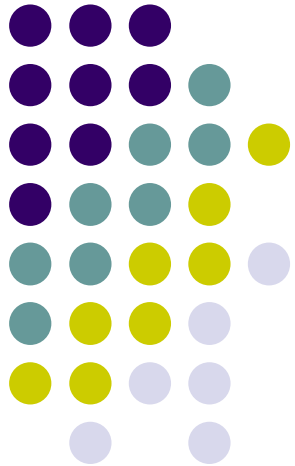
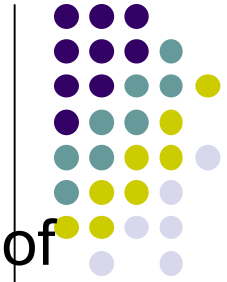
Neural network model for electricity demand prediction

DUŠAN MARČEK
Department of Applied Informatics,
Faculty of Economic, VŠB-
Technical University of Ostrava,
Dusan.Marcek@vsb.cz

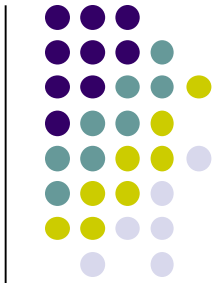


Outlines



- Basic principles of identifying input-output functions of systems and forecasting
- An overview of statistical and CI methods for time series modelling and forecasting
- Statistical modeling
- SVR model
- Neural Approach (classic NN)
- Neural Approach (RBF NN)
- Genetic learning algorithm implementation
- Parameter settings
- Results and empirical comparison
- Conclusion

Basic principles of identifying input-output functions of systems and forecasting



There are three major approaches to forecasting

- **Explanatory modelling and forecasting.**
- **Time series**
- **In a special case machine learning, CI such as SV regression or ANN offers relatively new ways for improving forecast method.**

An overview of statistical and CI methods for time series modelling and forecasting

Statistical *CI (SC)*

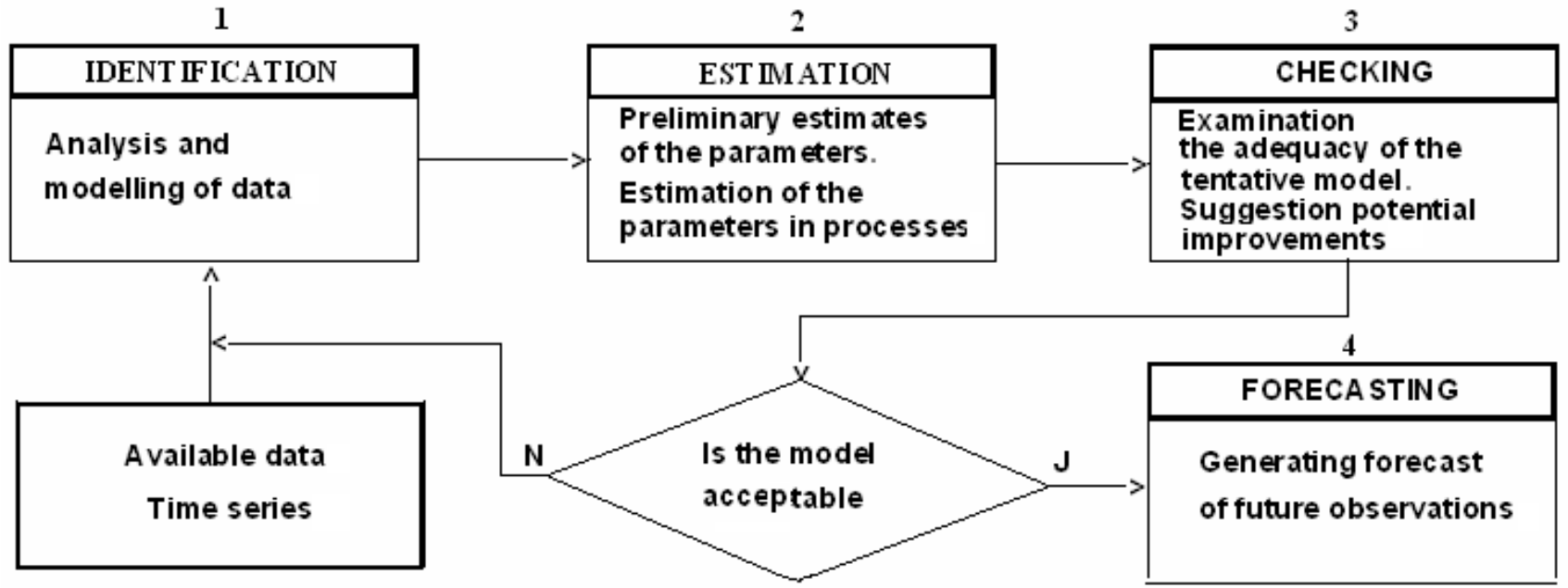
Standard regression/ econometric models	Without or with Seasonal comp.	Integration of 3 IT:	ANN
			Fuzzy logic systems
Latest models: State-Space Models (K. filtration)	Structural models. EC, VEC models		Machine learning
Transfer Function M.		ANN:	Learning
Models: ARIMA,		Perceptron-type	classic BP
ARCH-GARCH		RBF NN	soft (fuzzy logic)
		Fuzzy NN	Granular
Special models:	Bayes M.	Convolution NN	GA, MGA, Hybrid Deep learning
		Recurrent	
		Believe	
- ARIMA/ARCH-GARCH models require more costs of development, - there is not a convenient way to modify or update the estimates of the model parameters as each new observation becomes available and one has to periodically completely develop and refit the model.		Computational NN offer exciting adwantages such as learning, adaptation, fault-tolerace, parallelizm, generalization.	

