Benchmarking Probabilistic Neural Network Algorithms

Florin GORUNESCU

Department of Mathematics, Biostatistics and Computer Science
University of Medicine and Pharmacy, Craiova, Romania
gorun@umfcv.ro

Abstract. The progress of research in probabilistic neural network (PNN) and related issues is straight related to directly compare the performance of different PNN algorithm versions. In most cases, the PNN application in real life issues involves the classical activation function (Parzen-Cacoulos estimator) only. Although this estimator has been used in most experimental works so far, it is not the only consistent estimator available for practical purpose. The aim of this paper is to introduce new activation functions and to present a performance benchmark for them, tested with a medical dataset containing information regarding hepatic diseases.
Keywords: probabilistic neural networks, benchmark rules, activation function
Math. Subjects Classification 2000: 90B15, 68T05

1 INTRODUCTION

A Neural Network (NN) is seen as an information paradigm inspired by the way the human brain processes information. NN's are applicable in virtually every situation in which a relationship between the inputs and outputs exists. The power of NN's comes to life when a pattern that has no output associated with it is given as an input. In this case, the network gives the output that corresponds to a taught input pattern that is least different from the given pattern.

A useful interpretation of the network outputs under certain circumstances is to estimate the probability of class membership, in which case the network is actually learning to estimate a probability density function (p.d.f.). This is the case of the probabilistic neural network (PNN), a special type of neural network using a kernel-based approximation to form an estimate of the p.d.f.'s of categories in a classification problem. This particular type of ANN provides a general solution to pattern classification problems by following the probabilistic approach based on the Bayes decision theory. The network paradigm basically uses the Parzen-Cacoulos estimator to obtain the corresponding p.d.f.'s of the classification categories. PNN uses a supervised training set to develop probability density functions within a pattern layer. Key advantages of PNN are: the

fast training process, an inherently parallel structure guaranteed to converge to an optimal classifier as the size of the representative training set increases and that training samples can be added or removed without extensive retraining. On the other hand, the main disadvantages refer to: not as general as back-propagation, large memory requirements, slow execution of the network and it requires a representative training set even more so than other types of NN's.

Most papers in the PNN area present either performance results for the (Specht) classical algorithm concerning a large range of classification problems, or introduce optimisation methods related to the training data set only. Lack of standard performance measures is widespread in many areas of benchmark processes. For many other fields it is clear that defining a reasonable set of such standard measures is a very difficult task, but the PNN case is not one of them.

Two problems are solved by this approach. First, we have introduced new activation functions and compared both the accuracy and the running speed of each PNN version in the training mode, these two training performance measures being considered the main characteristic of a classification algorithm. Second, we have compared the corresponding accuracy in the testing mode, since the idea is that the performance of a neural network on the test set estimates its performance in real use.

2 PNN ARCHITECTURE

The PNN consists of nodes allocated in three layers after the inputs:

- *pattern layer/unit*: there is one pattern node for each training example. Each pattern node/unit forms a product of the input pattern vector x (for classification) with a weight vector W_i, $Z_i = x \cdot W_i$, and then perform a a nonlinear operation on Z_i before outputting its activation level to the summation node/unit. Let us note that, instead of the sigmoid activation function (back-propagation algorithm), the nonlinear operation used here is $\exp\left[(Z_i - 1)/\sigma^2\right]$. Assuming that both x and W_i are normalized to unit length, this is equivalent to using $\exp\left[-(W_i - x)^\tau (W_i - x)/(2\sigma^2)\right]$.
- *summation layer/unit*: each summation node/unit receives the outputs from pattern nodes associated with a given class. It simply sums the inputs from the pattern units that correspond to the category from which the training pattern was selected, $\sum_i \exp\left[-(W_i - x)^\tau (W_i - x)/(2\sigma^2)\right]$.
- *output or decision layer/unit*: the output nodes/units are two-input neurons. These units produce binary outputs, related to two different categories $\Omega_r, \Omega_s, r \neq s, r, s = 1, 2, ..., q$, by using the classification criterion: $\sum\limits_i \exp\left[-(W_i - x)^\tau (W_i - x)/(2\sigma^2)\right] > \sum\limits_j \exp\left[-(W_j - x)^\tau (W_j - x)/(2\sigma^2)\right]$. These units have only a single corresponding weight C, given by the loss parameters the *prior* probabilities and the number of training patterns in each category. Concretely, the corresponding weight is the ratio of *a priori* probabilities, divided by the ratio of samples and multiplied by the ratio

of losses, $C = -\frac{h_s l_s}{h_r l_r} \cdot \frac{n_r}{n_s}$. This ratio can be determined only from the significance of the decision.

