Aprendizaje Automático

Based on material by Andrew Ng from Stanford University (Machine Learning course)

Dimensionality Reduction

Máster en Bioinformática y Biología Computacional

Dimensionality Reduction

Dimensionality Reduction algorithms:

- Map high-dimensional data to a lower dimension
- While preserving structure

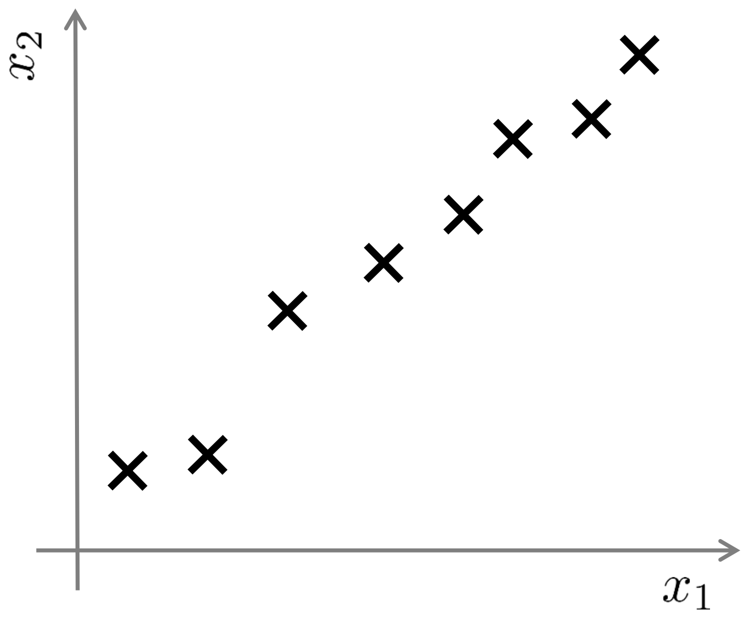
They are used for:

- Visualization
- Performance
- Curse of dimensionality

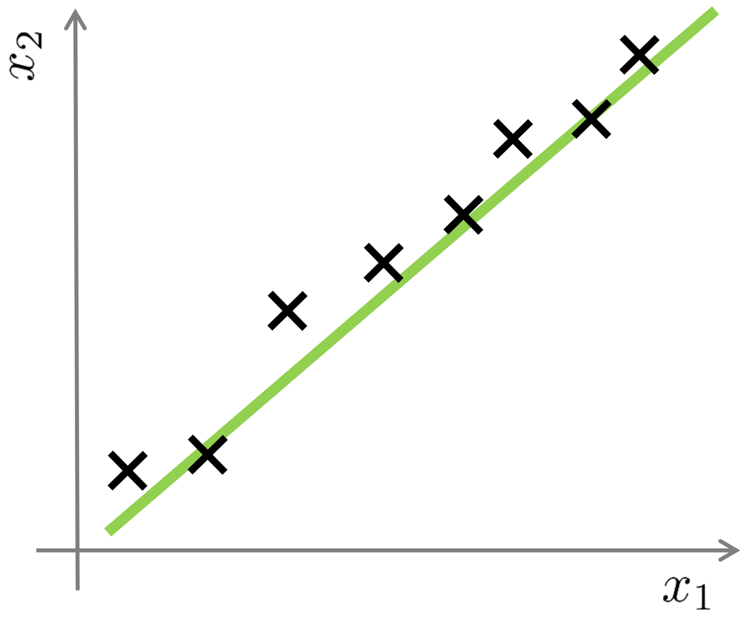
A ton of algorithms exist:

- t-SNE is specialised for visualization
- ... and has gained a lot of popularity

From 2D to 1D

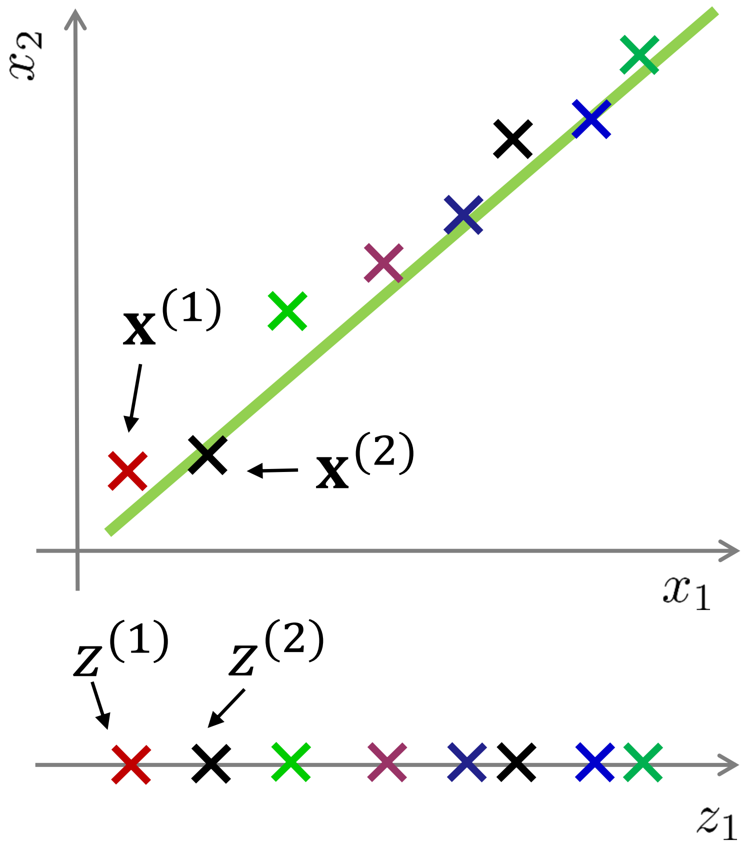


From 2D to 1D



Reduce correlation of feats

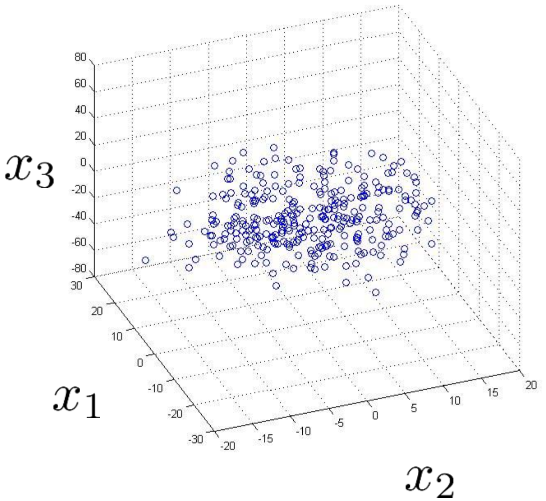
From 2D to 1D



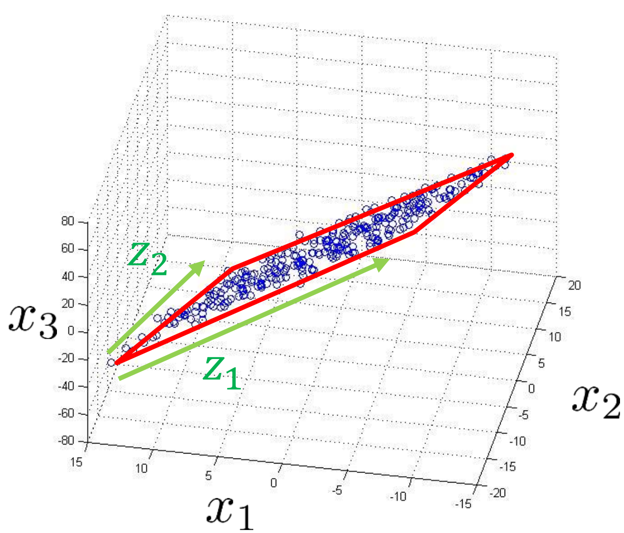
$$\mathbf{x}^{(1)} \in \Re^2 \rightarrow z^{(1)} \in \Re$$
$$\mathbf{x}^{(2)} \in \Re^2 \rightarrow z^{(2)} \in \Re$$
$$\vdots$$
$$\mathbf{x}^{(N)} \in \Re^2 \rightarrow z^{(N)} \in \Re$$

