Expeditive Extensions of Evolutionary Bayesian Probabilistic Neural Networks

Vasileios L. Georgiou[1], Sonia Malefaki[2], Konstantinos E. Parsopoulos[1], Philipos D. Alevizos[1], and Michael N. Vrahatis[1]

[1] Department of Mathematics, University of Patras, Patras, Greece
{vlg,kostasp,philipos,vrahatis}@math.upatras.gr
[2] Department of Statistics and Insurance Science, University of Piraeus, Piraeus, Greece
smalefak@unipi.gr

Abstract. Probabilistic Neural Networks (PNNs) constitute a promising methodology for classification and prediction tasks. Their performance depends heavily on several factors, such as their spread parameters, kernels, and prior probabilities. Recently, Evolutionary Bayesian PNNs were proposed to address this problem by incorporating Bayesian models for estimation of spread parameters, as well as Particle Swarm Optimization (PSO) as a means to select prior probabilities. We further extend this class of models by introducing new features, such as the Epanechnikov kernels as an alternative to the Gaussian ones, and PSO for parameter configuration of the Bayesian model. Experimental results of five extended models on widely used benchmark problems suggest that the proposed approaches are significantly faster than the established ones, while exhibiting competitive classification accuracy.

1 Introduction

Classification models exhibit a rapid development in the past few years, due to their wide applicability in modern scientific and engineering applications. Probabilistic Neural Networks (PNNs) [1] is a widely used classification methodology, which has been used in several applications in bioinformatics [2, 3, 4, 5], as well as in different scientific fields [6, 7] with promising results. PNNs constitute a variant of the well–known Discriminant Analysis [8], presented in the framework of artificial neural networks. Their main task is the classification of unknown feature vectors into predefined classes [1], where the Probability Density Function (PDF) of each class is estimated by kernel functions. For this purpose, the Gaussian kernel function is usually employed.

The type of kernels and their spread parameters, as well as the prior probability of each class affect the performance of PNNs, significantly [9, 10]. Evolutionary Bayesian PNNs (EBPNNs) [11] were proposed as variants of the standard PNNs, where the spread parameters are estimated by Bayesian models, while the prior probabilities are determined by the Particle Swarm Optimization (PSO) algorithm. However, the employed Bayesian models included also several parameters, configured through a time consuming exhaustive search procedure.

The present work aims at extending the EBPNN model in order to reduce the required execution time. For this purpose, new features are introduced, such as the Epanechnikov kernels. Also, besides the prior probabilities, PSO is used for determining the constants of the Bayesian model's prior distributions. The new class of models is called Extended EBPNN (EEBPNN), and five models are compared with different established EBPNN and PNN approaches on four widely used classification problems from the UCI repository, with promising results.

The paper is organized as follows: Section 2 contains the necessary background information on PNNs and PSO. The proposed EEBPNN model is described in Section 3, and experimental results are reported in Section 4. The paper concludes in Section 5.

2 Background Material

PNNs and PSO are briefly described in this section for presentation completeness.

2.1 Probabilistic Neural Networks

PNNs are supervised neural network models, closely related to the Bayes classification rule [7, 12] and Parzen nonparametric probability density function estimation theory [1, 13]. Their training procedure consists of a single pass over all training patterns [1], thereby rendering PNNs faster to train, compared to the Feedforward Neural Networks (FNNs).

Consider a p–dimensional classification task and let K be the number of classes. Let also $\mathcal{T}_{\mathrm{tr}}$ be the training data set with a total of N_{tr} feature vectors, while N_k be the number of training vectors that belong to the k–th class, $k = 1, 2, \ldots, K$. The i–th feature vector of the k–th class is denoted as $X_{ik} \in \mathbb{R}^p$, where $i = 1, 2, \ldots, N_k$, $k = 1, 2, \ldots, K$. Then, a PNN consists of four layers: the *input*, *pattern*, *summation* , and *output layer*, as depicted in Fig. 1 [1, 9].

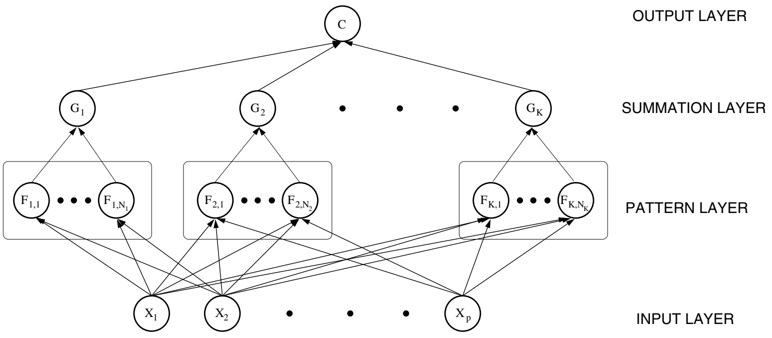

Fig. 1. The probabilistic neural network model

An input feature vector, $X \in \mathbb{R}^p$, is applied to the p input neurons that comprise the input layer, and it is passed to the pattern layer. The latter is fully interconnected with the input layer and organized into K groups of neurons. Each group of neurons in the pattern layer consists of N_k neurons, and the i–th neuron in the k–th group computes its own output by using a kernel function. The kernel function is typically a multivariate Gaussian of the form,

$$f_{ik}(X) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2} \left(X - X_{ik}\right)^{\top} \Sigma^{-1} \left(X - X_{ik}\right) \right), \tag{1}$$

where $X_{ik} \in \mathbb{R}^p$ is the center of the kernel, and Σ is the matrix of spread (smoothing) parameters. PNNs that exploit a global smoothing parameter are called *homoscedastic*, while the use of a different parameter per class is referred to as *heteroscedastic* PNN [14].

