Joint Regression and Linear Combination of Time Series for Optimal Prediction

Dries Geebelen [1], Kim Batselier [1], Philippe Dreesen [1], Marco Signoretto [1], Johan Suykens [1], Bart De Moor [1], Joos Vandewalle [1] *

1- ESAT-SCD K.U.Leuven/ IBBT-K.U.Leuven Future Health Department, Department of Electrical Engineering, Kasteelpark Arenberg 10, B-3001 Leuven, Belgium.

Abstract. In most machine learning applications the time series to predict is fixed and one has to learn a prediction model that causes the smallest error. In this paper choosing the time series to predict is part of the optimization problem. This time series has to be a linear combination of a priori given time series. The optimization problem that we have to solve can be formulated as choosing the linear combination of a priori known matrices such that the smallest singular vector is minimized. This problem has many local minima and can be formulated as a polynomial system which we will solve using a polynomial system solver. The proposed prediction algorithm has applications in algorithmic trading in which a linear combination of stocks will be bought.

1 Introduction

It is beneficial to have a linear combination of assets instead of one asset for several reasons: in modern portfolio theory [1] the aim is selecting a set of assets that has collectively lower risk than any individual asset, in pairs trading [2] a market neutral trading strategy is used that enables traders to profit from virtually any market conditions (uptrend, downtrend, or sideways movement). In this paper we select the linear combination of assets that minimizes the prediction error. The proposed optimization problem, which is the main contribution of this paper, is solved using a polynomial solver that uses solely basic linear algebra tools. Solving multivariate polynomial systems is normally done in the field of computational algebraic geometry [3]. Another class of polynomial solvers are homotopy continuation methods. These are a symbolic-numerical hybrid [4, 5].

*Research supported by: Research Council KUL: GOA/11/05 Ambiorics, GOA/10/09 MaNet , CoE EF/05/006 Optimization in Engineering (OPTEC) en PFV/10/002 (OPTEC), IOF-SCORES4CHEM, several PhD/postdoc & fellow grants; Flemish Government: FWO: PhD/postdoc grants, projects: G0226.06 (cooperative systems and optimization), G0321.06 (Tensors), G.0302.07 (SVM/Kernel), G.0320.08 (convex MPC), G.0558.08 (Robust MHE), G.0557.08 (Glycemia2), G.0588.09 (Brain-machine) research communities (WOG: ICCoS, ANMMM, MLDM); G.0377.09 (Mechatronics MPC); IWT: PhD Grants, Eureka-Flite+, SBO LeCoPro, SBO Climaqs, SBO POM, O&O-Dsquare; Belgian Federal Science Policy Office: IUAP P6/04 (DYSCO, Dynamical systems, control and optimization, 2007-2011); IBBT; EU: ERNSI; FP7-HD-MPC (INFSO-ICT-223854), COST intelliCIS, FP7-EMBOCON (ICT-248940), FP7-SADCO (MC ITN-264735), ERC HIGHWIND (259 166); Contract Research: AMINAL; Other: Helmholtz: viCERP; ACCM. The scientific responsibility is assumed by its authors.

The remainder of this paper is organized as follows. First we give a motivating example in Section 2. Section 3 explains the optimization problem. The polynomial solver is discussed Section 4. In Section 5 the algorithm is applied on a toy data set. Finally, Section 6 concludes the paper.

2 Motivating Example

Suppose time series $v_1(t)$ and $v_2(t)$ are defined as follows

$$v_1(1) = e_1(1), \quad v_1(t) = 0.8v_1(t-1) + 0.6e_1(t) \qquad t = 2, ..., m \qquad (1)$$
$$v_2(1) = e_2(1), \quad v_2(t) = -0.6v_2(t-1) + 0.8e_2(t) \qquad t = 2, ..., m \qquad (2)$$

where for any t, $e_1(t)$ and $e_2(t)$ are mutually independent i.i.d. gaussian variables with mean 0 and variance 1. Suppose $y_1(t)$ and $y_2(t)$ are generated as follows

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = A \begin{bmatrix} v_1(t) \\ v_2(t) \end{bmatrix} \text{ with } A = \begin{bmatrix} \sqrt{0.5} & \sqrt{0.5} \\ \sqrt{0.25} & \sqrt{0.75} \end{bmatrix}. \qquad (3)$$

One can notice $y_1(t)$, $y_2(t)$ have mean 0 and variance 1 as well. The expected square errors for the optimal first order autoregressive predictor equal

$$E(y_1(t) - b_1 y_1(t-1))^2 = 0.99 \qquad \text{with } b_1 = 0.1,$$
$$E(y_2(t) - b_2 y_2(t-1))^2 = 0.9375 \qquad \text{with } b_2 = -0.25.$$

