Adaptive Neural Nets Filter Using a Recursive Levenberg-Marquardt Search Direction

Lester S.H. Ngia, Jonas Sjöberg and Mats Viberg
Department of Signals and Systems
Chalmers University of Technology
412 96 Gothenburg, Sweden
ngia@s2.chalmers.se

Abstract

This paper proposes a recursive Levenberg-Marquardt (LM) search direction as the training algorithm for non-linear adaptive filters, which use multi-layer feed forward neural nets as the filter structures. The neural nets can be considered as a class of non-linear adaptive filters with transversal or recursive filter structures. In the off-line training, the LM method is regarded as an intermediate method between the steepest descent (SD) and Gauss-Newton (GN) methods, and it has better convergence properties than the other two methods. In the echo cancellation experiments, the recursive LM algorithm converges faster and gives higher echo return loss enhancement (ERLE) than the recursive SD and GN algorithms.

1 Introduction

Neural nets can be trained by batch algorithms, eg. [3, 8, 10], where the nets are initially trained or estimated with a finite number of data samples and subsequently used with the fixed estimated parameters. They can also be trained by recursive algorithms, where a new data sample is available in each training iteration, eg. [2, 7].

This paper proposes a recursive algorithm, that is based on a LM search direction, to train neural nets for non-linear adaptive filtering. This is motivated by the fact that it is often the best method in off-line training of neural nets [3, 10]. There are several approaches to describe recursive training algorithms [6]. The chosen approach is to derive the proposed recursive algorithm from the off-line training algorithms. The selected technique is similar to the one called *Recursive Maximum Likelihood* algorithm in [6].

The applications of neural nets as adaptive filters are not new. For example, adaptive neural net filters are trained by a variant of the recursive SD algorithm in [7] and a recursive GN algorithm in [2]. To the authors' knowledge, the only work that is close to this paper is found in [4]. However, to reduce the computation complexity, the recursive algorithm does not use the off-diagonal elements (second-order information) of the Hessian matrix. It is also not suitable to estimate time-varying system. On the contrary, the proposed recursive Levenberg-Marquardt algorithm can estimate time-varying system and uses the second-order information without much increase in computation complexity.

The neural net filter structure is given in Sec. 2 and the proposed recursive LM algorithm is explained in Sec. 3. The new non-linear adaptive filter is applied as a network hybrid echo canceller in the experiments in Sec. 4. The data are echo of two hybrid paths that are recorded at a mobile switching center in Sweden [1]. The conclusions are provided in Sec. 5.

2 Filter Structures

In the case of a linear filter, the predictor can be expressed as

$$\hat{y}(t,\theta) = \theta^T \varphi(t) \tag{1}$$

where $\hat{y}(t,\theta)$ is an estimated output, $\varphi(t)$ is a regressor vector and θ is a parameter vector.

This paper investigates the case of a non-linear filter:

$$\hat{y}(t,\theta) = g(\varphi(t),\theta) \tag{2}$$

where the memoryless function $g(\varphi(t),\theta)$ can be chosen to be a feed-forward or a radial basis function neural net. However, the presented theory is generally valid for any smooth function $g(\cdot,\cdot)$.

Depending on the choice of regressor one can obtain the

[1] The data is kindly contributed by Ericsson Radio Systems AB, Sweden.

following standard models:

$$\varphi(t) = [u(t-n_k), \cdots, u(t-n_k-n_b)]^T \quad \text{FIR} \quad (3a)$$

$$\varphi(t) = [-y(t-1), \cdots, -y(t-n_a)]^T \quad \text{AR} \quad (3b)$$

$$\varphi(t) = [-y(t-1), \cdots, -y(t-n_a) \cdots u(t-n_k), \cdots, u(t-n_k-n_b)]^T \quad \text{ARX} \quad (3c)$$

where $u(t)$ and $y(t)$ are the observed input-output pairs of a dynamic system with a delay of n_k and orders of n_a, n_b. The linear filter structures of models in (3) are called as linear FIR, AR and ARX models respectively [5, 6]. Recursive estimation of such models are extensively covered in eg. [6]. The non-linear filter structures of models in (3) are named as NFIR, NAR and NARX models respectively [9]. The derivation of recursive algorithms for these models is given in Sec. 3.

Mainly for clarity reason, the derivation is constrained to the single input single output (SISO) case. It is straightforward to generalize the result to the multi-input multi-output (MIMO) case. Fig. 1 shows a SISO NARX model where the nonlinear dynamics consist of a feed-forward neural net with sigmoidal hidden neuron units.

