Classifying Streaming Data using Grammar-based Immune Programming

Jaspreet Bassan
Department of Computer Science
Ryerson University
Toronto, Ontario M5B 2K3
Email: jaspreet.bassan@ryerson.ca

Marcus Santos
Department of Computer Science
Ryerson University
Toronto, Ontario M5B 2K3
Email: m3santos@ryerson.ca

Abstract—**This work proposes a technique for classifying unlabelled streaming data using grammar-based immune programming, a hybrid meta-heuristic where the space of grammar generated solutions is searched by an artificial immune system inspired algorithm. Data is labelled using an active learning technique and is buffered until the system trains adequately on the labelled data. The proposed system is tested and evaluated using synthetic and real-world data. The performance of the system is compared with two benchmark problems. The proposed classification system adapted well to the changing nature of streaming data and the active learning technique made the process less computationally expensive by retaining only those instances which favoured the training process.**

I. Introduction

The problem of inducing classifiers in offline environments[1] is well understood and has a long history [1], [2]. This is not the case, however, for streaming environments[2]. The techniques used in inducing classifiers offline are not directly transferrable to streaming environments, for the following reasons. Firstly, streaming data is continuous; thus one can not afford to store all the historical data. Also data in streaming environments may not be representative, which could lead to biased models and therefore to incorrect predictions affecting the system's performance. Streaming data is unlabelled. Hence, it needs to be labelled before it can be used in training, and this might prove expensive. Another major issue of streaming data is "concept drift." Concept drift causes data which is being classified to change in various unpredictable ways. For example: after watching a series of videos on quantum computing on YouTube, the recommender will most likely recommend videos related to this topic. The person may gradually lose interest in quantum computing, thereby causing a *gradual* drift in her choices. Conversely, after watching a video on quantum computing, the person has *suddenly* drifted from this topic to a completely different one.

Machine Learning (ML) has been quite successful in mining information from data. One could employ a neural network [3], support vector machine [4], and evolutionary algorithms (EAs) [5] among others for classifying data. EAs induce a classifier by evaluating the population against all the training instances for a number of iterations. This is a well understood problem in the case of an offline environment where all the training cases are available at once. On the other hand, one cannot employ an iterative approach in a streaming environment as a streaming classification system should output a result as soon as it receives the data. It is argued that EAs perform better in dynamic environments because of the availability of a diverse set of solutions and their ability to adapt which potentially makes it easier to detect change, or to tackle concept drift [6]. In spite of the apparent reasons for using evolutionary inspired ML techniques that are able to develop self adapting solutions to their ever changing environments, little research has been done in EAs in dynamic environments [7], [8], and adequate amount of research of EAs in streaming environments is lacking.

Inspired from the biological immune system, Immune Programming (IP) is a metaheuristic similar to Genetic Programming (GP) in the sense that it also automatically generates solutions but employs a different search engine. IP employs CLONALG[3] [9] in its search space. A Grammar-based IP (GIP) [10] system is an IP system that utilizes a grammar to generate and preserve syntactically correct programs. GIP has been quite successful in regression problems (more successful than Grammatical Evolution [11]), but it has not been applied yet to classification problems. In GIP, hypermutation helps maintain a balance between exploration and exploitation and aids in restoring diversity, and maintaining a diverse set of solutions is critically important in dynamic environments [8]. We conjecture that these built-in evolutionary properties of CLONALG and GIP make them amenable to streaming environments.

In this work, we propose a novel technique for employing GIP for classification problems while labelling uncertain instances in real-time. In this paper, we make use of the IP terminology where a population of candidate solutions for a problem is known as a *repertoire* of *antibodies* (Abs). Training and testing data are known as *Antigens* (Ags). A fitness measure assigned to the Abs is known as *affinity* which

[1]Offline environments are static in nature and all data is available at once.

[2]Streaming environments are dynamic environments, where the data streams in real-time.

[3]Note that, CLONALG falls under Artificial Immune Systems (AIS). EAs and AIS have more similarities than distinctions. So, it is safe to assume that the process undertaken by AIS to solve dynamic problems should be similar to EAs.

is based on the number of correct classifications. Initially, the system is exposed to a small set of labelled Ags. The system then generates a repertoire of Abs using a defined grammar and the repertoire evolves over time by subjecting the Abs to affinity computation, selection, cloning, hypermutation, affinity computation, reselection, and purging. In the first few generations, the system behaves like an offline system as it learns from the labelled Ags. New unlabelled Ags are accepted from the raw streaming data into a buffer at every iteration. They are then labelled by a committee of Abs at fixed intervals. The entire process is repeated as long as the Ags keep streaming. Testing is performed periodically on the Ags available in the buffer.

Two variants of the GIP system are created to be employed in the streaming environment: one that learns from the labelled streaming data and another that learns from the unlabelled streaming data; henceforth called LDS and UDS variants respectively. The variants are compared with each other, and are also compared with the benchmark problems presented by Atwater et al. in his works [12], [13]. Our proposed LDS variant performed better than the Atwater et al.'s work. The labelling procedure was accurate as the unlabelled streaming variant performed similar to its labelled counterpart. Our labelling method is compared with the minimum-variance method presented by Zhu et al. in his work [14]. The UDS variant performed better than the second benchmark problem when in the case of concept drift. The experiments performed indicate both the variants achieved sufficient diversity to maintain satisfactory performance.