Statistical modeling, time series approach

Box and Jenkins developed a new modeling approach based on time series analysis and derived from the linear filter known as AR or ARIMA (AutoRegressive Integrated Moving Average) models. The basic ARMA model of orders p, q (ARMA(p,q)) has the form:

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ \dots \ + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \ \dots$$

The basic steps in Box-Jenkins Procedure are: (1) Identification, (2) Estimation, (3) Diagnostic checking, (4) Forecasting. See next figure



Flow chart of building an appropriate time series forecast model

Statistical modeling, ARCH-GARCH: Models

ARCH (GARCH) model for time sequence $\{y_t\}$ in the following form:

$$y_t = \nu_t \sqrt{h_t}, \qquad h_t = \alpha_0 + \sum_{i=1}^{m} \alpha_i y_{t-i}^2 + \sum_{j=1}^{s} \beta_j h_{t-j}, \tag{4}$$

Nelson (1991) proposed the following **exponential GARCH model** abbreviated as **EGARCH** to allow for leverage effects in the form:

$$\log h_t = \alpha_0 + \sum_{i=1}^{p} \alpha_i \frac{|\varepsilon_{t-i}| + \gamma_i \varepsilon_{t-i}}{\sigma_{t-i}} + \sum_{j=1}^{q} \beta_j h_{t-j} \tag{5}$$

γ_i denotes the coefficient of leverage effects (see Zivot and Wang (2005, p. 243))

The basic GARCH model can be extended to allow for leverage effects. This is performed by treating the basic GARCH model as a special case of the **power GARCH (PGARCH)** model proposed by Ding, Granger and Engle (1993):

$$\sigma_t^d = \alpha_0 + \sum_{i=1}^{p} \alpha_i (|\varepsilon_{t-i}| + \gamma_i \varepsilon_{t-i})^d + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^d \tag{6}$$

where *d* is a positive exponent, and γ_i denotes the coefficient of leverage effects (see Zivot and Wang (2005, p. 243)).

Statistical modeling

The Auto-Regressive (AR) process is a linear combination of previous values, the Moving-Average (MA) process is a linear combination of previous errors, 'I' is an operator for differencing a time series. An ARMA(p, q) model of orders P and Q is defined by

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \ldots + \theta_q \varepsilon_{t-q}$$

A logical extension of the ARMA process is a seasonal model abbreviated as ARMA(P, Q)$_s$ process in the form

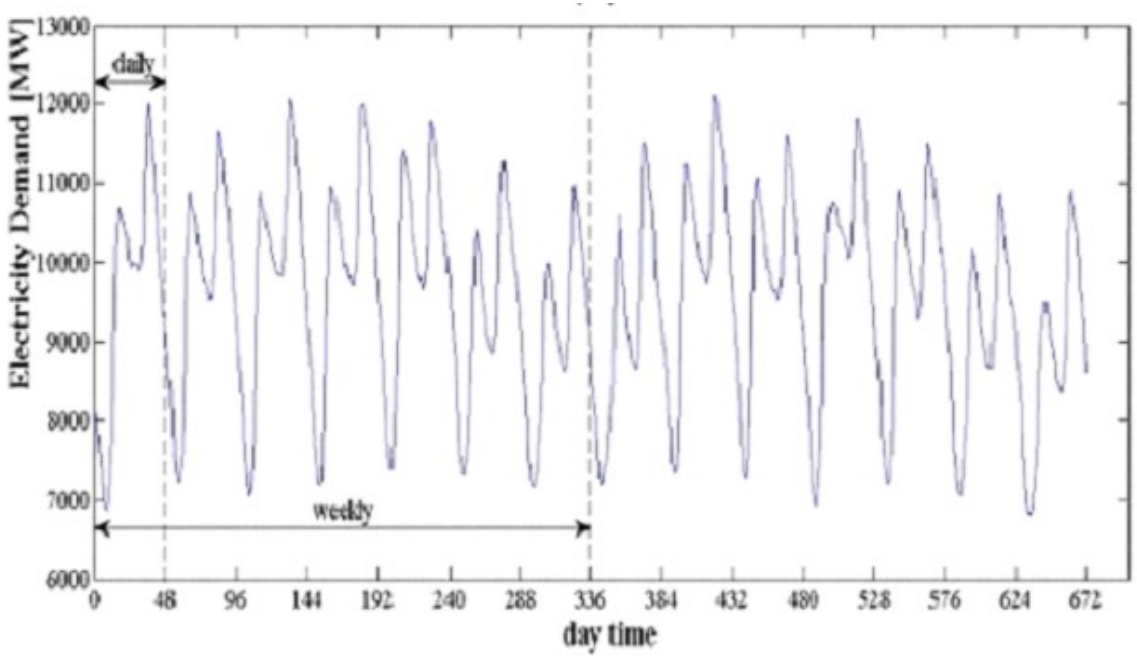
$$y_t - \lambda_1 y_{t-s} + \lambda_2 y_{t-2s} + \ldots + \lambda_P y_{t-Ps} = \gamma_1 \varepsilon_{t-s} - \gamma_2 \varepsilon_{t-2s} - \ldots - \gamma_Q \varepsilon_{t-Qs}$$

where the subscripts s denotes the number of seasonal cycles.

To illustrate the statistical methodology, consider half-hourly electricity demand data for the state of New South Wales in Australia for three months. It contains 4416 electricity demand values recorded at half-hourly intervals.

Statistical modeling

Electricity demand data recorded at half-hourly intervals shows two main cycles: daily and weekly.



