WEEK - 8

⇒ In unsupervised learning, the training set is of the form $\{x^{(1)}, x^{(2)}, x^{(3)} \ldots x^{(m)}\}$ without labels (y).

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

⇒ K-means is an iterative algorithm and it does two things —
1) = Cluster assignment step
2) = Move centroid step

⇒ **K-means Algo**

Input → 1) = K (number of clusters)
2) = Training set (m)

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

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

→ Repeat {

cluster assignment:
for $i = 1$ to m {
$c^{(i)}$:= index (from 1 to K) of cluster centroid closest to $x^{(i)}$ } → $\min_k \|x^{(i)} - \mu_k\|$

centroid move:
for $k = 1$ to K {
μ_k := average (mean) of points — assigned to cluster k }

}

$\Rightarrow \quad c^{(i)} = \min_k \|x^{(i)} - \mu_k\|^2$ (cluster assignment)

$\Rightarrow$

$\Rightarrow \quad \mu_2 = \frac{1}{4}\left[x^{(i)}, x^{(j)}, x^{(k)}, x^{(n)}\right] \in R^n$ → centroid movement

the points whose $c = 2$ (belong to cluster 2)

Optimization Objective of K-means algorithm

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

aka Distortion f^n

- K should be less than m ($K < m$)

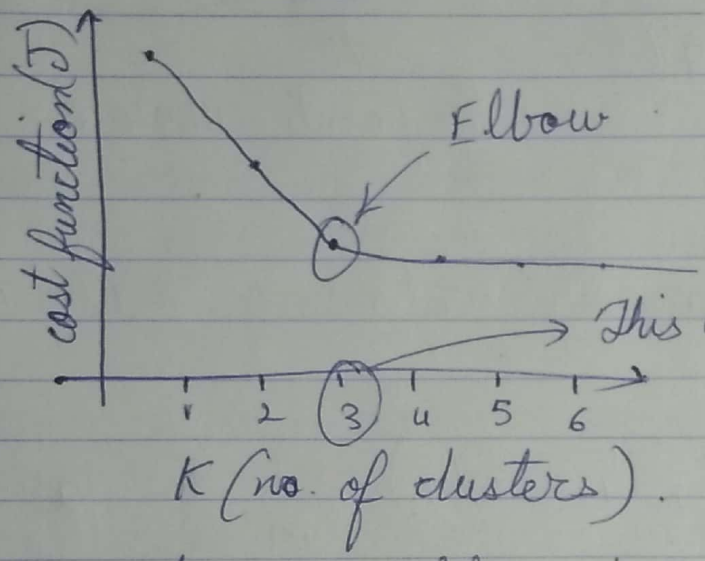
m → no. of examples.

- If $K = 2-10$, then having multiple random initialization can help to find min cost f^n.
- If $K > 100$, then having multiple random initialization would not help much.

P.T.O

⇒ For choosing the value of K.

→ Elbow Method.



⇒ If the above graph varies at any point, that means k-means got stuck in a bad local minimum. We should try ~~return~~ rerunning k-means with multiple random initializations.

⇒ By hand → manually → by visualizing for each value of K and deciding which no. of clusters fit correctly.

⇒ Based on a metric to use for later/downstream purpose.

⇒ Dimensionality Reduction is the second type of unsupervised machine learning algo.

⇒ DIMENSIONALITY REDUCTION (DR)

→ DATA COMPRESSION

⇒ Reduce data from 2D to 1D or 3D to 2D → we can compress two similar features (features who have the same meaning) into a single feature which holds or represents both their data.

→ This saves memory and also helps the algo run faster.

⇒ Most Common (DR) algo is PCA
Principal Components Analysis.

⇒ PCA tries to find a lower dimensional surface onto which to project the data so that the distance b/w the points & the surface is minimum.

⇒ The distance b/w the points & the surface onto which they are projected is called projection error.

⇒ Formula →

Reduce 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.

→ For n-directions, we define k vectors
$u^{(1)}, u^{(2)} \ldots u^{(k)}$

⇒ $\|u^{(1)}\| = \sqrt{(u_1^{(1)})^2 + (u_2^{(1)})^2}$

PCA Algo

⇒ Before performing PCA, we have to preprocess the data ie: perform feature scaling or mean normalization.

Mean normalization

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

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

⇒ To reduce data from n-dimensions to k-dimensions

→ Compute → "covariance matrix"

$$\underset{\text{(Sigma)}}{\Sigma} = \frac{1}{m} \sum_{i=1}^{n} \underbrace{(x^{(i)})}_{n \times 1} \underbrace{(x^{(i)})^T}_{1 \times n}$$

Summation Symbol.

