Equation Discovery System And Neural Networks For Short-Term Dc Voltage Prediction

Irena Nančovska, Anton Jeglič and Dušan Fefer
Faculty of Electrical Engineering,
Tržaška 25, Ljubljana, Slovenia
Phone: +386 61 1768 216, Fax: +386 61 1768 214
E-mail: {Irena.Nancovska,Anton.Jeglic,Dusan.Fefer}@fe.uni-lj.si
AND
Ljupčo Todorovski,
Jozef Stefan Institute,
Jamova 39, Ljubljana, Slovenia
Phone: +386 61 1773 307, Fax: +386 61 1258 058
E-mail: Ljupco.Todorovski@ijs.si

Keywords: neural networks, equation discovery, machine learning

Edited by: Cene Bavec and Matjaž Gams

Received: October 12, 1999 **Revised:** November 25, 1999 **Accepted:** December 15, 1999

The aim of the paper is to compare the predictive abilities of the novel method for time series prediction that is based on equation discovery with neural networks. Both methods are used for short-term (one-step ahead) prediction and have the ability to learn from examples. With purpose to validate the predictive models, they are applied to several data sets. The successful predictive models could be used for voltage monitoring in a high precision solid-state DC voltage reference source (DCVRS) without presence of a high level standard, and further for voltage correction as a segment in the software controlled voltage reference elements (VRE).

1 Introduction

Measured time series could be described as mixtures of dynamic, deterministic part which drives the process and observational noise which is added in the measurement process, but does not influence the future behavior of the system. Many up-to-date scientific researches on predicting the future behavior of system are based on modelling of the deterministic part. Examples ranges from the irregularity in the annual number of sunspots to the changes of currency exchange rates. To make a forecast if the underlying deterministic equations of the observed system are not known, one must find out both the rules governing system dynamics and the present state of the system (Gershenfeld & Weigend 1992). Mainstream statistical techniques for predicting include variations of the auto-regressive technique that Yule invented in 1927. The technique uses weighted sum of previous observations of the series to predict the next value. However, there are a number of cases for which this paradigm is inadequate because of the non-linearity of the underlying model (Gershenfeld & Weigend 1992). In the paper we present two different paradigms for forecasting: neural networks and equation discovery. Neural networks used for prediction are characterized as black-box models whereas models obtained with equation discovery systems are transparent (white-box).

Neural networks represent an emerging technology with some important characteristics: universal approximation (input-output mapping), ability to learn from and adapt to their environment and the ability to evoke weak assumption about the underlying physical system which generates the input data (Haykin 1998). In the paper we use three types of neural networks. The first one is a supervised multilayer feedforward network, which is trained with back-propagation learning algorithm (Haykin 1998, Nielsen 1990, Pham 1995). The second type emphasizes the role of time as an essential dimension of learning. It is a natural extension of the first type, replacing the ordinary synaptic weights with finite-duration impulse response (FIR) filters (Haykin 1998, Gershenfeld & Weigend 1992). The third type of network a recurrent structure with a hidden neurons which introduce time in the network processing by virtue of the built-in feedback loop (Alippi 1996, Haykin 1998).

Equation discovery systems explore the hypothesis space of all equations that can be constructed given a set of arithmetical operators, functions and variables, searching for an equation that fits the input data best. In the paper, we present an equation discovery system LAGRAMGE that uses context free grammars for restricting the hypothesis space of equations. The hypothesis space of LAGRAMGE is a set of equations, such that the expressions on their right hand sides can be derived from a given context free grammar. For the purpose of time series prediction, we use difference equations, that predicts the present value of the time series. Three different grammars for linear, quadratic and piecewise linear equations are used.

In order to compare the predictive abilities of two described paradigms we performed experiments in two synthetic and three real world time series prediction problems. The domains used in the experiments present models with different amounts of non-linear dynamics (determinism) and noise (randomness). The predictive models obtained in the experiment with reference voltage domain can be used in to improve the metrological characteristics of a DCVRS in two different manners: voltage monitoring and voltage correction. For the purpose of voltage monitoring, predictive models could be used during the inter-calibration period without presence of a high level standard while the predictors are obtained during the calibration period by using a high precision instrument. Further, the models could be used for voltage correction, as a segment in a software controlled VRE. By implementation of a control loop for voltage correction, based on the obtained predictors, the sensitivity of the reference source could be reduced, which contributes to enhancement of the robustness of the system and thereby the stability of the reference voltage (Nancovska 1997).