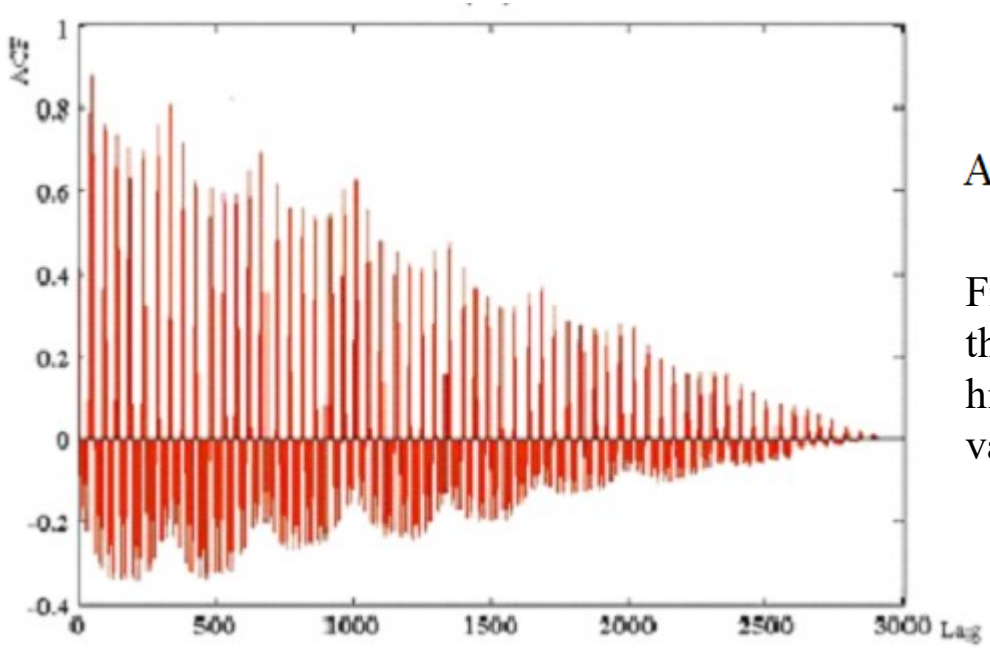
Electricity demand data for 2 consecutive weeks

The daily pattern shows that the demand is lowest at 4:30am and then reaches its first maximum at 9:30am and its second maximum at 6:30pm, in agreement with the human routine. The weekly pattern shows that the same days of the week (e.g. Mondays) have similar demand profiles. The weekly and daily cycles are 336 and 48 half-hour periods, respectively.

.

Statistical modeling

The primary tool used in identification process is Auto Correlation Function (ACF). Actually, the theoretical ACF is unknown and must be estimated by the sample ACF.



Autocorrelation function for the training set

Fig. shows the sample autocorrelation function for the training data; values close to 1 and -1 indicate high positive and negative linear correlations and values close to 0 indicate lack of correlation.

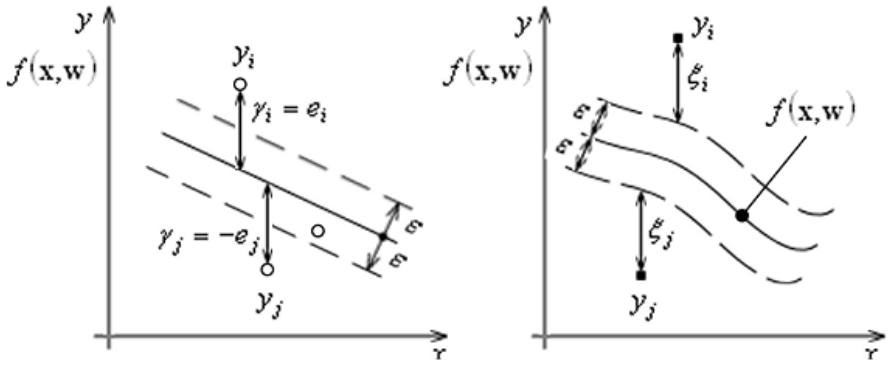
$$\text{ARIMA}(p,d,q)(P,D,Q)_s = \text{ARIMA}(6,1,1)(1,1,1)_{48}$$

$$y_t(1-\lambda_1 B-\lambda_2 B^2-\ldots-\lambda_6 B^6)=(1-\gamma B^{48})\varepsilon_t$$

where B is the backward-shift operator defined as $By_t = y_{t-1}$, $B^j y_t = y_{t-j}$

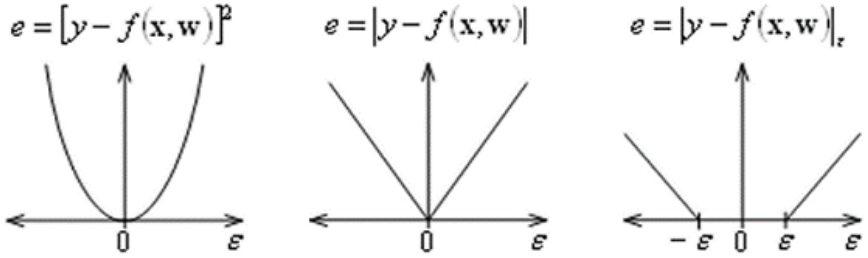
The parameters of the model above were estimated by Maximum Likelihood procedure in e-views software (http://www.eviews.com).

Causal Models Experimenting with non-linear SV Regression



Using an ε-insensitive loss function

$$|y - f(\mathbf{x},\mathbf{w})|_\varepsilon = \begin{cases} 0 & \text{if } |y - f(\mathbf{x},\mathbf{w})| \le \varepsilon, \\ |y - f(\mathbf{x},\mathbf{w})| - \varepsilon & \text{othewise} \end{cases}$$



SVR model

SVR models may take the following forms

$$\begin{cases} f(\mathbf{y},\mathbf{w},b) = K(\mathbf{x}_i^T,\mathbf{x}_j)\mathbf{w} + b & \text{or} \\ f(\mathbf{y},\alpha,b) = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)K(\mathbf{x}_i^T,\mathbf{x}_j) + b. \end{cases}$$

where $K(\mathbf{x}_i^T,\mathbf{x}_j)$ are relevant kernel functions

$(\mathbf{x}_i^T,\mathbf{x}_j)$, are training data, b is a real constant (bias).

