X Congresso Brasileiro de Inteligência Computacional (CBIC'2011), 8 a 11 de Novembro de 2011, Fortaleza, Ceará

Meta-learning to optimize the number of hidden nodes of MLP networks trained by Extreme Learning Machine algorithm

Tarcísio Lucas, Ricardo Prudêncio and Teresa Ludermir
Center of Informatics, Federal University of Pernambuco
{tdpl,rbcp,tbl}@cin.ufpe.br

Carlos Soares
Faculty of Economic, University of Porto
csoares@fep.up.pt

Abstract – The optimization of Artificial Neural Networks (ANNs) is an important task to the success of using these models in real-world applications. The solutions adopted to this task are expensive in general, involving trial-and-error procedures or expert knowledge which is not always available. In this work, we investigate the use of meta-learning to chose the number of hidden nodes in MLP networks trained by Extreme Learning Machine algorithm. Meta-learning is a research field aiming to automatically acquiring knowledge which relates features of the learning problems to the performance of the learning algorithms. The meta-learning techniques were originally proposed and evaluated to the algorithm selection problem. In recent years, meta-learning was investigated to optimize parameters specifically for Support Vector Machines. However, meta-learning can be adopted as a more general strategy to optimize ANN parameters, which motivates new efforts in this research direction. Meta-learning treats the parameter selection is just another supervised learning task. In our work, we generated a base of meta-examples associated to 93 regression problems. Each meta-example was generated from a regression problem and stored: 16 features describing the problem (e.g., number of attributes and correlation among the problem's attributes) and the best number of nodes for this problem, empirically chosen from a range of possible values. This set of meta-examples was given as input to a meta-learner which was able to predict the best number of nodes for new problems based on their features. The experiments performed in this case study revealed satisfactory results.

Keywords – Meta-learning, parameter selection, neural network optimization, Multilayer Perceptron, Extreme Learning Machine

1 INTRODUCTION

Artificial Neural Networks (ANNs) have been widely used in the literature to solve problems in different areas of knowledge [1]. Despite its potential advantages, the performance of ANNs in a problem is very dependent on the values of its hyper-parameters, including training parameters and its architecture. The choice of adequate values for ANN parameters is dependent on the problem at hand (i.e. the learning task being solved) and it is commonly a difficult task. Consider, for instance, the task of choosing the number of nodes in the hidden layer of Multilayer Perceptron (MLP) networks. If the number of nodes is too small, it may not be enough for the network to learn the relationships present in the dataset (i.e. *underfitting*). On the other hand, if the number of nodes is too large, the network may memorize the training set (i.e. *overfitting*), thus harming the generalization power of the network [1]. The adequate choice of this parameter depends on the size of the learning dataset, the complexity of the problem, the amount of noise present in the data, among other aspects to consider.

As said, the optimization of ANN parameters is a complex problem. Different approaches have been proposed to solve it. A trial-and-error procedure is a very simple approach, however it can demand too much effort on the user and the results may be not satisfactory. There are more sophisticated and systematic approaches, such as the use of meta-heuristic search algorithms [2] [3] [4] [5]. In this approach, the best parameters are found during an optimization process. In this optimization process, the estimated performance of the ANN in the learning problem at hand corresponds to the objective function. The set of possible configurations of parameters corresponds in turn to the search space. The search algorithm iteratively evaluates different solutions in the search space, aiming to find a good configuration of parameters which optimizes the estimated performance (the objective function) of the ANN. A limitation of this approach is that the knowledge acquired in a problem is not used to help solving new problems. The search commonly starts at solutions randomly defined for a new problem, which may lead to a slower convergence of the optimization process.

