Extending DEAP with Active Sampling for Evolutionary Supervised Learning

Sana Ben Hamida[1] [a] and Ghita Benjelloun[2]

[1] *Paris Dauphine University, PSL Research University, CNRS, UMR[7243], LAMSADE, 75016 Paris, France*

[2] *Paris Dauphine University, PSL Research University, 75016 Paris, France*

Keywords: Genetic Programming, DEAP, Active Learning, Random Sampling, Weighted Sampling, Occupancy Detection, Pulsar Detection.

Abstract: Complexity, variety and large sizes of data bases make the Knowledge extraction a difficult task for supervised machine learning techniques. It is important to provide these techniques additional tools to improve their efficiency when dealing with such data. A promising strategy is to reduce the size of the training sample seen by the learner and to change it regularly along the learning process. Such strategy known as active learning, is suitable for iterative learning algorithms such as Evolutionary Algorithms. This paper presents some sampling techniques for active learning and how they can be applied in a hierarchical way. Then, it details how these techniques could be implemented into DEAP, a Python framework for Evolutionary Algorithms. A comparative study demonstrates how active learning improve the evolutionary learning on two data bases for detecting pulsars and occupancy in buildings.

1 INTRODUCTION

Learning from complex, large and unbalanced databases is a major issue for most of the current data mining and machine learning algorithms. More powerful knowledge discovery techniques are needed to deal with the related challenges. Evolutionary Data Mining tools, such as Genetic Programming (GP), are powerful meta-heuristics with an empirically proven efficiency on complex machine learning problems. Thus, the use of evolutionary data mining procedures becomes a hot trend. Thanks to their global search in the solution space, evolutionary computation techniques help in the information retrieval from a voluminous pool of data in a better way compared to traditional retrieval techniques. Thus, they are part of the promising tools to discover insights within the growing complexity and volume of available data. However, as all machine learning techniques, GP needs some extensions to improve its efficiency when learning from large, complex or unbalanced datasets. This paper investigates the use of active learning paradigm to scale evolutionary data mining techniques to complex data sets.

Active learning (Atlas et al., 1990; Cohn et al., 1994) is a learning paradigm based on active sampling techniques. The goal of any sampling approach is to extract subsets from the original training set in order to reduce the size or to handle some anomalies such unbalanced classes . With active learning, subset sampling is applied regularly along the training process in order to change frequently the examples seen by the algorithm. Subset selection could be done randomly or according to a given strategy such data difficulty based strategy (Gathercole and Ross, 1997) or data topology based strategy (Lasarczyk et al., 2004; Hmida et al., 2016b). Active learning is successfully implemented with iterative learning algorithm such as evolutionary data mining methods.

This work outlines how we extended an existing GP implementation with active sampling techniques using three strategies: random selection, balanced selection and weighted selection based on age and difficulty of exemplars. Random and weighted sampling methods are applied on a balanced set extracted from the training data base in order to handle the classes unbalance problem. We also investigate the application of multi-level sampling using different sampling techniques on a hierarchical way. These strategies are implemented on a Python framework for evolutionary algorithms, DEAP. They are applied to solve two classification problems: pulsar classification and occupancy detection in buildings.

[a] https://orcid.org/0000-0003-4202-613X

Ben Hamida, S. and Benjelloun, G.
Extending DEAP with Active Sampling for Evolutionary Supervised Learning.
DOI: 10.5220/0010604605740582
In *Proceedings of the 16th International Conference on Software Technologies (ICSOFT 2021)*, pages 574-582
ISBN: 978-989-758-523-4

This paper is organized as follows. First we remind briefly the evolutionary loop, the GP engine and the their implementation over the Python framework DEAP. Section 3 explains the active learning paradigm and details the sampling techniques implemented for this work: Random Sampling (RSS), Balanced sampling (SBS) and Weighted Sampling (DSS). Details about the implementation of these techniques over DEAP are given in section 4. Section 5 summarizes the classification problems for the experimental study. The main results are summarized and discussed in section 5.2 before the conclusion.

