SPARSE KERNEL ORTHONORMALIZED PLS FOR FEATURE EXTRACTION IN LARGE DATA SETS

Jerónimo Arenas-García, Kaare Brandt Petersen and Lars Kai Hansen
Informatics and Mathematical Modelling, Technical University of Denmark

OVERVIEW

- We propose a kernel extension of Orthonormalized PLS for feature extraction, within the framework of Kernel Multivariate Analysis (KMVA)
- KMVA methods have dense solutions and, therefore, scale badly for large datasets
- By imposing sparsity, we propose a modified KOPLS algorithm with reduced complexity (rKOPLS)
- The resulting scheme is a powerful feature extractor for regression and classification problems, including the multi-class and multi-label cases

1 Orthonormalized PLS (OPLS)

- Training data: $\mathbf{\Phi} = [\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_l)]^T$; $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_l]^T$
- Find projection vectors for feature extraction:

$$\tilde{\mathbf{\Phi}}' = \tilde{\mathbf{\Phi}}\mathbf{U}$$

where each column in $\mathbf{U}$ $(dim(\mathcal{F}) \times n_p)$ is a projection vector, and $\tilde{\mathbf{\Phi}}$ is a centered version of $\mathbf{\Phi}$

- OPLS:

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

- OPLS properties:
  - Only projections for input data; projected data is white
  - Optimal features for linear prediction (with a bottleneck) of the training labels with square loss, i.e.,

$$\mathbf{U} = \arg\min_{\mathbf{U}} \|\tilde{\mathbf{Y}} - \tilde{\mathbf{\Phi}}'\hat{\mathbf{B}}\|_F^2, \quad \hat{\mathbf{B}} = (\tilde{\mathbf{\Phi}}'^T\tilde{\mathbf{\Phi}}')^{-1}\tilde{\mathbf{\Phi}}'^T\tilde{\mathbf{Y}}$$

2 Kernel OPLS

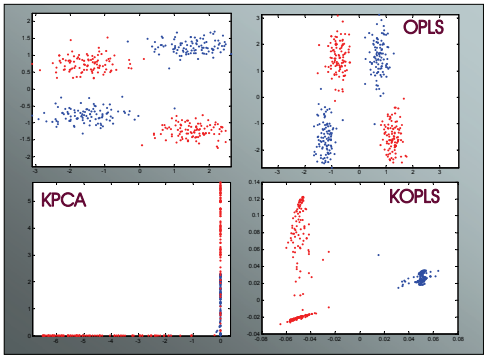
- Representer Theorem: $\mathbf{U} = \tilde{\mathbf{\Phi}}^T\mathbf{A}$
- KOPLS:

$$\begin{array}{ll} \text{maximize:} & \mathrm{Tr}\{\mathbf{A}^T\mathbf{K}_x\mathbf{K}_y\mathbf{K}_x\mathbf{A}\} \\ \text{subject to:} & \mathbf{A}^T\mathbf{K}_x\mathbf{K}_x\mathbf{A} = \mathbf{I} \end{array}$$

- Matrix $\mathbf{A}$ is the solution of a gen. eigenvalue problem

$$\mathbf{K}_x\mathbf{K}_y\mathbf{K}_x\mathbf{A} = \mathbf{K}_x\mathbf{K}_x\mathbf{A}\mathbf{\Lambda}$$

with $\mathbf{K}_x = \tilde{\mathbf{\Phi}}\tilde{\mathbf{\Phi}}^T$ and $\mathbf{K}_y = \tilde{\mathbf{Y}}\tilde{\mathbf{Y}}^T$



Pros

+ Expressive FE for regression and classification problems, including the multi-class and multi-label cases
+ Unlike other KMVA, KOPLS involves projections in the input space only
+ Associated to a well-known cost function

Cons

– Kernel computations: l^2
– Memory requirements: $2\,l^2$
– Dense solution: l nodes
– $\mathbf{K}_x\mathbf{K}_x$ is typically singular
– Solution may overfit

3 Sparse approximation of the KOPLS solution (rKOPLS)

- We use $\mathbf{U} = \mathbf{\Phi}_R^T\mathbf{B}$, where $\mathbf{\Phi}_R$ is a subset of the train dataset including R patterns ($R < l$)
- This leads to the generalized eigenvalue problem

$$\mathbf{K}_R\mathbf{K}_y\mathbf{K}_R^T\mathbf{B} = \mathbf{K}_R\mathbf{K}_R^T\mathbf{B}\mathbf{\Lambda}$$

where $\mathbf{K}_R = \mathbf{\Phi}_R\tilde{\mathbf{\Phi}}^T$ is a reduced ($R \times l$) kernel matrix

