An Optimization Perspective on Kernel Partial Least Squares Regression

K. P. Bennett and M. J. Embrechts[*]
Mathematical Sciences Dept.
Decision Science and Engineering Systems Dept.
Rensselaer Polytechnic Institute
Troy, NY 12180

Abstract. This work provides a novel derivation based on optimization for the partial least squares (PLS) algorithm for linear regression and the kernel partial least squares (K-PLS) algorithm for nonlinear regression. This derivation makes the PLS algorithm, popularly and successfully used for chemometrics applications, more accessible to machine learning researchers. The work introduces Direct K-PLS, a novel way to kernelize PLS based on direct factorization of the kernel matrix. Computational results and discussion illustrate the relative merits of K-PLS and Direct K-PLS versus closely related kernel methods such as support vector machines and kernel ridge regression.

[*] This work was supported by NSF grant number IIS-9979860. Many thanks to Roman Rosipal, Nello Cristianini, and Johan Suykens for many helpful discussions on PLS and kernel methods, Sean Ekans from Concurrent Pharmaceutical for providing molecule descriptions for the Albumin data set, Curt Breneman and N. Sukumar for generating descriptors for the Albumin data, and Tony Van Gestel for an efficient Gaussian kernel implementation algorithm. This work appears in J.A.K. Suykens, G. Horvath, S. Basu, C. Micchelli, J. Vandewalle (Eds.) *Advances in Learning Theory: Methods, Models and Applications,* NATO Science Series III: Computer & Systems Sciences, Volume 190, IOS Press Amsterdam, 2003,p. 227-250.

1 Introduction

Partial Least Squares (PLS) has proven to be a popular and effective approach to problems in chemometrics such as predicting the bioactivity of molecules in order to facilitate discovery of novel pharmaceuticals. PLS-type algorithms work very well. The algorithms are very resistant to overfitting, fast, easy to implement and simple to tune. Chemometric problems frequently have training data with few points and very high dimensionality. On this type of problem, simple linear least squares regression fails, but linear PLS excels. This ability to do inference in high-dimensional space effectively makes PLS an ideal candidate for a kernel approach. Rosipal and Trejo extended PLS to nonlinear regression using kernels functions [22]. As demonstrated in this chapter, kernel partial least squares (K-PLS) is a very effective general purpose regression approach.

The goal of this work is to make PLS and K-PLS more accessible to machine learning researchers in order to promote the use and extension of this powerful approach. The first step is to understand where PLS comes from and how it works. Thus, we develop a relatively simple yet rigorous derivation of PLS and two K-PLS variants from an optimization perspective. Many published papers contain PLS algorithms and analysis of these algorithms. But, typically no derivation of the algorithm is provided. The second step is to understand the strengths and weaknesses of PLS and K-PLS. So we provide a computational comparison with other kernel regression methods and discuss the relative merits of the approaches and directions for future work. The third step is to get researchers to try K-PLS. The K-PLS code is publicly available as part of the Analyze/StripMiner software at www.drugmining.com.

For simplicity, we focus on the simple regression problem of predicting a single response variable and derive the version of PLS that has been previously kernelized. But the analysis performed here can be easily altered to produce other PLS and K-PLS variations both existing and new. To demonstrate this, we derive the novel DK-PLS algorithm. The mathematical models underlying PLS are closely related to those of principal component analysis (PCA) regression. Thus in Section 2.1, we begin with a brief review of PCA. PCA computes components based on input data but does not take into account the response. In Section 2.2, we show how PLS can be derived from a logical extension of PCA to include information about the response. In Section 3, we investigate two possible methods for constructing a nonlinear PLS algorithm via kernels. In Section 4 we discuss practical aspects of a successful K-PLS implementation. In Sections 5-7, we provide a computational comparison between K-PLS and other kernel regression algorithms such as support vector machines (SVM) and kernel ridge regression (KRR). We conclude with a discussion of the relative merits of the PLS and K-PLS approaches, and future directions for research.

We limit our discussion to regression problems with a single dependent variable. More specifically, given a training set of data $((\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_\ell, \mathbf{y}_\ell))$ $\mathbf{x}_i \in R^{n \times 1}$, $\mathbf{y}_i \in R$, the problem is to construct some function f such that $f(x_i)$ approximately equals $\mathbf{y}_i$ and the function generalizes well on future data. Note PLS and K-PLS are also applicable to multiple regression problems with $\mathbf{y}_i \in R^c$, $c > 1$. The following notation is used. Each data point, $\mathbf{x}_i$, is represented as the i^{th} row in the data matrix $\mathbf{X}$. The i^{th} response is denoted by $\mathbf{y}_i$ where $\mathbf{y}$ is a column vector. Let ℓ denote the number of points, and $\mathbf{n}$ denote the dimensionality of the data, so $\mathbf{X} \in R^{\ell \times \mathbf{n}}$ and $\mathbf{y} \in R^{\ell \times 1}$. In general, matrices are denoted by bold capital letters such as $\mathbf{A}$. Vectors are denoted by bold lower case letters like $\mathbf{u}$. With the exception of the data points $\mathbf{x}_i$, vectors are presumed to be column vectors unless otherwise noted. The 2-norm or Euclidean norm is denoted by $||\mathbf{y}||$ if $\mathbf{y}$ is a vector, thus $||\mathbf{y}||^2 = \mathbf{y}'\mathbf{y}$. For a matrix $\mathbf{A}$, $||\mathbf{A}||^2$

denotes the square of the Frobenius norm which equals $\sum_i \sum_j (\mathbf{A}_{ij})^2$. Greek letters are used to denote scalars. The transpose of a matrix is denoted by $\mathbf{X}'$. The dot product of two column vectors $\mathbf{u}$ and $\mathbf{v}$ is denoted by $\mathbf{u}'\mathbf{v}$. The outer product of $\mathbf{u}$ and $\mathbf{v}$ is denoted by $\mathbf{u}\mathbf{v}'$. The reader should be careful to distinguish the use of dot products from that of outer products. Note that we assume a data vector $\mathbf{x}$ is a row vector. Thus $\mathbf{x}\mathbf{w}$ denotes the inner product of $\mathbf{x}$ and $\mathbf{w}$. Subscripts are used to indicate components of a matrix or a vector. Superscripts indicate values of a matrix or vector at each iteration.

2 PLS Derivation

As the name implies, PLS is based on least-squares regression. Consider the problem of constructing a linear function consisting of the inner product of $\mathbf{x}$ with $\mathbf{w}$, $f(\mathbf{x}) = \mathbf{x}\mathbf{w}$, such that $f(\mathbf{x})$ is approximately $\mathbf{y}$. Note that here $\mathbf{x}$ is a row vector and $\mathbf{w}$ is a column vector. We assume that $\mathbf{y}$ and the data matrix $\mathbf{X}$ have been scaled to have zero column means so that no bias term is necessary. A least squares optimization method constructs $\mathbf{w}$ by minimizing the sum of the squared residuals between the predicted and actual response. The simplest least squares problem is:

$$\min_{\mathbf{w}} \sum_{i=1}^{\ell} \frac{1}{2}(\mathbf{x}_i\mathbf{w} - \mathbf{y}_i)^2. \tag{1}$$

where $\mathbf{x}_i$ is a row vector representing the i^{th} data point. For high-dimensional data, the problem must be regularized to prevent overfitting. In ridge regression [12] this is accomplished by by penalizing large values of $||\mathbf{w}||^2$ to yield:

$$\min_{\mathbf{w}} \sum_{i=1}^{\ell} \frac{1}{2}(\mathbf{x}_i\mathbf{w} - \mathbf{y}_i)^2 \; + \; \frac{\lambda}{2}||\mathbf{w}||^2. \tag{2}$$

PLS's method of regularization or capacity control distinguishes it from other least squares algorithms. In Principal Component Analysis (PCA) regression, the capacity control is performed by mapping the data to a lower-dimensional space and constructing the least squares estimate in that space. PCA ignores all information about $\mathbf{y}$ while constructing this mapping. PLS utilizes information from the response $\mathbf{y}$ to help select a mapping better suited for regression problems. Since PLS is very similar to PCA, we begin with a review of PCA in the next section and then show how this type of analysis can be extended to PLS in the subsequent section.

2.1 PCA Regression Review

PCA regression consists of two steps. First a linear projection or mapping of the data is constructed, using standard PCA. Then the final regression function is constructed by minimizing the least squares error between the projected data and the response $\mathbf{y}$. Selecting a lower dimensional subspace for the mapping restricts the set of possible regression functions, thus limiting the capacity of the resulting function to overfit the data. This is a form of regularization.

