Review of feature extraction techniques

Florian YGER

Sugiyama lab, Tokyo Institute of Technology - Japan

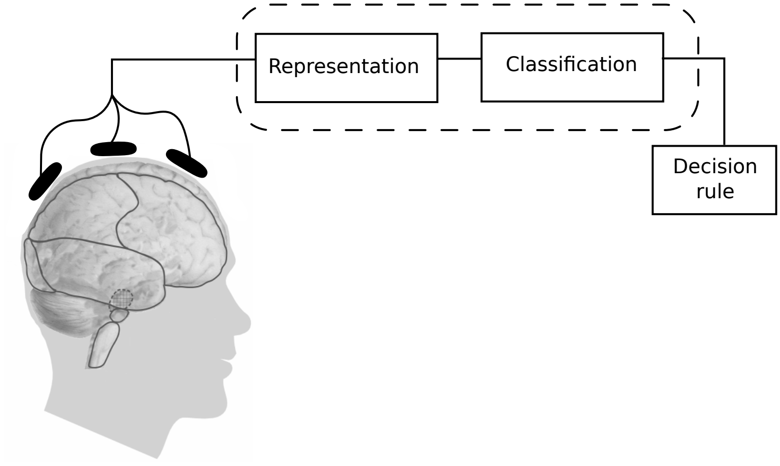
www.yger.fr - florian@sg.cs.titech.ac.jp

May 20, 2014

Introduction

In today's lecture :
Our final goal is classification (supervised learning setting) but we consider the complete data processing chain (from representation to decision).
In this chain, we focus on the first step : representation.

Illustration : application to Brain-Computer-Interfaces



Overview

Content

Summary

- motivations to feature extraction.
- different explanations/formulations for the PCA.
- links among various methods and variants of the litterature.

Ockham's razor

Often quoted in the litterature, it have been named by willian Hamilton (1788-1856) after William of Ockham (1287-1347), it is a principle of parcimony and economy when solving and modeling problems.
Usually quoted as *Pluralitas non est ponenda sine necessitate*, it can be translated as "a plurality is not to be posited without necessity".
It is a heuristic principle stating that among competing hypotheses/solutions, the simplest one should be selected.

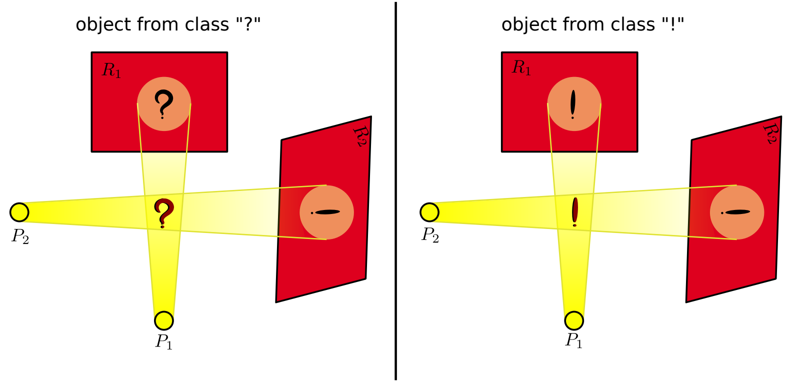
What is feature extraction ?

Dimension reduction is a field of Statistics and Machine learning that focuses on reducing the number of variables (used by a learning algorithm). It assumes that the data contains redundant and/or irrelevant variables.
It is divided into two closely related sub-fields :

- **feature selection** : selecting subsets of useful variables.
- **feature extraction** : transforming the data (combining the variables into more informative ones).

Plan

Answer in one picture



Outline

Main reasons for feature extraction

- reduction of the number of variables (computational issue).
- denoising the data.
- better interpretability.
- data summary and graphical representation.
- enhance the generalisation and reduce overfitting.

Key concepts

- Close relationship between representating and classifying the data.
- With a bad representation of the data, even the best classifier is helpless.

