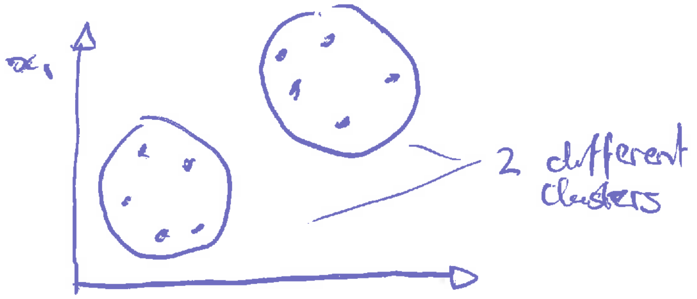
week 8

# Unsupervised learning - introduction : Clustering

## Unsupervised learning

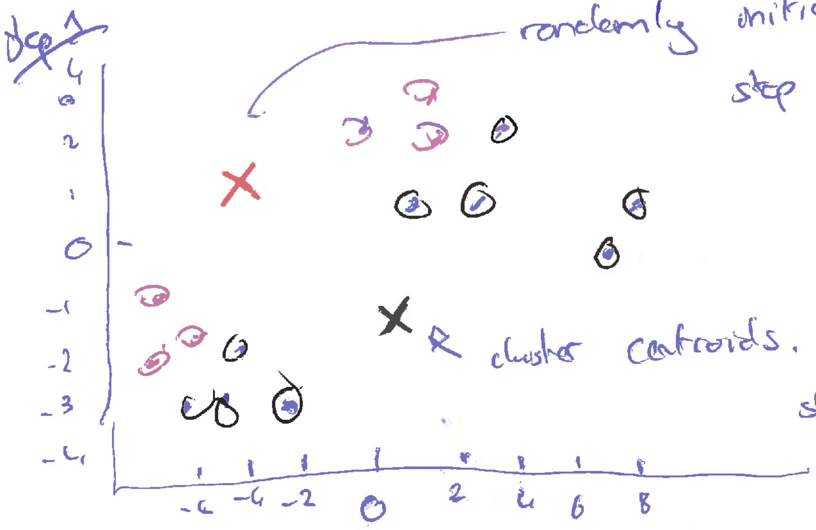


Training set: $\{x^{(1)}, x^{(2)}, x^{(3)} \ldots x^{(m)}\}$

## Applications of clustering

- Market segmentation
- Social network Analysis
- Organise computing clusters
- astronomical Analysis

# K-means Algorithm



randomly initializ two cluster center points

step 1) Cluster assignment step

So if a data point is closer to a cluster centroid → it is assigned to that cluster centroid.

step 2) move centroid step

→ compute average of all the assigned ~~that~~ datapoints of a cluster and move the centroid to this new centrepoint.

now closer together - centroids "moved"

→ after awhile the centroid locations no longer change.

# K-means algorithm

Input:

- k (number of clusters) ✓
- training set $\{x^{(1)}, x^{(2)}$ … ✓ (no y…)