Iterative rKOPLS algorithm

1) Compute matrices $\mathbf{K}_R\mathbf{K}_R^T$ and $\mathbf{K}_R\mathbf{Y}$ using

$$\mathbf{K}_R\mathbf{K}_R^T = \sum \mathbf{k}_i\mathbf{k}_i^T; \quad \mathbf{K}_R\mathbf{Y} = \sum \mathbf{k}_i\mathbf{y}_i^T \qquad (\mathbf{k}_i = \mathbf{\Phi}_R\phi(\mathbf{x}_i))$$

2) Repeat until desired or maximum number of projections:
   i. Find largest gen. eigenvalue and eigenvector pair: $\{\lambda_i, \boldsymbol{\beta}_i\}$
   ii. Normalize eigenvector so that $\boldsymbol{\beta}_i\mathbf{K}_R\mathbf{K}_R^T\boldsymbol{\beta}_i = 1$
   iii. Deflate $\mathbf{K}_R\mathbf{K}_y\mathbf{K}_R^T$ according to

$$\mathbf{K}_R\mathbf{K}_y\mathbf{K}_R^T \leftarrow \mathbf{K}_R\mathbf{K}_y\mathbf{K}_R^T - \lambda_i\mathbf{K}_R\mathbf{K}_R^T\boldsymbol{\beta}_i\boldsymbol{\beta}_i^T\mathbf{K}_R\mathbf{K}_R^T$$

(equivalent to $\mathbf{K}_y \leftarrow \mathbf{K}_y - \lambda_i\mathbf{K}_R^T\boldsymbol{\beta}_i\boldsymbol{\beta}_i^T\mathbf{K}_R$)

Now:

+ Kernel computations: $R \cdot l$
+ Memory requirements: $R(R + \dim(\mathbf{y}))$
+ Sparse solution with R nodes
+ $\mathbf{K}_R\mathbf{K}_R^T$ is better conditioned than $\mathbf{K}_x\mathbf{K}_x$
+ R acts as a sort of regularizer

But:

– Random selection of centroids $\mathbf{\Phi}_R$
– Additional parameter R: subject to a tradeoff performance vs complexity

Comparison with other MVA methods

	KOPLS	KPLS	KPLS2
Max. Problem	$\max \mathrm{Tr}\{\mathbf{A}^T\mathbf{K}_x\mathbf{K}_Y\mathbf{K}_x\mathbf{A}\}$ s. t. $\mathbf{A}^T\mathbf{K}_x\mathbf{K}_x\mathbf{A} = \mathbf{I}$	$\max \mathrm{Tr}\{\mathbf{A}^T\mathbf{K}_x\mathbf{Y}\mathbf{V}\}$ s. t. $\mathbf{A}^T\mathbf{K}_x\mathbf{A} = \mathbf{I}$ $\mathbf{V}^T\mathbf{V} = \mathbf{I}$	$\max \mathrm{Tr}\{\mathbf{A}^T\mathbf{K}_x\mathbf{Y}\mathbf{V}\}$ s. t. $\mathbf{A}^T\mathbf{K}_x\mathbf{A} = \mathbf{I}$ $\mathbf{v}_i^T\mathbf{v}_i = 1$
GEP / SVD	$\mathrm{GEP}[\mathbf{K}_x\mathbf{K}_Y\mathbf{K}_x, \mathbf{K}_x\mathbf{K}_x]$	$\mathrm{SVD}[\mathbf{K}_x\mathbf{Y}]$	$\mathrm{SVD}[\mathbf{K}_x\mathbf{Y}]$
Deflation	$\mathbf{K}_x\mathbf{K}_Y\mathbf{K}_x = \mathbf{K}_x\mathbf{K}_Y\mathbf{K}_x - \mathbf{K}_x\mathbf{K}_x\boldsymbol{\alpha}\boldsymbol{\alpha}^T\mathbf{K}_x\mathbf{K}_x$	$\mathbf{Y} = \mathbf{Y}[\mathbf{I} - \mathbf{v}\mathbf{v}^T]$	$\mathbf{K}_x = \mathbf{P}\mathbf{K}_x\mathbf{P}$ $\mathbf{P} = \mathbf{I} - \frac{\mathbf{K}_x\boldsymbol{\alpha}\boldsymbol{\alpha}^T\mathbf{K}_x}{\boldsymbol{\alpha}^T\mathbf{K}_x\mathbf{K}_x\boldsymbol{\alpha}}$
Allows sparse	YES	YES	NO
Max n_p	$\min\{\mathrm{rank}(\mathbf{K}_x), \mathrm{rank}(\mathbf{Y})\}$	$\min\{\mathrm{rank}(\mathbf{K}_x), \mathrm{rank}(\mathbf{Y})\}$	$\mathrm{rank}(\mathbf{K}_x)$

