

ARTIFICIAL NEURAL NETWORKS ASSISTED PREDICTION OF PROSTATE CANCER CONFINEMENT AFTER RADICAL PROSTATECTOMY

Razvan Bardan¹, Corina Botoca², Mircea Botoca¹, Viorel Bucuras¹, Georgeta Budura²

REZUMAT

Obiectiv: De a prezice penetrarea capsulei prostatice după prostatectomia radicală, utilizând rețelele neuronale artificiale. **Material și metode:** Variabilele de intrare ale rețelilor neuronale au fost reprezentate de datele clinice selecționate și prelucrate dintr-o bază de date conținând 650 de pacienți cu cancer de prostată, tratați prin prostatectomie radicală. Diferite arhitecturi de rețele neuronale artificiale și diferiți algoritmi de predicție au fost testate. Pentru evaluarea performanței predicției am utilizat ca și comparator regresia logistică, una dintre cele mai bune metode statistice de predicție.

Rezultate și concluzii: În toate experimentele rețelele neuronale artificiale au obținut rezultate superioare regresiei logistice. Modelul cu cea mai mare acuratețe a fost o rețea neuronală de tip recurent. Performanțele obținute (eroare medie pătratică = 0,0428, acuratețe a predicției globale = 97,37%) au fost mai bune decât cele ale experimentelor publicate anterior. Limitările performanței predictive a rețelilor neuronale artificiale sunt date în general de dimensiunile reduse ale bazelor de date și de modul defectuos de colectare a datelor, elemente care în mod cert pot fi ameliorate.

Cuvinte cheie: rețele neuronale artificiale, predicție, cancer de prostată

ABSTRACT

Aim of the study: To predict prostate cancer capsule penetration after radical prostatectomy by using artificial neural networks (ANN). **Material and methods:** The neural networks input variables were clinical data selected and preprocessed from a database record of 650 patients with prostate cancer, treated by radical prostatectomy. Different ANN architectures and algorithms have been tested. For the performance assessment we have used logistic regression, one of the best prediction statistical tools. **Results and conclusions:** In all the experiments the ANN models performed better than the logistic regression. The model with the best prediction accuracy was a recurrent ANN. The obtained performances (mean square error = 0.0403, global prediction accuracy = 97.37%) were better than the results of similar experiments available previously published. The limitations of the predictive performance of ANN are generally caused by the reduced database sample size and by the biased data collection methods, elements that could be further improved.

Key Words: artificial neural networks, prediction, prostate cancer

INTRODUCTION

The field of oncology needs new, better prognosis methods in all the major forms of cancer, assisting the physicians in the complex process of diagnosis and, thereafter, in the therapeutic decisions, making correct disease classifications and risk stratifications.¹

Although the TNM staging is widely used, making the anatomical description of lesion extent, it has some limitations, because is not including new tumor markers, or other recently described pathological elements, which are necessary for a specific diagnosis, leading finally to the most appropriate therapy. This is the main reason for the necessity of new predictive tools, which should include a larger number of clinical parameters, increasing the accuracy of the prognosis.²

The artificial neural networks (ANN) represent a viable option, because they allow the assessment of non-linear relationships between any of the clinical/biological parameters, with faster and better results, when compared with traditional statistical methods.³ The recent developments, of improved computational power, and the advances in artificial neural networks software encouraged us to investigate new potential applications, as the finding of correlations between different predicting factors, prognosis prediction for different groups of patients, or new risk group stratifications.

¹ Department of Urology, Victor Babes University of Medicine and Pharmacy, ² Department of Communications, Polytechnic University, Timisoara

Correspondence to:
Razvan Bardan, Dept. of Urology, Timisoara Clinical Emergency Hospital,
10 I. Bulbuca Blvd., 300376 Timisoara, Tel. +40-748-331235
Email: razvan.bardan@umft.ro

Received for publication: Oct. 19, 2008. Revised: Mar. 14, 2009.

