Gabriela Andrejková; Miroslav Levický

Neural networks using Bayesian training

Kybernetika, Vol. 39 (2003), No. 5, [511]--520

Persistent URL: http://dml.cz/dmlcz/135552

KYBERNETIKA — VOLUME 39 (2003), NUMBER 5, PAGES 511 - 520

NEURAL NETWORKS USING BAYESIAN TRAINING

GABRIELA ANDREJKOVÁ AND MIROSLAV LEVICKÝ

Bayesian probability theory provides a framework for data modeling. In this framework it is possible to find models that are well-matched to the data, and to use these models to make nearly optimal predictions. In connection to neural networks and especially to neural network learning, the theory is interpreted as an inference of the most probable parameters for the model and the given training data. This article describes an application of Neural Networks using the Bayesian training to the problem of Predictions of Geomagnetic Storms.

Keywords: neural network, Bayesian probability theory, geomagnetic storm, prediction

AMS Subject Classification: 62M45, 62F15

1. INTRODUCTION

Neural networks continue to offer an attractive paradigm for the design and analysis of adaptive, intelligent systems for many applications in artificial intelligence [7], [8]. This is true for a number of reasons: for example, amenability to adaptation and learning, robustness in the presence of noise, potential for massively parallel computation.

Predictions of the hourly D_{st} index from the interplanetary magnetic field and solar plasma density, based on Artificial Neural Networks (ANN), were made and analysed by Lundstedt and Wintoft (feedforward networks) [10] and Andrejková et al (recurrent networks, fuzzy neural networks) [2], [3]. Recent results have shown that it is possible to use dynamic neural networks for predictions of GeoMagnetic Storms (GMS) and modeling of the solar wind-magnetosphere coupling. In this paper we are reporting preliminary results obtained with the help of a neural network model using Bayesian probability at its training.

There has been increased interest in using of Bayesian networks but our model is different. A Bayesian network is a compact, graphical model of a probability distribution [5]. Our model represents a combination of artificial neural networks with Bayesian probability. Anděl [1] and Bernardo and Smith [4] describe the probability theory and the Bayesian probability theory which have proved very successful in a variety of applications, for example MacKay [11], [12], Schlessinger and Hlaváč [16] and Müller and Insua [13]. The effectiveness of the models representing nonlinear input-output relationships depends on the representation of the input-output space. The method belongs to a large family of approximation techniques working

with some training samples. Families of approximation techniques are described in Dechter and Rish [6].

A designed Neuro-Bayesian model will predict the occurrence of geomagnetic storms on the basis of input parameters n, v, σ_{B_z} and B_z: n ... the plasma density of solar wind, v ... the bulk velocity of solar wind, B_z, σ_{B_z} ... z-component of the interplanetary magnetic field and its fluctuation.

To follow the changes of the geomagnetic field values we use the quantity D_{st} *index*. Its values are in the interval ± 10 nT (nano Tesla) during a normal situation but during the geomagnetic storm they can decrease as much as some hundreds nT in a few hours.

In Section 2, we describe some basic definitions and properties of the Bayesian probability theory. In Section 3, we briefly describe the neural networks as a probabilistic models. Section 4 contains the starting point to finding the weights of neural networks. Some interesting results for GMS prediction are presented in Section 5.

2. BAYESIAN PROBABILITY

A Bayesian data-modeller's aim is to develop probabilistic model that is well matched to the data and makes optimal predictions using that model. A very good description of this theory is presented in Bernardo at al [4]. Bayesian inference satisfies the likelihood principle: *Inferences depend only on the probabilities assigned to the data that were obtained, not on properties of the data which might have occurred.*

We shall use the following notation for *conditional* probabilities: $\Omega, \Omega \neq \emptyset$ – the space of elementary events; $\mathcal{H}$ – σ-algebra of some nonempty subsets of Ω (a model of computation), $A, B \in \mathcal{H}$ – events, $P(A), P(B)$ – probabilities of the events A, B, $(\Omega, \mathcal{H}, P)$ – a probability space, $p(\mathbf{x})$ is a density of the random vector $\mathbf{x}$ on a probability space $(\Omega, \mathcal{H}, P)$, $P(A|B, \mathcal{H})$ is pronounced "the probability of A, given B and $\mathcal{H}$" and it explains the conditional probability; B and $\mathcal{H}$ mean the conditional assumptions on which this measure of plausibility is based.

The Bayesian approach requires:

— specifying a set of prior distributions for all of weights in the network (and variance of the error) and

— computing the posterior distributions for the weights using Bayes' Theorem.