PCA constructs a linear projection of the data that preserves the characteristics of the data as much as possible. The first component of PCA computes the linear projection of the data with maximum variance. But there are alternative derivations of PCA. The first principal component can be derived by minimizing the distance between a data point and its projection on a vector. These are equivalent problems.

If we take a data point $\mathbf{x}_i \in R^{1\times n}$ and project it on the vector $\mathbf{w}' \in R^{1\times n}$, the projected point is $(\mathbf{x}_i\mathbf{w})\mathbf{w}'$. To compute the linear projection that minimizes the distortion between $\mathbf{x}$ and it's projection, one can minimize

$$\min_{\mathbf{w}} \sum_{i=1}^{\ell} ||\mathbf{x}_i - \mathbf{x}_i\mathbf{w}\mathbf{w}'||^2 \; s.t. \; \mathbf{w}'\mathbf{w} = 1. \tag{3}$$

Note that $\mathbf{x}_i$ is a row vector, $\mathbf{w}$ is a column vector, and $\mathbf{w}\mathbf{w}'$ is an outer product. By simple linear algebra, Problem (3) is equivalent to maximizing the variance of the projected data, i.e.

$$\max_{\mathbf{w}} var(\mathbf{X}\mathbf{w}) \; s.t. \; \mathbf{w}'\mathbf{w} = 1. \tag{4}$$

Using the constraint to simplify the objective, problem (3) can be rewritten as

$$\begin{array}{ll} \min_{\mathbf{w}} & \sum_i [(\mathbf{x}_i - \mathbf{x}_i\mathbf{w}\mathbf{w}')(\mathbf{x}_i' - \mathbf{w}\mathbf{w}'\mathbf{x}_i')] = \sum_i (\mathbf{x}_i\mathbf{x}_i' - \mathbf{x}_i\mathbf{w}\mathbf{w}'\mathbf{x}_i') \\ s.t. & \mathbf{w}'\mathbf{w} = 1. \end{array} \tag{5}$$

After dropping the constant terms and converting the problem to a maximization problem, the problem becomes:

$$min_{\mathbf{w}} - \sum_i \mathbf{x}_i\mathbf{w}\mathbf{w}'\mathbf{x}_i' \; s.t. \; \mathbf{w}'\mathbf{w} = 1 \tag{6}$$

or equivalently

$$max_{\mathbf{w}} \sum_i ||\mathbf{x}_i\mathbf{w}||^2 \; s.t. \; \mathbf{w}'\mathbf{w} = 1. \tag{7}$$

Assuming that $\mathbf{x}_i$ has mean $\mathbf{0}$, this problem is equivalent to Problem (4).

The optimal solution for $\mathbf{w}$ can be easily constructed using the first order optimality conditions [18]. To derive the optimality conditions for Problem (7), convert the problem to a minimization problem, construct the Lagrangian function, and set the derivative of the Lagrangian with respect to $\mathbf{w}$ to zero. With λ as the Lagrangian multiplier of the constraint, the Lagrangian function is

$$L(\mathbf{w}, \lambda) = -\sum_i ||\mathbf{x}_i\mathbf{w}||^2 + \lambda(\mathbf{w}'\mathbf{w} - 1).$$

The optimality conditions are

$$\mathbf{X}'\mathbf{X}\mathbf{w} = \lambda\mathbf{w} \qquad \mathbf{w}'\mathbf{w} = 1. \tag{8}$$

So we know the optimal $\mathbf{w}$ is just the maximal eigenvalue of the covariance matrix $\mathbf{X}'\mathbf{X}$.

But this yields only a one-dimensional representation of the data. A series of orthogonal projections can be computed by the NIPALS (Nonlinear Iterative Partial Least Squares) algorithm [34]. The data matrix is reduced in order to account for the part of the data explained by $\mathbf{w}$. The data matrix is "deflated" by subtracting away the part explained by $\mathbf{w}$. So at the next iteration the method computes the best linear projection $\mathbf{w}^2$ of $[\mathbf{X}^1 - \mathbf{X}^1\mathbf{w}^1\mathbf{w}^{1\prime}]$ which exactly corresponds to the eigenvector with second largest eigenvalue of $\mathbf{X}'\mathbf{X}$. If $\hat{M} < n$

orthogonal projects are desired, this procedure is repeated $\hat{M}$ times. If the projection vectors $\mathbf{w}^i$ are represented as columns in the matrix $\mathbf{W}$, the reduced dimensionality data or latent variables are now $\mathbf{XW}$. We call each column of $\mathbf{XW}$ a latent variable. The matrix $\mathbf{X}$ has now been approximated by the low-rank matrix $\mathbf{XWW}'$. By construction $\mathbf{W}'\mathbf{W} = I$.

In PCA regression, the least-squares loss function is used to compute the final regression function. The least-squares problem in the projected space:

$$\min_{\mathbf{v} \in R^{\hat{M}}} \frac{1}{2} ||\mathbf{XWv} - \mathbf{y}||^2 \tag{9}$$

has optimality conditions

$$\mathbf{W}'\mathbf{X}'\mathbf{XWv} - \mathbf{W}'\mathbf{Xy} = 0 \tag{10}$$

The optimal value of the regression coefficients in the projected space is $\bar{\mathbf{v}}$. In the original space, the coefficients of the final linear function, $f(\mathbf{x}) = \mathbf{xb}$, are

$$\mathbf{b} = \mathbf{W}\bar{\mathbf{v}} = \mathbf{W}(\mathbf{W}'\mathbf{X}'\mathbf{XW})^{-1}\mathbf{W}'\mathbf{Xy}. \tag{11}$$

The final regression function is

$$f(\mathbf{x}) = \mathbf{xb} = \mathbf{xW}(\mathbf{W}'\mathbf{X}'\mathbf{XW})^{-1}\mathbf{W}'\mathbf{Xy} \tag{12}$$

assuming $\mathbf{x}$ is a row vector. The final regression function looks complicated as an equation, but conceptually it is fairly simple.

In PCA regression, the degree of capacity control or regularization is controlled by $\hat{M}$, the sole parameter in PCA. $\hat{M}$ controls the number of principal components $\mathbf{w}$, or equivalently the number of latent variables $\mathbf{xw}$. If $\hat{M} = n$, then the method reduces to simple least squares.

2.2 PLS Analysis

Like PCA, PLS also constructs a mapping of the data to a $\hat{M} \le n$ dimensional space and then solves a least squares regression problem in the $\hat{M}$ dimensional space to calculate the final regression function. Like PCA, this mapping is achieved by constructing a low-rank approximation of the data matrix $\mathbf{X}$. If $\hat{M} = n$, PLS also becomes simple least-squares regression. PLS methods differ from PCA in only two respects: the mathematical model for computing the mapping and the mathematical model used to compute the final regression coefficients. Unlike PCA, PLS utilizes both the input and the response data, $\mathbf{X}$ and $\mathbf{y}$ respectively, to construct the mapping to a lower dimensional space. In this process it constructs low-rank approximations for both $\mathbf{X}$ and $\mathbf{y}$. The final optimization model optimization utilizes these approximations plus the linear projections to compute the final regression coefficients.

2.3 Linear PLS

The entire PLS procedure can be derived by making two simple but profound changes to the optimization problems underlying PCA Regression. The primary problem with PCA Regression, is that PCA does not take into account the response variable when constructing the principal components or latent variables. Thus even for easy classification problems such as

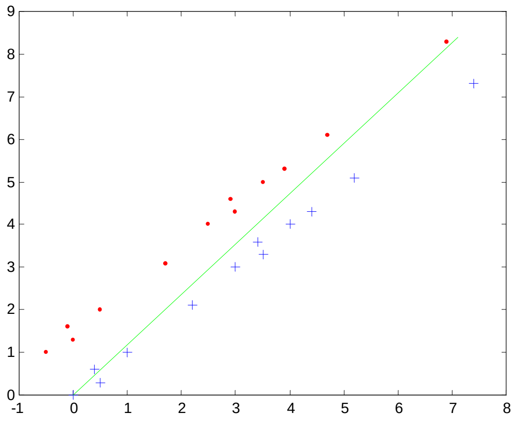

Figure 1: Line is direction of maximum variance (w) constructed by PCA.

