1. Orthonormalized PLS (OPLS)
- Training data: \( \Phi = [\phi(x_1), \ldots, \phi(x_i)]^T; Y = [y_1, \ldots, y_i]^T \)
- Find projection vectors for feature extraction:
  \[ \tilde{\Phi} = \Phi U \]
  where each column in \( U \) is a projection vector, and \( \tilde{\Phi} \) is a centered version of \( \Phi \)
- OPLS:
  \[ \begin{align*}
  \text{maximize:} & & \text{Tr}\{U^T \tilde{\Phi}^T \tilde{\Phi} Y \tilde{U} \} \\
  \text{subject to:} & & U^T \Phi \Phi^T U = I
  \end{align*} \]
- 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.,
    \[ U = \arg \min_U \| Y - \tilde{\Phi} B \|^2_F, \quad B = (\tilde{\Phi}^T \tilde{\Phi})^{-1} \tilde{\Phi}^T Y \]

2. Kernel OPLS (KOPLS)
- Representer Theorem: \( U = \tilde{\Phi}^T A \)
- KOPLS:
  \[ \begin{align*}
  \text{maximize:} & & \text{Tr}\{A^T K_x K_y X A\} \\
  \text{subject to:} & & A^T K_x K_y A = I
  \end{align*} \]
- Matrix \( A \) is the solution of a gen. eigenvalue problem
  \[ K_x K_y K_x A = K_x K_y AA \]
  with \( K_x = \tilde{\Phi} \tilde{\Phi}^T \) and \( K_y = Y Y^T \)

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 with a well-known cost function

Cons
- Kernel computations: \( I^2 \)
- Memory requirements: \( 2 l^2 \)
- Dense solution: \( l \) nodes
- \( K_x, K_y \) is typically singular
- Solution may overfit

3. Sparse approximation of the KOPLS solution (rKOPLS)
- We use \( U = \Phi_R^T B \), where \( \Phi_R \) is a subset of the train dataset including \( R \) patterns \((R < l)\)
- This leads to the generalized eigenvalue problem
  \[ K_R K_y K_R^T B = K_R K_R^T B A \]
  where \( K_R = \Phi_R \Phi_R^T \) is a reduced \((R \times l)\) kernel matrix

Now:
- Kernel computations: \( R \cdot l \)
- Memory requirements: \( R(R + \text{dim}(Y)) \)
- Sparse solution with \( R \) nodes
- \( K_R K_R^T \) is better conditioned than \( K_x K_y \)
- \( R \) acts as a sort of regularizer

But:
- Random selection of centroids \( \Phi_R \)
- Additional parameter \( R \): subject to a tradeoff performance vs complexity

Iterative rKOPLS algorithm
1. Compute matrices \( K_R K_R^T \) and \( K_R Y \) using
   \[ K_R K_R^T = \sum k_i k_i^T \quad K_R Y = \sum k_i y_i^T \]
2. Repeat until desired or maximum number of projections:
   - Find largest gen. eigenvalue and eigenvector pair: \( \{ \lambda_i, \beta_i \} \)
   - Normalize eigenvector so that \( \beta_i^T K_R K_R^T \beta_i = 1 \)
   - Deflate \( K_R K_R^T \) according to
     \[ K_{R'} K_{R'}^T - \lambda_i K_{R'} \beta_i \beta_i^T K_{R'} \]
     (equivalent to \( K_y = K_y - \lambda_i K_{R'} \beta_i \beta_i^T K_{R'} \))

Comparison with other MVA methods

<table>
<thead>
<tr>
<th>MVA Methods</th>
<th>KOPLS</th>
<th>KPLS</th>
<th>KPLS2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max. Problem</td>
<td>max Tr{A^T K_x K_y A</td>
<td>A = I}</td>
<td>max Tr{A^T K_y Y</td>
</tr>
<tr>
<td>GEP / SVD</td>
<td>GEP(K, K_y, K_x)</td>
<td>SVD(K_y)</td>
<td>SVD(K_y)</td>
</tr>
<tr>
<td>Deflation</td>
<td>K_y K_x - K_y K_x^T K_y</td>
<td>Y = Y(I - v v^T)</td>
<td>K_y = P K_y P^T</td>
</tr>
<tr>
<td>Allows sparse</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>Max n ( n )</td>
<td>min {rank(K_y), rank(Y)}</td>
<td>min {rank(K_y), rank(Y)}</td>
<td>rank(K_y)</td>
</tr>
</tbody>
</table>

An illustrative example: sinc regression

- Training data: 100 points in [-4,4]; \( y = \text{sinc}(x) + \text{noise} \)
- 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

UCI Benchmark data sets

- 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 computational cost)
- Gaussian kernel; width parameter selected using 10-CV
- We use two different classification schemes:
  1) Linear regression + w.t.a.;
  2) Linear \( \nu \)-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

### 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 \( \Phi_R \) in feature space

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