PLS based regression algorithms and their use in Multi-Agent Systems

Raúl Pierre Rentería
renteria@inf.puc-rio.br
Ruy Luiz Milidiú
milidiu@inf.puc-rio.br

PUC-RioInf.MCC18/04 June, 2004

Abstract: We present new algorithms aimed at prediction, based on traditional techniques such as partial least-squares and principal component analysis regression. Along, some experiments are reported showing the possible use of these techniques as an AI engine in a Multi-Agent System.

Keywords: Multi-Agent System, TAC, PLS, PCR

Resumo: Neste trabalho, são apresentados novos algoritmos para predição baseados em técnicas tradicionais como regressão por mínimos quadrados parciais e regressão por componentes principais. Juntamente, são relatados resultados experimentais mostrando a viabilidade do uso destas técnicas como componente IA em Sistemas Multi-Agentes.

Palavras-Chave: Sistemas Multi-Agentes, TAC, PLS, PCR

1 Introduction

Partial Least Squares regression (1; 2) has been widely used in the chemometric field for the robustness of the generated model when the number of variables is large when compared to the number of samples. This led to its application to many other areas, such as process monitoring, marketing analysis and image processing (3; 4; 5; 6; 7; 8).

In this paper, we review MKPLS, a multi-kernel based algorithm for Partial Least-Squares regression. As shown in (9), using different kernels at the training phase provides a better adaptation to the input data, resulting in not only a more compact model but also a better prediction quality. We present also MKPCR, a multi-kernel based algorithm for Principal Component Regression, showing that the approach developed for MKPLS can be used to other regression algorithms. Finally, we report some experiments on the linear PLS used as the AI engine in a known Multi-Agent System, the Trading Agent Competition.

In section 2, the traditional linear PLS is revisited. In section 3, our multi-kernel approach is reviewed, followed by the new multi-kernel principal component regression. In section 4, the empirical results obtained with the linear PLS on the Trading Agent Competition, a well known Multi-Agent System, are shown. Finally, in section 5, we summarize our findings.

2 Partial Least-Squares

Partial Least Squares (PLS) is a multivariate statistical method, based on the use of factors, which is aimed at prediction (10). The goal is to predict the values of a set of variables y based on the observed values of a set of variables x. In our case, x may be formed by the values of a time series window and y be taken as the value of a single future observation, or as the values of a set of future points in the same time series. The construction of a PLS model, requires a set of observation samples (patterns) and also their respective future values. Let X be the matrix containing in its rows the patterns of observations and Y be the matrix containing in its rows the future values to be predicted.

The PLS method provides additional predictive power when compared with both the methods of Multiple Linear Regression (MLR) and Principal Components Regression (PCR) (11).

MLR models Y from X using a least-squares criterion. All of X is available to model Y, and the method is dependent on the inversion of a matrix. PCR first models X by a score matrix U and then uses MLR to estimate the relationship between Um and Y.

The possibility of deleting factors and thus reducing dimensionality is considered its major advantage.

The PLS method is a modeling procedure that simultaneously estimates underlying factors in both X and Y. These factors are then used to define a subspace in X that is more adequate to model Y. With PCR, the rotation defined by the eigenvectors is used to find a subspace in X that subsequently is used to model Y. The approach taken by PLS is very similar to that of Principal Component Analysis (PCA), except that factors are chosen to describe the variables in Y as well as in X. This is accomplished by using the columns of the Y matrix to estimate the factors for X. At the same time, the columns of X are used to estimate the factors of Y. The resulting models are

$$\mathsf{X} = \mathsf{TP} + E \text{ and } \mathsf{Y} = \mathsf{UQ} + F$$

where the elements of T and U are called the scores of X and Y, respectively, and the elements of P and Q are called the loadings. The matrices E and F are the errors associated with modeling X and Y with the PLS model.

The T factors are not optimal for estimating the columns of X as is the case with PCA , but are rotated so as to simultaneously describe the Y matrix.

