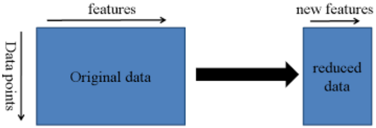
Feature extraction techniques

Vanessa Gómez Verdejo

Machine Learning 4 Data Science Group
Universidad Carlos III de Madrid

Summary

Feature extraction



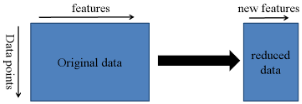
Reduce the dimensionality of the input space

- Use only relevant data, i.e, remove irrelevant/noisy/correlated components minimizing the loss of RELEVANT information
- Discover good combinations of input variables (features)

Simplify the ML stage

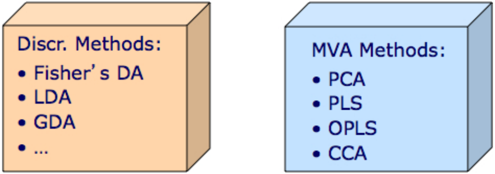
- Minimize the number of parameters in the classifier (curse of dimensionality)
- *Bend* the input space to better fit our task
- Compact representation of data (crucial for large datasets)

Some notation



- Input data: $\mathbf{x} = [x_1, \dots x_N]$ (N input dimensions)
- Output data (labels) $\mathbf{y} = [y_1, \dots y_M]^T$ (M target variables)
- Transformed input data: $\mathbf{x}' = \left[x'_1, \dots x'_{n_p}\right]$ ($n_p < N$ new input dimensions)
- Matrix notation for training data: $\mathbf{X}$ ($L \times N$), $\mathbf{X}'$ ($L \times n_p$), $\mathbf{Y}$ ($L \times M$)
- Some useful matrix: $\mathbf{C}_{\mathbf{XX}} = \mathbf{X}^T\mathbf{X}$ and $\mathbf{C}_{\mathbf{XY}} = \mathbf{X}^T\mathbf{Y}$
- Transformation matrix ($N \times n_p$): $\mathbf{U} = \begin{bmatrix} u_{1,1} \dots u_{1,n_p} \\ \vdots \ddots \vdots \\ u_{N,1} \dots u_{N,n_p} \end{bmatrix}$
- Data transformation is given by: $\mathbf{x}' = \mathbf{U}^T\mathbf{x}^T$ ($\mathbf{X}' = \mathbf{U}^T\mathbf{X}^T$)

Classification of FE techniques



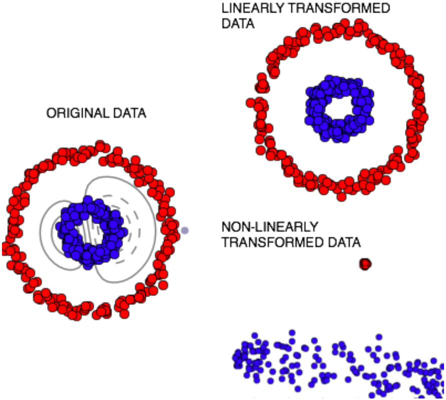
Discriminative methods vs. MVA

- Both families provide linear projections for FE
- MVA algorithms require classical linear algebra methods (EIG, Generalized EIG, SVD)
- Certain equivalences are known under a classification context.

Classification of FE techniques

Linear vs. non-linear projections

- We can generate new features by linear combinations of the original ones
- or non-linear combinations can be applied...



LDA as Feature extractor

- LDA considers that the data follow a gaussian distribution with the same covariance matrix.

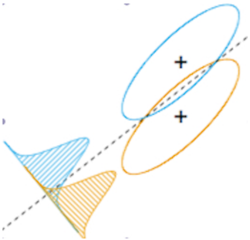
$$p(\mathbf{x}|y=-1) \sim G(\mathbf{m}_0, V) \qquad p(\mathbf{x}|y=1) \sim G(\mathbf{m}_1, V)$$

- Then, the optimum classifier is

$$\hat{y} = \text{sign}\left(\mathbf{w}^T\mathbf{x}\right)$$

where $\mathbf{w} = V^{-1}(\mathbf{m}_1 - \mathbf{m}_0)$.

- To decide, we project the input data over $\mathbf{w}$ and apply a threshold.
- These projections provide a new data representation -¿ **new features**.
- In multiclass problems there are as many linear discrimination functions as number of classes minus one.