An alternative approach to ANN parameter selection is the use of *meta-learning*, which was originally proposed for the algorithm selection task [6]. Meta-learning is an area that tries to extrapolate the knowledge acquired in past problems to solve new problems. Meta-learning can be defined as the process of automatic acquisition of knowledge that relates the performance of learning algorithms to the characteristics of problems [7]. The knowledge in meta-learning is acquired from a set of meta-examples. In general, each meta-example is associated to a learning problem and it is composed by: (1) meta-attributes that describe the problem (e.g., number of attributes, number of examples, class entropy,...) and (2) meta-label (the label that indicates

the algorithm that performed better in the problem) [8]. Several studies in meta-learning have been developed for selection of algorithms with success in different fields of application [6].

In recent years, meta-learning has been extrapolated to the task of parameter selection. In this context, each meta-example stores as meta-label the configuration of parameters with best performance in the learning problem associated to the meta-example. The meta-knowledge acquired from the set of meta-examples is used to predict which is the best configuration of parameters based on the characteristics of the learning problems at hand. The meta-learning techniques were adopted to optimize parameters of algorithms, more specifically, the parameters of Support Vector Machines (SVM) [9] [10] [11]. The good performance of meta-learning in these works motivates the use of meta-learning in other models of ANNs.

In the current work, we developed a case study that applies meta-learning to choose the number of hidden neurons for MLP networks with one hidden layer trained by ELM. We built a set of meta-examples from the evaluation of MLP networks in 93 regression problems, systematically varying the number of hidden layer neurons. Each meta-example is associated with a regression problem and stores: (1) 16 features describing the dataset of the problem; (2) the optimal number of neurons to the MLP in the problem, defined empirically. A learning algorithm adopted as meta-learner received this set of meta-examples as input and was used to predict the best number of neurons to new regression problems based on their characteristics. In our experiments, we evaluated three different algorithms as meta-learner: linear regression, k-NN and SVM. The case study validated the meta-learning as an alternative to optimize the number of neurons in the hidden layer. It was verified in the experiments that the characteristics of problems (meta-attributes) can be used to suggest an adequate number of neurons in the hidden layer of MLP networks trained by ELM.

The remaining of this paper is organized as follows. Section 2 is a brief review on the problem of optimizing ANNs. Following, Section 3 shows the basic concepts of meta-learning and its use for parameter selection. Next, the Section 4 shows the basic concepts of ELM. Then, in Section 5, the developed work is presented in detail. Finally, in Section 6, we present conclusions and future work.

2 NEURAL NETWORK OPTIMIZATION

ANNs have been successfully applied to different problems in the literature [12] [13] [14] [15]. The success of these algorithms in applications motivated the development of new models with different levels of complexity. However, the quality of a neural network model is strongly dependent on the definition of the values of its parameters (e.g., network type, number of layers, number of neurons, learning rates, among others). For a given problem, there is an specific set of parameters to be applied in the neural model which can lead to adequate results. Defining the values of these parameters is not easy since the number of possible configurations is usually very high. Moreover, there is a computational cost of evaluating how good a configuration is. This cost varies with the problem, the neural model and training algorithm considered. Obviously, an exhaustive search in parameter space is not feasible in practice.

There are several approaches to optimize the parameters of an ANN. In many cases, the optimization of parameters is done by a not systematic trial-and-error procedure by the user. This approach may not lead to good results if the user has little experience to guide the search process. In this context, different authors have dealt with the optimization of ANNs as an actual optimization task, using meta-heuristic search algorithms to systematically explore the space of possible configurations of parameters. Different algorithms have been used in this context such as Simulated Annealing, Tabu Search, Genetic Algorithms and Particle Swarm Optimization [2] [3] [4] [5], but the computational cost also use to be expensive.

An alternative way to optimize ANNs is the use of knowledge (empirical or theoretical). There is in fact a large amount of rules and heuristics that can be used to guide the choice of some parameters of specific models [16]. For instance, in [17], it is suggested that the number of hidden neurons of an MLP would be smaller than the number of inputs (i.e., number of attributes of the problem) and bigger than number of network outputs. Although these *rules of thumb* may be useful in some contexts, they are limited and not very extensible. In fact, it is generally difficult to use more complex features of the data to guide the parameter selection due to correlations among features and noise in the data. Moreover, it is difficult to elicit knowledge about more complex ANN models since experts are not always available. In this context, meta-learning is an alternative to automatically acquire knowledge that uses the features of data to recommend the best algorithms and parameters. This approach is less computationally expensive when compared with the search techniques. At the same time, it provides a systematic approach to acquiring knowledge that can be adopted for optimization of parameters in different types of ANNs.

