Using Neuroevolution for Predicting Mobile Marketing Conversion

Pedro José Pereira[1], Pedro Pinto[2], Rui Mendes[2], Paulo Cortez[1], and Antoine Moreau[3]

[1] ALGORITMI Centre, Department of Information Systems, University of Minho, 4804-533 Guimarães, Portugal
id6927@alunos.uminho.pt, pcortez@dsi.uminho.pt
[2] ALGORITMI Centre, Department of Informatics, University of Minho, 4710-057 Braga, Portugal
a71929@alunos.uminho.pt,rcm@di.uminho.pt
[3] OLAmobile, Spinpark, 4805-017 Guimarães, Portugal
antoine.moreau@olamobile.pt

Abstract. This paper addresses user Conversion Rate (CVR) prediction within the context of Mobile Performance Marketing. Specifically, we adapt two main neuroevolution methods: Neuroevolution of Augmenting Topologies (NEAT) and Hypercube-based NEAT (HyperNEAT). First, we discuss two mechanisms for increasing execution speed (parallelism and data sampling); a strategy for preventing excessive network complexity with NEAT; and a rolling window scheme for performing an online learning. Then, we present experimental results, using distinct datasets and testing both offline and online learning environments.

Keywords: Marketing · Classification · Neuroevolution.

1 Introduction

The massive usage of portable computing devices (e.g., tablet, smartphone) increased the value of mobile markets, giving rise to Demand-Side Platforms (DSPs). A DSP is a broker that matches users to advertisements and involves users, publishers and advertisers. Publishers attract a vast audience of users, which want to access a popular content web site (e.g., games or news portal). The web site is funded by requiring users to click a dynamic ad link before accessing the content. The goal of the DSP is to select the ad to be displayed to the user. If there is a product or service acquisition (a conversion), then the DSP automatically returns a portion of the advertiser profit to the publisher.

In this paper, we approach the Conversion Rate (CVR) task [5], aiming to predict if a user will produce a conversion when seeing an ad. Such prediction is a key tool to assist the DSP in better assigning ads to users. The CVR task has been approached using several machine learning models, mostly linear models, such as the linear Poisson regression [3] or Logistic Regression (LR) [5]. More

sophisticated methods, such as Gradient Boosting Decision Trees [16], Random Forest [5], XGboost [11] or Deep Learning [15], have also been proposed.

This paper describes the implementation of a data-driven approach for CVR user prediction, with application to a real-world DSP, managed by Olamobile, which is a mobile marketing worldwide company. Specifically, we explore *neuroevolution* algorithms, which use Evolutionary Algorithms (EAs) to design and fit Artificial Neural Networks (ANNs). An important advantage of these methods is the ability to automatically optimize the topology and weights of the networks [7]. The automatic design of ANN is particularly valuable in this marketing domain, since data is created with high velocity and there are several dynamic changes (e.g., new campaigns, changes in online user buying behaviors). Thus, new data-driven models need to be constantly created, which is clearly facilitated by the usage of automatic data-driven model selection procedures. Within our knowledge, the application of neuroevolution to Mobile Performance Marketing is non-existent. Moreover, several related works tend to consider only prediction classification measures and not the computational effort. For example, the deep learning method used in [15] is much more complex than LR and the classification performance of the deep learning models only improved very slightly (e.g., 0.1 percentage points) when compared with LR. Also, several related studies (e.g., [16,5,10,15]) only address static offline learning scenarios, with a single holdout train and test split.

In this paper, we adapt and compare two neuroevolution algorithms for CVR prediction: NeuroEvolution of Augmenting Topologies (NEAT) [13] and Hypercube-based NEAT (HyperNEAT) [12]. We test the algorithms with two categorical data transforms, two traffic modes (TEST and BEST), and with static and dynamic environments, measuring the predictive classification performance and computational effort. This document is organized as follows: Section 2 presents the collected data and neuroevolution methods; Section 3 details the performed experiments and obtained results; finally, Section 4 describes the main conclusions.

