Learning from Non-Stationary Data using a Growing Network of Prototypes

Alejandro Cervantes and Pedro Isasi
Department of Computer Science
University Carlos III de Madrid, Leganés, Madrid 29911, Spain
Emails: acervant@inf.uc3m.es, isasi@ia.uc3m.es

Christian Gagné and Marc Parizeau
Laboratoire de Vision et Systèmes Numériques (LVSN)
Université Laval, Québec (Québec), Canada G1V 0A6
Emails: {christian.gagne,marc.parizeau}@gel.ulaval.ca

***Abstract*—Learning from non-stationary data requires methods that are able to deal with a continuous stream of data instances, possibly of infinite size, where the class distributions are potentially drifting over time. For handling such datasets, we are proposing a new method that incrementally creates and adapts a network of prototypes for classifying complex data received in an online fashion. The algorithm includes both an accuracy-based and time-based forgetting mechanisms that ensure that the model size does not grow indefinitely with large datasets. We have performed tests on seven benchmarking datasets for comparing our proposal with several approaches found in the literature, including ensemble algorithms associated to two different base classifiers. Performances obtained show that our algorithm is comparable to the best of the ensemble classifiers in terms of accuracy/time trade-off. Moreover, our approach appears to have significant advantages for dealing with data that has a complex, non-linearly separable topology.**

I. Introduction

Learning from non-stationary data, or Non-Stationary Learning (NSL), has become a promising field specially with the increasing availability of continuous data sources from ubiquitous computing and networking devices.

The relation between Incremental Learning (IL) and Non-Stationary Data (NSD) depends on the features of the data source. Different research fields address topics specific to applications of different nature, such as:

- **Learning with Concept Drift** [1]. Concept drift appears when new data is not consistent with past data. This means that some unobserved aspect of reality was correlated with expected results, and varying values of that "hidden context" must force variation on learned models. Most references in this field assume a principle of batch learning, that is, complete training data is available and can be processed before the algorithm is requested to provide an answer.
- **Learning from Large Datasets** [2]. Defined as scenarios where a huge amount of learning data is available, which imposes constraints on the space required for representation.
- **Learning from Data Streams**. Defined by a continuous stream of learning data, which makes preprocessing of data unfeasible. These problems are also assumed to include the restriction that data can only be read once (one-pass learning) and usually assume non-stationarity of data [3]. A recent review of current techniques in this field can be found in [4].
- **Real-time Learning**. A particular case of Learning from Data Streams where hard time constraints are also imposed. The learner has to be ready to respond in a specific (usually reduced) period of time. This may also require special-purpose hardware for many applications [5].

TABLE I. Relation between research topics and features required in the algorithms. SP: Sequential Processing, OP: Online Processing, ATR: Anytime Response, AC: Adaption to Changes, TI: Time Invariance, SI: Space Invariance

Learning domain	SP	OP	ATR	AC	TI	SI
Concept drift	Yes	No	No	Yes	Weak	Weak
Large datasets	No	Yes	Yes	No	No	Hard
Data streams	Yes	Yes	Yes	Usually	Weak	Hard
Real-time from streams	Yes	Yes	Yes	Usually	Hard	Hard
Non-stationary data	Yes	Yes	Yes	Yes	Weak	Weak

An early definition from Aha [6] limits incremental learning to sequential processing of examples in an instance based learning algorithm. This restrictive definition, however, provides little insight on the key issues of IL. Other more detailed definitions [7]–[10] include a mixture of task features that incremental algorithms need to address. They vary slightly depending on whether the topic of the paper is related to learning from large datasets, or to learning from data streams.

Table I summarizes the different application fields with respect to the task and algorithm features, and specifies the "Incremental Learning from Non-Stationary Data" field which is the scope of our paper. Algorithms may be described in terms of several features: Sequential Processing (SP), Online Processing (OP), Anytime Response (ATR), and Adaptation to Changes (AC). SP means that data is learnt as it arrives, thus taking data ordering into account. OP implies that a complete or final data set is never expected, data should be processed in a single pass. ATR states that a prediction of the data class label must be available at any time. Finally, AC specifies that embedded class models must be able to adapt to changes in data, either stationary or non-stationary.

Depending on the research topic, two constraints may also be imposed on algorithms: Time Invariance (TI), and Space Invariance (SI). TI means that each training sample should be processed in constant time, regardless of the amount of data encountered so far. SI implies that internal data structures should require an approximately constant amount of memory, also independent of the amount of previously processed data.

This paper presents an algorithm designed for incremental learning of non-stationary data. It complies with the aforementioned features: sequential, online processing, anytime

response and adaptation to changes. Throughout this work we shall refer to it as Growing Prototype Network Classifier (GPNC).

The rest of this paper is organized as follows: in Section II, we describe relevant contributions to the field, while in Section III we describe the proposed GPNC algorithm. Section IV describes the experimental setting, in terms of datasets to be used and reference algorithms selected for comparison. In Section V we first present some preliminary experimentation on GPNC to specify deciding values for some of its parameters, and then we report experiments on the defined datasets and compare the results of GPNC with those of selected IL algorithms proposed elsewhere. Finally, we summarize the results of this work in Section VI.

II. Related work

One of the first models that directly addresses many of the concerns specific to non-stationary learning was Adaptive Resonance Theory (ART [11]). The authors address a core issue for non-stationary systems in what they call the stability versus plasticity dilemma: learners must be able to react to significant events (plasticity) but not every event has to be learnt (stability) as that can be proved to lead to unstable behavior. ART deals with the problem by being able to change between two different "learning" modes, and autonomously recognize when the algorithm should change between them.

