Accepted to appear in Neural Networks - Elsevier

Active Learning for Classifying Data Streams with Unknown Number of Classes

Saad Mohamad[a,b,*], Moamar Sayed-Mouchaweh[b], Abdelhamid Bouchachia[a]

[a]*Department of Computing, Bournemouth University, Poole, UK*
[b]*Department of Informatics and Automatics, Ecole des Mines, Douai, France*

Abstract

The classification of data streams is an interesting but also a challenging problem. A data stream may grow infinitely making it impractical for storage prior to processing and classification. Due to its dynamic nature, the underlying distribution of the data stream may change over time resulting in the so-called *concept drift* or the possible emergence and fading of classes, known as *concept evolution*. In addition, acquiring labels of data samples in a stream is admittedly expensive if not infeasible at all. In this paper, we propose a novel stream-based active learning algorithm (SAL) which is capable of coping with both *concept drift* and *concept evolution* by adapting the classification model to the dynamic changes in the stream. SAL is the first AL algorithm in the literature to explicitly take account of these concepts. Moreover, using SAL, only labels of samples that are expected to reduce the expected future error are queried. This process is done while tackling the problem of *sampling bias* so that samples that induce the change (i.e., drifting samples or samples coming from new classes) are queried. To efficiently implement SAL, the paper proposes the application of non-parametric Bayesian models allowing to cope with the lack of prior knowledge about the data stream. In particular, Dirichlet mixture models and the stick breaking process are adopted and adapted to meet the requirements

*Saad Mohamad is the corresponding author
Email addresses: `smohamad@bournemouth.ac.uk` (Saad Mohamad), `moamar.sayed-mouchaweh@imt-lille-douai.fr` (Moamar Sayed-Mouchaweh), `abouchachia@bournemouth.ac.uk` (Abdelhamid Bouchachia)

of online learning. The empirical results obtained on real-world benchmarks demonstrate the superiority of SAL in terms of classification performance over the state-of-the-art methods using average and average class accuracy.

Keywords: Data Streams, Active Learning, Concept Drift, Concept Evolution, Bayesian Online Learning.

---

1. Introduction

Classification has been the focus of a large body of research due to its key relevance to numerous real-world applications. A classifier is trained by learning a mapping function between input and pre-defined classes. In an offline setting, the training of a classifier assumes that the training data is available prior to the training phase. Once the training is exhausted, the classifier is deployed and, cannot be trained any further even if performs poorly. This can happen if the training data used does not exhibit the true characteristics of the underlying distribution. Moreover, for many applications, data arrives over time as a stream and therefore the offline assumptions cannot hold. Data streams classification presents many challenges because of the continuous and evolving nature of streams, in addition to the problem of labelling such large data. Data streams are assumed to be unbounded in size, which makes it infeasible to store all the data to train the proposed model. Hence, online learning algorithms is more appropriate [1, 2, 3, 4, 5].

Data streams may evolve over time so that the mapping between the input space and the output space (classes) changes, leading to *concept drift* [6, 7, 8]. Thus, to deal with data streams classification, the classifier must self-adapt online over time [9, 1, 10, 11].

The evolving nature of data streams poses another challenge which is rarely addressed in the literature and known as *concept evolution.* This occurs when new classes emerge or existing classes vanish. The classifier must be able to identify the new classes and incorporate them into the decision model [12, 13, 14, 15]. Emergence of new classes has been studied in the context of novelty detection, where the task is to identify the non-conforming instances. This is seen as *one-class classification*, in which a very large number of data samples describing *normal* condition is available while the data samples describing the *abnormalities* are rare [16, 17]. In contrast, *Concept evolution* involves the emergence of different normal and abnormal classes.

Given the large volume and high velocity of data streams, it is impractical to acquire the label of each instance. Active learning (AL) is a promising way to efficiently building up training sets with minimal supervision. AL queries particular instances to train the classifier using as few labelled data instances as possible. AL for data streams is more challenging because both *concept drift* and *concept evolution* can occur at any time and anywhere in the feature space. Another challenge associated with AL, in general, is the *sampling bias* [18] where the sampled training set does not reflect on the underlying data distribution. Basically, AL seeks to query samples that, if labelled, significantly improve the learning. AL becomes increasingly confident about its sampling assessment. That confidence could lead, however, to negligence of valuable samples that reflect on the true data distribution, creating a bias towards a certain set of instances, which could become harmful.

AL stands as a very interesting opportunity to handle *concept drift* and *concept evolution* by querying the data samples representative of this change (i.e., its characteristics). In contrast to standard *concept drift* and *concept evolution* handling techniques, where only automatic detection mechanisms are applied, in AL an access to the oracle that provides the ground truth (true labels of data) is granted. In this paper, we propose an AL methodology, called Stream Active Learning (SAL) that not only works under all the aforementioned challenges but also handles *concept drift* and *concept evolution* by querying the data samples representative of them (i.e., their characteristics). In contrast to most of the existing AL approaches which adopt heuristic AL criteria, SAL aims at directly reducing the expected future error [19] in a unified and systematic way. Similar AL approaches are proposed in [20, 19, 21], however, they work in offline setting and do not take into account the challenges associated with data streams. SAL queries the samples which incur high reduction in the expected future error, while using both labelled and unlabelled data.

In our previous work [22], we proposed a bi-criteria stream-based AL approach (BAL) that seeks to select data samples which are expected to reduce the future expected error. Because closed form calculation of the expected future error is intractable, BAL uses approximations by combining online classification and online clustering. The classification model estimates the conditional distribution of the labels given the data, while the clustering model estimates the marginal distribution of the data. BAL only considers binary classification and does not deal with *concept evolution*.

SAL adopts the same concept as in [22], but with some differences. While BAL has used existing classification and clustering algorithms, SAL uses a unified non-parametric Baysian model that allows multi-class classification without the need to fix the number of classes in advance as in BAL. Furthermore, this model uses both labelled and unlabelled data for estimating the marginal and conditional distributions. BAL does not take advantage of the unlabelled data when estimating the conditional distribution.

The proposed model is a Dirichlet process mixture model [23] with a stick breaking prior [24] attached to each mixture component. This prior is applied over the classes of the data in the mixture components. Dirichlet process mixture model is based on the most common non-parametric prior, the Dirichlet Process. This prior allows the number of the components forming the mixture to grow, if necessary, as more data is seen. Such a characteristic is useful in the case of data streams as not much prior knowledge is available. The proposed model can approximate both the conditional and marginal distributions. In contrast to BAL, SAL allows multi-class classification with dynamic number of classes and hence it is capable of dealing with *concept evolution*. The application of stick-breaking prior over the classes allows the potential growth of the number of classes. We employ a particle filter method [25, 26] to perform online inference. Note that to ensure enough flexibility of SAL, we explicitly distinguish between the learning engine and the selection engine. The learning engine uses a supervised online learning algorithm to train a classifier on the existing labelled data, while the selection engine uses an AL algorithm to select influential samples from the data stream for labelling. Here, the selection engine consists of SAL which includes the AL approach and the unified estimation model.

As in [22, 27], SAL handles *sampling bias* problem caused by AL using importance weighted empirical risk [28]. Such problem is more severe in online setting as the underlying classifier used by AL has to adapt. On the other hand, the adaptation can depend on the queried data. Hence, if drift occurs for samples which the model is confident about their labels, they will not be queried and the model will not be adapted. SAL handles *concept drift* by querying the data samples representative of this drift (i.e., its characteristics). Similarly, it handles *concept evolution* by querying the data samples coming from a new class. To this end, we use the *importance weighting* principle to weight labelled data samples that drive a change in order to increase the importance of their regions in the feature space. Thus, by mitigating the *sampling bias* problem, *concept drift* and *concept evolution* will be efficiently handled. The *importance weighting* principle has also been theoretically proven to correct sampling bias [28]. Similar technique was used in [22, 27], but they are limited to binary classification. To the best of our knowledge, SAL is the first AL algorithm in the literature that is change aware and considers *concept drift* and *concept evolution* together. To sum up, our contribution to the state-of-the-art, SAL is the first approach satisfying the following needs together:

1. directly reducing the expected future error online while taking the data streams challenges (*infinite length*, *concept drift* and *concept evolution*) into account.
2. mitigating the *sampling bias* problem which implicitly allows SAL to be aware of *concept drift* and *concept evolution*.

The rest of this paper is organized as follows. Section 2 presents the related work. Section 4.1 describes SAL including the formulation of the objective function that it is intended to be optimised as well as the proposed AL approach.

Section 4.2 provides the details of the proposed model to estimate the objective function described in Sect. 4.2. Section 5 discusses the experimental results for a number of well-known real-world datasets. Finally, Sec. 6 concludes the paper and presents the future work.

2. Related work

Online AL from data streams in presence of drift has been investigated using batch-based learning, where the stream is split into batches [31, 32]. Batch-based methods often assume that the data is stationary within each batch, so that pool-based AL strategies (PSS) can be applied. According to PSS, the selection of instances is made by exhaustively searching in a large collection of unlabelled data gathered in a pool. PSS evaluates and ranks the entire collection before selecting the best query. Authors in [33] use a sliding window approach which discards the oldest instances instead of explicitly detecting the changes. The method proposed in [34] works completely online by evaluating data instances on-the-fly allowing faster adaptation to change. However, instead of explicitly handling the problem of *sampling bias*, it combines naive randomization with an *uncertainty criterion* [35]. By doing so, the budget is wasted on some random queries. On the contrary, in [22, 27], *Sampling bias* is studied using *importance weighting* principle which has been theoretically proven to be effective [28]. However, the methods in [27, 22] are restricted to binary classification and the data distribution effect is ignored in [27].

Online AL approaches that address the data stream challenges, such as *infinite length*, *concept drift* and *concept evolution* are the least investigated among all AL approaches. Authors in [36] handle the problem of *concept evolution* by defining a nonparametric Bayesian prior on the classes using Pitman-Yor Processes [37]. However, they use *Query-by-committee (QBC)* [38] which aims at reducing the version-space instead of directly optimizing the error rate. QBC often fails by spending effort eliminating areas of parameter space that have no effect on the error. It does not consider the data distribution effect and ignores the problem of *sampling bias*.

Authors in [39] apply a hybrid batch-incremental learning approach, where the data is divided into fixed-size chunks and an offline classifier is trained from each chunk. An ensemble of M classifiers is maintained to classify the unlabelled data. To detect the novel classes, a clustering technique is used in order to isolate odd data instances. If the isolated samples are enough and sufficiently close to each other (coherent), they get queried. Otherwise, the algorithm considers them as noise. The algorithm also uses the uncertainty sampling within the current chunk to query the label of instances for which it is most uncertain. Similar principal was adopted in [40, 41], where authors adopt batch-based approach to learn from data streams while considering *concept drift* and *concept evolution*. SAL adopts a different approach based on probabilistic non-parametric Bayesian modelling and addresses the *sampling bias* problem. SAL is also a stream-based where no need to store the data instances in different batches. The application of non-parametric Bayesian models allows to cope with the lack of prior knowledge about the data stream.

3. Preliminary

In this section, we present some preliminary materials that will be needed for developing SAL. Firstly, we introduce the offline AL approach from which our work is inspired. This approach adopts the same view for AL as ours. That is, it aims at labelling data samples that minimizing the expected future error. we also define some notations and propose a technique to handle the *sampling bias* problem.

Secondly, we give a brief background on Dirichlet process (DP) which is the core of our model. DP is used as a non-parametric prior in Dirichlet process mixture model (DPMM) which, in contrast to parametric model, allows the number of components to grow, if necessary, to accommodate the data.