2 Materials and Methods

2.1 Collected Data

The analyzed real-world DSP produces redirects and sales data events. Redirects are created each time a user clicks an ad, while a sale is produced when there is a product acquisition. CVR is modeled using binary classification. This task is complex since the DSP generates big data, with high volume and velocity properties. There are millions of redirects and thousands of sales per hour. Moreover, only a small fraction of redirects lead to sales. Also, only a partial set of characterization features are available due to privacy and technological constraints (e.g., it is not possible to identify a single user). And the nominal input features often present a high cardinality, with hundreds or thousands of distinct levels.

We had access to an Intel Xeon 1.70GHz server with 56 cores and 2TB of disk space, which is limited when compared with the DSP datacenter. Due to

server limitations (e.g., storage, communication costs), we work with sampled data, retrieved from the DSP datacenter over a two-week period, from 2018-05-30 to 2018-06-13. The data includes redirects and sales related with two traffic modes: TEST and BEST. The former is used to test the performance for new incoming campaigns (20% of the traffic), while the latter includes only the best TEST performing campaigns (80%). The sampled data contains 484,665 BEST mode observations, of which 156,637 (32.3%) were sales, and the TEST dataset contains 319,875 observations, of which only 29,596 (9.25%) resulted in sales.

The collected eleven input features are summarized in Table 1, partially characterizing the advertiser, publisher and user. The table details the cardinality (number of levels) for each feature and traffic mode. The datasets contain two other attributes (not shown in the table): a time-stamp – when the sale or redirect occurred; and the target – binary variable with the sale (1) or no sale (0) label.

Table 1: Features.

Feature	Description	Cardinality		Examples
		BEST	TEST	
campaign	advertisement campaign	1389	1741	Numeric ID
vertical	advertisement type	5	4	`Video`, `Mainstream`
application	advertised product	1018	1101	Numeric ID
partner	publisher	167	200	Numeric ID
account	publisher type	8	9	`Network`, `Developer`
manager	publisher account manager	19	34	Numeric ID
operator	user mobile carrier or WiFi	404	448	`Vodafone`, `WiFi`
browser	user web browser	14	14	`Chrome`, `Safari`
region	user region	23	23	`Asia`, `South America`
country	user country	198	225	`India`, `Brazil`
city	user city	13423	10690	`Dhaka`, `Sao Paulo`

All features are nominal, including the numeric identifiers. Since the neural network base learner requires numeric inputs, we compare two feature handling modes: RAW and Inverse Document Frequency (IDF). RAW uses original numeric identifier raw values. For features that contain text, Raw Encoding (RAW) converts each category into a number $1..N$, where N is the cardinality of the feature, by order of appearance. The IDF encodes each level as $IDF(x) = ln(\frac{N}{f_x})$, where N is the total number of instances, and f_x is the number of occurrences of category x [2]. The levels are ranked according to their frequency, with values that are more frequent being closer to 0, and those that are less frequent ranging up to a maximum value of $ln(N)$, for $f_x = 1$. The transformations are performed using only training data, with an encoding mapping being stored in

order to transform test data values. Any unseen input value is transformed in a special way, depending on the encoding: with RAW, it takes the value 0; and with IDF, it attains the maximum value $ln(N)$.

2.2 Neuroevolution Models

The predictive models employ an ANN trained by an EA, aiming to estimate the target probability: $p(1|\mathbf{x}) \in [0, 1]$, where $\mathbf{x}$ denotes the input vector for a particular redirect. Two neuroevolution algorithms are tested: NEAT [13] and HyperNEAT [12]. To implement the algorithms, we used the modern MultiNEAT library (`http://multineat.com`), which includes recent deep learning features, such as usage of the ReLU activation function [8] (see Table 2).

