# Modeling Time-Varying Processes by Unfolding the Time Domain

Lars Kindermann[a] & Thomas P. Trappenberg[b]

[a] FORWISS, Bavarian research center for knowledge based systems, Germany, kindermann@forwiss.de
[b] RIKEN Brain Science Institute, Lab. for Information Synthesis, Japan, thomas@brain.riken.go.jp

## Abstract

*Most current technologies in modeling time varying processes aim to adapt a static model over time in what has become to be known as continuous learning. We propose here a different approach to the same problem domain that of including the time explicitly in the modeling. An example implementation of this strategy is given in form of a multilayer perceptron with explicit time input. The performance of this approach is evaluating on a benchmark that was constructed to illustrate typical problems in industrial applications.*

## Introduction

The environment in which we live is not static, it can change rapidly over time. Hence, it is important for us to be able to adapt continuously to new situations, and our ability of ongoing learning is thought to be instrumental in coping with such changing environments. It is this kind of adaptation in time-varying systems that we aim to attack with continuous learning procedures [1]. A major problem for such procedures is thereby the plasticity-stability dilemma [2], that of keeping previous acquired knowledge (stability) while enabling optimal adaptation to new situations (plasticity).

Time varying systems can also be a major complication in controlling industrial processes. For examples, picture a production facility such as a steel rolling mill. The obsolescence of individual components does typically lead to a time varying behavior of the whole machine on a time scale dependent on the wearing process. Furthermore, the machine might be used in different production modes such as the rolling of different types of steel on a daily or weekly basis. Models of such machines, which are necessary for the control process, have to be able to account for the changing characteristics of the machine over time. In addition, the modeling of time varying processes of real-world applications is often complicated by at least two more problems [3]. First, the processes we aim to describe are most often highly non-linear so that conceptually more difficult non-linear models have to be derived. Secondly, sample data of the process that we aim to model are often sparse either in time, or in the possible state space of the process, or in both domains.

Current technology targeting continuous learning focus on locally adaptive methods like RBF networks, local linear maps, or hybrid systems (see [4] and references therein) as they can easily be constructed to change only in regions where new data arrives and stay stable elsewhere. Those approaches do in effect try to separate the time domain from the model itself, which we think is not always justified. We propose to include the time domain as parametric input in the modeling of the inherently time dependent process. This simple strategy has, to our knowledge, not yet been explored in the area of continuous learning. Our results show that this strategy should indeed be considered.

## Problem illustration

Our aim is to approximate an unknown time varying process

$$y = y(\underline{x}, t) + \eta(t)$$

from examples (training set), where $\underline{x}$ stands for the parametric degrees of freedoms of the model and $\eta$ represents a stochastic process (noise) that itself can be time dependent. An example of a time varying process without noise and only one degree of freedom beside the temporal domain is illustrated in Figure 1. The training set from which the system surface has to be reconstructed is shown in the figure with dots on the function surface. This training set consists of one training example at each time step in a sequence of discrete time steps. A working point process that is independent of the time varying process we try to model thereby determines the corresponding x-values of the training points. Such training sets are typically produced by measurements of the machine characteristics during operation.

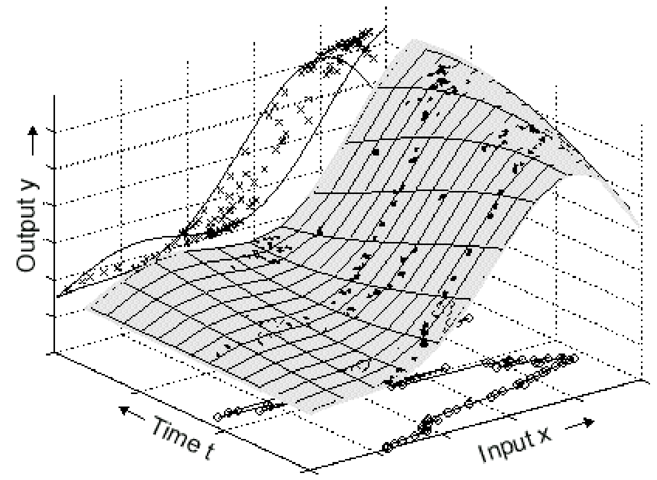

Figure 1: *Example of a function y(x) that changes in time. Only one training point is available for each time value (dots on the function surface). The task at time t is to predict the shape of y(x) at time t+1.*

In practical applications it is often desired to make predictions of the function (machine characteristics) at the next time step. This is typically needed for control purposes. However, a global model can also be desired, for example in optimization applications or when a production machine should be switched to a new operation mode.

