COMPARATIVE ANALYSIS OF DIFFERENT APPROACHES TOWARDS MULTILAYER PERCEPTRON TRAINING*

A. Valishevsky

Keywords: neural net; time-series prediction; adaptive learning algorithms; backpropagation; QuickProp; Rprop

1. Introduction

Error Backpropagation algorithm appeared in mid-80's and caused a true revolution in the field of neural nets, as it allowed to train multilayer neural networks, which are capable of solving non-linear classification tasks. Just after the introduction of the algorithm, articles showing its weaknesses started to appear. The principal weakness of backpropagation is that a comparatively big amount of iterations has to be made during the learning process. Three fundamentally different approaches towards decrease of the number of iterations are considered in this paper: arbitrary error steepness, employment of the second derivative and an adaptive learning rule.

Share price forecasting task is being considered in order to compare the above-mentioned algorithms. This task can be considered as a non-linear classification task.

There are several approaches towards forecasting of the behaviour of a given system. The first one is based on exact knowledge of regularities that underlie a system. If the regularities can be expressed in terms of precise equations, which can in principle be solved, it's possible to predict the future behaviour of the system. However, the knowledge of the rules governing the behaviour of the system usually is not available. This is particularly true for most macroeconomic problems. The main cause is the non-stationary character of the system parameters. The second approach is based on the creation of statistical models, which approximate the considered system. However, most of these models are stationary and are intended to approximate stationery processes, but the majority of economic processes are non-stationary. The third approach is empirical. It relies on the discovery of empirical regularities in the system. But there're problems as far as the latter approach is concerned, because the regularities are not always evident and are often masked by noise.

Several researchers attempted to use neural networks to search for hidden regularities in macroeconomic systems and to predict future values of the economic time-series [1], [2]. The popularity of the application of neural networks for economic problems solution can be explained by the fact that multilayer perceptrons are capable to find regularities in non-stationary systems [3], and are capable of learning when the training set contains both regularities and exceptions [1].

2. Time Series Prediction

Two approaches to forecasting economic time-series can be considered: a fundamental and a technical analysis. The difference between these approaches is, that in the case of the fundamental analysis the correlating time-series are examined, while the adherents of the technical analysis deem that the economic time-series embody all the knowledge that is

* This research has been financially supported by A/S "DATI"

necessary to forecast its values. The rationale behind the latter approach, also adopted in this paper, is that any changes in economic policy or other indexes are instantly reflected in the securities rate [1].

3. Related Works

Several researchers have attempted to use neural networks to forecast future values of the economic time-series [1], [2] et al. The authors of the mentioned papers study the possibilities of using multilayer neural networks to forecast the values of the time-series, but they use different approaches (both to perform the forecast and to train the neural nets).

In the paper [2] a fundamental analysis is exposed. There are only 3 hidden units in the neural net that was used to make the forecasts. It can be due to the fact that when using a fundamental analysis, the network solves a problem, which is close to the linear classification task [2].

The author uses a QuickProp algorithm in order to train the neural net. After determining the neural network geometry, the attempts were made to decrease the convergence time of the learning algorithm. For this purpose the author increased the value of the learning rate, but as a consequence the convergence time has also increased. It can be caused by the fact, that in this case the gradient step of the learning algorithm increases and the neural network is not able to find an optimal solution.

In the paper [1] the technical analysis is considered. The author uses the error backpropagation algorithm to train the neural net. It's one of the most commonly used learning rules and, as the author notes, it is believed to be an effective learning procedure when the mapping from input to output vectors contains both regularities and exceptions.

The author also notes, that the generalisation abilities of the neural network depend on the number of free parameters. It's desirable to decrease this number, but not to the point when the neural net won't be able to solve the classification task anymore. The number of free parameters is equal to the number of dendrites in the neural network. In order to decrease this number, the number of processing units in the hidden layer can be decreased (the number of the units at input and output levels is determined by other factors). Nevertheless, the author uses the neural net with 32 hidden units.

The author considers two kinds of activation functions: a unipolar function and a bipolar one. The convergence time of the neural net with the bipolar function is about six times faster than that of the network with the unipolar function. Besides that, a forecast of the neural net with a bipolar function is more precise, which means that this net generalises the data better. The analytical data is adduced showing that the convergence time could be reduced by the factor of 9.25 by using the bipolar activation function.