in Figure 2.3, the method may select poor latent variables. Thus, the first change is to incorporate information about the response in the model used to construct the latent variables. To simplify notation, we switch to matrix notation and write the PCA Problem (4) as

$$\min_{\mathbf{w}} ||\mathbf{X} - \mathbf{X}\mathbf{w}\mathbf{w}'||^2 s.t. \mathbf{w}'\mathbf{w} = 1 \tag{13}$$

where $||\mathbf{X} - \mathbf{X}\mathbf{w}\mathbf{w}'||^2 = \sum_i \sum_i (\mathbf{x}_{ij} - \mathbf{x}_i\mathbf{w}\mathbf{w}_j)^2$. If $\mathbf{X}\mathbf{w}$ is approximately $\mathbf{y}$, the regression problem is solvable using only one latent variable. So we simply substitute $\mathbf{y}$ for $\mathbf{X}\mathbf{w}$ into the PCA Problem (7) to yield

$$\min_{\mathbf{w}} ||\mathbf{X} - \mathbf{y}\mathbf{w}'||^2 \ s.t. \ \mathbf{w}'\mathbf{w} = 1. \tag{14}$$

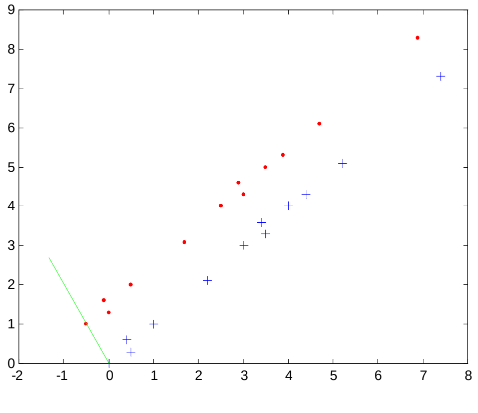

Figure 2: Line is direction w constructed by PLS using response (1 or -1) information.

Does this problem make sense? Figure 2.3 illustrates the resulting PLS direction on the above sample problem. Now the regression function (here for a classification problem) can be constructed using a single latent variable. We know Problem (14) constructs the rank-one function of y that mostly nearly maps to the input data x. How does this relates to the regression problem of finding a function of the inputs x that maps to the response y? Using

$||\mathbf{w}|| = 1$ and the Cauchy-Schwartz Theorem, we can show that $||\mathbf{X}-\mathbf{y}\mathbf{w}'||^2$ bounds the usual least-squares regression loss function. More precisely for each point training point $(\mathbf{x}_i, \mathbf{y}_i)$:

$$||\mathbf{x}_i\mathbf{w} - \mathbf{y}_i||^2 = ||(\mathbf{x}_i - \mathbf{y}_i\mathbf{w}')\mathbf{w}||^2 \leq ||\mathbf{x}_i - \mathbf{y}_i\mathbf{w}'||^2||\mathbf{w}||^2 = ||\mathbf{x}_i - \mathbf{y}_i\mathbf{w}'||^2.$$

Thus Problem (14) minimizes a bound on the least squares loss function but the choice of $\mathbf{w}$ is now greatly restricted.

To our knowledge, Problem (14) has not been used before to derive PLS in the literature. It is well known that in PLS the first latent variable $\mathbf{X}\mathbf{w}$ maximizes the sample covariance of $\mathbf{X}\mathbf{w}$ and $\mathbf{y}$. Just as in PCA, the covariance and low rank approximation problems can be shown to be equivalent. Note that this derivation depends on the assumptions that $\mathbf{y}'\mathbf{y} = \mathbf{w}'\mathbf{w} = 1$, $mean(\mathbf{X}) = \mathbf{0}$, and $mean(\mathbf{y}) = 0$. For any data point $\mathbf{x}$,

$$\begin{aligned} ||\mathbf{x} - \mathbf{y}\mathbf{w}'||^2 &= (\mathbf{x} - \mathbf{w}\mathbf{y}')(\mathbf{x} - \mathbf{y}\mathbf{w}') = \mathbf{x}'\mathbf{x} - 2\mathbf{w}\mathbf{x}'\mathbf{y} + \mathbf{y}\mathbf{w}'\mathbf{w}\mathbf{y}' \\ &= \mathbf{x}'\mathbf{x} - 2\mathbf{w}\mathbf{x}'\mathbf{y} + \mathbf{y}\mathbf{y}' = -2cov(\mathbf{x}\mathbf{w}, y) + constant. \end{aligned}$$

Thus after converting the problem to a maximization problem and removing constant terms, Problem (14) is equivalent to

$$\max_{\mathbf{w}} cov(\mathbf{X}\mathbf{w}, \mathbf{y}) \; s.t. \; \mathbf{w}'\mathbf{w} = 1. \tag{15}$$

A wonderful quality of these problems is that they have a closed form solution. The optimality conditions are $\mathbf{X}'\mathbf{y} = \lambda\mathbf{w}$ and $\mathbf{w}'\mathbf{w} = 1$, so the optimal solution is $\mathbf{w} = \frac{\mathbf{X}'\mathbf{y}}{(\mathbf{y}'\mathbf{X}\mathbf{X}'\mathbf{y})}$. The latent variable can be computed in linear time. It is easy to show that $\mathbf{w}$ is the eigenvector of $\mathbf{X}'\mathbf{y}\mathbf{y}'\mathbf{X}$.

Note that these problems are well defined even if the constraint $||\mathbf{w}|| = 1$ is dropped. We would like to derive a version of PLS that is convenient for kernel functions when $\mathbf{w}$ is in feature space. So we will not use the normalized form of $\mathbf{w}$. For the rest of the chapter we will use the unnormalized $\mathbf{w} = \mathbf{X}'\mathbf{y}$. We can now derive the rest of the PLS algorithm using the analogous approaches to PCA regression, but we must alter any step that assumes $\mathbf{W}'\mathbf{W} = I$.

As in PCA we want to compute a series of directions that provide good low-rank approximations of our data. We can now deflate our data matrix to take into account the explained data. Our current approximation of the original matrix ($\mathbf{X}^1$) is $\mathbf{X}^1\mathbf{w}^1\mathbf{w}^{1'}$ But we can make this approximation a bit better. Define $\mathbf{t}^1 = \frac{\mathbf{X}^1\mathbf{w}^1}{||\mathbf{X}^1\mathbf{w}^1||}$. The best approximation is the matrix $\mathbf{t}^1\mathbf{p}^{1'}$ where $\mathbf{p}^1$ solves

$$\min_{\mathbf{p}} ||\mathbf{X}^1 - \mathbf{t}^1\mathbf{p}'||^2. \tag{16}$$

By Lemma 1 in the appendix, $\mathbf{p}^1 = \mathbf{X}^{1'}\mathbf{t}^{1^i}$. Thus deflating $\mathbf{X}^1$ can be accomplished as follows:

$$\mathbf{X}^2 = \mathbf{X}^1 - \mathbf{t}^1\mathbf{p}^{1'} = \mathbf{X}^1 - \mathbf{t}^1\mathbf{t}^{1'}\mathbf{X}^1.$$

This process will generate a series of vectors $\mathbf{t}^i$ that are orthogonal. The matrix $\mathbf{T}$, created using $\mathbf{t}^i$ as the columns of $\mathbf{T}$, is orthogonal.

Recall that $\mathbf{X}^i\mathbf{w}^i$ is also an approximation of $\mathbf{y}$. So it seems reasonable to calculate the residual for the current estimate of $\mathbf{y}$ since that part of $\mathbf{y}$ is in some sense explained. Since $\mathbf{t}^1$ is proportional to $\mathbf{X}^i\mathbf{w}^i$, we calculate the best least-squares fit of $\mathbf{t}^1$ to $\mathbf{y}^1 = \mathbf{y}$. So our best estimate of $\mathbf{y}^1$ is $\mathbf{t}^1\mathbf{c}^{1'}$ such that $\mathbf{c}^1 \in R$ solves

$$\min_{\mathbf{c}} ||\mathbf{y}^1 - \mathbf{t}^1\mathbf{c}'||^2.$$

By Lemma 1, $\mathbf{c}^1 = \mathbf{t}^{1'}\mathbf{y}^1$. So the residual is just

$$\mathbf{y}^2 = \mathbf{y}^1 - \mathbf{t}^1\mathbf{c}^{1'} = \mathbf{y}^1 - \mathbf{t}^1\mathbf{t}^{1'}\mathbf{y}^1.$$

For the next latent variable, this process can be repeated using $\mathbf{X}^2$ and $\mathbf{y}^2$ to compute $\mathbf{w}^2$, $\mathbf{t}^2$ etc. until the desired number of latent variables, $\hat{M}$ is reached.

2.4 Final Regression Components

Once the latent variables have been constructed, the next step is to compute the final regression functions. PLS and PCA use different mathematical models to compute the final regression coefficients. PCA Regression maps the original data into a lower-dimensional space using the projection matrix $\mathbf{W}$ and then computes the least squares solution in this space. Recall that PLS computes an orthogonal factorization of both the input $\mathbf{X}$ and response $\mathbf{y}$ data in the process of computing $\mathbf{W}$. This factorization constructs low rank approximations of $\mathbf{X}$ and $\mathbf{y}$. *The least squares model for PLS is based on these approximations of the input and response data, not the original data.* This is the other key difference between K-PLS and PCA regression. The use of the approximate data instead of the actual training data contributes both to the robustness and computational efficiency of the approach.

