Applied Soft Computing 25 (2014) 26–39

Contents lists available at ScienceDirect

Applied Soft Computing

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

Training algorithms for Radial Basis Function Networks to tackle learning processes with imbalanced data-sets

M.D. Pérez-Godoy*, Antonio J. Rivera, C.J. Carmona, M.J. del Jesus

Department of Computer Science, University of Jaén, 23071 Jaén, Spain

ARTICLE INFO

Article history:
Received 9 November 2012
Received in revised form 6 May 2014
Accepted 12 September 2014
Available online 22 September 2014

Keywords:
Imbalanced data-sets
Radial Basis Function Networks
Training algorithms
Evolutionary computation

ABSTRACT

Nowadays, many real applications comprise data-sets where the distribution of the classes is significantly different. These data-sets are commonly known as imbalanced data-sets. Traditional classifiers are not able to deal with these kinds of data-sets because they tend to classify only majority classes, obtaining poor results for minority classes. The approaches that have been proposed to address this problem can be categorized into three types: resampling methods, algorithmic adaptations and cost sensitive techniques.

Radial Basis Function Networks (RBFNs), artificial neural networks composed of local models or RBFs, have demonstrated their efficiency in different machine learning areas. Centers, widths and output weights for the RBFs must be determined when designing RBFNs.

Taking into account the locally tuned response of RBFs, the objective of this paper is to study the influence of global and local paradigms on the weights training phase, within the RBFNs design methodology, for imbalanced data-sets. Least Mean Square and the Singular Value Decomposition have been chosen as representatives of local and global weights training paradigms respectively. These learning algorithms are inserted into classical RBFN design methods that are run on imbalanced data-sets and also on these data-sets preprocessed with re-balance techniques. After applying statistical tests to the results obtained, some guidelines about the RBFN design methodology for imbalanced data-sets are provided.

1. Introduction

Many real applications are associated to data-sets with an implicit imbalance of the existing classes. This fact implies that the number of instances of certain classes is much lower than the instances of the other classes. The importance of these data-sets resides in the fact that a minority class usually represents the concept of interest, for example patients with illness in a medical diagnosis problem; whereas the other class represents the counterpart of that concept (healthy patients). For this kind of problems standard classifier algorithms have a bias toward the majority class. This is due to the fact that the mechanisms inside these classifiers are positively weighted to predict the majority class defined by the accuracy metrics. The different methods proposed for addressing this problem can be categorized into three groups [1]: resampling methods [2], algorithmic adaptations [3], and cost sensitive techniques [4].

Radial Basis Function Networks (RBFNs) [5] are one of the most important Artificial Neural Network paradigms in the field of Machine Learning. RBFNs have important features such as: a simple topological structure; the possibility of extracting rules [6,7]; a universal approximation capability [8]; and that each neuron/RBF has a characteristic locally-tuned response. RBFNs have been successfully applied in most important machine learning areas [9] such as classification [10,11], regression [12,13] or time series forecasting [14,15]. Examples of application fields where RBFNs have been demonstrated their good behavior are: engineering problems [16–18], medical diagnosis [19,20] or web mining [21,22], among others. They have also obtained outstanding results in imbalanced problems [23]. Typically, RBFNs are designed in two stages: First, RBF parameters (center and width) are calculated with different techniques [9]; and then, weights are obtained with local or more global training algorithms. A global training algorithm considers all the data-set instances at the same time in a matrix that is resolved using numerical methods, whereas a local training algorithm separately trains each instance, only modifying weighs of the nearest RBFs.

From these premises, the objective of this paper is to carry out a study that characterizes the most suitable RBFN training methodology (local and global) taking into account the known locally-tuned response of the RBFs and avoiding the above mentioned bias toward the majority class on imbalanced data-sets.

* Corresponding author: Tel.: +34 953 212892; fax: +34 953 212472.
E-mail address: lperez@ujaen.es (M.D. Pérez-Godoy).

http://dx.doi.org/10.1016/j.asoc.2014.09.011
1568-4946/© 2014 Elsevier B.V. All rights reserved.

Thus, this paper analyzes the characteristics of Least Mean Square (LMS) [24] and Singular Value Decomposition (SVD) [25], as local and global weights training representative methods respectively. In this way, the behavior of both training algorithms is studied with different RBFN design methods: an Incremental algorithm, a Clustering method, a traditional Evolutionary Computation method (Pittsburgh codification) and CO^{2}RBFN [10], an evolutionary cooperative–competitive RBFN design method.

During experimentation, the RBFN design algorithms were applied to a largest group of data-sets with different imbalance ratio (IR) and to the same group of data-sets preprocessed in order to re-balance them. The analysis of the results let us to obtain a set of guidelines for the use of weights training algorithms in imbalanced frameworks.

The paper is organized as follows: First, Section 2 introduces RBFNs. Section 3 describes LMS and SVD, the two weights training algorithms for RBFNs. Section 4 details the most important paradigms for RBFN design. In Section 5, the imbalance problem is analyzed. The experimental framework of this research is specified in Section 6 and the results and their analysis are shown in Section 7. Finally, the conclusions of the paper are outlined in Section 8.

2. Radial Basis Function Networks

From a structural point of view, an RBFN is a feed-forward neural network with three layers: an input layer with n nodes, a hidden layer with m neurons or RBFs, and an output layer (Fig. 1).

The m neurons of the hidden layer are activated by a radially-symmetric basis function, $\phi_i : R^n \to R$, which can be defined in several ways [26], the Gaussian function being the most widely used (Eq. (1)):

$$\phi_i(\vec{x}) = \phi_i(e^{-(\|\vec{x}-\vec{c}_i\|/d_i)^2}) \tag{1}$$

where $\vec{c}_i \in R^n$ is the center of basis function ϕ_i, $d_i \in R$ is the width (radius), and $\|\|$ is typically the Euclidean norm on R^n. This expression is the one used in this paper as the Radial Basis Function (RBF). The output node implements the following function, where weights w_{ij} show the contribution of an RBF to the output node (Eq. (2)):

$$f_j(\vec{x}) = \sum_{i=1}^{m} w_{ij}\phi_i(\vec{x}) \tag{2}$$

The objective of any RBFN design process is to determine centers, widths and the linear output weights connecting the RBFs to the output neuron layer. The most traditional learning procedure has two stages [27–29,13,30]: first, learning of centers and widths, and then, training of output weight. In order to obtain the weights in the second stage, two paradigms are usually followed: gradient descent based methods (with a local behavior) [29,31,30] and global linear regression methods [12,32,13,31,33].

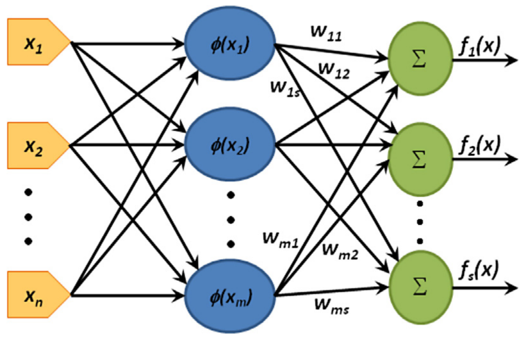

Fig. 1. RBFN topology.

One early technique to adjust the centers and widths is Clustering [34]. *K*-means algorithm [35] is a well-know algorithm based on clustering, which has been used for the RBFN design in [36]. For this kind of methods the center of the clusters are established as the center of the RBFs and RBF widths are typically set to the width of the previously calculated clusters or as the average distance between RBFs.

RBFNs were initially used for function approximation. For this reason, most of the RBFN design methods are based on traditional optimization techniques such as regularization [37], orthogonalization of regressors [38], gradient-based [39], or Levenberg–Marquardt [40]. These techniques can be used to decide if the RBFs aggregate or eliminate and may be considered as forward or backward selection methods [41]. The RAN algorithm [42] is one of the most well-known methods based on an incremental (aggregation) scheme.

Another important paradigm for RBFN design is Evolutionary Computation (EC) [43–45], which uses natural evolution and stochastic searching to design optimization algorithms. Reviews of EC applied to RBFN design can be found in [46,9]. In most of the proposals within this evolutionary paradigm, an individual represents a whole RBFN, and different operators are applied to the entire population to improve individual fitness [46,47]. This approach is known as the Pittsburgh representation scheme. An alternative to this evolutionary approach is the cooperative–competitive evolutionary or cooperative–coevolutionary strategy [48,49]. It provides a framework within which an individual of the population represents only a part of the solution, as it evolves in parallel and competes to survive, but at the same time cooperates to find a common solution (the complete RBFN).

In the engineering field, it can be found other RBFN design methods [17]. In [50–52] control systems are modeled using RBFNs. To developed them, first genetic algorithms are employed to determine the initial parameters of the networks. Then, the parameters of the decoupled adaptive neural network controllers are updated regarding some stability theories, such as the Lyapunov theory, and boundary-layer functions that guarantee the convergence of the state errors within a specified error bound. In the experimentation, the success of the design methods are demonstrated testing the stability and the error convergence of the modeled adaptive control laws.

Pérez-Godoy et al. developed an evolutionary cooperative–competitive RBFN design method, CO^{2}RBFN [10]. The standard version of CO^{2}RBFN, which uses the LMS algorithm for training weights, was applied to the classification problem of imbalanced data-sets [23] and achieved significant results. Other conclusions deduced in [23] were the good behavior of some RBFN design methods dealing with imbalanced data-sets.

As mentioned in this paper a more general objective is established, specifically to characterize the weights training phase inside a RBFN design process for imbalance problems. Thus, we will analyze the behavior of local and global training algorithms with different RBFN design methods such as an Incremental algorithm, a Clustering method, a traditional Evolutionary Computation method (Pittsburgh codification) and CO^{2}RBFN. Theses methods are described in Section 4.

3. Training algorithms for RBFN weights

During the training phase for RBFN weights a known set of input and output data pairs were delivered to the RBFN to compute the output layer weights. Taking into account Eq. (2), the least squares recipe is then to minimize the sum-squared-error.