4. Methods

Multilayer neural network with the bipolar activation function is used to forecast the values of the time-series. Four learning rules are used for neural net training: the 'common' error backpropagation and three algorithms, in which three different approaches are applied in order to improve the convergence time (arbitrary error steepness, employment of the second derivative and an adaptive learning rule).

Technical approach is used in time-series analysis and windowing is employed for training set creating.

4.1. Windowing

Windowing is a commonly used method of identifying regularities within a time-series contaminated by noise. The basic idea of this method is to use two windows to look into the dataset: the input window W^i of size n and the output window W^o of size m. It's assumed that a sequence of values $W_1^i,...,W_n^i$ is related in some way to the sequence $W_1^o,...,W_m^o$. In such a way, the vector $W^i \to W^o$ can be used as a training vector.

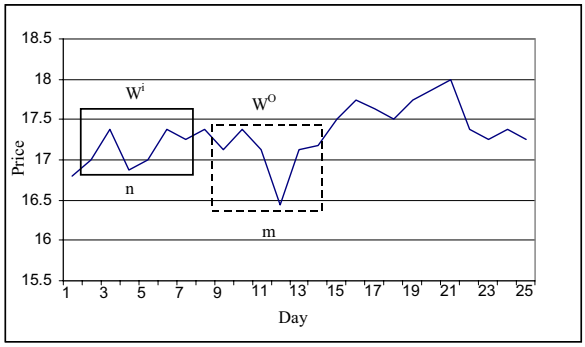

Fig. 1. Windowing - looking for hidden regularities in time series

4.2. Error Backpropagation

The idea behind the error backpropagation algorithm is to minimise the square error at the output level of the neural net. The minimisation is performed using the gradient descent method. Detailed description of the error backpropagation algorithm is given in [3].

4.3. Error Backpropagation with Improved Learning Speed

This algorithm was introduced in [4]. It's based on the error backpropagation algorithm. The main difference between the algorithms is that in the new one the error can be of arbitrary power. More formally, the error at the output of the network is defined in the following way:

$$E_p = \frac{1}{\beta}\sum_{j=1}^{N}\left|d_j - y_j\right|^{\beta}. \tag{1}$$

As the new error definition is introduced, the formulas that determine the synaptic weight modification alter as well. Weight change at the output level is performed in accordance with the expression (2) and in the hidden layer, according to the expression (3):

$$\Delta W_{ij}^{(o)} = \eta \operatorname{sgn}(d_j - y_j)\left|d_j - y_j\right|^{\beta-1} f'(net)x_j \ , \tag{2}$$

$$\begin{aligned} \Delta W_{ki}^{(h)} &= \eta\sum_{j=1}^{N}\delta_j W_j f'(net)x_k \ , \\ \delta_j &= \operatorname{sgn}(d_j - y_j)\left|d_j - y_j\right|^{\beta-1} f'(net) \ , \end{aligned} \tag{3}$$

where η is the learning rate, *sgn()* is the sign function and β is the error steepness coefficient.

Another innovation contained in this learning rule is that the weights are not altered for the learning patterns, which are already convergent. In order to perform it, the threshold γ is introduced, and during every iteration the following condition is being checked: $\max_j\left|d_j - y_j\right| > \gamma$. If the condition is satisfied, the weight update is made; otherwise the weights are not updated and the algorithm processes the next learning pattern.

As the authors note, the speed of convergence of this algorithm is about 2 times faster than that of the error backpropagation algorithm.

4.4. QuickProp

The QuickProp algorithm is probably one of the most well-known speed-up modifications of the error backpropagation learning rule. It was introduced in [5]. In order to accelerate the speed of convergence, the value of the second derivative that characterises curvature of the weight space is considered. The given algorithm differs from the error backpropagation by the fact that in the QuickProp the approximate value of the second derivative is used instead of the momentum term (4):

$$\Delta w(t) = -\eta \frac{\partial E}{\partial w} + \frac{\frac{\partial E}{\partial w}(t)}{\frac{\partial E}{\partial w}(t-1) - \frac{\partial E}{\partial w}(t)} * \Delta w(t-1) . \quad (4)$$

