POPULATION SIZE TRADEOFFS IN DE AND PSO-BASED METHODS FOR PNN TRAINING

Nikolay T. Dukov
Technical University of Varna

Todor D. Ganchev
Technical University of Varna

Dimitar M. Kovachev
Technical University of Varna

Michael N. Vrahatis
University of Patras

Abstract

We report on a comparative evaluation of three evolutionary methods for training the probabilistic neural network (PNN). The specific focus here is on an investigation of the acceptable tradeoffs, in terms of accuracy vs. computational and memory demands, depending on the population size. An empirical evaluation is carried out on the well-known Parkinson Speech Dataset with Multiple Types of Sound Recordings following a common experimental protocol. The numerical results identify the Unified PSO-based training as the most appropriate due to its superior classification accuracy and lower computational demands.

Keywords: probabilistic neural network (PNN), differential evolution (DE), particle swarm optimization (PSO), statistical significance test.

INTRODUCTION

The probabilistic neural network (PNN) [1] implements a robust statistical classification method which operates well even when limited amount of representative training data is available. The original PNN [1] has only one adjustable parameter, referred to as the spread factor *sigma*. In many applications the value of *sigma* is not crucial for the PNN operation, however adjusting accurately this parameter could result in a significant improvement in the overall recognition accuracy.

Hardware implementations of the PNN impose certain restrictions on the choice of method for adjusting *sigma*. Among these are: (i) it has to be computationally inexpensive, (ii) insensitive to accumulation of rounding errors, (iii) if possible should avoid the computation of derivatives etc. Evolutionary methods, such as differential evolution (DE) and swarm intelligence algorithms satisfy many of these requirements, and have been successfully used for training of the PNN [2-9].

In particular, a particle swarm optimization (PSO)-based algorithm was used for an enhanced training of the *sigma* parameter and also for the recurrent layer weights of the locally recurrent probabilistic neural network (LRPNN) [2]. The Unified PSO (UPSO) was found more successful than several other considered algorithms. A self-adaptive PNN trained with PSO showed improvement over the original PNN on two protein localization problems and two medical diagnostic tasks [3]. A differential evolution (DE)-based training method was also used successfully for the adjustment of *sigma* on a multiclass decision problem for pathogen classification [4].

In the present study we evaluate three DE and PSO-based methods for adjusting *sigma* of the PNN in a common setup with experimental protocol based on the Parkinson Speech Dataset. The main focus here is on the exploration of acceptable tradeoffs, in terms of accuracy vs. computational and memory demands, depending on the population sizes used in the DE and PSO algorithms.

PROBABILISTIC NEURAL NETWORK

The PNN [1] is a three-layer structure which in the first and second layer estimates the probability density function (PDF) for each and every class and then in the third layer makes use of a Bayesian optimal classification scheme to decide the winning class. Most often, the PDF is estimated by employing a sum of spherical Gaussian functions that are centered at each training vector:

$$p_i(\mathbf{x} \mid k_i) = \frac{1}{(2\pi)^{d/2}\sigma_i^{\,d}} \cdot \frac{1}{M_i} \times \sum_{j=1}^{M_i} \exp\left(-\frac{1}{2\sigma_i^{\,2}} (\mathbf{x}-\mathbf{x}_{ij})^T (\mathbf{x}-\mathbf{x}_{ij}) \right), \quad (1)$$

where $i = 1,...,K$ is the class index, σ_i^d is the *sigma* aka *smoothing factor*, which regulates the receptive field of the kernel function. The input vector $\mathbf{x}$ and the centers $\mathbf{x}_{ij} \in R^d$ of the kernel are of dimensionality d, and M_i is the number of pattern units in a given class k_i. Finally, exp stands for the exponential function, and the superscript T denotes the transpose of the vector.

Consequently, the PNN makes the classification decision in accordance with the Bayes' strategy for decision rules.

$$D(\mathbf{x}) = \arg\max_i \left\{ P(k_i) p_i(\mathbf{x} \mid k_i) \right\}, \; i = 1,...,K, \quad (2)$$

where $P(k_i)$ is the *a priori* probability for class k_i. Besides the decision, PNNs also provide a probability and reliability measure of each classification.