This key issue is the concern of most of the algorithms that are specifically designed to deal with adaptation to change in an incremental way. Underlying representation, however, may vary, from several types of classification trees to neural network or support vector approaches.

IB3 [6] is often considered one of the first efforts that uses the nearest neighbor approach. Incremental learning is based on an adaptive learning and forgetting mechanism of instances. IB3 keeps or deletes instances based on the accuracy of the instance compared to the relative frequency of its class.

In the area of incremental learning of decision trees, the basic algorithm is called CVFDT [12]. It is a decision tree classifier that is able to process new data incrementally in a very fast way, thus making it adequate to learning from real-time data streams. It is also able to learn from non-stationary data, using a sliding window of stored patterns (that is, a time-based instance forgetting mechanism) that is adjusted dynamically. It also replaces parts of the tree if they are invalidated by new data. Recent versions of this family of algorithms can be found in [12]–[15].

Other approaches use neural networks [10], dynamic Generalized Linear Models [16], a learning Kalman filter framework [17], or Genetic Programming [18].

Closely related to the present work, clustering algorithms based on an incremental building of data topologies have been modified to perform incrementally such as in Supervised Growing Neural Gas (GNG [19]) and adapt to change if change is slow enough [20]. This clustering methods can be adapted to classification, like in Incremental Learning Vector Quantization (ILVQ [21]). Our approach belongs to this family of classifiers.

A field that has increasing interest in non-stationary environments is ensemble-based learning. A review of the specific mechanisms used in this field can be found in [22]; one strong reason to use ensembles is that the incremental and adaptive part of the algorithm can be achieved independently of the nature of the base classifier. This is done by just dividing data in chunks and later providing those chunks to any base classifier. However this introduces a dependency on the data chunk size and has performance drawbacks. Two different types of algorithms in this area are:

- In Learn++ [23] a dynamic combination of classifiers is used to deal with non-stationary data. The ensemble combines classifiers based on the dynamic weights assigned to each of them depending on the success rate on the current data set. Several versions of this classifier have been developed for different purposes [7], [24], [25].
- Algorithms derived from the bagging or boosting batch learning algorithms, where weighting is associated with patterns, not classifiers. A review of several versions of algorithms of this family can be found in [26]. A recent algorithm that seems able to achieve good accuracy at the cost of speed is online non-stationary learning using boosting (ONSBoost [27]).

III. Description of the GPNC algorithm

GPNC is an incremental learning algorithm that generates a network of linked prototypes P, each labeled with one of the class labels in training data. This set P is available at any time to classify unlabeled data using the nearest neighbor rule.

Each of the network nodes $P_i \in \{P\}$ is defined by the following attributes:

- A position $\mathbf{x_i}$ vector, and a corresponding velocity vector $\mathbf{v_i}$.
- An objective position vector $\mathbf{o_i}$.
- The prototype class label c_i.
- A fitness value f_i that indicates the degree of success in classification. This value is 0 when a prototype is newly inserted in P and is adjusted in step 7 of the algorithm.
- A set of non-directed links $\{L_i\}$ with other prototypes, where the link between prototypes P_i and P_j is labeled L_{ij}. Each link L_{ij} has an integer *age* value, $a_{ij} \geq 0$. For convenience, the set of prototypes with a link with P_i is called hereafter $N(P_i)$ (neighbors of P_i).

The algorithm learns in time steps. At each time step it accepts an input vector (training pattern) T defined by its position and class $\{\mathbf{x_T}, c_T\}$. In this work, time steps will be indicated as exponent: $\mathbf{v_i^t}$ is the velocity vector of prototype P_i at time step t.

A. Description of the Algorithm

The learning algorithm can be described as follows.

1) Start with an empty $\{P\}$
2) Receive a training pattern $T = \{\mathbf{x_T}, c_T\}$
3) Find the nearest (P_w) and second nearest (P_z) prototypes in P, if $|P| > 2$.
4) Compute the Boolean decision variable GC (Growth Condition) using Eq. 1. This variable is later used to decide if the network of prototypes has to **grow** or **adapt**.

$$GC = |P| < 2 \vee \forall P_i \in \{P\},\ c_i \neq c_T \vee \forall i \in \{w, z\},\ \|\mathbf{x_{P_i}} - \mathbf{x_T}\| > r_i \quad (1)$$

That is, GC is $true$ if either there are less than two prototypes in P, or there is no prototype in P of class c_T, or distance from T to P_w or P_z is greater than the corresponding "influence window" value r_w and r_z. These values are computed using the algorithm in ILVQ [21] based on the distance between P_i and its linked neighbors $N(P_i)$:

- r_{within} is the average distance to neighbors of the same class as P_i
- r_i is then calculated as the distance to the most distant neighbor of different class that is closer than r_{within}

- **Growth**: If $IC = true$, a new prototype P_T (Eq. 2) is inserted in {P}. Also, this new prototype becomes linked to P_w and P_z, but only *if* distance from those prototypes to P_T is less than their corresponding windows r_w and r_z (Eq. 3).

$$\{P\} \leftarrow P_T = \{\mathbf{x_T}, c_T\} \quad (2)$$

$$\{L_i\} \leftarrow L_{Ti},\ \forall i \in \{w, z\} \mid \|\mathbf{x_{P_i}} - \mathbf{x_T}\| \leq r_i \quad (3)$$

- **Adaptation**: If $IC = false$, the model is updated. This phase involves the following steps:

  i) Ages of all links from or to P_w are increased by one unit, and if they reach a_{max} the link is removed:

$$a_{wi} = a_{wi} + 1 \quad \forall i\, P_i \in N(P_w)$$
$$\forall a_{wi} \geq a_{max} \quad \mathbf{remove}(L_{wi}) \quad (4)$$

  ii) If exists, age of the link L_{wz} is reset to 0; if it does not exist, the link is created.

