This is a repository copy of *The Kernel Adaline: A New Algorithm for Non-Linear Signal Processing and Regression*.

White Rose Research Online URL for this paper:
http://eprints.whiterose.ac.uk/82396/

Monograph:
Frieb, Thilo-Thomas and Harrison, R.F. (1998) The Kernel Adaline: A New Algorithm for Non-Linear Signal Processing and Regression. Research Report. ACSE Research Report 731 . Department of Automatic Control and Systems Engineering

---

Reuse
Unless indicated otherwise, fulltext items are protected by copyright with all rights reserved. The copyright exception in section 29 of the Copyright, Designs and Patents Act 1988 allows the making of a single copy solely for the purpose of non-commercial research or private study within the limits of fair dealing. The publisher or other rights-holder may allow further reproduction and re-use of this version - refer to the White Rose Research Online record for this item. Where records identify the publisher as the copyright holder, users can verify any specific terms of use on the publisher's website.

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk/

The Kernel Adaline: A New Algorithm for Non-linear Signal Processing and Regression

Thilo-Thomas Frieß and Robert F Harrison

Department of Automatic Control and Systems Engineering

The University of Sheffield, Mappin Street, Sheffield, S1 3JD, UK

Research Report No. 731

18 November 1998

Abstract

A certain class of non-linear algorithm for signal processing and machine learning is based on the same intrinsic principle. Some given samples (training points) are in the first stage mapped into a very high-dimensional *linearisation* space (the feature space of pattern recognition theory), and then a linear algorithm performs its work in this space. The expensive expansion into the linearisation space can be performed efficiently using Mercer's kernel functions, as studied by Aizerman *et al.* in the 1960s. In this work a non-linear adaptation of the (so far linear) Adaline algorithm by Widrow and Hoff is proposed. The new algorithm combines the conceptual simplicity of a least mean square algorithm for linear regression, but exhibits the power of a universal non-linear function approximator. The kernel Adaline algorithm is introduced, and the first experimental results are given.

SHEFFIELD UNIV. LIBRARY ST. GEORGE'S

Introduction

In the field of function approximation it is well known that by expanding the function in a series form it can be represented to any desired degree of accuracy by taking enough terms. It is therefore possible, in principle, to conduct a linear regression on a new set of variables that have been pre-processed by a fixed mapping (e.g. the polynomials of Taylor series, the sines and cosines of Fourier series, etc.). This approach has one major drawback generally known as the curse of dimensionality. Essentially, the number of terms to be found is exponential in the order of the expansion. This leads not only to a large computational burden but also to the more practical problem of requiring an unreasonably large amount of data from which the coefficients must be estimated for a given level of (statistical) confidence. A general series expansion can therefore lead to an intractable problem.

In 1964 Aizerman and colleagues (Aizerman, Braverman et al. (1964)) introduced the so-called *method of potential functions* into the pattern recognition literature. The non-linear algorithm studied there is a linear Perceptron (Rosenblatt (1962)), which is computed *implicitly* in a space of infinite-dimension (called the linearisation space) using kernel functions. The impact of this work has been rediscovered in the nineties by Vapnik and colleagues (Boser, Guyon et al. (1992)), where it has been used to construct a non-linear adaptation of the generalised portrait algorithm – the support vector (SV) machine (Cortes & Vapnik (1995); Vapnik (1995)).

A recently introduced algorithm, the kernel Adatron (Friess, Cristianini et al. (1998); Friess & Harrison (1998b); Friess & Harrison (1998a); Friess & Harrison (1998c)), provides a fast, conceptually simple, and computationally robust alternative to SVM classifiers which are strongly based on solving quadratic programming problems. The

200447005

kernel method is used to provide arbitrary, large margin discriminant functions via an iterative algorithm in contrast to the batch processing of the SVM approach.

Another interesting alternative to SV machines are linear programming machines (Friess & Harrison (1998a)), these learning machines represent Perceptrons with weight decay regularisation in the hidden unit space of a two-layered radial basis function neural network, which is equivalent to a non-linear classification boundary in the input space. As the name implies, this kernel algorithm requires only the solution of a linear programme which is computationally much more attractive than the quadratic programming required by SV machines.

SVMs can also be used for function approximation but become intractable for large samples. Motivated by the success and simplicity of the kernel Adatron we now use kernels to develop a non-linear version of the (so far linear) Adaline algorithm of Widrow and Hoff (Widrow & Hoff (1960)), yielding a very general non-linear mapping through an algorithm with known convergence properties.

The problem of function approximation will then be seen to reduce to one of selecting an appropriate kernel and its parameter, which specifies the mapping to the linearisation space. This parameter can be selected empirically via a train and test cycle as is usual in neural network training.

Linear Minimum Mean Square Estimation (MMSE)

Consider the arrangement shown in figure 1 where x, y, t and e denote the input, output, target and error signals, respectively. The input may be an n-vector, while all other quantities are scalar and the subscript, p, indicates the index in the sample of length L. The quantity, w, denotes the adjustable system parameters or weights and b, an adjustable bias. The objective is to design $f = \langle w, x \rangle + b$ (choose the weights and bias) so that the mean square error between the system output, y, and the target (measured data), t, is minimised. This leads naturally to the cost functional

$$J = E\{e^2\} \cong \frac{1}{2L}\sum_{p=1}^{p=L} e_p^2 = \frac{1}{2L}\sum_{p=1}^{p=L}\left(t_p - f\left(w, b, x_p\right)\right)^2 \tag{1}$$

where the expectation is replaced by its empirical estimate.