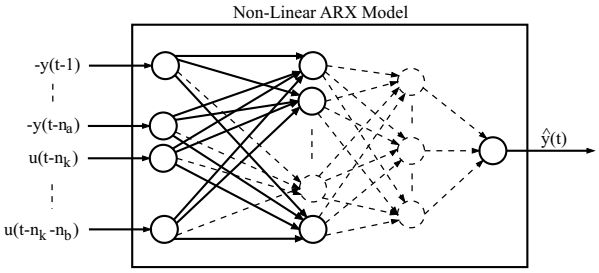

Figure 1. The SISO nonlinear ARX model.

3 Recursive Algorithms

A brief description of batch algorithms for off-line training is given to define the concepts to describe the recursive algorithms for on-line training. A thorough explanation is provided in eg. [1, 10].

Given a fixed set of data $[y(t)\ u(t)]_{\{t=1\}}^N$, the following mean squared error criterion is used:

$$V_N(\theta) = \frac{1}{2N} \sum_{t=1}^{N} \varepsilon^2(t,\theta) \quad (4)$$

where $\varepsilon(t,\theta) = y(t) - \hat{y}(t,\theta)$.

Starting at an initial estimate $\hat{\theta}_N^{(0)}$, many off-line algorithms can be described by the iteration of

$$\hat{\theta}_N^{(i+1)} = \hat{\theta}_N^{(i)} - \mu_N^{(i)} [R_N^{(i)}]^{-1} V_N'(\hat{\theta}_N^{(i)}) \quad (5)$$

where $\hat{\theta}_N^{(i)}$ is the parameter estimate at iteration i, $R_N^{(i)}$ is a matrix which modifies the local search direction defined by the gradient $V_N'(\hat{\theta}_N^{(i)})$ and $\mu_N^{(i)}$ is a step-length to assure that $V_N(\hat{\theta}_N^{(i)})$ decreases in every iteration.

Following are some common off-line algorithms which are described by (5):

- *Steepest descent*: Take

$$R_N^{(i)} = I \quad (6)$$

If $\mu_N^{(i)}$ is constant we have the traditional back-propagation error algorithm [8]. The convergence rate is slow in an ill-conditioned problem, which is common in neural nets minimization.

- *Gauss-Newton*: Take

$$R_N^{(i)} = \frac{1}{N} \sum_{t=1}^{N} \psi(t,\hat{\theta}^{(i)}) \psi(t,\hat{\theta}^{(i)})^T \quad (7)$$

where

$$\psi(t,\theta) = \frac{d}{d\theta} g(\varphi(t),\theta) \quad (8)$$

The term $R_N^{(i)}$ is guaranteed to be positive semi-definite.

- *Levenberg-Marquardt*: Set $\mu_N^{(i)} = 1$ and use

$$R_N^{(i)} = \frac{1}{N} \sum_{t=1}^{N} \psi(t,\hat{\theta}^{(i)}) \psi(t,\hat{\theta}^{(i)})^T + \delta^{(i)} I \quad (9)$$

Now $\delta^{(i)} \geq 0$ is used instead of $\mu_N^{(i)}$ to assure a criterion decrease in each iteration. Choosing $\delta^{(i)} = 0$ gives a full Gauss-Newton step which is the largest possible update with the algorithm. Taking $\delta^{(i)} \to \infty$ will cause the diagonal elements of $R_N^{(i)}$ to dominate and thus lead to a steepest descent direction with a short step length.

Now, consider the error criterion with an exponential forgetting mechanism at time t:

$$V_t(\theta) = \frac{1}{2} \sum_{\tau=1}^{t} \lambda^{t-\tau} \varepsilon^2(\tau,\theta) \quad (10)$$

where $0 < \lambda \leq 1$ is the *forgetting factor* [2] So, the old measurements in the criterion are exponentially discounted and this is important to track time-varying systems.

The following derivation of a general recursive algorithm is close to the one applied in the *Recursive Maximum Likelihood* algorithm in [6], which is used to obtain a recursive

[2] Typically the choice of λ is $0.95 \leq \lambda \leq 1$.

algorithm for a pseudo-linear regression model. The model is expressed as $\hat{y}(t,\theta) = \theta^T \varphi(t,\theta)$.