From 3D to 2D

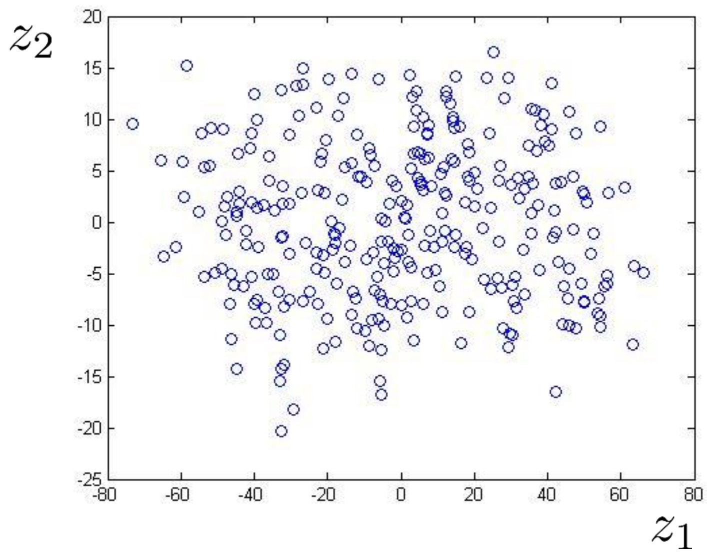
Original dataset



Projected dataset



Reduce data from 3D to 2D



$$\mathbf{x}^{(i)} \in \Re^3 \rightarrow \mathbf{z}^{(i)} \in \Re^2$$

Data Visualization

	x_1	x_2	x_3	x_4	x_5	x_6	x_{2000}
Sub_ID	rs307377	rs7366653	rs41307846	rs3753242	rs35082957	rs34154371	…
HG00001	1	0	1	1	0	0	…
HG00002	0	0	1	1	1	0	…
HG00003	1	1	0	0	0	1	…
HG00004	0	0	0	1	0	1	…
HG00005	0	0	1	1	1	1	…
…							

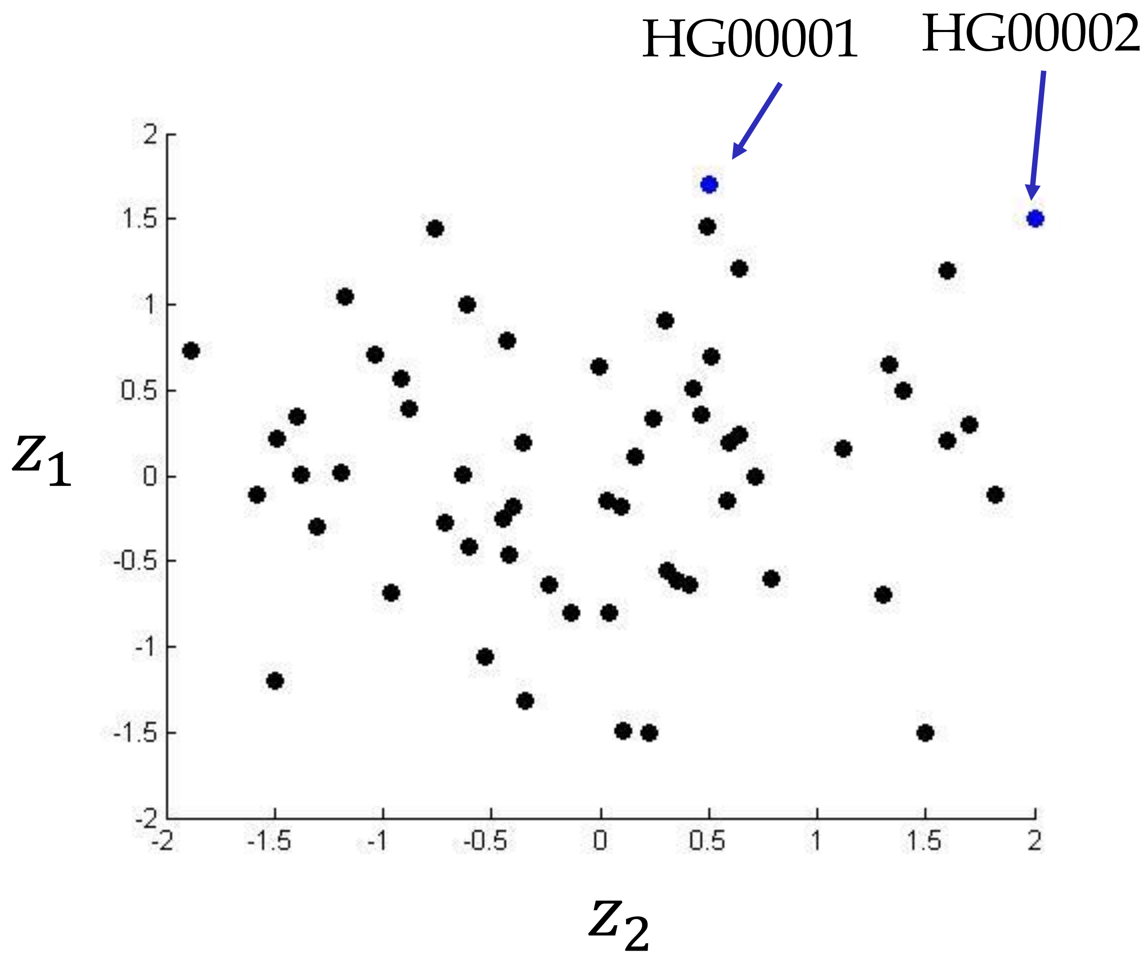
$$\mathbf{x}^{(i)} \in \mathbb{B}^{2000}$$

How can we understand our data better?
How can we reduce from 2000D to 2D

Data Visualization

$$\mathbf{z}^{(i)} \in \Re^2$$

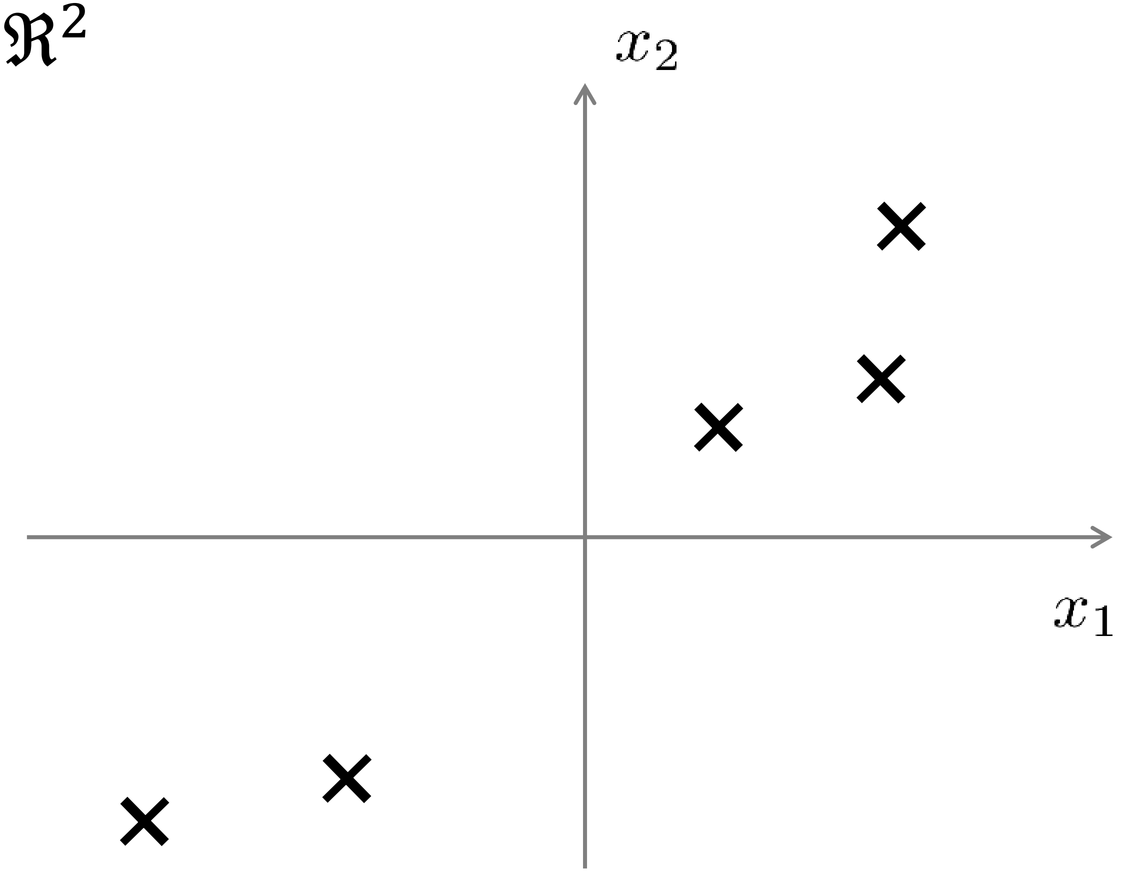
Sub_ID	z_1	z_2
HG00001	0.65	0.71
HG00002	0.43	2.43
HG00003	0.03	1.14
HG00004	5.40	2.11
HG00005	2.33	0.46
…		



