International Journal on Artificial Intelligence Tools
Vol. 15, No. 3 (2006) 371–396

World Scientific
www.worldscientific.com

NEW SELF-ADAPTIVE PROBABILISTIC NEURAL NETWORKS IN BIOINFORMATIC AND MEDICAL TASKS

V. L. GEORGIOU*, N. G. PAVLIDIS†, K. E. PARSOPOULOS‡, PH. D. ALEVIZOS§ and M. N. VRAHATIS¶

Computational Intelligence Laboratory (CI Lab), Department of Mathematics, University of Patras Artificial Intelligence Research Center (UPAIRC), University of Patras, GR–26110 Patras, Greece

*vlg@math.upatras.gr
†npav@math.upatras.gr
‡kostasp@math.upatras.gr
§philipos@math.upatras.gr
¶vrahatis@math.upatras.gr

We propose a self–adaptive probabilistic neural network model, which incorporates optimization algorithms to determine its spread parameters. The performance of the proposed model is investigated on two protein localization problems, as well as on two medical diagnostic tasks. Experimental results are compared with that of feedforward neural networks and support vector machines. Different sampling techniques are used and statistical tests are conducted to calculate the statistical significance of the results.

Keywords: Probabilistic neural networks; bioinformatics; particle swarm optimization.

1. Introduction

The field of Bioinformatics has rapidly developed during the past 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 artificial neural networks have been used to perform prediction and classification tasks[1,2].

Medicine is another field in which computational intelligence methods are frequently applied to address signal processing and classification tasks. Radiation therapy treatment planning systems, for example, can be designed and optimized through soft computing simulation methodologies[3]. Moreover, artificial neural networks have been employed for the investigation of pathological cases, like heart disease and tumor diagnosis[4].

Research in bioinformatics and medical tasks is driven by experimental data. Typically, there is a vast amount of data but little theory available. To this end machine learning techniques are extensively applied in these fields. An advantage of probabilistic neural networks (PNNs) as classifiers is their ability to function

as an "intelligent memory" so that there is no loss of the available information. Furthermore, a PNN not only classifies a new pattern but also provides a measure of the uncertainty of that classification. For example, in a cancer classification task, a PNN estimates the probability of having a benign or malignant cancer and not just a true-false answer.

PNNs which were introduced by Specht in 1990[5], 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. In contrast to feedforward neural networks that are black-box systems, PNNs use Bayesian strategies for pattern classification, a familiar process to decision makers. Moreover, the contribution of each neuron of the PNN can be calculated, something that is not possible when using FNNs. The classical PNN can be viewed as an "intelligent memory" since each training pattern is stored as a neuron of the network[6]. 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[5].

PNNs have been implemented in a plethora of Bioinformatics and Medical tasks. In Ref. 7, a comprehensive study of the capability of PNNs along with a feature extraction method to perform cancer classification is presented. The experimental results show that the conjugation of the probabilistic neural network and the feature selection method can achieve 100% recognition accuracy in the AAL/AML classification[7], and also attain satisfactory results in two colon cancer data sets. In Ref. 8, PNNs are employed to develop accurate NMR-based metabonomic models for the prediction of xenobiotic-induced toxicity in experimental animals and their potential use in accelerated drug discovery programs is highlighted. PNNs have also been considered for the design of an automatic, reliable and efficient prediction system for protein subcellular localization which is needed for large-scale genome analysis[9]. In Ref. 9, a boosting algorithm and a PNN are integrated into a prediction system which yielded superior prediction performance compared to existing algorithms. In Ref. 10, PNNs are successfully employed in magnetic resonance image analysis to identify subtle changes in brain tissue quantities and volumes which are relevant in the diagnosis of neurological diseases.

The performance of PNNs is largely influenced by the spread parameters (see Subsection 2.1). To alleviate this influence, we propose a self–adaptive PNN model, which incorporates an optimization algorithm to set the spread parameters of the network to appropriate values[11]. The selection of appropriate spread parameters involves the minimization of a non–differentiable error function. To this end, we employ only algorithms that do not require gradient information. When considering PNNs that employ a single spread parameter (*homoscedastic* PNNs) we consider the golden section search method as well as a swarm intelligence algorithm, namely the *Particle Swarm Optimization* (PSO) algorithm. For the PNNs that employ a number of spread parameters equal to the dimensionality of the input

space (*heteroscedastic* PNNs) we consider only the PSO method. The selection of PSO is based on its efficiency in solving a plethora of applications in science and engineering[3,12,13,14,15,16,17].

The proposed model is applied on two data sets from the field of Bioinformatics, namely the E.Coli and Yeast data sets[18], as well as, on the Breast Cancer[19,20] and Diabetes[19,21] medical data sets. The performance achieved through different sampling techniques, namely *Stratified Random Sampling*, *Cross–Validation*, and the *Train–Validation–Test Partitioning* is also investigated. The performance of the new model is compared to that of feedforward neural networks (FNNs) and Support Vector Machines (SVMs) and extensive statistical tests are conducted to justify the significance of the results.

The paper is organized as follows: in Section 2 the basic concepts of PNNs and PSO are briefly described. In Section 3 the proposed approach is presented along with the experimental results, including descriptions of the data sets and the employed sampling techniques. The paper concludes in Section 4.

2. Background Material

2.1. *Probabilistic neural networks*

PNNs constitute a supervised neural network model that is widely used in the area of pattern recognition, nonlinear mapping, and estimation of probability of class membership and likelihood ratios[22]. PNNs are closely related to the Bayes classification rule[23,24] and Parzen nonparametric probability density function estimation theory[5,25], thereby offering a way to interpret the network's structure in terms of probability density functions[6]. The standard training procedure of PNNs requires a single pass over all the patterns of the training set[5]. This characteristic renders PNNs faster to train, compared to FNNs.

The structure of a PNN is similar to that of FNNs, although the architecture of a PNN is always limited to four layers; the *input layer*, *pattern layer*, *summation layer*, and *output layer*, as illustrated in Fig. 1. Let $X_{i,k} \in \mathbb{R}^n$ be the ith pattern of the training set that belongs to category k, with $i = 1, 2, \ldots, M_k$, $k = 1, 2, \ldots, K$, where M_k is the size of class k, and K is the number of categories. For each $X_{i,k}$, a neuron in the pattern layer of the network is created. The center of the Gaussian kernel activation function of this neuron is $X_{i,k}$. The output of the specific pattern neuron is connected to the neuron of the summation layer that corresponds to the class, k, in which the training pattern is classified.

An input vector, $X \in \mathbb{R}^n$, is applied to the input neurons, x_i, $i = 1, 2, \ldots, n$, 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,

$$f_{i,k}(X) = \frac{1}{(2\pi)^{n/2} \det(\Sigma)^{1/2}} \exp\left(-\frac{1}{2}(X - X_{i,k})^\top \Sigma^{-1} (X - X_{i,k})\right), \qquad (1)$$

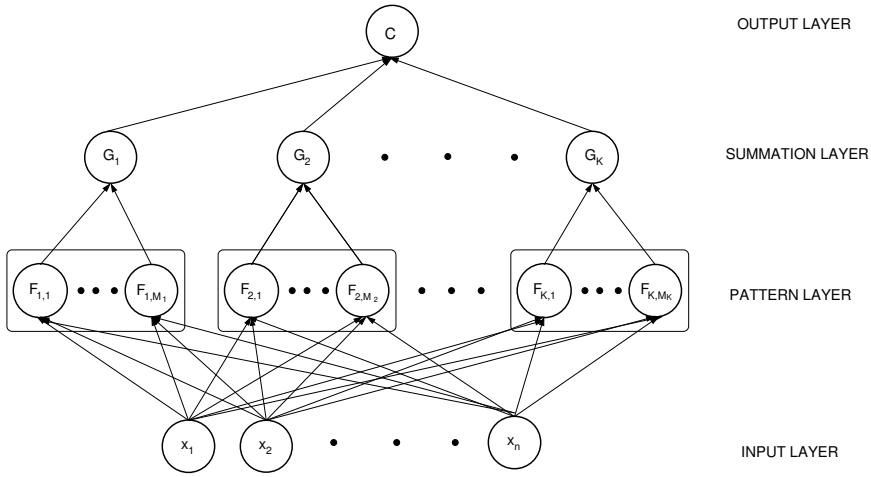


Fig. 1. A probabilistic neural network.

where $X_{i,k} \in \mathbb{R}^n$ is the center of the kernel; Σ, also known as the matrix of *spread (smoothing) parameters*, determines the size and shape of the receptive field of the kernel, and $\det(\Sigma)$ denotes the determinant of the matrix Σ. In his original contribution Specht[5] restricted Σ to one global smoothing parameter, σ^2. Substituting $\Sigma = \sigma^2 I$, where I is the identity matrix, Eq. (1) becomes,

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

PNNs that exploit a global smoothing parameter are called *homoscedastic*. On the other hand, with the term *heteroscedastic* PNN we refer to networks that use a diagonal matrix Σ instead of a scalar (σ^2)[22].

