbb{R}^2$

Reduce data from 50D to 2D

Andrew Ng

## Data Visualization

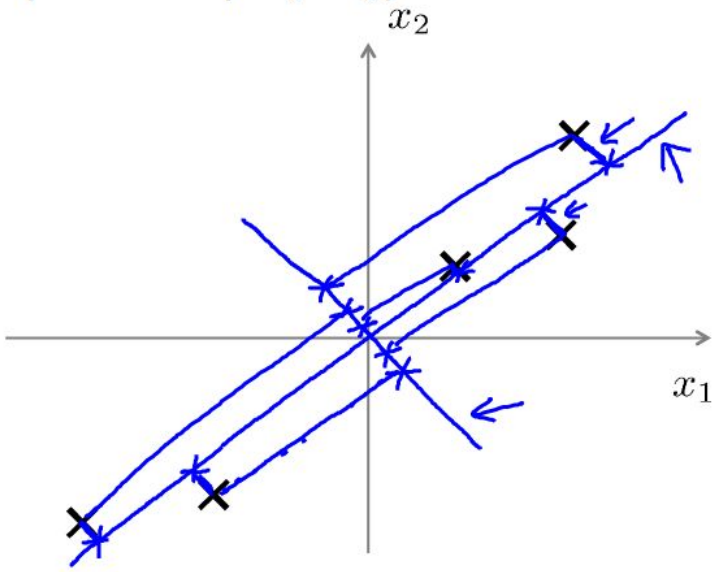



Andrew Ng

# Principal Component Analysis problem formulation :

For the problem of dimensionality reduction, by far the most popular algorithm is principal component analysis or pcr. By that algorithm we will find out a line on which every point of the training set will have a minimum projection distance (or have the minimum projection error).

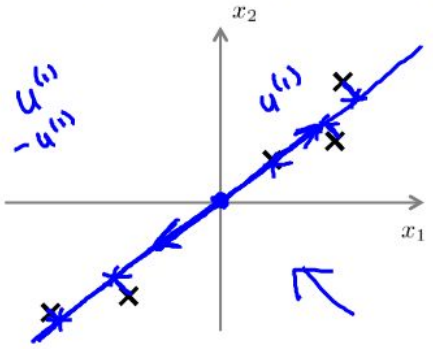
## Principal Component Analysis (PCA) problem formulation