Drawback, the new axes do not have any physical meaning

Principal Component Analysis: Formulation

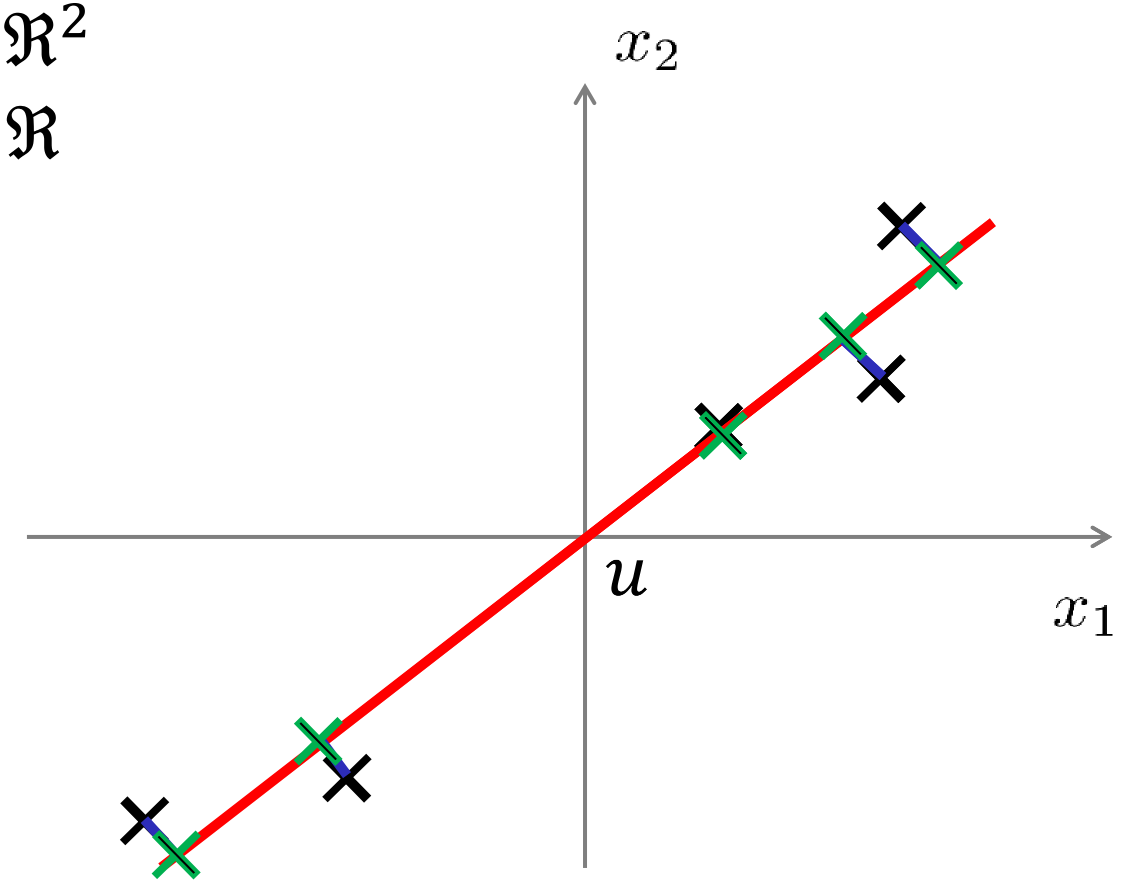
$\mathbf{x}^{(i)} \in \Re^2$



Principal Component Analysis: Formulation

$\mathbf{x}^{(i)} \in \Re^2$

$u^{(i)} \in \Re$

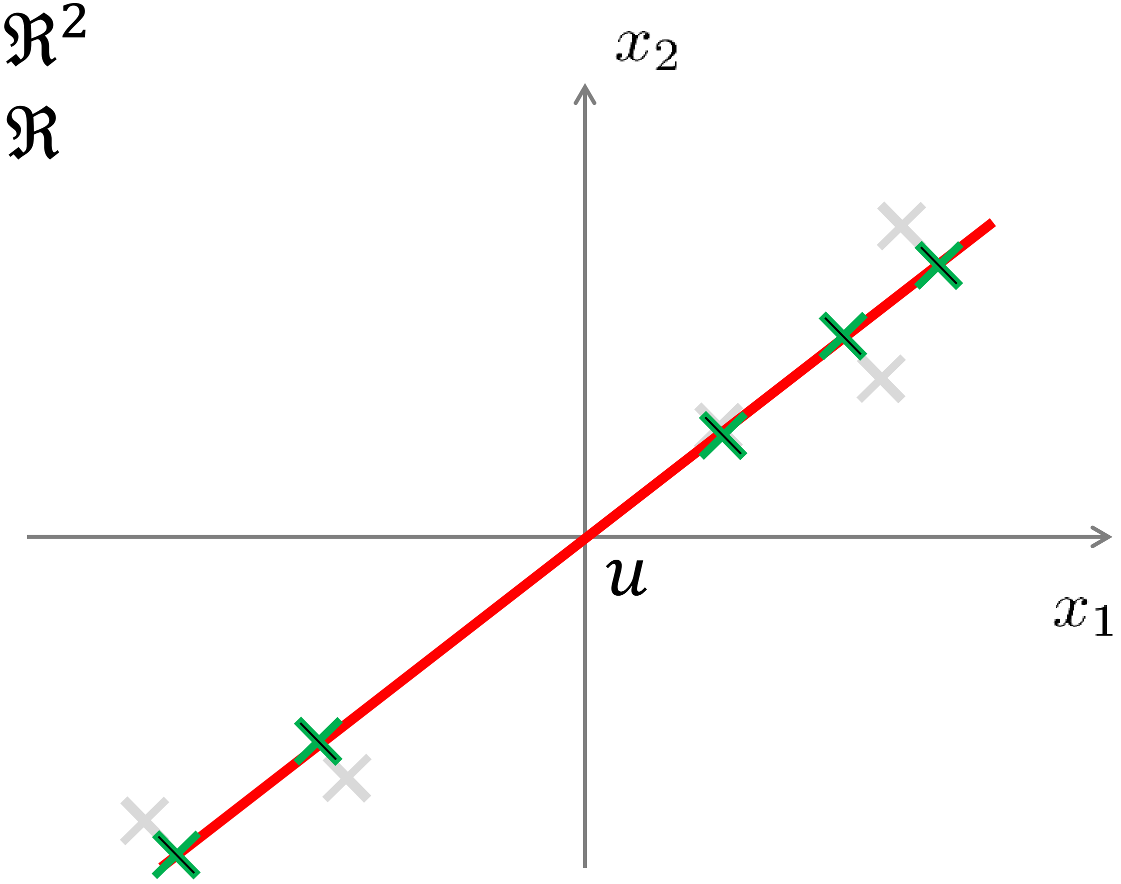


Projection to a line

Principal Component Analysis: Formulation

$\mathbf{x}^{(i)} \in \Re^2$

$u^{(i)} \in \Re$

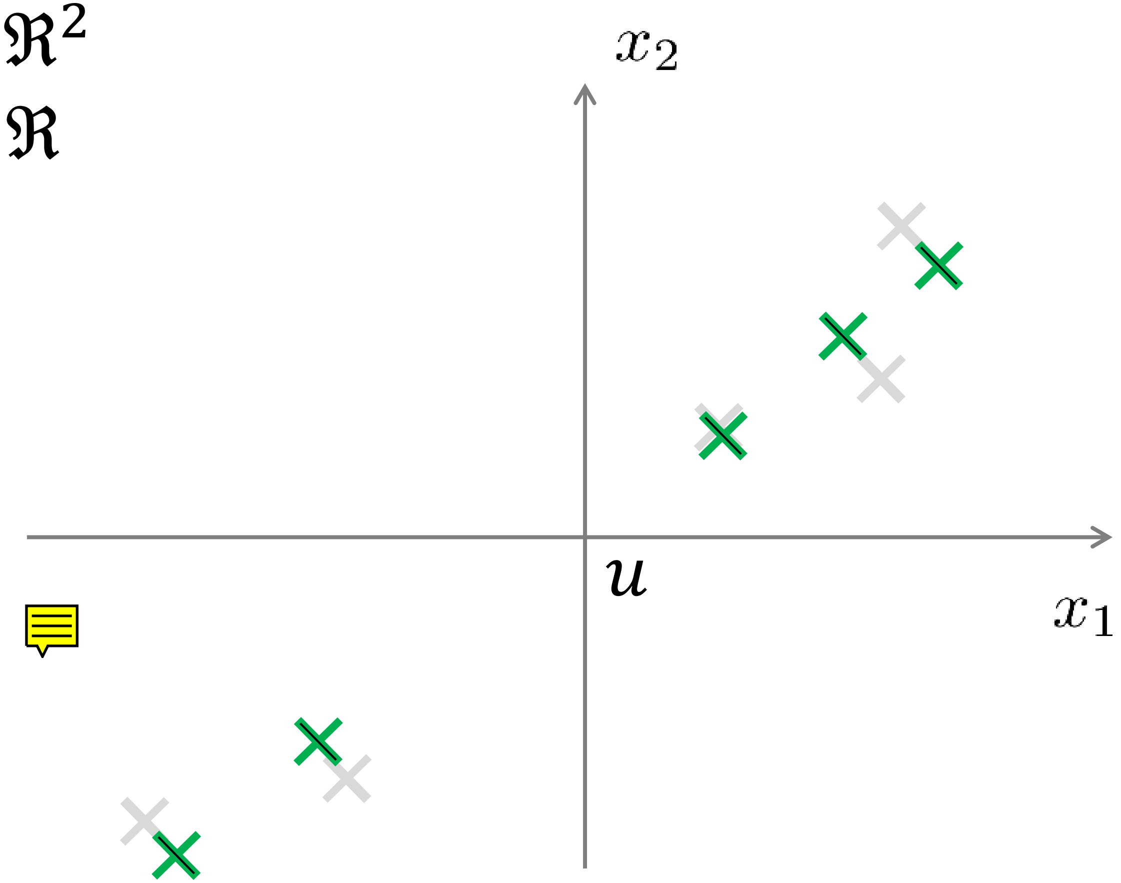


PCA tries to minimize the projection "error"

