From Raw Data to Meaningful Features

Andreas Damianou

University of Sheffield, UK

Summer School on Machine Learning and Data Science

27 June, 2016
Makerere University, Uganda

Working with data

- Data-science: everything revolves around a **dataset**

- Dataset: the set of data (to be) collected for our algorithms to learn from

- Example: child development dataset

height	weight	
1.33	20	child 1
1.40	28	child 2
1.20	12	child 3
1.28	25	child 4
1.15	18	child 5

Working with data

- Data-science: everything revolves around a **dataset**

- Dataset: the set of data (to be) collected for our algorithms to learn from

- Example: child development dataset

$$
Y = \begin{array}{cl}
\begin{array}{cc} \text{height} & \text{weight} \end{array} & \\
\begin{bmatrix} 1.33 & 20 \\ 1.40 & 28 \\ 1.20 & 12 \\ 1.28 & 25 \\ 1.15 & 18 \end{bmatrix} &
\begin{array}{l} \rightarrow \text{child 1} \\ \rightarrow \text{child 2} \\ \rightarrow \text{child 3} \\ \rightarrow \text{child 4} \\ \rightarrow \text{child 5} \end{array}
\end{array}
$$

Working with data

- Data-science: everything revolves around a **dataset**
- Dataset: the set of data (to be) collected for our algorithms to learn from
- Example: child development dataset

5 rows (instances)
*2 columns (**features**)*

$$Y = \begin{array}{c} \begin{array}{cc} \text{height} & \text{weight} \end{array} \\ \begin{bmatrix} h_1 & w_1 \\ h_2 & w_2 \\ h_3 & w_3 \\ h_4 & w_4 \\ h_5 & w_5 \end{bmatrix} \end{array} \begin{array}{l} \\ \rightarrow y_1 \\ \rightarrow y_2 \\ \rightarrow y_3 \\ \rightarrow y_4 \\ \rightarrow y_5 \end{array}$$

Notation

It's convenient to use notation from linear algebra.

$$
Y = \begin{bmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,d} \\ y_{2,1} & y_{2,2} & \cdots & y_{2,d} \\ \vdots & \vdots & & \vdots \\ y_{n,1} & y_{n,2} & \cdots & y_{n,d} \end{bmatrix}
$$

n rows → *n* instances

d columns → *d* features (dimensions)

So, the matrix Y contains *d*-dimensional data

Notation

It's convenient to use notation from linear algebra.

$$Y = \begin{bmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,d} \\ y_{2,1} & y_{2,2} & \cdots & y_{2,d} \\ \vdots & \vdots & & \vdots \\ y_{n,1} & y_{n,2} & \cdots & y_{n,d} \end{bmatrix}$$

n rows → *n* instances

d columns → *d* features (dimensions)

So, the matrix Y contains *d*-dimensional data

With linear algebra notation, we can write operations more succinctly.

E.g., if *W* is a *d x k* matrix, what does *Y*W* do?
What happens if $k < d$? If $k > d$?

What is “dimensionality”?

- Simply, the number of features *used* for describing each instance.
- In the previous example: height and width.
- The number of features depends on our selection or limitations during data collection.

Feature selection

- Out of many possible feature-sets describing our data, we want to keep only those that help with the particular task **(task-dependent)**.

> ***Feature selection:*** *The task of selecting which features to include in our dataset, out of all the possible features that we could have considered.*

Feature selection

Example: Task is to classify children into the normal vs under-development class.

- Which feature combination makes more sense? Why?
  - [height, weight, gender]

Feature selection

Example: Task is to classify children into the normal vs under-development class.

- Which feature combination makes more sense? Why?
  - [height, weight, gender]
  - [height, gender]

Feature selection

Example: Task is to classify children into the normal vs under-development class.

- Which feature combination makes more sense? Why?
  - [height, weight, gender]
  - [height, gender]
  - [height, weight, gender, BMI] $\quad$ *where* $BMI = \dfrac{weight}{height^2}$

Feature selection

Example: Task is to classify children into the normal vs under-development class.