$$a_{wz} \leftarrow 0, \quad \text{if } L_{wz} \in \{L_w\}$$
$$\{L_w\} \leftarrow L_{wz}, \quad \text{if } L_{wz} \notin \{L_w\} \quad (5)$$

5) The objective vector for P_w is updated using Eq. 6. Also, objectives for all direct topological neighbors of P_w ($N(P_w)$) are updated using Eq. 7.

$$\mathbf{o_w^{t+1}} \leftarrow \begin{cases} \mathbf{o_w^t} + \alpha \cdot (\mathbf{x_T} - \mathbf{o_w^t}), & \text{if } c_{P_w} = c_T \\ \mathbf{o_w^t} - \gamma \cdot \alpha \cdot (\mathbf{x_T} - \mathbf{o_w^t}), & \text{if } c_{P_w} \neq c_T \end{cases} \quad (6)$$

$\forall i, P_i \in N(P_w)$,

$$\mathbf{o_i^{t+1}} \leftarrow \begin{cases} \mathbf{o_i^t} + \beta \cdot (\mathbf{x_T} - \mathbf{o_i^t}), & \text{if } c_{P_i} = c_T \\ \mathbf{o_i^t} - \gamma \cdot \beta \cdot (\mathbf{x_T} - \mathbf{o_i^t}), & \text{if } c_{P_i} \neq c_T \end{cases} \quad (7)$$

Objective update equations depend on the parameters α, β and γ. α and β represent the different influence of T depending on whether the prototype is P_w or a neighbor of P_w. This type of adaptation assumes that $\beta < \alpha$. That is, influence of the training pattern decays for neighbors.

Note that for any P_i considered for update, if $c_T = c_i$, P_i shall be *attracted* toward $\mathbf{x_T}$ due to the positive sign in the equation; on the contrary, if $c_T \neq c_i$, P_i shall be *repelled* from $\mathbf{x_T}$. The parameter $\gamma > 0.0$ is used to balance the relative importance of both effects.

6) Simulate the attractive force towards the prototypes' objective positions. First Eq. 8 calculates an acceleration vector that is proportional to the distance to the prototype objective; then it updates velocity; secondly, Eq. 9 updates the positions of all the prototypes in P using the calculated velocity. These equations are simplified versions of movement equations in the PSO algorithm [28]. The parameter $\nu \leq 1.0$ acts as a dissipating factor for velocity, and μ determines if movement will be attenuated or oscillatory around the objective vector.

$$\mathbf{v_i^{t+1}} \leftarrow \nu \cdot \mathbf{v_i^t} + \mu \cdot (\mathbf{o_i} - \mathbf{x_i^t}) \quad (8)$$
$$\mathbf{x_i^{t+1}} \leftarrow \mathbf{x_i^t} + \mathbf{v_i^{t+1}} \quad (9)$$

7) Update fitness of P_w using Eq. 10. If f_w is decreased under the value of a parameter ϕ, the prototype P_w is deleted.

$$f_w^{t+1} \leftarrow \begin{cases} (1-\delta) \cdot f_w^t + 1, & \text{if } c_w = c_T \\ (1-\delta) \cdot f_w^t - 1, & \text{if } c_w \neq c_T \end{cases} \quad (10)$$

B. Comments on the algorithm

The algorithm implements three different mechanisms in order to comply to the features we need in incremental learning from non-stationary data:

- **Growth** phase adapts the network of prototypes as needed. The GC criterion adapts P to the presence of data of new classes. The set of prototypes also grows if a new pattern is so distant to prototypes in P that it cannot be considered represented by the current prototypes. That is the role of the "influence windows" r_w and r_z. If the new pattern is outside the region defined by those values, then the pattern will be inserted as a new prototype.
- **Adaptation** phase aims to fix an objective position that is a weighted sum of the new pattern and the previous position. Prototypes are given velocities as a result of calculating an acceleration toward the objective position, as a way to be able to track more efficiently changes in data distributions.
- **Forgetting** mechanisms are of two types: first, the fitness based mechanism detects contradiction between new data and previous data. This mechanism also deletes noisy prototypes, because their fitness will be quickly reduced under the deletion threshold. Secondly, the decay factor in fitness (δ) and ageing of links reduce influence of past data, because even prototypes that had many good classifications in the past will be eventually removed if they are not "refreshed" by new data. They will also be more prone to deletion if new contradictory data appears.
- **Locality**. The algorithm is local by design: all adjustments are based on the local information stored in each prototype, and propagate only to its directly linked neighbors. This means that the algorithm shall be able to deal with simultaneous data distributions that have different frequencies or dynamics.

IV. Experimental setting

A. Measures of accuracy

Evaluation of performance of non-stationary learning algorithms can be done in several ways that take into account the nature of the data:

- **TestThenTrain** (TT) (also known as Prequential error evaluation) performs prediction on each pattern before using it to train the model. It requires no parameters.
- **TestThenTrain by blocks** (TB) is evaluated as TestThenTrain but uses a given number of patterns for testing before using that same block of patterns for training. It requires a single parameter, block size.
- **Training followed by testing on a Holdout set** (HT). This requires setting two parameters: the training and test block sizes (the number of patterns used for each phase). Then it performs several blocks of training followed by testing on the corresponding block of a separate testing dataset. Block phases need not be of the same size.