Principal Component Analysis: Formulation

$\mathbf{x}^{(i)} \in \Re^2$

$u^{(i)} \in \Re$

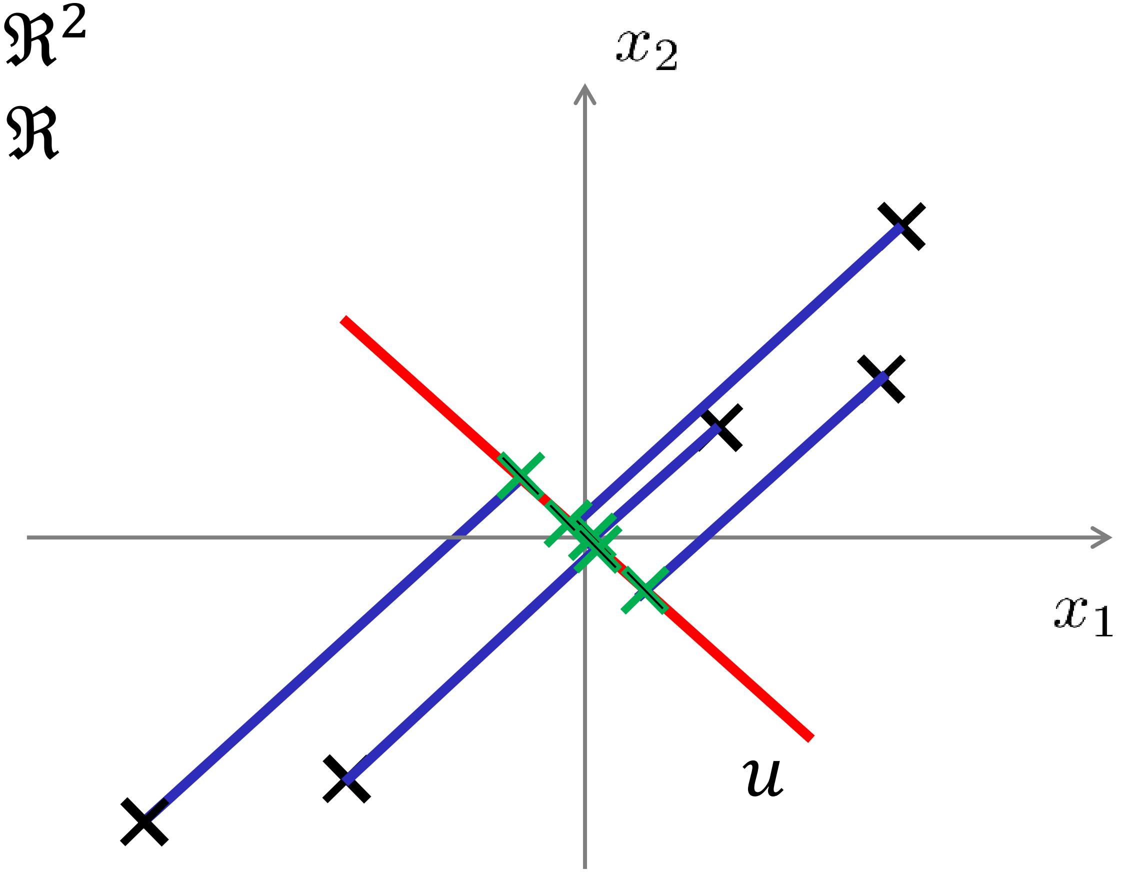


PCA tries to minimize the projection "error"