These expected square errors are much worse than the lowest achievable expected square errors for $v_1(t)$ and $v_2(t)$ which are respectively 0.36 and 0.64. The linear combination of $y_1(t)$ and $y_2(t)$ with mean square value equal to 1 that causes the smallest expected square error for the optimal first order autoregressive predictor is $v_1(t) = 3.35y_1(t) - 2.73y_2(t)$. This is the solution that we hope to approximate with our algorithm given a finite number of datapoints of time series $y_1(t)$ and $y_2(t)$. In case $y_1(t)$ and $y_2(t)$ represent stock returns, buying the linear combination resulting in $v_1(t)$ will probably cause higher returns than both $y_1(t)$ and $y_2(t)$ because $v_1(t)$ can be predicted much better.

3 Optimization Problem

The most general optimization problem that we consider in this paper is

$$\min_{a,b,c} J(a,b,c) = \|Ya - f(a,b,c)\|_2^2 \quad s.t. \begin{cases} \|Ya\|_2 = 1 \\ f(a,b,c) = \sum_{i=1}^{n_b} b_i X_i a + Uc. \end{cases} \qquad (4)$$

This optimization problem has the following parameters and variables:

- Matrix $Y \in \mathbb{R}^{m \times n_a}$ represents n_a time series, each containing m datapoints. To avoid overfitting m should be much larger than n_a. With reference to the motivating example Y is $\begin{bmatrix} y_1(2) & y_1(3) & ... & y_1(m) \\ y_2(2) & y_2(3) & ... & y_2(m) \end{bmatrix}^T$.

- Vector $a \in \mathbb{R}^{n_a}$ contains the coefficients corresponding to a linear combination of Y. The resulting time series becomes Ya.
- Matrix $X_i \in \mathbb{R}^{m \times n_a}$ contains time series dependent inputs such that input $X_i a$ corresponds to time series Ya for every a. In the motivating examples n_b equals 1 and X_1 becomes $\begin{bmatrix} y_1(1) & y_1(2) & ... & y_1(m-1) \\ y_2(1) & y_2(2) & ... & y_2(m-1) \end{bmatrix}^T$.
- The vector $b \in \mathbb{R}^{n_b}$ contains linear regression coefficients such that b_i corresponds to input $X_i a$.
- The matrix $U \in \mathbb{R}^{m \times n_c}$ contains the external inputs which are independent from the time series. The n_c inputs at row j are the external inputs corresponds to the jth datapoint in the time series. To add a constant offset one can add a column containing ones to U. In the motivating example there is no U.
- The vector $c \in \mathbb{R}^{n_c}$ contains regression coefficients corresponding to the external inputs.

The constraint $\|Ya\|_2 = 1$ ensures the mean square value of the resulting time series equals 1 and thus avoids the solution in which a equals the zero vector. The optimization problem can be reformulated with $U^\dagger$ the pseudo-inverse of U

$$\min_{a,b} J(a,b) = a^T (Y' - \sum_{i=1}^{n} b_i X_i')^T (Y' - \sum_{i=1}^{n_b} b_i X_i') a \tag{5}$$

$$\text{s.t. } \|Ya\|_2 = 1 \tag{6}$$

where $c = U^\dagger (Y - \sum_{i=1}^{n_b} b_i X_i) a$, $Y' = (I - UU^\dagger)Y$ and $X_i' = (I - UU^\dagger)X_i$.

In the optimum, a equals the right singular vector of $(Y' - \sum_{i=1}^{n} b_i X_i')$ corresponding to the smallest singular value σ_{n_a}. The prediction error in this optimum equals $\sigma_{n_a}^2$. Adding the Lagrange multiplier λ to enforce the constraint $\|Ya\|_2 = 1$ gives the Lagrangian

$$\mathcal{L}(a,b,\lambda) = a^T (Y' - \sum_{i=1}^{n_b} b_i X_i')^T (Y' - \sum_{i=1}^{n_b} b_i X_i') a + \lambda (1 - a^T Y^T Y a) \tag{7}$$

The first-order optimality conditions that need to be satisfied are

$$\frac{\partial \mathcal{L}(a,b,\lambda)}{\partial a} = 2(Y' - \sum_{i=1}^{n_b} b_i X_i')^T (Y' - \sum_{i=1}^{n_b} b_i X_i') a - 2\lambda Y^T Y a = 0 \tag{8}$$

$$\frac{\partial \mathcal{L}(a,b,\lambda)}{\partial b_i} = 2a^T X_i'^T (Y' - \sum_{i=1}^{n_b} b_i X_i') a = 0 \qquad i = 1, ..., n_b \tag{9}$$

$$\frac{\partial \mathcal{L}(a,b,\lambda)}{\partial \lambda} = 1 - a^T Y^T Y a = 0. \tag{10}$$

This is a system of $n = n_a + n_b + 1$ polynomial equations of degree 3 or less.