There were developed nonparametric techniques for estimating univariate (or multivariate) p.d.f. from random samples. Briefly, by using the multivariate Gaussian approximation, that is a sum of multivariate Gaussian distributions centered at each training sample, we obtain the following form of the p.d.f. $f_r(x)$

$$f_r(x) = \frac{1}{(2\pi)^{p/2}\sigma^p} \cdot \frac{1}{m} \cdot \sum_{i=1}^{m} \exp\left(-\frac{\|x - x_i\|^2}{2\sigma^2}\right), r = 1, 2, ..., q, \qquad (1)$$

where i is the (vector) pattern number, m is the total number of training patterns, x_i is the i-th training pattern from category (class) Ω_r, p is the input space dimension and σ is an adjustable "smoothing" parameter using the training procedure. The network is trained by setting the W_i weight vector in one of the pattern units equal to each x pattern in the training set and then connecting the pattern unit's output to the appropriate summation unit.

3 BENCHMARK RULES

The dataset used for performing benchmark on PNN must be split into at least two parts: one part on which the training is performed -the training set- and another part on which the performance of the resulting network is tested -the testing set. Such an evaluation is called cross-validation and it is necessary to avoid the overfitting (overtraining) phenomenon. For two networks trained on the same dataset, the one with larger training set error may actually be better, since the other has concentrated on peculiarities of the training set at the cost of losing much of the regularities needed for good generalization to new (unknown) datasets. Classification performance has been reported in percent of incorrectly classified examples, i.e. the classification error. This is better than reporting the percentage of correctly classified examples, because the latter makes important differences insufficiently clear. For instance, an accuracy of 98% is actually twice as good as one of 96%, which is easier to see if the errors are reported (2% compared to 4%). Only the corresponding errors on the training set and testing set were reported, even if one considers that a better benchmark would involve three categories: training, validation and testing. The training running speed has been reported in comparison with the classical PNN activation function f_1, to which corresponds the normalized speed equalling 1 (standard unit). Thus, we computed the speed for all activation functions in competition, corresponding to the same training accuracy. Basically, smaller the number of division knots N, faster the algorithm.

4 BENCHMARK COMPETITORS

For the benchmark process we have chosen a number of 19 types of activation functions. Firstly, we have considered the standard PNN algorithm, corresponding to the classical activation function:

$$f_1(x) = \frac{1}{(2\pi)^{p/2}\sigma^p} \cdot \frac{1}{m} \cdot \sum_{j=1}^{m} \exp\left(-\frac{\|x - x_j\|^2}{2\sigma^2}\right) \tag{2}$$

Next, new activation functions were introduced in order to compare their performance to the standard one. Below we list the new classifiers:

(a) Generalizations of the classical case, given by:

$$f_{kn} = \frac{1}{(2\pi)^{p/2}\sigma^p} \cdot \frac{1}{m} \cdot \sum_{j=1}^{m} \exp\left[k^n \cdot \left(-\frac{d(x, x_j)^2}{2\sigma^2}\right)\right], k \geq 2, n \geq 1$$

(b) An alternative to the classical case, based on Taylor's polynomial approximation of exponentials:

$$f_{Tr} = \frac{1}{(2\pi)^{p/2}\sigma^p} \cdot \frac{1}{m} \cdot \sum_{j=1}^{m} \sum_{k=1}^{r} \frac{\left(-\frac{d(x,x_j)^2}{2\sigma^2}\right)^k}{k!}, r \geq 1.$$

Note. For the benchmark process we considered $r = 1, 2, ..., 10$.

5 BENCHMARKING APPLICATION

The dataset used for the benchmarking process has been chosen from the medical field, involving the hepatic diseases domain. This data set has a lot of difficulties during the classification task. The dataset has 299 instances with 15 attributes, divided into 4 classes, two classes having only 30 instances each and one class having 60 instances, making thus the classification task harder. The goal of this choice was two-fold. On the one hand, PNN in general is widely applied in the medical domain and it is believed that it will receive extensive application to biomedical systems in the future. On the other hand, such data are far from being too homogeneous and, thus, represent a good choice for benchmarking learning algorithms. The data consisted of medical records of 299 individuals from the Department of Internal Medicine, Division of Gastroenterology, University Emergency Hospital of Craiova, Romania, split in 60 patients with chronic hepatitis (CH), 179 patients with liver cirrhosis (LC), 30 patients with hepatic cancer (HCC) and 30 healthy people (HP). The PNN has been applied to data in order to classify the group of individuals into four categories, depending on the diagnosis type: 1 = HCC, 2 = LC, 3 = CH and 4 = HP.

We have used an incremental-based searching technique, consisting into equally dividing the searching domain of the "smoothing" parameter σ by N knots and using them in the training process to maximize the cost function. Searching techniques based on Genetic Algorithm or Monte Carlo simulation are also available to deepen the benchmark process.

5.1. COMPARING THE TRAINING RUNNING SPEED

The first comparison test involving PNN refers to the training speed, which represents a major characteristic of this particular NN, since the main criticism regarding PNN is that all training samples must be stored and used in classifying new patterns, resulting in a very rapid increase in memory and computing time. The benchmark for the training speed is represented by the training speed of the classical PNN activation function f_1, considered as 100% speed and equalling the normalized value 1. In Figure 1 we displayed the corresponding speeds.

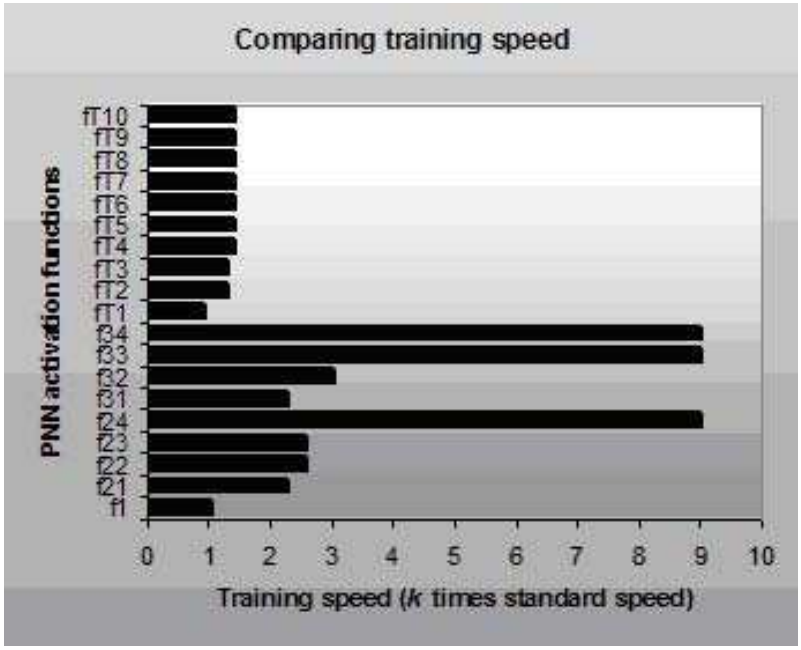

Fig. 1. Comparing the training speed

5.2. COMPARING THE CLASSIFICATION ERRORS

The second comparison test involving PNN refers to the classification performance of each version. As we mentioned above, instead of considering accuracies we considered the corresponding classification errors. We compared

the classification performance of alternative approaches both in training mode and testing mode applying the k-fold cross-validation, used for relative small datasets. Basically, each time we randomly split the initial data set (299 instances) into two subsets, training and testing, using the 10-fold cross-validation technique. A number of 254 instances (85%) of the initial dataset were withheld from the initial set for the smoothing factor adjustment (the training process).

The highest performance is done by f_{34} (training -0% error/testing -19.61% error) followed by f_{T8} and f_{31}. On the other hand, the fastest PNN version was f_{32}, more than six times faster than the standard PNN version at the same accuracy. As we remarked above, the choice of an optimal balance between accuracy and speed is up to user. This benchmark underlies the idea that the PNN application efficaciousness strongly depends on the activation function. Let us note that in each case the testing accuracy is less than the training accuracy, which represents a 'natural' behavior.

Finally, we have studied the problem of choosing the appropriate number N of dividing knots for the incremental search. Figure 2 shows the evolution of errors (training process for the entire dataset -299 instances) with the number of dividing knots (a decreasing trend of the error graph, becoming flat (zero value) when the number of dividing knots increases).

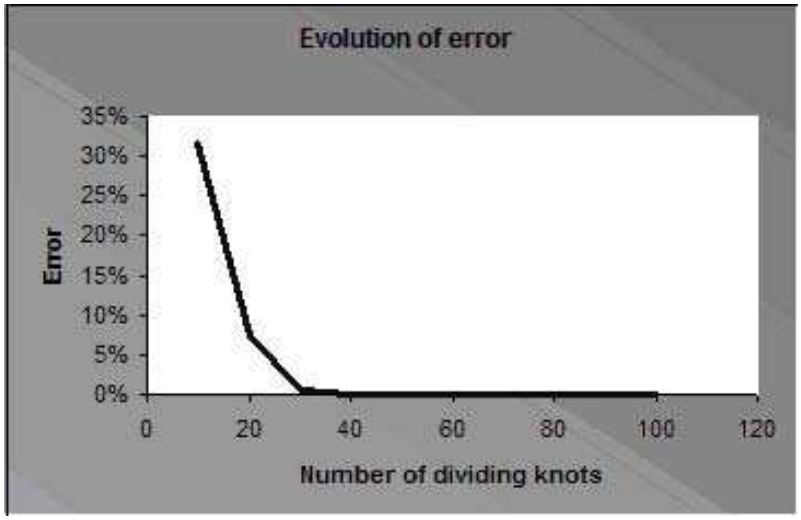

Fig. 2. Comparing the training speed

6 CONCLUSIONS AND FURTHER WORK

This paper describes the benchmarking rules and the dataset used to test the performances of some PNN algorithms. We also introduced new activation functions and tested them in comparison with the classical case. It also gives some basic performance measures indicating the difficulty of the various problems. These measures can be used as baselines for comparison. Further work must use different databases in order to test the scalability of PNN, that is the ease with which the network can be modified to fit different problems.

Acknowledgement. This paper has been conceived during my stay at the Harrow School of Computer Science, University of Westminster, London, UK, under REWARDING and DEVELOPING STAFF-HR Grant, September 2004-July 2005.

References

[1] **C. Bishop**,Neural Networks for Pattern Recognition, Oxford University Press, 1995.

[2] **F. Gorunescu, M. Gorunescu, E. El-Darzi, M. Ene, S. Gorunescu**, Statistical Comparison of a Probabilistic Neural Network Approach in Hepatic Cancer Diagnosis, *Proceedings Eurocon2005 -IEEE International Conference on "Computer as a tool"*, Belgrade, Serbia, November 21-24, 1-4244-0049-X/05/20.00 ©2005 IEEE, 2005, 237–240.

[3] ***, Proben 1-A Set of Neural Network Benchmark Problems and Benchmarking Rules, Technical Report 21/94, Fakultat fur Informatik, Universitat Karlsruhe, Germany, 1994.

[4] **D.F. Specht**, Probabilistic Neural Networks. *Neural Networks.* **3***1,* 1990, 109–118.

[5] **A. Zaknich**, Introduction to the modified probabilistic neural network for general signal processing applications. *IEEE Transactions on Signal Processing,* **46***,* 1998, 1980–1990.