Use of the data factorization makes computing the final regression coefficients more efficient. Let $\mathbf{T} \in R^{\ell \times K}$ denote the matrix formed by using each $\mathbf{t}^i$ as a column and similarly for $\mathbf{W}$. We know $\mathbf{T}$ was constructed to be a good factorization of both $\mathbf{X}$ and $\mathbf{y}$. By Lemma 1 in the Appendix and the fact that $\mathbf{T}'\mathbf{T} = I$, one can show $\mathbf{P} = \mathbf{X}'\mathbf{T}$ solves $\min_{\mathbf{P}} ||\mathbf{X} - \mathbf{T}\mathbf{P}'||^2$ and therefore $\mathbf{T}\mathbf{P}'$ is the best approximation of $\mathbf{X}$ based on $\mathbf{T}$. Similarly the best approximation of $\mathbf{y}$ is $\mathbf{T}\mathbf{C}'$ where $\mathbf{C} = \mathbf{y}'\mathbf{T}$ solves $\min_{\mathbf{C}} ||\mathbf{y} - \mathbf{T}\mathbf{C}'||^2$. The final regression problem is: construct $\mathbf{v}$ such that

$$\min_{\mathbf{v}} ||(\mathbf{T}\mathbf{P}')\mathbf{W}\mathbf{v} - \mathbf{T}\mathbf{C}'||^2 \quad = \quad \min_{\mathbf{v}} ||\mathbf{T}(\mathbf{P}'\mathbf{W}\mathbf{v} - \mathbf{C}')||^2 \tag{17}$$

Note this is identical to the problem solved in PCA Regression except the approximation of $\mathbf{X}$ and $\mathbf{y}$ are used instead of the actual data. If $(\mathbf{P}'\mathbf{W}\mathbf{v} = \mathbf{C}')$ has a solution $\bar{\mathbf{v}}$, then $\mathbf{v}$ is the optimal solution of Problem (17) since the objective will then be zero, the lowest possible value. It turns out that $\mathbf{P}'\mathbf{W}$ is always a lower triangular matrix (see [22, 13]) and thus nonsingular. Thus we know that $\bar{\mathbf{v}}$ exists satisfying $\mathbf{P}'\mathbf{W}\mathbf{v} = \mathbf{C}'$. The solution $\hat{\mathbf{v}}$ can be computed efficiently using forward substitution. For notational convenience we will say $\bar{\mathbf{v}} = (\mathbf{P}'\mathbf{W})^{-1}\mathbf{C}'$. But in fact the inverse matrix need not be calculated explicitly.

Having found $\bar{\mathbf{v}}$, the final regression function can be computed using the same approach in PCA Regression (see (11),(12)). The final regression functions is $f(\mathbf{x}) = \mathbf{x}\mathbf{b}$ where

$$\mathbf{b} = W(\mathbf{P}'\mathbf{W})^{-1}\mathbf{C}' = W(\mathbf{T}'\mathbf{X}\mathbf{W})^{-1})\mathbf{T}'\mathbf{y}. \tag{18}$$

By exploiting the facts that $\mathbf{C}' = \mathbf{T}'\mathbf{y}$ and $\mathbf{P}' = \mathbf{T}'\mathbf{X}$, $\mathbf{b}$ can be calculated solely in terms of $(\mathbf{X}, \mathbf{y}, \mathbf{T}, \mathbf{W})$.

To summarize, the final PLS algorithm for computing a single response variable is:

Algorithm 1. Basic PLS Algorithm *Assume data* $\mathbf{X}^1 = \mathbf{X}$ *and response* $\mathbf{y}^1 = \mathbf{y}$ *have been normalized by column to have mean* 0 *and standard deviation* 1. *The only algorithm parameter is the number of latent variables,* $\hat{M}$.

```
1. For $m = 1$ to $\hat{M}$
2.     $\mathbf{w}^m = \mathbf{X}^{m\prime}\mathbf{y}^m$
3.     $\mathbf{t}^m = \mathbf{X}^m\mathbf{w}^m$
4.     $\mathbf{t}^m = \mathbf{t}^m/||\mathbf{t}^m||$
5.     $\mathbf{t}^{m+1} = \mathbf{X}^m - \mathbf{t}^m\mathbf{t}^{m\prime}\mathbf{X}^m$
6.     $\mathbf{y}^{m+1} = \mathbf{y}^m - \mathbf{t}^m\mathbf{t}^{m\prime}\mathbf{y}^m$
7.     $\mathbf{y}^{m+1} = \mathbf{y}^{m+1}/||\mathbf{y}^{m+1}||$
8. Compute final regression coefficients $\mathbf{b}$
       $\mathbf{b} = \mathbf{W}(\mathbf{T}'\mathbf{X}\mathbf{W})^{-1})\mathbf{T}'\mathbf{y}.$
   where the $m^{th}$ columns of $\mathbf{W}$ and $\mathbf{T}$ are $\mathbf{w}^m$ and $\mathbf{t}^m$ respectively.
```

The final regression function is $f(\mathbf{x}) = \mathbf{x}\mathbf{b}$ where the data vector $\mathbf{x}$ is a row vector. PLS requires far less computational time than PCA because PLS computes the latent vector by simply multiplying the data matrix by the residual, e.g. $\mathbf{w}^m = \mathbf{X}^{m\prime}\mathbf{y}^m$. while PCA must compute the eigenvectors of $\mathbf{X}^{m\prime}\mathbf{X}^m$ at each iteration. Because the solution can be represented in terms of the data matrix and works very well on high-dimensional collinear data, PLS is a natural candidate for a kernel method.

3 Nonlinear PLS via Kernels

While other nonlinear extensions to PLS have previously been proposed [1, 35, 36, 2], PLS has only just recently been extended to nonlinear regression through the use of kernels. K-PLS exhibits the elegance that only linear algebra is required, once the kernel matrix has been determined. There are two general approaches for kernelizing PLS. The first approach by Rosipal and Trejo is based on the now classic methodology used in SVM [30, 22]. Each point is mapped nonlinearly to a higher dimensional feature space. A linear regression function is constructed in the mapped space corresponding to a nonlinear function in the original input space. In the dual space, the mapped data only appears as dot products and these dot products can be replaced by kernel functions in the final K-PLS algorithm. The second approach is to use PLS to factorize the kernel matrix directly. K-PLS as first introduced by Rosipal and Trejo [22] is derived using approaches analogous to those used for kernel PCA [23]. Direct Kernel PLS (DK-PLS), introduced here, is based on a direct factorization of the kernel matrix. DK-PLS explicitly produces a low rank approximation of the kernel matrix. Thus it is more closely related to other kernel matrix approximation approaches based on sampling or factorization [16, 10, 17, 9, 27, 24]. DK-PLS has the advantage that the kernel does not need to be square. When combined with sampling of the columns of the kernel matrix such as in [17, 16, 10], it is more scalable than the original KPLS.

3.1 Feature Space K-PLS

A full discussion of the derivation of K-PLS from PLS using the approach of mapping the data to feature space and constructing a linear function in feature space is given in [22]. To generate this approach one defines a mapping Φ and replaces $\mathbf{X}$ with $\Phi(\mathbf{X})$ and propagates the changes required in the algorithm. Thus the basic problem becomes:

$$\min_{\mathbf{w}} ||\Phi(\mathbf{X}) - \mathbf{y}\mathbf{w}'||^2 \quad s.t. ||\mathbf{w}||^2 = 1. \tag{19}$$

PLS Algorithm 1 can be easily kernelized. Using approaches such as for LS-SVM [26], the optimality conditions of this problem can be constructed in the dual space. This produces a formulation equivalent to [22] and the reader should consult that paper for more details. We just summarize here and provide the simplified algorithm for one response variable. In Algorithm 1, there is no need to explicitly calculate $\mathbf{W}$. Steps 2 and 3 can be combined. The final regression coefficient can be rewritten to exploit the fact that $\mathbf{w}^m = \mathbf{X}^m\mathbf{y}^m = \mathbf{X}\mathbf{y}^m$. In feature space, step 6 cannot be explicitly performed. But since $\mathbf{X}^m$ only appears in the expression $\mathbf{X}^m\mathbf{X}^{m\prime}$, the kernel matrix can be deflated directly. The K-PLS simplified for one response becomes:

Algorithm 2. Kernel Partial Least Squares [22]

