Lecture 04: Nonlinear feature extraction with kernels

Gustavo Camps-Valls

Image Processing Laboratory (IPL) – Universitat de Valncia. Spain.
gustavo.camps@uv.es, http://www.uv.es/gcamps

VNIVERSITAT DE VALÈNCIA

The organization of the course:

1. Fundamentals of kernel methods
2. Supervised and unsupervised kernel-based classification
3. Kernel methods for regression and time series analysis
4. Nonlinear feature extraction with kernels $<<<$

Motivation

- Feature selection/extraction is essential before classification or regression
- High number of correlated features leads to:
  - Collinearity
  - Overfitting
  - Hughes phenomenon
- Linear methods offer Interpretability $\sim$ knowledge discovery.
- Linear algorithms are commonly used: PCA, PLS, CCA, ...
- Linear algorithms fail when data distributions are curved (nonlinear feature relations)

Motivation

- Feature selection/extraction is essential before classification or regression
- High number of correlated features leads to:
  - Collinearity
  - Overfitting
  - Hughes phenomenon
- Linear methods offer Interpretability $\sim$ knowledge discovery.
- Linear algorithms are commonly used: PCA, PLS, CCA, ...
- Linear algorithms fail when data distributions are curved (nonlinear feature relations)

Outline

- PCA is widely used

Motivation

- Feature selection/extraction is essential before classification or regression
- High number of correlated features leads to:
  - Collinearity
  - Overfitting
  - Hughes phenomenon
- Linear methods offer Interpretability $\sim$ knowledge discovery.
- Linear algorithms are commonly used: PCA, PLS, CCA, ...
- Linear algorithms fail when data distributions are curved (nonlinear feature relations)

Outline

- PCA is widely used
- PCA is not optimal in supervised problems: PLS is very good here

Motivation

- Feature selection/extraction is essential before classification or regression
- High number of correlated features leads to:
  - Collinearity
  - Overfitting
  - Hughes phenomenon
- Linear methods offer Interpretability $\sim$ knowledge discovery.
- Linear algorithms are commonly used: PCA, PLS, CCA, ...
- Linear algorithms fail when data distributions are curved (nonlinear feature relations)

Outline

- PCA is widely used
- PCA is not optimal in supervised problems: PLS is very good here
- PLS is *suboptimal* in the mean-square-error sense

Motivation

- Feature selection/extraction is essential before classification or regression
- High number of correlated features leads to:
  - Collinearity
  - Overfitting
  - Hughes phenomenon
- Linear methods offer Interpretability $\sim$ knowledge discovery.
- Linear algorithms are commonly used: PCA, PLS, CCA, ...
- Linear algorithms fail when data distributions are curved (nonlinear feature relations)

Outline

- PCA is widely used
- PCA is not optimal in supervised problems: PLS is very good here
- PLS is *suboptimal* in the mean-square-error sense
- Orthonormalized PLS (OPLS) is optimal in MSE sense (Roweis†, 1999)

Motivation

- Feature selection/extraction is essential before classification or regression
- High number of correlated features leads to:
  - Collinearity
  - Overfitting
  - Hughes phenomenon
- Linear methods offer Interpretability $\sim$ knowledge discovery.
- Linear algorithms are commonly used: PCA, PLS, CCA, ...
- Linear algorithms fail when data distributions are curved (nonlinear feature relations)

Outline

- PCA is widely used
- PCA is not optimal in supervised problems: PLS is very good here
- PLS is *suboptimal* in the mean-square-error sense
- Orthonormalized PLS (OPLS) is optimal in MSE sense (Roweis[†], 1999)
- Unfortunately, *real* problems are commonly non-linear $\rightarrow$ *Kernels methods*

Notation preliminaries

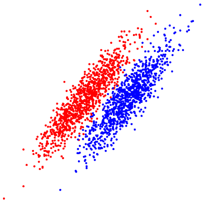
Notation

Data	$\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{l}$, $\mathbf{x}_i \in \mathbb{R}^N$, $\mathbf{y}_i \in \mathbb{R}^M$.
Input Data Matrix	$\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_l]^\top$
Label Matrix	$\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_l]^\top$
Number of projections	n_p
Projected Inputs	$\mathbf{X}' = \mathbf{XU}$
Projected Outputs	$\mathbf{Y}' = \mathbf{YV}$
Projection matrices	$\mathbf{U}$ $(N \times n_p)$, and $\mathbf{V}$ $(M \times n_p)$
Covariance	$\mathbf{C}_{xy} = E\{(\mathbf{x} - \boldsymbol{\mu}_x)(\mathbf{y} - \boldsymbol{\mu}_y)\} \sim \frac{1}{l}\mathbf{X}^\top\mathbf{Y}$
Frobenius norm of a matrix	$\Vert A\Vert_F^2 = \sum_{ij} a_{ij}^2$

Linear feature extraction

Toy example

- Imagine a classification problem in which labels matter (a lot!).
- “Blind” feature extraction is not a good choice.
- Let’s see what happens with different methods ...

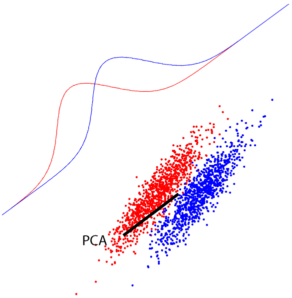


Linear feature extraction

Principal Component Analysis (PCA)

- *"Find projections maximizing the variance of the data:"*

$$\text{PCA:} \quad \begin{array}{ll} \text{maximize:} & \text{Tr}\{(\mathbf{XU})^\top(\mathbf{XU})\} = \text{Tr}\{\mathbf{U}^\top\mathbf{C}_{xx}\mathbf{U}\} \\ \text{subject to:} & \mathbf{U}^\top\mathbf{U} = \mathbf{I} \end{array}$$



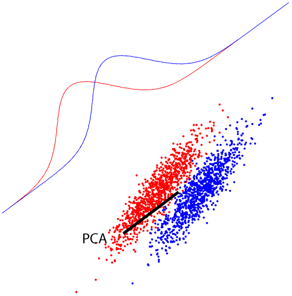
Linear feature extraction

Principal Component Analysis (PCA)

- *"Find projections maximizing the variance of the data:"*

$$\text{PCA:} \quad \text{maximize:} \quad \mathrm{Tr}\{(\mathbf{XU})^\top(\mathbf{XU})\} = \mathrm{Tr}\{\mathbf{U}^\top\mathbf{C}_{xx}\mathbf{U}\}$$
$$\text{subject to:} \quad \mathbf{U}^\top\mathbf{U} = \mathbf{I}$$

- `>> [U D] = eig(C);` [Prove it!]