The real constants are obtained from the solution of the following quadratic programming (QP) problem

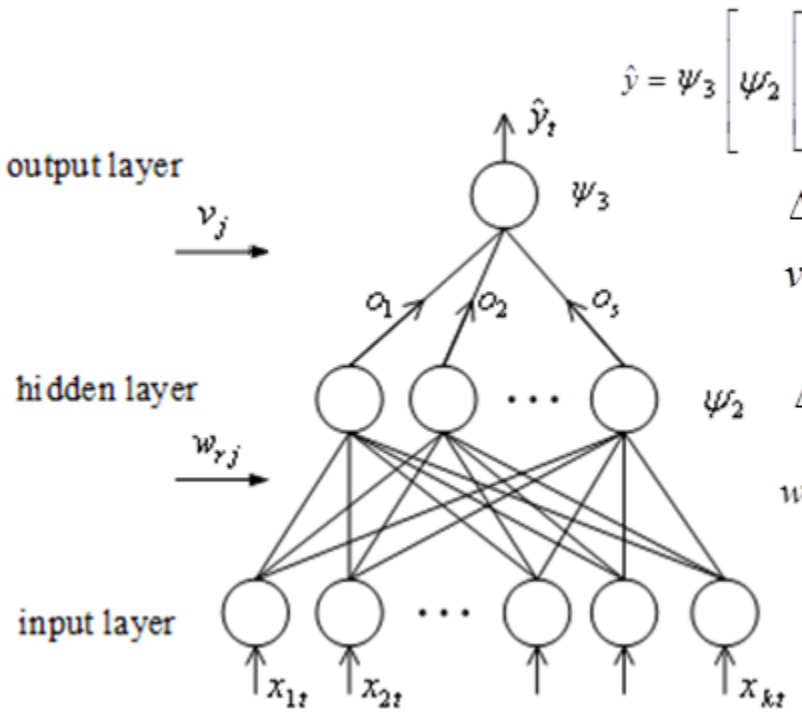
$$\begin{cases} \max\ L(\boldsymbol{\alpha},\boldsymbol{\alpha}_\mathbf{i}^*) = \\ -\frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\,K(\mathbf{x}_i^T\mathbf{x}_j) - \varepsilon\sum_{i=1}^{n}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{n} y_i(\alpha_i - \alpha_i^*) \end{cases}$$

subject to constrains: $\sum_{i=1}^{n}(\alpha_i^* - \alpha_i) = ($, $0 \le \alpha_i^* \le C, i = 1,$, $0 \le \alpha_i \le C, \quad i = 1,n$

where L is the Lagrangian with Lagrange multipliers α_i, α_i^*

In the SV regression, to estimate its parameters the user must further choose some attributes that affect their estimates. These are the following attributes: measure of error approximation (Loss Function ε), the regularization and weights vector norm C, kernel function K and its degree

Neural Approach (classic NN)



$$\hat{y} = \psi_3\left[\psi_2\left[\begin{pmatrix} w_{11} & \cdots & w_{1k} \\ \vdots & \ddots & \vdots \\ w_{s1} & \cdots & w_{sk} \end{pmatrix} \times \begin{bmatrix} x_1 \\ \vdots \\ x_k \end{bmatrix}\right] \times \begin{bmatrix} v_1 \\ \vdots \\ v_s \end{bmatrix}\right]$$

$$\Delta_3 = (y - \hat{y})\dot{\psi}$$

$$v_j^{new} = v_j^{old} + o_j \Delta_3 \quad \text{for } j = 1, 2, \ldots, s$$

$$\Delta_{2j} = \Delta_3 \psi_2'(u_j)\, v_j^{old}, \quad \text{for } j = 1, 2, \ldots, s$$

$$w_{rj}^{new} = w_{rj}^{old} + \Delta_{2j} x_r, \quad \text{for } j = 1, 2, \ldots, s;\ r = 1, 2, \ldots, k$$

$$c_j^{(t+1)} = c_j^{(t)} + \lambda(t)\,(x^{(t)} - c_j^{(t)})$$

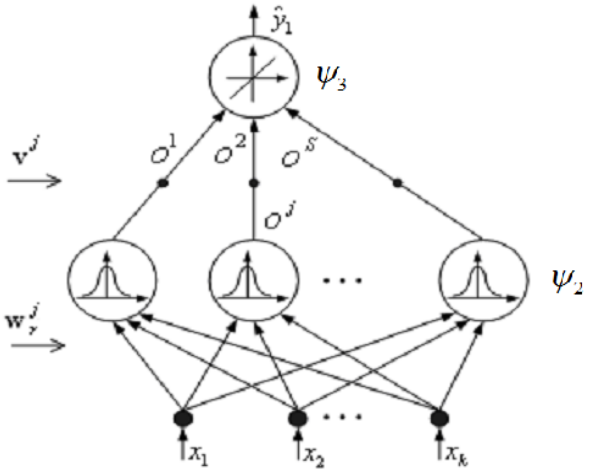
Fig. 3. An example of the three layer feed-forward network notation for units and weights with architecture $k - s - 1$

The structure of a neural network is defined by its architecture (processing units and their interconnections, activation functions, methods of learning and so on). In Figure above, each circle or node represents the neuron. This neural network consists an input layer with input vector $\mathbf{x}$ and an output layer with the output value $\hat{y}_t$. The layer between the input and output layers is normally referred to as the hidden layer and its neurons as RBF neurons. Here, the input layer is not treated as a layer of neural processing units .

Typically, the updating process is divided into epochs. Each epoch involves updating all the weights for all the examples. This algorithm can be generalized to other network architectures such as for networks with more hidden layers than 1.