While we could go directly to a closed form solution for the optimal set of weights via the normal equations, we shall see that this is not feasible in the general, non-linear, case. To pre-empt this we proceed to the direct gradient descent algorithm, which is known to be robust and computationally simple. It is straightforward to show that the incremental update rule is given by:

$$\begin{aligned} y\left(x_p\right) &= \langle w, x_p \rangle \\ w &\leftarrow w + \eta\left(t_p - y\left(x_p\right)\right)x_p \end{aligned} \tag{2}$$

where η is the adaptation rate (possibly a decreasing function of time) and that for a suitable value of adaptation rate the solution will converge towards the neighbourhood of the global minimum of J e.g. (Clarkson (1993)).

So far it has been assumed that the hyper-plane to be learned by the Perceptron passes through the origin. If, additionally, a bias parameter is required the samples, x, can be augmented $x \rightarrow [x\ 1]^T$ and then used in the algorithm above. Now the weight vector has one more component, this new component is the bias parameter, b.

Now observe from (2) that, as in any Perceptron, the weights can equally well be represented as a weighted sum of the data samples, i.e. $w = \sum_{i=1}^{i=L} \alpha_i x_i$ so that we may re-write the linear output function in its *data dependent* form (Friess & Harrison (1998a)):

$$y_p = \left\langle \sum_{i=1}^{i=L} \alpha_i x_i, x_p \right\rangle + b = \sum_{i=1}^{i=L} \alpha_i \langle x_i, x_p \rangle + b \tag{3}$$

The expression of the output function as a weighted-sum of inner products of data vectors is central to the extension of the procedure to non-linear systems.

To derive the data dependent version of a Perceptron two things are required (Friess & Harrison (1998a)); the weight update rule and the decision function must be re-written. The data dependent decision function is already given in (3).

In a conventional Perceptron, for both classification and regression, the decision function is given by: $f(x_i) =< w, x_i > +b$, and the update function is given by: $w \leftarrow w + p_u x_i$ (p_u is a scalar).

It can be shown that, for data dependent Perceptrons (using the weighted expansion of w on the training points, x, the decision function is given by:

$f(x_i) = \sum_{p=1}^{L} \alpha_p < x_p, x_i > +b$, and that the equivalent update rule is given by:

$\alpha_i \leftarrow \alpha_i + p_u$, $b \leftarrow b + p_u$ (Friess & Harrison (1998a)). Both the data dependent representation and the formulation given above represent the same Perceptron.

Note that in the data dependent formulation the whole update process is realised by adding values to the multipliers α, and by computing dot products between training patterns.

Data dependent non-linear MMSE via kernel functions

Consider now a fixed mapping of the input data, thus: $z_p = \varphi(x_p)$, where $\varphi(.)$ is in some sense rich enough to capture the underlying functional form of f (Aizerman, Braverman et al. (1964)). Clearly, if we knew this function, we might pre-process our data and use linear MMSE to find the optimal linear combination of the zs. We call these new data, *features* in pattern recognition.

As stated earlier, it is desirable but impractical to allow explicit, high dimensional expansions of the data, but through the use of the class of functions known as Mercer kernels it is possible to perform the mapping implicitly, thus avoiding the undesirable explosion of terms to be estimated. Mercer kernels represent inner products in some Hilbert space (Courant & Hilbert (1953); Wahba (1998)) and thus allow the mapping of two samples, x_u, x_v into a high dimensional linearisation space defined by $(x_u, x_v) \mapsto (z_u, z_v) = (\varphi(x_u), \varphi(x_v))$ and the subsequent calculation of inner products there, thus:

$$k(x_u, x_v) = \langle \varphi(x_u), \varphi(x_v) \rangle = \langle z_u, z_v \rangle \tag{4}$$

Any function, $k(.,.)$, satisfying Mercer's conditions may be used to represent inner products in linearisation space. Examples of common functions satisfying the

conditions in addition to the scalar product are the Gaussian radial basis function, k_{rbf} and the polynomial kernel, k_{pol}, given by:

$$k_{rbf}(x_u, x_v) = \exp\left(-\|x_u - x_v\|^2 / 2\sigma^2\right),\ k_{pol}(x_u, x_v) = \left(\langle x_u, x_v \rangle + 1\right)^d$$

Note that these kernels each have an associated parameter that can loosely be thought of as a "smoothing" parameter.

The use of the kernel allows the replacement of the scalar product in the function, *f*, with an inner product in a high dimensional linearisation space without explicitly performing the expansion. Notice though, that the weight vector, *w*, now resides in linearisation space (which will, in some cases, be infinite dimensional (Smola, Schölkopf et al. (1998))) and is not, therefore, accessible for update. In the data dependent representation, however, the multipliers *are* accessible and we therefore replace the scalar products in the foregoing with a kernel function and perform the update of the alphas as described above.

If, in the data dependent perceptron, dot products are replaced by kernel functions the algorithm automatically operates in the linearisation space. All that is required is to compute the proposed updates p_u given by the Adaline scheme (that is $p_u = t_i - f(x_i)$)and update the weight vector (by updating the multipliers, α).