4.5. Rprop

This algorithm was introduced in [6]. **Rprop** stands for '**R**esilient back**prop**agation'. This is an adaptive learning scheme, performing supervised batch learning. Is differs from the algorithms described above by the fact that in this algorithm the partial derivatives are used only to determine the *direction* of the weight step, but not the *size* of the change (the author points to a harmful and unforeseeable influence of the size of the partial derivative on the weight step). Only a sign of the derivative is used to indicate the *direction* of the weight update. The *size* of the weight change $\Delta w_{ij}^{(t)}$ is determined by the so-called 'update-value' $\Delta_{ij}^{(t)}$ (5). The second step of the Rprop learning is to determine the new update-values $\Delta_{ij}^{(t)}$. This is based on a sign-dependent adaptation process (6).

$$\Delta w_{ij}^{(t)} = \begin{cases} -\Delta_{ij}^{(t)}, & if \ \frac{\partial E}{\partial w_{ij}}^{(t)} > 0 , \\ +\Delta_{ij}^{(t)}, & if \ \frac{\partial E}{\partial w_{ij}}^{(t)} < 0 , \\ 0, & else , \end{cases} \quad (5)$$

$$\Delta_{ij}^{(t)} = \begin{cases} \eta^{+}\Delta_{ij}^{(t-1)}, & if \ \frac{\partial E}{\partial w_{ij}}^{(t-1)} * \frac{\partial E}{\partial w_{ij}}^{(t)} > 0 , \\ \eta^{-}\Delta_{ij}^{(t-1)}, & if \ \frac{\partial E}{\partial w_{ij}}^{(t-1)} * \frac{\partial E}{\partial w_{ij}}^{(t)} < 0 , \\ \Delta_{ij}^{(t-1)}, & else , \end{cases} \quad (6)$$

where $\frac{\partial E}{\partial w_{ij}}^{(t)}$ denotes the summed gradient information over all patterns of the pattern set (batch learning).

In words, the adaptation rule works as follows: every time the partial derivative of the corresponding weight w_{ij} changes its sign, which indicates that the last update was too big and the algorithm has jumped over a local minimum, the update-value $\Delta_{ij}^{(t)}$ is decreased by the factor η^{-}. If the derivative retains its sign, the update-value is increased by the factor η^{+} in order to accelerate convergence in shallow regions. Additionally, in case of a change in sign, there should be no adaptation in the succeeding learning step.

The author of the algorithm proposes to use the following values of the parameters:

- η^{-} = 0.5, i.e. if a jump over a minimum occurs, we halve the update-value;
- η^{+} = 1.2, as it gave very good results independent of the examined problem.

Rprop follows the principle of 'batch learning'. It means that weight update and adaptation are performed after the gradient information of the whole learning set is computed.

5. Experimental Set-Up

Experiments are carried out using the following scheme: at the beginning the neural network topology is determined and remains invariant throughout successive experiments. Afterwards the experiments, which help to determine the optimal parameters for each of the considered learning rules, are carried out.

5.1. Windowing

In this paper windowing is used to produce the training set and to help to identify regularities within a time-series. As mentioned above, it's necessary to determine the size of the input and output windows (the sizes of these windows determine the neural network architecture at the input and output levels). Input window of size equal to 1 is used in the experiments. The rationale behind this is to avoid overloading the smooth interpolation properties of the learning procedure [1].

In turn, the size of the output window should be determined empirically, because the underlying regularities in the considered time-series are not known a priori.

5.2. Neural Network Geometry

Size of the output window is equal to 1, but the input window size is not known in advance. Thus, the output layer of the neural net consists of one processing unit, and the number of neurons in the input layer should be determined empirically. Neural nets with 3 to 10 inputs are considered in the experiments, what corresponds to the input window size from half a week to two weeks. The number of neurons in the hidden layer should be determined empirically as well, because it's not known a priori what is the size of the input data internal representation, which is necessary to approximate precisely the regularities in the time-series.

5.3. Error Backpropagation

The following parameters concerning the error backpropagation algorithm need to be determined empirically:

- *steepness of the activation function*, because it determines the steepness of the error surface in weight space;
- *learning rate*, because it strongly influences the learning process;
- and *momentum*, which is used to help to avoid learning algorithm from getting stuck in local minima [7].

5.4. Error Backpropagation with Improved Learning Speed

Besides the learning rate and the momentum, this algorithm is supplemented by two more adjustable parameters: *error steepness* and *threshold*. The authors of the algorithm do not give any details concerning the choice of the given parameters. But judging from the results of the mentioned experiments, it can be concluded that for most problems the optimal value of the error steepness in the range of 1.7 and that of the threshold is in the range of 0.1.