3 META-LEARNING

Meta-learning can be defined as the process of automatic acquisition of knowledge which relates the performance of learning algorithms with the characteristics of the learning problems [7]. Figure 1 presents the meta-learning approach. The knowledge in meta-learning is acquired from a set of meta-examples generated from the evaluation of a pool of candidate algorithms on a number of learning datasets. Each meta-example consists of: (1) meta-attributes that characterizes a learning problem; and (2) a meta-label indicating the best candidate algorithm for the problem. The meta-attributes are, in general, statistics that describe the dataset, such number of examples, number of attributes, correlation between attributes, average entropy of attributes, among others [18] [19]. The meta-label, in turn, is in general a class label indicating the best algorithm for the problem, usually determined by an empirical evaluation (for instance, by a cross validation experiment). Given such meta-data, another learning algorithm is applied on the meta-level aiming to produce a model for selecting among the candidate algorithms for new problems based on their meta-attributes.

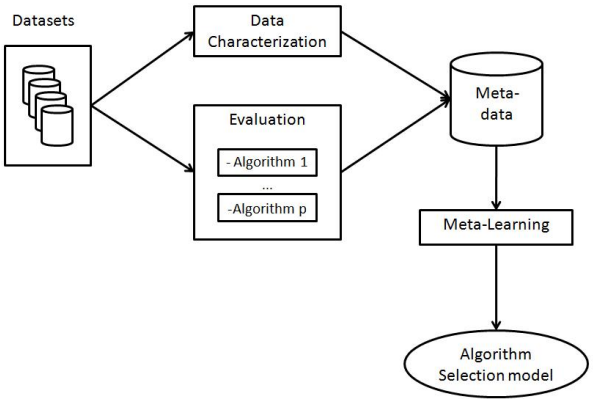

Figure 1: The meta-learning process for algorithm selection.

Meta-learning has been extensively evaluated for selection algorithms in different context. However, recently meta-learning has also been applied to optimization of parameters (instead of selecting algorithm). In fact, the application of meta-learning for parameter selection is straightforward. The set of possible configurations of parameters becomes the pool of candidates to be chosen by the meta-learner. Hence, the meta-learning in this case can select the best parameters given the characteristics of a problem. Different from simple heuristics and the rules of thumb, the knowledge in meta-learning is acquired by a systematic learning process which takes advantage of the empirical experience implicitly stored in the meta-examples.

Previous work using meta-learning for parameter optimization was applied to Support Vector Machines (SVM). In [9], the authors try to predict the best kernel function for classification problems. In this case, the meta-learning task was a classification task, in which the meta-label was the best kernel function among five candidates. In this work, a set of meta-examples were built from 112 classification datasets. In [10], meta-learning was used to select the width parameter of the Gaussian kernel for regression problems. Although the width parameter of Gaussian kernel is continuous, the problem was treated as a classification task, with the meta-label variable assuming 11 different discrete values. In this study, a set of 42 meta-examples was generated from 42 regression datasets. In [11], meta-learning was used to optimize two SVM parameters: the parameter γ of the RBF kernel and the regularization constant C. A set of 40 meta-examples was produced from different regression problems. An instance-based learner was used in the meta-level to recommend the SVM parameters among a set of 399 candidate combinations of γ and C.