Now the data dependent, non-linear function, *f*, which is linear in the linearisation space, is given by: $f(x_i) = \sum_{p=1}^{L} \alpha_p < \phi(x_p), \phi(x_i) > + b = \sum_{p=1}^{L} \alpha_p k(x_p, x_i) + b$, which can be calculated efficiently if a cache matrix is used (note that $k(x_i, x_j)$ will be computed several times while cycling through the training set).

The updates of this non-linear Perceptron, using augmented patterns in the feature space $x \rightarrow [\varphi(x)\ 1]'$ are given by: $\alpha_i \leftarrow \alpha_i + p_u$, $b \leftarrow b + p_u$. A pseudocode algorithm is given below.

Kernel Adaline Learning

1. Choose a starting point (e.g. $\alpha_i = 0,\ i = 1 \ldots L,\ b = 0$) and a learning rate, η
2. WHILE error criterion not met
3. FOR $i = 1 \ldots L$
   choose an input x_i $i \in [1, L]$
   calculate output of current kernel Adaline $f(x_i) = \sum_{p=1}^{p=L} \alpha_p k(x_p, x_i) + b$
   or use a cache matrix for speed
   update the corresponding multiplier and bias by the Adaline rule:
   $\alpha_i \leftarrow \alpha_i + \eta(t_i - f(x_i))$
   $b \leftarrow b + \eta(t_i - f(x_i))$
4. END FOR
5. END WHILE

Termination

Note that in the above formulation the algorithm terminates when its error criterion is met. In the Adaline, and more generally in neural networks, one might stop the learning process as soon as the empirical error (say the RMS or LMS error) lies below a certain threshold which may have been defined a priori. An alternative approach is to implement early stopping. Usually this is realised in neural networks by monitoring the training and validation error. At the minimum point of the *validation* cost (e.g. MSE or other more appropriate measure) the learning process is stopped. The method can also be applied directly to the kernel Adatron.

It can be shown that after q iterations using a learning rate, η, the vector of multipliers, α, lies in a box of size $q*\eta$ in the dual space of the parameters α. Since the data dependent structure is fixed the weight vector, w, will also lie in a box. Early stopping, therefore, implies weight decay regularisation both in the convex dual space of multipliers, and in the linearisation space implied by the kernel function (where weight vector, w, resides).

Convergence Properties of the Kernel Adaline

Since the kernel Adaline implements the original Adaline algorithm in the kernel linearisation space the original proof of the Adaline given in (Widrow & Hoff (1960)) can be applied. It is known that Mercer kernel functions represent inner products in linearisation space. The original proof of the Adaline does not depend on the distribution of training patterns, and therefore proves that, after mapping training patterns, x, to their corresponding vector, z, the algorithm will still converge. Therefore the original proof of the Adaline also applies to the Adaline in the kernel linearisation space, as presented above.

Multiple output prediction

So far we have only examined the prediction of one output variable. As with neural networks our approach can easily be generalised for cases where many output signals are to be predicted from one input signal simply by using one kernel Adaline for each output signal and maintaining for each output signal one separate set of multipliers. The different outputs can be learned sequentially, or in parallel as in neural networks. In both cases early stopping techniques are applicable. Clearly, this approach is applicable for any kernel Perceptron, e.g. the kernel Adatron, etc.

Examples

Static curve fitting

Here we use the method to fit a simple one-dimensional curve. A set of measurements at a number of experimental conditions is made and these are assumed to comprise the true value plus an uncorrelated measurement noise. Here the sinc function is uniformly sampled on the interval (-10, 10) and uncorrelated, zero-mean Gaussian noise with a standard deviation of 0.2 is added. Here we present results from a single realisation only; thus statistical variations are to be expected. A Gaussian kernel is used with three values of smoothing parameter to demonstrate the effects of over-, correctly- and under-fitting. Figure 2(a) shows how the MSE varies with iteration number for both training and testing data with a very narrow kernel $\sigma = 0.1$. The near zero value of MSE for training error indicates the learned function is severely over-

specialised to the training data (figure 2(b)). By widening the kernel to a value of 2 this effect can be controlled (figure 2(c)) leading to a good fit. Further widening of the kernel to a value of 5 yields serious under-fitting (figure 2(d)).

Non-linear filtering

Here a generalisation of the FIR linear optimal filtering theory is made using kernels for non-linear applications such as systems identification, time series analysis and prediction, active noise cancellation, channel equalisation, signal deconvolution, delay estimation and inverse-based control. We use our new optimal filtering theory to identify an unknown non-linear, moving average system:
$y(n) = 0.5x(n) + x(n-1)^3 + \varepsilon(n)$ with $\varepsilon \sim N(0,4)$ and $x \sim N(0,1)$ with independent increments. Figure 3(a) shows the reduction in MSE for an assumed correct model structure (single lag, cubic non-linearity). The minimum value of validation-set MSE occurs at around 25 iterations, so we stop learning at this point to regularise the solution. Figures 3(b) and 3(c) show the fits to the training and validation data repectively. We note the excellent performance on both sets. For reasons of space we do not show further MSE plots but observe that early stopping is used when there is a clear minimum in the validation MSE plot or close to where an asymptote is first achieved. In figure 4 the polynomial order is increased to five. Here a degree of overfitting is evident but it is not severe. In figure 5 we overestimate the required number of lags instead. Again overfitting is evident and preditive performance worsens. Finally, in figure 6 both the number of lags and the polynomial degree are set to five leading to the poorest performance of all. However, this still captures many of the major features of the time history, which may be influenced by the FIR nature of the model.