5.5. QuickProp

There's only one adjustable parameter in this algorithm – the *learning rate*. Thus, it's necessary to determine the optimal value only for this parameter.

5.6. Rprop

The Rprop algorithm is probably the most easily adjustable learning rule. Despite the fact that there's a comparatively large amount of adjustable parameters, most of them can be used with the values set by default. As the author of the algorithm points out, slight variations of the values of parameters neither improve nor deteriorate the convergence time. The Rprop algorithm uses the following parameters:

- *initial update-value* Δ_0. The value of this parameter is not critical, as it's adapted as learning proceeds. By default, $\Delta_0 = 0.1$;
- *maximum weight-step*. This parameter sets the upper bound for the weight-step. By default, $\Delta_{\max} = 50.0$;
- *minimal weight-step*. By default, $\Delta_{\min} = 10^{-6}$;
- *increase factor* η^+. Independently of the examined problem, $\eta^+ = 1.2$ gave very good results;
- *decrease factor* η^-. Independently of the examined problem, $\eta^- = 0.5$ gave very good results.

5.7. Comparison of the Results

In order to compare the results, four neural networks designed for forecasting the share price values, but trained with the help of different learning algorithms, are created. Four 15-day long forecasts are produced with the help of each of the neural networks and a comparative analysis of the obtained forecasts is provided.

6. Analysis of the Experiments

The main criterion during the validation of the neural networks is the error value on the test set. As can be seen in Fig. 2, a neural network with seven neurons in the input layer has the minimal error value.

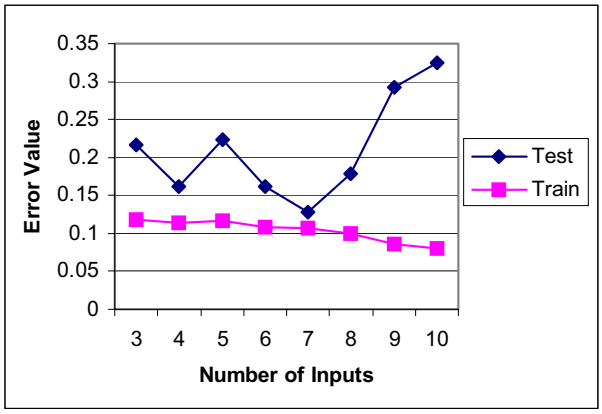

Fig. 2. Neural Nets with Different Number of Inputs

After determining the optimal number of neurons in the input layer, the optimal number of hidden units has been determined. This layer is particularly important, since internal representation of the input data is created in the hidden layer. Error values of neural networks with different number of hidden units are listed in Fig. 3. As can be seen, a neural network with 8 units in the hidden layer has the minimal error value on the test set.

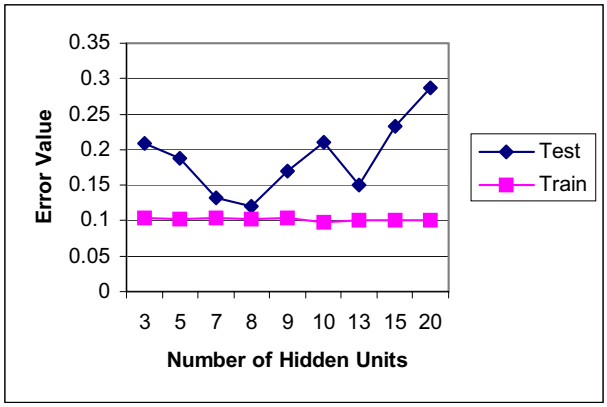

Fig. 3. Neural Nets with Different Number of Hidden Units

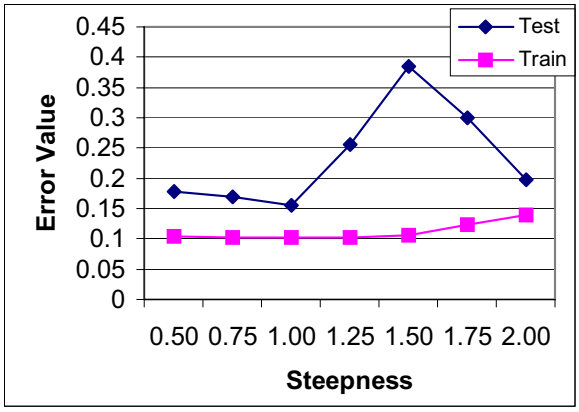

Fig. 4. Different Steepness Coefficients of the Activation Function

After determining the optimal neural network geometry, the optimal parameters for the learning rules have been determined with the help of experiments. Initially the error backpropagation has been considered.