- Which feature combination makes more sense? Why?
  - [height, weight, gender]
  - [height, gender]
  - [height, weight, gender, BMI]
  - [height, weight, eye color]

$$\textit{where } BMI = \frac{weight}{height^2}$$

Feature selection

Example: Task is to classify children into the normal vs under-development class.

- Which feature combination makes more sense? Why?
  - [height, weight, gender]
  - [height, gender]
  - [height, weight, gender, BMI]
  - [height, weight, eye color]
  - Other Suggestions?

$$\textit{where } BMI = \frac{weight}{height^2}$$

High-dimensional data

- Data having large number of features, *d*
- Examples
  - Micro-array data
  - Images

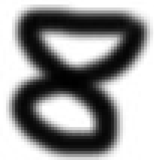
High-dimensional data

- Data having large number of features, *d*
- Examples
  - Micro-array data
  - Images

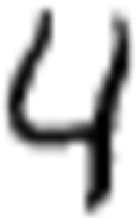
...

$$Y = \begin{bmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,d} \\ y_{2,1} & y_{2,2} & \cdots & y_{2,d} \\ \vdots & \vdots & & \vdots \\ y_{n,1} & y_{n,2} & \cdots & y_{n,d} \end{bmatrix}$$

High-dimensional data



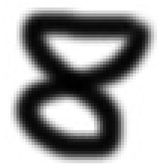
[0, 2, 0, 190, 10, 0, 0, 0, 255, 255, 120, 0, 0, 20, ...]



[2, 4, 0, 8, 20, 120, 0, 255, 187, 42, 0, 122, 0, 1, ...]

[0, 12, 150, 22, 70, 0, 255, 255, 120, 0, 0, 12, 5, ...]

High-dimensional data



[0, 2, 0, 190, 10, 0, 0, 0, 255, 255, 120, 0, 0, 20, ...]

[2, 4, 0, 8, 20, 120, 0, 255, 187, 42, 0, 122, 0, 1, ...]

[0, 12, 150, 22, 70, 0, 255, 255, 120, 0, 0, 12, 5, ...]

With the human eye we can spot the similarity of the 1st and 3rd image. But the computer only sees huge sequences of numbers (pixel intensities)!!

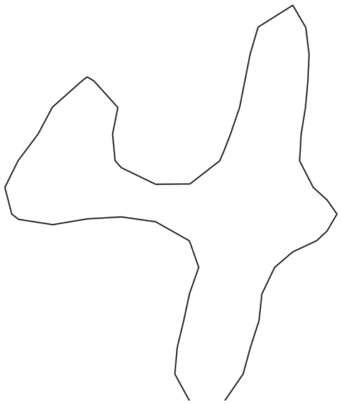
Feature extraction

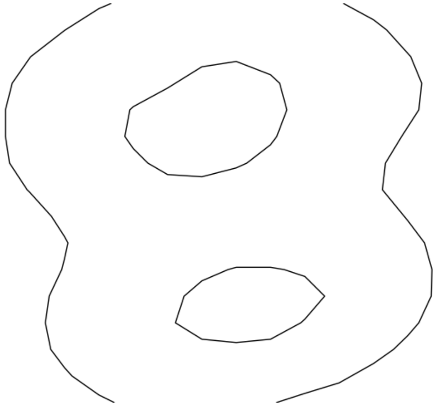
*Create a **new** set of features out of the original ones which came with the raw data. This is done through some algorithm or transformation **dependent on the nature of the data**.*

Contours from images

Original feature space

Extracted feature space





SURF features

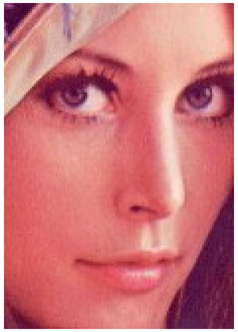
Source: docs.opencv.org

Parenthesis:

Q: How would we do ***Feature Selection*** *to image data?*

Original data

Feature Extraction

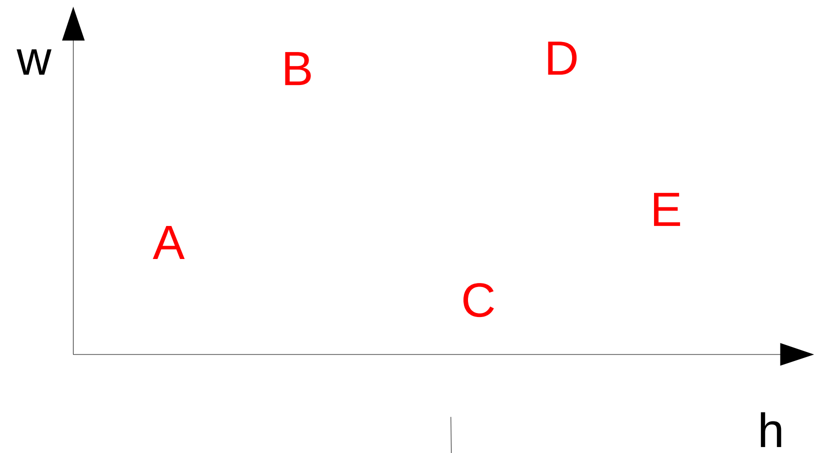


Back to Feature Selection

Dropping one of the oriinal features:

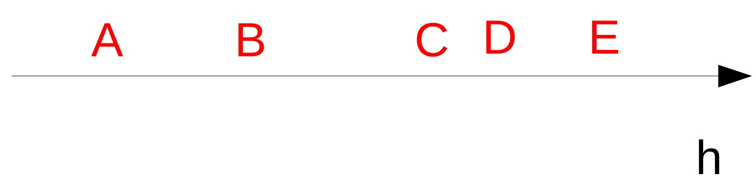
$$
Y = \begin{array}{cc} \text{height} & \text{weight} \\ \end{array}
$$

$$
Y = \left[\begin{array}{cc} h_1 & w_1 \\ h_2 & w_2 \\ h_3 & w_3 \\ h_4 & w_4 \\ h_5 & w_5 \end{array} \right]
$$



$$
\begin{array}{c} \text{height} \end{array}
$$

$$
X' = \left[\begin{array}{c} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \end{array} \right]
$$



