Scale Genetic Programming for large Data Sets: Case of Higgs Bosons Classification

Hmida Hmida, Sana Ben Hamida, Amel Borgi, Marta Rukoz

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 126 (2018) 302–311

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Knowledge Based and Intelligent Information and Engineering Systems, KES2018, 3-5 September 2018, Belgrade, Serbia

Scale Genetic Programming for large Data Sets: Case of Higgs Bosons Classification

Hmida Hmida[a,b], Sana Ben Hamida[b], Amel Borgi[a], Marta Rukoz[b]

[a] *Tunis El Manar University, LIPAH, Tunis, Tunisia*
[b] *Paris Dauphine University, PSL Research University, CNRS, LAMSADE UMR 7243,75016 Paris, France*

Abstract

Extract knowledge and significant information from very large data sets is a main topic in Data Science, bringing the interest of researchers in machine learning field. Several machine learning techniques have proven effective to deal with massive data like Deep Neuronal Networks. Evolutionary algorithms are considered not well suitable for such problems because of their relatively high computational cost. This work is an attempt to prove that, with some extensions, evolutionary algorithms could be an interesting solution to learn from very large data sets. We propose the use of the Cartesian Genetic Programming (CGP) as meta-heuristic approach to learn from the Higgs big data set. CGP is extended with an active sampling technique in order to help the algorithm to deal with the mass of the provided data. The proposed method is able to take up the challenge of dealing with the complete benchmark data set of 11 million events and produces satisfactory preliminary results.

Keywords: Cartesian Genetic Programming, Active Sampling, Higgs Bosons Classification, large dataset, Machine Learning ;

1. Introduction

The scientific and technical challenges that have arisen with the beginning of the era of the Big Data are identified and recognized by the scientific communities. These challenges include learning from very large data sets. In fact, learning from very large datasets may limit the applicability of most of the usual techniques. Currently, there is a great quantity of very large public datasets; for example, the Machine Learning Repository contains more than 400 datasets classified by kind of task, attribute type, data type, etc[1]. In 2014, a small group of ATLAS physicists and data scientists have organized a machine learning challenge allowing to analyze the Higgs dataset, a recent high energy

* Corresponding author. Tel.: +0-000-000-0000 ; fax: +0-000-000-0000.
E-mail address: sana.mrabet@dauphine.fr

[1] http://archive.ics.uci.edu/ml/datasets

10.1016/j.procs.2018.07.264

experimental physic database[1] . The main tools of high energy experimental physicists are modern accelerators to collide protons and/or antiprotons to create exotic particles that occur only at extremely high energy densities. Here, the problem is that particle accelerators create a huge amount of data, however the vast majority of particle collisions do not produce exotic particles. For example, though the Large Hadron Collider produces approximately 100 billion collisions per hour, approximately 300 of these collisions result in a Higgs boson, on average [6]. So, a good data analysis depends on distinguishing collisions which produce particles of interest (signal) from those producing other particles (background). A description of the Higgs dataset is given in section 2. Several machine learning techniques were proposed such as Deep Neural Networks [1] and decision trees [9]. Gbor Melis, the winner of the Higgs Boson challenge, states that ”his original plan for this contest was to use an evolutionary algorithm, but he withdrew because evolutionary algorithms do not scale to larger problem sizes”. Nevertheless, Genetic Programming (GP) evolutionary techniques have shown to be a very promising solution to deal with the large data classification scenario. However, this algorithm suffers from an increased computational cost induced mainly by the evaluation step.

There have been some research efforts to improve the performance of GP when applied to very large data sets. A promising algorithmic solution is to reduce the learning set size by adding an active data sampling technique[15]. The specific interest of the active data sampling lies in reducing the original training dataset size by substituting it with a representative subset much smaller.

This paper intends to show that GP is a good solution for such kind of data. We propose an active sampling method to learn from the Higgs dataset using a Cartesian Genetic Programming (CGP). In addition, the dataset used for the challenge is a sub-sample of the original database. The challenge training set contains only 250K instances. In this work, we propose to use a CGP to learn from the enlarged big dataset proposed by Baldi [6] containing 11 million events. To deal with the high size of the training data, CGP is extended with an active sampling technique.