2 BACKGROUND

2.1 EAs: A Brief Overview

Evolutionary algorithms (EAs) are zeroth order optimization methods based on a random search of the parameter space by a population of optimizers undergoing "evolutionary pressure" based on an analogy with Darwinian selection of species EAs are based on the evolution of a population of solutions X_i (*individuals*), where X_i a candidate solution to the problem. Each solution X_i is evaluated to give some measure of its performance (*fitness*). The minimisation of this measure during the process drive the evolution of the population. At each generation, a set of new individuals (*offspring*) are generated by selecting the more fit individuals. These offspring undergo transformations (alter step) by means of "genetic" operators : mutation (perturbation of a solution) and crossover (combination of solutions) (Petrowski and Ben-Hamida, 2017). A new population is then formed by a stochastic selection among offspring proportionally to their fitness.

2.2 Genetic Programming

Genetic Programming (GP) is a dynamic tree based representation for EAs. It is applied to evolve complex objects such as sequential sub-programs or linear functions. GP is considered as the evolutionary technique having the widest range of domain application. Thanks to its great flexibility, GP is able to solve different classes of machine learning problems such as data classification and symbolic regression. In this paper, GP is used to solve two classification tasks described in section 5.

GP respects the same EAs main steps as EA as described above. The GP population is an ensemble of dynamic syntax trees composed of a set of leaves, called terminals and a set of nodes, called non-terminals. The non-terminal set may include arithmetic operations, mathematical functions, boolean operations, conditional operators (such as If-Then-Else), functions causing iteration (such as Do-Until),etc. The terminals are, typically, variable atoms (i.e, representing the inputs) or constant atoms or functions with no explicit arguments. To evolve the GP population, specific initialization and variation operators are needed. DEAP framework, described below, proposes a great variety of mutation and crossover operators ready to use.

GP has shown its great potential when applied to supervised learning, especially classification problems. GP evolves a population of classifiers (genetic programs) by means of genetic operators through generations. The general GP loop for supervised learning is illustrated in Fig 1. Two data sets (at least) are needed: the training set and the test set. The training set is used by the algorithm along the learning process to evaluate the evolved models (steps 1 and 4 in Fig 1). The test set is used to assess the accuracy of the final models (step 6 in Fig 1). The two sets must be independent such as all test instances are unknown from the learner. Each pattern is evaluated according to its corresponding error across the test set. For a classification problem, the error might be estimated according to the number of misclassified cases.

2.3 DEAP

DEAP (or Distributed Evolutionary Algorithm in Python) (Fortin et al., 2012) is a Python framework released in 2012 at the Vision and Numerical Systems Laboratory (LVSN) at the Laval university of Canada. It is freely available and is presented as a rapid prototyping and testing framework. It implements several EA: Genetic Algorithms, Evolution Strategies (ES) and Genetic Programming.

The core architecture of DEAP is built around different components that define the specific parts of what is an evolutionary algorithm and what is an evolutionary learning loop (Fig 1). DEAP's core is composed of three modules: *base, creator*, and *tools*. The base module contains objects and data structures frequently used in Evolutionary Computation that are not already implemented in the Python standard libraries.

To implement functional programming paradigm used by GP, DEAP propose different data structure such as lists and trees. For each data structure, a set of crossover and mutation operator is available. For this work, we choose the basic GP trees. The non terminal set, DEAP propose a collection of unary and binary arithmetic operators, a collection of mathemat-