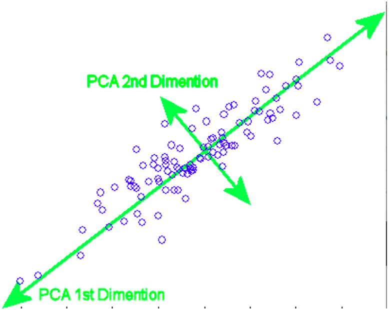


Principal Component Analysis (PCA)

Goal

Find projections maximizing the variance of the projected data

- $\mathbf{u}_1^T \mathbf{x}^T$ projects the maximum variance of the data
- $\mathbf{u}_2^T \mathbf{x}^T$ the second one, ...
- Computing $n_p < N$ new features, we remove the directions with less variance



Principal Component Analysis (PCA)

Mathematical formulation

- Find projections maximizing the variance of the projected data

$$\mathbf{U} = \underset{\mathbf{U}}{\operatorname{argmax}}\ \operatorname{Tr}\left\{\mathbf{U}^T\mathbf{X}^T\mathbf{X}\mathbf{U}\right\} = \underset{\mathbf{U}}{\operatorname{argmax}}\ \operatorname{Tr}\left\{\mathbf{U}^T\mathbf{C_{XX}}\mathbf{U}\right\}$$

$$\text{s.t.}\quad \mathbf{U}^T\mathbf{U} = \mathbf{I}$$

- Which leads to the eigenvalue problem

$$\mathbf{C_{XX}}\mathbf{u} = \lambda\mathbf{u}$$

- Thus, U consists of the first eigenvectors of C_{xx} (i.e., those associated with largest eigenvalues)

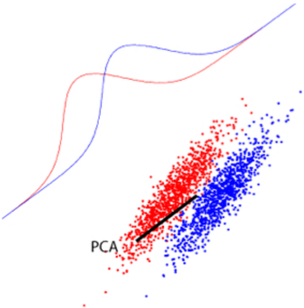
$$\mathbf{U} = \operatorname{eigs}(\mathbf{C_{XX}}) \qquad \mathbf{U} = \operatorname{svd}(\mathbf{X})$$

Principal Component Analysis (PCA)

Somme comments

- It is an **unsupervised** algorithm!!!!

- Which direction will PCA consider as the most relevant one?
- If we had to extract only one projection, which is the most relevant for the task?



- Clearly, when dealing with supervised problems, we should consider the labels to obtain good features $\longrightarrow$ PLS and CCA algorithms

Partial Least Squares (PLS)

Goal

Find the projections of the input and output data with maximum covariance

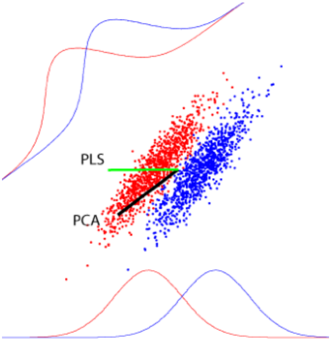
- Mathematical formulation

$$\begin{aligned}\mathbf{U},\mathbf{V} = & \underset{\mathbf{U},\mathbf{V}}{\text{argmax}}\ \text{Tr}\left\{\mathbf{U}^T\mathbf{X}^T\mathbf{Y}\mathbf{V}\right\}\\ = & \underset{\mathbf{U},\mathbf{V}}{\text{argmax}}\ \text{Tr}\left\{\mathbf{U}^T\mathbf{C}_{\mathbf{XY}}\mathbf{V}\right\}\\ & \text{s.t.}\ \ \mathbf{U}^T\mathbf{U} = \mathbf{V}^T\mathbf{V} = \mathbf{I}\end{aligned}$$

- Which leads to

$$\mathbf{U},\mathbf{V} = \text{svd}(\mathbf{C}_{\mathbf{XY}})$$

- The maximum number of projections is limited by the number of output classes



Canonical Correlation Analysis (CCA)

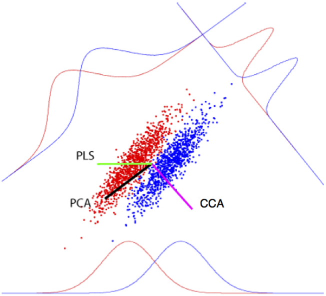
Goal

Find the directions of maximum correlation between input and output data

- Mathematical formulation

$$\mathbf{u}, \mathbf{v} = \underset{\mathbf{u},\mathbf{v}}{\operatorname{argmax}} \frac{\left(\mathbf{u}^T \mathbf{C}_{\mathbf{XY}} \mathbf{v}\right)^2}{\mathbf{u}^T \mathbf{C}_{\mathbf{XX}} \mathbf{u} \mathbf{v}^T \mathbf{C}_{\mathbf{YY}} \mathbf{v}}$$

$$\begin{aligned} \mathbf{U}, \mathbf{V} = & \underset{\mathbf{U},\mathbf{V}}{\operatorname{argmax}} \operatorname{Tr}\left\{\mathbf{U}^T \mathbf{C}_{\mathbf{XY}} \mathbf{V}\right\} \\ \text{s.t.} \quad & \mathbf{U}^T \mathbf{C}_{\mathbf{XX}} \mathbf{U} = \mathbf{V}^T \mathbf{C}_{\mathbf{YY}} \mathbf{V} = \mathbf{I} \end{aligned}$$