The paper is organized as follows. First two sections describe the techniques used for time series predicting. In Section 2 a brief description of used neural networks is given and Section 3 gives overview of the equation discovery system LAGRAMGE. The results of applying both techniques on five time series data sets are presented in Section 4. Finally, Section 5 concludes with a summary of the results and directions for further work.

2 Neural Networks

The time series $x(1), x(2), x(3)\dots$, which describes the system is given. From the series we generate vectors $\mathbf{x(n)} = [x(n-1), x(n-2), \dots, x(n-p)]^T$, which describe the last p values of the phenomenon until time $n-1$. We are trying to find a map $F(\mathbf{x(n)}) = \widehat{x}(n)$ such that the predicted value $\widehat{x}(n)$ in time n is the most similar to the original signal value $x(n)$ in time n. For accomplishment of F we use three different types of neural networks (Gershenfeld & Weigend 1992, Narendra 1990, Pham 1995).

2.1 Multilayer perceptron (MP)

We use a general multilayer feed-forward network (Lippmann 1987) whose learning algorithm is generalized δ-rule or back-propagation (BP). The user interface provides regulation of the following parameters: number of layers, number of neurons in each layer, learning rate η and momentum term α.

Parameters η and α could be changed during the training. Neurons in input layer act as buffers for distributing the input signals $\mathbf{x(n)}$ to neurons in the hidden layer (Haykin 1998, Lippmann 1987, Nielsen 1990, Pham 1995). MP is usually used as pattern recognition tool, but from a systems theoretic point of view it can be also used for approximation of non-linear maps (Narendra 1990).

2.2 FIR multilayer perceptron (FIR-MP)

In order to allow time to be represented by the effect it has on signal processing or to make the network to be dynamic, time delays are introduced into the synaptic structure of the network and their values are adjusted during the learning phase (Haykin 1998). In fact each synapse is represented by a finite-duration impulse response (FIR) filter (Figure 1).

FIR-MP network is a vector generalization of the MP and its learning algorithm is a vector generalization of the standard BP algorithm, called temporal BP (TBP). The basic form of TBP is non-causal because the computation of weights requires knowledge of future values of weights' changes δ-s and weights w-s. It could be made causal by adding a finite number of delay operators on the feedback connections so that only present and past values of δ-s and w-s are used. We hypothesize that by introducing tapped delay feedbacks the NN performance on problems involving time dependencies could be improved. In (Lin 1996) FIR-MP is compared to the NARX recurrent network by its computational power. NARX is computationally as strong as fully connected recurrent network thus is Turing machine equivalent (Siegelmann 1995, Siegelmann & Sontag 1995).

2.3 Recurrent network in real time (RN)

The net (Haykin 1998, Pham 1995) consists of connected input-output layers and processing layer. RN has ability to connect the external time-varying input with its previous output by using delay operator.

The learning algorithm used is real time recurrent learning (RTLL) (Haykin 1998), which is gradient-descent learning algorithm and minimizes the error function by changing the weights of all visible neurons. This architecture is capable of representation of arbitrary non-linear dynamical system and it is Turing equivalent (Alippi 1996, Siegelmann 1995, Siegelmann & Sontag 1995). However, learning simple behavior can be quite difficult by using gradient descent[1]. RTRL is not guaranteed to follow the negative gradient of the error function. This is a consequence of the feedback connection and it can be improved by slow changing of weights. Although RN has difficulty capturing the global behavior (Lin 1996) it is useful for learning short-term dependencies and thus can be used for short-term predictions.

[1]For example, even it is Turing equivalent, it has been difficult to get it successfully learn finite-state machines from example strings encoded as sequences.

