Incremental learning with the minimum description length principle

Pierre Alexandre Murena, Antoine Cornuéjols, Jean-Louis Dessalles

Incremental Learning with the Minimum Description Length Principle

Pierre-Alexandre Murena
Télécom ParisTech
Université Paris-Saclay
46 rue Barrault, 75013 Paris, France
murena@telecom-paristech.fr

Antoine Cornuéjols
UMR MIA-Paris
AgroParisTech, INRA, Université Paris-Saclay
16 rue Claude Bernard, 75005 Paris, France
antoine.cornuejols@agroparistech.fr

Jean-Louis Dessalles
Télécom ParisTech
Université Paris-Saclay
46 rue Barrault, 75013 Paris, France
dessalles@telecom-paristech.fr

***Abstract*—Whereas a large number of machine learning methods focus on offline learning over a single batch of data called *training data set*, the increasing number of automatically generated data leads to the emergence of new issues that offline learning cannot cope with. Incremental learning designates *online* learning of a model from streaming data. In non-stationary environments, the process generating these data may change over time, hence the learned concept becomes invalid. Adaptation to this non-stationary nature, called *concept drift*, is an intensively studied topic and can be reached algorithmically by two opposite approaches: active or passive approaches. We propose a formal framework to deal with concept drift, both in active and passive ways. Our framework is derived from the Minimum Description Length principle and exploits the algorithmic theory of information to quantify the model adaptation. We show that this approach is consistent with state of the art techniques and has a valid probabilistic counterpart. We propose two simple algorithms to use our framework in practice and tested both of them on real and simulated data.**

I. Introduction

The current development of sensors and of the Internet of Things leads to a dramatically increasing number of automatically generated data: this volume was estimated to 2.8 ZB in 2012 by the Digital Universe Study [1]. Many of these data appear in a *stream* fashion and thus cannot be stored in the machine's memory.

This new way of collecting data, as well as the gathering of data in ever-growing social media, induces two major problems that traditional machine learning algorithms cannot cope with. First, older data in streams have to be removed from the system memory and thus cannot be used anymore by the learner. Secondly, the data generating process may vary over time, in particular because of ageing effect [2] or of changes in the environment (e.g. in fraud detection [3], [4]). This non-stationary nature of environments is called *concept drift*.

There exists several ways a concept drift can occur in a stream. If the data distribution changes permanently at a given time step, the drift is called *abrupt*. On the contrary, an *incremental concept drift* will be characterized by a continuous change in the distribution which can possibly end up in a new stationary state. In *recurrent concept drift*, the system alternates between several stationary distributions.

To deal with concept drift, two strategies can be adopted: *active* algorithms and *passive* algorithms. Active algorithms aim at identifying the distribution changes (in particular by using a triggering mechanism) and adapt the model only once a change is detected. On the contrary, passive algorithms adapt the model at each time step without detecting a change.

Many methods have been proposed to deal with concept drifts and state of the art algorithms are known to obtain good performances in streaming environments [5], [6]. However, a formal approach to the problem is less often addressed.

In this paper, we propose a formal approach based on the Minimum Description Length (MDL) Principle. The MDL principle is an inductive principle assuming that the best theory to describe a system is the theory the programming of which has the shortest expression on a universal Turing machine. In particular, we will see that several existing methods are particular cases of our proposed framework. The MDL principle has already been applied to a wide variety of problems, among which model selection [7], analogy reasoning [8] and transfer learning [9].

In section II, we will expose the general framework and the Minimum Description Length Principle. We will show that our approach is robust to all concept drift types we previously exposed. After developing a general theory, we will provide simplifications to deal with problems of different natures. In section III, we will then propose two general algorithms which can be used to deal directly with the MDL principle. In section IV, we provide three different classes of models that can be used in our framework: probabilistic models, prototype-based models (inspired by Learning Vector Quantization and Self Organizing Maps [10]) and models on finite alphabets. Using the prototype-based class of models, we give some experimental results we obtained using the proposed algorithms both for active and passive incremental learning.

II. A framework for Incremental Learning

A. Notations for Incremental Learning

Let $\mathcal{P}$ be a problem space and $\mathcal{S}$ a solution space. At a time step t, the system receives a problem $X_t \in \mathcal{P}$ and aims at predicting the solution $\widehat{Y}_t \in \mathcal{S}$. After predicting, the system may receive the actual solution Y_t to the given problem.

This formalism is consistent with several usual situations. In particular it covers both cases of *instance-incremental* and *batch-incremental learning* [11].

In *instance-incremental learning*, the system receives data one by one. At a step t, the learner receives a point $x \in \mathcal{X}$ where $\mathcal{X}$ is an input space (typically, $\mathcal{X} = \mathbb{R}^d$) and has to predict an output $y \in \mathcal{Y}$ (where the output space $\mathcal{Y}$ can be either continuous in regression, or finite in classification).

In *batch-incremental learning*, the learner receives a batch of data $x_1, \ldots, x_p$ and has to attribute a label $y_1, \ldots, y_p$ to each of the input points.

Using our notation is direct in both cases. In instance-incremental learning, the problem space and the input space are the same. In batch-incremental learning, a problem consists of a batch of instances of the input space $\mathcal{X}$.

