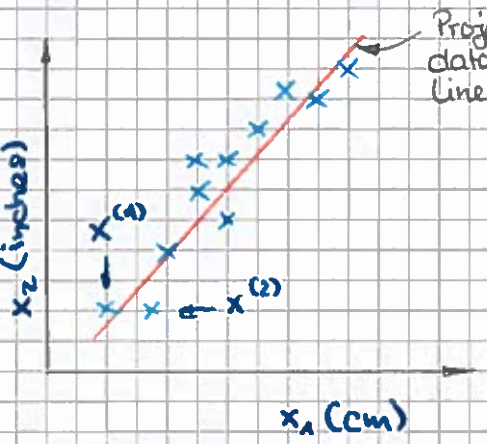
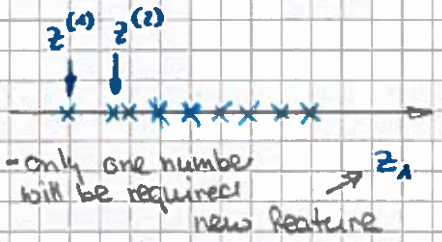
14. Dimensionality Reduction (PCA)

Data Compression:



Reduce Data from 2D to 1D

- Redundant data set - different units for same attributes
  ⇒ can happen when different teams are working independently

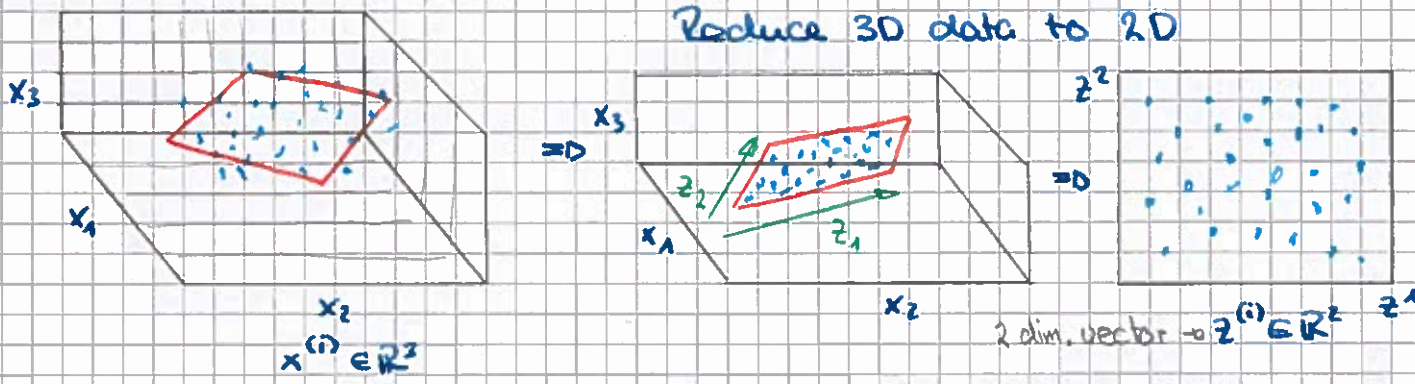


- only one number will be required
  new feature

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

⇒ compression lead to faster algorithm

Reduce 3D data to 2D



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

Dimensionality reduction of a dataset $(x^{(1)}, x^{(2)}, \dots x^{(m)})$ where $x^{(i)} \in \mathbb{R}^n$.

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

Data Visualization:

Country	x_1 GPD	x_2 per capita GPD	x_3 Human dev.	x_4 life	x_5 pov. index.
Canada	1.577	39.17	0.908	80.7	32.6
China	⋮	⋮	⋮	⋮	⋮
India	⋮	⋮	⋮	⋮	⋮
Singapore	0.223	56.69	0.866	80	42.5
USA	14.527	46.86	0.91	78.3	40.8

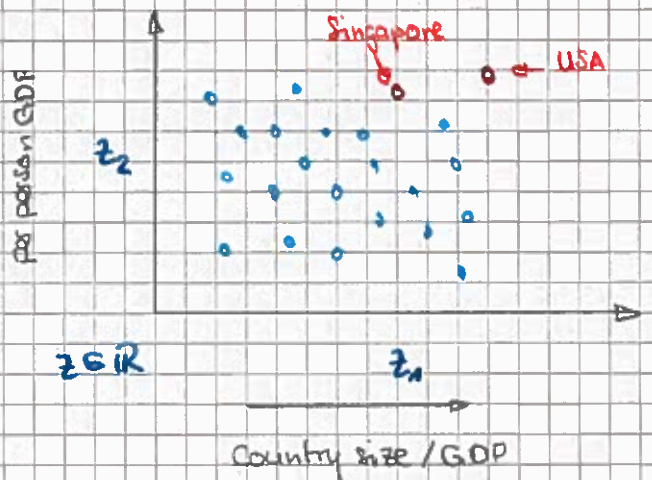
$m = 50$ $x^{(i)} \in \mathbb{R}^{50}$

⇓

Country	z_1	z_2 ← no particular meaning
Canada	1.6	1.2
China	⋮	⋮
India	⋮	⋮
Singapore	0.5	1.7
USA	2	1.5

