Probabilistic Neuro-Fuzzy System in Medical Diagnostic Task and its Lazy Learning-Selflearning

Yevgeniy Bodyanskiy[a], Anastasiia Deineko[a], Iryna Pliss [a] and Olha Chala [a]

[a] *Control systems research laboratory, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine*

Abstract

The computational intelligence system that is a hybrid of probabilistic neural network and the neuro-fuzzy system is proposed for solving the medical diagnostic tasks. The distinctive feature of the proposed system is the ability to process data that are given in different scales: numerical, ordinal, nominal, and binary. Also, the tuning process of the system is a hybrid of lazy learning and selflearning, according to T. Kohonen. Moreover, it is characterized by high processing speed, comparing to deep neural networks which are learning with error backpropagation procedures. The diagnostic system that is under consideration is characterized by uncomplicated computational implementation. It is intended to work with both short and long datasets in conditions of overlapping classes of diagnosis, which is typical for medical applications.

Keywords 1

Medical Data Mining, Probabilistic Neural Network, Neuro-Fuzzy System, Membership Function, Lazy Learning, Pattern Recognition

1. Introduction

Data mining methods are currently widely used in the analysis of medical information [1-3] and, first of all, in diagnosis problems based on the available data on the patient's state. As a rule, medical diagnostics problems from a data mining standpoint are considered either problem of pattern classification-recognition, clustering - recognition without a teacher, or forecasting-prediction of the disease course. Methods of computational intelligence [4-6] adapted for solving medical problems [7-10] proved to be the best mathematical apparatus here. Artificial neural networks have proved to be efficient due to their ability to train parameters - synaptic weights (and sometimes architecture) to process a training dataset, which ultimately allows or restores distributing hypersurfaces between classes of diagnoses arbitrarily false shapes. Here deep neural networks have effectively demonstrated their capabilities [11, 12], which provide recognition accuracy entirely inaccessible for other approaches.

Simultaneously, there is a broad class of situations when deep neural networks are either ineffective or generally inoperable. Here, notably the problems with a short training dataset, which often happens in real medical cases. Also, medical information is often presented not only in a numerical scale of intervals and relationships but also in a nominal, ordinal (rank) or binary scale.

Probabilistic neural networks (PNNs) [13] are well suited for solving recognition-classification problems under conditions of a limited amount of training data [13], which, however, are crisp systems operating in conditions of non-overlapping classes and learning in a batch mode. In [14-18], fuzzy and online PNN modifications were introduced to solve recognition problems under overlapping classes and trained in sequential mode. The main disadvantages of these systems are their

IDDM'2020: 3rd International Conference on Informatics & Data-Driven Medicine, November 19–21, 2020, Växjö, Sweden
EMAIL: yevgeniy.bodyanskiy@nure.ua (Ye. Bodyanskiy); anastasiia.deineko@nure.ua (A. Deineko); iryna.pliss@nure.ua (I. Pliss); olha.chala@nure.ua (O. Chala)
ORCID: 0000-0001-5418-2143 (Ye. Bodyanskiy); 0000-0002-3279-3135 (A. Deineko); 0000-0001-7918-7362 (I. Pliss); 0000- 0002-7603-1247 (O. Chala)

cumbersomeness (the size of the training dataset determines the number of nodes in the pattern layer) and the ability to work only with numerical data. The ability to work with data in different scales is an advantage of neuro-fuzzy systems [19]. Here, for the problem under consideration ANFIS, Takagi-Sugeno-Kang, Wang-Mendel and other systems can be noted.

Unfortunately, training these systems (setting their synaptic weights, and some-times membership functions) may require relatively large amounts of training datasets [20]. In this regard, it seems expedient to develop a hybrid of a probabilistic neural network (PNN) and a neuro-fuzzy system for solving classification-diagnostics-recognition problems in the conditions of overlapping classes and training data in different scales, as well as the ability to instantaneous tuning based on lazy learning [21].