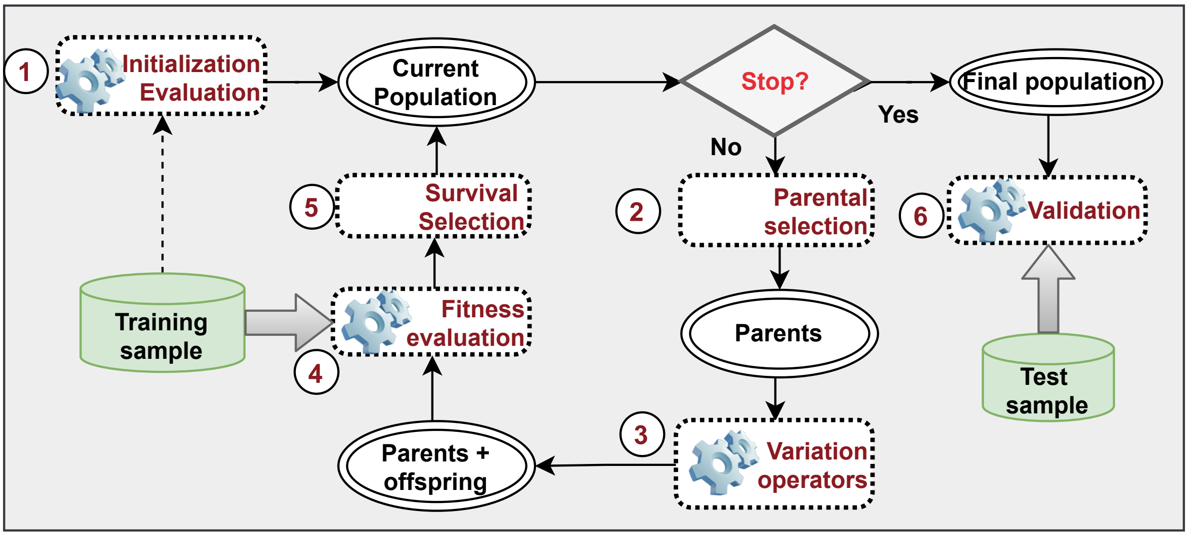

Figure 1: Genetic Learning Evolutionary loop. Steps 2,3 and 5 concern the traditional EA loop. Steps 1 and 4 need training data sample. Model validation in step 6 needs test data sample.

ical functions such as logarithmic functions and a set of logic operators.

3 ACTIVE LEARNING

With Evolutionary data mining techniques, it is possible to train all models on a single subset $\mathcal{S}$. The subset $\mathcal{S}$ is then used to evaluate all individuals (models) throughout an evolutionary run. This sampling approach known as the *Static Sampling*. If any sampling technique is called by the learner to change the training set along the evolution, then it is the *Active Learning* paradigm that is applied.

Active Learning (Atlas et al., 1990; Cohn et al., 1994) could be defined as:

> 'any form of learning in which the learning program has some control over the inputs on which it trains.'

Active learning is implemented essentially with active sampling techniques. The goal of any sampling approach is to reduce the original size of the training set, and thus the computational cost, and enhance the learner performance. Sampling training data set has been first used to boost the learning process and to avoid over-fitting (Iba, 1999). Later, it was introduced for Genetic learners as a strategy for handling large input databases. With the increasing size of available training datasets, this practice is widely used.

With active sampling, the training subset is changed periodically across the learning process. We distinguish one-level sampling methods using a single selection strategy and multi-level sampling (hierarchical sampling) methods using multiple selection strategies associated in a hierarchical way.

3.1 One-level Active Sampling

One-level sampling methods use a single selection strategy based on dynamic criteria, such random selection, weighted selection, incremental selection, etc. Records in the training subset S are selected before the application of the genetic operators each η ($\eta \geq 1$) generations. When $\eta = 1$, the population is evaluated on a different data subset each generation and the sampling approach is called *generation-wise*.

If the main goal of the sampling technique is the reduction of the training fitness cases for the evaluation step, the subset selection can be performed randomly, as described in the following (subsection 3.1.2). However, if the sampling method has an additional purpose (avoiding over-fitting, improve learning quality, handling imbalanced data, etc), it needs a specific record selection strategy. It is the case of several sampling approaches in the literature. For example, the Topology Based Sampling method (Lasarczyk et al., 2004; Hmida et al., 2016a) studies relationship between fitness cases in the data base in order to avoid to select similar fitness cases in the same training subset. Its objective is to maximize the generalization ability of the applied machine learning technique. To overcome imbalance in the original data base, a series of balanced sampling techniques are proposed in (Hunt et al., 2010). For a classification problem, these techniques aim to improve the classifiers accuracy by correcting the data imbalance within majority and minority class instances in the selected subset. For this work, we are interesting on the balanced, random and weighted sampling.

3.1.1 Balanced Sampling (SBS)

The main purpose of balanced sampling is to overcome imbalance in the original data sets. The well known techniques in this category are those proposed by Hun et al. (Hunt et al., 2010) aiming to improve classifiers accuracy by correcting the original dataset imbalance within majority and minority class instances. Some of these methods are based on the minority class size and thus reduce the number of instances. In this paper, we focus on the *Static Balanced Sampling (SBS)*. SBS is an active sampling method that selects cases with uniform probability from each class without replacement until obtaining a balanced subset. This subset contains an equal number of majority and minority class instances of the desired size.