As can be seen in Fig. 4, the optimal value of activation function steepness coefficient is equal to 1.0. In the following two experiments the learning rate and the momentum have been examined. The results are listed in Fig. 5 and Fig. 6. As can be seen, the optimal values of these parameters are equal, respectively, to 0.1 and 0.5.

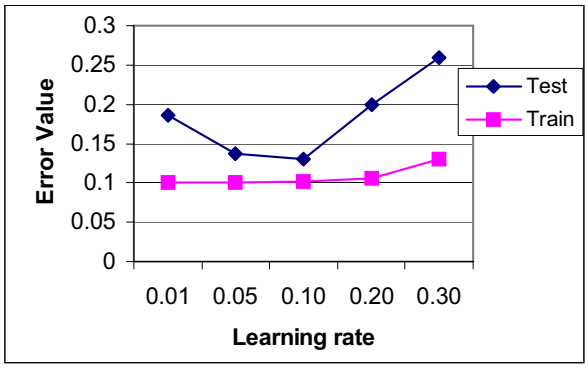

Fig. 5. Experimenting with Learning Rate, Backpropagation

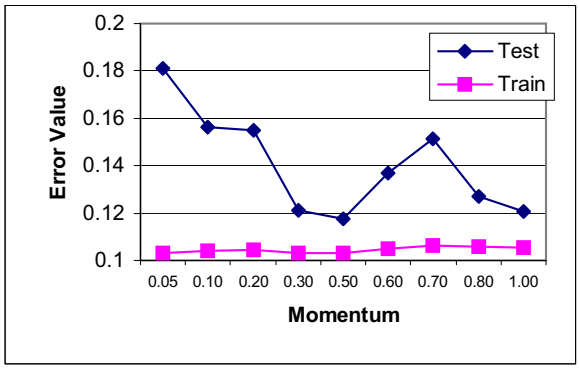

Fig. 6. Experimenting with Momentum, Backropagation

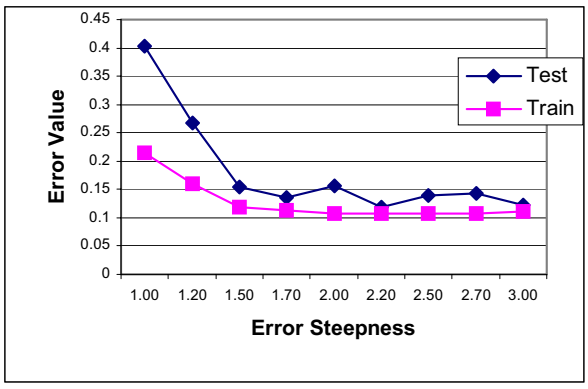

Fig. 7. Experimenting with Error Steepness, Backpropagation with Improved Learning Speed

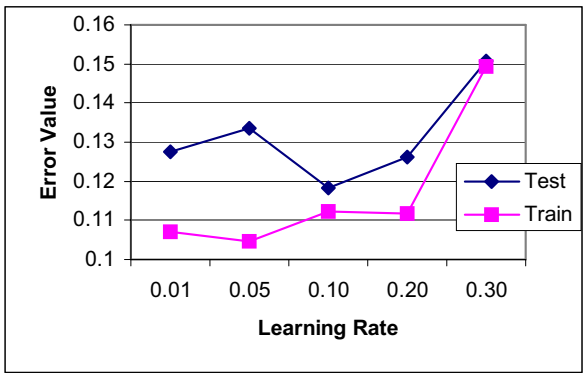

Fig. 8. Experimenting with Learning Rate, Backpropagation with Improved Learning Speed

The backpropagation with improved learning speed has been considered next. Based on the experiment results (Fig. 7 and Fig. 8), it can be concluded, that the optimal value of the error steepness for this algorithm at solving the forecasting task is equal to 2.2 and the optimal value of learning rate is equal to 0.1.

The results of the experiment with the QuickProp algorithm are listed in Fig. 9. The optimal value of the learning rate for this algorithm is equal to 0.05.