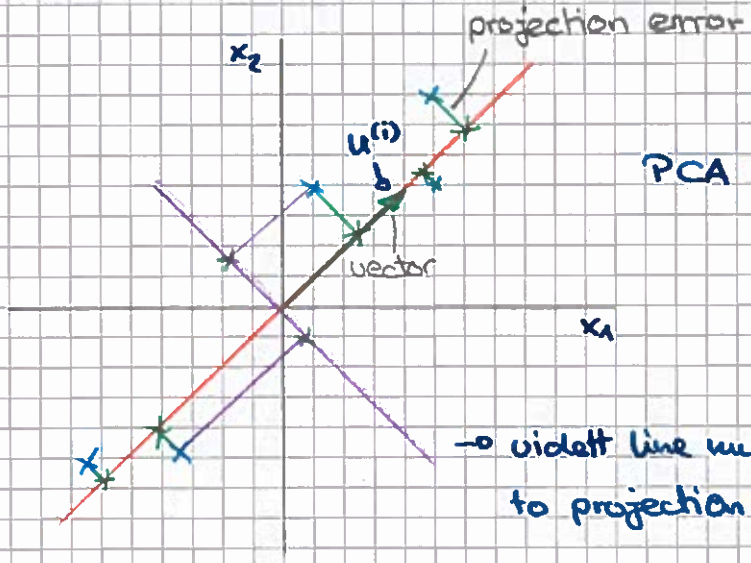
$z^{(i)} \in \mathbb{R}^2$

Reduce the data from 50D to 2D



$k \leq n$

Principal Component Analysis (PCA)



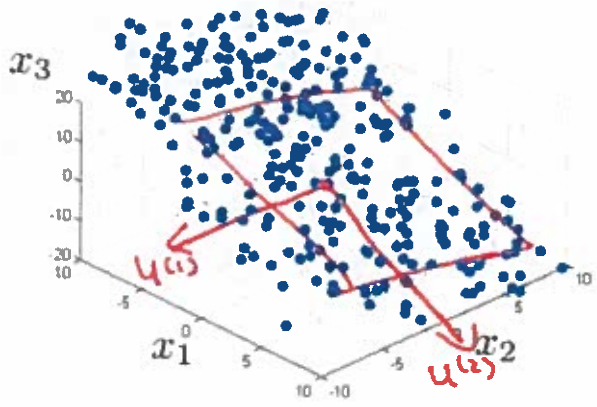
$x \in \mathbb{R}^2$

PCA tries to find lower dimensional projection $\Rightarrow$ projection error to be min.

$\rightarrow$ violett line much worse then red line in respect to projection error.

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.
$\Rightarrow$ it doesent matter if vector is $u^{(1)}$ or $-u^{(1)}$

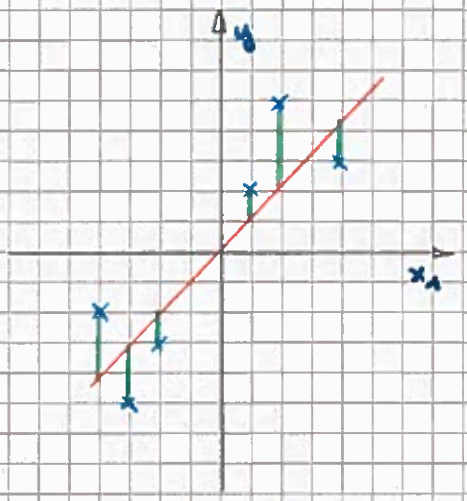
Reduce from n-dimensional to k-dimensional: Find vectors $u^{(1)}, u^{(2)} \ldots u^{(k)}$ onto which to project the data, so to minimize the projection error.



3D $\rightarrow$ 2D
$k = 2$

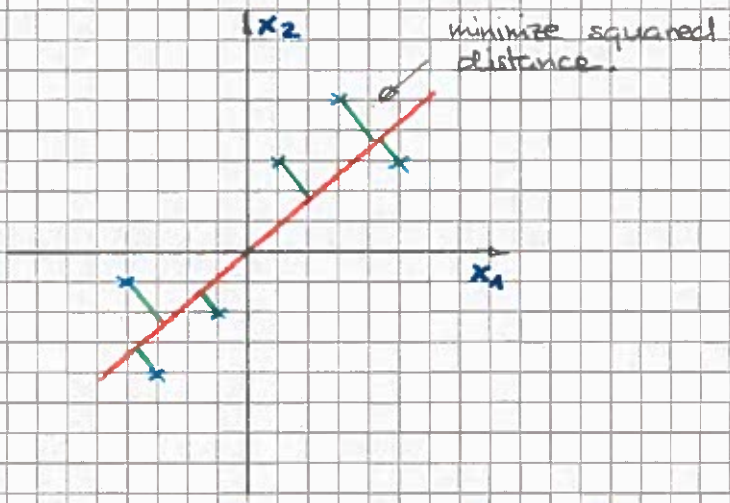
- tries to find a line to minimize the projection error
- with PCA all features are treated equally

