2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

A modular neural network architecture that selects a different set of features per module

Diogo S. Severo, Everson Veríssimo, George D. C. Cavalcanti and Tsang Ing Ren

***Abstract*—Modular Neural Network (MNN) divides a problem into smaller and easier sub-problems, and each sub-problem is solved by a neural network called expert. In previous MNN architectures, all experts used the same set of features. This work proposes a modular neural network architecture in which a specialized set of features is selected per expert. As each expert deals with a different sub-problem, it is expected an improvement in the accuracy rate when different and specialized features are selected per expert. The feature selection procedure is an optimization method based on the binary particle swarm optimization. Experimental results over public datasets show that the proposed modular neural network obtains better accuracy rates than literature MNNs.**

I. Introduction

Artificial neural networks are inspired by the way the human brain performs a task, however they lack modularization. In contrast, our brain has distinct specialized areas that are responsible for specific tasks, such as: vision, audition, and speech [1]. Modular neural networks (MNNs) aim to construct specialized neural networks using the divide-to-conquer approach [2][3][4].

In a supervised scenario, where C represents the set of all possible classes, a modular neural network works as follows. First, the original problem is divided into smaller and simpler sub-problems and these sub-problems are distributed into different modules. Each sub-problem contains only a subset C' of the whole set of classes C ($C' \subseteq C$). After, each module is responsible for solving a specific sub-problem by training a classifier, called expert. So each expert is a specialized classifier that responds only for a subset of the whole possible classes of the problem at hand.

Modular Neural Networks take advantage of modularization to overcome the accuracy rate of single neural networks when dealing with complex problems. This modularization is performed by Task Decomposition methods [5], [6], [7], [8] that aim to divide a complex problem into easier sub-problems. After decomposition, each module uses the whole set of features to train each expert. However, it is expected that different modules require different subsets of the original features to better perform their task. Thus, it is important to choose the features that best preserve the discriminant information among classes per module.

This paper proposes a modular neural network in which a different subset of the original features is selected per module. The proposed MNN architecture is based on the Pattern Distributor (PD) architecture proposed by Guan et al. [10]. As well as other MNN architectures, the PD architecture divides the problem into sub-problems where each one is treated by a different module. The main difference between PD architecture and other MNN architectures is the presence of a special module called distributor. The distributor module selects the expert that is responsible for the final classification of a given query pattern. Thus, the selection of the winner module is made before classification unlike other MNN architectures where the selection of the winner module is usually made after classification of the query pattern.

This paper is organized as follows. Section II describes the proposed method that selects a different set of features per module. Section III gives an overview about particle swarm optimization (PSO) which is the optimization approach used to select the subset of features. Section IV presents the experimental study and Section V draws the final remarks.

II. The proposed architecture

Figure 1 shows the training architecture of the proposed approach. It is composed of three main parts: Task Decomposition, Pattern Distributor Training and Training Expert$_k$. The aim of the Task Decomposition is to split the database $\mathbf{G}$ into k disjoint subsets $\{G_1, G_2, \ldots, G_k\}$. The Pattern Distributor PD_{mod} is a classifier that returns one of the possible subsets $\{1, 2, \ldots, k\}$ given a pattern $x \in \mathbf{G}$. The first step of the *Training Expert*$_k$ phase is to select the best features M_k in the G_k dataset. These best features are used to train the expert E_k.

1) *Task Decomposition*: The original dataset is divided into the modules using a task decomposition technique. The architecture is flexible enough to admit different task decomposition methods. The choice of the algorithm is a crucial factor in the MNN since a bad decomposition can lead to a bad system performance. Given a training dataset $\{(x_1, c_1), (x_2, c_2), \ldots, (x_n, c_n)\}$ where x_i is a pattern and c_i is its respective class, all entries are used by the decomposition technique to find the number of groups represented by k. Ended the decomposition, a set of data groups is returned: $G = \{G_1, G_2, \ldots, G_k\}$

2) *Training of the PD Module*: The patterns from all groups are used to train the pattern distributor module (PD). The patterns in the groups have their classes relabeled to a new class that indicates the group where they come from. Thus, the patterns from G_1 have their classes relabeled to 1, the patterns from G_2 have their classes relabeled to 2 and so on until the group G_k. Afterwards, the PD module is trained to decide to which group a pattern belongs. The distributor module already trained is represented by PD_{mod} (Figure 1).