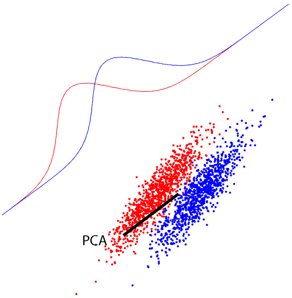
Linear feature extraction

Principal Component Analysis (PCA)

- *"Find projections maximizing the variance of the data:"*

$$\text{PCA:} \quad \text{maximize:} \quad \text{Tr}\{(\mathbf{XU})^\top(\mathbf{XU})\} = \text{Tr}\{\mathbf{U}^\top \mathbf{C}_{xx}\mathbf{U}\}$$
$$\text{subject to:} \quad \mathbf{U}^\top\mathbf{U} = \mathbf{I}$$

- `>> [U D] = eig(C);` [Prove it!]
- `>> opts.disp = 0; Nf=3; [U D] = eigs(C,Nf,'LM',opts);`

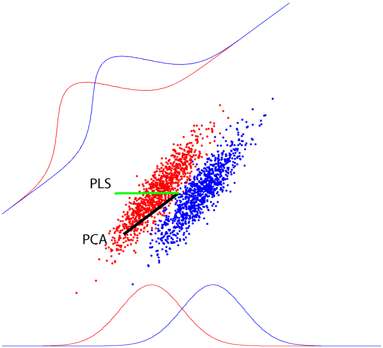


Linear feature extraction

Partial Least Squares (PLS)

- *“Find directions of maximum covariance between the projected input and output data:”*

$$\text{PLS:} \qquad \begin{array}{ll} \text{maximize:} & \mathrm{Tr}\{(\mathbf{XU})^\top(\mathbf{YV})\} = \mathrm{Tr}\{\mathbf{U}^\top\mathbf{C}_{xy}\mathbf{V}\} \\ \text{subject to:} & \mathbf{U}^\top\mathbf{U} = \mathbf{V}^\top\mathbf{V} = \mathbf{I} \end{array}$$



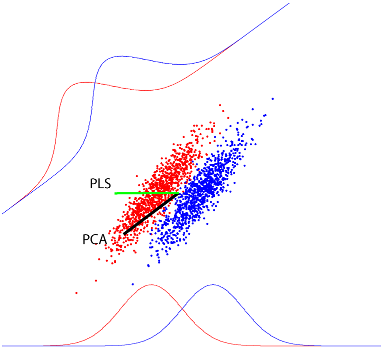
Linear feature extraction

Partial Least Squares (PLS)

- *"Find directions of maximum covariance between the projected input and output data:"*

$$\text{PLS:} \quad \text{maximize:} \quad \mathrm{Tr}\{(\mathbf{XU})^\top(\mathbf{YV})\} = \mathrm{Tr}\{\mathbf{U}^\top \mathbf{C}_{xy}\mathbf{V}\}$$
$$\text{subject to:} \quad \mathbf{U}^\top\mathbf{U} = \mathbf{V}^\top\mathbf{V} = \mathbf{I}$$

- `>> [U Sx Dx] = svds(X'*Y,Nf);` [Prove it!]



Linear feature extraction

Canonical correlation analysis (CCA), Hotelling (1936)

- Unlike PCA or PLS, CCA looks for directions of max I/O correlation:

$$\text{CCA:} \qquad \mathbf{u}, \mathbf{v} = \arg\max_{\mathbf{u},\mathbf{v}} \frac{(\mathbf{u}^\top \mathbf{C}_{xy} \mathbf{v})^2}{\mathbf{u}^\top \mathbf{C}_{xx} \mathbf{u} \; \mathbf{v}^\top \mathbf{C}_{yy} \mathbf{v}}$$

Linear feature extraction

Canonical correlation analysis (CCA), Hotelling (1936)

- Unlike PCA or PLS, CCA looks for directions of max I/O correlation:

$$\text{CCA:} \qquad \mathbf{u}, \mathbf{v} = \arg\max_{\mathbf{u},\mathbf{v}} \frac{(\mathbf{u}^\top \mathbf{C}_{xy} \mathbf{v})^2}{\mathbf{u}^\top \mathbf{C}_{xx} \mathbf{u} \; \mathbf{v}^\top \mathbf{C}_{yy} \mathbf{v}}$$

- This is invariant to a scaling of the projection vectors **u** and **v**, so ...

$$\text{CCA(2):} \qquad \mathbf{u}, \mathbf{v} = \arg\max_{\mathbf{u},\mathbf{v}} \; \mathbf{u}^\top \mathbf{C}_{xy} \mathbf{v}$$

$$\text{subject to: } \mathbf{u}^\top \mathbf{C}_{xx} \mathbf{u} = \mathbf{v}^\top \mathbf{C}_{yy} \mathbf{v} = 1$$

Linear feature extraction

Canonical correlation analysis (CCA), Hotelling (1936)

- Unlike PCA or PLS, CCA looks for directions of max I/O correlation:

$$\text{CCA:} \qquad \mathbf{u}, \mathbf{v} = \arg\max_{\mathbf{u},\mathbf{v}} \frac{(\mathbf{u}^\top \mathbf{C}_{xy} \mathbf{v})^2}{\mathbf{u}^\top \mathbf{C}_{xx} \mathbf{u} \; \mathbf{v}^\top \mathbf{C}_{yy} \mathbf{v}}$$

- This is invariant to a scaling of the projection vectors $\mathbf{u}$ and $\mathbf{v}$, so ...

$$\text{CCA(2):} \qquad \mathbf{u}, \mathbf{v} = \arg\max_{\mathbf{u},\mathbf{v}} \; \mathbf{u}^\top \mathbf{C}_{xy} \mathbf{v}$$

$$\text{subject to: } \; \mathbf{u}^\top \mathbf{C}_{xx} \mathbf{u} = \mathbf{v}^\top \mathbf{C}_{yy} \mathbf{v} = 1$$

- CCA in terms of the complete projection matrices $\mathbf{U}$ and $\mathbf{V}$:

$$\text{CCA(3):} \qquad \mathbf{U}, \mathbf{V} = \arg\max_{\mathbf{U},\mathbf{V}} \; \mathrm{Tr}\{\mathbf{U}^\top \mathbf{C}_{xy} \mathbf{V}\}$$

$$\text{subject to: } \; \mathbf{U}^\top \mathbf{C}_{xx} \mathbf{U} = \mathbf{V}^\top \mathbf{C}_{yy} \mathbf{V} = \mathbf{I}$$

Linear feature extraction

Canonical correlation analysis (CCA), Hotelling (1936)