NEAT is a popular neuroevolution technique with three main characteristics: tracking of genes through historical markings; protection of innovation through speciation; and minimization of dimensionality through incremental growth from a minimal structure [13]. NEAT uses a direct encoding, where individuals contain every connection of the ANN. In contrast, HyperNEAT [12] uses an indirect encoding, allowing to evolve large-scale ANNs. In HyperNEAT, the individual is a Compositional Pattern-Producing Network (CPPN), an intermediate neural network which is used to generate the weights of the final network connections. The method requires a grid of nodes (neurons), called the *substrate*, to be previously defined by the user. Then, for each potential connection in the substrate, the CPPN takes as inputs the geometric positions of the two neurons and outputs the connection weight. A connection is not expressed if the magnitude of its weight is below a minimal threshold.

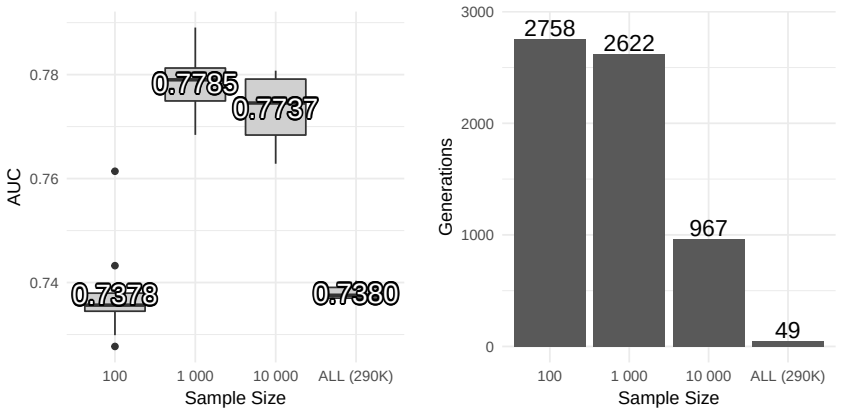
These neuroevolution methods share the same EA, which includes two phases in each generation: evolution and evaluation. The evolution uses the selection, crossover and mutation operators that are applied to generate a new population. The evaluation requires the highest computational effort and it is based on the Area Under the ROC Curve (AUC) of the Receiver Operating Characteristic (ROC) [6], computed using the ANN individual predictions.

To speed up the evaluation, two mechanisms were implemented: parallel evaluation of each individual of the population and sampling of the training data. Since the evaluation of each individual of the population is independent, each AUC calculation is executed as a parallel task that is run in a unique core. Moreover, the fitness computation is applied only to a random sample of data. We note that working with the full data would require a high computational effort and in particular a computational effort would be "wasted" to compute the fitness of very weak solutions. The sampling procedure works as follows: in every generation, a balanced sample (with both sale and no sale redirects) of a predefined size is randomly selected from the whole training dataset. All individuals are then evaluated over the same sample. Balanced sampling is used to avoid classifiers that are too biased towards the more prevalent "no sale" class.

The sampling calls into question which individual should be returned at the end of the execution, since the fitness scores represent the performance of individuals over a portion and not all of the training data. This issue is addressed

by storing in memory the best individuals along the generations: the *elite*. When the termination criterion is met, an extra evaluation is performed over the elite using the whole training dataset, aiming to select the best ANN.

The sample size becomes an extra hyperparameter of the algorithm. The optimum sample size should be small enough to provide a fast execution speed, allowing for a high number of generations to be completed, but not so small that it hurts the algorithm capability of adjusting to the training data. The trade-off is shown in Figure 1, where the two extremes on the low and high end of four predefined sample sizes provide the worst results. Note that, for a meaningful comparison, the execution time was the same for all sample sizes (total of 20 minutes).



