2

Artificial Neural Networks Numerical Forecasting of Economic Time Series

Michael Štencl and Jiří Šťastný
Mendel University in Brno, Faculty of Business and Economics, Dept. of Informatics
Czech Republic

1. Introduction

The current global market is driven by many factors, e.g. by the facts that we live in the information age and that information is distributed in short times, large amounts and by many data channels. It is practically impossible to analyse all kinds of incoming information flows and transform them to data by classical methods. New requirements call for new methods. Artificial neural networks once trained on patterns can be used for forecasting and they are able to work with extremely big datasets in reasonable time. Traditionally, this is solved by means of a statistical analysis - first a time-series model is constructed and then statistical prediction algorithms are applied to it in order to obtain future values. The common point for both methods is the learning process from samples of past data, or learning from the past. From many of the uncommon points the input conditions for the model creation and the length of the time series pattern set could be pointed out. On one hand, very sophisticated statistical methods exist that have strictly defined input conditions for datasets; on the other hand, practically open input conditions of artificial neural networks can be used. Regarding the length of the time series, the main problem of the Czech Republic, short and middle term predictions are valuable datasets. The lengths of selected economic values are not huge enough for quality of prediction or forecasting. Hand-in-hand with typical problems of real datasets (noisiness and/or missing data), there is the issue of the quality of the numerical forecasting. In addition, the strong nonlinearity of the models leads to an unsolvable usage of classical methods or construction of models that are not representing the reality. These are only few of the difficulties related to economic and financial modelling and prediction. Possible problems of numerous types of the artificial neural networks with *n*-setups make the issue even more complicated.

The aim of this chapter is to compare different types of artificial neural networks using short and middle terms predictions of a real-world economic index. A number of papers dealing with artificial neural networks used for particular problems and often for the test do not use real-world economic indexes.

The chapter is divided into four sections. The first simply presents the introduction to the research domain. The second section describes state-of-the-art artificial intelligence approaches to both prediction and forecasting of economic indexes. In the third section, neural network types and learning algorithms dealing with the prediction of time series and learning optimization are presented. In detail, the third section also includes methods of verification and validation of artificial neural networks and description of real-world economic indexes

used in the experiments. In addition, the experimental approach including methods and strategies for neural network adaption to real-world data approximation and prediction is included as a subchapter. The last chapter presents a number of prediction experiment results of the real-world economic indexes, including the learning process optimization by different learning algorithms, multithread implementation of artificial neural networks. Evaluating the different artificial neural networks types on short and middle term prediction with commented results is another part of the fourth section. Global conclusions end this chapter and give further perspectives for future development of proposed approaches.

2. Artificial Neural Network approach

Artificial neural networks do not need to know the algorithm to reach forecast as in the statistics methods and this makes the main difference (Novák, 1998; Sarle, 1994; Šnorek & Jiřina, 1998). The forecasting of future values with artificial neural networks is based on learned past pattern sets for a defined length. The principle of artificial neural networks is based on learning values from past periods and then approximating the future values. The accuracy of the prediction is influenced by several attributes such as the topology of the selected artificial neural network, the learning rule, the types of activation function, the number of inputs, the length and also the structure of input time series. Globally, there are two main categories of artificial neural network models – feed-forward networks and recurrent networks. A feed-forward network represents a function of its current input; thus, it has no internal state other than the weights themselves. A recurrent network feeds its outputs back into its own inputs. (Russel & Norvig, 2003) These authors formulate learning as an optimization search in weight space. The definition means the reset of the weights of inputs on each input node. Artificial neural networks use two types of learning – supervised and unsupervised. When supervised learning is used a training set for output validation must be supplied. Training sets are used as inputs for the network and the computed outputs are compared with sample results. Weights of all neurons are adjusted backwards according to the output error. The learning algorithm used defines a specific algorithm of resetting the weights. Both genetic algorithm and back-propagation algorithm were tested as the learning algorithms. Back-propagation seems to yield better results in prediction tasks (Štastný & Škorpil, 2007).

One of the basic but also very powerful types of the network is the Multi Layer Perceptron Network (MLP); it belongs to the group of feed-forward neural networks. The configuration variations of MLP networks including the selection of different learning algorithms are a very complex task. The MLP network with the back-propagation learning algorithm (Štastný & Škorpil, 2005) is also one of the most widely used methods in time series forecasting. Often the MLP model is also combined with statistical models in hybrid systems (Tseng & Tzeng, 2002). The MLP model is one of the basic models but often brings very good results. According to the article (Štastný & Štencl, 2008), there has been wide interest in making the comparison study of the published MLP forecast with other models such as the radial basis function (RBF-NN) or competitive networks. The prediction of economic values has its own specifics. First of all, a large number of variables causes a strong non-linear increase in complexity of its analysis and usually the time series (Mostafa, 2009), especially in the conditions of the Czech Republic, is not very long. Another notable problem is incompleteness of and uncertainty in the datasets. Making the models with statistic methods is often impossible, but certain artificial intelligence methods are able to solve that problem.

3. Methods

This part describes two learning algorithms for training MLP networks. Commonly known Back Propagation learning algorithm and Levenberg-Marquardt algorithm are described. Both were selected during previous research (Štencl & Šťastný, 2009; Štencl & Šťastný, 2010; Šťastný & Škorpil, 2005) focused on learning process optimization and both showed the ability to be used for short and middle term prediction of real-world economic time series. As the second type of the artificial neural network, the Radial Basis Function network was selected (Štencl et al., 2009; Štencl & Šťastný, 2009). At the end of the upcoming Results part of this chapter, the comparison of selected artificial neural network results with genetic algorithm approach is presented (Štencl et al., 2009; Šťastný & Škorpil, 2007).

3.1 Neural Networks learning algorithms

First, and most known, the Back Propagation (BP) learning algorithm is the most common learning algorithm for Multi-layer perceptron networks (MLP NN). The algorithm is based on minimizing the error of neural network output compared to the target value. A more detailed description of BP was published previously (e.g. Bishop, 2000; Šťastný & Škorpil, 2005; Štencl & Šťastný, 2009). Standard BP algorithm could be modified to BP with momentum and variable step learning.

