



# Machine Learning

---

ITAM



# Menú

---

- Techniques for dimensionality reduction
  - Attribute selection
    - Select a subset of attributes
  - Feature extraction
    - Principal component analysis (PCA)



# Motivation

---

- The curse of dimensionality
  - This phenomenon refers to how the number of instances needed to learn grows with the number of dimensions (attributes) the instances have



# The curse of dimensionality

---

- Suppose we divide the space in unitary cubes (hyper-cubes) and think we need to have training data in each cube so as to say we are doing appropriate sampling
- How does the number of hyper-cubes change as the dimension is increased?
  - One line of length 10 is “filled” with 10 hypercubes of dimension 1
  - A square with a side of length 10 is filled with 100 hypercubes of dimension 1
  - A cube with a side of length 10 is filled with 1000 hypercubes of dimension 1
- For every extra dimension the space grows an order of magnitude



# The curse of dimensionality

---

- Another way to look at it, is that when we have a distance metric that defines a hyper-sphere (say a unit radius), an increase in dimension makes points to grow further away from its center
- Essentially what happens is that increasing the dimension of the space decreases the volume of the unit hyper-sphere



# The curse of dimensionality

---

- The formula for the volume of a radius 1 hypersphere :

$$v_n = \left( \frac{2\pi}{n} \right) v_{n-2}, n \geq 3$$

n	Vn
1	2
2	3.141592654
3	4.188790205
4	4.934802201
5	5.263789014
6	5.16771278
7	4.72476597
8	4.058712126
9	3.298508903
10	2.55016404
11	1.884103879
12	1.335262769
13	0.910628755
14	0.599264529



# The curse of dimensionality

x1	x2	x3	x4	x5
0.920265056	0.26452019	0.30489182	0.04169455	0.84777899
0.650880888	0.72257056	0.93792103	0.81626444	0.99629299
0.584418079	0.41394878	0.74564532	0.07076269	0.80383761
0.079327314	0.43830391	0.49845982	0.64060199	0.74885878
0.32663081	0.3137468	0.72152193	0.38348997	0.32952658
0.631366981	0.75508859	0.67049779	0.60748443	0.31974318

D(x1)	D(x1,x2)	D(x1,x2,x3)	D(x1,x2,x3,x4)	D(x1,x2,x3,x4,x5)
0.92026506	0.95752739	1.00489687	1.00576149	1.31540312
0.65088089	0.97249892	1.35109215	1.57852388	1.86663798
0.58441808	0.71616903	1.03386896	1.03628779	1.31150573
0.07932731	0.44542467	0.66847986	0.92587053	1.19080885
0.32663081	0.45290699	0.85189121	0.93422866	0.99064169
0.63136698	0.98426777	1.19094514	1.33693218	1.37463572

Average distance	0.53214819	0.75479913	1.01686237	1.13626742	1.34160552
to origin					



# Dimensionality reduction

---

- The complexity of a model depends in part on then number of attributes of its data
  - And the number of instances need to train it grows with the number of attributes
- Additionally many regression and classification techniques are incapable of identifying irrelevant attributes
- Irrelevant attributes can hurt model perfomance





# Dimensionality reduction

---

- Reducing the number of attributes reduce the complexity of the model
  - Simple models tend to generalize better
- Also, sometimes gathering attributes is costly. If the attribute is irrelevant we can save on the recollection expense



# Dimensionality reduction

---

- There are supervised and unsupervised techniques
  - That require or not the value of the objective function
- Further, methods can be divided in:
  - Attribute selection
    - Select a subset  $k$  out of the  $p$  attributes
  - Feature extraction
    - Find  $k$  features created using the  $p$  attributes
- We will look at a supervised technique for attribute selection and a unsupervised one for feature extraction
  - Subset selection
  - Principal component analysis



# Attribute selection

## Subset selection

---

- The objective is to find the subset of attributes that best helps the performance of the model
  - How many subsets are there?
  - Too many
- Its impractical to explore every possible combination of the  $p$  attributes
- There are two related techniques that find a reasonably good subsets using a simple (and familiar) heuristic
- Since they are supervised we need the value of the objective function for the training data