3.1. ***Offline Active Learning Approach***

Many active learning approaches seek to minimize an approximation of the expected error (Eq. (1)) [20, 19, 21]. SAL follows the same methodology but with more challenging setting where the data comes as a stream,

$$R = \int_{\boldsymbol{x}} L\big(\hat{p}(\hat{y}|\boldsymbol{x}), p(y|\boldsymbol{x})\big)p(\boldsymbol{x})d\boldsymbol{x} \tag{1}$$

where $(\boldsymbol{x}, y)$ is a pair of random variables, such that $\boldsymbol{x}$ represents the data instance (observation) and y is its class label. $\hat{y}$ represents the predicted label of $\boldsymbol{x}$, $p(y|\boldsymbol{x}))$ and $p(\boldsymbol{x})$ are the true conditional and marginal distributions respectively. $\hat{p}(\hat{y}|\boldsymbol{x})$ is the learner's conditional distribution used to classify the data. The learner receives observations drawn from $p(\boldsymbol{x})$ with latent labels y unless they are queried by the AL algorithm. We denote the labelled observations up to time t as X_{L_t} and their labels as Y_{L_t}. The unlabelled observations up to time t are denoted as X_{U_t}. We also use X_t to denote $\{X_{U_t}, X_{L_t}\}$, D_{L_t} to denote $\{X_{L_t}, Y_{L_t}\}$ and D_t to denote $\{X_t, Y_{L_t}\}$. We separate the learner algorithm or the hypothesis class from the AL algorithm so that we can simply plug in any learner to test the AL algorithm. Let $p(\hat{y}|\boldsymbol{x}, \boldsymbol{\phi})$ refer to the learner's conditional distribution $\hat{p}(\hat{y}|\boldsymbol{x})$, where $\boldsymbol{\phi}$ is the parameter vector that governs the learner's distribution.

In the following, we discuss the offline AL approaches used to minimize an approximation of Eq. (1) since a closed form solution is not available. Then, we present our online AL approach. Authors in [20] approximate the expected error using the empirical risk over the unlabelled data:

$$\hat{R}_{X_U}(\boldsymbol{\phi}_{D_L}) = \frac{1}{|X_U|}\sum_{\boldsymbol{x}\in X_U} L\big(p(\hat{y}|\boldsymbol{x}, \boldsymbol{\phi}_{D_L}), p(y|\boldsymbol{x})\big). \tag{2}$$

We refer to the classifier parameters after being trained on D_L as $\boldsymbol{\phi}_{D_L}$. Different types of loss functions can be adopted according to the classification problem. Active learning seeks to optimize Eq.(2) by asking for the labels of the samples that, once incorporated in the training set, the empirical risk drops the most.

Ideally, the selection should depend on how many queries can be made. However, the solution of such optimization problem is NP hard, since the number of trials is combinatorial. Hence, most commonly used AL strategies greedily select one example at a time [20, 27, 21].

$$\tilde{\boldsymbol{x}} = \arg\min_{\boldsymbol{x} \in X_U} \hat{R}_{X_{U-(\boldsymbol{x})}}(\boldsymbol{\phi}_{D_{L+(\boldsymbol{x},y)}}). \tag{3}$$

The empirical risk over the labelled and unlabelled samples is considered in [21]:

$$\hat{R}_D(\boldsymbol{\phi}_{D_L}) = \frac{1}{|D|} \sum_{\boldsymbol{x} \in D} L\big(p(\hat{y}|\boldsymbol{x}, \boldsymbol{\phi}_{D_L}), p(y|\boldsymbol{x})\big). \tag{4}$$

The risk incurred when training the learner is the one related to the labelled data:

$$\hat{R}_{D_L}(\boldsymbol{\phi}_{D_L}) = \frac{1}{|D_L|} \sum_{\boldsymbol{x} \in D_L} L\big(p(\hat{y}|\boldsymbol{x}, \boldsymbol{\phi}_{D_L}), p(y|\boldsymbol{x})\big). \tag{5}$$

In active learning, a subset of unlabelled samples is selected for labelling. Let q be a random variable distributed according to a Bernoulli distribution with parameter a, $q \sim Ber(a)$. That is, an instance $\boldsymbol{x}$ is queried if $q = 1$. The data instances used to train the model are sampled from a distribution induced by the AL queries instead of the data underlying distribution. That is, the distribution of the queried data $p(\boldsymbol{x}|q = 1)$ is different from the original one $p(\boldsymbol{x})$. Hence, Equation (5) is a biased estimator of (1) and the learned classifier may be less accurate than when learned without using AL (*Sampling bias*). Similar to [22, 27], we use the *importance weighting* technique [28] in order to come up with an unbiased estimator. Thus, Eq. (5) can be written as follows:

$$\hat{R}'_{D_L}(\boldsymbol{\phi}_{D_L}) = \frac{1}{|D_L|} \sum_{\boldsymbol{x} \in D_L} \frac{1}{p(q = 1|\boldsymbol{x})} L\big(p(\hat{y}|\boldsymbol{x}, \boldsymbol{\phi}_{D_L}), p(y|\boldsymbol{x})\big). \tag{6}$$

Hence, the unbiased estimation above can be shown as [42]:

$$E_{\boldsymbol{x} \sim p(\boldsymbol{x}|q=1)}\big[\hat{R}'_{D_L}(\boldsymbol{\phi}_{D_L})\big] = E_{x \sim p(\boldsymbol{x})}\big[\hat{R}_{D_L}(\boldsymbol{\phi}_{D_L})\big]. \tag{7}$$

3.2. Dirichlet process

DP is one of the most popular prior used in the Bayesian non-parametric model. Its first use by the machine learning community dates back to [45, 46]. In general, a stochastic process is a probability distribution over a space of paths which describe the evolution of some random values over time. DP is a family of stochastic processes whose paths are probability distributions. It can be seen as an infinite-dimensional generalization of Dirichlet distribution, where it is a prior over the space of countably infinite distributions. In the literature, DP has been constructed in different ways, the most well-known constructions are: infinite mixture model [46], distribution over distributions [47], Polya-urn scheme [48] and stick-breaking [44]. For more details, the interested reader is referred to [49].

Figure 1 shows two graphical models, DP mixture model and the finite mixture model with number of clusters L which becomes an infinite mixture model when L goes to ∞. Infinite mixture model is simply a generalization of the finite mixture model, where DP prior with infinite parameters is used instead of Dirichlet distribution prior with fixed number of parameters. The finite mixture model can be represented by the following equations:

$$\begin{aligned}\boldsymbol{\pi}|\alpha_0 &\sim Dirichlet(\alpha_0/L, ..., \alpha_0/L)\\ z_i|\boldsymbol{\pi} &\sim Discrete(\pi_1, ..., \pi_L)\\ \boldsymbol{\theta_k}|G_0 &\sim G_0\\ \boldsymbol{x_i}|z_i, \boldsymbol{\theta} &\sim F(\boldsymbol{\theta_{z_i}}) \end{aligned} \tag{8}$$

$F(\boldsymbol{\theta_{z_i}})$ denotes the distribution of the observation $\boldsymbol{x_i}$ given $\boldsymbol{\theta_{zi}}$, where $\boldsymbol{\theta_{zi}}$ is the parameter vector associated with component z_i. Here z_i indicates which latent cluster is associated with observation $\boldsymbol{x_i}$. Indicator z_i is drawn from a discrete distribution governed by parameter $\boldsymbol{\pi}$ drawn from a dirichlet distribution parametrized by α_0. We can simply say that $\boldsymbol{x_i}$ is distributed according to a mixture of components drawn from prior distribution G_0 and picked with probability given by the vector of mixing proportions $\boldsymbol{\pi}$. The model represented by Eq.(8) above is a finite mixture model, where L is the fixed number of parameters (components). The infinite mixture model can be derived by letting $L \to \infty$, then $\boldsymbol{\pi}$ can be represented as an infinite mixing proportion distributed according to stick-breaking process $GEM(\alpha_0)$ [44]. Thus, Eq.(8) can be equivalently expressed according to the graphical representation as:

$$\begin{aligned} G|\alpha_0, G_0 &\sim DP(\alpha_0, G_0) \\ \boldsymbol{\theta_i}|G &\sim G \\ \boldsymbol{x_i}|\boldsymbol{\theta_i} &\sim F(\boldsymbol{\theta_i}) \end{aligned} \tag{9}$$

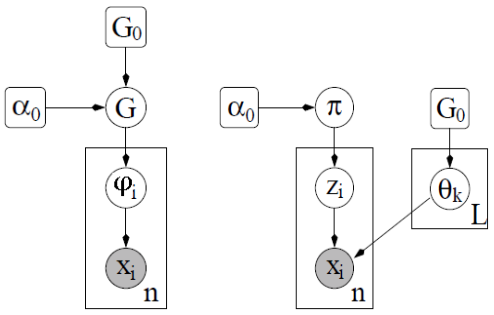
where $G = \sum_{k=1}^{\infty} \pi_k \delta_{\boldsymbol{\theta_i}}$ is drawn from DP prior, $\delta_{\boldsymbol{\theta_i}}$ is a Dirac delta function centered at $\boldsymbol{\theta_i}$. Technically, DP is a distribution over distribution [47], where $DP(G_0, \alpha_0)$, is parametrized by the base distribution G_0, and the concentration parameter α_0. Since DP is distribution over distributions, a draw G from it is a distribution. Thus, we can sample $\boldsymbol{\theta_i}$ from G. Back to Eq.(8), by integrating over the mixing proportion $\boldsymbol{\pi}$, we can write the prior for z_i as conditional probability of the following form [50]:

$$p(z_i = c|z_1, ..., z_{i-1}) = \frac{n_c^{-i} + \alpha_0/L}{i - 1 + \alpha_0} \tag{10}$$

where n_c^{-i} is the number of z_i for $j < i$ that are equal to c. By letting L goes to infinity we get the following equations:

$$\begin{aligned} P(z_i = c|z_1, ..., z_{i-1}) &\to \frac{n_c^{-i}}{i - 1 + \alpha_0} \\ P(z_i \neq z_j \, for \, all \, j < i|z_1, ..., z_{i-1}) &\to \frac{\alpha_0}{i - 1 + \alpha_0} \end{aligned} \tag{11}$$

For an observation $\boldsymbol{x_i}$ with $z_i \neq z_j \, for \, all \, j < i$, a new component gets created with indicator $z_i = c_{new}$. For more details about the process of obtaining the



(a) DP mixture model (b) Finite mixture model

Figure. 1 Graphical model

prior distribution, reader is referred to [50].

4. Stream-based Active Learning Approach

In this section, we develop the active learning approach SAL along with the estimator model that will be used to estimate the marginal and conditional distributions needed for SAL to work online.

4.1. Active Learning Approach

In Sec. 3.1, we assumed that the underlining conditional distribution $p(y|\boldsymbol{x})$ is known, but in reality it is not. Thus, we need to estimate it. Furthermore, in online setting, comparing the effect of labelling certain data instances against that of other data instances (as done in Eq.(3)) is not possible. Thus, storing pools of data seen so far might be a choice. However, it will violate the online learning assumption. We, instead, estimate the probability of unlabelled and labelled data at time t. Consider $p(y|\boldsymbol{x}, D_t)$, $p(\boldsymbol{x}|X_{U_t})$ and $p(\boldsymbol{x}|X_{L_t})$ as estimators for the true conditional distribution, the unlabelled data distribution and the labelled data distribution at time t, where D_t represents the set of the previously seen data instances with the labels of the queried ones. Thus, Eq. (4) equipped with the *importance weighting* on the labelled data can be written as the sum of the following:

$$\hat{R}_{D_t}(\boldsymbol{\phi}_t) = \int_{\boldsymbol{x}} L\big(p(\hat{y}|\boldsymbol{x},\boldsymbol{\phi}_t), p(y|\boldsymbol{x}, D_t)\big) p(\boldsymbol{x}|X_{U_t}) d\boldsymbol{x} \tag{12}$$

$$\hat{R}'_{D_t}(\boldsymbol{\phi}_t) = \int_{\boldsymbol{x}} \frac{L\big(p(\hat{y}|\boldsymbol{x},\boldsymbol{\phi}_t), p(y|\boldsymbol{x}, D_t)\big)}{p(q=1|\boldsymbol{x})} p(\boldsymbol{x}|X_{L_t}) d\boldsymbol{x} \tag{13}$$

where $\boldsymbol{\phi}_t$ denotes the classifier parameters after being trained on D_{L_t}. Based on Eq. (12) and Eq. (13), we can develop an online querying strategy similar to the one proposed in Eq. (3). The data instance can be assessed on-the-fly by comparing the error reduction incurred by labelling it against the highest error reduction. To compute the highest error reduction, we can sample a pool of unlabelled data at each time step from $p(\boldsymbol{x}|X_{U_t})$. Then, we search for the sample that incurs the highest error reduction. A more direct approach would be to use a non-convex optimizer to find the highest error reduction to be taken as an error reference. Both approaches are nonetheless computationally expensive as they involve estimation of integrals. Thus, we need to compute the expectation of the error reduction since the labels are unknown. This is however still computationally very demanding.