Let $\hat{\theta}(t-1)$ be the estimate at time $t-1$ and assume that it minimizes $V_{t-1}(\theta)$. The goal is to compute $\hat{\theta}(t)$. Consider a Taylor expansion on $V_t(\theta)$ at the current estimate:

$$V_t(\theta) = V_t(\hat{\theta}(t-1)) + V_t'(\hat{\theta}(t-1))[\theta - \hat{\theta}(t-1)] + \frac{1}{2}[\theta - \hat{\theta}(t-1)]^T V_t''(\hat{\theta}(t-1))[\theta - \hat{\theta}(t-1)] + o(|\theta - \hat{\theta}(t-1)|^2) \quad (11)$$

Neglecting the remainder $o(|\theta - \hat{\theta}(t-1)|^2)$, the parameter update can be chosen as the minimizer of (11):

$$\hat{\theta}(t) = \hat{\theta}(t-1) - [V_t''(\hat{\theta}(t-1))]^{-1} V_t'(\hat{\theta}(t-1))^T \quad (12)$$

Differentiating (10) with respect θ gives

$$\begin{aligned} V_t'(\theta)^T &= -\sum_{\tau=1}^{t} \lambda^{t-\tau} \psi(\tau,\theta)\varepsilon(\tau,\theta) \\ &= \lambda V_{t-1}'(\theta)^T - \psi(t,\theta)\varepsilon(t,\theta) \end{aligned} \quad (13)$$

and differentiating once more:

$$V_t''(\theta) = \lambda V_{t-1}''(\theta)^T + \psi(t,\theta)\psi(t,\theta)^T + \varepsilon''(t,\theta)\varepsilon(t,\theta) \quad (14)$$

According to the assumption that $\hat{\theta}(t-1)$ is indeed the optimal estimate at time $t-1$, then $V_{t-1}'(\hat{\theta}(t-1)) = 0$. Use an approximation that $\varepsilon''(t,\hat{\theta}(t-1))\varepsilon(t,\hat{\theta}(t-1)) = 0$. This approximation should be good when the estimate $\hat{\theta}(t-1)$ is close to its true value, since then $\varepsilon(t,\hat{\theta}(t-1))$ is almost white and uncorrelated with $\varepsilon''(t,\hat{\theta}(t-1))$.

Thus, (12) can be re-written as

$$\hat{\theta}(t) = \hat{\theta}(t-1) + H(t)^{-1}\psi(t,\hat{\theta}(t-1))\varepsilon(t,\hat{\theta}(t-1)) \quad (15a)$$

$$H(t) = \lambda H(t-1) + \psi(t,\hat{\theta}(t-1))\psi(t,\hat{\theta}(t-1))^T \quad (15b)$$

Now the recursive versions of the steepest-descent, Gauss-Newton and Levenberg-Marquardt algorithms can be defined by comparing the off-line algorithm in (5) and (15a). Update the parameters according to

$$\hat{\theta}(t+1) = \hat{\theta}(t) + \mu(t) R(t)^{-1}\psi(t,\theta)\varepsilon(t,\theta) \quad (16)$$

where $\mu(t)$ is a gain at time t. Select $R(t)$ and $\mu(t)$ as follows:

- *Steepest descent*: Choose $R(t) = I$ to provide

$$\hat{\theta}(t+1) = \hat{\theta}(t) + \mu(t)\psi(t,\theta)\varepsilon(t,\theta) \quad (17)$$

and $\mu(t)$ can be a normalized gain at time t, ie. $\mu(t) = \mu/|\varphi(t)|^2$, where μ is a constant gain.

- *Gauss-Newton*: Choose $R(t) = (1-\lambda)H(t)$ to normalize $H(t)$ in (15b) and $\mu(t) = (1-\lambda)$ to give

$$\hat{\theta}(t+1) = \hat{\theta}(t) + (1-\lambda)R(t)^{-1}\psi(t,\theta)\varepsilon(t,\theta) \quad (18a)$$

$$R(t) = \lambda R(t-1) + (1-\lambda)\psi(t,\theta)\psi(t,\theta)^T \quad (18b)$$

- *Levenberg-Marquardt*: Choose $R(t) = (1-\lambda)H(t) + \delta I$ and $\mu(t) = (1-\lambda)$ to yield

$$\hat{\theta}(t+1) = \hat{\theta}(t) + (1-\lambda)R(t)^{-1}\psi(t,\theta)\varepsilon(t,\theta) \quad (19a)$$

$$R(t) = \lambda R(t-1) + (1-\lambda)[\psi(t,\theta)\psi(t,\theta)^T + \delta I] \quad (19b)$$