Canonical Correlation Analysis (CCA)

Somme comments

- This problem can be solved as a generalized eigenvalue problem
- As many extracted features as output classes
- It is usually applied to obtain a common space to work with input and output features
- For classification purposes:
  - It tends to outperform PLS approaches
  - It's equivalent to LDA as feature extractor

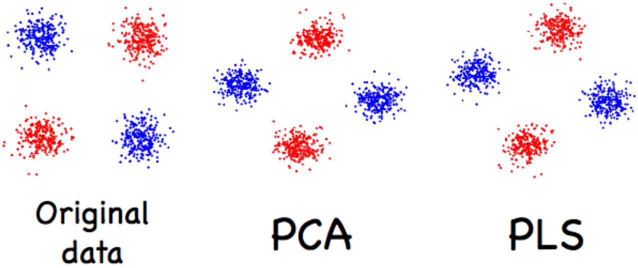
Linear methods

Advantages

- Simplicity
- Easy to understand
- Robust
- Lead to convex problems

Disadvantages

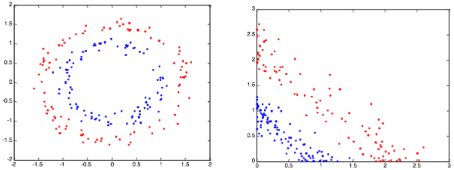
- Lack expressive power



Kernel methods

Idea

- Project Data into a High Dimensional Space



- ...so that a linear algorithm run in the "Feature Space" is non-linear in the original input space.

Kernel examples:

- polynomial, gaussian, ...

Working with kernels

Kernel trick

- It is possible to compute inner products in many ∞-dimensional space:

$$k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$$

- If the linear algorithm can be reformulated in terms of inner products, we can replace them by kernel functions.

Representer theorem

- states that the solutions of certain optimization problems can be written as an expansion in terms of training samples

$$\mathbf{u} = \sum_{l=1}^{L} a^{(l)} \phi(\mathbf{x}^{(l)}) = \Phi^T \mathbf{a}$$

- where the vector $\mathbf{a} = \left[a^{(1)}, \ldots, a^{(L)}\right]^T$ contains the dual variables which are indicating the weight that takes each data to represent the solution.

Kernel Principal Component Analysis (KPCA)

Extending the formulation to the feature space

Find projections maximizing the variance of the data in the *feature space*

- Project the data to the feature space ($X \longrightarrow \Phi$)

$$\mathbf{U} = \underset{\mathbf{U}}{\operatorname{argmax}} \ \operatorname{Tr}\left\{\mathbf{U}^T \Phi^T \Phi \mathbf{U}\right\} \qquad \text{s.t.} \ \ \mathbf{U}^T\mathbf{U} = \mathbf{I}$$

- Apply the Representer Theorem ($\mathbf{U} = \Phi^T \mathbf{A}$)

$$\mathbf{A} = \underset{\mathbf{A}}{\operatorname{argmax}} \ \operatorname{Tr}\left\{\mathbf{A}^T \Phi\Phi^T \Phi\Phi^T \mathbf{A}\right\} \qquad \text{s.t.} \ \ \mathbf{A}^T \Phi\Phi^T \mathbf{A} = \mathbf{I}$$

- Replacing $\Phi\Phi^T$ by the kernel matrix

$$\mathbf{A} = \underset{\mathbf{A}}{\operatorname{argmax}} \ \operatorname{Tr}\left\{\mathbf{A}^T \mathbf{K}\mathbf{K}\mathbf{A}\right\} \qquad \text{s.t.} \ \ \mathbf{A}^T \mathbf{K}\mathbf{A} = \mathbf{I}$$

- Which leads to the eigenvalue problem: $\mathbf{K}\mathbf{a} = \lambda\mathbf{a}$
- Thus, $\mathbf{A}$ consists of the first eigenvectors of $\mathbf{K}$ (i.e., those associated with largest eigenvalues)

$$\mathbf{A} = \operatorname{eigs}(\mathbf{K})$$

Kernel Partial Least Square (KPLS)