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

vosannamane
@gmail.com

## K-means algorithm

total number of centroids

Randomly initialize k clu… 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 $x^{(i)}$
                                   $\min_k \|x^{(i)} - \mu_k\|^2$ & length.
    move centroid step.
    for k = 1 to K
        $\mu_k$ := average (mean) of points assigned to cluster k
}
```

- cluster assignment step: distance between ith training example and cluster centroid

↳ so ∴ picking the cluster centroid $c^{(i)}$ that is closest to our training example $x^{(i)}$

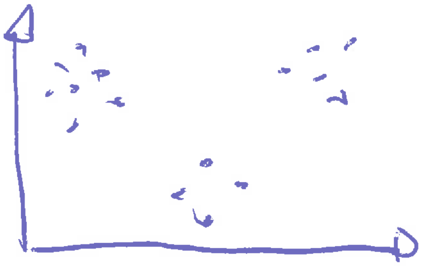
↳ let assume this cluster ($\mu_2$) has the values $x^{(1)}, x^{(5)}, x^{(6)}, x^{(10)}$ assigned to it.

$$\therefore\ c^{(1)} = 2,\ c^{(5)} = 2,\ c^{(6)} = 2,\ c^{(10)} = 2.$$

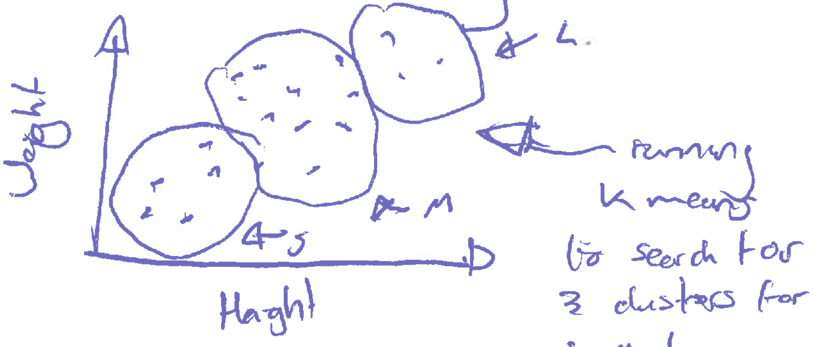
$$\mu_2 \leftarrow \frac{1}{4}[x^{(1)} + x^{(5)} + x^{(6)} + x^{(10)}) \in \mathbb{R}^n$$

notes: ~~del delte~~ delete a cluster if it has no points assigned to it
↳ or reinitialize it ~~b~~ through randomization

## K-means for non-separated Clusters



T-shirt sizing S, M, L



running K means
↳ search for 3 clusters for S, M, L.

Week 8

# Clustering - optimization objective

## K-means optimization objective

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

lower case $k$ is an index into   $K = \{1, 2, \dots K\}$

$\mu_k$ = cluster ~~centroid~~ $K$ $(\mu_k \in \mathbb{R}^n)$

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

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

$\mu_{c^{(i)}} = \mu_5 = ?$

$x$ has been assigned to cluster 5

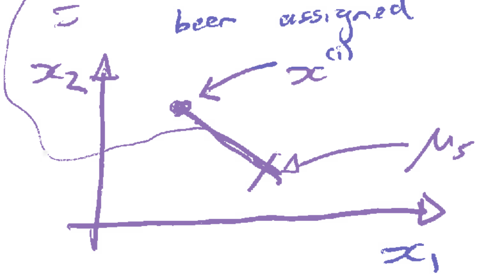
optimization objective:

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

$\Leftarrow$ location of the cluster centroid to which $x^{(i)}$ has been assigned

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

$\rightarrow$ Sometimes called the "distortion cost function"



$\rightarrow$ step 1 | The cluster assignment step is minimizing $J(\dots)$ with respect to $c^{(1)}, c^{(2)}$ etc ... $c^m$ (holding $\mu_1, \dots, \mu_K$ fixed)

$\rightarrow$ step 2 | minimises $J(\dots)$ w.r.t. $\mu_1 \dots \mu_K$.

## Clustering - Random initialization

should have $K < m$

Randomly pick $K$ training examples

Set $\mu_1, \dots, \mu_K$ equal to these $K$ examples

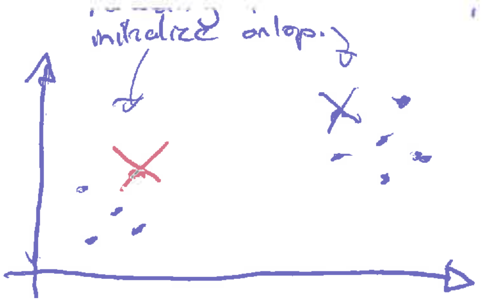
$\mu_1 = x^{(i)}$

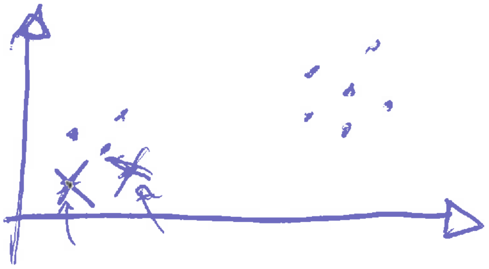
$\mu_2 = x^{(j)}$

$\vdots$

lets say: $\boxed{K=2}$

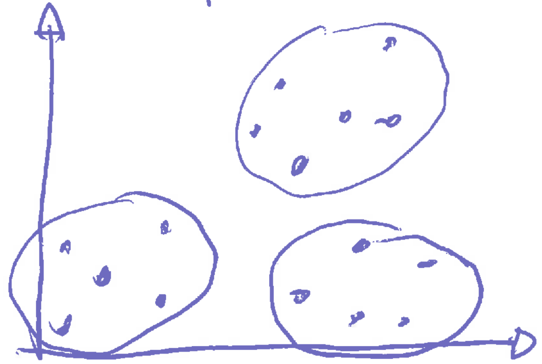
two examples





K means can converge to different solutions depending how the clusters were initialized

Local optima. → can get stuck at different Local optima.



initialize K means lots of times (randomly) to increase the chances of finding the global minimum.

## Random initialization

typical 50 → 1000

