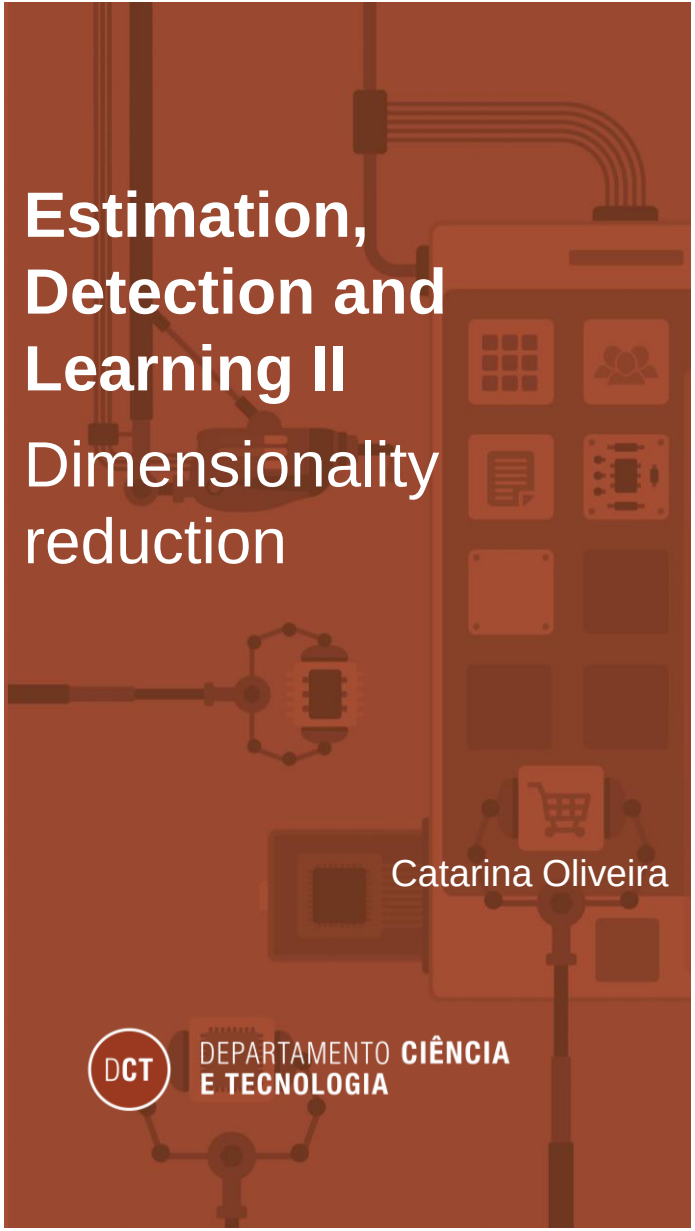


UNIVERSIDADE PORTUCALENSE

CONTENT

1. Main Component Analysis (PCA)
2. Linear Discriminant Analysis (LDA)
3. Self- Organizing Map (SOUND)

Main Component Analysis (PCA)

PCA

Transforms a *dataset* with many variables into a *dataset* with fewer variables with (almost) the same information

Steps:

1. Standardization
2. Computation of the covariance matrix
3. Computation of the *eigenvectors* and *eigenvalues* of the covariance matrix to identify the principal components
4. *feature vector*
5. Reshape the data along principal component axes

PCA: 1 - Standardization

Objective: to standardize the range of continuous initial variables so that each of them contributes equally to the analysis

Reason: The PCA is very sensitive to the variances of the initial variables

Process: for each observation v_i of each variable v with mean μ and standard deviation σ, its value becomes:

$$z = \frac{v_i - \mu}{\sigma}$$

PCA: 2 – Computation of the covariance matrix

Purpose: to understand how the input dataset variables vary from the average to each other, i.e. to see if there is any relationship between them.

Reason: sometimes variables are highly correlated in such a way that they contain redundant information.