Notations

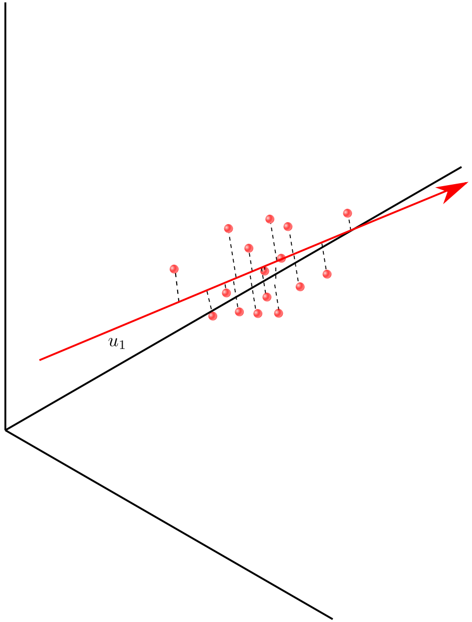
We assume that we have a dataset of n observations of x (vector of size d). X denotes the concatenation of the observations in a matrix of size $d \times n$.
In the remainder, for a given vector v, $||v||_2 = \sqrt{\langle v, v \rangle}$ denotes the euclidean norm, built from the scalar product $\langle v, w \rangle = v^\top w$.

Plan

Introduction

It was first proposed by Karl Pearson in 1901 (*On Lines and Planes of Closest Fit to Systems of Points in Space*) and then independently re-developped by Harold Hotelling in 1933 (*Analysis of a complex of statistical variables into principal components*).
Since then, it has been studied extensively and has known a lot of extensions.

A first illustration ...



... two interpretations...

1. u_1 is direction where the data are the most spreaded (maximize the information).
2. u_1 is the line where the data can be projected with a small error (minimize the residual of the projection).

... leading to two formulae

1. information maximization - maximize the variance of the produced feature :

$$\max_u \frac{1}{n} \sum_i ||x_i^\top u||_2^2$$

2. projection error/residual minimization :

$$\min_u \frac{1}{n} \sum_i ||x_i - \langle x_i, u \rangle u||_2^2$$

To ensure that there is a unique (and non-trivial solution), we add this constraint :

$$||u||_2^2 = 1$$

Question

Are these two formulations equivalent ?

Quick demonstration

$$
\begin{aligned}
& \arg\min_u \frac{1}{n} \sum_i ||x_i - \langle x_i, u \rangle u||_2^2 \\
\Leftrightarrow & \arg\min_u \frac{1}{n} \sum_i (x_i^\top x_i - 2x_i^\top u u^\top x_i + x_i^\top u \underbrace{u^\top u}_{||u||_2^2=1} u^\top x_i) \\
\Leftrightarrow & \arg\min_u \frac{1}{n} - x_i^\top u u^\top x_i \\
\Leftrightarrow & \arg\max_u \frac{1}{n} ||x_i^\top u||_2^2
\end{aligned}
$$

How to solve it ?

Expressing the lagrangian of the first problem :

$$\mathcal{L}(u,\lambda) = \frac{1}{n}||x_i^\top u||_2^2 - \lambda(||u||_2^2 - 1)$$

Setting to zero the derivative w.r.t the primal variable u, we obtain :

$$\begin{aligned}&\frac{\partial}{\partial u}\mathcal{L} = 0\\ \Leftrightarrow &\frac{2}{n}\sum_i x_i x_i^\top u - 2\lambda u = 0\\ \Leftrightarrow &\frac{1}{n}\underbrace{\sum_i x_i x_i^\top}_{XX^\top} u = \lambda u\end{aligned}$$

u is the **leading eigenvector** (associated to the biggest eigenvalue) of the covariance matrix $\frac{1}{n}XX^\top$.

and now what ?

In order to create more than one feature, how can we extract more than one principal direction ?
Should we add additionnal constraints ?

More than one principal component

$$\max_{u_1,u_2} \frac{1}{n}\sum_i ||x_i^\top u_1||_2^2 + \frac{1}{n}\sum_i ||x_i^\top u_2||_2^2$$

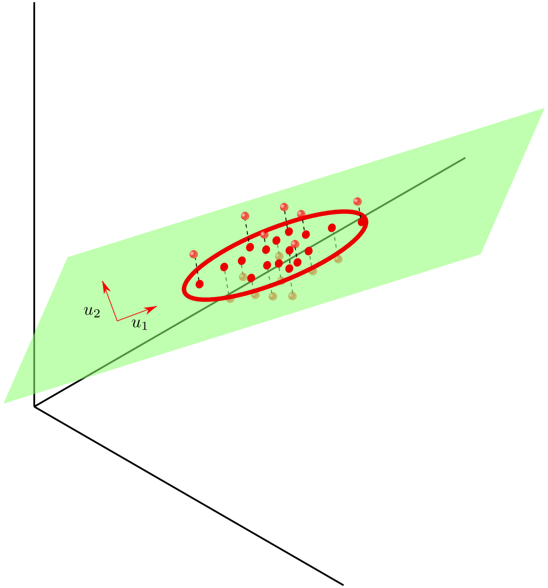
with $||u_1||_2^2 = 1$ and $||u_2||_2^2 = 1$