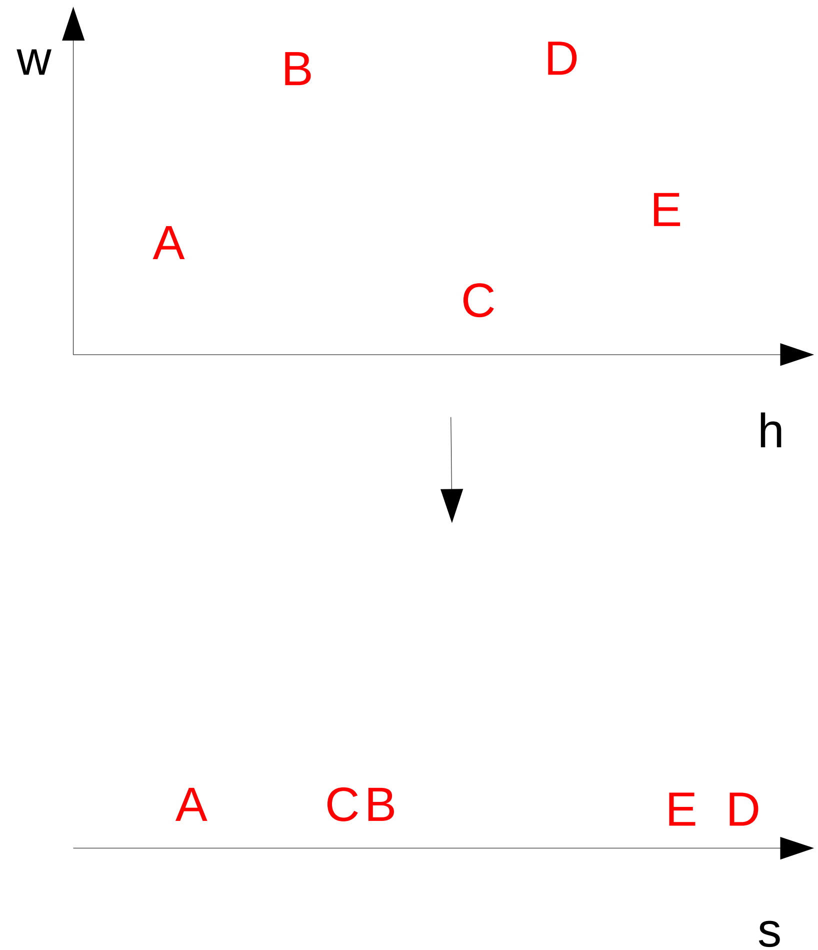
Dimensionality reduction

2nd way: Another solution is to transform the two features into one by projecting them to a **new manifold (geometrical subspace)**.

$$
\begin{array}{cc}
 & \begin{array}{cc} \text{height} & \text{weight} \end{array} \\
Y = & \begin{bmatrix} h_1 & w_1 \\ h_2 & w_2 \\ h_3 & w_3 \\ h_4 & w_4 \\ h_5 & w_5 \end{bmatrix}
\end{array}
$$

$$
\begin{array}{cc}
 & \text{size} \\
X'' = & \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \end{bmatrix}
\end{array}
$$

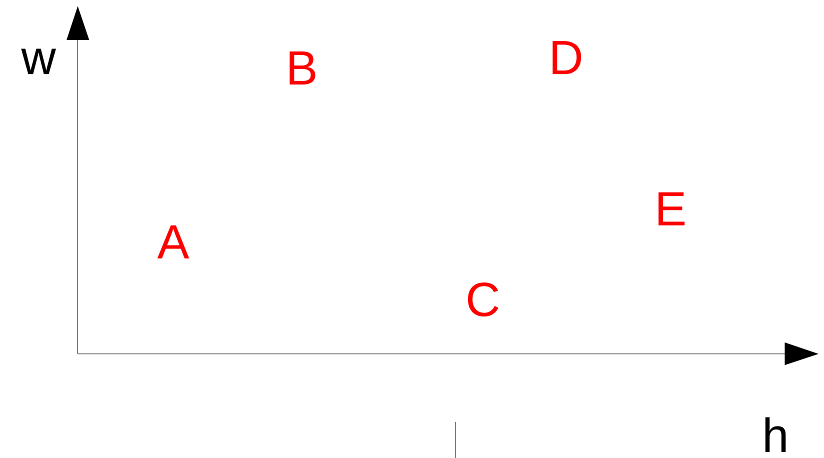
$s = f(h,w)$



Dimensionality reduction

2nd way: Another solution is to transform the two features into one by projecting them to a **new manifold (geometrical subspace)**.

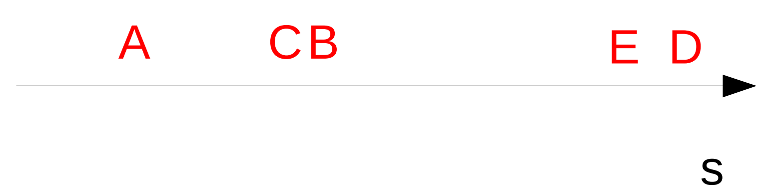
$$Y = \begin{array}{c} \begin{matrix} \text{height} & \text{weight} \end{matrix} \\ \begin{bmatrix} h_1 & w_1 \\ h_2 & w_2 \\ h_3 & w_3 \\ h_4 & w_4 \\ h_5 & w_5 \end{bmatrix} \end{array}$$



<u>But how do we find s?</u>

$$X'' = \begin{array}{c} \text{size} \\ \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \end{bmatrix} \end{array}$$