The summation layer of the network computes the approximation of the conditional class probability functions through a combination of the previously computed densities,

$$G_k(X) = w_k \sum_{i=1}^{M_k} f_{i,k}(X), \qquad k \in \{1, 2, \ldots, K\}, \tag{2}$$

where w_k is a positive coefficient standing for the prior probability of class k, satisfying the relation,

$$\sum_{k=1}^{K} w_k = 1.$$

An input vector X is classified to the class that corresponds to the summation unit with the maximum output,

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

Note that the prior class probabilities, w_k, are problem–dependent, and it may not always be feasible to estimate them from the training data since the training set may contain little meaningful information regarding the prior class probabilities[26]. In our experiments, we assume that all classes are equiprobable, i.e.,

$$w_k = 1/K, \qquad 1 \leqslant k \leqslant K\,.$$

An important feature of PNNs compared to alternative classification methods is that, when a new input vector is given, the outputs of a PNN correspond to the conditional posterior probabilities of class membership. This is particularly important in bioinformatics and medical applications because a decision threshold can be set and if all the posterior probabilities are below this threshold the pattern can be classified as unknown. Moreover, Bayesian confidence intervals can be computed.

Heteroscedastic PNNs can also provide information for feature selection in cases where the dimension of the problem is large. When the appropriate spread parameter that corresponds to the ith variable (feature), σ_i^2, is large, it is an indication that the specific variable is not important in the computation of the outputs of the PNN, and can be ignored.

A limitation of PNNs is the curse of dimensionality. When the dimension of the dataset is large, PNNs usually do not yield good results. The performance of PNNs is also inhibited by the presence of dependent variables in the dataset. Both these problems can be alleviated to an extent using principal component analysis (PCA) prior to training the PNN. This procedure yields new variables that are independent and furthermore quantifies the information content of each new variable. Although PCA may succeed in reducing the dimensionality, the new variables can be difficult to interpret.

2.2. *Particle swarm optimization*

Nature constitutes source of inspiration for the development of new, distributed optimization algorithms that can address difficult optimization problems regardless of any special structure and mathematical properties of the underlying objective function. Evolutionary algorithms have been developed based on ideas that imitate procedures that take place in human DNA and they have been applied with success in a plethora of applications[27]. Particle Swarm Optimization (PSO) was introduced in 1995 by Eberhart and Kennedy[28,29]. The inspiration for its development sprang from the dynamics of socially organized groups of individuals. The algorithm adheres the five basic principles of swarm intelligence as they were defined by Millonas[30], therefore it is categorized as a swarm intelligence algorithm[12].

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 in a memory the best position it ever encountered. This best position is shared with other particles in the swarm.

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[12]. Thus, the global variant can be considered as a special case of the local variant in which, the entire swarm is considered as the neighborhood of each particle. The most usual neighborhood topology is the ring topology, where all particles are assumed to lie on a ring, i.e., Z_1 is considered to be the particle that follows immediately after Z_N. This topology was adopted in our experiments.

Assume a d–dimensional search space, $\mathcal{S} \subset \mathbb{R}^d$, and a swarm consisting of NP particles. The ith particle is a d–dimensional vector,

$$Z_i = (z_{i1}, z_{i2}, \ldots, z_{id})^\top \in \mathcal{S}.$$

The velocity of this particle is also a d–dimensional vector,

$$V_i = (v_{i1}, v_{i2}, \ldots, v_{id})^\top.$$

The best previous position encountered by the ith particle in $\mathcal{S}$ is denoted by,

$$BP_i = (bp_{i1}, bp_{i2}, \ldots, bp_{id})^\top \in \mathcal{S}.$$

Assume g_i to be the index of the particle that attained the best previous position among all particles in the neighborhood of Z_i, and t to be the iteration counter. Then, the swarm is manipulated by the equations[31],

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

$$Z_i(t+1) = Z_i(t) + V_i(t+1)\,, \tag{5}$$

where $i = 1, 2, \ldots, NP$; 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 vectors that consist of values uniformly distributed within $[0, 1]$. All vector operations in Eqs. (4) and (5) are performed componentwise. Alternatively, the velocity update can be performed through the equation[32],

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

where χ is a parameter called *constriction coefficient*, giving rise to a different version of PSO. The best positions are then updated according to the equation

$$BP_i(t+1) = \begin{cases} Z_i(t+1), & \text{if } \; f\left(Z_i(t+1)\right) < f\left(BP_i(t)\right), \\ BP_i(t), & \text{otherwise,} \end{cases}$$

and the indices g_i, $i = 1, 2, \ldots, NP$, are updated. The particles are always bounded in the search space $\mathcal{S}$.

Although Eqs. (4) and (6) are algebraically equivalent, there are significant differences regarding the selection of the corresponding parameters. Specifically, the constriction coefficient is derived analytically through the formula[32],

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

for $\phi > 4$, where $\phi = c_1 + c_2$, and $\kappa = 1$, based on the stability analysis of Clerc and Kennedy[32,33]. On the other hand, the inertia weight, ω, is usually determined empirically. An initial value close to one and gradually decline towards zero is considered a good configuration, since it promotes exploration (better global search) at the beginning and exploitation (refined local search) at later stages of the algorithm's execution[34].

In general, the global variant of PSO exhibits faster convergence rates, although, in some cases it may reduce the swarm's diversity very fast, thereby getting trapped in local minimizers. On the other hand, the local variant, especially when the neighborhood size is small, exhibits superior exploration capabilities at the cost of slower convergence.

3. The Proposed Approach and Experimental Results

The matrix of spread parameters, Σ, is crucial for the classification accuracy of a PNN. In our study, we allow Σ to be a diagonal matrix whose entries can differ. Obviously, in the one–dimensional case, Σ becomes σ^2. We refer to the former case as heteroscedastic PNN, while the latter case is referred to as homoscedastic. For the homoscedastic case the optimization task is in one dimension, whereas for the heteroscedastic case the dimension of the optimization problem equals that of the pattern vectors.

We employ the PSO algorithm to determine σ^2 in the one–dimensional case or Σ in the multidimensional case, while, for the univariate case, the golden section search method is also used. Specifically, for the PSO method, a swarm of particles, which are diagonals of matrices Σ, is initialized randomly in $[0, 5]^d$, where d denotes the dimension of the optimization problem, and the optimization algorithm is performed for a specific sample taken from the data set at hand. The range in which the diagonal elements of Σ typically lie is $[0, 2]$, as proposed in (Ref. 35). However, we extended the range to $[0, 5]$, since preliminary experiments for the multidimensional case resulted in promising values of the spread parameter that were larger than 2. The obtained value for the elements of Σ is the best achieved value with respect to the specific sample. This procedure can be applied for any sample to provide the corresponding optimal Σ. We call this model *self–adaptive probabilistic neural network* (SA–PNN). A pseudocode of the proposed approach is given in Table 1.

As outlined in Table 1, the first stage of the proposed approach is identical to the training procedure proposed by Specht[5]. More specifically, for each pattern of the training set, a neuron in the pattern layer is constructed and the center of its kernel activation function is set equal to the feature vector. Finally the output of

Table 1. Pseudocode for the SA–PNN algorithm.

PNN input: Training set $S_{\text{train}} = \{X_1^{\text{train}}, \ldots, X_N^{\text{train}}\}$ and test set S_{test}.
PSO input: Swarm size NP, χ, c_1, c_2, bounding box $\mathcal{B} = [0, 5]^d$.

Perform a pass over the training set to construct the PNN.
Set $t = 0$.
Initialize a PSO swarm, $Z_i(t) = \text{diag}(\Sigma_i(t)) \in \mathcal{B}$, $V_i(t) \in \mathcal{B}$, $i = 1, 2, \ldots, NP$.
Initialize best positions, $BP_i(t)$, $i = 1, 2, \ldots, NP$, and the indices g_i.
Do
  Update velocities, $V_i(t+1)$, $i = 1, 2, \ldots, NP$, using Eq. (4) or Eq. (6).
  Update particles, $Z_i(t+1) = Z_i(t) + V_i(t+1)$, $i = 1, 2, \ldots, NP$.
  Constrain each particle $Z_i(t+1)$ in $\mathcal{B}$.
  Compute the fitness function: $f(Z_i(t+1))$.
  Update best positions, $BP_i(t+1)$, $i = 1, 2, \ldots, NP$, and the indices g_i.
  Update iteration counter, $t = t + 1$.
While (the maximum number of iterations is not reached)
Write the optimal spread parameter and the corresponding performance of the PNN on S_{test}.

the neuron is connected to the summation layer neuron that collects the outputs of all the pattern neurons that correspond to the same class as that of the specific training pattern. The only difference is that, at this stage, a matrix Σ need not be determined by the user. Subsequently, the steps taken by the PSO algorithm are outlined.