In both cases, the problem at time t is denoted by X_t. In instance-incremental learning, $X_t \in \mathcal{X}$ is directly an element of the input space. If $\mathcal{X}$ is a vector space, we denote by X_t^i the i-th coordinate of the vector X_t. More generally, we will use the upper-script index to designate the coordinate of a vector. In batch-incremental learning, X_t is given in form of a list and we will designate by $X_{t,n}$ the n-th element of X_t. In particular, $X_{t,n}$ is an element of the input space $\mathcal{X}$.

B. Minimum Description Length Principle

The idea of our approach is to use the Minimum Description Length Principle as an inductive principle in incremental learning. Firstly introduced by Solomonoff's theory of induction [12], this inductive principle is exposed formally by Rissanen [13]. The MDL principle is defined as follows:

The best theory chosen to describe observed data is the one which minimizes the sum of the description length (in bits) of:

- *the theory description*
- *the data encoded using the theory*

The description lengths are calculated using the notion of Kolmogorov complexity [14]. Given a string $\mathbf{x}$, the Kolmogorov complexity of $\mathbf{x}$, denoted by $K(\mathbf{x})$ is defined as the length (in bits) of the shortest program on a prefix Universal Turing Machine $\mathcal{M}$ (i.e. a Turing machine which produces decodable codes) generating the string $\mathbf{x}$:

$$K_{\mathcal{M}}(\mathbf{x}) = \min_{p \in \mathcal{P}_{\mathcal{M}}} \{l(p); p() = \mathbf{x}\} \tag{1}$$

In equation 1, the term $\mathcal{P}_{\mathcal{M}}$ designates the set of all programs on the Universal Turing Machine $\mathcal{M}$; $l(p)$ is the length of the program p and $p()$ is its output.

A similar definition is proposed for the conditional Kolmogorov complexity of $\mathbf{x}$ given a string $\mathbf{y}$:

$$K_{\mathcal{M}}(\mathbf{x}|\mathbf{y}) = \min_{p \in \mathcal{P}_{\mathcal{M}}} \{l(p); p(\mathbf{y}) = \mathbf{x}\} \tag{2}$$

where $p(\mathbf{y})$ is the output of the program p taking $\mathbf{y}$ as input parameter.

It is important to notice in equations 1 and 2 that the quantity is defined for a given Universal Turing Machine $\mathcal{M}$. The *invariance theorem* guarantees that the complexities for two different machines are equal up to a constant. Based on this theorem, a machine independent definition of Kolmogorov complexity can be defined but will not be used in this article. This choice is motivated by two reasons: first, the exact Kolmogorov complexity is not calculable, as it requires an evaluation over all programs on all Universal Turing Machines. On the other hand, the choice of $\mathcal{M}$ can be used and exploited as an inductive bias: by selecting a specific Turing machine, we will control how the compression is operated. This choice may seem arbitrary at first sight; but it is of a same nature as the choice of a restricted class of hypotheses which is used in Machine Learning for both theoretical and practical purpose. In the following, we will abusively omit to mention the machine in the expression of the complexity.

The ideal MDL can formulated using the notion of Kolmogorov complexity. Given data D, the model M chosen to describe an observation is the model minimizing the objective $K(M) + K(D|M)$.

C. Simplifying Assumptions for Incremental Learning

The following assumptions correspond to a choice for Turing machine $\mathcal{M}$.

To deal with incremental learning, we will consider that each time step t will be associated to a model M_t. This model is used to describe both the problem X_t and the solution Y_t. The problem X_t is described directly with the model M_t. The solution Y_t is described with the help of a decision function β_t (the classifier in a classification problem or regressor in a regression problem). The decision function β_t is described directly with the help of the model M_t.

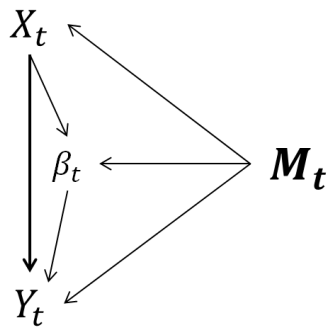

Fig. 1. Simplified framework for the description of problem X_t and solution Y_t using model M_t. On this graphic representation, an arrow from **A** to **B** means that string **B** is described using string **A**.

In order to apply this description (summarized graphically in figure 1) to the mathematical expression of MDL principle, we use the chain rule: if **A** and **B** are two strings, the following equality is verified: $K(A, B) = K(A) + K(B|A) + \mathcal{O}(1)$. Applying this equality to our description, we obtain:

$$\begin{aligned} K(X_t, Y_t|M_t) = K(X_t|M_t) + K(\beta_t|M_t, X_t) \\ + K(Y_t|M_t, X_t, \beta_t) \end{aligned} \tag{3}$$

This value of complexity expressed in equation 3 applies only to one time step. Further hypotheses over model transfer have to be expressed to deal with the whole streaming process.

In the following, we will use the notation $X_{:T}$ to designate the vector $(X_1, \ldots, X_T)$ of the T first values of X. We will use the equivalent notation for the solutions ($Y_{:T}$).

We decompose the global model as a description of all models up to time T. This decomposition is denoted $M_{:T}$. Besides, we suppose that the data (X_t, Y_t) at any time step t are described only by the corresponding model M_t. In particular, they are not described by any previous data. As a consequence, at a time step T, the global description of the data is given by:

$$K(D|M) = K(X_{:T}, Y_{:T}|M_{:T}) = \sum_{t=1}^{T} K(X_t, Y_t|M_t) \quad (4)$$

D. Model Complexity

The term $K(M_{:T}) = K(M_1, \ldots, M_T)$ is of major importance in the sense that it encodes the whole transfer process between time steps and will be responsible for the adaptation to the concept drift.

