Optimizing the Performance of Probabilistic Neural Networks in a Bionformatics Task

V.L. Georgiou, N.G. Pavlidis, K.E. Parsopoulos, Ph.D. Alevizos, M.N. Vrahatis
Department of Mathematics,
University of Patras Artificial Intelligence Research Center (UPAIRC),
University of Patras, GR–26110 Patras, Greece
email: {vlg, npav, kostasp, philipos, vrahatis}@math.upatras.gr

ABSTRACT: A self adaptive probabilistic neural network model is proposed. The model incorporates the Particle Swarm Optimization algorithm to optimize the spread parameter of the probabilistic neural network, enhancing thus its performance. The proposed approach is tested on two data sets from the field of bioinformatics, with promising results. The performance of the proposed model is compared to probabilistic neural networks, as well as to four different feedforward neural networks. Different sampling techniques are used, and statistical tests are performed to justify the statistical significance of the results.

KEYWORDS: Bioinformatics, Probabilistic Neural Networks, Particle Swarm Optimization.

INTRODUCTION

The field of bioinformatics has rapidly developed during the last few years. A significant area of interest in this field is the prediction of protein localization sites in cells. Numerous systems, including rule–based systems (decision trees, decision rules), statistical learning systems (naive Bayes, Support Vector Machines) and neural networks, have been used to perform prediction and classification tasks on this problem [1].

An effective approach for addressing such tasks is the *Probabilistic Neural Network* (PNN) (cf. [1]). PNNs were introduced by Specht in 1990 [2]. They constitute a class of neural networks that combine some of the best attributes of statistical pattern recognition and feedforward neural networks. PNNs are the neural network implementation of kernel discriminant analysis. The classical PNN can be viewed as an "intelligent memory" since each training pattern is stored as a neuron of the network [3]. PNNs require small training times and produce outputs with Bayes posterior probabilities. These desirable features come at the expense of larger memory requirements and slower execution speed for the prediction of unknown patterns [2].

The performance of a PNN is largely influenced by the spread parameter (see next Section). In this paper, we propose a self adaptive probabilistic neural network model that incorporates the *Particle Swarm Optimization* (PSO) algorithm, to optimize the spread parameter of the PNN. PSO is a swarm intelligence optimization algorithm, motivated by the dynamics of socially organized colonies [4], which has proved to be very efficient on a plethora of applications in science and engineering [5]. The proposed, self adaptive, model is applied on two data sets from the field of bioinformatics, namely the E.coli and the Yeast data sets, using alternative sampling techniques. Further, its performance is compared to that of four different Feedforward Neural Network (FNN) models.

The paper is organized as follows: in the following Section the basic concepts of PNNs and PSO are briefly described. Next, the proposed approach is presented. Subsequently, the experimental setup, including descriptions of the data sets and the sampling techniques used, as well as, a presentation of the obtained results and their statistical analysis are reported. The paper ends with conclusions.

PROBABILISTIC NEURAL NETWORKS

The probabilistic neural network (PNN) was introduced by Specht [2]. It is a supervised neural network that is widely used in the area of pattern recognition, nonlinear mapping, and estimation of the probability of class membership and likelihood ratios [6]. It is closely related to Bayes classification rule, and Parzen nonparametric probability density function estimation theory [2], [7]. The fact that PNNs offer a way to interpret the network's structure in terms of probability density functions [3] is an important merit of this type of networks. The standard training procedure for PNNs

requires a single pass over all the patterns of the training set [2]. This characteristic renders PNNs faster to train compared to feedforward neural networks.

The structure of a PNN is similar to that of FNNs, although the architecture of a PNN is limited to four layers; the *input layer*, the *pattern layer*, the *summation layer*, and the *output layer*, as illustrated in Fig. 1. An input vector $X = (x_1, \ldots, x_n)^\top \in \mathbb{R}^n$, is applied to the n input neurons and is passed to the pattern layer. The neurons of the pattern layer are divided into K groups, one for each class. The ith pattern neuron in the kth group computes its output using a Gaussian kernel of the form,

$$F_{k,i}(X) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{\|X - X_{k,i}\|^2}{2\sigma^2}\right), \tag{1}$$

where $X_{k,i} \in \mathbb{R}^n$ is the center of the kernel, and σ, also known as the *spread (smoothing) parameter*, determines the size of the receptive field of the kernel. The summation layer of the network computes the approximation of the conditional class probability functions through a combination of the previously computed densities.