# Subset selection

## Incremental selection

---

- Let  $A=\{x_1, \dots, x_p\}$  be the set of attributes and ERROR the maximum acceptable error
- $F=\{\}$
- Do{
  - For each attribute  $x_j$  of  $A$ 
    - Compute the classification or regression error  $\varepsilon$  using  $F \cup \{x_j\}$
    - Store the attribute  $x_m$  (and  $\varepsilon$ ) that causes the greatest diminution in error
  - $F \leftarrow F \cup \{x_m\}$
  - $A \leftarrow A - \{x_m\}$
- Until  $(\varepsilon < \text{ERROR})$  or  $(A=\{\})$



# Subset selection

## Decremental selection

---

- Let  $A=\{x_1, \dots, x_p\}$  be the set of attributes and ERROR the maximum acceptable error
- $F \leftarrow A$
- Do{
  - For each attribute  $x_j$  de  $F$ 
    - Compute the classification or regression error  $\varepsilon$  using  $F - \{x_j\}$
    - Discard the attribute  $x_m$  (and  $\varepsilon$ ) the causes the least reduction in the error
  - $F \leftarrow F - \{x_m\}$
- Until  $(\varepsilon < \text{ERROR})$  or  $(A = \{\})$



# Subset selection

---

- It's important (as ALWAYS) to train the model and compute the error with two disjoint sets of data
- Decremental selection is more costly
  - But it may find subsets that reduce the error only when they are together (and that incremental selection wouldn't)
  - E.g. Saldo, deuda y edad. Puede ser que individualmente (o en parejas) ni el saldo de un cliente, ni su deuda, ni su edad ayuden a un clasificador, pero que juntos sí lo hagan
- They work for every supervised learning technique



# Feature extraction

---

- Instead of selecting a subset of attributes this technique constructs new features using the original attributes
  - E.g. Instead of having *balance* and *debt* we create *balance/debt*



# Principal Components Analysis (PCA)

---

- This technique finds relations among the attributes to create new features
- Is unsupervised
  - It does not need the value of the objective function in the training data
- It only finds linear relationships
- It assumes the covariance is a good indicator for useful relationships between attributes





# Principal Components Analysis (PCA)

---

- Suppose we have  $N$  training instances with  $p$  attributes each ( $p$  dimensions)
- The objective is to find a new space with  $k \leq p$  dimensions that highlights the useful attribute relationships for learning



# Principal Components Analysis (PCA)

---

- The general idea is to find linear combinations of the attributes keeping those that maximize the variance in the data
  - The features that explain the most variability in the data are the ones that can be best leveraged for learning
  - These are the features that best characterize the differences between data groups and that can be used to describe similar instances



# Algorithm (idea)

---

- Compute the covariance matrix
- Find the direction in which there is more variability. Repeat with the other, perpendicular, directions
- Keep only some the directions (the ones that capture the most variability)
- Transform the original data using the found features



# Brief review

---

- Covariance
- Eigenvectors and eigenvalues



# Covariance matrix

---

- It indicates the variability of an attribute with respect to another
- $\text{Cov}(A_1, A_2) =$   
$$\left( \sum (A_{1,i} - \text{avg}(A_1)) (A_{2,i} - \text{avg}(A_2)) \right) / (n-1)$$
- This matrix contains all attribute combinations:  $A_1$  with  $A_2$ ,  $A_2$  with  $A_3, \dots, A_p$  with  $A_1, \dots, A_p$  with  $A_p$ .
- This results in a matrix with  $p^2$  entries
- Note that the diagonal contains the variance of each attribute