Process: for each dataset variable v_i, *calculate* the covariance matrix:

$$\begin{bmatrix} Cov(v_1, v_1) & Cov(v_1, v_2) & \cdots & Cov(v_1, v_n) \\ Cov(v_2, v_1) & Cov(v_2, v_2) & \cdots & Cov(v_2, v_n) \\ \vdots & \vdots & & \vdots \\ Cov(v_n, v_1) & Cov(v_n, v_2) & \dots & Cov(v_n, v_n) \end{bmatrix}$$

Covariance sign:

- Positive: variables are correlated
- Negative: variables are inversely correlated

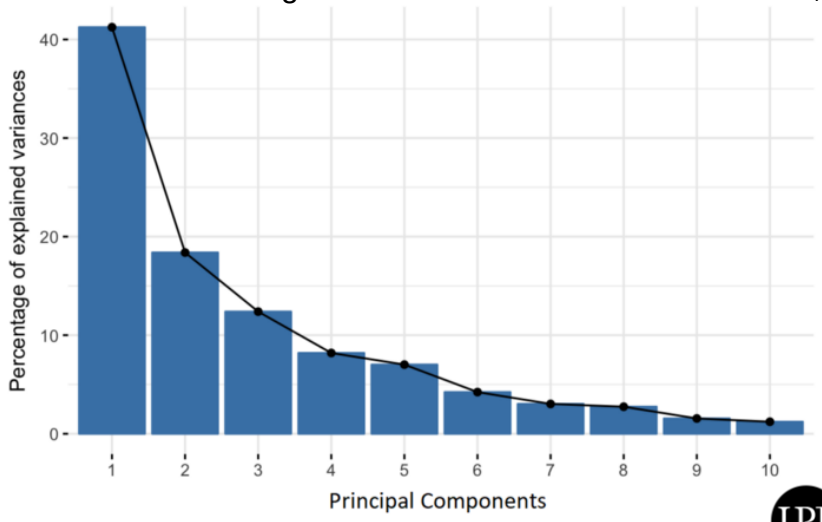
PCA: 3 - Computation of *eigenvectors* and *eigenvalues*

Eigenvectors and *eigenvalues* : linear algebra concepts calculated from the covariance matrix to determine the principal components of the data

Principal components: new variables constructed as linear combinations or mixtures of the initial variables, made in such a way that:

- The new variables (principal components) are not correlated
- Most of the information within the initial variables is compressed into the first components.

Example: a *dataset* with 10 dimensions provides 10 main components, but the PCA tries to put as much information as possible in the first component, the maximum remaining information in the second and so on, until it has something like:



PCA: Main Components

The organization of the main components in this way allows to reduce the dimensionality without losing much information, discarding the components with little information and considering the rest as new variables

Note: principal components are less interpretable and have no real meaning, as they are constructed as linear combinations of initial variables

Geometrically, the principal components represent the direction of the data that explain most of the variance, that is: the lines that capture most of the information in the data.

- The greater the variance of a line, the greater the dispersion of data along that line
    - The greater the dispersion, the more information has

Principal components can be seen as new axes that show the best angle to view and evaluate the data so that differences between observations are more visible.

PCA: construction of the main components

The *eigenvectors* of the covariance matrix are the directions of the axes where the greatest variance is (more information), called principal components.

The *eigenvalues* are coefficients of the *eigenvectors* , and give the amount of information for each principal component

By ordering the *eigenvectors* in order of their *eigenvalues* , from the largest to the smallest, we obtain the principal components in order of significance.

PCA: 4 – Feature vector

After computing the *eigenvectors* and sorting them by descending *eigenvalues* , finding the principal components in order of significance

Determine if we are going to keep all components (n) or discard those with lower significance (smallest *eigenvalues*), forming with the remaining ones (p - not discarded) a matrix of vectors that we call *feature vector*

feature vector : matrix that has as columns the *eigenvectors* that we decided to keep

Dimensionality reduction: we choose to keep only p components of the initial n

PCA: 5 – Reshape data along principal component axes

use the *feature vector* formed using the covariance matrix *eigenvectors to reorient the data from the original axes to those represented by the principal components*