2. The architecture of the proposed system

The proposed probabilistic neuro-fuzzy system contains four layers of information processing: the first hidden layer of fuzzification, formed by one-dimensional bell-shaped membership functions, the second hidden layer - aggregation one, formed by elementary multiplication blocks, the third hidden layer of adders, the number of which is determined by the number of classes plus one per which should be split the original data array, and, finally, the fourth - the output defuzzification layer, formed by division blocks, at the outputs of which signals appear that determine the levels of fuzzy membership of each observation to each of the possible classes.

Unlike classical neuro-fuzzy systems, here is no layer of tuning weights parameters. As will be shown below, the proposed method's learning process is implemented in the first hidden layer by adjusting the membership functions' parameters. It is clear that this approach simplifies the numerical implementation of the system and improves its performance.

The initial information for the system synthesis is a training dataset formed by a set of n-dimensional images-vectors $x(k)=(x_1(k),x_2(k),...,x_i(k),...x_n(k))^T$ each of which (here $1\le k\le N$ observation number in the original array or the moment of the current discrete time in Data Stream Mining tasks) belongs to a specific class Cl_j, $j=1,2,...,m.$ It is convenient to rearrange the original training dataset so that the first N_1 observations belong to the first class Cl_1 following N_2 observations to Cl_2 and finally N_m latest observations to class Cl_m. Moreover, for each class, instead of the index number *k*, it is convenient to introduce an intraclass numbering so that for the first class Cl_1 $k=t_1=1,2,...,N_1$; for class Cl_2 $k=t_2=N_1+1,N_1+2,...,N_1+N_2$; and, finally, for the last *m*-th class Cl_m $k=t_m=N_1+N_2+...+N_{m-1}+1,...,\ N_1+N_1+...+N_m=N$.

Based on this training dataset, the first hidden fuzzification layer is formed by Gaussian membership functions

$$\mu_{l_i}(x_i,w_{l_i})=\exp\left(-0.5\sigma^{-2}\left(x_i-w_{l_i}\right)^2\right), \quad (1)$$

where w_{l_i} - fixed or adjustable (more generally) centers of the corresponding membership functions, σ^2 - parameter specifying the width of the corresponding function also fixed or tuned $l_i=1,2,...,h_i;i=1,2,...,n.$

Note that in a standard probabilistic neural network, the first hidden layer of patterns is formed by multidimensional Gaussians, the number of which is determined by the training dataset size *N*. In the proposed system, the number of membership functions at each input can be different, for example, if a binary variable of type 1 or 0, "Yes" or "No," "there is a symptom," or "there is no symptom," is supplied to the input then two functions are enough at this input ($h_i=2$); if at the *i*-th input, the corresponding variable can take an arbitrary number of values, then $2\le h_i\le N.$ The total number of one-dimensional functions in the system varies in the interval

$$2n\le h\le Nn \quad (2)$$

where $h = \sum_{i=0}^{n} h_i$.

At the input of the first hidden layer, *h* signal-values of the corresponding Gaussians appears

$$o_{l_i}^{[1]} = \mu_{l_i}(x_i, w_i). \quad (3)$$

Then they are fed to the second - aggregation hidden layer, which, similarly to standard neuro-fuzzy systems, is formed by ordinary multiplication blocks, which is equal to *N*.

In this layer from one-dimensional membership functions that multidimensional kernel activation functions are formed

$$\mu_{t_j}^{j}(x_i, w_{t_j}) = \prod_{i=1}^{N} \exp\left(-0.5\sigma^{-2}\left(x_i - w_{l_i}\right)^2\right) = \exp\left(-0.5\sigma^{-2}\left\|x - c_{\tau_j}\right\|^2\right), \quad (4)$$

the vector centers of which $w_{t_j} = \left(w_{l_1}, ..., w_{l_i}, ..., w_{l_n}\right)^T$ are formed with the centers of one-dimensional membership functions. Moreover, for each *j*-th class, multidimensional activation functions N_j are formed. As a result, a signal is generated at the output of the second hidden layer

$$o_{t_j i}^{[2]} = \mu_{t_j}^{j}(x(t_j), w_{t_j}). \quad (5)$$

The third hidden layer is formed from the blocks of summation, the number of which is determined by the value *m + 1*. The first *m* adders calculate the data density distribution for each class