```
for i = 1 to 100 {
    Randomly initialize K-means
    Run K means to 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 gives lowest cost $J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots \mu_K)$
```

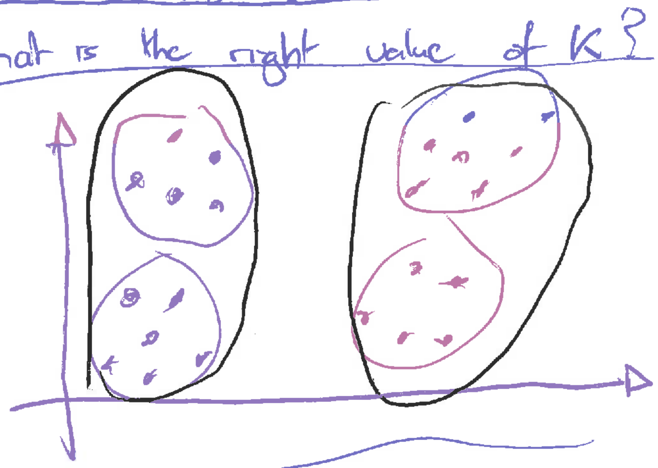
K=2 → 10

If you have K>10, multiple random initializations will not make much difference

week 6

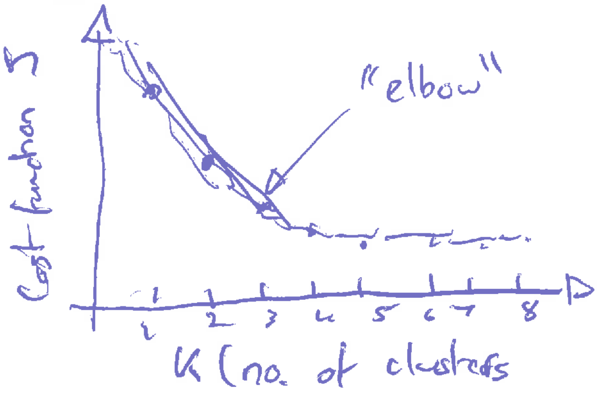
# clustering - choosing the number of clusters

## What is the right value of K?

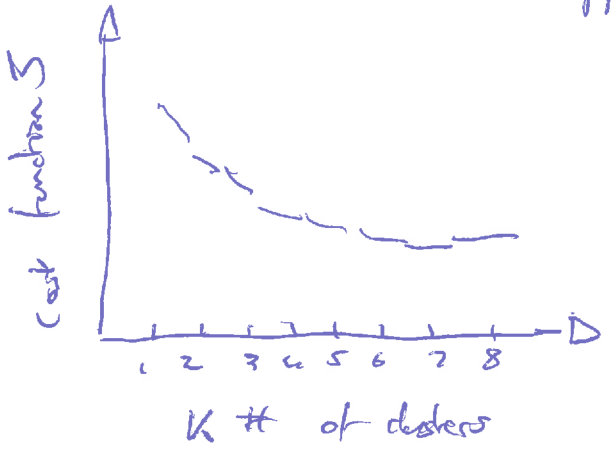


## choosing the value of K

Elbow method: "worth a shot"



Sometimes this happens:

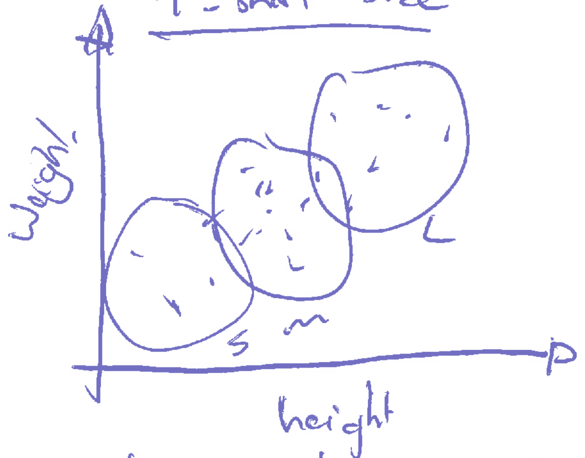


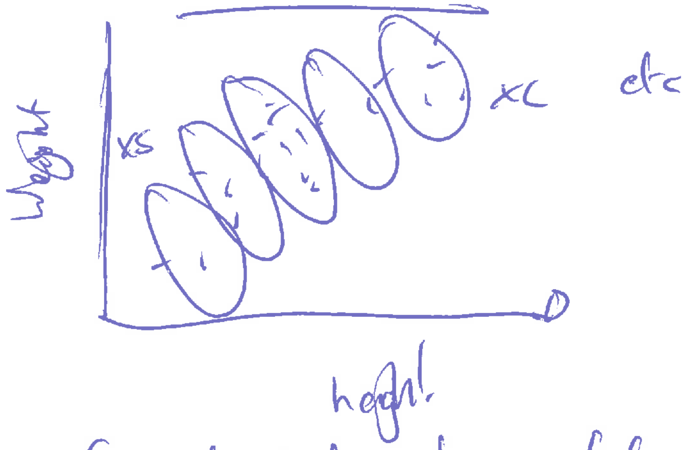
## Choosing the value of K

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

K=3 T-shirt size S, M, L

K=5 T-shirt size XS, S, M, L, XL

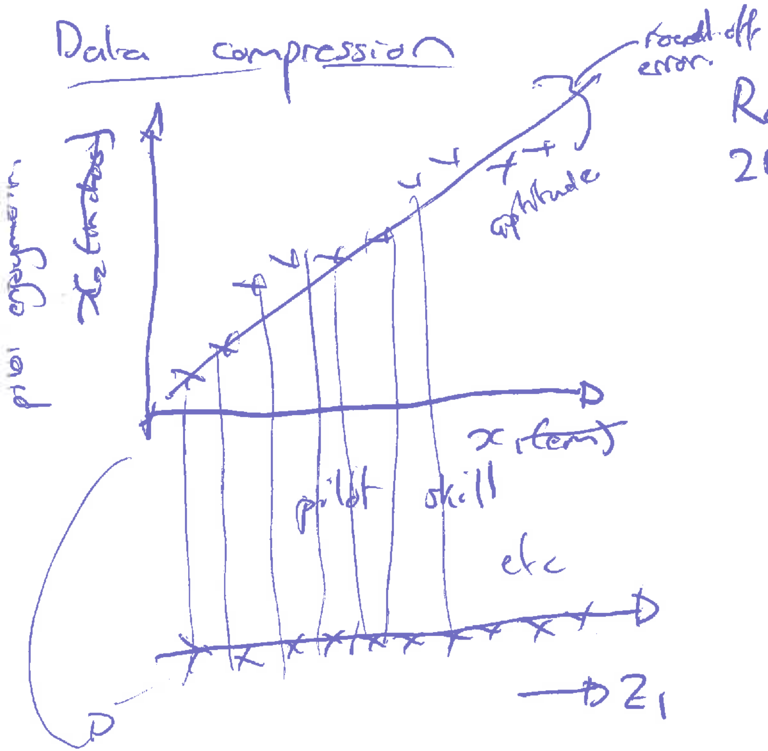




deciding what set of sizes for t shirts to manufacture is week8

# Dimensionality Reduction – Motivation I: data compression

## Data compression



need only 1 real number to specify a point on a line.

Reduce data from 2D to 1D

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

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

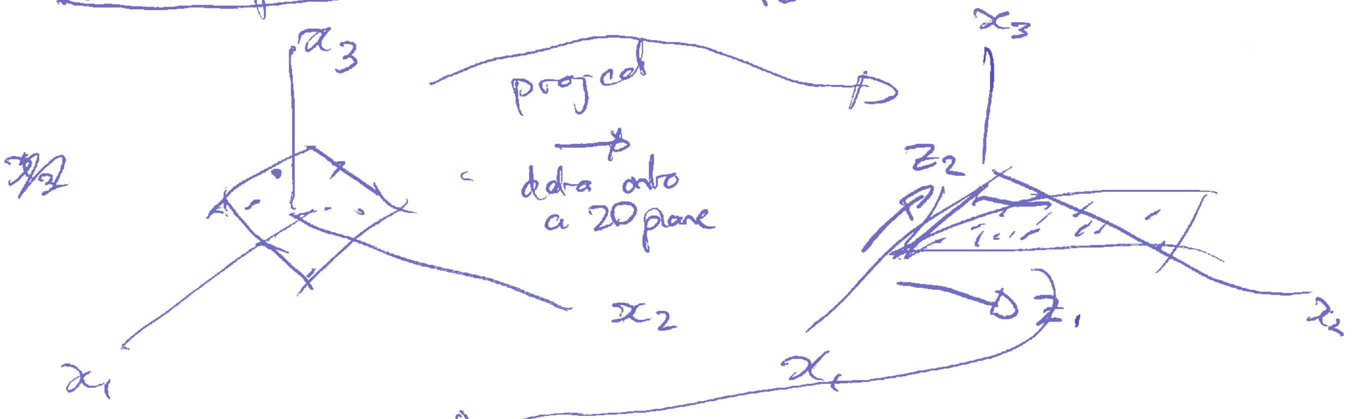
$\vdots$

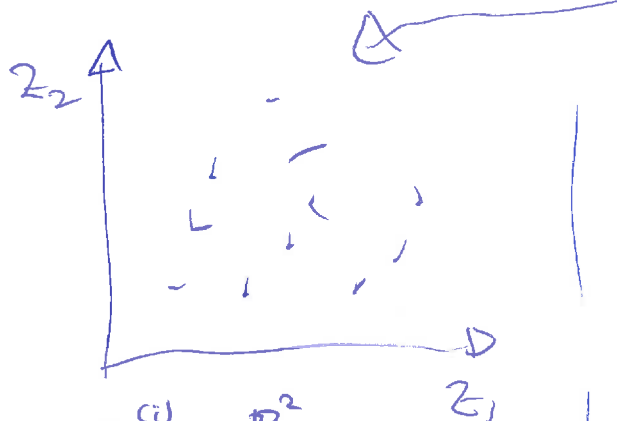
$x^{(m)} \to z^{(m)}$