THEORETICAL BACKGROUND

Neural networks are nonlinear systems consisted of a large number of relative simple processing elements, named neurons, that operate simultaneously, in parallel. The neurons interact through excitatory and inhibitory connections, which have associated weights, in a similar manner to the biological neurons. Learning is performed by weights changes conform to a learning rule.

During the years various types of ANN architectures have been developed, which from the point of view of the information flow can be classified as feed forward ANN or recurrent ANN.

1. Multilayer perceptron

Multilayer perceptron (MLP) is the most usual type of feed forward ANN, which propagates the information from the input towards the output layer. MLP has an input layer of neurons, a number of hidden layers and an output layer. It is an universal approximator. This means that MLP can map any nonlinear input–output function to an arbitrary degree of accuracy, provided that a sufficient number of hidden neurons are used. The architecture of a MLP with a single hidden layer is represented in Figure 1.

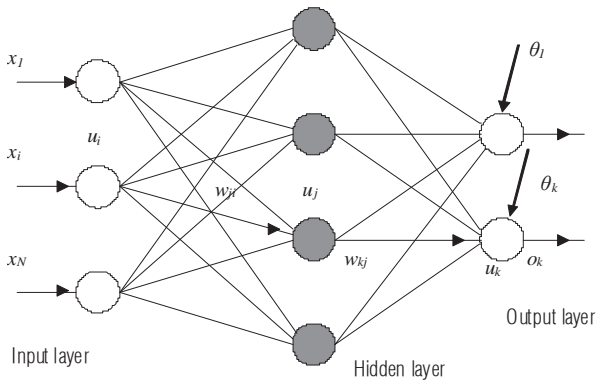


Figure 1. The architecture of a multilayer neural network.

Typically, all the neurons from a layer have the same continuous activation function. The continuous activation functions allow the use of gradient based methods to train the NN parameters. Usually a sigmoid function is used, for example an exponential function, given by the relation:

$$o_{pj} = \frac{1}{1 + e^{-\beta(\sum_i w_{ji}x_{pi}(t) + \theta_j)}} \quad (1)$$

where:

- x_{pi} is the input to the i neuron for the p input model;

- o_{pj} is the output of the neuron j for the p input model;

- θ_j is the bias of the j neuron;

- w_{ji} is the weight of the interconnection between the i input and j neuron;

- β is a proportionality parameter, chosen in the (0,1) interval.

The advantage of the previous function is a very simple derivate, given by:

$$f'(x) = f(x)[1 - f(x)] \quad (2)$$

Sometimes also the hyperbolic tangent is used, because it has values in the interval [-1, 1]:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (3)$$

Its derivate is also very simple, given by:

$$\frac{d \tanh(x)}{dx} = [\text{sec h}(x)]^2 = \frac{4}{(e^x + e^{-x})^2} \quad (4)$$

To train the NN parameters, respectively the weights and biases, usually the back-propagation error algorithm (BKP), a supervised training algorithm is used. It requires pairs of input model- output desired model (x_p, d_p) and it has two stages. During the first stage the information propagates through layers, from input towards output. In the second stage, the errors between the desired output and the current output propagate from output towards input, determining changes in the NN parameters. The BKP algorithm minimizes the overall mean square error.³

First stage

Let's use the following notations:

- N - number of inputs in the NN (which is the dimension of the input models);

- N_h - number of neurons in the hidden layer;

- N_{out} - number of neurons in the output layer;

As in Figure 1, the inputs are indexed by i , the hidden neurons by j and the output neurons by k . The net input for the j hidden neuron, net_{pj} , is given for the p input model by:

$$net_{pj} = \sum_{i=1}^N w_{ji}x_{pi} + \theta_j \quad i=1,2, \dots, N \quad (5)$$

The hidden neuron output o_{pj} is calculated as a function of the net input:

$$o_{pj} = f\left(\sum_{i=1}^N w_{ji}x_{pi} + \theta_j\right) \quad j=1, \dots, N_h \quad (6)$$