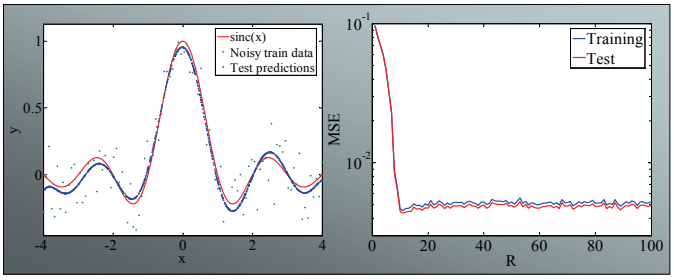
[1] R. Rosipal and L. J. Trejo, "Kernel partial least squares regression in reproducing kernel hilbert spaces," *JMLR*, 2001
[2] J. Shawe-Taylor and N. Christianini, *Kernel Methods for Pattern Analysis*, 2004

SPARSE KERNEL ORTHONORMALIZED PLS FOR FEATURE EXTRACTION IN LARGE DATA SETS

4 An illustrative example: sinc regression

- Training data: 100 points in [-4,4]; $\mathbf{y}_i = \text{sinc}(\mathbf{x}_i) + \text{noise}$
- Validation data: 1000 points in [-4,4]; noise free targets
- Gaussian kernel with $\sigma = \sqrt{1/2}$
- 200 independent experiments for $R = 1{:}100$

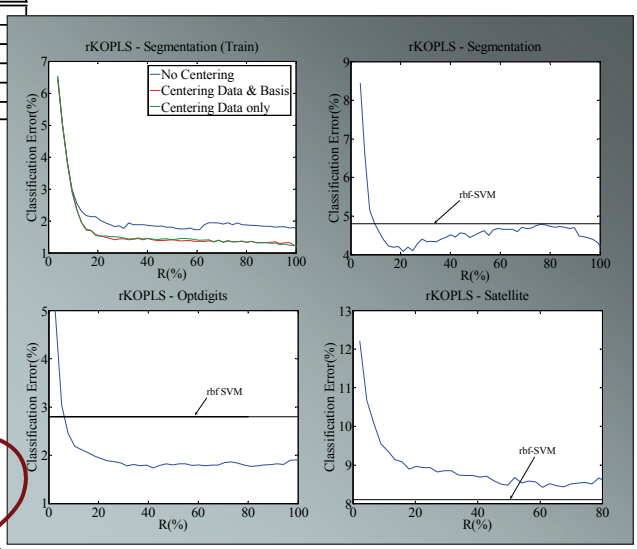
- Only 1 projection for rKOPLS
- Optimal performance for $R = 10$
- Overfitting for $R > 10$ is almost negligible



5 UCI Benchmark data sets

	# Train/Test	# Clases
vehicle	500 / 346	4
segmentation	1310 / 1000	7
optdigits	3823 / 1797	10
satellite	4435 / 2000	6
pendigits	7494 / 3498	10
letter	10000 / 10000	26

- Goal: Analyze the discriminative power of rKOPLS features in a benchmark of UCI multi-class classification problems.
- We compare with KPLS2, both without and with subsampling (for same training compupational cost)
- Gaussian kernel; width parameter selected using 10-CV
- We use two different classification schemes: 1) Linear regression + w.t.a.; 2) Linear ν- SVM