## Data Compression

1000D to 100D

reduce data from 3D to 2D.



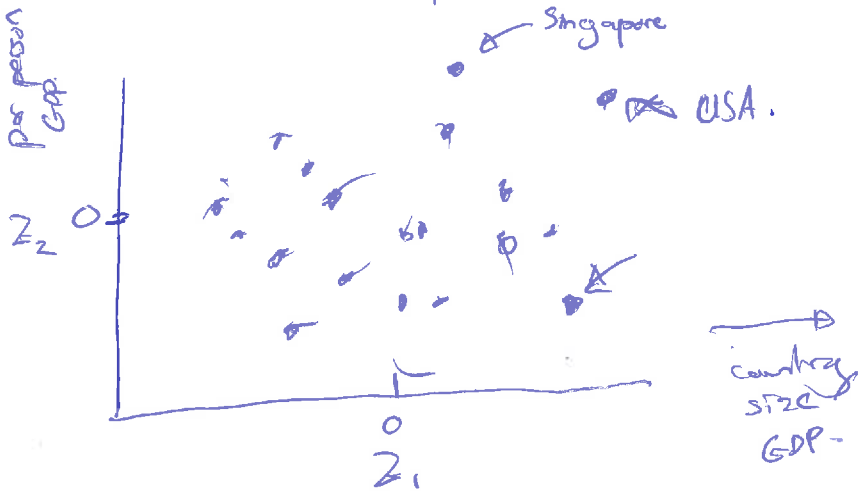


$z^{(i)} \in \mathbb{R}^2$

$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$ $z^{(i)} = \begin{bmatrix} z_1^{(i)} \\ z_2^{(i)} \end{bmatrix}$

# Dimensionality Reduction: Motivation II: data visualization

Reduce data from 50D to 2D.



# Dimensionality Reduction - 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, $x_2$ = number of bedrooms), scale features to have comparable range of values.

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

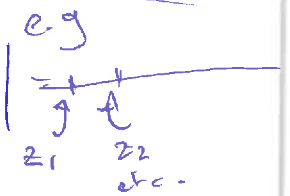
$s_j$ ← standard deviation of feature j.

## Principal Component Analysis Algorithm

- Reduce data from 2D to 1D
- Reduce data from 3D to 2D.

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

$$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$



## Principal Component Analysis (PCA) Algorithm

Reduce data from n-dimensions to k-dimensions

Compute "covariance matrix"

not to be confused with summation....

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

Compute "eigenvectors" of matrix $\Sigma$ → $n \times n$ matrix

→ [U, S, V] = svd(Sigma); (or eig(sigma))

← octave ↳ singular value decomposition.

$U \leftarrow \mathbb{R}^{n\times n}$