The summation layer consists of K neurons and each one estimates the conditional probability of the corresponding class given the input feature vector, X, according to the Bayes decision rule:

$$G_k(X) = \sum_{i=1}^{N_k} \pi_k f_{ik}(X), \quad k \in \{1, 2, \ldots, K\}, \tag{2}$$

where π_k is the prior probability of the k–th class, and $\sum_{k=1}^{K} \pi_k = 1$. Thus, X is classified in the class that achieves the maximum output of the summation neurons.

A limitation of PNNs is the curse of dimensionality. When the dimension of the data set is large, PNNs usually do not yield good results. A faster version of the PNN can be obtained by using only a part, instead of the whole training data set. Such a training set can be obtained either by randomly sampling from the available data or by finding “representative” training vectors for each class through a clustering technique.

For this purpose, the widely used K–medoids clustering algorithm [15] can be applied on the training data of each class. The extracted medoids are then used as centers for the PNN's kernels, instead of using all the available training data. The resulted PNNs are significantly smaller with respect to the number of neurons in the pattern layer, although there is no sound procedure for estimating the optimal required number of medoids.

2.2 The Particle Swarm Optimization Algorithm

Particle Swarm Optimization (PSO) was introduced in 1995 by Eberhart and Kennedy [16, 17], drawing inspiration from the dynamics of socially organized groups. PSO is a stochastic, population–based optimization algorithm that exploits a population of individuals to synchronously probe the search space. In this context, the population is called a *swarm* and the individuals (i.e., the search points) are called the *particles*.

Each particle moves with an adaptable velocity within the search space and retains in memory the best position it has ever encountered. This position is also shared with other particles in the swarm. In the *global* PSO variant, the best position ever attained by all individuals of the swarm is communicated to every particle at each iteration. On the other hand, in *local* PSO, best positions are communicated only within strict neighborhoods of each particle.

Assume a d–dimensional search space, $\mathcal{S} \subset \mathbb{R}^d$, and a swarm consisting of N particles. Let

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

be the i–th particle and

$$V_i = (v_{i1}, v_{i2}, \ldots, v_{id})^\top, \quad B_i = (b_{i1}, b_{i2}, \ldots, b_{id})^\top \in \mathcal{S},$$

be its velocity and best position, respectively. Assume g to be the index of the particle that attained the best previous position among all particles, and t be the iteration counter. Then, the swarm is manipulated by the equations:

$$V_i(t+1) = \chi \Big[V_i(t) + c_1 r_1 \big(B_i(t) - Z_i(t) \big) + \ c_2 r_2 \big(B_g(t) - Z_i(t) \big) \Big], \qquad (3)$$

$$Z_i(t+1) = Z_i(t) + V_i(t+1), \qquad (4)$$

where $i = 1, 2, \ldots, N$; χ is a parameter called the *constriction coefficient*; c_1 and c_2 are two positive constants called the *cognitive* and *social* parameter, respectively; and r_1, r_2, are random vectors uniformly distributed within $[0,1]^d$ [18]. All vector operations in Eqs. (3) and (4) are performed componentwise.

The best positions are then updated according to the equation:

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

The particles are bounded within the search space $\mathcal{S}$, while the constriction coefficient is derived analytically through the formula:

$$\chi = \frac{2\kappa}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|},$$

for $\varphi > 4$, where $\varphi = c_1 + c_2$ and $\kappa = 1$, based on the stability analysis due to Clerc and Kennedy [18].

3 The Proposed Extended Model

EBPNN models were proposed as a new variant of PNNs that estimate the spread parameters through Bayesian models. More specifically, a different diagonal matrix of spread parameters,

$$\Sigma_k = \text{diag}\ \left(\sigma_{1k}^2,\ \sigma_{2k}^2,\ \ldots, \sigma_{pk}^2 \right), \quad k = 1, 2, \ldots, K,$$

for each one of the K classes is used to increase model flexibility [10, 11].

The centered data per class, received from the preprocessing phase with the K–medoids algorithm as described in Section 2.1, are modeled by:

$$X_{ik} \overset{\text{iid}}{\sim} \mathcal{N}_p(\mu_k, \Sigma_k), \quad i = 1, 2, \ldots, N_k, \quad k = 1, 2, \ldots, K.$$

The prior distributions of the model parameters are defined as:

$$\begin{aligned} \mu_{jk} &\sim \mathcal{N}(0, \nu^2), \\ \tau_{jk} &\sim \mathcal{G}(\alpha, \beta), \quad j = 1, 2, \ldots, p, \end{aligned}$$

where $\tau_{jk} = \sigma_{jk}^{-2}$ and $\alpha, \beta, \nu > 0$.