Let $\mathbf{K}^0 = \Phi(\mathbf{X})\Phi(\mathbf{X})'$*, i.e.* $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$*, be the Gram matrix in feature space. Let* $\mathbf{K}^1$ *be the centered form of* $\mathbf{K}^0$*. Let response* $\mathbf{Y}^1 = \mathbf{y}$ *be normalized to have mean* 0 *and standard deviation* 1*. Let* $\hat{M}$ *be the desired number of latent variables.*

```
1. For $m = 1$ to $\hat{M}$
2.     $\mathbf{t}^m = \mathbf{K}^m\mathbf{y}^m$
3.     $\mathbf{t}^m = \mathbf{t}^m/||\mathbf{t}^m||$
4.     $\mathbf{K}^{m+1} = (I - \mathbf{t}^m\mathbf{t}^{m\prime})\mathbf{K}^k(I - \mathbf{t}^m\mathbf{t}^{m\prime})$
5.     $\mathbf{y}^{m+1} = \mathbf{Y}^m - \mathbf{t}^m\mathbf{t}^{m\prime}\mathbf{Y}^m$
6.     $\mathbf{y}^{m+1} = \mathbf{y}^{m+1}/||\mathbf{y}^{m+1}||$
7. Compute final regression coefficients $\alpha$
```

$$\alpha = \mathbf{Y}(\mathbf{T}'\mathbf{K}^{1\prime}\mathbf{Y})^{-1})\mathbf{T}'\mathbf{y}$$

where the m^{th} *columns of* $\mathbf{Y}$ *and* $\mathbf{T}$ *are* $\mathbf{y}^m$ *and* $\mathbf{t}^m$ *respectively.*

$$f(x) = \sum_{i=1}^{\ell} \hat{K}(\mathbf{x}, \mathbf{x}_i)\alpha_i$$

Note that the training and testing kernels must be centered. See equation 21.

3.2 Direct Kernel Partial Least Squares

Direct Kernel Partial Least Squares factorizes the kernel matrix directly and then computes the final regression function based on this factorization. Let $\mathbf{K} = \Phi(\mathbf{X})\Phi(\mathbf{X})'$, i.e. $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$, be the Gram matrix in feature space. Assume $\mathbf{K}$ and $\mathbf{y}$ have been centered so that no bias term is required. The underlying problem for DK-PLS is:

$$\min_{\alpha} ||\mathbf{K} - \mathbf{y}\alpha'||^2 \tag{20}$$

DK-PLS constructs a low rank approximation of the kernel matrix, and then uses this approximation to construct the final function. Strategies for improving kernel methods through factorization have been receiving increasing attention [29, 9]. DK-PLS not only computes a factorization very efficiently relative to eigenvector methods, but it produces a low-rank approximation biased for good performance on the regression task. Algorithm 1 is converted to DK-PLS by simply substituting $\mathbf{K}$ for $\mathbf{X}$. Any variant of PLS can be adapted using the direct kernel approach. The resulting algorithms may be more efficient especially if the kernel matrix is not square. DK-PLS does not assume that the kernel matrix is square so the data maybe sampled to construct the kernel matrix such as in RSVM [10]. When coupled with such a sampling strategy, DK-PLS is more scalable than K-PLS.

Algorithm 3. Direct Kernel Partial Least Squares *Let* $\mathbf{K}^0 = \Phi(\mathbf{X})\Phi(\mathbf{X})'$*, i.e.* $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$*, be the Gram matrix in feature space. Let* $\mathbf{K}^1$ *be the centered form of* $\mathbf{K}^0$*. Let response* $\mathbf{Y}^1 = \mathbf{y}$ *be normalized to have mean* 0 *and standard deviation* 1*. Let* $\hat{M}$ *be the desired number of latent variables.*

```
1. For $k = 1$ to $\hat{M}$
2.     $\mathbf{t}^m = \mathbf{K}^m\mathbf{K}^{m\prime}\mathbf{y}^m$
3.     $\mathbf{t}^m = \mathbf{t}^m/||\mathbf{t}^m||$
4.     $\mathbf{K}^{m+1} = \mathbf{K}^m - \mathbf{t}^m\mathbf{t}^{m\prime}\mathbf{K}^m$
5.     $\mathbf{y}^{m+1} = \mathbf{y}^m - \mathbf{t}^m\mathbf{t}^{m\prime}\mathbf{Y}^m$
6.     $\mathbf{y}^{m+1} = \mathbf{y}^{m+1}/||\mathbf{y}^{m+1}||$
7. Compute final regression coefficients $\alpha$
       $\alpha = \mathbf{K}^1\mathbf{Y}(\mathbf{T}'\mathbf{K}^1\mathbf{K}^{1\prime}\mathbf{Y})^{-1}\mathbf{T}'\mathbf{y}$
   where the $m^{th}$ columns of $\mathbf{Y}$ and $\mathbf{T}$ are $\mathbf{y}^m$ and $\mathbf{t}^m$ respectively.
       $f(x) = \sum_{i=1}^{\ell} \hat{K}(x_i, x)\alpha_i$
   Note that the testing kernel must be centered.
```

4 Computational Issues in K-PLS

K-PLS requires that the user specify the number of latent variables. Selecting a modest number of latent variables effectively constitutes regularization for K-PLS. The number of latent variables can be determined by either i) tuning on a validation set or ii) adhering to a policy. It has been our experience that for particular types of datasets the optimal number of latent variables does not change by much and the prediction performance is generally not extremely sensitive to the optimal choice for the number of latent variables. On chemometric data, we generally adopt the policy of using five latent variables, regardless of the dataset (but Mahalanobis scale the data first). For large datasets or data that are known to be highly nonlinear (e.g., twisted spiral), the optimal number of latent variables is estimated using a validation set.

As in Kernel PCA, centering the kernel is important to make K-PLS (and especially DK-PLS) work properly. The idea of kernel centering is to force the bias term to be zero. The important thing to remember when centering kernels is that the training set and the test set kernels should be centered in a consistent manner. Kernel centering can be implemented using the following formulas for centering the training kernel and test kernel as suggested by Wu et al [39, 23, 22]. It is important to note that the equation for the test kernel is based on the un-centered training kernel:

$$\begin{array}{rcl} K^{train}_{center} & = & (\mathbf{I} - \frac{1}{\ell}\mathbf{1}\mathbf{1}')K^{train}(\mathbf{I} - \frac{1}{\ell}\mathbf{1}\mathbf{1}') \\ K^{test}_{center} & = & (K^{test} - \frac{1}{\ell}\mathbf{1}\mathbf{1}'K^{train})(\mathbf{I} - \frac{1}{\ell}\mathbf{1}\mathbf{1}') \end{array} \tag{21}$$

where $\mathbf{1}$ is a vector of ones and I is an identity matrix of appropriate dimension. Above expressions for kernel centering are mathematically elegant and can be rapidly programmed in MATLAB, but are numerically inefficient. In our implementation we essentially adhere to the formulation above, but micro-encode for optimal performance.

A numerically equivalent but more efficient kernel centering proceeds as follows. For the centered training kernel subtract the average for each column of the training kernel from the column entries, and then subtract the average row value of the modified training kernel row value from each row entity. The training average column values can be kept in memory and used to center the test kernel in a similar way. For centering the test kernel, the average value of the columns of the training kernel is subtracted from each column entry in the test kernel, succeeded by subtracting the average row value from each row entry of the recently modified test kernel. Note that the kernel need not be square.

5 Comparison of Kernel Regression Methods

In this section, we compare several different types of kernel regression methods with kernel PLS.