In Figure 3, we have omitted any thresholds from our description, they can be treated as connections to an input terminal that has permanently the value equal to -1 with connections weights $w_{0j} = \Theta_r$.

Neural Approach (RBF NN)



$$\psi_2(u^j) = e^{\frac{-(x-w^j)^T \Sigma^{-1}(x-w^j)}{2}}$$

$$u^j = \| \mathbf{x} - \mathbf{w}^j \|^2 \quad \text{for } j = 1, 2, \ldots, s$$

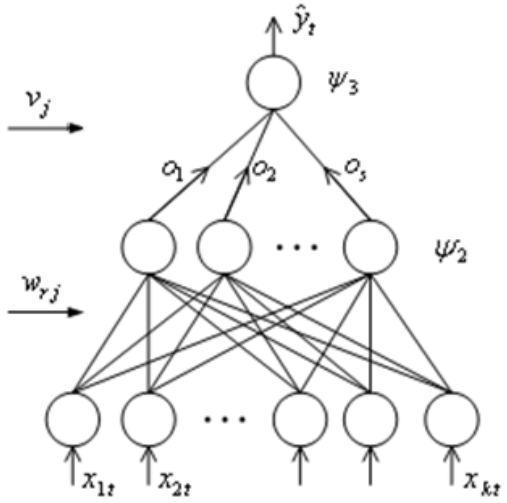
In the hidden layer the weights $\mathbf{w}^j$ represent the centers $\mathbf{c}^j$

To finds the weights or centers of activation functions we used the adaptive (learning) version of the *K*-means clustering algorithm.

$$c_j^{(t+1)} = c_j^{(t)} + \lambda(t)\,(x^{(t)} - c_j^{(t)})$$

NN model trained by GA algorithm

Back-propagation algorithm is based on the gradient method. A disadvantage of the gradient method is slipping into local minimum.



RBF neural network architecture

In the RBF neural network we used instead of back-propagation learning algorithm the genetic learning algorithm.

In neural networks, the weights may be adapted by genetic algorithms (GA). E.g. to predict half-hourly 1-step ahead-electricity demand time series Australian data the genetic algorithm was proposed [7] to train the weight v_j of the output neuron in soft RBF (Radial Basic Function) network.

Genetic algorithms (GA) are implemented as a computer simulation in which a population of abstract representations (called chromosomes) of candidate solutions (called individuals) to an optimization problem evolves toward better or faster solutions

The common phases of GA flow are initialization, elitism, selection, crossover, mutation, new population and termination

Genetic learning algorithm implementation

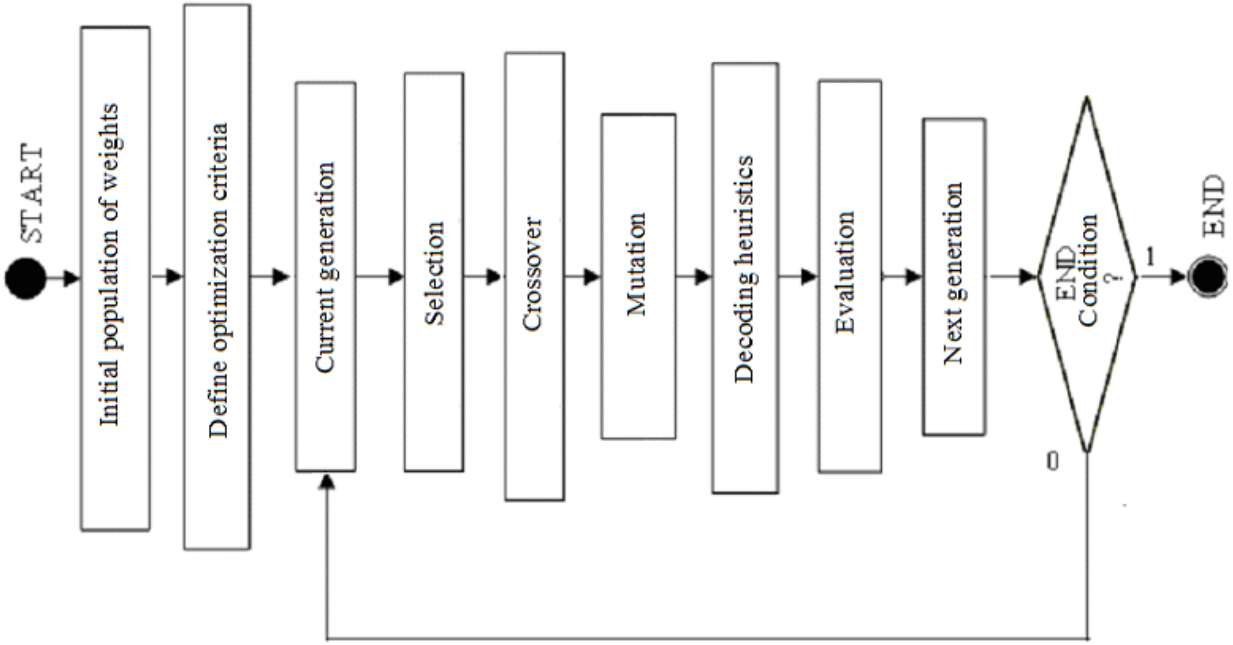
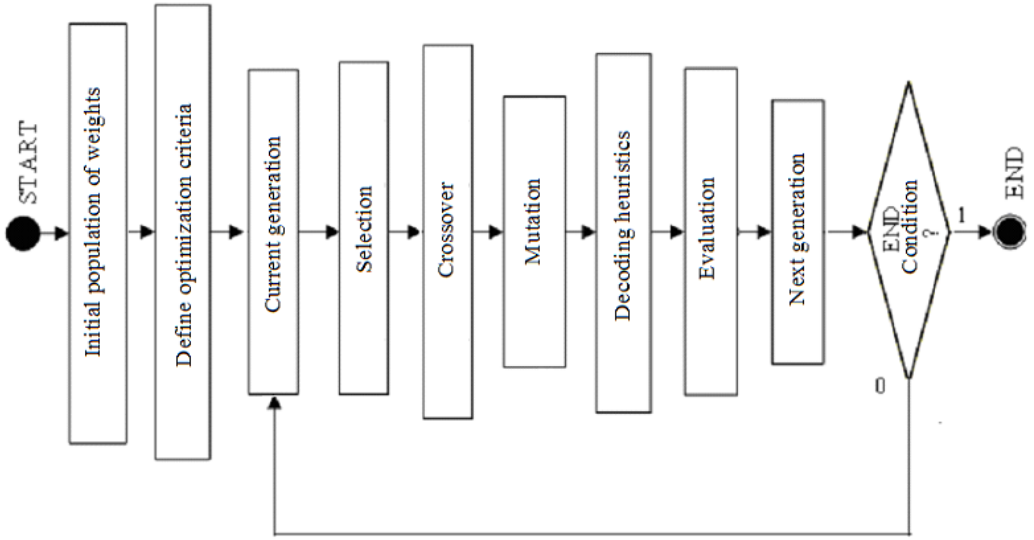
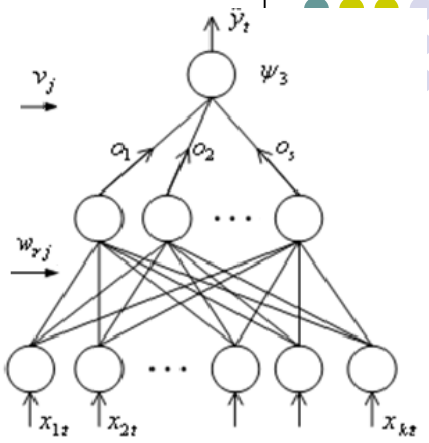