3.1 Avoiding the matrix inversion

A direct solution of (16) is not practical for the recursive Gauss-Newton and Levenberg-Marquardt algorithms because the inversion of $R(t)$ requires a computation complexity of $O(d^3)$ (d=dim θ). Using the *matrix inversion lemma* can reduce the computation complexity from $O(d^3)$ to $O(d^2)$. The *lemma* expresses the following inversion:

$$[A + BCD]^{-1} = A^{-1} - A^{-1}B[DA^{-1}B + C^{-1}]^{-1}DA^{-1} \quad (20)$$

For the recursive Gauss-Newton algorithm, choose $A = \lambda R(t-1)$, $B = (1-\lambda)\psi(t,\theta)$, $C = 1$ and $D = \psi(t,\theta)^T$. Introduce $P(t) = (1-\lambda)R(t)^{-1}$ and the recursive Gauss-Newton algorithm in (18) can be re-written as

$$\hat{\theta}(t+1) = \hat{\theta}(t) + P(t)\psi(t,\theta)\varepsilon(t,\theta) \quad (21a)$$

$$P(t) = \{P(t-1) - \frac{P(t-1)\psi(t,\theta)\psi(t,\theta)^T P(t-1)}{\lambda + \psi(t,\theta)^T P(t-1)\psi(t,\theta)}\}/\lambda \quad (21b)$$

Now, the recursion (21) contain an inversion of a matrix of only dimension 1 because the matrix $[DA^{-1}B + C^{-1}] = 1 + \lambda^{-1}\psi(t,\theta)^T P(t-1)\psi(t,\theta)$ has rank 1. The reason is that the updating quantity $(1-\lambda)\psi(t,\theta)\psi(t,\theta)^T$ in (18b) is of rank 1.

Unfortunately, the *lemma* cannot be used directly on the recursive Levenberg-Marquardt algorithm in (19) since the updating quantity in (19b) is of rank d. A possible remedy to this problem is to add δd to $\psi(t,\theta)\psi(t,\theta)^T$ at one diagonal element a time, as suggested in [6]. Then (19b) is approximated by

$$R(t) = \lambda R(t-1) + (1-\lambda)[\psi(t,\theta)\psi(t,\theta)^T + \delta d Z_d(t)] \quad (22)$$

where $Z_d(t)$ is a $d \times d$ zeroes matrix except one of its diagonal element at $(t \mod d) + 1$ is 1. After an elapse of d time samples, (22) is virtually the same as (19b).

The expression (22) can be re-written as:

$$R(t) = \lambda R(t-1) + (1-\lambda)[\psi^*(t,\theta)\Lambda^*(t)^{-1}\psi^*(t,\theta)]^T \quad (23)$$

where $\psi^*(t,\theta)$ is now a dx2 matrix with the position of element 1 depends on $(t \mod d)+1$:

$$\psi^*(t,\theta) = \begin{pmatrix} 0 & \cdots & \psi(t,\theta)^T & 1 & \cdots & 0 \end{pmatrix}^T \quad (24)$$

$$\Uparrow$$

$$\text{position}=(t \mod d)+1$$

and the weighting matrix is

$$\Lambda^*(t)^{-1} = \begin{pmatrix} 1 & 0 \\ 0 & \delta d \end{pmatrix} \quad (25)$$

The *lemma* can now be applied to (23) with $A = \lambda R(t-1)$, $B = (1-\lambda)\psi^*(t,\theta)$, $C = \Lambda^*(t)^{-1}$ and $D = \psi^*(t,\theta)^T$. Using the substitution $P(t) = (1-\lambda)R^{-1}(t)$ will result the following recursive Levenberg-Marquardt algorithm to replace (19):

$$\hat{\theta}(t+1) = \hat{\theta}(t) + P(t)\psi(t,\theta)\varepsilon(t,\theta) \quad (26a)$$

$$P(t) = \{P(t-1) - P(t-1)\psi^*(t,\theta) S(t)^{-1}\psi^*(t,\theta)^T P(t-1)\}/\lambda \quad (26b)$$

$$S(t) = \psi^*(t,\theta)^T P(t-1)\psi^*(t,\theta) + \lambda\Lambda^*(t) \quad (26c)$$

The dimension of $S(t)$ is now only 2 instead of d, so that a much smaller matrix needs to be inverted every time.

The only structure dependent quantity in (17),(21) and (26) is the gradient vector $\psi(t,\theta)$[3]. It is straight-forward to derive $\psi(t,\theta)$ by using the same technique as in the back-propagation error algorithm, eg. [8].