More than one principal component

$$\max_{u_1,u_2} \frac{1}{n} \sum_i ||x_i^\top u_1||_2^2 + \frac{1}{n} \sum_i ||x_i^\top u_2||_2^2$$

with $||u_1||_2^2 = 1$ and $||u_2||_2^2 = 1$
AND $\langle u_1, u_2 \rangle = 0$.

Illustration

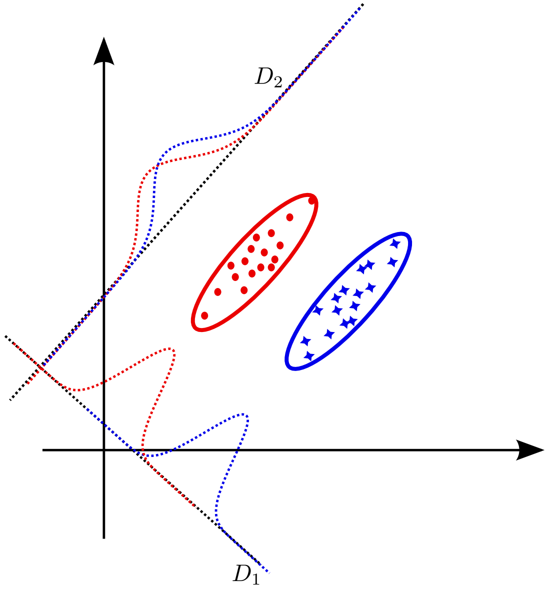


Solution

u_1 is the first leading eigenvector of covariance matrix $\frac{1}{n}XX^\top$. Hence u_2 is the second leading eigenvector.

Consequently, if we need k principal direction for the PCA, we need to extract k eigenvectors.

However PCA may fail ...



... and has some limitations

1. only capable of handling vector data.
2. linear transformation of the data.
3. interpretation of the feature is sometimes difficult.
4. guarantees for the Gaussian case.

... and has some limitations

1. only capable of handling vector data.
2. linear transformation of the data.
3. interpretation of the feature is sometimes difficult.
4. guarantees for the Gaussian case.

but there are alternatives :

1. kernel PCA.
2. kernel PCA and auto-encoders.
3. sparse PCA (not covered today).
4. ICA (cf. Special Lecture by Dr. Sasaki)

Other non covered issues

1. what happens when the data change ?
2. how to handle data having two views/representations ?
3. can we add other constraints than sparsity ?

and some keywords for finding solutions in the litterature :

1. online, stochastic and incremental PCA / subspace tracking.
2. Canonical Correlation Analysis (CCA) and its own variants.
3. dictionary learning and Non-Negative Matrix Factorization (NMF).

Plan

Motivation I

Motivation II

Introduction

Proposed in 1998 by Scholkopf, Smola and Muller (*Nonlinear Component Analysis as a Kernel Eigenvalue Problem*), it is on of the numerous applications of the *kernel trick* to a linear algorithm.
Since then, it has been applied for novelty detection, manifold learning and image de-noising.

Kernel trick

A linear algorithm can be transformed in a non-linear algorithm. To do so, we can transform the input data x by a non-linear function $\phi(.)$ and then apply our linear algorithm on $\phi(x)$.

However, we need to compute $\phi(x)$, which can be computationnaly impossible.

Alternatively, if the linear algorithm is expressed in term of scalar product of the input variables $\langle x_i, x_j \rangle$, then, we can replace the scalar product by a positive-definite kernel function $k(x_i, x_j)$ (under some conditions).

For a given decision function, it means :

$$f(x) = \langle w, x \rangle + b$$

Kernel trick

A linear algorithm can be transformed in a non-linear algorithm. To do so, we can transform the input data x by a non-linear function $\phi(.)$ and then apply our linear algorithm on $\phi(x)$.