The output o_{pk} of the k neuron is expressed in terms of the information propagated from the

hidden layer:

$$o_{pk} = f(\text{net}_{pk}) \quad \text{where} \quad \text{net}_{pk} = \sum_{j=1}^{N_h} w_{kj} o_{pj} + \theta_k \quad k=1, \dots, N_{out} \quad (7)$$

and $w_k = [w_{k1} \ w_{k2} \ \dots \ w_{kj}]$ is the weight vector of the k output neuron. The global k neuron output function is:

$$o_{pk} = f\left(\sum_{j=1}^{N_h} w_{kj} f\left(\sum_{i=1}^N w_{ji} x_{pi} + \theta_j\right) + \theta_k\right) \quad k=1, \dots, N_{out} \quad (8)$$

The current output o_{pk} is compared with the desired output d_{pk} , generating in the output neurons an error δ_{pk} :

$$\delta_{pk} = (d_{pk} - o_{pk}) f'(\text{net}_{pk}) \quad (9)$$

where f' is the derivate of the activation function.

Second stage

During the second stage the errors are propagated from output towards input from one layer to the previous one, generating changes in the weights connections, in order to minimize the overall mean square error. The learning rule for the connections between the output and hidden neurons is:

$$\Delta_p w_{kj} = \eta \delta_{pk} o_{pj} \quad (10)$$

where $\Delta_p w_{kj}$ represents the variation of the weight interconnection w_{kj} between the hidden neuron j and the k output neuron and η is the learning rate, with a value in the interval (0,1).

The hidden neurons error δ_{pj} are determined using the errors δ_{pk} of the k output neurons with which the neuron j is connected, with the relation:

$$\delta_{pj} = \left(\sum_k \delta_{pk} w_{kj} \right) f'(\text{net}_{pj}) \quad (11)$$

The weights interconnections between input and hidden neurons are modified with the relation:

$$\Delta_p w_{ji} = \eta \delta_{pj} o_{pi} \quad (12)$$

If there are more hidden layers, the errors are evaluated for each layer with the relation (11) and then the interconnections with the precedent layer are determined. If there are outputs in the hidden layer, these sum two type of errors, errors obtained by difference between the current output and the desired output, calculated with relation (9) and errors determined by the propagation of output neurons errors, determined with relation (11). Sometimes, the algorithm gets stucked in a local minimum of the error and not in the global one, which is one of the BKP deficiencies. If the performance of the NN in this

local minimum is acceptable, it can be used in practice. Otherwise, the literature offers different methods to avoid the local minimums.

There are different stages in developing a MLP application: a training stage, a testing one and the practical operation.

Gathering the data base and splitting it in the training and testing sets have a major influence on the success of the ANN application. During training each pair input model –desired model (x_p, d_p) are repeatedly applied to the NN, in order to minimize the error. This conducts to a relatively long training time, which is another deficiency of the BKP algorithm. During testing, new input models are applied to the ANN and its performance is verified.

Radial basis function neural networks

Recently, the radial basis function (RBF) neural network received considerable attention, since the MLP network is plagued by long training times and may be trapped in bad local minima. The RBF network is able to approximate, as the MLP does, any arbitrary nonlinear function in the complex multi-dimensional space but with a reduced computing complexity comparative with other ANN. In Figure 2 we present the RBF structure, which contains an input layer, one single hidden layer and an output layer.

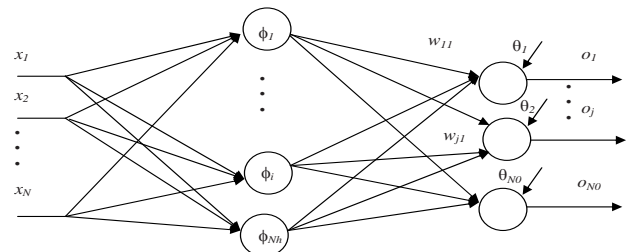


Figure 2. The structure of a radial basis function neural network.