This paper is organized in the following manner: Section II presents the materials and methodology used in this work, Section III analyses the results obtained from the different variants of our implementation, and provides a comparison with the benchmark problems, and lastly, Section IV provides a summary of the proposed work and results, and provides the work to be undertaken in the future.

II. Methodology

Any learning algorithm can be understood in terms of three essential components, as seen in the equation, *Learning* = *Representation* + *Evaluation* + *Optimization* [15]. In this section, we present our methodology by describing our design choice for representation, evaluation, and optimization. First, we present in Figure 1 our proposed methodology and briefly explain how it works.

The algorithm depicted in Figure 1 provides a bird's eye view of the UDS variant: Assume L and U are buffers representing sliding windows that follow a FIFO policy where L is the labelled set of instances and U is the unlabelled set of instances. The first step was to load the grammar and initialize all the parameters (such as population size, mutation rate, number of clones, and so on.) Then we loaded some amount of the labelled data into the L set and unlabelled data (which is the streaming data) into U. We then generated a repertoire of Abs (candidate solutions) using the grammar and calculated their affinities. If *time to slide* L is true, then n new instances from the *Lbuffer* are transferred into L by replacing its oldest n instances. If *time to label* is true, we label some instances from U using an Active Learning method. If *time to test* is true, testing is performed. The Abs undergo evolution using the selection and genetic operators, and finally U slides at the end of every iteration. The entire process repeats as long as the instances are available in U.

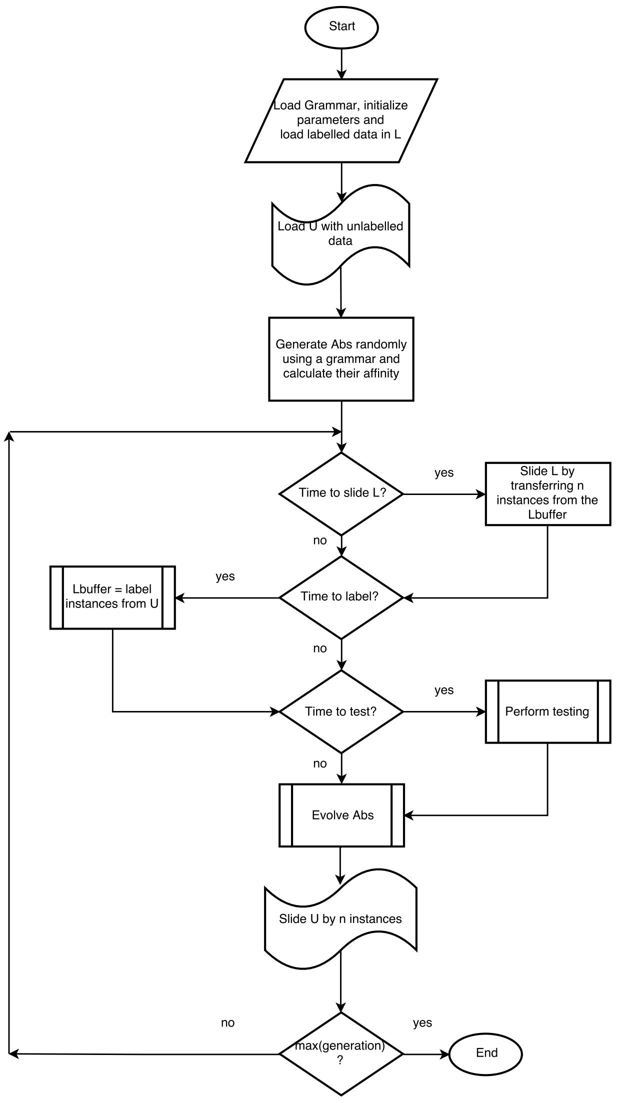

Start
Load Grammar, initialize parameters and load labelled data in L
Load U with unlabelled data
Generate Abs randomly using a grammar and calculate their affinity
Time to slide L?
yes
Slide L by transferring n instances from the Lbuffer
no
Time to label?
yes
Lbuffer = label instances from U
no
Time to test?
yes
Perform testing
no
Evolve Abs
Slide U by n instances
no
max(generation) ?
yes
End

Fig. 1: Flowchart of the UDS variant

A. Representation

An appropriate representation must be selected which can be easily handled by the computer. A correct representation is crucial to ensure that the models learn effectively from the environment, especially when the models have to undergo continuous change.

In our approach we employ an "indirect" representation à la Grammar-based Immune Programming (GIP) [10] which in turn was inspired from Grammatical Evolution [11], where each candidate classifier (i.e., Ab) is encoded as a fixed length binary string (or *genotype*) that maps to a decision tree (or *phenotype*) by means of a user-provided grammar that defines the tree's structure. In our experiments we used two grammars, one for each dataset (see Section II-D). The grammar shown in Figure 2 was used to define the syntax of the decision trees used in the experiments involving the Real-world Dataset, and the grammar shown in Figure 3 was used in the experiments involving the Synthetic dataset.