Diogo S. Severo, Everson Veríssimo, George D. C. Cavalcanti and Tsang Ing Ren are with the Centro de Informática, Universidade Federal de Pernambuco, Brazil, site: http://www.cin.ufpe.br/∼viisar (email: {dss2, evs2, gdcc, tir}@cin.ufpe.br).

This work was supported by Brazilian agencies: CNPq, Capes, and Facepe.

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

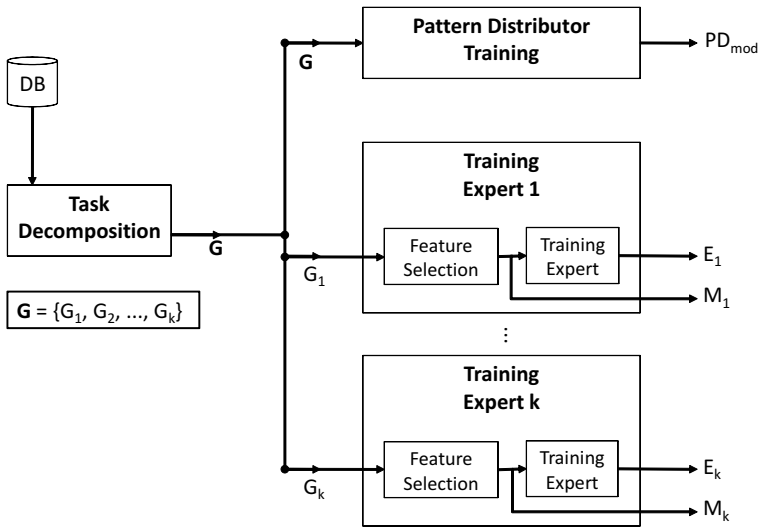


Fig. 1: Proposed architecture: Training phase. The database $\mathbf{G}$ is composed of k disjoint parts. The output of this phase is composed of: one Pattern Distributor classifier PD_{mod}; k experts E_k and the features selected M_k per expert

3) *Feature Selection*: The best features are selected per group G_k and this procedure aims to preserve the discriminant information required to classify the patterns of each group. An optimization procedure is used to this task and the output is a binary vector (Figure 2) where each position represents a feature; "1" means presence and "0" means absence of the respective feature. After the definition of the binary vector, all features that have value equal to "0" are removed. This dimensionality reduction procedure can be performed in parallel because the modules are independent.

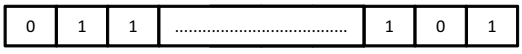


Fig. 2: Particle coded as a binary vector

4) *Training of the Modules*: After performing the feature selection, the new patterns of each module are used to train their respective experts (single NNs). Thus, the *i*-th module uses only the patterns that belong to group G_i with their original classes to train the expert E_i. If a group has only one class, it is not necessary to train a classifier, since only one output is possible. Otherwise, a classifier is trained to discriminate patterns among the existing classes. As well as the feature selection procedure, the training of the modules can be performed in parallel since the modules are independent. At the end of the training procedure, the system returns the experts trained (E_1, E_2, ..., E_k) as well as their masks (M_1, M_2, ..., M_k) as shown in Figure 1.

After training, the unseen patterns (x_{query}) are classified by the system. First, the PD module selects the proper expert which in turn applies the mask to the input pattern to filter the features and then the expert gives the predicted class for the input pattern. Depending on the decomposition technique used, the results are non-redundant groups, i.e., patterns from a particular class are present only in one group.

Thus, if the PD module makes a mistake, the expert selected is not able to fix the problem and the error is propagated. This test procedure of the proposed approach is shown in Figure 3.

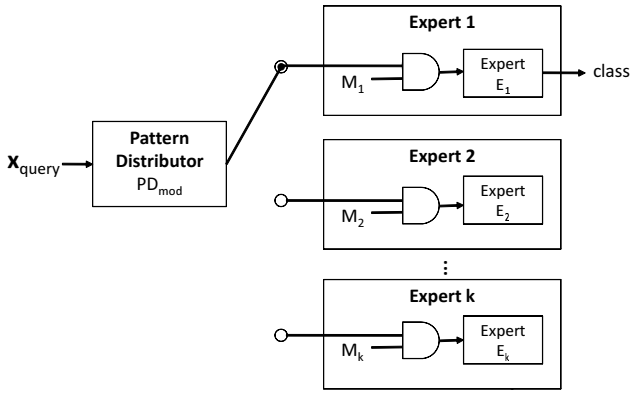


Fig. 3: Proposed architecture: Test phase

III. Particle Swarm Optimization

Particle Swarm Optimization was developed originally by Kennedy and Eberhart [12] in 1995. The idea of the algorithm is to simulate the social behavior of animals looking for resources, for example, birds in a flock and fish in a school. Basically, the standard PSO is composed by three vectors of *n* components. The size *n* is determined by the search space dimensionality. Therefore, for each dimension *i* of a particle *p*, we have:

(i) x_i, the current position.
(ii) v_i, the current velocity.
(iii) pb_i, the best position visited by that particle.

During the iterative process, the particles of the swarm search for the best solution for the problem exploring the search space. For that purpose, each particle is guided by the

combination of the best position reached by the particle (pb_i - cognitive factor) and the best position reached by the whole swarm (gb_i - social factor) in a period of time *t*. The velocity and position of each particle are updated as follows:

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (1)$$

$$v_i(t+1) = v_i(t) + c_1 r_1[pb_i(t) - x_i(t)] + c_2 r_2[gb_i(t) - x_i(t)] \quad (2)$$

In equation 2, the variables r_1 and r_2 are random numbers in the interval [0,1] and c_1 and c_2 are acceleration coefficients with fixed and equal values, generally in the interval [0,2] [13]. Shi and Eberhart [13] introduced the inertia weight (w) in the velocity update rule (equation 2) to control the degree of exploration of the search space and generally its value decreases linearly from 0.9 to 0.4. The new velocity update rule becomes:

$$v_i(t+1) = wv_i(t) + c_1 r_1[pb_i(t) - x_i(t)] + c_2 r_2[gb_i(t) - x_i(t)] \quad (3)$$

In this work, we use the IBPSO (*Improved Binary Particle Swarm Optimization*) algorithm [9] that is based on the BPSO (*Binary Particle Swarm Optimization*) algorithm [11] which in turn is based on the well-known PSO [12] defined above.

PSO deals with continuous spaces. However, there are problems that are represented in a discrete feature space. With this in mind, Kennedy and Eberhart proposed the binary PSO (BPSO) [11], i.e., a PSO approach that can be applied to discrete problems. One of these discrete problems is feature selection. As an optimization problem, a possible solution to the problem of feature selection can be seen as a binary vector whose length is equal to the size of the feature space of a given problem. Therefore, bit of value 0 in the binary vector indicates the absence (non-selected) of the feature, while a bit of value 1 reveals the presence (selected) of the feature in the subset of features to be returned by BPSO algorithm.

During the search for the best solution, the algorithm can be trapped in a local optimum and the search can be restricted to this poor region. To deal with this problem, Chuang et al. [9] proposed the IBPSO method where *gbest* (best solution found by the whole swarm) is reset to zero if its value does not change after three iterations. This is the only difference between the BPSO and IBPSO algorithms. The meaning of reseting *gbest* is that local optimum may be avoided, better results and a lower number of features are expected.

IV. Experimental study

A modular neural network based on pattern distributor architecture was developed using crosstalk tables (CT-PD) and genetic algorithms (GA-PD) as task decomposition methods [10]. These two MNNs were compared with the proposed approach. In addition to these experiments, a single NN with and without the feature selection procedure was also used in the comparison procedure.