For each particle, $Z_i = \text{diag}(\Sigma_i)$, $i = 1, 2, \ldots, NP$, the leave-one-out misclassification proportion on the training set is computed and this value is used as the fitness value $f(Z_i)$ of the particle. According to the leave–one–out method, a PNN is trained using all but one of the patterns from the training set. The excluded pattern is subsequently used to assess the classification ability of the network. This process is repeated excluding a different pattern of the training set each time, until all patterns of this set are excluded once. The leave-one-out method is computationally very expensive for FNNs as it requires the network to be trained as many times as the number of training patterns. However, the PNN architecture renders the application of the leave-one-out procedure very simple. It is sufficient to exclude the output of the pattern layer neuron, $f_{i,k}$, that corresponds to the "left-out" pattern, $X_{i,k}$, from the summation that takes place at the corresponding neuron, G_k, in the summation layer. The adaptation process is terminated when a maximum number of iterations is reached.

The proposed model was evaluated on two data sets from the field of Bioinformatics, namely the E.coli and the Yeast data set, as well as, on the Cancer and Diabetes medical data sets. The spread parameter was determined using PSO with swarms of 5 and 10 particles for the one–dimensional and multidimensional case, respectively, that were evolved for 50 iterations. The local variant of the algorithm was used, as it exhibited better performance, compared to the global variant, due to its enhanced exploration capability. The particles were constrained in the range $[0, 5]^d$, since this is the range of interest regarding the values of the elements of Σ. Both

the constriction coefficient and the inertia weight PSO version were considered. In the constriction coefficient version, the default parameter values, $c_1 = c_2 = 2.05$, $\chi = 0.729$, were used[32]. An upper bound, $V_{\max}$, on the absolute value of the components of the velocity vectors of the particles was also used and set to 0.3, in order to hinder the particles from possible bouncing on the bounds of the search space. In the inertia weight version, the inertia weight was initialized to 1.0 and it gradually declined towards 0.1 throughout the first 75% of the available iterations (i.e., for 37 iterations). Then it remained fixed to the value 0.1[16].

The performance of the SA–PNN for different sampling techniques was investigated using the Stratified Random Sampling and the λ–Fold Cross–Validation sampling techniques, as well as, a Train–Validation–Test partitioning of the data set. Statistical tests were performed to determine whether one of the considered sampling techniques yields a superior performance. Furthermore, the performance of the resulting SA–PNN was compared to that of a 7-16-8 FNN for the E.coli dataset[36], an 8-16-10 FNN for the Yeast case[36], an 8-4-2-2 FNN for the Breast Cancer dataset[19], and an 8-2-2-2 FNN for the Diabetes dataset[19]. All the FNNs were trained using the well–known and widely used Resilient Backpropagation (Rprop) method[37]. The choice of this technique was based on its efficiency and effectiveness on several problems. For a brief exposition of FNNs and the Rprop training algorithm refer to Appendix A.

3.1. *Description of the 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[18,38]. 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 size of the data set is 336 without any missing values and all variables are continuous.

The Yeast Data Set: The objective is similar to the E.coli data, i.e., determine the cellular localization of the Yeast proteins[18,38]. There are 10 sites, namely the 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. Thus, there are 8 continuous inputs of 1484 instances without missing values.

The Breast Cancer Data Set: This data set was provided by the University of Wisconsin hospitals in 1992[19,20]. There are two possible classes for each record: benign or malignant. The input features are the uniformity of cell size and shape; bland chromatin; single epithelial cell size; and mitoses. So there are 9 continuous inputs and 699 instances. Also, there are no missing values.

The Pima Indians Diabetes Data Set: This data set was provided by the John Hopkins University in 1992[21]. It concerns the Pima Indians diabetes and the input features are the diastolic blood pressure; triceps skin fold thickness; plasma glucose concentration in a glucose tolerance test; and diabetes pedigree function. The classification is performed regarding the exhibition of diabetes signs of the patient based on the criteria of the World Health Organization. The 8 inputs are all continuous without missing values and there are 768 instances. The aim is to classify whether someone is infected by diabetes or not, therefore, there are two classes.

3.2. *Description of the sampling techniques*

The considered sampling techniques are described below.

Stratified Random Sampling: In Stratified Random Sampling (SRS), a data set of size N is divided in K non–overlapping subpopulations, called *strata*, each containing the N_k patterns that belong to class $k \in \{1, 2, \ldots, K\}$ (where K is the total number of classes present in the dataset), 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 which is used as training set by the classifiers has size α, where,

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

while the rest of the data form the test set. Proportionate allocation uses a constant sampling fraction, α_k/N_k, for each stratum.

λ–Fold Cross–Validation: According to this technique, the initial set of data is divided into λ parts of approximately equal size. Subsequently, each 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. The 4–Fold and 10–Fold Cross–Validation techniques[36], denoted as 4–CV and 10–CV, respectively, were used in our experiments.

Table 2. Test set classification accuracy percentage for E.coli data set.

Sampling	Model	Mean	Median	St.Dev.	Min.	Max.	CI low	CI up
SRS	PNN	86.33	**86.75**	2.28	80.72	89.76	85.26	87.39
	GSS	**86.42**	86.14	2.18	**82.53**	89.76	85.40	87.44
	PCU	86.39	**86.75**	**2.10**	**82.53**	90.36	85.40	87.37
	PIU	85.72	86.14	3.33	74.70	90.36	84.16	87.28
	PCM	81.42	81.33	5.27	70.48	90.36	78.95	83.88
	PIM	81.78	81.93	4.65	74.70	**90.96**	79.60	83.95
	FNN 7–16–8	85.39	84.94	2.40	81.33	90.36	84.27	86.52
	SVM	69.67	69.28	2.58	64.46	74.10	68.46	70.88
4-CV	PNN	**86.86**	86.76	0.72	**85.42**	**88.10**	86.52	87.20
	GSS	86.73	86.76	0.90	84.52	**88.10**	86.30	87.15
	PCU	86.79	86.76	**0.58**	**85.42**	87.80	86.51	87.06
	PIU	86.77	**86.90**	0.67	85.12	**88.10**	86.46	87.08
	PCM	83.44	82.89	1.62	81.25	86.31	82.68	84.20
	PIM	83.07	82.89	2.48	79.17	87.50	81.90	84.23
	FNN 7–16–8	85.98	86.01	1.00	83.04	87.20	85.51	86.45
	SVM	75.00	75.00	0.79	72.02	75.89	74.63	75.37
TVT	PNN	85.48	85.54	3.92	77.11	91.57	83.65	87.32
	GSS	**86.45**	85.54	**2.79**	**81.93**	91.57	85.14	87.75
	PCU	86.33	**86.14**	3.21	80.72	**92.77**	84.82	87.83
	PIU	86.08	**86.14**	3.92	79.52	**92.77**	84.25	87.92
	PCM	81.69	81.93	5.33	71.08	90.36	79.19	84.18
	PIM	82.41	81.93	5.48	71.08	**92.77**	79.85	84.97
	FNN 7–16–8	85.06	**86.14**	3.57	78.31	90.36	83.39	86.73
	SVM	68.07	67.47	6.19	55.42	78.31	65.18	70.97

Train–Validation–Test Partitioning: The practice of partitioning the data set into three components is standard in the neural network literature[39]. In the context of SA–PNN, we construct the network using the patterns of the train set. The optimization algorithm, subsequently, determines the matrix, Σ, of spread parameters by minimizing the leave–one–out misclassification error on the training set, while the misclassification error on the validation set is monitored at each iteration of the algorithm. In order to avoid over–training the network, when an increase on the validation error for μ subsequent iterations is observed, the algorithm stops and returns as Σ, the matrix that yielded the lowest error on the validation set. We denote this technique by TVT.

3.3. *Presentation of the results and statistical analysis*

In Tables 2–5 the mean (Mean), median (Median), standard deviation (St.Dev.), minimum (Min.), maximum (Max.), and 95% confidence interval (CI low, CI up) of the success rates (in percentage terms) for the E.coli, Yeast, Breast Cancer and Diabetes data sets, are reported for each sampling technique and for all the implemented methods. The methods employed are: the classical homoscedastic PNN, homoscedastic PNN optimized by golden section search (GSS), homoscedastic PNN

Table 3. Test set classification accuracy percentage for Yeast data set.