4 Real Data Experiments

Two hybrid echoes, namely Hybrid 1 and 2, are recorded at a mobile switching center and the near-end equipment are ordinary fixed telephones. The input signals are white Gaussian noise sequences and the sampling time is 8KHz.

The following recursive algorithms are applied to the same NFIR filter structure (3a), which are then used as echo cancellers (ECs):

- *Recursive steepest-descent algorithm (RecSD)*: The algorithm suggested in [7] is used, which is a variant of (17). It has a rectangular sliding window of length N_c and K_n iterations of gradient computation between time t and $t+1$. The use of N_c and K_n helps to improve the tracking and convergence rate at the expense of increased computational complexity. If $N_c = K_n = 1$, then the suggested algorithm is the same as the one in (17).
- *Recursive Gauss-Newton algorithm (RecGN)*: It uses the algorithm in (21), which is the same as a recursive linearized least squares algorithm in [2].
- *Recursive Levenberg-Marquardt algorithm (RecLM)*: It applies the algorithms described in (26).

To demonstrate the degree of nonlinearity in the network echo channel, a linear FIR filter is also compared. It has a linear regression structure as in (1) and uses the same $\varphi(t)$ of (3a). The filter parameters are estimated using the NLMS algorithm with $\mu = 0.5$, which is similar to (17) except that $\psi(t) = \varphi(t)$ in the linear FIR filter.

Table 1 compares the number of flops required to implement the RecSD, RecGN and RecLM algorithms. Note that the number of flops does not include the computation of $\psi(t,\theta)$. To simplify the analysis, the values for n_k and n_b are pre-determined to be $n_k = 203, n_b = 25$ and $n_k = 41, n_b = 65$ for the respective Hybrid 1 and 2.

	RecSD	RecGN	RecLM
Flops	$d(4N_c+2)K_n$	$5.5d(d+1)$	$d(6.5d+12.5)$

Table 1. Comparison of the number of flops per recursion to implement each algorithm.

The values of user-design variables in RecSD, RecGN and RecLM are shown in Table 2. Only the values for N_c and K_n are different in Hybrid 1 and 2. They use 1 hidden layer with 5 (4) hidden neurons in Hybrid 1 (Hybrid 2) as a tradeoff between the achieved ERLE and over-fitting due to too many *spurious* parameters. In this configuration, the computational complexities of RecSD is similar to RecLM, and RecLM needs 15% (Hybrid 1) and 18% (Hybrid 2) more flops than RecGN. The linear FIR filter has the lowest number of parameters with d equals to the predetermined value of n_b, after considering that the input signal is delayed by n_k. All algorithms have initial parameter estimate $\hat{\theta}(0)$ equal to uniformly distributed random values in $[-0.5, 0.5]$.

The figure of merit for the effectiveness of an EC is the echo return loss enhancement (ERLE) which is defined as:

$$ERLE = 10log_{10}\frac{E[y(t)^2]}{E[\varepsilon(t,\theta)^2]} \quad (27)$$

Fig. 2 shows that among the algorithms, the linear FIR has the lowest ERLE because the echo path is nonlinear and it converges the fastest because it uses the least number of parameters. RecSD has the least ERLE improvement of $\approx$ 3dB and RecLM has the highest ERLE gain

[3]The computation of the vector $\psi(t,\theta)$ is the same for all search direction methods.

RecSD		RecGN	RecLM
$\mu = 1$		$P(0) = 1.10^4$	$P(0) = 1.10^4$
Hybrid 1	$N_c = 6$	$\lambda = 0.999$	$\lambda = 0.999$
	$K_n = 30$		$\delta = 10^{-5}$
Hybrid 2	$N_c = 10$		
	$K_n = 40$		

Table 2. The values of design variables in each algorithm

of $\approx$ 8dB over the linear filter. RecLM converges $\approx$ 0.3s faster and has an ERLE of $\approx$ 2dB higher than RecGN. A similar trend of results are also obtained for Hybrid 2 as illustrated in Fig. 3, except that Hybrid 2 is quite linear. Thus, RecLM has only an ERLE of $\approx$ 2dB and $\approx$ 0.5dB higher than the linear filter and RecGN respectively. To an extent, the results show the superior efficiency of the recursive Levenberg-Marquardt algorithm.