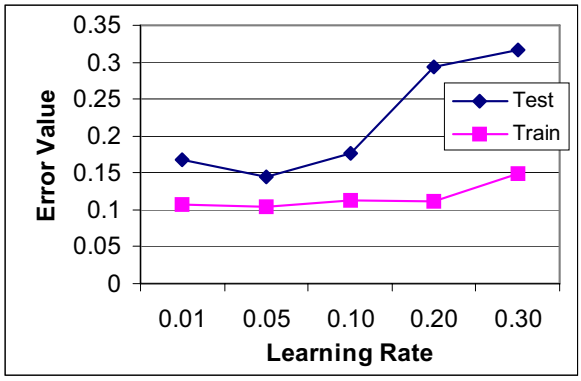

Fig. 9. Experimenting with Learning Rate, QuickProp

This experiment concludes the study of the optimal values of the learning rules. In Fig. 10 convergence lines of the considered learning algorithm are presented.

The advantage of the Rprop algorithm during the learning process is evident, as the convergence is much more rapid compared to other algorithms. It should be noted that in contrast to other learning rules, no experiments have been held in order to adjust its parameters.

Another interesting result is that the convergence time of the QuickProp and of the Error Backpropagation with Improved Learning Speed algorithms, when solving the considered non-linear classification task, is worse than that of the 'common' Backpropagation. The reason of it is probably that the error surface is not smooth and it contains a big amount of local minima. So, the modification of its curvature doesn't improve the convergence time.

And now let's examine the forecasts, which have been obtained with the help of the neural networks, that are described above. Forecasts of the nets trained with the above-mentioned algorithms are shown in Fig. 11.

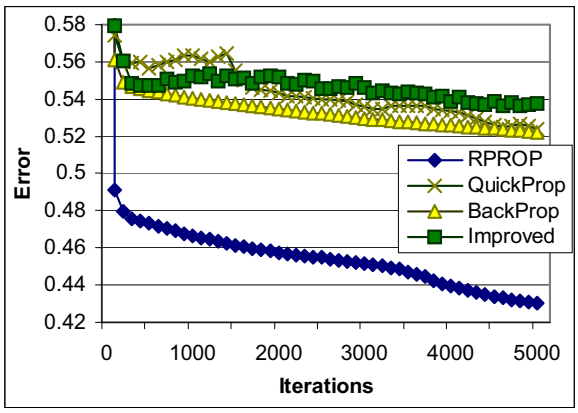

Fig. 10. Convergence Lines

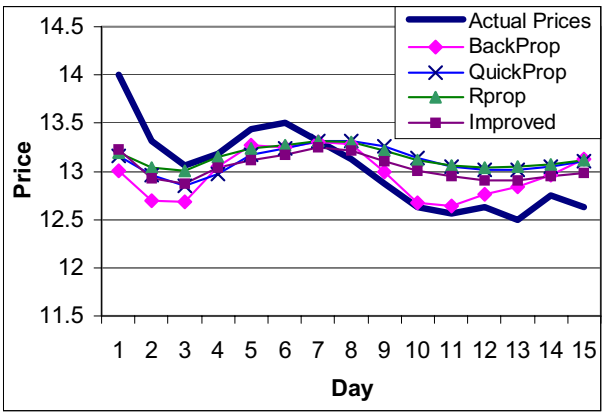

Fig. 11. Forecasts

As can be seen, all three forecasts follow the common trend and downfalls/growth of the values occur virtually simultaneously. Nevertheless, perceptron that is trained with the help of backpropagation predicts changes of the share prices better (quantitatively). As can be seen from the data of Table 1, neural network that is trained with the help of backpropagation has the minimal error value.

Table 1. Errors in Forecasts

	BackProp	QuickProp	Rprop	Improved
Error	*2.178*	*2.508*	*2.276*	*2.205*

From what is said above it can be concluded that not all the algorithms, which during the training process or when solving other, less complicated tasks, give better results than backpropagation, are capable to solve in the same good manner such a complicated classification problem as economic time-series forecasting.

7. Other Results

While performing the experiments, the problem of delay in the forecast has been encountered, i.e. the neural net did correctly forecast the time-series values, but there has been a little delay. Possible causes of this problem are listed below:

- The size of the output window is too small, i.e. the number of input parameters is not enough to find hidden regularities.
- There are too many neurons in the hidden layer, which causes that the neural net memorises training patterns better.
- The value of the activation function steepness coefficient is too high, as a result of which too big gradient steps are made and the algorithm jumps over the optimal solutions.
- The value of the learning rate is too high, which causes too big gradient steps.