5.1 Methods

Different methods considered in this benchmark study include: i) Linear Partial Least Squares [32, 33, 34, 37] (PLS); ii) Linear Proximal Support Vector Machines [10] (P-SVM Lin); iii) K-PLS algorithm as proposed by Rosipal and Trejo [22] with kernel centering (K-PLS); iv) direct kernel PLS (DK-PLS), which factorizes the centered kernel matrix directly as described above; v) LS-SVM also known as Kernel Ridge Regression [26, 7, 8] (LS-SVM) applied to the centered kernel; vi) the reduced form of Least-Squares Support Vector Machines [10] (LS-RSVM) applied to the centered kernel; and viii) classic SVM as implemented in SVM-Torch [30, 5]. The kernel was not centered for SVM-TORCH. More precisely, the LS-SVM solution is produced by solving the following set of equations (using notation in [3]):

$$(\mathbf{K} + \lambda \mathbf{I})\alpha = \mathbf{y} \tag{22}$$

to produce the following function

$$f(\mathbf{x}) = y'(\mathbf{K} + \lambda \mathbf{I})^{-1}\mathbf{k} \tag{23}$$

where $\mathbf{k}_i = K(x, x_i), i = 1, \ldots, m$. The LS-RSVM method is constructed by solving the following equations:

$$(\mathbf{K}'\mathbf{K}\alpha - \lambda \mathbf{I})\alpha = \mathbf{K}'\mathbf{y} \tag{24}$$

to produces the final regression functions:

$$f(\mathbf{x}) = y'\mathbf{K}(\mathbf{K}'\mathbf{K} + \lambda \mathbf{I})^{-1}\mathbf{k} \tag{25}$$

where $\mathbf{k}_i = K(x, x_i), i = 1, \ldots, m$. For all kernel methods except SVM-Torch, the kernel was centered. Note that LS-RSVM does not require the kernel matrix to be square to be well defined. All the computer coding was efficiently implemented in C in the Analyze/StripMiner code available from the DDASSL website [6]. The Least Squares SVM variants (LS-SVM, LS-RSVM and LS-RSVM lin) apply an extension of scaled conjugate gradient method introduced by Möller (in a very different context) [19] for fast equation solving. The scaled conjugate gradient method is effectively a Krylov method [14] and the computation time for solving a linear set of ℓ equations scales roughly as $50\ell^2$, rather than ℓ^3 for traditional equation solvers.

5.2 Benchmark Cases

The benchmark studies comprise four binary classification problems and three true regression cases. The classification cases were treated as if they were regression problems. The classification benchmark data sets were obtained from the UCI Machine Learning Repository [20] and are BUPA Liver, Ionosphere, Tic-Tac-Toe, and Mushroom. The regression cases include Boston Housing, Abalone, and Albumin. The Boston Housing data were obtained from the UCI data repository. The Abalone data, which relate to the prediction of the age for horseshoe crabs [21], were obtained from http://ssi.umh.ac.be/abalone.html. The Albumin dataset [4] is a public QSAR drug-design-related dataset and can be obtained from the DDASSL homepage [6]. The aim of the Albumin dataset is predicting the binding affinities of small molecules to the human serum albumin.

With an exception for the mushroom data, all the computations were performed on a 300 MHz Pentium II with 128 MB of memory. The calculations for the mushroom and abalone data were performed on a 1.8 GH Pentium IV. Because the compiler was not optimized for a Pentium IV Processor, the reported execution times for the mushroom and abalone data were recalibrated to the estimated run time on a 300 MHz Pentium II.

Method	Data Set	BUPA Liver	Ionosphere	Tic-Tac-Toe	Mushroom
	mxn	345x6	351x34	958x9	8124x22
PLS	Test (%)	30.0	12.7	31.1	12.3
(linear)	Time (s)	0.52	1.89	2.69	56.18
P-SVM	Test	30.0	13.7	31.5	12.4
(linear)	Times	0.05	4.26	1.37	37.6
K-PLS	Test	27.9	**4.2**	**0.1**	**0.0**
	Time	47.47	63.48	1044.24	$2.01*10^5$
DK-PLS	Test	28.1	5.5	6.8	3.75
	Time	40.51	54.62	799.31	$1.05*10^5$
LS-SVM	Test	29.0	5.5	3.9	0.04
	Time	77.00	54.40	590.4	$6.76 *10^4$
LS-RSVM	Test	**27.5**	4.3	9.5	0.11
	Time	211.55	300.08	4313	$1.12 *10^5$
SVM-Torch	Test	28.9	5.0	1.5	0.08
	Time	258.03	242.00	1161.81	$1.44*10^5$

Table 1: Binary Classification Benchmark Cases Average Misclassification Rate (100 x, leave 10% out mode)

5.3 Data Preparation and Parameter Tuning

All data were first Mahalanobis scaled before generating the kernel. The kernel function used in this study is a Gaussian kernel ($K(u, v) = \exp(-||u - v||^2/2\sigma^2)$). The value of σ for each dataset was tuned on a validation set before proceeding with the calculations. The λ parameter for penalizing the two-norm of the weights in all the least squares methods (LS-SVM, LS-RSVM, and LS-RSVM lin) were heuristically determined by the following relationship:

$$\lambda = 0.05 \left(\frac{\ell}{200}\right)^{\frac{3}{2}} \tag{26}$$

where ℓ is the number of training data. We found that this particular policy choice for λ gives near optimal performance for 40 different benchmark datasets that we have tested so far. SVM-Torch is executed in a regression mode and the C parameter is heuristically determined by the above formula, but now $C = 1/\lambda$. The ϵ parameter for SVM-Torch was tuned on a validation set. All cases were executed in a 100-times leave-10-percent-out mode (100xLOO10). The number of latent variables for PLS and the K-PLS methods was held fixed by policy to five for most cases, but tuned on a validation set if the optimal value was considered to be relevantly different. The results for the binary classification benchmark cases are summarized in Table 1, and the results for the regression cases are shown in Table 2. The best performance for prediction accuracy are indicated in bold. The results for both regression and classification are summarized in Table 3.

For the binary classification cases, the results are presented as the number of cases that were misclassified without providing any details on the false positive/false negative ratio. For the regression cases the results are presented by the least-mean square error and Q2 defined below. Q2 is the square of the correlation between the actual and predicted response. Since Q2 is independent on the scaling of the data [11], it is generally useful to compare the relative

Method	Data Set	Boston Housing	Albumin	Abalone
	mxn	506x13	94x551	4177x8
PLS	Test Q2	0.28	0.36	0.49
(linear)	LMSE	4.92	0.35	2.22
	Time(s)	1.35	10.73	17.75
P-SVM	Test Q2	0.28	0.42	0.49
(linear)	LMSE	4.88	0.4	2.22
	Times(s)	1.18	69.7	5.85
K-PLS	Test Q2	**0.13**	0.35	**0.44**
	LMSE	3.40	0.35	2.12
	Time(s)	181.05	34.24	26224
DK-PLS	Test Q2	0.18	**0.33**	0.45
	LMSE	3.9	0.34	2.12
	Time(s)	147.99	32.37	14202
LS-SVM	Test Q2	0.14	0.35	0.47
	LMSE	3.49	0.35	2.18
	Time(s)	167.60	36.71	8286
LS-RSVM	Test Q2	0.18	0.33	0.45
	LMSE	3.88	0.34	2.14
	Time(s)	661.49	39.17	170925
SVM-Torch	Test Q2	0.16	0.38	0.44
	LMSE	3.70	0.37	2.15
	Time(s)	212.43	368.0	6675

Table 2: Regression Benchmark Cases Q2 Error (100 x leave 10% out mode)

prediction performance for different datasets for regression. Let $\mathbf{y}_i$ be the i^{th} response, $\hat{\mathbf{y}}_i$ be the predicted response, $\mu_{\mathbf{y}}$, $\sigma_{\mathbf{y}}$, $\mu_{\hat{\mathbf{y}}}$, and $\sigma_{\hat{\mathbf{y}}}$ be the sample means and standard deviations of the actual and predicted responses, then

$$Q^2 = \frac{\sum_{ij}\left((\hat{\mathbf{y}}_i - \mu_{\hat{\mathbf{y}}})(\mathbf{y}_j - \mu_{\mathbf{y}})\right)}{\sigma_{\hat{\mathbf{y}}}\sigma_{\mathbf{y}}} \tag{27}$$

5.4 Results and Discussion

The best performance method and particular choice of tuning parameters for each benchmark problem is summarized in Table 5.4. The reported execution times in this table are for K-PLS only, to allow for a consistent comparison between different datasets. It can be observed that K-PLS generally compares well in prediction performance and computing time with other kernel methods.

The K-PLS method has the general advantage that the tuning is robust and simpler than other methods. Other than the choice of kernel, the only parameter is the number of latent variables. One need only consider a few discrete values as opposed to the continuous parameters in SVM and Ridge Regression. Only for datasets that exhibit pronounced non-linearity was it necessary to choose more than 5 latent variables. As currently implemented, K-PLS

Data Set	nxm	σ	λ	K	type	error	method	Time (s)
BUPA Liver	345x6	3.5	0.09695	5	class	27.5%	LS-SVM	47.47
Ionosphere	351x34	3.5	0.09930	5	class	4.2%	K-PLS	63.48
Tic-Tac-Toe	958x9	2.0	0.44817	20	class	0.1%	K-PLS	1044.00
Mushroom	8124x22	2.0	11.05298	12	class	0.0%	K-PLS	2.01*10^5
Boston Housing	506x13	5.0	0.17214	12	reg	0.13	K-PLS	181.00
Albumin	94x551	40.0	0.01385	5	reg	0.33	DK-PLS	34.00
Abalone	4177x8	4.0	4.07574	12	reg	0.44	K-PLS	26224.00

Table 3: Classification and Regression Results Summary

Table 4: Benchmark Summary Results

	Classified as negative class	Classified as positive class
Belonging to negative class	998 (TN)	165 (FP)
Belonging to positive class	17 (FN)	122 (TP)

Table 5: Confusion matrix for the checkerboard example of Fig. 3 with a zero threshold for discriminating between the negative class and the positive outlier class

and DK-PLS have the restriction that they require a machine with sufficient memory to store the full kernel matrix in memory. To allow processing of large sets in DK-PLS, a smaller rectangular kernel can be constructed using sampling. This was not required in these experiments. Results for DK-PLS are generally comparable to those obtained from K-PLS.