Process: multiply the transpose of the original *dataset by the transpose of the feature vector*

$$DatasetFinal = FeatureVector^T \times DatasetOriginalStandardizado^T$$

Linear Discriminant Analysis (LDA)

LDA

Linear Discriminant Analysis (LDA): prediction algorithm for *multi-class classification*

Used for dimensionality reduction, providing a projection of the *dataset* that better separates (discriminates) examples by classes

Tries to find a linear combination of input variables that allows for maximum sample separation between classes (computing centroids or means) and minimum sample separation within each class

We can use LDA to calculate the projection from the *dataset* and select a number of dimensions or components from the projection to use as input to a model.

LDA Algorithm

Takes on:

- The data follows a Gaussian distribution.
- Each attribute has the same variance: values of each variable vary around the mean, on average the same value

Process:

- Estimates the mean μ_k and variance σ^2 of the data for each class *k*

Forecast:

- Estimates the probability (using Bayes ' theorem) that a new input belongs to each class. The class with the highest probability is predicted.

LDA for dimensionality reduction in python

```
# prepare dataset
date = ...

# define transform
lda = LinearDiscriminantAnalysis ()

# prepare transform on dataset
lda.fit (date)

# apply transform to dataset
transformed = lda.transform (data)
```

Self- Organizing Map (SOUND)

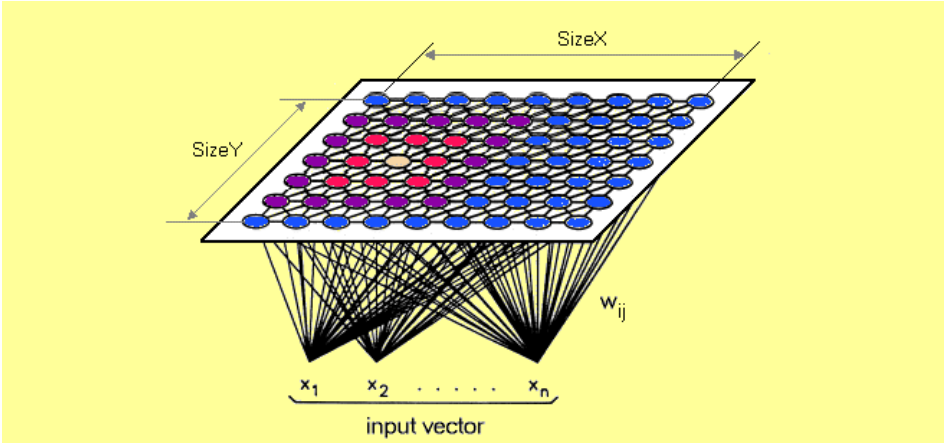
Self- Organizing Map (SOUND)

Type of artificial neural network trained using *unsupervised learning* to produce a representation with fewer dimensions (usually two) based on an *input space*

Different from neural networks because they apply *competitive learning* instead of *error- correction lerarnig* (eg : *backpropagation* with *gradient descent*) and use a neighborhood function to preserve the topological properties of the *input space*

Self- organizing map = Kohonen map

Kohonen , T. Self-organized formation of topologically correct feature maps. Biol. Cybern . 43, 59–69 (1982). https://doi.org/10.1007/BF00337288



SOUND Algorithm

1. The weights of each node are initialized.
2. A vector is randomly chosen from the training dataset.
3. Each node is examined to calculate which weights are most similar to the input vector. The winning node is called *Best Matching unit* (BMU).
4. The neighborhood of the BMU is calculated. The number of neighbors decreases with time.
5. The winning weight is rewarded by becoming more like the chosen vector by 2. Neighbors also become more like that vector. The closer a node is to the BMU, the more its weights are changed and the further the neighbor is from the BMU, the less it learns.
6. Repeat step 2 for N iterations

best Matching Unit : technique that calculates the distance (eg Euclidean) of each weight to the chosen vector. The weight with the shortest distance is the winner.

UNIVERSIDADE
PORTUCALENSE

Do conhecimento à prática.