- Unlike PCA or PLS, CCA looks for directions of max I/O correlation:

$$\text{CCA:} \qquad \mathbf{u}, \mathbf{v} = \arg\max_{\mathbf{u},\mathbf{v}} \frac{(\mathbf{u}^\top \mathbf{C}_{xy} \mathbf{v})^2}{\mathbf{u}^\top \mathbf{C}_{xx} \mathbf{u}\ \mathbf{v}^\top \mathbf{C}_{yy} \mathbf{v}}$$

- This is invariant to a scaling of the projection vectors **u** and **v**, so ...

$$\text{CCA(2):} \qquad \mathbf{u}, \mathbf{v} = \arg\max_{\mathbf{u},\mathbf{v}} \ \mathbf{u}^\top \mathbf{C}_{xy} \mathbf{v}$$

$$\text{subject to:}\ \ \mathbf{u}^\top \mathbf{C}_{xx} \mathbf{u} = \mathbf{v}^\top \mathbf{C}_{yy} \mathbf{v} = 1$$

- CCA in terms of the complete projection matrices **U** and **V**:

$$\text{CCA(3):} \qquad \mathbf{U}, \mathbf{V} = \arg\max_{\mathbf{U},\mathbf{V}} \ \mathrm{Tr}\{\mathbf{U}^\top \mathbf{C}_{xy} \mathbf{V}\}$$

$$\text{subject to:}\ \ \mathbf{U}^\top \mathbf{C}_{xx} \mathbf{U} = \mathbf{V}^\top \mathbf{C}_{yy} \mathbf{V} = \mathbf{I}$$

- Introducing Lagrange multipliers ...

$$\begin{pmatrix} \mathbf{0} & \mathbf{C}_{xy} \\ \mathbf{C}_{xy}^\top & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix} = \lambda \begin{pmatrix} \mathbf{C}_{xx} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{yy} \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix}$$

Linear feature extraction

Canonical correlation analysis (CCA), Hotelling (1936)

- Unlike PCA or PLS, CCA looks for directions of max I/O correlation:

$$\text{CCA:} \qquad \mathbf{u},\mathbf{v} = \arg\max_{\mathbf{u},\mathbf{v}} \frac{(\mathbf{u}^\top \mathbf{C}_{xy}\mathbf{v})^2}{\mathbf{u}^\top \mathbf{C}_{xx}\mathbf{u}\ \mathbf{v}^\top \mathbf{C}_{yy}\mathbf{v}}$$

- This is invariant to a scaling of the projection vectors **u** and **v**, so ...

$$\text{CCA(2):} \qquad \mathbf{u},\mathbf{v} = \arg\max_{\mathbf{u},\mathbf{v}} \ \mathbf{u}^\top \mathbf{C}_{xy}\mathbf{v}$$
$$\text{subject to: } \ \mathbf{u}^\top \mathbf{C}_{xx}\mathbf{u} = \mathbf{v}^\top \mathbf{C}_{yy}\mathbf{v} = 1$$

- CCA in terms of the complete projection matrices **U** and **V**:

$$\text{CCA(3):} \qquad \mathbf{U},\mathbf{V} = \arg\max_{\mathbf{U},\mathbf{V}} \ \mathrm{Tr}\{\mathbf{U}^\top \mathbf{C}_{xy}\mathbf{V}\}$$
$$\text{subject to: } \ \mathbf{U}^\top \mathbf{C}_{xx}\mathbf{U} = \mathbf{V}^\top \mathbf{C}_{yy}\mathbf{V} = \mathbf{I}$$

- Introducing Lagrange multipliers ...

$$\begin{pmatrix} \mathbf{0} & \mathbf{C}_{xy} \\ \mathbf{C}_{xy}^\top & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix} = \lambda \begin{pmatrix} \mathbf{C}_{xx} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{yy} \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix}$$

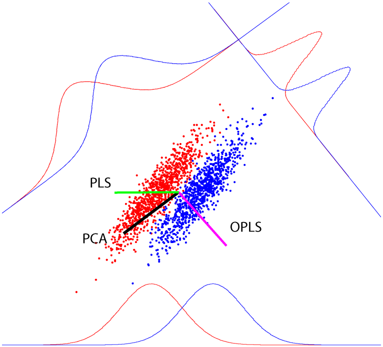
- `>> A = [0 Cxy;Cxy' 0]; B = [Cxx 0;0 Cyy]; [UV D] = eig(A,B);`

Linear feature extraction

Orthonormalized Partial Least Squares (OPLS)

- *"OPLS chooses the projection* $\mathbf{U}$ *to make* $\mathbf{X}'$ *the best approximation to* $\mathbf{X}$ *in a reduced dimensionality space:"*

$$\text{OPLS:} \qquad \begin{array}{ll} \text{find:} & \mathbf{U} = \arg\min\{\|\mathbf{Y} - \mathbf{X}'\mathbf{W}\|_F^2\} \\ \text{where:} & \mathbf{W} = (\mathbf{X}'^{\top}\mathbf{X}')^{-1}\mathbf{X}'\mathbf{Y} \end{array}$$

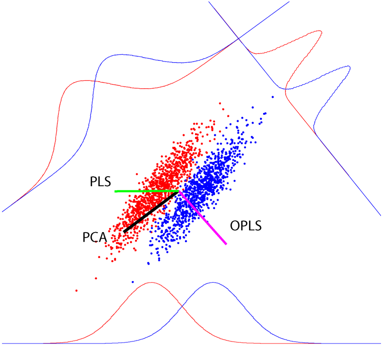


Linear feature extraction

Orthonormalized Partial Least Squares (OPLS)

- *"... which can be rewritten as [Worsley98]:"*

$$\text{OPLS:} \quad \begin{array}{ll} \text{maximize:} & \mathrm{Tr}\{\mathbf{U}^\top \mathbf{C}_{xy} \mathbf{C}_{xy}^\top \mathbf{U}\} \\ \text{subject to:} & \mathbf{U}^\top \mathbf{C}_{xx} \mathbf{U} = \mathbf{I} \end{array}$$



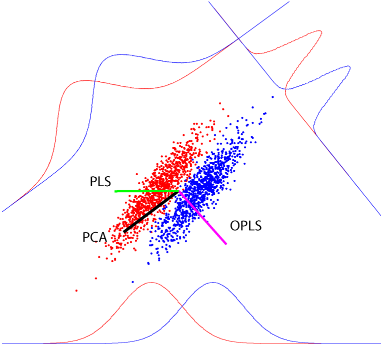
Linear feature extraction

Orthonormalized Partial Least Squares (OPLS)

- *"... which can be rewritten as [Worsley98]:"*

$$\text{OPLS:} \quad \text{maximize:} \quad \mathrm{Tr}\{\mathbf{U}^\top \mathbf{C}_{xy} \mathbf{C}_{xy}^\top \mathbf{U}\}$$
$$\text{subject to:} \quad \mathbf{U}^\top \mathbf{C}_{xx} \mathbf{U} = \mathbf{I}$$

- `>> [U,D] = eig((X'*Y)*(Y'*X),X'*X);` [Prove it!]



