ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 407-412

Yield Curve Forecasting by Error Correction Neural Networks and Partial Learning

H. G. Zimmermann*, Ch. Tietz and R. Grothmann
SIEMENS AG, Corporate Technology, CT IC 4, Munich, Germany.

Abstract. Error correction neural networks (ECNN) are an appropriate framework for the modeling of dynamical systems in the presents of noise or missing external influences. Combining ECNNs with the concept of variants-invariants separation in form of a bottleneck coordinate transformation, we are able to handle high-dimensional problems.
Further on, we propose a new learning rule for the training of neural networks, which evaluates only specific gradients for the adaptation of the network weights. By this, we are able to generate time invariant localized structures and thus, support the optimization of the network.
Forecasting the German yield curve, an ECNN including the separation of variants-invariants is superior to traditional neural networks.

1 Introduction

Most dynamical systems are partly driven by an autonomous development and partly by external influences. Recurrent networks enable us to model such open systems in a direct way [5]. Unfortunately, due to e. g. missing external influences or noise, our knowledge about the dynamics is often limited. On this problem, we introduce error correction neural networks (ECNN), which use the last model error as an auxiliary input. Now, the learning can interpret the model misfit as an external shock which guides the dynamics afterwards.

Another problem is the handling of high dimensional systems. The complexity of such systems can be reduced, if it is possible to separate the dynamics into time variant and invariant structures. Clearly, only the variants have to be predicted, while the invariants remain constant. Such a dimensionality reduction in form of a bottleneck network can be combined with ECNNs.

Besides, learning from data is also a major issue of the model building. Here, we are interested in finding time invariant structures out of varying time series. We develop a new learning rule which evaluates only specific gradients during the training, such that localized structures are generated in the network.

The outline of the paper is a follows: First, we introduce the concept of ECNN. Second, we combine the separation of variants-invariants with ECNN in order to handle high dimensional systems. Third, we introduce a new learning rule, which enables us to enforce the construction of localized structures in the network. Finally, we apply our techniques to forecast the German yield curve.

*To whom correspondence should be addressed: Georg.Zimmermann@mchp.siemens.de

ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 407-412

2 Error Correction Neural Networks (ECNN)

Most dynamical systems are partly internally and partly externally driven [5]. For discrete time grids, such an open system can be modeled by a

$$\begin{array}{lllll} \textit{state transition} & s_{t+1} & = & f(s_t, u_t, y_t - y_t^d) & \text{and an} \\ \textit{output equation} & y_t & = & g(s_t)\ . & \end{array} \tag{1}$$

The state transition s_{t+1} is a mapping from the previous state s_t, external influences u_t and a comparison between the model output y_t and the observation y_t^d. If the model error $(y_t - y_t^d)$ is zero, we have a perfect description of the dynamics. However, due unknown influences u_t or noise, our knowledge about the dynamics is often limited. Being so, the model error $(y_t - y_t^d)$ at time t quantifies the model misfit. The output equation computes the model output.

Using weight matrices A, B, C, D of appropriate dimensions corresponding to s_τ, u_τ and $(y_\tau - y_\tau^d)$, a neural network model of the open system in Eq. 1 is

$$\begin{array}{lcl} s_{t+1} & = & \tanh(As_t + Bu_t + D\tanh(Cs_t - y_t^d)) \\ y_t & = & Cs_t \end{array} \tag{2}$$

$$\frac{1}{T} \cdot \sum_{t=1}^{T} \left(y_t - y_t^d\right)^2 \rightarrow \min_{A,B,C,D} \tag{3}$$

In Eq. 2, the output y_t is recomputed by Cs_t and compared with the observation y_t^d. Different dimensions in s_τ are adjusted by D. The system identification task of Eq. 3 is solved by finite unfolding in time using shared weights [1]. Fig. 1 depicts the resulting spatial neural network architecture [5].