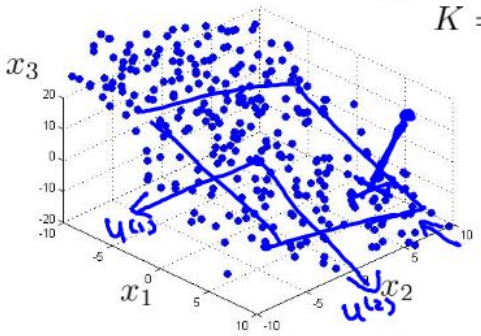



Andrew Ng

## Principal Component Analysis (PCA) problem formulation

$3D \to 2D$
$K = 2$



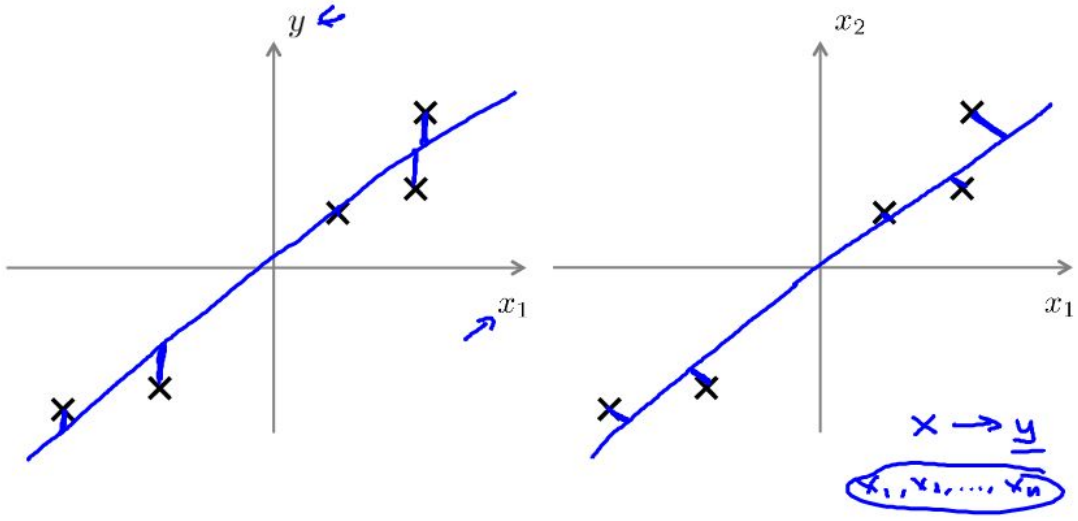

Reduce from 2-dimension to 1-dimension: Find a direction (a vector $u^{(1)} \in \mathbb{R}^n$) onto which to project the data so as to minimize the projection error.

Reduce from n-dimension to k-dimension: Find $k$ vectors $u^{(1)}, u^{(2)}, \ldots, u^{(k)}$ onto which to project the data, so as to minimize the projection error.

Andrew Ng

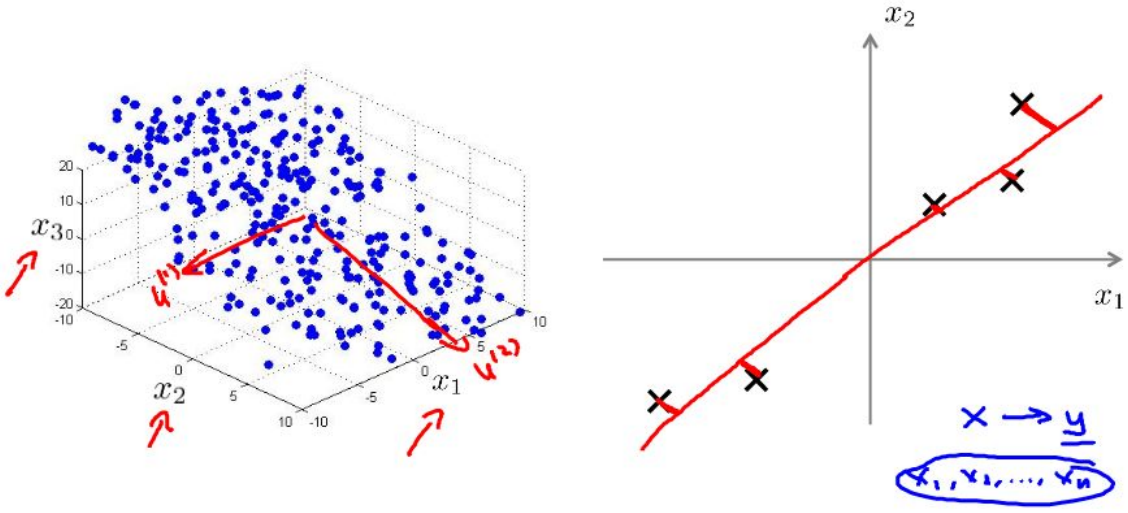
Sometimes it may appear that linear regression and PCA are similar. But they are not. In linear regression we try to fit a straight line so as to minimize the square error between point and this straight line. On the other hand PCA tries to minimize the orthogonal distance between the straight line and the ponts.

## PCA is not linear regression




Andrew Ng

## PCA is not linear regression




Andrew Ng

## PCA algorithm :

## Data preprocessing

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 $x_j - \mu_j$.

If different features on different scales (e.g., $x_1$ = size of house, $x_2$ = number of bedrooms), scale features to have comparable range of values.

$x_j^{(i)} \leftarrow \frac{x_j^{(i)} - \mu_j}{s_j}$



## Principal Component Analysis (PCA) algorithm

Reduce data from $n$-dimensions to $k$-dimensions

Compute "covariance matrix":

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

Compute "eigenvectors" of matrix $\Sigma$: → Singular value decomposition

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

eig (Sigma)