$$U = \begin{bmatrix} | & | & | & & | \\ u^{(1)} & u^{(2)} & u^{(3)} & \cdots & u^{(n)} \\ \underbrace{| \quad | \quad |}_{K} & & & & \end{bmatrix}$$

$u^{(1)}, \dots, u^{(k)}$ : ← k direction onto which we want to project the data.

vectors for plane

# Dimensionality reduction - principal component analysis algorithm

from $[U,S,V] = svd(Sigma)$, we get:

$$U = \underbrace{\begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(n)} \\ | & | & & | \end{bmatrix}}_{k.} \in \mathbb{R}^{n \times n}.$$

$$X \in \mathbb{R}^n \longrightarrow z \in \mathbb{R}^k$$

$$z = \underbrace{\begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(k)} \\ | & | & & | \end{bmatrix}^T}_{n \times k.} X^{(i)} = \underbrace{\underbrace{\begin{bmatrix} \text{---} & u^{(1)} & \text{---} \\ \text{---} & u & \text{---} \\ \text{---} & (u^{(k)})^T & \text{---} \end{bmatrix}}_{k \times n} \underset{n \times 1}{X^{(i)}}}_{k \times 1}$$

$U_{reduce}$

$z = \mathbb{R}^k$

---

## PCA Algorithm summary

$$Sigma = \frac{1}{m} \sum_{i=1}^{m} (x^{(i)})(x^{(i)})^T$$ implemented on octave looks like → $sigma = (1/m) * X' * X;$

$$\left.\begin{aligned} [U,S,V] &= svd(Sigma); \\ Ureduce &= U(:,1:k); \\ z &= Ureduce' * x; \end{aligned}\right\} \text{octave code}$$

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