The RBF ANN output is determined with the relation:

$$o_j = \sum_{i=1}^{N_h} w_{ji} \phi_i(\| \mathbf{x} - \mathbf{c}_i \|) + \theta_j, \quad j = \overline{1, N_o} \quad (13)$$

where

- o_j is the neuron j output;
- w_{ji} is the weight interconnection from the j neuron to the i neuron;
- θ_j is the bias of j neuron;
- $\phi_i(\bullet)$ is the radial function of the i hidden neuron;
- \mathbf{c}_i is the centre vector associated to the i neuron;
- $\| \bullet \|$ is a metric distance, between its arguments;
- N_h is the number of the hidden neurons;
- N_o is the number of the output neurons.

The RBF NN parameters are the centers vectors $\{c_i\}$ and the weights vectors $\{w_j\}$. Each of the neurons of the hidden layer calculates a metric distance between the input model x and its centre vector c_i . This distance can be of different types, but usually the Euclidian distance is used.

Given the x input model, $x = [x_1 \ x_2 \ \dots \ x_N]^T$, belonging to the RN space, the Euclidean distance is defined by the relation:

$$\|x - c_i\| = \sqrt{(x_1 - c_{i1})^2 + (x_2 - c_{i2})^2 \dots + (x_N - c_{iN})^2} \quad (14)$$

The closer the input x is to the vector centre c_i , the smaller the distance will be. In case that x is identical with c_i the Euclidian distance is zero. The result is passed through a real, nonlinear, continuous activation function, named radial function. This function also gives the name to the ANN.

Several radial functions can be used, for example: the Gauss function, the multiquadratique one, the inverse multiquadratique, the Cauchy function. Usually the Gauss function, represented in Figure 3, is used:⁴

$$\phi(x) = e^{-\frac{x^2}{\rho^2}} \quad (15)$$

where ρ is a parameter named radius, width or spread. The radius may be chosen proportionally with the input dispersion.

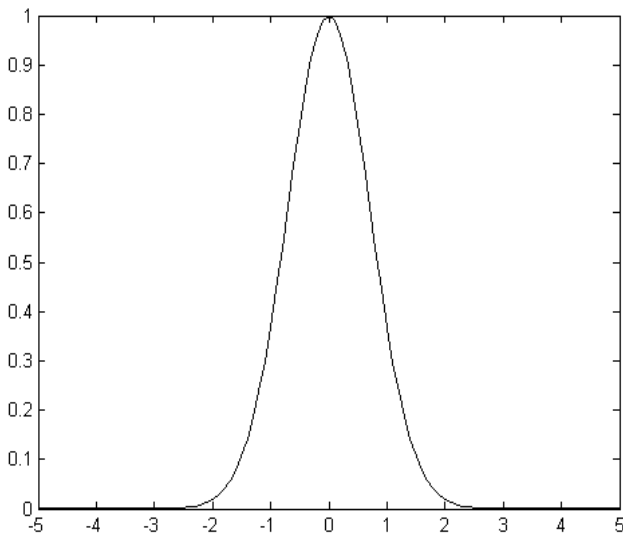


Figure 3. The Gauss function with centre $c = 0$ and radius $\rho = 1$.

For the inputs close to the centre, the radial basis function output is greater, approaching to 1 as the Euclidian distance is decreasing to 0 and it is converging to zero as the distance between the input and the centre is increasing. Thus the RBF-ANN is capable of a local modeling of the inputs.

Theoretical and practical studies prove that the

type of radial function does not essentially influence the performances RBF-ANN.

Training a RBF-ANN means determining the number of basis functions (hidden neurons)(1), the centers, the radius the radial functions (2) and the output interconnections weights (3). For some algorithms these steps are separately treated, while in others the parameters are found simultaneously. An unsupervised learning algorithm is usually used to find the number and positions of the hidden neurons and a supervised learning algorithm for the weights of the output layer.³

The design and training of RBF-ANN essentially depend on the centers, thus a lot of studies deals with this problem. The following strategies have imposed into practice: randomly choosing a subset of fixed centers from the database, the standard competitive algorithm (the k mean), the competitive algorithm with conscience, the rival penalization competitive algorithm, the dynamic rival penalization algorithm, a subset selection from the database using the orthogonal least squares method and the supervised centers selection.³⁻⁹

The basic approaches to determine the output layer weights can be classified into off line and on line methods. Off line methods include the usual least squares method, while the others include the least mean square algorithm (LMS) or recursive least square (RLS). The simplest way is to use the LMS algorithm, which updates the weight interconnections with the relation:³

$$\Delta_p w_{ji} = \eta e_{pi} \phi_i \quad (16)$$

where e_p is the difference between the desired d_p and current output o_p for the p input model:

$$e_p = d_p - o_p \quad (17)$$

This algorithm minimizes the mean square error (MSE):

$$MSE = \frac{1}{P} \sum_{i=1}^P e_p^2 \quad (18)$$

where P is the number of input models.

Recurrent neural network

The recurrent NN (RNN) is the most general case of ANN, which stores a record of the prior inputs and factor them with the current data, to produce the output. In a RNN, information about past inputs is fed back into and mixed with the inputs, through recurrent or feedback connections, for hidden or

output neurons.

In this way, the neural network contains a memory of the past inputs via the activations. (Fig. 4).

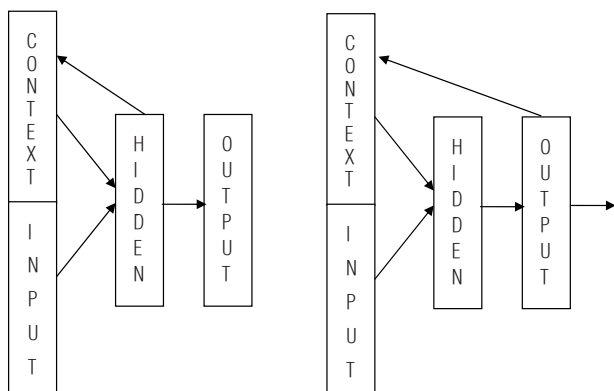


Figure 4. Partial recurrent neural networks.

Two types of RNN are known: partial and fully RNN.

Two major architectures for limited recurrent networks are widely used, the Elman structure and the Jordan structure. Elman suggested allowing feedback from the hidden neurons to a set of additional inputs called context neurons. The output neurons function may be linear or nonlinear.¹⁰

Earlier, Jordan described a network with feedback from the output back to a set of context neurons. This form of recurrence is a compromise between the simplicity of a feed-forward network and the complexity of a fully recurrent neural network because it still allows the popular back propagation training algorithm to be used.¹¹

Fully recurrent NN have each neuron connected to all the others. The output of a neuron depends not only from the others neurons outputs, but also from their previous outputs. The RNN neurons have nonlinear activation functions and a complex dynamic behaviour, so these networks are especially recommended for real time applications.

RNN with the same structure have different dynamic evolution if they are trained with a different algorithm. Thus a RNN is completely defined with both architecture and training algorithm.

Some of the training algorithms are back propagation through time with its variants and real time recurrent learning. These algorithms minimize the gradient of the global error. The numerical complexity of calculus is N^4 , where N is the number of the RNN neurons.

The reduced dimension of the RNN if compared with the MLP and RBF-NN, in conditions of the same performance, is the major argument of using RNN in real time applications.

AIM OF THE STUDY

The objective of our research was to predict prostate cancer capsule penetration after radical prostatectomy by using different architectures of artificial neural networks.

MATERIAL AND METHODS

In order to perform the simulations, we have used an original database, containing the records of 650 patients which underwent radical prostatectomy for prostate cancer between January 1, 1992 and December 31, 2005, at the Department of Urology of the Radboud University, Nijmegen, Holland. The impressive list of recorded parameters included:

- Date of birth;
- Age at diagnosis;
- Height;
- Weight;
- Date of positive diagnosis;
- Total Gleason Score in initial specimen (after biopsy or TURP);
- Primary, secondary, tertiary and quaternary Gleason score;
- Preoperative total PSA value;
- Preoperative TNM staging;
- Prostate volume;
- Date of prostatectomy;
- Capsule penetration;
- Seminal vesicles invasion;
- Positive margins;
- PSA recurrence;
- Time interval from prostatectomy to PSA recurrence.