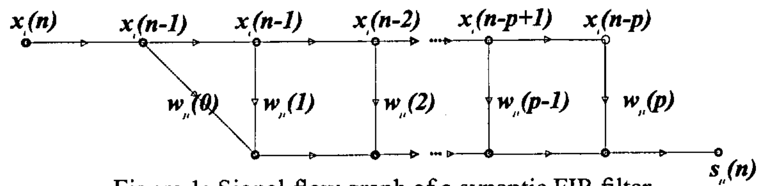

Figure 1: Signal-flow graph of a synaptic FIR filter

3 Equation Discovery

The problem of equation discovery, as addressed by LAGRAMGE, can be defined as follows.
Given are

- a context free grammar $G = (N, T, P, S)$ (see next section) and
- input data $D = (V, v_d, M)$, where
  - $V = \{v_1, v_2, \dots v_m\}$ is a set of domain variables,
  - $v_d \in V$ is the dependent variable and
  - M is a set of one or more measurements. Each measurement is a table of measured values of the domain variables at successive time points:

time	v_1	v_2	$\dots$	v_m
t_0	$v_{1,0}$	$v_{2,0}$	$\dots$	$v_{m,0}$
t_1	$v_{1,1}$	$v_{2,1}$	$\dots$	$v_{m,1}$
t_2	$v_{1,2}$	$v_{2,2}$	$\dots$	$v_{m,2}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
t_N	$v_{1,N}$	$v_{2,N}$	$\dots$	$v_{m,N}$

Find an equation for expressing the dependent variable v_d in terms of variables in V. This equation is expected to minimize the discrepancy between the measured and calculated values of the dependent variable. The equation can be:

- differential, i.e. of the form $\partial v_d/\partial t = \dot{v}_d = E$, or
- ordinary, i.e. of the form $v_d = E$,

where E is an expression that can be derived from the context free grammar G.

3.1 Restricting the space of possible equations

The syntax of the expressions on the right hand side of the equation is prescribed with a context free grammar (Hopcroft & Ullman 1979). A context free grammar contains a finite set of variables (also called nonterminals or syntactic categories) each of which represents expressions or phrases in a language (in equation discovery, nonterminals represent sets of expressions that can appear in the equations). The expressions represented by the nonterminals are described in terms of nonterminals and primitive symbols called terminals. The rules relating the nonterminals among themselves and to terminals are called productions.

The original motivation for the development of context free grammars was the description of natural languages. For example, a simple grammar for deriving sentences consists of the productions *sentence* $\rightarrow$ *noun verb*, *noun* $\rightarrow$ *network*, *noun* $\rightarrow$ *equation*, and *verb* $\rightarrow$ *predicts*. Here *sentence*, *noun* and *verb* are nonterminals, while words that actually appears in sentences (i.e. network, predicts) are terminals. The sentences *networkpredicts* and *equationpredicts* can be derived with this grammar.

We denote a context free grammar as a tuple $G = (N, T, P, S)$, where N and T are finite disjoint sets of nonterminals and terminals, respectively. P is a finite set of productions; each production is of the form $A \rightarrow \alpha$, where A is a nonterminal and α is a string of symbols From $N \cup T$. We use the notation $A \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_k$ for a set of productions for the nonterminal A: $A \rightarrow \alpha_1, A \rightarrow \alpha_2, \dots, A \rightarrow \alpha_k$. Finally, S is a special nonterminal called starting symbol.

Grammars used in equation discovery system LAGRAMGE have several symbols with special meanings. The terminal *const* $\in T$ is used to denote a constant parameter in an equation that has to be fitted to the input data. The terminals v_i are used to denote variables from the input domain D. Finally, the nonterminal $v \in N$ denotes any variable from the input domain. Productions connecting this nonterminal symbol to the terminals v_i are attached to v automatically, i.e., $\forall v_i \in V : v \rightarrow v_i \in P$.

The only restriction on the grammar G is that the right sides of the productions in P have to be expressions that are legal in the C programming language. This means that we can use *all* C built-in operators and functions in the grammar. Additional functions, representing background knowledge about the domain at hand can be used, as long as they are defined in conjunction with the grammar. Note that the derived equations may be non-linear in both the constant parameters and the system variables.

Expressions can be derived by grammar G from the nonterminal symbol S by applying productions from P. We start with the string w consisting of S only. At each step, we replace the leftmost nonterminal symbol A in string w with α, according to some production $A \rightarrow \alpha$ from P. When w consists solely of terminal symbols, the derivation process is over.