Prior distribution is a probability distribution on the unknown parameter vector $\omega \in \Omega$ in the probability model, typically described by its density function $p(\omega)$ which encapsulates the available information about the unknown value of ω. In our case the weight vector $\mathbf{w}$ has no known prior distribution; this is therefore replaced by a reference prior function.

Posterior distribution is a probability distribution on the unknown parameter vector $\omega \in \Omega$ in the probability model, typically described by its density function $p(\omega|D)$, conditionally on the model, encapsulates the available information about the unknown value of ω, given the observed data D and the knowledge about ω, which the prior distribution $p(\omega)$ might contain. It is obtained by Bayes' Theorem.

Bayes Theorem: Given data D generated by the probability model $\{p(D|A), A \in \Omega\}$ and a prior distribution $p(A)$, the posterior distribution of A is $p(A|D) \propto p(D|A)p(A)$. The proportionality constant is $\{\int_\Omega p(D|A)p(A)dA\}^{-1}$.

Two approaches have been explored in the finding of the posterior probability:

— To find the most probable parameters (weights) using methods similar to the conventional training and then approximate the distribution over weights using information available at this maximum.

— To use the Monte Carlo method to sample from the distribution over weights. This was the method we applied initiation to using Markov chains.

There are two rules of probability which can be used:

— *The product rule* relates to joint probability of A and $B, P(A, B|\mathcal{H})$ to the conditional probability:

$$P(A, B|\mathcal{H}) = P(B|\mathcal{H})P(A|B, \mathcal{H}) \tag{1}$$

— *The sum rule* relates the *marginal* probability distribution of A, $p(A|\mathcal{H})$, to the joint and conditional distributions:

$$p(A|\mathcal{H}) = \sum_B p(A, B|\mathcal{H}) = \sum_B p(A|B, \mathcal{H})p(B|\mathcal{H}) \tag{2}$$

Having specified the joint probability of all variables as in equation, we can use the rules of probability to evaluate the way in which our beliefs and predictions should change when we get new information.

3. NEURAL NETWORKS AS PROBABILISTIC MODELS

A supervised neural network is a non-linear parametrized mapping from an input $\mathbf{x}$ to an output $\widehat{\mathbf{y}} = \mathbf{f}(\mathbf{x}, \mathbf{w}; \mathcal{A})$. The output is a continuous function of the parameters $\mathbf{w}$, which are called weights and $\mathcal{A}$ is an architecture of the network.

The network is trained in the classical way using a data set $D = \{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), \ldots, (\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}$ by the backpropagation algorithm, n is the length of a training sample. It means the following sum squared error is minimized

$$E_D(\mathbf{w}) = \frac{1}{2} \sum_{m=1}^{n} \sum_i (y_i^{(m)} - f_i(\mathbf{x}^{(m)}; \mathbf{w}))^2 \tag{3}$$

The weight decay is often included in the objective function for minimization. This means that

$$M(\mathbf{w}) = \beta E_D(\mathbf{w}) + \alpha E_W(\mathbf{w}), \tag{4}$$

where $E_W(\mathbf{w}) = \frac{1}{2} \sum_i w_i^2$, α and β are regularization constants.

The learning process above can have the following probabilistic interpretation. We suppose $\Omega = \mathcal{R}^d$, where d is a weight vector dimension. The error function is interpreted as minus the log likelihood for a noise model:

$$p(D|\mathbf{w}, \beta, \mathcal{H}) = \frac{1}{Z_D(\beta)} \exp(-\beta E_D(\mathbf{w})) \tag{5}$$

where parameter β defines a noise level $\sigma_n^2 = \frac{1}{\beta}$ and $Z_D(\beta)$ is a suitably chosen constant.

Similarly, the regularization is denoted as a log prior probability distribution over parameters $\mathbf{w}$

$$p(\mathbf{w}|\alpha, \mathcal{H}) = \frac{1}{Z_W(\alpha)} \exp(-\alpha E_W(\mathbf{w})) \tag{6}$$

where $\sigma_W^2 = \frac{1}{\alpha}$, α is a regularization constant and $Z_W(\alpha)$ is a suitably chosen constant.

The function E corresponds to the deduction of parameters $\mathbf{w}$ according to data D. It means that

$$p(\mathbf{w}|D, \alpha, \beta, \mathcal{H}) = \frac{p(D|\mathbf{w}, \alpha, \beta, \mathcal{H}) p(\mathbf{w}|\alpha, \mathcal{H})}{p(D, \alpha, \beta, \mathcal{H})} \tag{7}$$

Bayesian inference for modelling problems may be implemented by analytical methods, by Monte Carlo sampling, or by deterministic methods using Gaussian approximations.