After giving further details about the Higgs dataset and some related works, we present in section 3 a short review of the active sampling techniques proposed for evolutionary methods. The proposed evolutionary machine learning technique based on an active sampling is presented in details in section 4. Sections 5 and 6 summarize the experimental setting and give some preliminary results. The last section concludes the present work and presents some further works to improve our algorithm.

2. Higgs dataset and related works

A Higgs or Z boson is a heavy state of matter resulting from a small fraction of the proton collisions at the Large Hadron Collider[7]. This heavy state quickly decays into more stable particles, so the intermediate states of matter are not observable by the detectors surrounding the point of collision. Thus, it is difficult to distinguish between two different processes with the same set of final stable particles without studying the intermediate states. An other approach is to examine closely the momentum and direction of the final state particles. Highly faithful collisions are then simulated by the ATLAS Simulator [10] using sophisticated Monte Carlo programs to reproduce the essential measurements provided by the detectors. The relevant variables are known as low-level variables. Otherwise, in order to better discriminate between Higgs-boson productions and Z-boson productions, a set of high-level variables is constructed using non-linear combinations of the low-level variables. The resulting dataset contains 80 million collision events, characterized by 28 real-valued features, 21 low-level variables describing the 3D momenta and energies of the collision products, and 7 high-level variables.

ATLAS experiment at CERN publicly released a portion of the simulated data used by physicists to optimize the analysis of the Higgs Bosons by machine learning techniques. A subset of this data was presented as a challenge in 2014 [1] in the kaggle platform[2]. The immediate goal of the challenge was to explore the potential of advanced classification methods to improve the statistical significance of the experiment. The supervised learning task is to distinguish between two types of physical processes: one in which a Higgs Boson decays into $\tau^+\tau$ leptons and a background process that produces a similar measurement distribution [6].

[2] https://www.kaggle.com/

From the machine learning point of view, the problem can be formally cast into a binary classification problem. The task is to classify events as a signal (event of interest) or a background (event produced by already known processes).

For the challenge, Kaggle provided a training set of 250K events and a test set of 550K events (a detailed description is given in [2]). Most of the advanced machines learning techniques were tried for this challenge. For example most top ranking solutions used ensembles of decision trees, and particularly the XGBoost software [4]. The winning solution of Gbor Melis[20] uses a deep learning method (an ensemble of 70 3-layers Neural Networks fully interconnected, with 600 hidden units per layer).

Just after the challenge, Baldi et al. [6] published for benchmarking machine-learning classification algorithms a big data set of simulated Higgs Bosons that contains 11 million simulated collision events and the 28 features presented above [3]. The instances in the positive class correspond to *signal events*, otherwise, they correspond to *background event*. Some recent works tried to learn a portion of this new benchmark dataset. For example, Alves [5] used an ensemble learning with various machine learning algorithms to learn form just 20% of the 11 million events. The work of Baldi [6] using the Deep Neural Network (DNN) on the same proportion of data have been used as reference to judge the performance of his method and concluded that the proposed algorithm performed only about 10% worse than the complex DNNs.

In this work, we propose to handle the complete dataset with 11 million instances, which is a big challenge when using evolutionary algorithms. The following table summarizes the main characteristics of the dataset:

Table 1: Higgs dataset composition

Total of events	11 millions
Number of Attributes	28 real-valued (21 low-level variables and 7 high-level variables)
Percentage of signals	53%
Training set size	10.5 millions events
Test set size	500K events

3. Active Sampling

Active learning is a subfield of machine learning where the learner (algorithm) could choose the data from which it learns. This hypothesis was developed essentially in the statistics literature. Active sampling could be considered as the main approach for active learning.

There have been a lot of algorithms and applications for active sampling in machine learning over the years [24]. Some of these techniques are specific to evolutionary algorithms and could be classified into two categories: one level sampling techniques and multi-level sampling techniques or hierarchical sampling. We present below the main methods in each category.