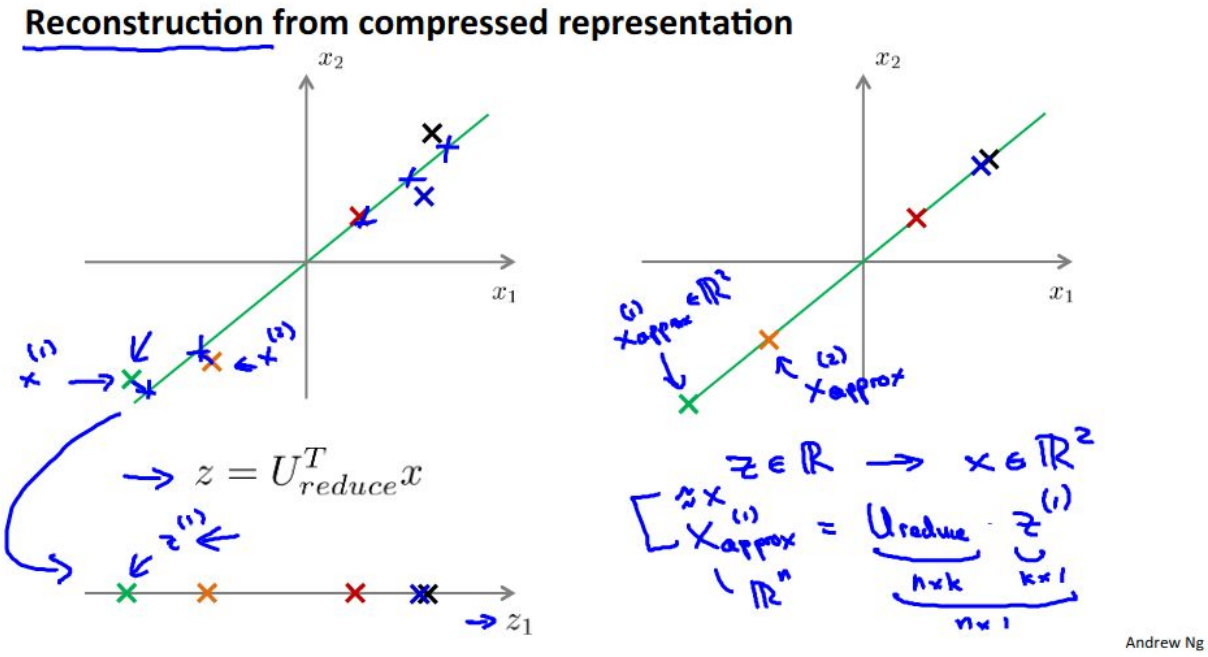
$n \times n$ matrix.

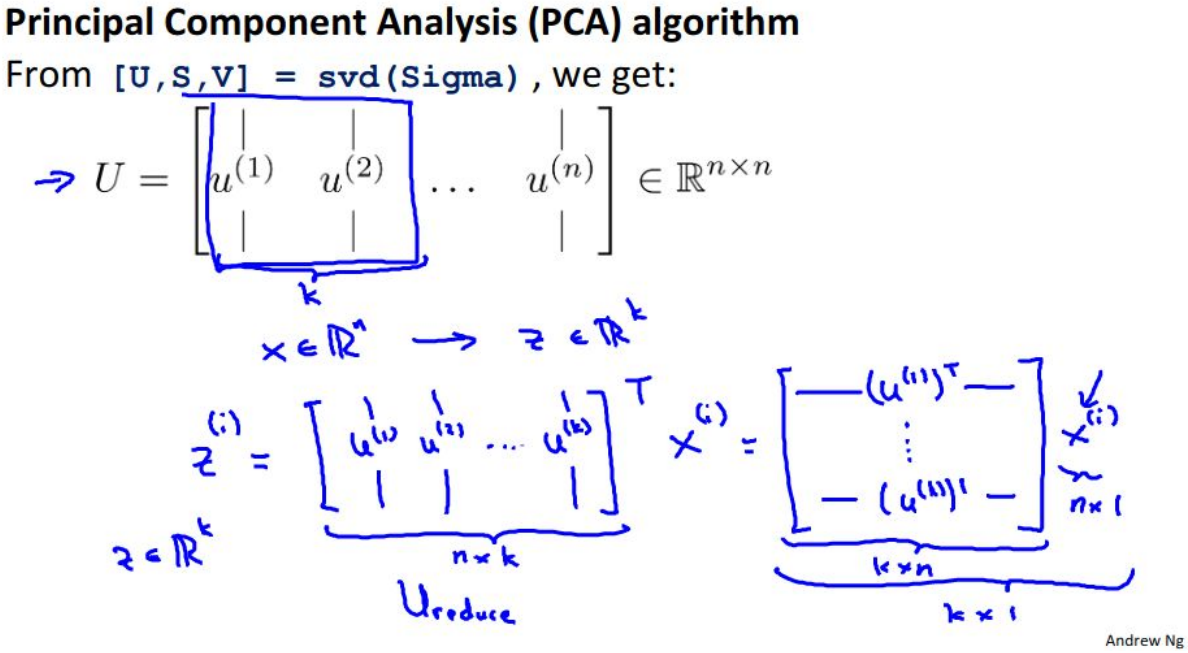
$U = \begin{bmatrix} | & | & | & & | \\ u^{(1)} & u^{(2)} & u^{(3)} & \cdots & u^{(n)} \\ | & | & | & & | \end{bmatrix}$ $\quad U \in \mathbb{R}^{n \times n}$