- Linear formulation

$$\mathbf{U}, \mathbf{V} = \underset{\mathbf{U},\mathbf{V}}{\text{argmax}} \ \text{Tr}\left\{\mathbf{U}^T \mathbf{X}^T \mathbf{Y} \mathbf{V}\right\} \qquad \text{s.t.} \ \ \mathbf{U}^T \mathbf{U} = \mathbf{V}^T \mathbf{V} = \mathbf{I}$$

- Kernel formulation

$$\mathbf{A}, \mathbf{V} = \underset{\mathbf{A},\mathbf{V}}{\text{argmax}} \ \text{Tr}\left\{\mathbf{A}^T \mathbf{K} \mathbf{Y} \mathbf{V}\right\} \qquad \text{s.t.} \ \ \mathbf{A}^T \mathbf{K} \mathbf{A} = \mathbf{V}^T \mathbf{V} = \mathbf{I}$$

- which solution is given by

$$\mathbf{A}, \mathbf{V} = \text{svd}\left(\mathbf{K}\mathbf{Y}\right)$$

Kernel Canonical Correlation Analysis (KCCA)

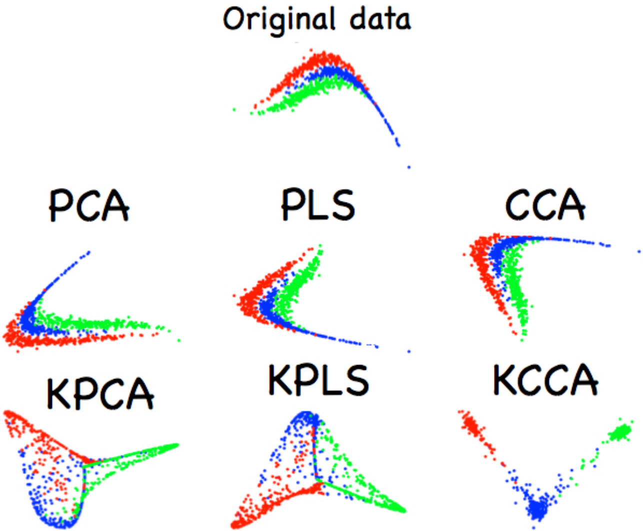
- Linear formulation

$$\begin{aligned} \mathbf{U}, \mathbf{V} = \quad & \underset{\mathbf{U},\mathbf{V}}{\operatorname{argmax}} \operatorname{Tr}\left\{\mathbf{U}^T\mathbf{X}^T\mathbf{Y}\mathbf{V}\right\} \\ & \text{s.t.} \quad \mathbf{U}^T\mathbf{X}^T\mathbf{X}\mathbf{U} = \mathbf{V}^T\mathbf{Y}^T\mathbf{Y}\mathbf{V} = \mathbf{I} \end{aligned}$$

- Kernel formulation

$$\begin{aligned} \mathbf{A}, \mathbf{V} = \quad & \underset{\mathbf{A},\mathbf{V}}{\operatorname{argmax}} \operatorname{Tr}\left\{\mathbf{A}^T\mathbf{K}\mathbf{Y}\mathbf{V}\right\} \\ & \text{s.t.} \quad \mathbf{A}^T\mathbf{K}\mathbf{K}\mathbf{A} = \mathbf{V}^T\mathbf{Y}^T\mathbf{Y}\mathbf{V} = \mathbf{I} \end{aligned}$$

KMVA: example



Some remarks

- The kernel matrix has to be centered (remove the mean in the feature space)
- KPCA, KPLS and KCCA can be computed as its linear counterparts (same functions), but taking into account:

	Linear	Kernel
Input data	$\mathbf{X}$	$\mathbf{K}$
Variables to compute	Eigenvectors ($\mathbf{U}$)	Dual variables ($\mathbf{A}$)
Projection vectors	$\mathbf{U}$	$\mathbf{U} = \Phi^T\mathbf{A}$ (no computed)
Projected data	$\mathbf{X}' = \mathbf{U}^T\mathbf{X}^T$	$\mathbf{X}' = \mathbf{A}^T\Phi\Phi^T = \mathbf{A}^T\mathbf{K}$

- KMVA overcomes the lack of expressiveness of the linear versions, but have serious **scalability** limitations and **overfitting** problems can emerge $\longrightarrow$ compact solutions

Compact solutions

- Reduce de number of possible *support data*

$$\mathbf{U} = \Phi_R^T \mathbf{A}$$

  where Φ_R is a subset of the training data with $R < L$ points
- We obtain a reduced kernel matrix

$$\mathbf{K}_R = \Phi_R \Phi^T \quad (R \times L)$$

- Are we subsampling the data??
  - $\mathbf{K}_R$ still contains information about all samples!!