Fig. 4. Flowchart of the GA

The evolution usually starts from a population of randomly generated individuals and happens in generations. In each generation, the fitness of every individual is evaluated in the population. Multiple individuals are stochastically selected from the current population (based on the fitness), and modified (recombined and mutated) to form a new population. The new population is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced or a satisfactory fitness level has been reached for the population.

Genetic learning algorithm implementation





The first blocks of GA we define the initial population of neural network weights, optimization criteria, and fitness functions. The fitness function was proposed as the summary measure of model´s forecast accuracy *RMSE* as follows

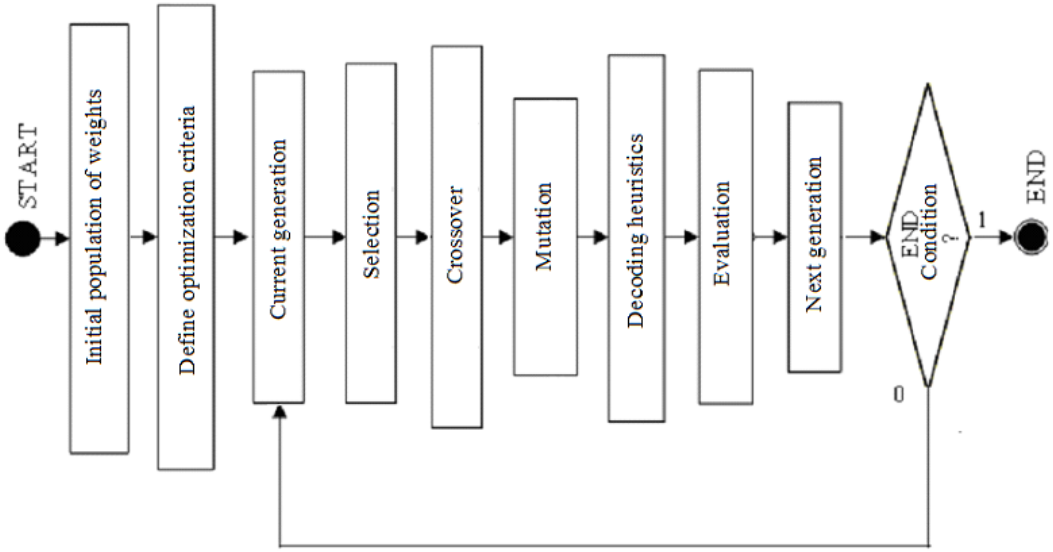
$$RMSE = \sqrt{\frac{\sum_{t=1}^{n} e_t^2}{n}}$$

Genetic algorithms traditionally work with genes either 0 or 1. The initial population of weights was generated randomly from the interval $(a, b) \equiv (-0.7 ; 0.7)$ and transformed into the integer digit denoted as l by the equation

$$l = \frac{v-a}{v-b}(2^k - 1)$$

where v is the value of the weight randomly chosen from the interval (a, b), k is the length of binary string in this case of size 16.