However, we need to compute $\phi(x)$, which can be computationnaly impossible.

Alternatively, if the linear algorithm is expressed in term of scalar product of the input variables $\langle x_i, x_j \rangle$, then, we can replace the scalar product by a positive-definite kernel function $k(x_i, x_j)$ (under some conditions).

For a given decision function, it means :

$$f(x) = \langle w, \phi(x) \rangle_{\mathcal{F}} + b$$

Positive-definite kernel

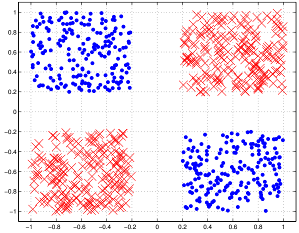
It is a non-linear similarity measure generalizing the concept of scalar product.
Its positive-definiteness is a key ingredient[1] and can be defined in terms of its Gram matrix $K_{ij} = k(x_i, x_j)$: the eigenvalue of K must be **non-negative**.

Hence, even if we cannot write it explicitely, there exists a mapping ϕ (eventually in an infinite dimensionnal functionnal space) such that :

$$k(x_i, x_j) = \langle \phi(x_i), \phi(x_i) \rangle_{\mathcal{H}}$$

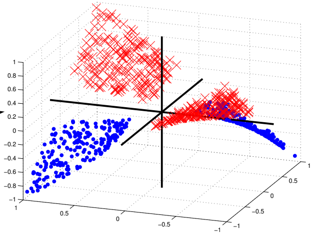
[1]For more about this topic, see the excellent tutorial : 'Positive definite kernels in machine learning' by Marco Cuturi.

Illustration



$\forall i \ X_i = [a_i \ b_i] \in \mathcal{X}$

$\phi(X) = [a \ b \ ab]$



$\forall i \ \phi(X_i) \in \mathcal{F}$

Examples and properties

Classical exmples

- linear kernel : $k(x, y) = \langle x, y \rangle$
- gaussian kernel : $k(x, y) = \exp(\frac{||x-y||^2}{2\sigma^2})$ (with $\sigma > 0$)
- polynomial kernel : $k(x, y) = \langle x, y \rangle^d$ (with the integer $d > 0$)

Operations producing kernels

- scalar multiplication : αk with $\alpha > 0$
- positive sum : $k_1 + k_2$
- elementwise product : $k_1 \times k_2$
- ...

Implicit vs explicit formula

Let us consider the following mapping $\phi(.)$ from $\mathbb{R}^2$ to $\mathbb{R}^3$:

$$(x_1, x_2) \mapsto (z_1, z_2, z_3) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

then :

$$\begin{aligned}
\langle \phi(x), \phi(y) \rangle &= \langle (x_1^2, \sqrt{2}x_1x_2, x_2^2), (y_1^2, \sqrt{2}y_1y_2, y_2^2) \rangle \\
&= x_1^2y_1^2 + 2x_1x_2y_1y_2 + x_2^2y_2^2 \\
&= (x_1y_1)^2 + 2(x_1y_1)(x_2y_2) + (x_2y_2)^2 \\
\langle \phi(x), \phi(y) \rangle &= \langle x, y \rangle^2
\end{aligned}$$

In this example, the scalar product can be computed more efficiently in its implicit form (the 2nd polymial kernel). Note that sometimes (for infinite dimensionnal mappings), the explicit formulat is not available (ex: Gaussian kernel).

Benefits

- using the kernel trick enables to make an algorithm non-linear.
- solving the problem with a kernel is equivalent to solving the linear algorithm in the higher dimensionnal feature space.
- no need to compute explicitely the mapping ϕ.
- applied to numerous algorithm (SVM, LDA, perceptrons ... and off course PCA)

but

- the algorithm's complexity grows as the square of the variable (since we now handle the matrix K).