4. DESCRIPTION OF ALGORITHMS

We deal only with neural networks used for regression. Assuming a Gaussian noise model, the conditional distribution for the output vector given the input vector based on this mapping will be as follows:

$$p(\mathbf{y}|\mathbf{x}, \mathbf{w}) = (2\pi\sigma^2)^{d/2} \exp\left(-\frac{|\mathbf{y} - f(\mathbf{x}, \mathbf{w})|^2}{2\sigma^2}\right) \tag{8}$$

where d is the dimension of the output vector and σ is the level of the noise in the outputs.

In the Bayesian approach to the statistical prediction, one does not use a single "best" vector of weights, but rather integrates the prediction from all possible weight vectors over the posterior weight distribution which combines the data with prior computed weights.

The best prediction for the given input from the testing data $\mathbf{x}_{n+1}$ can be expressed by

$$\widehat{\mathbf{y}}_{n+1} = \int_{R^d} f(\mathbf{x}_{n+1}, \mathbf{w}) p(\mathbf{w}|(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)) d\mathbf{w}. \tag{9}$$

The densities of the posterior probabilities of weight vectors are as follows (according to the Bayes theorem):

$$p(\mathbf{w}|(\mathbf{x}_1,\mathbf{y}_1),\ldots,(\mathbf{x}_n,\mathbf{y}_n)) = \frac{p(\mathbf{w})p((\mathbf{x}_1,\mathbf{y}_1),\ldots,(\mathbf{x}_n,\mathbf{y}_n)|\mathbf{w})}{p((\mathbf{x}_1,\mathbf{y}_1),\ldots,(\mathbf{x}_n,\mathbf{y}_n))} = \frac{p(\mathbf{w})p(\mathbf{y}_1,\ldots,\mathbf{y}_n|\mathbf{x}_1,\ldots,\mathbf{x}_n,\mathbf{w})}{p(\mathbf{y}_1,\ldots,\mathbf{y}_n|\mathbf{x}_1,\ldots,\mathbf{x}_n)} \tag{10}$$

The training data are independent of each other which means that the following relationship is satisfied:

$$\frac{p(\mathbf{w})p(\mathbf{y}_1,\ldots,\mathbf{y}_n|\mathbf{x}_1,\ldots,\mathbf{x}_n,\mathbf{w})}{p(\mathbf{y}_1,\ldots,\mathbf{y}_n|\mathbf{x}_1,\ldots,\mathbf{x}_n)} = \frac{p(\mathbf{w})\prod_{i=1}^{n} p(\mathbf{y}_i|\mathbf{x}_i,\mathbf{w})}{p(\mathbf{y}_1,\ldots,\mathbf{y}_n|\mathbf{x}_1,\ldots,\mathbf{x}_n)} \tag{11}$$

To the formulation of Bayesian problem it is necessary to add the prior distribution of weights. One of the possibilities is using the Gaussian noise function: $p(\mathbf{w}) = (2\pi\omega^2)^{-\frac{d}{2}}\exp\left(-\frac{|\mathbf{w}|^2}{2\omega^2}\right)$, where ω is an expected weight scale, it should be set by hand.

High-dimensional integrals mentioned above for predictions are in general analytically unsolvable and numerically difficult to compute. This leads to the problem of Bayesian learning. While in traditional training we deal with an optimization problem, in Bayesian training we deal with evaluation of high-dimension integrals. Metropolis algorithm presents a method for evaluation of the integrals. However, this algorithm works slowly for our problem. But it forms the basis for Hybrid Monte Carlo method, which should be more effective for evaluating the integrals. Now we will describe used methods in some basic steps. The developed algorithm for our problem was applied according to the construction of Neal [14].

Suppose that we wish to evaluate

$$\langle g\rangle = \int_{R^d} g(\mathbf{q})p(\mathbf{q})\mathrm{d}\mathbf{q}. \tag{12}$$

The Metropolis algorithm generates a sequence of vectors $\mathbf{q}_0, \mathbf{q}_1, \ldots$, which forms a Markov chain with the stationary distribution $p(\mathbf{q})$. The integral in equation (12) is then approximated as

$$\langle g\rangle \approx \frac{1}{M}\sum_{t=I}^{I+M-1} g(\mathbf{q}_t), \tag{13}$$

where I stands for the number of initial values which will not be used in evaluation and M stands for the number of functional values g. Averaging those values one gets an approximate value for $\langle g\rangle$. In limit case, as M increases, approximation converges to the real value $\langle g\rangle$. It is difficult to determine how long it takes to reach the stationary distribution (and hence to determine the value of I), or determine how the values of $\mathbf{q}_t$ are correlated in following iterations (and hence how large M should be). One cannot avoid such difficulties in applications that utilize the Metropolis algorithm.