PCA is not Linear Regression



prediction of y based on x_1 value (feature)

$x \rightarrow y$



⇒ projection, each feature is treated equally

$x_1, x_2 \ldots x_n$

both are very different algorithms.

Data Preprocessing

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

Processing (feature scaling / mean normalization):

↳ for unlabeled data

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

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

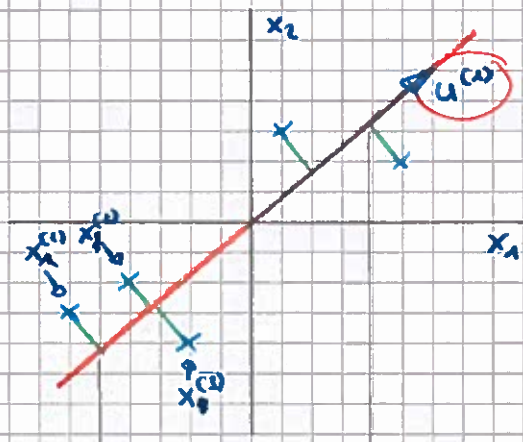
If different features on different scales (e.g. x_1 = size of house, x_2 = # of bedrooms), scale features to have comparable ranges of values.

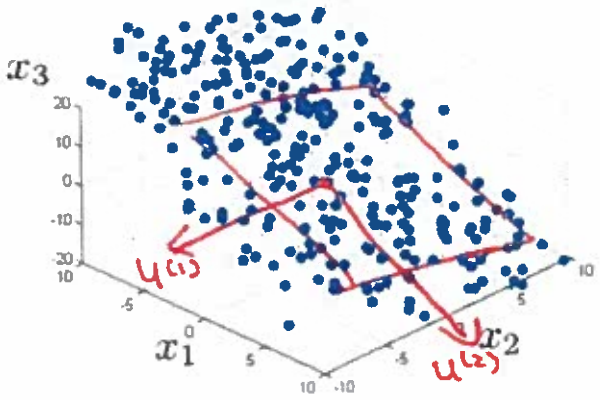
$$x_j \leftarrow \frac{x_j^{(i)} - \mu_j}{s_j}$$

→ resp. standard dev.

or max - min value

Principal Component Analysis Algorithm





Reduce data from 2D to 1D

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

Reduce data from 3D to 2D

$x^{(i)} \in \mathbb{R}^3 \rightarrow z^{(i)} \in \mathbb{R}^2$

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

Reduce data from n-dimensions to k-dimensions

Compute the "covariance matrix"

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

Compute "eigenvectors" of matrix Σ:

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

singular value decomposition

eig(Sigma) => less stable as svd

U ↓ required matrix

Sigma ↓ $n\times n$ matrix ~~vector~~

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

$x\in\mathbb{R}^n \rightarrow z\in\mathbb{R}^k$

$$z^{(i)} = \underbrace{\begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \cdots & u^{(k)} \\ | & | & & | \end{bmatrix}^T}_{n\times k} \cdot x^{(i)} = \underbrace{\begin{bmatrix} - & (u^{(1)})^T & - \\ & \vdots & \\ - & (u^{(k)})^T & - \end{bmatrix}}_{k\times n} \cdot \underbrace{x^{(i)}}_{n\times 1}$$

$k\times 1$

Ureduced

$z\in\mathbb{R}^k$

=> After mean normalization (ensure every feature has zero mean) and optionally feature scaling.