We will adopt a markovian point of view on the model description. At time step T, we consider that all the models M_t with $t \leq T$ are described using previously defined models. For any $t \leq T$, we define an association function

$$\Delta_t : \{1, \ldots, t-1\} \mapsto \{0, 1\} \quad (5)$$

such that $\Delta_i(j) = 1$ if model i can be described with the help of model j, and $\Delta_i(j) = 0$ otherwise.

Using these association functions, we can express the complexity of the models up to time T as:

$$K(M_1, \ldots, M_T) = \sum_{t=1}^{T} K(M_t|M_{\Delta_t^{-1}(\{1\})}) \quad (6)$$

In this equation, the notation $\Delta_t^{-1}(\{1\})$ designates the set of indices i such that $\Delta_t(i) = 1$.

In practice, the association functions Δ_t can be either fixed by the system or learned online.

The choice of the functions Δ_t is of major importance in theory and in practice, because it offers to the system the possibility to store previously acquired knowledge and thus to memorize states of interest. A constant effort for obtaining this property has been deployed in recent techniques [15]. Several choice scenarios can be considered in our case, which all correspond to state of the art methods (figure 2):

- When $\Delta_i(j) = 0$ for all i, j: all models are *a priori* independent in terms of description. The model is learned completely at each time step.
- When $\Delta_i(j) = 1$ for all i, j: the whole past models are taken into account to describe the present model.
- When $\Delta_i(j) = 1$ only for $j \geq i - h$ with a fixed h: the present model can be described with the last h models, which correspond to a sliding window of fixed size. The fixed sliding window is used in several algorithms like FLORA [16].
- When $\Delta_i(j) = 1$ only for $j \geq i - h$ with a size h estimated by the system: the size of the sliding window is not fixed anymore but heuristically adapted to the current problem. Such techniques are used in ADWIN [17].

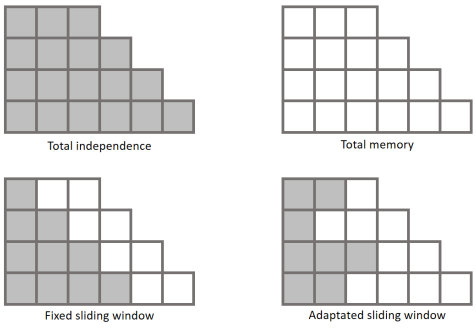

Fig. 2. Possible choices for Δ function. The color of the square at line i and column j indicates the value of $\Delta_i(j)$. If the square is dark, $\Delta_i(j) = 0$; If the square is white, $\Delta_i(j) = 1$

Besides the choice of the association functions Δ_t have consequences over the minimization objective at each time step. Indeed, 1 values of these functions impose to add terms relative to previous observations in the complete Kolmogorov complexity. In practice, this is not necessarily possible, in particular because old data are not stored in memory anymore. Hence, considering old states in the complexity is possible only with simplifying assumptions or with the use of a stochastic gradient descent for the minimization.

E. Optimization problems

On the contrary to *offline learning* which proceeds to model learning in only one step, *incremental learning* has to operate in multiple phases: At each time step, the system has to answer to the new occurring problem X_t and to adapt to the actual solution Y_t.

In a strict use of the MDL principle, it would be possible to optimize the model M when observing the new problem X_t: Indeed the data D can be considered as the delimited concatenation of all previously observed problem and solution couples and of the new observation:

$$D = \langle (X_1, Y_1), \ldots, (X_{t-1}, Y_{t-1}), X_t \rangle \quad (7)$$

Using the previously defined notations and complexity terms, the global complexity of the system at time step T is then defined as:

$$K_T = K(M_{:T}) + \sum_{t=1}^{T-1} K(X_t, Y_t|M_t) \quad (8)$$

The learning problem that the system is facing, from the Minimum Description Length point of view, is the minimization of this global complexity K_T term over the whole history of models $M_{:T}$ up to time T.

This minimization is complex for various reasons. One of them is the interdependency of the M_t which appear in several terms of the sum (because of the Δ_t functions). In particular, all models have to be learned again at each time step. In the perspective of data stream mining, this global update is obviously impossible and algorithmic hypotheses have to be chosen in order to overcome this difficulty.

At time step T, we choose to optimize the complexity over M_{T-1} and M_T only. The past models $M_{:T-2}$ are supposed to be fixed and can be used for the description of the last two models. This assumption is coherent with the incremental paradigm in which data are not stored and computations must be fast. Re-optimizing the whole modeling process would require to store all data $X_{:T}$ and would be highly time-consuming.

Hence, the optimization problem to solve at each time step is the following:

$$\begin{aligned}\underset{M_{T-1},M_T}{\text{minimize}} \quad & K_T - K_{T-1} + C(X_{T-1}|M_{T-1}) \\ & + C(M_{T-1}|M_{\Delta_{T-1}^{-1}}(\{1\})) \end{aligned} \tag{9}$$

F. Probabilistic interpretation

A use of Kolmogorov complexity to measure the quality of a learning is not the most frequent paradigm in Machine Learning: A large range of methods rely on statistics and probability theory.