Principal Component Analysis: Formulation

$\mathbf{x}^{(i)} \in \Re^2$

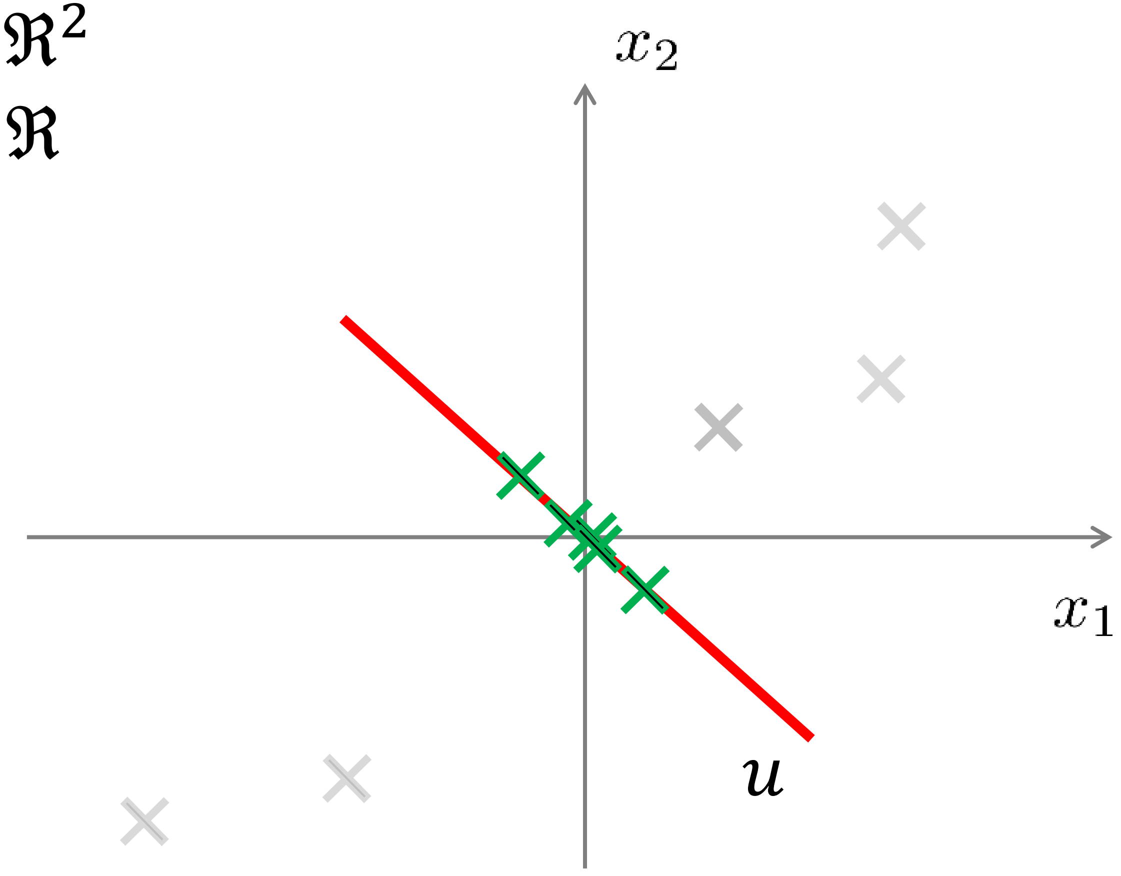
$u^{(i)} \in \Re$



Principal Component Analysis: Formulation

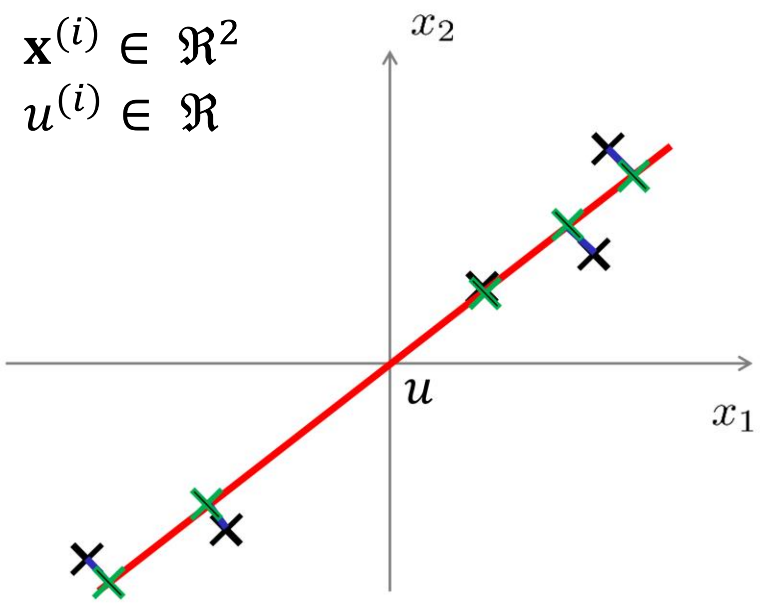
$\mathbf{x}^{(i)} \in \Re^2$

$u^{(i)} \in \Re$



Principal Component Analysis: Formulation

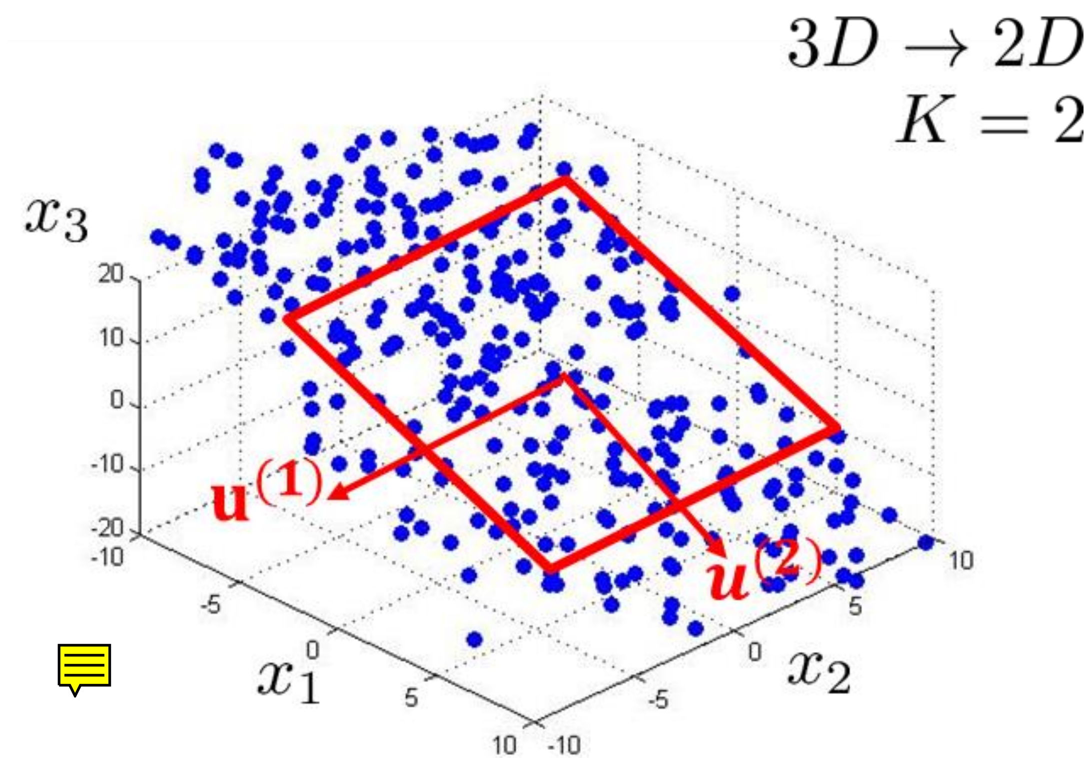
$\mathbf{x}^{(i)} \in \Re^2$
$u^{(i)} \in \Re$



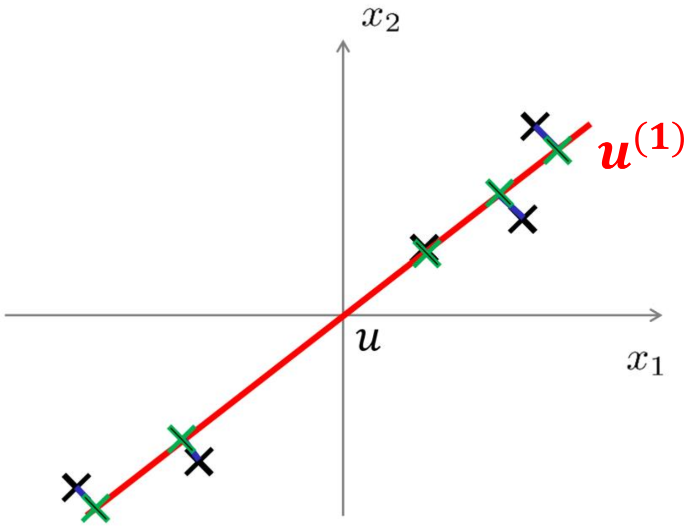
- Reduce from 2-D to 1-D: Find a direction (a vector $u^{(1)} \in \Re$) onto which to project the data so as to minimize the projection error.

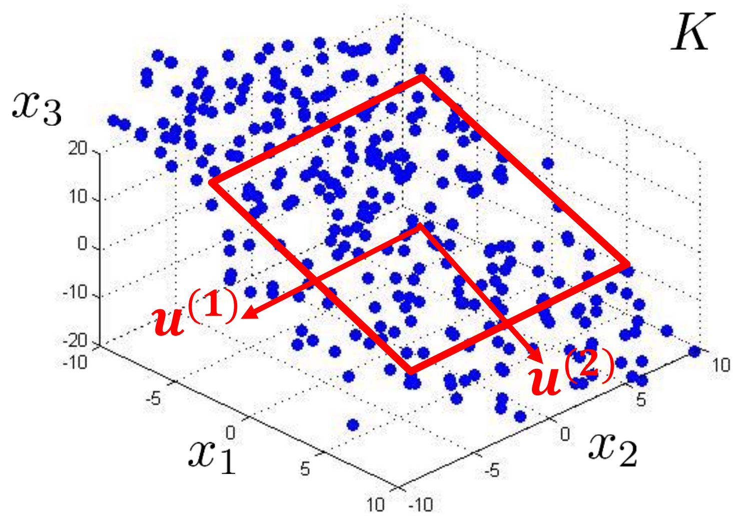
Principal Component Analysis: Formulation

- Reduce from n-dimension to k-dimension: Find k vectors $u^{(1)}, u^{(2)}$, $u^{(3)}$,…. $u^{(k)}$ onto which to project the data, so as to minimize the projection error.



PCA: Algorithm



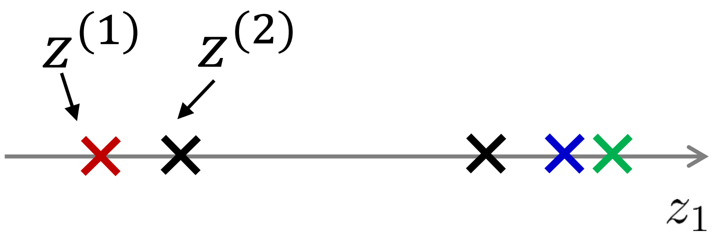


Reduce data from 2D to 1D

$$\mathbf{x}^{(i)} \in \Re^2 \to z^{(i)} \in \Re$$

Reduce data from 3D to 2D

$$\mathbf{x}^{(i)} \in \Re^3 \to \mathbf{z}^{(i)} \in \Re^2$$



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

Principal Component Analysis: Algorithm

- Training set: $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(M)}$
- Data Preprocessing (feature scaling/mean normalization):

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

- If different features on different scales (e.g., $x_1 =$ area, $x_2 =$ growth rate), scale features to have comparable range of values.

$$x_j^{(i)} = \frac{x_j^{(i)} - \mu_j}{\sigma_j}$$

- Where, the standard deviation: $\sigma_j = \sqrt{\frac{1}{M} \sum_{i=1}^{M} (x_j^{(i)} - \mu_j)^2}$

Principal Component Analysis: Algorithm

- Compute "covariance matrix" of $\mathbf{x}$:

$$\Sigma = \frac{1}{M}\sum_{i=1}^{M}\left(\mathbf{x}^{(i)}\right)\left(\mathbf{x}^{(i)}\right)^{T} \in \Re^{n\times n}$$

- Implementation. If we have the X matrix defined as:

$$\mathbf{X} = \begin{bmatrix} - & \left(\mathbf{x}^{(1)}\right)^{T} & - \\ & \vdots & \\ - & \left(\mathbf{x}^{(M)}\right)^{T} & - \end{bmatrix} \in \Re^{M\times n} \rightarrow \Sigma = \left(\frac{1}{M}\right) \times \mathbf{X}^{T} \times \mathbf{X}$$

Principal Component Analysis: Algorithm

- Compute "eigenvalues" of matrix Σ:

$$\boldsymbol{\lambda} = \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_n \end{bmatrix} \in \Re^{n \times 1}$$

- Compute the "eigenvectors" of matrix Σ:

$$\mathbf{U} = \begin{bmatrix} | & & | \\ \mathbf{u}^{(1)} & \dots & \mathbf{u}^{(n)} \\ | & & | \end{bmatrix} \in \Re^{n \times n}$$

Principal Component Analysis: Algorithm

- Order "eigenvectors" according to its "eigenvalues" from higher to lower values. It means higher significance to lower significance

$$\mathbf{U} = \begin{bmatrix} | & & | \\ \mathbf{u}^{(1)} & \dots & \mathbf{u}^{(n)} \\ | & & | \end{bmatrix} \in \Re^{n \times n}$$

$$\boldsymbol{\lambda}' = \begin{bmatrix} \lambda'_1 \\ \vdots \\ \lambda'_n \end{bmatrix} \in \Re^{n \times 1} \qquad \lambda'_1 > \lambda'_2 > \cdots > \lambda'_n$$

$$\mathbf{U}' = \begin{bmatrix} | & & | \\ (\mathbf{u}')^{(1)} & \dots & (\mathbf{u}')^{(n)} \\ | & & | \end{bmatrix} \in \Re^{n \times n}$$

