ADAPTIVE RANK FILTERING BASED ON ERROR MINIMIZATION

Bert de Vries

David Sarnoff Research Center
CN5300, Princeton, NJ 08543-5300
bdevries@sarnoff.com

ABSTRACT

A method for adaptive (on-line) pruning and constructing a (layered) computational network is introduced. The dimensions of the network are updated for every new available sample, which makes this technique highly suitable for tracking nonstationary sources. This method extends work on predictive least squares by Rissanen [1] and Wax [2] to an adaptive updating scheme. The algorithm is demonstrated by an application to adaptive prediction of exchange rates.

1. INTRODUCTION

For modeling of nonstationary signals, it is not appropriate to fix the model parameters after training on a "representative" data set. Here we present a technique which adaptively tracks the model dimensions. For ordered data such as a time series, Rissanen developed the *Predictive Least Squares* (PLS) framework, which extends the ordinary least squares criterion by including the estimation of the model order [1]. The PLS criterion makes no assumptions about the statistics of data or model nor does the result rely on asymptotically large datasets. These features make the PLS criterion highly suited for order tracking in non-stationary time-series such as speech or financial rates. Our work extends the PLS framework to adaptive updating and non-linear models such as neural networks.

2. PREDICTIVE LEAST SQUARES

Assume we have a time series $\{y(1), y(2), \ldots, y(n)\}$ for which we want to construct a linear autoregressive model $\hat{y}(t) = \underline{w}^T \underline{x}_t$, where $\underline{w}^T = [w(1), w(2), \ldots, w(p)]$ and $\underline{x}_t = [y(t-1), \ldots, y(t-p)]^T$. The least-squares approach dictates to minimize

$$\sum_{t=p+1}^{n} (y(t) - \underline{w}^T \underline{x}_t)^2 \qquad (1)$$

with respect to $\underline{w}$. We write the resulting least-squares estimate $\underline{w}_n = \left(\sum_{t=p+1}^{n} \underline{x}_t \underline{x}_t^T\right)^{-1} \sum_{t=p+1}^{n} \underline{x}_t y(t)$ with subscript n to indicate that we used information up to time step n in order to estimate $\underline{w}$. The least-squares residual $\sigma_n^2 = \sum_{t=p+1}^{n} (y(t) - \underline{w}_n^T \underline{x}_t)^2$ will not increase for increasing order p. Hence, the standard least-squares method does not estimate an appropriate model order p. Many extended approaches which include order estimation exist. For instance, information-based criteria minimize instead

$$\log \sigma_n^2 + c_n / n \qquad (2)$$

where $c_n = 2p$ identifies *Akaike's Information Criterion* (AIC) and $c_n = p \log n$ reduces to a form of the *Minimum Description Length* (MDL) principle (see e.g. [3] and references therein). The derivation of these criteria rely on asymptotically large n and certain assumptions about the Gaussianity of data or model. Hence, for many real-world processes, characterized by non-stationary (effectively small n) and non-Gaussian data (e.g. speech, biomedical and economic data), these criteria perform worse than desired.

For ordered data such as a time series, Rissanen [1] proposed to minimize instead

$$s_n^2 = \sum_{t=p+1}^{n} (y(t) - \underline{w}_{t-1}^T \underline{x}_t)^2 \qquad (3)$$

where $\underline{w}_{t-1}$ is now the least-squares estimate based on $\{y(i), i \le t-1\}$. The residuals $e(t) = y(t) - \underline{w}_{t-1}^T \underline{x}_t$ are sometimes called "*honest*" or "*true*" prediction errors to emphasize that only information from the past is used at any time. This in contrast to the residuals $y(t) - \underline{w}_n^T \underline{x}_t$, which have used information from both the past and future.

Minimization of (3) is called the *Predictive Least Squares* (PLS) method. The PLS criterion, in contrast to standard LS, does include order estimation as it minimizes for the order that worked best in the past. The optimal order is determined by $p_{PLS} = \arg min_p(s_n^2)$. In fact, it can be shown that PLS delivers an *asymptotically consistent* estimate of model order. An important feature of PLS is that there is no large data record assumption and hence the criterion is very attractive for small data sets and non-stationary signals. In this paper we report work on extending the PLS principle to track the order in a non-stationary environment and to non-linear computational (neural) networks.

3. PREDICTIVE ADAPTIVE RANK FILTERING

Consider the linear filter given by

$$y(t) = \underline{w}_{t-1}^T \underline{x}_t = \sum_{i=1}^{p} w_i(t-1)x_i(t) . \quad (4)$$

If we define a quadratic cost function

$$\varepsilon(t) = \lambda\varepsilon(t-1) + e^2(t) \quad , \quad (5)$$

where $e(t) = z(t) - y(t)$ are "true" prediction errors, $z(t)$ is a target signal, and λ a forgetting factor ($0 \le \lambda \le 1$); then it is easy to derive the error gradient $\partial\varepsilon(t)/\partial\underline{w}$, which can be used for on-line adaptation of the weights. We would like to derive a similar gradient for the filter length *p*, but this is not possible since *p* is an integer. However, we can use a local search algorithm which updates the filter length to track minimal "true" prediction cost.