Sampling	Model	Mean	Median	St.Dev.	Min.	Max.	CI low	CI up
SRS	PNN	57.79	57.37	1.55	**55.48**	**61.16**	57.06	58.51
	GSS	56.29	57.24	5.93	31.80	59.68	53.51	59.06
	PCU	57.73	57.51	1.59	54.13	60.89	56.99	58.48
	PIU	57.73	57.44	1.60	54.13	60.89	56.98	58.48
	PCM	**58.01**	**58.80**	1.71	53.99	60.08	57.21	58.81
	PIM	57.66	57.98	1.73	52.64	60.22	56.85	58.47
	FNN 8–16–10	55.99	56.22	1.89	52.10	59.40	55.10	56.87
	SVM	39.30	39.58	**1.03**	37.21	40.60	38.81	39.78
10-CV	PNN	58.37	58.25	0.57	57.13	59.58	58.10	58.63
	GSS	58.87	58.73	0.43	**58.28**	59.76	58.67	59.08
	PCU	58.84	58.82	0.39	58.08	59.64	58.66	59.02
	PIU	58.87	58.86	0.33	58.21	59.51	58.72	59.03
	PCM	58.86	**58.90**	0.69	57.61	60.04	58.53	59.18
	PIM	**58.92**	58.89	0.77	57.41	**60.05**	58.56	59.28
	FNN 8–16–10	55.55	55.55	0.68	54.32	56.81	55.24	55.87
	SVM	42.97	42.99	**0.32**	42.38	43.80	42.82	43.12
TVT	PNN	57.38	57.43	**1.92**	54.32	61.08	56.48	58.28
	GSS	55.69	**57.70**	7.30	33.78	61.62	52.27	59.11
	PCU	57.70	57.43	2.10	**54.32**	61.62	56.72	58.69
	PIU	57.76	57.57	2.25	54.05	**61.89**	56.71	58.81
	PCM	**57.89**	**57.70**	2.62	52.70	**61.89**	56.67	59.12
	PIM	57.50	56.89	2.38	53.24	61.62	56.39	58.61
	FNN 8–16–10	55.46	56.08	2.33	48.65	58.11	54.37	56.55
	SVM	40.03	39.59	3.40	35.14	48.65	38.44	41.62

optimized by constriction coefficient PSO (PCU), homoscedastic PNN optimized by inertia weight PSO (PIU), heteroscedastic PNN optimized by constriction coefficient PSO (PCM), heteroscedastic PNN optimized by inertia weight PSO (PIM), feedforward neural network (FNN) and C–support vector machines (SVM). Note that by classical homoscedastic PNN we refer to a PNN whose single smoothing parameter, σ^2, was the one that yielded the best training set classification accuracy out of 250 points equally distributed in the range $[0, 5]$. The choice of 250 points was made to render comparable the outcomes of this approach with those of PSO (5 particles for 50 iterations).

For a better visual presentation of the obtained results we also provide box–and–whiskers plots (boxplots) in Figs. 2–5. The boxplot is a diagram for conveying location and variation information about a certain variable. The median of attained classification accuracies is displayed as a horizontal line and a box is drawn between the first and third quartile of observations. Then, the minimum and maximum values that lie into the range with center the median and length 1.5 multiplied by the interquartile range are connected to the box. If a value lies outside this range, then it is considered as an outlier and displayed as a dot.

As previously mentioned, a powerful feature of PNNs is that Bayesian confidence intervals for the class membership probabilities can be estimated. This infor-

Table 4. Test set classification accuracy percentage for Cancer data set.

Sampling	Model	Mean	Median	St.Dev.	Min.	Max.	CI low	CI up
SRS	PNN	96.22	96.28	**0.70**	94.56	97.71	95.89	96.55
	GSS	96.15	96.13	0.79	94.84	97.71	95.78	96.52
	PCU	96.20	96.42	0.82	94.56	97.42	95.82	96.58
	PIU	96.20	96.13	0.78	94.56	97.42	95.84	96.57
	PCM	94.71	94.84	1.44	90.26	96.56	94.04	95.39
	PIM	95.11	95.27	1.23	92.26	97.13	94.54	95.69
	FNN 9–4–2–2	95.40	95.42	0.98	93.41	97.42	94.94	95.86
	SVM	**96.73**	**96.71**	0.72	**95.70**	**98.28**	96.40	97.07
10-CV	PNN	95.83	95.85	0.32	95.13	96.71	95.68	95.98
	GSS	95.92	95.86	0.27	95.42	96.42	95.80	96.05
	PCU	95.83	95.78	0.26	95.42	96.28	95.70	95.95
	PIU	95.85	95.85	0.23	95.42	96.14	95.74	95.96
	PCM	95.13	95.14	0.46	94.42	96.28	94.91	95.34
	PIM	94.96	94.99	0.61	94.13	96.27	94.67	95.24
	FNN 9–4–2–2	95.93	96.14	0.91	92.56	**97.14**	95.50	96.36
	SVM	**96.79**	**96.85**	**0.18**	**96.42**	**97.14**	96.71	96.88
TVT	PNN	95.09	95.40	1.68	91.95	97.70	94.30	95.87
	GSS	95.32	95.69	1.56	91.95	97.13	94.58	96.05
	PCU	95.49	95.98	1.47	92.53	97.13	94.80	96.18
	PIU	95.49	95.69	1.41	93.10	97.13	94.83	96.15
	PCM	95.72	95.98	1.36	93.10	97.13	95.08	96.36
	PIM	95.55	95.40	1.34	**93.68**	98.28	94.92	96.17
	FNN 9–4–2-2	95.37	95.11	2.17	90.23	**98.85**	94.36	96.39
	SVM	**96.21**	**96.55**	**1.32**	**93.68**	**98.85**	95.59	96.83

mation can be used to assess the degree of confidence with which a classification is performed and to reject classifications with very low posterior probability. In the current study, instead of exploiting this feature, we use the maximum estimated posterior probability to classify patterns of the test set, using Eq. (3). We adopt this approach to render the results obtained by PNNs directly comparable to those obtained by FNNs and SVMs.

Regarding the stratified sampling technique (SRS), the size of the random sample for each stratum, in all experiments, was set to 50% of the size of the corresponding class. In the TVT sampling technique, μ was set to 5. The highest maximum values in all data sets were obtained using TVT.

The level of significance in all statistical tests is set to 0.05. To choose which statistical tests are applicable in our experimental design, a Kolmogorov–Smirnov test for the normality assumption[40] was applied to each run. The test showed that all the runs were normally distributed indicating that parametric tests should be used for the statistical comparison of the results. For the comparisons of the different methods applied to each dataset, the "corrected resampled t–test" is used[41,42]. This test takes into account the underestimation of the variance since the training and test samples that were created by resampling overlap and, hence, they are not independent. The same holds for the k–fold cross–validation comparisons where

Table 5. Test set classification accuracy percentage for Diabetes data set.

Sampling	Model	Mean	Median	St.Dev.	Min.	Max.	CI low	CI up
SRS	PNN	73.19	73.05	1.74	69.01	76.82	72.38	74.00
	GSS	71.76	72.79	3.95	65.10	78.65	69.91	73.61
	PCU	73.59	73.31	2.08	68.75	78.65	72.62	74.57
	PIU	73.40	73.18	2.01	68.75	78.65	72.46	74.34
	PCM	73.96	74.09	1.87	70.83	77.08	73.08	74.83
	PIM	74.05	73.83	2.10	70.05	77.86	73.07	75.03
	FNN 8–2–2–2	**76.58**	**76.56**	1.55	**73.44**	**78.91**	75.85	77.30
	SVM	73.45	73.44	**1.22**	70.57	75.52	72.88	74.02
10-CV	PNN	74.07	73.97	0.75	73.04	75.65	73.72	74.42
	GSS	73.26	73.44	1.11	71.09	75.14	72.75	73.78
	PCU	74.13	74.16	0.82	72.64	75.78	73.74	74.51
	PIU	74.12	74.09	0.77	72.79	75.52	73.76	74.48
	PCM	75.29	75.53	0.72	73.95	76.05	74.95	75.63
	PIM	75.11	75.07	0.87	73.45	76.69	74.70	75.51
	FNN 8–2–2–2	76.38	**77.02**	1.51	71.89	**77.87**	75.68	77.09
	SVM	**76.42**	76.44	**0.39**	**75.51**	76.95	76.24	76.60
TVT	PNN	72.67	73.56	3.60	62.83	79.06	70.98	74.36
	GSS	72.46	73.82	3.71	62.83	76.96	70.73	74.20
	PCU	72.93	73.04	**1.76**	69.63	75.92	72.11	73.75
	PIU	72.72	73.04	1.92	69.11	75.92	71.82	73.62
	PCM	74.32	74.35	2.89	69.11	**79.58**	72.97	75.67
	PIM	74.21	74.35	3.19	69.11	79.06	72.72	75.71
	FNN 8–2–2-2	**76.83**	**76.96**	1.93	**73.30**	**79.58**	75.93	77.74
	SVM	73.35	74.08	3.83	61.78	78.53	71.56	75.14

the "corrected repeated k–fold CV t–test" is applied. To compare the different sampling techniques on each dataset, a Kruskal–Wallis median test[40] is performed. This test is used due to the heterogeneity in the variances of each sampling technique (a Bartlett test was conducted), and it is equivalent to the Analysis of Variance (ANOVA) test.

If there is no statistically significant difference in pairwise statistical comparisons between methods, then only the minimum p–value is reported because this p–value shows the comparison of the two methods that have the greatest non significant difference. Similarly, the maximum p–value is reported when there are statistically significant differences between methods, as it reports the pair of methods that have the smallest statistically significant difference.

3.4. *Comparisons of methods*

E.coli Dataset: As reported in Table 2 for the E.coli dataset, the classification accuracy of SVMs is substantially lower than that of PNNs and FNNs, irrespective of the sampling technique employed. This can be immediately derived also from Fig. 2. For the SRS sampling technique, utilizing pairwise t–tests, we compare the mean performance of SVMs with that of all other methods and observe that the

differences in mean accuracy are statistically significant. The maximum p–value for this series of comparisons is 0.038 and corresponds to the comparison of mean classification accuracy of SVM and PCM. The pairwise t–tests performed on all combinations of the other methods (excluding SVMs) did not yield a statistically significant difference in their mean performance.

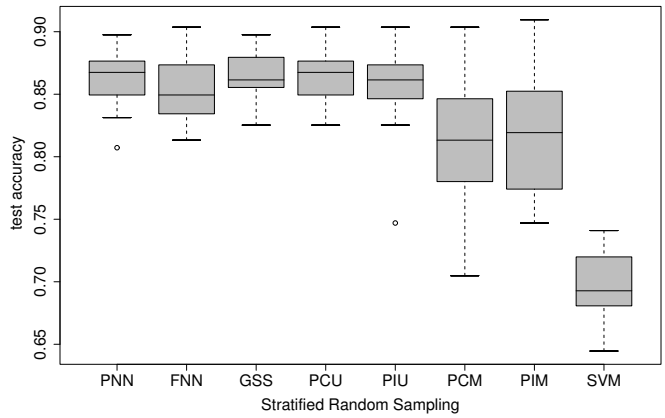
For the CV sampling technique, the worst performance is achieved by SVMs with maximum p–value in all pairwise t–tests less than 0.001. Performing pairwise t–tests among all the other considered methods, three groups of methods with respect to mean classification accuracy were identified. The first and best group is comprised by all the homoscedastic PNNs and the FNN; the second group contains heteroscedastic PNNs, while the third consists of the SVM.

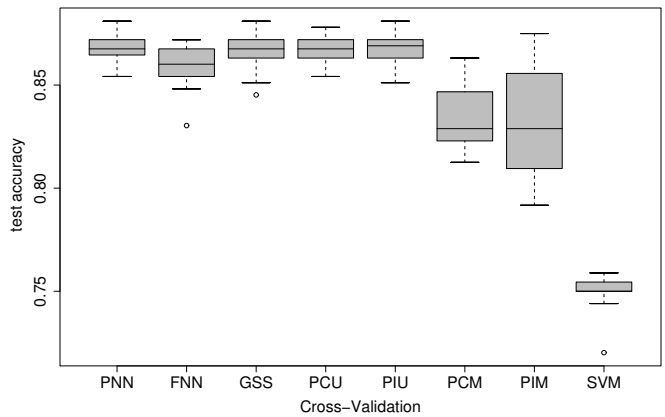
Using the TVT sampling technique similar results to SRS were obtained. Again, the only method that yields a statistically significant poor performance was the SVM. The maximum p–value was 0.032 and corresponds to the comparison of SVM with PIM. At this point we would like to point out that the maximum performance attained for the E.coli data set and TVT sampling is 92.77%, and it compares favorably with the results obtained for this dataset through a data selection method proposed in Ref. 43.

Yeast Dataset: On the Yeast dataset the performance of the SVMs is relatively low, as reported in Table 3 and depicted in Fig. 3. It is also worth noting that the relative performance of the heteroscedastic PNNs was much better compared to that on the E.coli dataset.

For the SRS technique, no statistically significant differences between the mean accuracies of the methods were identified, with the only exception being the SVM whose performance was found statistically inferior to all other methods (maximum p–value 0.010 from the comparison with GSS). For the CV technique three groups were again identified. These groups were (in descending order of mean performance) the PNNs, the FNN and the SVM. The maximum p–values from all pairwise t–tests between all the methods of one group and those of the immediately inferior group (in terms of mean accuracy) are PIM–FNN with p–value 0.001 and FNN–SVM with p–value less than 0.001. For the TVT technique, the only method that has a statistically different mean performance from all the others is SVM (SVM–GSS with maximum p–value 0.013).

Breast Cancer: From the results reported in Table 4 and Fig. 4 it is evident that all methods performed quite well and especially SVMs. An important difference from the previous datasets was that there are only two classes in the Breast Cancer dataset, while in the E.coli and Yeast datasets there are eight and ten classes, respectively. For the SRS technique, no method exhibited a statistically superior performance (minimum p–value 0.199 corresponding to the t–test of FNN–SVM). For the CV technique, the SVM exhibited a statistically superior performance over the other methods (maximum p–value 0.011 from the comparison between PNN





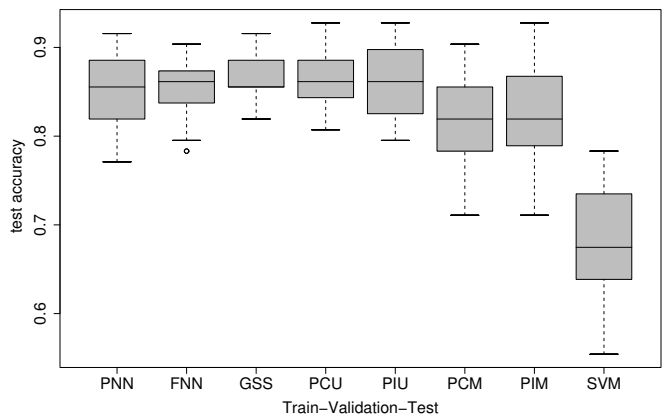
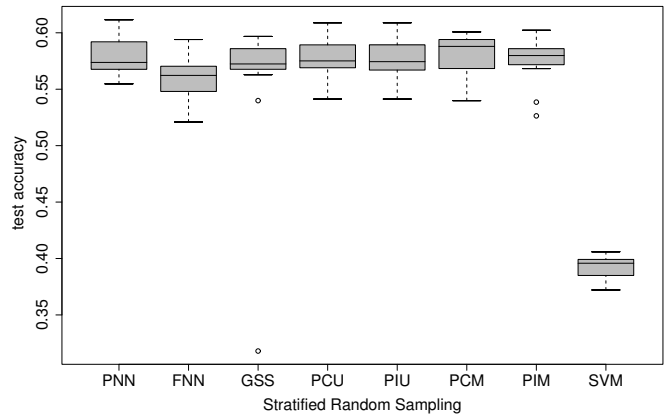
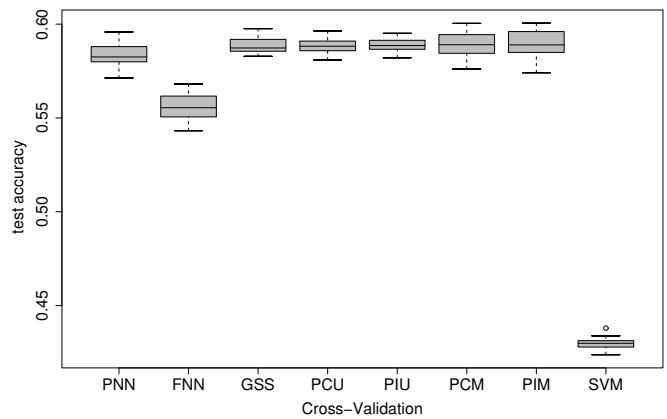