After the analysis of the records, we have selected a number of relevant parameters for prostate cancer diagnosis and staging, excluding the cases with incomplete information. Thus, we have formed a new database of 548 cases for our retrospective study.

The selected parameters were:

- Preoperative stage (using the TNM 2002 classification): the patients in stages T_{2a} - T_{3b} were included;
- Preoperative total PSA value (range: 0-84 ng/mL);
- Total Gleason score (primary + secondary) in the initial specimen;
- Age;
- Capsule penetration (no/yes).

The final cohort included 365 cases with no prostate capsule penetration (organ-confined prostate cancer) and 183 cases diagnosed after radical

prostatectomy with capsule penetration.

As it was already demonstrated, the success of a ANN application depends significantly on the appropriate partition of the database. According to the accepted empirical rules, we have used the following datasets:³

- Set 1: 350 cases for training;
- Set 2: 50 cases for cross-validation;
- Set 3: 148 cases for testing.

The input data was randomized; pre-processing included the following input parameters:

- Age (absolute value);
- Total Gleason score (absolute value);
- Preoperative PSA (absolute value);
- Preoperative TNM stage (allocation method:

$T_{2a} = 1; T_{2c} = 2; T_{3a} = 3; T_{3b} = 4$).

Values of 0 and 1 were assigned to the predicted output, the capsule penetration, for the negative case (no penetration), respectively for the positive one.

As ANN development software application we have used Neuro Solutions 5.0 for Excel. Various ANN architectures have been developed and tested: multilayer perceptron (MLP), radial basis function NN (RBF-NN), competitive NN (CNN) and recurrent NN (RNN).

As performance parameters, the mean square error, the positive predictive value (PPV), negative predictive value (NPV) and global percentage of correct classification were considered.

A comparison of the performance of different NN architectures has been accomplished, in order to determine the best complexity/accuracy ratio. The best NN performances were afterwards compared with the best statistical prediction method, the logistic regression (LR).

RESULTS

A significant number of prostate capsule penetration prediction simulations were developed using various NN architectures, different activation functions and different training algorithms, in order to obtain the best complexity/accuracy ratio.

The input data were pre-processed and weighted in different ways. A stratification of the input parameters according to the latest statistical and representation concepts used in the current medical practice was tested.¹² The accuracy of predictions obtained with stratified input parameters has been almost equal with that obtained in the initial conditions, proving once again that NN can manage large quantities of data.

The MLP NN with the best performance had a structure of eight neurons on the first hidden layer,

four neurons on the second hidden layer and one neuron as output. A nonlinear activation function, the hyperbolic tangent, was used to update all the neurons and the backpropagation algorithm with momentum to train the ANN parameters.³

Another ANN with a good performance was a RBF-NN, combined with a MLP. The resulting RBF-MLP NN had a structure of eight neurons on the first hidden layer, four neurons on the second (the MLP layer) and one neuron as output. The Gaussian activation function and Euclidean distance were used for the RBF neurons and the hyperbolic tangent for the MLP. The competitive algorithm with conscience was used to obtain the RBF centers, while the backpropagation algorithm with momentum was used to determine the MLP parameters.

The best RNN, of Jordan-Elmann type, had a structure of four neurons on the first hidden layer, four neurons on the second hidden layer and one output neuron.

The performance parameters, respectively the mean squared error (MSE), positive predictive value (PPV) and negative predictive value (NPV) of these networks are given in Table 1. The MSE, the PPV and NPV in our study were better than the results of similar experiments available in literature.¹³⁻¹⁵

Table 1. ANN performance parameters.