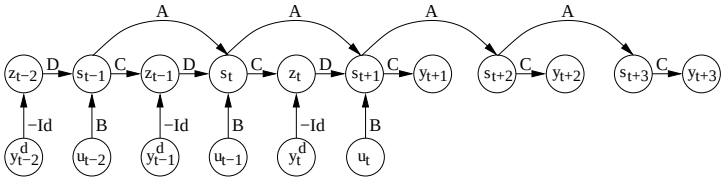

Figure 1: ECNN incorporating overshooting [5]. Note, that $-Id$ is the fixed negative of an identity matrix, while $z_{t-\tau}$ are output clusters with target values of zero in order to optimize the error correction mechanism.

The ECNN is understood best by analyzing the dependencies of s_{t+1} and s_t, u_t as well as $z_t = Cs_t - y_t^d$. The ECNN has two different inputs: $(i.)$ the externals u_t, which directly influence the state transition, and $(ii.)$ the targets y_t^d. Only the difference between y_t and y_t^d has an impact on s_{t+1} [5]. At future time $t+\tau$, there is no compensation of the internal expectations $y_{t+\tau}$ and thus the system offers forecasts $y_{t+\tau} = Cs_{t+\tau}$. An ECNN forecast is based on an autonomous dynamics, external influences and the error correction part [5].

The autonomous part of the ECNN is extended into the future by *overshooting* [5], i. e. we iterate matrices A and C in future direction (Fig. 1). Overshooting regularizes the learning and thus may improve the model performance. Of course we have to supply the additional output clusters $y_{t+1}, \ldots, y_{t+n}$ with target values. Because of shared weights, no additional parameters are used.

ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 407-412

3 ECNN & Variants-Invariants Separation

Modeling high dimensional dynamical systems [5], we integrate the concept of variants-invariants separation into the ECNN of Fig. 1. This dimension reduction is realized by a bottleneck neural network (Fig. 2, left). The compressor F removes time invariant structures from dynamics, i. e. we single out time variant structures of the dynamics. The reconstruction of the complete dynamics (decompression) is done by matrix E. As depicted in Fig. 2, the bottleneck network seems to be disconnected from the ECNN. However, this isn't true: the two sub-systems are implicitly connected via shared matrices E and F [5].

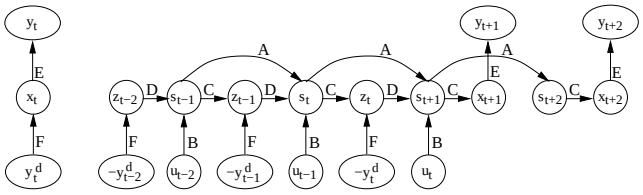

Figure 2: Combining Variants - Invariants Separation and ECNN

Using variants-invariants separation, the ECNN has to predict a low dimensional vector x_τ instead of the high dimensional vector y_τ. Note, that the negative inputs $-y^d_\tau$ are required by the ECNN in order to generate the transformed targets $-x^d_\tau$ in each z_τ cluster. By this, we can compensate the internal forecasts $x_\tau = Cs_{\tau-1}$ with the transformed targets $-x^d_\tau = F(-y^d_\tau)$.

4 Exploring Invariant Structures in Time

Training neural networks, we are interested in identifying time invariant structures in varying time series. Standard learning algorithms like error backpropagation [4] in combination with a particular learning rule generate a hypothesis of such time invariant structures. Fitting our model $y_{t+1} = f(x, w)$ to the observed data x, forecasting is only feasible, if we assume the time invariance of the explored structures (so-called invariance hypothesis) [3].

Unfortunately, a pure fitting of the data often leads to inadequate results, because the underlying dynamics may drift over time. As a remedy, the structures which are created during the learning have to be revised. Only if we cannot falsify the invariance hypothesis, the forecasts of our model are reliable.

Instability pruning: Having trained the neural network until convergence, we suggest to use instability pruning in order to test the invariance hypothesis. Instability pruning uses an important property of the minimal training error: If the learning has reached a minimum, the cumulative gradient g of every weight is approximately zero on the training set $t = 1, \ldots, T$:

$$g = \frac{1}{T}\sum_t^T g_t = \sum_t^T \frac{1}{T} g_t \approx 0. \tag{4}$$

If this criteria is still valid, in case we apply a time varying weighting to the gradients, the invariance hypothesis is true and our model is stable. The weighting

ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 407-412

is done by emphasizing the near to present time gradients on the training set:

$$\sum_t^T \alpha_t g_t = 0 \qquad \text{with} \quad \sum_t^T \alpha_t = 1 \tag{5}$$

We define a weighted average, which favors the most recent gradients on the training set by using a weighting factor $\alpha_t = \alpha t$. We choose $\alpha_t = 2t/(T(T+1))$.