Discussion on how prequential error relates to holdout error can be found in [9], and analysis of prequential statistics is available in [29]. For the field of classification of data streams it is unlikely that a separate sample of the distributions may be available, so TT is usually proposed as the preferred evaluation method. However, HT evaluation has the advantage that train and test datasets can be designed with different characteristics. For instance, it may be decided that train and test datasets have different levels of noise, or different lengths.

B. Benchmarking Datasets

Some datasets that are commonly used in the literature are listed in Table II. In order to be able to compare our results to published data ([7], [10]) we also provide the evaluation method used for each dataset. We shall use this same evaluation method in our experiments. We also list the data block value used for either test data, train data, or both.

TABLE II. Relation of datasets frequently used in literature for learning from non-stationary data

Dataset	Size	Attributes	Classes	Evaluation
Artificial Datasets				
sea	50000	3	2	HT, block 250
CB (4 sets)	10000	2	2	HT, train block 25, test bl. 1024
Real Datasets				
elec2	43512	7	2	TT
weather	18160	8	2	TB, block 30

The **SEA Concepts dataset** [8] (*sea*) has a linear boundary that divides the two classes. At fixed moments in time, the linear boundary is moved. The training dataset includes some noise, typically 10% to make the problem more challenging.

The **Checkerboard datasets** (*CBconstant, CBexponential, CBpulse, CBsinusoidal*) were introduced in [7]. Distributions are spatially located as an XOR-like pattern ("checkerboard") and rotate with four types of movement: constant speed, exponentially increasing speed, sudden change and sinusoidal speed. Data is sampled only from a small window of the attribute space. These datasets are periodical, so they may be used to detect long-term memory in algorithms, that is, ability to remember previous data.

The **Nebraska Weather Prediction dataset** (*weather*) contains meteorological data from a single station over 50 years. We have used the dataset processed in [7], which contains only eight features. The two classes mean "rain" (31%) and "no rain" (69%). This dataset should show yearly periodicity that might be exploited by algorithms.

The **New South Wales Electricity Market Pricing dataset** (*elec2*) was first proposed in [30] and is also used in [10], [31]. It contains samples collected at 30-minute intervals that detail demand in New South Wales, Victoria, and amount of electricity scheduled to be exchanged between those Australian states. Prediction is whether the price will raise or go down. From the initial attributes we removed the date attribute. As the original dataset had missing attributes we completed those with the first value found for each of the missing attributes.

We have not considered in this work other simple artificial datasets using rotating hyperplanes such as those used in [12]. Other real-world datasets can be found in [16], [17], [32]–[34] but most of them either are not publicly available or are not used besides the original work, so we have not considered them for benchmarking purposes.

C. Algorithms

We are performing analysis of performance of our algorithm using the previously mentioned datasets and comparing its results to the results of some of the latest algorithms in the field of NSL. To encompass different tendencies in the field, we have performed experiments both using single-classifier approaches and ensemble approaches.

The following single classifier algorithms have been used:

- Adaptive VFDT [15] is a version of VFDT that uses a change detection mechanism in order to define the length of a window of relevant recent patterns. For tests we have used the ADWIN [14] change detector, as it seems to provide good results. ADWIN detects changes in a series of patterns; it only uses a confidence bound parameter, δ_{ADWIN}. ADWIN is used to adjust a learning window over training data for any learning algorithm in the following way: if changes are detected in data, length of the window is reduced; when change is not present window length is increased.
- Stochastic Gradient Descent (SGD) techniques have been proved to be successful options for online classification of large datasets using SVMs in [35]. The accuracy loss due to its approximated nature (compared to exact SVM techniques) is easily balanced when computational cost are taken into account. The method is incremental as SGD does not remember which examples where used to adjust the classifier in previous iterations. SGD update functions are provided for several classification and regression algorithms, including Adaline, Perceptron, and linear SVMs. SGD is not really designed for NSL, but it is able to gradually update a single surface of separation of the underlying SVM in order to adapt changes, and retains no knowledge of previous phases of learning. As such, it adapts well to many slow-changing datasets even if new data contradicts previous data. However as it works with a linear SVM, success with this algorithm depends on separability of the data distribution.

Base algorithms for the ensemble classifiers have been selected in order to be able to compare a fast but simple approach with a slower approach more able to represent complex data. For each of these algorithms we have performed experiments with the following two different base incremental classifiers commonly used in the field of learning from data streams. Note that these algorithms are not designed for non-stationary learning at all; non-stationarity will be addressed by the ensemble mechanism.

- Incremental Naive Bayes (INB) incrementally builds a Naive Bayes classifier that represents data on the assumption of attribute independence. INB is not designed to adapt its model to change, but can provide a fast and accurate representation of stationary data in certain cases.
- VFDT-HoeffdingTree (HOEFT) [5] is an incremental method that is able to incrementally construct a representation of data in terms of Hoeffding trees. A Hoeffding tree is a decision-tree that is able to learn patterns in constant time (independent of size). We have selected VFDT as base algorithm because of its ability to classify datasets with complex representation.

With these methods as base classifiers, experimentation was performed using the following ensemble-based learning algorithms.