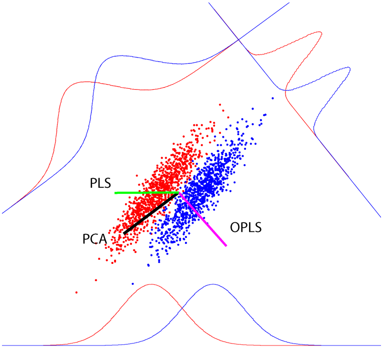
Linear feature extraction

Orthonormalized Partial Least Squares (OPLS)

- *"... which can be rewritten as [Worsley98]:"*

$$\text{OPLS:} \qquad \text{maximize:} \quad \mathrm{Tr}\{\mathbf{U}^\top \mathbf{C}_{xy} \mathbf{C}_{xy}^\top \mathbf{U}\}$$
$$\text{subject to:} \quad \mathbf{U}^\top \mathbf{C}_{xx} \mathbf{U} = \mathbf{I}$$

- `>> [U,D] = eig((X'*Y)*(Y'*X),X'*X);` [Prove it!]
- `>> [U,D] = eig(inv(X'*X)*(X'*Y)*(Y'*X));` [Prove it!]



Remarks

Remarks on linear feature extraction for supervised problems

- Feature extraction is important for *understanding* and *processing* (classification and regression)
- Labels *must* play an important role in feature extraction
- Traditional PCA fails since labels are obviated
- Traditional PLS does a good, yet suboptimal, job
- Orthonormalized PLS excels in linear feature extraction

Remarks

Remarks on linear feature extraction for supervised problems

- Feature extraction is important for *understanding* and *processing* (classification and regression)
- Labels *must* play an important role in feature extraction
- Traditional PCA fails since labels are obviated
- Traditional PLS does a good, yet suboptimal, job
- Orthonormalized PLS excels in linear feature extraction
- Optimality:
  - PCA is optimal for reconstruction error
  - CCA is optimal for maximizing correlation with output
  - PLS is optimal for maximizing covariance with output
  - OPLS is optimal for minimizing MSE

Linear vs. Non-linear feature extraction

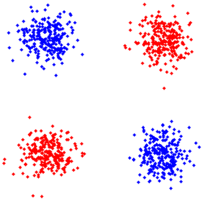
Linear feature extraction. Advantages

- Simplicity.
- Easy to understand.
- Leads to convex optimization problems.

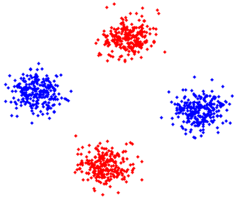
Linear feature extraction. Drawbacks

- Unsuitable for non-linear problems
- More dimensions than points?

Linear vs. Non-linear feature extraction

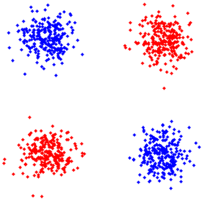


Original data

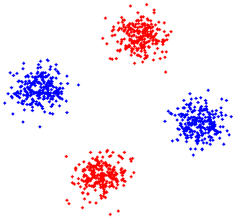


PCA

Linear vs. Non-linear feature extraction



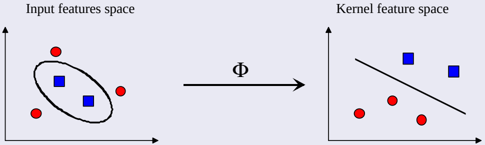
Original data



OPLS

Kernel methods for non-linear feature extraction

Kernel methods



1. Map the data to an ∞-dimensional feature spaces, $\mathcal{H}$.
2. Solve a linear problem there.

Kernel trick

- No need to know ∞ coordinates for each mapped sample $\phi(\mathbf{x}_i)$
- *Kernel trick: "if an algorithm can be expressed in the form of dot products, its non-linear (kernel) version only needs the dot products among mapped samples, the so-called kernel function:"*

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$$

- Using this trick, we can implement K-PCA, K-PLS, K-OPLS, etc.

Kerneling PCA ...

Principal Component Analysis (PCA)

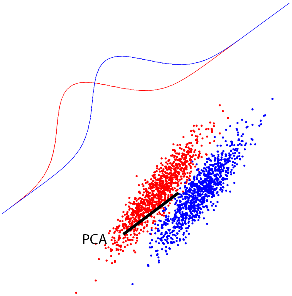
- *"Find projections maximizing the variance of the data:"*

$$\text{PCA:} \quad \text{maximize:} \quad \mathrm{Tr}\{(\mathbf{X}\mathbf{U})^\top(\mathbf{X}\mathbf{U})\} = \mathrm{Tr}\{\mathbf{U}^\top \mathbf{C}_{xx}\mathbf{U}\}$$
$$\text{subject to:} \quad \mathbf{U}^\top\mathbf{U} = \mathbf{I}$$

- Including Lagrange multipliers λ, this problem is equivalent to

$$\mathbf{C}_{xx}\mathbf{U} = \lambda\mathbf{U}$$

```
>> [U lambda] = eig(C);
>> [U lambda] = eigs(C,p);
```



Kerneling PCA ...

Kernel Principal Component Analysis (KPCA)

- *"Find projections maximizing the variance of the mapped data:"*

$$\text{KPCA:} \qquad \begin{aligned} &\text{maximize:} && \text{Tr}\{(\mathbf{\Phi U})^\top(\mathbf{\Phi U})\} = \text{Tr}\{\mathbf{U}^\top\tilde{\mathbf{\Phi}}^\top\tilde{\mathbf{\Phi}}\mathbf{U}\} \\ &\text{subject to:} && \mathbf{U}^\top\mathbf{U} = \mathbf{I} \end{aligned}$$

- The term $\tilde{\mathbf{\Phi}}^\top\tilde{\mathbf{\Phi}}$ is $d_{\mathcal{H}} \times d_{\mathcal{H}}$!!!

Kerneling PCA ...