6 Case Study for Classification with Uneven Classes

Results from applying K-PLS to a 8x8 checkerboard problem for uneven classes are shown in Figure 3. In this case we used 12 latent variables and a Gaussian kernel with a kernel width of 0.08. There are 1000 training samples (including 121 minority patterns) and 1302 test samples (including 139 minority patterns). The results obtained from LS-SVM and LS-RSVM were comparable to those obtained with K-PLS and are not shown. The confusion matrix for the test set for a zero threshold for classification between the +1 and the 1 patterns is shown in Table 5.

7 Feature Selection with K-PLS

Feature selection, the process of determining which features to include in the input space, can further enhance the performance of K-PLS. Analyze/StripMiner has a feature selection method incorporated based on sensitivity analysis as described by Kewley and Embrechts [15]. Sensitivity analysis monitors the response as the features are tweaked one-at-a-time within their allowable range, while holding the other input features constant at their average value. Features that cause a larger variation in the response are deemed more important. Sensitivity analysis for feature selection generally requires multiple passes where just a small fraction of the features (10-30%) are dropped at a time. The features selection method implemented in Analyze/StripMiner operates in a leave-several-out mode where the sensitivities are averaged over multiple bootstraps.

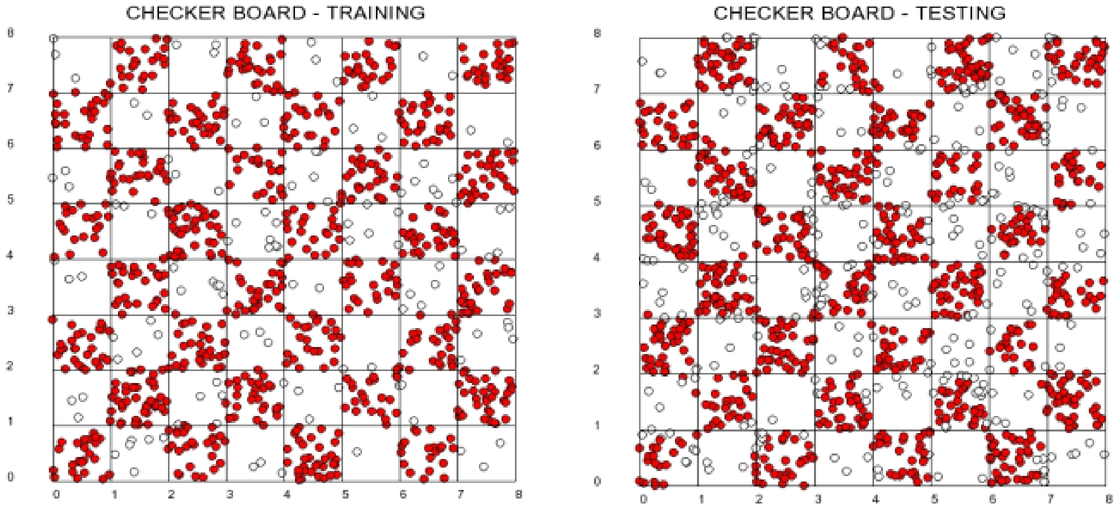

Figure 3: Training set and test set performance for an 8x8 checkerboard toy problem, where the red class (the 1 class) is the majority class. There are 1000 training samples (including 121 minority patterns) and 1302 test samples (including 139 minority patterns).

To evaluate the relative importance of a feature, the dataset can be augmented with a random gauge variable: features that are less sensitive than the random gauge variables are not likely to be important and dropped from the dataset. After these features have been dropped a new model is built, the sensitivities are determined again, and more features are dropped. This process is then repeated until all features show higher sensitivities than the random gauge variable.

In this section we will report on feature reduction studies on the albumin dataset. The Albumin dataset is a pharmaceutical dataset for predicting binding affinities to the human serum Albumin [4]. The basic descriptive features consist of 511 MOE and wavelet descriptors generated and made available as a public benchmark dataset in the Drug Design and Semi-Supervised Learning (DDASSL) project [6].

Figure 4 shows the change of Q2 as a function of the number of features. The optimal performance is for 35 features with a Q2 of 0.084 for 5 latent variables. This performance changes relatively little until the number of features is further reduced to 20. Figure 4 shows the scatterplot for the test set for the albumin data when the number of features is successively reduced from the original 511 to 35 features. Figure 5 shows the change in Q2 when the number of latent variables changes from 1 to 12.

8 Thoughts and Conclusions

This chapter presents a novel derivation of PLS from an optimization perspective. This derivation provides a basis for the machine learning researcher to better understand how and why

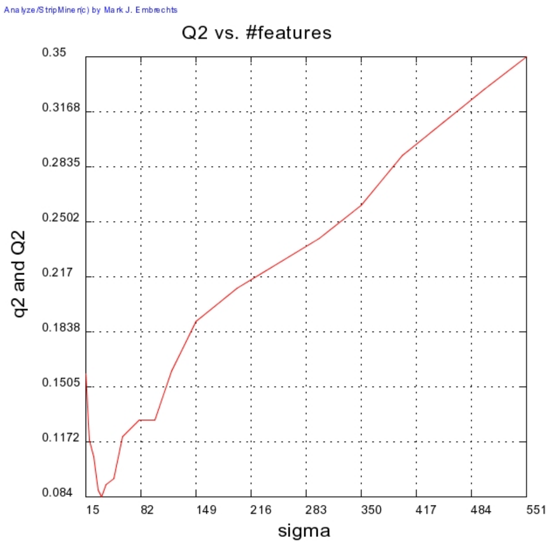

Figure 4: Increased prediction performance on the test set for the albumin dataset for 5 latent variables as indicated by the Q2 measure as a function of the reduced number of selected features by successively applying sensitivity analysis to K-PLS. The optimal number of features is 35 with a Q2 of 0.084.

PLS works and how it can be improved. PLS has two basic tricks. PLS produces low-rank approximations of the data that are aligned with the response and then uses low-rank approximations of both the input data and response data to produce the final regression model. We showed that from this perspective there are two general options for introducing kernels into PLS. The K-PLS algorithm of Rosipal and Trejo results from applying PLS to the data mapped into feature space. An alternative approach is to use PLS to make low-rank approximations of the kernel or Gram matrix. This produces the Direct K-PLS algorithm. DK-PLS approach can be used with any of the many PLS variants. Rectangular kernels can be factorized with DK-PLS. Combined with sampling of the kernel columns, this is a significant advantage on large datasets where full evaluation of the kernel is not possible. Several benchmark studies show that K-PLS and DK-PLS are comparable with other kernel-based support vector machine approaches in prediction performance and execution time, and tend to show a slightly better performance in general. PLS and variants are much easier to tune than SVM methods, because the only parameter (beyond choice of kernel) is the number of latent variables. Various options for K-PLS are implemented in the Analyze/StripMiner code, which operates on Windows and Linux platforms. The executable code and complimentary JAVA scripts for graphics are available for academic evaluation and scholarly use from www.drugmining.com. K-PLS and DK-PLS are outstanding methods when the kernel matrix can fit in memory.

Future research is need to fully exploit the potential of K-PLS type algorithms. Statistical learning theory may be able to shed light on why K-PLS generalizes well. Variants of K-PLS may help address its limitations. Current implementations require the full kernel matrix in memory because of the lack of sparsity in the solution and to support kernel centering and deflation. New variants of K-PLS with sparse solutions are needed that do not require full evaluation of the kernel matrix for both training and prediction of new points. Kernel

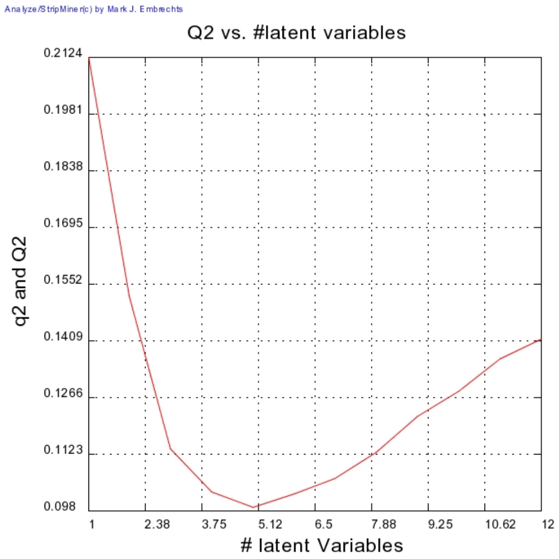

Figure 5: Prediction performance on the test set for the albumin dataset as indicated by the Q2 measure as a function of the selected number of latent variables for the optimal selection of 35 features.

centering could be eliminated by introducing bias when constructing the latent variables. Efficient algorithms for deflating the kernel matrix are needed.