3.1. One level sampling methods

3.1.1. Random Sampling

The simplest method to choose fitness cases to build the training sample is random. The selection of fitness cases is based on a uniform probability among the training subset. This stochastic selection helps to reduce any bias within the full dataset on evolution. Random Subset Selection (RSS) is the first implementation given by Gathercole et al. [13]. In RSS, at each generation g, the probability of selecting any case i is equal to $P_i(g)$ such that :

$$\forall i : 1 \leq i \leq T, \quad P_i(g) = \frac{S}{T}. \tag{1}$$

[3] See UCI Machine Learning Repository at http://archive.ics.uci.edu/ml/datasets/HIGGS

where T is the size of the full dataset and S is the target subset size (for all the following equations). The sampled subset has a fluctuating size around S. A second variant of random sampling called *Fixed Random Selection* [27] uses a fixed number of cases selected at every generation. *The Stochastic Sampling* [22] is a third variant of random sampling using the same probabilistic selection to construct subsets, but for each individual per generation. The single parameter of this technique is the subset size and it is set like an additional GP parameters.

3.1.2. Weighted Sampling

Weighted Sampling techniques have two objectives:
(i) focus the algorithm abilities on difficult cases, i.e. fitness cases frequently unsolved by the best solutions,
(ii) check fitness cases that have not been looked at for several generations.
They are highly inspired by boosting techniques in the machine learning field, originally used to improve accuracy of weak learners.
The first algorithm in this category is *Dynamic Subset Selection* (DSS) [13, 14, 12]. This algorithm is intended to preserve training set consistency while alleviating its size by keeping only the difficult cases with ones not selected for several generations. To each dataset record is assigned a difficulty degree $D_i(g)$ and an age $A_i(g)$ starting with 0 at first generation and updated at every generation. The difficulty is incremented for each misclassification and reset to 0 if the fitness case is solved. The age is equal to the number of generations since last selection, so it is incremented when the fitness case has not been selected and reset to 0 otherwise.
The resulting weight W of the i^{th} fitness case is calculated as follows:

$$\forall i : 1 \leq i \leq T, \quad W_i(g) = D_i(g)^d + A_i(g)^a. \tag{2}$$

where d is the difficulty exponent and a is the age exponent. The selection probability of a record i is biased by its weight W_j such that:

$$\forall i : 1 \leq i \leq T, \quad P_i(g) = \frac{P_i(g) * S}{\sum_{j=1}^{T} W_j(g)}. \tag{3}$$

DSS needs three parameters to be tuned: difficulty exponent, age exponent and target size.

3.1.3. Incremental Data Selection: (IDS)

The Incremental Data Selection, called by the authors Incremental Data Inheritance [27], associates a training subset to each individual. During evolution, the individual subsets are recombined and enlarged by small numbers of fitness cases taken from the original data set. This procedure should ensure diversity of the training data. In the final generation, all individuals are evaluated according to the entire dataset.

3.1.4. Topology Based Subset Selection (TBS)

The Topology Based Subset Selection technique extends the DSS method to take into account the topology of the fitness space, creating relationships between fitness cases [18]. The relation between two fitness cases is strengthened if a single individual of the population is able to solve both fitness cases. Then, cases having a tight relationship with respect to a threshold cannot be selected together in the same subset assuming that they have an equivalent difficulty for the population.

3.1.5. Balanced sampling

Balanced sampling[17] aims to improve classifier accuracy by correcting the original dataset imbalance within majority and minority class instances. It has some methods based on the minority class size and thus reduce the number of instances like the methods studied in this paper. Several approaches are proposed, we summarize hereafter three sampling techniques used with GP. First, Static Balanced Sampling that selects cases with uniform probability from each class without replacement until obtaining a balanced subset of the desired size. Then, Basic Under-sampling (resp. Basic Over-sampling) selects all minority (resp. majority) class instances and then an equal number from the majority (resp. majority) class randomly.