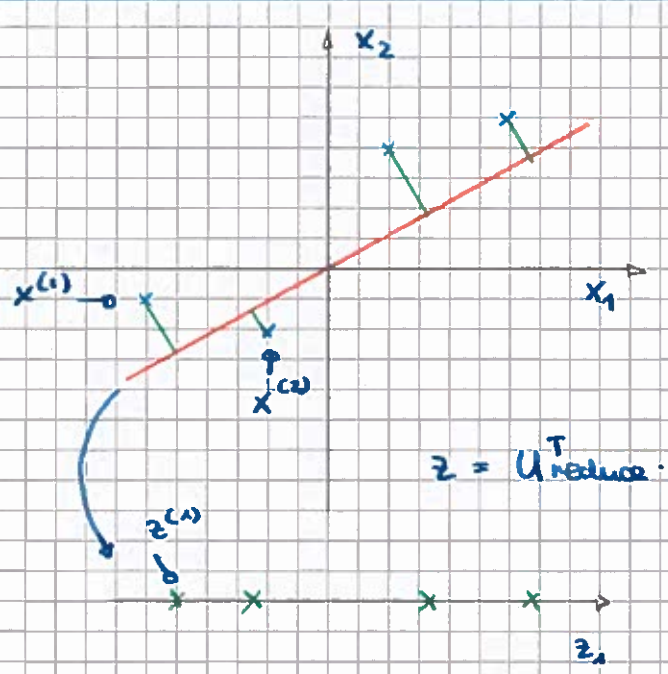
$$\text{Sigma} = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)})(x^{(i)})^T \Rightarrow X = \begin{bmatrix} - & x^{(1)T} & - \\ & \vdots & \\ - & x^{(m)T} & - \end{bmatrix}$$

vectorized: `sigma = (1/m) * X' * X;`

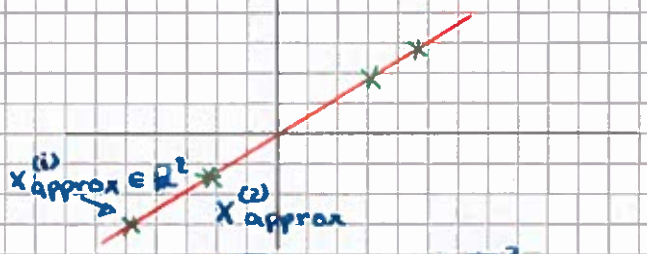
```
[U, S, V] = svd(Sigma)
Ureduce = U(:, 1:k);
z = Ureduce' * x;
```

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

Reconstruction from Compressed Representation



$$z = U_{reduce}^{T} \cdot x$$



$z \in \mathbb{R}$ $\Rightarrow$ $x \in \mathbb{R}^2$

$$x_{approx}^{(1)} = U_{reduce} \cdot z$$

$n \times k$ $\quad$ $k \times 1$

$\Downarrow$

$\mathbb{R}^n$ $\quad$ $n \times 1$

Choosing K (number of Principal Components)

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

Typically choose K to be smallest value so that

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

K → validation

⇒ "99% of variance retained" ⇒ 95% – 99% are usual

Algorithm:

Try PCA with K = 1

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

then check if

→ Formula ⇒ K-Validation $\leq 0{,}01$

if not start over again with K = 2 ...

⇓

Approach inefficient!

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

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

For a given K

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

K = 3 ⇒ $\leq 0{,}01$

⇓

Equivalent to K-Validation formula

$$\frac{\sum_{i=1}^{K} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \geq 0{,}99$$

calculate this for different K values

Smallest value of K

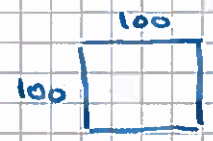
Pick smallest value of K for which the result is: "99% of variance retained"

svd(Sigma) must only be called once

Supervised learning speedup

Very high feature vector
assume computer vision problem

$\Rightarrow (x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots (x^{(m)}, y^{(m)})$ $\Rightarrow x^{(i)} \in \mathbb{R}^{10'000}$



100 × 100 pixels

Extract inputs:

Unlabeled data set: $x^{(1)}, x^{(2)}, \dots, x^{(m)} \in \mathbb{R}^{10'000}$

$\Downarrow$ PCA

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

for prediction x

reduced by 10 x

z

New training set:

$(z^{(1)}, y^{(1)}), (z^{(2)}, y^{(2)}), \dots, (z^{(m)}, y^{(m)})$ $\Rightarrow h_\theta(z) = \frac{1}{1+e^{-\theta^T z}}$

Note: Mapping $x^{(i)} \to z^{(i)}$ 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 set.

$\Rightarrow$ is calculating U_{reduce} $\Rightarrow$ parameter learned by PCA as well as mean normalization

$\Rightarrow$ PCA makes a mapping between $x \to z$

Application of PCA

- Compression
  - Reduce memory/disk needed to store data
  - Speed up learning algorithm
  - Choose k by % of variance retained.

- Visualization

  only $k = 2$ or $k = 3$ could be used to plot data.

Bad use of PCA: To prevent overfitting

Use $z^{(i)}$ instead $x^{(i)}$ to reduce the number of features to $k < n$
⇒ this makes it less likely that overfitting occures.

⇒ BAD use!

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

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

⇒ PCA throughs away ⇒ therefore might loose valuable information

PCA is sometime used where it shouldn't be

Design of ML system:
- Get training set: $(x^{(i)}, y^{(i)})$
- 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 $(z_{test}^{(1)}, y_{test}^{(1)}), \dots (z_{test}^{(m)}, y_{test}^{(m)})$

} Project plan

⇒ How about doing without PCA?

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