Generation of the Markov chain is described by an energy function, defined as

$$E(\mathbf{q}) = -\ln(p(\mathbf{q})) - \ln(Z_E), \tag{14}$$

where Z_E is a positive constant chosen for convenience. The algorithm starts by random sampling of $\mathbf{q}_0$. In every t-th iteration, a random candidate $\widetilde{\mathbf{q}}_{t+1}$ for the next state is sampled from distribution $p(\widetilde{\mathbf{q}}_{t+1} \,|\, \mathbf{q}_t)$. The candidate is accepted if its energy is lower than that of the previous state; if its energy is higher it is accepted with probability $\exp(-\Delta E)$, where $\Delta E = E(\widetilde{\mathbf{q}}_{t+1}) - E(\mathbf{q}_t)$. In other words,

$$\mathbf{q}_{t+1} = \begin{cases} \widetilde{\mathbf{q}}_{t+1} & \text{if } U < \exp(-\Delta E) \\ \mathbf{q}_t & \text{otherwise} \end{cases} \tag{15}$$

and U is a random number from uniform distribution from interval $\langle 0, 1)$.

The Hybrid Monte Carlo method as an improvement on the Metropolis algorithm eliminates much of the random walk in weight space, and further accelerates exploration of the weight space. The method uses a gradient information provided by the backgropagation algorithm.

Unlike the Metropolis algorithm, the Hybrid Monte Carlo method generates sequence of vector couples $(\mathbf{q}_0, \mathbf{r}_0), (\mathbf{q}_1, \mathbf{r}_1), \ldots$, where vectors $\mathbf{q}$ are called position vectors and vectors $\mathbf{r}$ are called momentum vectors. Both these vectors are of the same dimension. Potential energy function $E(\mathbf{q})$ used in Metropolis algorithm is extended to Hamiltonian function $H(\mathbf{q}, \mathbf{r})$ that combines potencial and kinetic energy:

$$H(\mathbf{q}, \mathbf{r}) = E(\mathbf{q}) + \frac{1}{2} |\mathbf{r}|^2 . \tag{16}$$

$p(\mathbf{q}, \mathbf{r}) = p(\mathbf{q})p(\mathbf{r})$ is fact for the stationary distribution of generated Markov chain. Marginal distribution of $\mathbf{q}$ is the same as the one for Metropolis algorithm. Thus, the value of $\langle g \rangle$ can be again approximated by use of equation (13). Momentum characteristics have Gaussian distributions, and they are independent of $\mathbf{q}$ and of each other. The Markov chain is generated by two types of transitions – dynamic and stochastic moves. The hybrid Monte Carlo method is described in [15].

5. RESULTS OF GEOMAGNETIC STORM (GMS) PREDICTIONS

We will discuss various implementation issues which are necessary for the actual prediction. The data are available from the NASA "OMNI database" and are distributed by National Space Science Data Center [17] and WDC-A for Rockets&Satellites. In the period 1963–1999, the quantities: B_z, σ_{B_z}, n, v and D_{st} are measured and saved at each hour. We will predict the values of D_{st} index with neural networks in depending on the next four quantities:

B_z – z-coordinate of the interplanetary magnetic field, the values are in the interval ± 50 nT,

σ_{B_z} – mean square error B_z characterizes the swings of the quantity,

n – the plasma density of solar wind in 1 cm^3,

v – the bulk velocity of solar wind, the values can be as much as 1200 km/s.

We suppose that the measured D_{st} index of geomagnetic storm is a function of parameters B_z, σ_{B_z}, n, v and in Figure 1 we can see the course of it.

Some data are not complete and we use linear interpolation to fill the gaps but only in the case if the gap is less then 30 hours. The reconstructed data are used for a choice of the samples to the training set according to the following criteria: if the value D_{st} decreases at least 40 nT during two hours then the training sample (the storm) is created from the measured values 36 hours before the decreasing, 2 hours of the identification of decreasing and 106 hours after the decreasing. The file of the values must satisfy requirements of completeness of measurements. This means that 144 hours describe one event – GMS. One storm is used for the learning of the neural network by moving the 8 hours window.

