Journal of Biomedical Informatics 63 (2016) 74–81

Contents lists available at ScienceDirect

Journal of Biomedical Informatics

journal homepage: www.elsevier.com/locate/yjbin

Boosting backpropagation algorithm by stimulus-sampling: Application in computer-aided medical diagnosis

Florin Gorunescu [a,*], Smaranda Belciug [b]

[a] Department of Biostatistics and Informatics, University of Medicine and Pharmacy of Craiova, Craiova 200349, Romania
[b] Department of Computer Science, University of Craiova, Craiova 200585, Romania

ARTICLE INFO

Article history:
Received 29 January 2016
Revised 25 July 2016
Accepted 3 August 2016
Available online 4 August 2016

Keywords:
Backpropagation/stimulus-sampling model
Colon cancer
Breast cancer
Diabetes
Thyroid
Fetal heartbeat

ABSTRACT

Neural networks (NNs), in general, and multi-layer perceptron (MLP), in particular, represent one of the most efficient classifiers among the machine learning (ML) algorithms. Inspired by the stimulus-sampling paradigm, it is plausible to assume that the association of stimuli with the neurons in the output layer of a MLP can increase its performance. The stimulus-sampling process is assumed memoryless (Markovian), in the sense that the choice of a particular stimulus at a certain step, conditioned by the whole prior evolution of the learning process, depends only on the network's answer at the previous step. This paper proposes a novel learning technique, by enhancing the standard backpropagation algorithm performance with the aid of a stimulus-sampling procedure applied to the output neurons. The network uses the observable behavior that varies throughout the training process by stimulating the correct answers through corresponding rewards/penalties assigned to the output neurons. The proposed model has been applied in computer-aided medical diagnosis using five real-life breast cancer, colon cancer, diabetes, thyroid, and fetal heartbeat databases. The statistical comparison to well-established ML algorithms proved beyond doubt its efficiency and robustness.

1. Introduction

Among the rich area of traditional applications of Bioinformatics, the use of ML tools in processing biological data, here including medical data, represents nowadays an important trend [1]. The intelligent medical data analysis, especially conceived for automatic diagnosis, became an unexpectedly fruitful niche within the Bioinformatics field for the intensive exploitation of ML algorithms. Medical diagnosis (or differential diagnosis) is commonly perceived as the process of discriminating between different diseases, possibly accounting for a patient's illness, based on the available medical data. In this context, the use of ML algorithms supports the human medical-decision process by exploiting the immense processing power of computers when using a huge amount of data. Fed with different medical data of a patient, the "intelligent" assistant will provide the most probable diagnosis by an automatic comparison with well-established data contained in medical databases from high quality sources. From the computer-aided medical diagnosis point of view, this is a classification problem, the ML algorithms becoming popular tools for solving such a task. ML algorithms can improve the human decision-making process, minimizing thus the possible physician's error. Although the use of ML techniques in medical decision-making is nowadays a current practice, the physicians are necessarily required to make the final decision based on the computer support, so doctors can never be replaced by computers.

Recent years have seen a large development of new approaches regarding NNs applied to the medical diagnosis. NNs, seen as "massively parallel distributed processors" [2], are used to model arbitrary non-linear data based on their unique property of mimicking the human brain knowledge-processing paradigm. Among the classical NNs algorithms, the most known and used architecture is the feedforward multi-layer network, also called MLP. MLP stores knowledge in its synaptic weights and the learning/training process achieves the estimation of the optimal values of these parameters for a certain task. As learning method, the back-propagation (BP) algorithm in conjunction with gradient descent applied to an error function is one of the most popular training techniques for MLP [3]. NNs have been used successfully in the automated medical diagnosis [4,5]. Breast cancer detection and recurrence, and lung cancer benefited from the contribution of NNs [6–9]. NNs have been applied to cardiovascular diseases

* Corresponding author at: Department of Biostatistics and Informatics, University of Medicine and Pharmacy of Craiova, 2 Petru Rares Str., Craiova 200349, Romania.
E-mail addresses: gorunef@gmail.com, fgorun@rdslink.ro, florin.gorunescu@webmail.umfcv.ro (F. Gorunescu), smaranda.belciug@inf.ucv.ro (S. Belciug).

http://dx.doi.org/10.1016/j.jbi.2016.08.004
1532-0464/

[10,11]. There are also NN-based approaches regarding diabetes [12,13]. NNs applications concern other diseases also, such as pancreatitis [14,15], Alzheimer's disease [16], prostate cancer [17], colon cancer [18,19], etc.

Different from other approaches trying to improve the accuracy of MLP trained with the BP algorithm, the current work proposes a novel technique inspired by the stimulus-sampling paradigm. Stimulus-sampling theory [20], developed within the statistical learning framework, has known a certain interest in the learning theory in attempting to develop a statistical explanation for learning phenomena. According to this theory, the learning process is gradual and cumulative, and environmental changes cause variability for learning. Consequently, a certain stimulus-response association learned in a certain training step will presumably result in a general improvement of the overall learning process. The underlying idea of this paper is to boost the traditional BP algorithm by considering the learning process as a mix BP/stimulus-sampling. Two distinct phases of computations are envisaged. The BP algorithm has the task of updating weights in a standard manner. The stimulus-sampling-based approach strengthens or weakens the responses of the neurons belonging to the output layer at a certain step depending on the correctness of the answers of the previous step.

The work has two main contributions. First, a novel mix BP/stimulus-sampling learning technique is developed. On the other hand, the model is validated on five real-world medical databases, publicly available, targeting different major diseases. The statistical comparison indicated that the novel mix learning approach outperforms in most cases the conventional techniques.

The remainder of this paper is organized in four sections. Section 2 presents both the design and implementation of the novel model, and the five real-world datasets for the benchmarking process. Section 3 presents the experimental results in terms of performance analysis and performance assessment. Section 4 briefly summarizes the main characteristic of the novel approach, while Section 5 deals with the conclusions and future work.

2. Materials and methods

2.1. Model description

This paper proposes the use of a stimulus-sampling technique applied to the output layer of a MLP in order to enhance the BP generalization capability. The model, denoted BPSS-MLP, is presented in two steps. Firstly, the parent MLP is described, and secondly, the novel stimulus-sampling boosting technique is detailed.

2.1.1. Parent MLP model

MLP is generally considered the most popular NN model in use today. The MLP structure involves some critical hyper-parameters in its design, such as the network depth, the number of hidden neurons per layer, the initial learning rate, and the momentum. The *universal approximation theorem* for nonlinear input-output mapping, which is directly applicable to MLP, allows the use of a single hidden layer (3-MLP) to uniformly approximate any continuous function, and such architecture is theoretically sufficient to model almost any real-life problem [2,3]. We have consequently chosen three layers (one hidden) for the parent MLP as the depth of the network. One suggests choosing the number of hidden units by taking into account both the number of input and output variables, and the desired approximation order [21]. We have tried different number of hidden units via cross-validation and depending on each dataset. On the one hand, the choice of the number of hidden units took into account the predictive accuracy in the testing phase relative to the predictive accuracy in the learning phase, in the sense that the two accuracies to be as similar as possible. On the other hand, the underlying idea has considered both the common empirical rule-of-thumb "*an optimal number of hidden units usually ranges between the number of input units and the number of output units*", and the criterion "*the smallest number of hidden units providing a performance close to the Bayesian classifier*". There is a clear approach for the appropriate selection of the model hyper-parameters (e.g., [2,3]). We have chosen heuristically the number of hidden units for each dataset as a trade-off between cross-validation accuracy and minimization of the network size [2]. Both the initial learning rate η_0, corresponding to the learning rate η varying over iterations, and the momentum constant μ are highly critical hyper-parameters for a faster convergence without divergent oscillation to a local minimum in the error surface [2,3]. Therefore, we have considered the common practice to select them via cross-validation. Using different combinations of initial learning rates and momentum values belonging to the interval (0, 1), we have chosen the most suitable couple (η_0, μ) via cross-validation for each dataset. The selected models through hyper-parameters optimization have been then used to analyze and assess the algorithm's performance.

The parent MLP is characterized by N training samples, a (multivariate) output, the (synaptic) weights w_{ij}, the tangent "*tanh*" as activation function for the hidden layer, and a *softmax* activation function for the network output units [3].

The training samples are represented as vectors $\mathbf{x_k} = (x_1^k, x_2^k, \ldots, x_p^k)$, $k = 1, 2, \ldots, N$, with the number p of the input neurons equaling the number of attributes. Normalized inputs have been used in order to increase the convergence speed [22]. The network has one output y_i for each class Y_i, so the (multivariate) output is represented as a vector $\mathbf{y} = (y_1, y_2, \ldots, y_q)$ corresponding to the decision classes $Y_1, Y_2, \ldots, Y_q$. The target data has a "1-*of*-*q*" coding scheme (rule), i.e., $y_1 \sim (0, 0, \ldots, 1)$, $y_2 \sim (0, 0, \ldots, 1, 0), \ldots, y_q \sim (1, 0, \ldots, 0)$. Appropriate synaptic weights initialization has been considered, allowing the training algorithm to produce a good set of weights and improve the training speed [2,3]. We have used the following strategy [2] for the weights initialization. The initial weights have been generated from a uniform distribution with a mean of zero and a variance equaling to the reciprocal of the number of synaptic connections of a neuron.

The activation function was chosen depending on the layer type. In this regard, the hyperbolic tangent:

$$y = f(u) = a \cdot \tanh(b \cdot u), \tag{1}$$

with the suitable values for its parameters $a = 1.7159$, and $b = 2/3$, suggested in [22], has been chosen as activation function for the hidden layer, because it converges faster in many instances. Taking into account the number of classes (at least two), the *softmax* activation function:

$$\text{softmax}\ (y_i) = \frac{\exp[y_i - \max(y)]}{\sum_{i=1}^{q} \exp[y_i - \max(y)]}, \tag{2}$$

has been used as a smooth version of the *winner-takes-all* (WTA) model for the output layer [3].

After each learning epoch is finished (i.e., a complete presentation of the entire training set during the learning process), a certain error function is computed. There are many choices of the error function for classification problems [2,3]. Even if the mean-square-error (MSE) is not the most appropriate for classification, we have considered it motivated by analytical simplicity and because it can approximate the posterior probabilities. Thus, MSE of the network, given by:

$$\frac{1}{N} \times \sum_{\text{training samples}} \sum_{\text{output neurons}} (\text{actual output} - \text{network output})^2, \tag{3}$$