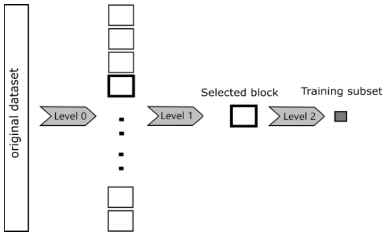

Fig. 1: Two level Hierarchical sampling

3.2. Multi-level sampling or Hierarchical Sampling

Hierarchical sampling combines several sampling algorithms applied at different levels. Its objective is to deal with large data sets that do not fit in the memory, and simultaneously provide the opportunity to find solutions with a greater generalization ability than those given by the one-level sampling techniques. The data subset selections at each level are independent.

3.2.1. RSS-DSS and DSS-DSS

Curry et al. conceived an extension to the DSS algorithm into a 3 levels hierarchy[23]. At level 0, the data set is first partitioned into blocks that are sufficiently small to reside within RAM alone. Then, at level 1, one block is chosen from this partition based on RSS or DSS. Finally, at level 2, the selected block is considered as the full dataset on which DSS was applied for several generations. Depending on the level 1 algorithm, two approaches are possible: *RSS-DSS* hierarchy or *DSS-DSS* hierarchy.

3.2.2. Balanced Block-DSS

Curry et al. proposed an extension of their method RSS-DSS/DSS-DSS and they introduce the *Balanced Block DSS* (BB-DSS) [11]. BB-DSS differs from the precedent methods on the level 0, where database partition is altered in order to obtain balanced blocks at level 1. To generate a balanced block that reflect the original data set classes' distribution, the method uses a fixed ratio for each class.

3.2.3. RSS-TBS

Based on the same idea, Hmida et al. proposed two new variants of hierarchical sampling : the RSS-TBS and the BUSS-RSS-TBS [15]. The RSS-TBS uses the Topology Based Subset Selection at level 2 instead of RSS or DSS. The second variant BUSS-RSS-TBS extends the first variant with a Basic Under-sampling at the level 0 block creation. BUSS favors the minority class by calculating the block size according to their cardinalities. For majority class, an equal number of instances are selected randomly.

4. Extending CGP with Active Sampling

As an evolutionary machine learning technique, we choose the Cartesian Genetic Programming (CGP) that is a graph-based GP. CGP is extended with a an active data sampling technique from one-level and two-level sampling categories. Details are given in the following two subsections.

4.1. Cartesian Genetic Programming

Cartesian Genetic Programming [21] is an extension of the standard GP where genotypes represent graph-like programs. The genotype is a list of integers that represents the program primitives and how they are connected together. Each integer encodes some information such as: from where a node gets its data, what operations the node performs on the data and where is the output data required by the user.

CGP shows several advantages over other GP approaches. Unlike trees, there are more than one path between any pair of nodes. This enables the reuse of intermediate results. A genotype can also have multiple outputs which make CGP able to solve many types of problems. Otherwise, CGP has the great advantage of counteracting the bloating effect (genotype growth), frequent phenomena with other GP representations. Note that bloating is undesirable, especially with very large datasets. CGP is easy to implement, and it is highly competitive compared to other GP methods.

The main operator implemented for CGP is the point mutation that chooses randomly an allele g_m and changes it according to its nature. If g_m corresponds to a function label, then it is replaced by a random element from the non-terminal set (Table 2). Otherwise, a random value is chosen from the output of any previous node in the genotype or from the terminal set. CGP uses also an arithmetic crossover for real-valued genomes.

[26].

4.2. The data sampling approach

To compare the efficiency of the data sampling approaches when applied to Higgs dataset, one technique is selected from each category: the RSS from the one-level sampling category and the RSS-DSS from the two-level sampling category. The RSS technique is implemented as described in section 3.1.1 where a random training subset is selected at each generation. The RSS-DSS sampling process is applied through three levels. First, the entire training data set is partitioned into equal blocks that are small enough to fit within RAM (level 0). The number of cases per class is proportional to the initial training set class distribution. Blocks are then saved into hard disk. In the next step, blocks are chosen randomly with uniform probability using RSS technique. This forms level 1. At level 2, DSS algorithm is used to sub-sample the selected block biased by exemplar difficulty and age as described for the DSS method (section 3.1.2). The general algorithm is shown below.