It can be noted as well that the value of the momentum coefficient doesn't directly influence the delays in the forecast

8. Conclusions

A comparative analysis of four multilayer neural network learning algorithms is exposed in this work. The comparison is based on the share price forecasting task, which can be related to the class of non-linear classification tasks. As a result, algorithms with a better convergence time and with a more precise forecast have been determined. Nevertheless, it's not possible to give a precise answer to the question, which algorithm is better for a specific task to be solved with the help of neural networks.

Acknowledgement I appreciate useful comments on the manuscript given by Professor Arkady Borisov from Riga Technical University.

References

1. Refenes A.N., Azema-Barac M. et al. Currency Exchange Rate Prediction and Neural Network Design Strategies // In: Neural Computing & Applications, Springer-Verlag, London, 1993 - p. 46-58.
2. Бэстенс Д.-Э., Ван Ден Берг В.-М. и др. Нейронные сети и финансовые рынки // Научное издательство Москва, 1997.
3. Rumelhart D.E., McClelland J.L., eds. Parallel Distributed Processing, Vol. 1 and Vol. 2 // MIT Press, Massachusetts, 1986.

4. Elhadi N., Csefalvay K. Backpropagation with Improved Learning Speed // In: Journal on Communications, vol. XLIV, May 1993 - p. 22-26.
5. Fahlman S.E. A Empirical Study of Learning Speed in Back-Propagation Networks // In: Progress Report CMU-CS-88-162, 1988.
6. Riedmiller M. Rprop – Description and Implementation Details // In: Technical Report. University of Karlsruhe, 1994.
7. Rogers J. Object-Oriented Neural Networks in C++ // Academic Press, 1997.

Alexander Valishevsky
Department of Computer Science, University of Latvia, Raina bulv. 19, Riga, LV-1586, Latvia.
E-mail: sd80022@lanet.lv

Vališevskis A.
Dažādu pieeju salīdzinošā analīze daudzslāņu neironu tīklu apmācīšanā
Šajā darbā tiek veikta četru daudzslāņu neironu tīkla apmācīšanas algoritmu salīdzinošā analīze: kļūdas atgriezeniskās izplatīšanas algoritma un triju algoritmu, kuros tiek pielietotas trīs principiāli atšķirīgas pieejas konverģences laika uzlabošanai. Algoritmu salīdzināšanai tiek izmantots akciju kursa prognozēšanas uzdevums. Darbā tiek noteikta šī uzdevuma risināšanai optimāla neironu tīkla arhitektūra. Pēc tam tiek salīdzināti četru neironu tīklu prognozes, kuriem ir vienāda arhitektūra, bet kuru apmācīšanas laikā tika pielietoti dažādi algoritmi. Īpaša uzmanība tiek veltīta neironu tīklu ieejas datu vispārināšanas iespējām. Tiek aprakstīti vairāki iemesli, kas var izraisīt kavējumu neironu tīklu prognozē.

Valishevsky A.
Comparative Analysis of Different Approaches towards Multilayer Perceptron Training
A comparative analysis of four multilayer perceptron learning algorithms is exposed in this work: the error backpropagation algorithm and three other algorithms with fundamentally different approaches towards the improvement of convergence time. Stock exchange share price prediction is at the basis of the comparison of the algorithms. The optimal neural network topology for the solution of the above-mentioned task is determined in this work. Furthermore the forecasts concerning four neural networks with the same topology, but trained with the help of different algorithms are being compared. Special attention is paid to the generalisation ability of neural networks. A series of reasons, which can cause neural network forecast delay problems, is mentioned.

Валишевский А.
Сравнительный анализ различных подходов при обучении многослойного перцептрона
В данной работе проводится сравнительный анализ четырех алгоритмов обучения многослойного перцептрона: обратного распространения ошибки и трех алгоритмов, в которых использованы принципиально различные подходы к улучшению скорости сходимости. Для сравнения алгоритмов используется задача прогнозирования курса акций. В работе определяется оптимальная архитектура нейронной сети для решения данной задачи, и впоследствии сравниваются прогнозы четырех нейронных сетей с одинаковой архитектурой, но обученных при помощи разных алгоритмов. Особое внимание уделяется способности нейронных сетей обобщать входные данные. Приведен ряд причин, из-за которых может возникнуть проблема задержки в прогнозе нейронной сети.