We can conclude from Eq. (12), (13) that by labelling the samples that have the largest contribution to the current error [43], there is a good chance for a large decrease of the expected future error. This contribution can be expressed through the following equations:

$$\hat{R}_{D_{t-1}}(\boldsymbol{\phi}_{t-1}, \boldsymbol{x}_t) = L\big(p(\hat{y}_t|\boldsymbol{x}_t,\boldsymbol{\phi}_{t-1}), p(y_t|\boldsymbol{x}_t, D_{t-1})\big) p(\boldsymbol{x}_t|X_{U_{t-1}}) \tag{14}$$

$$\hat{R}'_{D_{t-1}}(\boldsymbol{\phi}_{t-1}, \boldsymbol{x}_t) = L\big(p(\hat{y}_t|\boldsymbol{x}_t,\boldsymbol{\phi}_{t-1}), p(y_t|\boldsymbol{x}_t, D_{t-1})\big) \tilde{p}(\boldsymbol{x}_t|X_{L_{t-1}}) \tag{15}$$

As stated in the introduction, we seek to mitigate harmful bias that is caused by dynamic changes in the data (i.e., *concept drift* and *concept evolution*). Thus, if $\boldsymbol{x}_t$ is queried and wrongly classified, SAL integrates the weight effect of $p(q_t = 1|\boldsymbol{x}_t)$ into the current labelled data marginal distribution $\big(p(\boldsymbol{x}_t|X_{L_{t-1}})\big)$. This is done by repetitive update $\big(\frac{1}{p(q_t=1|\boldsymbol{x}_t)}\big)$ iterations (Sec. 4.2). Hence, $\tilde{p}(\boldsymbol{x}_t|X_{L_{t-1}})$

represents the labelled data marginal distribution with the weight effect of the previously queried samples integrated.

Equation (14) encourages querying samples that have strong representativeness among the unlabelled data and that are expected to be wrongly classified; while Eq.(15) encourages querying those which have strong representativeness among the labelled data, but, still wrongly classified. Such samples are rare. However, Eq. (15) allows the learner to be completely independent from the sampling approach, as it integrates the *sampling bias* independently from the learner algorithm. Thus, as the learner proceeds, Eq. (15) also helps to switch the focus from only representative samples to samples which are underestimated.

The querying probability is computed by comparing the samples with the one that has the largest contribution to the error. A solution can be devised by trying to optimize Eq.(14) and Eq.(15). However, to avoid time-consuming computation and keep the AL algorithm independent of the learner, we maintain the largest contribution to the error among the already seen data in variable A_t. Hence, this latter becomes a comparison reference for computing the probability of querying. A forgetting factor β empirically set to 0.9 is used to consider the dynamic nature of the data:

$$A_t = \max\big((\hat{R}_{D_{t-1}}(\boldsymbol{\phi}_{t-1}, \boldsymbol{x}_t) + \hat{R}'_{D_{t-1}}(\boldsymbol{\phi}_{t-1}, \boldsymbol{x}_t)), \beta A_{t-1}\big) \tag{16}$$

$$p(q_t = 1|\boldsymbol{x}_t, D_{t-1}, \boldsymbol{\phi}_{t-1}) = \frac{1}{A_t}\big(\hat{R}_{D_{t-1}}(\boldsymbol{\phi}_{t-1}, \boldsymbol{x}_t) + \hat{R}'_{D_{t-1}}(\boldsymbol{\phi}_{t-1}, \boldsymbol{x}_t)\big) \tag{17}$$

The number of classes evolves over time such that new classes may emerge and old ones may vanish. Thus, $p(y_t|\boldsymbol{x}_t, D_{t-1})$ (in Eq. (14) and Eq. (15)) must account for all classes in the data stream. Potentially, the length of the stream is infinite, which means that the probability of receiving infinite different classes is not zero. Hence, the support of the distribution over the classes must be infinite. To allow that, stick-braking distribution is imposed as a prior over the classes. Intuitively, this prior allows to foresee a probability on the creation of new classes. To remove obsolete classes, we propose an online estimator of $p(y_t|\boldsymbol{x}_t, D_{t-1})$ equipped with a forgetting factor to handle the evolving nature of data. The same model estimates $p(\boldsymbol{x}_t|X_{L_{t-1}})$ and $p(\boldsymbol{x}_t|X_{U_{t-1}})$ online. More details about the estimators are found in the next section. *Concept evolution* is implicitly handled through the loss function in Eq. (14) and Eq. (15). While the support of the classifier's distribution $p(\hat{y}_t|\boldsymbol{x}_t, \boldsymbol{\phi}_{t-1})$ is set over the already seen classes, the estimator of $p(y_t|\boldsymbol{x}_t, D_{t-1}))$ poses a probability on the creation of a new class. Thus, the losses in Eq. (14) and Eq. (15) are high if the probability of a new class is high. Hence, the probability of querying (Eq. (17)) becomes high. We use the 0-1 loss funtion:

$$l(\hat{y}_t, y_t) = \begin{cases} 0 & if \quad \hat{y}_t = y_t \\ 1 & otherwise \end{cases} \tag{18}$$

hence, the loss in Eq. (14) and Eq. (15) can be rewritten as follows:

$$L\big(p(\hat{y}_t|\boldsymbol{x}_t, \boldsymbol{\phi}_{t-1}), p(y_t|\boldsymbol{x}_t, D_{t-1})\big) = E_{\hat{y}_t \sim p(\hat{y}_t|\boldsymbol{x}_t, \boldsymbol{\phi}_{t-1})}\big[E_{y_t \sim p(y_t|\boldsymbol{x}_t, D_{t-1})}[l(\hat{y}_t, y_t)]\big]. \tag{19}$$

Since the support of the classifier's distribution is over the already seen classes, $\hat{y}_t \in C_{t-1}$ with C_{t-1} is the set of classes discovered up until time $t-1$. On the other hand, the estimator support includes the novel class, $y_t \in C_{t-1} \cup \{|C_{t-1}|+1\}$. Thus, the loss is high if the probability of a new class $\big(p(y_t = |C_{t-1}|+1|\boldsymbol{x}_t, D_{t-1})\big)$ is high and the probability of querying becomes high. The steps of SAL are portrayed in Fig.2.

Under the constraint of limited labelling budget, a rational querying strategy needs to be applied. To implement this constraint, in [34], two counters were maintained: the number of labelled instances $f_t = |X_{L_t}|$ and the budget spent so far: $b_t = \frac{f_t}{|\text{data seen so far}|} = \frac{|X_{L_t}|}{|X_t|}$.

As data arrives, we do not query unless the budget is less than a constant B and querying is granted by the sampling model. However, over infinite time

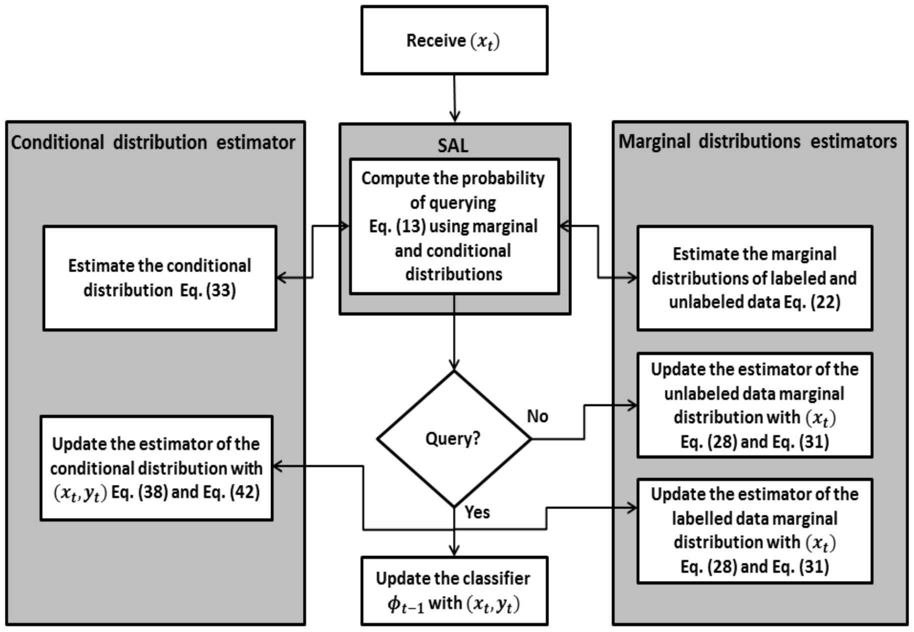

Receive (x_t)
Conditional distribution estimator
Estimate the conditional distribution Eq. (33)
Update the estimator of the conditional distribution with (x_t, y_t) Eq. (38) and Eq. (42)
SAL
Compute the probability of querying Eq. (13) using marginal and conditional distributions
Query?
No
Yes
Update the classifier ϕ_{t-1} with (x_t, y_t)
Marginal distributions estimators
Estimate the marginal distributions of labeled and unlabeled data Eq. (22)
Update the estimator of the unlabeled data marginal distribution with (x_t) Eq. (28) and Eq. (31)
Update the estimator of the labelled data marginal distribution with (x_t) Eq. (28) and Eq. (31)

Figure. 2 General scheme of SAL

horizon this approach will not be effective. The contribution of every query to the budget will diminish over time and a single labelling action will become less and less sensitive. Authors in [34] propose to compute the budget over fixed memory windows of size *wnd*. To avoid storing the query decisions for each window, an estimation of f_t and b_t were proposed:

$$\hat{b}_t = \frac{\hat{f}_t}{wnd} \tag{20}$$

where $\hat{f}_t$ is an estimate of how many instances were queried within the last *wnd* of incoming data.

$$\hat{f}_t = (1 - 1/wnd)\hat{f}_{t-1} + labelling_{t-1} \tag{21}$$

where $labelling_{t-1} = 1$ if instance x_{t-1} is labelled, and 0 otherwise. Using the

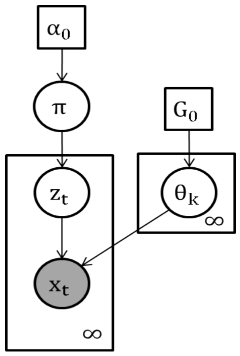


Figure. 3 Infinite mixture model

forgetting factor $(1-(1/wnd))$, the authors showed that $\hat{b}_t$ is unbiased estimate of b_t.

In this paper, SAL adopts this notion of budget; so that it can be assessed against the active learning algorithms proposed in [34]. Note that in our experiments, we set $wnd = 100$ as in [34].