- DWM [31] is an ensemble method that constructs a set of classifiers (experts) based on their measured accuracy on successive blocks of patterns of a fixed size. Each received pattern is first used to test the ensemble and weight the experts. Member classifiers weights are adjusted by an amount β depending on whether they provide correct or incorrect predictions. After evaluation, incremental learning is performed for all the experts. Each p patterns a classifier creation/removal phase decides if a new classifier is inserted, and classifiers whose accuracy was worse than a parameter γ are deleted.
- Oza Online Bagging with ADWIN (OBAG$_A$) [26] adds the ADWIN change detection mechanism to the Online Bagging method proposed in [36], [37]. In bagging, training data is resampled to decide the number of copies of each pattern used to train the base classifiers. Prediction is based on the unweighted voting of the base classifiers. The authors report that OBAG$_A$ is able to achieve very good results in terms of accuracy but has a high cost in terms of memory and time when compared to other algorithms. ADWIN makes Online Bagging suited to NSL by providing an adaptive window of training data.
- Oza Online Boosting with ADWIN (OBOOST$_A$) [26] the combination of Online Boosting [36], [37] and the ADWIN change detection method, that plays the same role as in (OBAG$_A$). Online Boosting is based on allocation and incremental update of a distribution of weights to patterns, in such a way that patterns that are misclassified raise their weights and patterns well classified decrease theirs. These weights are then used to evaluate each classifier and also to decide the influence of the vote of each classifier on the final output of the ensemble.
- Online Non-stationary Boosting (ONSBOOST) [27] is also based in Oza version of Online Boosting [36], [37] to adapt the base classifiers weights based on their performance. It uses a fixed-size window for testing patterns which is updated on periodical intervals of K patterns. This is done as part of an ensemble update phase that is executed periodically. In this phase, ONSBoost performs two operations: it slides the testing window and checks all of the base classifiers against the testing data. If the ensemble works better *without* any of the classifiers, it removes the classifier and inserts a new one that replaces the former. As a result, the ensemble size is kept constant, which makes it suitable for large datasets as memory does not grow with time.

We have used the Massive Online Analysis (MOA [38]) software environment to provide the algorithms and run the experiments. The following algorithms were used as provided by that environment. The SGD implementation came from the WEKA [39] classifiers suite and ONSBoost was implemented by the authors.

V. Experimentation

A. Results of preliminary experimentation on our algorithm

Preliminary experimentation was performed using the *sea* dataset as it is widely used in literature. We have not performed exhaustive optimization of the GPNC parameters but used this dataset to provide the sample values found in Table III. These values were then used for evaluation of all other datasets.

TABLE III. GPNC parameters

	Description	Value
a_{max}	Maximum link age	20
α	Objective update for closest prototype	1
β	Objective update for neighbors	0.25
γ	Factor for repulsion	0.05
ν	Inertia coefficient	0.2
μ	Memory coefficient	0.025
δ	Fitness decay factor	0.01
ϕ	Deletion threshold for fitness	0.1

A separate analysis was performed to decide the effect of the decay factor for fitness δ, as this mechanism has an important effect on the algorithm behavior. Resulting models P generated by each experiment of the algorithm will have different number of prototypes; this number is both a measure of space requirements and time performance of the algorithms, as classification involves calculating distances to all the prototypes.

Results for the experiments on the *sea* dataset are summarized in Table IV.

It is clear that as the number of prototypes increases, so does accuracy; however more prototypes also means a significant increase in processing time.

In Figure 1 we represent the evolution of the number of prototypes in P during processing of this dataset for three

TABLE IV. Results for the dataset *sea* for different values of δ. Accuracy shows classification success rate in % using the Holdout Set (HT) performance measure

	$\delta = 0.0$	$\delta = 0.001$	$\delta = 0.01$	$\delta = 0.025$
# prototypes	1507	360	35	7
Accuracy (%)	88.42	88.77	86.16	84.06
Execution time (s)	962.98	267.86	19.56	10.40

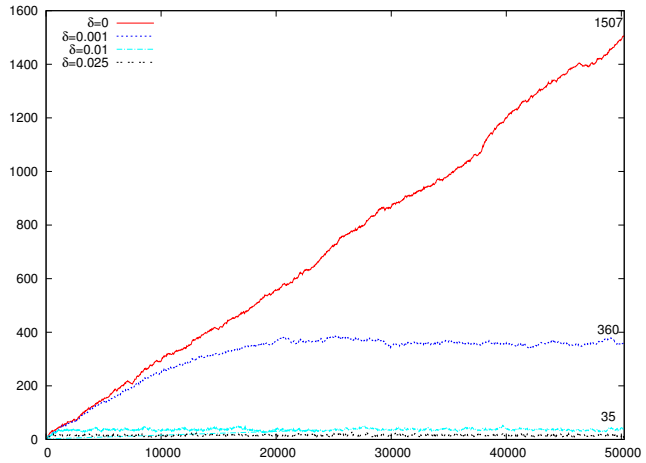

Fig. 1. Evolution of the number of prototypes for the *sea* dataset with different values of the δ parameter: 1.0, 0.025, 0.01, 0.001.

different values of the δ parameter, including a base version where fitness does not age.

It is clear that the algorithm requires setting this parameter in order to comply with the time or space invariance constraints (TI and SI). With $\delta = 0.0$, the number of prototypes increases linearly with the number of patterns being processed. Conversely, it reaches an stable value for the rest of the configurations. Obviously the value for this parameter might be optimized for each problem; however, in this work we have selected a value of $\delta = 0.01$ for all the experiments, as the number of prototypes stabilizes quite early.

B. Results on the datasets provided in literature

Table V presents the results of experiments on each of the datasets.

An issue that was quickly raised was that SGD was greatly influenced by the way evaluation was performed. It seems that the algorithm is not able to perform adequately on separate blocks of train and test data, but it performs much better when we use TT evaluation such as in the *elec*$_2$ dataset. Using TT evaluation with SGD reported results of 81.65% in *sea* and 67.88% in *weather*. Its results did not change significantly in the rest of the datasets. This effect was also present to a lower extent when using the DWM classifier.