In the ideal situation, the sources of variation in X are exactly equal to the sources of variation in Y, and the factors for X and Y are identical. In real applications, X varies in ways not correlated to the variation in Y, and therefore $\mathsf{t} \neq \mathsf{u}$ (t and u are columns of T and U, respectively). However, when both matrices are used to estimate factors, the factors for the X and Y matrices have the following relationship:

$$\mathsf{u} = b\mathsf{t} + \epsilon$$

where b is termed the inner relationship between u and t and is used to calculate subsequent factors if the intrinsic dimensionality of X is greater than one. Geometrically, this states that the vectors u and t are approximately equal except for their lengths.

The main advantage of PLS over PCR is that it incorporates more information in the model-building phase. The advantages gained by employing factor-based methods, such as PLS, are due to the characteristics of the factors.

The first advantage of factor models comes from a computational limitation of MLR. MLR cannot be used in cases where the columns of measurement variables in X are linear combinations of each other. This is because MLR is based on the inversion of $\mathsf{X}^\top\mathsf{X}$, which is not possible if X has some columns that are linear combinations of the others. Furthermore,

the inversion process is unstable when some columns are nearly linear combinations of others. A model built on such an inverse will give large errors in the output variable estimates when noise is present in the X matrix. Factor-based models can eliminate collinearities without removing useful information.

The second main advantage of factor-based models is the possibility of removing noise from the data matrix. Again, using PCA as an example, one can demonstrate that the inclusion of all the eigenvectors can actually decrease the predictive ability of the model. To figure out when this is occurring, the analyst can use cross-validation and build a factor based calibration model using a set of samples named the *training set*. This calibration model is then applied to the X matrix of an independent set of samples named the *test set*, and an estimate of the Y matrix for the test set is obtained. Observe that, for the test set, the Y matrix is also known. The sum of squared deviations of the predicted values for the y_{ij}'s and the true y_{ij}'s are calculated for the test set. This value is termed the PRESS value, an acronym for predictive residual error sum of squares(10). Factors are then added one at a time, and the PRESS value is then calculated. What usually occurs is that the PRESS value either levels off or reaches a minimum before all the factors have been included. Inclusion of additional factors beyond this point actually results in an increased PRESS value, which means poorer prediction. This is because the random variation or noise in the X matrix is being used to fit the Y matrix, that is, over-fitting is occurring. By not including the factors that describe mainly noise, it is possible to increase the predictive ability of the model. Both PCR and PLS offer such a noise reduction capability.

3 Multi-Kernel Strategies

3.1 MKPLS: Multi-Kernel PLS regression algorithm

The main motivation for MKPLS was the PRESS curve obtained with one kernel PLS when compared to the standard linear PLS. For example, if both curves are plotted (Fig. 1) for a data set described in (9), we see that one kernel PLS outperforms PLS if one uses more than 13 factors. However, the performance of one kernel PLS is really poor for the first factors. The polynomial kernel defined as $K_{i,j} = (x_i \cdot x_j + 1)^2$ can barely model the predicted variable y for the first 10 factors. It would be interesting to have one regression model that would be as sharp as PLS on the first factors, and as sharp as one kernel PLS on the remaining ones. MKPLS generalizes the one kernel PLS by using a kernel matrix K_1 for the first f_1 factors and then switching to a different kernel K_2 for the remaining

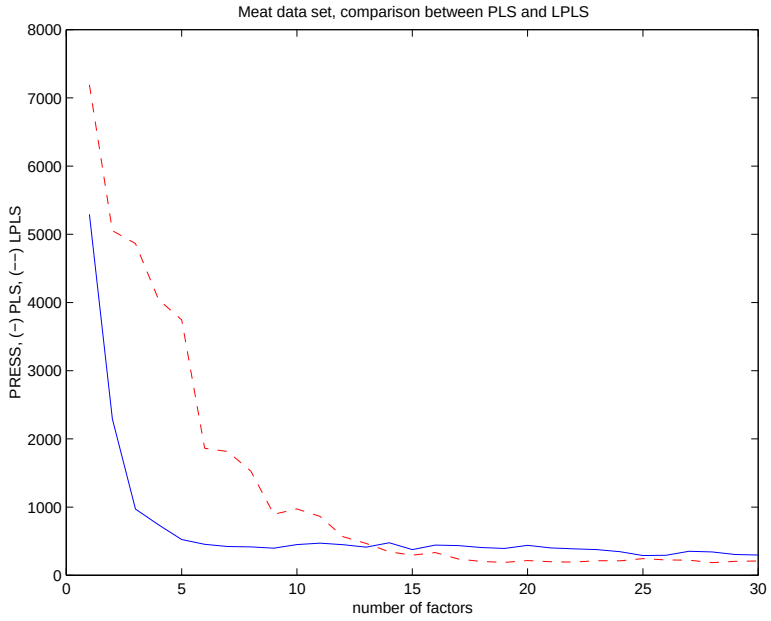

Figure 1: PRESS values for PLS and LPLS, Meat data set.