Several datasets with similar examples have been produced by Protzel et al. to illustrate typical circumstances in modeling industrial processes and have been used in the NIPS'98 Workshop on Continuous Learning to compare different methods [5]. These benchmark datasets, which can be accessed over the Internet, were generated by AR(3) processes to which discontinuities and sometimes noise was added. Two datasets (A and B) have 10 input parameters (one of which was obsolete) with 10.000 training points. Dataset A corresponds thereby to dataset B without noise. Datasets C-E are generated by the same time dependent process with two input parameters and have all 2000 training data. They only differ in the way the first 1800 training points are chosen.

A set goal of these benchmarks is to enable comparative discussions with two kinds of prediction tasks. The one-step-prediction-task was to predict the function at each consecutive time step at the parameter values given in the training set from the data seen until the previous time step. In the model-evaluation-task the global approximation of the function at the final time step was evaluated at the points given by an evaluation set which is included in the benchmarks.

## Method

Contrary to most existing methods of continuous learning we propose to include the time domain explicitly in the model. To model the time varying processes of the benchmarks described in the previous section we used a simple multilayer perceptron (MLP) that receives the available data (which we termed parameter input). The central idea of our method described here is that we included in addition an artificial input stream of data representing the ongoing time. This is illustrated in Figure 2.

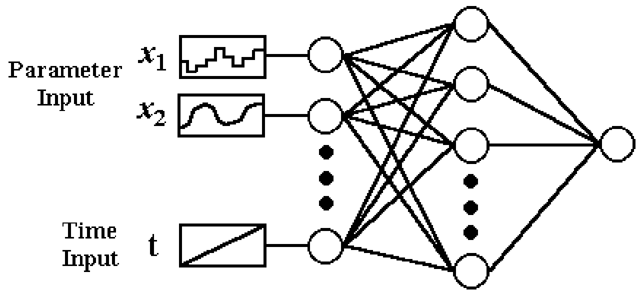

Figure 2: *MLP with additional input representing the time.*

The inclusion of the additional time-input increases the dimensionality of the model, which is realized by the MLP, by 1 and has the effect of an explicit unfolding of the time domain. This avoids the stability-plasticity dilemma because the time-varying $n$-dimensional process is mapped to a $n$+1-dimensional static function. While adaptive methods have to deal with the order and the time the data arrives, our method on the other hand models the behavior in time inherently by trying to approximate the *n+1*-dimensional function surface instead of its $n$-dimensional projection (see figure 1). Therefore it is well suited for a short-term extrapolation in any dimension, including the time domain, so that it can be used in forecasting.

The general idea is not restricted to MLP type networks. Practically any architecture can profit by this simple trick. Indeed, Back and Chen [6] and Back [7] have discussed similar techniques in conjunction with hybrid systems that included recurrent networks. However, contrary to these hybrid system applications, we discuss in this paper the general strategy of increasing the degree of freedom of the network model for applications in continuous learning problems.

## Results

All results for the prediction tasks of the Continuous Learning benchmarks [5] were achieved by using a simple MLP with 20 sigmoidal hidden nodes and a linear output nodes. This network was incrementally trained on all available data up to one time step before the time for which predictions were made. In the case of data set A and B we trained the network only ever 10 time steps, whereas the predictions for the other data sets where done with incremental learning after each new arrival of a training data. In all cases we employed the LM algorithm for training with only up to 5 training epochs for each additional training session.

The performance of the network in the one step prediction task was evaluated with the Normalized Mean Square Error (NMSE) defined by

$$NMSE(t_0) = \frac{1}{\sigma^2(t_0)}\left(y(t_o) - \hat{y}(t_0)\right)^2$$

for each time step $t_0$ for which the prediction was made. The value $\sigma^2$ represents thereby the variance of all the data up to time $t_0$. Hence, only a value of NMSE below 1 indicates a prediction that is better than just predicting the mean value of the available data.

The values for NMSE for the one step ahead predictions of datasets A-E are shown on a logarithmic scale in the left column of Figure 3. In the right column we display a failure rate which counts the relative number of NMSE values above 1 in a sliding window of 100 time steps. After a short transient period most of the data of the dataset A and D-E have been predicted often with high accuracy.

Our method was not able to cope with dataset B that was analog with dataset A except that it also included noise. Noise can be regarded as an unstructured time dependent process. It is hence obvious that our method is not able to model this kind of variations. Further methods to separate structured time dependencies from unstructured variations (noise) have to be included in such circumstances.