# PCA

## Eigenvectors

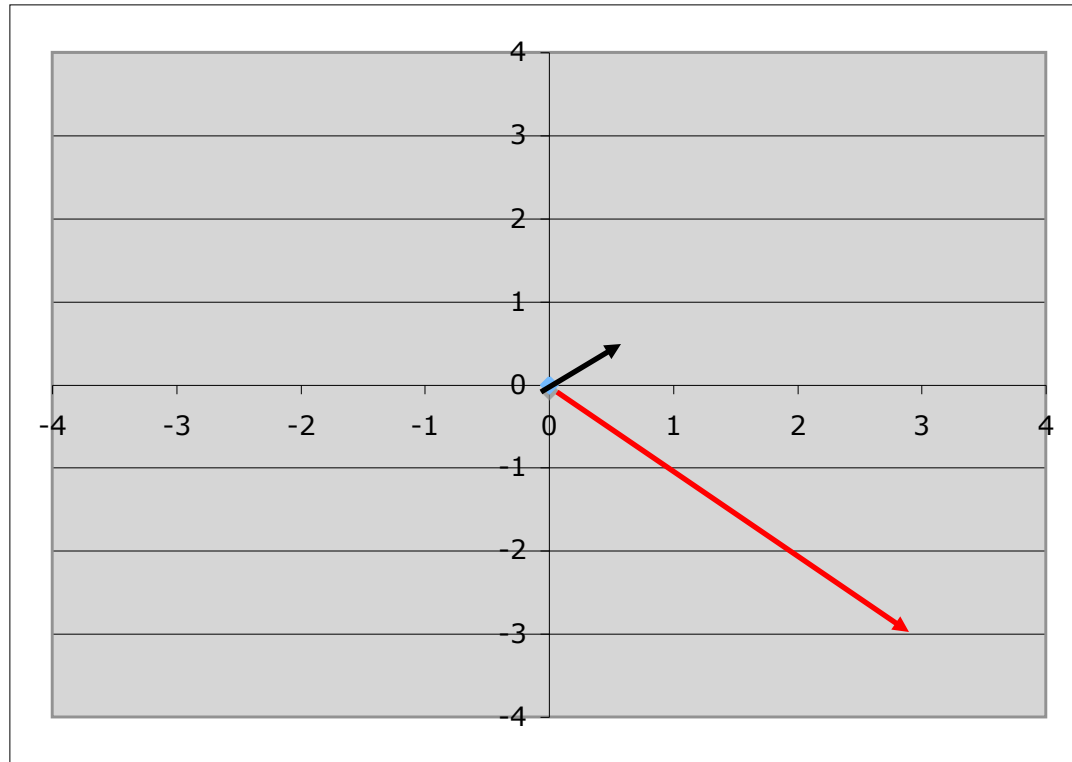
---

- An eigenvector is a vector that does not change direction (only magnitude) under a given transform
- We think of a matrix as a transformation

# PCA

## Example (not an eigenvector)

Transformation  $\begin{bmatrix} 4 & 2 \\ -4 & -2 \end{bmatrix}$  vector  $\begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$



Transforms  
into

$$\begin{bmatrix} 3 \\ -3 \end{bmatrix}$$

# PCA

## Example (an eigenvector)

Transformation  $\begin{bmatrix} 4 & 2 \\ -4 & -2 \end{bmatrix}$  vector  $\begin{bmatrix} 1 \\ -1 \end{bmatrix}$



Transforms  
into

$$\begin{bmatrix} 2 \\ -2 \end{bmatrix}$$





# Eigenvalue

---

- An eigen value is the amount by which an eigenvector changes its magnitude under a given transformation

$$\begin{bmatrix} 2 & 3 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 12 \\ 8 \end{bmatrix} = 4 \times \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 3 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 6 \\ 4 \end{bmatrix} = \begin{bmatrix} 24 \\ 16 \end{bmatrix} = 4 \times \begin{bmatrix} 6 \\ 4 \end{bmatrix}$$

- The eigenvalue is 4



# Properties of eigenvectors and their relation to PCA

---

- They only exist for square matrices
  - The covariance matrix is square
- An  $n \times n$  matrix has at most  $n$  different eigenvectors of unit length
  - Estos usamos para PCA
- Eigenvectors are orthogonal to each other
  - This causes the transformed covariance matrix to have zeros everywhere except in the diagonal
- The magnitude of an eigenvalue indicates the strength (importance) of an attribute relationship
  - You can retrieve the proportion of the variance a feature represents by dividing its eigenvalue over the sum of all eigenvalues



# PCA Algorithm

---

Let  $\mathbf{X}$  be the data matrix with  $N \times p$  dimensions

1. For each attribute  $i$ , compute the mean  $m_i$  and subtract it from each attribute value. Let  $\mathbf{X}'$  be the resulting matrix
2. Compute the covariance matrix  $\Sigma$  of  $\mathbf{X}'$ . This will yield a  $p \times p$  matrix
  - Its the average of the multiplication of each attributes difference from the mean
  - It represents how much each attribute varies wrt another