EXPERIMENTAL SETUP

An evaluation of three optimization algorithms based on DE and PSO is performed with focus on exploring the scaling effects of the population size for the estimation of optimal *sigma* for a heteroscedastic PNN. The experimental evaluation is carried out on the Parkinson Speech Dataset with Multiple Types of Sound Recordings from the UCI machine learning repository [10,11]. The dataset contains speech feature vectors with dimensionality 26, computed from voice recordings. Each of the feature vectors is tagged either Parkinson positive or negative. The experimental setup is based on the *ten times cross-validation* scheme, with division of the data 90% to 10% for training and testing, respectively.

Specifically, three training methods DE/best/1/bin, Simple PSO, and UPSO with u=0.5 were evaluated. For each of those optimization algorithms we carried out experiments with population size of 10, 20, 30, 40 and 50 particles. Every algorithm was run ten times for each of the five population sizes. Each run was with 1000 iterations and a stop condition was applied if the objective function did not change after 250 consequent iterations.

For convenience, in the rest of this paper we refer to the 15 setups of the PNN training method with a provisional index (*a-p*), where *a* stands for the subsequent number of the training method and *p* for the population size (*cf.* Table I).

Table I. Evaluation setups.

Optimization algorithm	Population size
(1-1) DE/best/1/bin	10
(1-2) DE/best/1/bin	20
(1-3) DE/best/1/bin	30
(1-4) DE/best/1/bin	40
(1-5) DE/best/1/bin	50
(2-1) Simple PSO	10
(2-2) Simple PSO	20
(2-3) Simple PSO	30
(2-4) Simple PSO	40
(2-5) Simple PSO	50
(3-1) UPSO – u=0.5	10
(3-2) UPSO – u=0.5	20
(3-3) UPSO – u=0.5	30
(3-4) UPSO – u=0.5	40
(3-5) UPSO – u=0.5	50

EXPERIMENTAL RESULTS

The average error is computed based on the ten runs for each population size and is reported in percentages (*cf.* Table II). For convenience of visualization the results in Table II are sorted in descending order with respect to the average error.

Table II. Average error in percentages computed for ten runs of each setup.

Training method setup	Error (%)
(1-5) DE/best/1/bin	25.13%
(3-4) UPSO – u=0.5	25.15%
(3-5) UPSO – u=0.5	25.17%
(1-4) DE/best/1/bin	25.19%
(1-3) DE/best/1/bin	25.44%
(3-3) UPSO – u=0.5	25.50%
(3-2) UPSO – u=0.5	25.78%
(1-2) DE/best/1/bin	26.09%
(3-1) UPSO – u=0.5	26.77%
(1-1) DE/best/1/bin	27.07%
(2-5) Simple PSO	27.37%
(2-4) Simple PSO	27.64%
(2-3) Simple PSO	27.94%
(2-2) Simple PSO	28.17%
(2-1) Simple PSO	29.43%