Having trained the neural network until convergence, the test value of instability pruning is now computed for a single weight w by comparing the weighted and unweighted criteria of the gradients:

$$test_w = \frac{\epsilon + \left| \frac{1}{T} \sum_t^T \frac{2t}{T+1} g_t \right|}{\epsilon + \left| \frac{1}{T} \sum_t^T g_t \right|} \tag{6}$$

If we have a stationary distribution of the gradients g_t, the *weighted* cumulative gradient is similar to the cumulative gradient (see Fig. 3, left). Thus, the test value of Eq. 6 is close to 1. Drifting structures over time cause instable gradient distributions (see Fig. 3, right). Thus, the test value is larger than 1.

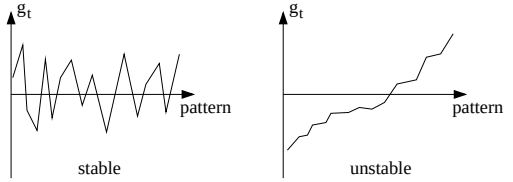

Figure 3: The gradients of valuable weights have a stationary distribution (left). Gradients of instable weights have a non-stationary distribution (right).

Instability pruning does not require a validation set for the calculations of the test values. In principle, the test values can also be smaller than 1. In this case, the system is even more stable in recent time and thus, the associated weights should not be pruned. We suggest to train the network until convergence and compute the test values. Then, we take out the most unstable weights and restart the learning. We iterate this scheme until the error on a certain validation set increases. The constant ϵ prevents numerical difficulties.

Partial Learning is a new learning rule, which improves of generation of invariant structures in the sense of sparse network architectures: A drawback of using standard learning rules together with error backpropagation [4] is that a distributed representation of the input to output relationship is generated, i. e. all parts of the network contain some relevant information. Each weight of the network is partly responsible for the explanation of the network error.

Unfortunately, due to the distributed structures, the falsification of the generated invariance hypothesis by pruning methods is complicated. In addition, structural redundancies in different parts of the network make the task of finding structural instabilities even more difficult [3].

As a remedy, we propose to support the learning of localized structures. We achieve the generation of localized structures by using *only* the $p\%$ largest gradients of each connector for the modification of the network weights. With η as the learning rate, the partial learning rule can be written as

ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 407-412

$$\hat{w} = w + \eta f(p,g) g_t \ , \quad \text{with} \tag{7}$$

$$f(p,g) = \left\{ \begin{array}{ll} 1 & \text{for the p\% largest gradients } |g_t| \text{ in each connector} \\ 0 & \text{otherwise} \end{array} \right\} \tag{8}$$

According to Eq. 8, a weight w is updated to a new value $\hat{w}$, if its associated gradient g_t is one of the p% largest gradients of the concerned connector. Here, we neglect the implication of error backpropagation, that the gradients which are created for a specific training pattern are able to evaluate *all* weights of the network. Partial learning focuses only on weights being in charge for a specific training pattern. As a result, the learning of single training patterns changes only parts of the network and thus, localized structures are preferred.

We choose only the p% largest gradients (Eq. 8), because these gradients often contain the most relevant structural information and are likely to generate highly non-linear structures. This is especially true in the beginning of the training. Large error signals indicate important mismatches of the model, which have to be handled. On the other hand, large gradients are often caused by outliers in the data set. On this problem, we propose to use a network internal preprocessing together with a robust error function [3].

Our experiments indicate, that choosing even the 90% smallest gradients for the learning leads to an inefficient learning and to a poor fit on the training set, while choosing only 10% of the largest gradients allows efficient learning and a perfect fit of the training data. We found that partial learning should *not* be applied to output and bias connections of the neural network.