Kernel Principal Component Analysis (KPCA)

- *"Find projections maximizing the variance of the mapped data:"*

$$\text{KPCA:} \qquad \begin{array}{ll} \text{maximize:} & \mathrm{Tr}\{(\mathbf{\Phi U})^\top(\mathbf{\Phi U})\} = \mathrm{Tr}\{\mathbf{U}^\top\tilde{\mathbf{\Phi}}^\top\tilde{\mathbf{\Phi}}\mathbf{U}\} \\ \text{subject to:} & \mathbf{U}^\top\mathbf{U} = \mathbf{I} \end{array}$$

- The term $\tilde{\mathbf{\Phi}}^\top\tilde{\mathbf{\Phi}}$ is $d_{\mathcal{H}} \times d_{\mathcal{H}}$!!!

Kernel Principal Component Analysis

- Apply the representer's theorem: $\mathbf{U} = \tilde{\mathbf{\Phi}}^\top\mathbf{A}$ where $\mathbf{A} = [\alpha_1, \ldots, \alpha_n]^\top$
- *"Find projections maximizing the variance of the mapped data:"*

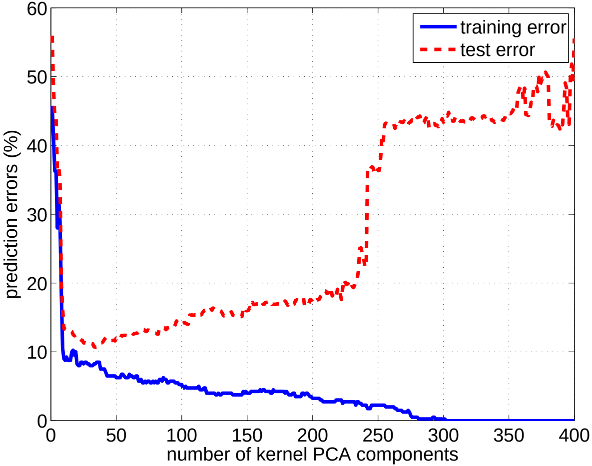
$$\text{KPCA (2):} \qquad \begin{array}{ll} \text{maximize:} & \mathrm{Tr}\{\mathbf{A}^\top\mathbf{K}_x\mathbf{K}_x\mathbf{A}\} \\ \text{subject to:} & \mathbf{A}^\top\mathbf{K}_x\mathbf{A} = \mathbf{I} \end{array}$$

- Including Lagrange multipliers λ, this problem is equivalent to

$$\mathbf{K}_x\mathbf{K}_x\alpha = \lambda\mathbf{K}_x\alpha \rightarrow \mathbf{K}_x\alpha = \lambda\alpha$$

Problem 1: the intrinsic dimensionality

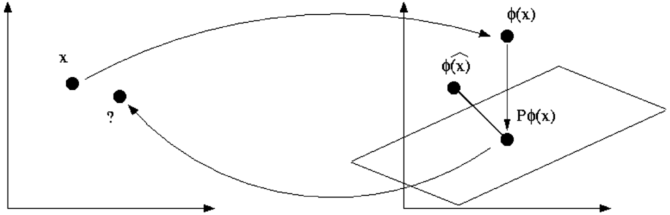
- Choosing the kernel and its parameter(s)
- Choosing the number of eigenvectors



Problem 2: Finding preimages

"Given a point in $\mathcal{H}$, find the corresponding point in $\mathcal{X}$"

- For many points in the feature space there is no exact pre-image in the input space
- Inverting the mapping ϕ is an ill-posed problem
- Some relaxed solutions exist.

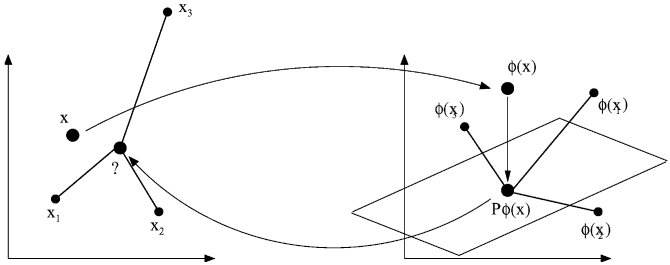


Problem 2: Finding preimages

- Mika99: 'minimize the feature space distance $\|\phi(\hat{\mathbf{x}}) - P\varphi(\mathbf{x})\|$'
  - Iterative procedure, very computationally demanding
  - local minimum
  - inestable solutions

Problem 2: Finding preimages

- Mika99: 'minimize the feature space distance $\|\phi(\hat{\mathbf{x}}) - P\varphi(\mathbf{x})\|$'
  - Iterative procedure, very computationally demanding
  - local minimum
  - inestable solutions
- Kwok04: 'constrain input distances by computing neighbor dist. in $\mathcal{H}$'



Problem 2: Finding preimages

noisy image; (300 training images) Mika *et al.*; proposed method; (60 training images) Mika *et al.*; proposed method

number of training images	σ^2	SNR		
		noisy images	our method	Mika *et al.*
300	0.25	2.32	**6.36**	5.90
	0.3	1.72	**6.24**	5.60
	0.4	0.91	**5.89**	5.17
	0.5	0.32	**5.58**	4.86
60	0.25	2.32	**4.64**	4.50
	0.3	1.72	**4.56**	4.39
	0.4	0.90	**4.41**	4.19
	0.5	0.35	**4.29**	4.06

Experiment 1: Image denoising

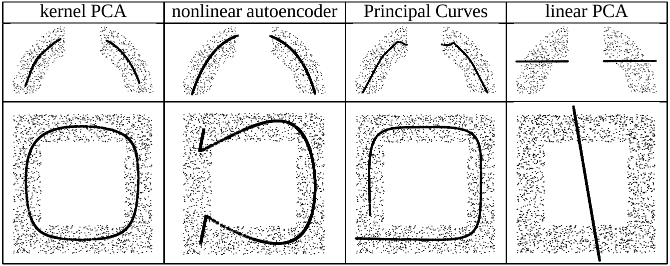


Figure 1: De-noising in 2-d (see text). Depicted are the data set (small points) and its de-noised version (big points, joining up to solid lines). For linear PCA, we used one component for reconstruction, as using two components, reconstruction is perfect and thus does not de-noise. Note that all algorithms except for our approach have problems in capturing the circular structure in the bottom example.