4.2. ***Estimator Model***

In this section, we develop the model that will be used to estimate the distributions in Eq. (14) and Eq. (15) needed for SAL to work online. We describe the proposed estimator model and develop an online particle inference algorithm for it. While DPMM estimates the marginal distributions, the conditional distribution is estimated by an upgrade of DPMM. It accommodates labelled data using a stick-breaking process [44] over the classes. These estimations are done on-the-fly by performing online inference using the particle inference algorithm. For the sake of simplification, we start with an unsupervised clustering model, then we add a new ingredient to accommodate labelled data to result in a semi-supervised clustering algorithm.

4.2.1. ***Unsupervised clustering***

Figure 3 shows the infinite mixture model where $\boldsymbol{\pi}$ is drawn from a stick-breaking process $GEM(\alpha_0)$ and G_0 is a Normal-Inverse-Wishart distribution $NIW(.|\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0, k_0, v_0)$. Where $\boldsymbol{\mu}_0$ is the prior of the clusters' means, $\boldsymbol{\Sigma}_0$ controls the variance among their means, k_0 scales the diffusion of the clusters means and v_0 is the degree of freedom of the Inverse-Wishart distribution.

Following [25], we introduce a state vector H_t that summarizes the data seen up to time t. Hence, $H_t = \{z_t, m_t, \boldsymbol{n}_t, \boldsymbol{s}_t\}$ can represent all the statistics used by the model. Here z_t assigns the component generating $\boldsymbol{x}_t$, m_t is the number of components, $\boldsymbol{n}_t$ is a vector of components cardinalities and $\boldsymbol{s}_t$ is the sufficient statistics for each mixture component i.e., means $\boldsymbol{su}$ and scatter matrices $\boldsymbol{sc}$.

Given the concentration parameter α_0 and the prior distribution parameters $\{\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0, k_0, v_0\}$, we aim at computing online the marginal distribution of the current data $p(\boldsymbol{x}_t|X_{t-1})$ without the need for saving all data $X_{t-1} \equiv \boldsymbol{x}_{1:t-1}$:

$$p(\boldsymbol{x}_t|X_{t-1}) = \sum_{z_{1:t}} p(\boldsymbol{x}_t|z_{1:t}, X_{t-1})p(z_{1:t}|X_{t-1}) \tag{22}$$

The estimation of $p(\boldsymbol{x}_t|X_{U_{t-1}})$ and $p(\boldsymbol{x}_t|X_{L_{t-1}})$ in SAL can be derived from Eq.(22) by simply replacing X_{t-1} by $X_{U_{t-1}}$ or $X_{L_{t-1}}$.

$$p(z_{1:t}|X_{t-1}) = p(z_t|z_{1:t-1})p(z_{1:t-1}|X_{t-1}) \tag{23}$$

The first term of Eq. (22) can be written as follows:

$$p(\boldsymbol{x}_t|z_{1:t}, X_{t-1}) = \int_{\boldsymbol{\theta}} p(\boldsymbol{x}_t|\boldsymbol{\theta}, z_t)p(\boldsymbol{\theta}|z_{1:t}, X_{t-1}) \tag{24}$$

If z_t refers to a new component, $p(\boldsymbol{\theta}|z_{1:t}, X_{t-1})$ becomes equivalent to the prior distribution $p(\boldsymbol{\theta}|G_0)$. Otherwise, z_t refers to an already existing component. Then, $p(\boldsymbol{\theta}|z_{1:t}, X_{t-1})$ becomes equivalent to $p(\boldsymbol{\theta}|\boldsymbol{s}_{z_t,t-1}, n_{z_t,t-1}, z_{1:t-1})$, where $n_{z_t,t-1}$ is the number of data samples which have been assigned to component z_t until time $t-1$, $\boldsymbol{s}_{z_t,t-1} = \{\boldsymbol{su}_{z_t,t-1}, \boldsymbol{sc}_{z_t,t-1}\}$ is the sufficient statistics i.e., mean and variance respectively.

$$\begin{aligned}\boldsymbol{su}_{z_t,t-1}(z_{1:t-1}) &= \frac{\sum_{z_i=z_t, i<t} \boldsymbol{x}_i}{n_{z_t,t-1}} \\ \boldsymbol{sc}_{z_t,t-1}(z_{1:t-1}) &= \sum_{z_i=z_t, i<t} (\boldsymbol{x}_i - \boldsymbol{su}_{z_t,t-1})(\boldsymbol{x}_i - \boldsymbol{su}_{z_t,t-1})^T \end{aligned} \tag{25}$$

Equation (24) can be solved given the sufficient statistics, the past assignments and the model hyper-parameters. More details can be found in Appendix A. The first term of Eq. (23) can be computed in the same way as Eq.(11):

$$p(z_t|z_{1:t-1}) \propto \begin{cases} n_{z_t,t-1} & z_t \text{ is an existing cluster} \\ \alpha_0 & z_t \text{ is a new cluster} \end{cases} \tag{26}$$

The second term of Eq. (23), $p(z_{1:t-1}|X_{t-1})$, is the probability of the different configurations. Such configurations determine the different statistics represented by H_{t-1}. Thus, $p(H_{t-1}|X_{t-1})$ has the same probability as the posterior $p(z_{1:t-1}|X_{t-1})$. We track this posterior online by approximating it with a set of (at maximum) N particles. Upon the arrival of a new data point, the particles are extended to include a new assignment z_t assuming that the previous assignments are known and fixed. Thus, the task is to update the posterior of the extended particles at time t, $p(H_t|X_t)$, given that the posterior at $t-1$, $p(H_{t-1}|X_{t-1})$ is known. In order to prevent combinatorial explosion, we use the re-sampling technique proposed in [26] which retains the maximum N particles. We approximate the posterior at time t in two steps:

Updating:

$$p(H_t|X_t) \propto \int_{H_{t-1}} p(H_t|H_{t-1}, \boldsymbol{x}_t) p(\boldsymbol{x}_t|H_{t-1}) p(H_{t-1}|X_{t-1}) \tag{27}$$

Given the N particles along with their weights, $p(H_{t-1}|X_{t-1}) = \sum_{i=1}^{N} w_{t-1}^{(i)} \delta(H_{t-1} - H_{t-1}^{(i)})$, the update can be written as follow.

$$p(H_t|\boldsymbol{x}_{1:t}) \propto \sum_{i=1}^{N} p(H_t|H_{t-1}^{(i)}, \boldsymbol{x}_t) p(\boldsymbol{x}_t|H_{t-1}^{(i)}) w_{t-1}^{(i)} \tag{28}$$

The solution of the second term of Eq. (28) can be computed in a similar way to Eq. (24) and Eq. (26). Following the update step, the number of resulting particles for each $H_{t-1}^{(i)}$ equals to the number of existing components $m_{t-1}^{(i)} + 1$. The new assignment z_t expresses the different configurations of the new particles. Therefore,

$$\begin{aligned} p(H_t^{(i_2)}|H_{t-1}^{(i)}, \boldsymbol{x}_t) &= p(z_t = j|H_{t-1}^{(i)}, \boldsymbol{x}_t) \\ &\propto p(\boldsymbol{x}_t|z_t = j, H_{t-1}^{(i)})p(z_t = j|H_{t-1}^{(i)}) \end{aligned} \tag{29}$$

We use Cantor pairing function to uniquely encode the jth assignment and the ith particle to a single natural number:

$$\begin{aligned} i_2 &= \Omega(i, j) \\ \Omega(i, j) &= \frac{1}{2}(i + j)(i + j + 1) + j \end{aligned} \tag{30}$$

By solving Eq. (29), we determine the weights of the new particle $w_t^{i_2}$. Equation (29) can be solved in a similar way to Eq. (24) and Eq. (26). The elements of the new state vector $H_t^{(i_2)}$ are updated as follows:

$$H_t^{(i_2)} = \begin{cases} z_t^{(i_2)} & = j \qquad j \text{ is an existing component} \\ n_{j,t}^{(i_2)} & = \lambda n_{j,t-1}^{(i)} + 1 \\ n_{k,t}^{(i_2)} & = \lambda n_{k,t-1}^{(i)} \quad \forall k \neq j, k \leq m_t^{(i)} \\ \boldsymbol{su}_{j,t}^{(i_2)} & = \frac{\lambda n_{j,t-1}^{(i)} \boldsymbol{su}_{j,t-1}^{(i)} + \boldsymbol{x}_t}{n_{j,t}^{(i)}} \\ \boldsymbol{sc}_{j,t}^{(i_2)} & = \lambda \boldsymbol{sc}_{j,t-1}^{(i)} + n_{j,t-1}^{(i)} \boldsymbol{su}_{j,t-1}^{(i)} \boldsymbol{su}_{j,t-1}^{(i)T} \\ & \quad - n_{j,t}^{(i)} \boldsymbol{su}_{j,t}^{(i)} \boldsymbol{su}_{j,t}^{(i)T} + \boldsymbol{x}_t \boldsymbol{x}_t^T \\ & \\ z_t^{(i_2)} & = m_{t-1}^{(i)} + 1 \qquad j \text{ is a new component} \\ m_t^{(i_2)} & = m_{t-1}^{(i)} + 1 \\ n_{j,t}^{(i_2)} & = 1 \\ n_{k,t}^{(i_2)} & = \lambda n_{k,t-1}^{(i)} \quad \forall k \leq m_{t-1}^{(i)} \\ \boldsymbol{su}_{j,t}^{(i_2)} & = \boldsymbol{x}_t \\ \boldsymbol{sc}_{j,t}^{(i_2)} & = \boldsymbol{0} \end{cases} \tag{31}$$

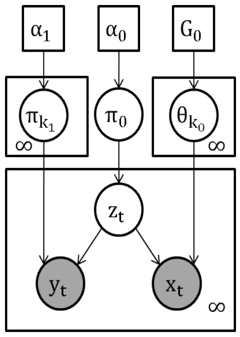


Figure. 4 Proposed semi-supervised clustering model

where λ is a forgetting factor which allows the components to adapt with changes. Having approximated all the terms of Eq.(28), we end up with $M = \sum_{i=1}^{N}(m_{t-1}^{(i)} + 1)$ new particles along with their weights $w_t^{(i_2)}$. So, we move to the next step which reduces the number of created particles to a fix number N.

300 **Re-sampling:**

We follow the re-sampling technique proposed in [26] which discourages the less-likely particles (configurations) and improves the particles that explain the data better. It keeps the particles whose weights are greater than $1/c$ and re-samples from the remaining particles. The variable c is the solution of the following equation:

$$\sum_{i_2=1}^{M} \min\{cw_t^{(i_2)}, 1\} = N \tag{32}$$

The weights of re-sampled particles are set to $1/c$ and the weights of the particles greater than $1/c$ are kept unchanged.

Next, we consider the labels by proposing stick-breaking prior over the classes.

4.2.2. ***Semi-supervised clustering***

305 The stick-breaking component assignment is the same as the Gaussian component assignment. That is, every Gaussian component is associated with a stick-breaking component where the variable z_t controls the components assignment (see Fig.4).