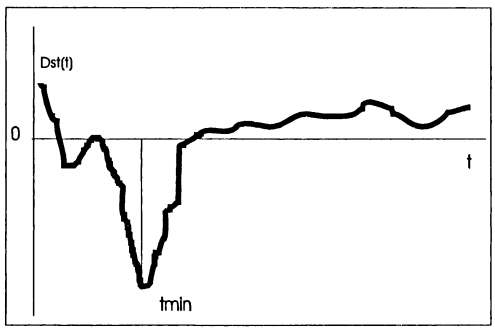

Fig. 1. The D_{st} index of geomagnetic storm measured in 1981, 62-nd day.

We have prepared the training data set and two data testing sets A and B. To prepare the A and B sets we used the data from years 1980–1984 and 1989–1999 because we had the continued values of parameters n, v, B_z, σ_{B_z} and D_{st}. The prepared data were represented by a sequence of

$$\mathbf{s}^t = (n^t, v^t, B_z^t, \sigma_{B_z}^t, D_{st}^t),$$

where $\mathbf{s}^t$ can be applied as time series.

The software of Levický described in [9] was modified and used in the present application. The algorithm based on the works of Neal and McKay was written in Delphi 5.

The feed-forward neural network calculating the following function $\mathbf{y} = f(\mathbf{x}, \mathbf{w})$ (the function f is corresponding to the function f in (9)) was used in the tests:

$$\begin{aligned} &\text{hidden layer:} \quad a_j^{(1)} = \textstyle\sum_{l=1}^{m} w_{jl}^{(1)} x_l + \theta_j^{(1)}; \quad h_j = \tanh(a_j^{(1)}), \quad j = 1, \ldots, p \\ &\text{output layer:} \quad a_1^{(2)} = \textstyle\sum_{j=1}^{p} w_{1j}^{(2)} h_j + \theta_1^{(2)}; \quad y_1 = a_1^{(2)}. \end{aligned}$$

The vectors $\mathbf{x} = (x_1, \ldots, x_m)$ (where m is the dimension of the data representing a geomagnetic storm) represent neural network inputs. Every vector $\mathbf{w}$ stands for

Table 1. Experimental Results – approximations of GMS.

Data	# Iteration	#Good Approximations	#Bad Approximations	Average Error	Success
A	6000	62	74	1.90585	45,59 %
B	6000	101	35	0.48040	74,26 %
A	12000	76	60	1.19665	55,88 %
B	12000	113	23	0.23863	83,09 %
A	18000	86	50	0.77771	63,24 %
B	18000	109	27	0.23801	80,15 %

coordinates of one GMS. Values y_1, represent outputs of the neural network. It means the results are values of some function.

The energy function explained by (14) in our problem is the function $M(\mathbf{w})$ described by (4). The sequence of weights is generated by the energy function as the Markov chain. The sequence of weights is used to the approximation of values D_{st}. It means that the function f in (9) is approximated by (13).

The computed results are in the following Table 1 and Table 2. The model was at the time in the initial testing stage. In Table 1 we present results computed with two data sets A and B. The measured and predicted data follow the D_{st} index in the interval $0-143$ hours, if both values are closed to each other then the prediction is good in the opposite case it is bad. The prediction performance is measured by *#Good Predictions, #Bad Predictions, Average Error* and *% of Success.* In Table 2 are presented results concerning of the predicted GMS numbers.

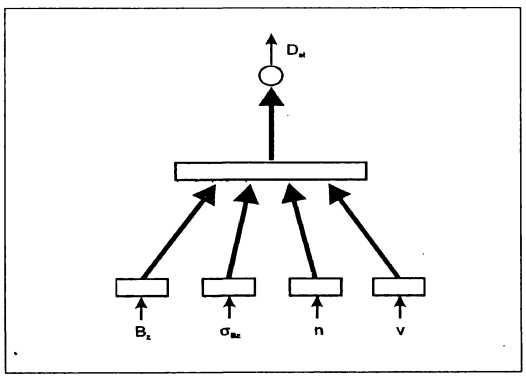

Fig. 2. The model of the neural network.

Table 2. Experimental Results – predictions of GMS.

# Iteration	*Average Error*	*# Predicted GMS*
20000	0.91344	21
60000	0.72604	21
100000	0.64452	19
120000	0.61764	18

Total test samples in testing sets A and B: 272, the number of input neurons in the neural network: 32 (for each block – B_z, σ_{B_z}, n, v, 8 input values), the number of hidden neurons in the neural network: 28, the number of output neurons: 1, as it is shown in Figure 2.