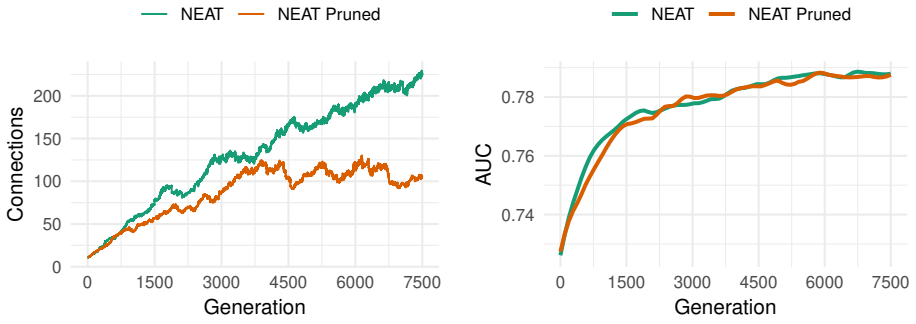
(a) Test AUC of best individuals (average of multiple runs).

(b) Average number of generations completed.

Fig. 1: Comparison of sample sizes.

There is a tendency for network complexity to increase with the number of generations. This growth is expected, and positive, as long as it leads to significantly better networks. However, this *bloat* phenomenon, if not controlled or limited, results in an ever increasing computational effort for both the evolutionary algorithm and the processing of network predictions (Figure 2a). To limit bloat, we dynamically adjust the mutation rates for addition and removal of neurons and connections. This strategy works by introducing a simplification phase whenever the mean complexity of the population overcomes a predefined limit. During the simplification phase, the probability of mutations that add complexity (i.e., neurons or connections) is gradually decreased, while the probability of mutations that remove complexity is increased by the same amount. Once the complexity of the population is diminished, the simplification phase ends. and the default behaviour of the EA is resumed. We employ this strategy over NEAT, calling it NEAT Pruned (NEATP), with the simplification phase starting

at 100 connections. A simplification strategy cannot be applied to HyperNEAT, because it uses an indirect encoding. A comparison between NEAT and NEATP is shown in Figure 2, revealing that NEATP limits the network complexity with no significant impact on performance.



(a) Population complexity.

(b) AUC of the best individual.

Fig. 2: Bloat control by adjustment of mutation rates.

The neuroevolution hyperparameters were optimized by using the irace tool [9]. Irace samples various configurations of hyperparameters and tests how the algorithm performs with them, according to the AUC of the best individual. To avoid overfitting, the AUC given to irace was calculated over a separate dataset, with earlier sampled DSP data. The final hyperparameters obtained for both NEAT and HyperNEAT are shown in Table 2. We note that the sample size value of $2,000$ is consistent with Figure 1a results.

Table 2: Hyperparameters set using the irace tool.

Parameter	NEAT	Hyper-NEAT	Parameter	NEAT	Hyper-NEAT
sample size	2 000	2 000	add neuron rate	3.25%	8.04%
substrate hidden layers	–	8	remove neuron rate	2.30%	6.68%
substrate neurons per layer	–	23	add link rate	13.3%	9.56%
population size	120	145	remove link rate	9.1%	4.65%
min species	6	4	mutation weight rate	64.8%	64.6%
max species	10	11	mutation bias rate	6.7%	5.8%
survival rate	66.8%	34.4%	mutation activation rate	0.4%	0.6%
crossover rate	89.0%	74.1%	Sigmoid neuron rate	33.4%	50%
interspecies rate	0.25%	0.23%	Relu neuron rate	33.3%	0%
mutation rate	34.8%	69.2%	Gaussian neuron rate	33.3%	0%
elitism	2.5%	2.8%	Sine neuron rate	0%	50%

2.3 Evaluation

We test both static (offline) and dynamic (online learning) scenarios. We use the AUC classification metric [6], since it is a popular metric in CVR [5,15]. The AUC metric is independent of false positive and negative costs, which might not be known during the training phase; also, it is independent of the class distribution, thus it can be used with highly unbalanced tasks, such as the CVR data. The quality of a AUC value is often interpreted as: 0.5, the performance of a random classifier; 0.6 to 0.7, reasonable; 0.7 to 0.8, good; 0.8 to 0.9, very good; 0.9 to 1, excellent.