Algorithm 1: CGP + RSS-DSS active sampling

```
Parameters:
$\mathcal{S}$: learning data set
$T_{l1}$: for level 1 maximum iterations
$T_{l2}$: for level 2 maximum iterations

Divide $\mathcal{S}$ into blocks ;                                   // level 0
repeat
    Conduct Block Selection using RSS method ;                    // level 1
    reset age and difficulty vectors ($A(g)$ and $D(g)$)
    repeat
        Conduct Subset Selection using DSS ;                      // level 2
        Evaluate individuals against block subset
        Update fitness case ages $A_i(g)$
        Update fitness case difficulty $D_i(g)$
        Evaluate best of generation individual against full block
        Update best of run according to full block performance
        Apply genetic operators
    until level 2 iterations = $T_{l2}$;
until  level 1 iterations = $T_{l1}$;
```

In addition to the known CGP parameters, two new parameters are used by the algorithm: T_{l1} and T_{l2}. T_{l1} and T_{l2} design respectively the number of CGP iterations for the first-level sampling (RSS) and for the second-level sampling (DSS).Thus, after each T_{l1} iterations, the level-one training data set is replaced with an other set with the RSS method. This set is used to generate the training dataset each T_{l2} iterations with the DSS approach.

4.3. Performance Metrics

By the end of each run, the best individual based on the fitness function is evaluated on the test dataset. Results are recorded in a confusion matrix from which accuracy, True Positive Rate (TPR) and False Positive Rate (FPR) are calculated. The objective function used with CGP is the classification accuracy according to the following formula.

$$Accuracy = \frac{True\ Positives + True\ Negatives}{Total\ patterns}. \tag{4}$$

Otherwise, to evaluate performance of the predictive models, Kaggle uses an evaluation metric known as Approximate Median Significance (AMS). This metric uses FPR value (corresponding to the rate of events classified as *signals*) and TPR value (corresponding to the rate of events classified as *background*), and aims to heavily penalize false positive cases. A detailed description of AMS is available in [3]. several formula have been proposed to compute AMS value. A simplified definition is given in [2] as follows:

$$AMS = \frac{TPR}{\sqrt{FPR + (FPR \times \epsilon)^2}} \text{ where } \epsilon < 0.1 \tag{5}$$

In addition to these metrics, we calculate the Area Under ROC curve (AUC or AUROC) of the best classifier. AUC is often used as a measure of quality of the classification models.

5. Experimental settings

5.1. Framework

Software framework. Among several evolutionary computation frameworks, Sean Luke's ECJ [19] was used in this work to implement and test the extended CGP. ECJ is an active project at George Mason University's Evolutionary Computation Laboratory having frequent releases It's an open source framework written in Java and benefit of many contribution packages as the one used here for implementing Cartesian GP developed by David Oranchak[8]. This framework provides a very flexible API using parameter files well documented in the ECJ owner's manual.

Hardware framework. Experiments are performed on an Intel $i7 - 4810MQ$ ($2.8GHZ$) workstation with $8GB$ RAM running under Windows $8.164 - bit$ Operating System. GP programs are trained to distinguish between signal events and background events in the Higgs database.

5.2. Terminal and function sets

The terminal set includes 28 features of the benchmark Higgs dataset. The function set includes basic arithmetic, comparison and logical operators reaching 17 functions (table 2).

Table 2: Terminal and function sets for GP.

Function (node) set	
Arithmetic operators: Comparison operators: Logic operators: Other:	$+, -, *, \%$ $<, >, <=, >=, =$ AND, OR, NOT, NOR, NAND NEGATE, IF (IF THEN ELSE), IFLEZE(IF <=0 THEN ELSE)
Terminal set	
Higgs Features Random Constants	28 8 in $[-2, 2[$

5.3. CGP and sampling parameter settings

The design of CGP parameters used in this work is summarized in Table 3. The parameter values are tuned thanks to a series of trials.

Table 3: CGP and sampling parameters.