# Dimension Reduction : Reconstruction from compressed representation

in the previous examples we ~~app~~ reduced a 2D (or $N^{th}$ D) dimension space, down to 1D using

$(x_2, x_1)$ $\Big\{ \; Z = U^T_{reduce} \, x$

$\uparrow$ new dimension (1D)

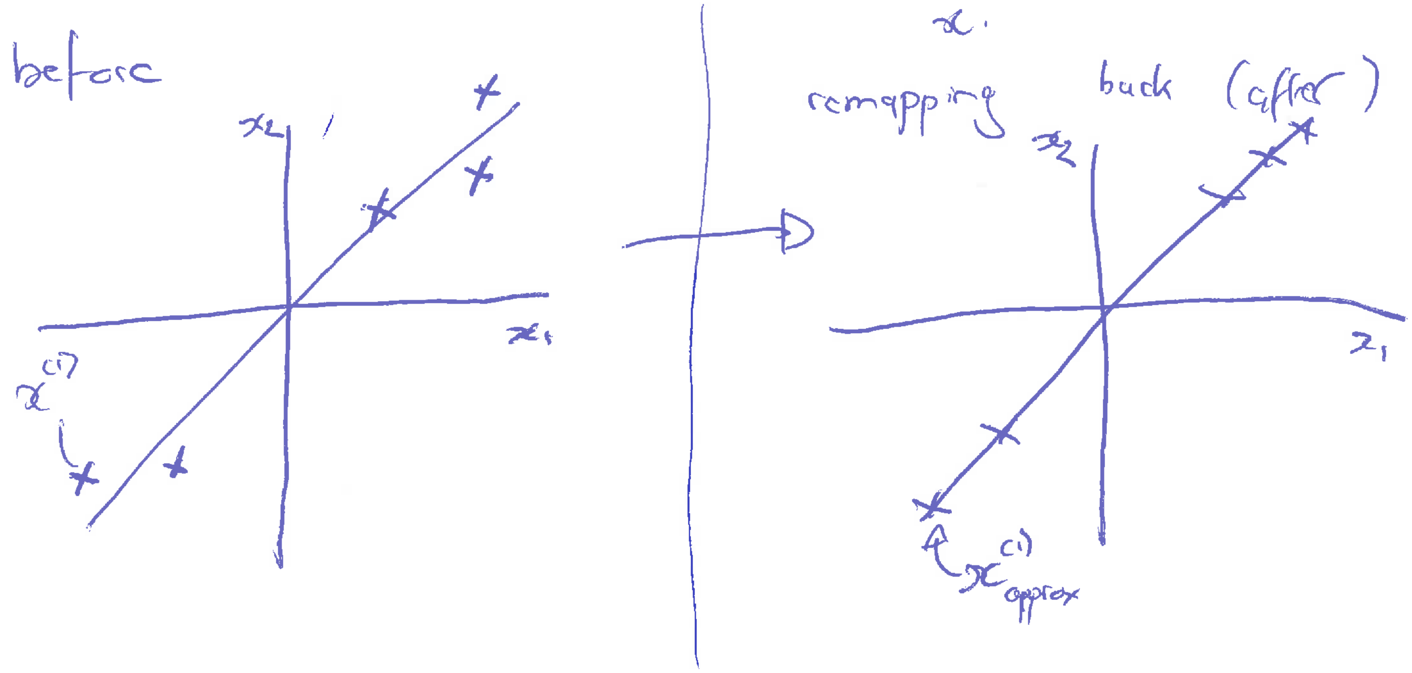
to reverse it, we need to do the following:

$$Z \in \mathbb{R} \longrightarrow x \in \mathbb{R}^2 .$$

$$x_{approx} = \underbrace{\underbrace{U_{reduce}}_{n \times k} \cdot \underbrace{Z}_{k \times 1}}_{n \times 1}$$

$\approx x$

$\uparrow$ original $x$.



# Dimensionality Reduction – choosing the number of principal components

## Choosing k (number of principal components)

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

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

Typically, choose k to be smallest value so that

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

or 0.05 (5%)

"99% of variance is retained" / 95% of variance retained

## Choosing k (number of principal components)

Algorithm:

Try PCA with $k = \cancel{1}$, $\cancel{k=}2$, $k = 3$ etc

if not less than 0.01 try 2 if not try 3

Compute $U_{reduce}, z^{(1)}, z^{(2)}, \ldots, z^{(m)}, x^{(1)}_{approx}, \ldots, x^{(m)}_{approx}$

Check if:

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

$[U, S, V] = svd(sigma);$

$$S = \begin{bmatrix} S_{11} & & & & 0 \\ & S_{22} & & & \\ & & S_{33} & & \\ & & & \ddots & \\ 0 & & & & S_{nn} \end{bmatrix}$$