$s = f(h,w)$

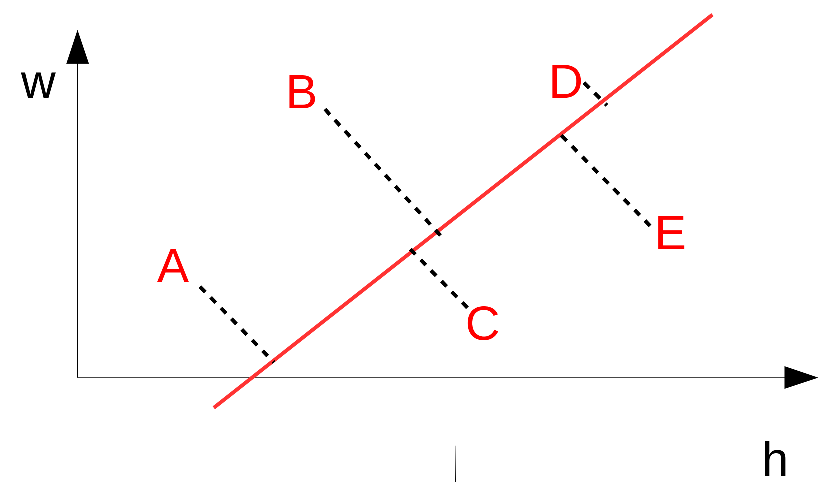


Dimensionality reduction

2nd way: Another solution is to transform the two features into one by projecting them to a **new manifold (geometrical subspace)**.

$$
Y = \begin{array}{cc} \text{height} & \text{weight} \\ \end{array}
$$

$$
Y = \begin{bmatrix} h_1 & w_1 \\ h_2 & w_2 \\ h_3 & w_3 \\ h_4 & w_4 \\ h_5 & w_5 \end{bmatrix}
$$

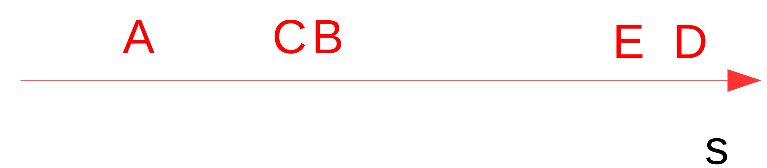


But how do we find s?

$$
\begin{array}{c} \text{size} \end{array}
$$

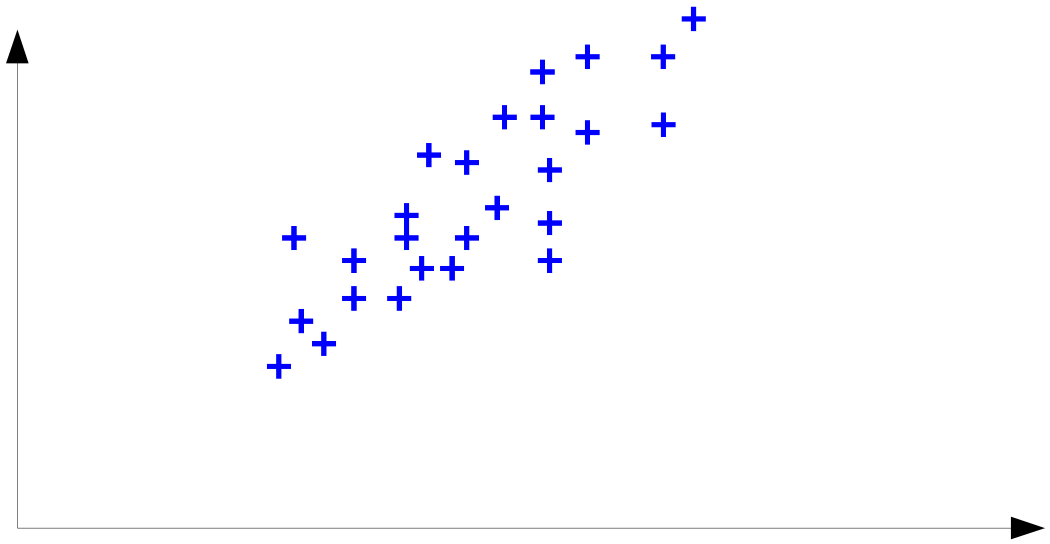
$$
X'' = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \end{bmatrix}
$$