$u^{(1)}, \ldots, u^{(k)}$

# Reconstruction from compressed representation :

We may have to get back our original feature data from compressed format. In that case we need this algorithm.



## Principal Component Analysis (PCA) algorithm summary

After mean normalization (ensure every feature has zero mean) and optionally feature scaling:

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

```
[U,S,V] = svd(Sigma);
Ureduce = U(:,1:k);
z = Ureduce'*x;
```

$X = \begin{bmatrix} — & x^{(1)T} & — \\ & \vdots & \\ — & x^{(m)T} & — \end{bmatrix}$

Sigma = (1/m) * X' * X;

$x \in \mathbb{R}^n$ ~~$x_0 = 1$~~

## Choosing the number of principle component (K):

We can't just choose a for K as our wish but we need to satisfy some terms

### Choosing $k$ (number of principal components)

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

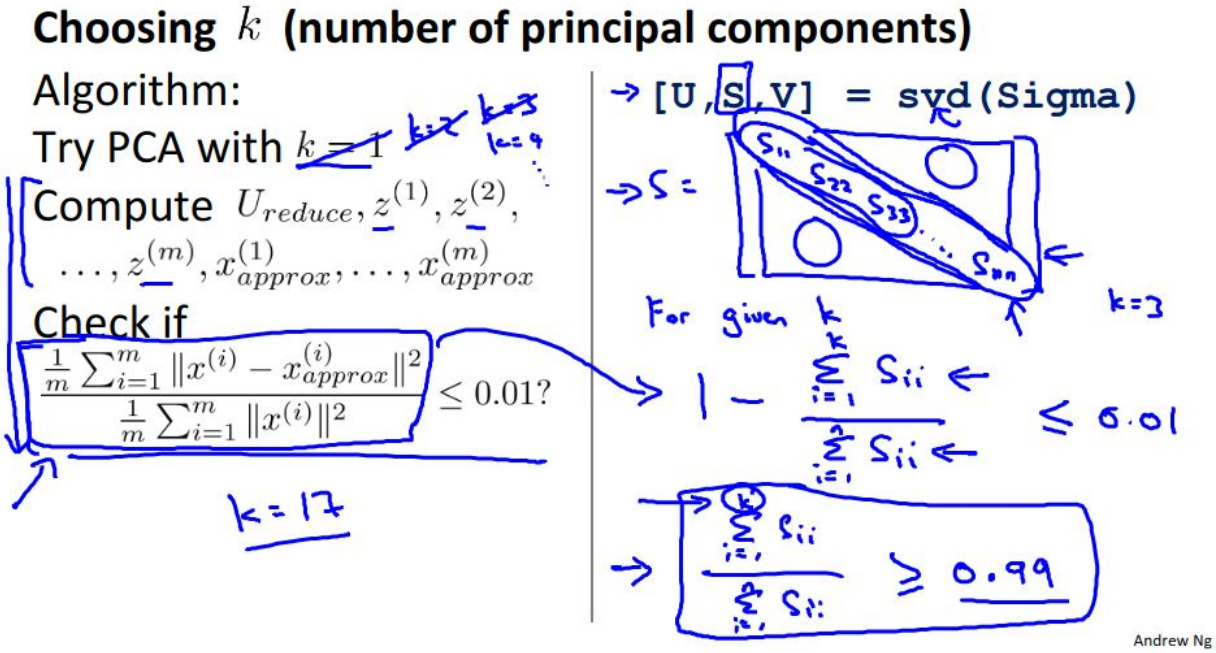
Total variation in the data: $\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}\|^2$