G = { N, T, P, S }

N = { DT, Attr, n-Attr }

T = { 0, 1, AGE, WORKCLASS, FNLWGT, EDUCATION, EDUCATION-NUM, MARITAL-STATUS, OCCUPATION, RELATIONSHIP, RACE, SEX, CAPITAL-GAIN, CAPITAL-LOSS, HOURS-PER-WEEK, NATIVE-COUNTRY }

S = { DT }

P =

DT ::= AGE 3-Attr | WORKCLASS 8-Attr | FNLWGT 5-Attr | EDUCATION 16-Attr | EDUCATION-NUM 16-Attr | MARITAL-STATUS 7-Attr | OCCUPATION 14-Attr | RELATIONSHIP 6-Attr | RACE 5-Attr | SEX 2-Attr | CAPITAL-GAIN 2-Attr | CAPITAL-LOSS 2-Attr | HOURS-PER-WEEK 4-Attr | NATIVE-COUNTRY 41-Attr

Attr ::= DT | 0 | 1

$\text{n-Attr ::= } \overbrace{\text{Attr Attr . . .Attr}}^{n}$

Fig. 2: Grammar constructed for the real-world dataset

G = { N, T, P, S }

N = { Expr, Op, UnitOp, Var }

T = {x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, +, -, *, /, cos, sin, log, exp }

S = { Expr }

P =

Expr := Op Expr Expr | UnitOp Expr | Var

Op ::= + | - | * | /

UnitOp ::= cos | sin | log | exp

Var ::= x1 | x2 | x3 | x4 | x5 | X6 | X7 | x8 | X9 | X10

Fig. 3: Grammar constructed for the synthetic dataset

In the case of Figure 2, *Attr* represents attribute value. *DT* can be derived into one or more of the possible attributes, for example *AGE*, where *AGE 3-Attr* represents three possible values of attribute *AGE*. Each Attr symbol can in turn derive into one of the possible attributes like *SEX* or the terminals (0 or 1) where 0 and 1 stand for the salaries $>$50K and $\leq$50K respectively.

B. Evaluation

Before optimization, all candidate classifiers (i.e., the Abs) have to be evaluated and assigned a measure of quality (or *fitness*) with respect to the training dataset. In this subsection we provide details about the fitness measures we used, the buffering method we employed for training in a streaming data environment, and how we labeled (i.e, assigned class labels) the incoming raw training instances from the data stream.

1) Fitness: Our classification system employs two measures of fitness *Rank* and *Affinity*. We evaluated the phenotype of the Abs.

Rank is the total number of correct classifications. This is the raw fitness of an Ab and is used to determine its affinity.

Affinity is the main measure of fitness used in CLONALG to evaluate and optimize the Abs. This measure of fitness is derived from the rank of an Ab. Equation 1 is used to calculate the affinity of Ab, where $rank$ is the rank of an Ab, min is the lowest rank of the Ab in the repertoire, and $range$ is $(max - min)$ where max is the highest rank of the Ab in the repertoire. If $range$ is 0, affinity is set to 0.5.

$$affinity = \frac{rank - min}{range} \tag{1}$$

Affinity is in the range 0 to 1, implying that the best possible affinity to achieve is 1 and the worst is 0.

2) Streaming Interfaces: Streaming data can arrive from disparate sources. For example, monitoring a boiler in a power plant may involve processing continuous data arriving from disparate sensors. It is necessary to employ a control mechanism to pool the data from these sources into a buffer, called *sliding window* (SW), an interface between streaming data and the model [8], [12], [16]. It is infeasible to store all the historical data as discussed before, hence employing a replacement policy to SW is inevitable.

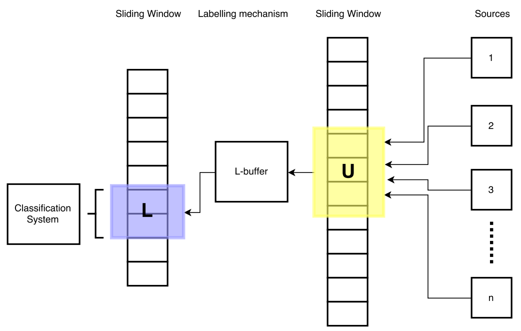


Fig. 4: Streaming interfaces

As depicted in Figure 4, two sliding windows are employed, one for buffering unlabelled data (U) and one for storing labelled data (L) from which the classifiers learn. Interfaces U and L are slid whenever it is *time to slide* them. A labelling mechanism truly labels the data and buffers them in *Lbuffer* from where the instances are transferred to L. All the interfaces employ a First In First Out replacement policy.