Fig. 2. Boxplots of classification accuracies on test set for E.coli dataset





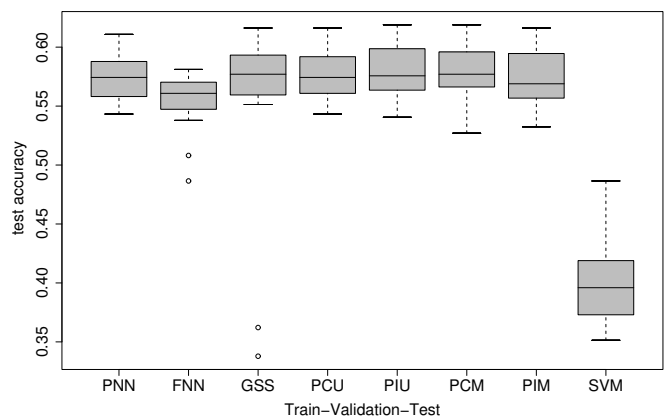
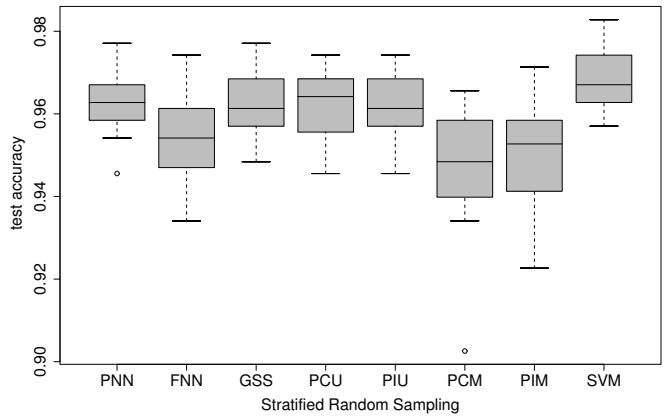
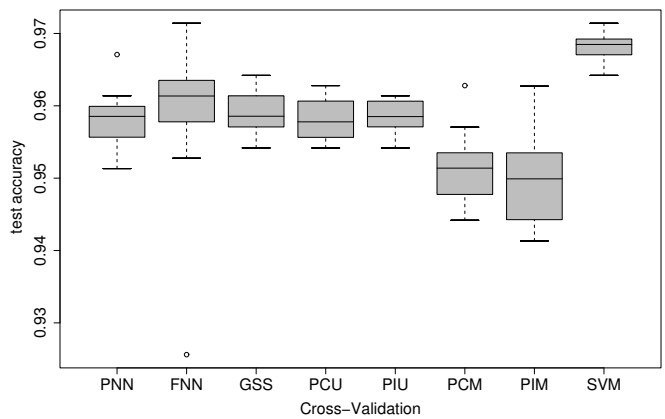


Fig. 3. Boxplots of classification accuracies on test set for Yeast dataset





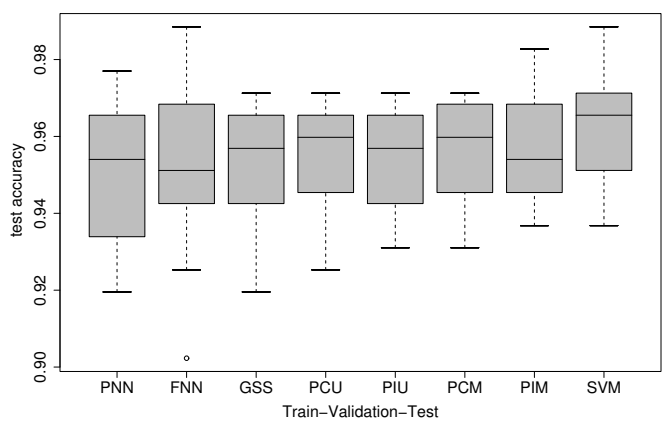


Fig. 4. Boxplots of classification accuracies on test set for Breast Cancer dataset

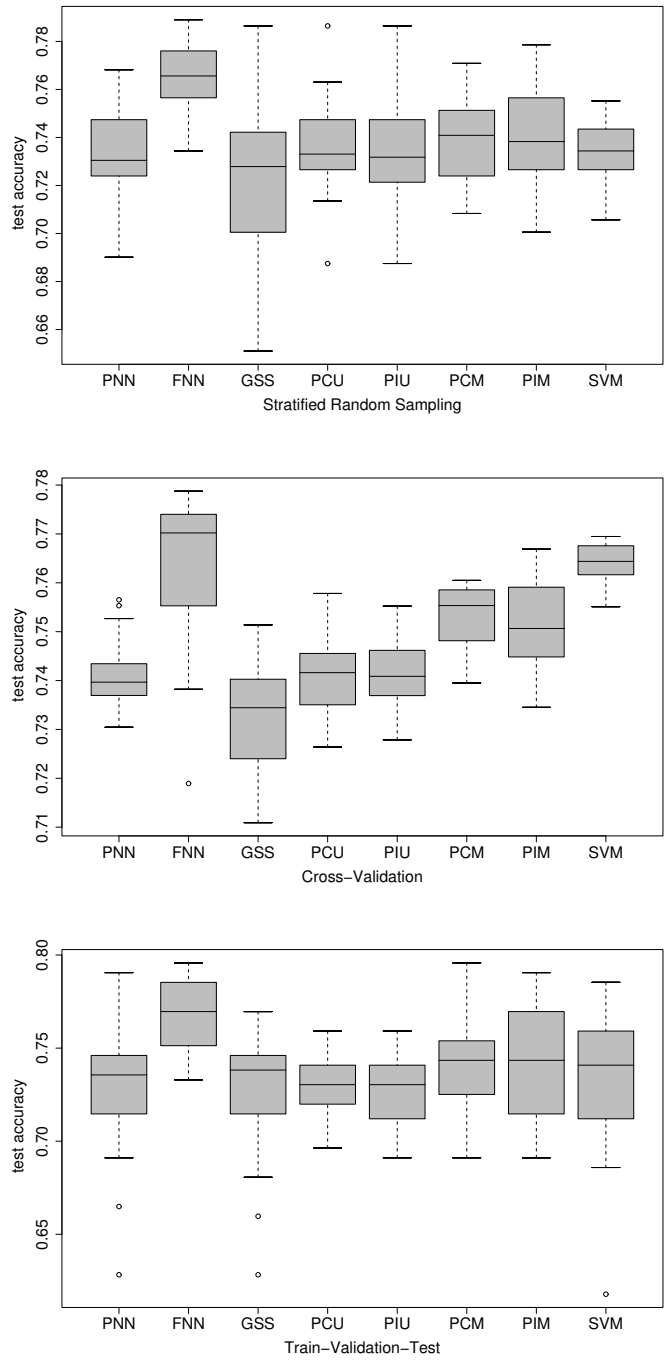


Fig. 5. Boxplots of classification accuracies on test set for Diabetes dataset

and SVM). Finally, for the TVT technique, there was no statistically significant difference between the methods (minimum p–value equal to 0.104 for PCU–SVM).

Diabetes Dataset: As it can be observed from Table 5 and Fig. 5, in most cases, the best performance was achieved by the FNN, although there were no significant differences. For the SRS technique, no statistically significant differences were identified (minimum p–value = 0.124 for FNN–SVM). For CV, we have statistically significant differences between the homoscedastic PNNs and the FNN, as well as, the SVM (maximum p–value = 0.022 for FNN–GSS). For the TVT there were a statistically significant difference between the performances of FNN and both PCU and PIU (p–value 0.011 for PCU and p–value 0.015 for PIU).

3.5. *Comparison of the sampling techniques*

In order to check if there is homogeneity on the variances on the three sampling techniques on each dataset, a Bartlett test[40] was conducted taking into account the mean accuracy of the methods on each sample. In all datasets, there was heterogeneity in the variances of the sampling techniques, so an ANOVA test of equality of means is not applicable (all p–values where less than 0.001). Consequently, the non–parametric Kruskal–Wallis test of equality of medians was performed.

E.coli Dataset: For the E.coli dataset, comparing the three sampling techniques we see that there is statistically significant difference among them (p–value 0.011). Conducting pairwise comparisons, there is a statistically significant superiority of CV compared with SRS (p–value less than 0.001) as well as with TVT (p–value less than 0.001). However, SRS and TVT yield the same performance (p–value equal to 0.070).

Yeast Dataset: The same findings also hold for the yeast dataset, where there is difference between all the sampling techniques (p–value less than 0.001). Similarly, CV is superior to both SRS (p–value less than 0.001) and TVT (p–value equal to 0.007), although there is no statistically significant difference between SRS and TVT (p–value equal to 0.914).

Breast Cancer Dataset: On the contrary, for the breast cancer data set, there is no statistically significant difference among all three sampling techniques (p–value equal to 0.997).

Diabetes Dataset: Similarly to E.coli and Yeast, for the diabetes dataset there is a statistically significant difference among all sampling techniques (p–value equal to 0.004). The CV technique differs from SRS and TVT with p–value equal to 0.002 and 0.008, respectively. Between SRS and TVT, there is no statistically significant difference (p–value equal to 0.071).

Summarizing the findings, we see that the CV technique yields superior performance in three out of the four test problems considered, compared to SRS and TVT.