The features and usage of the momentum in BP is described in classical BP algorithm. Basically, the algorithm remembers in a parameter the direction in which the current state in error space was reached. This parameter prevents the algorithm from being stuck in local minima. The momentum is added to the learning rule fitting up to changes in weights which are equal to the sum of the recent changes and new variations calculated using BP. The momentum constant defines the effect of the momentum. If the constant is equal to zero, the momentum is ignored, if it is equal to one, the changes are ignored and weights remain the same.

Variable step learning (η) provides learning acceleration. MLP is learning faster with a bigger learning step. However, when the learning step reaches the maximal value, the learning process becomes unstable. Variable step learning is set up by the maximal (initial) learning step, the minimal (final) learning step and the type of the function.

$$\eta = \eta_{\max} - i^{\exp}\left(\frac{\eta_{\max} - \eta_{\min}}{i_{cel}^{\exp}}\right) \tag{1}$$

At the beginning, the maximal (initial) learning step is set to boost the network learning progress. The network looks first for the rough values of weights. After that, the values are decreasing with each learning step according to the selected type of the function. The curve defined by the type of the function connects the maximal and the minimal step value. The type of the curve can be anything from abscissa, over quadratic function to the cubic function. The exponent value determines the decreasing speed at the beginning of the learning process and at the end of the learning process. The value of learning step in the *i*-th iteration is obtained by formula (1); where exp is representing the exponent of the calculated curve, the i_{cet} is the global amount of iteration, $\eta_{\max}$ and $\eta_{\min}$ is the maximal, resp. minimal, learning step.

The momentum and variable learning steps are some of the optimization methods for the BP algorithm. The set-up of the BP algorithm with a momentum and a variable step is presented at Fig. 1.

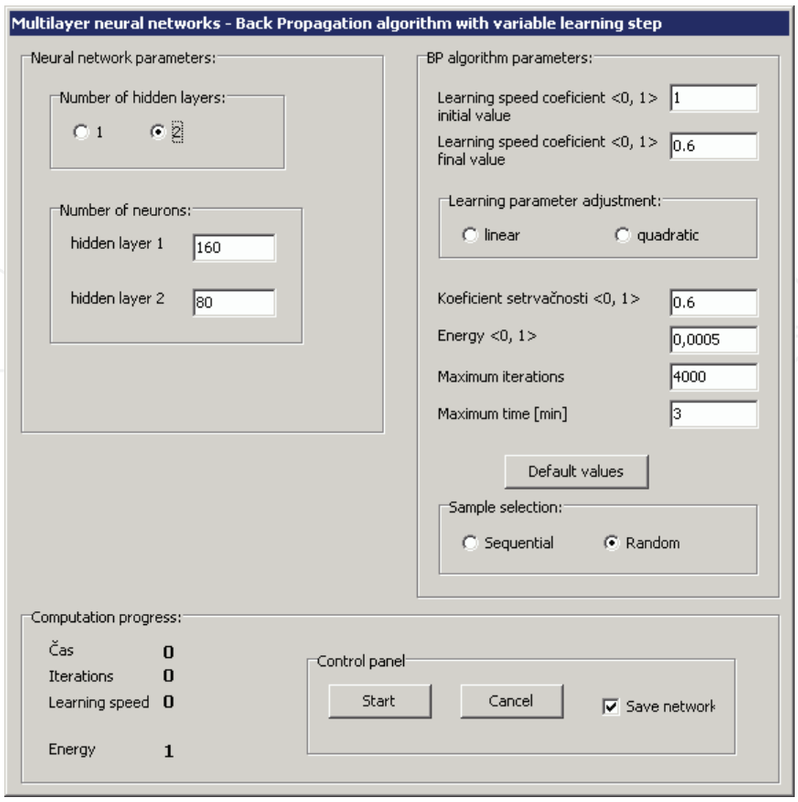


Fig. 1. Set-up window of MLP Network for BP algorithm with a variable learning step

There are also many different techniques including the Levenberg–Marquardt algorithm. The Levenberg–Marquardt method is one of the fastest learning algorithm methods for MLP networks (Hagan & Menhaj, 1999; Sotirov, 2005). The Levenberg-Marquardt (LM) algorithm is an iterative technique that locates the minimum of a multivariate function which is expressed as the sum of squares of non-linear real-valued functions (Sotirov, 2005). It has become a standard technique for non-linear least-squares problems, widely adopted in a broad spectrum of disciplines. LM can be thought of as a combination of steepest descent and the Gauss-Newton method.

As noticed before, the LM algorithm is a variant of the Gauss-Newton method and was designed to approach second-order training speed without having to compute the Hessian matrix (Hagan & Menhaj, 1999). Typically, for the learning of feed-forward neural networks, a sum of squares is used as the performance function. Then the Hessian matrix can be approximated as

$$H = J^T J \tag{2}$$

and the gradient can be computed as

$$g = J^T e \tag{3}$$

where J is the Jacobian matrix (for single neuron shown at (4), where w is vector of the weights, w_0 bias of the neuron, ε error vector) that contains first derivate of the network error with respect to the weights and biases, and e is a vector of network errors (Hagan & Menhaj, 1999; Sotirov, 2005).

$$J = \begin{bmatrix} \frac{\partial \varepsilon_1}{\partial w_1} & \cdots & \frac{\partial \varepsilon_1}{\partial w_n} & \frac{\partial \varepsilon_1}{\partial w_0} \\ \vdots & & \vdots & \vdots \\ \frac{\partial \varepsilon_p}{\partial w_1} & \cdots & \frac{\partial \varepsilon_p}{\partial w_n} & \frac{\partial \varepsilon_p}{\partial w_0} \end{bmatrix} = \begin{bmatrix} x_{1_1} & \cdots & x_{n_1} & 1 \\ \vdots & & \vdots & \vdots \\ x_{1_p} & \cdots & x_{n_p} & 1 \end{bmatrix} \tag{4}$$

The LM algorithm uses the (2) approximation of the Hessian matrix, and the determination of new weight configuration is calculated as follows:

$$w_{k+1} = w_k - \left[J^T J + \mu I \right]^{-1} J^T \varepsilon_k \tag{5}$$

When the scalar μ is zero, the algorithm uses an approximation of the Hessian matrix. When μ is large, this becomes gradient descent with a small step size. Each iteration decreases μ after a successful step, which reduces the performance function and increases μ only when a tentative step would increase the performance function (Hagan & Menhaj, 1999; MathWorks, 2010). Matlab R2010a has an efficient implementation of LM. Because the solution of the matrix equation is a built-in function, its attributes become even more pronounced in a Matlab environment (MathWorks, 2010).