# PCA Algorithm

---

3. Compute the eigenvectors and eigenvalues of  $\Sigma$ 
  - The eigenvectors represent the directions (axis) that best describe the covariance of the attributes
4. Choose the  $k$  principal components (those eigenvectors with the highest eigenvalues) and create the feature matrix **R**
  - **R** has the components (eigenvectors) in each column. Its a  $p \times k$  matrix
5. Finally apply the transformation to our data **X'**
  - Transformed data =  $\mathbf{R}^T \mathbf{X}'^T$

# Example PCA

(Data from Lindsay I Smith)

## Paso 1

Datos

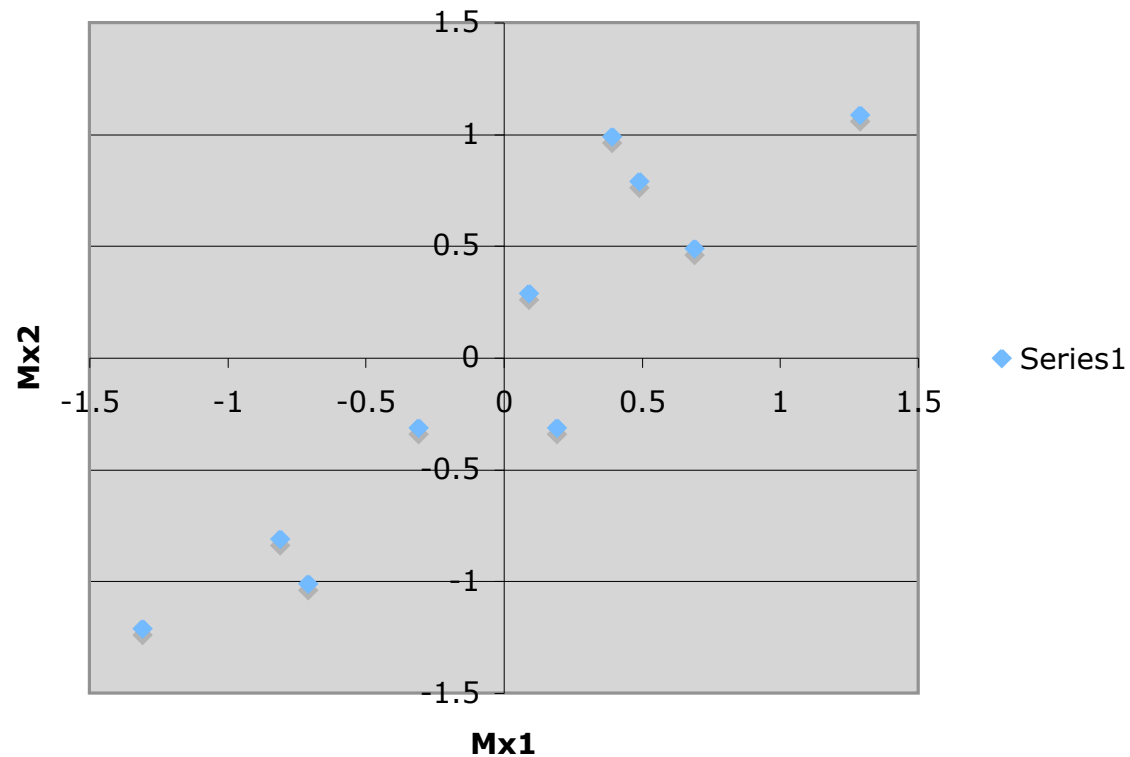
x1	x2	x1-m1	x2-m2
2.5	2.4	0.69	0.49
0.5	0.7	-1.31	-1.21
2.2	2.9	0.39	0.99
1.9	2.2	0.09	0.29
3.1	3	1.29	1.09
2.3	2.7	0.49	0.79
2	1.6	0.19	-0.31
1	1.1	-0.81	-0.81
1.5	1.6	-0.31	-0.31
1.1	0.9	-0.71	-1.01

mi, m2

1.81

1.91

# PCA Example



- Plot of data-average

# PCA example

## Step 2

- Covariance

$E((x_1 - m_1)(x_1 - m_1))$	$E((x_1 - m_1)(x_2 - m_2))$
$E((x_2 - m_2)(x_1 - m_1))$	$E((x_2 - m_2)(x_2 - m_2))$