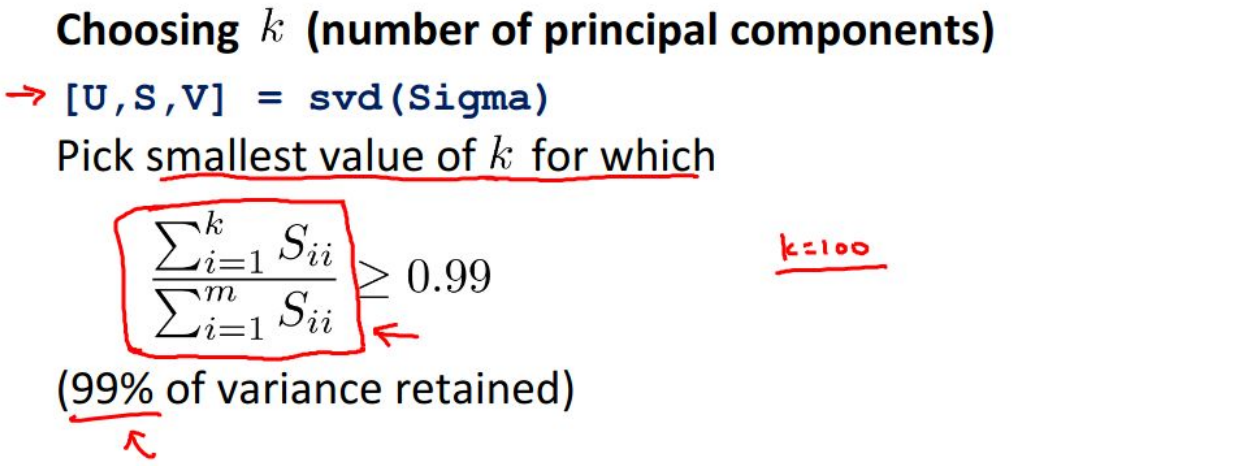
Typically, choose $k$ to be smallest value so that

$$\frac{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)} - x_{approx}^{(i)}\|^2}{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}\|^2} \leq \text{~~0.01~~ 0.05 0.10} \quad \text{~~(1%)~~ ~~5%~~ (10%)}$$

"~~99%~~ 95% 90% of variance is retained"

It is better to retain 95-99% of variance. We have two ways to ensure that as pictured below. But the right one is more efficient.