3.2 RBF Neural Networks

Radial Basis Function neural networks (RBF-NN) belong to the feed-forward models of neural networks. RBF-NN consists of three layers of nodes. The first is the input layer that transports the input vector to each of the nodes in the hidden (second) layer. The third layer consists of one node. It sums up the outputs of the hidden layer of nodes to yield the decision value (Wedding & Cios, 1996).

As defined in (Wedding & Cios, 1996; Šíma & Neruda, 1996) the hidden layer of nodes, each node represents a data cluster, which is centred at a particular point and has a given radius and could be named as local unit. As in (Šíma & Neruda, 1996) the local units have the relevant output located at the point in close neighbourhood defined by its parameters. When an input vector goes on each node of hidden layer simultaneously, each node then calculates the distance from the input vector to its own centre (Wedding & Cios, 1996). The MLP nodes, on the other hand, divide the input space into subspaces in which there is a big difference on output.

Radial Basis Functions are used for the approximation and interpolation in numerical mathematics. The approximation process is based on a function. Usually, the linear combination of base functions, here the radial functions, is used. The basis function realizes the transformation of the distance value, calculated from the input vector to its own centre, to the output value of the node. The output value is then multiplied by a weighting value or a constant.

Problems with the creation of RBF-NN consist in the determination of the number of neurons in the hidden layer, the determination of the middles of these neurones and the determination of the neurones width. A powerful method for the determination of the number and quality of neurons of the hidden layer is the algorithm APC-III (Štencl & Šťastný, 2009). This single-pass associating algorithm unlike others uses a constant radial (Ripley, 1996).

```
Algorithm APC-III:
C:      the number of neurons
cj:     the middle of  j-th  neuron
nj:     the number of samples in j-th neuron
dij:    the distance between  xi and j-th neuron

{
       C=1; c1 ← x1; n1 = 1;
          for(i = 2; i =< P; i++) // for every pattern from training set
       {
              for(i = 1; i =< C; i++) // for every neuron
              {
                calculatedij;
                if(dij=< R0) // insert xi into j-th neuron
                {
                      cj = (cjnj + xi)/(nj + 1);
                      nj = nj + 1;
                      break;
                      };
              };
              if(xi is not in any neuron) // create new neuron
              {
                      C = C + 1;
                      cc ← xi;
                      nc= 1;
              };
       };
};
```

Fig. 2. Algorithm APC-III symbolic implementation (Štencl & Štastný, 2009)

The learning process consists in a precept of the given network to answer correctly to an entire training set. As the hidden layer was in this network represented by so-called areas and the middles of the areas are fast added to it, the learning process oversimplifies only to the setting of scales and thresholds of the output layer. The gradient method and the Least Mean Square (LMS) method were tested for learning of the neuron network.

The gradient method uses relations derived for the outgoing layer for algorithm Back-Propagation (BP). In contrast to the BP method, this method only optimizes scales and thresholds of the outgoing layer.

In the learning stages, the network

1. estimates centres of c_j with $x(t)$,
2. estimates widths b_j,
3. determines the weights w_{sj} of input neurones ($x(t)$,$y(t)$).

At the first stage, centres c_j are determined for each RBF unit. The centres c_j are represented by the weights between the input and the hidden layer. For example, the algorithms for cluster analysis are used. To speed up this stage, non-adaptive methods can also be used such as uniform or random distribution of RBF neuron centres over the input space. The second stage sets up other values of RBF neurons. The setup values of RBF units (b_j) determine the wideness of the area around estimated centres of c_j. The objective of the third stage of learning is to determine the weights of input neurons, for example the least square method or gradient algorithms can be used.

In global view, the RBF-NN learning includes the unsupervised learning at the first stage. At the second stage the setup of RBF units is made. The typically used function for this setup is the Gaussian Radial Basis Function (Šíma & Neruda, 1996) defined as in (6).

$$\varphi(x) = e^{-\left(\frac{\|x-c\|}{b}\right)^2} \tag{6}$$

The RBF unit determines the important output values in the radial zone with the centre in c; b represents the width of φ and determines the size of the radial zone. The setup parameters of RBF units determine the wideness of the controlled area and affect the generalization capability of the network. If the parameters are smaller it means a lower generalization capability; on the other hand, for a wider area the units lost their local mean.
At the last stage, the supervised learning is used. The last stage sets up the weights w_{sj}. The setup is made by mineralization process of typical error function (Šíma & Neruda, 1996).
After the learning process the RBF-NN is ready to approximate training sets and also provide good results for answers outside the training set. Different techniques for regularization have been discussed for a long time. For example, Bishop (1991) works with the same RBF units as training patterns. This technique brings a uniform resolution and wideness of the Gaussian function, provided that the input data have the same time of generation.

4. Results

All of the above described methods are applied to solve the prediction of real numerical time series represented by Czech household consumption expenditures. The tested dataset includes twenty-eight observations between the years 2001 and 2007. The observations are represented by quarterly data and the goal is to predict three future values for the first three quarters of 2008. As the second real-world economic index, the Czech Republic Goods transport indexes were used. The data are originally from the Czech Statistical Office and are measured quarterly. The length of the time series is 32 units and represents quarters of the years 2000 and 2008. The number of data used to train networks and to test network is a representative for the generalization ability testing of each selected method. The predicted values of both experiments are compared with the measured values. In the next step, a comparison of neural network topology efficiency with respect to learning algorithms is made. The used dataset includes all of the previously specified problems of the real-world economic index. Conditions for all the experiments remain defined as following:

- the tests were performed at the same hardware with monitoring of the kernel processes to keep them on the same level
- all the comparison tests were performed in Matlab 2010a environment with the neural Network Toolbox
- the same input, targets, validation and cross-validation datasets are used
- the short and middle term prediction is performed.

The parameters compared are based on previous research and they reflect the objective comparison with other methods. The first of the parameters is the output precision measured with the Mean Square Error and the Normalized Mean Square Error respectively. The second comparison parameter is the absolute comparison of the output value with the absolute value of the specific index.