First, we compare NEAT, NEATP and HyperNEAT over the two types of numerical transformations (RAW and IDF) with static data, using a simpler holdout validation, in which the data is randomly split into train (70%) and test (30%) sets. The algorithms run for 10,000 generations, with 6 parallel processes being used for the evaluations. Then, we test a dynamic scenario by using a rolling window scheme [14]. In the first iteration, the last $W = 4$ days of data are used to fit the model, which is tested to predict the next $T = 1$ day events. After a predefined number of generations, there is a shift in time, which results in the second iteration. It is assumed that one day has passed, thus the training data slides $S = 1$ day. The neuroevolution population is continuously adjusted to the new training data and then predictions are computed for the next $T = 1$ day. And so on.

The continuous update of data requires the IDF transform to be updated after each rolling window iteration. The two inputs of IDF – f_x (frequency of category x) and N (size of the data sample) – are then calculated as a weighted average over time:

$$f_x^t = \lambda \cdot f_x^{t-1} + f_x^{new} \tag{1}$$

$$N^t = \lambda \cdot N^{t-1} + N^{new} \tag{2}$$

where the index t represents the rolling window iteration; f_x^{new} and N^{new} are the frequency of category x and size of the data sample over the latest window; and λ is a coefficient within the range $[0, 1]$ used to progressively "forget" information from past iterations. In this work, and after some experimentation with an older dataset (collected before 30th May of 2018), λ was set to 0.8.

3 Results

The progression of fitness along the generations is represented in Figure 3, and a summary of the results obtained is given in Table 3. To establish a baseline, the performance of a Logistic Regression (LR) model is also presented, implemented using the Broyden–Fletcher–Goldfarb–Shanno (BFGS) offline learning algorithm of the **rminer** R package [4].

Regarding the two numerical transformations, IDF presents a slight improvement over RAW in terms of the AUC metric for all BEST traffic cases and also HyperNEAT and TEST data. Focusing on the algorithms, it is clear that HyperNEAT does not perform as well as NEAT, which can be due to two reasons.

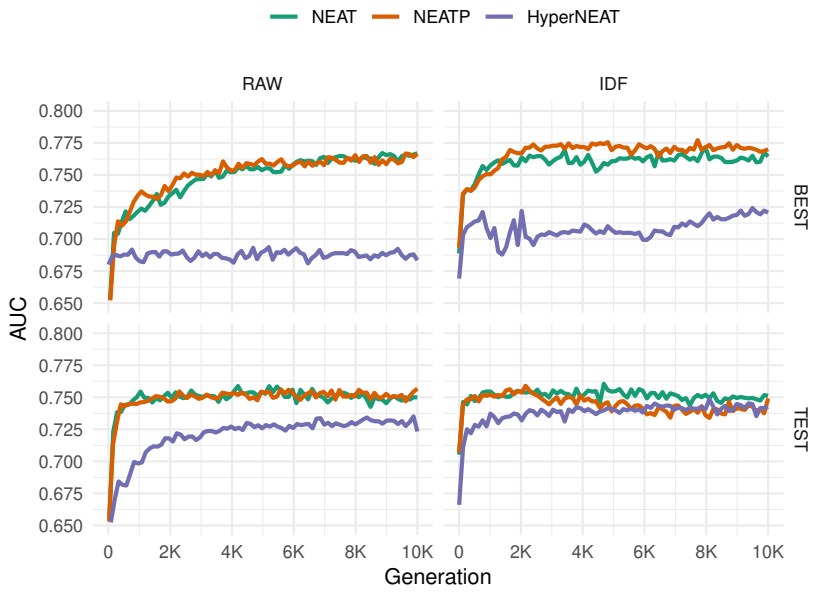


Fig. 3: Fitness of the best individual per generation with static data (top graphs are related with BEST traffic, while bottom plots consider TEST traffic).