Independent execution of the base algorithms was useful to provide some additional information to explain results on these datasets. For instance, dataset *sea* can be classified very accurately by single INB (92.38%) or HOEFT (93.20%) classifiers. This means that effect of non-stationary data is probably averaged over the whole dataset in this form of evaluation, as in SEA changes occur abruptly at certain moments of time. That is, measure of the total error is not the best measure for datasets when they are stationary during most of the testing period.

In the *sea* dataset, results of all versions of GPNC achieve worse performance than the rest of the algorithms (except SGD). This may be a weakness of our algorithm that has to be studied. As GPNC is based on a Nearest Neighbor approach, it may be misled by the fact that training data in this dataset has a significant degree of added noise, which may be affecting results. It is also clear that in this dataset reducing the number of prototypes is detrimental to accuracy.

In the *CB* datasets, reference algorithms have a low accuracy because of the limited representation ability of the base classifiers themselves, as data is organized in a XOR-pattern. The HoeffdingTree representation is clearly better, but not enough as AHOEFT only achieves between 54.6% and 61.28% success in these datasets. Boosting ensembles are able to increase success between 65% and 75%. However GPNC clearly provides the best results. It seems that GPNC can represent the XOR-like moving distribution of data more accurately. In these datasets, increasing δ helps the algorithm to achieve good results.

In the *weather* dataset, GPNC has results comparable to the rest of the algorithms ranging between 72.37% for $\delta = 0.01$ to 74.82% for $\delta = 0$). The accuracy/performance trade-off is of the same type as in *sea* dataset. OBAG$_A$ and ONSBOOST$_A$ perform slightly better ($74 - 75\%$) on this dataset.

In dataset *elec*$_2$, some of the reference algorithms clearly outperform the rest: SGD, ONSBOOST and DWM, with HoeffdingTree base classifier. The reason may be that, in this dataset classification has a strong dependency on very recent history of patterns. That is, this dataset can be classified more accurately if only the most recent patterns are retained and used for classification. This is supported by the fact that GPNC achieves better results with $\delta = 0.01$, performing almost as well as the former algorithms. Retaining past knowledge in this case seems to be detrimental for accuracy. OBAG$_A$ and OBOOST$_A$ have an intermediate result when using the HoeffdingTree representation. The fact that SGD is the best classifier means that data is separable by a single linear boundary.

The lower part of Table V shows execution time for each of the algorithms. It is clear that using a Hoeffding tree base classifier has also a major impact in the ensemble methods over the simpler INB. It is also clear that the better performance of ONSBOOST in terms of accuracy is balanced by a much greater cost in terms of time over the OBOOST$_A$ algorithm. ONSBOOST is clearly the slowest of the ensemble algorithms except in very specific cases, and DWM is the fastest of the ensemble algorithms.

Comparing GPNC times with times of the reference algorithm, when $\delta = 0.01$ it has a reasonable execution time; version with $\delta = 0$ however has a much increased computational cost, comparable to the ensemble classifiers with HoeffdingTree base classifier.

VI. Conclusions

Incremental learning on non-stationary data is becoming an increasing subject of study. Many algorithms have been developed to deal with different aspects of learning in such environments. Our definition of a proper algorithm for non-stationary learning (NSL) includes several aspects: Sequential Processing, Online Processing, Any-Time Response, and Adapts to Change. Also, in many applications both space and time invariance constraints become mandatory, in the sense that algorithms should have stable time/size performance over the learning process.

TABLE V. ACCURACY AND EXECUTION TIME OF COMPARED CLASSIFIERS OVER THE SEVEN DATASETS. THE FIRST SET OF EXPERIMENTS ON THE ENSEMBLE ALGORITHMS USES A NAIVE BAYES BASE CLASSIFIER, THE SECOND SET USES A HOEFFDINGTREE CLASSIFIER. THE "BASE" COLUMN SHOWS THE RESULTS OF A SINGLE CLASSIFIER OF THE RELEVANT TYPE. *: 81.65% WITH TT EVALUATION. †: 67.88% WITH TT EVALUATION.

		GPNC		Non-ensemble		Ensemble with Naive Bayes					Ensemble with HoeffdingTree				
	Dataset	$\delta = 0$	$\delta = 0.01$	SGD	AHOEFT	Base (NB)	DWM	OBAG$_A$	OBOOST$_A$	ONSBOOST	Base (HOEFT)	DWM	OBAG$_A$	OBOOST$_A$	ONSBOOST
Accuracy (%)	sea	88.42	86.16	35.83*	96.19	92.38	90.57	95.87	92.64	92.24	93.20	88.57	95.79	92.42	93.34
	CBconstant	68.41	82.08	53.97	54.60	51.38	57.14	57.66	54.45	56.22	54.29	57.11	56.04	67.25	70.97
	CBexponential	72.78	82.68	52.48	56.36	52.22	57.76	56.93	58.05	58.68	56.48	60.05	59.72	66.82	69.46
	CBpulse	76.16	83.54	53.43	59.57	46.68	53.81	51.34	52.60	54.56	60.83	54.22	56.86	72.59	72.05
	CBsinusoidal	74.10	83.38	52.38	61.28	45.52	55.24	50.67	54.41	58.88	62.32	55.30	56.78	72.49	75.74
	weather	74.82	72.37	32.58†	74.06	69.24	69.65	72.76	73.91	73.84	73.12	69.70	75.58	74.18	75.83
	elec$_2$	78.57	87.42	90.86	82.07	73.26	82.70	78.29	77.58	66.44	78.36	86.50	82.59	83.68	89.02
Execution time (s)	sea	962.98	19.56	2.21	3.48	1.18	2.50	6.56	6.01	112.77	4.75	4.44	20.79	263.36	129.26
	CBconstant	67.63	11.17	4.04	5.31	3.45	6.55	11.48	7.15	547.80	7.52	7.38	17.04	62.84	503.75
	CBexponential	48.08	13.33	4.19	5.77	3.57	6.61	11.55	7.29	467.03	7.31	7.58	18.03	46.54	515.72
	CBpulse	47.11	11.59	4.15	4.54	3.50	6.51	11.53	6.99	454.02	7.16	7.29	16.54	64.92	507.30
	CBsinusoidal	68.84	11.86	4.20	4.91	3.48	6.57	11.59	8.69	446.58	8.00	7.36	17.03	53.81	487.45
	weather	178.64	12.01	2.54	6.70	1.61	3.05	11.47	10.81	47.01	7.32	5.36	44.47	357.45	154.41
	elec$_2$	947.88	19.46	1.87	7.60	2.46	5.10	17.91	16.06	103.69	25.49	10.43	59.89	964.23	1546.09