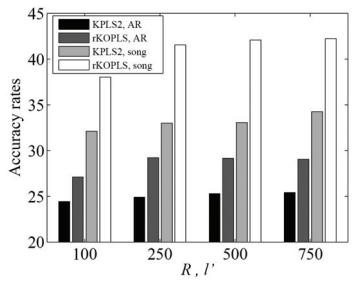
- Classifiers relying on rKOPLS generally achieve higher accuracies, even when no subsampling is used for KPLS2
- KPLS2 requires many more projections than rKOPLS
- Competitive performance when compared to rbf-SVM

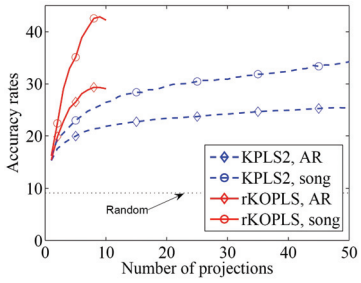
	rKOPLS - pseudo+w.t.a.			KPLS2 - pseudo + w.t.a			
	$R = 250$	$R = 500$	$R = 1000$	$l' = \sqrt{250\ l}$	$l' = \sqrt{500\ l}$	$l' = \sqrt{1000\ l}$	$l' = l$
vehicle	80.4 ± 1.2	79.9	—	$\mathbf{81.3} \pm 1.3$	80.5	—	80.5
segmentation	$\mathbf{95.7} \pm 0.4$	95.5 ± 0.3	—	93.9 ± 0.5	94.2 ± 0.5	—	95.1
optdigits	97.4 ± 0.2	97.7 ± 0.1	$\mathbf{98.2} \pm 0.2$	96.5 ± 0.3	97 ± 0.3	97 ± 0.2	97.6
satellite	89.8 ± 0.2	90.6 ± 0.3	91 ± 0.2	89.7 ± 0.4	90.3 ± 0.6	91.1 ± 0.3	$\mathbf{91.8}$
pendigits	97.6 ± 0.1	$\mathbf{98.2} \pm 0.1$	98.1 ± 0.2	97.4 ± 0.2	97.6 ± 0.1	97.7 ± 0.2	96.9
letter	84.8 ± 0.3	90 ± 0.2	$\mathbf{92.9} \pm 0.4$	84 ± 0.6	86 ± 0.6	86.2 ± 0.4	—
	rKOPLS - SVM			KPLS2 - SVM			rbf-SVM
vehicle	$\mathbf{81.2} \pm 1$	80.3	—	$\mathbf{81.2} \pm 1.1$	80.6	—	83
segmentation	95.1 ± 2	95.4 ± 0.4	—	$\mathbf{95.6} \pm 0.5$	94.8 ± 0.3	—	95.2
optdigits	97.3 ± 0.2	97.6 ± 0.1	$\mathbf{98.2} \pm 0.2$	96.4 ± 0.2	96.9 ± 0.2	96.9 ± 0.3	97.2
satellite	89.6 ± 0.6	90.5 ± 0.4	$\mathbf{91} \pm 0.2$	89.7 ± 0.5	90.4 ± 0.6	90.8 ± 0.5	91.9
pendigits	97.6 ± 0.2	$\mathbf{98.2} \pm 0.1$	98.1 ± 0.2	96.9 ± 0.1	97.1 ± 0.2	97.3 ± 0.2	98.1
letter	88.8 ± 1.5	92.1 ± 0.2	$\mathbf{93.9} \pm 0.3$	85.8 ± 0.5	85.9 ± 1.1	87.7 ± 1.2	96.2

Further issues

- Linear SVM is able to better exploit the projections, improving the performance in *vehicle* and *letter*
- Overfitting for increasing R is not very serious, if it occurs at all
- There is no need to center the centroids ($\mathbf{\Phi}_R$) in feature space

6 Feature Extraction for Music Genre Classification





- Objective: Predict musical genre from the audio stream
- Input features: AR models of MFCC coefficients (1.2 sg)
  Each AR model is summarized in a 135 length vector
- Training data: 57,388 AR vectors, approx. evenly distributed
  Test data: 36,556 AR vectors corresponding to 500 songs
- Most kernel MVA methods cannot handle such a training set
- Gaussian kernel; width parameter selected with CV
- Classifier: SLP + softmax network

- rKOPLS significantly outperforms KPLS2 with only 10 proj.
- Accuracy does not increase significantly for $R > 500$
- This system is running on-line inside a plug-in for winamp

rKOPLS provides expressive power + controlled CPU cost