3) Labelling: We employed an Active Learning approach to label the data stream as it is infeasible to label all the incoming instances in real-time. Labelling was performed at regular intervals. Algorithms 1 and 2 depict the labelling procedure. If *time to label* is true, a *committee* of size c is created by selecting top c Abs from the repertoire. To make the system less computationally expensive, we randomly selected a subset (U_{temp}) of size r from U to participate in the labelling procedure.

```
   Data: $Ab$, a repertoire of $Abs$ ;
   $U$, an unlabelled set of $Ags$ ;
   $r$, length of the subset of $U$ ;
   $c$, length of the committee ;
   $bn$, number of $Ags$ to be permanently labelled ;
   Lbuffer, a set of permanently labelled $Ags$ ;
   Result: $bn$ labelled $Ags$
1  committee ← select top $c$ $Abs$ from $Ab$ ;
2  $U_{temp} \leftarrow$ randomly select $r$ $Ags$ from $U$ ;
3  $U_{aff-temp} \leftarrow$ assign-affinity($U_{temp}$) ;
4  $buf_{bn} \leftarrow$ select $bn$ least certain $Ags$ from $U_{temp}$ ;
5  Lbuffer ← append $buf_{bn}$ ;
6  $U \leftarrow$ remove $buf_{bn}$ ;
```

Algorithm 1: Algorithm for permanently labelling the Ags

Ags in U_{temp} are assigned an affinity, this process is shown in Algorithm 2. The *committee* temporarily labels the Ag by taking a majority vote of the labels predicted.

The *evaluation-set* is formed by appending the recently labelled Ag to L_{temp} which is the copy of the labelled set of instances. The affinity of each Ab in the repertoire is computed using the *evaluation-set* and an average is taken. The average affinity is then assigned to the Ag. This process repeats for all the Ags available in U_{temp}.

```
   Data: $cmt$, a committee of $c$ best $Abs$ ;
   $U_{temp}$, a randomly selected subset of $U$ ;
   $L_{temp}$, the copy of labelled set of instances ;
   $Ab$, a repertoire of $Abs$ ;
   Result: Label $Ags$ from $U_{temp}$
1  $i \leftarrow 0$ ;
2  while $i < |U_{temp}|$ do
3      Labels ← predict labels for $Ag_i$ using the $cmt$ ;
4      label ← select a label from Labels using a
         majority vote ;
5      $Ag_i \leftarrow$ assign label ;
6      evaluation-set ← $L_{temp} \cup Ag_i$ ;
7      avg-aff ← calculate the average affinity of the $Ab$
         using the evaluation-set;
8      $Ag_i \leftarrow$ assign avg-aff ;
9      $i++$ ;
10 end
```

Algorithm 2: Algorithm for temporarily labelling the Ags

After assigning affinities to Ags, bn Ags with the lowest affinities are selected to permanently include them in *Lbuffer*. That is, the most uncertain Ags are permanently labelled so that the candidate solutions can learn from them. Changes can occur either on the boundary, near the boundary, or further away from the boundary of the search space. Labelling uncertain instances captures the changes occurring on or near the boundary of the search space. Another reason for randomly selecting a subset of U is to potentially increase the search space in order to capture both near and remote changes.

C. Optimization

The Abs are optimized after the process of affinity calculation, labelling, and testing (only when their *times to slide* are true). The evolve process shown in Figure 1 is similar to the CLONALG algorithm [9]. After calculating the affinity of the Abs, n best Abs are selected for *cloning*. The cloned Abs undergo *hypermutation*. The mutated clones are added to the repertoire of Abs after computing their affinities. We then replace d worst Abs from the *reselected* set with d randomly generated Abs. This is performed to potentially increase the search space of the solutions.

D. Datasets

We made use of two datasets for training and testing our system, a real-world dataset and synthetic dataset whose properties are well-known in various data mining (DM) and machine learning (ML) applications in order to compare the performance of our system's variants with two benchmark problems: Atwater et al. works presented in [12], [13] and Zhu et al. work presented in [14].

1) Real-world Dataset: The adult dataset, available freely in the UCI Machine Learning Repository [17], has been widely used to evaluate the performances of various DM and ML-based system and is thus used to compare the performance of the employed variants with the benchmark problems. All the attributes of this dataset were discretized and made nominal, and the missing values were treated as noise and were thus discarded.

2) Synthetic Dataset: For training and testing the proposed system, we need a large volume of ordered data whose temporal and behavioural properties are known. Another reason to generate a synthetic dataset is to simulate concept drift which is lacking in real-world datasets. Synthetic data was generated using a similar process described in Atwater et al. and Zhu et al. [13], [14] which contains all numeric values.

The properties of these datasets can be found in Table I.

classes	Adult	Synthetic with CD
class 1	75%	3.3%
class 2	25%	4%
class 3	-	92.7%
no. of attributes	14	10
no. of instances	45,222	2M

TABLE I: Properties of the datasets used in training and testing. CD stands for concept drift.

III. Empirical Evaluation

We created two variants of the system: labelled-data streaming system and unlabelled-data streaming system. The first variant assumes that the data is labelled by some oracle and the second variant performed labelling.