(a) CGP parameters

Parameter	Value
Population size	128
Number of generations	3500
CGP nodes	500
CGP levels-back	3
Inputs/Outputs	36/1
Tournament size	4
Crossover probability	0.9
Mutation probability	0.04

(b) Sampling Parameters

Method	Parameter	Value
One level RSS	Target Size	7000
Two level RSS-DSS	Target Size(level 2 block size)	7000
	Level 0 block size	50000
	RSS iterations	50
	Max DSS iterations	70
	Difficulty/Age exponent	1/3.5
	Difficulty/Age Roulette	70%/30%

Active sampling parameters are exposed in Table 3(b). The one-level sampling method (RSS) has a unique parameter which is the target subset size. It is set to the value of the two-level RSS-DSS target for comparability purposes. RSS-DSS uses a large number of parameters having their values adjusted according to our previous works in [16]. For this study, only three parameters are tuned: target size, RSS iterations and Max DSS iterations.

6. Results and discussion

For the experimental study, nine measures are recorded across the 5 runs performed: the best and the mean values for the accuracy(4), TPR, FPR and AMS metric (5) whereas AUC is recorded only for the best classifier. To compute AMS, ϵ (eq. 5) is set to 0.05. The corresponding values are illustrated in Table 4.

Table 4: CGP results according to the performance metrics.

		Accuracy	TPR	FPR	AMS	AUC
One level Sampling RSS	Best	0,637854	0,6803448	0,3139881	1,214	0,6405
	Average	0,6293352	0,6620886	0,35647769	1,08	
Two level sampling RSS-DSS	Best	0,65038	0,686341	0,39	1,1	0,6482
	Average	0,6459072	0,6426552	0,3504401	1,0855	

Besides the performance metrics, we also recorded the confusion matrix according to the test set. It shows the distribution of fitness cases between the two classes (S: Signal, B: Background) as predicted by the best CGP classifier against their real classes as marked on the test set.

Table 5: Confusion Matrices of the test set.

(a) One level RSS

		Real class	
		S	B
Predicted class	S	157376	73942
	B	107131	161551

(b) Two level RSS-DSS

		Real class	
		S	B
Predicted class	S	181542	91845
	B	82965	143648

The big challenge of this work is to prove that a Genetic Programming algorithm, with some extensions, is able to learn from very large data sets. This goal is well achieved. Extended with an active sampling approach, CGP is able to train its population on a data set having 10.5 million observations in a reasonable time. Indeed, with the configuration described in section 3, a complete run of CGP took around 2.5 hours, which could be considered as an interesting runtime viewing the large scale of the problem and the hardware framework. Otherwise, the spent time for a single CGP run is lower than for other machine learning techniques such as Logistic Regression or Linear SVM (table 6). Moreover, the comparison between the results obtained with the two implemented active sampling techniques shows that the performance of the two approaches are quite similar with a small superiority of the RSS-DSS method (table 4). These results prove the validity of the active learning for Genetic Programming.

The most known results in Higgs Boson detection are those achieved in Kaggle Higgs challenge. Though, comparing our result with this challenge ”leaderboard” is not fair for the following reasons:

- Kaggle’s challenge uses a different dataset having a fewer number of events,
- this dataset have 30 features, 13 of them are derived and selected by ATLAS physicists,
- ranking is based on AMS score that depends on each event weight set by experts for this challenge,
- performance enhancing measures are extensively used by competitors such as feature reduction, ensemble classifiers, bagging, parameter tuning, etc.

Note that the winning method is an ensemble of 70 neural networks whose predicted probabilities were combined by simple arithmetic averaging [20]. The neural networks only differed in their initialization and training sets. Other published works are based on the same data base of the present work, but they used a selected subset with a reduced size for the experimental study.

The first machine learning technique applied on the Higgs data set is a Deep Neural Network (DNN) proposed by Baldi et al.[7]. They compared this technique with the boosted decision tree and the Shallow Neural Network. They used a subset of Higgs data base of 2.6 million examples and 100K validation examples. They demonstrated how DNN can be trained on such data set with a high degree of accuracy.