Our method is not in opposition to those techniques; on the contrary, they can be shown to be limit cases of or complexity-based framework. Basically there exists a strong correlation between Kolmogorov complexity and probabilities. This correlation is justified intuitively by the Shannon-Fano code. Over a finite alphabet, the Shannon-Fano code consists in associating each character to a complexity based on its occurrence frequency with the rule $Pr(X) = 2^{-K(X)}$ (or equivalently $K(X) = -\log P(X)$, where $\log$ designates the logarithm with base 2 as always in this paper).

The precise theoretical conditions to substitute the terms $K(M)$ and $K(X|M)$ by their probabilistic counterparts are discussed in [18].

Focusing on this interpretation, the minimization objective in equation 9 corresponds to Bayes rule and the minimization over the model (hence the probability distribution or the classifier) corresponds to the Bayesian criterion used in *Maximium A Posteriori* and *Maximum Likelihood* Principles.

The very nature of the objective remains unclear. Two approaches may be adopted, which correspond to the notions of *discriminative* and *generative* learning [19]. The purpose of discriminative learning is to establish a separation between classes without estimating the density of the input points. On the contrary, generative learning focuses on the estimation of the generating process of the points.

In the terms of our framework, generative learning will consist in choosing a class of models describing the data: in a probabilistic setting, this class may be the class of joint distributions over the input and output space. Discriminative learning corresponds to a situation where the model M cannot be used to describe data, which means in terms of complexity that $K(X|M) = K(X)$. In generative learning, the model will thus correspond directly to the decision function β.

G. A general principle

As stated above, the proposed framework offers a common description to a large variety of existing methods. We propose to examine how state of the art methods can be expressed in the terms of our framework.

1) Active methods: Active approaches of incremental learning aim to detect change time steps t explicitly. Previous knowledge is used while no change has been detected.

Adaptive sliding window approach (ADWIN) [17] stores all previous data points into a *sliding window*. At each step, the sliding window is split in two sub-windows: the method detects a drift if the difference of the errors for each sub-window is higher than a parameter. ADWIN framework can be interpreted in terms of Kolmogorov complexity as follows. The model M_t at time t corresponds to the concatenation of data memorized in the sliding window and of the corresponding classifier. Considering constant sizes for the description of data and description functions, and assuming that the decision function β is a constant during the whole process, the objective to minimize over β is $\sum_{t=1}^{T} K(Y_t|X_t, \beta)$. If a drift is detected at time step n, the last $T-n$ terms in the sum are changed in take into account the introduction of a new decision function β_1 and of the new data description using β_1. As the complexity term $K(Y_t|X_t, \beta)$ corresponds to a correction term, a link can be established with empirical risk. Denoting $R_{n+1:T}(\beta)$ the empirical risk of binary classifier β on the last $T-n$ data, the detection of a break point obeys the criterion:

$$R_{n+1:T}(\beta) - R_{n+1:T}(\beta_1) > \frac{K(\beta_1)}{T-N} \tag{10}$$

Similar arguments may be given to other methods based on sliding windows. In Probabilistic Adaptive Windowing (PAW) [20], the probabilistic selection of samples to remove from the window corresponds to a non-deterministic way to minimize the global complexity.

All methods based on a statistical test are closely related to our model: they all check a regularity in data, hence a way to compress them. For instance, the validity of the *Sequential Probability Ratio Test* (SPRT) [21] in our context can be established in the same way as done for ADWIN.

2) Passive methods: Passive approaches of incremental learning do not consider abrupt changes but continuously adapt the learned decision functions to new incoming data.

Most passive methods rely on ensemble learning, and in particular on *bagging*. The key point of bagging is that at any time t the system relies on a pool of base learners $\{h_t^i\}_{1\leq i\leq N}$. We consider that this pool corresponds to the model M_t. In order to deal efficiently with concept drift, ensemble methods have to remove outdated experts from the pool. Several strategies are used to select learners to eliminate, including systematic removal of experts with performance lower than a threshold, or elimination of the worst base learner. The elimination of a base learner corresponds to a change in the model, which has a cost in terms of complexity. The higher value of $K(M_t|M_{\Delta_t^{-1}(\{1\})})$ has to be compensated by a lower value of $K(Y_t|M_t, X_t, \beta_t)$, hence a better performance. To express the value of Y_t using M_t and β_t, a method based on a majority vote of all expert learners is used, thus the correction term $K(Y_t|M_t, X_t, \beta_t)$ can depend on all experts.

III. Algorithms

In this section, we will develop methods to solve optimization problem 9 in a context of incremental learning (i.e. with low memory and high speed).

A. Dealing with previous models

We propose to classify algorithms depending on the way they deal with previously acquired models. As mentioned earlier, the dependency on the past is given by the complexity term $K(M_t|M_{:t-1})$ which encodes the description length of model M_t at step t and the previous models $M_{:t-1}$ up to step t. Using the previously defined association functions Δ, the expression has already been simplified into $K(M_t|M_{\Delta_t^{-1}(\{1\})})$.

In order to describe model M_t with the help of the set $M_{\Delta_t^{-1}(\{1\})}$, two strategies may be chosen: either use many models in the set or select one single model.

