A Computationally Light Pruning Strategy for Single Layer Neural Networks based on Threshold Function

Edoardo Ragusa, Christian Gianoglio, Rodolfo Zunino and Paolo Gastaldo
Department of Electrical, Electronic, Telecommunication Engineering and Naval Architecture DITEN
University of Genoa
Genova, Italy
Email: {edoardo.ragusa,christian.gianoglio}@edu.unige.it, {rodolfo.zunino,paolo.gastaldo}@unige.it

***Abstract*—Embedded machine learning relies on inference functions that can fit resource-constrained, low-power computing devices. The literature proves that single layer neural networks using threshold functions can provide a suitable trade off between classification accuracy and computational cost. In this regard, the number of neurons directly impacts both on computational complexity and on resources allocation. Thus, the present research aims at designing an efficient pruning technique that can take into account the peculiarities of the threshold function. The paper shows that feature selection criteria based on filter models can effectively be applied to neuron selection. In particular, valuable outcomes can be obtained by designing ad-hoc objective functions for the selection process. An extensive experimental campaign confirms that the proposed objective function compares favourably with state-of-the-art pruning techniques.**

***Index Terms*—neural networks, threshold function, embedded systems**

I. Introduction

Machine learning (ML) methods provide a powerful tool to tackle classification and regression problems. Nowadays, several applications exploit such technologies. Indeed, an increasing amount of such applications rely on resource-constrained embedded systems that are expected to yield online inferences in real-time. [1].

Deep learning paradigms offer state-of-the-art performances in terms of prediction accuracy; on the other hand, the hardware implementation of the corresponding decision function still does not fit low-resources device. Moreover, the number of hyper parameters that characterizes these models discourage their usage when learning process should involve little computational cost and human intervention [2].

Random networks, conversely, represent a valuable solutions in term of training cost and ease of training process automation. Furthermore, the resulting predictors are very suitable for low cost implementations [1]. In general, randomness is one of the basic elements of learning from the very beginning. Despite ever green criticism due to the obvious structural limitation with respect to fully optimized approach, the literature [3], [4] ensures learning, approximation and compression capabilities of random based models.

This paper focuses on random Single Layer Feedforward Networks (SLFNs) based on the hard limit (threshold) activation function. Such configuration leads to a prediction function that can be supported by a very simple architecture, as multipliers are no longer needed [1]. In fact, the number of neurons in the hidden layer directly impacts on computational complexity and resources allocation, both in the case of training and inference phases. However, the most efficient training procedure for SLFNs based on threshold function is not designed to find a proper balance between the size of the hidden layer and the generalization performance [5]. In fact, the algorithm exploits a remapping of the input datum in a higher dimensional space, with explicit size N. As a consequence, demanding pruning strategies can be replaced by feature selection procedures because each neuron identifies a unique feature of the remapped space.

In this regard, the paper offers two main contributions. First, it proposes the use of feature selection algorithms as pruning procedure for the selection of hidden neurons. This in turn leads to the second contribution: the design of a novel objective function for the pruning of inactive neurons derived by learning process. The proposed objective function fit in the framework of filter models [6]. Experimental results show that the proposed approach can almost reach the performance of the state-of-the-art pruning algorithm for random networks [7], which in fact is less efficient in terms of computational costs.

II. BACKGROUND

Let $\mathcal{X} \in R^D$ be an input domain and let $\mathcal{T} = \{(\boldsymbol{x}, y)_i; \boldsymbol{x} \in \mathcal{X}; y \in \{-1, 1\}; i = 1, .., Z\}$ be a labeled training set. In addition, let $\phi(\boldsymbol{x} \cdot \boldsymbol{w} + b)$ be an activation function, where $w \in R^D$ and $b \in R$ are adjustable parameters; standard choices for ϕ are sigmoid, tanh and Gaussian. The prediction function of a SLFN for bi-class classification is given by:

$$y(\boldsymbol{x}) = sign\Big(\sum_{n}^{N} \beta_n \phi(\boldsymbol{x} \cdot \boldsymbol{w}_n + b_n)\Big) \tag{1}$$

The authors acknowledge financial support from Compagnia di San Paolo, grant number: 2017.0559, ID ROL: 19795

Models such as Extreme Learning Machine (ELM) [8], Random Vector Functional Link (RVFL) [9], and Weighted Sum of Random Kitchen Sinks [10] rely on training procedures where parameters $\boldsymbol{w}_n, b_n$ are not learned but simply extracted from random distributions. The common rationale is to exploit the hidden layer to complete a prearranged remapping of the input space: $\mathcal{X} \Rightarrow R^N$. Accordingly, the parameters of the hidden layer are set randomly or by exploiting some characteristics of the dataset [11]. As a major result, training can be accomplished by solving a linear regression problem in the remapped space R^N. In the usual configuration the training objective function includes a Tikhonov regularization term [8]. Notably, the learning problem admits a unique closed form solution:

$$\boldsymbol{\beta} = (\lambda \boldsymbol{I} + \boldsymbol{H}^T \boldsymbol{H})^{-1} \boldsymbol{H}^T \boldsymbol{y} \tag{2}$$

where $h_{in} = \phi(\boldsymbol{x}_i^t \boldsymbol{\omega}_n + b_n)$ and λ is the smoothing hyper-parameter.

III. RELATED WORKS

Optimizing the trade off between number of neurons and generalization capabilities of the eventual predictors is a well known problem. In [7], [12] the optimization problem has been modified to the purpose of obtaining sparse solution. As a major consequence, ineffective neurons are removed. In [12] the authors exploited a l_1 regularization term to sparsify the solution; then, the selected neurons were involved in the training of a standard ELM. Optimally Pruned ELM (OPELM) [7] combined multiple sparse regressions and leave-one-out mechanism to prune the less informative neurons. In both the approaches, tough, the resulting pruning process is computationally demanding; moreover, a few hyper-parameters are added to the optimization problem.

Biologically-inspired solutions have stimulated various works: self-adaptive evolutionary ELM [13], dolphin swarm ELM [14], genetic ensemble of ELM [15], particle swarm optimization based ELM [16] and Artificial Immune System based ELM [17]. All these approaches exploit strategies derived from the observation of natural phenomena. In general, such models are computationally demanding. In fact, most of them involve non-convex optimization problems.

In [1], [18] the target was the implementation of the predictor on resource-constrained device. Hence, the underlying mapping strategy was designed to fulfill specific constraints on the admissible activation function, i.e., respectively, hard-limiter function and tristate function.

The original solution proposed by this paper is to approach the pruning of ineffective neurons as a feature selection problem. Such approach takes advantage of a peculiarity of random SLFNs: the hidden layer performs a remapping of the input data in a space with explicit dimensionality N [11]. To the purpose of limiting the computational cost of the pruning process, feature selection is implemented by using filtering models [6].

IV. CONTROLLING THE SIZE OF THE HIDDEN LAYER

The size of the hidden layer (i.e., the number of neurons) represents a crucial aspect, since it affects generalization capabilities, computational complexity and resource occupancy. Ideally, one would the training to yield a decision function that optimizes the trade-off between size of the hidden layer and generalization ability.

In the following, $\boldsymbol{h}_n$ denotes the n-th feature in the space $\{-1, 1\}^N$, i.e., the activation of the n-th neuron. Correspondingly, $\boldsymbol{h}_n$ is a Z-dimensional vector, where

$$h_{i,n} = sign(\boldsymbol{x}_i \cdot \boldsymbol{w}_n + b_n); \tag{3}$$

here, $\boldsymbol{x}_i$ is a pattern belonging to the labeled training set $\mathcal{T}$.

In principle, N affects differently training and inference phase. The cost of the training phase can be divided into two parts: the first part concerns the computation of matrix $\boldsymbol{H}$, which involves the evaluation of $Z \times N$ neuron activation. The number of floating point operations for each neuron can be formalized as:

$$\#_{flops\_act} = 2 * D, \tag{4}$$

because D multiplications and D sums should be completed. Since a threshold function is involved, a straightforward operation leads to the activation of the neuron.

The second part is the solution of a linear equation system (LES). It is worth to note that the closed form solution shown in eq. (2) is not the optimal approach in term of computational cost and numerical stability. State-of-the-art approaches employ sets of solvers and select the optimal one based on the peculiarities of the system at hand [19].

Overall, the number of floating point operations of the training phase can be formalized as:

$$\#_{flops\_training} = Z * N * \#_{flops\_act} + \#_{LES} \tag{5}$$

where $\#_{LES}$ is the number of operation required by the LES solver.

Inference phase consists in the evaluation of eq. (1). Thus, N linearly affects the number of floating point operations and the number of network parameters. The memory consumption of the eventual predictor can be formalized as follow:

$$\#_{par\_predictor} = N(D + 1) + N \tag{6}$$

where the first term takes into account the number of hidden parameters and the second term refers to the number of parameters in the linear separator. Similarly, the number of floating point operation is:

$$\#_{flops\_test} = N * \#_{flops\_act} + N \tag{7}$$

In general, SLFNs that exploits randomness to complete a prearranged remapping of the input space does not address effectively the trade-off between number of neurons and generalization abilities. This drawback is indeed related to the choice of not adjusting the neurons' parameters according to

a cost function. In principle, one may tackle this issue by keeping in mind that the hidden layer actually remaps the original input space $\mathcal{X}$ into a new feature space R^N [11]. As a result, feature selection methods can be applied in such space. In this case, feature selection eventually drives a 'neuron selection'. Hence, the SLFN is first trained by setting the size N of the hidden layer. Then, feature selection is exploited to downsize the layer from N to M ($M < N$).

The literature provides different techniques designed to shrink the number of features [6]. This paper focuses on a specific category of feature selection methods: filter models [6]. These methods share a common, iterative approach to select M relevant features out of N candidate features. Initially, the set $\mathcal{S}$ of selected features is empty, while the set $\mathcal{F}$ includes all the N candidate features. At each iteration, an objective metric is used to identify in $\mathcal{F}$ the most relevant feature; such feature is added to $\mathcal{S}$ and removed from $\mathcal{F}$. The procedure ends when $\mathcal{S}$ include M features.

Actually, standard objective functions for filter models do not fit properly the problem at hand. Threshold function admits only two values. As a consequence, their behaviour is different from the common activation function that characterizes hidden layers. On one hand, this non linearity suits theoretical constraints for universal approximation capability [8]. On the other hand, in terms of feature selection, one has to deal with binary features. Interestingly, though, thresholds and entropy gain criteria are two of the core elements of random forest (RF) classifiers [20]. RFs proved to be very effective in several application; hence, they established de facto the suitability of the pair {entropy gain, threshold function}.

Following this rationale, the present research adopts the min entropy (mEN) method as objective function. Since a threshold function is involved, the entropy gain can be assessed by observing that the generic n-th neuron separates the training set $\mathcal{T}$ into two subsets:

$$\Omega_{n,u} = \{(\boldsymbol{x}, y)_i | \boldsymbol{x}\boldsymbol{w}_n + b_n < 0\},$$
$$\Omega_{n,o} = \{(\boldsymbol{x}, y)_i | \boldsymbol{x}\boldsymbol{w}_n + b_n > 0\} \quad (8)$$

Accordingly, one identifies the most relevant feature in $\mathcal{F}$ as follows:

$$\arg\max_{\boldsymbol{h}_n \in \mathcal{F}} \Big\{ E(\mathcal{T}) - E(\Omega_{n,u}) - E(\Omega_{n,o}) \Big\} \quad (9)$$

The second objective function presented in this paper is a linear combination of (9) and the standard min redundancy (mRD) (10) criterion. Among the criteria available for feature selection, diversity is well-known as a reliable metric. In general, two features that are highly correlated do not introduces information. Thus, mRD is formalized as:

$$\arg\min_{\boldsymbol{h}_n \in \mathcal{F}} \Big\{ \sum_{\boldsymbol{h}_q \in \mathcal{S}} P(\boldsymbol{h}_n, \boldsymbol{h}_q) \Big\} \quad (10)$$

where $P()$ is the Pearson correlation coefficient between two vectors. The eventual objective function is called min entropy min redundancy (mENmRD).

V. EXPERIMENTAL RESULTS

The experimental session evaluated the ability of the feature selection methods described above to yield a predictor that can balance accuracy and size of the hidden layer.

The performance of the feature selection methods has been evaluated on 5 multiclass problems and 9 bi-class problems. All the datasets belong to the UCI database [21]: Glass Identification, Wine, Image Segmentation, Statlog (Vehicle Silhouettes), Pima Indians Diabetes, Ionosphere, Connectionist Bench, LSVT voice rehabilitation, Ozone level detection, QSAR Biodegradation, Breast Cancer Wisconsin (Diagnostic), Statlog (Australian Credit Approval) and Breast Cancer Wisconsin (Original). For each dataset, the experiment was organized as follows:

- randomly split the dataset into three sets: training (70% of the dataset), development (15%), and test (15%).
- generate 2,000 neurons with as many random pairs $(\boldsymbol{w}_n, b_n)$.
- select M neurons out of the 2,000 by using feature selection
- train the SLFN with 2,000 neurons by exploiting the development set to configure the regularization parameter λ
- train the SLFN with M neurons by exploiting the development set to configure the regularization parameter λ.

The present work compares the proposed objective functions with three state-of-the-art metrics that can support filter models: max relevance (MRL) [21], min redundancy (mRD) [21], max relevance min redundancy (MRLmRD) [21]. Besides, an additional approach based on MultiResponse Sparse Regression (MRSR) algorithm is added to the comparison. Such algorithm supports the pruning procedure in OP-ELM, i.e., a state-of-the-art SLFN based on random parameters [22]. Accordingly, its performance provides a reference value for the other methods. It is worth noting, though, that the MRSR algorithm is expected to be order of magnitude more demanding of filter models in terms of computational complexity. MRSR should solve multiple regression problems, while filter models implies the solution of a single regression problem.

Table I compares the performance of the different feature selection methods. Each row of the table refers to a specific method and provides the outcome of the experiment for different values of M. The outcome is expressed as the difference -in terms of classification accuracy on the test set- between the SLFN with M neurons and the SLFN with 2,000 neurons; thus, a positive value means an increased accuracy. The accuracy is expressed as percentage, in the range $[0, 1]$, over the size of the test set. The table actually gives the average value of such quantity over the 14 problems along with its standard uncertainty (between brackets).

As expected, the table shows that filter models were not able to improve the performance of MRSR. MRSR indeed proved able to not affect classification accuracy even when 200 neurons out of 2,000 were selected. Nonethless, among the filter models, the proposed mENmRD metric achieved

TABLE I
EXPERIMENTAL RESULTS

Method	M= 100	M = 200	M = 500
mEN	-0.042(0.004)	-0.030(0.004)	-0.012(0.003)
mENmRD	-0.045(0.006)	-0.015(0.003)	-0.012(0.004)
MRL	-0.067(0.005)	-0.037(0.004)	-0.016(0.004)
mRD	-0.169(0.003)	-0.102(0.004)	-0.026(0.004)
MRLmRD	-0.058(0.003)	-0.024(0.002)	-0.013(0.003)
MRSR	-0.003(0.003)	+0.006(0.003)	+0.004(0.003)

satisfactory performances with $M = 200$. Overall, this means that feature selection can support an effective, computationally light pruning strategy for the proposed SLFN.

The experiments can also reveal which of the filtering method has been the most consistent over the different datasets and the different training/set pairs. Accordingly, MRSR is not involved in this campaign. Thus, for each benchmark and for each training/set pair, one point was assigned to the pruning that scored the best classification accuracy. The graph in Fig. 1 reports on the results of this experiment. The y-axis gives the cumulative score over the 560 tests (14 dataset $\times$ 40 training/test splits). On the x-axis, bars are grouped according to the size of $M = \{100, 200, 500\}$. In each group, bars refers, respectivley, to mEN, mENmRD, MRL, mRD, MRLmRD starting from the left.

The plot shows that the proposed objective functions, mEN and mENmRD, proved to be the very consistent, i.e., the corresponding predictors often occupied the highest position in the rank order over the different experiments. The MRL method slightly outperformed mENmRD only when $M = 500$.

VI. CONCLUSION

This paper proposed the use of feature selection techniques for implementing pruning procedures in random SLFNs. Two objective functions designed to deal with binary features have been introduced to the purpose of dealing with threshold activation functions. Experimental results involving 14 real world problems proved the effectiveness of the proposed solution in balancing the size of the hidden layer and the generalization performance of the eventual predictor.

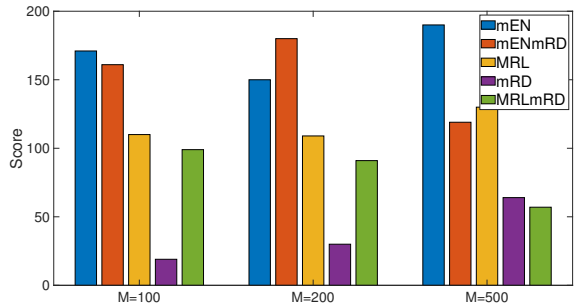

Fig. 1. Consistency evaluation of the five filtering methods; the plot gives the number of neurons M on the x-axis and the corresponding total amount of collected points on the y-axis.

REFERENCES

[1] E. Ragusa, C. Gianoglio, P. Gastaldo, and R. Zunino, "A digital implementation of extreme learning machines for resource-constrained devices," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 8, pp. 1104–1108, 2018.

[2] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," *arXiv preprint arXiv:1605.07678*, 2016.

[3] C. Gallicchio, J. Martin-Guerrero, A. Micheli, and E. Soria-Olivas, "Randomized machine learning approaches: Recent developments and challenges," in *Proceedings of the 25th European Symposium on Artificial Neural Networks (ESANN)*. i6doc. com, 2017, pp. 77–86.

[4] Y. Miche, B. Schrauwen, and A. Lendasse, "Machine learning techniques based on random projections." in *ESANN*, 2010.

[5] G.-B. Huang, Q.-Y. Zhu, K. Mao, C.-K. Siew, P. Saratchandran, and N. Sundararajan, "Can threshold networks be trained directly?" *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 53, no. 3, pp. 187–191, 2006.

[6] J. Xu, B. Tang, H. He, and H. Man, "Semisupervised feature selection based on relevance and redundancy criteria," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 9, pp. 1974–1984, 2017.

[7] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, and A. Lendasse, "Op-elm: optimally pruned extreme learning machine," *IEEE transactions on neural networks*, vol. 21, no. 1, pp. 158–162, 2010.

[8] G. Huang, G.-B. Huang, S. Song, and K. You, "Trends in extreme learning machines: A review," *Neural Networks*, vol. 61, pp. 32–48, 2015.

[9] Y.-H. Pao, G.-H. Park, and D. J. Sobajic, "Learning and generalization characteristics of the random vector functional-link net," *Neurocomputing*, vol. 6, no. 2, pp. 163–180, 1994.

[10] A. Rahimi and B. Recht, "Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning," in *Advances in neural information processing systems*, 2009, pp. 1313–1320.

[11] P. Gastaldo, F. Bisio, C. Gianoglio, E. Ragusa, and R. Zunino, "Learning with similarity functions: a novel design for the extreme learning machine," *Neurocomputing*, vol. 261, pp. 37–49, 2017.

[12] S. Decherchi, P. Gastaldo, A. Leoncini, and R. Zunino, "Efficient digital implementation of extreme learning machines for classification," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 59, no. 8, pp. 496–500, 2012.

[13] J. Cao, Z. Lin, and G.-B. Huang, "Self-adaptive evolutionary extreme learning machine," *Neural processing letters*, vol. 36, no. 3, pp. 285–305, 2012.

[14] T. Wu, M. Yao, and J. Yang, "Dolphin swarm extreme learning machine," *Cognitive Computation*, vol. 9, no. 2, pp. 275–284, 2017.

[15] X. Xue, M. Yao, Z. Wu, and J. Yang, "Genetic ensemble of extreme learning machine," *Neurocomputing*, vol. 129, pp. 175–184, 2014.

[16] Y. Xu and Y. Shu, "Evolutionary extreme learning machine–based on particle swarm optimization," in *International Symposium on Neural Networks*. Springer, 2006, pp. 644–652.

[17] H.-y. Tian, S.-j. Li, T.-q. Wu, and M. Yao, "An extreme learning machine based on artificial immune system," in *The 8th International Conference on Extreme Learning Machines (ELM2017), Yantai, China*, 2017.

[18] A. Patil, S. Shen, E. Yao, and A. Basu, "Hardware architecture for large parallel array of random feature extractors applied to image recognition," *Neurocomputing*, vol. 261, pp. 193–203, 2017.

[19] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU press, 2012, vol. 3.

[20] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[21] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 8, pp. 1226–1238, 2005.

[22] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, and A. Lendasse, "Op-elm: optimally pruned extreme learning machine," *IEEE transactions on neural networks*, vol. 21, no. 1, pp. 158–162, 2010.