We propose to include the classes information in the state vector H_t so that it becomes $H'_t = \{H_t, \boldsymbol{n}'_t\}$, where $\boldsymbol{n}'_t$ is a matrix of the number of different classes assigned to each component. Hence H'_t comprises all the statistics used in the model. Assume that at time t, we need to predict the distribution over y_t given all the data and their labels seen so far.

$$p(y_t|\boldsymbol{x}_t, D_{t-1}) = \sum_{z_t} p(y_t|z_t, D_{t-1}) p(z_t|D_{t-1}, \boldsymbol{x}_t). \tag{33}$$

The first term of Eq. (33) can be computed in a similar way to Eq. (26), where z_t selects the stick-breaking component generating y_t. Hence, the probability of y_t depends only on the data assigned to component z_t. More details can be found in Appendix B.

$$p(y_t|z_t, D_{t-1}) \propto \begin{cases} n'_{z_t,y_t,t} & y_t \text{ is an existing class} \\ \alpha_1 & y_t \text{ is a new class} \end{cases} \tag{34}$$

where $n'_{z_t,y_t,t}$ refers to the number of the data samples which are assigned to component z_t and have label y_t at time t. So, to compute (34), the distribution of the labels to the data in each component must be memorized. The second term of Eq.(33) can be written as follows:

$$p(z_t|\boldsymbol{x}_t, D_{t-1}) = \sum_{z_{1:t-1}} p(z_t|z_{1:t-1}, \boldsymbol{x}_t, D_{t-1}) p(z_{1:t-1}|\boldsymbol{x}_t, D_{t-1}) \tag{35}$$

$$p(z_t|z_{1:t-1}, \boldsymbol{x}_t, D_{t-1}) \propto p(\boldsymbol{x}_t|z_{1:t}, D_{t-1}) p(z_t|z_{1:t-1}) \tag{36}$$

$$p(z_{1:t-1}|\boldsymbol{x}_t, D_{t-1}) \propto p(\boldsymbol{x}_t|z_{1:t-1}, D_{t-1}) p(z_{1:t-1}|D_{t-1}) \tag{37}$$

Equation (35) can be solved by following similar steps to (22) but with additional observation $Y_{L_{t-1}}$. Hence, $p(\boldsymbol{x}_t|z_{1:t}, D_{t-1})$, $p(\boldsymbol{x}_t|z_{1:t-1}, D_{t-1})$ is solved by following the same steps in Eq. (24) after replacing H_{t-1} by H'_{t-1}. The second term of Eq. (37), $p(z_{1:t-1}|D_{t-1})$, has the same probability as the posterior $p(H_{1:t-1}|D_{t-1})$. Thus, Similar to Eq.(28), the solution of Eq. (33) depends only on the elements of the state vector H'_t along with its posterior distribution. We track this posterior online. Similar to Sec. 4.2.1, we approximate the posterior at time t by a set of N weighted particles using two steps; updating and re-sampling. The re-sampling step is the same as in Sec. 4.2.1. The updating step follows the same way:

$$p(H'_t|D_t) \propto \sum_{i=1}^{N} p(H'_t|H'^{(i)}_{t-1}, \boldsymbol{x}_t, y_t) p(\boldsymbol{x}_t, y_t|H'^{(i)}_{t-1}) w'^{(i)}_{t-1} \tag{38}$$

$$p(H'^{(i_2)}_t|H'^{(i)}_{t-1}, \boldsymbol{x}_t, y_t) = p(z_t = j|H'^{(i)}_{t-1}, \boldsymbol{x}_t, y_t) \propto$$
$$p(\boldsymbol{x}_t, y_t|z_t = j, H'^{(i)}_{t-1}) p(z_t = j|H'^{(i)}_{t-1}) \tag{39}$$

$$p(\boldsymbol{x}_t, y_t|z_t = j, H'^{(i)}_{t-1}) = p(\boldsymbol{x}_t|z_t = j, H'^{(i)}_{t-1}) p(y_t|z_t = j, H'^{(i)}_{t-1})) \tag{40}$$

The second term of Eq.(39) is computed in Eq.(26). The first and second terms of Eq. (40) can be solved in the same way as in Eq. (24) and Eq. (34) respectively. The second term of Eq. (38) can be written as follows:

$$p(\boldsymbol{x}_t, y_t|H'^{(i)}_{t-1}) = \sum_{z_t} p(\boldsymbol{x}_t, y_t|z_t, H'^{(i)}_{t-1}) p(z_t|H'^{(i)}_{t-1}) \tag{41}$$

The elements of the new state vector are updated in the same way as in Eq.(31):

$$H'^{(i_2)}_t = \begin{cases} H^{(i_2)}_t & j \text{ is an existing component} \\ n'^{(i_2)}_{j,y_t,t} & = \lambda' n'^{(i)}_{j,y_t,t-1} + 1 \\ \boldsymbol{n}'^{(i_2)}_{k,t} & = \lambda' \boldsymbol{n}'^{(i)}_{k,t-1} \quad \forall k \neq j, k \leq m^{(i)}_t \\ & \\ H^{(i_2)}_t & j \text{ is a new component} \\ n'^{(i_2)}_{j,y_t,t} & = 1 \\ \boldsymbol{n}'^{(i_2)}_{k,t} & = \lambda' \boldsymbol{n}'_{k,t-1} \quad \forall k \leq m^{(i)}_{t-1} \end{cases} \tag{42}$$

The estimation of $p(y_{t+1}|\boldsymbol{x}_{t+1}, D_t))$ in SAL is computed by Eq. (33). The estimation of $p(\boldsymbol{x}_t|X_{U_{t-1}})$ and $p(\boldsymbol{x}_t|X_{L_{t-1}})$ are computed by Eq. (22). We maintain three state vectors, one for the unlabelled data H_{t-1}^u, one for the labelled data H_{t-1}^l and one for all the data samples and their labels seen up to time $t-1$, H'_{t-1}. Hence, three estimators represented by state vectors associated with their weights: $\{(H'_t, w'_t)$, (H_t^u, w_t^u) and $(H_t^l, w_t^l)\}$ and their hyper-parameters: $\{(\alpha_0, \alpha_1, \boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0, k_0, v_0)$, $(\alpha_0^u, \alpha_1^u, \boldsymbol{\mu}_0^u, \boldsymbol{\Sigma}_0^u, k_0^u, v_0^u)$ and $(\alpha_0^l, \alpha_1^l, \boldsymbol{\mu}_0^l, \boldsymbol{\Sigma}_0^l, k_0^l, v_0^l)\}$ are maintained.

Having introduced the AL approach in Sec. 4.1 and developed the needed estimator model in Sect 4.2, the full details of the algorithm are provided in Alg. (1).

5. Experiments

In this section, we present the algorithms that SAL is compared against, then we describe the datasets on which the experiments are carried out. SAL is compared against two types of stream-based AL approaches. The first set of approaches introduced in [34] takes into consideration the challenges of data stream, namely the *infinite length* of the data and *concept drift*, but ignores *concept evolution*. These methods are:

- *VarUn*: Variable Uncertainty, stream-based AL.

- *RanVarUn*: Variable Randomized Uncertainty, stream-based AL.

We also consider a baseline random sampling: *Rand*. The aim of this comparison is to show how SAL performs against these methods just cited (with restricted budget). Fortunately, these methods are integrated in the MOA data stream software suite [51] which helps carry out the experiments without the need to implement them.

The second set of stream-based AL approaches are developed to cope with *concept drift* and *concept evolution* [36, 52]. However, they do not explicitly handle *concept drift* as well as the *sampling bias* problem. We consider the following:

Algorithm 1 Steps of SAL

```
1: Input: data stream, hyper-parameters of the estimators, forgetting factors $\lambda$ and $\lambda'$ $\big($Eq. (31) and Eq. (42) resp.$\big)$, the input of the classifier, forgetting factor $\beta$ (Eq. (16)), budget $B$, $wnd$ (Eq. (20)), the maximum number of particles $N$
2: initialize: the weight of the first particle of all the three estimators to 1, the number of components for all the estimators' state vector to 0, $\hat{f}_t = 0$ (Eq. (20)), $A_0 = 0$ (Eq. (16)), $t = 1$
3: while (true) do
4:     $t \leftarrow t + 1$,
5:     Receive $\boldsymbol{x_t}$
6:     if $\hat{b}_t < B\big($Eq. (20)$\big)$ then                              ▷ enough budget
7:         compute $a_t = p(q_t = 1|\boldsymbol{x}_t, D_{t-1}, \boldsymbol{\phi}_{L_{t-1}})$ that refers to the probability of querying $\boldsymbol{x}_t$ $\big($Eq. (17)$\big)$
8:         $q_t \sim Bern(a_t)$
9:         if $q_t = 1$ then                                              ▷ querying
10:            $y_t \leftarrow query(\boldsymbol{x_t})$
11:            Update the classifier (represented by its vector parameter $\boldsymbol{\phi}_{t-1}$)
12:            Update the estimator of the labelled data distribution $\big($represented by its state vector's particles along with their weights $(H^{l(i)}_{t-1}, w^{l(i)}_{t-1})\big)$ $\big($Eq. (28) and Eq. (31)$\big)$
13:            Update the estimator of the conditional distribution $\big($represented by its state vector's particles along with their weights $(H'^{(i)}_{t-1}, w'^{(i)}_{t-1})\big)$ $\big($Eq. (38) and Eq. (42)$\big)$
14:        else
15:            Classify the instance $\boldsymbol{x}_t$ using the classifier (base learner represented by its vector parameter $\boldsymbol{\phi}_{t-1}$)
16:            Update the estimator of the unlabelled data distribution $\big($represented by its state vector's particles along with their weights $(H^{u,(i)}_{t-1}, w^{u,(i)}_{t-1})\big)$ $\big($Eq. (28) and Eq. (31)$\big)$
17:        end if
18:    end if
19:    Compute $\hat{f}_{t+1}$ $\big($Eq. (21)$\big)$
20: end while
```

- *lowlik*: Low-likelihood criterion specialized for quick unknown class discovery [52].

- *qbc*: Query-by-Commitee, a stream-based version proposed by [36].

- *qbc-pyp*: Stream-based joint exploration-exploitation AL proposed in [52].

These methods have shown good class discovery performance on unbalanced

Table. 1 Benchmark Datasets properties used for comparing SAL against [34]

Datasets	N	d	N_c	$S\%$	$L\%$	E	CD	CE
Electricity	45312	8	2	42	58	0.98	1	0
Airlines	53938	7	2	36	64	0.94	1	0
Forest	10000	55	7	12.6	16.2	1	0	1
Digits	13184	25	10	0.1	50.05	0.62	0	1

Table. 2 Benchmark Datasets properties used for comparing SAL against [52, 36]

Datasets	N	d	N_c	$S\%$	$L\%$	E	CD	CE
Pageblocks	5473	10	5	0.49	89.28	0.26	0	1
Shuttle	20000	9	7	0.02	78.4	0.34	0	1
Thyroid	7200	21	3	2.47	92.47	0.28	0	1
Forest	5000	10	7	3.56	24.36	0.94	0	1
KDD	33650	41	10	0.04	51.46	0.17	1	1
Digits	13184	25	10	0.1	50.05	0.62	0	1

data including the datasets that we use in this study. Hence, by comparing against them, we highlight the efficiency of SAL in dealing with *concept evolution* in challenging setting where the class of the datasets are highly unbalanced. We set up the same settings described in [36]. It is worth noting that although these methods are stream-based AL, they memorize all labelled samples and re-use them at each iteration for updating the model. On the contrary, SAL does not reuse past instances in a strict stream-based learning environment. That is, its complexity does not grow with time.

As we have shown previously, SAL is flexible and any learner (classifier) can be plugged in. Here, we use online Naive Bayes as a learner like in [34]. For all the experiments, the number of particles N is set to 5. Normally, as we increase the number of particles, the estimator model gives better estimation, but the computation becomes heavier.

5.1. ***datasets***

SAL is evaluated on eight real-world benchmark datasets widely used in the AL area: Pageblocks, Shuttle, Thyroid, Covertype (Forest), KDDCup 99 network intrusion detection (KDD), Electricity, Airlines and MNIST handwritten digits (Digits). The first five of these datasets are downloaded from UCI repository

[53]. Electricity [54] and Airlines [55] are popular real-world benchmark used in evaluating classification in the context of evolving data streams. Digits is a well-known vision dataset[1].