3.1.2 Random Sampling (RSS)

The simplest method to choose fitness cases (data record) to build the training sample S is random. 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. (Gathercole and Ross, 1994). In RSS, at each generation g, the probability of selecting any case i is equal to $P_i(g)$ such that :

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

where T_B is the size of the full dataset B and T_S is the target subset size. The sampled subset has a fluctuating size around T_S.

3.1.3 Weighted Sampling (DSS)

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.
The first algorithm in this category is *Dynamic Subset Selection* (DSS) (Gathercole and Ross, 1997; Gathercole, 1998). 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:

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

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.

3.2 Multi-level Active Sampling

Multi-level sampling or hierarchical sampling combines several sampling algorithms applied in 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. The data subset selections at each level are independent (Fig 2).

The hierarchical sampling was first experimented for an intrusion detection problem in (Robert Curry, 2004). RSS and DSS was combined to implement two level sampling methods: RSS-RSS and RSS-DSS. Authors proposed also in (Curry et al., 2007) the BB-DSS techniques where a balanced sampling is applied in the first level. The aim is to handle classes' unbalance by generating balanced blocks from the starting data base. These blocks are then used as input for the second level sampling. The same approaches are also experimented in (Hmida et al., 2018) to solve the Higgs Boson classifications problem.

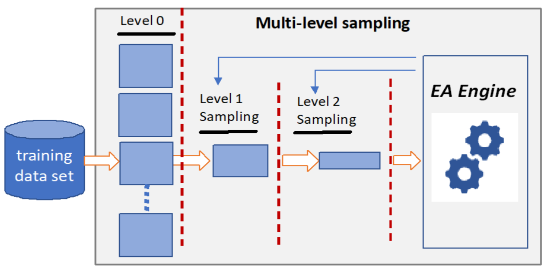

Figure 2: Multi-level Sampling for Evolutionary Machine Learning. Level 1 and 2 sampling are iterated within the Evolutionary loop.

3.3 Implemented Schema

In this work, four sampling schema are implemented that are variants from the one-level and multi-level sampling methods described above:

- **2L-RSS:** two level random sampling. At level 0, balanced bloks are created with a same ratio for each class using the SBS technique. At level 1, the random sampling is applied on the current block.
- **2L-DSS:** as for 2L-RSS, the weighted sampling at level 1 is applied on the balanced blocks created at level 0.
- **3L-RSS-DSS:** It is a 3 level sampling technique based on the 2L-RSS and extended with a third level where a weighted sampling is applied on the subset given by RSS.
- **3L-RSS-RSS:** It respects the same steps as 3L-RSS-DSS. However, the random selection is applied twice, at level 1 and 2.

To implement these techniques, additional parameters are needed to define the block size, the training set size and the intermediate training set size for the three level sampling. Otherwise, the sampling frequency for each level is also an input parameter to be defined.

4 ENABLING ACTIVE SAMPLING OVER DEAP

This section details how DEAP is extended with the different sampling techniques proposed in this work. First, the whole train set is loaded in a pandas data frame that is used as an input argument for SBS to create Blocks (Fig 2 - Level 0).

The code of the SBS and RSS sampling methods used in this work is given in Fig 3. For the DSS algorithm, the training data is extended with 3 additional columns : Age (A), Difficulty (D) and Weight (W) that are updated each generation according to the equation 2. The parameters *d* and *a* are set to 1 and 3.5 as in (Hmida et al., 2016b). However, the selection step is slightly modified. Before selecting a new train set from the input sample, the data frame is divided into two blocks (one for each class) in order to use both balance and weight criteria for the sampling method.

