ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0.
Available from http://www.i6doc.com/en/.

An evolutionary approach for optimizing weightless neural networks

Maurizio Giordano[1] and Massimo De Gregorio[2]

1- Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR – CNR)
Via P. Castellino, 111 – 80131 Naples – ITALY

2- Istituto di Scienze Applicate e Sistemi Intelligenti "E. Caianiello" (ISASI – CNR)
Via Campi Flegrei, 34 – 80078 Pozzuoli (NA) – ITALY

Abstract. WiSARD is a *weightless neural network* model using RAMs to store the function computed by each neuron rather than storing it in connection weights between neurons. Non-linearity in WiSARD is implemented by a mapping that splits the binary input into tuples of bits and associate these tuples to neurons. In this work we apply the evolutionary $\mu + \lambda$ algorithm [1] to make evolve an initial population of mappings toward the generation of new mappings granting significant improvements in classification accuracy in the conducted experiments.

1 Introduction

WiSARD[1] is a *weightless neural network* (WNN) model of computing widely used in literature as a pattern recognition method in image processing [2] and machine learning [3]. WNNs differ from artificial neural networks (ANNs) in that learned information is stored in memory cells (RAMs) located in neurons connected each other with weightless connections.

Learning and classification capabilities of WiSARD are very depending on the choice of the input-to-neuron mapping function [4]. In this work we provide evidence, by means of experiments on several case studies, that evolutionary optimization of WiSARD mappings is not only a viable approach but it also provides significant improvements in terms of accuracy. In particular, by applying a well-known evolutionary method [1] and by defining our own crossover and mutation operators, we developed a simple framework for the optimization of the input-to-neuron mapping which is a very critical hyperparameter of WiSARD.

2 The WiSARD model

WiSARD derived from the n-tuple recognition method introduced by Bledsoe & Browning (1959) [5]. The basic computation unit is the n-tuple neuron or RAM neuron: a RAM with 2^n memory cells. The simplest RAM network with properties of generalization is called *discriminator* and consists of a layer of m RAM nodes with n inputs so that the single layer receives a binary pattern of $m \times n$ bits. WiSARD is a multi-discriminator system in which each discriminator is trained to a different class of patterns. Training and classification phases of

[1] **W**ilkie, **S**tonham and **A**leksander **R**ecognition **D**evice.

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0.
Available from http://www.i6doc.com/en/.

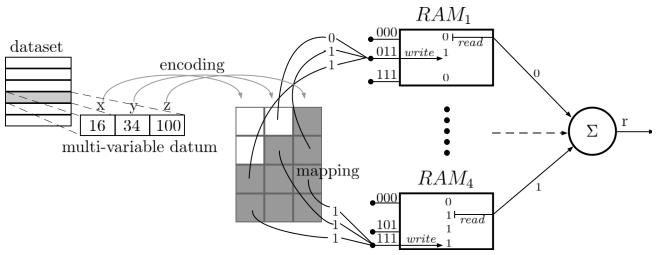


Fig. 1: WiSARD discriminator model

a WiSARD consist of writing and, respectively, reading the RAM node memory contents. In particular, training consists of storing (writing) a nonzero value in those RAM memory locations addressed by the binary input pattern; while classification consists of reading and then adding (*response*) the RAM memory contents addressed by a previously unseen pattern.

Although originally conceived to recognize binary patterns in image processing, WiSARD can also be applied to the classification of multi-variable numeric data in machine learning [3]. Figure 1 describes a WiSARD discriminator and how a 3-variable numeric datum (the vector x, y, z) is encoded into a binary pattern then mapped to a layer of RAMs. Classifiication accuracy in WiSARD depends on the choice of the input-to-neuron mapping. In this work we face the problem of optimizing this mapping which is represented by a 2D integer array formed by m rows (number of neurons) and n columns (bit positions, n-tuples).

In our optimization problem, the *fitness* of mappings is a "black-box" function that associates a mapping to the accuracy of a 10-fold cross-validation of a WiSARD using that mapping. Gradient-based optimization cannot be applied since it does not work with "black-box" systems. Although Grid (or Random) search could be applied to the optimization of "black-box" objective functions, this technique performs an exhaustive (or randomly selective) searching in a subset of the hyperparameter space. Even if we have only one parameter to vary (the mapping array), the number of possible mappings is so huge,[2] that it is unpractical to limit the search in a significant subset of it. Bayesian optimization builds a probabilistic model for "black-box" objective functions and uses an acquisition function to determine the next position x in the parameter space to locate the optimum. Since in our case the parameter value is a very large array, it is quite complex to develop such an acquisition function.

3 The evolutionary approach

Evolutionary optimization is used in hyperparameters optimization for statistical machine learning algorithms, deep neural network architecture search and

[2]The number of different partitions of s indices into m groups of n elements is: $Pr_{m,n} = \binom{s}{m,n} = \frac{s!}{m!n!}$, which, for m=8 and n=4, gives almost 10^{29} possible mappings!

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0. Available from http://www.i6doc.com/en/.

training of weights [6, 7]. We adopted an evolutionary approach for our optimization problem for the following reasons: 1) it applies to "black-box" systems; 2) the searching is not limited to a subset of the parameter space; 3) the next position in the parameter space is computed by simple crossover and mutation operators; and 4) during the evolution, multiple paths towards optimal solutions are explored at the same time.

Mappings as individuals of a population. We denote with $\underset{nm}{\mathcal{M}}$ the space of mappings (2D arrays) formed by a number m of n-tuples of indices between 0 and s-1, where $s{=}m{\times}n$. The trivial mapping is the linear one (see Eq. 1), and each permutation of elements in a mapping produces a licit mapping for WiSARD discriminators with m neurons and n-tuple based addressing (see Eq. 2).

$$M_0 = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \\ 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 \\ 20 & 21 & 22 & 23 \\ 24 & 25 & 26 & 27 \\ 28 & 29 & 30 & 31 \end{bmatrix} \in \underset{nm}{\mathcal{M}} \qquad (1) \qquad M_i \in \underset{nm}{\mathcal{M}} \rightarrow Perm(M_i) \in \underset{nm}{\mathcal{M}} \qquad (2)$$

Starting form a mapping, an equivalent one is obtained by swapping 1) two indices in the same n-tuple (same row), and/or 2) two n-tuples (rows) in the mapping, since exchanging two memory cells in a RAM and/or two neurons does not change the WiSARD's behaviour. From now on, with M_i we indicate the equivalence classes of mappings in $\underset{nm}{\mathcal{M}}$. The *fitness* of M_i (see Eq. 4) is a real function that associates the 10-fold cross-validation accuracy of a WiSARD with parameters n, m and M_i trained on a dataset $\mathcal{D}$.

$$\mathcal{P} = \{M_1, \dots M_\mu\} \in \underset{nm}{\mathcal{M}}^{\mu} \qquad (3) \qquad F : \underset{nm}{\mathcal{M}} \times \mathcal{D} \rightarrow [0,1] \in \mathbb{R} \qquad (4)$$

$$select : \underset{nm}{\mathcal{M}}^{\lambda} \rightarrow \underset{nm}{\mathcal{M}}^{\mu} \quad mate : \underset{nm}{\mathcal{M}}^{2} \rightarrow \underset{nm}{\mathcal{M}}^{2} \quad mutate : \underset{nm}{\mathcal{M}} \rightarrow \underset{nm}{\mathcal{M}} \qquad (5)$$

Selection, mutation, and recombination are described as operators that transform the entire population. The *select* operator extracts μ individuals from a population of size λ. In case $\lambda{=}\mu$, then a 1:1 replacement of the population occurs, although the selection procedure is stochastic and the same individual can be selected multiple times, according to the fitness values. The *mate* operator applies on a pair of parent individuals and produces two new children individuals. The *mutate* operator generates individuals by mutating other individuals. The iterative application of recombination and mutation of individuals is named *variation loop* and new individuals produced at each iteration form the *offspring*. The way operators are applied in the variation loop is one of the characteristic of an evolutionary algorithm. Mutation and recombination of individuals is stochastic: the application of mutation depends on a parameter Θ_u representing the probability that an offspring is produced by mutation, while recombination depends on a parameter Θ_r representing the probability that an offspring is produced by crossover. To describe the algorithm, we also need to define the following utility functions: the $clone(M_i)$ function generates perfect copy of individuals; the $rndInd(\mathcal{P})$ and $rndPairInd(\mathcal{P})$ functions randomly extract, respectively, an individual and a pair of individuals from the population.

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0. Available from http://www.i6doc.com/en/.

Crossover and mutation. The crossover operator swaps the sub-arrays of two mappings at the middle row. Let us consider two mappings each formed by eight quadruples (n=4, m=8). The crossing point of the two matrices is the 5^{th} row, and after crossover application the two mappings become:

$$mate\left(\begin{bmatrix}11&22&10&2\\16&14&28&26\\20&13&24&5\\17&8&30&25\\23&1&31&6\\4&18&29&19\\9&7&27&3\\0&21&15&12\end{bmatrix},\begin{bmatrix}27&29&3&30\\8&25&2&11\\28&5&4&16\\22&18&13&31\\14&6&21&26\\9&19&1&23\\12&17&20&24\\15&10&0&7\end{bmatrix}\right)\rightarrow\begin{bmatrix}11&22&10&2\\16&14&28&26\\20&13&24&5\\17&8&30&25\\31&6&21&4\\9&19&1&23\\12&\mathbf{18}&\boxed{29}&\underline{27}\\15&|3|&0&7\end{bmatrix},\begin{bmatrix}27&29&3&30\\8&25&2&11\\28&5&4&16\\22&18&13&31\\23&1&14&6\\\overline{26}&\mathbf{17}&\boxed{20}&19\\9&7&\underline{24}&|10|\\0&21&15&12\end{bmatrix}. \quad (6)$$

After crossing matrices some duplicated indices may occur: they are identified and then swapped, in such a way the output mappings have no duplicates.[3]

In a mapping mutation at most n pairs of n-tuples are involved and selected at random. In each pair of tuples, a swapping of indices at the same position occurs. The position is different for each pair, so that n exchanges occur at different columns of the mapping. Only $2{\times}n$ rows are involved in the swapping of indices, no matter if the mapping contains a larger number of rows. If the mapping has less than $2{\times}n$ rows, then the swapping operation will be in a number of $m/2$. An example of mutation could be the following:

$$mutate\left(\begin{bmatrix}11&\boxed{22}&10&2\\16&14&\underline{28}&26\\20&13&24&|5|\\17&\boxed{8}&30&25\\31&6&21&|4|\\9&19&\underline{1}&23\\\mathbf{12}&18&29&27\\\mathbf{15}&3&0&7\end{bmatrix}\right)\rightarrow\begin{bmatrix}11&\boxed{8}&10&2\\16&14&\underline{1}&26\\20&13&24&|4|\\17&\boxed{22}&30&25\\31&6&21&|5|\\9&19&\underline{28}&23\\\mathbf{15}&18&29&27\\\mathbf{12}&3&0&7\end{bmatrix}. \quad (7)$$

Evolution. The evolution of WiSARD mappings is described in Algorithm 1. First, a population of μ mappings is initialized with invalid fitness. The evolution loop begins and lasts τ steps. At each evolution step a variation loop iterates λ times. Two threshold values Θ_r and Θ_u are set such that $\Theta_r + \Theta_u < 1$. At each variation step a random *choice* in the range [0,1] is extracted: if *choice* is less than Θ_r two individuals from the parental population are randomly extracted, cloned and then mated. Only the first child is appended to the offspring. Otherwise, if *choice* is in the range $[\Theta_r, \Theta_r{+}\Theta_u[$ one individual from the parental population is randomly selected, cloned and then mutated. The resulting mutant is appended to the offspring. Otherwise, if *choice* is in the range $[\Theta_r{+}\Theta_u, 1]$ one individual is selected at random from the parental population, cloned and appended to the offspring (reproduction). At the end of the evolution step, the offspring is appended to the parental population and its fitness is *evaluated*. From the parental-plus-offspring set the *select* operator randomly extracts μ individuals with best fitness which form the new population for the next evolution step.

In [8] the authors proposed a similar evolutionary approach for optimization of input-to-neurons mappings, although they only considered mutation operations.[4] The fitness of the mutated mapping is evaluated and then adopted if it is

[3] While this is not a requirement in the WiSARD's definition, in our adoption of this neural model we assume that the input-to-neuron mapping has no duplicate indices, that is neurons span the entire input pattern with no overlaps.

[4] A number t of discriminator inputs (binary input points) are swapped at each mutation.

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0. Available from http://www.i6doc.com/en/.

Algorithm 1 $(\mu + \lambda)$ algorithm

Require: μ, τ, λ, Θ_r, Θ_u
Ensure: $\mathcal{M}(\tau)$ best population after τ runs
  $t \leftarrow 1$
  $\mathcal{M}(t) \leftarrow initialize(\mu)$ ▷ create initial population
  $\mathbf{F}(t) \leftarrow evaluate(\mathcal{M}(t))$ ▷ initialize fitness as invalid
  while $t < \tau$ **do** ▷ evolution loop
    $\mathcal{M}_o \leftarrow \{\}$ ▷ init offspring with empty set
    for $i = 1, 2, \ldots, \lambda$ **do** ▷ variation loop
      choice = random()
      if $choice < \Theta_r$ **then** ▷ crossover
        $M_x, M_y \leftarrow clone(randomPairInd(\mathcal{M}(t))$
        $M_x, M_y \leftarrow recombine(M_x, M_y)$
      else if $choice < \Theta_r + \Theta_u$ **then** ▷ mutation
        $M_x \leftarrow clone(rndInd(\mathcal{M}(t))$
        $M_x \leftarrow mutate(M_x)$
      else ▷ reproduction
        $M_x \leftarrow clone(rndInd(\mathcal{M}(t)))$
      end if
      $\mathcal{M}_o \leftarrow append(M_x, \mathcal{M}_o)$
    end for
    $t \leftarrow t + 1$
    $\mathcal{M}(t) \leftarrow \mathcal{M}_o + \mathcal{M}(t-1)$ ▷ add offspring to population
    $\mathbf{F}(t) \leftarrow evaluate(\mathcal{M}(t))$ ▷ calculate fitness of offspring
    $\mathcal{M}(t) \leftarrow select(\mathcal{M}(t), \mathbf{F}(t), \mu)$ ▷ select μ individuals from population
  end while

greater than the previous one. To avoid the risk of limiting the search within a local optimum, the simulating annealing technique is used.[5] The $(\mu+\lambda)$-algorithm copes with the same problem of local optimum by mixing, at each stage of the evolution, the parental population with the offspring. The authors in [8] showed a 10% increase in the discrimination power in a character recognition application. In our experiments we observed higher percentage increases in performance.

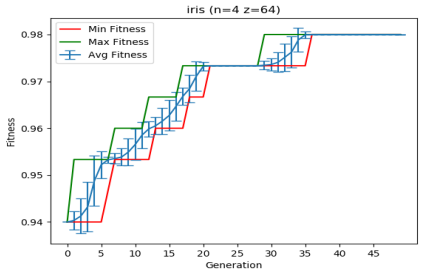
4 Experiments

For the validation of our approach we used WiSARD-Classifier,[6] i.e. a supervised classification method [3] based on WiSARD for multi-variable numeric data. In all experiments we chose an initial populations with μ=100 linear mappings, and λ=μ. The evolution loop iterated for τ=50 steps, while crossover and mutation probabilities were set to Θ_r=0.5 and Θ_u=0.2. We computed optimal mappings for ten datasets selected from the UCI Machine Learning repository.[7] In all experiments we used WiSARD models with parameters n=4 and z=64, with model fitness being the 10-fold cross-validation accuracy on each dataset.

Figure 2a shows the evolution of the average accuracy of models trained on the *iris* dataset: after 40 steps the algorithm converges and the population of mappings with zero deviation gives the highest accuracy. Figure 2b compares the 1^{st} generation average accuracy of models with that at the end of the evolution on 10 different datasets. Performance comparison of WiSARD and Random Forests

[5] The number of swaps t exponentially decays from an initial value, thus allowing large jumps in the search space at the early stages of the evolution loop.
[6] Available for download at: https://github.com/giordamaug/WisardClassifier-C_vectors.
[7] http://archive.ics.uci.edu/ml

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0. Available from http://www.i6doc.com/en/.



(a) accuracy plot on iris dataset

dataset	WiSARD n=4, z=64 start	WiSARD n=4, z=64 end	RF n.estimators=100 criterion=entropy
balance	0.701	**0.814**	0.701
ecoli	0.806	**0.881**	0.857
glass	0.668	0.748	**0.780**
hayes	0.575	**0.887**	0.819
iris	0.940	**0.980**	0.967
monks-1	0.746	**0.996**	0.985
phoneme	0.749	0.837	**0.912**
sonar	0.702	0.740	**0.754**
vehicle	0.624	0.731	**0.753**
wine	0.972	**0.989**	0.977

(b) comparison with Random Forests (RF)

Fig. 2: WiSARD's accuracy improvements on some UCI datasets

is also reported. The results are very good if we consider that WiSARD was configured with a limited addressing scheme (n=4) and a rough discretization (z=64) of numeric values. This is due to the algorithm that, at each generation, mixes parental and offspring individuals with high fitness: in this way, the individuals not selected to form the current offspring are not discarded and, later on, they could produce new and more profitable paths of evolution for mappings.

5 Conclusions

In this work we faced the problem of optimizing input-to-neuron mappings in a weightless neural network called WiSARD. We used an evolutionary algorithm, called (μ+λ)-algorithm [1], together with recombination and mutation operators we designed and implemented for the purpose. The experimental results on ten UCI datasets validated the approach and its good performance.

References

[1] T. Back, D.B. Fogel, and Z. Michalewicz, editors. *Basic Algorithms and Operators*. 1999.

[2] Massimo De Gregorio and Maurizio Giordano. Background estimation by weightless neural networks. *Pattern Recognition Letters*, 96:55–65, 2017.

[3] Massimo De Gregorio and Maurizio Giordano. An experimental evaluation of weightless neural networks for multi-class classification. *J. of Applied Soft Computing*, 13, July 2018.

[4] I. Aleksander and T. J. Stonham. Guide to pattern recognition using random-access memories. *Computers and Digital Techniques, IEE Journal on*, 2(1):29–40, February 1979.

[5] W. W. Bledsoe and I. Browning. Pattern recognition and reading by machine. In *Eastern Joint IRE-AIEE-ACM Computer Conference*, pages 225–232, New York, NY, USA, 1959.

[6] Steven R. Young *et al.* Optimizing deep learning hyper-parameters through an evolutionary algorithm. In *Proc. Workshop on Machine Learning in High-Performance Computing Environments*, MLHPC '15, pages 1–5, New York, NY, USA, 2015. ACM.

[7] Erik Bochinski, Tobias Senst, and Thomas Sikora. Hyper-parameter optimization for convolutional neural network committees based on evolutionary algorithms. *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3924–3928, 2017.

[8] J.M. Bishop A.A. Crowe P.R. Michinton R.J. Mitchell. Evolutionary learning to optimise mapping in n-tuple networks. In *IEE Colloquium on Machine Learning*, pages 1–3, 1990.