First, HyperNEAT might be more suitable to problems with geometric relationships among inputs [12]. Second, results suggest that large-scale networks do not seem to significantly increase AUC, as evidenced by the similar performances of NEAT and NEATP. And HyperNEAT tends to evolve more complex networks when compared with NEAT. Overall, NEAT and NEATP yield the best results. In particular, NEATP successfully maintained the ANN complexity near the predefined limit of 100 connections without any significant decrease in AUC, which reflects in a shorter training time. When compared to the LR model, the neuroevolution algorithms present better predictive AUC results for BEST data (e.g., 0.78 versus 0.74) and a similar discrimination level for TEST mode data (AUC of 0.76).

We only compare the neuroevolution models in the dynamic rolling window scenario because the standard LR model only works in an offline learning scenario. The neuroevolution results are compared in Figure 4, which plots the test AUC value of the best ANN per rolling window iteration; and in Table 4, as the median of all iterations.

The performance of the algorithms and numerical transformations with dynamic data is similar to their performance with static data. The highlight of these results, and focusing in particular in Figure 4, is that the algorithms are capable of building upon previous training in order to adjust to new data, while maintaining a steady (TEST data) or even improved (BEST traffic) performance.

Table 3: Results with static data (averaged over 10 runs).

Mode	Algorithm	AUC[1]		Time[2]		Complexity[3]	
		RAW	IDF	RAW	IDF	RAW	IDF
BEST	NEAT	**0.77**	**0.78**	81	107	31 · 54	40 · 103
	NEATP	**0.77**	**0.78**	81	85	**29 · 50**	**31 · 50**
	HyperNEAT	0.70	0.74	215	225	184 · 620	184 · 355
	LR	0.71	0.74	**0.15**	**0.15**	-	-
TEST	NEAT	**0.76**	**0.76**	94	157	53 · 139	79 · 281
	NEATP	**0.76**	**0.76**	88	100	**41 · 96**	**56 · 109**
	HyperNEAT	0.73	0.75	198	216	184 · 316	184 · 643
	LR	**0.76**	0.75	**0.15**	**0.15**	-	-

[1] Calculated over the test set.
[2] Total training time in minutes.
[3] Complexity (nodes · connections) of the returned network.

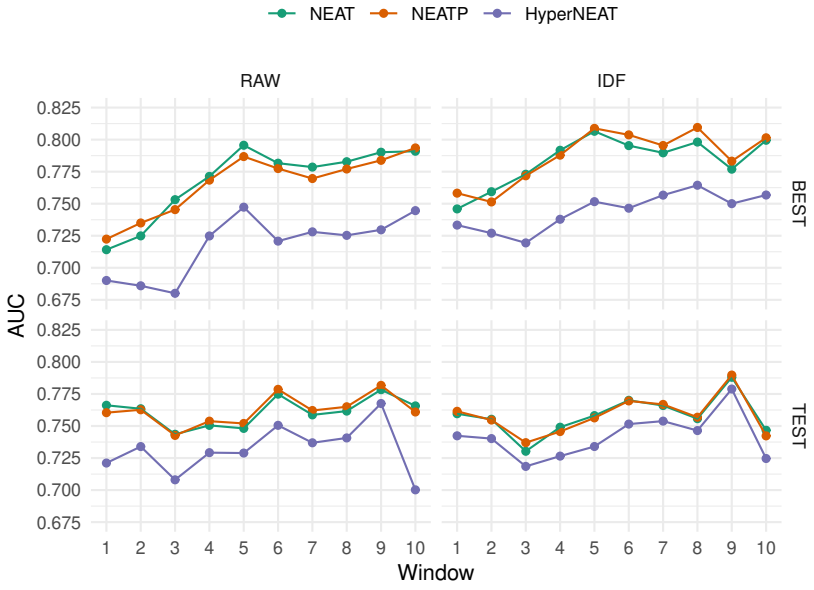


Fig. 4: AUC per rolling window iteration (top graphs are related with BEST traffic, while bottom plots consider TEST traffic).

Thus, it is possible to update the predictive model according to the latest set of data using a relatively small number of generations. In this case, the total of 10,000 generations is split over 10 rolling window iterations, each, therefore, running for 1,000 generations. Consequently, although the total training time is

Table 4: Results with dynamic data (median of 10 rolling window iterations and averaged over 10 runs).