The results of the experiments identify the main differences in the used neural networks architectures together with numerical forecasting of a real-world economic index. The detected differences are then verified by means of practical comparative examples.
The most important and also difficult part of working with ANN generally is the right architecture setup of selected network. For the multi-layer networks the number of hidden layers and units is one of the most important setups. The number of hidden units and layers estimates the flexibility of the network to approximate nonlinearization in the data. For estimating the "right" number of hidden layers and unit there is no universal approach. Hastie (2001) claims that the choice of the number of hidden layers is guided by background knowledge and experimentation. Typically, the number of hidden units is somewhere in the range of 5 to 100, the number increasing with the number of inputs and number of training cases (Hastie et al., 2001). Kecman (2001) gives a reasonable approach when deciding about the MLP networks architecture. We should specify the cost function for the neural network performance, including the size of the neural network learning time, implement ability in hardware, accuracy achieved, and the like (Kecman, 2001). Based on author's experience, the pseudo system approach has been applied in presented experiments. We covered all areas defined by Kecnam (2001) and used automated implementation of testing different numbers of hidden units and layers. The system approach starts with a huge number (it depends on the number of inputs) of hidden units, decreasing in several steps depending on the generalization capability. The presented architectures brings best setups of selected neural networks models.

4.1 Learning Algorithms comparison

The first set of experiments was performed using our own modification of MLP network with BP learning algorithm tested through Matlab R2010a environment. The second part of the experiment was performed using the MLP network with LM algorithm implementation in Matlab R2010a based on the previously described principles. The dataset was divided into input data and validation data in order to obtain better generalization of the results.
Both networks described in the following section are based on one-hidden-layer architecture. The stopping criterions for learning process were normalized mean square error (MSE), and total amount of epochs set to 2000 epochs. The observed values for comparison are the total amount of learning epochs and the number of neurons in the hidden layer respecting the specified learning algorithm. In order to compare the MLP network using the BP learning algorithm and the MLP network using the LM learning algorithm, the dataset was divided into a training set (70 % of the dataset) and a test set (remaining 30 % of the dataset). The training set was then divided into the training set itself (80 %) and a validation set (20 %). The performance of both experiments was evaluated for the test set. For the purposes of the experiment, the comparison starts at the third quarter of 2002.
The comparison process stops at the fourth quarter of 2007. The prediction is made for the first three quarters of 2008. The computed results of both MLP networks are then compared with real measured values. The hardware used for both experiments was an Intel Core i5, with 2.66 GHz and 4 GB of RAM. The operating system was the UNIX based Mac OS X. Both experiments had the same conditions so that the learning process is not affected.
The first setup is made by the MLP network with the BP learning algorithm. The architecture had twenty neurons in the hidden layer and the learning process ended when the stopping criterion defined by MSE (valued to 0.01) was reached after 890 epochs. The

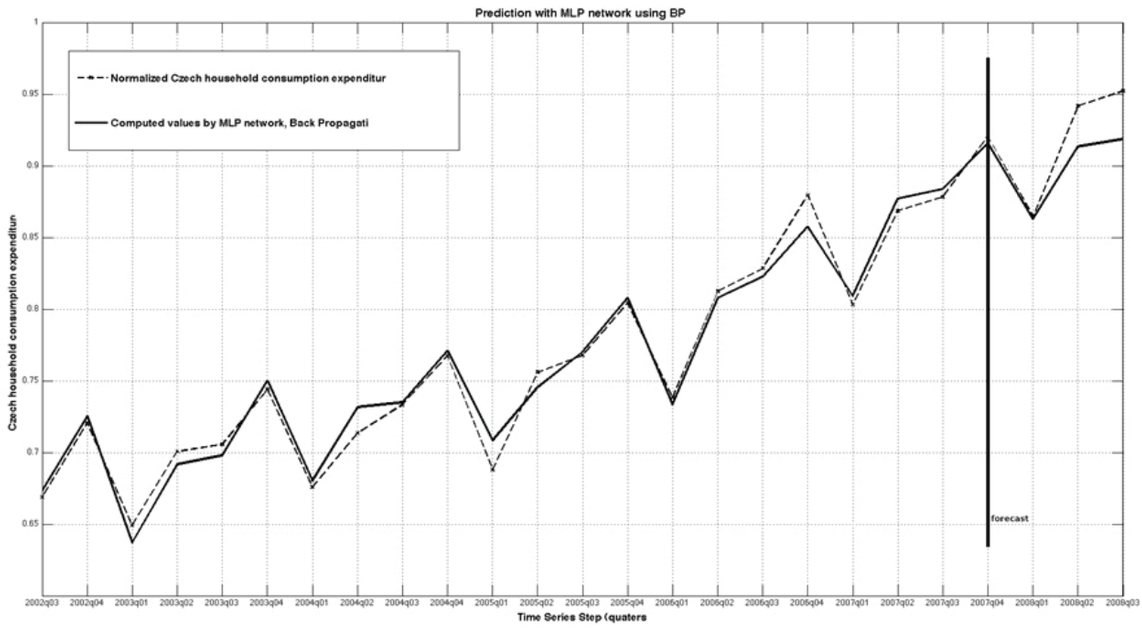


Fig. 3. Prediction with MLP network using BP learning algorithm (Štencl & Štastný, 2010)

results of the experiment are presented at Figure 3. The dashed line represents the original real input values and the solid line represents the calculated output of the MLP network with the BP learning algorithm. The prediction starts with the first quarter of 2008 (as indicated by the solid line and description forecast). The difference between the calculated value and the real value for the first quarter of 2009 is 0.00169. With the next two values the prediction gains more error.

The second setup works with the MLP network using the LM learning algorithm. The architecture was different because the LM learning algorithm works with smaller networks topologies. The test was performed with more network architectures. The best result, by reaching the stopping criterion of MSE, has been reached by the MLP network with ten (10) neurons in the hidden layer. The performance of the learning and the validation processes is presented at Fig. 4. The plot presents training, validation and test values calculated using mean square errors (MSE). The MLP network reached the stopping criterion on epoch 20 with the best validation performance value of 0.0058 (MSE).

Figure 4 represents the comparison of the values calculated by the MLP network with the LM learning algorithm implemented in Matlab R2010a (by function *trainlm*). Again, the dotted line represents normalized real Czech household consumption expenditure. The solid line represents the calculated values of the MLP network with the LM learning algorithm. Again, the comparison starts at the third quarter of 2002 and ends with the last quarter of 2007. The prediction is made for the first three quarters of 2008. When comparing the real and the calculated values of the MLP network, the main difference is in the first quarter of 2008, where the empirical variance is -0.076. For the next calculated values the empirical value decrease to vales of -0.00047 and 0.00074 respectively.

