Online Ridge Regression method using sliding windows

Paola Arce
Center for Technological Innovation in High Performance Computing CTI-HPC, UTFSM
Valparaíso, Chile
Email: paola.arce@usm.cl

Luis Salinas
Informatics Dpt. and Centro Científico-Tecnológico de Valparaíso - CCTVal
UTFSM
Valparaíso, Chile
Email: lsalinas@inf.utfsm.cl

***Abstract*—A new regression method based on the *aggregating algorithm for regression* (AAR) is presented. The proposal shows how *ridge regression* can be modified in order to reduce the number of operations by avoiding the inverse matrix calculation only considering a sliding window of the last input values. This modification allows algorithm expression in a recursive way and therefore its use in an online context. Ridge regression, AAR and our proposal were compared using the closing stock prices of 45 stocks from the technology market from 2000 to 2012. Empirical results show that our proposal performs better than the other two methods in 28 of 45 stocks analyzed, due to the lower MSE error.**

***Keywords*-Ridge Regression; Machine Learning; Online Learning;**

I. Introduction

The learning from examples problem is an ill-posed problem which admits an infinite number of solutions. In order to restrict the space of admissible solutions, the regression problem is usually formulated in terms of regularization theory [1] as an optimization problem, which minimizes the functional:

$$J(\mathbf{w}) = \sum_{t=1}^{m}(y_t - f(\mathbf{x}_t))^2 + \gamma R(\mathbf{w}), \qquad f \in \mathcal{H}, \quad (1)$$

where m is the number of samples $(\mathbf{x}_t, y_t)$ with $\mathbf{x}_t \in \mathbb{R}^l$, l correspond to the number of features, $y_t \in \mathbb{R}$ is the target, $R(\mathbf{w}) = ||w||^2$ where $||.||_K^2$ is a norm in a *reproducing kernel hilbert space* $\mathcal{H}$ defined by the positive definite form K, and γ is a regularization parameter [2]. When the hypothesis space is reduced, the risk of overfitting the training data decreases and therefore leading to better generalization capability. *Ridge regression* (RR) is a batch method generally used to solve this problem, which is a generalization of least squares method (LS).

However, online algorithms are more attractive than batch algorithms because their simplicity and ability to manage large data sets, this is why they are popular in financial applications.

There are several popular online methods such as perceptron [3], passive-aggressive [4], stochastic gradient descent [5], aggregating algorithm [6] and the second order perceptron [7]. In [8] it is provided an in-deph analysis of online learning.

The *aggregating algorithm for regression* (AAR) method formulates a recursive formulation of RR in an online way. AAR consider all data to make a prediction, but in highly variant scenarios the old data could be useless.

In this paper, we propose an online method based on the idea presented by [6] considering in our case a single sliding window of the most recent data. This proposal also reduces the number of operations at every step of the algorithm by expressing the inverse matrix in an iterative form. Our algorithm is later tested with financial data from stock market.

II. Ridge Regression

A. Regression problems

The objective of regression problems is to find a function f which explains the relation between an input $\mathbf{x}_t \in \mathbb{R}^l$, and an output $y_t \in \mathbb{R}$ such that: $y_t = f(\mathbf{x}_t) + \epsilon_t$ for a set of m data points $\{(\mathbf{x}_t, y_t)\}_{t=1}^m$. If the relationship between y_t and $\mathbf{x}_t$ is thought to be linear, f can be written as:

$$f(\mathbf{x}_t) = \mathbf{w}^\intercal \mathbf{x}_t = \sum_{i=1}^{l} w(i)x_t(i)\,,$$

where $\mathbf{w}$ is a weight vector determined in a training phase.

The least squares method is a well known way to solve a regression problem. This method consists of minimizing the sum of squared error:

$$J(\mathbf{w}) = \sum_{t=1}^{m}(f(\mathbf{x}_t) - y_t)^2 = \sum_{t=1}^{m}(\mathbf{w}^\intercal \mathbf{x}_t - y_t)^2\,, \quad (2)$$

which is equivalent to minimize $J(\mathbf{w})$ with $\gamma = 0$ in equation (1). Expressed in matrix form this amounts to:

$$J(\mathbf{w}) = \left|\left| \begin{bmatrix} x_1(1) & \cdots & x_1(l) \\ \vdots & \ddots & \vdots \\ x_m(1) & \cdots & x_m(l) \end{bmatrix} \begin{bmatrix} w(1) \\ \vdots \\ w(l) \end{bmatrix} - \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} \right|\right|_{L^2}^2$$

or a more reduced expression is:

$$J(\mathbf{w}) = \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2,$$

where

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\intercal \\ \vdots \\ \mathbf{x}_m^\intercal \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} \quad \text{and} \quad \mathbf{w} = \begin{bmatrix} w(1) \\ \vdots \\ w(l) \end{bmatrix}$$