4 Polynomial Solver

A quick overview of the basic polynomial root-finding algorithm is given for the case that there are no roots with multiplicities and roots at infinity. More details for the case of multiplicities and roots at infinity can be found in [6, 7]. The main computational tool of the algorithm is either the singular value decomposition (SVD) or rank-revealing QR decomposition and the eigenvalue decomposition.

Algorithm 4.1
Input: system of the n-variate polynomials $F = f_1, \ldots, f_s$ *of degrees* $d_1, \ldots, d_s$
Output: kernel K

1: $M \leftarrow$ *coefficient matrix of* F *up to degree* $d = \sum_{i=1}^{s} d_i - n + 1$
2: $s \leftarrow$ *nullity of* M
3: $Z \leftarrow$ *basis null space from SVD*(M) *or QR*(M)
4: $S_1 \leftarrow$ *row selection matrix for* s *linear independent rows of* Z
5: $S_2 \leftarrow$ *row selection matrix for shifted rows of* $S_1\, Z$
6: $B \leftarrow S_1\, Z$
7: $A \leftarrow S_2\, Z$
8: $[V, D] \leftarrow$ *solve eigenvalue problem* $B\, V\, D = A\, V$
9: $K \leftarrow Z\, V$

The first step in the algorithm is to construct the coefficient matrix of the polynomial system F containing equations (8), (9) and (10) up to a degree $d = \sum_{i=1}^{s} d_i - n + 1$. In order to explain how this coefficient matrix is made we first need to explain how multivariate polynomials are represented by their coefficient vectors. This is achieved by simply storing the coefficients of the polynomial into a row vector according to a certain monomial ordering. In principle any monomial ordering can be used. We refer to [3] for more details on monomial orderings. The following example illustrates this for a bivariate polynomial of degree 2.

Example 4.1 *The vector representation of* $2 + 3x_1 - 4x_2 + x_1x_2 - 7x_2^2$ *is*

$$\begin{array}{cccccc} 1 & x_1 & x_2 & x_1^2 & x_1x_2 & x_2^2 \\ \big(2 & 3 & -4 & 0 & 1 & -7\big). \end{array}$$

We can now define the coefficient matrix of a multivariate polynomial system up to a degree d.

Definition 4.1 *Given a set of* n*-variate polynomials* $f_1, \ldots, f_s$*, each of degree* $d_i\, (i = 1, \ldots, s)$ *then the coefficient matrix of degree* d*,* $M(d)$*, is the matrix containing the coefficients of*

$$M(d) \;=\; \big(f_1^T \quad x_1 f_1^T \quad \ldots \quad x_n^{d-d_1} f_1^T \quad f_2^T \quad x_1 f_2^T \quad \ldots \quad x_n^{d-d_s} f_s^T\big)^T \qquad (11)$$

where each polynomial f_i *is multiplied with all monomials from degree 0 up to* $d - d_i$ *for all* $i = 1, \ldots, s$*.*

Note that the coefficient matrix not only contains the original polynomials $f_1, \ldots, f_s$ but also 'shifted' versions where we define a shift as a multiplication with a monomial. The dependence of this matrix on the degree d is of crucial importance, hence the notation $M(d)$. It can be shown [8] that the degree $d = \sum_{i=1}^{s} d_i - n + 1$ provides an upper bound for the degree for which all the solutions of the polynomial system appear in the kernel of $M(d)$. This brings us to step 2 of Algorithm 4.1. The number of solutions of F are counted by the dimension of the kernel of $M(d)$. For the case that there are no multiplicities and no solutions at infinity, this is then simply given by the Bezout bound $m_B = \prod_{i=1}^{s} d_i$. Steps 3 up to 9 find all these solutions from a generalized eigenvalue problem which is constructed from exploiting the structure of the canonical kernel. The canonical kernel K is a n-variate Vandermonde matrix. It consists of columns of monomials, ordered according to the chosen monomial ordering and evaluated in the roots of the polynomial system. This monomial structure allows to use a shift property which is reminiscent of realization theory. This shift property tells us that the multiplication of rows of the canonical kernel K with any monomial corresponds with a mapping to other rows of K. This can be written as the following matrix equation