For given value k

k=3

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

n=n

## Summary

→ Pick smallest value of k for which

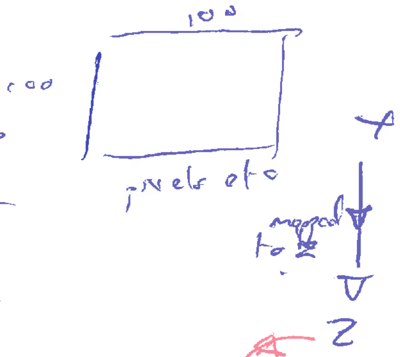
$$\frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{m} S_{ii}} \geq 0.99 \quad (99\% \text{ variance retained})$$

"Choose what dimension to reduce the data too"

# Dimensionality reduction - advice for applying PCA

## Supervised learning speedup

$\boxed{(x^{(1)}, y^{(1)})}, (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$ $\quad x^{(i)} \in \mathbb{R}^{10000}$



extract inputs:

Unlabeled dataset: $\underbrace{x^{(1)}, x^{(2)}, \ldots, x^{(m)}}$ $\in \mathbb{R}^{10000}$

$\downarrow$ PCA

new representation

$z^{(1)}, z^{(2)}, \ldots, z^{(m)} \in \mathbb{R}^{1000}$.

New training set

$\boxed{(z^{(1)}, y^{(1)})}, (z^{(2)}, y^{(2)}), \ldots, (z^{(m)}, y^{(m)})$

if using logistic regression for example:

$$h_\theta(z) = \frac{1}{1 + e^{-\theta^T z}}$$

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 $x^{(i)}_{cv}$ and $x^{(i)}_{test}$ in the cross validation and test sets.

Ureduce

parameter learned by PCA

$\rightarrow$ fit parameters only to training set. not cv or test set. } $\rightarrow$ applying mapping too ~~train~~ cv & test sets

## Application of PCA

- Compression