Experiment 1: Image denoising

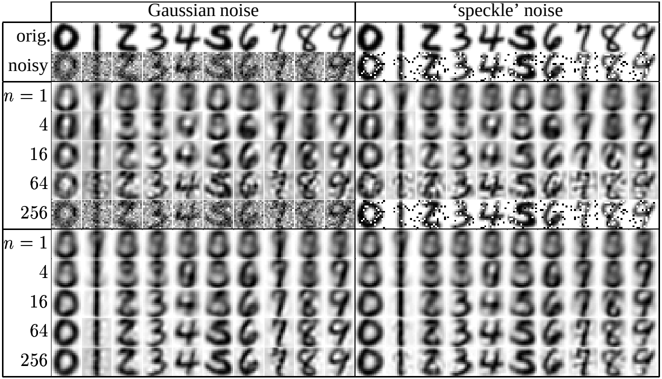
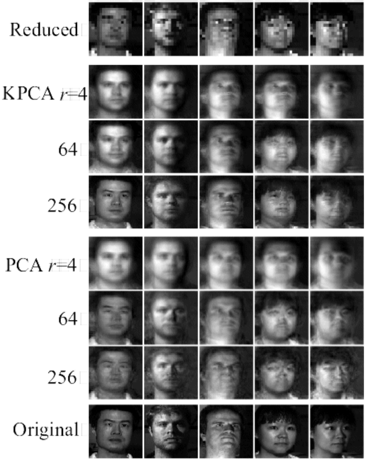


Figure 4: De-Noising of USPS data (see text). The left half shows: *top:* the first occurrence of each digit in the test set, *second row:* the upper digit with additive Gaussian noise ($\sigma = 0.5$), *following five rows:* the reconstruction for linear PCA using $n = 1, 4, 16, 64, 256$ components, and, *last five rows:* the results of our approach using the same number of components. In the right half we show the same but for 'speckle' noise with probability $p = 0.4$.

Experiment 2: Image superresolution

- Collect high-res face images
- Use KPCA with RBF-kernel to learn non-linear subspaces
- For new low-res image:
  - scale to target high resolution
  - project to closest point in face subspace



reconstruction in r dimensions

Signal and noise

Signal vs noise

- Signal: magnitude generated by an inaccesible system, $\mathbf{s}_k$
- Noise: magnitude generated by the medium corrupting the signal, $\mathbf{n}_k$
- Observation: signal corrupted by noise, $\mathbf{x}_k = \mathbf{s}_k + \mathbf{n}_k$, $k = 1, \ldots, n$

Separating signal from noise

- Eigenvalue perspective: the noise is in the low eigenvalues
- Feature extractors
  - PCA: retain the eigenvectors with higher eigenvalues
  - ICA: find the non-orthogonal projection of the signal with maximal independent axes
  - PLS: find projections maximally aligned with the labels
- Many feature extractors have been kernelized ...
- ... but all of them disregard the noise characteristics!

Signal-to-noise ratio transformation

Notation

- Observation: $\mathbf{x}_i \in \mathbb{R}^N$, $i = 1, \ldots, n$
- Additive noise model: $\mathbf{x}_i = \mathbf{s}_i + \mathbf{n}_i$
- Matrix notation: $\mathbf{X} = \mathbf{S} + \mathbf{N}$, $\mathbf{X} \in \mathbb{R}^{n \times N}$.

The SNR transformation

- Define a linear transform $\mathbf{\Psi}$ such that maximizes the SNR:

$$\text{SNR} = \max_{\mathbf{\Psi} \neq 0} \frac{\|\mathbf{S\Psi}\|^2}{\|\mathbf{N\Psi}\|^2} \approx \max_{\mathbf{\Psi} \neq 0} \frac{\|\mathbf{X\Psi}\|^2}{\|\mathbf{N\Psi}\|^2},$$

- Assumed that signal and noise are mutually orthogonal:

$$\mathbf{S}^\top \mathbf{N} = 0, \mathbf{N}^\top \mathbf{S} = 0$$

- This is equivalent to solving the generalized eigenproblem:

$$\mathbf{X}^\top \mathbf{X\Psi} = \mu \mathbf{N}^\top \mathbf{N\Psi}$$

- We only need to estimate the signal covariance, $\mathbf{C}_{xx} = \mathbf{X}^\top \mathbf{X}$, and the noise covariance, $\mathbf{C}_{nn} \approx \mathbf{N}^\top \mathbf{N}$.

Signal-to-noise ratio transformation

The noise covariance estimation

Assume stationary processes in wide sense:

- Differentiation: $\mathbf{n}_i \approx \mathbf{x}_i - \mathbf{x}_{i-1}$
- Smoothing filtering: $\mathbf{n}_i \approx \mathbf{x}_i - \frac{1}{M}\sum_{k=1}^{M} a_k \mathbf{x}_{i-k}$
- Wiener estimates
- Wavelet domain estimates
- ....

The MatLab SNR code

```
>> X = standardize(X);
>> N = diff(X);
>> [V D] = eig(X'*X,N'*N);
```

Standard kernelization

KSNR through kernel trick

- Replace $\mathbf{X} \in \mathbb{R}^{n \times N}$ with $\mathbf{\Phi} \in \mathbb{R}^{n \times N_{\mathcal{H}}}$
- Replace $\mathbf{N} \in \mathbb{R}^{n \times N}$ with $\mathbf{\Phi}_N \in \mathbb{R}^{n \times N_{\mathcal{G}}}$

$$\mathbf{\Phi}^\top \mathbf{\Phi} \mathbf{\Psi} = \mu \mathbf{\Phi}_N^\top \mathbf{\Phi}_N \mathbf{\Psi},$$

- Not solvable in its present form given the inaccessibility and high dimensionality of the involved matrices, $N_{\mathcal{H}} \times N_{\mathcal{H}}$ and $N_{\mathcal{G}} \times N_{\mathcal{G}}$.
- Left multiply both sides by $\mathbf{\Phi}$, and use representer's theorem, $\mathbf{\Psi} = \mathbf{\Phi}^\top \mathbf{L}$:

$$\mathbf{K}^2 \mathbf{L} = \mu \mathbf{K}_N \mathbf{K}_N^\top \mathbf{L},$$

  where
  - $\mathbf{K} = \mathbf{\Phi}\mathbf{\Phi}^\top$ has elements $K(\mathbf{x}_i, \mathbf{x}_j)$
  - $\mathbf{K} = \mathbf{\Phi}\mathbf{\Phi}_N^\top$ has elements $K_N(\mathbf{x}_i, \mathbf{n}_j)$
- Easy and simple to program!
- Potentially useful when signal and noise are nonlinearly related: occlusion, strips, saturation, etc.
- Two critical parameters to estimate!