We propose GPNC, a Nearest Prototype classifier that incrementally constructs a network of labeled prototypes that represent the topology of the data. This network is able to grow and evolve depending on the new information received as training data. The algorithm is able to incrementally add new classes and follow data even if the data generation process is non-stationary. Adaptation of the network is also achieved by moving the prototypes when new data appears in their proximity.

Also, a fitness based mechanism is introduced to be able to deal with contradiction in data. In a non-stationary environment with sudden changes, new data may be of different class than the prototypes in its area of the attribute space. A fitness value that represents performance is updated for each prototype, increasing when it performs good classifications, and decreasing when it contradicts data. When fitness drops under a certain level, the prototype is deleted.

To meet time and space invariance constraints, the size of the prototype set must not grow in an uncontrolled way as more data becomes available for training. This is achieved by adding a time-based forgetting mechanism, not to patterns, but to prototypes' performance. For each prototype, fitness decays with age, so old prototypes become more likely to be deleted.

We have tested our algorithm in several datasets used in the literature. Our results show that it performs better than most state of the art algorithms when dataset has a complex representation. When it does not, its performance in terms of accuracy is usually worse than the best ensemble-based algorithms. However, its computational cost is significantly better than those of most of the ensemble algorithms.

Further work is required to provide self-adaptation of the parameters of the algorithm. We have found that the trade-off between accuracy and performance depends strongly on the δ parameter we use for time-based forgetting.

As a general consideration, we have detected that the evaluation procedure may have an impact on the measured accuracy (such as with SGD). Also datasets are usually tested with a specific value for the chunk size parameter, that is important to the ensemble approach. But this parameter has to be estimated when dealing with unknown data. It is also unclear to what extent datasets in literature contain a significant amount of non-stationary data, as parts of the datasets may be invariant. Further work on categorizing or generating benchmarks that measure different features of NSL algorithms would be of great interest to the field.

ACKNOWLEDGEMENTS

This article has been funded by the Spanish Ministry of Science and Innovation under the project MOVES with grant reference TIN2011-28336, and NSERC-Canada.

REFERENCES

[1] D. P. Helmbold and P. M. Long, "Tracking drifting concepts by minimizing disagreements," *Mach. Learn.*, vol. 14, pp. 27–45, January 1994. [Online]. Available: http://portal.acm.org/citation.cfm?id=189226.189233

[2] L. Bottou and O. Bousquet, "Learning using large datasets," in *Mining Massive DataSets for Security*, ser. NATO ASI Workshop Series. Amsterdam: IOS Press, 2008, to appear. [Online]. Available: http://leon.bottou.org/papers/bottou-bousquet-2008b

[3] C. C. Aggarwal, "A framework for diagnosing changes in evolving data streams," in *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '03. New York, NY, USA: ACM, 2003, pp. 575–586. [Online]. Available: http://doi.acm.org/10.1145/872757.872826

[4] M. Kholghi, H. Hassanzadeh, and M. Keyvanpour, "Classification and evaluation of data mining techniques for data stream requirements," in *Computer Communication Control and Automation (3CA), 2010 International Symposium on*, vol. 1, may 2010, pp. 474 –478.

[5] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '00. New York, NY, USA: ACM, 2000, pp. 71–80. [Online]. Available: http://doi.acm.org/10.1145/347090.347107

[6] D. W. Aha and D. Kibler, "Instance-based learning algorithms," in *Machine Learning*, 1991, pp. 37–66.

[7] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1517–1531, 2011.

[8] W. N. Street and Y. Kim, "A streaming ensemble algorithm (sea) for large-scale classification," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '01. New York, NY, USA: ACM, 2001, pp. 377–382. [Online]. Available: http://doi.acm.org/10.1145/502512.502568

[9] J. a. Gama, R. Sebastião, and P. P. Rodrigues, "Issues in evaluation of stream learning algorithms," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '09. New York, NY, USA: ACM, 2009, pp. 329–338. [Online]. Available: http://doi.acm.org/10.1145/1557019.1557060

[10] D. Martínez-Rego, B. Prez-Snchez, O. Fontenla-Romero, and A. Alonso-Betanzos, "A robust incremental learning method for non-stationary environments," *Neurocomputing*, vol. 74, no. 11, pp. 1800 – 1808, 2011.

[11] G. Carpenter and S. Grossberg, "The art of adaptive pattern recognition by a self-organizing neural network," *Computer*, vol. 21, no. 3, pp. 77 –88, mar 1988.

[12] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD 01. New York, NY, USA: ACM, 2001, pp. 97–106.

[13] J. a. Gama, P. Medas, and R. Rocha, "Forest trees for on-line data," in *Proceedings of the 2004 ACM symposium on Applied computing*, ser. SAC '04. New York, NY, USA: ACM, 2004, pp. 632–636. [Online]. Available: http://doi.acm.org/10.1145/967900.968033

[14] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *In SIAM International Conference on Data Mining*, 2007.