$$G_k(X) = \sum_{i=1}^{M_k} w_{ki} F_{k,i}(X), \quad k \in \{1, \ldots, K\}, \tag{2}$$

where M_k is the number of pattern neurons of class k, and w_{ki} are positive coefficients satisfying, $\sum_{i=1}^{M_k} w_{ki} = 1$. Pattern vector X is classified to belong to the class that corresponds to the summation unit with the maximum output,

$$C(X) = \arg \max_{1 \leqslant k \leqslant K} (G_k). \tag{3}$$

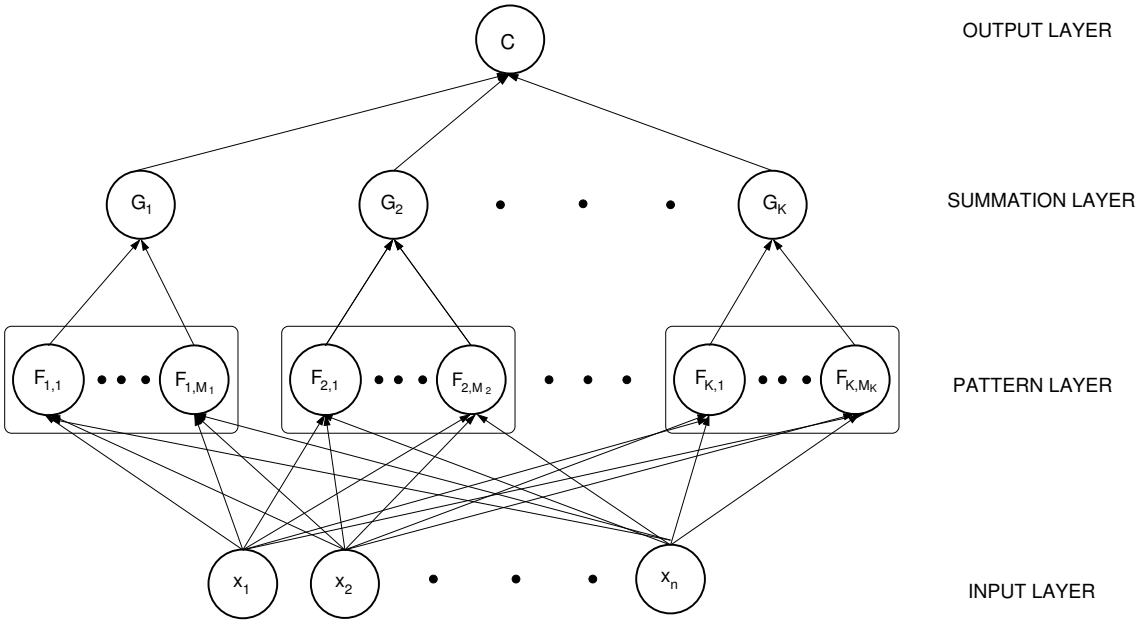

Figure 1: A probabilistic neural network.

PARTICLE SWARM OPTIMIZATION

Particle Swarm Optimization (PSO) is a stochastic, population–based optimization algorithm. It exploits a population of individuals to synchronously probe promising regions of the search space. In this context, the population is called a *swarm* and the individuals (i.e. the search points) are called *particles*. Each particle moves with an adaptable velocity within the search space, and retains a memory of the best position it ever encountered. In the *global* variant of PSO, the best position ever attained by all individuals of the swarm is communicated to all the particles at each iteration. In the *local* variant, each particle is assigned to a neighborhood consisting of prespecified particles. In this case, the best position ever attained by the particles that comprise the neighborhood is communicated among them [4].

Assume an d–dimensional search space, $S \subset \mathbb{R}^d$, and a swarm consisting of *NP* particles. The i–th particle is an d–dimensional vector, $Z_i = (z_{i1}, z_{i2}, \ldots, z_{id})^\top \in S$. The velocity of this particle is also a d–dimensional vector,