These datasets are presented in Tab.1 and Tab.2, where N is the number of instances, d is the number of features/attributes, N_c is the number of classes, $S\%$ and $L\%$ are the proportions of smallest and largest classes respectively, E represents the class entropy, CD and CE flag if the data experiences *concept drift* and *concept evolution*. According to [34], Electricity data experiences more frequent and abrupt drift than Airlines data. Both enjoy well-balanced binary classes (high entropy). Such properties will help manifest the capability of SAL to efficiently cope with *concept drift* and *sampling bias*. Although Forest data enjoys high class entropy, its classes are unbalanced through time; Hence, it experiences *concept evolution*. Pageblocks, Shuttle, Thyroid and KDD show multiple classes in naturally unbalanced proportions. Such property will help manifest the capability of SAL to efficiently discover the unknown classes. Digits dataset is used in its preprocessed version [56]

In this paper, we use the full Pageblocks, Thyroid and Electricity datasets but only portions of Shuttle, Forest, KDD and Digits datasets. When comparing against the first type of competitors (*VarUn*, *RanVarUn* and *Rand*) [34], 10000 and 53938 (10%) instances are used from Forest (hight entropy) and Airlines datasets respectively. As for the second type of competitors (*lowlik*, *qbc* and *qbc-pyp*) [52, 36], we follow the setting in [36] where 20000, 33650 and 5000 instances are used from Shuttle, KDD and Forest respectively. For both types of competitors, 13184 instances are used from Digits data. These settings are observed to ensure a fair comparison of SAL against the competitors.

All datasets were collected and saved in flat files. To simulate streams from these files, SAL reads through the data in the same order it was collected. It processes the samples sequentially before they are discarded. If SAL decides to query a certain sample, this latter is sent along with its label to the online

[1]http://yann.lecun.com/exdb/mnist/

Table. 3 Learning rate parameters

Datasets	Pageblocks	Shuttle	Thyroid	Forest	KDD	Electricity	Airlines	Digits
λ	0.6	0.7	0.8	0.6	0.7	0.7	0.7	0.8
λ'	0.7	0.7	0.7	0.7	0.6	0.6	0.6	0.7

Table. 4 SAL hyper-parameters setting

Hyper-parameters	$\alpha_0\ \alpha_0^u\ \alpha_0^l$	$\mu_0\ \mu_0^u\ \mu_0^l$	$v_0\ v_0^u\ v_0^l$	$k_0\ k_0^u\ k_0^l$
Values	1	$\mathbf{0}$	$d+2$	0.01

classifier in order to update itself. Note that it is assumed that the ground truth is available immediately after a query is made.

5.2. ***Classification performance***

In this section, we evaluate SAL against the methods presented in [34] on all datasets (see Tab. 1) that have well-balanced classes (E is high, except for Digits). Such datasets will help manifest the classification performance of SAL compared to the others. Following the competitors setting, the evaluation of SAL is based on a prequential methodology. The classification performance of SAL is measured according to the average accuracy which is the ratio of correctly classified testing samples:

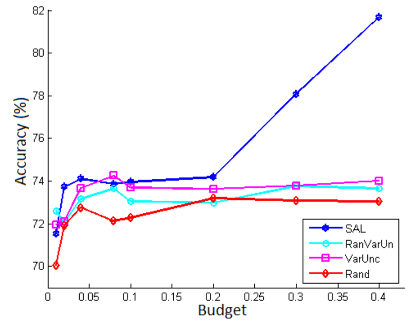
$$AA = \frac{\sum_{i \in T} 1_{(\hat{y}_i, y_i)}}{|T|} \tag{43}$$

$$1_{(\hat{y}_i, y_i)} = \begin{cases} 1 & if \quad \hat{y}_i = y_i \\ 0 & otherwise \end{cases} \tag{44}$$

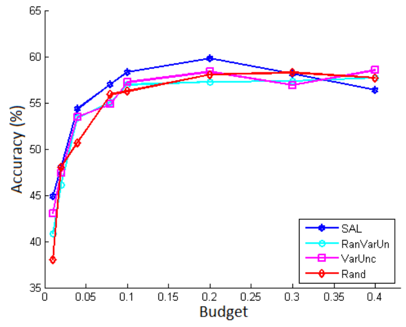
where the elements of T are the indices of all testing samples, $|T|$ is the total number of testing samples. All results are averaged over 30 runs in order to capture the real performance of SAL.

5.2.1. ***Settings***

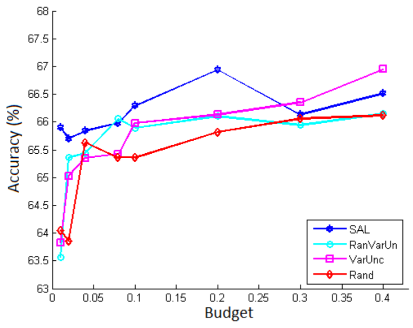
The hyper-parameters of the estimator model are fixed apriori (see Tab. 3). In order to allow vague prior, we set α_0, α_0^u and α_0^l to 1. The means u_0, u_0^u and u_0^l are set to $\mathbf{0}$. The covariance matrices Σ_0, Σ_0^u and Σ_0^l are roughly set to be



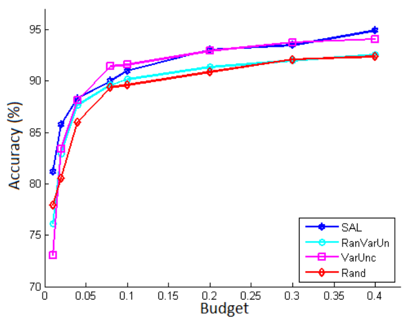
(a) Electricity



(b) Forest



(c) Airlines



(d) Digits

Figure. 5 Classification performance

410 as large as the dispersion of the data. The degree of freedom of the Wishart distributions v_0, v_0^u , v_0^l must be greater than d. We set them to $d+2$. The hyper-parameters k_0, k_0^u and k_0^l are empirically set to 0.01. Because at this stage, we are interested only in the classification error, the hyper-parameter α_1 which controls the prior over the classes is set to a low value. It can be seen
415 from Eq. (34) that when α_1 is low, the model tends to put low probability on the emergence of new classes. We empirically set it to 0.01. The effect of the forgetting factors λ and λ' in Eq.(31) and Eq.(42) on SAL's performance is studied and the parameters are set to the values that give the best performance (see Table 4). Note that we also consider the best results of the competitors.

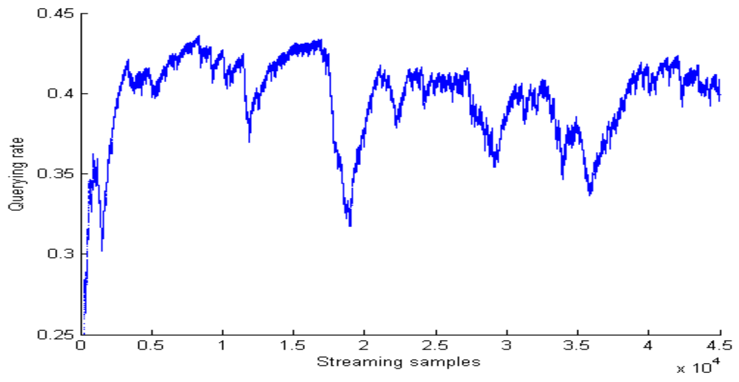
5.2.2. ***Performance Analysis***

Following similar setting in [34], we carry out the experiments on the four datasets shown in Tab. 1 using different budget values. Figure 5 shows the classification accuracy of both SAL and the competitors for different values of budget B.

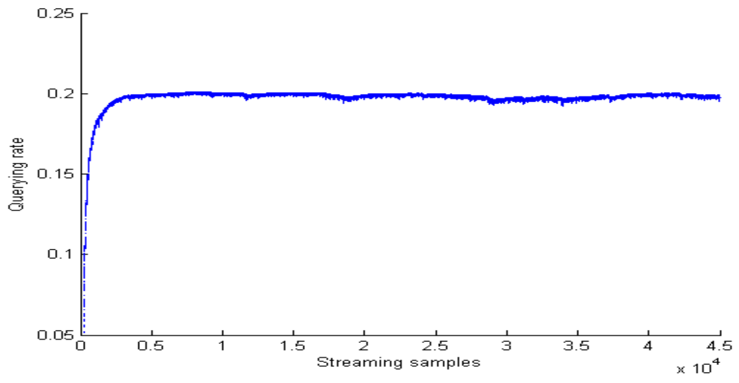
Using budget less than 0.05, SAL outperforms all competitors on the four datasets: Electricity, Forest, Airlines and Digits datasets. Such superiority with low budget is a very strong point for SAL as it aligns with the goal of AL which is high accuracy with low budget. SAL shows the best performance on Electricity data for all budgets except for 0.1 and for budget less than 0.02. On Airlines and Forest datasets, SAL has the best performance when using budget less than 0.3. SAL also gives the best results on Digits dataset using budget less than 0.05 or more than 0.3.

As stated earlier, Airlines and Electricity datasets suffer from *concept drift*. The *concept drift* occurring for Airlines datasets is less frequent and softer than that of Electricity dataset [34]. Having frequent aggressive drift makes *sampling bias* more likely to occur. Indeed, AL's confident sampling assessment is more likely to miss drifting valuable samples. Thus, the contrary behaviour between SAL's accuracies for Electricity and Airlines datasets as budget exceeds 0.2 could be explained by SAL' capability of coping with *concept drift* and *sampling bias*. As querying budget of Electricity data exceeds 0.2, the accuracy of competitors converges to a certain value while SAL's accuracy increases linearly. That is, the competitors do not exploit the budget properly. Since drift of Airlines dataset is less tricky, competitors are able to handle it when higher budget is granted. However, for low budget SAL enjoys the best performance.

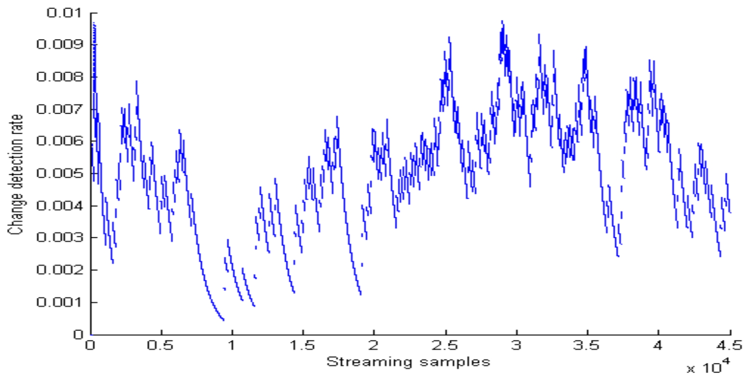
Figure 6 shows SAL behaviour along the stream of Electricity dataset with two different budgets 0.4 and 0.2. Figure 6a and Fig6b present the prequential labelling (querying) rate for budget=0.4 and 0.2 respectively. Figure.6c presents the prequential drift (change) rate. To smooth the curve, a fading factor of 0.999 is used. The drift is detected using the drift detection method (DDM) proposed



(a) Querying rate for budget=0.4



(b) Querying rate for budget=0.2



(c) Drift rate

Figure. 6 Active learning behaviour along the stream for Electricity

by [57]. We can notice the correspondence between the drift and querying rate. SAL tends to query less intensively when drift rate is low. However, it is not only the drift that drives SAL querying behaviour and it is not guaranteed that DDM detects all occurring drifts. To understand better the behaviour of SAL, we plot the querying rate on Electricity data for budget=0.2 corresponding to the inflection point of SAL's accuracy Fig.6b. The experiments show that the querying rate is less sensitive to the drift rate compared to that for budget=0.4. That supports the explanation (see previous paragraph) for the extremely high performance of SAL on Electricy when budget exceeds 0.2.