It should be noted, however, that the CV technique exploits 90% of the available data for training in each sample, compared to 50% used by the the SRS and TVT. Also, the computational cost is more than 10 times greater, since 10 PNNs are constructed for each sample. The training sets are also substantially larger compared to the ones from SRS and TVT.

4. Conclusions

PNNs are recognized as a powerful classification tool in numerous Bioinformatics and Medical applications[7,8,9,10]. In contrast to other established classification methodologies, like FNNs and the k-nearest neighbor method, PNNs are capable of providing a measure of confidence for the classification, which is highly desirable from the point of view of the practitioner. In this paper, a self–adaptive model for probabilistic neural networks (PNN) was proposed. The proposed approach incorporates an optimization algorithm to find appropriate spread parameters for the PNN with respect to the resulting classification accuracy. The effectiveness of the proposed model is assessed on the E.coli and Yeast data sets from the field of Bioinformatics, as well as, on the Cancer and Diabetes datasets from the field of medicine, with promising results.

From the obtained experimental results, it appears that the selection of an appropriate spread parameter for the homoscedastic PNN significantly enhances its performance and renders its classification accuracy comparable to the alternative classification methods considered, namely feedforward neural networks and support vector machines. In particular, support vector machines produced a relatively worse performance on the two multi–class classification tasks. On the contrary, their performance was marginally superior to that of the other classifiers on the binary classification task of Breast Cancer prediction. On the Diabetes binary classification task, the performance of both SVMs and PNNs was satisfactory, but the best performing method when using cross–validation proved to be the FNN.

PNNs were never ranked as the worst among the three considered classification methods. For the E.coli and the Yeast datasets, their performance was particularly good. Statistically significant superiority in the performance of PNNs is identified using the cross–validation sampling technique for these two datasets. An important finding is that heteroscedastic PNNs, which are considered to be more powerful classifiers than their homoscedastic counterparts, do not exhibit a statistically superior performance compared to homoscedastic PNNs. On the E.coli dataset, using the cross–validation sampling technique the homoscedastic PNNs exhibit a statistically significant superior performance compared to heteroscedastic PNNs. This finding can be attributed to the phenomenon of overfitting (over–training) which is frequently encountered in neural network applications. In these cases, the optimization algorithm was capable of detecting a point that yielded a particularly low classification error on the training set, but which did not produce a satisfactory performance on the test set.

The method employed to obtain a suitable spread parameter for the homoscedastic PNNs appears to bear a small impact on the resulting classification accuracy, as can be derived from the comparison between PSO and the golden section method.

The sampling method that yielded statistically significant superior performance was λ–fold cross–validation. This finding, however, can be attributed to the larger training set size that is employed by this method, compared to stratified random sampling and train–validation–test set partitioning. In the context of PNNs, a larger training set size imposes larger computational and storage requirements for the construction and training of PNNs, and yields lower response times for the classification of new pattern vectors.

Future work will include the application of different evolutionary computation algorithms, like Differential Evolution and Evolution Strategies, on the problem of determining the optimal spread parameters of PNNs. Criteria for avoiding overfitting in the case of the multidimensional optimization problem will also be investigated. Furthermore, a generalization of the proposed self–adaptive scheme to determine all the elements and not only the diagonal entries of the matrix of spread parameters will be considered.

Acknowledgment

We would like to thank the editors and the anonymous reviewers for their valuable comments and suggestions. We thank European Social Fund (ESF), Operational Program for Educational and Vocational Training II (EPEAEK II) and particularly the Programs IRAKLEITOS and PYTHAGORAS, for funding the above work.

Appendix A: Feedforward Neural Networks and the Rprop Algorithm

FNNs have been widely used in pattern recognition tasks and they are considered to be very powerful classifiers compared to classical algorithms, such as the nearest neighbor method. More specifically, FNNs have proved capable of finding a good classifier based on a limited, and generally small, number of training examples. This capability, also referred to as *generalization*, is of particular interest from a pattern recognition point of view, since a large set of parameters is estimated using a relatively small data set.

In FNNs, the inputs form an input layer, while the output neurons form the output layer. All other neurons are assigned to a number of hidden layers. Each neuron in a layer is fully connected to all neurons of the next layer. This structure renders it possible to describe FNNs with a series of integers that represent the number of neurons at each layer. For example a network with a topology 4-5-5-1 is a network with 4 inputs at the input layer, two hidden layers with 5 neurons each, and an output layer with a single neuron.

The computational power of neural networks derives from their capability to adapt to specific problems. Two critical parameters for the successful application

of FNNs are the appropriate selection of network architecture and the training algorithm. The problem of identifying the optimal network architecture for a specific task remains up to date an open problem. The efficient supervised training of FNNs is a subject of considerable ongoing research and numerous algorithms have been proposed to this end. Supervised training amounts to the global minimization of the network error function. The rapid computation of a set of weights that minimizes this error is a difficult task since, in general, the number of network weights is large and the resulting error function generates a complex surface in the weight space. Next, a brief exposition of the Resilient backpropagation (Rprop) algorithm is provided.

Rprop is a local adaptive learning scheme performing supervised training of FNNs. To update each weight of the network, Rprop exploits information concerning the sign of the partial derivative of the error function. The size of the weight change, Δw_{ij}, is determined by a weight-specific update-value, $\Delta_{ij}^{(t)}$:

$$\Delta w_{ij}^{(t)} = \begin{cases} -\Delta_{ij}^{(t)}, & \text{if } \dfrac{\partial E^{(t)}}{\partial w_{ij}} > 0, \\ +\Delta_{ij}^{(t)}, & \text{if } \dfrac{\partial E^{(t)}}{\partial w_{ij}} < 0, \\ 0, & \text{otherwise}, \end{cases} \tag{8}$$

where $\partial E^{(t)}/\partial w_{ij}$ denotes the summed gradient information over all patterns of the training set. The second step of the Rprop algorithm is to determine the new update–values,

$$\Delta_{ij}^{(t)} = \begin{cases} \eta^{+}\Delta_{ij}^{(t-1)}, & \text{if } \dfrac{\partial E^{(t-1)}}{\partial w_{ij}} \dfrac{\partial E^{(t)}}{\partial w_{ij}} > 0, \\ \eta^{-}\Delta_{ij}^{(t-1)}, & \text{if } \dfrac{\partial E^{(t-1)}}{\partial w_{ij}} \dfrac{\partial E^{(t)}}{\partial w_{ij}} < 0, \\ \Delta_{ij}^{(t-1)}, & \text{otherwise}, \end{cases} \tag{9}$$

where, $0 < \eta^{-} < 1 < \eta^{+}$, i.e., each time the partial derivative with respect to w_{ij} changes its sign, which is an indication that the last update was too big and the algorithm has jumped over a local minimum, the update-value $\Delta_{ij}^{(t)}$ is decreased by η^{-}. If the derivative retains its sign, the update–value is slightly increased to accelerate convergence in shallow regions.

Regarding the five parameters of Rprop that were used in our experiments,

(1) the increase factor was set to $\eta^{+} = 1.2$;
(2) the decrease factor was set to $\eta^{-} = 0.5$;
(3) the initial update–value is set to $\Delta_0 = 0.07$;
(4) the maximum step, which is used in order to prevent the weights from becoming too large, was $\Delta_{\max} = 50$;

(5) the minimum step, which is used to avoid too small weight changes, was $\Delta_{\min} = 10^{-6}$.

The error goal for the Rprop was set to 10^{-2}, and it was applied for 500 epochs, as proposed in Ref. 44.

Appendix B: Support Vector Machines

Support Vector Machines is a classification method based on separating hyperplane classifiers. This method is used particularly in binary classification tasks with promising results, but also in multiclass classification[45,46]. We state briefly the motivation of this method for binary tasks.

Each instance from the training set consists of a pair, (x_i, y_i), where $x_i \in \mathbb{R}^n$ is the feature vector of the ith instance and y_i its class label, $y_i \in \{-1, 1\}$, $i = 1, 2, \ldots, N_t$, where N_t is the size of the training set. SVMs require the solution of the optimization problem,

$$\min_{w,b,xi} = \frac{1}{2} w^T w + C \sum_{i=1}^{N_t} \xi_i \,, \tag{10}$$

subject to the constraints,

$$\xi_i \geqslant 0, \qquad i = 1, 2, \ldots, N_t \tag{11}$$

$$y_i \left(w^\top \varphi(x_i) + b \right) \geqslant 1 - \xi_i, \qquad i = 1, 2, \ldots, n, \tag{12}$$

where $C > 0$ is a regularization constant. This particular version of SVM is called C–SVM.

The function φ in Eq. (12) maps the training vectors x_i into a higher dimensional space. Then, SVMs try to find a linear separating hyperplane with the maximal margin in this higher–dimensional space. The kernel function of the SVM is given by $K(x_i, x_j) = \varphi(x_i)^\top \varphi(x_j)$. In our study, we used as kernel function the Radial Basis Function,

$$K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2) \,.$$

The parameter γ is called the *kernel parameter* and it was set equal to $1/n$, where n is the number of features.