Multi-Kernel PLS regression approach
1. Apply one kernel PLS regression to first kernel $\mathtt{K}_1$
2. Deflate second kernel $\mathtt{K}_2$ using model obtained in step 1
3. Apply one kernel PLS regression to the deflated kernel $\mathtt{K}_2$

Figure 2: MKPLS main steps.

factors, as indicated by the high level algorithm on Fig. 2. For the interpretability of this operation it is important that the mapping done with K_2 includes the mapping done by K_1. Since the addition of two kernels is still a kernel, this can be simply done by defining $K_2 = K_1 + K$ where K denotes the kernel for the additional non-linearity, Gaussian or polynomial for instance.

This is meaningful in MKPLS since the switching of the kernels is done by deflating the factors t_i found with K_1. This would make no sense if the mapping of K_1 was not also done with K_2.

3.2 Training step

Given two kernel functions with the corresponding kernel matrices K_1 and K_2 for some training data set, the MKPLS model can be constructed through the following procedure:

1. obtain the first f_1 factors $\{t_i, b_i\}$ applying the one kernel PLS algorithm using the first kernel matrix K_1;

2. deflate the second kernel matrix K_2 and the dependent variable Y by applying the deflating algorithm described in Fig. 3;

3. apply again the one kernel PLS algorithm to the deflated kernel obtaining the remaining f_2 model factors.

At the end of this procedure, the set of $f_1 + f_2$ factors corresponding to the non-linear multi-kernel model will be available.

$\mathtt{K_2}$ and $\mathtt{Y}$ deflation for the training step	
1	for $\mathtt{i} = \mathtt{1}$ to $\mathtt{f_1}$
2	$\mathtt{g'_i} \leftarrow \mathtt{K_2 t_i}$
3	$\mathtt{a_i} \leftarrow \mathtt{t_i^\top t_i}$
4	$\mathtt{U} \leftarrow \mathtt{g'_i t_i^\top / a_i}$
5	$\mathtt{K_2} \leftarrow \mathtt{K_2 - U - U^\top + (t_i^\top g'_i) t_i t_i^\top / a_i^2}$
6	$\mathtt{Y} \leftarrow \mathtt{Y - t_i b_i^\top}$
7	end

Figure 3: MKPLS training deflating algorithm.

3.3 Prediction step

Since the one kernel PLS prediction algorithm uses a specific kernel matrix related to the test data set, it will be also necessary to switch the kernel matrices K'_1 and K'_2 during the prediction phase.

1. apply the same prediction algorithm starting with K'_1. However the set of f_1 score t'_i should be retained for deflation along with the predicted y;

2. deflate K'_2 using the algorithm in Fig. 4. Note the use of g'_i obtained during training deflation;

3. apply the prediction algorithm starting with the deflated K'_2 and the predicted y obtained in step 1.

$\mathtt{K}'_2$ deflation for the prediction step	
1	for $\mathtt{i} = 1$ to $\mathtt{f}_1$
2	$\mathtt{U}_1 \leftarrow \mathtt{K}'_2 \mathtt{t}_\mathtt{i} \mathtt{t}_\mathtt{i}^\top / \mathtt{a}_\mathtt{i}$
3	$\mathtt{U}_2 \leftarrow \mathtt{t}'_\mathtt{i} \mathtt{g}_\mathtt{i}'^\top / \mathtt{a}_\mathtt{i}$
4	$\mathtt{K}'_2 \leftarrow \mathtt{K}'_2 - \mathtt{U}_1 - \mathtt{U}_2 + (\mathtt{t}_\mathtt{i}^\top \mathtt{g}'_\mathtt{i}) \mathtt{t}'_\mathtt{i} \mathtt{t}_\mathtt{i}^\top / \mathtt{a}_\mathtt{i}^2$
5	end

Figure 4: MKPLS prediction deflating algorithm.