  - Reduce memory/disk needed to store data.
  - Speed up learning algorithm
  - $\rightarrow$ choose k by % of variance retained.
- Visualization ✓
  - $\rightarrow$ k=2 or k=3 ✓

# Dimensionality reduction - advice for applying PCA

## Bad use of PCA: To prevent overfitting:

Use $\underline{z}^{(i)}$ instead of $x^{(i)}$ to reduce the number of features $\underline{k < n}$ 1000 1000

Thus, fewer features, less likely to overfit. Bad!

This might work ok, but it is not a good way to address overfitting. Use regularization instead.

$\boxed{\lambda}$ ftw. ($\lambda$ knows what the values of y are ... & minimization term)

(PCA)
-> it throws away data, without knowing what the values of y are.

PCA is somtimes used where it should not be "

Design of ML system:

- -> Get training set $((x^{(1)}, y^{(1)}) \dots (x^{(m)}, y^{(m)}))$
- ~~-> Run PCA to reduce $x^{(i)}$ in dimension to get $z^{(i)}$~~
- -> train logistic regression on $((z^{(1)}, y^{(1)}), \dots (z^{(m)}, y^{(m)}))$
- -> test on test set: Map $x^{(i)}_{test}$ to $z^{(i)}_{test}$. Run $h_\theta(z)$ on $(z^{(1)}_{test}, y^{(1)}_{test} \dots z^{(m)}_{test}, y^{(m)}_{test})$
- -> how about the whole thing without PCA
- -> try running the normal thing with original / raw data $x^{(i)}$. If the implementation is too slow etc try implementing $z^{(i)}$ etc