Principal Component Analysis: Algorithm

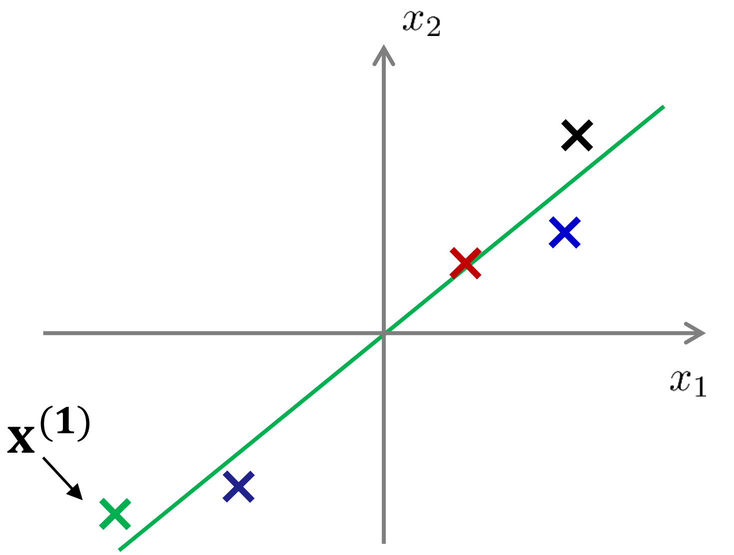
- Select the components with higher significance:

$$\mathbf{U}' = \begin{bmatrix} | & & | \\ (\mathbf{u}')^{(1)} & \dots & (\mathbf{u}')^{(n)} \\ | & & | \end{bmatrix} \in \Re^{n\times n}$$

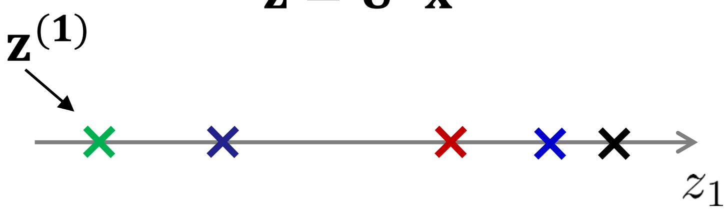
$$\mathbf{x} \in \Re^{n} \rightarrow \mathbf{z} \in \Re^{k}$$

$$\mathbf{z} = \underbrace{\begin{bmatrix} | & & | \\ (\mathbf{u}')^{(1)} & \dots & (\mathbf{u}')^{(k)} \\ | & & | \end{bmatrix}^T}_{n\times k} \underbrace{\mathbf{x}}_{n\times 1} = \underbrace{\begin{bmatrix} - & \left((\mathbf{u}')^{(1)}\right)^T & - \\ & \vdots & \\ - & \left((\mathbf{u}')^{(k)}\right)^T & - \end{bmatrix}}_{\substack{k\times n \\ \breve{\mathbf{U}}^T = \mathbf{U}'^T \text{ reduced}}} \underbrace{\mathbf{x}}_{n\times 1} = \underbrace{\mathbf{z}}_{k\times 1} \in \Re^{k}$$

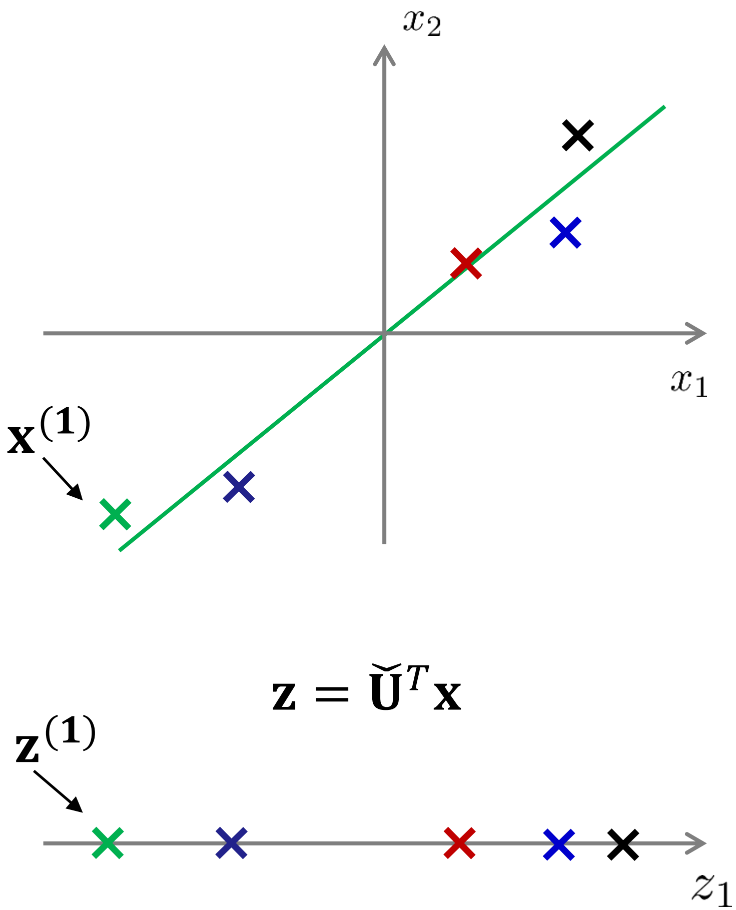
Reconstruction from compressed representation



$$\mathbf{z} = \breve{\mathbf{U}}^T \mathbf{x}$$



Reconstruction from compressed representation



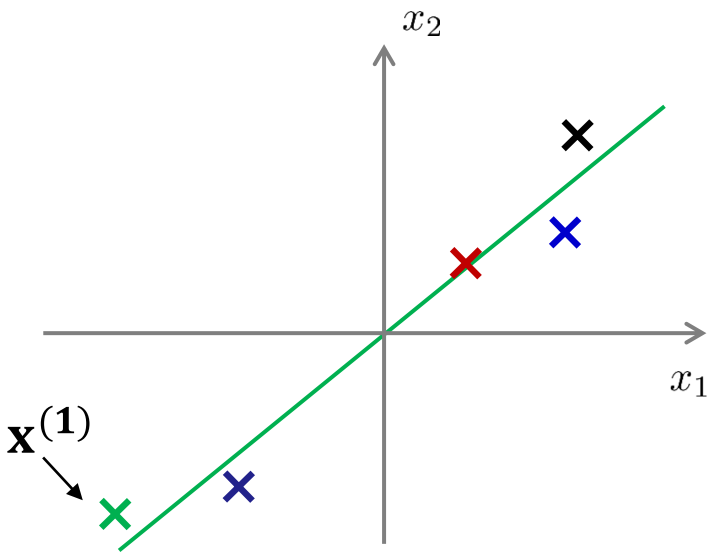
$$\mathbf{z} = \breve{\mathbf{U}}^T \mathbf{x}$$

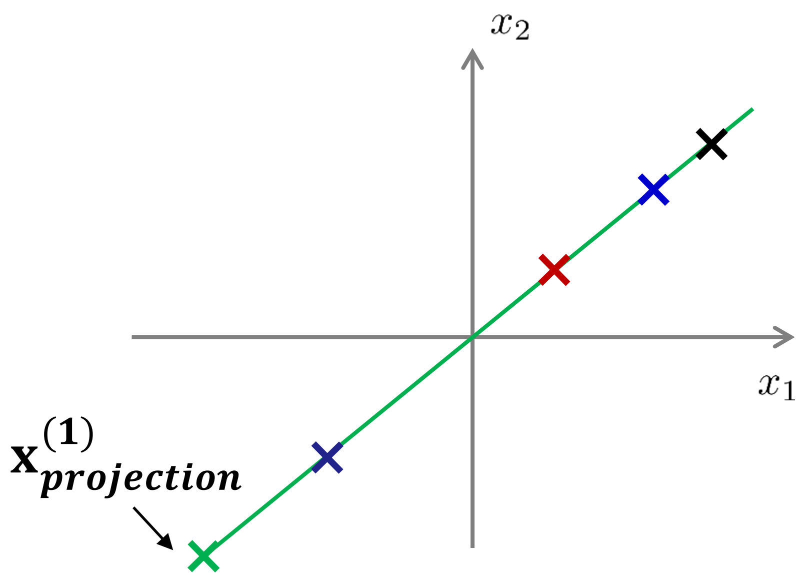
$$\mathbf{z} \in \Re \rightarrow \mathbf{x} \in \Re^2$$

$$\mathbf{x}_{projection} = \breve{\mathbf{U}}\,\mathbf{z} \approx \mathbf{x}$$

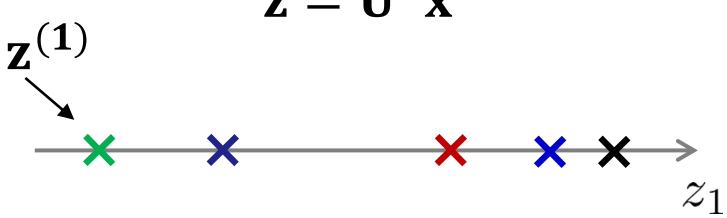
$n \times 1$ $\quad$ $n \times k$ $\quad$ $k \times 1$