In EBPNNs, it is assumed that the class centers, X_{ik}, are conditionally independent given μ_k and τ_{jk}. Also, μ_k and τ_{jk} are also considered independent, with joint posterior distribution:

$$\pi(\mu_{jk}, \tau_{jk} | X_{\cdot k,j}) \propto \tau_{jk}^{\frac{N_k}{2}+\alpha-1} \times \exp\left\{ -\tau_{jk} \left(\frac{\sum_{i=1}^{N_k} (X_{ik,j} - \mu_{jk})^2}{2} + \beta \right) - \frac{\mu_{jk}^2}{2\nu^2} \right\},$$

where $X_{ik,j}$ stands for the j–th component of the p–dimensional vector X_{ik}.

Simulation from the posterior distribution of (μ_{jk}, τ_{jk}), for $j = 1, 2, \ldots, p$, $k = 1, 2, \ldots, K$, requires the application of an indirect method, such as Gibbs sampler, since direct simulation is not feasible. The Gibbs sampler [19] produces a Markov chain by an iterative, recursive sampling from the conditional distributions that converges in distribution to the joint distribution. The full conditional distributions are given as follows:

$$\mu_{jk} | \tau_{jk}, X_{\cdot k,j} \sim \mathcal{N}\left(\frac{\tau_{jk} \sum_{i=1}^{N_k} X_{ik,j}}{\tau_{jk} N_k + \frac{1}{\nu^2}}, \frac{1}{\tau_{jk} N_k + \frac{1}{\nu^2}} \right), \tag{5}$$

$$\tau_{jk} | \mu_{jk}, X_{\cdot k,j} \sim \mathcal{G}\left(\frac{N_k}{2} + \alpha, \ \frac{\sum_{i=1}^{N_k} (X_{ik,j} - \mu_{jk})^2}{2} + \beta \right). \tag{6}$$

Starting from any point in the support of the joined distribution, we draw successively from the conditional distributions of μ_{jk} and τ_{jk}, each in turn, using the previously drawn value of the other parameter, and the obtained sequence converges to the joint distribution.

Conjugated prior distributions were chosen in EBPNNs, such that closed forms are available for the full conditional distributions. The choice of conjugated prior distributions has minor importance, since any distribution can be used as prior. In such cases, we can use a hybrid Gibbs sampler (Gibbs sampler embedding a Metropolis Hastings step) or different Monte Carlo or Markov Chain Monte Carlo simulation methods, such as Importance Sampling and Metropolis Hastings [20].

Based on the aforementioned Bayesian model, EBPNNs estimate the spread parameters of their kernels. Thus, instead of estimating $p \times K$ spread parameters, only the values of α, β, and ν, need to be determined. In recent implementations [10, 11], an exhaustive search was carried out in the range $[10^{-4}, 10]$, using

Table 1. Pseudocode of the sampling procedure and the determination of the prior probabilities with PSO in EBPNNs

```
Create the clustered training set $\mathcal{T}_{\mathrm{tr}}^{\mathrm{c}}$ from the original training set $\mathcal{T}_{\mathrm{tr}}$.
// Estimation of the spread parameters using the Gibbs sampler //
Do ($k = 1, 2, \ldots, K$)
   Do ($j = 1, 2, \ldots, p$)
         Select initial value for $\mu_{jk}$.
         Do ($m = 1, 2, \ldots, G_{\max}$)
               Draw from Eq. (6) a new $\tau_{jk}^{\mathrm{new}}$, using $\mu_{jk}$.
               Draw from Eq. (5) a new $\mu_{jk}^{\mathrm{new}}$, using $\tau_{jk}^{\mathrm{new}}$.
               Set $\mu_{jk} \leftarrow \mu_{jk}^{\mathrm{new}}$ and $\tau_{jk}^{m} \leftarrow \tau_{jk}^{\mathrm{new}}$.
         End Do
         Compute mean value, $\tau_{jk}$, of $\tau_{jk}^{m}$, $m = 1, 2, \ldots, G_{\max}$.
   End Do
   Set the spread matrix $\Sigma_k$ of class $k$ by using the relation $\tau_{jk} = \sigma_{jk}^{-2}$.
End Do
// Estimation of the prior probabilities by PSO //
Initialize a swarm of particles $Z_i$, $i = 1, 2, \ldots, N$, within the range $[0, 1]^K$.
Initialize best positions, $B_i$, and velocities, $V_i$, $i = 1, 2, \ldots, N$.
While (stopping condition not met)
      Update swarm using Eqs. (3) and (4).
      Constrain particles within $[0, 1]^K$.
      Evaluate particles based on the classification accuracy on $\mathcal{T}_{\mathrm{tr}}$.
      Update best positions.
End While
Write the obtained spread matrices $\Sigma_k$, $k = 1, 2, \ldots, K$, and prior weights.
```

variable step size, for the selection of α and β. Furthermore, the value of ν was set arbitrarily to $\nu = 1$ [10, 11].