In [25], authors evaluate machine learning frameworks on Higgs dataset in very close configuration to our work. They run Logistic Regression and Linear SVM using Wekam, Scikit-Learn and Spark. Only Spark managed to run algorithms on the data due to its big size. Our method outperforms their results in a shorter time as reported in Table 6.

Table 6: Results of Shashidhara and al.[25].

Classifier	Training time	Accuracy
Logistic Regression	4 hours	0.6076
Linear SVM	5 hours	0.5290

The present work is the first GP based attempt to deal with Higgs dataset and is among few works that trains on the complete dataset. The extended CGP takes up the challenge to deal with the complete benchmark data set of 11 million events thanks to the active sampling. Further extensions and improvements could be conducted especially on the hierarchical sampling in many ways. For example, data preprocessing and feature engineering are a promising directions to improve classifier quality. Tuning CGP intrinsic parameters, like fitness function, crossover and mutation probabilities and much more settings ..., is a hard problem but remains another path to investigate.

7. Conclusion

We proposed in the present work the use of the Cartesian Genetic Programming to classify Bosons in the Higgs benchmark dataset. To deal with the high size of the training data, CGP is extended with two active sampling approaches: one-level sampling technique where events in the generated subset are selected randomly, and hierarchical sampling with three level data selection techniques. The second and the third levels use respectively a random and a weighted data sampling.

CGP was able to yield satisfactory results. However, some improvements are needed to both raise the quality of the performance metrics and reduce the computing time.

First, other configurations of the proposed method will be tried. The objective is to better tune CGP and RSS-DSS parameters (such as T_{l1} and T_{l2}) in order to improve accuracy. Otherwise, CGP will be tested with AMS as fitness function rather than the accuracy metric. Finally, a second extension will be added to CGP to parallelize the CGP evaluation step.

References

[1] Adam-Bourdarios, C., Cowan, G., Germain, C., Guyon, I., Kegl, B., Rousseau, D., 2014a. Learning to discover: the higgs boson machine learning challenge URL: http://higgsml.lal.in2p3.fr/documentation.

[2] Adam-Bourdarios, C., Cowan, G., Germain, C., Guyon, I., Kegl, B., Rousseau, D., 2014b. The ATLAS Higgs Boson Machine Learning Challenge, in: International Conference on High Energy Physics(ICHEP) Conference, Valencia, Spain.

[3] Adam-Bourdarios, C., Cowan, G., Germain, C., Guyon, I., Kégl, B., Rousseau, D., 2016. How machine learning won the higgs boson challenge, in: European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning.

[4] Adam-Bourdarios, C., Cowan, G., Germain-Renaud, C., Guyon, I., Kgl, B., Rousseau, D., 2015. The higgs machine learning challenge. Journal of Physics: Conference Series 664, 072015. URL: http://stacks.iop.org/1742-6596/664/i=7/a=072015.

[5] Alves, A., 2017. Stacking machine learning classifiers to identify higgs bosons at the lhc. Journal of Instrumentation 12, T05005. URL: http://stacks.iop.org/1748-0221/12/i=05/a=T05005.

[6] Baldi, P., Sadowski, P., Whiteson, D., 2014. Searching for exotic particles in high-energy physics with deep learning. Nature communications 5.

[7] Baldi, P., Sadowski, P., Whiteson, D., 2015. Enhanced higgs boson to τ+ τ- search with deep learning. Physical review letters 114, 111801.

[8] CGP, . Cartesian gp website. URL: http://www.cartesiangp.co.uk.

[9] Chen, T., He, T., 2014. Higgs boson discovery with boosted trees., in: HEPML@ NIPS, pp. 69–80.

[10] collaboration, A., 2014. Dataset from the atlas higgs boson machine learning challenge 2014. http://opendata.cern.ch/record/328 doi:10.7483/OPENDATA.ATLAS.ZBP2.M5T8.

[11] Curry, R., Lichodzijewski, P., Heywood, M.I., 2007. Scaling genetic programming to large datasets using hierarchical dynamic subset selection. IEEE Transactions on Systems, Man, and Cybernetics: Part B - Cybernetics 37, 1065–1073. URL: doi:10.1109/TSMCB.2007.896406.