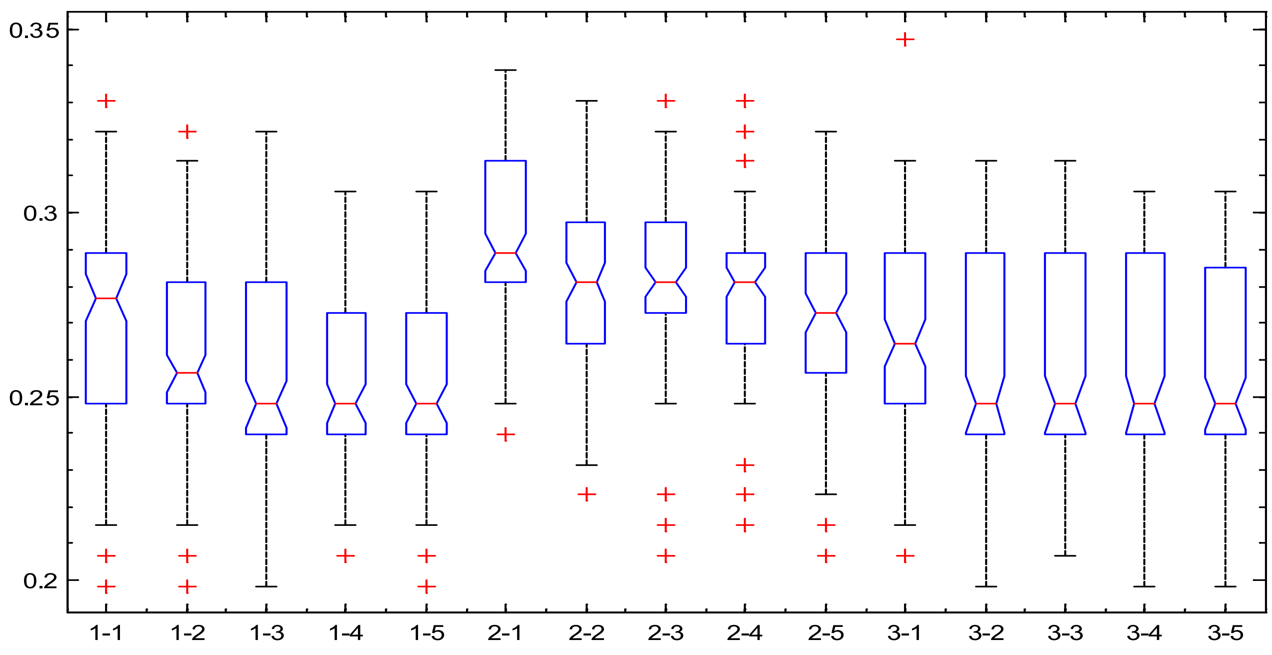

Fig. 1. Boxplot of all evaluated algorithms and populations.

In Table II we observe that seven instances have average error lower than 26%. However, in the current experimental protocol none of the five Simple PSO setups (with different population size) obtained an average error below 27%.

All experimental results are summarized in Fig. 1. In order to analyze the significance of the observed differences, we performed the Friedman statistical significance test with Holm post-hoc procedure [12]. This helps for controlling the accumulation of the Family-Wise Error Rate (FWER) in the multiple comparisons analysis.

In Table III we show the ranks after a Friedman test for the different populations of the optimization algorithm DE/best/1/bin as well as the overall *p*-value. The low *p*-value indicates that there is a significant difference among the considered instances. Regarding the achieved ranks for DE/best/1/bin, we see that the best performing population in terms of accuracy (*cf.* Table II) also achieves the best rank. This trend of correspondence applies for the other setups of the population size as well, which results in population size 10 being the worst case with the highest error rate and the lowest rank. The ranks and the overall *p*-value for different setups of the Simple PSO-based methods are presented in Table IV. Likewise the results for DE/best/1/bin, shown in Table III, the low *p*-value for Simple PSO (*cf.* Table IV) indicates for a significant difference among the populations or at least among some of them. The best performing population in terms of accuracy achieves the best rank and the other setups follow in descending order starting from population size 40 downwards to population size 10.

Table III. Ranks after the Friedman test for DE/best/1/bin and different population size.

Population size (by rank)	**Population size ranks**
Population size 50	20.53
Population size 40	21.70
Population size 30	22.71
Population size 20	27.76
Population size 10	34.82
p-value	3.0667e-22

Table IV. Ranks after the Friedman test for Simple PSO and different population size.

Population size (by rank)	Population size ranks
Population size 50	19.07
Population size 40	21.20
Population size 30	24.57
Population size 20	26.62
Population size 10	36.05
p-value	4.3042e-19

The overall *p*-value for the UPSO with u=0.5 (*cf.* Table V) and different population sizes is low enough to conclude that a significant difference exists between all or some of the population sizes.

Table V. Ranks after the Friedman test for UPSO with u=0.5 and different population size.