3.2 LAGRAMGE - the algorithm

Expressions generated by the context free grammar G contain one or more special terminal symbols *const*. A nonlinear fitting method is applied to determine the values of these parameters. The fitting method minimizes the value of the error function $Error(\mathbf{c})$, i.e. if $\mathbf{c}$ is the vector of constant parameters in expression E, then the result of the fitting algorithm is a vector of parameter values $\mathbf{c}^*$, such that $Error(\mathbf{c}^*) = \min_{\mathbf{c}\in R^{n_c}}\{Error(\mathbf{c})\}$. The error function *Error* is a sum of squared errors function, defined in the following manner:

- for a differential equation of the form $\partial v_d/\partial t = E$:

  $Error(\mathbf{c}) =$
  $\sum_{i=0}^{N}\left[v_{d,i} - \left(v_{d,0} + \int_{t_0}^{t_i} E(\mathbf{c}, v_1, \ldots v_m)\right)\right]^2$, and

- for an ordinary equation of the form $v_d = E$:

  $Error(\mathbf{c}) =$
  $\sum_{i=0}^{N}(v_{d,i} - E(\mathbf{c}, v_{1,i}, \ldots v_{d-1,i}, v_{d+1,i}, \ldots v_{m,i}))^2$,

where N is the size of the measurement table and $v_{j,i}$ the value of the system variable v_j at time t_i. Note that in the case of calculating the error function for differential equations we use the integral of the expression on the right hand side of the equation instead of the derivative of the dependent variable. This is done because the error of algorithms for numerical integration is in general smaller than the error involved of numerical derivation. We use a simple trapezoid formula for numerical integration with the same step size as the time step between successive measurements in the measurement table. The downhill simplex and Levenberg-Marquardt algorithms (Press et al. 1986) can be used to minimize the error function.

Furthermore, the value of a heuristic function for the expression is evaluated. It is equal to the sum of squared errors value *SSE* calculated by the fitting method ($SSE(E) = Error(\mathbf{c}^*)$). An alternative heuristic function *MDL* (minimal description length) can be used, that takes into account the length l of expression E:

$$MDL(E) = SSE(E) + \frac{l}{10 \cdot l_{max}}\sigma_{v_d},$$

where l_{max} is the length of the largest expression generated by the grammar and σ_{v_d} is the standard deviation of the dependent variable v_d. The length is measured as the number of terminals in the expression. The *MDL* heuristic function prefers shorter equations.

A context free grammar can in principle derive an infinite number of expressions (equations). LAGRAMGE thus uses a bound on the complexity (depth) of the derivation used to produce the equation (Todorovski & Džeroski 1997). The LAGRAMGE algorithm exhaustively or heuristically searches for the best equation (according to the selected heuristic function) within the allowed complexity (depth) limits.

3.3 Time series prediction with equation discovery

We reformulate the problem of time series prediction into the equation discovery problem in the following way. Given a time series $x(1), x(2), x(3), \ldots$, we choose a constant p and build matrix M as follows:

time	v_1	v_2	$\ldots$	v_{p+1}
t_0	$x(1)$	$x(2)$	$\ldots$	$x(p+1)$
t_1	$x(2)$	$x(3)$	$\ldots$	$x(p+2)$
t_2	$x(3)$	$x(4)$	$\ldots$	$x(p+3)$
$\vdots$	$\vdots$	$\vdots$		$\vdots$

Now the input domain for equation discovery problem equivalent to the problem of time series prediction is $D = (\{v_1, v_2, \ldots, v_{p+1}\}, v_{p+1}, M)$. We search for ordinary equation of the form $v_{p+1} = F(v_1, v_2, \ldots v_p)$. The obtained equation can be interpreted as difference equation for predicting the next value of the time series $\widehat{x}(n) = F(x(n-1), x(n-2), \ldots, x(n-p))$.