4.1 Extending DEAP with 2 Level Active Sampling

As introduced in section 3, all active sampling techniques are applied on a balanced data (*BBtrain*) having *BBSize* records. It is created at level 0 of each sampling algorithm according to the following procedure. This code is then integrated in the GP loop.

```
#each g_level0 generations
if g(\% g_level0)==0
   BBtrain=SBS([trainSet,BBSize)
```

To implement the level 1 sampling, according to the configuration, one method from RSS, SBS and DSS is applied on the balanced block BBtrain each g_level1 generations.

```
#each g_level1 generations
if g(\% g_level1)==0
    #RSS or SBS or DSS
        finalSet=BBtrain.RSS(setSize)
```

4.2 Extending DEAP with 3 Level Active Sampling

With the three-level sampling, an additional sampling step is performed just after the level 1 such as the final training sample is selected from an intermediate data set (*intermSet*) generated from *BBtrain* randomly or with a balanced sampling.

$$\underset{(BBsize)}{\text{BBtrain}} \longrightarrow \underset{(IntSetSize)}{\text{IntermSet}} \longrightarrow \underset{(setSize)}{\text{finalSet}}$$

The aim of this additional step is to train GP population on a smaller sample in order to further reduce the GP computational cost and the DSS sampling cost when it applied. At each generation, the population of classifiers is evaluated on the *finalSet* sample.

5 APPLICATION

To assess the efficiency of the active sampling for supervised learning over DEAP, we selected two data sets with different characteristics and difficulties.

ODS Data Set for Occupancy Detection in Buildings:

Occupancy detection in an office room uses data from light, temperature, humidity and CO_2 sensors (Candanedo and Feldheim, 2016a). Ground-truth occupancy was obtained from time stamped pictures that were taken every minute. The Occupancy is the response variable indicated with 0 and 1 (occupied/empty). Occupancy detection using the ODS

```
def RSS(InputSet,setSize):
    OutputSet=InputSet.sample(setSize) #select randomly a sample of setsize instances
    return OutputSet
def SBS(InputSet,setSize):
    # Create two random sub-samples from each class
    OutputSet_class0=InputSet[InputSet.iloc[:,0]==0].sample(setSize//2)
    OutputSet_class1=InputSet[InputSet.iloc[:,0]==1].sample(setSize//2)
     # contact the two sub-samples
    OutputSet=pd.concat(OutputSet_class0,OutputSet_class1)
    return OutputSet
```

Figure 3: DEAP implementing RSS and SBS sampling methods.

```
def DSS(InputSet,setSize):
    #update weights of all exemplars
    InputSet.iloc[:,-1] = InputSet.iloc[:,-2]**d + InputSet.iloc[:,-3]**a
    InputSet_class0=InputSet[InputSet.iloc[:,0]==0]
    InputSet_class1=InputSet[InputSet.iloc[:,0]==1]
    IntputSet_class0.sort_values(by = 'W', ascending = False)
    IntputSet_class1.sort_values(by = 'W', ascending = False)
    OutputSet = pd.concat(IntputSet_class0.iloc[:setSize//2],
                IntputSet_class1.iloc[:setsize//2])
    return OutputSet
```

Figure 4: DEAP implementing DSS sampling method.

data set is a medium difficulty classification problem that has been solved with different machine learning techniques. The classification accuracy might reach 99% with some techniques such as SVM (Elkhoukhi et al., 2018). The purpose of using this data in the present work is to study the effect of adding one-level and multi-level sampling techniques to a powerful machine learning technique such that GP to solve a non difficult classification problem.

HTRU Data Set for Pulsar Detection:

Pulsars are a rare type of Neutron star that produce radio emission detectable on Earth. They are of considerable scientific interest as probes of space-time, the inter-stellar medium, and states of matter (Lyon et al., 2016a). As pulsars rotate, their emission beam sweeps across the sky, and when this crosses our line of sight, produces a detectable pattern of broadband radio emission. As pulsars rotate rapidly, this pattern repeats periodically. Thus pulsar search involves looking for periodic radio signals in observational data.

Machine learning classifiers are applied for pulsar detection since 2010. A state of the art is published in (Wang et al., 2019). The first data base made publicly available contains 22 features. Lyon et al. (Lyon et al., 2016b) demonstrated that some features may be suboptimal for machine learning classifier and proposed the HTRU data set with 8 features. It is this second data base studied in the present work.

The HTRU data set contains 16,259 spurious examples caused by RFI/noise (the majority negative class), and 1,639 real pulsar examples . These examples have all been checked by human annotators. Classification machine learning tools are used to automatically label pulsar candidates to facilitate rapid analysis. The ability of machine learning technique applied for pulsar detection is proved in several works (Azhari et al., 2020).