The first strategy is employed in all ensemble learning methods [22], [23], [24]. Such as in classical machine learning, ensemble learning methods construct the solution to a new problem by considering a weighted sum of the predictions of previous models.

In the second strategy, the key idea is to select the optimal model inside the set of predecessors $M_{\Delta_t^{-1}(\{1\})}$. The selected predecessor is the best model in the sense of MDL principle.

In the perspective of selecting one single predecessor for each model, the total objective of equation 9 can be divided in two parts. The first part corresponds to the description of completed data at time step t once the solution has been given to the system:

$$\begin{aligned}\phi_1(M_{t-1}, M) &= K(M_{t-1}|M) + K(X_{t-1}|M_{t-1}) \\ &+ K(\beta_{t-1}|M_{t-1}, X_{t-1}) + K(Y_{t-1}|M_{t-1}, X_{t-1}, \beta_{t-1})\end{aligned} \tag{11}$$

The second part corresponds to the description of incomplete data at time step t:

$$\phi_2(M_t, M) = K(M_t|M) + K(X_t|M_t) \tag{12}$$

Equation 9 can be reformulated as:

$$\underset{M_t, M_{t-1}, \tilde{M}_1, \tilde{M}_1}{\text{minimize}} \quad \phi_1(M_{t-1}, \tilde{M}_1) + \phi_2(M_t, \tilde{M}_2) \tag{13}$$

In the following, we propose two algorithms to solve problem 13 with a general class of models. The first algorithm calculates model transformations at each step; The second algorithm exploits a dynamic memory of previously calculated models.

B. First Algorithm: Continuous Adaptation

In Continuous Adaptation Incremental Learning, the system infers a new model at each time step t for both model M_{t-1} and M_t. The system has access to all previously learned models $M_{:t}$ and chooses a predecessor among the models $M_{\Delta_{t-1}^{-1}(\{1\})}$ and $M_{\Delta_t^{-1}(\{1\})}$.

In practice, we separate the choice of the predecessor for M_{t-1} and for M_t. Such a separation is motivated by the fact that the description of data at time t depend on model M_{t-1} only by the transfer term in the case where the predecessor of M_t is M_{t-1}.

Consequently, and using the notations introduced previously, we can describe the learning algorithm in three steps:

1) Minimize the objective $\phi_2(M_t, M)$ over predecessor $M \in M_{\Delta_t^{-1}(\{1\})} \backslash \{M_{t-1}\}$ and M_t
2) Minimize the objective $\phi_1(M_{t-1}, M)$ over predecessor $M \in M_{\Delta_{t-1}^{-1}(\{1\})}$ and M_{t-1}
3) Minimize the objective $\phi_1(M_{t-1}, M) + \phi_2(M_t, M_{t-1})$ over predecessor $M \in M_{\Delta_{t-1}^{-1}(\{1\})}$, models M_{t-1} and M_t

In practice, this algorithm can have interesting properties in terms of comprehension of the underlying process. We propose to represent the dependency between two models by a vertex in a graph of models. Such a graphical representation makes the dependencies obvious and would enable a user interpret the decision, for example by detect easily periodic behaviours.

C. Second Algorithm: Memory Based Learning

The second algorithm we propose uses an ordered memory of models. These models are used as a simplification in the minimization.

At time step t, the system has access to a memory of m_t models denoted $M_0^m, \ldots, M_{m_t}^m$. The stored models are used as an approximation for the description models: Each model M_t is associated to a model in memory. On contrary to the previous algorithm, a perfect adaptation to the data is not required at any step, which implies that the algorithm is based on approximations and is less precise than the first algorithm. However, it is more convenient for high speed streaming data.

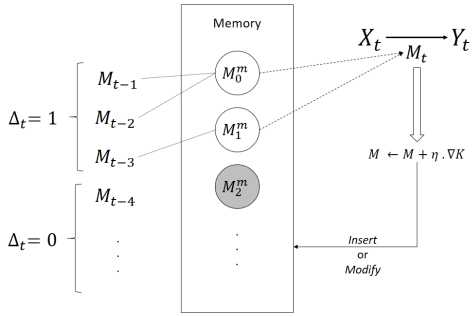

Fig. 3. General schema of Memory based algorithm

The algorithm relies on several approximations. First, a model is chosen only among the previously learned models. Once a model is chosen in the list, it can be updated using the new data. If the complexity difference between the base model and the updated version is high, the updated model may either replace the base model or be inserted as a new memorized model (figure 3). Another approximation consists in using the difference of position in ordered memory as complexity for model transfer:

$$K(M_k^m|M_l^m) = \log(1 + |l - k|) \tag{14}$$

This algorithm is more efficient than the first proposed algorithm in terms of computation time, in the sense that it avoids optimizing the model at each step. The model choice is made only by comparing the values of the complexity with different accessible models; the optimization of the model will be done only in a case where no model fits the observed data.

Hence, this algorithm corresponds to a process of *active learning*: the change of model has to be detected before the system actually calculates the new model.

IV. Applications

A. Symbolic incremental learning

A first application of our model concerns non-continuous data. The use of MDL principle has been introduced naturally to analogy reasoning [8]. Analogy reasoning can be defined as a form of reasoning in which one entity is inferred to be similar to another entity in a certain respect, on the basis of the known similarity between the entities in other respects [25]. In particular, proportional analogy concerns situations of the form "A is to B as C is to D" (which will be denoted $A : B :: C : D$).