As is well known, the optimal solution $\mathbf{w}_*$ obtained minimizing equation (2) is:

$$\mathbf{w}_* = (\mathbf{X}^\intercal \mathbf{X})^{-1} \mathbf{X}^\intercal \mathbf{y} \tag{3}$$

Thus, when a new input $\mathbf{x}_t$ arrives, the prediction of the target value y_t is defined as:

$$f(\mathbf{x}_t) = \mathbf{w}_*^\intercal \mathbf{x}_t \,.$$

B. Ridge Regression

In order to avoid the singularity of the matrix $\mathbf{X}^\intercal\mathbf{X}$ in equation (3), a regularization term is introduced. The optimization problem including the regularization term is shown in equation (1).

When $R(\mathbf{w}) = \|\mathbf{w}\|^2$, the method is called ridge regression and the optimal solution $\mathbf{w}_*$ is well known:

$$\mathbf{w}_* = (X^\intercal X + \gamma \mathbb{I})^{-1} X^\intercal y \,, \tag{4}$$

Equation (4) can be also be expressed as:

$$\begin{aligned} \mathbf{w}_* &= \Big(\sum_{t=1}^{m} \mathbf{x}_t \mathbf{x}_t^\intercal + \gamma \mathbb{I}\Big)^{-1} \sum_{t=1}^{m} y_t \mathbf{x}_t \\ \mathbf{w}_* &= \mathbf{A}^{-1}\mathbf{b}\,, \end{aligned}$$

where

$$\mathbf{A} = \sum_{t=1}^{m} \mathbf{x}_t \mathbf{x}_t^\intercal + \gamma \mathbb{I} \qquad \text{and} \qquad \mathbf{b} = \sum_{t=1}^{m} y_t \mathbf{x}_t \,.$$

Algorithm 1 Ridge Regression

```
Input:
    {$\mathbf{x}_1, \dots, \mathbf{x}_m$}: $m$ input vectors
    {$y_1, \dots, y_m$}: $m$ targets
Output:
    {$f(\mathbf{x}_1), \dots, f(\mathbf{x}_m)$}: model predictions
1: Initialize $\mathbf{A} = \gamma\mathbb{I}$ and $\mathbf{b} = 0$
2: for $t = 1$ to $m$ do
3:     read new $\mathbf{x}_t$
4:     output prediction $f(\mathbf{x}_t) = \mathbf{b}^\intercal \mathbf{A}^{-1} \mathbf{x}_t$
5:     $\mathbf{A} = \mathbf{A} + \mathbf{x}_t \mathbf{x}_t^\intercal$
6:     Read new $y_t$
7:     $\mathbf{b} = \mathbf{b} + y_t \mathbf{x}_t$
8: end for
```

The ridge regression method is shown in algorithm 1 where the prediction value for a new input $\mathbf{x}_{m+1}$ is:

$$\begin{aligned} f(\mathbf{x}_{m+1}) &= \mathbf{w}_*^\intercal \mathbf{x}_{m+1} \\ &= \mathbf{b}^\intercal \mathbf{A}^{-1} \mathbf{x}_{m+1} \,. \end{aligned}$$

III. The Aggregating Algorithm for Regression

The AAR, proposed by [6], is an application of the aggregating algorithm to the problem of regression.

The prediction formula for AAR is given by equation (5):

$$f(\mathbf{x}_{m+1}) = \Big(\sum_{t=1}^{m} y_t \mathbf{x}_t\Big)^\intercal \Big(\sum_{t=1}^{m+1} \mathbf{x}_t \mathbf{x}_t^\intercal + \gamma\mathbb{I})\Big)^{-1} \mathbf{x}_{m+1} \,, \tag{5}$$

which is very similar to the RR method except because AAR includes information of the new input $\mathbf{x}_{m+1}$. This means that AAR updates matrix A with the new input $\mathbf{x}_{m+1}$ before the prediction $f(\mathbf{x}_{m+1})$ is made.