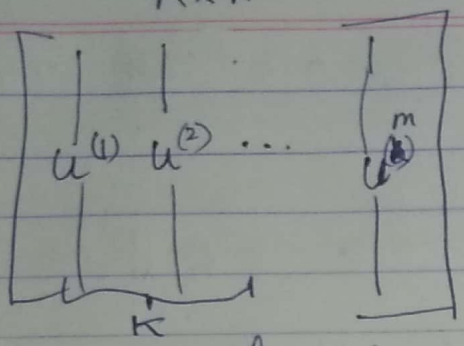
→ Compute "eigen vectors" of matrix Σ

$[U, S, V] = \text{svd}(\text{Sigma})$; → $n \times n$ matrix

or

⇒ $\text{eig}(\text{Sigma})$;

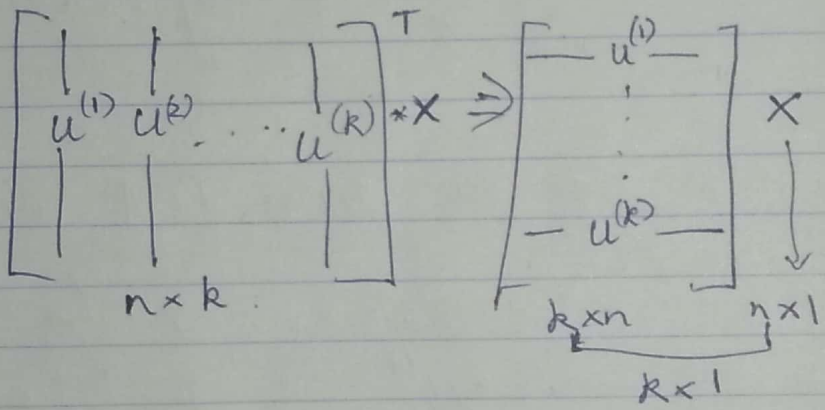
svd → Singular Value Decomposition

$\Rightarrow$ $U =$ 

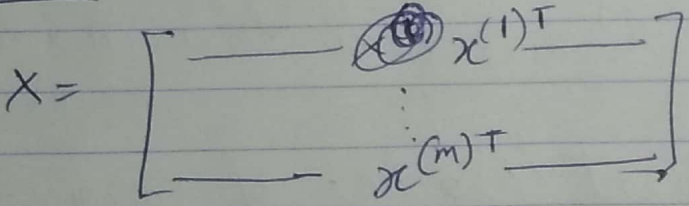
$u \in \mathbb{R}^{n \times n}$.

and we are compressing from n dimensions to k dimensions

$\rightarrow$ we take the first k columns of U

we get $Z =$ (Ureduce) 

$\Rightarrow$ Vectorized $\Rightarrow$ Sigma = $(1/m) * X' * X;$

$X =$ 

$\Rightarrow$ $[U, S, V] = svd(Sigma);$

$\rightarrow$ Ureduce = $U(:, 1:k);$

$\rightarrow$ $Z = Ureduce' * x;$

⇒ Reconstruction from compressed representation

→ $z \in \mathbb{R} \longrightarrow x \in \mathbb{R}^2$

$\therefore x_{approx} = \underbrace{U_{reduce}}_{n \times k} * \underbrace{z^{(i)}}_{k \times 1}$

$n \times 1$

which is very near to the original data.

⇒ If we run PCA with $k = n$ (no reduction)

$\therefore x_{approx} = x$ for every example x

$\therefore$ The percentage of variance retained will be 100%.

$\therefore$ U_{reduce} will be a $n \times n$ matrix.

⇒ Percent/Fraction of variance retained is given by

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

⇒ k = no of principal components.

⇒ 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 →

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

or

0.05 (5%)

or

0.10 (10%)

or

0.15 (85%)

→ "99% of variance is retained"

⇒ Algo for choosing k (very inefficient)

→ Try PCA with k = 1

→ Compute $U_{reduce}, z^{(1)}, z^{(2)} \ldots z^{(m)}, x_{approx}^{(1)} \ldots x_{approx}^{(m)}$.

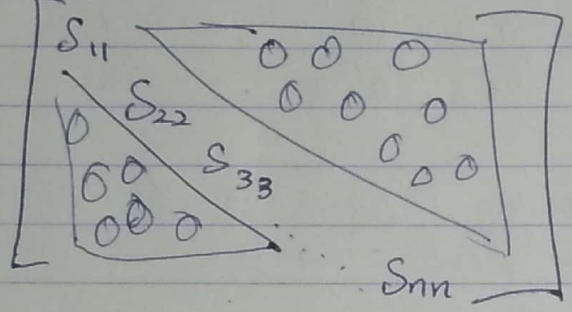
→ Check if 99% of variance is retained i.e → by using the big formula above

$m \Rightarrow$ no. of training examples.

~~//~~ $\Rightarrow$ Easier & faster method (using svd())

$\rightarrow$ $[U, S, V] = \text{svd}(\text{sigma})$

$\Sigma =$



For given k.

$$1 - \frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \leq 0.01$$

OR

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

$\Rightarrow$ PCA tries to minimize $\frac{1}{m} \sum_{i=1}^{m} \| x^{(i)} - x_{approx}^{(i)} \|^2$

$\Rightarrow$ You should run the PCA only on the training set.