Back to the PCA I

Using the eigenproblem :

$$\frac{1}{n}\sum_i x_i x_i^\top u = \lambda u$$

Note that an eigenvector u can be expressed w.r.t the data x : $u = \sum_i \alpha_i x_i$
Then, the problem can be rewritten as :

$$\frac{1}{n}\sum_i x_i (x_i^\top u) = \lambda u$$

$$\Leftrightarrow \frac{1}{n}\sum_i x_i (x_i^\top \sum_j \alpha_j x_j) = \lambda \sum_i \alpha_i x_i$$

$$\Leftrightarrow \frac{1}{n}\sum_i \sum_j \alpha_j x_i (x_i^\top x_j) = \lambda \sum_i \alpha_i x_i$$

Back to the PCA II

From this equation :

$$\frac{1}{n}\sum_i \sum_j \alpha_j x_i (x_i^\top x_j) = \lambda \sum_i \alpha_i x_i$$

we can left-multiply by any $x_k^\top$:

$$\frac{1}{n}\sum_i \sum_j \alpha_j x_k^\top x_i (x_i^\top x_j) = \lambda \sum_i \alpha_i x_k^\top x_i$$

Hence, by multiplying by every $x_k^\top$, we obtain :

$$KK\alpha = \lambda K\alpha$$

with K the Gram matrix of the linear kernel (i.e. $K_{i,j} = \langle x_i, x_j \rangle$)
Finally, we obtain the following eigenproblem :

$$K\alpha = \lambda\alpha$$

Kernelization in practice

Now, in order to obtain a k-PCA, we only need to replace K by the Gram matrix of the choosen kernel and then solve the eigenproblem.

For the PCA, for a new given points x_m, the representation along the first eigenvector u will be :

$$u^\top x_m = \sum_i \alpha_i {x_i}^\top x_m$$

For the kPCA, it will simply be :

$$z_m = \sum_i \alpha_i k(x_i, x_m)$$

Plan

Introduction

Proposed in 1988 by Bourlard and Kamp (*Auto-association by multilayer perceptrons and singular value decomposition*), it is one the numerous non-linear extension of the PCA for neural networks.
Thanks to the revival of neural networks (through the so-called *Deep Learning* trend), it has been applied to the pre-training of deep neural networks. Since then, several variants have been proposed (notably by the team of Yoshua Bengio in Canada).

Back to the PCA... again

When several principal directions are needed at once, we can reformulate the residual formulation as :

$$\min_U \frac{1}{n} \sum_i ||x_i - \underbrace{U \overbrace{U^\top x_i}^{\text{encoding}}}_{\text{decoding}} ||_2^2$$

$$\text{s.t. } U^\top U = \mathcal{I}_k$$

Using two non-linear functions g and h, an auto-encoder is obtained by solving the (non-convex) problem :

$$\min_U \frac{1}{n} \sum_i ||x_i - \underbrace{g(U \overbrace{h(U^\top x_i + b_h)}^{\text{encoding}} + b_g)}_{\text{decoding}} ||_2^2$$

Insights

The idea is to encode and decode the data using g and h and then to find their parameter U that minimize the error.

Usually, this problem is solved using a stochastic gradient descent.
Recently, Autoencoders have been used for pre-training (initialize) deep neural networkds. Autoencoders are stacked and the layers are learned incrementally in order to pre-train every layer of the deep network.

Short comment on ICA

Independant Component Analysis can also be seen as a variant of PCA. In this variant, instead of using the variance minimization problem :

$$\max_u \frac{1}{n} \sum_i ||x_i^\top u||_2^2$$
$$\text{with } ||u||_2^2 = 1$$

we use another statistical moment :

$$\max_u \frac{1}{n} \sum_i m(x_i^\top u)$$
$$\text{with } ||u||_2^2 = 1$$

using m a non-linear function.

Note that, since some architecture come from non-linear PCA, several connections have been spotted between ICA and neural networks (such as Autoencoders).

Further references on those topics

Feat.	"The elements of statistical learning" 10th edition - chap 14 - Hastie, Tibshirani and Friedman
PCA	"Principal Component Analysis", I.T. Joliffe (2005)
kPCA	"Advances in Kernel Methods - Support Vector Learning" - chap 20 - Burges, Scholkopf and Smola (1998)
AA-DL	"Representation Learning: A Review and New Perspectives" Bengio, Courville and Vincent - arXiv (v2-2014)
Sparse	"Sparse principal component analysis" -JCGS - Zou, Hastie and Tibshirani (2006)

Take home message

- data representation/feature extraction is an important task for classification.
- lots of feature extraction techniques can be linked to one of the various PCA formulations.

Homework due for today

Please hand me the printed version of your homework before leaving the class.

Homework for next lecture

Write you opinion about the special lecture today.

Directly submit the printed report to the lecturer next week.

Thank you for your attention.