An interesting example of proportional analogies have been suggested by D. Hofstadter [26]. In these analogies, the terms are strings made up of alphabet letters. In this micro-world, it is possible to define a model as a generic construction rule for both words and the associated transformation β.

The framework we proposed for incremental learning can be applied directly to sequences of proportional analogies. We define a sequence of proportional analogy as a sequence of the form $X_1 : Y_1 :: X_2 : Y_2 :: \ldots :: X_T : Y_T$.

Some empirical methods have been proposed to solve one single problem of proportional analogy (for example the Copycat project [27]) but the issue of sequences of analogies has never been addressed, whereas it is of major importance in human cognition. Our framework offers a direct way to deal with this issue.

B. Probabilistic Models

Probabilistic models form another class of models which can be used in the Minimum Description Length Principle and especially in our approach of incremental learning. We define a *probabilistic model* as a joint distribution $\mathbb{P}_{\mathcal{X}\times\mathcal{Y}}$ over the input space $\mathcal{X}$ and the output space $\mathcal{Y}$.

Using the probabilistic model $M = \mathbb{P}_{\mathcal{X}\times\mathcal{Y}}$ to describe data, we obtain:

$$K(X|M) = \sum_{i=1}^{n} -\log \mathbb{P}_{\mathcal{X}}[X_i] \tag{15}$$

The classifier β is obtained from M as follows:

$$\beta(x) = \arg\max_{y\in\mathcal{Y}} \mathbb{P}_{\mathcal{Y}|\mathcal{X}}[Y = y|X = x] \tag{16}$$

and thus $K(\beta|X, M) = 0$.

If the models M_t and M_u correspond to the respective distributions $\mathbb{P}_t$ and $\mathbb{P}_u$, then the model transfer term $K(M_t|M_u)$ can be interpreted as the cross-entropy H betwen $\mathbb{P}_u$ and $\mathbb{P}_t$:

$$K(M_t|M_u) = H(\mathbb{P}_u, \mathbb{P}_t) \tag{17}$$

The properties of such models in relation with information theory opens new theoretical perspectives regarding incremental learning.

C. Prototype-based models

In order to test our framework, we have selected a simple visual class of models. Our model is inspired by the Nearest Prototype Classifiers such as Learning Vector Quantification (LVQ) [10]. Such algorithms rely on a set of virtual points called *prototypes* used to approximate the classification locally: the classifier maps a point in the input space to the class of its closest prototype.

Consider a prototype set P of size K. A prototype $P_k \in P$ is a tuple $P_k = (p_k, c_k)$ where $p_k \in \mathcal{X}$ and $c_k \in \mathcal{Y}$. Using the prototype set P, a vector $x \in \mathcal{X}$ is linked to the closest prototype in P and its complete description includes the specification of the index of the prototype and the encoding of the relative position between the two vectors. With this modeling, the complexity of the data point x is:

$$K(x|P) = \min_{k=1,\ldots,K} K(x - p_k) \tag{18}$$

We choose to work on a basic Universal Turing Machine which does not compress over the structure of a set of data: A set of n vectors is encoded by the individual description of each of the vectors. Besides, a vector is given by the description of all his coordinates one by one. Hence, if X is a design matrix, then:

$$K(X) = \sum_{i=1}^{n} K(X_i) = \sum_{i=1}^{n}\sum_{j=1}^{d} K(X_i^d) \tag{19}$$

Given a prototype model P, a classifier β is built by selecting the class of the closest prototype. The method used to build β simply enumerates the prototypes, hence $K(\beta|M, X) = 0$.

Given a classifier β and input data $X \in \mathcal{X}^n$, the complexity $K(Y|P, X, \beta)$ corresponds to a correction term between the predictions $\widehat{Y} = \beta(X)$ and the actual outputs Y. The correction method enumerates a list of correction terms, displaying both the identifier of the point (of description length $\log n$) and the actual class (of description length $\log |\mathcal{Y}|$). Hence:

$$K(Y|P, X, \beta) = (\log n + \log |\mathcal{Y}|) \sum_{i=1}^{n} \mathbb{I}(Y_i \neq \beta(X_i)) \tag{20}$$

where $\mathbb{I}$ designates the identity function the value of which is 1 if its argument is true and 0 otherwise.

In order to define the complexity $K(M_t|M_{\Delta_t^{-1}(\{1\})})$, we propose the following process: Each model M_t is described as a transformation $\Phi(M)$ of a single model $M \in \Delta_t^{-1}(\{1\})$. Thus, the transfer algorithm is divided in two parts:

1) For all models $M \in \Delta_t^{-1}(\{1\})$, calculate the optimal transformation Φ
2) Select the past model M and the transformation Φ with lower density.

The first step is obviously the longest and its time complexity depends on the number of previous models considered by

the system (i.e. the cardinality of $\Delta_t^{-1}(\{1\})$). In case of a first-order markovian process, only the model M_{t-1} is considered by the algorithm.

In our prototype-based framework, we choose basic transformations for the models. Three transformations are considered:

1) *Prototype addition*: a prototype is added to the model (for example when a new class appears)
2) *Prototype deletion*: a prototype is removed because it has become useless.
3) *Prototype shifting*: the prototype position is changed.