Standard kernelization

The MatLab KSNR code

```
>> X = standardize(X);
>> sigma1 = estimateSigma(X,X);
>> Ks = kernelmatrix('rbf',X,X,sigma1);
>> Ksc = centering(Ks);
>> N = diff(X);
>> sigma2 = estimateSigma(X,N);
>> Kn = kernelmatrix('rbf',X,N,sigma2);
>> Knc = centering(Kn);
>> [V D] = eig(Ksc*Ksc,Knc*Knc');
```

Results in unsupervised change detection

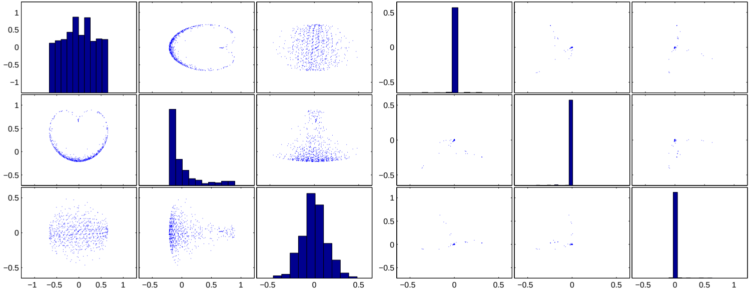
- RGB data from the DLR 3K camera system
- 3 cameras (16 Megapix) mounted in a plane
- Speed: 3 Hz.
- Two images acquired 0.7 seconds apart cover a busy motorway
- Changes dominated by car movement
- Additional changes: aircraft movement and different viewing angles

t_1 **image**

t_2 **image**

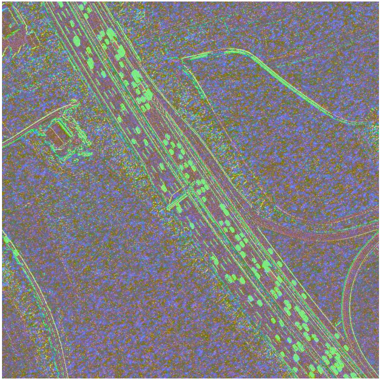
$|t_2 - t_1|$ **image**

Results in unsupervised change detection

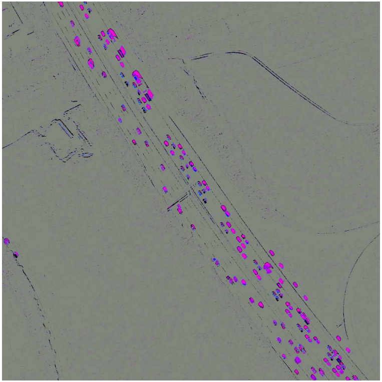


(a) kPCA. (b) kMAF.

Results in unsupervised change detection



KPCA (first 3 PCs)



KSNR (first 3 PCs)

Objectives

- Optimality: We focus on the OPLS.
- Kernelization: We present the Kernel Orthonormalized PLS (KOPLS).
- Scalability: We also make the method algorithmically feasible.
- We analyze and characterize the method:
  1. *Theoretically:*
     - Computational cost.
     - Memory.
     - Number of projections.
  2. *Experimentally:*
     - Toy examples.
     - Remote Sensing image classification.
     - Biophysical parameter estimation.

Kernel PLS

Notation

Data	$\{\phi(\mathbf{x}_i), \mathbf{y}_i\}_{i=1}^{l}$
Mapping	$\phi(\mathbf{x}) : \mathbb{R}^N \rightarrow \mathcal{H}$
Mapped inputs matrix	$\mathbf{\Phi} = [\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_l)]^\top$
Output matrix	$\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_l]^\top$
Number of projections	n_p
Projections of Mapped Inputs	$\mathbf{\Phi}' = \mathbf{\Phi}\mathbf{U}$
Projections of Outputs	$\mathbf{Y}' = \mathbf{Y}\mathbf{V}$
Projection matrices	$\mathbf{U}$ $(dim(\mathcal{H}) \times n_p)$, and $\mathbf{V}$ $(M \times n_p)$

Formulation

- *"The objective of KPLS is to find directions for maximum covariance:"*

$$\text{KPLS:} \qquad \begin{array}{ll} \text{maximize:} & \mathrm{Tr}\{\mathbf{U}^\top \tilde{\mathbf{\Phi}}^\top \tilde{\mathbf{Y}}\mathbf{V}\} \\ \text{subject to:} & \mathbf{U}^\top\mathbf{U} = \mathbf{V}^\top\mathbf{V} = \mathbf{I} \end{array}$$

  where $\tilde{\mathbf{\Phi}}$ and $\tilde{\mathbf{Y}}$ are centered versions of $\mathbf{\Phi}$ and $\mathbf{Y}$, respectively.
- Only a matrix of inner products of the patterns in $\mathcal{H}$ is needed (Shawe-Taylor, 2004).

Kernel Orthonormalized PLS

Formulation of the KOPLS

- *"The objective of KOPLS is:"*

$$\text{KOPLS:} \qquad \begin{array}{ll} \text{maximize:} & \text{Tr}\{\mathbf{U}^\top \tilde{\mathbf{\Phi}}^\top \tilde{\mathbf{Y}}\tilde{\mathbf{Y}}^\top \tilde{\mathbf{\Phi}}\mathbf{U}\} \\ \text{subject to:} & \mathbf{U}^\top \tilde{\mathbf{\Phi}}^\top \tilde{\mathbf{\Phi}}\mathbf{U} = \mathbf{I} \end{array}$$

- The features derived from KOPLS are optimal (in the MSE sense).

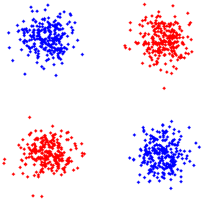
Kernel trick for the KOPLS

- All projection vectors (the columns of $\mathbf{U}$) can be expressed as a linear combination of the training data, $\mathbf{U} = \tilde{\mathbf{\Phi}}^\top \mathbf{A}$.
- The maximization problem is reformulated as:

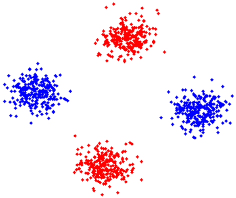
$$\text{KOPLS:} \qquad \begin{array}{ll} \text{maximize:} & \text{Tr}\{\mathbf{A}^\top \mathbf{H}_x \mathbf{H}_y \mathbf{H}_x \mathbf{A}\} \\ \text{subject to:} & \mathbf{A}^\top \mathbf{H}_x \mathbf{H}_x \mathbf{A} = \mathbf{I} \end{array}$$

- Centered kernel matrices: $\mathbf{H}_x = \tilde{\mathbf{\Phi}}\tilde{\mathbf{\Phi}}^\top$ and $\mathbf{H}_y = \tilde{\mathbf{Y}}\tilde{\mathbf{Y}}^\top$.
- **This is a generalized eigenproblem:** $\mathbf{H}_x\mathbf{H}_y\mathbf{H}_x\alpha = \lambda\mathbf{H}_x\mathbf{H}_x\alpha$
- $\mathbf{H}_x$ and $\mathbf{H}_y$ can be approximated without computing and storing the whole matrices.