A. Labelled-data Streaming System (LDS)

1) Parameterization: One sliding window was used which accepts streaming data labelled by an oracle and its size was initialized to 200 to decrease the cost of computation. At every generation, the sliding window would slide and accept sp (sp stands for sliding parameter) new instances by replacing the old sp instances (see Table II). The number of generations was 30000 for the synthetic dataset, and 750 for the adult dataset due to the scarcity of the training instances. The GIP parameters of the labelled-data streaming system were initialized to the values stated in Table II. Size of the population was inspired from Atwater et al. [12]. Out of 120 instances, 40 best genotypes were selected for optimization and 3 clones were generated for each selected genotype. A high mutation rate must be initialized to sustain diversity. The implemented system negates this value, thus the value 1 is the highest possible mutation value. Too much mutation could exploit the genomes; therefore, this parameter was initialized to 3.5.

2) Results and Discussion: Tests were performed every 200 generation using the data available in the window. At generation 1, tests were performed on the untrained candidate solutions with the default affinity of -100. Thus, the first randomly generated Ab was selected to test the Ags in the sliding window. Therefore, for the all variants, a steep growth or drop was observed in generation 1.

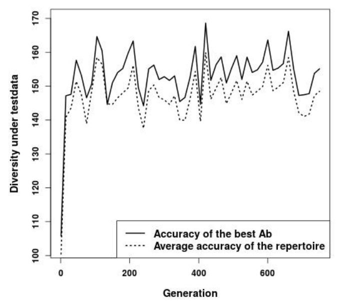
Accuracy: The solid line of Figure 5 represents the number of true positives of the best Ab and the dotted line represents the average number of true positives of the Abs available in the repertoire. The streaming system trained using the labelled adult dataset achieved an average accuracy of approximately 77% (the total number of instances is 200) which is illustrated in Figure 5a. The data available in the window may not be representative which explains the variation in the accuracy. Also, the best Ab could be different at every generation which explains the variation in performance. At generation 135, the best and the average accuracy as shown in Figure 5a was the same implying that all the solutions of the repertoire were the same or performed similarly. But the solutions after generation 135 were diverse and a steady diversity was maintained after generation 400. The average accuracy achieved by the system trained using synthetic dataset with concept drift was approximately 93% (see Figure 5b). Unlike the best Ab in the adult dataset which never performs equivalently, the accuracy in Figure 5b appears to be stable after generation 7,000. Synthetic dataset (see Figure 5b) managed to reach an almost-converged state after generation 2000. A very small amount of diversity was maintained by the system which was enough to achieve good performance. However, the amount of diversity preserved may not be sufficient for the system to perform well in the long run.

Average Detection Rate:Average Detection Rate (DR) is the average of true positives for every class. The mean of average DR for the adult dataset was 0.66, see Figure 6a. The average DR of the *w01P* case[4] of the benchmark problem introduced by Atwater et al. in his work [12] varied from (approximately) 45% to 68% with distinct fluctuations. On the other hand, our average DR varied from 65% to 71% implying that in spite of having a smaller sliding window size and less number of generations, our streaming system achieved better performance. As shown in Figure 6b, the mean of the average DR for the synthetic dataset was approximately 0.70. Contrasting average DRs were achieved at different generations. One of the reasons behind obtaining poor average DR could be the absence of one or more classes of the test cases available in the window. The absence of one class affects the average DR by 33.33%. Both, the *planar* dataset[5] of the benchmark problem and our synthetic dataset were produced through the same process. The average DR of the planar dataset of [13] varied from approximately 30% to 90% whereas the average DR of our synthetic dataset (as shown in Figure 6b) varied from 30% to 100% confirming that the GIP system performed better in the dynamic environment than the GP system under comparison while using a very small window size of 200.

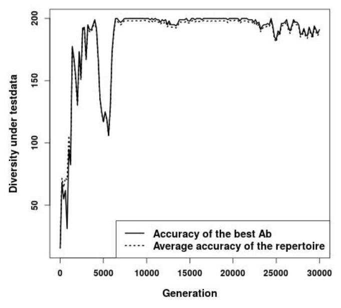
Diversity:Measuring diversity using the training dataset provides more information about evolvability or survivability. We computed the genotypic (binary strings) and phenotypic (decision trees) diversities of the repertoire under *training* dataset using the measure of variance. As shown in Figure 7a, the system managed to maintain a stable genotypic diversity after generation 150. In spite of achieving a stable genotypic diversity, we achieved a gradual increase in the phenotypic diversity which was what we aimed for. The system maintained a diverse repertoire of binary strings throughout its life time. The graph in Figure 7b depicts the diversity achieved by the candidate solutions of the synthetic dataset during training. This is the only experiment where the binary strings are the most diverse and the decision trees reached an almost converged state after generation 5000. Neutral mutations can cause the system to behave in such a way. In spite of maintaining a very diverse set of binary strings, neutral mutations always select the same production rule to generate its corresponding decision tree. Overall, the LDS variant for synthetic dataset did not quite achieve a satisfactory phenotypic diversity under training data. This is not desired for streaming systems, as a converged solution may not incorporate environmental changes which could affect the system's performance.

[4]The *w01P* case of Atwater et al. works [12] employed a sliding window of size 1% of the dataset. Thus for the adult dataset, the size of the sliding window was 480 and the number of generations was initialized to 30,000.