$$p_j(x) = o_j^{[3]} = \sum_{t_j = N_1 + N_2 + ... N_{j-1} + 1}^{N_1 + N_2 + ... + N_j} o_{t_j j}^{[2]} \quad (6)$$

and *(m+1)*-th one overall data density distribution

$$p(x) = o^{[3]} = \sum_{j=1}^{m} p_j(x) = \sum_{j=1}^{m} o_j^{[3]}. \quad (7)$$

In the output layer of defuzzification, the probability level is calculated that the presented observation *x* belongs to the *j*-th class

$$\hat{y}_j(x) = o_j^{[3]}(x) o^{[3]}(x)^{-1} = o_j^{[3]}(x)\left(\sum_{j=1}^{m} o_j^{[3]}(x)\right)^{-1}. \quad (8)$$

It is obvious that

$$\sum_{j=1}^{m} \hat{y}_j(x) = 1. \quad (9)$$

3. Combined training of probabilistic neuro-fuzzy system

In general, the proposed system's settings can be implemented based on the so-called lazy learning [21] in the same way as a standard PNN is configured. Lazy learning is based on the principle "Neurons at data points", when the kernel activation functions' centers coincide with the observations from the training set. For each observation $x(t_j)$, a multidimensional bell-shaped activation function $\mu_{t_j}^{j}(x_i, w_{t_j})$ (where $w_{t_j} \equiv x(t_i)$) is formed. It is clear that such a learning process is implemented rapidly. Still, if the amount of the training sample *N* is large enough, the PNN system becomes too cumbersome.

Following this approach, *N* membership functions should be formed at each input in a neuro-fuzzy system in the fuzzification layer. However, suppose the training signals on different inputs are specified either in the nominal or binary or in the rank scales. In this case, the number of membership functions at the corresponding inputs decreases significantly. In addition, in medical applications, numerical variables, such as the patient's temperature, are often repeated, leading to the conjugation of the number of membership functions. Finally, the most straightforward case compensates when all input signals are specified in a binary scale: "there is a symptom" – "there is no symptom". Only two membership functions with center coordinates 0 and 1 are formed at each input.

In the case when all the input variables are specified on a numerical scale, the number of one-dimensional membership functions is determined by the value $h_i = N;\ h = Nn$ that, with larger volumes of the training dataset, we can make the system too cumbersome. It is possible to overcome this problem using the self-learning procedure of the centers of membership functions [22], while their number h_i at each input remains constant.

Let's set the maximum possible value of the number of membership functions at the *i*-th input h_i^* and, before starting the learning process, place them evenly along the axis x_i on the interval [0, 1] so that the value determines the distance between the original centers $w_{l_i}(0)$ and $w_{l_i+1}(0)$ determined by the value

$$\Delta_i(0) = \left(h_i^* - 1\right)^{-1}. \tag{10}$$

When the first vector from the training dataset is fed to the system input $x(1) = \left(x_1(1),...,x_i(1),...,x_n(1)\right)^{-1}$ (it does not matter which of the classes Cl_j it belongs to), the center-"winner" $w_{l_i}^*(0)$ is determined at the beginning, which is the nearest $x_i(1)$ in the sense of distance

$$d_{l_i i} = \left|x_i(1) - w_{l_i}(0)\right|, \tag{11}$$

i.e.

$$w_{l_i}^*(0) = \arg\min\left\{d_{l_i 1},...,d_{l_i i},...,d_{l_i h_i^*}\right\}. \tag{12}$$

After this center-"winner" is pulled up to the input signal $x_i(1)$ component according to the expression

$$w_{l_i}(1) = w_{l_i}^*(0) + \eta_i(1)(x_i(1) - w_{l_i}^*(0)). \tag{13}$$

where $0 \le \eta_i(1) \le 1$ - is the learning rate. It is clear that when $\eta_i(1) = 1$ center-"winner" moves to a point $x_i(1)$ using the principle of "neurons at data points".

At the *k*-th iteration, the tuning procedure can be written in the form

$$w_{l_i}(k) = \begin{cases} w_{l_i}^*(k-1) + \eta_i(k)(x_i(k) - w_{l_i}^*(k-1)) \\ if\ w_{l_i}^*(k-1) - "winner", l_i = 1,2,...,h_i; i = 1,2,...,n, \\ w_{l_i}(k-1)\ otherwise. \end{cases} \tag{14}$$

It is easy to see that the last expression implements the self-learning principle of T.Kohonen [23] "Winner Takes All" (WTA).

Thus, the combination of lazy learning and self-learning can significantly simplify both the architecture and the process of tuning the probabilistic neuro-fuzzy system.

4. Results of the experiment

The proposed probabilistic neuro-fuzzy system is designed to work with different data types such as numerical and binary data that are presented in long and short datasets. Therefore, two datasets with different data types were taken from the UCI repository for the experimental evaluation.

The first dataset, "Heart Disease," contains 303 instances. Each of them includes detailed information about the patient, his or her physiological parameters, and symptoms of a disease. This dataset is a mix of numerical and binary data. Physiological parameters have numerical form, and symptoms typically have a binary form.

The second dataset, "Diabetes 130-US hospitals for years 1999-2008", is a long dataset that contains 100000 instances. It includes features that represent outcomes of treatment for patients: the length of stay at the hospital, information about the laboratory tests, and medications administered when patients were at the hospital. This dataset also contains numerical and binary data.

The first experiment was carried out with a short set of medical data. This small set has been further subdivided into subsets in order to determine the minimum number of data items to obtain practical classification results. The classification accuracies for machine learning method KNN – K-

nearest neighbour, EFPNN – evolving fuzzy-probabilistic neural network [24], and the proposed one were compared. The experimental evaluation results are represented in Table 1.

Table 1
The algorithms' accuracy comparison for small datasets

Algorithms for comparison	Classification accuracies				Max time, sec
	100	150	200	250	
KNN	50.24	51.63	50.7	49.03	0.03
EFPNN	56.07	61.9	71.83	79.02	0.1
PNFSL	51.14	57.7	69.34	77.52	0.79

The experiment results show that the KNN algorithm is fast, but an accuracy of it is close to 50%. Thus, the algorithm is not intended for classification of very short samples. Unlike KNN, the proposed network's classification accuracy increases as the number of elements in the sample increases. Even on very small samples, it achieves an accuracy of 77%. EFPNN also allows for greater accuracy as the sample size increases. However, it is significantly, more than 20% slower than the proposed network. The fastest method is KNN, but it should take into account that neural networks are implemented on Python and run on the central processor, and not on the GPU like KNN. It means that with the same hardware implementation, the time costs for all methods will be comparable. But the accuracy of the proposed network is higher.

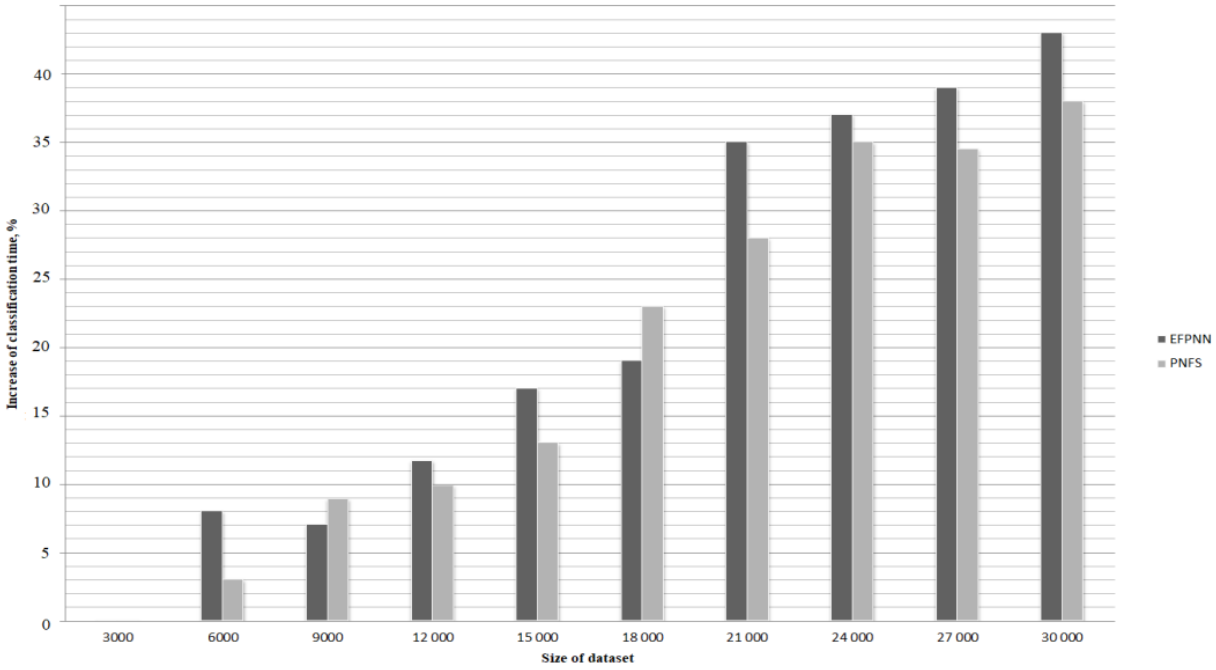

Figure 1: The dependency of dataset size and the increase of classification time

The second experiment was performed on the long dataset, which is called "Diabetes 130-US hospitals for years 1999-2008". From the initial dataset, a number of subsets that have different sizes, from 3000 to 30 000 instances were formed. The experiment is intended to compare the increase of the classification time with the dataset size grows because the absolute time consumption depends on the computer platform and used processor (CPU, GPU). Based on the results of the first experiment, for the second one, two PNFS and EFPNN neural networks, which provide a higher classification accuracy, were selected. The experiment showed that the proposed approach requires less computational cost than EFPNN. The increase in time required for larger subsets increases significantly compared to small subsets. This trend apparently exposes the influence of the software

on the classification time. Smaller datasets are usually allocated in RAM, while long datasets require swapping of data from external memory.

In general, according to the results of two experiments, the proposed approach in comparison with EFPNN provides slightly lower classification accuracy for small datasets but requires significantly lower computational costs when the dataset size grows.

5. Conclusion

The probabilistic neuro-fuzzy system is proposed for solving problems of classification-medical diagnostics in terms when information about the patient's condition is set simultaneously in numerical, rank (ordinal), nominal and binary scales. A feature of the system that is under consideration is the ability to work in the conditions of both short and growing long training sets when further they are sequentially fed into the system in online mode. The system configuration process is based on both lazy learning and self-learning, which significantly simplifies the system's computational implementation. The proposed method is characterized by high speed (just-in-time learning) and simplicity of numerical implementation, confirmed by the experiment results.

6. References

[1] Eu. Giannopoulou, Data Mining in Medical and Biological Research. N.Y.: ITAC, 2008. doi: 10.5772/95.

[2] P. Berka, S. Rauch, D. Zighed, Data Mining and Medical Knowledge Management: Cases and Applications. N.Y.: Herskey, 2009. doi: 10.4018/978-1-60566-218-3.

[3] A. Karahoca, Data Mining Applications in Engineering and Medicine. InTechOpen, 2012. doi: 10.5772/2616.

[4] C. Mumford, L. Jain, Computational Intelligence, Collaboration, Fuzzy and Emergence, Berlin: Springer, Vergal, 2009. doi: 10.1007/978-3-642-01799-5.

[5] R. Kruse, C. Borgelt, F. Klawonn, C. Moewes, M. Steinbrecher, P. Held, Computational Intelligence. A Methodological Introduction. Berlin: Springer-Verlag, 2013. doi: 10.1007/978-1-4471-5013-8.

[6] J. Kacprzyk, W. Pedrycz, Springer Handbook of Computational Intelligence, Berlin Hei-delberg: Springer, Verlag, 2015. doi: 10.1007/978-3-662-43505-2.

[7] R. Kantchev, R.: Advances in Intelligent Analysis of Medical Data and Decision Support Systems. Springer, 2013. doi: 10.1007/978-3-319-00029-9.

[8] M. Schmitt, H.-N. Teodorescu, A. Jain, A. Jain, S. Jain, Computational Intelligence Processing in Medical Diagnosis. Springer-Verlag Berlin Heidelberg , 2002. doi: 10.1007/978-3-7908-1788-1.

[9] Yu. Syerov, N. Shakhovska, S. Fedushko, Method of the Data Adequacy Determination of Personal Medical Profiles. Advances in Artificial Systems for Medicine and Education II. Springer Nature Switzerland AG, 2018, pp. 333-343. doi: 10.1007/978-3-030-12082-5_31.

[10] Ye. Bodyanskiy, I. Perova, O. Vynokurova, I. Izonin, Adaptive wavelet diagnostic neuro-fuzzy network for biomedical tasks. Conference: 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET), 2018, pp. 711 – 715. doi: 10.1109/TCSET.2018.8336299.

[11] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning. MIT Press, 2016.

[12] D. Graupe, Deep Learning Neural Networks: Design And Case Studies. N.Y.: World Scientific, 2016.

[13] D. F. Specht, Probabilistic neural networks, Neural Networks, volume 3, pp. 109-118 (1990). doi: 10.1016/0893-6080(90)90049-Q.

[14] Ye. Bodyanskiy, Ye. Gorshkov, V. Kolodyazhniy, J. Wernstedt, A learning of probabilistic neural network with fuzzy inference. Proc. Sixth. Int. Conf. on Artificial Neural Nets and Generic Algorithms “ICANNGA 2003”, Wien: Springer Verlag, pp. 13-17 (2003). doi: 10.1007/978-3-7091-0646-4_3.

[15] Ye. Bodyanskiy, Ye. Gorshkov, V. Kolodyazhniy, J. Wernstedt, Probabilistic neuro-fuzzy network with non-conventional activation functions, volume 2773 of Lecture Notes in Artificial

Intelligence. Berlin Heidelberg New York: Springer, 2003. doi: 10.1007/978-3-540-45226-3_133.
[16] L. Rutkowski, Adaptive probabilistic neural networks for pattern classification in time-varying environment. IEEE Trans. on Neural Networks, 2004, pp. 811-827. doi: 10.1109/TNN.2004.828757.
[17] J.-H. Yi, J. Wang, G.-G. Wang, Improved probabilistic neural networks with self-adaptive strategies for transformer fault diagnosis problem advances, Mechanical Engineering, vol. 8, pp. 1-13 (2016).
[18] P. Zhernova, I. Pliss, O. Chala, Modified fuzzy probabilistic neural network. Intellectual Systems For Decision Making and Problems of Computational Intelligence ISDMCI'2018, pp. 228-230, Kherson: PP Vyshemirsky V. S., (2018).
[19] Souza, P.V.C.: Fuzzy neural networks and neuro-fuzzy networks: A review the main techniques and applications used in the literature. Applied Soft Computing, vol. 92 (2020).
[20] S. Osowski, Sieci neuronowe do przetwarzania informacji, Warszawa: Oficijna Wydawnicza Politechniki Warszawskiej, 2006.
[21] O. Nelles, Nonlinear Systems Identification. Berlin: Springer, 2001. doi:10.1007/978-3-662-04323-3.
[22] Ye. Bodyanskiy, O. Tyshchenko, A. Deineko, Evolving Neuro-fuzzy Systems with Kernel Activation Functions: Their Adaptive Learning for Data Mining Tasks, volume 58, Saar-brüken: LAP LAMBERT Academic Publishing, 2015.
[23] T. Kohonen, Self-Organizing Maps. Berlin: Springer-Verlag (1995). doi: 10.1007/BF02844683.
[24] Ye. Bodyanskiy, A. Deineko, I. Pliss, O. Chala, Evolving fuzzy-probabilistic neural network and its online learning. 2020 10th International Conference on Advanced Computer Information Technologies, ACIT 2020 – Proceedings, 2020, pp. 465-468.