An evolutionary computational approach to probabilistic neural network with application to hepatic cancer diagnosis.

Florin Gorunescu[1]
Marina Gorunescu[2]
Elia El-Darzi[3]
Smaranda Gorunescu[2]

[1] Department of Mathematics, Biostatistics & Computer Science, University of Medicine and Pharmacy of Craiova, Romania
[2] Faculty of Mathematics & Computer Science, University of Craiova, Romania
[3] Harrow School of Computer Science, University of Westminster

An Evolutionary Computational Approach to Probabilistic Neural Network with Application to Hepatic Cancer Diagnosis

Florin Gorunescu
Department of Mathematics, Biostatistics and Computer Science, University of Medicine and Pharmacy of Craiova, Romania
E-mail: gorun@umfcv.ro
Marina Gorunescu
Faculty of Mathematics and Computer Science, University of Craiova, Romania
E-mail: mgorun@inf.ucv.ro
Elia El-Darzi
Harrow School of Computer Science, University of Westminster, London, UK
E-mail: eldarze@westminster.ac.uk
Smaranda Gorunescu
Faculty of Mathematics and Computer Science, University of Craiova, Romania
E-mail: smaranda.gorunescu@gmail.com

Abstract

The performance of a probabilistic neural network is strongly influenced by the smoothing parameter. This paper introduces an evolutionary approach based on genetic algorithm to optimise the search of the smoothing parameter in a modified probabilistic neural network. A Java implementation is introduced and the computational results showed the viability of this hybrid approach to determine the optimum diagnosis for hepatic diseases.

1. Introduction

It is a matter of fact that there are a lot of examples for human trials to mimic forms, structures or processes from various domains of natural prototypes. The human brain and the process of organic evolution are of broad interest as models for creating such 'artificial intelligent' methods like neural networks or evolutionary algorithms.

The field of medical informatics had a rapid growth over the last few years and computer assisted diagnosis represents an important area of this field. Many applications using rule-based systems, statistical learning systems, genetic algorithms, neural networks have been reported to predict good diagnostic decisions in this domain.

An effective and easy to use method for addressing such task is represented by the Probabilistic Neural Networks (PNN) [3]. The PNN was developed by Specht (1988) as a supervised neural network, consisting of a 3-layer, feed-forward, and one-pass training algorithm. PNN are widely used in the areas of pattern recognition, nonlinear mapping, estimation of the probability of class membership and likelihood ratio. They are closely related to Bayesian decision rule and use Parzen or Parzen like probability density function estimators. They combine some of the best attributes of statistical pattern recognition and feed-forward neural networks.

Evolutionary algorithms (EA) form a class of probabilistic optimisation methods that are inspired by some presumed principles of organic evolution and used to solve difficult optimization problems by intelligent exploitation of a random search.

This paper introduces an EA approach, based on genetic algorithm (GA) techniques, to optimise the PNN smoothing parameters. An application to determine the optimum diagnosis for hepatic diseases and the corresponding Java implementation are presented as well.

Proceedings of the 18th IEEE Symposium on Computer-Based Medical Systems (CBMS'05)
1063-7125/05 $20.00 © 2005 **IEEE**

The paper is organised as follows: in the following two Sections the basic concepts of PNN and EA are briefly described. Next, the proposed approach is presented. Subsequently, the PNN methodology is applied to the diagnosis process as well as the Java implementation are reported. The paper ends with conclusions.

2. Probabilistic Neural Networks

The PNNs are basically classifiers. The general classification problem is to determine the category membership of a multivariate sample data (i.e. a p-dimensional random vector $\mathbf{x}$) into one of q possible groups Ω_i, $i = 1, 2,\ldots, q$, based on a set of measurements. If we know the probability density functions (p.d.f.) $f_i(\mathbf{x})$, usually the Parzen-Cacoulos or Parzen like p.d.f. classifiers:

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

the *a priori* probabilities $h_i = P(\Omega_i)$ of occurrence of patterns from categories Ω_i and the *loss* (or *cost*) parameters l_i associated with all incorrect decisions given $\Omega = \Omega_i$, then, according to the Bayesian decision rule, we classify $\mathbf{x}$ into the category Ω_i if the inequality $l_i\, h_i\, f_i(\mathbf{x}) > l_j\, h_j\, f_j(\mathbf{x})$ holds true. The standard training procedure for PNN requires a single pass over all the training patterns, giving them the advantage of being faster than the feed-forward neural networks.

Basically, the architecture of PNN is limited to three layers: the *input*/*pattern layer*, the *summation layer* and the *output layer*. Each input/pattern node forms a product of the input pattern vector $\mathbf{x}$ with a weight vector W_i and then perform a nonlinear operation, that is $\exp[-(W_i - x)^\tau (W_i - x)/(2\sigma^2)]$ (assuming that both $\mathbf{x}$ and W_i are normalized to unit length), before outputting its activation level to the summation node. Each summation node receives the outputs from the input/pattern nodes associated with a given class and simply sums the inputs from the pattern units that correspond to the category from which the training pattern was selected, $\sum_i \exp[-(W_i - x)^\tau (W_i - x)/(2\sigma^2)]$. The output nodes produce binary outputs by using the inequality:
$\sum_i \exp[-(W_i - x)^\tau (W_i - x)/(2\sigma^2)] > \sum_j \exp[-(W_j - x)^\tau (W_j - x)/(2\sigma^2)]$, related to two different categories Ω_i and Ω_j.

3. Evolutionary Algorithms-

Briefly, an EA is characterised by: (a) an encoding of the search space through chromosomes, (b) a method of generating the initial population, (c) a fitness function to measure the chromosomes performance, (d) a set of variation operators to create new chromosomes, (e) assigning values for the parameters of the algorithm. The standard structure of an EA is outlined below:

1) $t = 0$;
2) Initialise P(t) – the chromosomes population at the t-generation;
3) Evaluate P(t);
4) While ($t <$ *number_of_generations*) do {$t := t + 1$
   select P(t) from P(t -1)
   crossover on P(t)
   mutation on P(t)
   survival in P(t)
   evaluate P(t) }

end.

GA, as a particular EA class, started and still mainly operates with binary strings for representing individuals. However, when solving real world problems, knowledge about it has to be encoded appropriately, so, in many cases the complex individuals are represented by vectors with real valued components.

4. Modified Probabilistic Neural Network model and application

Different from the classical case, with unique smoothing parameter σ and Euclidian distance, this section introduces a simple and efficient GA-based method of estimating each of the smoothing parameters σ_i, i = 1, 2,…, q and using an appropriate measure of similarity between patterns.

Input. Consider q decision classes Ω_1, Ω_2,…, Ω_q, each class Ω_i containing m_i training patterns.

1) For each decision class Ω_i compute the appropriate distance between any pair of training patterns; compute the average distance D_i as well as the standard deviation and consider the corresponding 99.7% confidence interval, denoted by I_{Ω_i}; I_{Ω_i} represents the domain of the smoothing factor σ_i.

2) Consider the Parzen-Cacoulos classifier $f_i(x)$ as the corresponding parent density of each decision class Ω_i; assign $\sigma_i = D_i$.

3) In each decision class Ω_i (randomly) choose a certain training pattern x_i^0 and compute $f_i(x_i^0)$.

4) *Bayesian decision rule*: IF $l_i h_i f_i(x_i^0) > l_j h_j f_j(x_i^0)$ (for all $j \neq i$) THEN $x_i^0 \in \Omega_i$ ELSE IF $l_i h_i f_i(x_i^0) \leq l_j h_j f_j(x_i^0)$ (for some $j \neq i$) THEN $x_i^0 \notin \Omega_i$.

5) The cost function is given by the sum of training patterns that are classified in the right way.

6) Repeat step 3 for all x_i^0 in Ω_i; repeat step 3 for all vectors x_j^0 in Ω_j, for all $j \neq i$.

7) *GA approach*: Each chromosome is defined by the variable $X = (\sigma_1, \sigma_2, ..., \sigma_q)$. Each gene corresponds to the smoothing factor σ_i which takes its value from its value domain I_{Ω_i}. A population of Y-chromosomes is used. Selection is carried out by the Monte Carlo procedure. The average crossover $(X_1, X_2) \rightarrow (X_1 + X_2)/2$ is used to generate new chromosomes and for the mutation the following technique is applied: assume we decide to mutate the gene σ_i of a chromosome. We will generate a random number, whose values are 0 or 1. Then the new value for the gene is determined by $\sigma_i \pm \delta$ (δ is a small enough value to fine tune the accuracy), "+" if 0 is generated, and "-" otherwise.

8) Find the maximum of the cost function.

Output. σ_i, i = 1, 2,…, q, corresponding to the maximum of the cost function represent the optimal values of the smoothing parameters σ's for each decision category Ω_i, i = 1, 2,…, q.

5. PNN application to hepatic cancer diagnosis

The PNN-based decision model was applied to discriminate a group of individuals into a certain categories of diagnosis in the area of hepatic diseases: chronic hepatitis (CH), liver cirrhosis (LC), hepatocellular carcinoma (HCC) and healthy people (HP), using significant medical analyses. Each individual in the data set has been represented by a 15-dimensional vector **x** = (x_1, x_2,…, x_{15}), where the components represent some of the most important serum enzymes: x_1 = TB (total bilirubin), x_2 = DB (direct bilirubin), x_3 = IB (indirect bilirubin), x_4 = AP (alkaline phosphatase), x_5 = GGT (gamma glutamyl transpeptidase), x_6 = LAP (leucine amino peptidase), x_7 = AST (aspartate amino transferase), x_8 = ALT (alanine amino transferase), x_9 = LDH (lactic dehydrogenase), x_{10} = PI (prothrombin index), x_{11} = GAMMA, x_{12} = ALBUMIN, x_{13} = GLYCEMIA, x_{14} = CHOLESTEROL, x_{15} = AGE.

The model was fitted to real data consisting of 299 individuals (both patients and healthy people) from the Department of Internal Medicine, Division of Gastroenterology, University Emergency Hospital of Craiova, Romania. This group of individuals consists of 60 patients with chronic hepatitis (CH), 179 patients with liver cirrhosis (LC), 30 patients with hepatocellular carcinoma (HCC) and 30 healthy people (HP).

The algorithm was coded in Java for the ease of implementation and we have used JDBC (*Java Database Connectivity*) for the processing of the data. Thus the program is connected to a database and the records of any specific table of this database can always be updated by the physicians themselves (in MS Access or MS Excel) without the need to modify the computer program.

6. Experimental results

The key to obtain a good classification using PNN is to optimally estimate the two parameters of the Bayes decision rule, the misclassification costs and the prior probabilities. In our practical experiment we have estimate them heuristically. Thus, as concerns the costs parameters, we have considered them depending on the average distances D_i, inversely proportional, that is $l_i = 1/D_i$. As concerns the prior probabilities, they measure the membership probability in each group and, thus, we have considered them equal to each group size, that is $h_i = m_i$.

To avoid overfitting, the data set was randomly partitioned into two sets: the training set and the validation set. A number of 254 persons (85%) of the initial group were withheld from the initial group for the smoothing factor adjustment (the training process). Once optimal smoothing parameters σ's for each decision category were obtained using the training set, the trained PNN was applied to the validation set (the remaining 45 persons). In order to obtain reliable results we have repeated 10 times the above procedure. The results concerning the accuracy rate in classification, obtained for a default number of 100 chromosomes and 5 generations are shown in Table 1.

Table 1. PNN classification accuracy: training vs. validation

Run	Training accuracy rate (%)	Validation accuracy rate (%)
1	83.27	91.11
2	79.93	88.89
3	73.57	90.00
4	80.26	91.11
5	73.91	86.67
6	75.25	86.67
7	69.89	85.56
8	85.28	75.56
9	71.23	84.44
10	70.90	85.56
Average	**76.35**	**86.56**

When the PNN was applied to the training process, the sensitivity analysis indicated that the proportion of the patients correctly diagnosed was (average) 76.35%. When the PNN was applied to the validation data set, which was not subjected to neural network training, the proportion was (average) 86.56%.

In Table 2 we have displayed the dependence between the accuracy rate and the number of chromosomes (for a fixed number of 5 generations).

Table 2. PNN classification accuracy: accuracy rate as function of the number of chromosomes (5 generations)

No. of chromosomes	Best generation rank	Training accuracy (%)	Validation accuracy (%)
10	2	44.14	68.89
20	1	64.88	82.22
30	1	70.23	84.44
40	1	63.21	82.22
50	1	85.95	95.56
60	1	71.23	84.44
70	1	70.23	85.56
80	2	60.20	81.11
90	5	55.85	80.00
100	1	70.23	85.56
110	2	65.21	83.33
120	1	93.97	98.89
130	3	85.95	94.44
140	3	68.56	83.33
150	2	60.20	81.11
160	2	89.96	90.00
170	1	89.96	96.67
180	4	87.95	94.44
190	1	76.25	87.78
200	2	81.27	92.22

From this table we could see that a good accuracy is obtained when the number of chromosomes is over 100 and the number of generations is around 2.

Finally, in Table 3 we have presented the connection between the rate accuracy and the number of generations (for a fixed number of 100 chromosomes), together with the number of generation necessary to obtain the best accuracy.

Table 3. PNN classification accuracy: accuracy rate as function of the number of generations.

No. of generations	Best generation rank	Training accuracy (%)	Validation accuracy (%)
5	2	67.89	85.55
10	2	72.24	85.55
15	1	72.57	86.66
20	1	76.25	87.77
25	14	70.23	85.55
30	5	78.26	87.77
35	9	72.57	85.55
40	2	82.94	93.33
45	9	83.27	90.00
50	12	64.88	82.22
55	49	75.58	86.66
60	40	89.96	88.88
65	2	71.23	82.22
70	8	69.89	85.55
75	9	91.30	97.77
80	3	80.93	92.22

85	3	76.25	91.11
90	3	67.22	83.33
95	6	89.29	95.55
100	4	71.23	85.55

From Table 3 we see that the GAs approach provides good accuracy after a small number of generations, so the 'evolution' may be stopped after a small number of steps.

7. Conclusions

In this paper we have developed an evolutionary algorithms approach to optimise the search of the smoothing parameters in a modified probabilistic neural network and demonstrated the applicability of a PNN-based model for decision-making in the hepatic diagnosis process. Different from the classical PNN approach, using a unique smoothing parameter, our modified PNN-based model provide each category with its proper parameter, seriously increasing the computational effort and searching time. In this context, the EAs are a means to overcome these constraints by finding good approximations in a shorter time with less computational effort.

8. References

[1] H.G Bayer, H.P. Schwefel, I. Wegener, "How to analyse evolutionary algorithms", Theoretical Computer Science, 287, 2002, 101-130.

[2] Fogel D.B., Evolutionary computation: toward a new philosophy of machine intelligence, IEEE Press, Picataway, NY, 1995.

[3] M. Hintz-Madsen, L. Hansen, J. Larsen, and D. Drzewiecki, "A probabilistic neural network framework for detection of malignant melanoma" (in R.N.G. Naguib and G.V. Sherbet, eds.) Artificial Neural Networks in Cancer Diagnosis, Prognosis and Patient Management, CRC Press, 2001, 141-183.

[4] Michalewicz Z., Genetic algorithms + data structures = Evolutionary programs, 2nd Ed, Springer Verlag, 1992.

[5] D.F. Specht, "Probabilistic neural networks for classification mapping or associative memory". Proceedings IEEE International Conference on Neural Networks, 1, 1988, 525-532.