$s = f(h,w)$



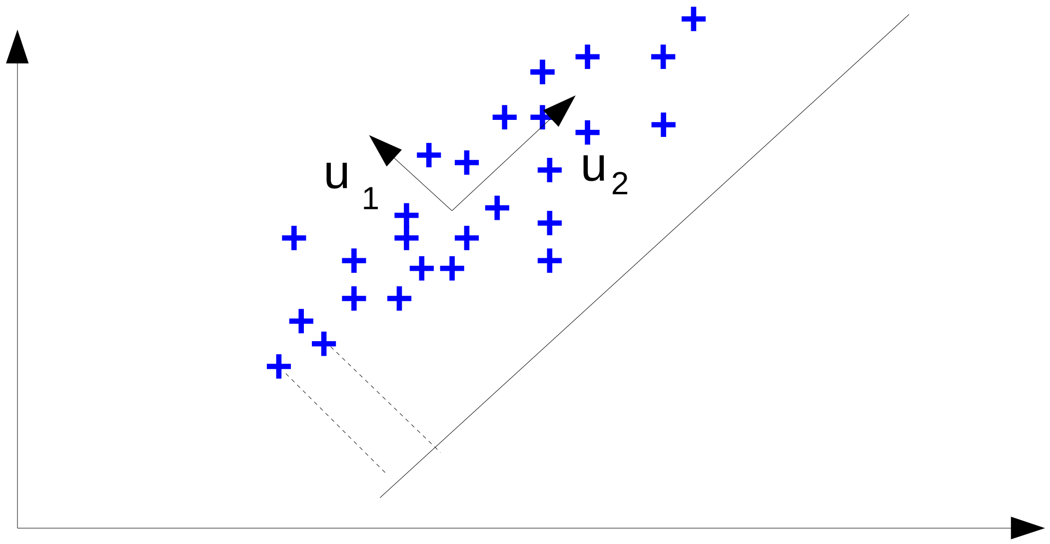
Eigendecomposition

- Gives a feeling of the properties of the matrix



Eigendecomposition

- Gives a feeling of the properties of the matrix



- u1 and u2 define the axes with maximum variances, where the data is most spread
- To reduce the dimensionality I project the data on the axis where data is the most spread
- There is no class information given

Notation

It's convenient to use notation from linear algebra.

$$Y = \begin{bmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,d} \\ y_{2,1} & y_{2,2} & \cdots & y_{2,d} \\ \vdots & \vdots & & \vdots \\ y_{n,1} & y_{n,2} & \cdots & y_{n,d} \end{bmatrix}$$

Remember this slide from before???

n rows → *n* instances

d columns → *d* features (dimensions)

So, the matrix Y contains *d*-dimensional data

With linear algebra notation, we can write operations more succinctly.

E.g., if *W* is a *d x k* matrix, what does *Y**W* do?
What happens if $k < d$?   If $k > d$?

- We need to optimise in order to minimise the distance between the true point and its reconstrcution:

$$x^* = \underset{x}{\operatorname{argmin}} \; || \; y - \operatorname{rec}(x) \; ||_2$$