The failure rates of datasets C-E all show a peak after time step 760 at which the benchmark function had a discontinuity with a relatively large step value. Such 'jumps' in the target function are a major challenge for all existing methods. Our network was able to adjust to the new situation after a short transient period. A smaller discontinuity at time step 573 had not this drastic effect.

Figure 4 shows a comparison of different algorithms for the global model evaluation at t=2001 for dataset C. While the other methods sometimes achieved locally more accurate results, the method discussed in this paper was the only one which captured the general structure of the underlying problem reasonable well. This can be expected in particular when the available data allow the MLP to interpolate. Extrapolations of MLPs are known to be poor.

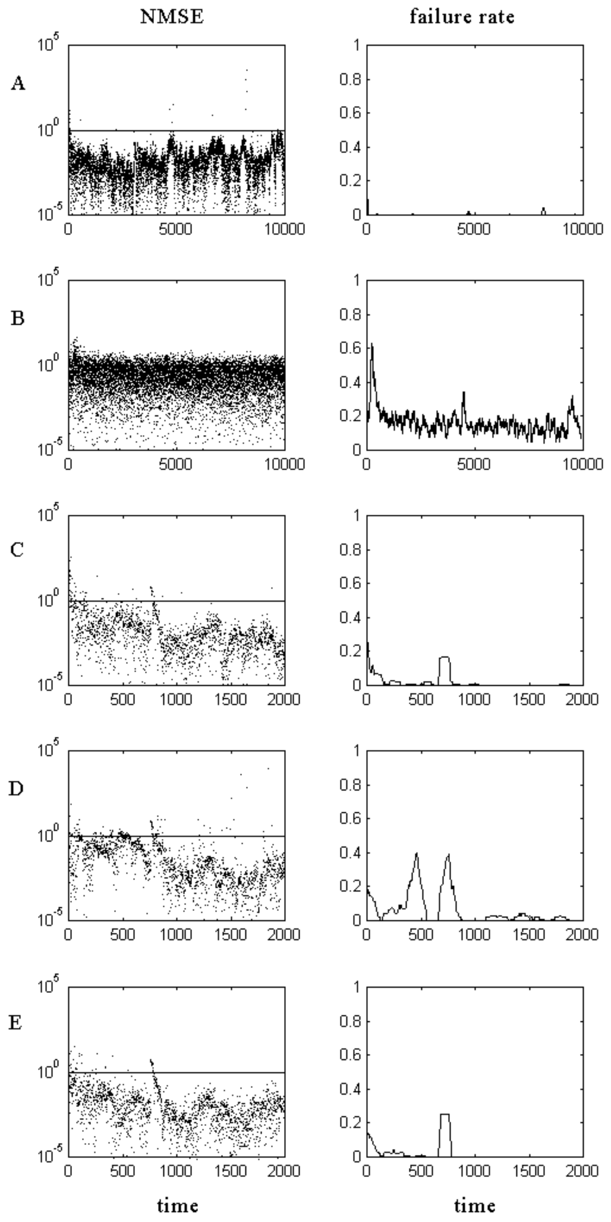

Figure 3. *Normalized Mean Square Error (left column) and failure rate (right column) in the one step prediction for all data sets of the NIPS'98 benchmarks.*

## Conclusion

Continuous learning and the modeling of time varying processes are becoming most important in applications such as controlling industrial processes. An easy approach for modeling time varying processes is proposed and tested on figurative benchmarks with excellent results. The MLP with the additional input of time gave comparable results and in most cases even clearly outperformed all adaptive