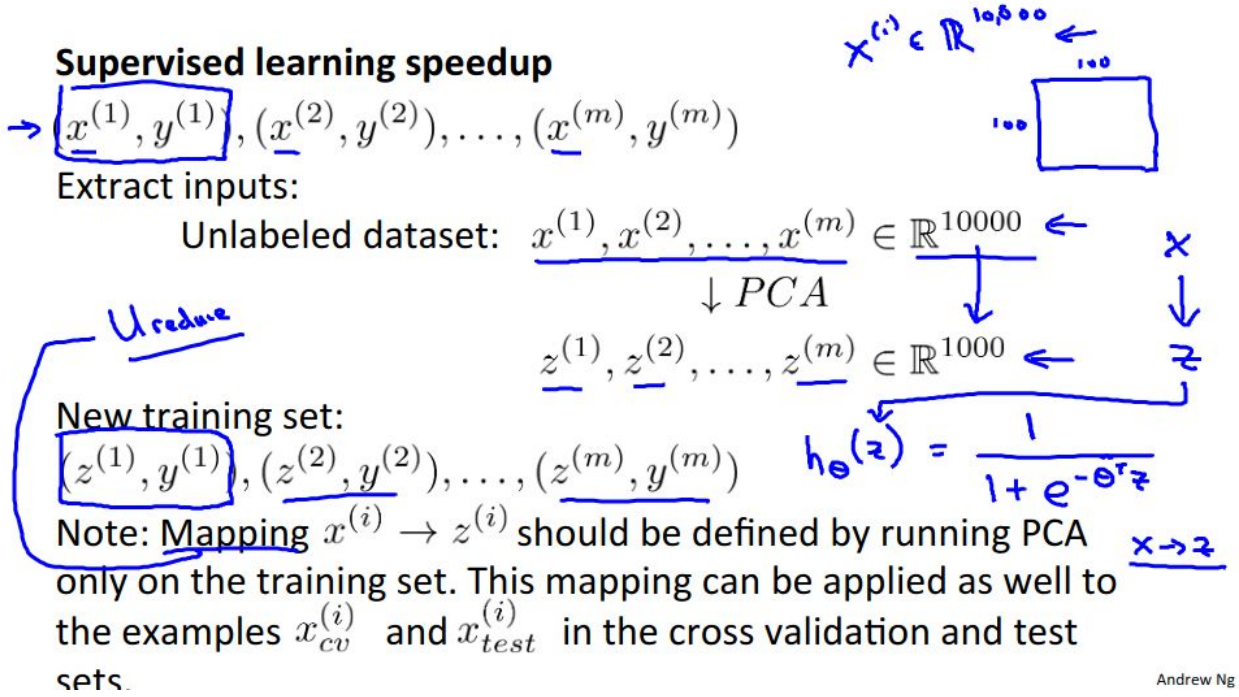


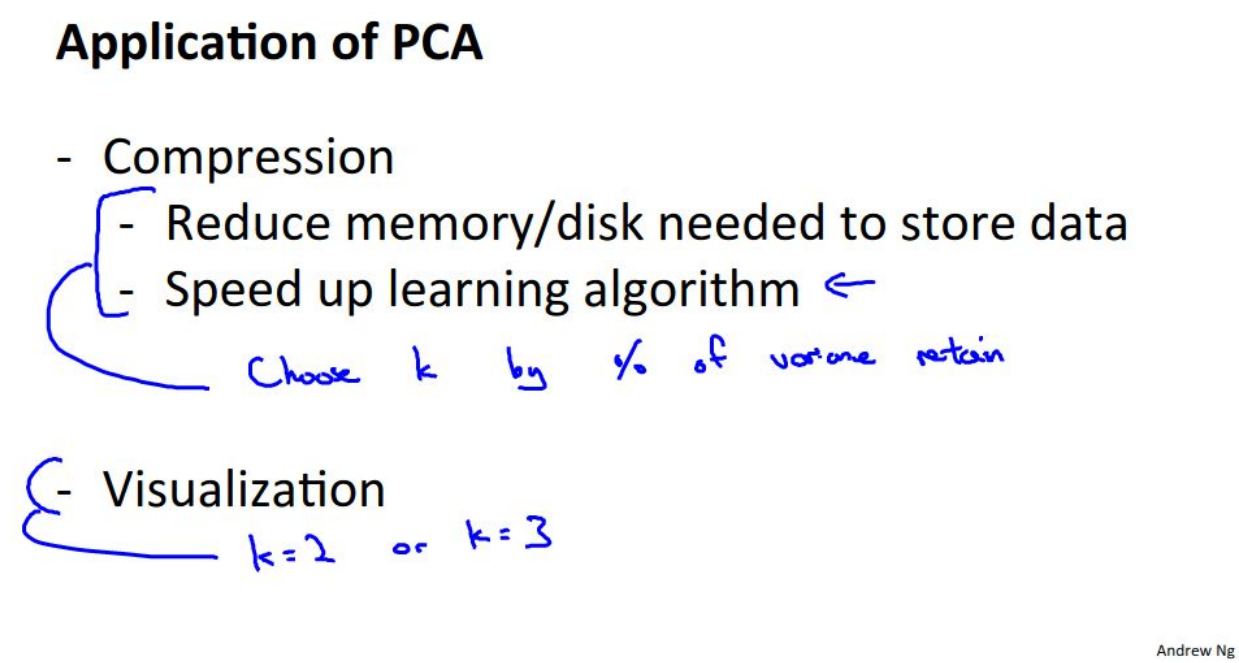




Advice for applying PCA :

We can apply PCA in supervised learning to speed up our learning algorithm. In order to do so we need to to first extract the input x'es and then reduce their dimension then

put them in the previous mapping. Then we can follow that reduce mapping for cross validation sets and new examples.




Though PCA helps us to reduce features but it is a bad idea to apply it to prevent overfitting over regularization.

## Bad use of PCA: To prevent overfitting

→ Use $z^{(i)}$ instead of $x^{(i)}$ to reduce the number of features to $k < n$.

Thus, fewer features, less likely to overfit.

Bad!

This might work OK, but isn't a good way to address overfitting. Use regularization instead.

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

Andrew Ng

It is also a bad habit to want to apply PCA without going another possible ways. We need to first implement our algorithm without PCA and if we cannot achieve our goal then we can use PCA . There, achieving goal means shortage of memory or slowness of our algorithm is referred.

## PCA is sometimes used where it shouldn't be

Design of ML system:

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

→ 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 $z^{(i)}$.

Andrew Ng