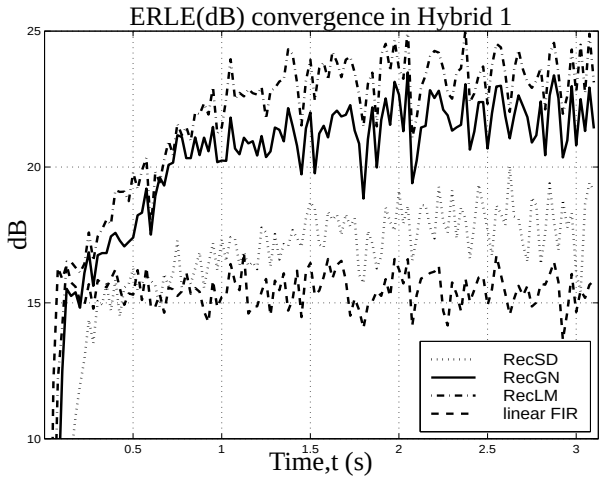

Figure 2. The comparison of ERLE convergence in Hybrid 1 ($\mathrm{d} = 136$).

5 Conclusion

This article presents a recursive LM algorithm that is used to adaptively train a non-linear ARX filter with a feed forward neural net dynamics. The recursive LM algorithm is compared with the recursive SD and GN algorithms using two real hybrid echo data. In the experiment, all algorithms use the same filter structures. The results indicate that the recursive LM algorithm is a more efficient algorithm for adapting highly non-linear systems than the recursive SD and GN algorithms, similarly with their corresponding off-line algorithms

Although the recursive LM algorithm consumes a higher number of flops than the other two algorithms, it is compensated abundantly by the increased efficiency. The high

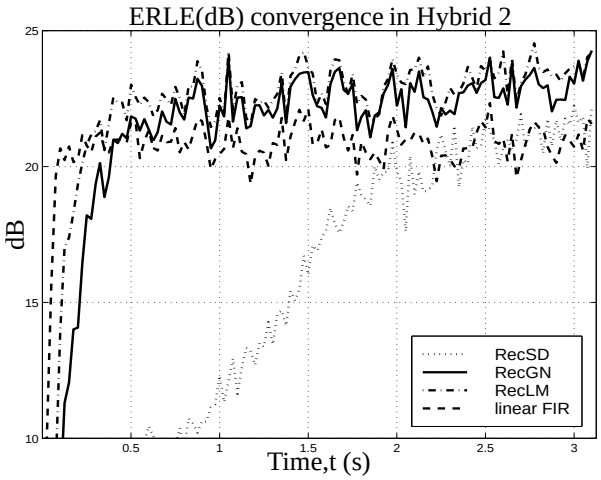

Figure 3. The comparison of ERLE convergence in Hybrid 1 ($\mathrm{d} = 269$).

efficiency is also unmatched by the recursive SD algorithm even when its computational complexity equals to the one of the recursive LM algorithm.

References

[1] J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, NJ, 1983.

[2] S. C. Douglas and T. H. Meng. Linearized least-squares training of multilayer feedforward neural networks. In *Int. Joint Conf. of Neural Networks*, pages 307–312, 1991.

[3] M. Hagan and M. Menhaj. Training feedforward networks with the Marquardt algorithm. *IEEE Trans. on Neural Networks*, 5(6):989–993, Nov. 1994.

[4] S. Kollias and D. Anastassiou. An adaptive least squares algorithm for the efficient training of artificial neural networks. *IEEE Trans. on Circuits and Systems*, 36(8):1092–1101, Aug. 1989.

[5] L. Ljung. *System Identification: Theory for the User*. Prentice-Hall, Englewood Cliffs, NJ, 1987.

[6] L. Ljung and T. Söderström. *Theory and Practice of Recursive Identification*. M.I.T. Press, Cambridge, MA, 1983.

[7] O. Nerrand, P. Roussel-Ragot, L. Personnaz, G. Drefys, and S. Marcos. Neural networks and nonlinear adaptive filtering: Unifying concepts and new algorithms. *Neural Computation*, 5(2):165–199, March 1993.

[8] D. Rumelhart, G. Hinton, and R. Williams. Learning representations by back-propagating errors. *Nature*, 323(9):533–536, Oct. 1986.

[9] J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B. Deylon, P. Y. Glorennec, H. Hjalmarsson, and A. Juditsky. Nonlinear black-box modeling in system identification: A unified overview. *Automatica*, 31(12):1691–1724, Dec. 1995.

[10] P. van der Smagt. Minimisation methods for training feedforward neural networks. *Neural Networks*, 7(1):1–11, 1994.