[5]The size for the planar dataset also used in [13] was initialized to 7500.

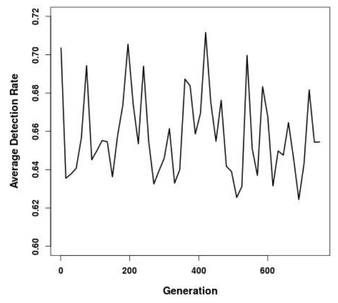


(a) Adult dataset

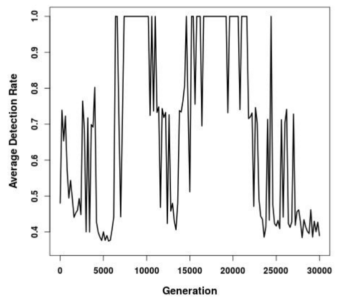


(b) Synthetic dataset with CD

Fig. 5: LDS variant: The dotted line represents the average accuracy of the population and the solid line represents the number of true positives of the best Ab.

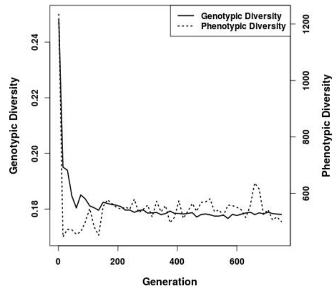


(a) Adult dataset

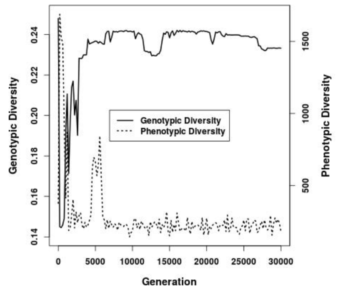


(b) Synthetic dataset with CD

Fig. 6: LDS variant: Average Detection Rate during *training*.



(a) Adult dataset



(b) Synthetic dataset with CD

Fig. 7: LDS variant: Measuring genotypic versus phenotypic diversity for synthetic dataset with concept drift *training*.

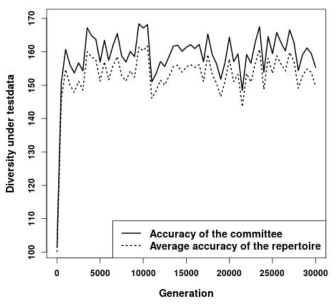
B. Unlabelled-data Streaming System (UDS)

1) Parameterization: Due to the satisfactory performance of the LDS variant, we initialized the GIP parameters to the same values depicted in Table II. The streaming interface parameters highly differ from the LDS variant mostly because of *labelling* and it's cost of computation. As stated in Table II, the size of U was initialized to 1000 as a larger window size consists of more representative data which helps improve performance which slides every generation for synthetic dataset and every 100 generation for adult dataset (to compensate for scarce data). The system was trained with new labelled data every 501 and 1001 generations for synthetic and adult datasets respectively because labelling was performed only at 500 and 1000 generations for these datasets. It was infeasible to perform labelling at every generation as it is a very expensive task which requires a powerful computer. All the interfaces, U, L, and *Lbuffer* slid by 60 instances. A committee of size 10 labelled r randomly selected instances from U into *Lbuffer* and bn most uncertain instances were transferred to L. This constituted in decreasing the cost of computation. The size of the test data was initialized to 200.

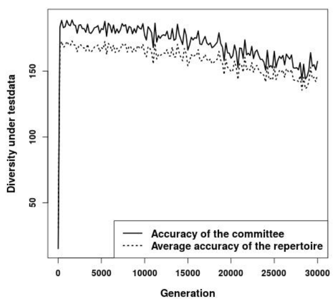
2) Results and Discussion: Analogous to the LDS variant, testing was performed every 200 generations for the synthetic dataset and every 500 generations for the adult dataset. It is impractical to use unlabelled data to perform tests. Therefore, we made use of another data stream of the labelled instances just for the testing purpose. No test exemplars were used to train the system.

Accuracy:The solid line of Figure 8 represents the number of true positives of the committee and the dotted line represents the average number of true positives of the Abs available in the repertoire. The average accuracy of the committee of the adult dataset (as shown in Figure 8a) was approximately 80%, which is 3.8% better than the LDS variant. The similar accuracies of the LDS and UDS variants prove that the labelling process was accurate. The accuracy of Zhu et al.'s system [14] under comparison which employed C4.5 algorithm varied from 81 to 84% making the average 82.5% which was 3% better than our accuracy making our performance a tad unsatisfactory. The average accuracy achieved by synthetic dataset (as shown in Figure 8b) was approximately 86%, which is 9% less than its (LDS) counterpart. The accuracy of our system was 9% better than the benchmark problem which employed C4.5 and 32% better than the benchmark problem which employed Naive Bayes (NB) [14]. Thus, our evolutionary system was more capable of addressing concept drift than its non-evolutionary counterpart. The accuracy of the committee remains stable until generation 10,000 which started declining thereafter. This is mostly because the duration after which the system was trained using new labelled data was not small enough to incorporate fast unpredictable changes (concept drift) making the performance decline gradually.