The promising results obtained by meta-learning in optimizing SVMs motivate its evaluation in other ANN models. In fact, there are many opportunities in the application of meta-learning to ANN optimization due to the variety of ANN models as well as the diversity of parameters to be chosen for each model. However, few relevant work can be pointed out in the literature in this research direction (as those ones mentioned above). This paper is a case study that introduces the idea of applying meta-learning in the optimization ANNs. The network chosen in our case study was the MLP with one hidden layer trained by ELM. The parameter selection task was to define the number of hidden nodes. Different from previous work that adopted meta-learning for parameter selection, in our work, we treated our specific parameter selection task as a regression problem and consequently we evaluated regression algorithms in the meta-level. Hence, we expect to introduce new insights that could be useful in other applications.

4 Extreme Learning Machine

Extreme Learning Machine is a very fast training algorithm that works on MLP neural networks with one hidden layer. It works in two big steps [20]:

1. assigns randomly values for the weights and biases of the input layer nodes; and
2. find the weights of the output layer analytically.

The ELM do not need a validation dataset and has only two parameters: number of hidden nodes end activation function. The Equation 1 shows how the ELM works in regression problem. The left side of Equation 1 determine the output of ANN (y_j) and the right side the correct answered (target - t), where M is the number of hidden nodes, β represent the weights of output layer, g is the activation function, w represent the weights of input layer, x is the network input, b is the bias of hidden layer, t is the target and N is the number of examples in the dataset. Thus, the ELM algorithm try to find analytically the values of β where $|y_j - t_j|$ is minimized for each example $j \in N$ [21].

$$\sum_{i=1}^{M} \beta_i g(w_i.x_j + b_i) = t_j, j = 1, 2, ..., N. \quad (1)$$

All the N equation represented in the Equation 1 can be written compactly as Equation 2.

$$H\beta = T \tag{2}$$

where:
$H(w_1, ..., w_M, b_1, ..., b_M, x_1, ..., x_N)$

$$= \begin{bmatrix} g(w_1.x_1 + b_1) & ... & g(w_M.x_1 + b_M) \\ ... & ... & ... \\ g(w_1.x_N + b_1) & ... & g(w_M.x_N + b_M) \end{bmatrix},$$

$$\beta = \begin{bmatrix} \beta_1^T \\ ... \\ \beta_M^T \end{bmatrix} and$$

$$T = \begin{bmatrix} T_1^T \\ ... \\ T_M^T \end{bmatrix}.$$

The matrix H are determined by the weight and bias chosen randomly, T is given by the problem, and we wish to find the β where the Equation 1 is true for every $j \in N$. We can rewrite the Equation 2 as Equation 3. So, in this linear system, in order to find β is necessary to find the inverse of the matrix H. If the number of hidden nodes is equal to the number of distinct training samples, $(M = N)$, the matrix H would be square and invertible. However, in most cases the number of hidden nodes is much less than number of training examples. So, the solution found by the author was to use the Moore-Penrose generalized inverse $(H^\dagger)$, a kind of inversion that can be applied in any matrix [21]. It is represented in the Equation 4.

$$\beta = TH^{-1} \tag{3}$$

$$\beta = TH^\dagger \tag{4}$$

More details on the operation of ELM can be found in [21].

5 META-LEARNING FOR OPTIMIZATION OF ARTIFICIAL NEURAL NETWORK

In this work, we investigate the use of meta-learning to chose the number of hidden nodes in MLP networks trained by Extreme Learning Machine algorithm. The construction of the meta-dataset usually has high computational cost. After that it is necessary to train the meta-learner using the meta-examples that composed the meta-dataset. However, the computational use cost of the model developed is very low.

The ELM is a training algorithm originally used to train MLP networks with one hidden layer. Its main advantage over other classics training algorithms as Backpropagation is the training speed, which is extremely fast. Another advantage is that ELM has only two parameters: number of hidden nodes and activation function, which have to be infinitely differentiable. More information about ELM can be found in [20] and [21]. These advantages motivated the use of ELM for this case study, because it improved more speed in the build of meta-dataset.