A. Datasets and Experimental Methodology

Many experiments were performed in order to demonstrate the accuracy of the proposed technique. Eight benchmarks were used for this purpose. Except for vowel [14], all the other datasets were obtained from the UCI repository [15]. A brief description of them can be seen in Table I. In this table, we can see the number of patterns, the number of features as well as the number of classes of each dataset.

TABLE I: Datasets Description

Datasets	Patterns	Features	Classes
Balance	625	4	3
Ionosphere	351	34	2
Iris	150	4	3
Libras	360	91	15
Liver	345	6	2
Semeion	1593	256	10
Vehicle	846	18	4
Vowel	990	10	11

All classifiers used in the proposed approach are Multilayer Perceptrons with fixed values of learning rate and *momentum*, 0.1 and 0.5 respectively. The *maximum* number of iterations for all neural networks was set to 1000. The number of hidden neurons were defined empirically in the range [10,180], for this purpose a single NN had its number of hidden neurons changed from 10 to 180 and was tested on the original datasets. The neural network that presented the highest accuracy classification rate provided the number of hidden neurons to be used in later experiments. Table II shows the number of hidden neurons used for each dataset as well as the number of modules found by the task decomposition methods.

TABLE II: Number of hidden neurons and number of modules per task decomposition procedure (GA - Genetic Algorithm; CT - Crosstalk Tables)

Datasets	Number of neurons	GA	CT
Balance	70	2	3
Ionosphere	170	2	2
Iris	20	2	2
Libras	90	3	12
Liver	10	2	2
Semeion	90	2	7
Vehicle	50	2	4
Vowel	170	2	5

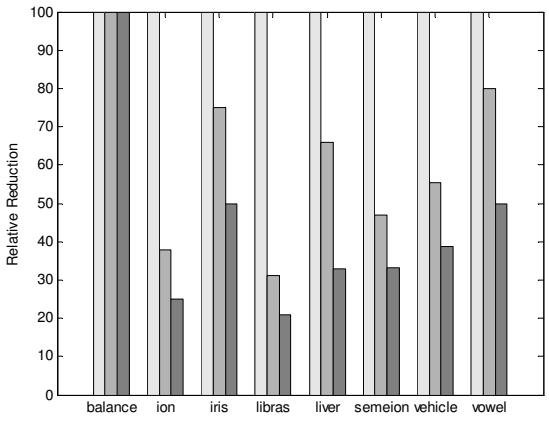
It was used 10-fold cross validation for all datasets: one fold for test and nine folds for the task decomposition. After creating the modules, 80% of the patterns of each group were used for training the correspondent expert (MLP Neural Network) and 20% were used to validate the training procedure of the MLP Neural Network. The IBPSO algorithm (described in Section III) is used to select the best features per expert.

B. Results

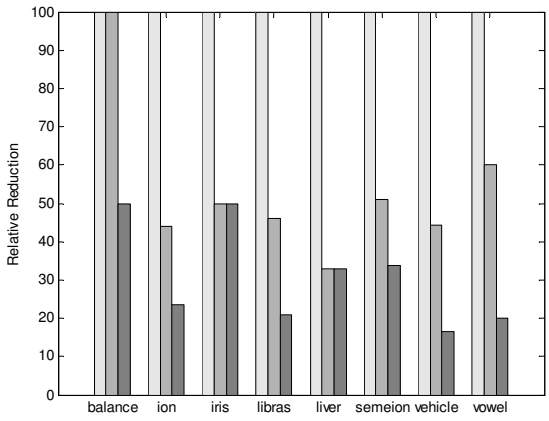
In a feature selection task, the goal is to select a small feature subset that is capable of retaining the discriminant information present in the original dataset. Therefore, the main results to be seen in this work are the dimensionality reduction obtained for each module of the proposed approach as well as the accuracy rate obtained in each dataset. Figure 4 presents the original dimensionality *d*, the *minimum* (d_{min}) and the *maximum* (d_{max}) dimensions after the feature selection in terms of percentage. Table III shows the accuracy rate and standard deviation for all datasets. The best performances are

TABLE III: Accuracy classification rate (%) and standard deviation in parenthesis

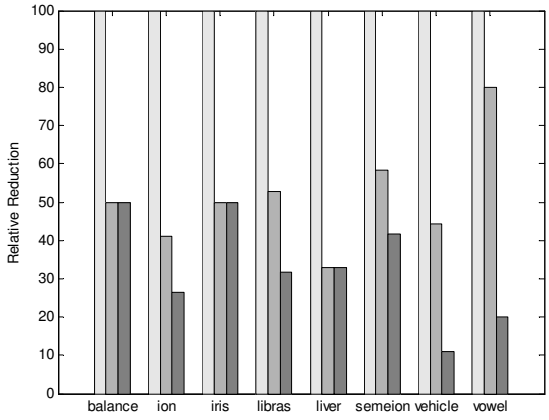
	Single NN		Modular Neural Networks			
Datasets	**NN**	**NN + IBPSO**	**GA**	**Proposed using GA**	**CT**	**Proposed using CT**
Balance	95.57 (2.82)	95.85 (2.38)	90.74 (3.92)	86.06 (7.85)	**96.20** (2.81)	**96.20** (2.81)
Ionosphere	92.87 (3.63)	91.17 (3.89)	**94.87** (2.95)	**94.87** (2.95)	94.31 (4.84)	94.31 (4.84)
Iris	95.33 (4.50)	91.33 (6.32)	**98.00** (3.22)	97.33 (3.44)	94.00 (10.16)	86.00 (14.89)
Libras	65.00 (23.36)	69.17 (9.75)	51.94 (29.34)	60.83 (26.53)	72.22 (9.07)	**74.44** (9.05)
Liver	67.91 (11.34)	57.88 (7.54)	**69.09** (9.68)	**69.09** (9.68)	62.14 (10.79)	62.14 (10.79)
Semeion	91.34 (1.21)	88.70 (1.66)	90.84 (2.44)	89.02 (2.02)	**91.40** (2.31)	**91.40** (2.40)
Vehicle	50.50 (6.43)	51.33 (12.26)	49.46 (6.42)	52.05 (9.20)	**53.13** (7.78)	**53.13** (7.78)
Vowel	32.22 (4.91)	31.11 (4.33)	34.75 (4.69)	31.31 (4.57)	36.67 (7.87)	**38.08** (4.44)



(a) Single Neural Network + IBPSO



(b) Proposed Architecture using the GA task decomposition algorithm



(c) Proposed Architecture using the CT task decomposition algorithm

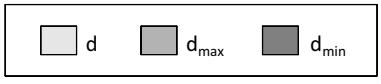


Fig. 4: Comparison of the dimensionality reduction percentage. d, d_{min}, and d_{max} are the original, minimum and maximum dimensionalities, respectively

in bold. The number of modules found by the decomposition techniques (CT and GA) that provided the best results is shown in Table II along with the hidden neurons.

Except for Iris, the proposed approach was better than the single neural networks (with and without feature selection) and better than the modular neural networks without feature selection (PD). In Table III, we can notice that the idea of applying feature selection to the individual modules from an MNN was effective, since for seven databases out of eight, the accuracy classification rates were higher with the feature space dimension reduced. For Libras and Vowel datasets, for example, the proposed approach using CT obtained the best performance when compared with other implemented methods. For Libras, only 48 features at most were used by each module instead of the original size which is 91 features. This shows that not all features are essential to perform this classification task.

For Balance, Ionosphere, Liver, Semeion and Vehicle datasets, the accuracy classification rates achieved by the PD approach (without feature selection) were similar to the proposed approach. However, the amount of features used by the proposed approach was much inferior when compared with the PD amount. In the experiments with Libras, for example, the proposed approach obtained similar performance when compared with the modular neural networks without feature selection (PD) using only 46% of the features present in the original dataset while for Ionosphere dataset only 41% of the features were used. These results are consistent with the purpose of feature selection tasks since feature selection aims to choose a small number of relevant features to achieve similar or even better results. Although the single NNs did not achieve better results than PD and the proposed approach, the use of feature selection was useful too. For some datasets, the biggest reduction was obtained by single NNs.

V. Conclusion

This paper proposed a modular neural architecture in which a different set of features is selected per module. The proposed approach discarded irrelevant features per module and this elimination improved the classification performance in terms of accuracy when compared with modular neural networks without feature selection. Given a smaller set of features, single neural networks had their storage and computational costs decreased since it is necessary less neurons in the input layer and consequently less connections between the input layer and the hidden layer. With less connections, the neural networks size decreased as well as the interference in weight-updating directions during the weight-update processing [10]. The experiments showed that the proposed approach obtains better results (accuracy rates combined with smaller subsets of features) than traditional modular neural networks.

As future works, other task decomposition methods, parameter that directly impacts the performance of the whole system, as well as other feature selection algorithms can be evaluated since the architecture proposed is flexible.

References

[1] T. Kohonen, *Self-Organization and Associative Memory*, Springer Series in Information Sciences, vol. 8, pp. 1-29, 1984.

[2] E. Micheli-Tzankou, *A Neural Network Model for Invertebrate Retina*, In IEEE Engineering in Medicine and Biology Society, pp. 13-16, 1987.

[3] E. Fiesler and A. Choudry, *Neural Networks as a Possible Architecture for the Distributed Control of Space Systems*, In Conference on Artificial Intelligence Space Applications, pp. 401-408, 1987.

[4] D. Hartline, *Models for Simulation of Real Neural Networks*, In International Neural Network Society (INNS) First Ann. Meeting (Boston, EUA), pp. 256-261.

[5] D. Acheson, *Trends in Neural Computation*, vol 35. Springer Berlin/Heidelberg, 2007.

[6] G. Auda, M. Kamel and H. Raafat, *Modular Neural Network Architecture for Classification*, In Proceedings of IEEE International Conference on Neural Networks, 1996, vol. 2, pp. 1279-1284.

[7] N. Takahashi and T. Nishi, *Global Convergence of Decomposition Learning Methods for Support Vector Machines*, IEEE Transactions on Neural Networks, vol. 17, no. 6, pp. 1362-1369, Nov. 2006.

[8] Y. Li, M. Dong and R. Kohari, *Classifiability-based Omnivariate Decision Trees*, IEEE Transactions on Neural Networks, vol. 16, no. 6, pp. 1547-1560, Nov. 2005.

[9] L. Y. Chuang, H. W. Chang, C. J. Tu and C. H. Yang, *Improved Binary PSO for Feature Selection Using Gene expression Data*, Computational Biology and Chemistry, pp. 29-38, 2007.

[10] S. U. Guan, C. Bao and T. Neo, *Reduced Pattern Training Based on Task Decomposition using Pattern Distributor*, IEEE Transactions on Neural Networks, vol. 18, pp. 1738-1740, 2007.

[11] J. Kennedy and R. C. Eberhart, *A Discrete Binary Version of the Particle Swarm Algorithm*, Systems, Man and Cybernetics, 1997. In: Proceedings of the IEEE International Conference on Computational Cybernetics and Simulation, vol. 5, October 12-15, pp. 4104-4108.

[12] J. Kennedy and R. C. Eberhart, *Particle Swarm Optimization*, In: Proceedings of the IEEE International Conference on Neural Networks, vol. 4, Perth, Australia, pp. 1942-1948, 1995.

[13] Y. Shi and R. C. Eberhart, *A Modified Particle Swarm Optimizer*, Evolutionary Computation Proceedings in IEEE World Congress on Computatational Intelligence, pp. 69-73, 1998.

[14] ELENA benchmark [http://www.dice.ucl.ac.be/neural-nets/Research/Projects/ELENA/elena.html].

[15] A. Asuncion and D. J. Newman, *UCI Machine Learning Repository*,[http://www.ics.uci.edu/mlearn/MLPRepository.html].