Genetic learning algorithm implementation



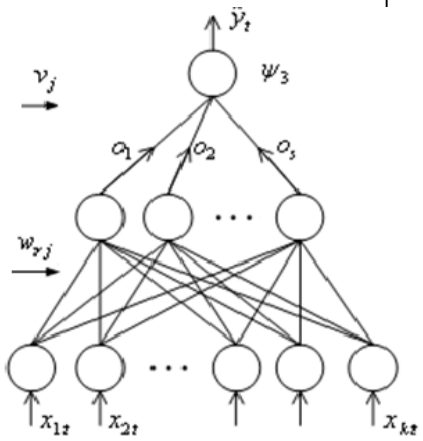


Fig. 4. Flowchart of the GA

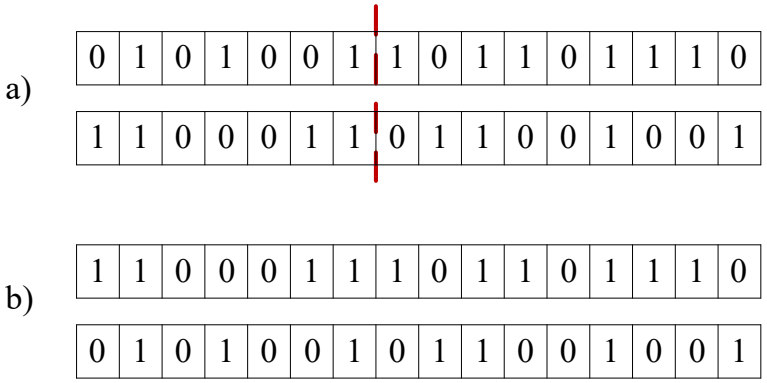
In this paper the roulette wheel selection was used. Mathematical foundation for the roulette wheel selection can be found in M. Obitko, *Genetic Algorithm*. 1998, available at: http://www.obitko.com/tutorials .

Very popular is also the **tournament** selection method, where *n* individuals are randomly selected from the population. From selected individuals, the one, who has the lowest fitness value is chosen and declared as the parent.

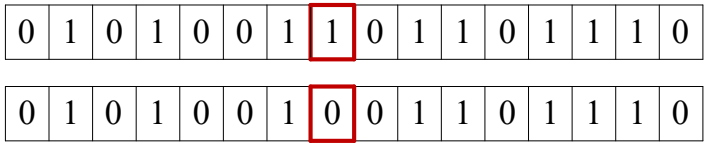
After the selection of the two chromosomes, the two basic operators of the genetic algorithm follow: the crossover and, possibly, the mutation.

Genetic learning algorithm implementation

The single-point crossover has been applied in this work. In the chromosome a randomly point was determined that divides chromosome into two parts. Then, those two parts of chromosomes were exchanged. In the first part in the next Figure there are two chromosomes selected for crossover. At the bottom of the Figure are their offsprings.



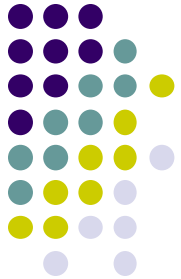
After the crossover is performed, **mutation m**ay take place. This is to prevent falling all solutions in population into a local optimum of solved problem. Mutation changes randomly the new offspring. For binary encoding, we can switch a few randomly chosen bits from 1 to 0 or from 0 to 1. Mutation can then be following:



The **New Population block** emerges from newly created offsprings to which one or more the best individuals are added from the previous generation. Thus, the next population can change their properties and also increases its diversity.

The binary string is transformed into decimal form y and then the calculation is inverse to the relation

Genetic learning algorithm implementation



The binary string is transformed into decimal value. The trasforation is inverse to the relation

$$l = \frac{w-a}{w-b}(2^k - 1)$$

Reproduction improves each generation by assessing chromosomes using a fitness feature. It is about finding the maximum / minimum of the fitness function, i.e. finding the best rated solution to the problem in the state space

For each new generation, the number of chromosomes is the same as before reproduction. Crossing and mutation give rise to new chromosomes and low-ranking chromosomes are replaced by higher-ranking chromosomes

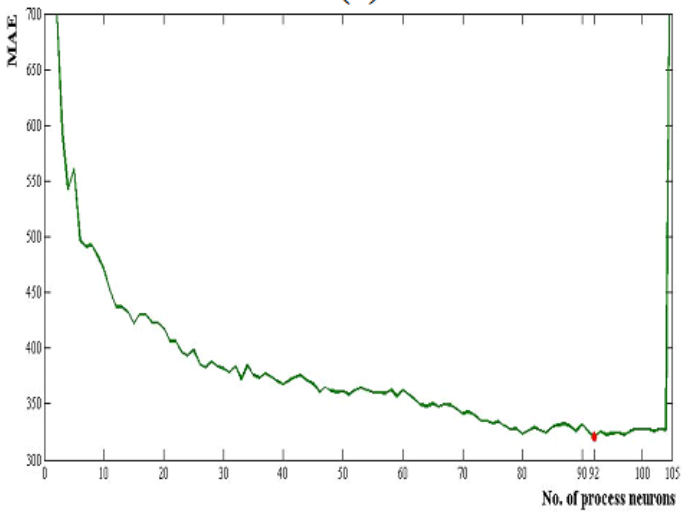
Parameter settings

The parameter settings we used and the details of network topology and learning parameters to estimate the weights for RBF NN classic (BP learning algorithm used) and RBF NN with GA learning algorithm are given in Table I.

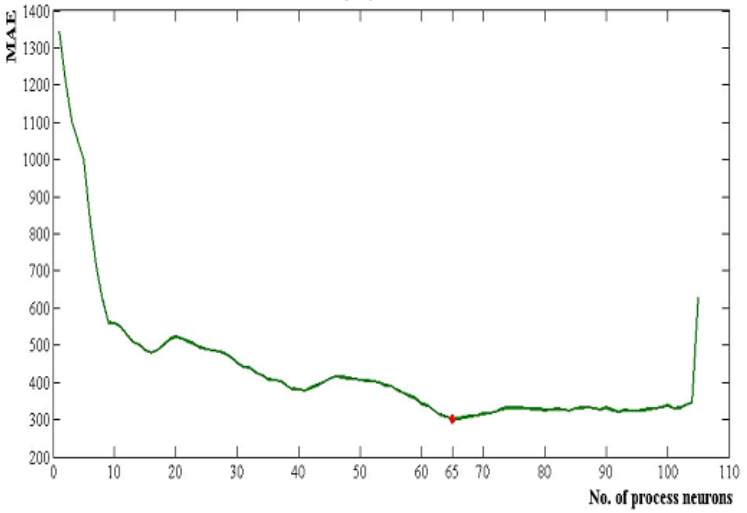
Table I.