Let us first treat the case where $x_i(t) = x(t-i+1)$, a transversal filter. We will call the optimal filter length $r(t)$ the *rank* of the filter. If we make the reasonable assumption that the rank of the model does not change faster than one dimension per time step, it is sufficient to keep track of the "true" prediction cost traces $\varepsilon_r(t)$ (subscript indicates filter order) for filter lengths $r(t) - 1$, $r(t)$, and $r(t) + 1$. Hence, the total filter length *p(t)* is always one tap larger than the rank. The number of taps is then updated to the value that minimizes the prediction cost trace:

$$r(t+1) = \arg min_r[\varepsilon_r(t)] \quad , \quad (6)$$

where *r* is taken over the range $r(t) - 1 \le r \le r(t) + 1$. If $r(t+1) = r(t) + 1$, we update the new filter length $p(t) = r(t+1) + 1$, and add a new weight $w_{p(t+1)}(t+1) = 0$ to the weight vector. Also, we initialize $\varepsilon_{p(t+1)}(t) = \varepsilon_{p(t+1)-1}(t)$. If $r(t+1) = r(t) - 1$, the filter length is reduced by one tap.

The just described strategy is simple and would possibly work well, but it can be improved at little extra cost. First, we have ignored that the common weights for optimally adapted transversal filters of different lengths have only the same values if the tap signals are mutually uncorrelated. On-line decorrelation of the tap signals can be achieved by using *lattice filters* rather than tapped delay lines. The *Least-Squares Lattice* (LSL) predictor was used to efficiently compute PLS by Wax ([2], although no rank tracking was treated here). The *Gradient Adaptive Lattice* (GAL) is computationally simpler than the LSL filter and performs on par in tracking a non-stationary signal [4]. It can be shown that the backward prediction errors in the GAL filter are mutually uncorrelated and are therefore highly suited as filter taps for the just described on-line rank adaptation method. Of course, a LSL filter (which enjoys faster initial convergence than a GAL filter) can be used as well.

Adaptive filters are prone to gradient measurement errors as well as "lag noise" due to shifting of the optimal weights in a non-stationary environment. The amount of excess mean squared error relative to the optimal residual is called *misadjustment* (e.g. Haykin, [4] p.708). As a result, the prediction cost traces $\varepsilon_r(t)$ are stochastic (noisy) signals and the adaptive rank algorithm (6) randomly shifts rank in proportional amount to the filter misadjustment. In other words, algorithm (6) transports the filter misadjustment to a rank noise or *rank misadjustment*. An improved adaptive rank algorithm would "guard" the rank adaptation against such misadjustment by making it harder to change ranks (if not from a physical viewpoint, then certainly from a computational cost viewpoint we like to minimize changing ranks due to noise). Thus, for a non-stationary environment, we only switch rank if

$$\varepsilon_{r(t)\pm1}(t) < \varepsilon_{r(t)}(t)[1-\delta] \quad , \quad (7)$$

where δ is a small "guarding" threshold, which should be proportional to the misadjustment. Since the estimation formulae for adaptive filter misadjustment are rather complex (Haykin [4], ch.16), it may be best to estimate the misadjustment from monitoring the variance of $\varepsilon_r(t)$ or set δ to a fixed value.
In analogy with the PLS method, we "predict" the rank based on what worked best for the past, and hence we will call this method *Predictive Adaptive Rank* (PAR) filtering. As an example, a GAL-PAR predictor implementation follows in Figure 1.

```
for t=1,2,...,n
    % UPDATE GAL PREDICTOR
    for i=1,2,...,r(t)+1 % go one beyond optimal order
        $f_i(t) = f_{i-1}(t) + \gamma_i(t) b_{i-1}(t-1)$ % forw. PE
        $b_i(t) = b_{i-1}(t-1) + \gamma_i(t) f_{i-1}(t)$ % backw. PE
        $\gamma_i(t+1) = \gamma_i(t) - \mu f_i(t) b_{i-1}(t-1)$ % refl coeff.
        $\varepsilon_i(t) = \lambda \varepsilon_i(t-1) + f_i^2(t)$ % pred. cost trace
    end%for i
    % NOW PRED. RANK ADAPTATION
    if $\varepsilon_{r(t)+1}(t) < \varepsilon_{r(t)}(t)[1-\delta]$ then
        $r(t+1) = r(t) + 1$ % increase rank
        $\gamma_{r(t+1)+1}(t) = 0$ % add new weight
        $\varepsilon_{r(t+1)+1}(t) = \varepsilon_{r(t+1)}(t)$ % init pred. cost
    elseif $\varepsilon_{r(t)-1}(t) < \varepsilon_{r(t)}(t)[1-\delta]$ then
        $r(t+1) = r(t) - 1$ % decrease rank
    else, $r(t+1) = r(t)$ % no rank change
    end%if
end%for t
```

Figure 1. pseudo-code for Gradient Adaptive Lattice--Predictive Adaptive Rank (GAL-PAR) predictor.

4. PREDICTIVE ADAPTIVE RANK ARRAY FILTERING

For array filtering problems or in multilayer networks, layers do not consist of time-shifted data only. In order to apply the adaptive rank algorithm, we need to additionally compute an *ordering* among the taps (nodes in neural net jargon) of $\underline{x}_t$ in order to decide when and which nodes to add or prune. Moreover, for added performance, it is beneficial to *decorrelate* the tap signals on-line. In a linear array filtering context, Yang et al. [5] proposed an adaptive *Principal Components Analysis* (PCA) algorithm followed by an MDL complexity measure of type (2) to track the rank. Adaptive principal components algorithms recursively update the eigenvectors and eigenvalues of the auto-correlation matrix of a multi-dimensional signal $\underline{u}_t$. The matrix of eigenvectors V_t can be used to decorrelate $\underline{u}_t$ by $\underline{x}_t = V_t^T \underline{u}_t$, where the elements in $\underline{x}_t$ are ordered along decreasing eigenvalues. The PCA-PAR algorithm, then, follows by predictive adaptive rank updating on $\underline{x}_t$. A large number of PCA tracking algorithms have been presented in various literatures. In our simulations we used the PASTd algorithm, a deflation algorithm which extracts the principal components sequentially (Yang et al., [5]).

There are a number of very attractive properties of the PCA-PAR algorithm relative to alternative methods for on-line construction and pruning of networks. In particular, we mention:

• The PCA-PAR algorithm smoothly constructs and prunes within the same framework. Many alternative on-line structuring algorithms such as the *Resource Allocating Network* (Platt, 1991) are restricted to network construction and cannot track downsizing of the model [6].
• The transformation of the input to its principal components space leads to the applicability of the self-orthogonalizing adaptive filtering algorithm for the weight matrix ($\underline{w}_t = \underline{w}_{t-1} + \eta D_t^{-1} \underline{x}_t e_t$, where D_t is the diagonal matrix of eigenvalues), which converges much faster than regular LMS or backpropagation-like updating [4].
• Furthermore, PAR algorithms, in contrast to algorithms that add complexity terms to the network cost, directly minimize the network cost, which is (for properly defined cost function) the ultimate goal of the computation.

As a computationally cheaper alternative to PCA tracking, we could use triangular matrices based on QR-decomposition to decorrelate $\underline{u}_t$. This road leads to algorithms such as Gram-Schmidt-PAR or Givens rotation followed by PAR. In general, in order to apply PAR, $\underline{x}_t$ needs to be ordered by some criterion and is preferably mutually uncorrelated.

5. APPLICATION TO FOREIGN EXCHANGE RATE PREDICTION

The PAR algorithm can be applied to various adaptive filtering protocols such as identification, inverse modeling, prediction, and interference canceling. We present here an application to adaptive *Non-linear AutoRegressive* (NAR) prediction of a non-stationary signal.

Consider the network shown in Figure 2.A, which aims to predict the time series $z(t)$ from its own past $\underline{x}_t = [z(t-1), z(t-2), \ldots, z(t-p)]^T$. The network separates the prediction problem into a linear part (by $\underline{w}$) and a non-linear prediction task (by $\underline{v}$). The linear part is estimated by $e_l(t) = z(t) - \underline{w}_{t-1}^T \underline{x}_t$ and LMS adaptation $\underline{w}_t = \underline{w}_{t-1} + \eta \underline{x}_t e_l(t)$. The residual $e_l(t)$ is due to misalignment of the adaptation algorithm, noise and nonlinear dependencies. The nonlinear part in $e_l(t)$ is subsequently estimated by a high dimensional *Random Representation* (RR) network: the *p*-dimensional input signal $\underline{x}_t$ is transformed by the nonlinear map $\underline{y}_t = \sigma(A\underline{x}_t)$ where *A* is a constant random *mxp* weight matrix $(m > p)$ and $\sigma(\ )$ a scalar nonlinear function. Next, the total prediction error $e(t)$ is estimated by $e(t) = e_l(t) - \underline{v}_{t-1}^T\underline{y}_t$ and $\underline{v}_t = \underline{v}_{t-1} + \eta\underline{y}_t e(t)$. Note that both the linear weights $\underline{w}$ as well as the weights $\underline{v}$ corresponding to the non-linear map can be updated by LMS without backpropagation, thus permitting fast tracking of non-stationarities (see Sutton [7], for a further discussion of the RR network). We compare the performance of this network to the case where PCA-PAR is applied to $\underline{y}_t$, the outputs of the nonlinear layer.

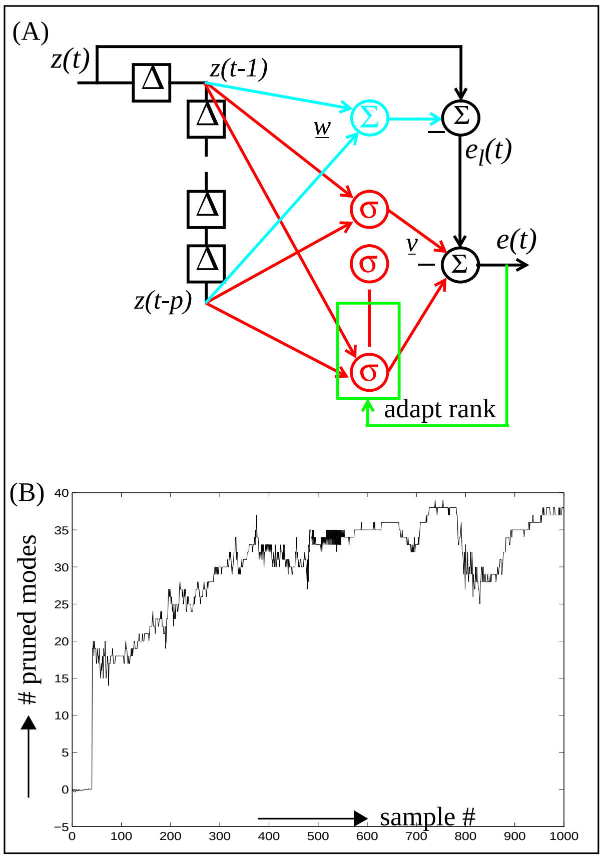

Figure 2. A) Network with PAR filtering in non-stationary time series prediction context. (B) The rank of pruned eigenspace for the nonlinear layer as a function of sample number for non-linear AR prediction of SFR/USD exchange rates.

This network configuration was applied to the problem of foreign exchange rate prediction. We selected 1,000 tick-by-tick samples (about one day) from May 1985 of the U.S. dollar versus Swiss Franc exchange rate. We used 40 '*tanh*' hidden nodes, LMS weight updating with constant learning rates η=0.01, and forgetting factor λ=0.98. The adaptive network without rank adaptation correctly predicted 57% of the directions $sign(z(t) - z(t-1))$. When PCA-PAR was added to the outputs of the non-linear layer, 72% of the direction of movements was correctly predicted. In Figure 2.B the rank of the pruned eigenspace (40-*r*(*t*)) versus sample number is shown. Note that PAR filtering serves both as an on-line construction and pruning algorithm. Also, the number of actually used nodes in eigenspace is much smaller than the number of pruned nodes (an average of about 10 out of 40). Since we use a deflation technique for PCA updating we only track the modes (plus one) that are used in the computation of the output. Clearly, this leads to tremendous savings over a symmetric PCA tracking algorithm.

CONCLUSIONS

An adaptive rank updating algorithm based on the principles of predictive least-squares was presented. This method can be applied both to time-shifted data vectors and multichannel filtering problems. In a neural network context, the proposed algorithm can be identified as an on-line pruning and construction algorithm.

REFERENCES

[1] J. Rissanen, A Predictive Least Squares Principle, *IMA Journal of Mathematical Control and Information*, vol.3, no.2-3, 211-222, 1986.
[2] M. Wax, Order Selection for AR Models by Predictive Least Squares, *IEEE Tr. Acoustics, Speech and Signal Pr.*, vol.36, no.4, 581-588, 1988.
[3] M. Wax and T. Kailath, Detection of Signals by Information Theoretic Criteria, *IEEE Tr. on ASSP 33(2)*, pp. 387-392, 1985.
[4] S. Haykin, *Adaptive Filter Theory 3rd ed.*, Prentice-Hall, 1996.
[5] B. Yang and F. Gersemsky, An Adaptive Algorithm of Linear Computational complexity for both rank and subspace tracking, *ICASSP'94*, pp. IV-33-36, 1994.
[6]J.C. Platt, A Resource-Allocating Network for Function Interpolation, *Neural Computation 3(2)*, pp. 213-225, 1991.
[7] R.S. Sutton and S.D. Whitehead, Online Learning with Random Representations, *Proc. 10th Int. Conf. on Machine Learning*, 314-321, 1993.