In standard PNNs, the prior probabilities of Eq. (2) are either estimated from the available data or set randomly. In contrast to this procedure, EBPNNs employ the PSO algorithm to determine the most promising values for the prior probabilities with respect to classification accuracy. Thus, a swarm of weights is randomly generated and probes the search space of weights to find the most promising values. The underlying objective function utilized by PSO is the classification accuracy of the PNN over the whole training data set [11]. The pseudocode of the Gibbs sampling phase as well as the determination of the priors with PSO, is presented in Table 1.

The proposed *Extended Evolutionary Bayesian Probabilistic Neural Network* (EEBPNN) model extends the aforementioned EBPNN models, as follows:

(1). The *Epanechnikov kernel*, which is defined as:

$$f_{ik}(X) = \max\left\{0,\, 1 - \frac{1}{2\kappa^2}\left(X - X_{ik}\right)^{\top} \Sigma_k^{-1} \left(X - X_{ik}\right)\right\}, \qquad (7)$$

where κ is the kernel's parameter [21], is used instead of the typical Gaussian kernels. This choice is based on the fact that the Epanechnikov kernel has the smallest asymptotic mean integrated squared error (AMISE) and it is considered as optimal kernel [22]. The expected gain is significantly faster execution time, since there is no need to calculate the time–consuming exponential functions included in the Gaussian kernel. The parameter κ can be set arbitrarily by the user or, alternatively, determined by using the PSO algorithm.

(2). The PSO algorithm is employed for the selection of the most appropriate values of α and β in the Bayesian model. The constant, ν, is set to the value 0.2, which is adequate to cover the range $[-0.5, 0.5]$ of the data in the considered problems.

The described EEBPNN model introduces several new features in different aspects of the standard PNN and EBPNN model. Generally, it is not necessary to use all the new features concurrently in the same model. Thus, one can define EBPNN models with Gaussian kernels, using PSO for determining the constants of the prior distributions in the Bayesian model, as well as the prior probabilities. Alternatively, Epanechnikov kernel can be used with the established EBPNN and BPNN models, where Bayesian constants are determined through exhaustive search. In the next section, we define several alternative models, and report their performance on widely used classification tasks.

4 Experimental Settings and Results

We considered four widely used benchmark problems from the Proben1 benchmark data sets [23] of the UCI repository [24]. Specifically, we used the following data sets:

1. *Wisconsin Breast Cancer Database* (WBCD): the aim is to predict whether a breast tumor is benign or malignant [25]. There are 9 continuous attributes based on cell descriptions gathered by microscopic examination and 699 instances.
2. *Card Data Set*: the aim is to predict the approval or non–approval of a credit card to a customer [26]. There are 51 attributes (not explicitly reported for confidential reasons) and the number of observations is 690.
3. *The Pima Indians Diabetes Data Set*: the aim is to predict the onset of diabetes, therefore, there are two classes [27]. The input features are the diastolic blood pressure; triceps skin fold thickness; plasma glucose concentration in a glucose tolerance test; and diabetes pedigree function. These 8 inputs are all continuous without missing values and there are 768 instances.
4. *Heart Disease Data Set*: the aim is to predict whether at least one of the four major vessels of the heart is reduced in diameter by more than 50%, so there are two classes [28]. The 35 attributes of the 920 patients are age, sex, smoking habits, subjective patient pain descriptions and results of various medical examinations such as blood pressure and cardiogram.

Table 2. Characteristics of the four benchmark data sets

	Cancer	Card	Diabetes	Heart
Number of Instances	699	690	768	920
Variables	9	51	8	35
Classes	2	2	2	2

The characteristics of the four data sets are summarized in Table 2. In our experiments, the number of medoids extracted from each class was only the 5% of the class size. This reduces the size of the pattern layer by a factor of 20, compared to the standard PNN that utilizes the whole training data set. The choice of 5% was based on numerous trials with different fractions of the class size. Also, for the Gibbs sampler, a number of $G_{\max} = 10^4$ draws was used.

The following new models of the EEBPNN class were considered in our experiments:

M1. **Epan.BPNN:** BPNN that uses Epanechnikov kernels with $\kappa = 1$, exhaustive search for the selection of the prior distributions' constants of the Bayesian model, and the prior probabilities are set explicitly based on the fraction of each class in the data set.

M2. **Epan.EBPNN:** EBPNN that uses Epanechnikov kernels with $\kappa = 1$, exhaustive search for the selection of the prior distributions' constants of the Bayesian model, and the prior probabilities are computed with PSO.

M3. **Gauss.MCPNN:** EBPNN with Gaussian kernels, prior distributions' constants of the Bayesian model estimated by PSO, and prior probabilities are set explicitly based on the fraction of each class in the data set.

M4. **Gauss.PMCPNN:** EBPNN with Gaussian kernels, prior distributions' constants of the Bayesian model as well as the prior probabilities are estimated by PSO.

M5. **Epan.EEBPNN:** EEBPNN with Epanechnikov kernels, where the prior distributions' constants of the Bayesian model, the prior probabilities and Epanechnikov's κ are all estimated by PSO.

Moreover, the following established PNN–based models were used for comparison purposes:

M6. **PNN**: Standard PNN with exhaustive search for the selection of the spread parameter σ.

M7. **CL.PNN:** Standard PNN that uses the clustered instead of the whole training set.

M8. **GGEE.PNN:** A variant of the standard PNN, proposed by Gorunescu *et al.* [29], which incorporates a Monte Carlo search technique.

M9. **Hom.EPNN:** Homoscedastic EPNN [9] that utilizes the whole training data set for the construction of the PNN's pattern layer.

M10. **Het.EPNN:** Heteroscedastic EPNN [9] that utilizes the whole training set.

M11. **CL.Hom.EPNN:** Same with the Hom.EPNN, where only the clustered training set was used for PNN's construction.

M12. **CL.Het.EPNN:** Same with the Het.EPNN, where only the clustered training set was used.

M13. **Bag.EPNN:** Bagging EPNN that incorporates the bagging technique for the prior weighting, clustered training set and generalized spread parameters' matrix [30].

M14. **Bag.P.EPNN:** Bagging EPNN with bagging, clustered training set, generalized spread parameters' matrix and prior probabilities estimation by PSO.

M15. **Gaus.BPNN:** BPNN with Gaussian kernels and prior distributions' constants of the Bayesian model are selected by an exhaustive search.

M16. **Gaus.EBPNN:** EBPNN with Gaussian kernels, prior distributions' constants of the Bayesian model are selected by an exhaustive search, and prior probabilities estimated by PSO.

Regarding PSO, we used the default parameter values, $c_1 = c_2 = 2.05$, and $\chi = 0.729$ [18]. The number of particles was set to 10, and a maximum number of 50 generations was allowed for the detection of the prior probabilities. In the Hom.EPNN (M9) model, the number of particles was set to 5, while in the Het.EPNN (M10), Bag.EPNN (M13), and Bag.P.EPNN (M14) models, 10 particles were used and a maximum number of 100 iterations was allowed. For the bagging EPNNs, 11 bootstrap samples were drawn from each clustered training data set. Based on these samples, an ensemble of 11 EPNNs was constructed, and the final classification was obtained by a majority voting procedure. In the proposed EBPNN variants (M1, M2), a swarm of 10 particles was used for 50 iterations, while for the MCPNN and EEBPNN variants (M3–M5), 5 particles were used for 30 iterations.

Every benchmark data set was applied 10 times using 10–fold cross–validation, where every time the folds were randomly selected. The mean, median, standard deviation, minimum and maximum of the obtained classification accuracies and CPU times were recorded for all models and they are reported in Tables 3–6. Each table consists of two parts divided by a horizontal line. The upper part contains all statistics for the five proposed models M1–M5, while the lower part contains the statistics for the rest of the models. The best classification performance for the proposed models, as well as for the rest of the models, is boldfaced. Thus, there is a boldfaced line in each of the two parts of the table, which corresponds to the best performing model of the corresponding part of the table, with respect to its classification accuracy.

In Table 3, which corresponds to the Cancer data set, the M14 model, i.e., EPNN model with bagging, clustered training set, generalized spread parameters' matrix and prior probabilities estimation by PSO, exhibited the highest classification accuracy of 97.17% and a CPU time of 90.01 seconds. On the other hand, the most promising of the proposed models was M3, i.e., EBPNN

Table 3. Test set classification accuracy percentages and CPU times for the Cancer data set

	Classification Accuracy (%)					CPU time (sec.)				
Model	Mean	Median	St.D.	Min	Max	Mean	Median	St.D.	Min	Max
M1	96.39	96.35	0.18	96.14	96.71	21.40	21.42	0.06	21.32	21.47
M2	96.53	96.56	0.22	96.14	96.85	24.39	24.42	0.07	24.28	24.48
M3	**96.75**	**96.71**	**0.22**	**96.42**	**97.14**	**41.12**	**41.11**	**0.69**	**40.04**	**42.72**
M4	96.75	96.71	0.17	96.42	97.00	65.04	65.65	2.91	57.23	67.78
M5	96.55	96.49	0.24	96.28	97.13	62.36	64.27	3.59	57.04	65.31
M6	95.79	95.85	0.25	95.27	96.14	42.09	42.42	0.66	40.66	42.69
M7	91.91	92.06	0.84	90.42	92.99	0.08	0.08	0.00	0.08	0.09
M8	96.39	96.42	0.20	95.99	96.71	1.52	1.61	0.17	1.22	1.65
M9	95.82	95.85	0.28	95.28	96.28	89.12	88.82	1.07	88.12	91.73
M10	95.32	95.21	0.57	94.42	96.14	171.78	171.75	1.07	170.21	174.04
M11	90.50	90.84	1.58	87.85	92.56	0.16	0.15	0.02	0.14	0.20
M12	87.89	87.78	1.74	85.27	90.56	0.32	0.33	0.06	0.24	0.43
M13	96.85	96.78	0.46	96.14	97.85	82.78	78.07	8.86	76.22	99.75
M14	**97.17**	**97.14**	**0.16**	**96.86**	**97.43**	**90.01**	**89.86**	**0.92**	**88.97**	**92.12**
M15	96.36	96.35	0.22	96.13	96.85	27.74	28.08	1.08	24.67	28.17
M16	96.51	96.49	0.14	96.28	96.71	31.62	32.02	1.33	27.84	32.16

Table 4. Test set classification accuracy percentages and CPU times for the Card data set