Model:	RBF NN classic	RBF NN with GA learning
Training/Testing Data	2928/1488	2928/1488
Initial value of v_J	0.1	(-0.7 ; 0.7)
Learning coefficient	0.3	0.025
Number of RBF neurons	92	65
Number of epochs	673	1528

(a)



(b)



1. Convergence MAE (error) function depending on the number of processing neurons for classic RBF NN (a). Convergence MAE function depending on the number of processing neurons for RBF NN + GA approach (b)

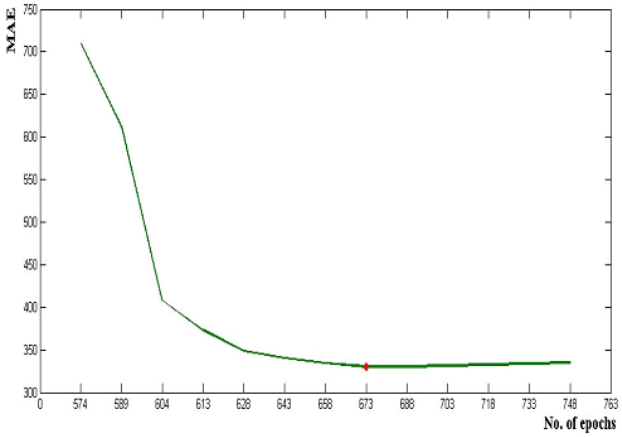
Parameter setting,

The parameter settings we used and the details of network topology and learning parameters to estimate the weights for RBF NN classic (BP learning algorithm used) and RBF NN with GA learning algorithm are given in Table I.

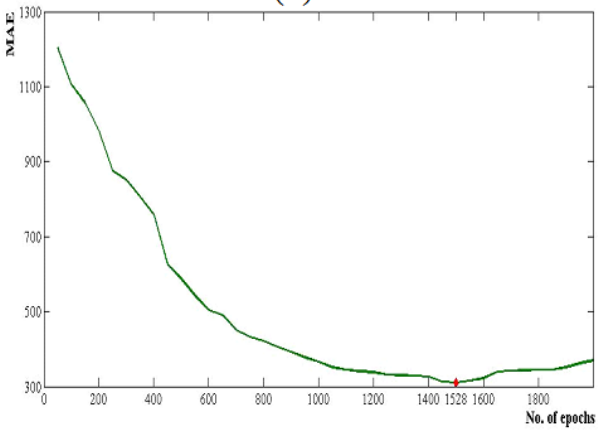
Table I.

Model:	RBF NN classic	RBF NN with GA learning
Training/Testing Data	2928/1488	2928/1488
Initial value of v_J	0.1	(-0.7 ; 0.7)
Learning coefficient	0.3	0.025
Number of RBF neurons	92	65
Number of epochs	673	1528

(a)

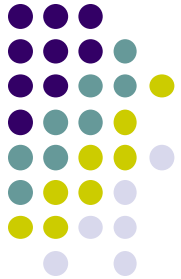


(b)



Convergence MAE (error) function depending on the number of trained epochs for classic RBF NN (a). Convergence MAE function depending on the number of trained epochs for RBF NN + GA approach (b)

Results, empirical comparison and conclusion



Next table shows the accuracy results of the ARIMA, RBF NN and SVR methods expressed in term of MAE and MAPE. MAE is a standard metric used by the research community and MAPE is the metric preferred by the industry forecasters.

Approach	MAE	MAPE [%]
Classic RBF NN (BP learning)	323.294	3.50
RBF NN (GA learning)	301.088	3.19
ARIMA$(6,1,1)(1,1,1)_{48}$	115.27	1.32
SVR	50.28	0.54

SVR is the most accurate method. (MAPE = 0.54), followed by ARIMA (MAPE = 1.32), RBF NN trained by GA (MAPE = 3.19%) and Classic RBF NN trained by BP (MAPE = 3.5%). All proposed forecast models based on advanced statistical and soft computing methods have MAPE measures much less than 5%, i.e. they indicate that all forecast models are very good.

The use of SVR and ARIMA models is a powerful approach to the solution of many forecasting problems. But, they are not without several limitations. In both AIRMA and SVR models, there is not conventional way to modify or update the estimates of the model parameters as each new observation becomes available. In contrast to NN, another drawback of ARIMA models is, that there is the learning speed very slow. The estimate of the parameters can be not parallelized.

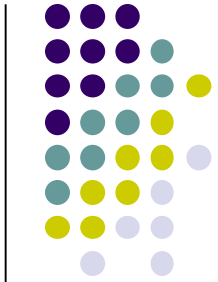
Conclusion

It was shown that:
SVR is the most accurate method. (MAPE = 0.54), followed by ARIMA (MAPE = 1.32), RBF NN trained by GA (MAPE = 3.19%) and Classic RBF NN trained by BP (MAPE = 3.5%).

But, all proposed forecast models based on advanced statistical and soft computing methods have MAPE measures much less than 5%, i.e. they indicate that all forecast models are very good.

ARIMA models are not without several limitations. In both AIRMA and SVR models, there is not conventional way to modify or update the estimates of the model parameters as each new observation becomes available.
In contrast to NN, another drawback of ARIMA models is that there is the learning speed very slow.
Although we cannot generally to say that statistical models generally outperform NN models, we can say that NN models have equivalent prediction performance comparing to statistical models.
The importance of having good intelligent forecasting tools for time series is ever more important with increasing number of data when more effort must be devoted to development of efficient data handling.

Future work will include exploring other ways of combining the prediction methods. In the future our main research objective is also to apply the developed metaheuristic on various datasets. Selected metaheuristics will be tested with different parameter combinations, and the combination of parameters which can yield approximate feasible solution in an acceptable computation time



Thank you!