References

1. P. Baldi and S. Brunak, *Bioinformatics: The Machine Learning Approach* (MIT Press, 2001).
2. A. C. Tan and D. Gilbert, An empirical comparison of supervised machine learning techniques in bioinformatics, in *Proc. 1st Asia Pacific Bioinformatics Conference (APBC)*, (2003) pp. 219–222.
3. K. E. Parsopoulos, E. I. Papageorgiou, P. P. Groumpos and M. N. Vrahatis, Evolutionary computation techniques for optimizing fuzzy cognitive maps in radiation therapy systems, *Lecture Notes in Computer Science* **3102**(2004) 402–413.

4. G. D. Magoulas, V. P. Plagianakos and M. N. Vrahatis, Neural network–based colonoscopic diagnosis using on–line learning and differential evolution, *Applied Soft Computing* **4**(2004) 369–379.
5. D. F. Specht, Probabilistic neural networks, *Neural Networks* **2**(1990) 109–118.
6. M. Berthold and J. Diamond, Constructive training of probabilistic neural networks, *Neurocomputing* **19**(1998) 167–183.
7. C. J. Huang, A performance analysis of cancer classification using feature extraction and probabilistic neural networks, in *Proc. 7th Conference on Artificial Intelligence and Applications*, (2002), pp. 374–378.
8. E. Holmes, J. K. Nicholson and G. Tranter, Metabonomic characterization of genetic variations in toxicological and metabolic responses using probabilistic neural networks, *Chem. Res. Toxicol.* **14**(2001) 182–191.
9. J. Guo, Y. Lin and Z. Sun, A novel method for protein subcellular localization based on boosting and probabilistic neural network, in *Proc. 2nd Asia-Pacific Bioinformatics Conference*, (Dunedin, New Zealand, 2004), pp. 20–27.
10. Y. Wang, T. Adali, S. Kung and Z. Szabo, Quantification and segmentation of brain tissues from mr images: A probabilistic neural network approach, *IEEE Trans. Image Processing* **7**(1998) 1165–1181.
11. V. L. Georgiou, N. G. Pavlidis, K. E. Parsopoulos, Ph. D. Alevizos and M. N. Vrahatis, Optimizing the performance of probabilistic neural networks in a bioinformatics task, in *Proc. EUNITE 2004 Conference*, (Aachen, Germany, 2004), pp. 34–40.
12. J. Kennedy and R. C. Eberhart, *Swarm Intelligence* (Morgan Kaufmann Publishers, 2001).
13. M. A. Abido, Optimal design of power system stabilizers using particle swarm optimization, *IEEE Trans. Energy Conversion* **17**(2002) 406–413.
14. A. R. Cockshott and B. E. Hartman, Improving the fermentation medium for *Echinocandin B* production part II: Particle swarm optimization, *Process Biochemistry* **36**(2001) 661–669.
15. C. O. Ourique, E. C. Biscaia and J. Carlos Pinto, The use of particle swarm optimization for dynamical analysis in chemical processes, *Computers and Chemical Engineering* **26**(2002) 1783–1793.
16. K. E. Parsopoulos and M. N. Vrahatis, Recent approaches to global optimization problems through particle swarm optimization, *Natural Computing* **1**(2002) 235–306.
17. K. E. Parsopoulos and M. N. Vrahatis, On the computation of all global minimizers through particle swarm optimization, *IEEE Trans. Evol. Comp.* **8**(2004) 211–224.
18. P. Horton and K. Nakai, A probabilistic classification system for predicting the cellular localization sites of proteins, in *Proc. 4th Int. Conf. ISMB*, (1996), pp. 109–115.
19. L. Prechelt, Proben1: A set of neural network benchmark problems and benchmarking rules, Technical Report 21/94, (Fakultät für Informatik, Universität Karlsruhe, 1994).
20. O. L. Mangasarian and W. H. Wolberg, Cancer diagnosis via linear programming, *SIAM News* **23**(1990) 1–18.
21. J. W. Smith, J. E. Everhart, W. C. Dickson, W. C. Knowler and R. S. Johannes, Using the adap learning algorithm to forecast the onset of diabetes mellitus, in *Proc. Symp. Comp. Appl. & Med. Care*, (1988), pp. 261–265.
22. D. F. Specht and H. Romsdahl, Experience with adaptive probabilistic neural network and adaptive general regression neural network, in *Proc. IEEE Int. Conf. Neural Networks*, (1994), pp. 1203–1208.
23. R. Romero, D. Touretzky and R. Thibadeau, Optical chinese character recognition using probabilistic neural network, *Pattern Recognition* **8**(1997) 1279–1292.
24. P. P. Raghu and B. Yegnanarayana, Supervised texture classification using a prob-

abilistic neural network and constraint satisfaction model, *IEEE Trans. Neural Networks* **9**(1998) 516–522.
25. E. Parzen, On the estimation of a probability density function and mode, *Annals of Mathematical Statistics* **3**(1962) 1065–1076.
26. R. L. Streit and T. E. Luginbuhl, Maximum likelihood training of probabilistic neural networks, *IEEE Trans. Neural Networks* **5**(1994) 764–783.
27. H.-P. Schwefel, *Evolution and Optimum Seeking* (Wiley, New York, 1995).
28. R. C. Eberhart and J. Kennedy, A new optimizer using particle swarm theory, in *Proc. 6th Symposium on Micro Machine and Human Science*, (Piscataway, NJ, 1995), pp. 39–43.
29. J. Kennedy and R. C. Eberhart, Particle swarm optimization, in *Proc. IEEE Int. Conf. Neural Networks*, (Piscataway, NJ, 1995), pp. 1942–1948.
30. M. M. Millonas, Swarms, phase transitions, and collective intelligence, in *Computational Intelligence: A Dynamic System Perspective*, eds. M. Palaniswami, Y. Attikiouzel, R. Marks, D. Fogel, and T. Fukuda, (IEEE Press, 1994), pp. 137–151.
31. Y. Shi and R. C. Eberhart, A modified particle swarm optimizer, in *Proc. IEEE Conf. Evol. Comp.*, (Anchorage, AK, 1998), pp. 69–73.
32. M. Clerc and J. Kennedy, The particle swarm–explosion, stability, and convergence in a multidimensional complex space, *IEEE Trans. Evol. Comput.* **6**(2002) 58–73.
33. I. C. Trelea, The particle swarm optimization algorithm: Convergence analysis and parameter selection, *Information Processing Letters* **85**(2003) 317–325.
34. Y. Shi and R. C. Eberhart, Parameter selection in particle swarm optimization, in *Evolutionary Programming*, eds. V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben (Springer, 1998), pp. 591–600.
35. R. K. Orr, Use of a probabilistic neural network to estimate the risk of mortality after cardiac surgery, *Medical Decision Making* **17**(1997) 178–185.
36. G. D. Magoulas, Building diverse neural ensembles for bioinformatics applications, in *Sp. sess. proc. EUNITE 2004 Symp.*, (Aachen, Germany, 2004), pp. 5–14.
37. M. Riedmiller and H. Braun, A direct adaptive method for faster backpropagation learning: The RPROP algorithm, in *Proc. IEEE Int. Conf. Neural Networks*, (1993), pp. 586–591.
38. C. L. Blake and C. J. Merz, UCI repository of machine learning databases, 1998.
39. S. Haykin, *Neural Networks: A Comprehensive Foundation* (New York, Macmillan College Publishing Company, 1999).
40. G. K. Kanji, *100 Statistical Tests* (Sage Publications, 1999).
41. R. R. Bouckaert and E. Frank, Evaluating the replicability of significance tests for comparing learning algorithms, in *Proc. Pacific-Asia Conference on Knowledge Discovery and Data Mining, LNAI 3056*, eds. H. Dai, R. Srikant and C. Zhang (Sydney, Australia, 2004), pp. 3–12.
42. T. G. Dietterich, Approximate statistical test for comparing supervised classification learning algorithms, *Neural Computation* **10**(1998) 1895–1923.
43. B. Bolat and T. Yildirim, A data selection method for probabilistic neural networks, in *Proc. XII Int. Turk. Symp. Artif. Intel. Neural Networks*, (2003).
44. C. Igel and M. Husken, Empirical evaluation of the improved Rprop learning algorithms, *Neurocomputing* **50**(2003) 105–123.
45. B. Boser, I. Guyon and V. Vapnik, A training algorithm for optimal margin classifiers, in *Proc. 5th Annual Workshop on Computational Learning Theory*, (1992).
46. C. Cortes and V. Vapnik, Support vector networks, *Machine Learning* **20**(1995) 273–297.