Reconstruction from compressed representation





$$\mathbf{z} = \breve{\mathbf{U}}^T \mathbf{x}$$



$$\mathbf{z} \in \Re \rightarrow \mathbf{x} \in \Re^2$$

$$\underbrace{\mathbf{x}_{projection}}_{n \times 1} = \underbrace{\breve{\mathbf{U}}}_{n \times k} \underbrace{\mathbf{z}}_{k \times 1} \approx \mathbf{x}$$

Choosing the number of principal components

- Average squared projection error $= \frac{1}{M}\sum_{i=1}^{M} \underbrace{\left\| \mathbf{x}^{(i)} - \mathbf{x}_{projection}^{(i)} \right\|^2}_{\text{Variance between data and projections}}$ (what PCA minimizes)

- Total variation in the data $= \frac{1}{M}\sum_{i=1}^{M} \left\| \mathbf{x}^{(i)} \right\|^2$

  (the average length squared of the examples)

- Typically, choose k to be smallest value so that:

$$\frac{\frac{1}{M}\sum_{i=1}^{M} \left\| \mathbf{x}^{(i)} - \mathbf{x}_{projection}^{(i)} \right\|^2}{\frac{1}{M}\sum_{i=1}^{M} \|\mathbf{x}^{(i)}\|^2} \leq 0.01$$

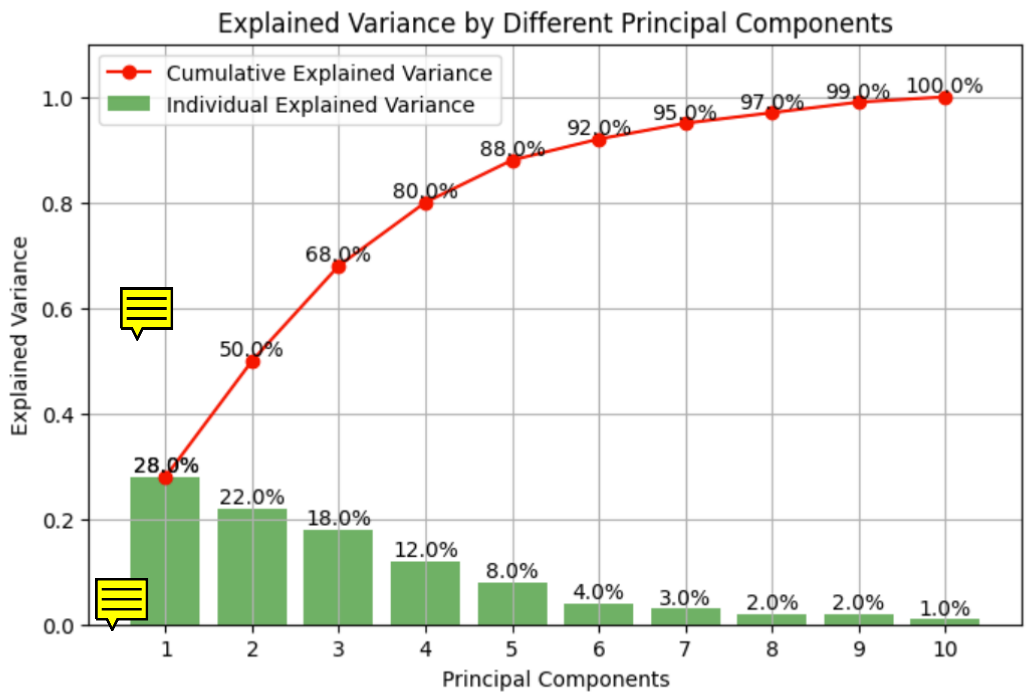
"99% of variance is retained"

Choosing the number of principal components

- Using the eigenvalues to estimate the variance retained in each component:

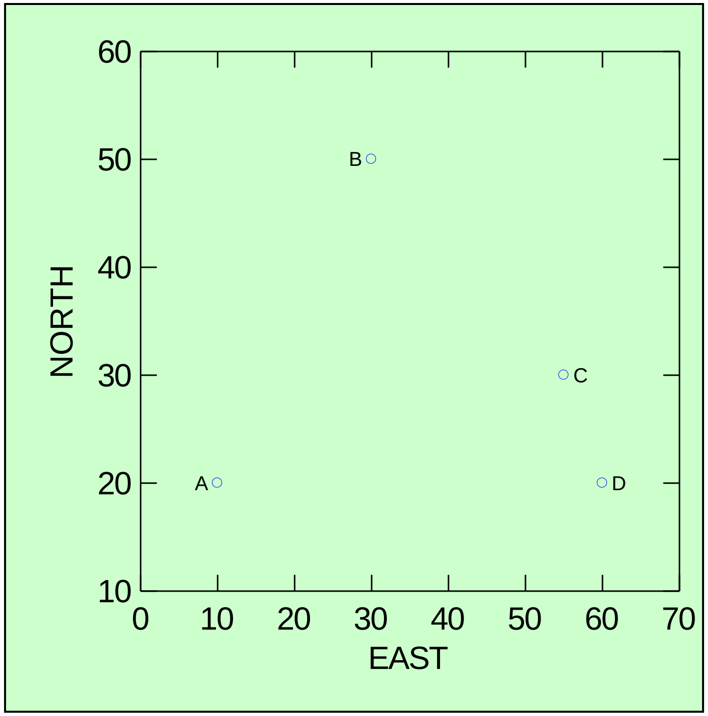
- Variance explained by the component j = $\dfrac{\lambda_j}{\sum_{i=1}^{n} \lambda_i}$

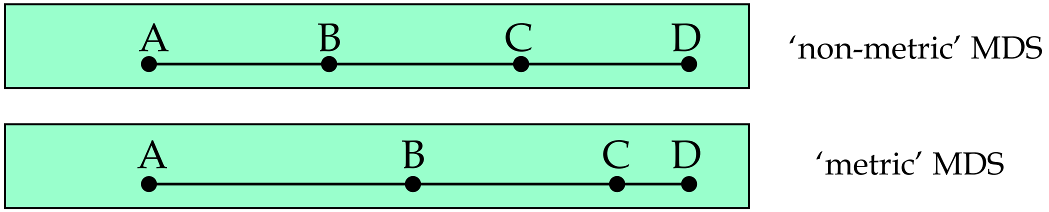
- Plot the cumulative variance ratio:



Dimensionality Reduction: Other Methods

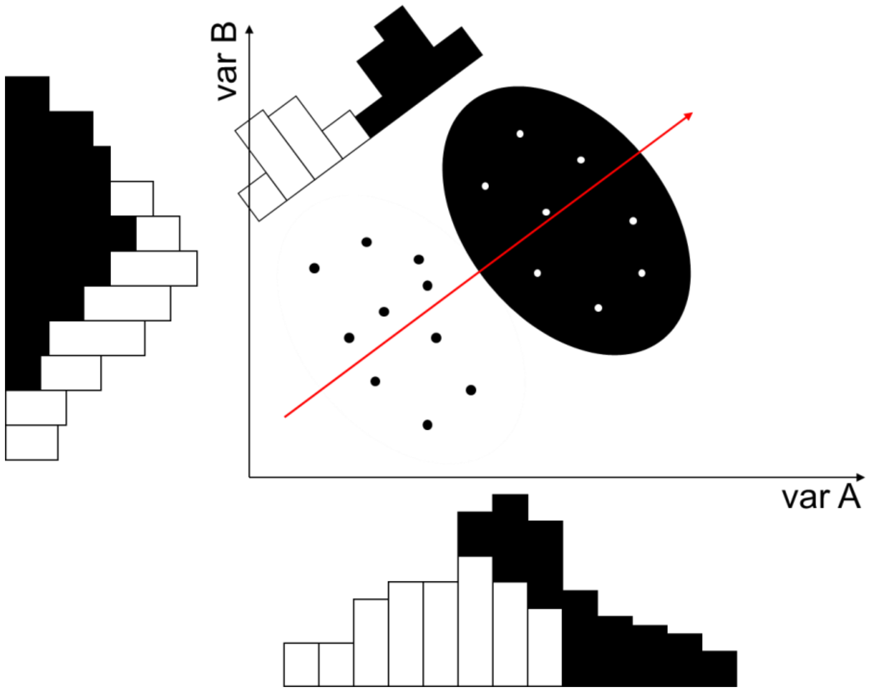
- **Muldimensional Scaling (MDS):** define low-dimension space that preserves the distance between cases in original high-dimension space.