Average Detection Rate:Analogous to the LDS variant, the mean of average DR in Figure 9a of the adult dataset was approximately 0.68. Similar accuracy and average DR also prove that the labelling process was effective. Similar to the accuracy, the graph of average DR of synthetic dataset as shown in Figure 9b experienced a linear decrease. The mean of average DR was 0.74 which was 6% better than the average DR of the labelled streaming system. This does not imply that UDS variant performed better than its (LDS) counterpart as the LDS variant achieved fluctuating data but without a gradual decrease in its value.

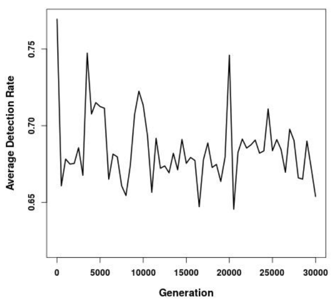


(a) Adult dataset

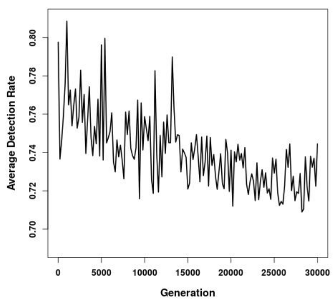


(b) Synthetic dataset

Fig. 8: UDS variant:The dotted line represents the average accuracy of the population and the solid line represents the number of true positives of the committee of *Abs*.

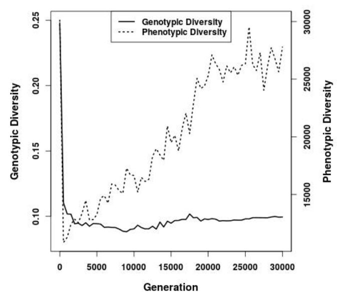


(a) Adult dataset

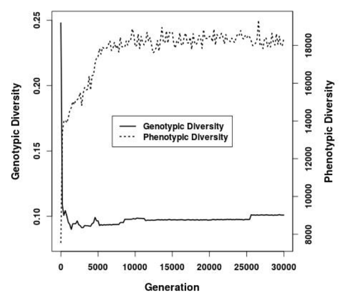


(b) Synthetic dataset

Fig. 9: UDS variant: Average Detection Rate of the best *Ab* during *training*.



(a) Adult dataset



(b) Synthetic dataset

Fig. 10: UDS variant: Average Detection Rate of the best *Ab* during *training*.

Diversity:The linear growth in the phenotypic diversity of the repertoire as shown in Figure 10a while maintaining genotypic diversity demonstrates the success of the UDS variant (using the adult dataset) in achieving evolvability. The phenotypic diversity increased linearly until generation 25,000 after which it appears to attain stability. It is very interesting to note the distinctions obtained in the phenotypic diversities of both the variants (see Figures 7a and 10a). Although, the genotypic diversity for both the variants remained stable after generation 1, the phenotypic diversity increased at different rates. It appears that the UDS variant for the adult dataset achieved more evolvability than its (LDS) counterpart. A very interesting trend was observed in the diversity of synthetic dataset using training data (see Figure 10b). A linear decrease in its accuracy and average DR fabricated an assumption that the results of the variance-based measures will also follow the same pattern. But instead, similar to the adult dataset, a linear growth was obtained in the phenotypic diversity of this variant while sustaining genotypic diversity using training data. A gradual increase in the phenotypic diversity is proof enough that the system did in fact achieve evolvability.

GIP parameters		Streaming parameters	
#gen	30000	$\|U\|$	1000
|pop|	120	$\|L\|$	500 (1000)
|genotype|	240	ttU	1 (100)
select	33.33%	ttL	501 (1001)
#clones	3	sp	60
mutate	3.5	r	500
purge	10%	bn	50
#runs	30	|committee|	10
-	-	$ttLab$	500 (1000)
|Test data|	200	|Test data|	200

TABLE II: Parameters of the UDS variant. Note that, the values within parentheses indicate the values of the adult dataset-based variants.

IV. Conclusions and Future Work

No research was found which employs GIP for classification problems. Also, to the best of our knowledge none of the EA-based systems simultaneously evolve classifiers while labelling streaming data in real-time. The LDS variant produced the most satisfactory results as it performed better than the benchmark problem. The LDS variant trained using the adult dataset achieved the most diverse solutions which is a by product of evolvability. The LDS variant trained using the synthetic dataset with concept drift did not maintain a diverse set of solutions at every time of its execution implying that it did not achieve much evolvability. Typically, in an EA, it is infeasible to introduce variations to a converged solution. But our AIS-based system was capable of introducing variations to it in small amounts whenever the environment changed. This proves that hypermutation does in fact aid in restoring diversity by maintaining a balance between exploration and exploitation, thereby verifying the hypothesis of Dempsey et al. stated in [8]. The amount of diversity achieved for both the datasets was plenty for addressing concept drift and achieving satisfactory performance. Our work also proved that a grammar-based AIS performs better than a non-grammar based system as our streaming system performed better than GP-based benchmark problem.