Type of NN/ Performance parameter	MSE	PPV[%]	NPV[%]
MLP (8-4-1)	0.0475	91.25	98.14
RBF-MLP (8-4-1)	0.0717	90.58	96.82
RNN (4-4-1)	0.0403	92.33	98.14

Table 2 represents the global accuracy of prediction.

Table 2. Global prediction accuracy.

MLP (8-4-1)	97.12%
RBF-MLP (8-4-1)	95.88%
RNN (4-4-1)	97.37%
LR	94.89%

For comparison, logistic regression (LR) applied to the same database in order to predict the prostate capsule penetration was used. The percentage of correctly classified cases was 94.89%. We can observe that all ANN prediction models have performed better than the LR. In the best case it can be noticed difference of 2.48%, which is a very encouraging result.

CONCLUSIONS

The performance of ANN's in predicting prostate capsule penetration was better than that of the logistic regression, actually the best statistical prediction method. Even if the best improvement using an ANN is only 2.48%, in clinical terms this might be beneficial, avoiding unnecessary therapy in cases with prostate capsule penetration. This could have also a positive psychological impact on the patients, reducing the number of not needed surgical interventions.

The performance limits of the artificial neural network predictions, at this moment, are given by the smaller sample sizes and especially by the reliability of the data gathering process.

ACKNOWLEDGMENT

We express our gratitude to Prof. Fred Witjes and to Ass. Prof. Inge van Ort from the Department of Urology of the Radboud University, Nijmegen, the Netherlands, who granted us the permission to use their database.

REFERENCES

1. Thompson I, Thrasher JB, Aus G, et al. Guidelines for the management of clinically localized prostate cancer. *J Urol* 2007;177(6):2106-31.
2. Snow PB, Smith DS, Catalona WJ. Artificial neural networks in the diagnosis and prognosis of prostate cancer: a pilot study. *J Urol* 1994;152:1923-6.
3. Haykin S. *Neural networks*. Mcmillan: Englewood Cliffs, 1994.
4. Hu YH, Hwang JN. *Handbook of neural networks signal processing*. CRC Press: New York, 2002.
5. Hecht-Nielsen W. *Neurocomputing*. Addison-Wesley: New York, 1990.
6. Ahalt SC, Krishnamurty AK, Chen P, et al. Competitive algorithms for vector quantization. *Neur Netw* 1990;3:277-91.
7. Xu L, Krzyzak A, Oja AE. Rival penalized competitive learning for clustering analysis. RBF net and curve detection. *IEEE Trans Neural Networks* 1993;4:636-48.
8. Botoca C, Budura G, Miclau N. A new competitive learning algorithm for data clustering. *Proceedings of ICMCS, Chisinau, 2005*, p. 75-8.
9. Chen S, Gowan CF, Grant PM. Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Trans Neural Networks* 1991;2(2):302-9.
10. Elman JL. Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning* 1991:195-225.
11. Jordan M. Attractor dynamics and parallelism in a connectionist sequential machine. *Proceedings Eight Annual Conf. Cognitive Sci. Soc., Amherst, MA, 1986*, p. 531-46.
12. Makarov DV, Trock BJ, et al.. Updated nomogram to predict pathologic stage of prostate cancer given prostate-specific antigen level, clinical stage, and biopsy Gleason score (Partin tables) based on cases from 2000 to 2005. *Urology* 2007;69(6):1095-101.
13. Tewari A, Narayan P. Novel staging tool for localized prostate cancer: a pilot study using genetic adaptive neural networks. *J Urol* 1998;60:430-6.
14. Han M, Snow PB, Brandt JM, et al. Evaluation of artificial neural networks for the prediction of pathologic stage in prostate carcinoma. *Cancer* 2001;91:1661-6.
15. Borque A, Sanz G, Allepuz C, et al. The use of neural networks and logistic regression analysis for predicting pathological stage in men undergoing radical prostatectomy: a population based study. *J Urol* 2001;166:1672-8.