An illustrative example (cont'd)

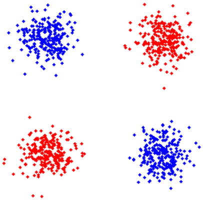


Original data

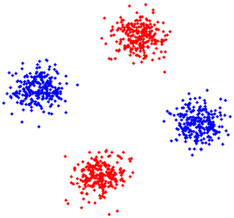


PCA

An illustrative example (cont'd)

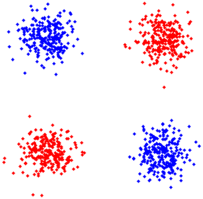


Original data

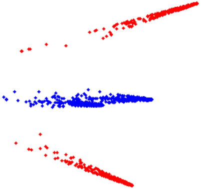


OPLS

An illustrative example (cont'd)

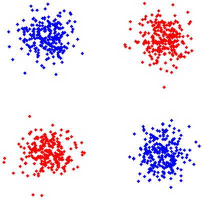


Original data

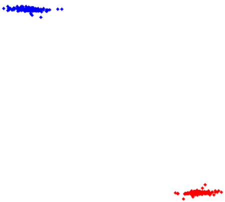


KPCA

An illustrative example (cont'd)



Original data



KOPLS

Remarks

Remarks on non-linear feature extraction

- Linear methods such as PCA, PLS or OPLS are not suitable for non-linear classification/regression tasks.
- Non-linear versions of these algorithms are readily obtained by applying the *kernel trick*.
- KPLS and KOPLS consider labels for the derivation of the projection vector, thus outperforming KPCA.
- KOPLS inherits mean-square-error optimality from its linear counterpart.

Methods Characterization

	KOPLS	KPLS
Kernel size	$l \times l$	$l \times l$
Storage	$O(l^2)$	$O(l^2)$
Max. n_p	$\min\{rank(\mathbf{\Phi}), rank(\mathbf{Y})\}$	$rank(\mathbf{\Phi})$

Experiment 1: Classification of LandSat images

Data collection

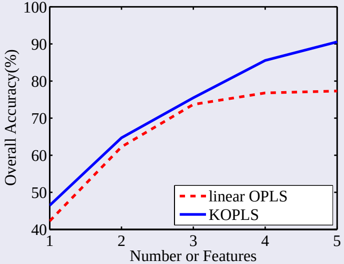
- LandSat image, 82×100 pixels with a spatial resolution of 80m×80m
- Six classes: red soil, cotton crop, grey soil, damp grey soil, soil with vegetation stubble and very damp grey soil.
- Contextual information: stack neighbouring pixels in 3×3 windows → **high-dimensional and redundant feature vectors!**.
- Training: 4435 samples.
- Testing: 2000 samples.

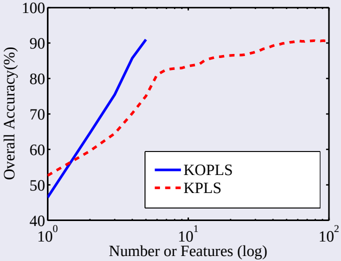
Experimental setup

- Methods: linear OPLS, KPLS and KOPLS.
- RBF kernel: $k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2\right)$
- 10-fold cross-validation on the training set to estimate σ.
- Classification procedure:
  1. Extract n_p projections ($n_p < rank(\mathbf{Y})$ for the KOPLS).
  2. Project test data.
  3. Linear discriminant with the pseudoinverse of the projected data.
  4. Winner-takes-all.

Experiment 1: Classification of LandSat images

Accuracy and feature expression





- The non-linear method provides a better representation of the discriminative information.
- KOPLS performance, with only 5 features, is 91%.
- KPLS needs 100 features to achieve similar performance.
- *Conclusions:*
  1. Non-linear OPLS methods provide much better results.
  2. KOPLS yields features which contain more discriminative information

Experiment 2: Oceanic chlorophyll concentration

Data collection

- *"Modeling the non-linear relationship between chlorophyll concentration and marine reflectance."*
- SeaBAM dataset (O'Reilly, 1998).
- 919 *in-situ* pigment measurements around the United States and Europe.
- Training: 460 samples
- Testing: 460 samples

Experimental setup

- Methods: linear PLS, KPLS and KOPLS.
- RBF kernel: $k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2\right)$
- Leave-one-out root mean square error (LOO-RMSE) to validate the model.
- σ tuned in the range $[10^{-2}, 10^4]$
- $n_p = rank(\mathbf{Y}) = 1$ for the KOPLS.

Experiment 2: Oceanic chlorophyll concentration

Accuracy and feature expression

Model	ME	RMSE	MAE	r
OPLS	-0.034	0.257	0.188	0.903
KPLS, $n_p = 1$	0.042	0.366	0.278	0.790
KPLS, $n_p = 5$	-0.013	0.189	0.140	0.947
KPLS, $n_p = 10$	-0.013	0.149	0.115	0.968
KPLS, $n_p = 20$	-0.009	0.138	0.106	0.972
KOPLS, $n_p = 1$	-0.015	0.154	0.111	0.967

- Linear OPLS performs poorly as the linear assumption does not hold.
- KPLS and the proposed KOPLS show a clear improvement in both accuracy and bias compared to linear OPLS
- KPLS and KOPLS show similar accuracy to SVR, and outperform in bias.
- Results obtained with a lower computational and storage burden
- The *only one* feature extracted with KOPLS provides a similar performance to the 10 *first features* from KPLS.

Conclusions

Conclusions

- Given definition of the most useful kernel methods for nonlinear feature extraction
- KPCA is nice but difficult to handle (proper sigma for a task?)
- Unlike KPLS, the proposed KOPLS is optimal in the sense of a minimum quadratic error approximation of the label matrix.
- Major problem: non-sparse computationally demanding methods
- Other kernel methods are available:
  - Kernel CCA
  - ...
- Everything relies on the proper definition of the kernel (again)

References

References

- J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press, 2004.
- G. Camps-Valls, J. L. Rojo and M. Martinez, *Kernel Methods in Bioengineering, Signal and Image Processing*, Idea Inc., 2007.
- R. Rosipal and N. Kramer, “Overview and recent advances in partial least squares,” *Subspace, Latent Structure and Feature Selection Techniques*, 2006.
- J. Arenas-García and G. Camps-Valls. “Efficient Kernel Orthonormalized PLS for Remote Sensing Applications.” IEEE Transactions on Geoscience and Remote Sensing, 2008, Volume: 46, Issue 10, Part 1. 2872-2881