The MKPLS training and prediction procedures are very close to the one kernel version of PLS. The main difference being the deflation algorithms for the kernel function switching.

3.4 MKPCR: Multi-Kernel Principal Component regression

Following the same strategy presented for MKPLS, one can use a KPCA algorithm (12) and adapt it to multiple kernels. This leads to the algorithm in Fig. 5.

Multi-Kernel PCR regression
1. Apply one kernel PCR regression to first kernel $\mathtt{K}_1$
2. Deflate second kernel $\mathtt{K}_2$ using model obtained in step 1
3. Apply one kernel PCR regression to the deflated kernel $\mathtt{K}_2$

Figure 5: MKPCR approach.

The training and prediction steps follow a very similar procedure than the MKPLS, maintaining the same algorithms for deflating. The main difference remains in the $\mathtt{t}$ factor calculation, as explained in section 2, PCR uses scores obtained from the X matrix only, without any bias from the Y matrix.

4 Experiments with a Multi-Agent System

The experiments made with the linear PLS on the Trading Agent Competition are presented next. The results show the effectiveness of PLS methods as the AI engine in a multi-agent system.

4.1 Trading Agent System

The Trading Agent Competition (TAC) (13) is an international forum designed to encourage high quality research on competitive trading agents. The multi-agent system in TAC operates in a shopping scenario of goods for traveling purposes. The artificial agents are *travel agents* that buy and sell airplane tickets, hotel rooms, and entertainment tickets for clients. TAC scores are based on the client's preferences for trips, and net expenditures in the travel auctions.

In TAC, each agent has a goal of assembling travel packages. Every package is from TACtown to Tampa, for a 5-day period. Each agent is acting on behalf of eight clients, who express their preferences for various aspects of the trip. The objective of the travel agent is to maximize the total satisfaction of its clients, with the minimum net expenditure in the travel auctions. The satisfaction is defined as the sum of all client utilities.

A run of the game is called an instance. Several instances of the game are played during each round of the competition in order to evaluate each agent's average performance and to smooth the random variations in clients preferences. Each game instance takes twelve minutes.

4.2 Experiments

For this experiment, we tried to predict the price of the more expensive hotel room on the first day of one period, called GH1. For that purpose the closing price of both types of hotel room (GH1−4 and BH1−4) were collected during 2003 competition, giving the price history for 160 instances. This led to an amount of 1280 closing prices, corresponding to 8 prices collected during 160 instances.

On table 1 some basic statistics are reported, the mean and standard deviation calculated on the 8 closing prices. On the last line the ratio between the deviation and the mean is computed, showing that the closing price doesn't have a uniform distribution, what makes the prediction task more difficult. For a better analysis, the closing price GH1 history is plotted on Fig. 6.

Table 1: Basic statistics for the 8 closing prices

	GH1	GH2	GH3	GH4	BH1	BH2	BH3	BH4
mean	44.2	82.3	89.8	61.4	10.5	52.1	56.5	23.3
std. dev.	35.2	87.3	84.7	46.9	23.2	79.5	74.8	46.8
ratio	0.79	1.06	0.94	0.76	2.21	1.52	1.32	2.00