Both networks ended the learning process on reaching the first stopping criterion defined by normalized mean square error (MSE). The first difference is in the architecture of the MLP networks. When using the LM with twenty neurons in the hidden layer, the network over fits the data. By decreasing the number of neurons in the hidden layer, it brings better

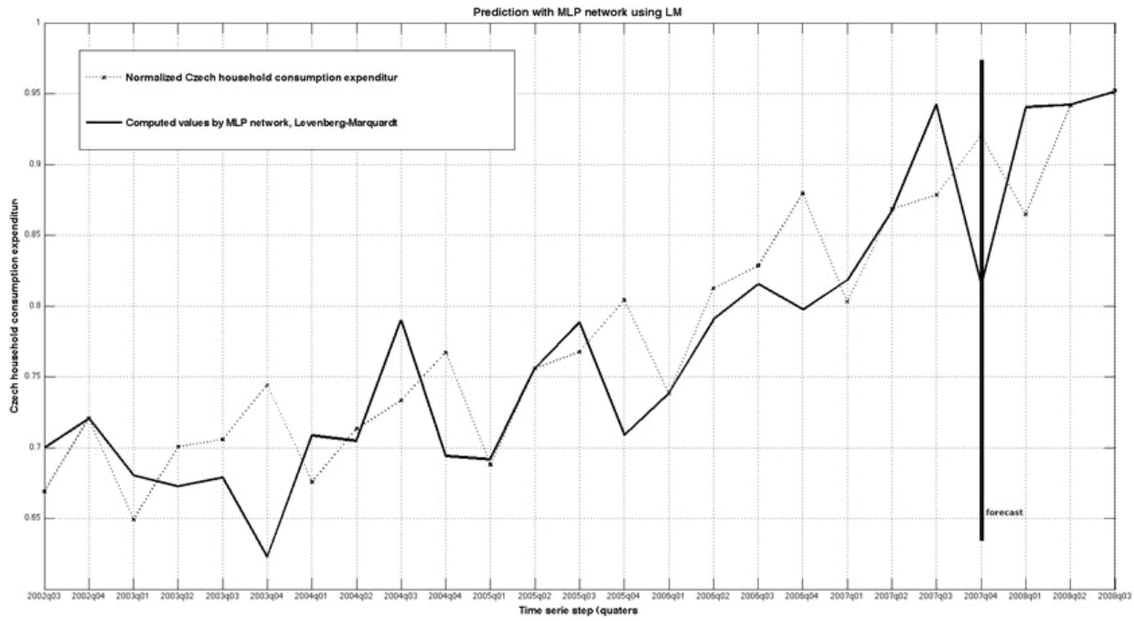


Fig. 4. Prediction with MLP network using LM learning algorithm (Štencl & Šťastný, 2010)

results. With ten neurons in the hidden layer the stopping criterion has been reached. Further decrease in the number of neurons in the hidden layer did not enhance the network error. Another difference is in the learning time. The MLP network with the BP needed 890 epochs to reach the stopping criterion defined by the mean square error. The MLP network with the LM finished the learning process after twenty epochs by reaching the stopping criterion. Regarding just to these two facts the LM algorithm seems to be more effective even on the time series models with limited data.

The MLP network with the BP brings better approximation (as presented at Fig. 2) and with a better prediction for the first quarter of 2008. Then the prediction error increases continually. The MLP network with the LM did not fit with the target data as well as the MLP network with the BP, but it computed better results for the second and the third quarters of 2008.

We can conclude by the fact that it is either possible to retrieve the results in short time (using the LM algorithm), or use the standard learning algorithm to obtain competitive computed values of the used dataset. The prediction results of the LM algorithm for the second and the third quarters of 2008 are definitely positive within the experiments. Both methods generally agree on the future values of the time-series.

4.2 Multithread MLP-NN implementation

This experimented was motivated by the learning process optimization on the computational level. Taking into account the results of the first experiments, we expected a better approximation ability of data. The multi-threaded calculation may allow faster computation (the value is lower than the maximum number of epochs) with a greater range of the network. The expectations of the multi-thread experiment results were defined as the achievement of more accurate values for the middle-term prediction in fewer learning epochs. The total number of epochs was chosen as a criterion because of relativity of time as evaluation criterion. The main evaluation criterion remains the MSE as used within the previous experiments to reach a qualified comparison.

For the multi-thread calculation, the open source Java framework Joone has been used (available at http://sourceforge.net/projects/joone). The chosen Joone function was executed through the Matlab 2010a environment to keep the experiment conditions. The calculation was performed on the workstation with an Intel Core i5 with a frequency of 2.66 GHz and 8 GB memory. The aim of the tests was to identify the key indicators for the optimization of the prediction of multithreaded real data with previously defined expectations. For multithreaded computations it was necessary to choose a greater range of MLP network topology. Based on the cross-validation a network with three layers with 5 neurons in the first hidden layer (linear activation function), 15 neurons in the second hidden layer (sigmoid activation function), 5 neurons in the second hidden layer (sigmoid activation function) and one neuron in output layer was selected for testing. As a training evaluation criteria the overall network error (set to the desired value 0.001) and the maximum number of epochs of network training (1000 epochs) were selected.

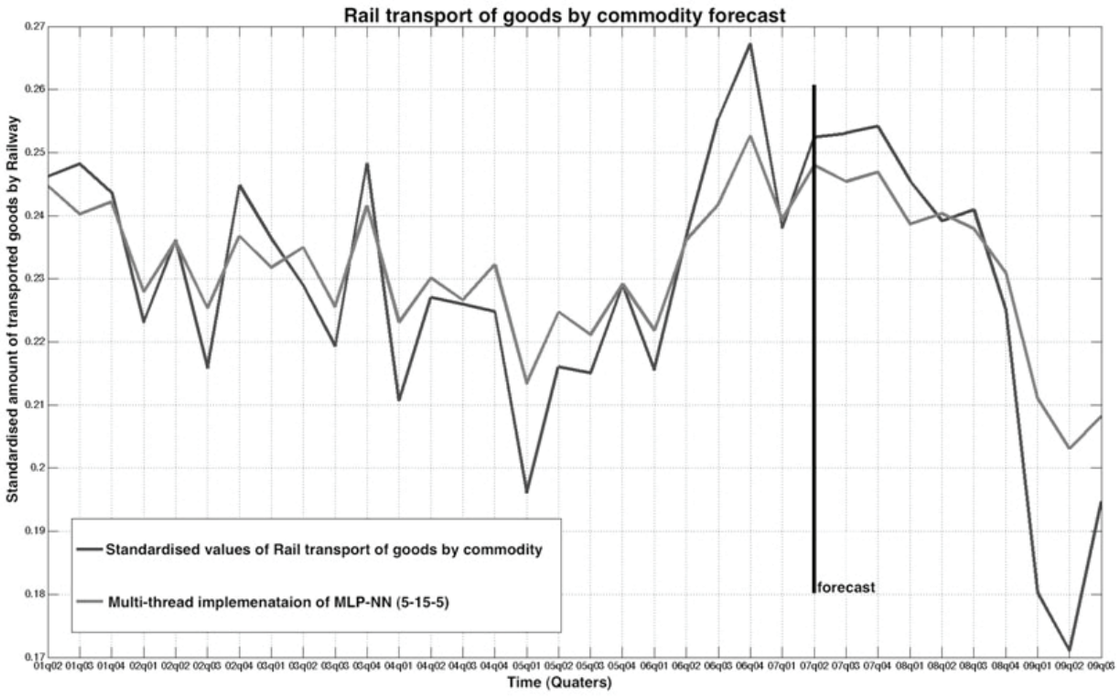


Fig. 5. Multi-thread prediction of standardised amount of transported goods by railway

Experiment Time series consists of goods transported by railway in as long-term trend of individual indicators by months and years as Volume indicators. Experiment dataset contains tons of goods transported by railways in the Czech Republic. The values where standardized to the interval <0, 1> for better adaptation of ANN. The length of the validation set was 35 observations (the last value of the second quarter of 2007). The forecast was set at 5 future values. The resulting values were compared with the real values from upcoming quarters in 2007 and 2008. The training of the ANN ended at 1320 epoch with a total network error (MSE) of 0.00098. The subsequent validation of the real data demonstrates the ability to predict the mean square error of 0.0012, which corresponds to the learning setup.

The validation result of the comparison experiment is shown at Figure 5. The figure includes the comparison of the computed values with the real-values of the selected index. The comparison starts at the second quarter of 2007 (signed as *forecast*). Networks of larger scale have a better capability of approximation for large datasets. A positive benefit of our experiment is the good ability of the trained network to successfully predict the trend of real-world index of a small input dataset.
The resulting total number of epochs in training may be due to the nature of the training dataset. Another positive conclusion is the ability of the network to predict the huge decrease in the future as is shown between the third quarter of 2008 and the second quarter of 2009. In this period the index fell down to the extreme – the global minimum. The trained network was, in spite of the unexpected decrease, able to forecast. The expectations of this experiment were not confirmed. Working with the multi-thread implementation did not optimize the learning process by decreasing the number of epochs with reaching the defined learning error. But the experiment confirmed a better ability of generalization of the multi-layer MLP implementation than the single layer implementation used in previous experiments.

4.3 Radial basis function experiment

The experiment consists of comparison tasks of the RBF NN implementation in Matlab R2007a and MLP NN used in (Štencl & Štastný, 2008). Figure 6 describes the standard topology of the Matlab RBF NN implantation algorithm. The experiment dataset consists of selected consumption expenditures time series violated by a random constant. The customization of the selected value has been made to get diverse time series model simulating different variations.

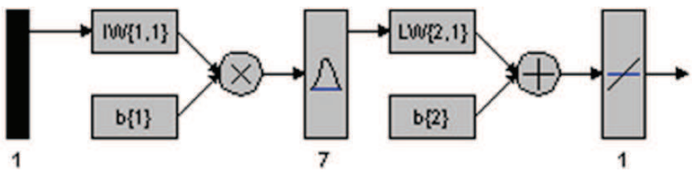


Fig. 6. Matlab RBF-NN topology (Štencl & Štastný, 2009)

The final model consists of 40 observations representing quarters of the years from 1998 to 2008. The absolute variety in the dataset is 3. The forecast is made for five future periods.
The stopping criterion for learning process was the normalized mean square error (MSE), and the total amount of epochs was set to 2000 epochs. The learning criterions are identical to the previous experiment because of the comparison aspect. The observed values for comparison are the total amount of learning epochs and the number of neurons in the hidden layer with respect to the radial basis neural network learning process. The learning process of the network was much faster than in the case of the MLP network with the Back-propagation algorithm.
The obtained values are not as exact as in the case of the MLP network. Precisely in this case, Matlab implementation does not allow a better configuration of the network. Figure 7 shows the performance progress for the Matlab implementation of RBF NN. The performance value for the MLP NN was 0.23124. The performance value for the Matlab was 0.6973.
Concluding this experiment, the RBF network should be used more for the short-term prediction. For smaller datasets there is a notable huge noisiness in the data computed by

RBF networks. In spite of that, RBF networks produce good results for datasets containing approximately fifty observations. For datasets with fewer observations, RBF networks are unable to approximate the selected data as the MLP networks in our case. As shown at figure 7, both types of ANN have shown a good generalization ability resulting in a better prediction for the MLP NN when reaching a smaller error. The RBF networks confirmed a better endurance for architecture changes than the MLP NN during the experiment and faster learning adoption.