The parameter selected for the case study was the number of hidden nodes in MLP networks. Thus, the activation function was fixed and all values of hidden nodes in the range $[1; 300]$ were tested for 93 different regression datasets. We also extracted the meta-attributes of each regression datasets. Thus, it was possible to build a meta-dataset with 93 meta-examples. Finally, some regression algorithms were tested as options for meta-learner.

5.1 Meta-examples

In order to build a set of meta-examples, it is necessary to collect a set of datasets (see Section 3). In our work, the set of meta-examples was produced from 93 regression datasets selected from WEKA website [1]. We selected regression datasets with more than 100 training examples and with the target attribute assuming more than 10 distinct values. Each regression dataset had their attributes normalized. The label (or target) was normalized between zero and one. Differently, the continuous input variables were normalized between -1 and 1. Symbolic attributes were binarized, in such a way that an attribute with k different values was represented by $k - 1$ bits zeros and a bit one.

In our work, each meta-example was created from a different regression problem and stored: (1) 16 meta-attributes that describe the problem and (2) the meta-label (optimal number of neurons used by MLP to solve the problem). These two aspects will be defined below.

[1] http://www.cs.waikato.ac.nz/ml/weka/

5.1.1 Meta-attributes

The meta-attributes were chosen in our work aiming to reflect the complexity of the regression problem. This is important since there is a relationship between the complexity of a problem and the number of neurons needed to learn the examples of the dataset's problem. In our study we used 16 meta-attributes. The choice of meta-attributes was based on earlier meta-learning studies which work with regression problems [10] [9]:

- Number of attributes;
- Number of examples;
- Mean skewness of attributes;
- Mean kurtosis of attributes;
- Mean absolute skewness of attributes;
- Maximum mean distance between each target value and its two neighbors (sorted by target value);
  - The goal of that meta-attribute is to measure the variability (or dispersion) of the target attribute.
- Maximum correlation between continuous attributes and the target;
- Number of continuous attributes with outliers;
- Proportion of continuous attributes with outliers;
- Existence of outliers in the target attribute: 1 if it does, 0 otherwise (calculated for continuous attributes);
- Target outlier value;
  - The ratio between the standard deviation and the standard deviation of alpha trimmed mean. If the standard deviation is 0, then the ratio is set to 1.
- R^2 coefficient of multiple linear regression (only numerical attributes are used);
- R^2 coefficient of multiple linear regression (symbolic attributes are binarized);
- Coefficient of variation: 0 if not sparse, 1 if sparse and 2 if extremely sparse;
- Absolute coefficient of variation: 0 if not sparse, 1 if sparse and 2 if extremely sparse;
- Check if the standard deviation is larger than mean.

Some considerations can be made concerning the above meta-attributes. The number of attributes and the number of examples, for instance, indicate the dimensionality of the regression problem. The meta-attributes based on skewness and kurtosis measure how far the regression dataset departs from normality. Correlation to the target attribute expresses the relevance of the predictor attributes in a regression dataset. The meta-attributes based on identifying outliers can be seen as measures of noise in a dataset. The R^2 coefficients of the linear regression can be seen as measures of problem complexity (values closer to 1 indicate linear relationships in the regression dataset). Coefficients of variation measure dispersion in the dataset. Obviously some of these meta-attributes may be correlated to each other at some degree. Also, there is a large number of other statistics that could be used to describe regression datasets. In future work, we intend to augment the number of meta-attributes in order to improve the meta-learning results.

5.1.2 Meta-label

In order to build a labelled meta-example in our case study, we have to identify the best number of hidden nodes for the MLP trained by ELM at each regression problem. For this we empirically evaluated the MLP varying the number of hidden nodes in the range $[1; 300]$ (all the number of hidden nodes in this range were evaluated). Then we chose the best one in terms of lowest error achieved.