The form of function F on the right-hand side of the equation is biased with a context free grammar G. We used three different context free grammars for restricting the space of possible equation in time series prediction domains. First grammar is used to produce linear models:

$$E \rightarrow const \mid const * v \mid E + const * v$$

The second grammar generates quadratic multivariate polynomials:

$$\begin{array}{lll} E & \rightarrow & const \mid const * F \mid E + const * F \\ F & \rightarrow & v \mid v * v \end{array}$$

Finally, the third grammar used in the experiments generates piecewise linear models. The breakpoint is set to 0.5, which is the middle of the interval of normalized values of time series:

```
double If(double v, double e1, dou-
ble e2) {
        return((v < 0.5) ? e1 : e2);
}
```

$$\begin{array}{lll} IfE & \rightarrow & E \mid \mathtt{If}(v, E, E) \\ E & \rightarrow & const \mid const * v \mid E + const * v \end{array}$$

4 Experiments

4.1 Data sets descriptions

We applied the techniques described in the previous two sections to two synthetic and three real world data sets:

Lorenz system Model of the Lorenz attractor is one of the most frequently used examples of the deterministic chaos system. It is described with the following differential equations:

$$\begin{aligned}\dot{x} &= \sigma(y-x)\\ \dot{y} &= x(R-z)-y\\ \dot{z} &= xy-bz\end{aligned}$$

The values of the constant parameters were chosen to be: $\sigma = 16.0, R = 451992, b = 4.0$. For initial state $x(0) = 0.06735, y(0) = 1.8841, z(0) = 15.7734$ the system is well-conditioned. The equations were simulated for 2000 time steps of length $h = 0.001$. For the prediction task we use the time series for variable z, because it clearly reflects non-linear dynamics of the system.

Reference voltage The observed time series are generated by solid-state VRE-s, based on 7V zener diodes LTZ1000 of a group DCVRS. They are produced by measuring the absolute voltage values of VRE, which is controlled by PC-computer. The PC communicates with DCVRS via serial port RS232. Measuring instrument is digital voltmeter HP3458A. The time series present 2000 samples taken in time intervals of 15 minutes during 500 hour measurement.

Fractional Brownian motions or $1/f$ noises FBM is a random function provided by Mandelbrot and Van Ness (FBM). The most important feature of FBM is that its increments $[B_H(t+T) - B_H(t)] = h^{-H}[B_H(t+hT) - B_H(t)]$ are stationary, statistically self-similar and have Gaussian distribution with a standard deviation $C_H T^H$, where C_H is constant. This is usually called T^H *law of FBM*. The parameter H is directly related to the fractal (Hausdorff) dimension D. For generation of the FBM signals we use the method of spectral synthesis (Nancovska 1997).

Lorenz-like chaos in NH_3-FIR lasers Far infrared lasers have been proposed as examples of a physical realization of the Lorenz model, mentioned earlier (Hubner et al. 1992). However, the actual laser systems are more complex then simple coherently coupled three-level systems. The data set was chosen to obtain several of the important quantities pertinent in comparison to the parameters of numerical data sets obtained by the integration of Lorenz equations. We took into consideration first 2000 time points of the time series.

Sunspots The data set is standard benchmark test for various techniques for time series prediction. It contains the observation of the number of annual sunspots for 280 years.

4.2 Experimental setting

The following methodology of the experiments with the time series prediction data sets was used. Each data set was divided in two parts of equal sizes (1000 time points per set, expect for the Sunspot data set which has only 280 time points). The first part was used as input for the learning system in the training phase, and the second one was used for testing the performance of the obtained predictive model. Furthermore, in the training phase 80% of the training set was used directly for learning and the rest is used for error evaluation only (the evaluation applies the estimation of the performance of the predictor learned so far). The length of the input vector p varied between 1 and 6. The criterion for choosing the best predictor was the root mean square error (RMSE).

In the experiments with neural networks, the value of parameter η is gradually decreased from 0.95 to 0.003 to avoid local minima of the error surface. When training the recurrent network η is set to the lower value from the beginning to make the time scale of the weight changes small enough to allow the learning algorithm to follow the negative gradient of error function.

We used beam search strategy in equation discovery system LAGRAMGE with beam width set to 50 and both heuristic functions (SSE and MDL) with downhill simplex method for constant parameters fitting. The depth complexity parameter was set to 10 and three different context free grammars (from the previous section) were used. The best equation was then chosen that minimizes the RMSE on the test training set.

4.3 Results

The results of the experiments for neural networks and equation discovery system LAGRAMGE are given in Table 1 and Table 2, respectively. The architecture of the MP neural network is represented with $x - y - z$, where x, y and z denote numbers of neurons in first, second and third layer, respectively. $1\overset{p}{-}y\overset{q}{-}1$ denotes FIR-MP neural network architecture with p and q time operators (taps) between corresponding layers and p equals the length of the input vector. The architecture of the recurrent neural network is represented with $x \hookleftarrow y - 1$, where x denotes the length of input vector and y the number of feedback connections.

Recurrent neural networks has the best performance for the Reference voltage and FBM data sets. Both data sets represent time series with very fast changing values without long-term trend. The recurrent neural network has worst performance for the time series with trend. In that case the MP and FIR-MP networks better identify the underlying system, as we can see from the results of the experiments for Lorenz and Sunspots data sets. For Lorenz-like chaos data set the best performance is surprisingly achieved with MP network[2]. For all data sets, expect the Lorenz-like, the prediction performance of different types of neural networks are comparable. In the experiments with Lorenz-like chaos MP is significantly better then other two types.

[2]Finding a simple representation for a complex signal might require looking for relationship among input variables. In the case of Lorenz-like chaos input vectors are representative enough to allow the MP to find the "simple" regression model which is good enough for local description of the model.

Data set	Winning NN			RMSE	
	Type	Architecture	p	Training	Testing
Lorenz	FIR-MP	$1\overset{5}{-}4\overset{2}{-}1$	5	$9.9 \cdot 10^{-4}$	$1.7 \cdot 10^{-2}$
Ref. voltage	Rec.	$6 \hookleftarrow 4-1$	6	0.0646	0.0749
FBM	Rec.	$5 \hookleftarrow 4-1$	5	0.0895	0.0823
Lorenz-like	MP	$6-3-1$	6	0.0153	0.0260
Sunspots	FIR-MP	$1\overset{5}{-}4\overset{4}{-}1$	5	0.09795	-

Table 1: Results of the experiments with neural networks

Data set	Winning equation		RMSE	
	Type	p	Training	Testing
Lorenz	piecewise linear	6	$1.919 \cdot 10^{-6}$	$2.465 \cdot 10^{-6}$
Ref. voltage	piecewise linear	6	0.06375	0.07405
FBM	linear	6	0.08846	0.0827
Lorenz-like	quadratic	3	0.03889	0.05939
Sunspots	quadratic	4	0.103	-

Table 2: Results of the experiments with equation discovery

In the experiments with equation discovery for the Lornez, Reference voltage and FBM data sets the discovered equations are linear. Although the Lorenz data set is obtained with simulating three non-linear differential equations, the interval enclosed in the data set do not expose the non-linearity of the underlying equations (due to the stability of the numerical integration a small time step was chosen). For Lorenz-like and Sunspots data sets quadratic equations were discovered, which was expected because of their non-linearity. The parameter p (number of previous values used for prediction) is significantly smaller in cases where quadratic equations were discovered. As in the experiments with neural networks, for all data sets, expect the Lorenz-like, the prediction performances of different types of equations are comparable. In the experiments with Lorenz-like chaos quadratic equations are significantly better then other two types.

Both methods manifest comparable performance on three data sets. Equation discovery outperforms neural networks for the Lorenz data set, which was expected because of the determinism of the underlying model. Neural networks have better performance on the Lorenz-like and Sunspots data sets where quadratic equations were discovered by LAGRAMGE.

Figure 2 shows the performance of the obtained predictors for different types of neural networks and equations.

5 Discussion

In the paper, we presented the equation discovery system LAGRAMGE that uses context free grammars for restricting the hypothesis space of equations. Background knowledge from the domain of use in the form of function definitions can be used along with common arithmetical operators and functions built in the C programming language. The hypothesis space of LAGRAMGE is a set of equations, such that the expressions on their right hand sides can be derived from a given context free grammar.

In contrast with system identification methods, where the structure of the model has to be provided explicitly by the human expert, LAGRAMGE can use a more sophisticated form of representing the expert's theoretical knowledge about the domain at hand. A context free grammar can be used to specify a whole range of possible equation structures that make sense from the expert's point of view. Therefore, the discovered equations are in comprehensible form and can give domain experts better or even new insight into the measured data. This also distinguishes LAGRAMGE from other system identification methods like neural networks, which can be used for obtaining black-box models, i.e., models with incomprehensible structure.

On the equation discovery side, the presented work is related to equation discovery systems, such as BACON (Langley et al. 1987), EF (Zembowitz & Zytkow 1992), E* (Schaffer 1993), LAGRANGE (Dzeroski & Todorovski 1993) and GOLDHORN (Krizman et al. 1995). However, none of them was applied to the task of time series prediction.

Various architectures of neural networks have already been used for system identification and prediction. Some of them are closely related to the architectures used in this paper (Haykin 1998, Lippmann 1987, Narendra 1990, Pham 1995), and others are different, such as radial basis function (RBF) neural network (Haykin 1998) and Group-Method-of-Data-Handling (Pham 1995). The NARX recurrent neural networks (Alippi 1996, Lin 1996) architecture is suitable for learning long-term dependencies in time series. For the task of short-term prediction, addressed in this paper, learning the local structure is good enough (Narendra

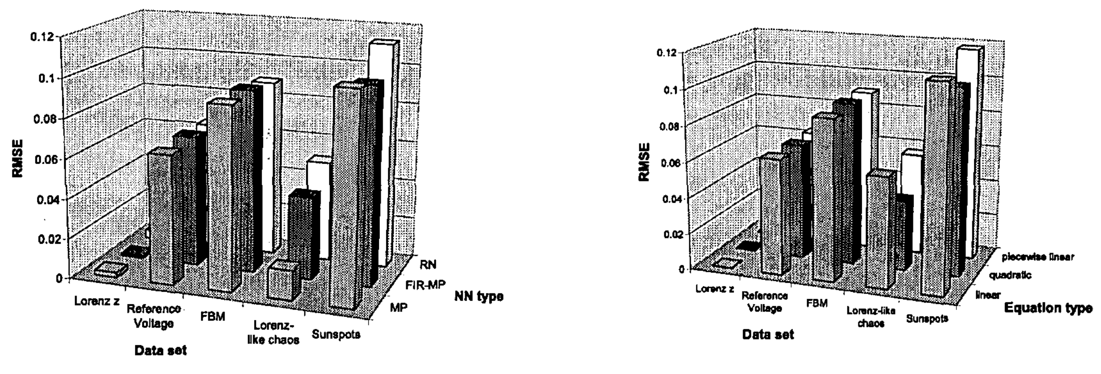

Figure 2: RMSE of the predictors for different types of neural networks and equations

1990). Support vector machine (SVM) for non-linear regression (Haykin 1998), which is approximate implementation of the method of structural risk minimization, could be also used. Finally, SVM may be implemented in the form of a polynomial learning machine, RBF network or MP.

The outcomes of the experiments confirm the potential applicability of both paradigms for time series prediction. For each domain, the performances of the predictors, obtained with both methods, are comparable. The best predictors were obtained for the deterministic Lorenz-z time series. The efficiency of the predictors for real world domains (Reference voltage and Lorenz-like chaos) are comparable. The predictors with worst efficiency were obtained for FBM and Sunspots time series, due to the randomness of the underlying model (FBM) and the small number of measurements available (Sunspots).

The predictive models could be used as a segment of software controlled voltage reference element (VRE), which consists of three main parts: measuring, predictive and control part. Stability of a reference voltage source could be enhanced by implementation of voltage control, which includes a function of correction (feedback loop). This could be done by correction of the current voltage by using a prediction based on measurements made before. It is anticipated for a solid-state voltage reference source to achieve the stability better than 1ppm/1000h (Nancovska 1997).

First step towards the further work will be the comparison of the methods described in the paper with mainstream statistical methods for time series prediction, such as ARIMA or exponential smoothing. It will be of great interest to apply these methods to the variety of data sets from different domains, where some background knowledge from the concrete domain can be used for restricting the equation space. Alternative types of neural networks architectures (NARX recurrent network, SVM for non-linear regression (Haykin 1998)) could be also implemented.

References

[1] Alippi, C., and Piuri, V. (1996) Experimental Neural Network for Prediction and Identification, In *IEEE Transactions on Instrumentation and Measurement*, Vol. 45, No. 2, 1996, pages 670–676.

[2] Džeroski, S., and Todorovski, Lj. (1993) Discovering dynamics In *Proc. Tenth International Conference on Machine Learning*, pages 97–103. Morgan Kaufmann, San Mateo, CA.