Algorithm 2 shows this procedure based on equation (5). This algorithm can be obtained by swapping lines 4 and 5 from algorithm 1.

Algorithm 2 *The aggregating algorithm for regression*

```
Input:
    {$\mathbf{x}_1, \dots, \mathbf{x}_m$}: $m$ input vectors
    {$y_1, \dots, y_m$}: $m$ targets
Output:
    {$f(\mathbf{x}_1), \dots, f(\mathbf{x}_m)$}: model predictions
1: Initialize $\mathbf{A} = \gamma\mathbb{I}$ and $\mathbf{b} = 0$
2: for $t = 1$ to $m$ do
3:     read new $\mathbf{x}_t$
4:     $\mathbf{A} = \mathbf{A} + \mathbf{x}_t \mathbf{x}_t^\intercal$
5:     output prediction $f(\mathbf{x}_t) = \mathbf{b}^\intercal \mathbf{A}^{-1} \mathbf{x}_t$
6:     Read new $y_t$
7:     $\mathbf{b} = \mathbf{b} + y_t \mathbf{x}_t$
8: end for
```

IV. Online Ridge Regression method

Despite of RR and AAR being very succesfull methods, they consider all the available data for making predictions. However some time series show a strong dependence on the latest information instead of all the data. The method proposed consists of a modification of AAR considering a sliding window which contains only the last L samples and the new input $\mathbf{x}_t$, i.e. $\{\mathbf{x}_i\}_{i=t-L}^{t}$.

In order to formulate the algorithm we need to first define the following matrices:

$$\mathbf{X}(t) = \begin{bmatrix} \mathbf{x}_{t-L}^\intercal \\ \vdots \\ \mathbf{x}_t^\intercal \end{bmatrix}, \mathbf{y}(t) = \begin{bmatrix} y_{t-L} \\ \vdots \\ y_t \end{bmatrix} \,.$$

The optimal solution using a sliding windows is then:

$$\mathbf{w}(t)_* = (\mathbf{X}(t)^\intercal \mathbf{X}(t) + \gamma \mathbb{I})^{-1} \mathbf{X}(t)^\intercal \mathbf{y}(t)\,. \qquad (6)$$

It is worth noticing that the matrix $\mathbf{X}(t+1)$ is slightly different from $\mathbf{X}(t)$:

$$\mathbf{X}(t+1) = \begin{bmatrix} \mathbf{x}_{t-L+1}^\intercal \\ \vdots \\ \mathbf{x}_{t+1}^\intercal \end{bmatrix}. \qquad (7)$$

Therefore the updated matrix $\mathbf{A}$ is obtained by:

$$\begin{aligned} \mathbf{A} &= \mathbf{X}(t+1)^\intercal \mathbf{X}(t+1) \\ \mathbf{A} &= \mathbf{X}(t)^\intercal \mathbf{X}(t) + \mathbf{x}_{t+1}\mathbf{x}_{t+1}^\intercal - \mathbf{x}_{t-L}\mathbf{x}_{t-L}^\intercal\,. \end{aligned}$$

Algorithm 3 shows the complete procedure, which is very similar to AAR (algorithm 2) except for the calculation of matrix $\mathbf{A}$.

Algorithm 3 Recursive Ridge Regression

```
Input:
    $\{\mathbf{x}_1, \dots, \mathbf{x}_m\}$: $m$ input vectors
    $\{y_1, \dots, y_m\}$: $m$ targets
    $L$: sliding window size ($L < m$)
Output:
    $\{f(\mathbf{x}_{L+1}), \dots, f(\mathbf{x}_m)\}$: model predictions
1: Initialize $\mathbf{A} = \sum_{t=1}^{L} \mathbf{x}_t\mathbf{x}_t^\intercal + \gamma\mathbb{I}$ and $\mathbf{b} = \sum_{t=1}^{L} y_t\mathbf{x}_t$
2: for $t = L+1$ to $m$ do
3:     read new $\mathbf{x}_t$
4:     $\mathbf{A} = \mathbf{A} + \mathbf{x}_t\mathbf{x}_t^\intercal - \mathbf{x}_{t-L-1}\mathbf{x}_{t-L-1}^\intercal$
5:     output prediction $f(\mathbf{x}_t) = \mathbf{b}^\intercal \mathbf{A}^{-1}\mathbf{x}_t$
6:     Read new $y_t$
7:     $\mathbf{b} = \mathbf{b} + y_t\mathbf{x}_t$
8: end for
```