The computed results are interesting from the following points of view:

— With the increase in the number of iteration the average error decreases. It is one of the criteria for the evaluation of the model.

– After 18000 iterations the success grows very slowly in the case of the testing data set A and decreases in the testing data set B (results in Table 1).

— The numbers of predicted Geomagnetic Storms (results in Table 2) are higher than the numbers of real GMS but after 120000 iterations the numbers of predicted GMS decrease.

— Bayesian neural networks that we used in the prediction of geomagnetic storms seems a very good model. They move the weight vector to the most probable part of the weight space.

— The using method has very slow convergence. It means that the trainig is very time-consuming.

6. CONCLUSION

In the present paper we dealt with feed-forward neural networks and their training for the prediction of Geomagnetic Storms. We use Bayesian probability theory and Monte Carlo methods. Monte Carlo methods provide good approximations of evaluating high-dimensional integrals which are needed to be computed at the training of neural networks. Their greatest disadvantage is the fact that they are extremely time-consuming because they need the high numbers of iteration steps.

ACKNOWLEDGEMENT

This research was supported by the Slovak Grant Agency through Grants 1/7557/20, 1/0385/03.

(Received February 3, 2003.)

REFERENCES

[1] J. Anděl: Mathematical Random Events (in Czech). Matfyzpress, Praha 2002.
[2] G. Andrejková, J. Azorová, and K. Kudela: Artificial neural networks in prediction D_{st} index. Proc. 1st Slovak Neural Network Symposium, ELFA, Košice, 1996, pp. 51–59.
[3] G. Andrejková, H. Tóth, and K. Kudela: Fuzzy neural networks in the prediction of geomagnetic storms. Proc. "Artificial Intelligence in Solar-Terrestrial Physics", Publisher European Space Agency, Lund 1997, pp. 173–179.
[4] J. M. Bernardo and A. F. M. Smith: Bayesian Theory. Wiley, New York 2002.
[5] A. Darwiche: A differential approach to inference in Bayesian networks. J. Assoc. Comput. Mach. *50* (2003), 2, 280–305.
[6] R. Dechter and I. Rish: Mini-buckets: A general scheme for bounded inference. J. Assoc. Comput. Mach. *50* (2003), 3, 107–153.
[7] M. H. Hassoun: Fundamentals of Artificial Neural Networks. MIT Press, Cambridge, MA 1995.
[8] J. Hertz, A. Krogh, and R. G. Palmer: Introduction to the Theory of Neural Computation (Santa Fe Institute Studies in the Science of Complexity, Vol. 1). Addison-Wesley, Reading 1991.
[9] M. Levický: Neural Networks in the Analysis and the Document Classification. Diploma Thesis, P. J. Šafárik University, Košice, 2002.
[10] H. Lundstedt and P. Wintoft: Prediction of geomagnetic storms from solar wind data with the use of a neural network. Ann. Geophysicae 12, EGS-Springer-Verlag, 1994, pp. 19–24.
[11] D. J. C. MacKay: Bayesian Methods for Neural Networks: Theory and Applications. Neural Network Summer School, 1995.
[12] D. J. C. MacKay: A practical Bayesian framework for backprop networks. Neural Computation 4, pp. 448–472.
[13] P. Müller and D. R. Insua: Issues in Bayesian analysis of neural network model. Neural Computation 10, pp. 749–770.
[14] R. M. Neal: Probabilistic Inference Using Markov Chain Monte Carlo Methods. Technical Report CRG-TR-93-1, University of Toronto, 1993.
[15] R. M. Neal: Bayesian Training of Backpropagation Networks by the Hybrid Monte Carlo Method. Technical Report CRG-TR-92-1, University of Toronto, 1992.
[16] M. I. Schlesinger and V. Hlaváč: Deset přednášek z teorie statistického a strukturního rozpoznávaní. ČVUT, Praha 1999.
[17] http://nssdc.gsfc.nasa.gov/omniweb/html/ow_data.html

Doc. RNDr. Gabriela Andrejková, CSc., Institute of Computer Science, Faculty of Science, P. J. Šafárik University, 041 54 Košice. Slovak Republic.
e-mail: andrejk@kosice.upjs.sk

Mgr. Miroslav Levický, Institute of Computer Science, Faculty of Science, P. J. Šafárik University, 041 54 Košice. Slovak Republic.
e-mail: levicky@science.upjs.sk