[3] Gershenfeld, N. A., and Weigend, A. S. (1992) The future of time series: learning and understanding. In *Time series prediction: Forecasting the future and understanding the past. Proceedings of the NATO Advanced Research Workshop on Comparative Time Series Analysis held in Santa Fe, New Mexico, May 14-17, 1992*, pages 1–70. Addison-Wesley Publishing Company, Reading, MA.

[4] Haykin, S. (1998) *Neural Network - A Comprehensive Foundation*, Second Edition, Macmillan College Publishing Company, Inc.

[5] Hopcroft J. E., and Ullman, J. D. (1979) *Introduction to automata theory, languages, and computation.* Addison-Wesley, Reading, MA.

[6] Hübner, U., and Weiss, C. O., and Abraham, N. B., and Dingyuan T. (1992) Lorenz-like chaos in NH_3-FIR lasers (Data Set A). In *Time series prediction: Forecasting the future and understanding the past. Proceedings of the NATO Advanced Research Workshop on Comparative Time Series Analysis held in Santa Fe, New Mexico, May 14-17, 1992*, pages 1–70. Addison-Wesley Publishing Company, Reading, MA.

[7] Hecht-Nielsen R. (1990) Introduction to backpropagation, In *Neurocomputing*, HNC, Inc. and University of California, San Diego, Addison-Wesley Publishing Company.

[8] Križman, V., and Džeroski, S., and Kompare, B. (1995) Discovering dynamics from measured data. *Electrotechnical Review*, 62: 191–198.

[9] Langley, P., and Simon, H., and Bradshaw, G. (1987) Heuristics for empirical discovery. In Bolc, L., editor, *Computational Models of Learning*. Springer, Berlin.

[10] Lin, T., and Horne, B. G., and Tino, P., and Giles, C. L. (1996) Learning Long-Term Dependences in NARX Recurrent Neural Networks. In *IEEE Transactions on Neural Networks*, Vol. 7, No. 6, November 1996, pages 1329 – 1338.

[11] Lippmann, R. P., (1987) An Introduction to Computing with Neural Nets, In *IEEE ASSP Magazine*, April 1987, pages 4–22.

[12] Mandelbrot, B., and Van Ness, J. W., (1968) Fractional Brownian Noises and Applications, In *SIAM* Rev. 10 (4), 1968, pages 422–436.

[13] Nančovska, I., and Kranjec, P., and Fefer, D., and Jeglič, D. (1998) Case Study of the Predictive Models Used for Improvement of the Stability of the DC Voltage Reference Source, In *IEEE Transactions on Instrumentation and Measurement*, vol.47, no. 6, 1998, pages 1487 - 1491.

[14] Narendra, K. S., and Parthasarathy, K. (1990) Identification and Control of Dynamical Systems Using Neural Networks. In *IEEE Transactions on Neural Networks*, Vol. 1, No. 1., March 1990, pages 4 – 27.

[15] Siegelmann, H. T., and Horne, B. G., and Giles, C. L. (1995) Computational capabilities of recurrent NARX neural network, In Tech. Rep. UMIACS-TR-95-12 nad CS-TR-3408, Inst. of Adv. Comp. Stud., Univ. of Maryland.

[16] Siegelmann, H. T., and Sontag E. D. (1995) On the Computational Power on Neural Networks. In *Journal of Comp. Systems in Science*, Vol. 50, No. 1, pages 132 – 150, 1995.

[17] Pham, D.T., and Liu, X. (1995) *Neural Networks for Identification, Prediction and Control*. Springer-Verlag, London, GB.

[18] Press, W. H., and Flannery, B. P., and Teukolsky, S. A., and Vetterling, W. T. (1986) *Numerical Recipes*. Cambridge University Press, Cambridge, MA.

[19] Schaffer, C. (1993) Bivariate scientific function finding in a sampled, real-data testbed. *Machine Learning*, 12: 167–183.

[20] Todorovski, Lj., and Džeroski, S. (1997) Declarative bias in equation discovery. In *Machine learning. Proceedings of the 14th international conference (ICML'97)*, pages 376–384. Morgan Kaufmann publishers, San Francisco, CA.

[21] Zembowitz, R., and Żytkow, J. (1992) Discovery of equations: experimental evaluation of convergence. In *Proc. Tenth National Conference on Artificial Intelligence*. Morgan Kaufmann, San Mateo, CA.