[12] Gathercole, C., 1998. An Investigation of Supervised Learning in Genetic Programming. Thesis. University of Edinburgh.

[13] Gathercole, C., Ross, P., 1994. Dynamic training subset selection for supervised learning in genetic programming, in: Davidor, Y., Schwefel, H.P., Männer, R. (Eds.), Parallel Problem Solving from Nature - PPSN III, Springer. pp. 312–321.

[14] Gathercole, C., Ross, P., 1997. Small populations over many generations can beat large populations over few generations in genetic programming, in: Koza, J.R., Deb, K., Dorigo, M., Fogel, D.B., Garzon, M., Iba, H., Riolo, R.L. (Eds.), Genetic Programming 1997: Proc. of the Second Annual Conf., Morgan Kaufmann, San Francisco, CA. pp. 111–118.

[15] Hmida, H., Hamida, S.B., Borgi, A., Rukoz, M., 2016a. Hierarchical data topology based selection for large scale learning, in: Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), 2016 Intl IEEE Conferences, IEEE. pp. 1221–1226.

[16] Hmida, H., Hamida, S.B., Borgi, A., Rukoz, M., 2016b. Sampling methods in genetic programming learners from large datasets: A comparative study, in: Angelov, P., Manolopoulos, Y., Iliadis, L.S., Roy, A., Vellasco, M.M.B.R. (Eds.), Advances in Big Data - Proceedings of the 2nd INNS Conference on Big Data, October 23-25, 2016, Thessaloniki, Greece, pp. 50–60. doi:10.1007/978-3-319-47898-2_6.

[17] Hunt, R., Johnston, M., Browne, W.N., Zhang, M., 2010. Sampling methods in genetic programming for classification with unbalanced data, in: Li, J. (Ed.), Australasian Conference on Artificial Intelligence, Springer. pp. 273–282.

[18] Lasarczyk, C.W.G., Dittrich, P., Banzhaf, W., 2004. Dynamic subset selection based on a fitness case topology. Evolutionary Computation 12, 223–242. URL: doi:10.1162/106365604773955157.

[19] Luke, S., . Ecj, a java-based evolutionary computation research system. URL: http://cs.gmu.edu/~eclab/projects/ecj/.

[20] Melis, G., 2014. Dissecting the winning solution of the higgsml challenge., in: HEPML@ NIPS, pp. 57–67.

[21] Miller, J.F., Thomson, P., 2000. Cartesian genetic programming, in: Poli, R., Banzhaf, W., Langdon, W.B., Miller, J., Nordin, P., Fogarty, T.C. (Eds.), Proceedings of the Third European Conference on Genetic Programming (EuroGP-2000), Springer Verlag, Edinburgh, Scotland. pp. 121–132.

[22] Nordin, P., Banzhaf, W., 1997. An on-line method to evolve behavior and to control a miniature robot in real time with genetic programming. Adaptive Behaviour 5, 107–140. doi:10.1177/105971239700500201.

[23] Robert Curry, M.H., 2004. Towards efficient training on large datasets for genetic programming. Lecture Notes in Computer Science 866, 161–174.

[24] Settles, B., 2012. Active Learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool.

[25] Shashidhara, B.M., Jain, S., Rao, V.D., Patil, N., Raghavendra, G.S., 2015. Evaluation of machine learning frameworks on bank marketing and higgs datasets, in: 2015 Second International Conference on Advances in Computing and Communication Engineering, pp. 551–555. doi:10.1109/ICACCE.2015.31.

[26] Wilson, G., Banzhaf, W., 2008. A comparison of cartesian genetic programming and linear genetic programming, in: European Conference on Genetic Programming, Springer. pp. 182–193.

[27] Zhang, B.T., Cho, D.Y., 1999. Genetic Programming with Active Data Selection. Springer, Berlin, Heidelberg. volume 1585. chapter Simulated Evolution and Learning. pp. 146–153. doi:10.1007/3-540-48873-1_20.