$V_i = (v_{i1}, v_{i2}, \ldots, v_{id})^\top \in S$. The best previous position encountered by the i–th particle in S is denoted by, $BP_i = (bp_{i1}, bp_{i2}, \ldots, bp_{id})^\top \in S$. Assume g_i to be the index of the particle that attained the best previous position among all the particles in the neighborhood of the i–th particle, and t to be the iteration counter. Then, the swarm is manipulated by the equations,

$$V_i(t+1) = \omega\, V_i(t) + c_1\, r_1\big(BP_i(t) - Z_i(t)\big) + c_2\, r_2\big(BP_{g_i}(t) - Z_i(t)\big), \quad (4)$$

$$Z_i(t+1) = Z_i(t) + V_i(t+1), \quad (5)$$

where $i = 1, \ldots, NP$, ω is a parameter called *inertia weight*; c_1 and c_2 are two positive constants called *cognitive* and *social* parameter, respectively; and r_1, r_2, are random variables uniformly distributed within $[0, 1]$. Alternatively, the velocity update can be performed through the following equation [8],

$$V_i(t+1) = \chi\Big[V_i(t) + c_1\, r_1\big(BP_i(t) - Z_i(t)\big) + c_2\, r_2\big(BP_{g_i}(t) - Z_i(t)\big)\Big], \quad (6)$$

where χ is a parameter called *constriction factor*, giving rise to a different version of PSO. Although Eqs. (4) and (6) are algebraically equivalent, there are significant differences regarding the selection of the corresponding parameters. Specifically, the constriction factor is derived analytically through the formula [8]

$$\chi = \frac{2\kappa}{|2 - \phi - \sqrt{\phi^2 - 4\phi}|}, \quad (7)$$

for $\phi > 4$, where $\phi = c_1 + c_2$, and $\kappa = 1$, derived through the stability analysis of the constriction factor version of PSO reported in [8],[9]. On the other hand, the inertia weight, ω, is usually determined empirically. An initial value close to 1 and gradually decline towards zero is considered a good configuration [10]. The initialization of swarm and velocities, is usually performed randomly and uniformly in the search space, although more sophisticated initialization techniques can enhance the overall performance of the algorithm [11].

THE PROPOSED APPROACH

As previously mentioned the spread parameter, σ, is critical for the classification accuracy of a PNN. In this work, the determination of σ is performed through PSO. Specifically, the PSO algorithm is initialized randomly with a swarm of values of σ, and optimization is performed for a specific sample taken from the data set. Thus, the obtained value of σ is the best achieved value with respect to the specific sample. This procedure can be applied for each sample to provide an appropriate σ with respect to classification accuracy. We call this model *Self Adaptive Probabilistic Neural Network* (SA–PNN). A pseudocode of the proposed approach is given in Table I.

Table I: Pseudocode for the SA–PNN algorithm.

PNN input: Normalized training set $S_{\text{train}} = \{X_1^{\text{train}}, \ldots, X_{N_{\text{train}}}^{\text{train}}\}$ and test set S_{test}.	
PSO input: Swarm size *NP*, χ, c_1, c_2, bounding box $\mathcal{B} = [0, 1]$	
Step 1	**Train** the PNN, using all the training set patterns in S_{train}.
Step 2	**Set** $t = 0$.
Step 3	**Initialize** a PSO swarm, $\sigma_i(t) \in \mathcal{B},\ V_i(t) \in \mathcal{B},\ i = 1, \ldots, NP$.
Step 4	**Initialize** best positions, $BP_i(t),\ i = 1, \ldots, NP$, and the index g.
	Do
Step 5	**Update** the velocities, $V_i(t+1),\ i = 1, \ldots, NP$, using Eq. (4), or Eq. (6).
Step 6	**Update** particles, $\sigma_i(t+1) = \sigma_i(t) + V_i(t+1),\ i = 1, \ldots, NP$.
Step 7	**Constrain** each particle $\sigma_i(t+1)$ in $\mathcal{B}$.
Step 8	**Compute** $f(\sigma_i(t+1))$ by the leave–one–out misclassification proportion on $S_{\text{train}},\ i = 1, \ldots, NP$.
Step 9	**Update** best positions, $BP_i(t+1),\ i = 1, \ldots, NP$, and the index g.
Step 10	**Update** iteration counter, $t = t + 1$.
	While (the error goal or the maximum number of iterations is not reached)
Step 11	**Write** the optimal spread parameter σ_g, and the corresponding performance of the PNN on S_{test}.