[15] A. Bifet and R. Gavald, "Adaptive learning from evolving data streams," in *Proceedings of the 8th International Symposium on Intelligent Data Analysis: Advances in Intelligent Data Analysis VIII*, ser. IDA 09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 249–260. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-03915-7_22

[16] S. M. Lee and S. J. Roberts, "Sequential dynamic classification using latent variable models," *Comput. J.*, vol. 53, pp. 1415–1429, November 2010. [Online]. Available: http://dx.doi.org/10.1093/comjnl/bxp127

[17] D. R. Lowne, S. J. Roberts, and R. Garnett, "Sequential non-stationary dynamic classification with sparse feedback," *Pattern Recogn.*, vol. 43, pp. 897–905, March 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1660180.1660692

[18] M. Riekert, K. Malan, and A. Engelbrect, "Adaptive genetic programming for dynamic classification problems," in *Evolutionary Computation, 2009. CEC '09. IEEE Congress on*, may 2009, pp. 674 –681.

[19] B. Fritzke, "Fast learning with incremental rbf networks," in *Neural Processing Letters*, 1994, pp. 2–5.

[20] Y. Prudent and A. Ennaji, "An incremental growing neural gas learns topologies," in *Neural Networks, 2005. IJCNN '05. Proceedings. 2005 IEEE International Joint Conference on*, vol. 2, july-4 aug. 2005, pp. 1211 – 1216 vol. 2.

[21] Y. Xu, S. Furao, O. Hasegawa, and J. Zhao, "An online incremental learning vector quantization," in *Proceedings of the 13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, ser. PAKDD '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 1046–1053. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-01307-2_112

[22] L. Kuncheva, "Classifier ensembles for changing environments," in *Multiple Classifier Systems*, ser. Lecture Notes in Computer Science, F. Roli, J. Kittler, and T. Windeatt, Eds. Springer Berlin / Heidelberg, 2004, vol. 3077, pp. 1–15.

[23] M. Muhlbaier and R. Polikar, "Multiple classifiers based incremental learning algorithm for learning in nonstationary environments," in *Machine Learning and Cybernetics, 2007 International Conference on*, vol. 6, aug. 2007, pp. 3618 –3623.

[24] M. Karnick, M. Muhlbaier, and R. Polikar, "Incremental learning in non-stationary environments with concept drift using a multiple classifier based approach," in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, dec. 2008, pp. 1 –4.

[25] M. Muhlbaier, A. Topalis, and R. Polikar, "Learn++.nc: Combining ensemble of classifiers with dynamically weighted consult-and-vote for efficient incremental learning of new classes," *Neural Networks, IEEE Transactions on*, vol. 20, no. 1, pp. 152–168, jan. 2009.

[26] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà, "New ensemble methods for evolving data streams," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '09. New York, NY, USA: ACM, 2009, pp. 139–148. [Online]. Available: http://doi.acm.org/10.1145/1557019.1557041

[27] A. Pocock, P. Yiapanis, J. Singer, M. Luján, and G. Brown, "Online non-stationary boosting," *Lecture Notes in Computer Science*, vol. 5997, pp. 205–214, 2010. [Online]. Available: http://eprints.gla.ac.uk/39591/

[28] J. Kennedy, R. Eberhart, and Y. Shi, *Swarm intelligence*. San Francisco: Morgan Kaufmann Publishers, 2001.

[29] A. P. Dawid, "Statistical theory: The prequential approach," *Journal of the Royal Statistical Society-A*, vol. 147, pp. 278–292, 1984. [Online]. Available: http://www.jstor.org/stable/2981683

[30] M. Harries and U. of New South Wales, "Splice-2 comparative evaluation: Electricity pricing," University of New South Wales, Tech. Rep., 1999.

[31] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," *J. Mach. Learn. Res.*, vol. 8, pp. 2755–2790, December 2007. [Online]. Available: http://dl.acm.org/citation.cfm?id=1314498.1390333

[32] M. Last, Y. Klein, and A. Kandel, "Knowledge discovery in time series databases," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 31, no. 1, pp. 160 –169, feb 2001.

[33] M. Last, "Online classification of nonstationary data streams," *Intell. Data Anal.*, vol. 6, pp. 129–147, April 2002. [Online]. Available: http://dl.acm.org/citation.cfm?id=1293986.1293988

[34] L. Cohen, G. Avrahami-Bakish, M. Last, A. Kandel, and O. Kipersztok, "Real-time data mining of non-stationary data streams from sensor networks," *Information Fusion*, vol. 9, no. 3, pp. 344 – 353, 2008, special Issue on Distributed Sensor Networks. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1566253505000655

[35] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT'2010)*, Y. Lechevallier and G. Saporta, Eds. Paris, France: Springer, August 2010, pp. 177–187. [Online]. Available: http://leon.bottou.org/papers/bottou-2010

[36] N. C. Oza and S. Russell, "Online bagging and boosting," in *In Artificial Intelligence and Statistics 2001*. Morgan Kaufmann, 2001, pp. 105–112.

[37] N. C. Oza, "Online ensemble learning," Ph.D. dissertation, The University of California, Berkeley, CA, Sep 2001.

[38] A. Bifet, G. Holmes, B. Pfahringer, P. Kranen, H. Kremer, T. Jansen, and T. Seidl, "Moa: Massive online analysis, a framework for stream classification and clustering," *Journal of Machine Learning Research - Proceedings Track*, vol. 11, pp. 44–50, 2010.

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