Population size (by rank)	Population size ranks
Population size 50	20.34
Population size 40	21.60
Population size 30	25.07
Population size 20	26.96
Population size 10	33.55
p-value	2.4636e-15

In terms of ranks however, it can be noticed that the best-performing setup in terms of accuracy is second in rank (population size 40) while the best rank is for the second setup in terms of accuracy (population size 50). Thus, further statistical tests are needed to investigate whether there is or not a significant difference between these two setups.

In order to establish a proper comparison between the population sizes for each algorithm, we considered a multiple-comparison test followed by a post-hoc procedure. For each algorithm ten hypothesizes for similarities were evaluated (*cf.* Tables VI-VIII).

Specifically, in Table VI we show the results for DE/best/1/bin, where seven out of the ten hypothesizes were rejected with a significance level $\alpha = 0.01$. The average error for the population sizes 30, 40 and 50 is considered similar. In Table VII we show the statistical significance tests for the comparisons among the population sizes of Simple PSO. Five hypothesizes were rejected with a significance level $\alpha = 0.01$ and two with $\alpha = 0.05$. The test fails to reject the similarity between population sizes: 40 – 50, 30 – 40 and 20 – 30. In the case of UPSO with u=0.5 (*cf.* Table VIII), a comparison among the results for different population sizes found out statistical differences in eight cases. Six hypothesizes were rejected with significance level $\alpha = 0.01$, one with $\alpha = 0.05$ and one with $\alpha = 0.1$. The two pairs found similar are population sizes 40 and 50, and population sizes 20 and 30.

Based on these results a further study on the effects of population size was performed for the three optimization algorithms considered here. For that purpose we selected the two best-performing setups (in terms of lowest average error rate, *cf.* Table II) for each algorithm and performed multiple-comparison tests. In Table IX we present these setups sorted in descending order.

Table VI. Adjusted values with Holm post-hoc for the Friedman multiple comparison test for DE/best/1/bin.

Comparison	Unadjusted	Holm
pop 10 – pop 20	4.0254e-05	0.0002
pop 10 – pop 30	2.2879e-11	1.8303e-10
pop 10 – pop 40	3.0369e-14	2.7332e-13
pop 10 – pop 50	7.0389e-16	7.0389e-15
pop 20 – pop 30	0.0016	0.0062
pop 20 – pop 40	2.8774e-05	0.0002
pop 20 – pop 50	2.5773e-06	1.8041e-05
pop 30 – pop 40	0.6093	0.6322
pop 30 – pop 50	0.1382	0.4146
pop 40 – pop 50	0.3161	0.6322

Table VII. Adjusted values with Holm post-hoc for the Friedman multiple-comparison test for Simple PSO.

Comparison	Unadjusted	Holm
pop 10 – pop 20	5.7720e-07	4.0404e-06
pop 10 – pop 30	1.8818e-09	1.5054e-08
pop 10 – pop 40	1.2851e-13	1.1566e-12
pop 10 – pop 50	4.9589e-16	4.9589e-15
pop 20 – pop 30	0.2799	0.4566
pop 20 – pop 40	0.0031	0.0123
pop 20 – pop 50	6.0226e-05	0.0004
pop 30 – pop 40	0.0607	0.1820
pop 30 – pop 50	0.0023	0.0116
pop 40 – pop 50	0.2283	0.4566

Table VIII. Adjusted values with Holm post-hoc for the Friedman multiple comparison test for UPSO with u=0.5.

Comparison	Unadjusted	Holm
pop 10 – pop 20	0.0002	0.0013
pop 10 – pop 30	1.1509e-06	9.2068e-06
pop 10 – pop 40	6.5373e-11	5.8835e-10
pop 10 – pop 50	4.6889e-12	4.6889e-11
pop 20 – pop 30	0.2526	0.5053
pop 20 – pop 40	0.0014	0.0069
pop 20 – pop 50	3.1443e-05	0.0002
pop 30 – pop 40	0.0198	0.0593
pop 30 – pop 50	0.0027	0.0109
pop 40 – pop 50	0.3708	0.5053

Table IX. Best two performing populations for each algorithm.

Optimization algorithm (by score)	Population size err (err in %)
(1-5) DE/best/1/bin	25.13%
(3-4) UPSO – u=0.5	25.15%
(3-5) UPSO – u=0.5	25.17%
(1-4) DE/best/1/bin	25.19%
(2-5) Simple PSO	27.37%
(2-4) Simple PSO	27.64%

In Table X we show the overall *p*-value for these instances (Table IX) as well as the rank for each setup. Again, the low *p*-value indicates for some significant differences. Here, the best rank is obtained for the best performing algorithm from Table IX. However, the positions of the following three ranks are inverted with respect to Table IX.

Table X. Ranks achieved by Friedman test for the two best performing populations from each algorithm.

Population size (by rank)	Population size ranks
(1-5) DE/best/1/bin	20.28
(1-4) DE/best/1/bin	21.33
(3-5) UPSO – u=0.5	23.40
(3-4) UPSO – u=0.5	25.11
(2-5) Simple PSO	45.53
(2-4) Simple PSO	47.37
p-value	1.0087e-63

As beforehand, Holm post-hoc procedure was used to investigate whether there are any differences among these setups. Table XI shows eight significantly different setups with significance level α = 0.01. A close inspection of the results shows that the eight occurrences of significant difference is between Simple PSO and the other two optimization algorithms (DE/best/1bin and UPSO with u=0.5).

Table XI. Adjusted values with Holm post-hoc for the Friedman multiple comparison test for the two best performing populations from each algorithm.

Comparison	Unadjusted	Holm
(1-4) – (1-5)	0.3161	1
(1-4) – (2-4)	4.8663e-32	7.2994e-31
(1-4) – (2-5)	2.7256e-28	3.5433e-27
(1-4) – (3-4)	0.1179	0.7076
(1-4) – (3-5)	0.7225	0.7416
(1-5) – (2-4)	1.2205e-31	1.7086e-30
(1-5) – (2-5)	5.1399e-28	6.1677e-27
(1-5) – (3-4)	0.0153	0.1069
(1-5) – (3-5)	0.2695	1
(2-4) – (2-5)	0.2283	1
(2-4) – (3-4)	3.3040e-19	2.9736e-18
(2-4) – (3-5)	6.3719e-21	7.0090e-20
(2-5) – (3-4)	5.7954e-18	4.6363e-17
(2-5) – (3-5)	1.3215e-19	1.3215e-18
(3-4) – (3-5)	0.3708	0.9484

In Table XII we show the average number of iterations for all setups (*cf.* Table I). Obviously, the Simple PSO-based setups need significantly lower amount of iterations for reaching a solution. The iterations needed when using DE/best/1/bin and UPSO with u=0.5 based optimization are considered similar when compared to the Simple PSO. However, DE/best/1/bin shows the average number of iterations -- above 70 and in three cases above 80. On the other hand, the UPSO with u=0.5 in most cases completes with 70 iterations on average. In one case there are 76.52 iterations on average (for population size 20) and 55.36 for population size 10.

Table XII. Average number of iterations per training method and population size.

Optimization algorithm	Average iterations
(1-1) DE/best/1/bin	70.47
(1-2) DE/best/1/bin	75.38
(1-3) DE/best/1/bin	85.36
(1-4) DE/best/1/bin	83.92
(1-5) DE/best/1/bin	86.61
(2-1) Simple PSO	26.00
(2-2) Simple PSO	31.37
(2-3) Simple PSO	20.29
(2-4) Simple PSO	10.74
(2-5) Simple PSO	16.94
(3-1) UPSO – u=0.5	55.36
(3-2) UPSO – u=0.5	76.52
(3-3) UPSO – u=0.5	71.20
(3-4) UPSO – u=0.5	71.34
(3-5) UPSO – u=0.5	70.67

CONCLUSION

On the PNN training problem considered here, the DE/best/1/bin and UPSO with u=0.5 based methods were found out to be similar and demonstrated advantage in terms of accuracy over the Simple PSO. However, UPSO with u=0.5 needs lower number of cost function evaluations, when compared to the DE/best/1/bin based methods.

In overall, when computational speed is the most restrictive criterion, Simple PSO might be the preferable choice, otherwise DE/best/1/bin or UPSO with u=0.5 would be the better option due to the better accuracy.

When hardware implementation of the PNN and its training is considered the best trade-off between complexity and computational demands is provided by the setups based on UPSO with u=0.5 and DE/best/1/bin.

Although, the overall time required for the completion of iterations in the hardware implementation will be considerably lower, the setups that require lower number of cost function evaluations will remain relatively faster and will require smaller area on the chip. Hardware implementations however will most likely result in a decreased overall accuracy due to the finite number of bits.

ACKNOWLEDGEMENTS

The authors acknowledge with thanks the support received through the research projects PD5 “Study of biologically substantiated architectures of artificial neural networks for the identification of heart diseases and neurological disorders”, SNP2 “Technological support for improving quality of life of people with the Alzheimer disease”, and the NP4 “Capacity building for object-oriented FPGA design in support of knowledge-based economy” financed by the Technical University of Varna, Bulgaria.

REFERENCES

[1] D. Specht, "Probabilistic Neural Networks," *Neural Networks,* vol. 3, (1990), pp. 109-118.

[2] T. Ganchev, “Enhanced training for the locally recurrent probabilistic neural networks”, *International Journal of Artificial Intelligence Tools*, vol. 18, no. 6, (2009), pp. 853-881.

[3] V. L. Georgiou, N. G. Pavlidis, K. E. Parsopoulos, P. D. Alevizos and M. N. Vrahatis, “New Self-Adaptive Probabilistic Neural Networks in Bioinformatics and Medical Tasks”, *International Journal on Artificial Intelligence Tools*, vol. 15, no. 3 (2006). pp. 371-396.

[4] W. Ford, K. Xiang, W. Land, R. Congdon, Y. Li, O. Sadik, “A Multi-class Probabilistic Neural Network for Pathogen Classification”, *Procedia Computer Science*, vol. 20, (2013), pp. 348-353.

[5] S. Hsieh and C. Chen, “Adaptive image interpolation using probabilistic neural network”, *Expert Systems with Applications,* vol. 36, no. 3, (2009), pp. 6025–6029.

[6] U. R. Acharya, M. R. K. Mookiah, S. V. Sree, R. Yanti, R. Martis, L. Saba, F. Molinari, S. Guerriero and J. S. Suri, “Evolutionary Algorithm-Based Classifier Parameter Tuning for Automatic Ovarian Cancer Tissue Characterization and Classification”, *Ovarian Neoplasm Imaging*, (2013), pp. 425-440.

[7] L. Li and G. Ma, "Optimizing the Performance of Probabilistic Neural Networks Using PSO in the Task of Traffic Sign Recognition," in *4th International Conference on Intelligent Computing*, (2008).

[8] P. M. Ciarelli, R. A. Krohling and E. Oliviera, "Particle Swarm Optimization Applied to Parameters Learning of Probabilistic Neural Networks for Classification of Economic Activities," in *Particle Swarm Optimization,* (2009).

[9] L. Padovese and S. Vicente, "Optimizing probabilistic neural networks by the use of genetic algorithms for rolling bearing fault diagnosis," in *Surveillance 5 CETIM*, (2004).

[10] UCI Machine Learning Data Repository, https://archive.ics.uci.edu/ml/datasets/Parkinson+Speech+Dataset+with++Multiple+Types+of+Sound+Recordings – Last accessed on 01 July 2015.

[11] B. Erdogdu Sakar, M. Isenkul, C.O. Sakar, A. Sertbas, F. Gurgen, S. Delil, H. Apaydin, O. Kursun, “Collection and Analysis of a Parkinson Speech Dataset with Multiple Types of Sound Recordings”, *IEEE Journal of Biomedical and Health Informatics*, vol. 17(4), (2013), pp. 828-834.

[12] J. Derrac, S. Garcia, D. Molina and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation,* vol. 1, no. 1, (2011), pp. 3-18.