0.5549	0.5539
0.5539	0.6449

# PCA example

## Step 3

- Eigenvectors and eigenvalues

Vectores característicos

eigen1	eigen2
-0.735179	0.677873
0.677873	0.735179

Valores característicos

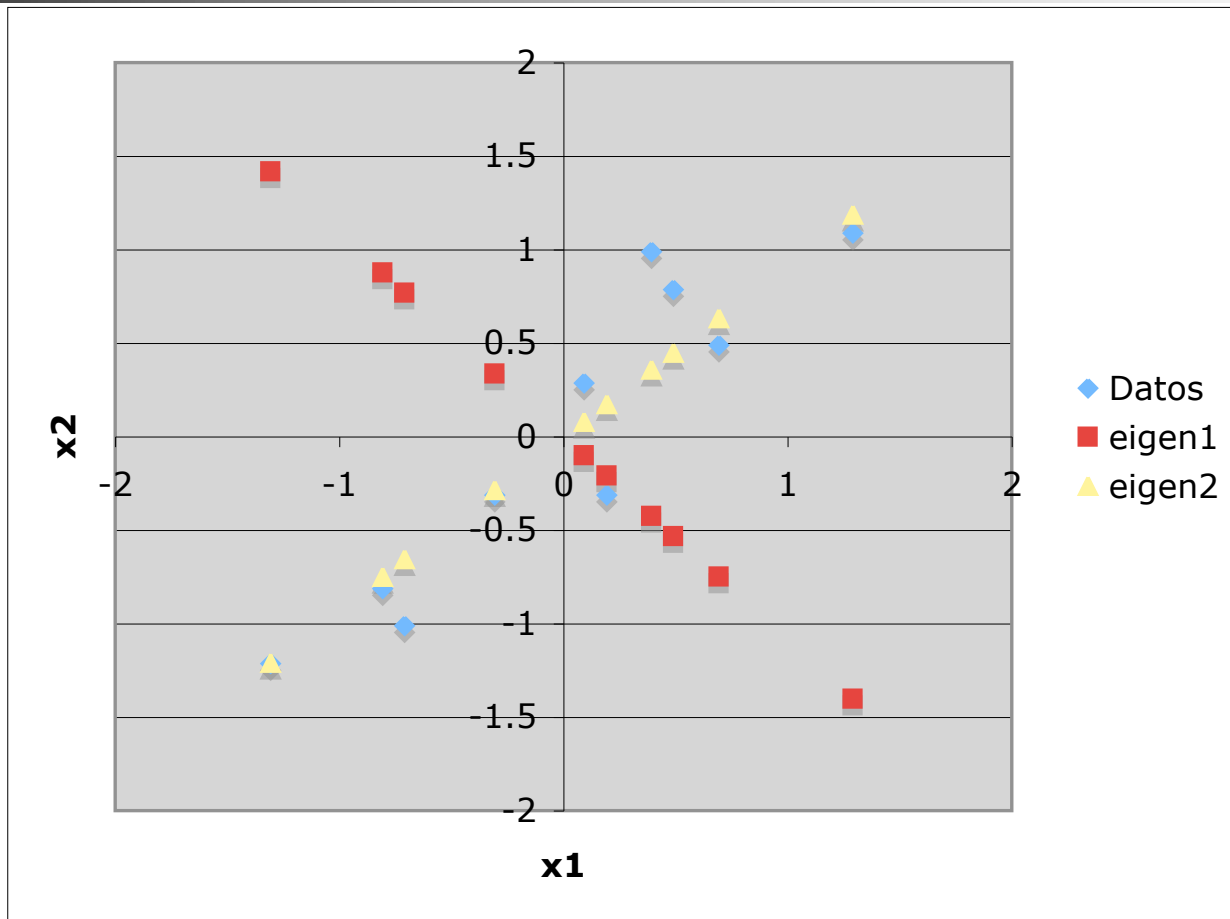
eigen1	0.044175
eigen2	1.155625

The second vector accounts for 72% of the variance

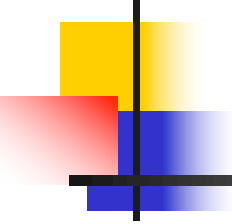
$$1.155/(1.155+0.044)=0.72$$



# PCA example



- Data and eigenvectors



# PCA example

## Steps 4 and 5

---

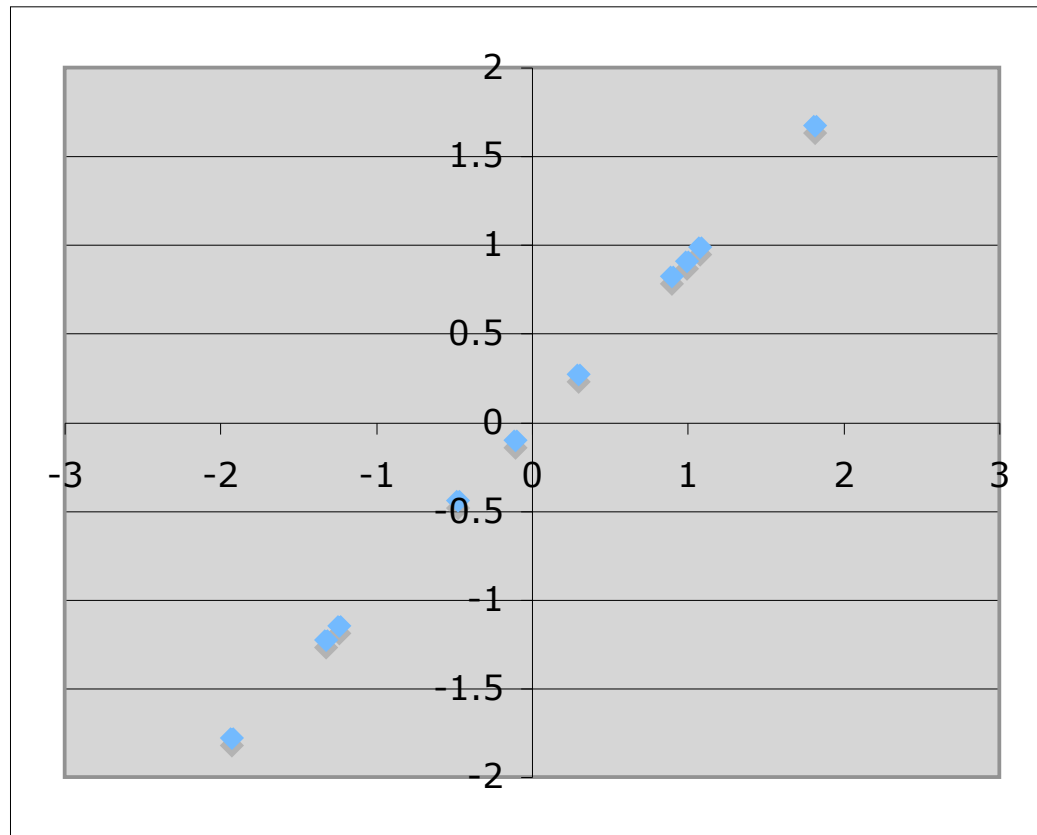
- We chose the vector with largest eigenvalue (the second one) and transform the data

Transformed data

0.82797008
-1.77758022
0.99219768
0.27421048
1.67580128
0.91294918
-0.09910962
-1.14457212
-0.43804612
-1.22382062

Transform  $\rightarrow$   $\text{eigen2}^T \mathbf{X}'^T$

# Plot of the transformed data



Each eigen vector defines the projection of a data point over the new axis  
The axis represent the linear combinations of the original attributes that maximize variation in the data, where it is better separated



# PCA

---

- In general we keep  $k$  eigenvectors and express the data in terms of the features that best describe their variation
- If  $k$  is smaller than  $p$  then we've also reduced the dimensionality of the problem



# Ejercicio

---

- Repita el ejercicio en python