Next an autoregressive term is added and the number of input lags increased, thus:
$y(n) = 0.5y(n-1) + x(n-1) + x(n-2)^3 + \varepsilon(n)$ leading to an IIR-type system. Figure 7(a) shows the behaviour of the MSE for a kernel Adaline model using three lagged inputs and third order polynomial kernel, i.e. no AR component. Clearly, for such a system the true temporal order is large to accommodate the IIR. Nontheless, good predictive accuracy is achieved once again using only three time lags (figures 7(b) and 7(c)). Using the same temporal structure but changing to a Gaussian radial basis function kernel of width $\sigma = 3$ we observe substantial degradation in performance(figure 8). In conductucting experiemntsthe ergression was found to be sensitive to the choice of the kernel width, perhaps indicating that the RBF choice is poor for this problem.

Conclusions

When Rosenblatt studied his Perceptron algorithm in the 1960s he suggested the idea of mapping training points into a high-dimensional linearisation space. The kernel Adaline is such a Perceptron, it uses kernel functions for large, in some cases infinite, expansions, and then applies Widrow's and Hoff's Adaline learning rule in the high dimensional linearisation space. Conceptually the problem of finding a good mapping has been solved in an elegant way using Mercer kernel functions as first suggested by Aizerman and co-workers in the 1960s.

These powerful techniques have been used to develop a non-linear version of Widrow's and Hoff's linear Adaline algorithm. The convergence of the algorithm is proven, since the original proof can be applied in the linearisation space (the kernel

feature space). In contrast to conventional neural networks such as the multi-layer Perceptron (Rumelhart, Hinton et al. (1986)) the cost function of the kernel Adaline is always convex, therefore it can be guaranteed that the algorithm always converges in finite time to an optimal solution, subject to a suitable choice of adaptation rate.

Early stopping techniques, which are strongly related to weight decay regularisation, can be applied to ensure a better generalisation ability of the learning machine, and, as a by product, to reduce the computational effort of the learning process.

Our initial experimental results have demonstrated the performance of the new algorithm for a simple curve-fitting task and for a problem in non-linear systems identification. It is clear form these that the procedure has a good deal to offer in non-linear signal processing and regression.

Future work will focus on methods of how to improve the learning technique, both algorithmically and in terms of speed by using more advanced optimisation techniques (e.g. line search in direction of the canonical base vectors of the quadratic cost function).

References

Aizerman, M., Braverman, E. et al. (1964) Theoretical foundations of the potential function method in pattern recognition learning. *Automation And Remote Control,* **25,** 821-837.

Boser, B.E., Guyon, I.M. et al. (1992) A training algorithm for optimal margin classifiers. *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, Pittsburgh, 144-152.

Clarkson, P.M. (1993) *Optimal and Adaptive Signal Processing*. Boca Raton: CRC Press.

Cortes, C. & Vapnik, V. (1995) Support-vector networks. *Machine Learning,* **20,** 273-297.

Courant, R. & Hilbert, D. (1953) *Methods of Mathematical Physics*. New York: Interscience Publishers Inc.

Friess, T.T., Cristianini, N. et al. (1998) The kernel-Adatron algorithm: a fast and simple learning procedure for support vector machines. In Shavlik, J.W., (Ed.), *Machine Learning: Proceedings of the 15th International Conference*. San Francisco, USA: Morgan Kauffman.

Friess, T.T. & Harrison, R.F. (1998a) Perceptrons in kernel feature spaces. Research Report No. **720**, The University of Sheffield, Sheffield.

Friess, T.T. & Harrison, R.F. (1998b) Kernel Adatron with bias unit: analysis of the algorithm. Research Report No. **729**, The University of Sheffield, Sheffield.

Friess, T.T. & Harrison, R.F. (1998c) Support vector neural networks: the kernel Adatron with bias and soft margin. Research Report No. **725**, The University of Sheffield, Sheffield.

Rosenblatt, F. (1962) *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Washington, DC: Spartan Books.

Rumelhart, D.E., Hinton, G.E. et al. (1986) Learning representations by back-propagating errors. *Nature,* **323,** 533-536.

Smola, A.J., Schölkopf, B. et al. (1998) The connection between regularization operators and support vector kernels. Anonymous *Neural Networks*

Vapnik, V. (1995) *The Nature of Statistical Learning Theory*. Springer Verlag.

Wahba, G. (1998) Support Vector Machines, Reproducing Kernel Hilbert Spaces and the Randomized GACV. Department of Statistics, Technical Report No. 984rr, University of Wisconsin, Madison.

Widrow, B. & Hoff, M. (1960) Adaptive switching circuits. *Proceedings of the 1960 IRE WESCON Convention*, Part 4 96-104.

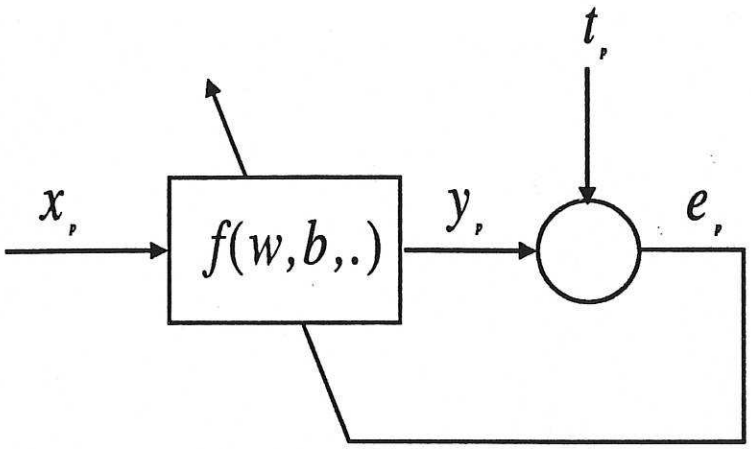


Figure 1: Adaline configuration

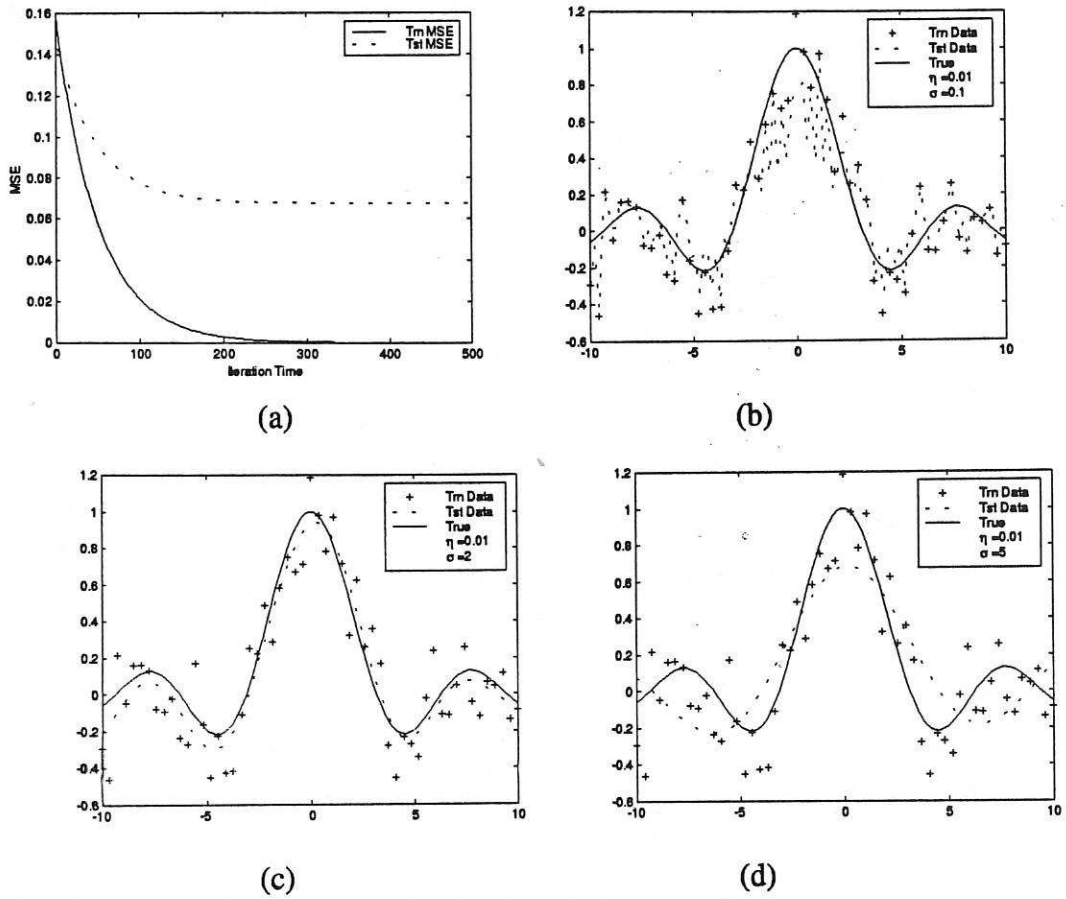
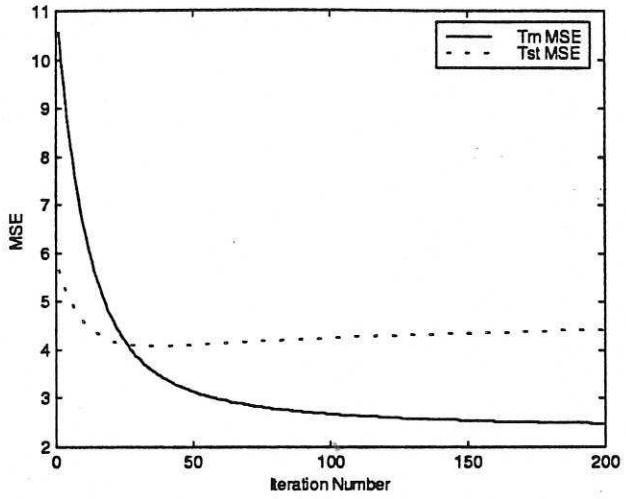
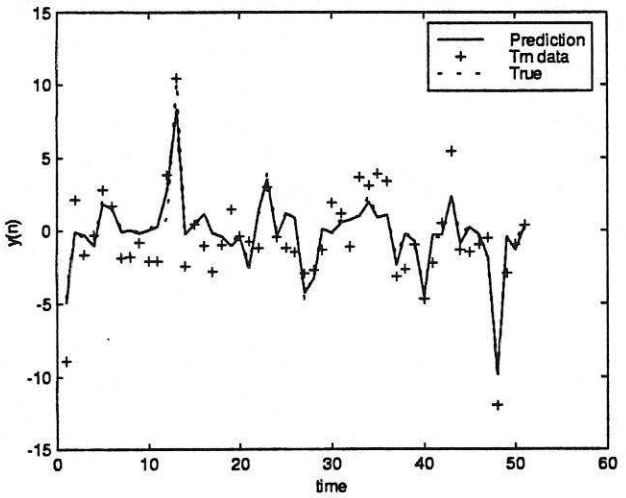


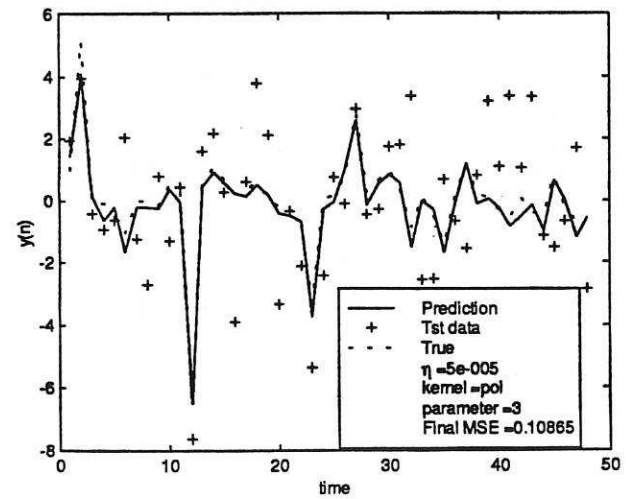
Figure 2: (a) MSE vs training iterations for training and validation data. True curve with training data and reconstructed curve (test data) superimposed: Gaussian kernel with (b) $\sigma = 0.1$, (c) $\sigma = 2.0$ and (d) $\sigma = 5.0$.



(a)

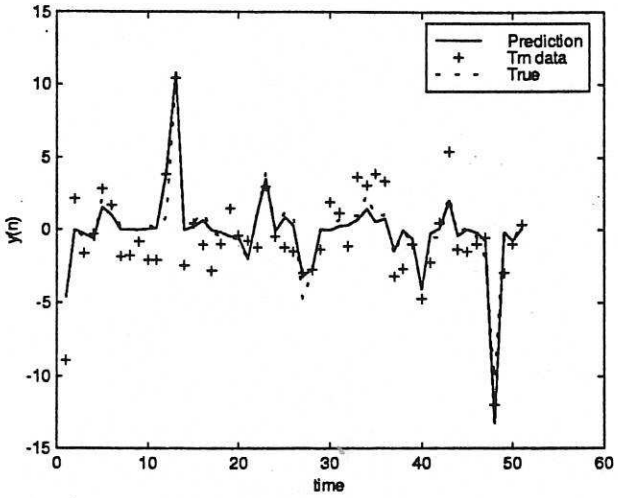


(b)

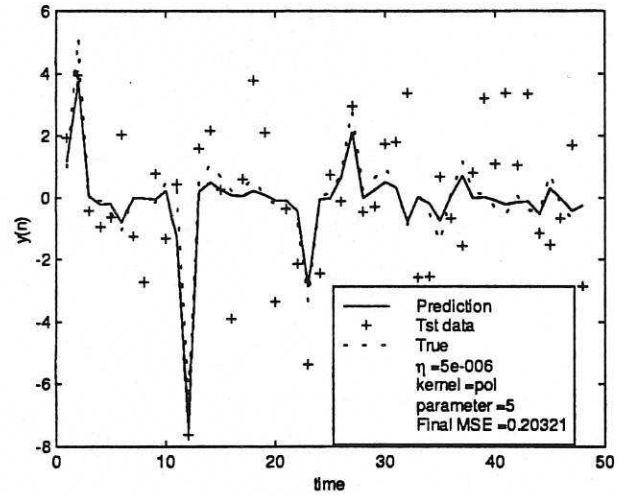


(c)

Figure 3: (a) MSE for training and validation data. FIR system with known structure (b) training data; (c) validation data.