	Classification Accuracy (%)					CPU time (sec.)				
Model	Mean	Median	St.D.	Min	Max	Mean	Median	St.D.	Min	Max
M1	80.58	80.94	1.03	78.55	81.59	193.86	193.69	1.37	191.86	195.82
M2	82.83	83.04	0.89	81.45	84.06	203.71	203.47	1.42	201.77	205.95
M3	84.84	84.57	0.76	84.06	86.23	223.64	221.09	20.04	199.17	262.71
M4	84.64	84.64	0.66	83.77	85.66	350.22	347.19	45.94	268.26	408.51
M5	**85.90**	**85.87**	**0.57**	**84.78**	**86.96**	**354.45**	**351.22**	**34.93**	**310.75**	**399.69**
M6	82.10	81.96	0.76	80.87	83.48	182.01	186.37	7.88	169.82	187.93
M7	80.49	80.58	0.66	79.13	81.45	0.23	0.23	0.00	0.22	0.24
M8	84.31	84.28	0.63	83.48	85.51	5.46	5.45	0.06	5.38	5.53
M9	85.35	85.22	0.38	84.93	86.09	266.10	274.39	74.56	168.72	342.27
M10	**87.67**	**87.76**	**0.51**	**86.96**	**88.55**	**521.60**	**510.24**	**142.74**	**327.08**	**671.83**
M11	82.02	81.81	1.15	80.73	84.49	0.49	0.47	0.06	0.46	0.66
M12	85.20	85.36	0.97	83.34	86.52	0.66	0.70	0.14	0.42	0.86
M13	86.64	86.67	0.51	85.80	87.39	309.85	309.36	1.88	307.58	314.33
M14	86.83	86.81	0.34	86.38	87.39	309.73	309.84	2.62	305.26	314.95
M15	84.93	85.00	0.25	84.49	85.22	215.39	214.92	1.24	214.14	217.49
M16	86.21	86.02	0.54	85.66	87.54	229.49	228.98	1.37	228.09	231.70

Table 5. Test set classification accuracy percentages and CPU times for the Diabetes data set

	Classification Accuracy (%)					CPU time (sec.)				
Model	Mean	Median	St.D.	Min	Max	Mean	Median	St.D.	Min	Max
M1	73.90	73.93	1.16	71.89	75.91	25.18	25.38	0.64	23.37	25.44
M2	71.68	71.79	1.08	69.92	73.55	28.61	28.86	0.81	26.31	28.93
M3	66.79	66.72	0.56	66.05	67.93	37.82	37.87	0.76	36.59	39.19
M4	73.88	73.64	0.53	73.35	74.49	49.92	49.52	1.18	48.80	51.59
M5	**74.64**	**74.47**	**1.18**	**72.80**	**76.69**	**56.29**	**56.53**	**1.42**	**54.15**	**58.18**
M6	65.08	65.08	0.05	64.99	65.15	49.58	49.64	0.38	49.06	50.09
M7	65.08	65.08	0.05	64.99	65.15	0.10	0.10	0.00	0.10	0.11
M8	69.43	69.24	0.68	68.53	70.38	1.87	1.87	0.03	1.83	1.90
M9	67.67	67.58	0.88	66.03	68.80	101.17	101.13	0.48	100.40	102.01
M10	69.37	69.46	0.80	67.73	70.54	195.27	195.66	0.92	193.82	196.62
M11	65.35	65.14	0.48	64.99	66.35	0.18	0.18	0.00	0.17	0.18
M12	69.30	69.18	1.59	67.08	72.36	0.36	0.36	0.01	0.35	0.38
M13	71.00	71.16	1.02	68.90	72.09	106.42	106.53	0.92	104.25	107.73
M14	71.22	71.39	1.00	69.75	72.54	106.24	106.26	0.81	105.31	108.06
M15	**74.21**	**74.35**	**0.93**	**72.43**	**75.91**	**25.18**	**25.63**	**1.09**	**22.48**	**25.79**
M16	72.93	73.26	1.50	69.92	75.06	29.62	30.27	1.51	25.94	30.43

Table 6. Test set classification accuracy percentages and CPU times for the Heart data set

	Classification Accuracy (%)					CPU time (sec.)				
Model	Mean	Median	St.D.	Min	Max	Mean	Median	St.D.	Min	Max
M1	72.26	72.12	0.48	71.52	72.94	88.26	88.17	0.47	87.52	89.27
M2	73.32	73.31	0.48	72.72	74.02	104.54	104.28	0.72	103.55	106.12
M3	**82.11**	**82.17**	**0.66**	**80.54**	**83.04**	**158.79**	**152.13**	**14.25**	**145.29**	**182.52**
M4	81.82	81.90	1.06	79.78	83.37	160.80	163.87	9.40	147.96	174.22
M5	81.82	81.90	1.06	79.78	83.37	151.42	150.62	8.70	143.03	173.56
M6	79.23	79.13	0.48	78.59	80.00	207.99	223.48	45.27	125.62	241.32
M7	79.84	79.78	0.71	78.48	80.98	0.32	0.32	0.02	0.30	0.35
M8	80.68	80.65	0.52	79.89	81.41	6.47	6.94	0.92	4.95	7.18
M9	81.50	81.52	0.27	80.87	81.74	223.28	224.35	4.28	215.15	228.97
M10	**82.60**	**82.45**	**0.40**	**82.07**	**83.26**	**438.10**	**440.29**	**6.82**	**422.45**	**449.24**
M11	79.96	79.95	0.56	79.24	81.09	0.67	0.63	0.08	0.61	0.83
M12	77.62	77.66	1.16	75.98	79.35	1.37	1.31	0.16	1.19	1.70
M13	82.28	82.34	0.62	81.20	83.15	394.49	392.36	5.93	387.13	404.55
M14	82.35	82.50	1.05	80.43	84.13	393.22	391.47	4.95	388.02	401.03
M15	80.46	80.43	0.69	79.13	81.52	88.55	88.54	0.38	87.92	89.03
M16	81.60	81.68	0.65	80.44	82.61	106.71	106.79	0.56	105.79	107.53

Table 7. The gain and loss percentages for classification accuracy and CPU between the best performing of the proposed and the rest of the models, for each benchmark problem. Negative values denote loss instead of gain.

	Best proposed model	Best of the rest models	Gain in classification accuracy	Gain in CPU time
Cancer	M3	M14	−0.4%	54.3%
Card	M5	M10	−2.1%	32.1%
Diabetes	M5	M15	0.5%	−55.2%
Heart	M3	M10	−0.5%	63.7%

with Gaussian kernels, prior distributions' constants of the Bayesian model are selected by PSO, and prior probabilities that are set explicitly based on the fraction of each class in the data set, with a classification accuracy of 96.75% and CPU time equal to 41.12 seconds. Thus, the proposed model provides a satisfactory performance that is almost 0.4% worst with respect to its classification accuracy than the best performing model, but at a 54.3% gain in CPU time.

In the results for the Card data set, reported in Table 4, the M10 model, i.e., Heteroscedastic EPNN trained with the whole training set, had the best performance, 87.67%, among all models, with a CPU time of 521.60 seconds. The most promising from the proposed models, was M5, i.e., EEBPNN with Epanechnikov kernels, where the prior distributions' constants of the Bayesian model, prior probabilities and Epanechnikov's κ are all estimated by PSO. M5 had a classification accuracy of 85.90% at the cost of a CPU time equal to 354.45 seconds. Thus, M5 had a competitive performance that is about 2% worst than the best model, although requiring 32% less CPU time.

In the Diabetes data set, reported in Table 5, the proposed M5 model had the best performance, 74.64%, among all models. However, this came with an increased CPU time of 56.29 seconds, compared to M15, which was the best performing among the rest of the models, with classification accuracy of 74.21% and CPU time 25.18 seconds. M15 consists of a BPNN with Gaussian kernels and prior distributions' parameters of the Bayesian model estimated by an exhaustive search.

In the Heart data set, reported in Table 6, M10 was again the best performing model as for the Card data set, with a classification accuracy of 82.60% and CPU time equal to 438.10 seconds. On the other hand, M3 was the best performing from the proposed models, with an accuracy of 82, 11%, which is 0.5% worse than M10, but at a computational cost of 158.79 seconds, i.e., it was 63.7% faster than M10.

In Table 7, we summarize all the gain and loss in classification accuracy and CPU time between the the best performing of the proposed and the rest of the models, for each benchmark problem, with negative values denote loss instead of gain. As we observe, the proposed models M3 and M5 have the best performance among the five proposed models M1–M5. They were able to achieve highly competitive classification accuracies but at significantly lower computational times, rendering them promising variants that can be useful especially in time–critical applications.

5 Conclusions

We proposed a class of Extended EBPNN models that incorporate several new features compared to the standard EBPNNs. These features include the use of the Epanechnikov kernel instead of the standard Gaussian kernels, as well as the selection of the prior distributions' constants of the Bayesian model by using the PSO algorithm. Five models are proposed that incorporate alternatively the aforementioned features, and four widely used benchmark classification problems from the UCI repository are employed to investigate their performance against several established PNN–based classification models.

The obtained results show that the proposed models can be competitive to the best performing of the rest models, while achieving significantly faster computation times in most cases. Thus, the proposed model can be considered as a promising alternative in time–critical PNN applications, although further research is needed to fully reveal the potential of EEBPNNs in such applications. However, in one of the test problems, the proposed model had the best performance overall, but at a higher computational cost.

Acknowledgements. The work of K.E. Parsopoulos was supported by the State Scholarship Foundation of Greece (I.K.Y.).

References

1. Specht, D.F.: Probabilistic neural networks. Neural Networks 2, 109–118 (1990)
2. Guo, J., Lin, Y., Sun, Z.: A novel method for protein subcellular localization based on boosting and probabilistic neural network. In: 2nd Asia-Pacific Bioinf. Conf. (APBC 2004), Dunedin, New Zealand, pp. 20–27 (2004)
3. Holmes, E., Nicholson, J.K., Tranter, G.: Metabonomic characterization of genetic variations in toxicological and metabolic responses using probabilistic neural networks. Chem. Res. Toxicol. 14(2), 182–191 (2001)
4. Huang, C.J.: A performance analysis of cancer classification using feature extraction and probabilistic neural networks. In: 7th Conference on Artificial Intelligence and Applications, Wufon, Taiwan, pp. 374–378 (2002)
5. Wang, Y., Adali, T., Kung, S., Szabo, Z.: Quantification and segmentation of brain tissues from mr images: A probabilistic neural network approach. IEEE Transactions on Image Processing 7(8), 1165–1181 (1998)
6. Ganchev, T., Tasoulis, D.K., Vrahatis, M.N., Fakotakis, N.: Generalized locally recurrent probabilistic neural networks with application to text-independent speaker verification. Neurocomputing 70(7-9), 1424–1438 (2007)
7. Romero, R., Touretzky, D., Thibadeau, R.: Optical chinese character recognition using probabilistic neural network. Pattern Recognition 8(30), 1279–1292 (1997)

8. Hand, J.D.: Kernel Discriminant Analysis. Research Studies Press (1982)
9. Georgiou, V.L., Pavlidis, N.G., Parsopoulos, K.E., Alevizos, P.D., Vrahatis, M.N.: New self–adaptive probabilistic neural networks in bioinformatic and medical tasks. International Journal on Artificial Intelligence Tools 15(3), 371–396 (2006)
10. Georgiou, V.L., Malefaki, S.N.: Incorporating Bayesian models for the estimation of the spread parameters of probabilistic neural networks with application in biomedical tasks. In: Int. Conf. on Statistical Methods for Biomedical and Technical Systems, Limassol, Cyprus, pp. 305–310 (2006)
11. Georgiou, V.L., Malefaki, S.N., Alevizos, P.D., Vrahatis, M.N.: Evolutionary Bayesian probabilistic neural networks. In: Int. Conf. on Numerical Analysis and Applied Mathematics (ICNAAM 2006), pp. 393–396. Wiley-VCH, Chichester (2006)
12. Raghu, P.P., Yegnanarayana, B.: Supervised texture classification using a probabilistic neural network and constraint satisfaction model. IEEE Transactions on Neural Networks 9(3), 516–522 (1998)
13. Parzen, E.: On the estimation of a probability density function and mode. Annals of Mathematical Statistics 3, 1065–1076 (1962)
14. Specht, D.F., Romsdahl, H.: Experience with adaptive probabilistic neural network and adaptive general regression neural network. In: Proc. IEEE Int. Conf. Neural Networks, vol. 2, pp. 1203–1208 (1994)
15. Kaufman, L., Rousseeuw, P.J.: Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley and Sons, New York (1990)
16. Eberhart, R.C., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings Sixth Symposium on Micro Machine and Human Science, Piscataway, NJ, pp. 39–43. IEEE Service Center, Los Alamitos (1995)
17. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proc. IEEE Int. Conf. Neural Networks, vol. IV, pp. 1942–1948 (1995)
18. Clerc, M., Kennedy, J.: The particle swarm–explosion, stability, and convergence in a multidimensional complex space. IEEE Tr. Ev. Comp. 6(1), 58–73 (2002)
19. Geman, S., Geman, D.: Stochastic relaxation, gibbs distributions and the Bayesian restoration of images. IEEE Trans. Pattn. Anal. Mach. Intel. 6, 721–741 (1984)
20. Gilks, W.R., Richardson, S., Spiegelhalter, D.J.: Markov Chain Monte Carlo in Practice. Chapman and Hall, Boca Raton (1996)
21. Looney, C.G.: A fuzzy classifier network with ellipsoidal epanechnikov functions. Neurocomputing 48, 489–509 (2002)
22. Herrmann, E.: Asymptotic distribution of bandwidth selectors in kernel regression estimation. Statistical Papers 35, 17–26 (1994)
23. Prechelt, L.: Proben1: A set of neural network benchmark problems and benchmarking rules. Technical Report 21/94, Fak. Informatik, Univ. Karlsruhe (1994)
24. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
25. Mangasarian, O.L., Wolberg, W.H.: Cancer diagnosis via linear programming. SIAM News 23(5), 1–18 (1990)
26. Quinlan, J.: Simplifying decision trees. International Journal of Man-Machine Studies 27(3), 221–234 (1987)
27. Smith, J.W., Everhart, J.E., Dickson, W.C., Knowler, W.C., Johannes, R.S.: Using the adap learning algorithm to forecast the onset of diabetes mellitus. In: Proc. Symp. Comp. Appl. & Med. Care, pp. 261–265 (1988)

28. Detrano, R., Janosi, A., Steinbrunn, W., Pfisterer, M., Schmid, J., Sandhu, S., Guppy, K., Lee, S., Froelicher, V.: International application of a new probability algorithm for the diagnosis of coronary artery disease. American Journal of Cardiology 64, 304–310 (1989)
29. Gorunescu, F., Gorunescu, M., Revett, K., Ene, M.: A hybrid incremental/monte carlo searching technique for the "smoothing" parameter of probabilistic neural networks. In: Int. Conf. Knowl. Eng., Princ. and Techn., KEPT 2007, Cluj-Napoca, Romania, pp. 107–113 (2007)
30. Georgiou, V.L., Alevizos, P.D., Vrahatis, M.N.: Novel approaches to probabilistic neural networks through bagging and evolutionary estimating of prior probabilities. Neural Processing Letters 27, 153–162 (2008)