Note that the reasons why SAL outperforms the competitors are not only because of its ability to explicitly handle *sampling bias* problem. SAL's superiority is rooted in the fact that it tries to directly reduce the expected future error instead of employing heuristic AL criteria as the competitors do. Forest and Digits datasets do not involve *concept drift*, however, SAL provides a good classification performance compared to the competitors.

To sum up, we highlight two main differences between SAL and the competitors AL approaches. Firstly, SAL explicitly deal efficiently with the problem of *sampling bias*. On the other hand, *RanVarUn* combines naive randomization with uncertainty criterion to deal with drift. By doing so, the budget is wasted on some random queries. *VarUnc* does not handle *sampling bias* problem. Secondly, SAL takes the importance of data marginal distribution into account; while the competitors do not.

5.3. ***Class discovery performance***

In this section, we evaluate SAL against the methods in [52, 36] on datasets that show multiple classes in naturally unbalanced proportions (see Tab. 2). Such datasets will help manifest the class discovery performance of SAL. Following the competitors setting, the class discovery performance of SAL is measured using the average class accuracy [56] which is given as:

$$AA_j = \frac{\sum_{i \in T_j} 1_{(\hat{y}_i, j)}}{|T_j|} \tag{45}$$

where the elements of T_j are the indices of all testing samples coming from class j.

$$ACA = \frac{\sum_{j \in C} AA_j}{|C|} \tag{46}$$

where the elements of C are the classes. It is worth mentioning that the final class accuracy is fairly penalized when there are misclassifications in small classes. All results are averaged over 15 runs with two-fold cross-validation.

5.3.1. ***Settings***

SAL takes the data density into account when querying. It might then consider the data representing small classes as outliers or noise and therefore never queries them. To avoid such a scenario and to improve the class discovery performance of SAL, we increase the importance of the small classes by integrating online their effect in the loss function $L(.)$. In other words, we weight the loss according to the size of the classes seen at time t. Thus, the loss in Eq.(18) is formulated as follows:

$$l(\hat{y}_t, y_t) = \begin{cases} 0 \quad if \quad \hat{y}_t = y_t \\ \frac{1}{s(y_t)} \quad otherwise \end{cases} \tag{47}$$

where $s(y_t)$ represents the importance of the class. It is proportional to the number of samples from a class y_t. To consider the dynamic nature of the data, we use a forgetting factor instead of counting all the samples seen so far:

$$\boldsymbol{nb} = fr * \boldsymbol{nb} + \mathbf{1}_{y_t} \tag{48}$$

where fr is the forgetting factor empirically set to 0.99, $\boldsymbol{nb}$ is a vector whose elements are the size of discovered classes, $\mathbf{1}_{y_t}$ is a vector whose elements are zeros except for the element indexed by y_t which is equal to 1.

$$s(y_t) \propto \begin{cases} \boldsymbol{nb}_{y_t} & \forall y_t \in C_{t-1} \\ 1 & y_t \quad \text{is a new class} \end{cases} \tag{49}$$

where C_{t-1} is the set of discovered classes at time $t-1$.

All the parameters of the estimator model excluding α_1 are set to the same values as in the previous section (see Tab. 3). Because α_1 controls the prior over the classes, it has impact on the class discovery performance. The model tends to put high probability on the emergence of new classes when α_1 is high (See Eq.(34)). We studied its effect on each dataset. So, it is set to 0.5 for Pageblocks, Shuttle, Thyroid and KDD and 0.4 for Forest and Digits. We can see that the value of α_1 for Forest and Digits datasets is less than the others. Such a difference can be interpreted as a result of the less unbalance classes in the Forset and Digits datasets compared to Pageblocks, Shuttle, Thyroid and KDD datasets.

5.3.2. ***Performance Analysis***

In these experiments, we follow the same setting as in the competitors [52, 36], where the maximum number of queries is set to 150 instances (SAL takes the budget ratio as input).

The results of the experiments are shown in Tab.5, Tab.6 and Fig. 7. Table 5 presents the number of classes discovered by the different methods. Table 6 presents the average class accuracy ACA achieved using the different methods. Figures 7 comprises four sub-figures; each one shows the discrepancy between ACA of each method and the highest ACA among the other methods. The discrepancy is negative when the method does not have the highest ACA among all methods.

The results show that SAL provides a comparable class discovery performance compared to the competitors. SAL was able to discover all the classes for Pageblocks, Thyroid and Forest datasets. For the Shuttle, KDD and Digits data,

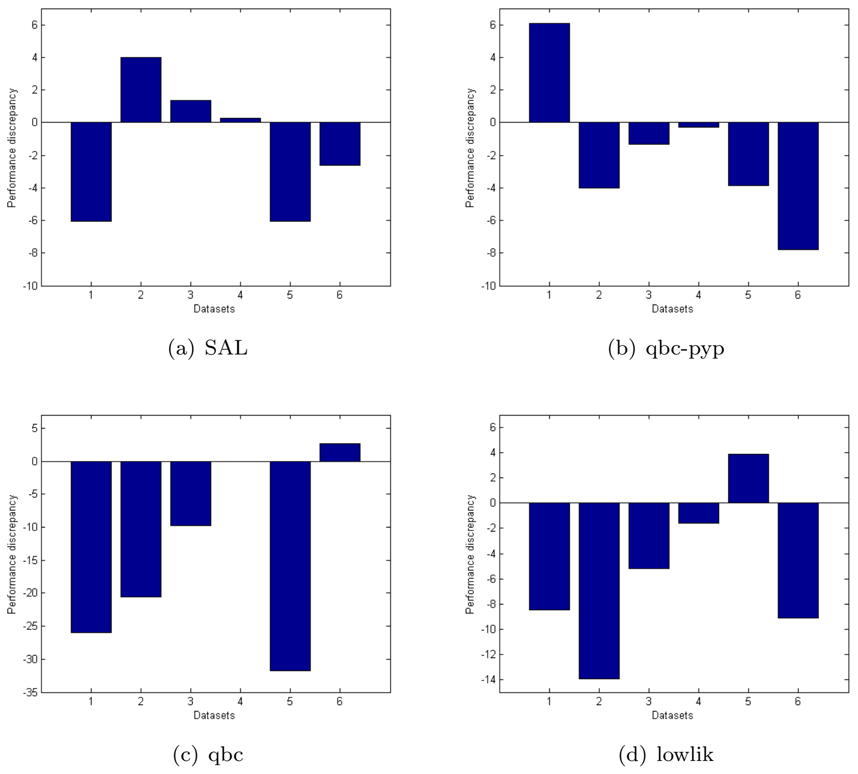
Table. 5 Number of classes discovered by different methods

No	Datasets	N_c	$lowlik$	qbc	$qbc - pyp$	SAL
1	Pageblocks	5	4.72	3.42	5	5
2	Shuttle	7	3.72	3.28	5	5.75
3	Thyroid	3	2.92	2.68	**3**	**3**
4	Forest	7	7	7	7	7
5	KDD	10	**9.76**	3.32	8.71	8
6	Digits	10	8.84	8.84	7.76	8.5

Table. 6 Average class accuracy achieved using different methods

No	Datasets	$lowlik$	qbc	$qbc - pyp$	SAL
1	Pageblocks	63.23	45.79	**71.72**	65.65
2	Shuttle	45.46	38.87	55.49	**59.42**
3	Thyroid	54.62	50.07	58.45	**59.8**
4	Forest	57.13	58.68	58.45	**58.71**
5	KDD	**51.21**	19.42	47.35	45.14
6	Digits	48.94	**58.04**	50.27	55.42
	Overall	53.93	45.14	56.95	**57.36**

around 5.75 oout of 7, 8 out of 10 and 8.5 out of 10 classes are discovered respectively (see Tab.5). SAL's average class accuracy is the best on the Shuttle, Forest and Thyroid datasets which may be explained by the fact that the proportion of the smallest classes are higher than the others (except for Shuttle which has relatively high entropy). That might be a result of SAL consideration of the data density which has shown good classification performance in the previous section. As for Digits, SAL has the second best results among the competitors. SAL has the third best result on KDD. This result may be explained by the fact that the data is severally unbalanced (percentage of the rarest class is around 0.04% and entropy is 0.17). On the other hand, SAL essentially selects the data instances that should reduce the error regardless of their classes percentage. SAL takes the density of the data samples into account, thus, instances from very rare classes might be deemed as outliers even if they are well isolated in the feature space. Nevertheless, SAL has the best results on three datasets (Shuttle, Thyroid and Forest); while each competitor gives the best result on one dataset (see Fig. 7).



(a) SAL (b) qbc-pyp

(c) qbc (d) lowlik

Figure. 7 Comparison of the class discovery performance

To sum up, the reason why SAL's class discovery performance is not the best on all datasets can be related to the fact that SAL avoids outliers. This avoidance along with the *sampling bias* mechanism have led to strong classification performance. Nevertheless, the class discovery performance is comparable to to that of strong competitors which some of them are specialized for detecting the outliers as novel classes. It can be clearly seen from Tab. 6 and Fig. 7 that SAL results are more consistent across different datasets compared to the competitors. Indeed, SAL has the best results on the majority of datasets and comparable results on the rest whilst each competitor performs well on a specific dataset. SAL can be reliably used for novelty detection tasks for datasets which are moderately unbalanced. Such datasets can be seen in applications where there are many normal and abnormal classes rather than one big normal class and many rare abnormal classes like the case of KDD.

6. Conclusion and future work

We proposed an active learning algorithm for data streams capable of dealing with data streams challenges: *infinite length*, *concept drift* and *concept evolution*. SAL labels samples that reduce the future expected error in a completely online setting. It also tackles the *sampling bias* problem of active learning. Experimental results on real-world data showed the limitation of the proposed approach regarding class discovery when applying to highly unbalanced datasets. However, the main goal of the proposed algorithm is to perform classification with unknown number of classes. Furthermore, its class discovery performance is comparable to the state-of-the-art and even better when the classes are not severely unbalanced. In future work, we will investigate the problem of class discovery from severely unbalanced datasets. We will also develop stronger mechanisms for novelty detection for noisy data streams.

Acknowledgment

A. Bouchachia was fully supported by the European Commission under the Horizon 2020 Grant 687691 related to the project: *PROTEUS: Scalable Online Machine Learning for Predictive Analytics and Real-Time Interactive Visualization*, whereas S. Mohamad was partly sponsored by the same project.

References

[1] G. Hulten, L. Spencer, P. Domingos, Mining time-changing data streams, in: the 7th ACM SIGKDD int. conf. on knowl. disc. and data mining, ACM, 2001, pp. 97–106.

[2] Y. Yang, X. Wu, X. Zhu, Combining proactive and reactive predictions for data streams, in: the 11th ACM SIGKDD int. conf. on knowl. disc. and data mining, ACM, 2005, pp. 710–715.

[3] C. C. Aggarwal, Data streams: models and algorithms, Vol. 31, Springer Science & Business Media, 2007.

[4] M. M. Gaber, A. Zaslavsky, S. Krishnaswamy, Mining data streams: a review, ACM sigmod record 34 (2) (2005) 18–26.

[5] P. Domingos, G. Hulten, Mining high-speed data streams, in: the 11th ACM SIGKDD int. conf. on knowl. disc. and data mining, ACM, 2000, pp. 71–80.

[6] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM computing surveys (CSUR) 46 (4) (2014) 44.

[7] M. Sayed-Mouchaweh, Handling concept drift, in: learning from data streams in dynamic environments, Springer, 2016, pp. 33–59.

[8] L. Hartert, M. Sayed-Mouchaweh, Dynamic supervised classification method for online monitoring in non-stationary environments, neurocomputing 126 (2014) 118–131.

[9] A. Bouchachia, C. Vanaret, Gt2fc: an online growing interval type-2 self-learning fuzzy classifier, IEEE trans. on fuzzy systems 22 (4) (2014) 999–1018.