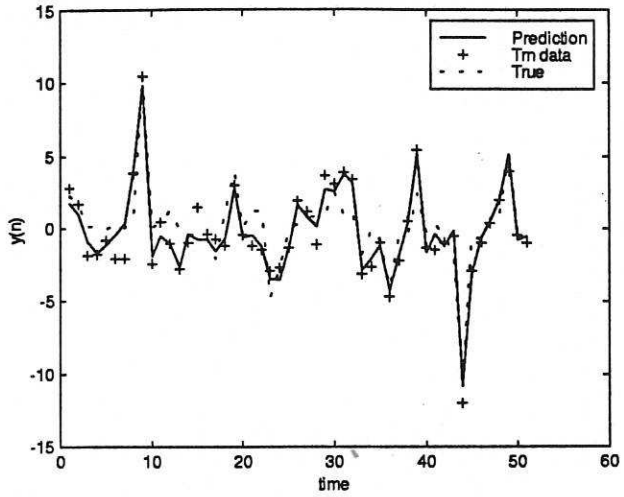


(a)

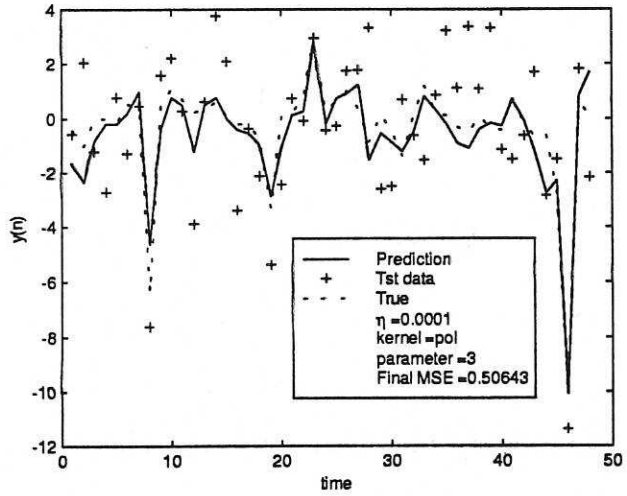


(b)

Figure 4: FIR-type system with overestimated polynomial order *but* correct temporal structure (a) training data; (b) validation data.

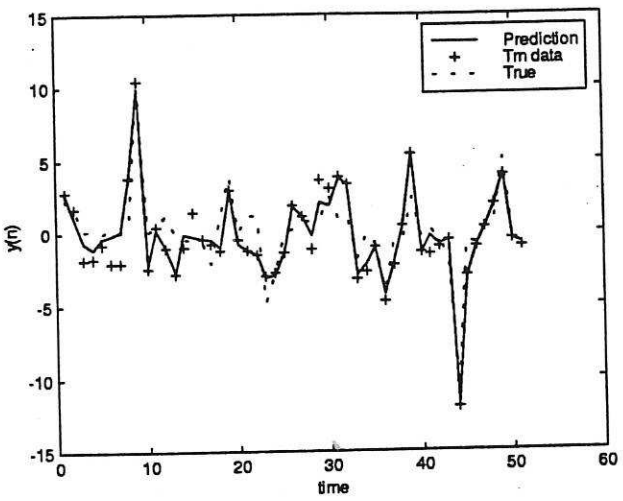


(a)

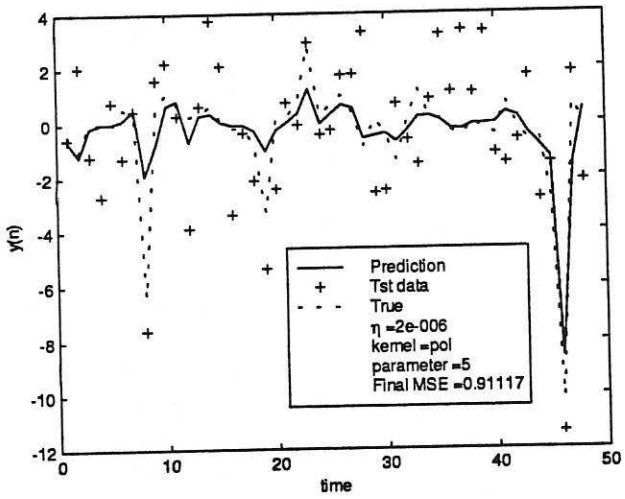


(b)

Figure 5: FIR-type system with correct polynomial order *but* overestimated temporal structure (a) training data; (b) validation data.

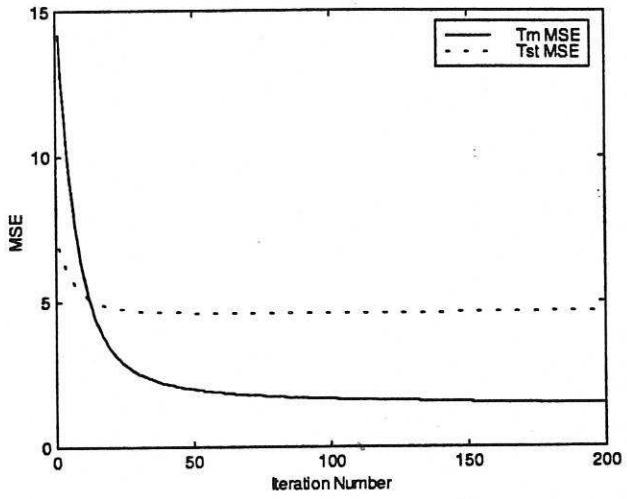


(a)