EXPERIMENTAL RESULTS

The proposed model was evaluated on two data sets from the field of bioinformatics, namely the E.coli and the Yeast data set [1]. To determine the spread parameter, a swarm of 20 particles was allowed to evolve for 100 generations. The global variant of the algorithm was considered because it exhibited faster convergence compared to the local one. The particles were constrained in the box $[0, 1]$ in order to avoid possible velocities explosion. In the constriction factor version, the values of the c_1 and c_2 parameters were set to 2.05, while χ was set to 0.729 [8]. An upper bound, V_{max}, on the absolute value of the velocities of the particles was used and set to 0.5. In the inertia weight version, the inertia weight, ω, was initialized to 1.0 and it gradually declined towards zero for the 75% of the available iterations.

The performance of the SA–PNN for different sampling techniques was investigated using the Stratified Random Sampling and the 10–Fold Cross–Validation sampling techniques, as well as, a Train–Validation–Test partitioning of the data set. The mean, standard deviation, minimum and maximum of the obtained success rates are reported for all sampling techniques. Statistical tests were performed to determine whether one of the considered sampling techniques yields a superior performance. Further, the performance of the resulting SA–PNN was compared to that of four different FNNs. The first three FNNs had a single hidden layer containing, 15, 20, and 25, neurons, respectively. The fourth FNN had two hidden layers with 10 neurons each. All the FNNs were trained using R–PROP [12] for 500 epochs, with learning rate equal to 0.01.

DESCRIPTION OF DATA SETS

A brief description of the considered data sets is provided below.

The E.coli Data Set
The goal for the E.coli data set is to predict the cellular localization sites of E.coli proteins [13]. There are 8 cellular sites, namely the cytoplasm (cp); inner membrane without signal sequence (im); periplasm (pp); inner membrane with uncleavable signal sequence (imU); outer membrane (om); outer membrane lipoprotein (omL); inner membrane lipoprotein (imL) and inner membrane with cleavable signal sequence (imS). The attributes are the McGeoch and Von Heijne recognition techniques for the signal sequence, the presence of charge on N–terminus of predicted lipoproteins, and 3 different scoring functions on the amino acid contents, predicted either as an outer or an inner membrane, cleavable or uncleavable sequence signal.

The Yeast Data Set
The objective is similar to the E.coli data, i.e. determine the cellular localization of the yeast proteins [13]. There are 10 sites: CYT (cytosolic or cytoskeletal); NUC (nuclear); MIT (mitochondrial); ME3 (membrane protein, no N–terminal signal); ME2 (membrane protein, uncleaved signal); ME1 (membrane protein, cleaved signal); EXC (extracellular); VAC (vacuolar); POX (peroxisomal) and ERL (endoplasmic reticulum lumen). The same attributes with the E.coli data set are considered, including additionally the nuclear localization information.

SAMPLING TECHNIQUES

The sampling techniques which were considered are briefly exposed below:

Stratified Random Sampling
In Stratified Random Sampling, a data set of size N, is divided in K non–overlapping subpopulations of size N_k, $k \in \{1, \ldots, K\}$, called *strata*, with

$$\sum_{k=1}^{K} N_k = N.$$

A random sample of size α_k, is selected from each stratum independently. The final stratified random sample has size α, where

$$\alpha = \sum_{k=1}^{K} \alpha_k,$$

and it is used as the training set, while the rest of the data form the test set. Proportionate allocation uses a sampling fraction of each strata that is proportional to that of the total population. The PSO algorithm determines the spread parameter by minimizing the leave–one–out misclassification proportion on the training set.

λ–Fold Cross–Validation
According to this technique, the initial set of data is divided into λ parts of approximately equal size. Subsequently, each

Table II: SA–PNN test set classification accuracy percentage for the E.coli data set.

Model	Sampling	Mean	St.Dev.	Min.	Max.
SA–PNN(con)	Stratif. Rand. Sampl.	86.02	6.17	62.27	91.26
SA–PNN(ine)	Stratif. Rand. Sampl.	**86.65**	5.83	64.67	**92.31**
SA–PNN(con)	10-fold Cross–Valid.	75.67	1.51	72.60	78.46
SA–PNN(ine)	10-fold Cross–Valid.	75.72	**1.45**	71.71	78.67
SA–PNN(con)	Train-Valid.-Test Set	82.50	5.02	64.54	90.00
SA–PNN(ine)	Train-Valid.-Test Set	82.63	4.86	**72.72**	90.00

Table III: SA–PNN test set classification accuracy percentage for the Yeast data set.

Model	Sampling	Mean	St.Dev.	Min.	Max.
SA–PNN(con)	Stratif. Rand. Sampl.	58.71	7.90	40.50	**76.33**
SA–PNN(ine)	Stratif. Rand. Sampl.	**63.47**	7.64	38.79	75.32
SA–PNN(con)	10-fold Cross–Valid.	48.58	**1.34**	**43.88**	50.74
SA–PNN(ine)	10-fold Cross–Valid.	48.65	1.37	43.87	50.75
SA–PNN(con)	Train-Valid.-Test Set	54.00	5.79	34.96	61.96
SA–PNN(ine)	Train-Valid.-Test Set	54.01	4.23	42.53	61.55

of the λ subsets is used as the test set while the other subsets form the training set. The average error across all λ trials is computed. As in stratified random sampling, the PSO algorithm determines the spread parameter by minimizing the leave–one–out misclassification proportion on the training set.

Train–Validation–Test Partitioning
The practice of partitioning the data set into three components is standard in the neural network literature [14]. In the context of SA–PNN, we train the network using the patterns of the train set. The PSO algorithm, subsequently, determines the spread parameter, σ, by minimizing the misclassification error on the validation set.

PRESENTATION OF THE RESULTS AND STATISTICAL ANALYSIS

In Table II, the mean (Mean), standard deviation (St.Dev.), as well as, the minimum (Min.) and maximum (Max.) success rates (in percentage terms) for the E.coli data set, are reported for each sampling technique and for both the inertia weight (ine), and the constriction factor (con), version of the PSO algorithm. The same information for the Yeast data set is reported in Table III. With respect to the stratified random sampling technique, the size of the random sample for each stratum was set to 50% of the size of the corresponding class. This practice was adopted in all the experiments with stratified random sampling. As can be seen from Tables II and III, SA–PNNs that used the training sets obtained through stratified random sampling, gave with the highest mean, and maximum, classification accuracy. For the E.coli data set the sampling technique that corresponded to the highest minimum classification accuracy was the Train–Validation–Test partitioning, while for the Yeast data set it was 10–fold cross validation. We would like to point out that the maximum attained performance on the E.coli data set (92.31%) compares favorably with the results obtained through a data selection method proposed in [15].

Applying a one–way ANOVA test on the mean performance of the different sampling techniques, for both data sets, the statistical significance of the observed variations in performance was verified. The corresponding p–value was 0 up to four decimal digits. Performing pair–wise t–tests, stratified random sampling was found to yield the best performance (corresponding p–value equal to 0 up to four decimal digits for both data sets). It is important to note that stratified random sampling produced smaller training sets compared to 10–fold cross–validation.

Despite the fact that the inertia weight version of the PSO algorithm appears to produce slightly superior mean classi-

Table IV: Test set classification accuracy (%) of FNNs on the E.coli data set with Stratified Random Sampling.

Topology	Mean	St.Dev.	Min.	Max.
7–15–8	85.86	2.40	78.91	89.75
7–20–8	**85.93**	**1.94**	79.51	**90.36**
7–25–8	84.90	2.55	78.31	89.75
7–10–10–8	85.03	2.13	**80.72**	**90.36**

Table V: Test set classification accuracy (%) of FNNs on the Yeast data set with Stratified Random Sampling.

Topology	Mean	St.Dev.	Min.	Max.
8–15–10	56.91	1.69	53.58	60.21
8–20–10	57.54	**1.42**	53.04	60.35
8–25–10	**58.26**	1.43	**55.20**	**61.97**
8–10–10–10	56.49	1.74	51.42	59.94

fication accuracy relative to the constriction factor version, the difference was not found to be statistically significant (the corresponding t–test p–value was 0.5431). This was not the case for the Yeast data set were the inertia weight version of the algorithm produced a statistically significant superior performance (the one–sided t–test p–value was 0.0083).

In Table IV, the mean, standard deviation, as well as, the minimum and maximum success rates (in percentage terms) for the E.coli data set, are reported for all the FNNs considered using stratified random sampling. Respectively, Table V illustrates this information for the Yeast data set. For both data sets, we performed t–tests to evaluate the statistical significance of the difference of mean classification accuracy between the best performing FNN with that of the SA–PNN obtained through stratified random sampling. For the E.coli data set the mean performance difference was not found to be statistically significant (p–value equal to 0.5725). For the Yeast data set the SA–PNNs using the inertia weight version of PSO were found to outperform the best performing FNNs (the corresponding p–value of the one–sided t–test was 3.74×10^{-6}).

CONCLUSIONS

A self adaptive model for probabilistic neural networks was proposed. The proposed approach incorporates the Particle Swarm Optimization Algorithm to find an appropriate spread parameter for the probabilistic neural network with respect to the resulting classification accuracy. The effectiveness of the proposed model is assessed on two data sets from the field of bioinformatics (E.coli and Yeast), with encouraging results. Among the three sampling techniques considered, stratified random sampling proved to yield the best classification performance despite the fact that the training sets it produces are smaller in magnitude compared to 10–fold cross validation. To evaluate further the capabilities of this model, comparisons with feedforward neural network models trained using the R–PROP algorithm were performed. These comparisons showed that the self adaptive model proposed achieves statistically significant superior performance on the Yeast data set, while on the E.coli data set the performance of the two models is not found to be statistically different.

Future work will include the generalization of the proposed self adaptive scheme to determine the corresponding matrix of spread parameters, to further optimize the performance of PNNs. We also intend to apply different evolutionary computation techniques on this problem.

ACKNOWLEDGMENT

The authors would like to thank the referees for their useful comments and suggestions. Also, the authors acknowledge partial support by the "Heraklitos" research grant, as well as, the "Pythagoras" research grant awarded by the Greek

Ministry of Education and Religious Affairs and the European Union.

REFERENCES

[1] A. C. Tan; D. Gilbert, 2003, "An empirical comparison of supervised machine learning techniques in bioinformatics", In *Proceedings of the* 1*st Asia Pacific Bioinformatics Conference (APBC 2003)*, pp. 219–222.

[2] D. F. Specht, 1990, "Probabilistic neural networks", *Neural Networks*, 1(3), pp. 109–118.

[3] M. Berthold; J. Diamond, 1998, "Constructive training of probabilistic neural networks", *Neurocomputing*, pp. 167–183.

[4] J. Kennedy; R.C. Eberhart, 2001, "Swarm intelligence", Morgan Kaufmann Publishers.

[5] K.E. Parsopoulos; M.N. Vrahatis, 2002, "Recent approaches to global optimization problems through particle swarm optimization", *Natural Computing*, 1(2–3), pp. 235–306.

[6] D. F. Specht; H. Romsdahl, 1994, "Experience with adaptive probabilistic neural network and adaptive general regression neural network", In *Proceedings of the IEEE International Conference on Neural Networks*, volume 2, pp. 1203–1208.

[7] E. Parzen, 1962, "On the estimation of a probability density function and mode", *Annals of Mathematical Statistics*, 3, pp. 1065–1076.

[8] M. Clerc; J. Kennedy, 2002, "The particle swarm–explosion, stability, and convergence in a multidimensional complex space", *IEEE Trans. Evol. Comput.*, 6(1), pp. 58–73.

[9] I. C. Trelea, 2003, "The particle swarm optimization algorithm: Convergence analysis and parameter selection", *Information Processing Letters*, 85, pp. 317–325.

[10] Y. Shi; R.C. Eberhart. "Parameter selection in particle swarm optimization". In V.W. Porto; N. Saravanan; D. Waagen; A.E. Eiben, editors, *Evolutionary Programming*, volume VII, pp. 591–600. Springer, 1998.

[11] K. E. Parsopoulos; M. N. Vrahatis. "Initializing the particle swarm optimizer using the nonlinear simplex method". In A. Grmela; N.E. Mastorakis, editors, *Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, pp. 216–221. WSEAS Press, 2002.

[12] M. Riedmiller; H. Braun, 1993, "A direct adaptive method for faster backpropagation learning: The rprop algorithm", In *Proceedings of the IEEE International Conference on Neural Networks, San Francisco, CA*, pp. 586–591.

[13] P. Horton; K. Nakai, 1996, "A probabilistic classification system for predicting the cellular localization sites of proteins", In *Proceedings of* 4*th International Conference on ISMB*, pp. 109–115.

[14] S. Haykin, 1999, "Neural networks: A comprehensive foundation", New York: Macmillan College Publishing Company.

[15] B. Bolat; T. Yildirim, 2003, "A data selection method for probabilistic neural networks", In *XII International Turkish Symposium on Artificial Intelligence and Neural Networks TAINN'2003*.