y is *1 times d*
x is *1 times q*

- *rec(x) = x W'*

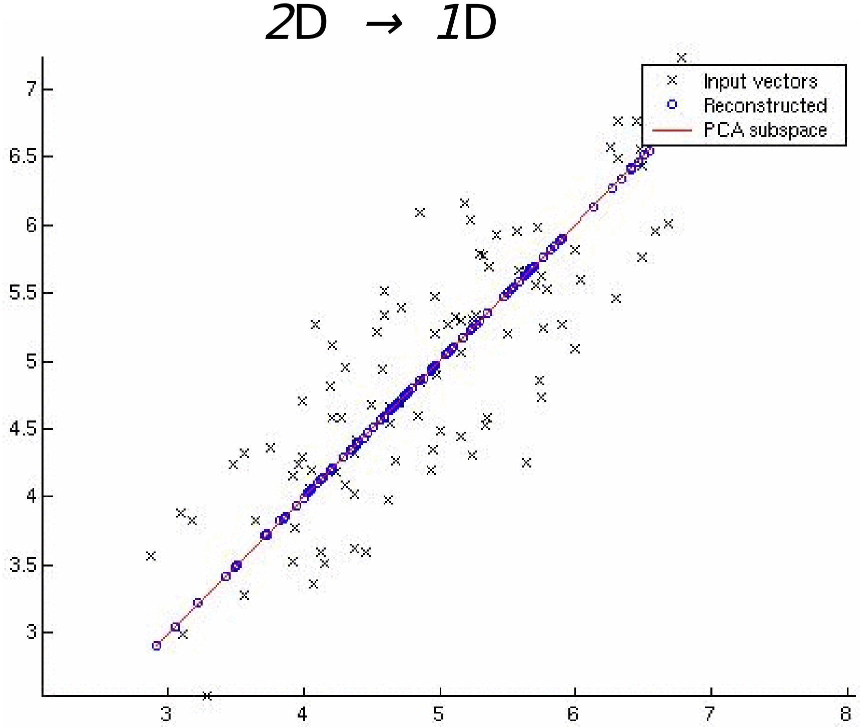
- What is the dimensionality of *W'*?
- Answer: *d times q*

- Then: *x = y W.* Now W is q times d.

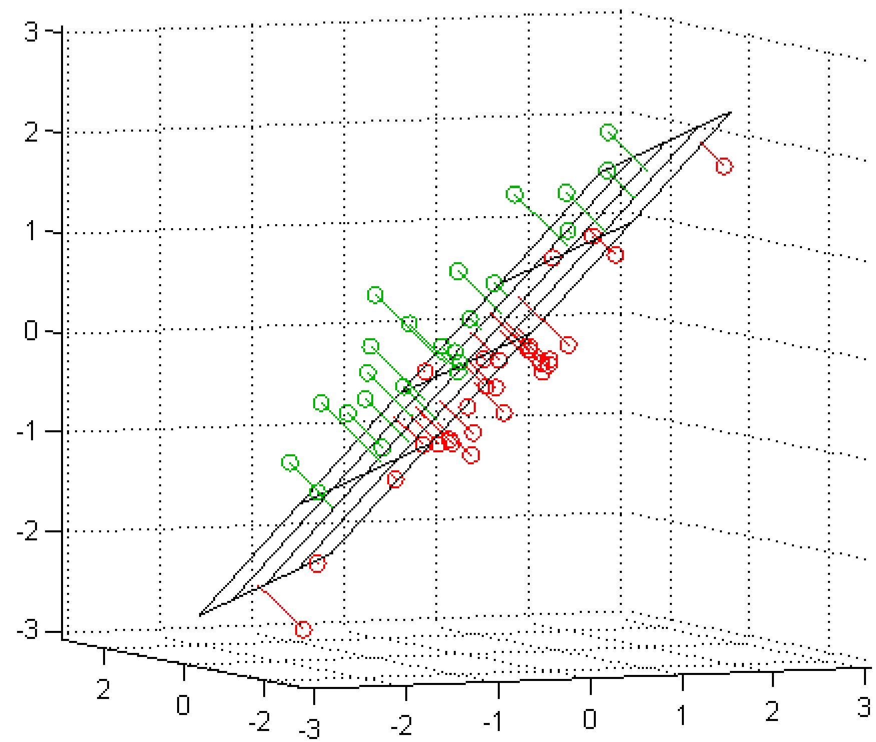
- We can determine both *x* and *W* by solving an optimisation problem (called eigenvalue problem)

Principal Component Analysis

*2*D → *1*D



*3*D → *2*D



- Remember: data are noisy!
- Trade-off: reduce size / noise without losing too much information

Why do dimensionality reduction?

- Pre-process data for another task (e.g. classification)
- Compression (lossy)
- Visualisation
- Data understanding / clearing

Recap

- Data = sets of features, consistent (in nature) across instances
- Raw features are not always ideal.
  - Feature Selection: Drop some of the original features
  - Feature Extraction: Create new features out of the original ones
  - Dimensionality Reduction: Project the features to a new geometrical subspace
- The above methods depend on the task and the nature of the data.