Appendix

In this chapter we are frequently required to compute the best rank-one approximation of the matrix $\mathbf{X} \in R^{\ell \times n}$ given the matrix $\mathbf{t} \in R^{\ell \times 1}$. Specifically we want to find $\mathbf{p} \in R^{n \times k}$ that solves

$$\min_{\mathbf{p}} \sum_{i=1}^{\ell} ||\mathbf{X}_i - \mathbf{t}\mathbf{p}'||^2 = \sum_i \sum_j (\mathbf{X}_{ij} - \mathbf{t}_i \mathbf{p}_j)^2 \tag{28}$$

Thus we prove the following Lemma.
Lemma 1. The solution of problem (28) is $\mathbf{p} = \frac{\mathbf{X}'\mathbf{t}}{||\mathbf{t}||^2}$.

Proof. Taking the partial derivatives and setting them equal to 0 yields the optimality conditions:

$$\sum_i (\mathbf{t}_i \mathbf{X}_{ij} - \mathbf{t}_i^2 \mathbf{p}_j) = 0 \text{ for } j = 1, \ldots, n.$$

Thus

$$\sum_i (\mathbf{t}_i \mathbf{X}_{ij}) = ||\mathbf{t}||^2 \mathbf{p}_j \text{ for } j = 1, \ldots, n.$$

*

References

[1] G. Baffi, E. B. Martin, and A. J. Morris, Non-Linear Projection to Latent Structures Revisited: the Quadratic PLS Algorithm, *Computers and Chemical Engineering* **23** (1999) 395-411.

[2] A. Berglund and S. Wold, INLR, Implicit Non-Linear Latent Variable Regression, *Journal of Chemometrics* **11** (1997) 141-156.

[3] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning methods*, Cambridge: Cambridge (2000).

[4] G. Colmenarejo, A. Alvarez Pedraglio, and J.-L. Lavandera, Chemoinformatic Models to Predict Binding Affinities to Human Serum Albumin, *Journal of Medicinal Chemistry* **44** (2000) 4370 4378.

[5] R. Collobert, and S. Bengio, Support Vector Machines for Large-Scale Regression Problems, IDIAP-RR-00-17 (2000).

[6] M. J. Embrechts, K. P. Bennett, and C. Breneman, www.drugmining.com (2002).

[7] T. Evgeniou, M. Pontil, and T. Poggio, T. Statistical Learning Theory: A Primer, *International Journal of Computer Vision* **38**(1) (2000) 9-13.

[8] T. Evgeniou, M. Pontil, and T. Poggio, Regularization Networks and Support Vector Machines, in *Advances in Large Margin Classifiers*, MIT Press, Boston (2000).

[9] S. Fine and K. Scheinberg, Efficient SVM Training Using Low-Rank Kernel Representations, *Journal of Machine Learning Research* **2** (2001) 243-264.

[10] G. Fung and O. L. Mangasarian, Proximal Support Vector Machine Classifiers, in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2001), 77-86.

[11] A. Golbraikh and A. Tropsha, Beware of q2!, *Journal of Molecular Graphics and Modelling* **20** (2002) 269-276.

[12] A.E. Hoerl and R.W. Kennard, Ridge regression: Biased estimation for nonorthogonal problems, *Technometrics* **12**(3) (1970) 55-67.

[13] A. Höskuldsson, PLS Regression Methods, *Journal of Chemometrics* **2** (1998) 211-218.

[14] I. C. F. Ipsen, and C. D. Meyer, The Idea behind Krylov Methods, *American Mathematical Monthly* **105** (1998) 889-899.

[15] R. H. Kewley, and M. J. Embrechts, Data Strip Mining for the Virtual Design of Pharmaceuticals with Neural Networks, *IEEE Transactions on Neural Networks* **11**(3) (2000) 668-679.

[16] Y.-J. Lee and O. L. Mangasarian, RSVM: Reduced Support Vector Machine Classifiers, in *CD Proceedings of the SIAM International Conference on Data Mining*, SIAM, Philadelphia (2001).

[17] K-M Ling and C-J Lin, A study on Reduced Support Vector Machines, Technical Report, Dep. of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan (2002).

[18] D. G. Luenberger, *Linear and Nonlinear Programming*, Addison-Wesley, Reading, MA (1989).

[19] M. F. Möller, A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning, *Neural Networks* **6** (1993) 525-534.

[20] P. M. Murphy and D. W. Ahu, UCI Repository of Machine Learning Databases, www.ics.uci.edu/∼mlearn/MLRepository.html (2002).

[21] W. J. Nash, T. L. Sellers, S. R. Talbot, A. J. Cawthorn and W. B. Ford, The Population Biology of Abalone (Haliotis species) in Tasmania. I. Blacklip Abalone (H. rubra) from the North Coast and Islands of Bass Strait, Sea Fisheries Division, Technical Report No. 48 (ISSN 1034-3288) (1994).

[22] R. Rosipal and L.J. Trejo, Kernel Partial Least Squares Regression in Reproducing Kernel Hilbert Space, *Journal of Machine Learning Research* **2** (2001) 97-123.

[23] B. Schölkopf, A. Smola, and K.-R. Müller, Nonlinear Component Analysis as a Kernel Eigenvalue Problem, *Neural Computation* **10** (1998) 1299-1319.

[24] B. Schölkopf, S. Mika, C.J.C. Burges, P. Knirsch, K.-R. Müller, G. Raetsch and A.J. Smola, Input space versus feature space in kernel-based methods, *IEEE Transactions On Neural Networks* **10**(5) (1999) 1000-1017.

[25] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press (2001).

[26] J. A. K. Suykens and J. Vandewalle, Least-Squares Support Vector Machine Classifiers, *Neural Processing letters* **9**(3) (1999) 293-300.

[27] A. J. Smola and B. Schölkopf, Sparse Greedy Matrix Approximation for Machine Learning, *Proceedings of the Seventeenth Intenational Conference on Machine Learning*, Morgan Kaufman, Stanford (2000) 911-918.

[28] J. A. Swets, R. M. Dawes, and J. Monahan, Better Decisions through Science, *Scientific American* (2002) 82-87.

[29] V. Tresp and A. Schwaighofer, Scalable kernel systems, In *International Conference on Artificial Neural Networks* (2001).

[30] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, New York (1995).

[31] H. Wold, Estimation of principal components and related models by iterative least squares, In *Multivariate Analysis*, Academic Press, NY (1966) 391–420.

[32] H. Wold, Soft Modeling, The Basic Design and Some Extensions, In *Systems under Direct Observation Causality-Structure Prediction*, North Holland, Amsterdam (1981).

[33] H. Wold in *Food Research and Drug Analysis*, Applied Science Publishers (1983).

[34] S. Wold, Cross-Validatory Estimation of the Number of Components in Factor and Principal Component Models, *Technometrics* **20**(4) (1987) 397–405.

[35] S. Wold, N. Kettanch-Wold, and B. Skegerberg, Non-Linear PLS Modelling, *Chemometrics and Intelligent Laboratory Systems* **7** (1989) 53–65.

[36] S. Wold, Non-Linear Partial Least Squares Modelling II: Spline Inner Function, *Chemometrics and Intelligent Laboratory Systems* **14** (1992) 71–84.

[37] S. Wold, PLS-Regression: a Basic Tool of Chemometrics, *Chemometrics and Intelligent Laboratory Systems* **58** (2001) 109–130.

[38] W. Wu, D. L. Massarat and S. de Jong, The Kernel PCA Algorithm for Wide Data. Part I: Theory and Algorithms, *Chemometrics and Intelligent Laboratory Systems* **36** (1977) 165-172.

[39] W. Wu, D. L. Massarat and S. de Jong, The Kernel PCA Algorithm for Wide Data. Part II: Fast Cross-Validation and Application in Classification of NIR Data, *Chemometrics and Intelligent Laboratory Systems* **37** (1977) 271-280.