In our study, we adopted the Extreme Learning Machine (ELM) algorithm for training the MLP networks. The ELM does not use validation dataset in their training. Thus, each dataset in our case study was divided into two sets: a training set (70% of the regression dataset) and test set (30% of the regression dataset). The training set was used to adjust the weights of MLP. Following, the test set was used to evaluate the accuracy of the trained MLP. Each experiment was repeated 10 times with different initial weights. All simulations were performed with the MATLAB 7.9 implementation of ELM provided by the ELM's author [2]. The best number of hidden nodes was selected for each problem according to the lowest average error in the test dataset over the

[2] http://www.ntu.edu.sg/home/egbhuang/

Table 1: Examples of meta-examples developed for the meta-learning of the ideal number of hidden neurons for a given problem.

Meta-attributes				**Meta-label**
Number of examples	Number of attributes	Number of outliers	...	Hidden Nodes
100	3	0	...	4
209	6	5	...	32
950	9	0	...	215
...	...	...	...	...
13750	40	0	...	142

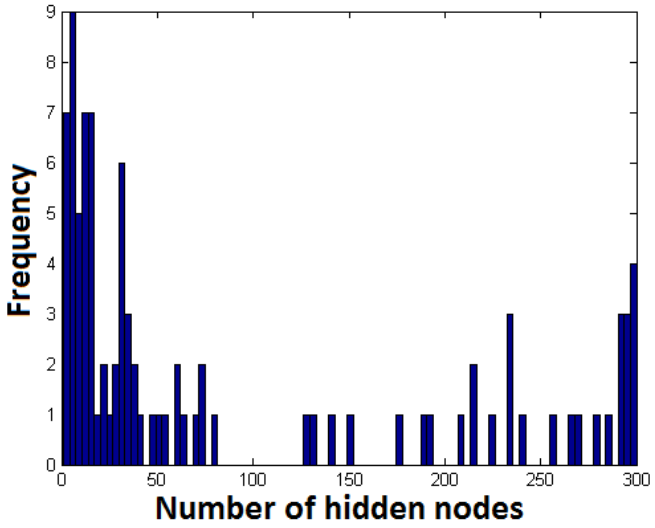

Figure 2: Histogram showing the distribution of the best options of hidden nodes for 93 regression problems.

10 repetitions. The error measure adopted in our work was the *Root Mean Square Error* (RMSE), presented in Equation 5, in which n is the number of examples in the test dataset and P_j and T_j are, respectively, the network output and the correct answer (target). Table 1 illustrates the final set of meta-examples built in our work.

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(P_j - T_j)^2} \tag{5}$$

Figure 2 shows a histogram with the distribution of the meta-label. The diversity of values indicates the complexity of the meta-learning task investigated (select the optimal number of hidden nodes). Once developed the meta-dataset, the next step is to submit it to the meta-learner in order to evaluate its accuracy.

5.2 Meta-learner

In this case study, the meta-learning task is to predict the optimum number of nodes in the MLP hidden layer trained by ELM based on the characteristics of the given problems. The optimum number of nodes in the hidden layer in this case is a number between 1 and 300. Thus, any regression algorithm can be used as meta-learner. In this study, we evaluated different regression algorithms as meta-learners which were previously in the literature [22]. All of them are implemented in the WEKA environment:

- 1-NN: a special case of instance-based learning algorithms [23];
- M5 algorithm: which was proposed by Quinlan [24] to induce regression trees;
- Linear regression model;
- SVM: using two different kernels (polynomial and RBF).

Table 2: Measure (RAE) and the correlation coefficient for five different meta-learners to predict the optimal number of nodes in the hidden layer in MLP networks.

Method	RAE	Correlation
Linear Regression	68,11%	0,66
SVM (Polynomial)	60,25%	0,66
SVM (RBF)	**52,84%**	**0,72**
1-NN	66,26%	0,51
M5	61,73%	0,62

All meta-learners were applied using the default parameters suggested in the WEKA environment, excepting the SVM with RBF kernel (for which, we tested two other parameter values based on the results obtained in [22]).

5.3 Experiments and results

In this section, we evaluate the performance of the meta-learners mentioned in the previous section. In order to evaluate the meta-learners, we adopted a leave-one-out procedure with the 93 meta-examples. In this procedure, each meta-example is used once to test the accuracy of meta-learner, while all others are used in the training process. The error measure adopted to evaluate the performance of meta-learning model was the Relative Absolute Error (RAE).

The RAE measure has high values (closer to 100%) when the meta-learner has an accuracy similar to the performance obtained by the mean of the meta-label, that is the mean of the best number of hidden nodes in all of 93 datasets (Figure 2). In this case, values closer to 100% indicate that the regression algorithm used as meta-learner is not useful, with a performance similar to an naive predictor. Besides the RAE, we also observed the correlation between the meta-learners' predictions and the actual meta-label.

The results obtained by the meta-learners are summarized in Table 2. All meta-learners had RAE lower than 100% (in fact, lower or equal than 68.11%). The correlation observed between the responses of the meta-learners ranged from 0.51 to 0.72. These results indicate that in fact there are regularities in the meta-attributes of the problems that can be used to predict the number of hidden nodes, i.e., that there is knowledge to be acquired from the meta-examples. The SVM with the RBF kernel function had the highest accuracy and correlation among all the meta-learners evaluated (RAE = 52.85% and correlation = 0.72). Similarly to previous meta-learning studies, such as [22], the SVM was the best option for meta-learning.

6 CONCLUSION

In this work, we investigate the use of meta-learning to chose the number of hidden nodes in MLP networks trained by Extreme Learning Machine algorithm. A meta-dataset with 93 meta-examples was built and different meta-learners were applied to solve the proposed meta-learning task. The RAE values found by the meta-learners achieved adequate values as well as the correlation between the meta-learners' predictions and the true values of the meta-label. The lowest error in the meta-learning task, as in previous work [22], was achieved by the SVM meta-learner (52.85%). In absolute terms, this error is still high, which reflects the complexity of the proposed meta-learning task.

Future experiments performed using search algorithms could eventually result on more accurate parameters. However, the cost to obtain a response from a meta-learning model is usually lower than the use of search algorithms. There is also an intermediate approach in which meta-learning is applied to provide initial parameter configurations to be refined by a search algorithm (as performed in [11]). In this case, the search algorithm would start from a promising region in the search space (provided by meta-learning) and could find satisfactory solutions faster. This hybrid approach will be investigated in future work.

We can point out other studies that can improve the precision obtained by meta-learning. The selection of meta-attributes was not well explored in the current work. In addition, other meta-learners and sets of meta-examples can be built in order to improve the results. Finally, meta-learning can be applied for optimizing parameters using different ANN models and training algorithms.

References

[1] A. d. P. Braga, A. P. d. L. F. Carvalho and T. B. Ludermir. *Redes Neurais Artificiais: Teoria e Aplicações*. LTC, Rio de Janeiro, 2007.

[2] C. Zanchettin, T. B. Ludermir and L. M. Almeida. "Hybrid Training Method for MLP: Optimization of Architecture and Training". *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 41, pp. 1–13, 2011.

[3] I. El-Henawy, A. Kamal, H. Abdelbary and A. Abas. "Predicting stock index using neural network combined with evolutionary computation methods". In *Informatics and Systems (INFOS), 2010 The 7th International Conference on*, pp. 1–6, 2010.

[4] Z. Y. Bin, L. L. Zhong and Z. Y. Ming. "Study on network flow prediction model based on particle swarm optimization algorithm and RBF neural network". In *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, volume 2, pp. 302–306, 2010.

[5] H. Li, M. Cai and Z.-Y. Xia. "Algorithm Research of RBF Neural Network Based on Improved PSO". In *Multimedia Communications (Mediacom), 2010 International Conference on*, pp. 87–89, 2010.

[6] K. A. Smith-Miles. "Cross-disciplinary perspectives on meta-learning for algorithm selection". *ACM Comput. Surv.*, vol. 41, pp. 1–25, 2008.

[7] C. Giraud-Carrier, R. Vilalta and P. Brazdil. "Introduction to the Special Issue on Meta-Learning". *Machine Learning.*, vol. 54, pp. 187–193, 2004.

[8] R. B. Prudêncio and T. B. Ludermir. "Active Generation of Training Examples in Meta-Regression". In *Proceedings of the 19th International Conference on Artificial Neural Networks: Part I*, ICANN '09, pp. 30–39. Springer-Verlag, 2009.

[9] S. Alia and K. A. Smith-Miles. "A meta-learning approach to automatic kernel selection for support vector machines". *Neurocomputing*, vol. 70, pp. 173–186, 2006.

[10] C. Soares, P. B. Brazdil and P. Kuba. "A Meta-Learning Method to Select the Kernel Width in Support Vector Regression". *Machine Learning*, vol. 54, pp. 195–209, 2004.

[11] T. Gomes, R. Prudrêandncio, C. Soares, A. Rossi and A. Carvalho. "Combining Meta-learning and Search Techniques to SVM Parameter Selection". In *Neural Networks (SBRN), 2010 Eleventh Brazilian Symposium on*, pp. 79–84, oct. 2010.

[12] N. Harun, S. Dlay and W. Woo. "Performance of keystroke biometrics authentication system using Multilayer Perceptron neural network (MLP NN)". In *Communication Systems Networks and Digital Signal Processing (CSNDSP), 2010 7th International Symposium on*, pp. 711–714, 2010.

[13] A. Gomperts, A. Ukil and F. Zurfluh. "Development and Implementation of Parameterized FPGA-Based General Purpose Neural Networks for Online Applications". *Industrial Informatics, IEEE Transactions on*, vol. 7, no. 1, pp. 78–89, 2011.

[14] J. Patra and K. Chua. "Neural network based drug design for diabetes mellitus using QSAR with 2D and 3D descriptors". In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pp. 1–8, 2010.

[15] R. Achkar and C. Nasr. "Real Time Application of an AMB Using MLP: Study of Robustness". In *Computational Intelligence, Modelling and Simulation (CIMSiM), 2010 Second International Conference on*, pp. 9–14, 2010.

[16] W. S. Sarle. "Neural Network FAQ, part 1 of 7: Introduction". ftp://ftp.sas.com/pub/neural/FAQ.html, 1997.

[17] A. Blum. *Neural networks in C++*. John Wiley e Sons, New York, 1992.

[18] P. Brazdil, C. Soares and J. da Costa. "Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results". *Machine Learning*, vol. 50, pp. 251–277, 2003.

[19] A. Kalousis, J. Gama and M. Hilário. "On Data and Algorithms: Understanding Inductive Performance". *Machine Learning*, vol. 54, no. 3, pp. 275–312, 2004.

[20] G. Huang, Q. Zhua and C. Siew. "Extreme Learning Machine: A New Learning Scheme of Feedforward Neural Networks". *International Joint Conference on Neural Networks*, vol. 2, pp. 985–990, 2004.

[21] G. Huang, Q. Zhua and C. Siew. "Extreme Learning Machine: Theory and applications". *Neurocomputing*, vol. 70, pp. 489–501, 2006.

[22] S. B. Guerra, R. B. C. Prudêncio and T. B. Ludermir. "Predicting the Performance of Learning Algorithms Using Support Vector Machines as Meta-regressors". In *Proceedings of the 18th international conference on Artificial Neural Networks, Part I*, ICANN '08, pp. 523–532. Springer-Verlag, 2008.

[23] D. W. Aha, D. Kibler and M. K. Albert. "Instance-based learning algorithms". *Machine Learning*, vol. 6, pp. 37–66, 1991.

[24] J. R. Quinlan. "Learning With Continuous Classes". In *Proceedings of the Australian Joint Conference on Artificial Intelligence*, pp. 343–348. World Scientific, 1992.