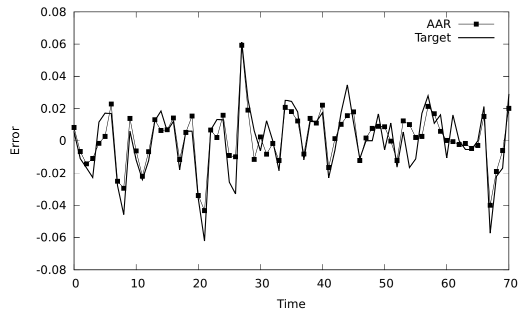
A. Matrix inverse

In order to avoid the matrix inverse calculation, we can use the Sherman Morrison formula.

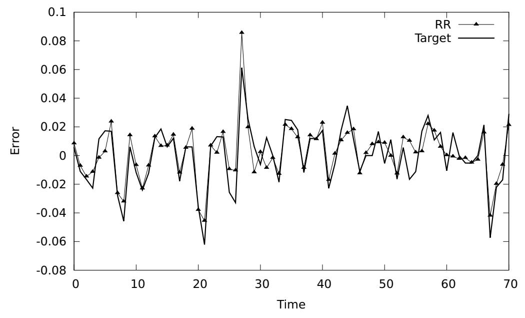
If $\mathbf{A}$ is a positive definite matrix and its inverse matrix is known, then the inverse of the matrix $\mathbf{B} = \mathbf{A} + \mathbf{x}\mathbf{x}^\intercal$ can be obtained as:

$$\mathbf{B}^{-1} = \mathbf{A}^{-1} - \frac{(\mathbf{A}^{-1}\mathbf{x})(\mathbf{A}^{-1}\mathbf{x})^\intercal}{1 + \mathbf{x}^\intercal \mathbf{A}^{-1}\mathbf{x}}\,.$$

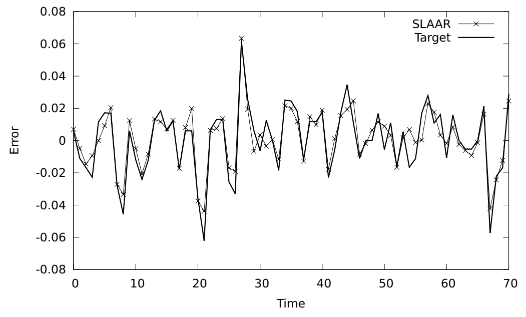
This reduces the inverse matrix computation order from $O(n^3)$ to $O(n^2)$.



(a) AAR v/s the target for SPY



(b) RR v/s the target for SPY



(c) SLAAR v/s the target for SPY

V. Experimental Results

In order to measure performance of our algorithm, we used daily stock returns of 45 stocks from the technology sector dated from the 1st of January 2000 to the 1st of July 2012.

Returns $\{x_t\}_{t=1}^{m-1}$ were defined in based on stock prices $\{p_t\}_{t=1}^{m}$ and are related as:

$$x_t = \frac{p_{t+1} - p_t}{p_t}\,.$$

The objective is to build a model of the returns of stock k based on returns of other stocks of the same financial sector.

Therefore, every input vector $\mathbf{x}_t$ will have 44 stock returns at time t without considering information of stock k. The target vector $\mathbf{y}$ will be the stocks returns of stock k.

We compared RR, AAR and our proposal called sliding windows AAR (SLAAR) and the results are shown in table I. The MSE was shown in bold when our method was better than RR and AAR. The table shows that our method outperforms in 28 of 45 stocks.

	AAR	RR	SLAAR	Best L
Stock	AAR error	RR error	SLAAR error	Best L
IBM	0.230307	0.231317	**0.228963**	70
MMM	0.199237	0.197172	**0.195550**	990
CVX	0.155454	0.152070	0.155150	750
UTX	0.260980	0.268351	0.261140	340
CAT	0.223338	0.222240	**0.210500**	360
MCD	0.268211	0.266997	0.267182	990
BA	0.292699	0.295623	**0.292180**	150
XOM	0.160376	0.157125	0.159074	780
JNJ	0.213276	0.214747	**0.212601**	680
PG	0.494937	0.493036	0.495395	950
KO	0.272203	0.274337	**0.270888**	870
WMT	0.272945	0.269190	**0.254820**	630
HPQ	0.276595	0.275459	0.280236	980
AXP	0.204162	0.211216	**0.195888**	230
DD	0.204220	0.202591	0.203177	330
JPM	0.238405	0.255824	**0.204777**	290
MRK	0.214294	0.211708	**0.196368**	850
DIS	0.324324	0.326105	**0.297895**	290
VZ	0.203936	0.206390	**0.196078**	490
HD	0.236339	0.237688	0.242606	990
T	0.235940	0.234002	**0.230230**	570
MSFT	0.332521	0.331694	0.334283	710
CSCO	0.227675	0.228196	0.233057	990
INTC	0.262545	0.264391	0.271204	1000
GE	0.168935	0.168258	0.168924	1000
PFE	0.200335	0.199163	0.204552	1000
BAC	0.299956	0.333396	**0.200134**	530
AA	0.249913	0.250263	**0.246067**	1000
XLF	0.300486	0.299604	0.301322	830
EWH	0.186584	0.188475	**0.180674**	260
EWG	0.156459	0.158946	**0.152092**	470
EWA	0.231424	0.233347	**0.224692**	960
XLV	0.140344	0.139772	**0.118340**	610
XLI	0.089607	0.087342	**0.086304**	450
XLU	0.107657	0.106929	**0.091362**	550
XLY	0.103068	0.102359	**0.095318**	590
XLB	0.105090	0.102791	**0.099280**	940
XLE	0.118998	0.117319	**0.111371**	1000
SPY	0.073800	0.071908	**0.067655**	230
EWS	0.223738	0.223721	0.228166	990
EWC	0.172954	0.174107	**0.164855**	330
EWU	0.146232	0.145031	0.146491	640
EWW	0.205101	0.203012	**0.199228**	80
EWM	0.249645	0.248694	0.257393	80
XLK	0.098086	0.098599	**0.097944**	440

Table I
MSE AND BEST L

The error was calculated considering the *mean squared error* (MSE) using the last 70 predictions made.

Table I also shows that the smaller error in our method was for the SPY stock, which is an expected result considering that the SPY value is constructed based on the main stocks of the markets.

A graphical comparison between AAR, RR and SLAAR with the target for stock SPY is shown in figures 1(a),1(b) and 1(c). It shows how AAR and our method fit better than RR to the target and also that our method is slightly better compared with AAR. It's also possible to notice that when returns change drastically none of the methods fit satisfactorily.

VI. Discussion and Conclusion

The existing approaches to solve regression problems are well known, however they are not used in an online context because they involve many calculations and in order to make a prediction, they consider all the data available. However, in some cases, only a portion of data is needed or available to make a prediction and sometimes the accuracy could be improved considering less data.

In this article, this historical dependence of data is studied and the results show that in some cases, if we use a sliding window of data instead of all the data, the MSE of the predictions is minimized. It's possible to check that there are some stocks which depend more on the historical data than others and this is valuable information for future algorithms.

On the other hand, RR and AAR calculates an inverse matrix at every step, which is computationally very expensive. In our proposal, this inverse matrix calculation is avoided and therefore it's more suitable to be used in an online context.

Acknowledgment

Partially supported by FONDECYT research grant 1100805.

References

[1] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks architectures," *Neural Computation*, vol. 7, pp. 219–269, 1995.

[2] T. Evgeniou, M. Pontil, and T. Poggio, "Regularization networks and support vector machines," in *Advances in Computational Mathematics*. MIT Press, 2000, pp. 1–50.

[3] F. Rosenblatt, "The Perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, pp. 386–408, 1958.

[4] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *Journal of Machine Learning Research*, vol. 7, pp. 551–585, MAR 2006.

[5] T. Zhang, "Solving large scale linear prediction problems using stochastic gradient descent algorithms," in *ICML 2004: Proceedings of the twenty-first international conference on Machine Learning. OMNIPRESS*, 2004, pp. 919–926.

[6] V. Vovk, "Competitive on-line statistics," *International Statistical Review*, vol. 69, p. 2001, 2001.

[7] N. Cesa-Bianchi, A. Conconi, and C. Gentile, "A second-order perceptron algorithm," *SIAM J. Comput.*, vol. 34, no. 3, pp. 640–668, 2005.

[8] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games*. New York, NY, USA: Cambridge University Press, 2006.