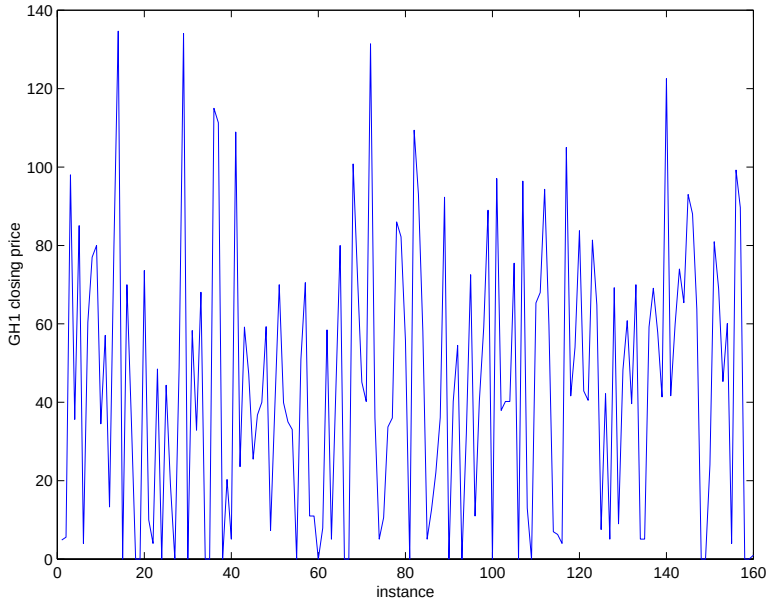

Figure 6: Expensive hotel room closing price on first day.

For the PLS modeling, we used the closing price GH1 of the last two weeks, resulting in a X matrix with 158 lines and 16 columns for the independent variables, and a 158 lines Y vector as the dependent variable. The variables were mean centered but not scaled. As explained in section 1, we splitted the set of 158 samples in two sets for training and testing purposes, containing 110 and 48 samples respectively. The mean of the PRESS value over 10 samplings was used as a quality of prediction measure.

After cross-validation, we selected 1 factor for prediction on the test set. For benchmarking purposes, we compared our results with (14), which uses

$$f'(t) = \alpha f(t-1) + (1-\alpha) f'(t-1)$$

with $\alpha = 3,5$, were $f'(t)$ corresponds to the predicted value on instance t and $f(t-1)$ the real closing price on instance $t-1$. On figure 7, is showed a comparison between the two models, for the first ten instances of the test set. As we can see, PLS performs better

than the model in (14). In fact, PLS produced a 18% lower prediction error, showing its superiority.

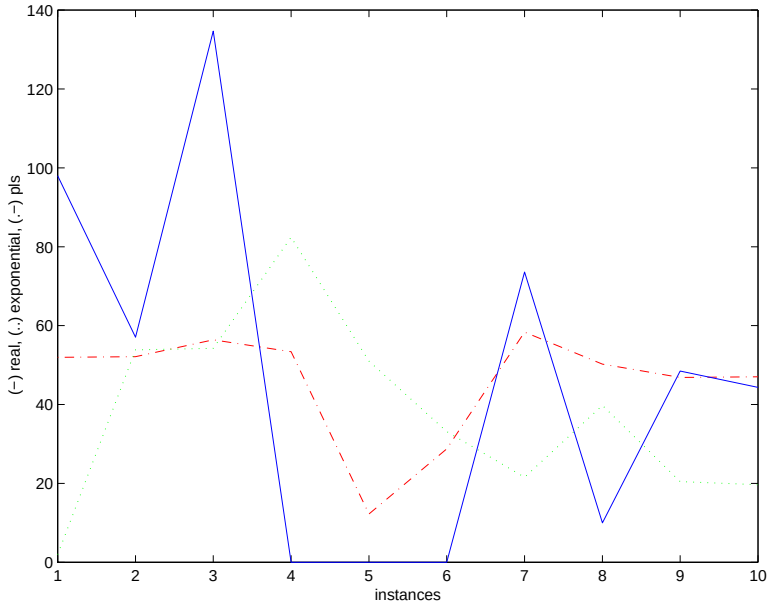

Figure 7: Comparison between models for selected instances.

However, analyzing the prediction for the entire test set (Fig. 8), we see that despite PLS best modeling, it's still unable to fit the real closing price, barely following its variation. After figure 8, the predicted values seem to behave like a white noise around the mean closing price. The main reason for that is the high deviation already observed. In fact, it seems to be complicate to predict all the GH1 closing prices with only one model. After some investigation, we can see that the closing prices (GH1 to GH4 and BH1 to BH4) are highly related during one instance. It follows that different models should be used for different instances following a strategy close to a mixture of expert models described in (4).