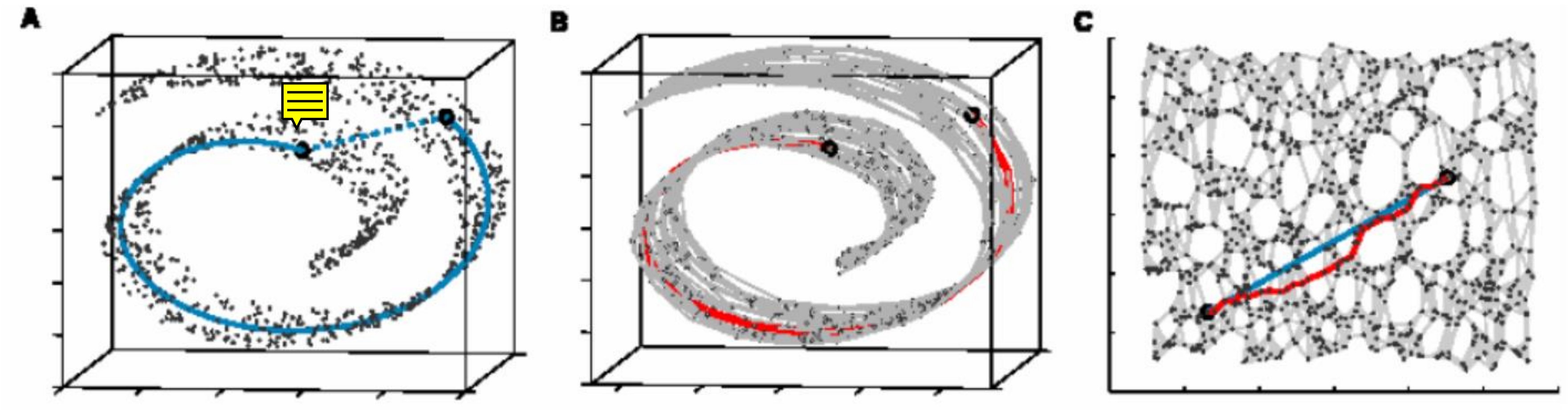
Dimensionality Reduction: Other Methods

- **Discriminant Analysis (e.g., LDA):** calculate a function that maximizes the ability to discriminate among 2 or more groups.



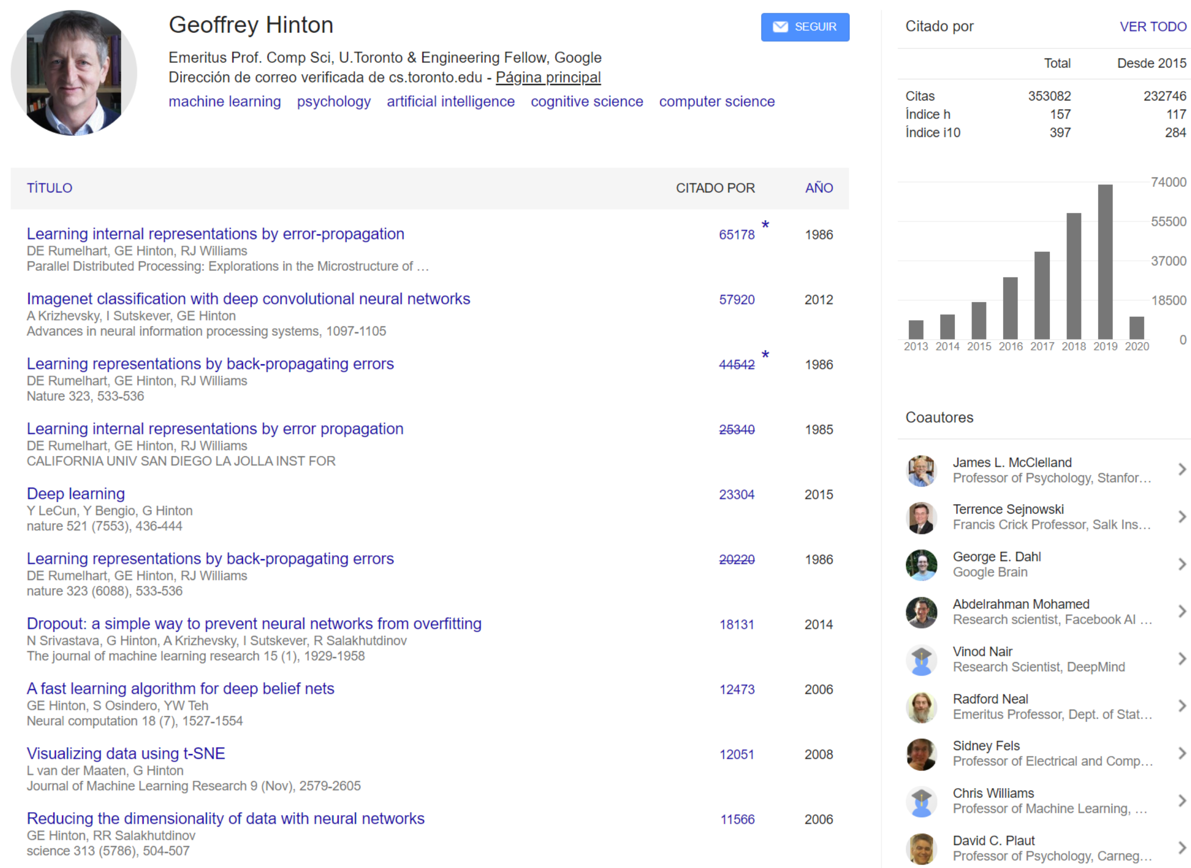
Dimensionality Reduction: Other Methods

- **Manifold Learning (e.g., Isomap)**: discover low dimensional representations in locally Euclidean smooth manifolds.

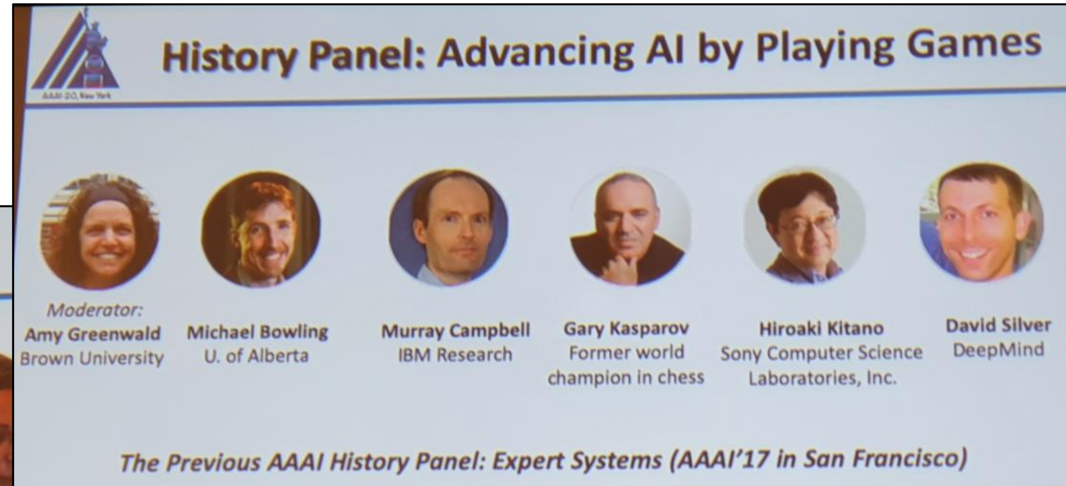


t-SNE

- t-SNE reduces dimensionality while preserving **local similarity**, has been build **heuristically**, and is commonly used to **visualize representations.**



AAAI 2020


History Panel: Advancing AI by Playing Games
AAAI-20, New York
Moderator:
Amy Greenwald
Brown University
Michael Bowling
U. of Alberta
Murray Campbell
IBM Research
Gary Kasparov
Former world champion in chess
Hiroaki Kitano
Sony Computer Science Laboratories, Inc.
David Silver
DeepMind
The Previous AAAI History Panel: Expert Systems (AAAI'17 in San Francisco)


Invited Speakers
AAAI-20, New York
Geoffrey Hinton
U. of Toronto
Yann LeCun
Facebook and NYU
Yoshua Bengio
U. de Montreal
Turing Award Winner Event!
Aude Billard
EPFL
(yesterday ☹)
Benjamin Shapiro
Abigail Zimmerman--Niefield
EAAI Invited Talk
Henry Kautz
NSF and U. Rochester
AAAI/IAAI Robert S. Engelmore Award Lecture
Dawn Song
UC Berkeley
AAAI/IAAI Joint Invited Talk
Susan Athey
Stanford U.
Stuart Russell
UC Berkeley
David Cox
IAAI Invited Talk


New York Hilton
MIDTOWN

Aprendizaje Automático

Based on material by Andrew Ng from Stanford University (Machine Learning course)

Dimensionality Reduction

Máster en Bioinformática y Biología Computacional