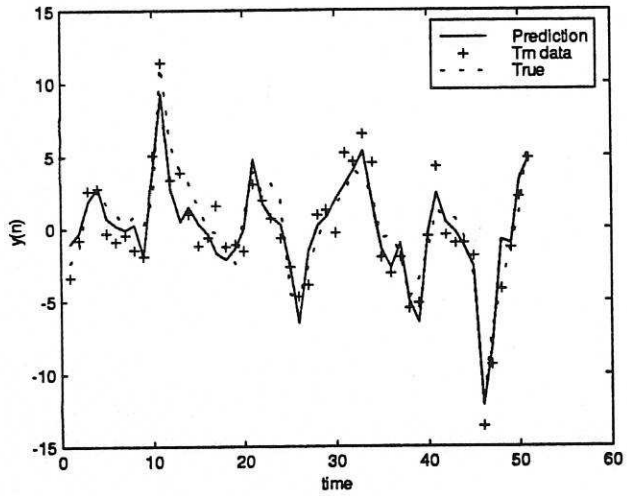


(b)

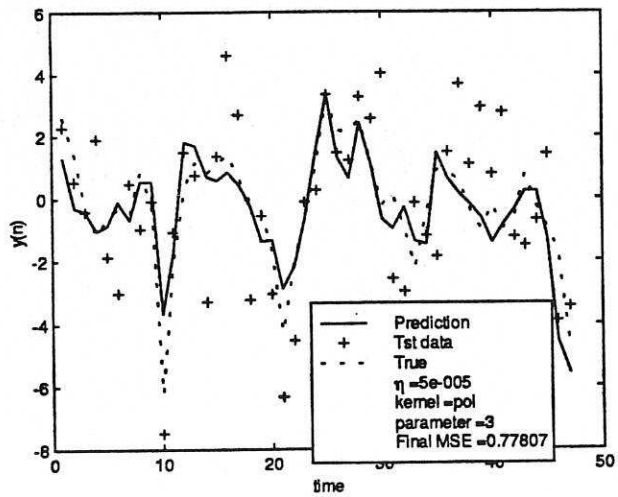
Figure 6: FIR-type system with both overestimated polynomial order *and* temporal structure (a) training data; (a) validation data.



(a)



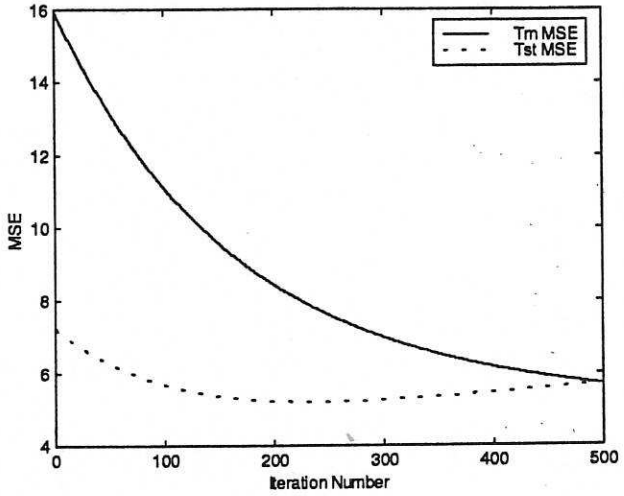
(b)



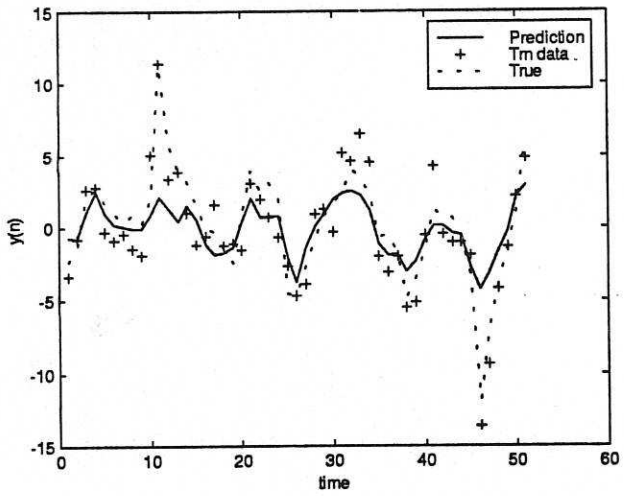
(c)

SHEFFIELD UNIV.
LIBRARY
ST. GEORGE'S

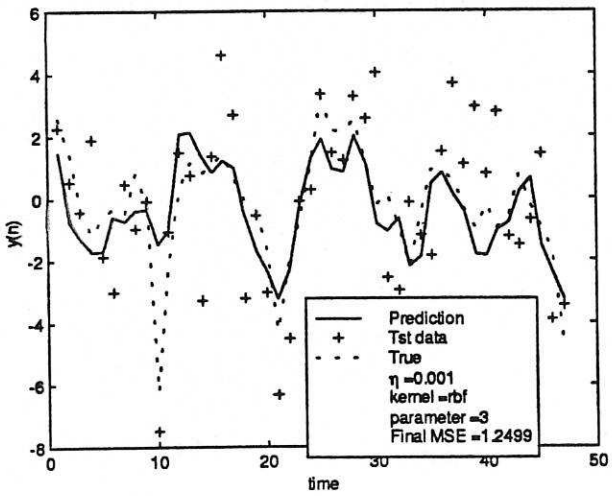
Figure 7: (a) MSE for training and validation data. IIR-type system with correct polynomial order and three time lags (b) training data; (c) validation data.



(a)



(b)



(c)

Figure 8: (a) MSE for training and validation data. IIR-type system with Gaussian kernel ($\sigma = 3$) and three time lags (b) training data; (c) validation data.