5 Conclusions

We showed that a statistical chemometric technique like PLS can be easily applied in a Multi-Agent System (MAS) as the core AI engine. For that purpose we presented the classical linear PLS regression and new, non-linear kernel based PLS algorithms, followed by an application in the TAC MAS. This experiment showed that PLS can outperform

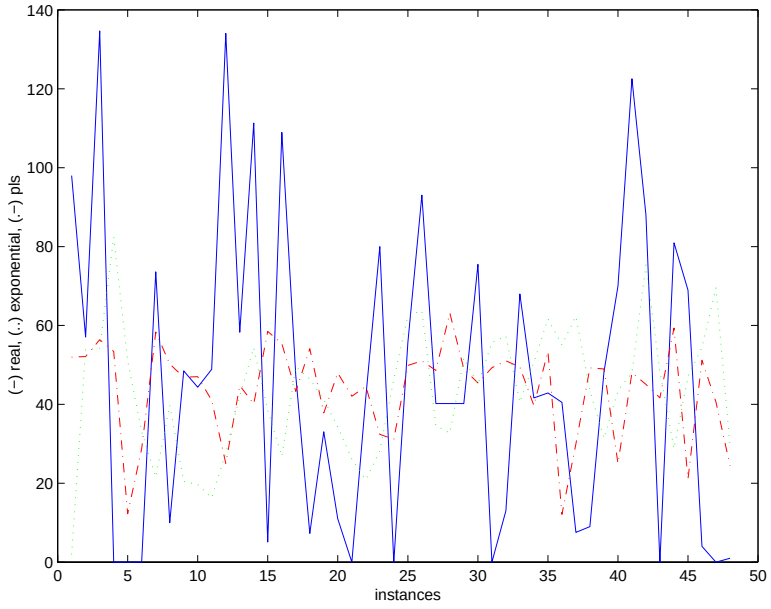

Figure 8: Comparison between models for all test set instances.

other techniques used in MAS (18% lower prediction error), but the overall performance shows that a better modeling of the TAC contest problem is still necessary.

References

[1] Wold, H.: Estimation of principal components and related models by iterative least squares. In Krishnaiah, P., ed.: Multivariate analysis II. Academic Press, New York (1966) 391–420

[2] Wold, S., Albano, C., III, W.D., Esbensen, K., Hellberg, S., Johansson, E., Sjöström, H.: Pattern recognition: finding and using regularities in multivariate data. In Martens, J., ed.: Proc. IUFOST Conf. Food Research and Data Analysis, London, Applied Science Publications (1983)

[3] Morineau, A., Tenenhaus, M., eds.: Les Méthodes PLS, Symposium International PLS'99, Cisia-Ceresta (1999)

[4] Milidiú, R., Machado, R., Rentería, R.: Time series forecasting throught wavelets and a mixture of experts models. Neurocomputing **28** (1999) 145–156

[5] Milidiú, R., Rentería, R.: Apls: A fast approximate algorithm for partial least-squares. In: V Brazilian Conference on Neural Networks, Rio de Janeiro - Brazil (2001) 661–666

[6] Milidiú, R.L., Rentería, R.: Ppls: An efficent parallel algorithm for partial least-squares. In: 12th Symposium on Computer Architecture and High Performance Computing, São Pedro, SP, Brazil (2000)

[7] Milidiú, R., Rentería, R., de Lucena, C.J.: Dpls and ppls: Two pls algorithms for large data sets. In: 2nd International Symposium on PLS and Related Methods, Capri, Italy (2001) 175–186

[8] Milidiú, R., Rentería, R.: Dpls and ppls: Two pls algorithms for large data sets. Computational Statistics and Data Analysis (2004) accepted for publication.

[9] Rentería, R., Milidiú, R.: A multi-kernel approach for non-linear kernel plsr. In: 3nd International Symposium on PLS and Related Methods, Lisbone, Portugal (2003) 175–186

[10] Geladi, P., Kowalski, B.R.: Partial least squares regression: A tutorial. Analytica Chimica Acta **185** (1986) 1–17

[11] Mardia, K., Kent, J., Bibby, J.: Multivariate Analysis. Academic Press, San Diego (1997)

[12] Schölkopf, B., Smola, A., Müller, K.R.: Nonlinear component analysis as a kernel eigenvalue problem. Technical report, Max-Planck-Institut (1996)

[13] Competition, T.A.: http://www.sics.se/tac (2001)

[14] Sardinha, J.A., Milidiú, R.L., de Lucena, C.J.P.: Engineering machine learning techniques into multi-agent systems. Technical report, PUC-Rio, Departamento de Informatica (2003)