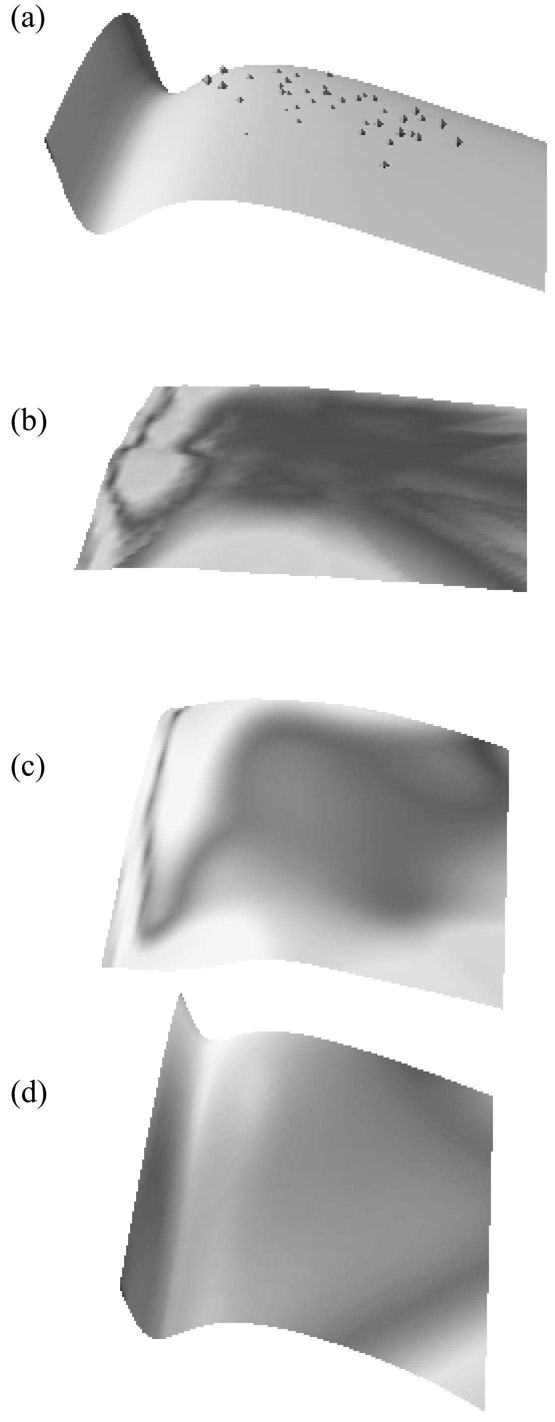

Figure 4: (a) *The target function with most recent training points.* (b)-(d) *Approximations by different methods. Dark shades correspond to good results while light areas denote large errors*. (b) *Adaptive local linear maps.* (c) *Steady state space model.* (d) *MLP with additional time input.*

methods, both in the one-step-prediction-task and in the model-evaluation-task. By avoiding dealing with explicit adaptation, the stability-plasticity dilemma is circumvented and much simpler algorithms like standard MLPs can be used.

Although we have used a simple MLP in the benchmark examples discussed in this paper it should be stressed that the method is not restricted to these architectures. Indeed, MLPs are known to have poor performance in function extrapolation and in regions with limited training examples. An additional estimation of reliability of the predictions as proposed in [8] should therefore be included in the method.

The effort of our method is minimal so that it should be applied before considering more complicated methods. To find out if an application will benefit from the additional time input, input variable selection methods, which evaluate the significance of each input, can be useful. Such a method using independent component analysis and higher order statistics is proposed in [9].

## References

[1] S. Haykin, *Neural Networks, A Comprehensive Foundation*, second edition, Prentice Hall, 1999.

[2] S. Grossberg, *Neural Networks and Natural Intelligence*, MIT Press, Cambridge, 1988.

[3] P. Prozel, L. Kindermann, M. Tagscherer and A. Lewandowski, *Computational Intelligence: Neuronale Netze, Evolutionäre Algorithmen, Fuzzy Control im industriellen Einsatz,*VDI Berichte 1381, VDI Verlag, Düsseldorf, 1998, pp. 347-359.

[4] M. Tagscherer, P. Protzel, *Adaptive Input-Space Clustering for Continuous Learning Tasks*, Proceedings in Artificial Intelligence –FNS'98, Munich, Germany, 1998, pp. 352-358.

[5] P. Prozel, L. Kindermann, A. Lewandowski and M. Tagscherer, *Continuous Learning Benchmarks,* http://www.forwiss.uni-erlangen.de/aknn/cont-learn/cl-benchmarks-NIPS98.html

[6] A.D. Back, T.-P. Chen, *Approxiamtion of Hybrid Systems by Neural Networks*, Proceedings of ICONIP 1997.

[7] A.D. Back, *Multiple and time varying dynamic modeling capable of recurrent neural networks*, Proc. Of the 1977 IEEE Workshop for Signal Processing 7, 1977.

[8] L. Kinermann, A. Lewandowski, M. Tagscherer, P. Prozel, *Computing Confidence Measures and Marking Unreliable Predictions by estimating Input Data Densities with MLPs*, submitted to ICONIP'99.

[9] A.D. Back and T.P. Trappenberg, *Input variable selection using independent component analysis*, IJCNN'99.