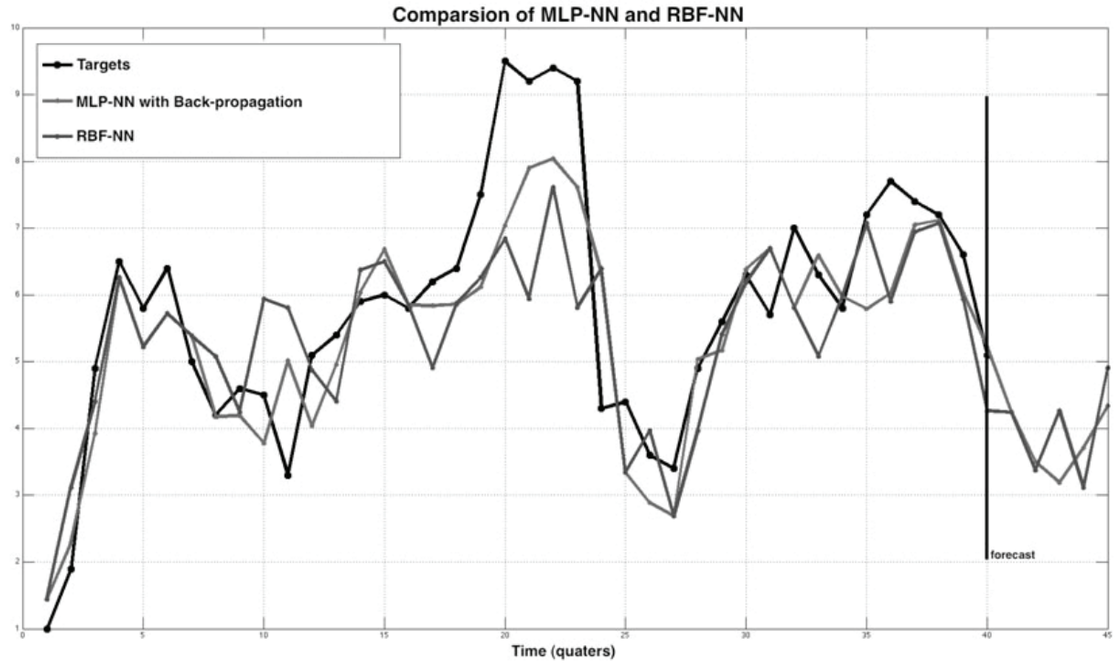


Fig. 7. Comparison of MLP-NN and RBF-NN (Štencl & Šťastný, 2009)

4.4 Comparison of RBF NN and two-level grammatical evolution

The experiment was previously published as (Štencl et al., 2009). The sample dataset for experiment consists of 40 observations defined at part 4.3; values lie in interval <1, 9.5>. The first part of the experiment was performed using RBF NN implementation in Matlab R2007a based on the previously described principles. The dataset was divided into input data and validation data in order to obtain better generalization of the results. The second part of the experiment was conducted using our own implementation of two-level grammatical evolution (Popelka, 2007).

Figure 8 shows the input data and two sample runs of both methods. Both methods generally agree on the future values of the time-series. The output of neural network is shown only with the values displayed, the output of the grammatical evolution is both the values displayed and the formula that makes the main difference between both the approaches. But if we focus on the learning time efficiency, the grammatical evolution provides us with more information, but its training would take much more time than training a neural network. In this simple case it would be about 10 generations of the underlying genetic algorithm, which takes about 40 minutes (about 4 times longer) using the same computer.

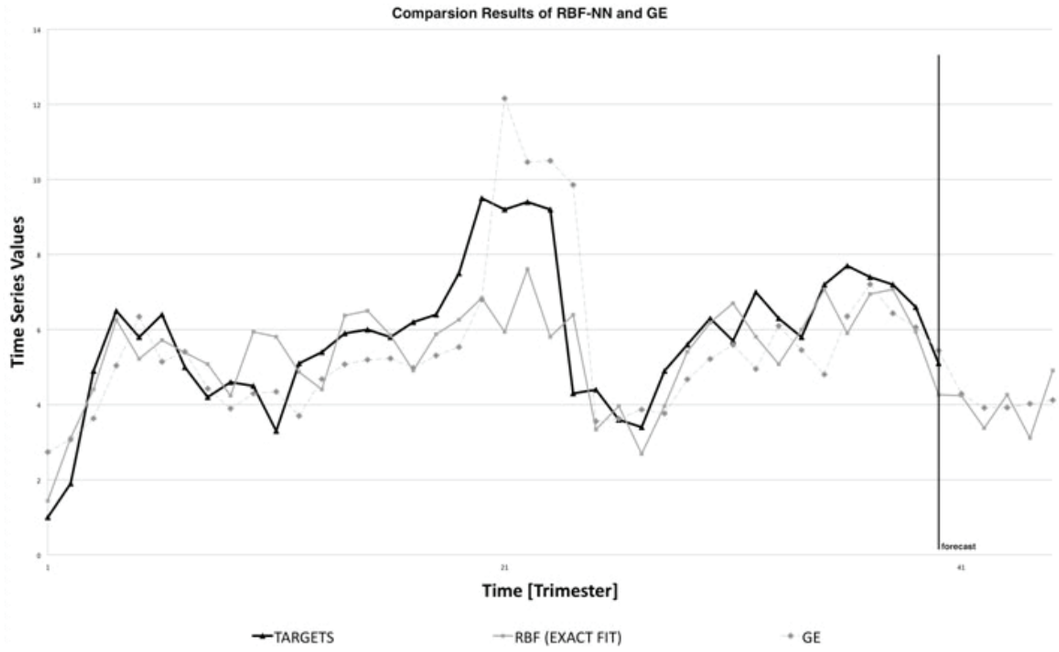


Fig. 8. Comparison of trained RBF NN and two-level grammatical evolution (Štencl et al., 2009)

Both methods provide comparable results in terms of accuracy. Briefly this could be described that it is either possible to obtain numerical results in a short time, or use a more complicated algorithm to obtain autoregressive formula of the time-series.

5. Conclusion

In this chapter we have presented comparison studies of different artificial neural networks on prediction of the real-world economic index. For this type of tasks the statistical analysis is used. For the statistical prediction, a time-series model must be constructed first and then the prediction is made. The set of input conditions is strictly defined which limits the generalization ability. The resulting forecast is then affected also by the noisiness and the length of the input dataset. The length of economic datasets is the typical problem of Czech conditions. Thus, the length of the input dataset and the noisiness, joined with the strong nonlinearity of the input models lead to an unsolvable usage of classical methods. Some of the mentioned difficulties can be reduced by using selected Artificial Neural Network models.

We evaluated the selected Artificial Neural Networks using different types of networks models by means of a systematic approach. The selection is based on a state-of-the-art analysis and also author's experience. The evaluation of the selected ANN is based on previously published papers with added consequences between them. For the experiments the real-world economic time series has been used. The input datasets include typical problems mentioned before. All the predictions are based on the comparison with real values of selected time-series model. For comparison purposes a few values of the dataset where excluded from the learning process to be compared later with the computed values.

The basic, but also very powerful, type of the ANN is the multi-layer perceptron network (MLP NN). The network architecture is the key element for an effective usage of the MLP

NN, together with the learning method. As we proposed in the text, there is no guaranteed approach to setup the right network architecture. For our experiment we used the systematic approach similar to cross-validation. We also tested different learning algorithms as part of the learning process optimization. The results of the experiments show better approximation of MLP NN with Back-propagation learning rule. The training of the MLP NN with Levenberg-Marquardt learning rule results in overfitting of the data. The comparison of both approaches leads to conclusion that both methods are effective. We also add an experiment with the multithread MLP NN. The expected results were not confirmed because of the maximal number of epochs (which was one of the comparison aspects). The multi-thread implementation provides a better generalization ability than in case of single thread computation.

As the second type of ANN used for prediction, the Radial Basis Function neural networks have been tested. The RBF NN generally provide a good forecasting ability. The RBF NN had a worse approximation ability than the MLP NN. Another interesting point is the endurance of the RBF NN for the architecture changes. The computed values show noisiness in the resulting values. This is probably produced because of the length of the input dataset. The RBF NN typically needs to have more data for better generalization. In the last experiment of this work we have compared the RBF NN with the two-level grammatical evolution. The experiment results confirmed the ability of both approaches with the main difference in time efficiency. We can conclude that the RBF NN are more effective for retrieving the values of the prediction.

Generally, we can conclude that these approaches have a good potential for short and middle term predictions of real-world economic index. We have also confirmed good efficiency of ANN when working with short or missing datasets. When comparing the different type of the ANN, the MLP NN showed the best generalization ability. The RBF NN is better to use with a longer input dataset, but they are more effective for obtaining the numerical values of time-series model as another Artificial Intelligence approach (a genetic algorithm in our case).

In future work we would like to use artificial neural network methods on different real-world data. Newly, the analysis of the business cycle opens a new application area for different types of artificial neural networks. A thorough analysis including the comparison with selected statistical methods shows that possible inconsistencies in the prediction of used methods can be described and quantified. The scope of further research will be also focused on testing of different architectures including the multi-thread implementations which show a very good generalization ability. The research will also include the modelling with Bayesian Networks.

This work has been supported by the Research design of MENDELU in Brno number 116/2101/IG1100791.

6. References

Bishop, C. M. (2000). Neural Networks for Pattern Recognition, *Oxford University press*, ISBN 0-19-853864-2.

Hagan, M.T. & Menhaj, M. (1999). Training feed-forward networks with the Marquardt algorithm, *IEEE Transactions on Neural Networks*, Vol. 5, No. 6, pp. 989-993.

Hastie, T.; Tibshirani, R. & Friedman, J. (2001). The elements of statistical learning; data minig, inference, and prediction, *Springer series in statistics*, ISBN 978-0387-95284-0.

Kecman, V. (2001). Learning and soft computing; support vector machines, neural networks, and fuzzy logic models, *A Bradford Book, The MIT Press*, ISBN 0-262-11255-8.

MathWorks™, Neural Network Toolbox Documentation. [online] Available at <http://www.mathworks.com/access/helpdesk/help/toolbox/nnet/backpro7.html#8119>

Mostafa, M. M. (2009). Consumer credit scoring models with limited data, *Expert Systems with Applications, 36*, DOI: 10.1016/j.eswa.2008.06.016.

Popelka, O. (2007). Two-level Optimization using Parallel Grammatical Evolution and Differential Evolution, *Proceedings of MENDEL'2007*, Prague, Czech Republic, pp. 88-92. 2007. ISBN 978-80-214-3473-8.

Sotirov, S. (2005). A Method of Accelerating Neural Network Learning, *Neural Process. Lett*. 22, 2 (Oct. 2005), 163-169. DOI= http://dx.doi.org/10.1007/s11063-005-3094-9.

Štencl, M.; Popelka, O. & Štastný, J. (2009). Time Series forecasting using machine learning methods, *Proceedings of 12th international Information Society*, pp. 66-69, ISSN 1581-9973, Ljubljana, Slovenia, October 2009, Jožef Stefan Institute, Ljubljana.

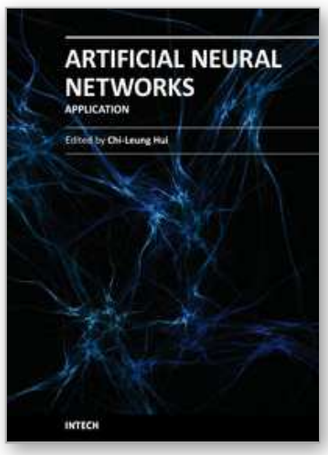
Štencl, M. & Štastný, J. (2010). Neural network learning algorithms comparison on numerical prediction of real data, *MENDEL 2010, 16th International Conference on Soft Computing*, pp. 280-285, ISSN 1803-3814, Brno, Czech Republic, Jun 2010, Brno University of Technology.

Štencl, M. & Štastný, J. (2009). Advanced approach to numerical forecasting using artificial neural networks. *Acta Universitatis agriculturae et silviculturae Mendelianae Brunensis,* Vol. 6, No. 2, (09/2010), ISSN 1211-8516.

Štencl, M. & Štastný, J. (2008). Nové metody predikce numerických dat, *Aktuálne manažerske trendy v teórii a praxi*, 1st Edition, Žilina: Žilinská univerzita v Žiline, EDIS, 2008, s. 28-33. ISBN 978-80-8070-966-2.

Štastný, J. & Škorpil, V. (2007). Genetic Algorithm and Neural Network. *WSEAS Applied Informatics & Communications*, ISSN 1790-5117.

Štastný, J. & Škorpil, V. (2005). Neural Networks Learning Methods Comparison. *International Journal WSEAS Transactions on Circuits and Systems,* Vol. 4, No. 4, ISSN 1109-2734.



Artificial Neural Networks - Application

Edited by Dr. Chi Leung Patrick Hui

ISBN 978-953-307-188-6
Hard cover, 586 pages
Publisher InTech
Published online 11, April, 2011
Published in print edition April, 2011

This book covers 27 articles in the applications of artificial neural networks (ANN) in various disciplines which includes business, chemical technology, computing, engineering, environmental science, science and nanotechnology. They modeled the ANN with verification in different areas. They demonstrated that the ANN is very useful model and the ANN could be applied in problem solving and machine learning. This book is suitable for all professionals and scientists in understanding how ANN is applied in various areas.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen