NEVE: A Neuro-Evolutionary Ensemble for Adaptive Learning

Tatiana Escovedo, André Da Cruz, Marley Vellasco, Adriano Soares Koshiyama

NEVE: A Neuro-Evolutionary Ensemble for Adaptive Learning

Tatiana Escovedo, André Vargas Abs da Cruz, Marley Vellasco, Adriano Koshiyama

Electrical Engineering Department
Pontifical Catholic University of Rio de Janeiro (PUC-Rio) - Rio de Janeiro – Brazil

Email: {tatiana,andrev,marley,adriano}@ele.puc-rio.br

Abstract. This work describes the use of a quantum-inspired evolutionary algorithm (QIEA-R) to construct a weighted ensemble of neural network classifiers for adaptive learning in concept drift problems. The proposed algorithm, named NEVE (meaning Neuro-EVolutionary Ensemble), uses the QIEA-R to train the neural networks and also to determine the best weights for each classifier belonging to the ensemble when a new block of data arrives. After running eight simulations using two different datasets and performing two different analysis of the results, we show that NEVE is able to learn the data set and to quickly respond to any drifts on the underlying data, indicating that our model can be a good alternative to address concept drift problems. We also compare the results reached by our model with an existing algorithm, Learn++.NSE, in two different nonstationary scenarios.

Keywords: adaptive learning, concept drift, neuro-evolutionary ensemble, quantum-inspired evolution.

1 INTRODUCTION

The ability for a classifier to learn from incrementally updated data drawn from a nonstationary environment poses a challenge to the field of computational intelligence. Moreover, the use of neural networks as classifiers makes the problem even harder, as neural networks are usually seen as tools that must be retrained with the whole set of instances learned so far when a new chunk of data becomes available.

In order to cope with that sort of problem, a classifier must, ideally, be able to [1]:

- Track and detect any sort of changes on the underlying data distribution;
- Learn with new data without the need to present the whole data set again for the classifier;
- Adjust its own parameters in order to address the detected changes on data;
- Forget what has been learned when that knowledge is no longer useful for classifying new instances.

A more successful approach consists in using an ensemble of classifiers. This kind of approach uses a group of different classifiers in order to be able to track changes on the environment. Several different models of ensembles have been proposed on the literature [2, 3, 4]:

- Ensembles that create new classifiers to each new chunk of data and weight classifiers according to their accuracy on recent data;
- Unweighted ensembles which can cope with new data that belongs to a concept different from the most recent training data;
- Ensembles that are able to discard classifiers as they become inaccurate or when a concept drift is detected.

Most models using weighted ensembles determine the weights for each classifier using some sort of heuristics related to the amount of mistakes the classifier does when working with the most recent data [5]. Although in principle any classifier can be used to build the ensembles, the ones which are most commonly used are decision trees, neural networks and naive Bayes [6].

In this work, we present an approach based on neural networks which are trained by means of a quantum-inspired evolutionary algorithm. Quantum-inspired evolutionary algorithms [7-11] are a class of estimation of distribution algorithms which present, for several benchmarks, a better performance for combinatorial and numerical optimization when compared to their canonical genetic algorithm counterparts. We also use the quantum-inspired evolutionary algorithm for numerical optimization (QIEA-R) to determine the voting weights for each classifier which is part of the ensemble. Every time a new chunk of data arrives, a new classifier is trained on this new data set and all the weights are optimized in order for the ensemble to improve its performance on classifying this new set of data.

Therefore, we present a new approach for adaptive learning, consisting of an ensemble of neural networks, named NEVE (Neuro-Evolutionary Ensemble). To evaluate its performance and accuracy, we used 2 different datasets to execute several simulations, varying the ensemble settings and analysing how do they influence the final result. We also compare the results of NEVE with the results of Learn++.NSE algorithm [2], an existing approach to address adaptive learning problems.

This paper is organized in four additional sections. Section 2 details the Quantum-Inspired Neuro-Evolutionary algorithm and the proposed model, the Neuro-Evolutionary Ensemble Classifier. Section 3 presents and discusses the results of the experiments. Finally, section 4 concludes this paper and present some possible future works.

2 The Proposed Model

2.1 The Quantum-Inspired Neuro-Evolutionary Model

Neuro-evolution is a form of machine learning that uses evolutionary algorithms to train artificial neural networks. This kind of model is particularly interesting for reinforcement learning problems, where the availability of input-output pairs is often difficult or impossible to obtain and the assessment of how good the network per-

forms is made by directly measuring how well it completes a predefined task. As training the weights in a neural network is a non-linear global optimization problem, it is possible to minimize the error function by means of using an evolutionary algorithm approach.

The quantum-inspired evolutionary algorithm is a class of "estimation of distribution algorithm" (EDA) that has a fast convergence and, usually, provides a better solution, with fewer evaluations than the traditional genetic algorithms [3, 6]. In this model, quantum-inspired genes are represented by probability density functions (PDF) which are used to generate classical individuals through an observation operator. After being observed, the classical individuals are evaluated, as in traditional genetic algorithms, and, by means of using fitness information, a set of quantum-inspired operators are applied to the quantum individuals, in order to update the information they hold in such a way that on the next generations, better individuals will have a better chance to be selected. Further details on how this optimization method works can be found in [7-11].

Based on this algorithm, the proposed quantum-inspired neuro-evolutionary model consists in a neural network (a multilayer perceptron (MLP)) and a population of individuals, each of them encoding a different configuration of weights and biases for the neural network. The training process occurs by building one MLP for each classical individual using the genes from this individual as weights and biases. After that, the full training data set (or the set of tasks to be performed) is presented to the MLP and the average error regarding the data set is calculated for each MLP. This average error is used as the fitness for each individual associated to that MLP, which allows the evolutionary algorithm to adjust itself and move on to the next generation, when the whole process will be repeated until a stop condition is reached.

This subsection presented the quantum-inspired neuro-evolutionary model. This model will be the basis for the algorithm proposed in this paper, to be presented in the next subsection.

2.2 NEVE: The Neuro-Evolutionary Ensemble Classifier

To some applications, such as those that use data streams, the strategy of using simpler models is most appropriate because there may not be time to run and update an ensemble. However, when time is not a major concern, yet the problem requires high accuracy, an ensemble is the natural solution. The greatest potential of this strategy for detecting drifts is the ability of using different forms of detection and different sources of information to deal with the various types of change [4].

One of the biggest problems in using a single classifier (a neural network, for example) to address concept drift problems is that when the classifier learns a dataset and then we need it to learn a new one, the classifier must be retrained with all data, or else it will "forget" everything already learned. Otherwise, using the ensemble, there is no need to retrain it again, because it can "retain" the previous knowledge and still learn new data.

Hence, in order to be able to learn as new chunks of data arrive, we implemented an ensemble with neural networks that are trained by an evolutionary algorithm, pre-

sented in section 2.1. This approach makes the ensemble useful for online reinforcement learning, for example. The algorithm works as shown in figure 3 and each step is described in detail on the next paragraphs.

On step 1 we create the empty ensemble with a predefined size equal to *s*. When the first chunk of data is received, a neural network is trained by means of the QIEA-R until a stop condition is reached (for example, the number of evolutionary generations or an error threshold). If the number of classifiers in the ensemble is smaller than *s*, then we simply add this new classifier to the ensemble. This gives the ensemble the ability to learn the new chunk of data without having to parse old data. If the ensemble is already full, we evaluate each classifier on the new data set and we remove the one with the highest error rate (including the new one, which means the new classifier will only become part of the ensemble if its error rate is smaller than the error rate of one of the classifiers already in the ensemble). This gives the ensemble, the ability to forget about data which is not needed anymore.

1. Create an empty ensemble P
2. Define the ensemble size s
3. For each chunk of data $D_i, i = 1, 2, 3, \ldots, m$ do
3.1. Train the classifier using the QIEA-R and a MLP and calculate it's error E' over the data chunk
3.2. If the ensemble is full (number of classifiers $= s$) then
  i) Calculate the classification error E_j for each classifier c_j in the ensemble
  ii) If $(E' > max(E_j))$
    A) Replace the classifier with $max(E_j)$ by the new classifier
3.3. else
  i) Add the new classifier to the ensemble
3.4. Evolve the voting weights w_j for each classifier in the ensemble using the last chunk of data D_i

Fig. 1. The neuro-evolutionary ensemble training algorithm.

Finally, we use the QIEA-R to evolve a voting weight for each classifier. Optimizing the weights allows the ensemble to adapt quickly to sudden changes on the data, by giving higher weights to classifiers better adapted to the current concepts governing the data. The chromosome that encodes the weights has one gene for each voting weight, and the population is evolved using the classification error as the fitness function. It is important to notice that when the first *s-1* data chunks are received, the ensemble size is smaller than its final size and thus, the chromosome size is also smaller. From the s data chunk on, the chromosome size will remain constant and will be equal to *s*.

In this work, we used only binary classifiers but there is no loss of generality and the algorithm can also be used with any number of classes. For the binary classifier, we discretize the neural network's output as "1" or "-1" and the voting process for each instance of data is made by summing the NN's output multiplied by its voting weight. In other words, the ensemble's output for one instance *k* from the i-th data chunk is given by:

$$P(D_{ik}) = \sum_{j=0}^{s} w_j c_j(D_{ik}) \tag{1}$$

where $P(D_{ik})$ is the ensemble's output for the data instance D_{ik}, w_j is the weight of the j-th classifier and $c_j(D_{ik})$ is the output of the j-th classifier for that data instance. If $P(D_{ik}) < 0$, we assume the ensemble's output is "-1". If $P(D_{ik}) > 0$, we assume the ensemble's output is "1". If $P(D_{ik}) = 0$, we choose a class randomly.

3 Experimental Results

3.1 Datasets Description

In order to check the ability of our model on learning data sets with concept drifts, we used two different data sets (SEA Concepts and Nebraska also used at [2]) upon which we performed several simulations in different scenarios.

The SEA Concepts was developed by [12]. The dataset consists of 50000 random points in a three-dimensional feature space. The features are in the [0; 10] domain but only two of the three features are relevant to determine the output class. Class labels are assigned based on the sum of the relevant features, and are differentiated by comparing this sum to a threshold.

Nebraska dataset, also available at [13], presents a compilation of daily weather measurements from over 9000 weather stations worldwide by the U.S. National Oceanic and Atmospheric Administration since 1930s, providing a wide scope of weather trends. As a meaningful real world dataset, [2] choosed the Offutt Air Force Base in Bellevue, Nebraska, for this experiment due to its extensive range of 50 years (1949–1999) and diverse weather patterns, making it a longterm precipitation classification/prediction drift problem. Class labels are based on the binary indicator(s) provided for each daily reading of rain: 31% positive (rain) and 69% negative (no rain). Each training batch consisted of 30 samples (days), with corresponding test data selected as the subsequent 30 days. Thus, the learner is asked to predict the next 30 days' forecast, which becomes the training data in the next batch. The dataset included 583 consecutive "30-day" time steps covering 50 years.

3.2 Running Details

On each simulation, we used a fixed topology for the neural networks consisting of 3 inputs for SEA Concepts dataset and 8 inputs for Nebraska dataset, representing the input variables for each dataset. In both datasets, we used 1 output, and we varied the number of the neurons for the hidden layer. Each neuron has a hyperbolic tangent activation function and, as mentioned before, the output is discretized as "-1" or "1" if the output of the neuron is negative or positive, respectively. The evolutionary algorithm trains each neural network for 100 generations. The quantum population has 10 individuals and the classical population 20. The crossover rate is 0:9 (refer to [8, 9] for details on the parameters). The same parameters are used for evolving the weights for the classifiers. The neural network weights and biases and the ensemble weights are allowed to vary between -1 and 1 as those values are the ones who have given the best results on some pre-evaluations we have made.

The first experiment was conducted in order to evaluate the influence of the variation of the parameters values in the results (number of the hidden layer neurons and

the size of the ensemble). We used 4 different configurations for each dataset. After running 10 simulations for each configuration, we performed some an Analysis of Variance (ANOVA) [14]. In order to use ANOVA, we tested the Normality assumption for the noise term with Shapiro-Wilk's test [20] and the homogeneity of variances with Bartlett's test [14]. All the statistical procedures were conducted in R package [15], admitting a significance level of 5%.

The second experiment, in turn, aimed to compare the results found by NEVE and Learn++.NSE algorithms and we used, for each dataset, the best configuration found by first experiment (ensemble size and number of neurons at hidden layer values). After running one simulation for each dataset, we made statistical comparisons between the results found by NEVE and Learn++.NSE algorithms. The results of Learn++.NSE can be found at [2]. Then, to evaluate NEVE we made 10 runs for each dataset used, due to the stochastic optimization algorithm used to train NEVE. Based on these runs, we calculate some statistical parameters (mean, standard deviation, etc.) that were used to compute the Welch t-test [14] to evaluate which algorithm had, in average, the best performance in test phase. The normality assumption necessary for Welch t-test was verified using Shapiro-Wilk test [16]. All the statistical analyses were conducted in R statistical package [15].

3.3 First Experiment

Based on the past subsections, we made 40 simulations using SEA dataset and 40 simulations using Nebraska dataset, using 4 different configurations on each dataset. Table 1 displays number of neurons at hidden layer and ensemble size with different levels (5 and 10) and the output (average error in test phase for 10 runs) for each configuration.

In order to evaluate which configuration provided a significant lower error, we have to perform multiple comparisons between the results of each configuration. For each dataset if we decide to use t-test [14] for example, we have to realize 6 comparisons between the configurations, and thus, the probability that all analysis will be simultaneous correct is substantially affected. In this way, to perform a simultaneous comparison between all configurations we fitted a one-way Analysis of Variance (ANOVA) [14] for each dataset, described by:

$$Y_{ij} = \mu + CF_j + \varepsilon_{ij}; \;\; \varepsilon_{ij} \sim N(0, \sigma^2) \tag{1}$$

where Yij is the i-th ouput for the j-th configuration, μ is the global mean, CFj is the run configuration with j-levels (j =1,2,3,4) for each dataset (A, B, C and D, and E, F, G, H, for SEA and Nebraska respectively) and ε_{ij} is the noise term, Normal distributed with mean zero and constant variance (σ^2). If CF_j is statistically significant, then some configuration demonstrated an average error different from the others. To verify which configuration has the average error less than other configuration we used Tukey's test [14].

Table 1. Results for SEA and Nebraska dataset.

SEA dataset				
Neurons Number	Ensemble Size	Config	Error in test phase	
			Mean	Std. dev.
10	10	A	24.99%	0.17%
5	5	B	24.88%	0.19%
10	5	C	25.06%	0.21%
5	10	D	24.75%	0.17%
Nebraska dataset				
Neurons Number	Ensemble Size	Config	Error in test phase	
			Mean	Std. dev.
10	10	E	32.30%	0.48%
5	5	F	32.85%	0.43%
10	5	G	33.04%	0.37%
5	10	H	32.10%	0.46%

Then, we performed the analysis for both dataset and exhibit the main results in Table 2.

Table 2. Results for SEA and Nebraska dataset.

ANOVA - SEA dataset		
Method	Test Statistic	p-value
Bartlett's test	0.4443	0.9309
Config	5.4360	0.0035
Shapiro-Wilk's test	0.9260	0.2137
ANOVA - Nebraska dataset		
Method	Test Statistic	p-value
Bartlett's test	0.6537	0.8840
Config	11.5900	< 0.0001
Shapiro-Wilk's test	0.9708	0.3803

Analyzing the results displayed in Table III, in both datasets the errors variance is homogeneous (Bartlett's test, p-value > 0.05). After verifying these two assumptions (Normal distribution and homogeneity of variances), we fitted the one-way ANOVA. In both datasets, some configurations (A, B, C and D for SEA, and E, F, G and H for Nebraska) demonstrated an average error different from the others (p-value < 0.05). In addition, in both fitted models, the noise term follows a Normal distribution (Shapiro-Wilk's test, p-value > 0.05).

In order to identify which configuration performed, in average, better than other, we made Tukey's test for difference of means. Table 3 present the results of this analysis.

Table 3. Tukey's test for SEA and Nebraska dataset.

SEA dataset	**Nebraska dataset**

Config	Mean difference	p-value	config	Mean difference	p-value
A-B	0.11%	0.5665	E-F	-0.55%	**0.0145**
A-C	-0.07%	0.8018	E-G	-0.74%	**0.0013**
A-D	0.24%	**0.0317**	E-H	0.20%	0.8849
B-C	-0.18%	0.1399	F-G	-0.19%	0.7434
B-D	0.13%	0.4010	F-H	0.75%	**0.0014**
C-D	0.31%	**0.0029**	G-H	0.94%	**0.0001**

It seems that in SEA dataset the D configuration performed significantly better than A and C, although its results is not statistically different than configuration B. In Nebraska the E and H configurations obtained error measures substantially lower than F and G. In fact, we can choose the configuration D as the best configuration to SEA and H to Nebraska dataset, considering the fixed parameters displayed in table II. This choice is based on two criterias: lower average error and computational cost to train these models.

3.4 **Second Experiment**.

Aiming to enable a better comparison with the results of the algorithm Learn +. NSE [2] in SEA Concepts dataset, we used 200 blocks of size 25 to evaluate the algorithm in the test phase. The best configuration previously achieved was 5 neurons in hidden layer and the size of the ensemble equal to 5 (see table 4). Also, with the results of Learn +. NSE in Nebraska dataset, we performed similarly to that used in [2]. The best configuration previously achieved was 10 neurons in hidden layer and the size of the ensemble equal to 10. Then, NEVE and Learn++.NSE results were displayed in Table 4.

Table 4. Results of SEA and Nebraska experiments.

Dataset	Algorithm	Mean	Standard Deviation
SEA	NEVE	98.21%	0.16%
	Learn++.NSE (SVM)	96.80%	0.20%
Nebraska	NEVE	68.57%	0.46%
	Learn++.NSE (SVM)	78.80%	1.00%

As can be seem, the mean accuracy rate of Learn++.NSE is lower than the best configuration of NEVE, and thus this difference is statistically significant (tcrit = -41.07, p-value < 0.0001), demonstrating that NEVE perrformed better in the test phase on SEA Concepts dataset.

However, in Nebraska the mean accuracy rate of NEVE is lower than the best configuration of Learn++.NSE, and thus this difference is statistically significant (tcrit = 18.26, p-value < 0.0001), demonstrating that NEVE performed better, in average, than Learn++.NSE. Figures 4 and 5 illustrates the hit rate on each test block obtained by NEVE on SEA and Nebraska, respectively.

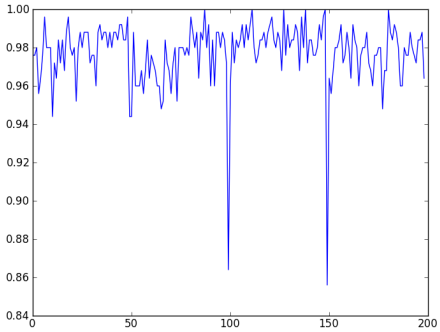

Fig. 4. Evolution of NEVE hit rate in SEA testing set.

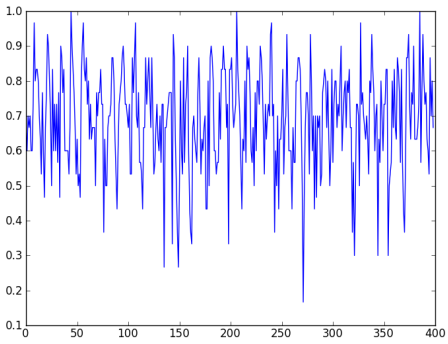

Fig. 5. Evolution of NEVE hit rate in Nebraska testing set.

4 Conclusions and Future Works

This paper presented a model that uses an ensemble of neural networks trained by a quantum-inspired evolutionary algorithm to learn data sets (possibly with concept drifts) incrementally. We analyzed the ability of the model using two different datasets and conducting two different experiments. In the first experiment, we found a good configuration for both datasets and demonstrated how the number of neurons and the ensemble size affect the average error produced by the model. As stated in the results, the ensemble size affected almost two times more the results of NEVE than the number of neurons. In the second experiment, the NEVE algorithm have demonstrated a better performance in SEA dataset compared to Learn++.NSE and yet lower accuracy when comparing with Learn++.NSE in Nebraska dataset.

Although the NEVE algorithm have demonstrated satisfactory performance for the datasets used in the analysis of this study, it is strongly recommended to perform further tests - using different configurations, different datasets and performing different analysis - to confirm the results presented here. We also intend in the future to continue this work, analyzing other existing approaches, such as [17] and [18], and performing new experiments in comparison with these and other algorithms. We still need to investigate other factors related to QIEA-R fine tunning (genetic operators, population size, etc.).

5 References

1. Schlimmer, J. C., Granger, R. H.: Incremental learning from noisy data. Machine Learning, vol. 1, no. 3, pp. 317–354 (1986).
2. Elwell, R., Polikar, R.: Incremental Learning of Concept drift in Nonstationary Environments. IEEE Transactions on Neural Networks 22(10): 1517-1531 (2011).

3. Kuncheva, L. I.: Classifier ensemble for changing environments. in Multiple Classifier Systems, vol. 3077. Springer-Verlag, New York. (2004).
4. Kuncheva, L. I.: Classifier ensemble for detecting concept change in streaming data: Overview and perspectives. Proc. Eur. Conf. Artif. Intell, pp. 5–10 (2008).
5. Ahiskali, M. T., M., Muhlbaier, M., Polikar, R.: Learning concept drift in non-stationary environments using an ensemble of classifiers based approach. IJCNN, pp. 3455–3462 (2008).
6. Oza, N. C.: Online Ensemble Learning. Dissertation, University of California, Berkeley, (2001).
7. Abs da Cruz, A. V., Vellasco, M. M. B. R., Pacheco, M. A. C.: Quantum-inspired evolutionary algorithms for numerical optimization problems. Proceedings of the IEEE World Conference in Computational Intelligence (2006).
8. Abs da Cruz, A. V.: Algoritmos evolutivos com inspiração quântica para otimização de problemas com representação numérica. Ph.D. dissertation, Pontifical Catholic University – Rio de Janeiro (2007).
9. Han, K.-H., Kim, J.-H.: Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. IEEE Trans. Evolutionary Computation, vol. 6, no. 6, pp. 580–593 (2002).
10. Han, K.-H., Kim, J.-H.: On setting the parameters of QEA for practical applications: Some guidelines based on empirical evidence. GECCO, Lecture Notes in Computer Science Volume 2723, 2003, pp 427-428 (2003).
11. Han, K.-H., Kim, J.-H.: Quantum-inspired evolutionary algorithms with a new termination criterion, He gate, and two-phase scheme. IEEE Trans. Evolutionary Computation, vol. 8, no. 2, pp. 156–169 (2004).
12. Street, W. N., Kim, Y.: A streaming ensemble algorithm (SEA) for large-scale classification. Proc. 7th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min., pp. 377–382 (2001).
13. Polikar, R., Elwell, R.: Benchmark Datasets for Evaluating Concept drift/NSE Algorithms. At: http://users.rowan.edu/~polikar/research/NSE. Last access at December 2012.
14. Montgomery, D. C. Design and analysis of experiments. Wiley (2008).
15. R Development Core Team. R: A language and environment for statistical computing. R Foundation for Statistical Computing. Vienna, Austria. Donwload at: www.r-project.org, 2012.
16. Royston, P.: An extension of Shapiro and Wilk's W test for normality to large samples. Applied Statistics, vol. 31, pp. 115–124 (1982).
17. Kolter, J., Maloof, M.: Dynamic weighted majority: An ensemble method for drifting concepts. Journal of Machine Learning Research, 8, pp. 2755-2790 (2007).
18. Jackowski, K.: Fixed-size ensemble classifier system evolutionarily adapted to a recurring context with an unlimited pool of classifiers. Pattern Analysis and Applications (2013).