Comparing Evolutionary Strategy Algorithms for Training Spiking Neural Networks

José S. Altamirano, Manuel Ornelas, Andrés Espinal, Raúl Santiago, Héctor Puga, Martín Carpío, and Sergio Tostado

Tecnológico Nacional de México, Instituto Tecnológico León, León, Gto., México
josesaltf@gmail.com,mornelas67@yahoo.com.mx,
andres.espinal@itleon.edu.mx

Abstract. Spiking Neural Networks are considered as the third generation of Artificial Neural Networks, these neural networks naturally process spatio-temporal information. Spiking Neural Networks have been used in several fields and application areas; pattern recognition among them. For dealing with supervised pattern recognition task a gradient-descent based learning rule (Spike-prop) has been developed, however it has some problems like no convergence. To overcome these problems, metaheuristic algorithms such as Evolutionary Strategy have been used. In this work, three variants of the Evolutionary Strategy algorithm are compared for training Spiking Neural Networks. Several well-known benchmark dataset are used to test the capabilities of the algorithms.

Keywords: Spiking Neural Network, Evolutionary Strategy, Pattern Recognition.

1 Introduction

The Artificial Neural Networks (ANNs) are capable of modeling complex no linear systems, and can be used to solve a great number of day-to-day problems such as pattern recognition, optimization, prediction, function approximation, etc. [1].

In the last years, the third generation of ANN [2], Spiking Neural Networks (SNNs), have gained importance due to the inclusion of the firing time component in the neuron's process. This is obtained by coding the information in spike trains instead of spike rates as in the Second Generation of ANNs. That makes SNNs more similar to the biological neurons [3,4,5], and increases their computational power [6].

For the training process of SNNs, there has been developed a gradient-descent based learning rule, Spikeprop [7]. However it has some drawbacks such as the limitation on using negative weight values, convergence not guaranteed due to its tendency to end trapped in local minima, etc. [8].

To overcome these disadvantages, there had been some works about the use of metaheuristic algorithms for the learning process of the SNN [9,10,8,11]. In this work we compare the performance of three variants of Evolutionary

Strategy algorithms for the training process of SNN by testing three classical benchmarks data sets: Breast Cancer Wisconsin, Iris Plant and Wine, (from the UCI Repository [12]).

This document is organized as follow: Section 2 gives fundamentals for simulating SNNs. Section 3 explains the implemented methodology used for training SNNs. The experimental design and results are showed in Section 4. Finally, in Section 5 conclusions and future work are presented.

2 Spiking Neural Networks

A neural network can be defined as an interconnection of neurons, such that neuron outputs are connected to other neurons, even with themselves; both lag-free and delay connections are allowed [13]. There are several models or topologies of ANNs, which are defined around three aspects: computing nodes, communication links and message types [14].

In this work a fully-conected feed-forward SNN was used, which is defined as follows: the computing nodes are spiking neurons defined by the Spike Response Model (SRM), the communication links are formed by synaptic weights (excitatories and inhibitories) and positive delays values, and the message types are ruled by the time-to-first spike coding scheme.

2.1 Spike Response Model

The SRM has been introduced in [15], and it is an approximation of the dynamics of integrate-and-fire neurons. The neuron status is updated through a linear summation of the postsynaptic potentials resulting from the impinging spike trains at the connecting synapses. A neuron fires whenever its accumulated potential reaches the threshold (θ) from below (Fig. 1).

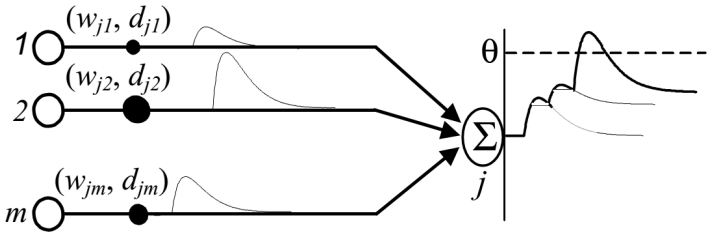

Fig. 1. Weighted input summed at the target neuron. Taken from [9]

In the SRM is consider that a neuron j has a set Γ_j of immediate predecessors called presynaptic neurons and receives a set of spikes with firing times t_i; $i \in \Gamma_j$. The internal state of a neuron is determined by Eq. (1), where w_{ji} is the synaptic

weight to modulate $y_i(t)$, which is the unweighted postsynaptic potential of a single spike coming from neuron i and impinging on neuron j.

$$x_j(t) = \sum_{i \in \Gamma_j} w_{ji} y_i(t) \tag{1}$$

The unweighted contribution $y_i(t)$ is given by Eq. (2), where $\epsilon(t)$ defines a spike response function describing a standard form of the postsynaptic potential.

$$y_i(t) = \epsilon(t - t_i - d_{ji}) \tag{2}$$

The function $\epsilon(t)$ is modeled by Eq. (3)

$$\epsilon(t) = \frac{t}{\tau} e^{1-(t/\tau)} \qquad for\ t > 0,\ else\ \epsilon(t) = 0 \tag{3}$$

where: t is the current time, t_i is the firing time of the presynaptic neuron i and d_{ji} is the associated synaptic delay. Finally the function has a τ parameter, that is the membrane potential time constant and define the decay time of the postsynaptic potential. Both θ and τ are constant and equal for all the neurons.

3 Metaheuristic Based Supervised Learning

Learning is a process by which the free parameters of a neural network are adapted through a process of stimulation from the environment in which the network is embedded. The type of learning is determined by the manner in which the parameter changes take place [16]. In this case, the learning is driven by an Evolutionary Strategy algorithm, and we refer to this learning process as Metaheuristic Based Supervised Learning.

In Metaheuristic-Based Supervised Learning, each individual contains all the free parameters of a previously structured SNN. Every individual is evaluated by means of a fitness function. To calculate the individual's fitness value the following steps are requiered: the first step makes a mapping process; this sets the individuals parameter as weights and delays in the SNN (Fig. 2). The second step uses the batch training as learning protocol, where all patterns are presented to the network before the learning takes place [17]. The third step is to calculate an error (to be minimized) according Eq. (4) (taken from [9]); where T are all training patterns, O are all output spiking neurons, $t_o^a(t)$ is the current timing output of the SNN and $t_o^t(t)$ is the desired timing output. The error calculated using Eq. (4) determines the fitness value of each individual and drives the supervised learning based on metaheuristic algorithms.

$$E = \sum_{t}^{T} \sum_{o \in O} \left(t_o^a(t) - t_o^t(t) \right)^2 \tag{4}$$

Next are presented the variants of Evolutionary Startegies used for training SNNs.

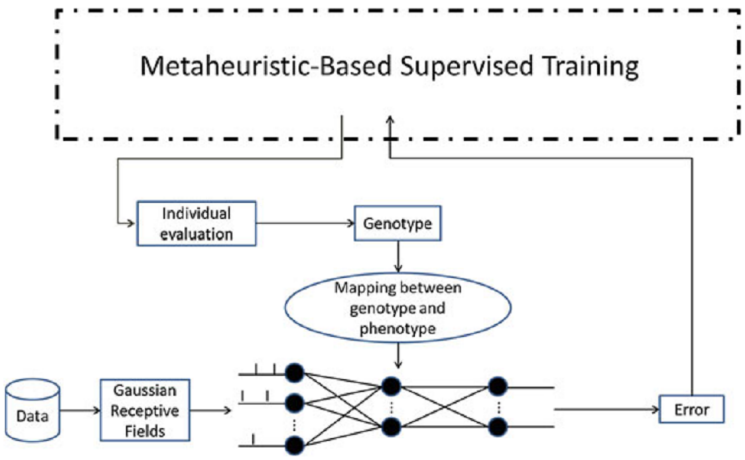

Fig. 2. Generic scheme for training SNNs with metaheuristic algorithms. Taken from [11]

3.1 Evolutionary Strategy

Evolutionary Strategy (ES), a variant of the Evolutionary Algorithms, was founded by students at the Technical University of Berlin (TUB), and although in the beginning it was not devised to compute minima or maxima of real-valued functions, it has proved to produce competitive solutions in such space ([18,19]). Next are presented the ES variants used in this work.

Evolutionary Strategies($\mu + \lambda$), (μ , λ) and Evolutionary Strategy There exists some variants for the Evolutionary Strategy, some of which depend on the selection for the new population, and others have a different mutation method.

One of the variants is the $(\mu + \lambda)$-ES, in which from a population of μ parents, there is generated λ descendants that are added to the original population, and, to keep the population size constant, the worst out of all $\mu + \lambda$ individuals are discarded [20].

Another variant is the (μ,λ)-ES, where from a population of μ parents, there is generated λ descendants and the selection takes place only among the generated offsprings, whereas their parents are forgotten no matter how good or bad their fitness was compared to that of the new generation. This strategy requires that $\lambda > \mu$ [20].

The third variant that is included in this work is a slightly modified Evolutionary Strategy (modified-ES), which is similar to the previous ones mentioned, but where the reproduction stage is not necessary, and there is the possibility of another type of operator for the mutation (Cauchy distribution mutation) [9].

The algorithm includes the parameter ρ, which is the number of parents that are going to take part in the reproduction. In the case of the first two variants, we considered a number of two parents giving the configuration $(\mu/2\overset{+}{,}\lambda)$; and $(\mu/1+\lambda)$ for the modified-ES, because only one parent was used.

The Algorithm 1 is based in [18], with some modifications to make it more general. The representation of each individual (χ) is composed of the object variables $(x_1, \ldots, x_n)$, being n the dimension of the problem, and some strategy parameters $(\eta_1, \ldots, \eta_n)$ of the mutation operator (the standard deviations). In the $(\mu/2\overset{+}{,}\lambda)$ version, there is a parent's selection, a recombination and a final modification using a random number from a Normal Distribution. On the other hand, in the $(\mu/1+\lambda)$ version the mutations are applied directly on every individual to generate the offspring. Finally, the population is replaced according to the applied version.

Algorithm 1 $(\mu \ / \ \rho \overset{+}{,} \lambda)$-ES

```
Begin
g ← 0
Initialize and evaluate pop_μ^g = {χ_i | i = 1, ..., μ}
repeat
    for l=1 to λ do
        if variant != modified-ES then
            parents = marriage (pop_μ^g, ρ) // selection through binary tournament
            χ̃_i= recombination(parents) // using intermediate recombination
            χ̃'_i= mutationNormal(X̃_i) // mutation using a Normal distribution
        if variant == modified-ES then
            r ← U[0, 1]
            if r < 0.5 then
                χ̃'_i= mutationNormal(χ̃_i) // mutation using a Normal distribution
            else
                χ̃'_i= mutationCauchy(χ̃_i) // mutation using a Cauchy distribution
    if typeSelection == (μ, λ) then
        pop_μ^{g+1} = selection(pop_λ^g) // Replace population with the created offspring
    else if typeSelection == (μ + λ) then
        pop_μ^{g+1} = selection(pop_μ^g,pop_λ^g) // Select new population from both the parents
        // and offspring populations
    g ← g + 1
until Stopping criteria
End
```

The marriage refers the way in which the ρ parents will be selected, in this work it was determined by using binary tournament selection. The intermediate recombination was made using Eq. (5) for the object variants and Eq. (6) for the standard deviations:

$$\widetilde{x}_l(j) = r\widetilde{x}_{r_1}(j) + (1-r)\widetilde{x}_{r_2}(j) \qquad \forall j \mid j = 1, \ldots, n \tag{5}$$

$$\widetilde{\eta}_l(j) = r\widetilde{\eta}_{r_1}(j) + (1-r)\widetilde{\eta}_{r_2}(j) \qquad \forall j \mid j = 1, \ldots, n \tag{6}$$

where r is a U[0,1] and r_1 and r_2 are the parents selected in the marriage. In the case of modified-ES it is not necessary to choose the parents due to the fact that each offspring is only a mutation of one parent.

The Normal mutation is made using the Eq. (7), and the Cauchy mutation for the modified-ES is made using Eq. (8). The standard deviations updates are part of each mutation and are made using Eq. (9).

$$\widetilde{x}'_l(j) = \widetilde{x}_l(j) + \widetilde{\eta}'_l(j)N(0,1) \tag{7}$$

$$\widetilde{x}'_l(j) = \widetilde{x}_l(j) + \eta(j)\delta_j \tag{8}$$

$$\widetilde{\eta}'_l(j) = \widetilde{\eta}_l(j)e^{\tau' N(0,1)+\tau N_j(0,1)} \tag{9}$$

Where:

- n represent the problem dimension
- $\tau' = 1/sqrt(2 \times (n))$ and $\tau = 1/sqrt(2 \times sqrt(n))$
- *N(0, 1)* denotes a normally distributed one dimensional random number with mean 0 and standard deviation 1. $N_j(0,1)$ indicates that the random number is generated anew for each value of j.
- δ_j is a Cauchy random variable, and it is generated anew for each value of j (Scale = 1).

The selection for the $(\mu + \lambda)$ includes elitism, to keep track of the better individuals, and is made through tournament selection.

4 Experiments and Results

Three classical benchmarks of pattern recognition from UCI[12] were used for experimentation: Brest Cancer Wisconsin (BCW), Iris Plant and Wine dataset.

4.1 Brest Cancer Wisconsin

The BCW data set consists of 683 samples belonging to two groups, namely benign and malignant cell tissues. Each data point is described with 9 attributes, represented by an integer ranging from 1 to 10 with larger numbers indicating a greater likelihood of malignancy. The data set is split into two parts, training and test data sets with 342 and 341 samples in each set respectively. The desired timing outputs where set to 6ms. for the benign class, and 10ms. for the malign class.

4.2 Iris Plant

The Iris plant dataset contains 3 classes of which 2 are not linearly separable, each class is formed by 50 patterns, where each one of them is described by 4 features. The desired timing outputs for setosa, versicolor and virginica classes are respectively 6, 10 and 14 ms.

4.3 Wine Data Set

These data are the results of a chemical analysis of wines grown in the same region in Italy but from three different cultivars. The analysis determined the quantities of 13 constituents (variables) found in each of the three types of wines. The desided timing outputs for each clase are 6ms. for class 1, 10ms. for class 2 and 14ms. for class 3.

4.4 Experimental Methodology

Due to computing times and statistical reasons, the experiments were carried out by applying 33 independently trainings for each dataset with every variant of Evolutionary Strategy. For feeding the SNN, pattern's dataset were codified by using four Gaussian Receptive Fields (GRFs) [7]. The SNN configuration for all problems was as follows: the neurons input layer depends on the GRFs, which varies by dataset features, 10 neurons into the hidden layer and 1 neuron into the output layer. All neurons from hidden and output layers had $\tau = 9$ and $\theta = 1$. The simulation time was 20ms. [11].

The Evolutionary Strategy configuration for all experiments was: $\mu = 30$, $\lambda = 30$ individuals, 15000 function calls as end criteria, and initial Standard deviation in the range $U[0, 1]$, which were choosen by empirical experimentation. The weight boundaries were [-1000, 1000] and the delay boundaries were [0.1,16] [9].

Table 1 shows the best fitness values achieved for each ES in each dataset over 33 training runs. The classification performance for both training and testing sets by the using the best results achieved by each ES are showed in Table 2.

Table 1. Results of the best fitness values for the training process of SNNs

	Fitness		
Data Set	(modified-ES)	$(\mu + \lambda)$-ES	(μ, λ)-ES
BCW	32	29	33
Iris Plant	0	16	15
Wine	6	21	14

Table 2. Comparison of the classification performance for the trained SNNs

	Training Set			Test Set		
Data Set	(modif-ES)	($\mu + \lambda$)-ES	(μ, λ)-ES	(modif-ES)	($\mu + \lambda$)-ES	(μ, λ)-ES
BCW	95.1%	94.13%	94.72%	95.91%	96.78%	95.61%
Iris Plant	100.0%	89.13%	84.0%	94.67%	73.33%	73.33%
Wine	93.18%	76.14%	90.91%	83.33%	60.0%	92.22%

The results show that the modified-ES version had a better performance in the training process for the three data sets, being able to achieve 100% in the classification for the training set of the Iris Plant dataset. On the other hand, even with lower fitness performance, the (μ, λ)-ES achieved good classification in the BCW and Wine datasets. The $(\mu+\lambda)$-ES version only had good classification performance in the BCW dataset.

5 Conclusions

This work compares three metaheuristics on the training of SNNs, and under the experiment circumstances, it was visible that even when the achieved fitness value was not too low, it is possible to obtain acceptable classification performance.

Based on the best results, the modified-ES showed better performance on both fitness value and classification.

For future work it is proposed to conduct experimentations with more metaheuristics and in more data sets, aiming for a more robust statistical analysis. Also we propone the research for some different fitness functions, and investigate the use of Grammar Evolution and Genetic Programming to evolve the neural network's structure.

Acknowledgments. Authors thank to Tecnológico Nacional de México, Instituto Tecnológico de León. The first author wants to thank to Consejo Nacional de Ciencía y Tecnología (CONACYT) for the economical support to his MS work.

References

1. Jain, A., Mao, J., Mohiuddin, K.: Artificial neural networks: a tutorial. Computer 29(3), 31–44 (Mar 1996). URL http://dx.doi.org/10.1109/2.485891
2. Maass, W.: Networks of spiking neurons: The third generation of neural network models. Neural Networks 10(9), 1659–1671 (Dec 1997). URL http://dx.doi.org/10.1016/S0893-6080(97)00011-7
3. Abeles, M.: Corticonics: Neural circuits of the cerebral cortex. Cambridge: Cambriedge University Press (1991)
4. Abeles, M., Prut, Y.: Spatio-temporal firing patterns in the frontal cortex of behaving monkeys. Journal of Physiology-Paris 90(3-4), 249–250 (Jan 1996). URL http://dx.doi.org/10.1016/S0928-4257(97)81433-7

5. Hopfield, J.J.: Pattern recognition computation using action potential timing for stimulus representation. Nature 376(6535), 33–36 (Jul 1995). URL http://dx.doi.org/10.1038/376033a0
6. Maass, W.: Noisy spiking neurons with temporal coding have more computational power than sigmoidal neurons. Advances in Neural Information Processing Systems 9, 211–217 (1997)
7. Bohte, S.M., Kok, J.N., La Poutr, H.: Error-backpropagation in temporally encoded networks of spiking neurons. Neurocomputing 48(1-4), 17–37 (Oct 2002). URL http://dx.doi.org/10.1016/S0925-2312(01)00658-0
8. Belatreche, A.: Biologically Inspired Neural Networks: Models, Learning, and Applications. VDM Verlag Dr. Mller, Saarbrcken (2010)
9. Belatreche, A., Maguire, L.P., McGinnity, M., Wu, Q.: An evolutionary strategy for supervised training of biologically plausible neural networks. In: The sixth international conference on computational intelligence and natural computing (CINC), proceedings of the 7th joint conference on information sciences, pp. 1524–1527. USA (2003)
10. Belatreche, A., Maguire, L.P., McGinnity, M.: Advances in design and application of spiking neural networks. Soft Computing 11(3), 239–248 (Oct 2006). URL http://dx.doi.org/10.1007/s00500-006-0065-7
11. Espinal, A., Carpio, M., Ornelas, M., Puga, H., Melin, P., Sotelo-Figueroa, M.: Comparing metaheuristic algorithms on the training process of spiking neural networks. Studies in Computational Intelligence pp. 391–403 (2014). URL http://dx.doi.org/10.1007/978-3-319-05170-3_27
12. Lichman, M.: UCI machine learning repository (2013). URL http://archive.ics.uci.edu/ml
13. Zurada, J.: Introduction to Artificial Neural Systems. West Publishing Co., St. Paul, MN, USA (1992)
14. Judd, J.: Neural network design and the complexity of learning. Neural Network Modeling and Connectionism Series, Massachusetts Institute Technology (1990)
15. Gerstner, W., Kistler, W.M.: Spiking Neuron Models: Single Neurons, Populations, Plasticity. Cambridge University Press (2002)
16. Haykin, S.: Neural Networks: A Comprehensive Foundation. Prentice Hall PTR, Upper Saddle River, NJ, USA (1998)
17. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification (2Nd Edition). Wiley-Interscience (2000)
18. Beyer, H.G., Schwefel, H.P.: Evolution strategies. a comprehensive introduction. Natural Computing 1(1), 3–52 (2002). URL http://dx.doi.org/10.1023/A:1015059928466
19. Rechenberg, I.: Evolutionsstrategie: optimierung technischer systeme nach prinzipien der biologischen evolution. Frommann-Holzboog (1973)
20. Schwefel, H.P.: Numerische optimierung von computer-modellen mittels der evolutionsstrategie (1977). URL http://dx.doi.org/10.1007/978-3-0348-5927-1