The methods suggested for both prototype addition and removal are highly similar to the methods proposed by [28] for Online Learning Vector Quantization. Our method just like theirs is based on a score for both operations, however the score we propose is more direct as it originates directly from the theory itself.

We consider that a prototype model P (of cardinal K) can be extended into a prototype model P^+ of cardinal $K+1$ if adding one extra prototype to the model makes the global complexity decrease. Besides, adding a prototype is compulsory when a new class appears. When adding a new prototype, an additional term K_{add} has to be added to the complexity $K(M_t|M_{:t})$): this term corresponds to the complexity of adding a prototype p_{K+1} to P:

$$K_{add} = K(p_{K+1}) + K\log(K+1) \tag{21}$$

In a similar way, a prototype model P of cardinal K can be reduced into a prototype model P^- of cardinality $K-1$ if removing one particular prototype from the model makes the global complexity decrease. This is the case in particular when no input point is linked to a prototype in a new configuration for a batch incremental learning problem. Removing prototype p_i from P requires to add an extra complexity term:

$$K_{rem} = K(i) + K\log(K-1) \tag{22}$$

The term $K(i)$ can be chosen either independent of i or not. If $K(i)$ is constant, the cost to remove any prototype is the same. On the contrary if $K(i) = \log(1+i)$, the cost to remove a recently added prototype is higher than the cost to remove an older prototype.

As explained before, we choose to apply a global transformation function Φ to the whole set of prototypes in order to describe the prototype shifting. In our proof of concept, we focus on linear functions of the form $\Phi(p) = p+\delta+\delta(p)$ where δ is a global translation vector and $\delta(p)$ a local translation vector. The translation vector δ applies to all prototypes in the model. The vector $\delta(p)$ depends on the prototype p and is a correction term. The complexity of the shifting process is:

$$K_{shift} = K(\Delta) + \sum_{k=1}^{K} C(\delta(p)) \tag{23}$$

The algorithm used to infer the prototype-based models at time step T is a variant of the Expectation-Maximization (EM) algorithm [29]. EM algorithm alternates two steps: the Expectation step (E step) and the Maximization step (M step).

In the E step, data points are associated to their closest prototype, both in data for time $T-1$ and for time T. Using this fixed point-prototype association, the position of the prototypes is modified by minimizing the global complexity. This new position is then supposed to be fixed.

In the M step, the classes of the prototypes are changed if necessary using their new positions. The decision to insert or remove prototypes from the considered model is also made in the M step.

D. Experimental results

In order to illustrate the pertinence of our framework in practice, we have tested its performances on classical data sets with the naive prototype-based model. We considered three datasets: SEA [30] (50000 instances, 3 attributes, artificial), Weather [31] (18159 instances, 8 attributes, real) and Electricity Market [32] (45312 instances, 3 attributes, real).

For all datasets, we tested the instance-incremental version of our algorithm (by streaming directly over the data) and the batch-incremental version (by grouping successive data into a same batch). The experiments were all done with a fix sliding window size: $|\Delta_t^{-1}(\{1\})| = 3$. We tested the two proposed algorithms.

Table I presents the performances for different size S of batches. When $S = 1$, the situation corresponds to a problem of instance-incremental learning. Otherwise, the situation corresponds to batch-incremental learning.

As the purpose of this paper is not to establish a competitive performance for the suggested algorithms with the prototype-based model, we do not propose any comparison to state of the art algorithms. The key idea is that the obtained results are not necessarily better but similar to existing methods. Developing more accurate algorithms will be an improvement perspective to this paper.

In practice, the calculation time with the passive approach makes impossible to use this algorithm directly for a real-time process: the optimization algorithms take too much time even in the case of instance-incremental and makes the system unable to deal with a real data stream.

TABLE I
ERROR RATE FOR SEVERAL BATCH SIZES

		$S=1$	$S=5$	$S=10$	$S=20$
SEA	Active	0.41	0.41	0.35	0.39
	Passive	0.32	0.37	0.37	0.34
Weather	Active	0.37	0.38	0.37	0.37
	Passive	0.32	0.28	0.36	0.36
Electricity	Active	0.33	0.31	0.36	0.35
	Passive	0.29	0.31	0.31	0.28

We notice that in general, the active algorithm has lower accuracy than passive algorithm. This is coherent with the fact that this algorithm relies on strong simplifying hypotheses.

The obtained results are not competitive with state of the art algorithms, which was expected: Our method is not

specifically designed to perform well and the class of models is very basic. We tested a very direct application of the equations presented above regardless of time complexity nor performance of the method. An effort has to be made in this direction in future works. The results are good enough to validate our framework yet.

V. Conclusion

In this paper, we have presented a new modeling the problem of incremental learning. This modeling offers a common description for several existing algorithms and proposes a unified approach of passive and active incremental learning by the mean of Kolmogorov complexity. In particular, it has been shown that our modeling is consistent with the sliding window methods, ensemble learning and Markovian models. We have proposed two elementary algorithms based on the chosen inductive principle and we have tested our framework with a simple class of models. By the mean of this tests, we have shown its efficiency to deal both with batch incremental and instance incremental learning in a context of concept drift.

Our work opens new perspectives in terms of theoretical approach of incremental learning. Besides, the formalism we developed can address a similar yet different problem: education. Instead of adapting models from step to step, education aims at finding the right steps to transfer a source concept into a target concept. Finally, the link with information geometry cannot be neglected: using probabilistic models, we have enlightened the use of well-known quantities to infer a discrete trajectory inside a manifold of probability distributions.

Acknowledgment

This research is supported by the programme Futur et Ruptures (Institut Mines-Telecom).

References

[1] J. Gantz and D. Reinsel, "The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east," 2012.

[2] Z. Akhtar, A. Ahmed, C. E. Erdem, and G. L. Foresti, *Adaptive Facial Recognition Under Ageing Effect*, pp. 97–117. Cham: Springer International Publishing, 2015.

[3] A. D. Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi, "Credit card fraud detection and concept-drift adaptation with delayed supervised information," in *2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, July 2015.

[4] D. Malekian and M. R. Hashemi, "An adaptive profile based fraud detection framework for handling concept drift," in *Information Security and Cryptology (ISCISC), 2013 10th International ISC Conference on*, pp. 1–6, Aug 2013.

[5] J. a. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, pp. 44:1–44:37, Mar. 2014.

[6] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in nonstationary environments: A survey," *IEEE Comp. Int. Mag.*, vol. 10, no. 4, pp. 12–25, 2015.

[7] B. Y. Mark H. Hansen, "Model selection and the principle of minimum description length," *Journal of the American Statistical Association*, vol. 96, no. 454, pp. 746–774, 2001.

[8] A. Cornuéjols and J. Ales-Bianchetti, "Analogy and induction : which (missing) link ?," in *Workshop "Advances in Analogy Research : Integration of Theory and Data from Cognitive, Computational and Neural Sciences"*, 1998.

[9] P. A. Murena and A. Cornuéjols, "Minimum description length principle applied to structure adaptation for classification under concept drift," in *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 2842–2849, July 2016.

[10] T. Kohonen, M. R. Schroeder, and T. S. Huang, eds., *Self-Organizing Maps*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 3rd ed., 2001.

[11] J. Read, A. Bifet, B. Pfahringer, and G. Holmes, "Batch-incremental versus instance-incremental learning in dynamic and evolving data," in *Proceedings of the 11th International Conference on Advances in Intelligent Data Analysis*, IDA'12, (Berlin, Heidelberg), pp. 313–323, Springer-Verlag, 2012.

[12] R. J. Solomonoff, "A Formal Theory of Inductive Inference: Parts 1 & 2," *Inform. Control*, vol. 7, 1964.

[13] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, pp. 465–471, 1978.

[14] M. Li and P. M. Vitanyi, *An Introduction to Kolmogorov Complexity and Its Applications*. Springer Publishing Company, Incorporated, 3 ed., 2008.

[15] M. J. Hosseini, Z. Ahmadi, and H. Beigy, "Using a classifier pool in accuracy based tracking of recurring concepts in data stream classification," *Evolving Systems*, vol. 4, no. 1, pp. 43–60, 2013.

[16] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Machine Learning*, vol. 23, no. 1, pp. 69–101, 1996.

[17] A. Bifet and R. Gavaldà, "Mining adaptively frequent closed unlabeled rooted trees in data streams," in *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 34–42, ACM, 2008.

[18] P. M. B. Vitanyi and M. Li, "Minimum description length induction, bayesianism, and kolmogorov complexity," *IEEE Transactions on Information Theory*, vol. 46, pp. 446–464, Mar 2000.

[19] A. Y. Ng and M. I. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," 2001.

[20] A. Bifet, B. Pfahringer, J. Read, and G. Holmes, "Efficient data stream classification via probabilistic adaptive windows," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, (New York, NY, USA), pp. 801–806, ACM, 2013.

[21] A. Wald, *Sequential Analysis*. John Wiley and Sons, 1st ed., 1947.

[22] 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*, KDD '01, (New York, NY, USA), pp. 377–382, ACM, 2001.

[23] R. Elwell and R. Polikar, *Incremental Learning of Variable Rate Concept Drift*, pp. 142–151. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[24] D. Brzezinski and J. Stefanowski, "Combining block-based and online methods in learning ensembles from concept drifting data streams," *Inf. Sci.*, vol. 265, pp. 50–67, May 2014.

[25] B. Indurkhya, "Modes of analogy," in *Proceedings of the International Workshop on Analogical and Inductive Inference*, AII '89, (London, UK, UK), pp. 217–230, Springer-Verlag, 1989.

[26] D. R. Hofstadter, *Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought*. New York, NY, USA: Basic Books, Inc., 1996.

[27] M. Mitchell, *Copycat: A Computer Model of High-level Perception and Conceptual Slippage in Analogy-making*. PhD thesis, Ann Arbor, MI, USA, 1991. UMI Order No. GAX91-16256.

[28] M. Grbovic and S. Vucetic, "Learning vector quantization with adaptive prototype addition and removal," in *2009 International Joint Conference on Neural Networks*, pp. 994–1001, June 2009.

[29] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, vol. 39, no. 1, pp. 1–38, 1977.

[30] 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*, KDD '01, (New York, NY, USA), pp. 377–382, ACM, 2001.

[31] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Transactions on Neural Networks*, vol. 22, pp. 1517–1531, Oct 2011.

[32] M. Harries, U. N. cse tr, and N. S. Wales, "Splice-2 comparative evaluation: Electricity pricing," tech. rep., 1999.