[10] C. C. Aggarwal, J. Han, J. Wang, P. S. Yu, A framework for on-demand classification of evolving data streams, IEEE trans. on knowl. and data engin. 18 (5) (2006) 577–589.

[11] A. Bouchachia, Incremental learning with multi-level adaptation, neurocomputing 74 (11) (2011) 1785–1799.

[12] Y. Sun, K. Tang, L. L. Minku, S. Wang, X. Yao, Online ensemble learning of data streams with gradually evolved classes, IEEE trans. on knowl. and data engin. 28 (6) (2016) 1532–1545.

[13] M. M. Masud, Q. Chen, L. Khan, C. Aggarwal, J. Gao, J. Han, B. Thuraisingham, Addressing concept-evolution in concept-drifting data streams, in: IEEE 10th int. conf. on data mining, IEEE, 2010, pp. 929–934.

[14] M. M. Masud, J. Gao, L. Khan, J. Han, B. Thuraisingham, Classification and novel class detection in concept-drifting data streams under time constraints, IEEE trans. on knowl. and data engin. 23 (6) (2011) 859–874.

[15] T. Al-Khateeb, M. M. Masud, L. Khan, C. Aggarwal, J. Han, B. Thuraisingham, Stream classification with recurring and novel class detection using class-based ensemble, in: IEEE 12th int. conf. on data mining, IEEE, 2012, pp. 31–40.

[16] M. A. Pimentel, D. A. Clifton, L. Clifton, L. Tarassenko, A review of novelty detection, signal processing 99 (2014) 215–249.

[17] R. P. Adams, D. J. MacKay, Bayesian online changepoint detection, arXiv preprint arXiv:0710.3742.

[18] S. Dasgupta, Two faces of active learning, theoretical computer science 412 (19) (2011) 1767–1781.

[19] D. A. Cohn, Z. Ghahramani, M. I. Jordan, Active learning with statistical models, journal of artificial intelligence research.

[20] N. Roy, A. McCallum, Toward optimal active learning through monte carlo estimation of error reduction, ICML, Williamstown (2001) 441–448.

[21] S. Vijayanarasimhan, K. Grauman, Multi-level active prediction of useful image annotations for recognition, in: NIPS, 2009, pp. 1705–1712.

[22] S. Mohamad, H. Bouchachia, M. Sayed-Mouchaweh, A bi-criteria active learning algorithm for dynamic data streams, IEEE trans. on neur. netw. and learn. syst., in press, 2016.

[23] Y. W. Teh, Dirichlet process, in: encyclopedia of machine learning, Springer, 2011, pp. 280–287.

[24] J. Sethuraman, A constructive definition of dirichlet priors, statistica sinica (1994) 639–650.

[25] C. M. Carvalho, H. F. Lopes, N. G. Polson, M. A. Taddy, et al., Particle learning for general mixtures, bayesian analysis 5 (4) (2010) 709–740.

[26] P. Fearnhead, Particle filters for mixture models with an unknown number of components, statistics and computing 14 (1) (2004) 11–21.

[27] W. Chu, M. Zinkevich, L. Li, A. Thomas, B. Tseng, Unbiased online active learning in data streams, in: the 17th ACM SIGKDD int. conf. on KDDM, ACM, 2011, pp. 195–203.

[28] A. Beygelzimer, S. Dasgupta, J. Langford, Importance weighted active learning, in: the 26th ann. int. conf. on mach. learn., ACM, 2009, pp. 49–56.

[29] M. Baena-Garcıa, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, R. Morales-Bueno, Early drift detection method, in: Fourth international workshop on knowledge discovery from data streams, Vol. 6, 2006, pp. 77–86.

[30] Q. Da, Y. Yu, Z.-H. Zhou, Learning with augmented class by exploiting unlabeled data, in: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, AAAI Press, 2014, pp. 1760–1766.

[31] D. H. Widyantoro, J. Yen, Relevant data expansion for learning concept drift from sparsely labeled data, IEEE trans. on knowl. and data engineering 17 (3) (2005) 401–412.

[32] M. M. Masud, C. Woolam, J. Gao, L. Khan, J. Han, K. W. Hamlen, N. C. Oza, Facing the reality of data stream classification: coping with scarcity of labeled data, knowledge and information systems 33 (1) (2012) 213–244.

[33] P. Lindstrom, S. J. Delany, B. Mac Namee, Handling concept drift in text data stream constrained by high labelling cost.

[34] I. Zliobaite, A. Bifet, B. Pfahringer, G. Holmes, Active learning with drifting streaming data, IEEE trans. on neur. netw. and learn. syst. 25 (1) (2014) 27–39.

[35] D. D. Lewis, J. Catlett, Heterogeneous uncertainty sampling for supervised learning, in: the 11th int. conf. on mach. lear., 1994, pp. 148–156.

[36] C. C. Loy, T. M. Hospedales, T. Xiang, S. Gong, Stream-based joint exploration-exploitation active learning, in: IEEE conf. on comp. vis. and pat. rec. (CVPR'12), IEEE, 2012, pp. 1560–1567.

[37] J. Pitman, M. Yor, The two-parameter poisson-dirichlet distribution derived from a stable subordinator, the annals of probability (1997) 855–900.

[38] H. S. Seung, M. Opper, H. Sompolinsky, Query by committee, in: the 5th ann. work. on comput. learn. theo., ACM, 1992, pp. 287–294.

[39] M. M. Masud, J. Gao, L. Khan, J. Han, B. Thuraisingham, Classification and novel class detection in data streams with active mining, in: advances in knowledge discovery and data mining, Springer, 2010, pp. 311–324.

[40] S. Khoshrou, J. S. Cardoso, L. F. Teixeira, Learning from evolving video streams in a multi-camera scenario, Machine Learning 100 (2-3) (2015) 609–633.

[41] Z. S. Abdallah, M. M. Gaber, B. Srinivasan, S. Krishnaswamy, Anynovel: detection of novel concepts in evolving data streams, Evolving Systems 7 (2) (2016) 73–93.

[42] A. Beygelzimer, D. J. Hsu, J. Langford, T. Zhang, Agnostic active learning without constraints, in: Advances in Neural Information Processing Systems, 2010, pp. 199–207.

[43] H. T. Nguyen, A. Smeulders, Active learning using pre-clustering, in: Proceedings of the twenty-first international conference on Machine learning, ACM, 2004, p. 79.

[44] J. Sethuraman, A constructive definition of dirichlet priors, Tech. rep., DTIC Document (1991).

[45] R. M. Neal, Bayesian mixture modeling, in: maximum mntropy and bayesian methods, Springer, 1992, pp. 197–211.

[46] C. E. Rasmussen, The infinite gaussian mixture model., in: NIPS, Vol. 12, 1999, pp. 554–560.

[47] T. S. Ferguson, A bayesian analysis of some nonparametric problems, the annals of statistics (1973) 209–230.

[48] D. Blackwell, J. B. MacQueen, Ferguson distributions via pólya urn schemes, the annals of statistics (1973) 353–355.

[49] Y. W. Teh, Dirichlet process, in: encyclopedia of machine learning, Springer, 2010, pp. 280–287.

[50] R. M. Neal, Markov chain sampling methods for dirichlet process mixture models, journal of computational and graphical statistics 9 (2) (2000) 249–265.

[51] A. Bifet, G. Holmes, R. Kirkby, B. Pfahringer, MOA: Massive online analysis, journ. of mach. learn. res. 11 (May) (2010) 1601–1604.

[52] C. C. Loy, T. Xiang, S. Gong, Stream-based active unusual event detection, in: asian conf. on comp. vis., Springer, 2010, pp. 161–175.

[53] A. Asuncion, D. Newman, UCI machine learning repository (2007).

[54] M. B. Harries, C. Sammut, K. Horn, Extracting hidden context, Machine learning 32 (2) (1998) 101–126.

[55] E. Ikonomovska, J. Gama, S. Džeroski, Learning model trees from evolving data streams, Data mining and knowledge discovery 23 (1) (2011) 128–168.

[56] T. M. Hospedales, S. Gong, T. Xiang, Finding rare classes: Active learning with generative and discriminative models, IEEE trans. on knowl. and data engin. 25 (2) (2013) 374–386.

[57] J. Gama, P. Medas, G. Castillo, P. Rodrigues, Learning with drift detection, in: Brazilian Symposium on Artificial Intelligence, Springer, 2004, pp. 286–295.

Appendix A. Compute Equ. (24):

- If z_t refers to a new component:

$$p(\boldsymbol{x}_t|z_t, X_{t-1}) = \int_{\boldsymbol{\theta}} p(\boldsymbol{x}_t|\boldsymbol{\theta})p(\boldsymbol{\theta}|G_0) = t_{v_1}(\boldsymbol{x}_t|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \tag{A.1}$$

where t refer to student's t-distribution which we end up with as a result of using a conjugate prior (i.e., the Normal Inverse Wishart prior) over the normal distribution parameter $\boldsymbol{\theta}$.

$$\boldsymbol{\mu}_1 = \boldsymbol{\mu}_0 \tag{A.2}$$

$$\boldsymbol{\Sigma}_1 = \frac{\boldsymbol{\Sigma}_0(k_0+1)}{k_0(v_0-d+1)} \tag{A.3}$$

$$v_1 = v_0 - d + 1 \tag{A.4}$$

where d is the dimension of the data.

- If z_t refers to an already seen component:

$$\begin{aligned} p(\boldsymbol{x}_t|z_{1:t}, X_{t-1}) &= \int_{\boldsymbol{\theta}} p(\boldsymbol{x}_t|\boldsymbol{\theta})p(\boldsymbol{\theta}|\boldsymbol{s}_{z_t,t-1}(z_{1:t-1}), n_{z_t,t-1}) \\ &= t_{v_2}(\boldsymbol{x}_t|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) \end{aligned} \tag{A.5}$$

$$\boldsymbol{\mu}_2 = \frac{k_0}{k_0+n_{z_t,t-1}}\boldsymbol{\mu}_0 + \frac{n_{z_t,t-1}}{k_0+n_{z_t,t-1}}\boldsymbol{su}_{z_t,t-1} \tag{A.6}$$

$$\begin{aligned} \boldsymbol{\Sigma}_2 = & \frac{1}{(k_0+n_{z_t,t-1})(v_0+n_{z_t,t-1}-d+1)} \\ & \big(\boldsymbol{\Sigma}_0 + \boldsymbol{sc}_{z_t,t-1} + \frac{k_0 n_{z_t,t-1}}{k_0+n_{z_t,t-1}}(\boldsymbol{su}_{z_t,t-1}-\boldsymbol{\mu}_0) \\ & (\boldsymbol{su}_{z_t,t-1}-\boldsymbol{\mu}_0)^T\big)(k_0+n_{z_t,t-1}+1) \end{aligned} \tag{A.7}$$

$$v_2 = v_0 + n_{z_t,t-1} - d + 1 \tag{A.8}$$

Appendix B. Compute the first term of Eq. (33):

Given z_t, y_t is independent of the observations $\boldsymbol{x}_{1:t-1}$. Hence,

$$p(y_t|z_t, D_{t-1}) = p(y_t|z_t, y_{t-1}) \tag{B.1}$$

As z_t selects the stick breaking component generating y_t, the distribution of y_t depends only on the label of the data assigned to components z_t. By marginalizing the selected stick breaking component the same way as in Eq.(11), we end up with the following equations:

$$p(y_t|z_t, D_{t-1}) = \begin{cases} \frac{n_{z_t,y_t,t}}{\alpha_1 + n'_{z_t,t} - 1} \propto n'_{z_t,y_t,t} & y_t \text{ is an existing class} \\ \frac{\alpha_1}{\alpha_1 + n'_{z_t,t} - 1} \propto \alpha_1 & y_t \text{ is a new class} \end{cases} \tag{B.2}$$