5 Forecasting the German Yield Curve

We employed an ECNN combined with the separation of variants-invariants (Fig. 2) to forecast the *complete* German yield curve (REX1 - REX10). We are working on the basis of monthly data to forecast the 3 and 6 month changes of the German yield curve. The training set is from Jan. 91 to Aug. 95, while the test set is from Oct. 95 to Dec. 97. The data base consists of stock, bond and foreign exchange market indices of Germany, USA and Japan. The preprocessing is done by computing the scaled momentum of each input [3].

The unfolding in time of the ECNN includes six past and an overshooting branch of six future time steps (Fig. 2). The overshooting branch provides us with the 3 and 6 month yield curve forecasts. We found that only 3 variants are important for the German yield curve. The ECNN was trained until convergence with *partial* learning using only 10% of the largest gradients.

The performance of the ECNN is evaluated by a comparison with *two* benchmarks: The *first* one refers to a time-delay recurrent neural network (RNN). In contrast to the ECNN the RNN does not include an error correction mechanism. The *second* one is a 3-layer feedforward network (MLP). We optimized the MLP by EBD-pruning [3]. The benchmarking is based on the return of investment (ROI) of each model. The ROI refers to a simple trading strategy using the yield curve forecasts, e. g. we sell bonds if we expect rising interest rates (Fig. 4). We included transaction costs of 1% per trade. Further more,

we computed the risk associated with each trading strategy. As a measure of risk, we consider the forecasting uncertainty of each model.

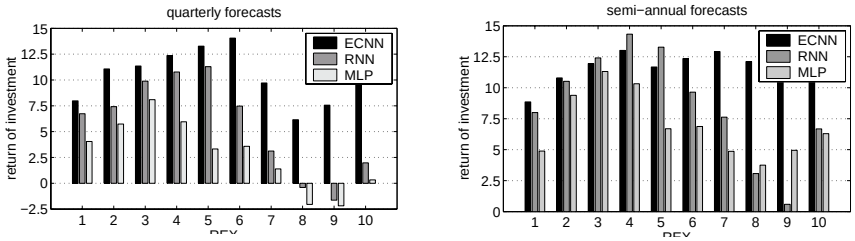

Figure 4: Return of investment of the ECNN and the benchmarks (test set).

As depicted in Fig. 4, the ECNN including variants-invariants separation outperforms both benchmarks. Due to the variants-invariants separation, the ECNN reaches a steady forecasting quality over the complete yield curve, i. e. there is *no* drawback in forecasting long-term instead of short-term maturities. Among the benchmarks, the RNN outperforms the MLP. Concerning the risk, we found that the ECNN has the lowest risk level ($\sigma = 0.0381$). Among the benchmarks, the RNN ($\sigma = 0.0440$) has a lower risk than the MLP ($\sigma = 0.0731$). These results indicate that the additional structures of the ECNN (e. g. variants-invariants separation) improve the forecasting.

6 Conclusion

An ECNN is able to handle external shocks without changing the identified dynamics of the underlying system. The ECNN can be extended by a variants-invariants separation, which allows the modeling of high dimensional, noisy dynamical systems. The training of the ECNN is supported by partial learning, which creates localized time invariant structures.

As indicated by our empirical study, the usage of prior knowledge in form of the proposed architectural enhancements of the network architecture enables us to outperform traditional neural networks. Future work will consider several extensions of the ECNN, e. g. *unfolding in space and time* [5]. We believe, that the ECNN is a promising framework for financial forecasting.

References

[1] S. Haykin. *Neural Networks. A Comprehensive Foundation.* N.Y., 1994.

[2] L. R. Medsker and L. C. Jain. Recurrent Neural Networks: Design and Application, CRC Press, Int. series on comp. intelligence, No. I, 1999.

[3] R. Neuneier and H. G. Zimmermann. How to Train Neural Networks. In *Neural Networks: Tricks of the Trade*, Springer, Berlin, 1998.

[4] P. J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioural Sciences.* PhD thesis, Harvard University, 1974.

[5] H. G. Zimmermann, R. Neuneier and R. Grothmann: *Modeling of Dynamical Systems by Error Correction Neural Networks*, in: Modeling & Forecasting Financial Data. Eds. Soofi, A. and Cao, L., Kluwer, March 2002.