Table 1: Data sets information.

Data Set	Attributes	Set size	Negative examples	Train data
ODS	5 Real Values	8143	6414	6000
HTRU	8 Real Values	17898	16259	14000

5.1 GP Settings and Performance Measures

GP are trained on the both data sets using parameters summarized in Table 2.

For each data set, five series of tests are performed. For each series, a different configuration is

Table 2: GP parameters.

Parameter	Value
Population size	200
Generations number	150
Tournament size	4
Crossover probability	0.5
Mutation probability	0.2
GP Tree depth	3
Fitness	Classification error
Selection operator	Selection by tournament
Crossover operator	Point crossover

applied according to the sampling strategy. For each data set, GP is extended with one of the following configurations:

- No Sampling : GP is run without active learning. The whole data set is seen by each classifier at each generation.
- Two Level Sampling: GP is extended with a two level sampling using SBS a the first level and RSS (2L-RSS) or DSS (2L-DSS) at the second level.
- Three Level Sampling: GP is extended with a three level sampling using the same hierarchy as Two Level Sampling and adding RSS (3L-RSS-RSS) or DSS (3L-RSS-DSS) at the last level.

Ten runs with different seeds are performed for each configuration. By the end of each run, the confusion matrix of best classifier according to the test set is recorded and its accuracy and recall are computed according to the following formula.

$$Accuracy = \frac{True\ Positives + True\ Negatives}{Total\ examples}. \quad (4)$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}. \quad (5)$$

where *True Positives* and *True Negatives* are the numbers of exemplars correctly classified in respectively class 1 and 0, *False Positives* and *False Negatives* are the numbers of exemplars incorrectly classified.

Additional parameters are set for the sampling techniques to define the size of the training set and the sampling frequency at each level. Their values are summarized in Table 3.

Experiments are performed on an Intel i7 (4 Core) workstation with 16GB RAM running under *Windows* 64-bit Operating System.

5.2 Results and Discussion

The main objective of an active sampling technique is to improve the performance of the learning algorithm according to either the quality of the obtained results by maximizing the generalization ability or according to the computational cost by minimizing the run time. Both measures are recorded for each run. Table 4 and 5 summarize the best and average accuracy, the best computing time and the best recall for each configuration.

Three main observations can be deduced from the experimental results. First, introducing a sampling technique to GP helps reducing the training cost up to 10 times, and often with improved performance. Fig 5 shows the relative time saving according to the computing time of the standard GP without sampling. It is near 90% for the ODS data set and 86% for the HTRU data set. The lower computing cost is recorded for the sampling techniques based on the RSS method. DSS algorithm differs from RSS by updating certain sampling parameters (age and difficulty) needed for the exemplar selection, which explains the difference of the computation time.

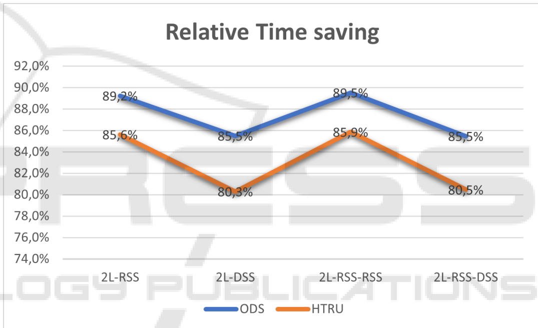

Figure 5: The Relative Time Saving for the four Sampling approaches (2L-RSS, 2L-DSS, 2L-RSS-RSS and 2L-RSS-DSS) according to the best computing time recorded for the Standard GP.

Second, training GP on smaller samples with active learning do not affect the quality of the obtained classifiers. Indeed, the average and best accuracy/recall are either quite close or better with active sampling. Thus, we can state that any active sampling technique is able to both reduce the computational cost and improve the generalization ability of the classifiers compared to the standard GP.

Third, using specific measurements for sample selection such as difficulty and age with DSS are expensive comparing to the random selection with RSS, without bringing a significant improvement in the cases of the ODS and HTRU data sets. On the contrary, for the case of the HTRU data set, which is a more complex base than ODS data set, RSS performs better with 2L and 3L sampling.

Otherwise, it is interesting to compare also the performance of the 2L and 3L sampling strategies.

Table 3: Sampling parameters.

Sampling method	**Parameter**	Level 0	Level 1	Level 2
No Sampling (1L)	size	6000	–	–
2L-Sampling	size	2000	200	–
	frequency (nb generations)	30 (ODS) - 50 (HTRU)	1	–
3L-Sampling	size	2000	1000	200
	frequency(nb generations)	30 (ODS) - 50 (HTRU)	5	2

Table 4: Results on ODS data set.

Test case	Computing time (en s)	Best Accuracy	Avg Accuracy	Best Recall	Avg Recall
No sampling	3209,42	97,8%	97%	100%	99,9%
2L-RSS	345,437	97,9%	97,6%	99,9%	99%
2L-DSS	465,53	**98%**	97,5%	100%	99,5%
3L-RSS-RSS	**336,15**	97,9%	97%	99,9%	97,4%
3L-RSS-DSS	466,54	97,9%	97,1%	100%	97,5%

Table 5: Results on HTRU data set.

Test case	Computing time (en s)	Best Accuracy	Avg Accuracy	Best Recall	Avg Recall
No sampling	5313,81	97,8%	97%	92%	88%
2L-RSS	764,183	**98,9%**	97,3%	89%	84%
2L-DSS	1047,20	98%	96,4%	92%	89%
3L-RSS-RSS	**751,607**	98,7%	97,2%	92%	86%
3L-RSS-DSS	1038,084	98,33%	96,21%	83%	80%

When increasing the number of levels for the hierarchical sampling, it is expected to note a decreasing quality. However, the accuracy is improved in the case of 3L-RSS-RSS for both data sets, and unchanged or improved in the case of 3L-RSS-DSS.

Results in Related Works. Occupancy detection using the same ODS data set has been studied in (Candanedo and Feldheim, 2016b) where different machine learning techniques were applied: Linear Discriminant Analysis, Classification and Regression Trees and Random Forest models. Each method is trained with different selected features. Accuracy obtained on test set with all experimented configurations varies between 95% to 99%. Results given by GP with active sampling are very close (or better in the case of some configurations) than those published in (Candanedo and Feldheim, 2016b).

The first machine learning technique applied to detect pulsars on the Lyon et al. HTRU data set is the Gaussian Hellinger very fast Decision Tree (Lyon et al., 2016b) with a precision equal to 89.9% and a recall around 83%. These results are improved in the study published in (Wang et al., 2019) where the precision and recall reach respectively 95% and 87.3% with the XG-Boost classifier and 96% and 87% with Random Forest classifier. The best accuracy value obtained by the fuzzy-K Nearest Neighbors method published in (Mohamed, 2018) is near 98%, which is the best performance published for this classification problem. Comparing these results with those given by GP in Table 5, it is clear that active learning applied to evolutionary machine learning such as GP is a promising track towards improving pulsar detection models.

6 CONCLUSION

This paper studies the effect of active learning on Genetic Programming as an evolutionary data mining technique. One-level and multi-level sampling strategies using random and weighted sampling are implemented on DEAP, a Python framework for evolutionary algorithms. The algorithm is then applied to classify pulsars and to detect building occupation. The experimental study yields to the following conclusion.

First, any active sampling technique is able to induce better generalizing classifiers compared to the standard GP using the full training set. Second, a simple sampling technique, such as RSS has very low computational cost and it is able to reach higher performance than sampling techniques using additional information such as difficulty and age.

Active learning paradigm based on data sampling is a promising solution that should be considered to scale evolutionary data mining to complex and large data sets. This paradigm could be combined with other paradigms such as Local learning or Ensemble learning for more powerful machine learning techniques.

REFERENCES

Atlas, L. E., Cohn, D., and Ladner, R. (1990). Training connectionist networks with queries and selective sampling. In Touretzky, D., editor, *Advances in Neural Information Processing Systems 2*, pages 566–573. Morgan-Kaufmann.

Azhari, M., Abarda, A., Alaoui, A., Ettaki, B., and Zerouaoui, J. (2020). Detection of pulsar candidates using bagging method. *Procedia Computer Science*, 170:1096–1101.

Candanedo, L. M. and Feldheim, V. (2016a). Accurate occupancy detection of an office room from light, temperature, humidity and co2 measurements using statistical learning models. *Energy and Buildings*, 112:28–39.

Candanedo, L. M. and Feldheim, V. (2016b). Accurate occupancy detection of an office room from light, temperature, humidity and co2 measurements using statistical learning models. *Energy and Buildings*, 112:28–39.

Cohn, D., Atlas, L. E., Ladner, R., and Waibel, A. (1994). Improving generalization with active learning. In *Machine Learning*, pages 201–221.

Curry, R., Lichodzijewski, P., and 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(4):1065–1073.

Elkhoukhi, H., NaitMalek, Y., Berouine, A., Bakhouya, M., Elouadghiri, D., and Essaaidi, M. (2018). Towards a real-time occupancy detection approach for smart buildings. *Procedia computer science*, 134:114–120.

Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., and Gagné, C. (2012). DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175.

Gathercole, C. (1998). *An Investigation of Supervised Learning in Genetic Programming*. Thesis, University of Edinburgh.

Gathercole, C. and Ross, P. (1994). Dynamic training subset selection for supervised learning in genetic programming. In *Parallel Problem Solving from Nature - PPSN III*, volume 866 of *Lecture Notes in Computer Science*, pages 312–321. Springer.

Gathercole, C. and 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., and Riolo, R. L., editors, *Genetic Programming 1997: Proc. of the Second Annual Conf.*, pages 111–118, San Francisco, CA. Morgan Kaufmann.

Hmida, H., Hamida, S. B., Borgi, A., and 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*, pages 1221–1226. IEEE.

Hmida, H., Hamida, S. B., Borgi, A., and 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., and Vellasco, M. M. B. R., editors, *Advances in Big Data - Proceedings of the 2nd INNS Conference on Big Data, October 23-25, 2016, Thessaloniki, Greece*, volume 529 of *Advances in Intelligent Systems and Computing*, pages 50–60.

Hmida, H., Hamida, S. B., Borgi, A., and Rukoz, M. (2018). Scale genetic programming for large data sets: Case of higgs bosons classification. *Procedia Computer Science*, 126:302 – 311. the 22nd International Conference, KES-2018.

Hunt, R., Johnston, M., Browne, W. N., and Zhang, M. (2010). Sampling methods in genetic programming for classification with unbalanced data. In Li, J., editor, *Australasian Conference on Artificial Intelligence*, volume 6464 of *Lecture Notes in Computer Science*, pages 273–282. Springer.

Iba, H. (1999). Bagging, boosting, and bloating in genetic programming. In Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., and Smith, R. E., editors, *Proc. of the Genetic and Evolutionary Computation Conf. GECCO-99*, pages 1053–1060, San Francisco, CA. Morgan Kaufmann.

Lasarczyk, C. W. G., Dittrich, P., and Banzhaf, W. (2004). Dynamic subset selection based on a fitness case topology. *Evolutionary Computation*, 12(2):223–242.

Lyon, R. J., Stappers, B. W., Cooper, S., Brooke, J. M., and Knowles, J. D. (2016a). Fifty years of pulsar candidate selection: from simple filters to a new principled real-time classification approach. *Monthly Notices of the Royal Astronomical Society*, 459(1):1104–1123.

Lyon, R. J., Stappers, B. W., Cooper, S., Brooke, J. M., and Knowles, J. D. (2016b). Fifty years of pulsar candidate selection: from simple filters to a new principled real-time classification approach. *Monthly Notices of the Royal Astronomical Society*, 459(1):1104–1123.

Mohamed, T. M. (2018). Pulsar selection using fuzzy knn classifier. *Future Computing and Informatics Journal*, 3(1):1–6.

Petrowski, A. and Ben-Hamida, S. (2017). *Evolutionary Algorithms*. Computer Engineering: Metaheuristics. Wiley.

Robert Curry, M. H. (2004). Towards efficient training on large datasets for genetic programming. *Lecture Notes in Computer Science*, 866(Advances in Artificial Intelligence):161–174.

Wang, Y., Pan, Z., Zheng, J., Qian, L., and Li, M. (2019). A hybrid ensemble method for pulsar candidate classification. *Astrophysics and Space Science*, 364.