$$S_1 K D = S_2 K \tag{12}$$

where S_1 and S_2 are row selection matrices and D a diagonal matrix which contains the shift monomial on the diagonal. S_1 will select the first m_B linear independent rows of K. The canonical kernel K is unfortunately unknown but a numerical basis Z for the kernel can be computed from either the SVD or QR decomposition. This basis Z is then related to K by means of a linear transform V, $K = ZV$. Writing the shift property (12) in terms of the numerical basis Z results in the following generalized eigenvalue problem

$$BVD = AV \tag{13}$$

where $B = S_1 Z$ and $A = S_2 Z$ are square nonsingular matrices. The eigenvalues D are then the shift monomial evaluated in the different roots of the polynomial system. The canonical kernel K is easily reconstructed from $K = ZV$. The monomial ordering used in Algorithm 4.1 is such that the first row of the canonical kernel corresponds with the monomial of degree 0. Therefore, after normalizing K such that its first row contains ones, all solutions can be read off from the corresponding first degree rows.

5 Example

In this Section we obtain the solutions using the algorithms discussed in this paper given 100 datapoints of the time series $y_1(t)$ and $y_2(t)$ as defined in (3). Table 1 shows that the solution with the lowest training error has, after normalization, a generalization error 0.363 close to the lowest achievable generalization error 0.360. This solution is approximately a multiple of $v_1(t)$ as we hoped for.

Table 1: Real-valued solutions of the polynomial system of the time series $y_1(t)$ and $y_2(t)$ for 100 datapoints per time series. The optimized parameters are a_1, a_2, b and λ, the mean square training error is denoted as E_{tr} and the generalization error, after normalization such that (6) is satisfied out-of-sample, is denoted as E_{gen}. Every linear combination of $y_1(t)$ and $y_2(t)$ can be written as a linear combination of $v_1(t)$ and $v_2(t)$ such that $a_1 y_1(t) + a_2 y_2(t) = a'_1 v_1(t) + a'_2 v_2(t)$. For each pair of symmetric solutions only one solution is shown in the table. The other solution has parameters equal to respectively $-a_1$, $-a_2$, b and λ.

	a_1	a_2	b	λ	E_{tr}	E_{gen}	a'_1	a'_2
Sol1	3.362	-2.705	0.808	0.308	0.308	0.363	1.025	0.035
Sol2	-2.279	3.047	-0.590	0.650	0.650	0.652	-0.088	1.027
Sol3	3.888	-4.057	0.000	1.000	1.000	1.000	0.720	-0.765
Sol4	0.398	0.616	0.000	1.000	1.000	1.000	0.590	0.815

6 Conclusion

In this paper we proposed an algorithm in which both the time series to predict and the prediction model are learned. On a toy example, the optimum found by our algorithm is close to the actual optimum. These results are promising towards real data sets such as predicting a linear combination of stock returns. Making the algorithm more scalable in the sense that a large number of time series and a large number of regression coefficients b_i can be handled, is one of the main challenges towards future research. Also, an extension towards blind source separation [9] is interesting to investigate.

References

[1] H.M. Markowitz, *Portfolio Selection: Efficient Diversification of Investments*, John Wiley & Sons, New York, 1959.

[2] G. Vidyamurthy, *Pairs trading: quantitative methods and analysis*, John Wiley & Sons, New York, 2004.

[3] D. A. Cox and J. B. Little and D. O'Shea, *Ideals, Varieties and Algorithms*, Springer-Verlag, 2007.

[4] J. Verschelde, Algorithm 795: PHCpack: a general-purpose solver for polynomial systems by homotopy continuation, *ACM Trans. Math. Softw.*, 25(2):251-276, ACM, 1999.

[5] A. Morgan, *Solving polynomial systems using continuation for engineering and scientific problems*, Prentice-Hall, Englewood Cliffs, N.J., 1987.

[6] P. Dreesen, K. Batselier and B. De Moor, Back to the Roots: Polynomial System Solving, Linear Algebra, Systems Theory. Technical Report 10-191, ESAT/SCD/SISTA, Katholieke Universiteit Leuven, Belgium, 2011.

[7] K. Batselier, P. Dreesen and B. De Moor, Global Optimization and Prediction Error Methods. Technical Report 11-199, ESAT/SCD/SISTA, Katholieke Universiteit Leuven, Belgium, 2011.

[8] F. S. Macaulay, On some formulae in elimination, *Proc. London Math. Soc.*, 35:3-27, 1902.

[9] S. Choi, A. Cichocki, H.M. Park, S.Y. Lee, Blind Source Separation and Independent Component Analysis: a Review, *Neural Inf. Proc. Lett. and Reviews*, 6(1):1-57, 2005.