Mode	Algorithm	AUC		Time		Complexity	
		RAW	IDF	RAW	IDF	RAW	IDF
BEST	NEAT	**0.78**	**0.79**	**8**	11	$23 \cdot 60$	$29 \cdot 124$
	NEATP	**0.78**	**0.79**	**8**	**9**	$\mathbf{19 \cdot 57}$	$\mathbf{22 \cdot 65}$
	HyperNEAT	0.73	0.75	21	23	$184 \cdot 315$	$184 \cdot 691$
TEST	NEAT	**0.76**	**0.76**	9	14	$24 \cdot 89$	$38 \cdot 156$
	NEATP	**0.76**	**0.76**	**8**	**10**	$\mathbf{24 \cdot 82}$	$\mathbf{31 \cdot 105}$
	HyperNEAT	0.73	0.74	20	21	$184 \cdot 340$	$184 \cdot 643$

about the same as with static data, the training time per rolling window iteration is $\frac{1}{10}$ of that amount: about 10 minutes with NEAT or NEATP, as shown in Table 4. This allows for new data to be learned very quickly, in the context of a dynamic environment.

The best predictive classification performances were achieved by the NEAT and NEATP models. As for the input attribute transformation methods, IDF improves the AUC values when compared with the RAW encoding for: BEST traffic – all methods; and TEST traffic – HyperNEAT. As for the computational effort, NEATP is the fastest method, requiring around just 8 to 10 minutes for each training, followed by NEAT and then HyperNEAT.

Considering the predictive accuracy and computational effort, we select IDF NEATP as the best neuroevolution strategy. Globally, interesting results were achieved, with an average (over 10 runs) rolling window median AUC of 0.79 (BEST) and 0.76 (TEST), which corresponds to a good discrimination level. This level compares favourably with other similar state of the art CVR prediction works, with an average AUC value of: 0.71 (LR) and 0.72 (random forest) in [5]; 0.76 (XGboost) and 0.80 (random forest) in [11]; and 0.71 (Deep Learning model) in [15]. In particular, the analyzed DSP currently employs a random user to advertisement matching when working with the TEST mode traffic, which corresponds to an AUC of 0.5. The NEATP classification performance is 26 percentage points better when compared with the random DSP assignment for new marketing campaigns. Also, NEATP can handle big data and produce daily predictions in real-time.

4 Conclusions

In this paper, we address user mobile marketing CVR prediction. As a case study, we had access to recent big data gathered from a real-world DSP company. We particularly focus on neuroevolution models, which present the advantage of automatically designing the ANN topology and weights. We compared two neuroevolution algorithms that automatically design ANN for CVR prediction: NEAT and HyperNEAT. NEAT uses a direct encoding, while HyperNEAT

employs an indirect encoding. We also compared two categorical to numerical transformations and two learning scenarios: static (offline) and dynamic (online). The prediction models were compared using both predictive classification performance and computational effort.

Considering the classification performance, computational effort and bloat, the best results were obtained by the NEATP model (a NEAT variant that limits the ANN growth). It produces better classification discrimination results when compared with HyperNEAT and an offline Logistic Regression (LR) in the static experiments. Also, it produces a steady or improved performance in the dynamic experiments, comparing favourably against HyperNEAT. Overall, a good classification discrimination level was obtained, resulting in a Area Under the ROC Curve (AUC) of 0.79 for BEST and 0.76 for TEST traffic. Moreover, under the tested experimental setup, NEATP requires a training time around 10 minutes, allowing its daily usage to perform real-time predictions. This model is particularly valuable for the TEST traffic, since the analyzed DSP uses a random selection of advertisements for new incoming TEST campains, which corresponds to an AUC of 0.5.

As future work, we wish to compare the proposed neuroevolution approaches with deep learning methods (e.g., Deep Feedforward Neural Network), using both classification performance and computational effort measures. Also, we intend to extend the proposed neuroevolution methods by exploring the use of a local search, based on gradient descent (e.g., backpropagation), to further tune the ANN connection weights. In particular, we aim to explore in the dynamic DSP learning what is the best mixed global and local search strategy: if the improved connections should be encoded back into the EA chromosomes (Lamarckian evolution) or not (Baldwin effect) [1].

Acknowledgements

This article is a result of the project NORTE-01-0247-FEDER-017497, supported by Norte Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, through the European Regional Development Fund (ERDF). This work was also supported by FCT – Fundação para a Ciência e Tecnologia within the Project Scope: UID/CEC/00319/2019.

References

1. Bereta, M.: Baldwin effect and lamarckian evolution in a memetic algorithm for euclidean steiner tree problem. Memetic Computing pp. 1–18 (2018)
2. Campos, G.O., Zimek, A., Sander, J., Campello, R.J., Micenková, B., Schubert, E., Assent, I., Houle, M.E.: On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. Data Mining and Knowledge Discovery **30**(4), 891–927 (2016)
3. Chen, Y., Pavlov, D., Canny, J.F.: Large-scale behavioral targeting. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009. pp. 209–218

(2009). https://doi.org/10.1145/1557019.1557048, http://doi.acm.org/10.1145/1557019.1557048
4. Cortez, P.: Data mining with neural networks and support vector machines using the r/rminer tool. In: Perner, P. (ed.) Advances in Data Mining. Applications and Theoretical Aspects, 10th Industrial Conference, ICDM 2010, Berlin, Germany, July 12-14, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6171, pp. 572–583. Springer (2010). https://doi.org/10.1007/978-3-642-14400-4_44, https://doi.org/10.1007/978-3-642-14400-4_44
5. Du, M., Brorsson, M., Avenesov, T., State, R.: Behavior profiling for mobile advertising. In: Proceedings of the 3rd IEEE/ACM International Conference on Big Data Computing, Applications and Technologies. pp. 302–307. ACM, New Shanghai, China (2016)
6. Fawcett, T.: An introduction to ROC analysis. Pattern Recognition Letters **27**, 861–874 (2006)
7. Floreano, D., Dürr, P., Mattiussi, C.: Neuroevolution: from architectures to learning. Evolutionary Intelligence **1**(1), 47–62 (2008)
8. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: Gordon, G.J., Dunson, D.B., Dudík, M. (eds.) Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011. JMLR Proceedings, vol. 15, pp. 315–323. JMLR.org (2011), http://www.jmlr.org/proceedings/papers/v15/glorot11a/glorot11a.pdf
9. López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L.P., Birattari, M., Stützle, T.: The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives **3**, 43–58 (2016)
10. Lu, Q., Pan, S., Wang, L., Pan, J., Wan, F., Yang, H.: A practical framework of conversion rate prediction for online display advertising. In: Proceedings of the ADKDD'17. p. 9. ACM, Halifax, Canada (2017)
11. Matos, L., Cortez, P., Mendes, R., Moreau, A.: A comparison of data-driven approaches for mobile marketing user conversion prediction. In: Proceedings of the 9th IEEE International Conference on Intelligent Systems (IS 2018). IEEE, Funchal, Madeira, Portugal (Sep 2018)
12. Stanley, K.O., D'Ambrosio, D.B., Gauci, J.: A hypercube-based encoding for evolving large-scale neural networks. Artificial life **15**(2), 185–212 (2009)
13. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary computation **10**(2), 99–127 (2002)
14. Tashman, L.: Out-of-sample tests of forecasting accuracy: an analysis and review. International Forecasting Journal **16**(4), 437–450 (2000)
15. Zhang, W., Du, T., Wang, J.: Deep learning over multi-field categorical data. In: European conference on information retrieval. pp. 45–57. Springer, Padua, Italy (2016)
16. Zhang, W., Yuan, S., Wang, J.: Real-time bidding benchmarking with ipinyou dataset. CoRR **abs/1407.7073** (2014)