The satisfactory results achieved by the LDS variant inspired us to continue our work in streaming environments using unlabelled data. Similar results achieved by the adult dataset for the LDS and UDS variants provided us a proof of concept that the active learning process accurately labelled the streaming data.

We noticed that when a dynamic system is set to operate as a static system for a certain amount of generations, it achieves a linear growth in its phenotypic diversity while sustaining genotypic diversity, which is the most essential in dynamic environments as maintaining different solutions can easily address environmental changes. The linearly decreasing performance and the linearly increasing diversity of the models of the unlabelled system using synthetic data with concept drift demands the need of training the system at every generation. Training the system at every generation requires labelling the streaming data at every generation which was infeasible because of its considerable computational expense and our limited access to a powerful computer. We also showed that our GIP system which is evolutionary in nature is more capable of addressing concept drift than the non-evolutionary ML algorithms of C4.5 and NB.

There is a lot of room for experimenting the implemented variants using different types of real-world datasets employed in different settings, static and dynamic. Due to the limitation of time, we could not compare our results with various other ML techniques such as neural networks, SVMs, and so on and thus we have left this as a future work. As future work, the control parameters will be made adaptive in accordance with the system's performance. For example, low performance would indicate the system to increase its mutation rate. Success of the symbiotic-based GP system [12], [13] in streaming environments is an inspiration to employ an archive for retaining past important exemplars which should aid in increasing the average detection rate of the system. Maintaining sub-populations on different parts of the training exemplars and co-evolving them thereby maintaining sub-optimal solutions could potentially aid in sustaining diversity. Our system could benefit from this as one of the goals of this work was to achieve evolvability. Another possible development could be to explore evolvable grammar in order to improve evolvability by achieving more diversity and survivability. Dempsey et al. introduced this concept in his study, Grammatical Evolution by Grammatical Evolution (GE^2) [8] which proved more successful than the traditional GE in dynamic environments. Finally, deploying our system to real-world dynamic applications such as sensor networks in a power plant, analysing electricity demand, and recommender systems to name a few is left as a future work.

References

[1] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[2] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," 2007.

[3] G. P. Zhang, "Neural networks for classification: a survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 30, no. 4, pp. 451–462, 2000.

[4] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *The Journal of Machine Learning Research*, vol. 2, pp. 45–66, 2002.

[5] A. A. Freitas, *A Survey of Evolutionary Algorithms for Data Mining and Knowledge Discovery*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 819–845. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-18965-4_33

[6] M. I. Heywood, "Evolutionary model building under streaming data for classification tasks: opportunities and challenges," *Genetic Programming and Evolvable Machines*, vol. 16, no. 3, pp. 283–326, 2015. [Online]. Available: http://dx.doi.org/10.1007/s10710-014-9236-y

[7] M. O'Neill, L. Vanneschi, S. Gustafson, and W. Banzhaf, "Open issues in genetic programming," *Genetic Programming and Evolvable Machines*, vol. 11, no. 3-4, pp. 339–363, 2010.

[8] I. Dempsey, M. O'Neill, and A. Brabazon, *Foundations in grammatical evolution for dynamic environments*. Springer, 2009, vol. 194.

[9] L. N. De Castro and F. J. Von Zuben, "Learning and optimization using the clonal selection principle," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 3, pp. 239–251, 2002.

[10] H. S. Bernardino and H. J. C. Barbosa, "Grammar-based immune programming," *Natural Computing*, vol. 10, no. 1, pp. 209–241, 2011. [Online]. Available: http://www.scopus.com/inward/record.url?eid=2-s2.0-79952756158&partnerID=40&md5=a8c4c74fb421deb0a8a4565f164d14f1

[11] M. O'Neill and C. Ryan, "Grammatical evolution," *Evolutionary Computation, IEEE Transactions on*, vol. 5, no. 4, pp. 349–358, Aug 2001.

[12] A. Atwater, M. I. Heywood, and N. Zincir-Heywood, "Gp under streaming data constraints: A case for pareto archiving?" in *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '12. New York, NY, USA: ACM, 2012, pp. 703–710. [Online]. Available: http://doi.acm.org/10.1145/2330163.2330262

[13] A. Atwater and M. I. Heywood, "Benchmarking pareto archiving heuristics in the presence of concept drift: diversity versus age," in *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. ACM, 2013, pp. 885–892.

[14] X. Zhu, P. Zhang, X. Lin, and Y. Shi, "Active learning from stream data using optimal weight classifier ensemble," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 40, no. 6, pp. 1607–1621, Dec 2010.

[15] P. Domingos, "A few useful things to know about machine learning," *Communications of the ACM*, vol. 55, no. 10, pp. 78–87, 2012.

[16] A. Vahdat, A. Atwater, A. R. McIntyre, and M. I. Heywood, "On the application of gp to streaming data classification tasks with label budgets," in *Proceedings of the 2014 Conference Companion on Genetic and Evolutionary Computation Companion*, ser. GECCO Comp '14. New York, NY, USA: ACM, 2014, pp. 1287–1294. [Online]. Available: http://doi.acm.org/10.1145/2598394.2611385

[17] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml