Bournemouth University **IMT Lille Douai**

Active Learning for Data Streams

by

Saad Mohamad

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

in the

Department of Computing at Bournemouth University and
Department of Informatique et Automatique at IMT Lille Douai

October 2017

Ask not about things which, if made plain to you, may cause you trouble

Abstract

With the exponential growth of data amount and sources, access to large collections of data has become easier and cheaper. However, data is generally unlabelled and labels are often difficult, expensive, and time consuming to obtain. Two learning paradigms have been used by machine learning community to diminish the need for labels in training data: semi-supervised learning (SSL) and active learning (AL). AL is a reliable way to efficiently building up training sets with minimal supervision. By querying the class (label) of the most interesting samples based upon previously seen data and some selection criteria, AL can produce a nearly optimal hypothesis, while requiring the minimum possible quantity of labelled data. SSL, on the other hand, takes the advantage of both labelled and unlabelled data to address the challenge of learning from a small number of labelled samples and large amount of unlabelled data. In this thesis, we borrow the concept of SSL by allowing AL algorithms to make use of redundant unlabelled data so that both labelled and unlabelled data are used in their querying criteria.

Another common tradition within the AL community is to assume that data samples are already gathered in a pool and AL has the luxury to exhaustively search in that pool for the samples worth labelling. In this thesis, we go beyond that by applying AL to data streams. In a stream, data may grow infinitely making its storage prior to processing impractical. Due to its dynamic nature, the underlying distribution of the data stream may change over time resulting in the so-called *concept drift* or possibly emergence and fading of classes, known as *concept evolution*. Another challenge associated with AL, in general, is the *sampling bias* where the sampled training set does not reflect on the underlying data distribution. In presence of *concept drift*, *sampling bias* is more likely to occur as the training set needs to represent the underlying distribution of the evolving data. Given these challenges, the research questions that the thesis addresses are: can AL improve learning given that data comes in streams? Is it possible to harness AL to handle changes in streams (i.e., concept drift and concept evolution by querying selected samples)? How can *sampling bias* be attenuated, while maintaining AL advantages? Finally, applying AL for sequential data steams (like time series) requires new approaches especially in the presence of *concept drift* and *concept evolution*. Hence, the question is how to handle *concept drift* and *concept evolution* in sequential data online and can AL be useful in such case?

In this thesis, we develop a set of stream-based AL algorithms to answer these questions in line with the aforementioned challenges. The core idea of these algorithms is to query samples that give the largest reduction of an expected loss function that measures the learning performance. Two types of AL are proposed: *decision theory* based AL whose losses involve the prediction error and *information theory*

based AL whose losses involve the model parameters. Although, our work focuses on classification problems, AL algorithms for other problems such as regression and parameter estimation can be derived from the proposed AL algorithms. Several experiments have been performed in order to evaluate the performance of the proposed algorithms. The obtained results show that our algorithms outperform other state-of-the-art algorithms.

Acknowledgements

This PhD thesis has been co-funded by Bournemouth University and IMT Lille Douai (Ecole Mines-Tlcom, IMT-Universit de Lille) under the supervision of Prof. Hamid Bouchachia and Prof. Moamar Sayed-Mouchaweh. My time at Bournemouth and Douai has been influenced and guided by a number of people to whom I am deeply indebted. Without their help, friendship and support, this thesis would likely never have seen the light of day.

I would like to thank the members of my thesis committee, Prof. Abdelhamid Bouchachia and Prof. Moamar Sayed-Mouchaweh for their insights and guidance. I feel most fortunate to have had the opportunity to receive their support. My supervisors, Prof. Abdelhamid Bouchachia and Prof. Moamar Sayed-Mouchaweh, have had the greatest impact on my academic development during my thesis. They have been a tremendous mentor, collaborator and friends, providing me with invaluable insights about research and academic skills in general. They both taught me how to do research, how to ask the right questions and how to answer them, how to have a clear vision and strategy. Their wisdom, creativity, integrity, humility, and generosity will continue to inspire me. I was indeed fortunate to have them as my supervisors. I feel exceedingly privileged to have had their guidance and I owe them a great many heartfelt thanks.

My deepest gratitude and appreciation is reserved for my family. Without their constant love, support and encouragement, I would never have been able to produce this thesis. I dedicate this thesis to them.

Contents

List of Figures

List of Tables

Abbreviations

AL	**A**ctive **L**earning
MQS	**M**embership **q**uery **S**ynthesis
PPP	**P**oole-based **S**elective **S**ampling
SSS	**S**tream-based **S**elective **S**ampling
QBC	**Q**uery-**B**y-**C**ommittee
HAR	**H**uman **A**ctivity **R**ecognition
BAL	**B**i-criteria **A**ctive **L**earning
GGMM	**G**rowing **G**aussian **M**ixture **M**odel
MAP	**M**aximum-**A**-**P**osteriori
MLE	**M**aximum **L**ikelihood **E**stimation
GMM	**G**aussian **M**ixture **M**odel
EM	**E**xpectation **M**aximisation
UAL	**U**ni-criterion **A**ctive **L**earning
SAL	**S**tream-based **A**ctive **L**earning
DP	**D**irichlet **P**rocess
DPMM	**D**irichlet **P**rocess **M**ixture **M**odel
NIW	**N**ormal-**I**nverse-**W**ishart
AA	**A**verage **A**ccuracy
ACA	**A**verage **C**lass **A**ccuracy
CRBM	**C**onditional **R**estricted **B**oltzmann **M**achine
OSC	**O**nline **S**emi-supervised **C**lassifier
BSAL	**B**ayesian **S**tream-based **A**ctive **L**earning
DL	**D**eep **L**earning
DT	**D**ecision **T**ree
SVM	**S**upport **V**ector **M**achine

Symbols

Symbol	Description
X	set of data samples
X_U	set of unlabelled data
X_L	set of labelled data
Y_L	set of labels associated with X_L
X_t	set of data samples seen up to time t
X_{L_t}	set of labelled data samples seen up to time t
X_{U_t}	set of unlabelled data samples seen up to time t
Y	set of data labels
Y_{L_t}	set of labels associated with X_{L_t}
D	$=(X, Y)$
D_L	$=(X_L, Y_L)$
D_t	$= (X_t, Y_{L_t})$
D_{L_t}	$= (X_{L_t}, Y_{L_t})$
C_t	the set of classes discovered up until time t
y_t	true data label at time t
$\hat{y}_t$	predicted data label at time t
$\boldsymbol{x_t}$	data input at time t
R	risk
$\hat{R}$	expected Risk
$R(\boldsymbol{\theta})$	risk given $\boldsymbol{\theta}$ (classifier parameter)
$\hat{R}_n(\boldsymbol{\theta})$	empirical risk over n samples
$\hat{R}_{X_U}(\boldsymbol{\theta})$	empirical risk over unlabelled data X_U
$L(.)$	loss function

Dedicated to my parents, sisters and brother

Chapter 1

Introduction

In this chapter, we present a general background on machine learning, its importance and paradigms; in particular online learning and active learning. We introduce the challenges of online learning from data streams. We also discuss the relevance of active learning as well as the main approaches, with particular focus on stream-based active learning. A literature review on this later is presented, where the state-of-the-art competitors of our new algorithms proposed in this thesis are described. Finally, the research questions, contributions and structure of the thesis are presented.

The organization of this chapter is as follows. Section 1.1 is a background. Section 1.2 introduces online learning and its challenges. Section 1.3 reviews active learning with focus on the state-of-the-art stream-based active learning algorithms. Section 1.4 presents the aim and objectives of the thesis. Section 1.5 discusses the contributions of the thesis. The structure of the thesis is presented in Sec. 1.6. A list of the publications on which the thesis is based is shown in Sec. 1.7.

1.1 Background

Our digital universe is rapidly growing. According to an updated digital universe study [4], in 2020, the amount of digital data produced will exceed 40 zettabytes which is the equivalent of $5,200$ GB of data for every man, woman and child on earth. The abundance of data has been exploited and generated a major impact in many fields of application like manufacturing, health-care, media, sports, science

and more. In order to exploit such data, for the sake of decision making, machine learning (ML) techniques can be used.

ML has a long history in applied mathematics and statistics. ML community has been exceedingly creative at taking existing ideas across many fields and mixing and matching them to solve problems in different domains [5]. From statistical inference point of view, ML can be categorised into two views: *Frequentist view* and *Bayesian view*. Bayesian inference is performed with probabilistic parameters (model's parameters) and fixed data, while Frequentist inference is performed with fixed parameters and random data samples. In the Frequentist approach, unknown set of parameters is treated as having fixed unknown values. It cannot be envisioned as random variables. In contrast, the Bayesian approach does allow probabilities to be associated with the unknown parameters. In ML literature, we can distinguish the following paradigms.

- Supervised learning (SL).
- Unsupervised learning (UL).
- Semi-supervised learning (SSL).
- Reinforcement learning (RL).
- Active learning (AL).

SL is about learning a mapping function, $f : X \rightarrow Y$, between a set of input instances and their corresponding output (e.g., labels of classes). The training set $D = (X, Y)$ is used to find the mapping (model) with the least expected loss. If the output is categorical, SL is a classification task; otherwise it is a regression task. Classification is a supervised learning task where the labels that we wish to predict take discrete values.

In UL, we are not given any output to use for training $D = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\}$. UL encompasses clustering where we try to find groups of data instances that are similar to each other. Loss function here is sometimes called *objective function* (e.g., K-Means algorithm).

SSL, as its name implies, falls between UL and SL. It can be thought of as a class of supervised learning that also makes use of unlabelled data for training. SSL makes use of one of the following assumptions.

- Smoothness assumption: points which are close to each other are more likely to share the same class label.
- Cluster assumption: the data tends to form discrete clusters and points in the same cluster are more likely to share a label (this is special case of the smoothness assumption)
- Manifold assumption: the data lies approximately on a manifold of much lower dimension than the input space.

In RL, the output is a reward given by the environment according to its state as a result of certain provided input (action) by the learner (agent). Thus, given a state $\boldsymbol{s}$ and an action a, the environment feedback is a reward function $\mathrm{R}(\boldsymbol{s}, a)$. Unlike SL, where we receive inputs and outputs from the environment and tries to learn a model, RL agent interacts with the environment and try to learn a policy f that maximizes cumulative rewards, where $f : A \times S \rightarrow R$ and A, S, R are the set of actions states and rewards respectively.

AL is about interacting with an oracle to query the label of selected data instances. In contrast to RL, the actions space is just queries and the reward is the true label. Once obtained, the data instances and their labels are used to train SL algorithms. We should point out that AL can be used to ask questions of different form than querying certain data [6, 7]. However in this research work, we consider AL that queries data labels for classification purposes. On the other hand, passive learning is a terminology that will be used for any learner that receives passively the information provided (class label in our case).

ML algorithms can also be classified according to the way data is processed. We can distinguish two paradigms: online learning and offline learning. In offline learning, once the training phase is exhausted, the model is no more capable to learn further knowledge from new instances, that is, it is not able to self-update in the future. On the contrary, in online setting, data comes in a form of streams and the model must predict and adapt online, that is, it keeps learning over time. For example, a spam detector receives online message and the task is to predict whether the messages are spam or not. The quality of the model's prediction is assessed by a loss function that measures the discrepancy between the predicted label and the true one. Then, an adaptation of the model is applied if needed.

Since this thesis is about online active learning, in the following, we provide more details related to online learning and active learning.

1.2 Online Learning

Online learning has been studied in several research fields including game theory, information theory, data mining and machine learning [8–13]. The concept of *online learning* may be interpreted differently. Any online learning algorithm must be able to learn from data streams. The characteristics of data stream include [14]:

- Continuous flow (data samples arrive one after another.
- Huge data volumes (possibly of an infinite length).
- Rapid arrival rate (relatively high with respect to the processing power of the system).
- Susceptibility to change (data distributions generating examples may change over time).

Therefore, an online learning algorithm should fulfil the following requirements:

- The algorithm can access data only once and sequentially,
- The time and space complexity of the algorithm must not scale with the number of instances,
- The algorithm should be able to self-adapt.

Incremental learning [15–18] is sometime mistaken for online learning, but it cannot fulfil all the listed requirements. Although, it processes one instance or a mini-batch of instances at each iteration, it requires to cycle over the whole data many times. Besides, incremental learning algorithms cannot adapt to changes. Online learning presented in [8, 9, 19], views data samples as the product of some unknown and unspecified mechanism that could be deterministic, stochastic, or even adversarial adaptive to the algorithm behaviour. In this thesis, we focus on online learning from data streams with non-adversarial actions. Many such algorithms can be derived from their offline counterparts. To do so, algorithms require

adjustment to make online update and mechanisms to forget outdated data and adapt to changes [20].

Some algorithms can be naturally updated online (e.g., k-nearest neighbours and naive-Bayes) and only require mechanisms to deal with changes [20–22]. Others, like decision trees, require, in addition to mechanisms for dealing with changes, substantial adjustments to make online update [23, 24]. Details on mechanisms and techniques for online learning methods can be found in [20, 25].

In this thesis, we adopt sequential Bayesian approaches for learning from non-stationary data streams [26–29]. That is, the sequential nature of Bayes theorem is exploited to recursively update some models. Forgetting factors [30] are employed to decay the contributions from old data points in favour of new better ones.

As mentioned earlier, data streams are susceptible to changes over time resulting in the so-called *concept drift* or the possible emergence of classes, known as *concept evolution*. *Concept drift* occurs when the statistical properties of the target variable, which the model is trying to predict, change over time in unforeseen ways [21]. More precisely, we can distinguish two types of drifts:

- Real concept drift refers to changes in the conditional distribution of the targets (labels) $p(y|\boldsymbol{x})$ given the observations. Such changes can happen either with or without change in the marginal distribution of the incoming data $p(\boldsymbol{x})$.
- Virtual drift occurs if the distribution of the incoming data changes $p(\boldsymbol{x})$ without affecting $p(y|\boldsymbol{x})$. Hence, in both cases the joint distribution $p(\boldsymbol{x}, y)$ changes.

Concept evolution occurs when new classes emerge [31–33]. Emergence of new classes has been studied in the context of novelty detection where classification models are unable to detect the novel class until they are trained with labelled instances of it. In this thesis, *concept evolution* is defined as emergence of new unseen classes as the data streams evolve over time.

Finally, we give an illustrative example of a simple online learning algorithm called perceptron. The perceptron [34–36] is perhaps the first and simplest online learning algorithm. Perceptron uses a class of linear separators in the input space,

where the prediction of the data label depends on the side of hyperplane the data instance falls in. If the predicted label and the true label are different, the algorithm adjusts itself.

$$h(\boldsymbol{x}) = \begin{cases} 1 & \boldsymbol{w}.\binom{1}{\mathbf{x}} > 0 \\ -1 & \text{otherwise} \end{cases}$$

A simple perceptron consists of the following steps:

1. Initialise the weights $\boldsymbol{w}$.
2. Receive the input $\mathbf{x}_t$.
3. Predict the output by computing $\hat{y} = h(\mathbf{x}_t)$.
4. Update $\boldsymbol{w}$ if $y_t \neq h(\mathbf{x}_t)$: $\boldsymbol{w_{t+1}} = \boldsymbol{w_t} + \alpha y_t \mathbf{x}_t$, where α is a learning rate and y_t is the true output.

1.3 Active Learning

In the context of classification, AL allows to label some selected data samples by an expert (oracle or annotator) according to some selection criteria. The overall goal of AL is to provide, at least, the same performance as that of passive learning while using less labelled examples.

1.3.1 Querying Criteria

There exist three main approaches of AL [37]: *membership query synthesis (MQS)*, *pool-based selective sampling (PSS)* and *stream-based selective sampling (SSS)*. According to MQS, the learner generates new data samples from the feature space that will be queried. However, labelling such arbitrary instances may be impractical, especially if the oracle is a human annotator as we may end up querying instances [38] that are hard to explain. PSS is the most popular AL method, according to which the selection of instances is made by exhaustively searching in a large collection of unlabelled data gathered at once in a pool. Here, PSS evaluates and ranks the entire collection before selecting the best query. On the other hand,

SSS scans through the data sequentially and makes query decisions individually. In the case of data streams, PSS is not appropriate, especially when memory or processing power is limited. It also assumes that the pool of data is stationary and uses the whole dataset. This will delay the adaptation and waste the resources. SSS, instead, adapts the classifier in real-time leading to fast adaptation.

While there have been many AL studies on the offline variant, only few ones have investigated the online setting. Most of the AL sampling criteria have been first introduced for offline setting, then adapted to work online. Authors in [39] introduce one of the most general frameworks for measuring informativeness, *label uncertainty* sampling criterion, where the queried instances are those which the model is most uncertain about their label. It has been since then used in many successful offline AL algorithms [40–44].

Another popular AL sampling criterion framework is the *query-by-committee* [45]. Here, a committee of models trained on the same dataset are maintained. They represent different hypotheses. The data label about which they most disagree is queried. To use the *query-by-committee* framework, one must construct a committee of models and have some measure of disagreement. *Query-by-committee* has shown both theoretical and empirical efficiency in several offline AL studies[46–48]. *Density-based* is another AL sampling criterion that differs from uncertainty and *query-by-committee* in that it uses unlabelled data for measuring the instance informativeness [44]. *Density-based* criterion assumes that the data instances in dense regions are more important. Many studies have involved *density-based* criterion by combining it with other criteria like uncertainty sampling [49] and *Query-by-committee* sampling [50].

Authors in [51] point out that there is no individual AL method superior to the others and that their combination could produce better results. A reasonable approach to combine an ensemble of active learning algorithms might be to evaluate their individual performance and dynamically switch to the best performer so far [51]. This idea of rewarding is rooted in RL [52]. There exists a classical trade-off in RL called the exploration/exploitation trade-off which can be explained as follows. If we have already found a way to act in the domain that gives a reasonable reward, then is it better to continue exploiting the explored domain or should we try to explore a new domain in the hope that it may improve the reward [53]. This concept dates back to the study of bandit problems in 1930s which are the most basic examples of sequential decision problems with an exploration-exploitation

trade-off. Because of the strong theoretical foundation of bandit problems, they are exploited by many authors to formulate AL [51, 54, 55]. However, none of them considers SSS active learning.

1.3.2 Online Active Learning

Online AL methods for data streams in presence of drift have been dealt with using batch-based learning, where data is divided into batches [56, 57]. These methods assume that the data is stationary within each batch and then PSS AL strategies are applied. Authors in [58] use a sliding window approach, where the oldest instances are discarded. *Label uncertainty* is, then, used to label the most informative instances within each new batch. In [59] an online approach is compared against a batch based approach, finding that both have similar accuracy, but the batch-based one requires more resources. Moreover, the batch-based approach requires to specify the size of the batch. Some studies consider fixed size; others use variable one. However, in both cases, more memory than the one with the online approach is needed. Another issue is that, in general, batch-based approaches cannot learn from the most recent examples until a new batch is full. This leads to more delay when responding to changes. This delay has a negative effect leading to late recovery. All these reasons make online learning more natural and suitable for AL.

Few SSS AL studies have been proposed [1, 3, 60–62]. Methods in [60, 61] assume that the data is stationary. They cannot work in the case of evolving data streams as the models cannot perceive the change of data distribution. Thus, these methods lead to *sampling bias* [63] problems when data evolves (see Sec. 1.3.3).

Authors in [3], handle the problem of *concept evolution* by defining a non-parametric Bayesian prior on the classes using Pitman-Yor Processes [64]. However, they use *Query-by-committee* (QBC) [45] 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*. It is worth noting that although this method is stream-based AL, it memorizes all labelled samples and re-use them at each iteration for updating the model.

Online AL approaches that address the three data stream challenges: *infinite length*, *concept drift* and *concept evolution* are the rarest. Authors in [65] also deal with *concept evolution* and *concept drift*. They 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. This algorithm does not address *sampling bias* and it needs to store the data instances in different batches.

1.3.3 Sampling Bias

Sampling bias problem is, in general, associated with AL, 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. It, therefore, creates a bias towards a certain set of instances, which could become harmful. *Sampling bias* 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.

Methods in [1, 62] takes *sampling bias* problem into account. In [1], SSS is adopted using randomization to avoid bias estimation of the class conditional distribution that may result from querying. This randomization is combined with *label uncertainty* to deal with *concept drift*. However, it results in wasting resources by randomly picking data to cover the whole input space. Moreover, randomization has no interaction when drift occurs and it naively keeps querying randomly. In [62], *sampling bias* is studied using *importance weighting* principle to re-weight labelled data and remove the bias. *Importance weighting* principle has been theoretically proven to be effective [66]. However, the method in [62] is restricted

to binary classification. Both methods [1, 62] assume that the number of classes is fixed and known in advance, namely there is no *concept evolution*. Furthermore, they ignore the effect of data distribution and do not benefit of abundant unlabelled data samples.

1.4 Aims and Objectives

Using AL in real world scenarios is not an easy task as many constraints and side effects arise. Most of the studies have been associated with unrealistic assumptions such as:

1 The data can be re-accessed at any time.

2 The number of classes is a-priori known (no *concept evolution*).

3 The data is stationary (no *concept drift*).

4 The data stream samples are temporally independent from each others.

The main aim of this thesis is to make AL applicable in realistic real-world scenarios. Since data comes as a stream, online learning classifiers and SSS active learning algorithms should be adopted. AL for data streams with *concept drift* could suffer the problem of *sampling bias*. For example, in classification, *label uncertainty* aims at selecting the most uncertain instances that typically lie close to the classification boundary. Training the classifier on those instances is expected to adjust the boundary and achieving better classification accuracy. However, given the non-stationary aspect of data coming over time in a stream, *concept drift* may occur any time and anywhere in the feature space. Applying *label uncertainty* may result in missing the changes occurring farther from the boundary. Here, the sampled instances are biased towards the boundary rather than being representative of the underlying data distribution. The evolving nature of data streams leads to *concept evolution* which poses another challenge for using AL to learn from data streams. That is, AL algorithms should not only query informative samples, but also discover the class space. This task becomes even more challenging in the case of data involving classes of unbalanced proportions. Applying AL for sequential data steams with temporal dependency requires new approaches especially in

presence of *concept drift* and *concept evolution*. That is, how to take the temporal dependency of the data into consideration when querying. To the best of our knowledge, this work is original considering the training data as a set of sequences (with temporal dependency within sequences); while existing AL work has focused on data assumed being drawn independently and identically (iid) from some joint distribution P.

In a nutshell, the objectives of this thesis:

- To provide a uniform probabilistic approach to cope with active learning taken into account inherent phenomena.
- To investigate active learning in the context of Streaming using various querying criteria.
- To investigate decision-theory based and information-theory based active learning in the context of stream-based selective sampling.
- To develop innovative application for AL from sequential data streams.

Section 1.5 provides a more precise description of contributions in line with these objectives.

1.5 Contributions

Most AL approaches, including those mentioned in Sec. 1.3, are basically derived from the general approach of finding queries that yield the largest reduction of the expected loss. Let Ω denote the set of variables that can be fully random or include some random elements. Let L refer to the loss function. The expected loss function that AL aims to reduce can be expressed as follows:

$$R = E_{\Omega}[L(\hat{\Omega}, \Omega)] \tag{1.1}$$

where the hat refers to an estimated term. Depending on the loss functions involved, AL can be divided to two main groups: information-based and decision-based AL. These groups mainly differ in the type of loss functions used. While *decision theory* based AL relies on the prediction error, *information theory* based

AL losses involve the model parameter. The set of variables in Ω includes labelled and unlabelled data.

The contributions of the thesis can be summarised as follows:

1. We propose *decision theory* and *information theory* AL algorithms that seek to directly reduce the expected loss online while taking the data streams challenges (infinite length, *concept drift* and *concept evolution*) into account.
2. The proposed algorithms can handle *concept drift* and *concept evolution* by querying the data samples representative of them. Hence, they are changes aware.
3. Some of the proposed AL algorithms handle *sampling bias* problem.
4. The proposed AL algorithms uses both labels and unlabelled data.

In this thesis, we propose AL algorithms that query the samples which incur the highest reduction in the expected loss (Eq. (1.1)), while using both labelled (like in *uncertainty* criteria) and unlabelled data (like in the *density-based* criterion). The proposed algorithms called BAL, SAL and BSAL (see abbreviations list for full names) address the challenges of AL for data streams (i.e., *concept drift*, *concept evolution*, *sampling bias* and temporal dependency). They handle *concept drift* and *concept evolution* by querying the data samples representative of them (i.e., their characteristics). In contrast to standard techniques for handling *concept drift* and *concept evolution* [22, 32, 33, 67, 68], where only automatic detection mechanisms are applied, AL assumes that an oracle provides the true labels of data. BAL and SAL are *decision theory* based AL approaches while BSAL is an *information theory* based AL approach. The information based approach seems better at coping with *concept evolution* when classes are unbalanced (See Chap. 4). Table 1.1 sums up the contributions of the thesis showing how such contributions compare against the closest state-of-the-art algorithms based on well-defined characteristics.

1.6 Structure of the Thesis

The structure of this thesis is as follows:

Table. 1.1 Comparisons

	AL uses		Type of Data		Type of AL		Data Streams Challenges		Sampling Bias
	Labelled data	Unlabelled data	Sequential	Non-sequential	Information-based	Decision-based	Concept drift	Concept evolution	
BAL	✓	✓		✓		✓	✓		✓
SAL	✓	✓		✓		✓	✓	✓	✓
BSAL	✓	✓	✓		✓		✓	✓	
[1]	✓			✓		✓	✓		
[3]	✓			✓	✓			✓	
[60]	✓	✓		✓		✓			
[61]	✓			✓		✓			
[62]	✓			✓		✓	✓		✓

- *Chapter 2* shows how AL can be applied for classifying drifting data streams. A new online AL algorithm called BAL is proposed. BAL queries data samples contributing to the reduction of the expected future error (Eq. (1.1)). Because the calculation of the expected future error is intractable, it is approximated using online classification and online clustering models. *Importance weighting* principle is used to curb the *sampling bias* problem, so that drifting samples are queried. The proposed approach only considers binary classification and does not deal with *concept evolution*.

- *Chapter 3* proposes a new online AL algorithm, called SAL. It is capable of coping with both *concept drift* and *concept evolution*. SAL seeks to minimize the expected future error. However, here the approximation is done using different estimation models (i.e., non-parameter Bayesian models). These models allow to cope with the lack of prior knowledge about the data stream, like the number of classes and data distribution. The minimization 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.

- *Chapter 4* proposes a new online information-based AL algorithm (BSAL) that is able to cope with both *concept drift* and *concept evolution*. In contrast to decision-based AL approaches, adopted by BAL and SAL whose losses involve the prediction error, information-based AL losses involve the model parameter. Hence, BSAL can elegantly handle *concept evolution* since there is no need to heuristically modify the loss function to account for the new classes. More importantly, it can efficiently handle *concept evolution* from data involving classes of unbalanced proportions. Another contribution in this chapter is applying AL for sequential data streams with temporal dependency. To cope with this challenge, we propose an algorithm with a three-module architecture composed of a feature extractor (CRBM) and an online semi-supervised classifier (OSC) equipped with the AL algorithm

BSAL. Knowing that the proposed approach is generic and not dedicated to specific application, human activity recognition (HAR) is used as a real-world application.

- *Chapter 5* summarizes the key contributions of this thesis and provides a broad view of the future work. Particularly, how to address the delayed and noisy labelling challenges.

1.7 List of Publications

A number of publications have emerged from this research work with some of them still under review.

- *Chapter 2* is based on the following publication:
  - S. Mohamad, A. Bouchachia; M. Sayed-Mouchaweh, A Bi-Criteria Active Learning Algorithm for Dynamic Data Streams, IEEE Transactions on Neural Networks and Learning Systems, 2016.

    Impact Factor: 6.108
- *Chapter 3* is based on the following publications:
  - S. Mohamad, M. Sayed-Mouchaweh, A. Bouchachia. Active Learning for Classifying Data Streams with Unknown Number of Classes. Neural Networks, accepted with minor revision, 2017.

    Impact Factor: 5.287
  - S. Mohamad, Moamar Sayed-Mouchaweh, and Abdelhamid Bouchachia. Active Learning for Data Streams under Concept Drift and concept evolution. Workshop STREAMEVOLV organised at the European Conference on Machine Learning, 2016.
- *Chapter 4* is based on the following publications:
  - S. Mohamad, M. Sayed-Mouchaweh, A. Bouchachia, Online Active Learning for Human Activity Recognition from Sensory Data Streams, Pattern Recognition, under review, 2017.

    Impact Factor: 4.582

– S. Mohamad, A. Bouchachia, M. Sayed-Mouchaweh. A Non-parametric Hierarchical Clustering Model. The IEEE International Conference on Evolving and Adaptive Intelligent Systems (EAIS), pp:1-6, IEEE Press, 2015.

Chapter 2

Active Learning for Data Streams under Concept Drift

In this chapter, we address the challenges of applying AL for data streams which are *concept drift* and *sampling bias*. We propose a novel bi-criteria AL approach (BAL) that relies on *label uncertainty* criterion and *density-based* criterion. While the first criterion selects instances that are the most uncertain in terms of class membership, the latter dynamically curbs the *sampling bias* and avoids querying outlier by weighting the samples to reflect on the true underlying distribution. In order to design and implement these two criteria for learning from the data stream, BAL adopts a Bayesian online learning approach and combines online classification and online clustering through the use of *online logistic regression* and *online growing Gaussian mixture models* respectively.

The organization of this chapter is as follows. Section 2.1 is an introduction. Section 2.2 describes the proposed selection criteria along with the used classification and clustering algorithms. Section 2.3 provides the details of the proposed online sampling. Section 2.4 presents the algorithm. Section 2.5 discusses the experimental results for a number of well-known academic and real world datasets and Sec. 2.6 concludes the chapter.

2.1 Introduction

Classification has been the focus on 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 representative and is available prior to the training phase. Once this latter is exhausted, the classifier is deployed and, therefore, 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. That is, the characteristics of data streams make it impractical to use offline learning algorithms: i) data streams are unbounded in size, ii) they arrive at high steady rate and iii) they may evolve over time. Thus, to deal with data streams efficiently, the classifier must *(self-)adapt* online over time [69, 70]. To do that, the classifier needs to be fed with labeled data continuously which is not feasible in most real-world streaming situations where the data is usually unlabeled. It is therefore very important to seek well-informed ways to obtain labels. AL methods provide a systematic approach to select data examples whose labels should be queried. It reduces the cost of obtaining labelled data by querying much less examples to reach the same performance as when data samples are randomly queried.

For the sake of illustration, consider the example of internet advert popping up on screen where both online and active learning are relevant. The goal is to predict if an advert item will be interesting to a given shopper at a given time. To this end, a classifier is built based on the feedback from the shoppers. However, the interest and preference of the shoppers may change over time leading to what is known as *concept drift*. Therefore, building a static model for such a scenario will not be effective; hence, the importance of an online adaptive model is manifested. In the streaming setting, obtaining unlabelled data is often cheap but labelling it is expensive. For instance, in the previous example, asking frequently for feedback whether an advert is interesting would annoy the shoppers; hence AL should be deployed in applications with caution.

Label uncertainty (or uncertainty sampling) is a simple and popular criterion and more important it is very suitable for online AL. However, selecting the data examples from the stream using *label uncertainty* leads to *sampling bias* as we

explained in Chap. 1. On the other hand, *density-based* allows covering the whole input space with only few data samples [71] and reducing the *sampling bias*.

In this work, we introduce a bi-criteria AL algorithm (BAL) for evolving data streams. Specifically, we combine *label uncertainty* and *density-based* labelling in an SSS-like setting. The uncertainty of data samples is evaluated using a classification model while density of regions is evaluated using a clustering model. In general, density criterion performs efficiently with few labelled data since it samples from maximum-density unlabelled regions. On the other hand, uncertainty criterion tunes the decision boundary after selective sampling from the uncertain regions [39, 72–74]. Thus, the density criterion is useful when there is regularity in the data which is the case of many applications. However, the density criterion alone would not provide an accurate classifier. In other words, the density criterion helps establishing the initial decision boundary; while uncertainty sampling "fine-tunes" that boundary by sampling the regions where the classifier is least certain. By combining both criteria, we can take advantage of the density criterion to reduce the data examples required by the uncertainty criterion to build an accurate classifier.

To ensure enough flexibility of BAL, we explicitly distinguish between the learning engine and the selection engine. The learning engine uses a supervised learning algorithm to train a classifier on the existing labeled data, while the selection engine selects influential samples from the data stream for labelling. Here, the selection engine includes a classifier which is different from the classifier used by the learning engine and the clustering model that serves to design and deploy the two criteria.

AL stands as a very interesting opportunity to handle *concept drift* by querying the data samples representative of this drift (i.e., its characteristics). In contrast to standard *concept drift* 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. To this end, we use *importance weighting* principle to weight labelled data samples that drives a drift in order to increase the importance of their regions in the feature space. *Importance weighting* principle has also been theoretically proved to correct *sampling bias* [75]. The BAL algorithm proposed in this work is the first AL algorithm that is *concept drift* aware.

To illustrate how BAL behaves, consider the example that pertains to internet advertisement. Using uncertainty criterion only would result in querying adverts for which the classifier is uncertain and therefore by interactively labelling such adverts, the classifier can improve its performance towards a better prediction of what to propose to the shopper. On the other hand, the density criterion takes the similarity of the adverts into account. Thus, from a dense group of similar adverts only one which represents the whole group will be queried. Clearly combining the two criteria allows querying representative adverts for which the algorithm is not sure. Furthermore, in the streaming case, the interest and preference of the shoppers may change over time; hence when the change (*drift*) occurs in relation to those adverts for which the classifier is certain, the *sampling bias* caused by the active learning will be harmful. By implementing the weighting mechanism, we aim at reducing the *sampling bias* by labelling more of data samples that drive the drift and that make the uncertainty criterion less important for these data.

In a nutshell, the contributions in this chapter are:

1. We propose a novel online AL algorithm for data streams based on a probabilistic model that combines two querying criteria: uncertainty and *density-based* criteria. To the best of our knowledge this is the first approach based on density-uncertainty to the online setting.

2. The proposed combination brings classification through logistic regression and clustering through growing Gaussian mixture models together to implement the two querying criteria in a uniform probabilistic way. The choice of these algorithms is therefore well motivated.

3. We propose mechanisms for making the proposed BAL algorithm aware of *concept drift*. It is the first study that shows the effectiveness of AL in dealing with *concept drift*.

2.2 Online bi-criteria AL

The steps of the bi-criteria active learning algorithm (BAL) proposed in this work are portrayed in Fig. 2.1 and the corresponding details are discussed in Sec. 2.4. In a nutshell, BAL consists of 4 steps. In the first step, given an instance $\boldsymbol{x}$, the

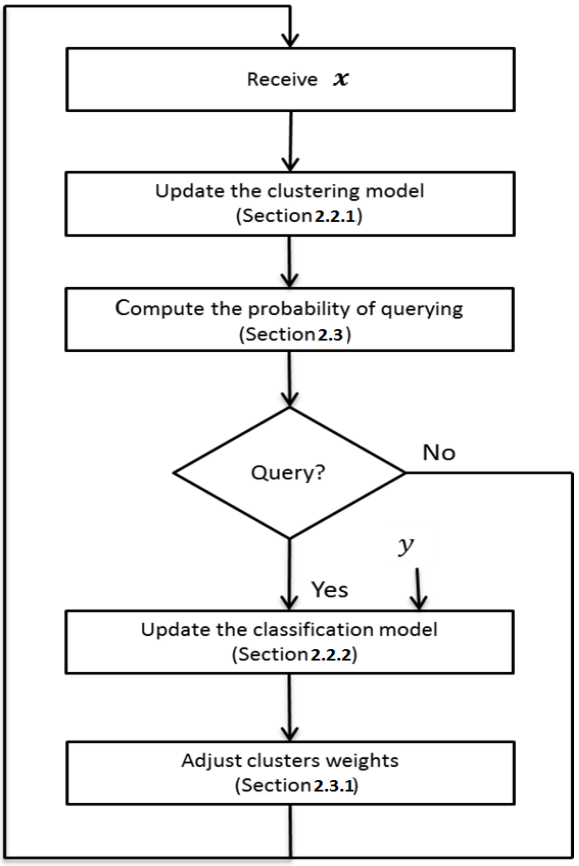


Figure. 2.1 General scheme of BAL

clustering model which is used to implement the density-based criterion is updated (see Sec. 2.2.1). Using BAL's sampling model in the second step, the probability of querying is computed and the decision whether to query and carry on to the third step or discard the instance and receive new one is made (see Sec. 2.3). In the third step, the label corresponding to the queried instance $\boldsymbol{x}$ is received and the classification model which is used to implement the label uncertainty criterion will be updated (see Section 2.2.2). In the fourth step, a weighting technique is used to curb the sampling bias (see Section 2.3.1).

In order to identify the data examples to query, we seek to minimize the current expected error which consequently leads to the minimization of the future expected error. It can be seen that this error can be derived from Eq. (1.1) (see Chap. 1) by using a prediction error loss function. The current expected error for an instance $\boldsymbol{x}$ can be computed using density-based and label uncertainty criteria which are estimated by dynamic classification and clustering models.

In [76] it is suggested to select samples that minimize the expected future classification error given as follows:

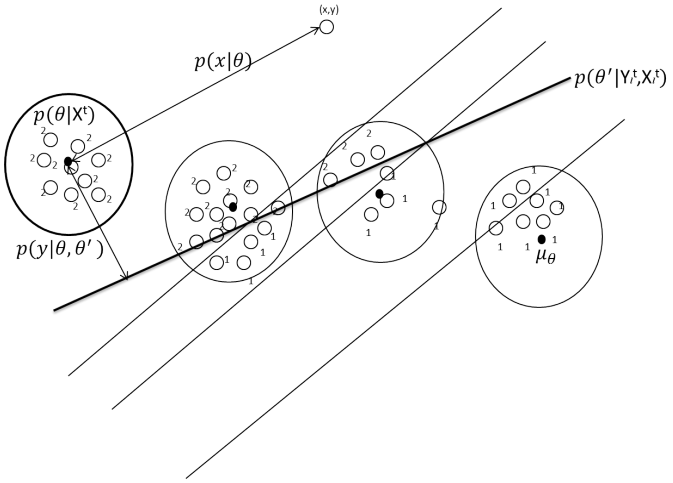


Figure. 2.2 Combining clustering and classification for AL

$$R = \int_{\boldsymbol{x}} E[(\hat{y} - y)^2|\boldsymbol{x}]p(\boldsymbol{x})d\boldsymbol{x} \tag{2.1}$$

where y is the true label of data instance $\boldsymbol{x}$ and $\hat{y}$ is the predicted output. $E[.|\boldsymbol{x}]$ denotes the expectation over $p(y|\boldsymbol{x})$. This integral cannot be computed due to its complexity. Authors in [49], who proposed an AL method that relies on an offline pool-based selective sampling setting, noted that instead of selecting data that produces the smallest future error, we can select the data that has the largest contribution to the current error in order to achieve a good approximation. Another issue pointed out is that the data distribution $p(\boldsymbol{x})$ affects the expected error.

In this study, we are concerned with online learning. Therefore, in order to have online approximation of Eq. (2.1), an online learning model that involves online sampling technique is needed. This model should be able to handle real drift and virtual drift [21]. The former refers to changes in the conditional distribution $p(y|\boldsymbol{x})$, whereas the latter refers to the changes in the distribution of the incoming data $p(\boldsymbol{x})$.

Our approach deals with the problems mentioned above and works independently from the classifier by adopting an online selection engine that uses the Bayesian inference. The random vector $\boldsymbol{z}$ is typically estimated from a training set of vectors Z [77, 78]. The Bayesian inference estimation process is given as follows:

$$p(\boldsymbol{z}|Z) = \int_{\Theta} p(\boldsymbol{z}, \Theta|Z)d\Theta = \int_{\Theta} p(\boldsymbol{z}|\Theta, Z)p(\Theta|Z)d\Theta \tag{2.2}$$

Assume that the selection engine has processed until time t, X_t data, among which X_{L_t} are labelled. Y_{L_t} is the set of labels associated with X_{L_t}. By replacing $\boldsymbol{z}$ with $(\boldsymbol{x}, y)$, Z with (X_t, Y_{L_t}), and Θ with $(\boldsymbol{\theta}, \boldsymbol{\theta'})$ where $\boldsymbol{\theta}$ and $\boldsymbol{\theta'}$ represent the parameters that govern respectively the clustering and the classification models. Equation (2.2) can be written as follows

$$p(\boldsymbol{x}, y|Y_{L_t}, X_t) = \int_{\boldsymbol{\theta'}} \int_{\boldsymbol{\theta}} p(\boldsymbol{x}, y|\boldsymbol{\theta}, \boldsymbol{\theta'}) p(\boldsymbol{\theta}, \boldsymbol{\theta'}|Y_{L_t}, X_t) d_{\boldsymbol{\theta}} d_{\boldsymbol{\theta'}} \tag{2.3}$$

$$p(\boldsymbol{\theta}, \boldsymbol{\theta'}|Y_{L_t}, X_t) = p(\boldsymbol{\theta}|Y_{L_t}, X_t, \boldsymbol{\theta'}) p(\boldsymbol{\theta'}|Y_{L_t}, X_{L_t}) \tag{2.4}$$

Assuming that $\boldsymbol{\theta}$ and $\boldsymbol{\theta'}$ are independent, then Eq. (2.4) can be rewritten as follows:

$$p(\boldsymbol{\theta}, \boldsymbol{\theta'}|Y_{L_t}, X_t) = p(\boldsymbol{\theta}|X_t) p(\boldsymbol{\theta'}|Y_{L_t}, X_{L_t}) \tag{2.5}$$

Assuming also that all the information about the class label y is encoded in the cluster parameter vector $\boldsymbol{\theta}$, then y and $\boldsymbol{x}$ are conditionally independent given $\boldsymbol{\theta}$. Therefore:

$$\begin{aligned} p(\boldsymbol{x}, y|\boldsymbol{\theta}, \boldsymbol{\theta'}) &= p(y|\boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{\theta'}) p(\boldsymbol{x}|\boldsymbol{\theta}, \boldsymbol{\theta'}) \\ &= p(y|\boldsymbol{\theta}, \boldsymbol{\theta'}) p(\boldsymbol{x}|\boldsymbol{\theta}) \end{aligned} \tag{2.6}$$

Equation (2.3) can then be rewritten as:

$$p(\boldsymbol{x}, y|Y_{L_t}, X_t) = \int_{\boldsymbol{\theta'}} \int_{\boldsymbol{\theta}} p(y|\boldsymbol{\theta}, \boldsymbol{\theta'}) p(\boldsymbol{\theta'}|Y_{L_t}, X_{L_t}) p(\boldsymbol{x}|\boldsymbol{\theta}) p(\boldsymbol{\theta}|X_t) d_{\boldsymbol{\theta}} d_{\boldsymbol{\theta'}} \tag{2.7}$$

This clearly explain how the selection engine consists of two models: a supervised learning classifier which is used to estimate the uncertain data examples and an unsupervised clustering model which is used to estimate the dense regions. These two models are used to approximate the future error. Equation (2.1) can be written as follows:

$$\int_{\boldsymbol{x}} \int_{y} (\hat{y} - y)^2 p(\boldsymbol{x}, y) dy d\boldsymbol{x} \tag{2.8}$$

$p(\boldsymbol{x}, y)$ can be estimated using Eq. (2.7). This later shows how label uncertainty and density-based criteria are combined. The data examples that minimize the current expected error are those located close to the center of clusters near the class boundaries. Figure 2.2 shows an example in two dimensions space illustrating the relationship between the clustering and classification models expressed in Eq. (2.7).

The distribution of $\boldsymbol{\theta'}$ and $\boldsymbol{\theta}$ are represented by lines and clusters. $p(y|\boldsymbol{\theta}, \boldsymbol{\theta'})$ depends on the distance between cluster $\boldsymbol{\theta}$ and line $\boldsymbol{\theta'}$. $p(\boldsymbol{x}|\boldsymbol{\theta})$ depends on the distance between $\boldsymbol{x}$ and cluster $\boldsymbol{\theta}$. As data is drawn from potentially changing distribution, $\boldsymbol{\theta}$ and $\boldsymbol{\theta'}$ may change.

In the following, the models exploited to define the two criteria used to decide when to query the labels are introduced. The two criteria are label uncertainty defined through logistic regression [28] and density-based criterion defined through Growing Gaussian Mixture Model (GGMM) [69, 79].

2.2.1 Growing Gaussian Mixture Model (GGMM)

To detect the dense regions, an online learning algorithm to estimate the density of data examples $\boldsymbol{x_t}$ is needed. According to Bayesian inference:

$$p(\boldsymbol{x_t}|X_{t-1}) = \int_{\boldsymbol{\theta_{t-1}}} p(\boldsymbol{x_t}|\boldsymbol{\theta_{t-1}}) p(\boldsymbol{\theta_{t-1}}|X_{t-1}) d\boldsymbol{\theta_{t-1}} \tag{2.9}$$

where t represents the time. Traditionally, computing the integral in Eq. (2.9) is not always straightforward and the Bayesian inference is often approximated via maximum-a-posteriori (MAP) or maximum likelihood estimation (MLE). In order to obtain an online approximation for Eq. (2.9), an online Gaussian Mixture Model algorithm (GMM) is used. The Gaussian mixture model perceives the data as a population with K different components where each component is generated by an underlying probability distribution [80].

$$p(\boldsymbol{x_t}|X_{t-1}) = \sum_{i=1}^{K} p(\boldsymbol{x_t}|\hat{\boldsymbol{\theta}}^{\boldsymbol{i}}_{\boldsymbol{t-1}}) \tau^i_{t-1} \tag{2.10}$$

where τ^i_{t-1} is the weight of the ith Gaussian in the mixture model, $\hat{\boldsymbol{\theta}}_{\boldsymbol{t}} = (\hat{\boldsymbol{\theta}}^{\boldsymbol{1}}_{\boldsymbol{t}}, ..., \hat{\boldsymbol{\theta}}^{\boldsymbol{K}}_{\boldsymbol{t}})$, where $\hat{\boldsymbol{\theta}}^{\boldsymbol{i}}_{\boldsymbol{t}}$ is parameter vector of cluster i.

In this case of incomplete data, MLE and MAP estimates are not directly computable. Therefore, it is standard to use iterative algorithms such as Expectation Maximization (EM). To accommodate online learning, the GGMM proposed in [69] is adopted. It estimates $\hat{\boldsymbol{\theta}}_{\boldsymbol{t}}$ from data by maximizing the likelihood of the joint distribution $p(X_t, \hat{\boldsymbol{\theta}}_{\boldsymbol{t}})$[81]. GGMM learns from labeled and unlabeled data and handles the complexity of the mixture model efficiently. Consequently by using

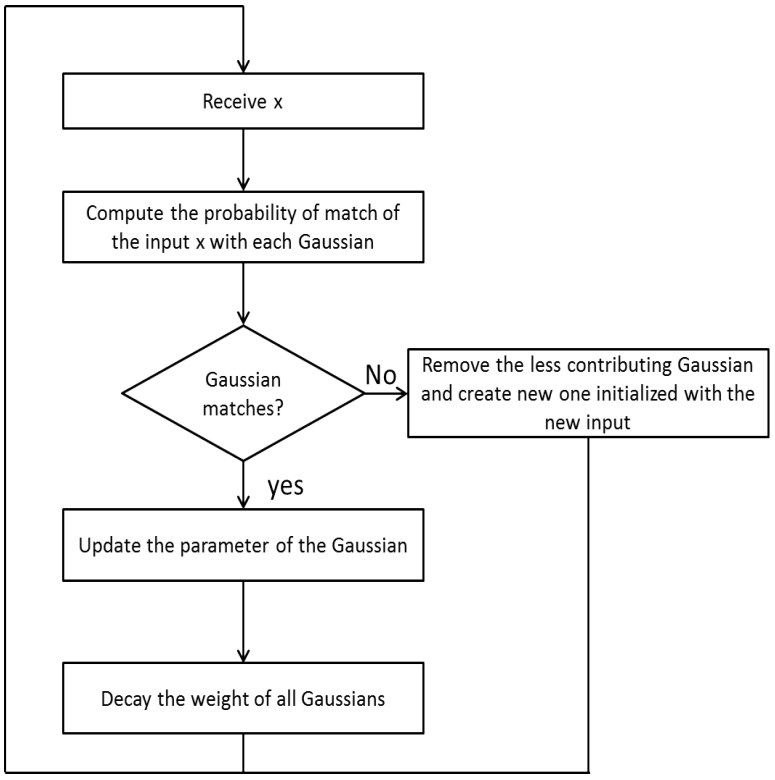


Figure. 2.3 The main steps of GGMM

GGMM, we wish to implement the density-based active learning criterion in an efficient way.

The parameters of GGMM are the clusters variance, the learning rate which determines the update step of the clusters' parameters, the maximum admissible number of clusters and the closeness threshold which controls the creation of new clusters. GGMM creates a new cluster when the Mahalanobis distance between a new input and the nearest cluster is more than the closeness threshold. To make the experiments easier, the closeness threshold is set equal to the variance. GGMM uses a constant fading factor (learning rate) to tune the contribution of clusters by updating $\hat{\boldsymbol{\theta}}_{\boldsymbol{i}}$. The least contributing clusters are discarded systematically and new ones are added dynamically over time. Figure 2.3 shows the main steps of GGMM.

Due to its incremental nature, GGMM can cope with concept drift [69] and its combination with the online classifier, logistic regression, (see Sec. 2.2.2) to devise BAL ensures real-time adaptation.

2.2.2 Logistic regression

To sample the uncertain data examples, the logistic classifier which is offline, probabilistic, and linear in the parameters is applied. To meet the online requirement of our approach, this classifier will be adapted as will be shown below.

Logistic regression corresponds to the following binary classification model:

$$y|\boldsymbol{x} \sim Bern(\mu') \tag{2.11}$$

$y|\boldsymbol{x}$ has the Bernoulli distribution with parameter μ' given as:

$$\mu' = p(y = 1|\boldsymbol{x}, \boldsymbol{\theta'}) = sigm(\boldsymbol{\theta'}^T \boldsymbol{x}) \tag{2.12}$$

where $sigm(a)$ refers to the sigmoid function, $sigm(a) = \frac{1}{1+\exp(-a)}$.

Note that in this work, binary classification is considered. However, it is easy to extend logistic regression to multi-class classification. In the following, the adaptation of logistic regression to online classification is discussed and then our method for handling concept drift using online logistic regression is introduced.

2.2.2.1 Bayesian view of online logistic regression

Logistic regression can be trained in an online mode either by using stochastic optimization or by adopting a Bayesian view which has the obvious advantage of returning posterior instead of just point estimate. In this work, we use the Bayesian view because we want to capture the non-deterministic nature (uncertainty) of the querying process that itself exploits the label uncertainty criterion. The Bayesian approach of the logistic regression is expressed through:

$$p(\boldsymbol{\theta'_t}|D_{L_t}) \propto p((\boldsymbol{x_t}, y_t)|\boldsymbol{\theta'_t}) p(\boldsymbol{\theta'_t}|D_{L_{t-1}}) \tag{2.13}$$

where D_{L_t} is the labeled data seen up to time t. Suppose that at time $t-1$ our knowledge about the parameters $\boldsymbol{\theta'_{t-1}}$ is summarized by the posterior distribution $p(\boldsymbol{\theta'_{t-1}}|D_{L_{t-1}})$. After receiving an observation $\boldsymbol{x_t}$, we consider the probability:

$$p(y_t|\boldsymbol{x_t}, D_{L_{t-1}}) = \int_{\boldsymbol{\theta'_t}} p(y_t|\boldsymbol{x_t}, \boldsymbol{\theta'_t}) p(\boldsymbol{\theta'_t}|D_{L_{t-1}}) d_{\boldsymbol{\theta'_t}} \tag{2.14}$$

where $p(\boldsymbol{\theta'_t}|D_{L_{t-1}})$ is the predicted posterior which can be expressed as:

$$p(\boldsymbol{\theta'_t}|D_{L_{t-1}}) = \int_{\boldsymbol{\theta'_{t-1}}} p(\boldsymbol{\theta'_t}|\boldsymbol{\theta'_{t-1}})p(\boldsymbol{\theta'_{t-1}}|D_{L_{t-1}})d_{\boldsymbol{\theta'_{t-1}}} \tag{2.15}$$

In order to calculate the expression $p(\boldsymbol{\theta'_t}|\boldsymbol{\theta'_{t-1}})$, we must specify how the parameters change over time. Following [28], we assume no knowledge of the drifting distribution $p(\boldsymbol{\theta'_t}|\boldsymbol{\theta'_{t-1}})$. Thus, Eq. (2.15) can be eliminated by estimating $p(y_t|x_t, D_{L_{t-1}})$ which is done as follows:

$$p(y_t|\boldsymbol{x_t}, D_{L_{t-1}}) = \int_{\boldsymbol{\theta'_{t-1}}} p(y_t|\boldsymbol{x_t}, \boldsymbol{\theta'_{t-1}})p(\boldsymbol{\theta'_{t-1}}|D_{L_{t-1}})d_{\boldsymbol{\theta'_{t-1}}} \tag{2.16}$$

Here, two challenges need to be dealt with:

- predicting the class of the new data example $\boldsymbol{x_t}$ by computing the posterior predictive distribution (Eq. (2.16))
- updating the posterior distribution $p(\boldsymbol{\theta'_t}|D_{L_t})$ as soon as the label of $\boldsymbol{x_t}$ is obtained by computing Eq. (2.13)

Both challenges require to approximate the prior distribution over the weight $\boldsymbol{\theta'_t}$ as a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu_t}, \boldsymbol{\Sigma_t})$. Two methods have been proposed in the literature to approximate the integral of Eq. (2.16): Monte Carlo approximation and probit approximation [82]. We use the probit approximation as it does not require sampling, thus it takes less computational time. However, it has been found that probit approximation gives very similar results to the Monte Carlo approximation [82]. The result of the approximation is as follows:

$$p(\hat{y}_t = 1|\boldsymbol{x_t}, D_{L_{t-1}}) \approx \hat{\nabla}_t = sigm(K(s_t)\bar{a}_t) \tag{2.17}$$

$$s_t^2 = \boldsymbol{x_t}^T \boldsymbol{\Sigma_{t-1}} \boldsymbol{x_t} \tag{2.18}$$

$$\bar{a}_t = \boldsymbol{\mu_{t-1}}^T \boldsymbol{x_t} \tag{2.19}$$

$$K(s_t) = (1 + \frac{\pi s_t^2}{8})^{-\frac{1}{2}} \tag{2.20}$$

For more details about the approximation steps, the interested reader is referred to [82].

In the following, we show how the parameters of the proposed online logistic regression classifier are updated. We use Newton's method as formulated in [83] to sequentially update $\boldsymbol{\theta'_t} = (\boldsymbol{\Sigma_t}, \boldsymbol{\mu_t})$ of Eq. (2.13) using Eq. (2.17). Therefore, the mean $\boldsymbol{\mu_t}$ and covariance $\boldsymbol{\Sigma_t}$ can be updated as follows:

$$\boldsymbol{\Sigma_t} = \boldsymbol{\Sigma_{t-1}} - \frac{\hat{\nabla}_t(1-\hat{\nabla}_t)}{1+\hat{\nabla}_t(1-\hat{\nabla}_t)s_t^2}(\boldsymbol{\Sigma_{t-1}}\boldsymbol{x_t})((\boldsymbol{\Sigma_{t-1}}\boldsymbol{x_t}))^T$$

$$\boldsymbol{\mu_t} = \boldsymbol{\mu_{t-1}} + \boldsymbol{\Sigma_t}\boldsymbol{x_t}(y_t - \hat{\nabla}_t) \tag{2.21}$$

These recursive equations reflect on the current and the past data. Nevertheless, the effect of data is implicitly decreasing as more data is processed due to the variance shrinkage.

2.2.2.2 Handling of concept drift

In non stationary setting, a variant version of (2.21) proposed in [28] is used. The situation is exactly the same as for the stationary case, except that the prior distribution is now $\mathcal{N}(\boldsymbol{\mu_{t-1}}, \boldsymbol{\Sigma_{t-1}} + v_t\boldsymbol{A})$. Here $\boldsymbol{A} = c\boldsymbol{I}$ with c is a constant and $\boldsymbol{I}$ is the identity matrix. $v_t\boldsymbol{A}$ assumes that the weight vector $\boldsymbol{\theta'_t}$ changes are of similar magnitude. Alternatively, one could use a separate forgetting matrix parameter for every weight coordinate as discussed in [84]. In this work, we consider the first assumption:

$$\begin{aligned}\boldsymbol{\Sigma_t} =& (\boldsymbol{\Sigma_{t-1}} + v_t\boldsymbol{A}) - \frac{\hat{\nabla}_t(1-\hat{\nabla}_t)}{1+\hat{\nabla}_t(1-\hat{\nabla}_t)s_t'^2}[(\boldsymbol{\Sigma_{t-1}} + v_t\boldsymbol{A})\boldsymbol{x_t}][(\boldsymbol{\Sigma_{t-1}} + v_t\boldsymbol{A})\boldsymbol{x_t}]^T \\ \boldsymbol{\mu_t} =& \boldsymbol{\mu_{t-1}} + \boldsymbol{\Sigma_t}\boldsymbol{x_t}(y_t - \hat{\nabla}_t)\end{aligned} \tag{2.22}$$

where $s_t'^2 = \boldsymbol{x_t}^T(\boldsymbol{\Sigma_{t-1}} + v_t\boldsymbol{A})\boldsymbol{x_t}$. Here v_t can be thought of as the Bayesian version of the window size in batch learning for data stream. In order to adjust the model as soon as it becomes unable to estimate the true changing $\boldsymbol{\theta'}$ distribution, we compute the discrepancy between the predictive class uncertainty after and before observing the true class label:

$$G_t = \tilde{U}(\boldsymbol{x_t}) - \hat{U}(\boldsymbol{x_t}) \tag{2.23}$$

where $\hat{U}(\boldsymbol{x_t})$ is the uncertainty of the label predicted from $\boldsymbol{x_t}$ and $\tilde{U}(\boldsymbol{x_t})$ is the uncertainty remaining after incorporating the true class label.

$$\begin{aligned}\hat{U}(\boldsymbol{x_t}) &= E[(\hat{y}_t - y_t)^2|\boldsymbol{x_t}] \\ \tilde{U}(\boldsymbol{x_t}) &= E[(\tilde{y}_t - y_t)^2|\boldsymbol{x_t}]\end{aligned} \tag{2.24}$$

$\hat{y}_t$ is the predicted label and $\hat{y}_t = \mathbb{1}(\hat{\nabla}_t > 0.5)$. $\tilde{y}_t$ is the predicted label after updating the classification model with the true label. $\tilde{y}_t = \mathbb{1}(\tilde{\nabla}_t > 0.5)$, where $\tilde{\nabla}$ can be computed as follows:

$$p(\tilde{y}_t = 1|\boldsymbol{x_t}, D_{L_t}) \approx \tilde{\nabla}_t = sigm(K(\tilde{s}_t)\tilde{a}_t) \tag{2.25}$$

$$\tilde{s}_t^2 = \boldsymbol{x_t}^T \boldsymbol{\Sigma_t} \boldsymbol{x_t} \tag{2.26}$$

$$\tilde{a}_t = \boldsymbol{\mu_t}^T \boldsymbol{x_t} \tag{2.27}$$

$$K(\tilde{s}_t) = (1 + \frac{\pi \tilde{s}_t^2}{8})^{-\frac{1}{2}} \tag{2.28}$$

The discrepancy G_t of the model uncertainty is monitored using a forgetting technique[85]:

$$v_t = \frac{\alpha v_{t-1} + \max(G_t, 0)}{NL_t} \tag{2.29}$$

$$NL_t = \alpha NL_{t-1} + l(\boldsymbol{x_t}) \tag{2.30}$$

NL_t is the prequential number of labeled data. $l(\boldsymbol{x_t})$ is 1 if $\boldsymbol{x_t}$ is labeled and 0 otherwise. α is a constant fading factor empirically set to 0.9. Equation (2.29) will be used to update Eq. (2.22).

Finally, the online logistic regression classifier is incrementally updated using (2.22) that addresses concept drift.

2.3 Online sampling

In the following, the sampling method used by BAL that avoids the problem of sampling bias is illustrated. Next, a technique to restrict the available resources in terms of labelling budget is described.

We have noted earlier that the computation of the future error in Eq. (2.1) is complicated. So instead, we select the sample that has the largest contribution to

the current error. Although such approach does not guarantee the smallest future error, there is a good chance for a large decrease in error. In an offline setting, the selection criterion based on the set of unlabeled data X_U is:

$$\boldsymbol{x} = \arg \max_{\boldsymbol{x_j} \in X_U} E[(\hat{y}_j - y_j)^2 | \boldsymbol{x_j}] p(\boldsymbol{x_j}) \tag{2.31}$$

$$E[(\hat{y}_j - y_j)^2 | x_j] = p(y_j = 1 | \boldsymbol{x_j})(\hat{y}_j - 1)^2 + p(y_j = 0 | \boldsymbol{x_j}) \hat{y}_j^2 \tag{2.32}$$

where $p(y|\boldsymbol{x})$ is unknown and needs to be approximated. In [49], the authors use the current estimation $p(y|\boldsymbol{x}, \hat{\boldsymbol{\theta}}')$ assuming that $\hat{\boldsymbol{\theta}}'$ is good enough. Unfortunately, in the online learning setting, the task is more challenging for three issues:

- No access to the already seen unlabeled data.
- The probability $p(y|\boldsymbol{x}, \hat{\boldsymbol{\theta}}')$ dynamically changes as more labeled data is seen. That means the approximation assumed above needs adjustment.
- Need to address the problem of sampling bias (which is detailed in the next Section).

To address these issues, we formulate the querying (sampling) probability in a recursive manner as follows. Let q be the binary random variable that determines whether $\boldsymbol{x}$ should be queried. The querying probability is defined by the following model:

$$q|\boldsymbol{x} \sim Bern(\mu') \tag{2.33}$$

Using Eq. (2.8), the querying probability can be reformulated as follows:

$$p(q = 1|\boldsymbol{x}) = \int_y (\hat{y} - y)^2 p(\boldsymbol{x}, y) dy \tag{2.34}$$

That is to say, a sample that has a large contribution to the current error is likely to be queried.

Now, let's see how this probability is formulated for the online setting so that we avoid the requirement to have access to the whole data. Let $D_t = (Y_{L_t}, X_t)$ be the set of labeled data seen so far. Using Eq. (2.7), we express the probability of

querying $\boldsymbol{x_t}$ as follows:

$$p(q_t = 1|\boldsymbol{x_t}, D_{t-1}) = \int_{y_t}\int_{\boldsymbol{\theta_t}} (\hat{y}_t - y_t)^2 p(y_t|\boldsymbol{\theta_t}, D_{L_{t-1}}) p(\boldsymbol{x_t}|\boldsymbol{\theta_t}) p(\boldsymbol{\theta_t}|X_t) d_{\boldsymbol{\theta_t}} d_{y_t}$$
$$= \int_{\boldsymbol{\theta_t}} E[(\hat{y}_t - y_t)^2|\boldsymbol{\theta_t}] p(\boldsymbol{x_t}|\boldsymbol{\theta_t}) p(\boldsymbol{\theta_t}|X_t) d_{\boldsymbol{\theta_t}} \tag{2.35}$$

$E[.|\boldsymbol{\theta_t}]$ denotes the expectation over $p(y_t|\boldsymbol{\theta_t}, D_{L_{t-1}})$. Using Eq. (2.10), $p(q_t = 1|\boldsymbol{x_t}, D_{t-1})$ can be estimated as follows:

$$p(q_t = 1|\boldsymbol{x_t}, D_{t-1}) = \sum_{i=1}^{K} E[(\hat{y}_t - y_t)^2|\hat{\boldsymbol{\theta}}_{\boldsymbol{t}}^{\boldsymbol{i}}] p(\boldsymbol{x_t}|\hat{\boldsymbol{\theta}}_{\boldsymbol{t}}^{\boldsymbol{i}}) \tau_t^i \tag{2.36}$$

where $E[(\hat{y}_t - y_t)^2|\hat{\boldsymbol{\theta}}_{\boldsymbol{t}}^{\boldsymbol{i}}] = \hat{U}(\hat{\boldsymbol{\theta}}_{\boldsymbol{t}}^{\boldsymbol{i}})$ can be computed as:

$$\hat{U}(\hat{\boldsymbol{\theta}}_{\boldsymbol{t}}^{\boldsymbol{i}}) = p(y_t = 1|\boldsymbol{\mu}_{\hat{\boldsymbol{\theta}}_{\boldsymbol{t}}^{\boldsymbol{i}}}, D_{L_{t-1}})(\hat{y}_t - 1)^2 + p(y_t = 0|\boldsymbol{\mu}_{\hat{\boldsymbol{\theta}}_{\boldsymbol{t}}^{\boldsymbol{i}}}, D_{L_{t-1}})\hat{y}_t^2 \tag{2.37}$$

$\boldsymbol{\mu}_{\hat{\boldsymbol{\theta}}_{\boldsymbol{t}}^{\boldsymbol{i}}}$ is the mean of cluster i at time t. $p(y_t = 1|\boldsymbol{\mu}_{\hat{\boldsymbol{\theta}}_{\boldsymbol{t}}^{\boldsymbol{i}}}, D_{L_{t-1}})$ can be computed using Eq. (2.17) after replacing $\boldsymbol{x_t}$ by $\boldsymbol{\mu}_{\hat{\boldsymbol{\theta}}_{\boldsymbol{t}}^{\boldsymbol{i}}}$. The resulting $p(y_t = 1|\boldsymbol{\mu}_{\hat{\boldsymbol{\theta}}_{\boldsymbol{t}}^{\boldsymbol{i}}}, D_{L_{t-1}})$ is the recursive approximation of the querying probability.

2.3.1 Tackling the problem of sampling bias

In general, the active learning model starts by exploring the environment. As training proceeds, the model becomes more certain. Then, data samples are queried based on their informativeness. As a result, the training set quickly will no more represent the underlying data distribution, hence the problem sampling bias.

Given a classification model with a parameter vector $\boldsymbol{\theta}$, the maximum a posteriori (MAP) estimate is $\hat{y} = argmax_y p(y|\boldsymbol{x}, \boldsymbol{\theta})$ and the risk associated is expressed as:

$$R(\boldsymbol{\theta}) = \int_{\boldsymbol{x}}\int_{y} L(\hat{y}, y) p(\boldsymbol{x}, y) dy d\boldsymbol{x} \tag{2.38}$$

where $L(.)$ is the loss function measuring the disagreement between prediction and the true label. Since $p(\boldsymbol{x}, y)$ is unknown the expected loss can be approximated by an empirical risk:

$$\hat{R}_n(\boldsymbol{\theta}) = \frac{1}{n}\sum_{j=1}^{n} L(\hat{y}_j, y_j) \tag{2.39}$$

where $(\boldsymbol{x_j}, y_j)$ are drawn from $p(\boldsymbol{x}, y)$ and n is the number of samples. In active learning instances are drawn according to an instrumental distribution $g(\boldsymbol{x})$. Thus, $(\boldsymbol{x_j}, y_j)$ are sampled from $g(\boldsymbol{x})p(y|\boldsymbol{x})$. In presence of drift, $g(\boldsymbol{x})$ may have low probability for data located far from the class boundary because it is considered as an uninformative region. If a drift occurs in that region, many instances with high loss $L(\hat{y}_j, y_j)$ will not be queried. This leads to a negative effect of active learning; sometimes, worse than learning from random sampling. In order to develop an unbiased estimator of the expected loss, we weight each drawn instance following the concept of weighted sampling. Thus, the empirical risk can be written as follows:

$$\hat{R}_{g,n}(\boldsymbol{\theta}) = \frac{1}{S}\sum_{j=1}^{n} S_j L(\hat{y}_j, y_j) \tag{2.40}$$

where $S_j = \frac{p(\boldsymbol{x_j})}{g(\boldsymbol{x_j})}$ is the *importance weighting* compensating for the discrepancy between the original and instrumental distributions, and $S = \sum_{j=1}^{n} S_j$ is the normalizer. Thanks to the *importance weighting*, Eq. (2.40) defines a consistent estimator [86]. That is, the expected value of the estimator $\hat{R}_{g,n}(\boldsymbol{\theta})$ converges to the true risk $R(\boldsymbol{\theta})$ for $n \to \infty$. The importance weighting was used in [75] to correct the sampling bias showing that by weighting the queried sample according to the reciprocals of the labelling probability, a statistical consistency is guaranteed. For any distribution and any hypothesis class, active learning eventually converges to the optimal hypothesis in the class. In the case of online learning, the importance weighting can be defined as follows:

$$\begin{aligned} S_t &= \frac{p(\boldsymbol{x_t})}{p(\boldsymbol{x_t})p(q_t = 1|\boldsymbol{x_t}, D_{t-1})} \\ &= \frac{1}{p(q_t = 1|\boldsymbol{x_t}, D_{t-1})} \end{aligned} \tag{2.41}$$

In order to apply the importance weighting, we interpolate the effect of weighting into the selection engine through the density estimation. Thus, the clustering model is updated with the importance sampling S_t. The effect of S_t on each cluster is represented by H_t^i as follows:

$$H_t^i = \begin{cases} S_t^{-1} p(\hat{\boldsymbol{\theta}}_{\boldsymbol{t}}^{\boldsymbol{i}}|\boldsymbol{x_t}) & \text{if } \boldsymbol{x_t} \text{ is classified correctly} \\ 0, & \text{if } \boldsymbol{x_t} \text{ is not queried} \\ (1 - S_t^{-1}) p(\hat{\boldsymbol{\theta}}_{\boldsymbol{t}}^{\boldsymbol{i}}|\boldsymbol{x_t}) & \text{otherwise} \end{cases} \tag{2.42}$$

Therefore, the new cluster weight can be written as follows:

$$\tau^i_{t,S_t} = (1 - H^i_t)\tau^i_t(1 - I_t) + ((1 - H^i_t)\tau^i_t + H^i_t)I_t \tag{2.43}$$

where $I_t = |\hat{y} - y|$, that is, I_t is 0 if x_t is correctly classified and 1 otherwise. The first term of Eq. (2.43) is used to decrease the effect of over-sampling by reducing the weight of clusters representing the instances correctly labeled. The second term is used to decrease the effect of under-sampling by increasing the weight of clusters representing the instances wrongly labeled.

2.3.2 Budget

Under limited labelling resources, a rationale querying strategy to optimally use those resources needs to be applied. To this end, the notion of budget was introduced in [1] in order to estimate the label spending. Two counters were maintained: the number of labeled instances f_t and the budget spent so far: $b_t = \frac{f_t}{|\text{data seen so far}|} = \frac{f_t}{|X_t|}$. As data arrives, we do not query unless the budget is less than a constant Bd and querying is granted by the sampling model. However, over infinite time horizon this approach will not be effective. The contribution of every label to the budget will diminish over the infinite time and a single labelling action will become less and less sensitive. Authors in [1] propose to compute the budget over fixed memory windows wnd. To avoid storing the query decisions within the windows, an estimation of f_t and b_t were proposed. It is computed as follows:

$$\hat{b}_t = \frac{\hat{f}_t}{wnd} \tag{2.44}$$

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

$$\hat{f}_t = (1 - 1/wnd)\hat{f}_{t-1} + Lab_{t-1} \tag{2.45}$$

where $Lab_{t-1} = 1$ if instance x_{t-1} is labelled, and 0 otherwise. Using the forgetting factor $(1 - 1/wnd)$, the authors showed that $\hat{b}_t$ is unbiased estimate of b_t.

In the present work, this notion of budget will be adopted in BAL so that we can assess it against the active learning proposed in [1]. Note that in our experiments in relation to the budget, we set $wnd = 100$ as in [1].

2.4 Algorithm

Having introduced the active learning criteria and the sampling technique based on Fig. 2.2, the full details of the algorithm are provided in Alg. (1). The lines 7, 15, and 16 are included in BAL only when the budget is considered. Otherwise, BAL is not constrained by the budget.

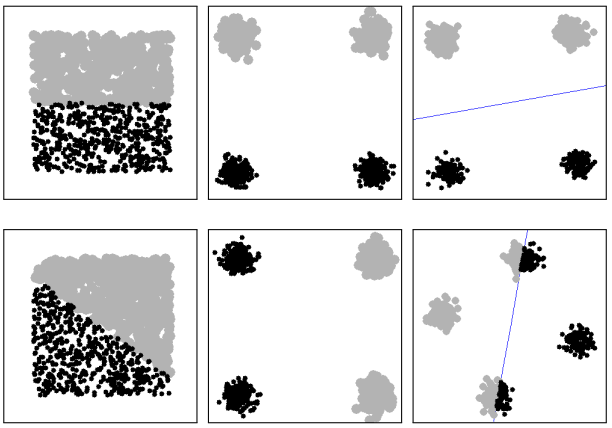
Algorithm 1 Steps of BAL

1: **Input:** data stream, parameters of GGMM: {maximum number of clusters, variance, learning rate }, BAL forgetting matrix $\boldsymbol{A}$ (Eq. (2.22)), budget Bd.
2: **initialize:** $t = 0$, $\boldsymbol{\mu_0} = \vec{0}$, $\boldsymbol{\Sigma_0} = 5\boldsymbol{I}$ (Eq. (2.22)), $v_t = 0$ (Eq. (2.29)), $\hat{f}_1 = 0$ (Eq. (2.45))
3: **while** (true) **do**
4: $t \leftarrow t + 1$,
5: Receive $\boldsymbol{x_t}$
6: Update the clustering model $\{\hat{\boldsymbol{\theta}}^{\boldsymbol{1}}_{\boldsymbol{t-1}}, ...\hat{\boldsymbol{\theta}}^{\boldsymbol{k}}_{\boldsymbol{t-1}}\}$ by $\boldsymbol{x_t}$.
7: **if** $\hat{b}_t < Bd$ (Eq. (2.44)) **then**
8: compute $\mu' = p(q_t = 1|\boldsymbol{x_t}, D_{t-1})$ that refers to the combination of uncertainty and density criteria using Eq. (2.36)
9: $q_t \sim Bern(\mu')$ (Eq. (2.33))
10: **if** $q_t = 1$ **then**
11: $y_t \leftarrow query(\boldsymbol{x_t})$
12: Update the classifier model $\{\boldsymbol{\mu_{t-1}}, \boldsymbol{\Sigma_{t-1}}\}$ by $(\boldsymbol{x_t}, y_t)$ using Eq. (2.22)
13: Remove the effect of sampling bias using Eq. (2.43)
14: **end if**
15: **end if**
16: Compute $\hat{f}_{t+1}$ (Eq. (2.45))
17: **end while**

2.5 Experiments

First, BAL is evaluated by analyzing its behavior under different data distributions and different types of concept drift. In order to have controlled settings with known distribution and drift type, $2D$ synthetic datasets proposed in [87] are considered. Second, BAL is compared against the state-of-art active learning methods designed for data streams on real-world datasets. Two real-world benchmark datasets are used: Electricity [88] and Airline [89].

The forgetting constant 'c' which determines the forgetting matrix $\boldsymbol{A}$ in Eq. (2.22) is empirically set to 5. The parameters of GGMM are empirically specified in



(a) Gradual drift (moving plane) (b) Abrupt drift (c) Mixture drift (moving Gaussian)

Figure. 2.4 Synthetic data

Tab. 2.1 (see App. A.1 for more details). To capture the real performance of BAL, all experiments are repeated 30 times and the results are averaged.

2.5.1 Simulation on synthetic data

The main goal of this section is to analyze BAL in terms of effect of concept drift and data distribution. Also the impact of the budget and the number of clusters on the performance of BAL are studied. Then the strengths and weaknesses of BAL are discussed. Note that here the online logistic regression algorithm serves as learning engine. Three synthetic datasets involving two different distributions, Gaussian and uniform, and three types of data drift: gradual, abrupt and mixture which involves both gradual and abrupt drifts are used. These types of drift occur in real-world applications as shown in the following:

- **Gradual drift:** consider two sources with gradual changes from one source to the other. The so-called moving plane data is used. A gradual changing environment is simulated by rotating the linear class boundary about the origin in a 2-d space. All data points come from a uniform distribution in the unit square (see Fig. 2.4a). The class definition (concept) changes with each new data point by rotating the boundary at a further angle of 1. The distribution $p(\boldsymbol{x})$ does not change over time, while $p(y|\boldsymbol{x})$ does. A real example is when a device begins to malfunction; the quality of the service that it provides starts to decrease. After a certain period, the service quality

reaches an unacceptable level. In this case, the device is considered to work under failure operation conditions [90].

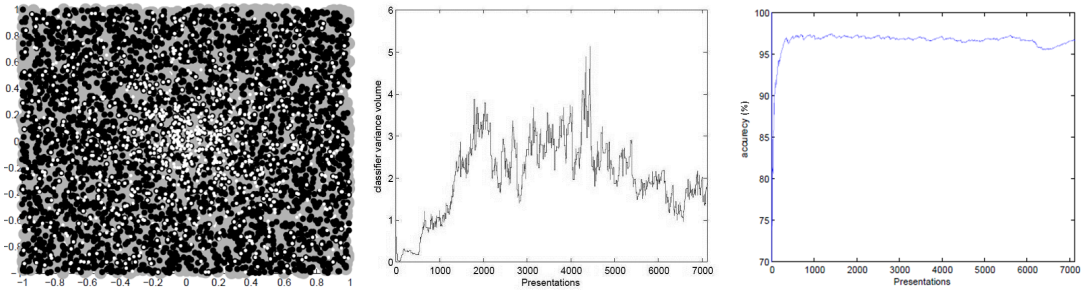
- **Abrupt drift:** consider many data sources each related to one of the target concepts. In the abrupt drift scenario, usually one data source is instantly replaced by another. An example where data is Gaussian distributed is used. Here, we have 4 Gaussian sources with half related to one concept and half to another. In the substitution process, one data source is instantly replaced by another yielding change in $p(y|\boldsymbol{x})$ with the same $p(\boldsymbol{x})$ (see Fig. 2.4c). A real example is the stuck on or stuck off faults in a valve or the failed on or failed off of a pump [90]. We also introduce *remote* abrupt drift for uniformly distributed data by instantly moving the boundary far away (see Fig. 2.12).

- **Mixture drift:** another version of Gaussian drift called moving Gaussian, where both $p(\boldsymbol{x})$ and $p(y|\boldsymbol{x})$ change. If $p(\boldsymbol{x})$ gradually changes, then $p(y|x)$ abruptly drifts and continues gradually drifting (see Fig. 2.4c). Consider, for example, textual news arriving as a data stream. The goal is to predict if a news item will be interesting to a given user at a given time. The preferences of the reader ($P(y|\boldsymbol{x})$) as well as the news popularity ($p(\boldsymbol{x})$) may change gradually or abruptly over time.

2.5.1.1 Simulation on synthetic data

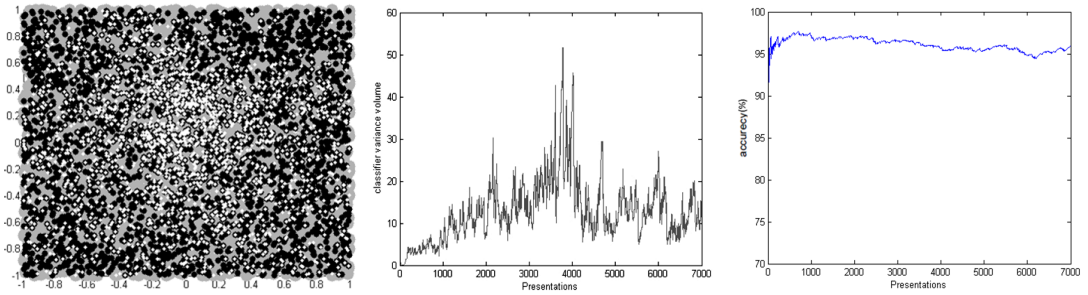
The notion of budget is not used here, thus there is no restriction on the use of resources. In order to study the behaviour of BAL on different types of drift and distributions, each dataset is experimented using only the uncertainty criterion (UAL). Then, another three experiments on the same datasets are carried out using BAL. Comparing the results allows us to study the impact (interest) of incorporating density-based criterion on different scenarios (different distributions and types of drift). The behavior of BAL is analyzed in terms of sequential querying probability and accuracy. In the following, the results of the experiment on these three datasets are shown.

Figures 2.5 and 2.6 show the results after applying UAL and BAL on the moving plane data to test the gradual drift. Figure 2.5a and 2.6a show the queried data in white. BAL and UAL algorithms adaptively pick the uncertain data around the center of the whole data as the rotation hyperplane crosses the data in the



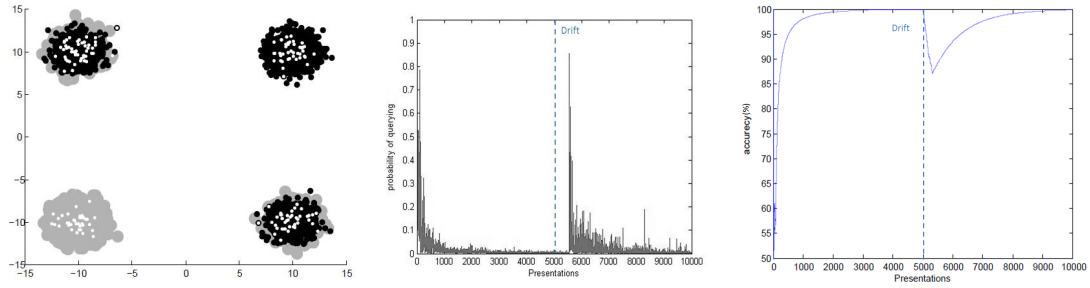
(a) Data (b) Classifier variance (c) Accuracy

Figure. 2.5 UAL gradual drift



(a) Data (b) Classifier variance (c) Accuracy

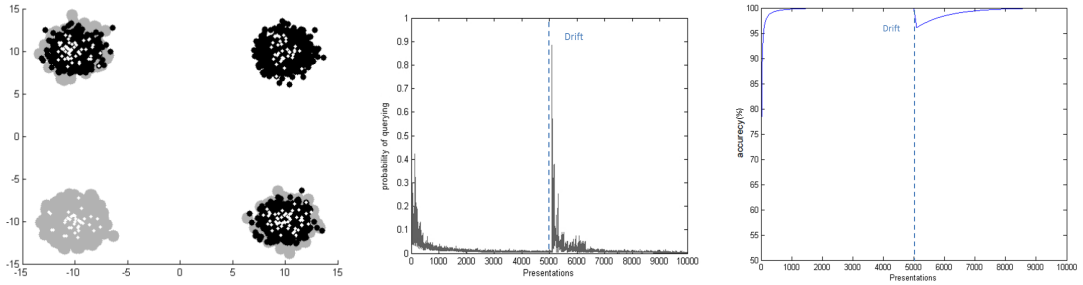
Figure. 2.6 BAL gradual drift



(a) Data (b) Querying probability (c) Accuracy

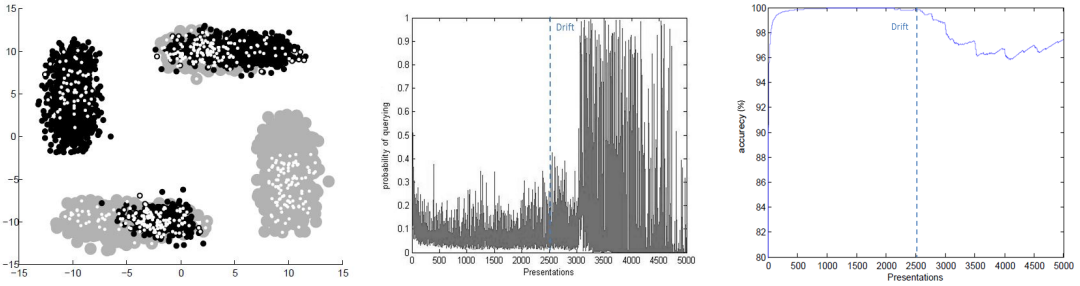
Figure. 2.7 UAL abrupt drift

center. However, it is clear that the number of queried data examples using BAL is larger. Figures 2.5b and 2.6b depict the determinant of the covariance $\boldsymbol{\Sigma_t}$ computed by Eq. (2.22). The variance is proportional to the classifier uncertainty. In particular, Fig. 2.5b and Fig. 2.6b show that the predictive model uncertainty is fluctuating over time. This fluctuation reflects the continuous data drift which proves the capability of the classifier to adapt to gradual drift. Higher uncertainty of the predictive model in Fig. 2.6b implies slow adaptation to drift. It is mainly caused by the uniform distribution of the data leading to inaccurate clustering. That is, the uncertainty criterion alone could be more valuable than its combination with the density criterion. Figures 2.5c and 2.6c show the accuracy over time. BAL uses more resources to obtain accuracy similar to UAL. In section B below,



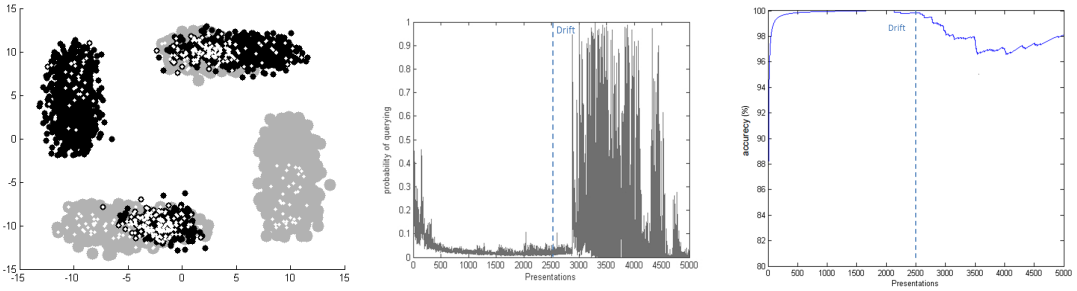
(a) Data (b) Querying probability (c) Accuracy

Figure. 2.8 BAL abrupt drift



(a) Data (b) Querying probability (c) Accuracy

Figure. 2.9 UAL mixture drift



(a) Data (b) Querying probability (c) Accuracy

Figure. 2.10 BAL mixture drift

this issue is investigated in detail.

After applying BAL and UAL on the Gaussian data with abrupt drifts, we obtain the results shown in Fig. 2.7 and Fig. 2.8. Figures 2.7b and 2.8b illustrate the querying probability over time highlighting the reaction of the model to change. In particular, Fig. 2.7b shows that the querying probability has a peak between the instance numbers 5000 and 6000; while the drift happens at sample 5000 which means that the model adapts to abrupt drift with some delay. Figure 2.8b shows that BAL reduces the querying probability and handles drift with less delay. Moreover, it is clear that BAL has better accuracy (compare Fig. 2.6c against Fig. 2.5c).

The results of UAL and BAL on mixture drift data are shown in Fig. 2.9 and Fig. 2.10. For both algorithms, it is clear that the majority of queried data is around the drifting regions (see Fig. 2.9a and Fig. 2.10a). Figure 2.10a shows slightly more concentrated queried data around the drifting regions using BAL. In contrast to abrupt drift, the probability of querying remains high because the gradual drift occurs after the abrupt drift. Figure 2.10b shows that BAL reduces the probability of querying when there is no drift and produces slightly better accuracy (compare Fig. 2.9c and Fig. 2.10c).

To sum up, BAL shows the ability to adapt to different types of drift but its performance may be affected by the data distribution. However, in the case of more complicated drift, BAL shows much more improvement as we will see also in the experiments related to real-world data in Sec. 2.5.3.

2.5.2 Performance Analysis

In the following, the effect of data distribution is studied in detail and the performance of both BAL and UAL with respect to the number of clusters is investigated. The notion of budget is also taken into account and its effect is analyzed.

Effect of the data distribution: In order to capture the data distribution's effect on BAL, we evaluate its performance on Gaussian (mixture drift) and uniformly (gradual drift) distributed data with respect to the number of clusters. Here, the budget is fixed to 0.05. Figure 2.11 illustrates the performance on both datasets. It is clear that incorporating density-based criterion improves the performance on the Gaussian distributed data.

For the case of uniformly distributed data, density-based criterion yields lower performance compared to UAL. However, for both datasets, the accuracy improves as the number of clusters increases (see Tab. 2.2). The uniform distribution is the most extreme break for the Gaussian assumption. However, BAL has good performance compared to UAL when remote drift occurs. Figure 2.12 shows that BAL can adapt faster to abrupt remote drift. Based on these experiments, BAL shows great potential in maintaining higher classification over time.

Effect of the budget: Table 2.3 illustrates the average accuracy of BAL compared to random sampling (baseline) on the three types of drift (gradual, abrupt

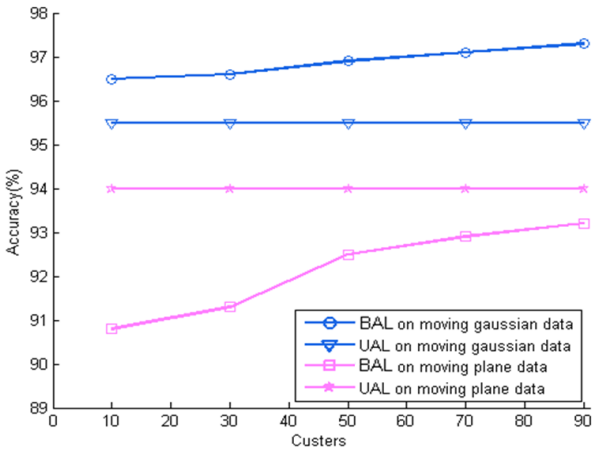


Figure. 2.11 Effect of the number of clusters on the performance

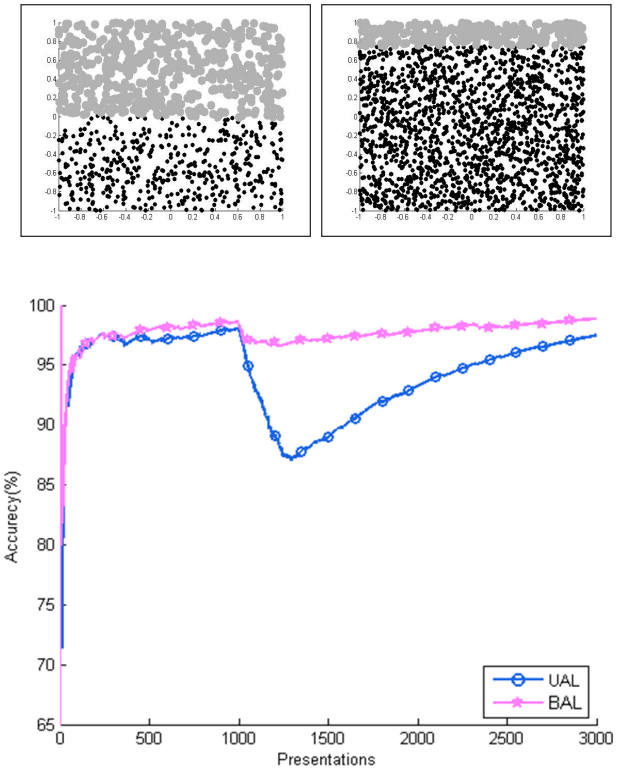


Figure. 2.12 Sensitivity to remote drift

and mixture) using different budgets. The values with star indicate that the additional budget is not used. To cope with abrupt drift, BAL does not need big budget to show high accuracy. However, budget has more impact on the accuracy in the case of gradual drift. Mixture drift requires less budget compared to the gradual drift and more budget compared to abrupt drift.

Table. 2.1 Clustering parameters (empirically obtained)

Datasets	Variance	Learning rate	Number of clusters
Gradual drift	0.1	0.01	50
Mixture drift	1	0.01	50
Electricity	0.5	0.6	50
Airlines	9	0.5	50

Table. 2.2 Effect of the number of clusters on BAL accuracy: case of plane and Gaussian data

Number of clusters	10	30	50	70	90
Accuracy (Gaussian)	96.5%	96.6%	96.9%	97.1%	97.3%
Accuracy (plane)	90.8%	91.3%	92.5%	92.9%	93.2%

Table. 2.3 Accuracy of BAL compared with random sampling using different budget values (Synthetic data)

Budget	0.01	0.05	0.1	0.2
Gradual drift (BAL)	80.35%	92.5%	93.5%	95.1%
Random sampling	80.7%	88.2%	91.6%	93%
Abrupt drift (BAL)	99.4%	99.4%*	99.4%*	99.4%*
Random sampling	96.5%	98.6%	99%	99.6%
Mixture drift (BAL)	94.6%	96.9%	96.9%*	96.9%*
Random sampling	92.6%	95%	96.9%	97.8%

Table. 2.4 Characteristics of the real-world datasets

Dataset	#Instances	#Features	#Classes
Airlines	539383	7	2
Electricity	45312	8	2

2.5.3 Simulation on real-world data

Two real-world benchmark datasets: Electricity and Airlines are used to evaluate BAL in more challenging settings. Their characteristics are shown in Tab. 2.4. Electricity data [88] is a popular benchmark used in evaluating classification in the context of data streams. The task is to predict the rise or fall of electricity prices (demand) in New South Wales (Australia), given recent consumption and prices in the same and neighboring regions. The Airlines dataset was collected by the USA flight control [89]. The task is to predict whether a flight will be delayed, given the information of the scheduled departure.

To illustrate the performance of BAL compared to the state-of-the-art active learning algorithms over data streams, two methods are considered: *Random* (baseline method) and *Variable Randomized Uncertainty* proposed in [1]. Unlike BAL, the

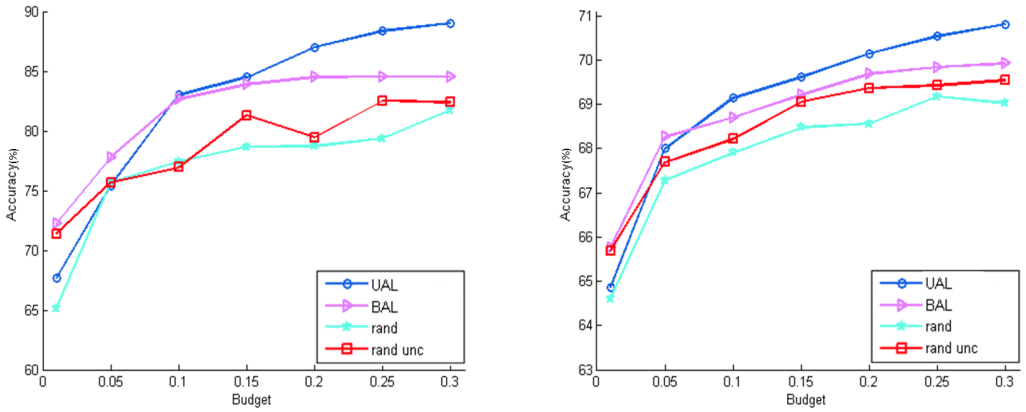
method described in [1] depends on the internal classification algorithm (learning engine). Here, we use Naive Bayes classifier as learning engine as in [1]. Only the first 10% of the whole data sequence for both datasets are used. The parameters of GGMM are empirically set to the values shown in Tab. 2.1.

In this experiment, the algorithms are evaluated using different budgets in $[0.01; 0.3]$. The final accuracy results are reported in Fig. 2.13 which show that UAL outperforms the rest of the competitors for high budget, while BAL works better for low budget (10 % for Electricity and $\tilde{7}$% for Airlines).

2.5.4 Discussion

While the performance of UAL and BAL are far better than the competitors, the application of the uncertainty criterion solely becomes more valuable as more data are queried (high budget). This finding was already shown in previous studies in the context of batch-based learning [74]. In the case of drifting data where the boundary between classes is very dynamic like with Electricity and Airlines datasets, the samples close to the boundary are more interesting since they drive the change in the distribution. Increasing the number of queries (i.e. high budget) will definitely enhance the accuracy irrespective of the data density, hence UAL performs better than BAL when the budget is high. This is in contrast to the case of drifting synthetic data where the simulations showed that overall BAL performs better and has a recovery speed faster than UAL.

In [1], authors claimed that the Electricity data has more aggressive drift compared to the Airlines data, which may explain the difference between the accuracy improvement associated with density-based criterion over the two datasets. BAL accuracy is better than UAL over the Airlines data only for budget less than 0.07, while it is up to 0.1 in the case of Electricity data. However, it is clear that the two proposed models UAL and BAL could be dynamically combined yielding one model that gives better result under any kind of drift over the whole budget line such that BAL selects the data to be queried for low budget, while UAL takes control for high budget.



(a) Electricity data

(b) Airlines data

Figure. 2.13 Results related to the real-world datasets

2.6 Conclusion

We proposed an active learning algorithm for data streams capable of dealing with changes of the data distribution. The algorithm labels the samples with high uncertainty and representativeness in a completely online scenario. It also tackles the *sampling bias* of active learning with potentially adversarial *concept drift*.

Experimental results on real-world data showed the limitation of the proposed approach when the budget is high or the drift occurrence is rare and smooth. However, the main goal of reducing the labelling cost in presence of *concept drift*, while maintaining good accuracy, has been achieved. Experimental results on synthetic data showed that as the data distribution becomes more uniform, more clusters are needed, and the time complexity increases. However, the maximum number of clusters is fixed, and we can have an upper bound on the time complexity. Therefore, based on the data stream velocity, we can decide the maximum number of clusters allowed. Finally, many experiments have been carried out in App. A.1 in order to come up with an approximation of local optimal values for GGMM's parameters. This is the main drawback of the proposed method and an improvement will be done in Chap. 3 by reducing the effect of parameter settings. A non-parametric model will be developed for this reason. We will also handle the *concept evolution* challenge.

Chapter 3

Active Learning for Data Streams under Concept Drift and Concept Evolution

In this chapter, we address the challenges of applying AL for data streams which are *concept drift*, *concept evolution* and *sampling bias* already found in Chap. 1. 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 take account of these concepts together. Moreover, using SAL, only labels of samples that minimize the expected future error are queried. The minimization 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, we propose 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 of online learning. In a nutshell, the main improvement achieved by SAL is that, in contrast to BAL, it uses a unified non-parametric Baysian model that can approximate both conditional and marginal distributions and accommodate growing complexity as more data is seen. Thus, the number of parameters need to be set will be reduced and the number of classes can dynamically changes allowing SAL to cope with *concept evolution*

The organization of this chapter is as follows. Section 3.1 presents an introduction. Section 3.2 describes SAL including the formulation of the objective function that it is intended to be optimised as well as the proposed AL approach. Section 3.3 provides the details of the proposed model to estimate the objective function described in Sect. 3.3. Section 3.4 discusses the experimental results for a number of well-known real-world datasets. Finally, Sec. 3.5 concludes the chapter.

3.1 Introduction

In this chapter, we relax the assumption of known and fixed number of classes taken in the previous chapter, where binary classification was performed. In fact, the evolving nature of data streams poses both *concept drift* and *concept evolution* challenges. *Concept evolution* occurs when new classes emerge or existing classes vanish. The classifier must be able to identify these new classes and incorporate them into the decision model [31, 91–93]. Emergence of new classes has been studied in the context of abnormality 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 [94, 95]. In contrast, *Concept evolution* involves the emergence of different normal and abnormal classes.

In this chapter, we propose an AL methodology, called Stream Active Learning (SAL) that addresses data stream challenges (i.e., *infinite length*, *concept drift* and *concept evolution*) and *sampling bias* in a unified and systematic way. In contrast to most of the existing AL approaches which adopt heuristic AL criteria, SAL aims at directly optimizing the expected future error [76]. This latter can be derived from Eq. (1.1) (see Chap. 1) by using a prediction error loss function. Similar AL approaches are proposed in [76, 96, 97]; however, they work in offline setting and do not take into account the challenges associated with data streams.

SAL adopts the same concept as BAL, but with some differences. While BAL has used existing classification and clustering algorithms, SAL uses a unified non-parametric Baysian model. The proposed model is a Dirichlet process mixture model [98] with a stick breaking prior [99] 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 [29, 100] to perform online inference.

As in Chap. 2 and [62], SAL handles *sampling bias* problem caused by AL using importance weighted empirical risk [66]. 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. In contrast to standard techniques for handling *concept drift* and *concept evolution* [32, 33, 67, 68], where only automatic detection mechanisms are applied, AL assumes that an oracle provides the true labels of data. 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. Similar technique was used in Chap. 2 and [62], but they are limited to binary classification. To the best of our knowledge, SAL is the first AL algorithm in the literature to consider both *concept drift* and *concept evolution*.

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*.

3.2 Active Learning Approach

Many active learning approaches seek to minimize an approximation of the expected error (Eq. (3.1)) [76, 96, 97]. SAL follows the same methodology, but with more challenging setting where the data comes as a stream,

$$R = \int_{\boldsymbol{x}} L\big(\hat{p}(\hat{y}|\boldsymbol{x}), p(y|\boldsymbol{x})\big) p(\boldsymbol{x}) d\boldsymbol{x} \tag{3.1}$$

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. (3.1) since a closed form solution is not available. Then, we present our online AL approach. Authors in [96] approximate the expected error using the empirical risk over the unlabelled data:

$$\hat{R}_{X_U}(\boldsymbol{\phi}_{X_L}) = \frac{1}{|X_U|} \sum_{\boldsymbol{x} \in X_U} L\big(p(\hat{y}|\boldsymbol{x}, \boldsymbol{\phi}_{D_L}), p(y|\boldsymbol{x})\big). \tag{3.2}$$

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.(3.2) 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 [62, 96, 97].

$$\tilde{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}\in X_U} \hat{R}_{X_{U-(\boldsymbol{x})}}(\boldsymbol{\phi}_{D_{L+(\boldsymbol{x},y)}}). \tag{3.3}$$

The empirical risk over the labelled and unlabelled samples is considered in [97]:

$$\hat{R}_D(\boldsymbol{\phi}_{D_L}) = \frac{1}{|D|}\sum_{\boldsymbol{x}\in D} L\big(p(\hat{y}|\boldsymbol{x},\boldsymbol{\phi}_{D_L}), p(y|\boldsymbol{x})\big). \tag{3.4}$$

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

$$\hat{R}_{D_L}(\boldsymbol{\phi}_{D_L}) = \frac{1}{|D_L|}\sum_{\boldsymbol{x}\in D_L} L\big(p(\hat{y}|\boldsymbol{x},\boldsymbol{\phi}_{D_L}), p(y|\boldsymbol{x})\big). \tag{3.5}$$

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, $q \sim Bern(a)$. 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 (3.5) is a biased estimator of (3.1) and the learned classifier may be less accurate than when learned without using AL (*Sampling bias*). Similar to Chap. 2 and [62], we use the *importance weighting* technique [66] in order to come up with an unbiased estimator. Thus, Eq. (3.5) can be written as follows:

$$\hat{R}'_{D_L}(\boldsymbol{\phi}_{D_L}) = \frac{1}{|D_L|}\sum_{\boldsymbol{x}\in D_L} \frac{1}{p(q=1|\boldsymbol{x})} L\big(p(\hat{y}|\boldsymbol{x},\boldsymbol{\phi}_{D_L}), p(y|\boldsymbol{x})\big). \tag{3.6}$$

Hence, the unbiased estimation above can be shown as:

$$E_{\boldsymbol{x}\sim p(\boldsymbol{x}|q=1)}\big[\hat{R}'_{D_L}(\boldsymbol{\phi}_{D_L})\big] = E_{x\sim p(\boldsymbol{x})}\big[\hat{R}_{D_L}(\boldsymbol{\phi}_{D_L})\big]. \tag{3.7}$$

So far, 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.(3.3)) 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. (3.4) equipped with the *importance weighting* on the labelled data can be written as the sum of the following equations:

$$\hat{R}_{D_t}(\boldsymbol{\phi}_t) = \int_{\boldsymbol{x}} L\big(p(\hat{y}|\boldsymbol{x}, \boldsymbol{\phi}_t), p(y|\boldsymbol{x}, D_t)\big) p(\boldsymbol{x}|X_{U_t}) d\boldsymbol{x} \tag{3.8}$$

$$\hat{R}'_{D_t}(\boldsymbol{\phi}_t) = \int_{\boldsymbol{x}} \frac{L\big(p(\hat{y}|\boldsymbol{x}, \boldsymbol{\phi}_t), p(y|\boldsymbol{x}, D_t)\big)}{p(q=1|\boldsymbol{x})} p(\boldsymbol{x}|X_{L_t}) d\boldsymbol{x} \tag{3.9}$$

where $\boldsymbol{\phi}_t$ denotes the classifier parameters after being trained on D_{L_t}. Based on Eq. (3.8) and Eq. (3.9), we can develop an online querying strategy similar to the one proposed in Eq. (3.3). 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. Furthermore, we need to compute the expectation of the error reduction because the labels are unknown which is again computationally very demanding given that the labels are unknown.

We can conclude from Eq. (3.8), (3.9) that the error can be reduced by labelling the samples that have the largest contribution to the current error. This contribution can be expressed through the following equations:

$$\hat{R}_{D_{t-1}}(\boldsymbol{\phi}_{t-1}, \boldsymbol{x}_t) = L\big(p(\hat{y}_t|\boldsymbol{x}_t, \boldsymbol{\phi}_{t-1}), p(y_t|\boldsymbol{x}_t, D_{t-1})\big) p(\boldsymbol{x}_t|X_{U_{t-1}}) \tag{3.10}$$

$$\hat{R}'_{D_{t-1}}(\boldsymbol{\phi}_{t-1}, \boldsymbol{x}_t) = L\big(p(\hat{y}_t|\boldsymbol{x}_t, \boldsymbol{\phi}_{t-1}), p(y_t|\boldsymbol{x}_t, D_{t-1})\big) \tilde{p}(\boldsymbol{x}_t|X_{L_{t-1}}) \tag{3.11}$$

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. 3.3). 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 (3.10) encourages querying samples that have strong representativeness among the unlabelled data and that are expected to be wrongly classified; while

Eq.(3.11) encourages querying those which have strong representativeness among the labelled data, but, still wrongly classified. Such samples are rare. However, Eq. (3.11) 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. (3.11) 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.(3.10) and Eq.(3.11). 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:

$$A_t = \max\left((\hat{R}_{D_{t-1}}(\boldsymbol{\phi}_{t-1}, \boldsymbol{x}_t) + \hat{R}'_{D_{t-1}}(\boldsymbol{\phi}_{t-1}, \boldsymbol{x}_t)), \beta A_{t-1}\right) \tag{3.12}$$

$$p(q_t = 1|\boldsymbol{x}_t, D_{t-1}, \boldsymbol{\phi}_{t-1}) = \frac{1}{A_t}\left(\hat{R}_{D_{t-1}}(\boldsymbol{\phi}_{t-1}, \boldsymbol{x}_t) + \hat{R}'_{D_{t-1}}(\boldsymbol{\phi}_{t-1}, \boldsymbol{x}_t)\right) \tag{3.13}$$

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. (3.10) and Eq. (3.11)) 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. (3.10) and Eq. (3.11). 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. (3.10) and Eq. (3.11) are high if the probability of a new class is high. Hence, the

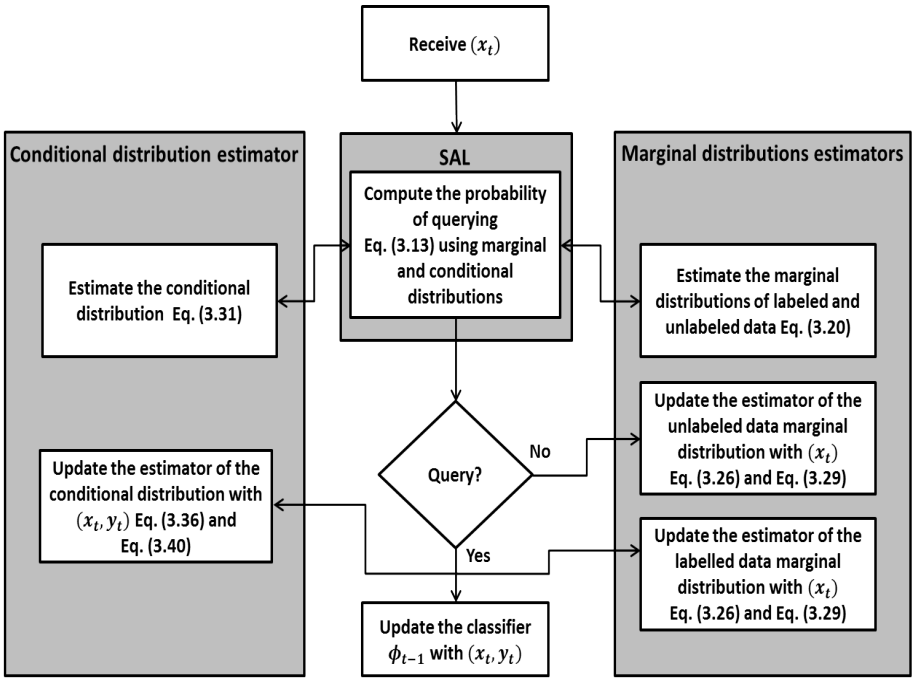


Figure. 3.1 General scheme of SAL

probability of querying (Eq. (3.13)) becomes high. We use the 0-1 loss funtion:

$$l(\hat{y}_t, y_t) = \begin{cases} 0 & if \quad \hat{y}_t = y_t \\ 1 & otherwise \end{cases} \tag{3.14}$$

hence, the loss in Eq. (3.10) and Eq. (3.11) can be rewritten as follows:

$$L\big(p(\hat{y}_t|\boldsymbol{x}_t, \boldsymbol{\phi}_{t-1}), p(y_t|\boldsymbol{x}_t, D_{t-1})\big) = E_{\hat{y}_t \sim p(\hat{y}_t|\boldsymbol{x}_t, \boldsymbol{\phi}_{t-1})}\big[E_{y_t \sim p(y_t|\boldsymbol{x}_t, D_{t-1})}[l(\hat{y}_t, y_t)]\big]. \tag{3.15}$$

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.3.1. Similar to Chap. 2, SAL adopts the notion of budget; so that it can be assessed against the active learning algorithms proposed in [1]. Details about this notion can be found in Sec. 2.3.2.

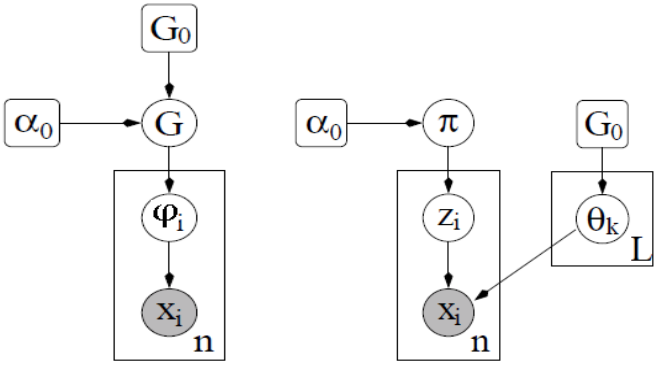
3.3 Estimator Model

In this section, we develop the model that will be used to estimate the distributions (Eq. (3.10) and Eq. (3.11)) needed for SAL to work online. First, 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. Second, 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 [101] over the classes. These estimations are done on-the-fly by performing online inference using the particle inference algorithm.

3.3.1 Dirichlet process

DP is one of the most popular prior used in Bayesian non-parametric modelling. It was first used by the machine learning community in [102, 103]. In general, stochastic process is probability distribution over a space of paths which describe the evolution of some random value 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. In the literature, DP has been constructed with different ways, the most well-known constructions are: infinite mixture model [103], distribution over distribution [104], Polya-urn scheme [105] and stick-breaking [101]. For more details, interested reader is referred to [106].

Figure 3.2 shows two graphical models, DP mixture model and the finite mixture model with a 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:



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

Figure. 3.2 Graphical model

$$\begin{aligned} \boldsymbol{\pi}|\alpha_0 &\sim Dirichlet(\alpha_0/L, ..., \alpha_0/L) \\ z_i|\boldsymbol{\pi} &\sim Discrete(\pi_1, ..., \pi_L) \\ \boldsymbol{\theta_k}|G_0 &\sim G_0 \\ \boldsymbol{x_i}|z_i, \boldsymbol{\theta} &\sim F(\boldsymbol{\theta}_{\boldsymbol{z_i}}) \end{aligned} \tag{3.16}$$

$F(\boldsymbol{\theta}_{\boldsymbol{z_i}})$ 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 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. (3.16) 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 $GEM(\alpha_0)$ [101]. Thus, Eq. (3.16) can be equivalently expressed according to the graphical representation as follows:

$$\begin{aligned} G|\alpha_0, G_0 &\sim DP(\alpha_0, G_0) \\ \boldsymbol{\theta_i}|G &\sim G \\ \boldsymbol{x_i}|\boldsymbol{\theta_i} &\sim F(\boldsymbol{\theta_i}) \end{aligned} \tag{3.17}$$

where $G = \sum_{k=1}^{\infty} \pi_k \delta_{\boldsymbol{\theta_i}}$ is drawn from the DP prior. $\delta_{\boldsymbol{\theta_i}}$ is a Dirac delta function centred at $\boldsymbol{\theta_i}$. Technically, DP is a distribution over distributions [104], where

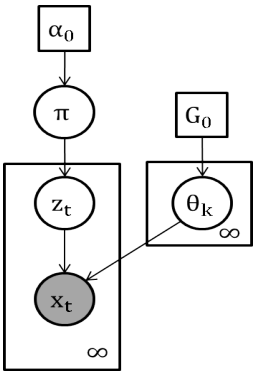


Figure. 3.3 Infinite mixture model

$DP(G_0, \alpha)$, is parametrized by the base distribution G_0, and the concentration parameter α. 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. (3.16), by integrating over the mixing proportion $\boldsymbol{\pi}$, we can write the prior for z_i as conditional probability of the following form [107]:

$$p(z_i = c|z_1, ..., z_{i-1}) = \frac{n_c^{-i} + \alpha_0/L}{i - 1 + \alpha_0} \tag{3.18}$$

where n_c^{-i} is the number of data samples excluding $\boldsymbol{x}_i$ that are assigned to component c. By letting L goes to infinity we get the following equations:

$$\begin{aligned} p(z_i = c|z_1, ..., z_{i-1}) &\rightarrow \frac{n_c^{-i}}{i - 1 + \alpha_0} \\ p(z_i \neq z_j \, for \, all \, j < i|z_1, ..., z_{i-1}) &\rightarrow \frac{\alpha_0}{i - 1 + \alpha_0} \end{aligned} \tag{3.19}$$

For an observation $\boldsymbol{x_i}$ with $z_i \neq z_j \, for \, all \, j < i$, a new component is created with indicator $z_i = c_{new}$. For more details about the process of obtaining the prior distribution, the reader is referred to [107].

3.3.2 Proposed Model

For the sake of simplification, we start with an unsupervised clustering model, then we add a new ingredient to the model in order to accommodate labelled data resulting in a semi-supervised clustering model.

3.3.2.1 Clustering

Figure 3.3 shows the infinite mixture model where $\boldsymbol{\pi}$ is drawn from a stick-breaking process $GEM(\alpha)$ 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 [29], 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 $X_{t-1} \equiv \boldsymbol{x}_{1:t-1}$:

$$p(\boldsymbol{x}_t|X_{t-1}) = \sum_{z_{1:t}} p(\boldsymbol{x}_t|z_{1:t}, X_{t-1}) p(z_{1:t}|X_{t-1}) \tag{3.20}$$

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.(3.20) by simply replacing X_{t-1} by $X_{U_{t-1}}$ or $X_{L_{t-1}}$.

$$p(z_{1:t}|X_{t-1}) = p(z_t|z_{1:t-1}) p(z_{1:t-1}|X_{t-1}) \tag{3.21}$$

The first term of Eq. (3.20) can be written as follows:

$$p(\boldsymbol{x}_t|z_{1:t}, X_{t-1}) = \int_{\boldsymbol{\theta}} p(\boldsymbol{x}_t|\boldsymbol{\theta}, z_t) p(\boldsymbol{\theta}|z_{1:t}, X_{t-1}) \tag{3.22}$$

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.

$$\begin{aligned} \boldsymbol{su}_{z_t,t-1}(z_{1:t-1}) &= \frac{\sum_{z_i=z_t,i<t} \boldsymbol{x}_i}{n_{z_t,t-1}} \\ \boldsymbol{sc}_{z_t,t-1}(z_{1:t-1}) &= \sum_{z_i=z_t,i<t} (\boldsymbol{x}_i - \boldsymbol{su}_{z_t,t-1})(\boldsymbol{x}_i - \boldsymbol{su}_{z_t,t-1})^T \end{aligned} \tag{3.23}$$

Equation (3.22) can be solved given the sufficient statistics, the past assignments and the model hyper-parameters. More details can be found in App. B.1. The first term of Eq. (3.21) can be computed in the same way as Eq. (3.19):

$$p(z_t|z_{1:t-1}) \propto \begin{cases} n_{z_t,t-1} & z_t \text{ is an existing cluster} \\ \alpha_0 & z_t \text{ is a new cluster} \end{cases} \tag{3.24}$$

The second term of Eq. (3.21), $p(z_{1:t-1}|X_{t-1})$, 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 [100] which retains the maximum N particles. We approximate the posterior at time t in two steps:

Updating:

$$p(H_t|X_t) \propto \int_{H_{t-1}} p(H_t|H_{t-1}, \boldsymbol{x}_t) p(\boldsymbol{x}_t|H_{t-1}) p(H_{t-1}|X_{t-1}) \tag{3.25}$$

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.

$$p(H_t|\boldsymbol{x}_{1:t}) \propto \sum_{i=1}^{N} p(H_t|H_{t-1}^{(i)}, \boldsymbol{x}_t) p(\boldsymbol{x}_t|H_{t-1}^{(i)}) w_{t-1}^{(i)} \tag{3.26}$$

The solution of the second term of Eq. (3.26) can be computed in a similar way to Eq. (3.22) and Eq. (3.24). 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,

$$\begin{aligned} p(H_t^{(i_2)}|H_{t-1}^{(i)}, \boldsymbol{x}_t) &= p(z_t = j|H_{t-1}^{(i)}, \boldsymbol{x}_t) \\ &\propto p(\boldsymbol{x}_t|z_t = j, H_{t-1}^{(i)}) p(z_t = j|H_{t-1}^{(i)}) \end{aligned} \tag{3.27}$$

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

$$\begin{aligned} i_2 &= \Omega(i, j) \\ \Omega(i, j) &= \frac{1}{2}(i+j)(i+j+1) + j \end{aligned} \tag{3.28}$$

By solving Eq. (3.27), we determine the weights of the new particle $w_t^{i_2}$. Equation (3.27) can be solved in a similar way to Eq. (3.22) and Eq. (3.24). The elements of the new state vector $H_t^{(i_2)}$ are updated as follows:

$$H_t^{(i_2)} = \begin{cases} z_t^{(i_2)} & = j \qquad\qquad j \text{ is an existing component} \\ n_{j,t}^{(i_2)} & = \lambda n_{j,t-1}^{(i)} + 1 \\ n_{k,t}^{(i_2)} & = \lambda n_{k,t-1}^{(i)} \quad \forall k \neq j, k \leq m_t^{(i)} \\ \boldsymbol{su}_{j,t}^{(i_2)} & = \frac{\lambda n_{j,t-1}^{(i)} \boldsymbol{su}_{j,t-1}^{(i)} + \boldsymbol{x}_t}{n_{j,t}^{(i)}} \\ \boldsymbol{sc}_{j,t}^{(i_2)} & = \lambda \boldsymbol{sc}_{j,t-1}^{(i)} + n_{j,t-1}^{(i)} \boldsymbol{su}_{j,t-1}^{(i)} \boldsymbol{su}_{j,t-1}^{(i)T} \\ & \quad -n_{j,t}^{(i)} \boldsymbol{su}_{j,t}^{(i)} \boldsymbol{su}_{j,t}^{(i)T} + \boldsymbol{x}_t \boldsymbol{x}_t^T \\ & \\ z_t^{(i_2)} & = m_{t-1}^{(i)} + 1 \qquad j \text{ is a new component} \\ m_t^{(i_2)} & = m_{t-1}^{(i)} + 1 \\ n_{j,t}^{(i_2)} & = 1 \\ n_{k,t}^{(i_2)} & = \lambda n_{k,t-1}^{(i)} \quad \forall k \leq m_{t-1}^{(i)} \\ \boldsymbol{su}_{j,t}^{(i_2)} & = \boldsymbol{x}_t \\ \boldsymbol{sc}_{j,t}^{(i_2)} & = \boldsymbol{0} \end{cases} \tag{3.29}$$

where λ is a forgetting factor which allows the components to adapt with changes. Having approximated all the terms of Eq.(3.26), 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

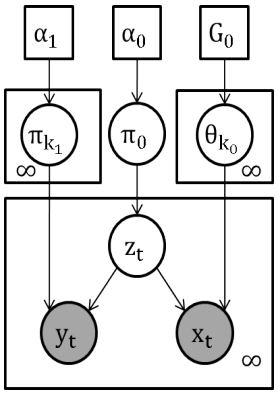


Figure. 3.4 Proposed semi-supervised clustering model

reduces the number of created particles to a fix number N.

Re-sampling:

We follow the re-sampling technique proposed in [100] 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:

$$\sum_{i_2=1}^{M} \min\{cw_t^{(i_2)}, 1\} = N \tag{3.30}$$

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.

3.3.2.2 Semi-supervised classifier

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.3.4).

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.

$$p(y_t|\boldsymbol{x}_t, D_{t-1}) = \sum_{z_t} p(y_t|z_t, D_{t-1})p(z_t|D_{t-1}, \boldsymbol{x}_t). \tag{3.31}$$

The first term of Eq. (3.31) can be computed in a similar way to Eq. (3.24), 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 App. B.2.

$$p(y_t|z_t, D_{t-1}) \propto \begin{cases} n'_{z_t,y_t,t} & y_t \text{ is an existing class} \\ \alpha_1 & y_t \text{ is a new class} \end{cases} \tag{3.32}$$

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 Eq. (3.32), the distribution of the labels to the data in each component must be memorized. The second term of Eq. (3.31) can be written as follows:

$$p(z_t|\boldsymbol{x}_t, D_{t-1}) = \sum_{z_{1:t-1}} p(z_t|z_{1:t-1}, \boldsymbol{x}_t, D_{t-1})p(z_{1:t-1}|\boldsymbol{x}_t, D_{t-1}) \tag{3.33}$$

$$p(z_t|z_{1:t-1}, \boldsymbol{x}_t, D_{t-1}) \propto p(\boldsymbol{x}_t|z_{1:t}, D_{t-1})p(z_t|z_{1:t-1}) \tag{3.34}$$

$$p(z_{1:t-1}|\boldsymbol{x}_t, D_{t-1}) \propto p(\boldsymbol{x}_t|z_{1:t-1}, D_{t-1})p(z_{1:t-1}|D_{t-1}) \tag{3.35}$$

Equation (3.33) can be solved by following similar steps to Eq. (3.20) 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. (3.22) after replacing H_{t-1} by H'_{t-1}. The second term of Eq. (3.35), $p(z_{1:t-1}|D_{t-1})$, has the same probability as the posterior $p(H_{1:t-1}|D_{t-1})$. Thus, similar to Eq.(3.26), the solution of Eq. (3.31) depends only on the elements of the state vector H'_t along with its posterior distribution. We track this posterior online. Similar to Sec. 3.3.2.1, 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. 3.3.2.1. The updating step follows the same way:

$$p(H'_t|D_t) \propto \sum_{i=1}^{N} p(H'_t|H'^{(i)}_{t-1}, \boldsymbol{x}_t, y_t)p(\boldsymbol{x}_t, y_t|H'^{(i)}_{t-1})w'^{(i)}_{t-1} \tag{3.36}$$

$$\begin{aligned} p(H'^{(i_2)}_t|H'^{(i)}_{t-1}, \boldsymbol{x}_t, y_t) &= p(z_t = j|H'^{(i)}_{t-1}, \boldsymbol{x}_t, y_t) \\ &\propto p(\boldsymbol{x}_t, y_t|z_t = j, H'^{(i)}_{t-1})p(z_t = j|H'^{(i)}_{t-1}) \end{aligned} \tag{3.37}$$

$$p(\boldsymbol{x}_t, y_t|z_t = j, H'^{(i)}_{t-1}) = p(\boldsymbol{x}_t|z_t = j, H'^{(i)}_{t-1})p(y_t|z_t = j, H'^{(i)}_{t-1})) \tag{3.38}$$

The second term of Eq.(3.37) is computed in Eq.(3.24). The first and second terms of Eq. (3.38) can be solved in the same way as in Eq. (3.22) and Eq. (3.32) respectively. The second term of Eq. (3.36) can be written as follows:

$$p(\boldsymbol{x}_t, y_t|H'^{(i)}_{t-1}) = \sum_{z_t} p(\boldsymbol{x}_t, y_t|z_t, H'^{(i)}_{t-1})p(z_t|H'^{(i)}_{t-1}) \tag{3.39}$$

The elements of the new state vector are updated in the same way as in Eq.(3.29):

$$H'^{(i_2)}_t = \begin{cases} H^{(i_2)}_t & j \text{ is an existing component} \\ n'^{(i_2)}_{j,y_t,t} & = \lambda' n'^{(i)}_{j,y_t,t-1} + 1 \\ \boldsymbol{n}'^{(i_2)}_{k,t} & = \lambda' \boldsymbol{n}'^{(i)}_{k,t-1} \quad \forall k \neq j, k \leq m^{(i)}_t \\ & \\ H^{(i_2)}_t & j \text{ is a new component} \\ n'^{(i_2)}_{j,y_t,t} & = 1 \\ \boldsymbol{n}'^{(i_2)}_{k,t} & = \lambda' \boldsymbol{n}'_{k,t-1} \quad \forall k \leq m^{(i)}_{t-1} \end{cases} \tag{3.40}$$

The estimation of $p(y_{t+1}|\boldsymbol{x}_{t+1}, D_t))$ in SAL is computed by Eq. (3.31). The estimation of $p(\boldsymbol{x}_t|X_{U_{t-1}})$ and $p(\boldsymbol{x}_t|X_{L_{t-1}})$ are computed by Eq. (3.20). We maintain three state vectors, one for the unlabelled data H^u_{t-1}, one for the labelled data H^l_{t-1} 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^u_t, w^u_t) and $(H^l_t, w^l_t)\}$ and their hyper-parameters: $\{(\alpha_0, \alpha_1, \boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0, k_0, v_0)$, $(\alpha^u_0, \alpha^u_1, \boldsymbol{\mu}^u_0, \boldsymbol{\Sigma}^u_0, k^u_0, v^u_0)$ and $(\alpha^l_0, \alpha^l_1, \boldsymbol{\mu}^l_0, \boldsymbol{\Sigma}^l_0, k^l_0, v^l_0)\}$ are maintained.

Having introduced the AL approach in Sec. 3.2 and developed the needed estimator model in Sect 3.3, the full details of the algorithm are provided in Alg. (2).

3.4 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

Algorithm 2 Steps of SAL

Input: data stream, hyper-parameters of the estimators, forgetting factors λ and λ' (Eq. (3.29) and Eq. (3.40) resp.), the input of the classifier, forgetting factor β (Eq. (3.12)), budget B, wnd (Eq. (2.44)), the maximum number of particles N
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. (2.44)), $A_0 = 0$ (Eq. (3.12)), $t = 1$
while (true) **do**
  $t \leftarrow t + 1$,
  Receive $\boldsymbol{x_t}$
  if $\hat{b}_t < B$(Eq. (2.44)){enough budget} **then**
    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 $\boldsymbol{x}_t$ (Eq. (3.13))
    $q_t \sim Bern(a_t)$
    if $q_t = 1$ {querying} **then**
      $y_t \leftarrow query(\boldsymbol{x_t})$
      Update the classifier (represented by its vector parameter $\boldsymbol{\phi}_{t-1}$)
      Update the estimator of the labelled data distribution (represented by its state vector's particles along with their weights $(H_{t-1}^{l(i)}, w_{t-1}^{l(i)})$) (Eq. (3.26) and Eq. (3.29))
      Update the estimator of the conditional distribution (represented by its state vector's particles along with their weights $(H_{t-1}^{\prime(i)}, w_{t-1}^{\prime(i)})$) (Eq. (3.36) and Eq. (3.40))
    else
      Classify the instance $\boldsymbol{x}_t$ using the classifier (base learner represented by its vector parameter $\boldsymbol{\phi}_{t-1}$)
      Update the estimator of the unlabelled data distribution (represented by its state vector's particles along with their weights $(H_{t-1}^{u,(i)}, w_{t-1}^{u,(i)})$) (Eq. (3.26) and Eq. (3.29))
    end if
  end if
  Compute $\hat{f}_{t+1}$ (Eq. (2.45))
end while

against two types of stream-based AL approaches. The first set of approaches introduced in [1] 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 [108] 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* [2, 3]. However, they do not explicitly handle *concept drift* as well as the *sampling bias* problem. We consider the following:

- *lowlik*: Low-likelihood criterion specialized for quick unknown class discovery [2].
- *qbc*: Query-by-Commitee, a stream-based version proposed by [3].
- *qbc-pyp*: Stream-based joint exploration-exploitation AL proposed in [2].

These methods have shown good class discovery performance on unbalanced 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 [3]. 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 [1]. 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.

3.4.1 datasets

SAL is evaluated on six real-world benchmark datasets widely used in the AL area: Thyroid, Covertype (Forest), KDDCup 99 network intrusion detection (KDD), Electricity, Airlines and MNIST handwritten digits (Digits). The first three of

Table. 3.1 Benchmark Datasets properties used for comparing SAL against [1]

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. 3.2 Benchmark Datasets properties used for comparing SAL against [2, 3]

Datasets	N	d	N_c	$S\%$	$L\%$	E	CD	CE
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

these datasets are downloaded from UCI repository [109]. Electricity [88] and Airlines [89] 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.3.1 and Tab.3.2, 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 [1], 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*. 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 [110]

In this work, we use the full Thyroid and Electricity datasets but only portions of Forest, KDD and Digits datasets. When comparing against the first type of competitors (*VarUn*, *RanVarUn* and *Rand*) [1], 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*) [2, 3], we follow the setting in [3] where 33650 and 5000 instances are used from KDD and Forest respectively.

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

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

3.4.2 Classification performance

In this section, we evaluate SAL against the methods presented in [1] on all datasets (see Tab. 3.1) 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 which is given as follows: Each time we get an instance, we first compute the probability of querying before using it to train the classifier if queried. Otherwise, it is used for testing the classifier. The classification performance of SAL is measured according to the average accuracy which is the ratio of correctly classified testing samples:

$$AA = \frac{\sum_{i \in T} 1_{(\hat{y}_i, y_i)}}{|T|} \tag{3.41}$$

$$1_{(\hat{y}_i, y_i)} = \begin{cases} 1 & if \quad \hat{y}_i = y_i \\ 0 & otherwise \end{cases} \tag{3.42}$$

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.

3.4.2.1 Settings

The hyper-parameters of the estimator model are fixed apriori (see Tab. 3.3). 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 as large as

Table. 3.3 Learning rate parameters

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

Table. 3.4 SAL hyper-parameters setting

Hyper-parameters	α_0 α_0^u α_0^l	μ_0 μ_0^u μ_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

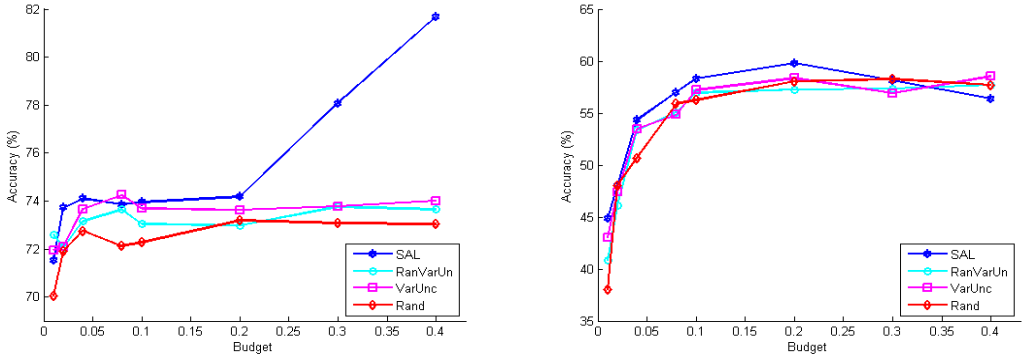
the dispersion of the data. The degree of freedom of the Wishart distributions v_0, v_0^u , v_0^l must be greater than d. We set them to $d+2$. The hyper-parameters k_0, k_0^u and k_0^l are empirically set to 0.01. Because at this stage, we are interested only in the classification error, the hyper-parameter α_1 which controls the prior over the classes is set to a low value. It can be seen from Eq. (3.32) 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.(3.29) and Eq.(3.40) on SAL's performance is studied and the parameters are set to the values that give the best performance (see Table 3.4). Note that we also consider the best results of the competitors.

3.4.2.2 Performance Analysis

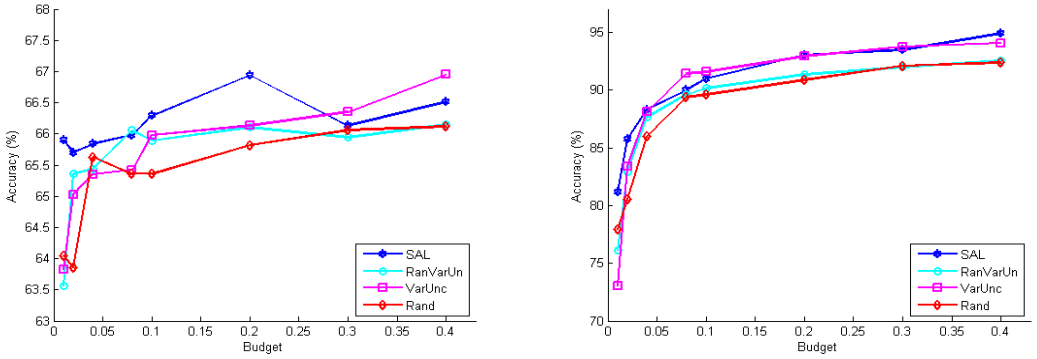
Following similar setting in [1], we carry out the experiments on the five datasets shown in Tab. 3.1 using different budget values. Figure 3.5 shows the classification accuracy of both SAL and the competitors for different values of budget B.

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 where *VarUn* slightly outperforms SAL. 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 [1]. Having frequent aggressive drift makes *sampling bias* more likely to occur. Indeed, AL's confident sampling assessment is more likely to miss

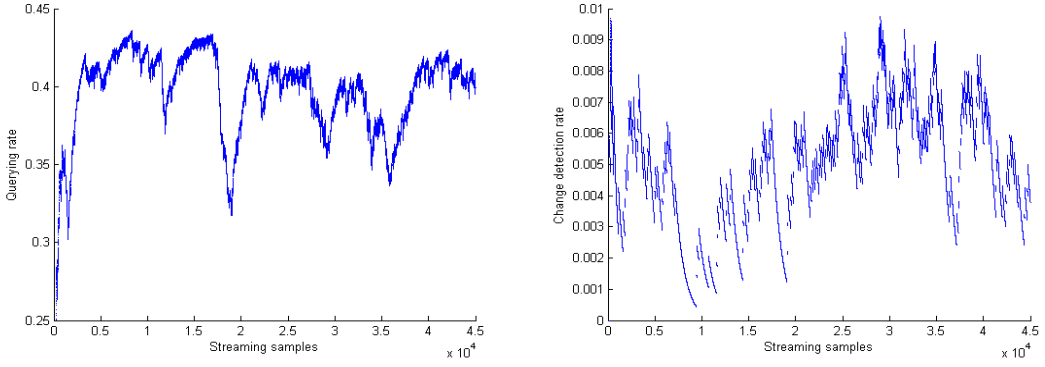


(a) Electricity (b) Forest



(c) Airlines (d) Digits

Figure. 3.5 Classification performance



(a) Querying rate (b) Drift rate

Figure. 3.6 Active learning behaviour along the stream for Electricity

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 3.6 shows SAL behaviour along the stream of Electricity dataset with budget fixed at 0.4. Figure. 3.6a presents the prequential labelling rate, and Fig.3.6b presents the prequential drift rate. To smooth the curve, a fading factor of 0.999 is used. The drift is detected using the drift detection method (DDM) proposed by [30]. 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.

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 minimise 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.

3.4.3 Class discovery performance

In this section, we evaluate SAL against the methods in [2, 3] on datasets that show multiple classes in naturally unbalanced proportions (see Tab. 3.2). 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 [110] which is given as:

$$AA_j = \frac{\sum_{i \in T_j} 1_{(\hat{y}_i, j)}}{|T_j|} \tag{3.43}$$

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

$$ACA = \frac{\sum_{j \in C} AA_j}{|C|} \tag{3.44}$$

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.

3.4.3.1 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.(3.14) is formulated as follows:

$$l(\hat{y}_t, y_t) = \begin{cases} 0 & if \quad \hat{y}_t = y_t \\ \frac{1}{s(y_t)} & otherwise \end{cases} \tag{3.45}$$

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:

$$\boldsymbol{nb} = fr * \boldsymbol{nb} + \mathbf{1}_{y_t} \tag{3.46}$$

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 expect for the element indexed by y_t which is equal to 1.

$$s(y_t) \propto \begin{cases} \boldsymbol{nb}_{y_t} & \forall y_t \in C_{t-1} \\ 1 & y_t \quad \text{is a new class} \end{cases} \tag{3.47}$$

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

Table. 3.5 Number of classes discovered by different methods

No	Datasets	N_c	*lowlik*	*qbc*	$qbc - pyp$	*SAL*
1	Thyroid	3	2.92	2.68	**3**	**3**
2	Forest	7	7	7	7	7
3	KDD	10	**9.76**	3.32	8.71	8
4	Digits	10	8.84	8.84	7.76	8.5

Table. 3.6 Average class accuracy achieved using different methods

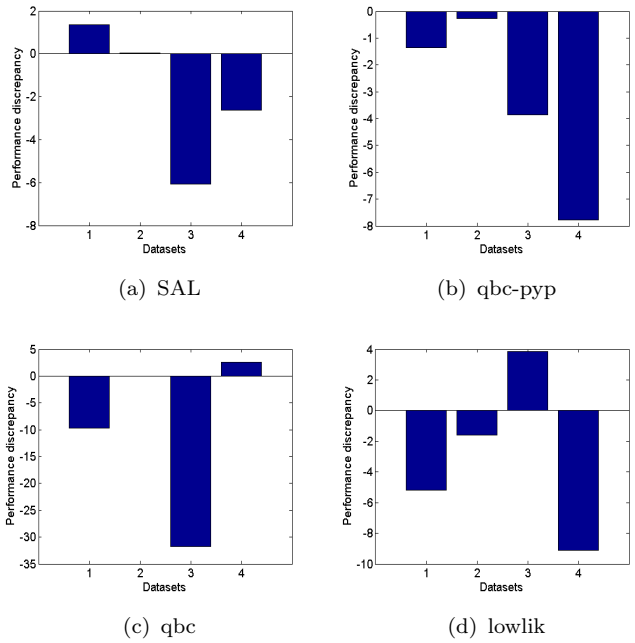
No	Datasets	*lowlik*	*qbc*	$qbc - pyp$	*SAL*
1	Thyroid	54.62	50.07	58.45	**59.8**
2	Forest	57.13	58.68	58.45	**58.71**
3	KDD	**51.21**	19.42	47.35	45.14
4	Digits	48.94	**58.04**	50.27	55.42
	Overall	53.73	46.55	53.63	**54.77**

All the parameters of the estimator model excluding α_1 are set to the same values as in the previous section (see Tab. 3.3). 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.(3.32)). We studied its effect on each dataset. So, it is set to 0.5 for 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 Thyroid and KDD datasets.

3.4.3.2 Performance Analysis

In these experiments, we follow the same setting as in the competitors [2, 3], 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.3.5, Tab.3.6 and Fig. 3.7. Table 3.5 presents the number of classes discovered by the different methods. Table 3.6 presents the average class accuracy ACA achieved using the different methods. Figures 3.7 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.



(a) SAL (b) qbc-pyp

(c) qbc (d) lowlik

Figure. 3.7 Comparison of the class discovery performance

The results show that SAL provides a comparable class discovery performance compared to the competitors. SAL was able to discover all the classes for Thyroid and Forest datasets. For the KDD and Digits data, around 8 out of 10 and 8.5 out of 10 classes are discovered respectively (see Tab.3.5). SAL's average class accuracy is the best on the Forest and Thyroid datasets which may be explained by the fact that the proportion of the smallest classes are higher than the others. 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 two datasets (Thyroid and Forest); while each competitor gives the best result on one dataset (see Fig. 3.7).

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 strong competitors

which some of them are specialized for detecting the outliers as novel classes. 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.

3.5 Conclusion

We propose an active learning algorithm, SAL, for data streams to deal 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 are very good in general. Unfortunately, SAL shows some limitations for class discovery when applied to highly unbalanced data. However, the main goal of the proposed algorithm is for classification in presence of unknown number of classes not specifically for unbalanced data. Furthermore, the performance of SAL with respect to class discovery is comparable to the state-of-the-art and is even better when the classes are not severely unbalanced. To further investigate this problem with unbalanced data, we introduce in Chap. 4 an information-based AL algorithm capable of coping with *concept evolution* in data streams involving classes of severely unbalanced proportion.

Chapter 4

Active Learning for Sequential Data Streams: An Application to Human Activity Recognition

The contribution of this chapter is two-fold. Firstly, we propose a novel online AL algorithm that can handle *concept evolution* in data stream involving classes of unbalanced proportions. Secondly, we propose an algorithm equipped with the proposed AL algorithm to deal data streams with sequential temporal dependency. The proposed AL algorithm is Bayesian stream-based; hence we call it BSAL. It can cope with both *concept drift* and *concept evolution*. In contrast to BAL and SAL which adopt decision-based AL approaches, BSAL is an information-based AL algorithm. While BAL and SAL rely on the prediction error, BSAL selects data samples which reduces the expected future error such that the loss involves the model parameter. Therefore, there is no need to heuristically modify the loss function as done in SAL to account for the new classes. More importantly, BSAL is more efficient at handling *concept evolution* in data streams involving classes of severely unbalanced proportions. The reason is that BSAL selects the data instances that lead to gaining information. SAL, on the contrary, essentially selects the data instances that contribute to the reduction of the error regardless of their classes percentage (see Chap. 3). While very rare novel classes are likely to be considered as outliers by SAL (they are not expected to have significant impact on the error), they bring significant information gain.

In this chapter, we also propose an approach to apply AL to a real-world application involving a sequential data stream, namely human activity recognition (HAR). It is highly relevant to many real-world domains like safety, security, and in particular healthcare. The current HAR technology faces many challenges among which : (1) the engineering of input features, (2) coverage of the training activity data, and (3) the annotation (the labelling) of activity data. In this chapter, we propose a new approach that consists of novel algorithms to deal with each of these problems. In particular, we apply a Conditional Restricted Boltzmann Machine (CRBM) to extract low-level features from unlabelled raw high-dimensional activity input. Because of the changes that may affect the sensor layout embedded in the living space and the change in the way activities are performed, we regard sensor data as a stream and human activity learning as an online continuous process. In such process the leaner can adapt to changes, incorporates novel activities and discards obsolete ones. Equally challenging, labelling sequential data with time dependency is highly time-consuming and difficult. Hence, the importance of AL algorithm is illustrated. It queries the user/resident about the label of particular activities in order to improve the model accuracy. The resulting approach will then tackle the problem of activity recognition using a three-module architecture composed of a feature extractor (CRBM), an online semi-supervised classifier (OSC) based on Dirichlet process mixture model (DPMM) equipped with BSAL.

The organization of this chapter is as follows. Section 4.1 presents an introduction. We discuss the related work and the motivation behind our work in Sec. 4.2. We describe OSC in Sec. 4.4. BSAL is described in Sec. 4.5. Empirical evaluation is presented in Sec. 4.6. Section. 4.7 concludes this chapter.

4.1 Introduction

The recent advances of sensor technologies have led to affordable sensors with excellent performance, low weight, and low power consumption. In smart-homes, these sensors are widely deployed to collect data for monitoring purposes. From such data, useful knowledge can be extracted allowing for a variety of applications. In many of these applications, human activity recognition (HAR) is an essential task, such as health-care [111, 112], ambient assistive living [69, 113–116] and surveillance-based security [117–119]. There are three main types of HAR, sensor-based [120], vision-based [117] and radio-based [121]. Sensor-based methods rely

on a large number of pervasive distributed sensors. Vision-based methods utilise image and video processing techniques to detect human activities. Radio-based methods use signal attenuation, propagation, and fading characteristics to detect human activities. In this work, we are interested in pervasive sensor-based methods, which, unlike the other two classes of methods, do not work under a limited coverage area, enjoy the merits of information privacy, use widely available and affordable sensors and do not expose the human body to radiation that may raise health concerns.

Recently, the field of deep learning (DL) has made a huge impact and achieved remarkable results in computer vision, natural language processing, and speech recognition. Yet it has not been fully exploited in the field of AR. DL provides an effective tool for extracting high-level feature hierarchies from high-dimensional data. Each layer of the deep architecture performs a non-linear transformation on the outputs of the previous layer. Thus, through DL, the data is represented in the form of a hierarchy of features, from low-level to high-level [122, 123]. Instead of relying on heuristic hand-crafted features, DL learns to extract features that allow for more discriminative power. Furthermore, in our experimental setting, hundreds of sensors, wearable and distributed in the environment, are deployed resulting in high-dimensional data. Hence, designing hand-crafted features is extremely hard and time-consuming. The variation of the sensor network layouts in different homes makes the task even harder if portability of the system is desired.

A key advantage of DL is that it can be trained from unlabelled data which is often massively available. In this work, we tackle the HAR problem by pre-trainning a Conditional Restricted Boltzmann Machine (CRBM) [124] to learn generic features from unlabelled raw high-dimensional sensory input. CRBM has been successfully applied in pattern recognition [124–130]. In this work, we apply CRBM to extract generic features from the sensory input. More details on CRBM are provided in Sec. 4.3.

A major contribution of this work is the application of online and active learning for HAR. To the best of our knowledge, this study is the first in the field to propose online learning to train a HAR algorithm. The HAR approaches have dominantly focused on traditional offline learning algorithms which assume that any new activities can be recognised using trained model on previously collected data. This is a strong assumption as it ignores natural changes in individuals' activity patterns and sensory measurements. In real-world situations, future data

deviates from historical data because of changes in the activities induced by the resident. Such changes take place for several reasons: the way the people perform activities changes over time, their health conditions change, they perform novel activities, etc. We adopt an online learning algorithm to cope with these changes over time; hence, we view the sensory input as continuous stream. In the vast majority of HAR approaches, training data is assumed to be manually annotated (labelled). Such manual annotation is extremely hard and time-consuming task. Furthermore, in the online setting, the data stream evolves, meaning that fresh labels are needed from time to time. Hence, using AL is a very practical solution. In the context of HAR, AL algorithm can query the user (individual carrying out the activities) about ambiguous or unknown activities in order to guide the learning process when needed.

In this work, we propose a novel and original architecture composed of an online semi-supervised classifier (OSC), stacked on the top of the deep learning (DL) feature extractor, CRBM, and equipped with an AL strategy. Similar to the model proposed in Chap. 3, OSC is based on the Dirichlet process mixture model (DPMM) [98] with a stick-breaking prior [99] over the classes. The basic difference is that OSC does not assume that the classes distribution of each instance is selected by the same variable selecting its cluster. OSC is a class-specific mixture model, where a mixture model is associated with each class. As in Chap. 3, we employ a particle filter method [29, 100] to perform online inference.

We also propose a Bayesian stream-based AL strategy called (BSAL). BSAL is an information theory based AL which aims at reducing the space of hypothesis by querying samples according to how much they are expected to reduce the model uncertainty [6]. On the contrary, decision theory based AL aims at reducing the prediction error by querying samples according to how much they are expected to reduce the future classification error [96]. While the two approaches seem quite distinct, they both aim at identifying data instances that give the largest reduction of the expected loss function Eq. (4.21). Thus, they mainly differ in the used type of loss functions. While decision theory based AL relies on the prediction error, information theory based AL losses involve the model parameter. One commonly used loss is the entropy of the model distribution.

Our proposed BSAL uses the Kullback-Leibler (KL) divergence as loss function (see Sec.4.5). We adopt an information based AL approach because it fits the Bayesian approach of the proposed semi-supervised classifier OSC (see Sec 4.4).

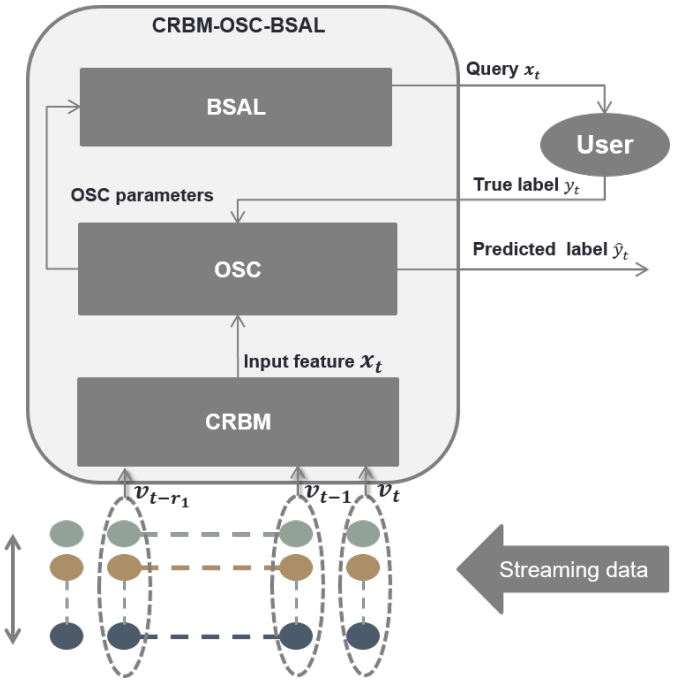


Figure. 4.1 General Architecture of CRBM-OSC-BSAL

Furthermore, there is no need to heuristically modify the loss function to account for the new classes because the loss involves the model distribution and not the prediction error. BSAL works completely online and is able to cope with the challenges associated with data streams, including the possible emergence of new classes. Since, BSAL selects the data instances that lead to gaining information, it is likely to query novel rare classes. Indeed, integrating these classes will result in significant changes in the model parameter. Hence, BSAL is capable of handling *concept evolution* even in data streams involving classes of severely unbalanced proportions.

Figure 4.1 shows a simple sketch of the proposed architecture (CRBM-OSC-BSAL).

4.2 Related Work and Motivation

Hand-crafted features have been the focus of most sensor-based HAR literature [131, 132], in which distinctive features are created or selected to train HAR systems. Statistical features such as mean, variance are utiliszed by [120, 133–137] as distinctive features of the sensory input. Such features are problem-specific and require the designer to understand the underlying problem to select and weight the most

effective features using the expensive trial and error process. Any variation of the environment implies re-crafting the features, which is inadequate.

On the contrary, DL can be used to learn discriminative features from the data automatically and in a systematic way. DL learns different layers of features from low-level generic features to high-level features. DL has made a tremendous impact on different fields such as computer vision and natural language processing [138]. Recently, few studies have considered deep learning in sensor-based AR [139–143]. However, all of these studies work offline.

The great majority of HAR research is based on offline learning algorithms, in which the adaptation of the learner after the training is not possible. Recently, authors in [144] used online learning for HAR where the data is considered as a stream and where AL was applied to query activity labels when necessary. However, the proposed approach requires labelled training set. In fact, there is two phases: (1) offline training phase and (2) online recognition and adaptation phase. In the offline phase, the model is built from a set of annotated sensory data that represents different activities. In the online phase, the recognition of unlabelled streaming data is performed. In this approach, the number of activities is assumed to be fixed. The proposed model does not extract the features and instead directly uses clustering technique.

Authors in [145] proposed to address some of HAR challenges such data annotation through active learning. However, instead of using online learning to adapt the model when necessary, transfer learning is used. Models are trained on different collected houses and persons where transfer learning is employed to share the knowledge among these models. Authors also apply offline AL to obtain labels. However, any change is assumed to be represented in the training data, but on a large scale where different houses and different persons are covered. Hence, any change not occurring in the training data (e.g., emergence of novel activities) cannot be handled.

A similar architecture to ours is proposed in [146], where a hierarchical non-parametric Bayesian model is plugged on top of a deep network. The deep network learns low-level generic features, then the hierarchical non-parametric Bayesian model learns high-level features that capture correlations among low-level features. However, the model works offline, does not use AL and does not process time-series data.

The present work goes beyond the sate-of-the-art methods by addressing the challenges of HAR in the smart-home setting with each component of the CRBM-OSC-BSAL architecture coping with one specific challenge.

1. CRBM allows to capture features that are less sensitive to the subtleties of sensory input. It learns generic features from the data in unsupervised way.
2. OSC helps overcome dynamic changes within the same environment. It allows CRBM-OSC-BSAL to be self-adaptive. OSC works online and adapts to change.
3. BSAL helps overcome hard and time-consuming activities annotation. It allows CRBM-OSC-BSAL to be self-exploring. BSAL is the first AL that directly reduces the expected loss online, while considering the challenges of data streams.

Note that this proposed architecture is generic, not dedicated only to HAR problems but can be applied to different applications with sequential time dependency. We review CRBM and Dirichlet process (DP) in Sec. 4.3 and Sec. 3.3.1 respectively. CRBM will be used to extract generic low level features which are used by our proposed online semi-supervised classifier (OSC) to learn activities. Being the core of OSC, DP is used as a non-parametric prior in Dirichlet process mixture model (DPMM) which, in contrast to the parametric prior, allows the number of components to vary during learning.

4.3 Conditional Restricted Boltzmann Machine

CRBM [124] is a non-linear generative model for time-series that uses an undirect model with binary latent variables, $\mathbf{h}$, connected to visible variables, $\mathbf{v}$. Unlike Hidden Markov models (HMMs) which rely on a single discrete K-state multinomial, CRBM allows for distributed binary representations for its hidden states. For, example, to model N bits of information about the past history, HMMs require 2^N hidden states, while CRBM only needs N binary latent variables. Linear dynamical systems are models with distributed hidden state, but, they cannot model the complex non-linear dynamics in the high-dimensional sensory input.

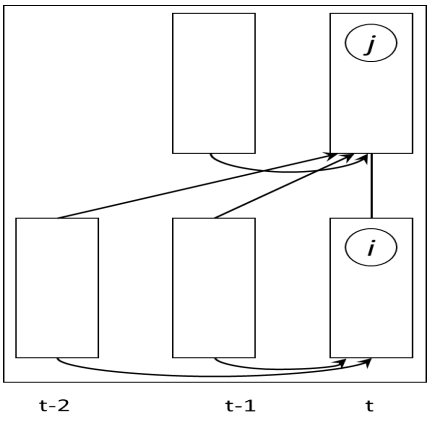


Figure. 4.2 Feature Extractor Architecture ($r_1 = 2$, $r_2 = 1$)

CRBM is a temporal extension of restricted Boltzmann machines (RBM). Typically, RBM uses binary units for both visible and hidden variables. But the sensory input in our data is continuous; therefore, we use real-valued Gaussian input units. CRBM has a layer of visible units which resembles to autoregressive model and a layer of hidden units. The visible variables $\mathbf{v}$ and hidden variables $\mathbf{h}$ in the current time slice receive directed connections from the visible variables at the previous few time slices. Also, there are undirected connections between layers at the current time slice like in RBM. Figure 4.2 shows a CRBM example with two layers, where the temporal order of the one at the bottom (r_1) is 2 and for the one in the top (r_2) is 1. CRBM defines a joint probability distribution over $\mathbf{v}$ and $\mathbf{h}$, conditional on the past n observations and model parameters Φ:

$$
\begin{aligned}
&p(\boldsymbol{v}, \boldsymbol{h} | \{\boldsymbol{v}\}_{t-n}^{t-1}, \Phi) \propto \exp(-E(\boldsymbol{v}, \boldsymbol{h} | \{\boldsymbol{v}\}_{t-n}^{t-1}, \Phi)) \\
&E(\boldsymbol{v}, \boldsymbol{h} | \{\boldsymbol{v}\}_{t-n}^{t-1}, \Phi) = \sum_i \frac{(v_i - b_i)^2}{2\sigma_i^2} - \sum_j h_j b_j - \sum_{ij} \phi_{ij} \frac{v_i}{\sigma_i} h_j
\end{aligned} \tag{4.1}
$$

where σ_i is the standard deviation of the Gaussian noise for visible unit i. Like in [124], it is set to one after rescaling the data to have zero mean and unit variance. The dynamic biases, b_i, b_j, are affine functions of the past n observations. The parameter ϕ_{ij} is a weight between elements v_i and h_j. The undirected connections between the hidden and visible variables in the current layer (time t) makes the inference easy because the hidden units become conditionally independent when the visible units are observed. The training is, therefore, easily done by minimizing contrastive divergence (for more details see [124]).

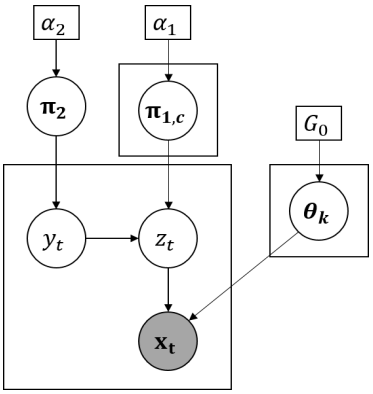


Figure. 4.3 Graphical model of OSC

A crucial characteristic of CRBM is that we can add layers like in Deep Belief Networks [147]. All layers in the CRBM architecture are trained similarly but sequentially. As more layers are added, CRBM can model higher-level features. In this paper, we use only two layers to retain the low-level inter-features correlations at a lower training computational cost.

4.4 Online Semi-supervised Classifier

The proposed online semi-supervised classifier (OSC) can be expressed as a Dirichlet process mixture model (DPMM) with a new latent label variable y_t (observed after querying the sample). Figure 4.3 shows the structure of OSC in the form of a graphical model. $\boldsymbol{\pi_{1,c}}$ and $\boldsymbol{\pi_2}$ are drawn from stick-breaking processes $GEM(\alpha_1)$ and $GEM(\alpha_2)$ respectively; G_0 is a Normal-Inverse-Wishart distribution $NIW(.|\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0, k_0, v_0)$ with $\boldsymbol{\mu}_0$ is the prior of the clusters' means; $\boldsymbol{\Sigma}_0$ controls the variance among the means; k_0 scales the diffusion of the clusters' means and v_0 is the degree of freedom of the Inverse-Wishart distribution.

The label y_t is generated from a stick-breaking prior. While z_t selects the component generating x_t, y_t selects the stick-breaking component generating z_t. Label y_t selects from different mixture models associated with different classes. The model is updated as follows. If the data samples received are unlabelled, the whole model is updated such that the class variable is marginalized out. Otherwise, only the mixture model associated with the same class as the sample is updated with the new data sample. A particle filter method derived from [29, 100] is used to perform the online inference.

Following [29], 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 replace all the statistics used in OSC, where m_t is the number of components; $\boldsymbol{n}_t$ is a matrix with rows referring to the number of data samples labled to the existing classes and columns referring to the number of data samples assigned to the existing components; and $\boldsymbol{s}_t$ is the sufficient statistics for all mixture components(i.e., means and scatter matrices).

Every time a new sample is received, OSC carries out three steps: prediction, updating and re-sampling. While updating step differs according to whether the data sample is labelled or not, prediction and re-sampling steps are applied in the same way for both cases. The 3 steps of OSC are described as follows:

4.4.1 Prediction:

Given the concentration parameters α_1, α_2 and the prior distribution parameters $\{\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0, k_0, v_0\}$, we aim at computing the conditional probability of the label given a data sample, $p(y_t|\boldsymbol{x}_t, D_{t-1})$:

$$p(y_t|\boldsymbol{x}_t, D_{t-1}) \propto p(\boldsymbol{x}_t|y_t, D_{t-1})p(y_t|D_{t-1}) \tag{4.2}$$

where D_{t-1} represents all data samples previously seen along with their labels if provided.

$$p(y_t|D_{t-1}) \propto \begin{cases} n_{y_t,:,t} & y_t \text{ is an existing class} \\ \alpha_2 & y_t \text{ is a new class} \end{cases} \tag{4.3}$$

where $n_{y_t,:,t}$ is the number of data samples labelled y_t at time t regardless of the components. The ':' denotes all the components.

$$p(\boldsymbol{x}_t|y_t, D_{t-1}) = \sum_{z_{1:t-1}} p(\boldsymbol{x}_t|y_t, z_{1:t-1}, D_{t-1})p(z_{1:t-1}|y_t, D_{t-1}) \tag{4.4}$$

$$p(\boldsymbol{x}_t|y_t, z_{1:t-1}, D_{t-1}) = \sum_{z_t} p(\boldsymbol{x}_t|y_t, z_t, z_{1:t-1}, D_{t-1})p(z_t|y_t, z_{1:t-1}, D_{t-1}) \tag{4.5}$$

$$p(z_{1:t-1}|y_t, D_{t-1}) \propto p(y_t|D_{t-1})p(z_{1:t-1}|D_{t-1}) \tag{4.6}$$

We use $z_{1:t}$ to denote the sequence $\{z_1, z_2, ..., z_t\}$. The first term of Eq. (4.5) can be computed as follows:

$$p(\boldsymbol{x}_t|y_t, z_t, z_{1:t-1}, D_{t-1}) = \int_{\boldsymbol{\theta}} p(\boldsymbol{x}_t|\boldsymbol{\theta}, z_t)p(\boldsymbol{\theta}|z_t, z_{1:t-1}, D_{t-1}) \tag{4.7}$$

If z_t refers to a new component, $p(\boldsymbol{\theta}|z_t, z_{1:t-1}, D_{t-1})$ becomes equivalent to the prior distribution $p(\boldsymbol{\theta}|G_0)$. Otherwise, z_t refers to an already existing component. Then, $p(\boldsymbol{\theta}|z_t, z_{1:t-1}, D_{t-1})$ becomes equivalent to $p(\boldsymbol{\theta}|\boldsymbol{s}_{z_t,t-1}, n_{z_t,t-1}, z_{1:t-1})$, where $\boldsymbol{s}_{z_t,t-1} = \{\boldsymbol{su}_{z_t,t-1}, \boldsymbol{sc}_{z_t,t-1}\}$ is the sufficient statistics (i.e., mean and scatter matrix respectively) defined as.

$$\begin{aligned} \boldsymbol{su}_{z_t,t-1}(z_{1:t-1}) &= \frac{\sum_{z_i=z_t, i<t} \boldsymbol{x}_i}{n_{z_t,t-1}} \\ \boldsymbol{sc}_{z_t,t-1}(z_{1:t-1}) &= \sum_{z_i=z_t, i<t} (\boldsymbol{x}_i - \boldsymbol{su}_{z_t,t-1})(\boldsymbol{x}_i - \boldsymbol{su}_{z_t,t-1})^T \end{aligned} \tag{4.8}$$

where $n_{z_t,t-1}$ is the number of data samples which have been assigned to component z_t until time $t-1$. Equation (4.7) can be solved given the sufficient statistics, the past assignments and the model hyper-parameters. It can be computed the same way as in Eq. (3.22) with X_t replaced by D_t. The second term of Eq. (4.5) can be written as follows:

$$p(z_t|y_t, z_{1:t-1}, D_{t-1}) = p(z_t|\{z_i\}_{y_i=y_t, i<t}) \tag{4.9}$$

Similar to Eq. (4.7), the solution is as follows:

$$p(z_t|\{z_i\}_{y_i=y_t, i<t}) \propto \begin{cases} n_{y_t,z_t,t} & z_t \text{ is an existing component} \\ \alpha_1 & z_t \text{ is a new component} \end{cases} \tag{4.10}$$

where $n_{y_t,z_t,t}$ is the number of data samples labeled y_t and assigned to component z_t at time t.

The second term in Eq. (4.6), $p(z_{1:t-1}|D_{t-1})$, is the probability of the different configurations. Such configurations determine the different statistics represented by H_t. Thus, $p(H_{t-1}|D_{t-1})$ has the same probability as the posterior $p(z_{1:t-1}|D_{t-1})$. $p(H_{t-1}|D_{t-1})$ is outlined and developed in Eq. (4.11).

The solution of Eq. (4.2) depends only on the elements of the state vector along with its posterior distribution. Thus, we need to track this posterior online by approximating it with a set of P 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, $p(H_t|D_t)$, given that the posterior at $t-1$, $p(H_{t-1}|D_{t-1})$ is known. In order to prevent combinatorial explosion, we use the

re-sampling technique proposed in [100] which retains only P particles. Therefore, we approximate the posterior at time t using the following updating and re-sampling steps:

4.4.2 Updating:

$$p(H_t|D_t) \propto \int_{H_{t-1}} p(H_t|H_{t-1}, y_t, \boldsymbol{x}_t)p(y_t, \boldsymbol{x}_t|H_{t-1})p(H_{t-1}|D_{t-1}) \tag{4.11}$$

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

$$p(H_t|D_t) \propto \sum_{i=1}^{P} p(H_t|H_{t-1}^{(i)}, y_t, \boldsymbol{x}_t)p(y_t, \boldsymbol{x}_t|H_{t-1}^{(i)})w_{t-1}^{(i)} \tag{4.12}$$

The solution of the second term of Eq. (4.12) can be computed in a similar way to Eq. (4.4) and Eq. (4.3). Following the updating step, the number of resulting particles for each $H_{t-1}^{(i)}$ becomes equal to the number of existing components $m_{t-1}^{(i)}+1$. The new assignments z_t lead to different configurations of the new particles. Therefore,

$$\begin{aligned} p(H_t^{(j)}|H_{t-1}^{(i)}, y_t, \boldsymbol{x}_t) &= p(z_t = j_1|H_{t-1}^{(i)}, y_t, \boldsymbol{x}_t) \\ &\propto p(\boldsymbol{x}_t|y_t, z_t = j_1, H_{t-1}^{(i)})p(z_t = j_1|y_t, H_{t-1}^{(i)}) \end{aligned} \tag{4.13}$$

where $j = f(i, j_1)$ and $f(a, b) = \frac{1}{2}(a + b)(a + b + 1) + b$ is the Cantor pairing function which uniquely encodes the assignment j_1 and the particle number i into a single natural number. By solving Eq. (4.13), we determine the weight of the new particle $w_t^{(j)}$. The first term of Eq. (4.13) is computed in Eq. (4.7), and the second term is computed in Eq. (4.9)). The elements of a new state vector $H_t^{(j)}$

are updated as follows:

$$H_t^{(j)} = \begin{cases} z_t^{(j)} & = j_1 \qquad\qquad j_1 \text{ is an existing component} \\ n_{y_t,j_1,t}^{(j)} & = \lambda n_{y_t,j_1,t-1}^{(i)} + 1 \\ n_{h,k,t}^{(j)} & = \lambda n_{h,k,t-1}^{(i)} \quad \forall h \neq y_t, \forall k \neq j_1, h \in C_t, k \leq m_t^{(i)} \\ \boldsymbol{su}_{j_1,t}^{(j)} & = \frac{\lambda n_{:,j_1,t-1}^{(i)} \boldsymbol{su}_{j_1,t-1}^{(i)} + \boldsymbol{x}_t}{n_{:,j_1,t}^{(i)}} \\ \boldsymbol{sc}_{j_1,t}^{(j)} & = \lambda \boldsymbol{sc}_{j_1,t-1}^{(i)} + n_{:,j_1,t-1}^{(i)} \boldsymbol{su}_{j_1,t-1}^{(i)} \boldsymbol{su}_{j_1,t-1}^{(i)T} \\ & \quad - n_{:,j_1,t}^{(i)} \boldsymbol{su}_{j_1,t}^{(i)} \boldsymbol{su}_{j_1,t}^{(i)T} + \boldsymbol{x}_t \boldsymbol{x}_t^T \\ & \\ z_t^{(j)} & = m_{t-1}^{(i)} + 1 \qquad j_1 \text{ is a new component} \\ m_t^{(j)} & = m_{t-1}^{(i)} + 1 \\ n_{y_t,j_1,t}^{(j)} & = 1 \\ n_{h,k,t}^{(j)} & = \lambda n_{h,k,t-1}^{(i)} \quad \forall h \neq y_t, \forall k \leq m_{t-1}^{(i)}, h \in C_t \\ \boldsymbol{su}_{j_1,t}^{(j)} & = \boldsymbol{x}_t \\ \boldsymbol{sc}_{j_1,t}^{(j)} & = \mathbf{0} \end{cases} \tag{4.14}$$

where λ is a forgetting factor which allows the components to adapt with change, C_t is the set of the labels of all existing classes. If the label y_t is unknown, we consider it as a latent variable. The posterior $p(H_t|D_t)$ in the update Eq. (4.12) can be written as follows:

$$p(H_t|D_t) = \sum_{y_t} p(H_t|D_{t-1}, \boldsymbol{x}_t, y_t) p(y_t|D_{t-1}, \boldsymbol{x}_t) \tag{4.15}$$

$$p(y_t|D_{t-1}, \boldsymbol{x}_t) \propto p(\boldsymbol{x}_t|y_t, D_{t-1}) p(y_t|D_{t-1}) \tag{4.16}$$

The first and second terms of Eq. (4.16) are computed in Eq. (4.4) and Eq. (4.3) respectively. The first term of Eq. (4.15) is already computed in Eq. (4.12), where the label is assumed to be known. We can see from Eq. (4.15) that for each new assignment z_t, there is a mixture of particles that depends on j and the different labels y_t. We re-define the state vector H_t to accommodate y_t as hidden variable when it is unknown. Thus, $H'_t = \{H_t, y_t\}$, where the different particles

are determined now by both z_t and y_t,

$$\begin{aligned} H_t^{\prime(j)} &= \{H_t^{(j_2)}, y_t = j_3\} \\ j &= f(j_2, j_3) \end{aligned} \tag{4.17}$$

where j_3 can be either an existing or a new class. To compute the weight and the update of the new state vector, we follow the same trend as in Eq. (4.12), Eq. (4.13) and Eq. (4.14).

$$p(H_t'|D_t) \propto \sum_{i=1}^{P} p(H_t'|H_{t-1}^{\prime(i)}, \boldsymbol{x}_t) p(\boldsymbol{x}_t|H_{t-1}^{\prime(i)}) w_{t-1}^{\prime(i)} \tag{4.18}$$

$$\begin{aligned} p(H_t^{\prime(j)}|H_{t-1}^{\prime(i)}, \boldsymbol{x}_t) &= p(z_t = j_1, y_t = j_3|H_{t-1}^{\prime(i)}, \boldsymbol{x}_t) \\ &\propto p(\boldsymbol{x}_t|y_t = j_3, z_t = j_1, H_{t-1}^{\prime(i)}) p(z_t = j_1, y_t = j_3|H_{t-1}^{\prime(i)}) \end{aligned} \tag{4.19}$$

After updating the P particles with all the possible new assignments z_t and y_t, we end up with M combinations of the P particles with the new assignments, along with the weights $w_t^{\prime(j)}$. So, we move to the next step which reduces the number of created particles to a fixed number P.

4.4.3 Re-sampling:

We follow the resampling technique proposed in [100] which discourages the less likely particles (configurations), and improves the particles explaining the data better. It keeps the particles whose weight is greater than $1/\kappa$, and re-samples from the remaining particles. The variable κ is the solution of the following equation:

$$\sum_{j=1}^{M} \min\{\kappa w_t^{\prime(j)}, 1\} = P \tag{4.20}$$

The weight of re-sampled particles is set to $1/\kappa$, and the weight of the particles greater than $1/\kappa$ is kept unchanged. All the re-sampled particles are ensured to be re-sampled once.

4.5 Active Learning Approach

We propose an AL algorithm that deliberately queries particular instances to train the OSC using as few labelled data instances as possible. In the context of HAR, the labels are the human activities, and the AL algorithm queries the user (individual carrying out the activities) about some activities. Thus, extensive queries will be annoying and must be avoided.

Most AL approaches are basically derived from the general approach of finding queries that result of the largest reduction in the expected loss. Let Ω denote the set of variables that can be fully random or include some random elements. Let L refer to the loss function. The expected loss function that AL aims to reduce can be expressed as follows:

$$R = E_{\Omega}[L(\hat{\Omega}, \Omega)] \tag{4.21}$$

where the hat refers to an estimated term. Depending on the loss functions involved, AL can be divided to two main groups: information-based and decision-based AL. If we take Ω as the set of vectors consisting of observed set X and latent set Y elements, we can end up with the expected classification error:

$$R = \int_{\boldsymbol{x}} L(\hat{p}(y|\boldsymbol{x}), p(y|\boldsymbol{x}))p(\boldsymbol{x})d\boldsymbol{x} \tag{4.22}$$

Here, the loss function involves the prediction error. In AL, we request information about certain samples. That is the learner queries the latent labels Y of some subset of X/Ω. We introduce a binary set Q whose elements are attached to the vectors in the set Ω. If an element $q \in Q = 1$, the latent elements of the corresponding vector in Ω is queried. Equation (4.22) can be seen as the core of most heuristic and non-heuristic decision-based AL. Many active learning approaches seek to minimize an approximation of the expected error of the learner Eq. (4.22) [76, 96, 97]. In our previous work [148, 149], we proposed an AL strategy that seeks to minimize Eq. (4.22) online, while considering the challenges of data streams.

If Ω is taken to be the set of the true model parameters $\boldsymbol{\psi}$, then the loss is over the model parameters. We obtain the following risk function:

$$R = E_{\boldsymbol{\psi}}[L(\hat{\boldsymbol{\psi}}, \boldsymbol{\psi})] \tag{4.23}$$

Similar to Eq. (4.22), Eq. (4.23) can be seen as the core of most heuristic and non-heuristic information-based AL. Authors in [150, 151] use the entropy of the model as the loss function. In this work, we propose a Bayesian stream-based AL (BSAL) inspired from information-based AL. BSAL is designed to cope with the challenges of data streams (infinite length, evolving nature, emergence of new classes). Information-based AL fits better the nature of the proposed Bayesian semi-supervised classifier (OSC), where the uncertainty over the model parameters is systematically expressed. Because information-based loss functions are over the model distribution, BSAL can easily deal with the challenges of data streams. In fact, the task is only to take the decision whether to online query or not, while OSC works online and accommodates novel classes. Thus, BSAL is completely compatible with OSC.

In the following, we discuss the offline AL strategy to minimize an approximation of Eq. (4.23), then we present our online AL strategy. The authors in [6] propose an algorithm that selects queries in a greedy way in order to improve the model accuracy as much as possible. Their original goal is to minimize the loss $L(\boldsymbol{\psi}, \hat{\boldsymbol{\psi}})$ incurred by using a single representation $\hat{\boldsymbol{\psi}}$ of the model instead of the true model parameters $\boldsymbol{\psi}$. As the true model parameters are unknown, authors estimate them using the posterior of the Bayesian model parameters $p(\boldsymbol{\psi}|X, Y)$. Therefore, the risk associated with a particular $\hat{\boldsymbol{\psi}}$ with respect to $p(\boldsymbol{\psi}|X, Y)$ can be expressed as follows:

$$R(p(\boldsymbol{\psi}|X, Y), \hat{\boldsymbol{\psi}}) = E_{\boldsymbol{\psi}\sim p(\boldsymbol{\psi}|X,Y)}[L(\boldsymbol{\psi}, \hat{\boldsymbol{\psi}})] \tag{4.24}$$

The point estimate $\hat{\boldsymbol{\psi}}$ that minimizes the risk is defined as the Bayesian point estimate. By fixing $\hat{\boldsymbol{\psi}}$ to the Bayesian point estimate, the resulting risk depends only on the model posterior. Authors in [6] adopted a pool-based AL approach, where the samples that reduce the risk the most are selected. Because the labels are unknown a priori, the expected risk, given the query, is computed as follows:

$$\hat{R}\big(p(\boldsymbol{\psi}|X, Y), \hat{\boldsymbol{\psi}}; Q = \boldsymbol{q}\big) = E_{Y_{S(Q)}\sim p(Y_{S(Q)}|X)}[R(p(\boldsymbol{\psi}|X, Y_{S(Q)}), \hat{\boldsymbol{\psi}})] \tag{4.25}$$

where clamping $\boldsymbol{q}$ to Q refers to the state of querying samples that correspond to the elements in $\boldsymbol{q}$ which are equal to one. The function $S(Q)$ returns the indices of the elements in Q that are equal to 1 (i.e., the samples to be queried). If $S(Q)$ is empty, the expected risk in Eq. (4.25) becomes:

$$\hat{R}\big(p(\boldsymbol{\psi}|X, Y), \hat{\boldsymbol{\psi}}; Q = \boldsymbol{q}\big) = R(p(\boldsymbol{\psi}|X), \hat{\boldsymbol{\psi}}) \tag{4.26}$$

The goal of AL is to query the data samples that result in maximizing the difference between the risk (Eq. (4.24)) and the expected risk (Eq. (4.25)).

$$\hat{\Delta}(X, Y|\boldsymbol{q}) = R(p(\boldsymbol{\psi}|X), \hat{\boldsymbol{\psi}}) - \hat{R}(p(\boldsymbol{\psi}|X, Y), \hat{\boldsymbol{\psi}}; Q = \boldsymbol{q}) \tag{4.27}$$

Given a stream of samples, BSAL evaluates each sample at time t before discarding it. Equation (4.27) can be reformulated as follows:

$$\hat{\Delta}(\boldsymbol{x}_t, y_t|D_{t-1}, q_t) = R(p(\boldsymbol{\psi}_t|D_{t-1}, \boldsymbol{x}_t), \hat{\boldsymbol{\psi}}_t) - \hat{R}(p(\boldsymbol{\psi}_t|D_{t-1}, \boldsymbol{x}_t, y_t), \hat{\boldsymbol{\psi}}_t; q_t) \tag{4.28}$$

where:

$$R(p(\boldsymbol{\psi}_t|D_{t-1}, \boldsymbol{x}_t), \hat{\boldsymbol{\psi}}_t) = E_{\boldsymbol{\psi}_t \sim p(\boldsymbol{\psi}_t|D_{t-1}, \boldsymbol{x}_t)}[L(\boldsymbol{\psi}_t, \hat{\boldsymbol{\psi}}_t)] \tag{4.29}$$

If the sample at time t is not queried ($q_t = 0$), the current expected risk (second term of Eq. (4.28)) is equal to the current risk (first term of Eq. (4.28)). Otherwise, the current expected risk can be written as follows:

$$\hat{R}(p(\boldsymbol{\psi}_t|D_{t-1}, \boldsymbol{x}_t, y_t), \hat{\boldsymbol{\psi}}_t; q_t = 1) = E_{y_t \sim p(y_t|D_{t-1}, \boldsymbol{x}_t)}[R(p(\boldsymbol{\psi}_t|D_{t-1}, \boldsymbol{x}_t, y_t), \hat{\boldsymbol{\psi}}_t)] \tag{4.30}$$

BSAL uses the KL divergence as the loss function. It turns out that by using the KL divergence loss, the mean value of the parameters turns into the Bayesian point estimate [6]. Similar to the Bayesian information theoretic AL proposed in [152], we consider the loss of the predictive posteriors parameterized by $\boldsymbol{\psi}_t$ and $\hat{\boldsymbol{\psi}}_t$:

$$L(\boldsymbol{\psi}_t, \hat{\boldsymbol{\psi}}_t) = \sum_y \int_{\boldsymbol{x}} p(\boldsymbol{x}, y|D_t, \boldsymbol{\psi}_t) \log \frac{p(\boldsymbol{x}, y|D_t, \boldsymbol{\psi}_t)}{p(\boldsymbol{x}, y|D_t, \hat{\boldsymbol{\psi}}_t)} d\boldsymbol{x} \tag{4.31}$$

$$\hat{\boldsymbol{\psi}}_t = E_{\boldsymbol{\psi}_t \sim p(\boldsymbol{\psi}_t|D_t)}[\boldsymbol{\psi}_t] \tag{4.32}$$

Our BSAL relies on our proposed Bayesian online semi-supervised classifier (OSC). OSC's parameters, $\boldsymbol{\psi}$, involve the stick-breaking components, the Gaussian components and the hidden configurations. We marginalize out the Gaussian and the stick-breaking components and keep the hidden configurations. The Bayesian point estimate is approximated by the mode of the different configurations induced by the particles. The details on how the discrepancy between the current risk and the current expected risk expressed in Eq. (4.28) is computed can be found in App. C.1.

As BSAL is an online-based AL, it must assess the data on the-fly and query those which incur highest risk reduction. The problem is how to decide whether the incurred reduction is high or not. A dynamically adaptive threshold, τ, is used so as to request the labels of samples whose current expected risk subtracted from their current risk (see Eq. (4.28)) breaches the threshold. This latter is adapted using a threshold adjustment step, s, following the Variable Uncertainty strategy in [1]. Further illustration may be found in Alg. 3. The binary input ALE in the algorithm activates/deactivates the active learning.

Another issue is the limited labelling resources. Hence, a rationale querying strategy to optimally use those resources needs to be applied. To this end, the notion of budget used in Chap. 2 is adopted in BSAL so that the labelling rate is controlled. Details about this notion can be found in Sec. 2.3.2.

Instead of fixing the precision hyper-parameters (also called concentration hyper-parameters) α_1 and α_2, we put hyper priors over them using $G(a, b)$ (gamma priors with shape a and scale b) and sample their values online following the sampling approach in [153]. More details on the sampling routine can be found in App. C.2.

Algorithm 3 Steps of CRBM-OSC-BSAL

Input: data stream, OSC hyper-parameters $\{\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0, k_0, v_0, a, b\}$ (see Sec. 4.6.2), forgetting factor λ, maximum number of particles P, budget Bd, ALE
initialize: set OSC precision parameters $\{\alpha_2, \alpha_{1i}\}_{i=1}^{m_2}$ to 1 (more details in Sec. 4.6.2 and App. C.2), set the OSC first particle weight $w_0^{(1)}$ to 1, threshold τ to 0.1, threshold adjustment step s to 0.1, $wnd = 100$, $\hat{f}_t = 0$ (Eq. (2.44)) and $t = 0$
while (true) **do**
  $t \leftarrow t + 1$
  extract low level features $\boldsymbol{x_t}$ from the current data samples {(see Sec. 4.3)}
  if $ALE = 1$ {active learning is activated} **then**
    if $\hat{b}_t < Bd$ {enough budget, Eq. (2.44)} **then**
      $a = \hat{\Delta}(\boldsymbol{x}_t, y_t | D_{t-1}, q_t = 1)$ {(see Eq. (4.28))}
      if $a > \tau$ **then**
        $Lab_t = 1$
        $y_t \leftarrow query(\boldsymbol{x_t})$
        $\tau = \tau(1 + s)$
      else
        $Lab_t = 0$
        $\tau = \tau(1 - s)$
      end if
    else
      Lab_t=0
    end if
  else
    Receive (Lab_t)
    if $Lab_t = 1$ {instance $\boldsymbol{x_t}$ label is known} **then**
      $y_t \leftarrow reveal(\boldsymbol{x_t})$
    end if
  end if
  if $Lab_t = 0$ **then**
    predict the label of $\boldsymbol{x_t}$ {(see Eq. (4.2))}
    update OSC model with instance $\boldsymbol{x_t}$ {(see Eq. (4.14), Eq. (4.17) and Eq. (4.18))}
  else
    update OSC model with instance $\boldsymbol{x_t}$ and label y_t {(see Eq. (4.12) and Eq. (4.14))}
  end if
  sample new precision parameters $\{\alpha_2, \alpha_{1i}\}_{i=1}^{m_2}$ {(see Alg. 4 in App. C.2)}
  compute $\hat{f}_{t+1}$ {(see Eq. (2.45))}
end while

4.6 Experiments

In this section, we evaluate CRBM-OSC-BSAL in two steps. In the first step, the active learning (AL) strategy is deactivated ($ALE = 0$) while the classification performance of CRBM-OSC is evaluated on the opportunity human activity recognition (HAR) dataset (see below). The data comes as a stream through CRBM which reduces its dimensions, then OSC classifies the currently performed activity and updates its particles set. To show the efficiency of the classification, we compare against static offline classification methods: Hoeffding decision tree (DT) and support vector machine (SVM). We also compare against the online method STAR [144] already discussed in Sec. II. In the second step, the active learning strategy, proposed in Sec. 4.5, is activated ($ALE = 1$) to evaluate the whole framework CRBM-OSC-BSAL on the same dataset. The classification performance is measured according to the average accuracy (AA) which is the correctly classified data samples divided by the total testing data samples. We also compute the average class accuracy (ACA) which is the average of the average accuracies across different activities (classes). This measurement manifests BSAL performance consistency across all activities by implicitly penalizing the accuracy when there are misclassifications of infrequent activities. Hence, it illustrates the class discovery performance of BSAL.

The Opportunity (Opp) dataset was the basis of the activity recognition challenge (http://www.opportunity-project.eu/challenge) proposed in the context of the European research project OPPORTUNITY [154]. Opp is a high-dimensional HAR dataset labelled for modes of locomotion, gestures and high-level activities. The data is acquired from four human subjects. In this work, we use a subset of the dataset corresponding to 3 subjects, denoted by Si and focus on recognition of gesture and modes of locomotion in order to show CRBM-OSC-BSAL high performance. Details of Opp are presented in Tab.4.1, Tab.4.3 and Tab.4.2, where N is the number of instances, d is the number of features/attributes, N_c is the number of classes. Opp was collected from subjects while performing daily activities in a sensor-rich environment of a room akin to an apartment with kitchen, deckchair, and outdoor access. 112 wearable and pervasive sensors with different sensing modalities were used. Each subject executed 4 motion activities and 17 gestures (e.g., *sitting*, *walking*, *standing*, *open fridge*, *clean table*, *move cup*, etc.). The task is to recognise the currently performed activity. Further details can be found in [154].

Table. 4.1 Real AR Dataset propreties used for evaluating CRBM-OSC-BSAL

Opp data	N	d	N_c
S2	133023	113	4 Locomotions and 17 Gestures
S3	124320	113	4 Locomotions and 17 Gestures
S4	105082	113	4 Locomotions and 17 Gestures

Table. 4.2 Class proportions for gesture activities

Subject	Open Door1	Open Door2	Close Door1	Close Door2	Open Fridge	Close Fridge	Open Dishwasher	Close Dishwasher	Open Drawer1
S2 (%)	5.1030	2.2899	1.3897	2.4755	2.3124	1.9242	7.0552	4.7204	6.3576
S3 (%)	3.9660	3.8557	3.0104	4.6303	5.1269	3.2421	6.5879	5.0188	3.3856
S4 (%)	3.9444	3.7998	3.3823	4.4454	4.0215	2.5503	5.8266	4.9208	3.7099

Close Drawer1	Open Drawer2	Close Drawer2	Open Drawer3	Close Drawer3	Clean Table	Drink Cup	Toggle Switch
2.6724	1.3278	2.9481	3.4151	2.0029	10.0259	40.2892	3.6908
3.9770	3.5511	5.5507	6.8153	3.3017	7.1000	27.1927	3.6879
4.0118	3.1446	7.1050	4.8566	3.9283	7.5450	29.3643	3.4433

Table. 4.3 Class proportions for locomotion activities

Subject	Stand	Walk	Lie	Sit
S2 (%)	36.13	24.43	3.71	35.74
S3 (%)	58.83	27.24	2.41	11.48
S4 (%)	52	25.16	5.53	17.32

Table. 4.4 Real AR Dataset propreties used for evaluating CRBM-OSC-BSAL

CRBM layers	input dimension	output dimension	temporal order
layer 1	113	150	8
layer 2	150	200	0
layer 3	200	10	0

4.6.1 Feature extractor

Before running experiments, we set up CRBM by training it on the Opp data. To ensure the independence between the online learning and feature extraction, the CRBM model for each subject is built from unlabelled data of all other subjects (all subjects except the one on which the current online learning is evaluated). We study the impact of different CRBM's parameter settings. The number of layers is fixed to 3 with the dimension of the last layer is set to 10 ($dim3 = 10$). The effect of the CRBM parameters effect (number of layers, dimensions and temporal orders) is studied and the parameters that give the best performance are chosen. The first and second layers' dimensions are set to $dim1 = 150$ and $dim2 = 200$ respectively, the temporal orders are set to $r1 = 8$, $r2 = 0$ and $r3 = 0$ (see

Tab. 4.4). In order to demonstrate what CRBM has learned about the structure of the data, we feed the trained CRBM with a data segment consisting of 22000 samples from subject 3 (S3) and plot the results.

Figure 4.4 is a gray-scale image showing the probability of the binary features extracted by the first layer. In Figure 4.5, the features extracted by the second layer is shown with a ribbon of different colours which illustrates some class labels of the activities through time. We can notice a regular output pattern for each activity. It can also be seen that the regularity of the features obtained by the second layer (Fig. 4.5) is sharper and less noisy than those obtained by the first layer (Fig. 4.4). Hence, features become more discriminative. To visualize the data samples in the features space, we set the last layer's dimensions to 2 and plot the output in Fig. 4.6. The data samples representing *standing* activity often overlap with the other activities especially *walking*. A potential explanation is that most transitions are from *standing* to *walking*. Besides, the gesture activities (not plotted) are usually performed while the user is standing. Thus, the *standing* activity region in the feature space gets expanded towards other activities' regions.

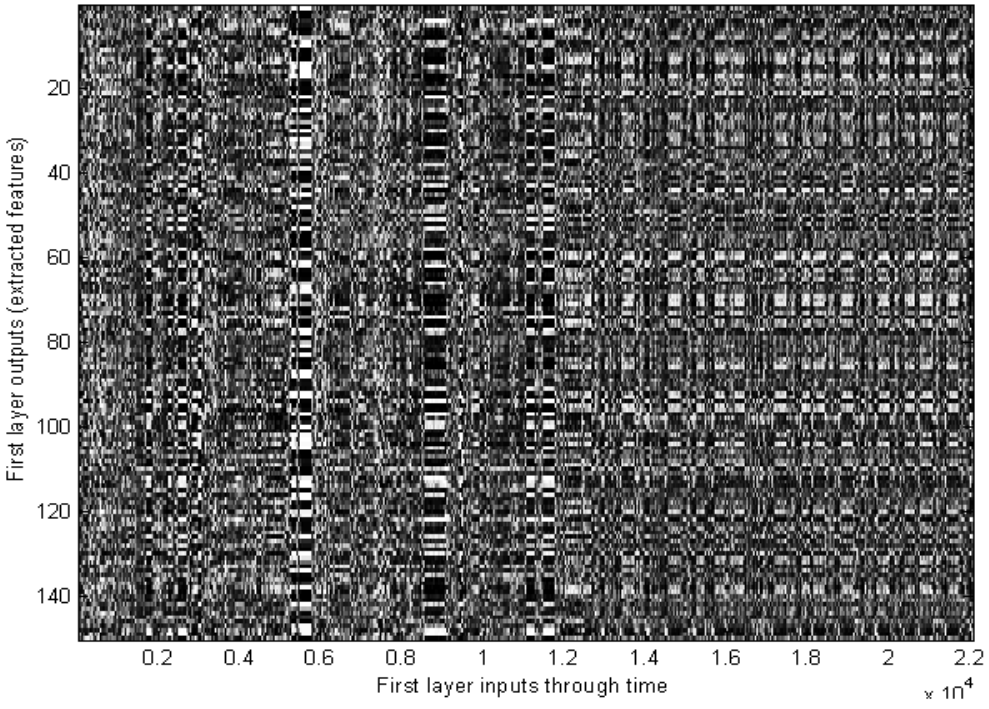


Figure. 4.4 Layer 1

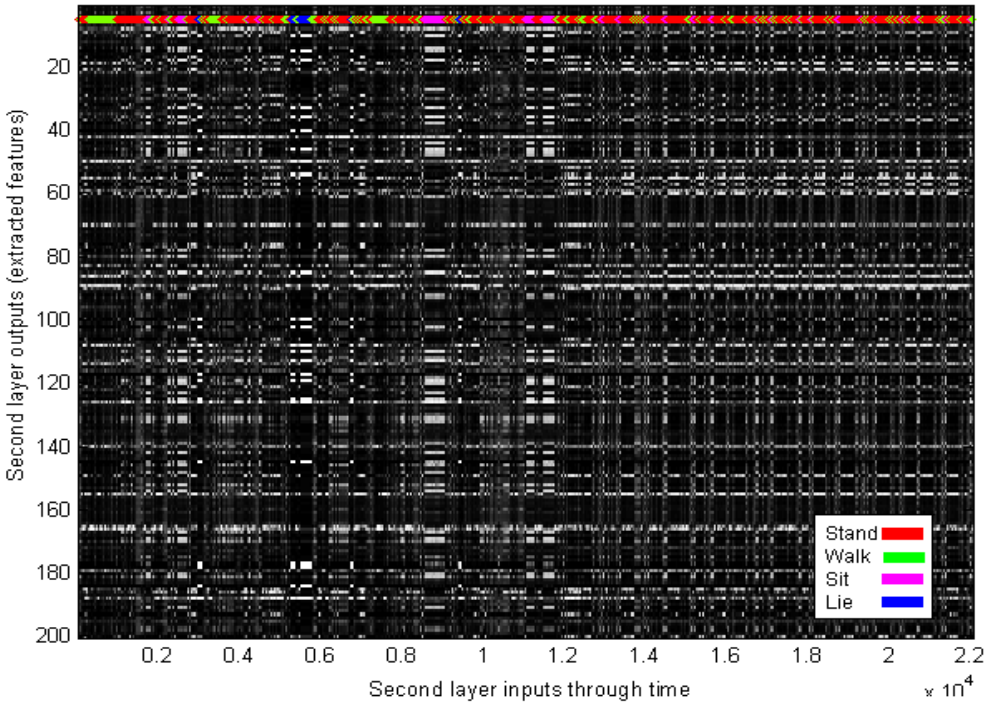


Figure. 4.5 Layer 2

4.6.2 Classification performance

In this set of experiments, we evaluate the classification performance of CRBM-OSC while active learning (AL) is not considered. The experiments are carried out on three subjects' Opp data with the goal of recognising user's modes of locomotion/gestures. CRBM parameters are set as explained in Sec. 4.6.1 (see Tab. 4.4). Hyper-prior can be put over the hyper-parameters of the Normal-Inverse-Wishart prior on cluster parameters (G_0). In [155], we propose a novel approach to use a non-parametric prior over the base distribution G_0 allowing for more flexible model which is much less sensitive to parameters (i.e., hyper-hyper-parameters). Alternatively, online non-parametric empirical Bayes can be proposed to find a point estimate of G_0 [156]. However, to keep the computation simple, we chose these hyper-parameters by hand, based on prior knowledge about the scale of the data. The mean u_0 is set to $\mathbf{0}$. The covariance matrix Σ_0 is roughly set to be large relative to the data. We set them to the identity matrix times the distance between the two farthest points in the data. The degree of freedom, v_0, must be greater than the number of dimensions d. We set it to $d+2$. The hyper-parameter k_0 is empirically set to 0.01. The forgetting factor λ in Eq.(4.14) is

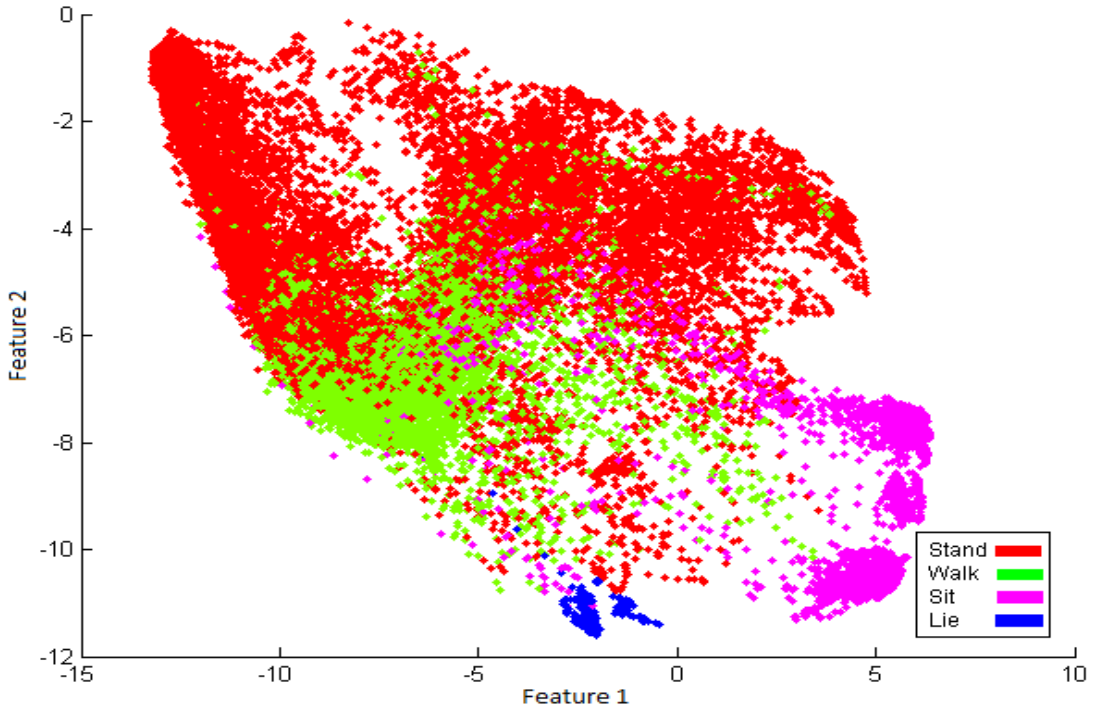


Figure. 4.6 Layer 3

empirically set to 0.95.

4.6.2.1 Locomotion

In order to show the efficiency of the proposed online algorithm, we compare it to well known offline classification models. DT (hoeffding decision tree) and SVM (support vector machines with polynomial kernel) have been efficiently applied for HAR in static environment [120]. We run two experiments with DT and SVM using two different training settings. In *setting 1*, SVM and DT are built from the datasets of all subjects excluding the one whose data is used for the evaluation. The results are presented in Tab. 4.5. In *setting 2*, training and testing are done using the data of the same subject with two-fold cross-validation. The results for each subject are averaged over 15 runs (see Tab. 4.6). We run SVM and DT with Weka 3.8 [157]. Although, DT and SVM are trained with more data in *setting 1*, all the results shown in *setting 2* are better. This variance may be explained by the distinct motion styles of the subjects. We adopt *setting 2* in the upcoming comparison. Next, we train our model CRBM-OSC online using *setting 2*(see Tab. 4.6). It can be seen that CRBM-OSC performs better on all subsets of the

Table. 4.5 Classification performance for locomotion activities under *setting 1*

Subject	Method	AA	Stand	Walk	Lie	Sit	ACA
S2	DT(%)	64.3	40	81.4	0	83.8	51.3
	SVM(%)	64.7	29.7	97.8	1.2	84	53.2
S3	DT(%)	41.9	27	95.6	0	0	41.1
	SVM(%)	73.6	84	59.7	0	68.4	53
S4	DT(%)	56.6	37.8	89.8	0	83.2	52.7
	SVM(%)	72.5	92	10.8	96.5	96.3	73.9

Table. 4.6 Classification performance for locomotion activities under *setting 2*

Subject	Method	AA	Stand	Walk	Lie	Sit	ACA
S2	CRBM-OSC(%)	**96**	**95.7**	**91.4**	98.4	**99.8**	**96.3**
	DT(%)	89.5	89.8	78.9	89.7	96.5	88.7
	SVM(%)	91.2	90.1	79.6	**99.3**	99.6	92.2
S3	CRBM-OSC(%)	**95.2**	**96.2**	**92.1**	98.3	**96.5**	**95.8**
	DT(%)	83.6	86.1	74.8	97.9	89.3	87
	SVM(%)	85	93.7	60.4	**99.9**	96.4	88.1
S4	CRBM-OSC(%)	**94.2**	**94.1**	**90.2**	99.3	**98.5**	**95.5**
	DT(%)	87.2	87.8	81.1	96	91.9	89.2
	SVM(%)	88	93.4	67.1	**99.9**	98.3	89.7
All	CRBM-OSC(%)	**96**	**95.7**	**94**	98.7	**98.4**	**96.7**
	DT(%)	83.3	94	56.6	96.9	88.6	84
	SVM(%)	84.5	92	60.7	**99.1**	96.6	87.2

data and for almost all activities. A reason for CRBM-OSC superiority may be explained by the ability of the algorithm to cope with dynamic changes in the individual activities. Based on this analysis and the previous one regarding the difference motion styles among the subjects, we can expect that the performance superiority of CRBM-OSC will increase if training is done on the datasets of all subjects. Thus, we run an experiment on the datasets of all subjects together using *setting 2* (see Tab. 4.6). It can be clearly seen that CRBM-OSC outperforms SVM and DT by far when all-subject data is considered compared to the case where only one-subject data is used.

Note that the performance of CRBM-OSC is consistent across all activities as its average class accuracy (ACA) is high. However, ACA is slightly better than the average accuracy (AA) for all methods. This is due to the plenty of available labels. In fact, the challenge lies in maintaining high ACA with few labels. Such challenge can be met by BSAL (to be discussed in the next section).

In order to show the effect of the feature extractor, we train OSC online using

Table. 4.7 Classification performance for locomotion activities

Subject	Method	AA	Stand	Walk	Lie	Sit	ACA
S2	CRBM-OSC(%)	**96**	95.7	**91.4**	**98.4**	**99.8**	**96.3**
	OSC(%)	93.52	**98.7**	77.8	99	89.7	91.3
S3	CRBM-OSC(%)	**95.2**	96.2	**92.1**	**98.3**	96.5	**95.8**
	OSC(%)	94.3	**99.1**	82.2	98.1	**97.7**	94.3
S4	CRBM-OSC(%)	**94.2**	94.1	**90.2**	**99.3**	**98.5**	**95.5**
	OSC(%)	92.6	**98.9**	74.9	96.5	98.2	92.1
All	CRBM-OSC(%)	**96**	**95.7**	94	**98.7**	**98.4**	**96.7**
	OSC(%)	93.8	80.3	**98.2**	88.6	97.1	91

setting 2, where CRBM is not considered (see Tab. 4.7). Generally, CRBM-OSC performs better in terms of average accuracy and average class accuracy. In addition, CRBM reduces the feature dimension from 113 to 10 which requires less computation and memory resources to process the streaming samples. We believe that with less discriminative low level features and more complex activities, CRBM will play more prominent role.

4.6.2.2 Gestures

Like the previous section (Sec. 4.6.2.1), we compare the performance of DT and SVM under the two training settings. SVM and DT are run with Weka 3.8 [157] and the results for each subject are averaged over 15 runs. The results obtained under both settings are presented in Tab. 4.8 and Tab. 4.9. Like in Sec. 4.6.2.1, DT and SVM, trained with less data under *setting 2*, show better performance than that of DT and SVM trained under *setting 1*. This variance confirms the assumtion drawn in Sec. 4.6.2.1 that there are changes in the data across different subjects. We will adopt *setting 2* in the upcoming comparisons.

Next, we train our model CRBM-OSC online with two-fold cross-validation. The results for each subject are averaged over 15 runs (see Tab. 4.9). CRBM-OSC average accuracy is better than SVM and DT's one on all the datasets. In addition, CRBM-OSC average accuracies are the best for the majority of the activities. Interestingly, the performance of CRBM-OSC increases when training is done on the datasets of all subjects. Hence, the analysis regarding the importance of adaptive learning is demonstrated for gesture activities too.

Unlike the case with the locomotion activities, we can notice that ACA is slightly lower than the average accuracy (AA) for all methods. This can be explained by

Table. 4.8 Classification performance for gesture activities under *setting 1*

Subject	Method	AA	Open Door1	Open Door2	Close Door1	Close Door2	Open Fridge	Close Fridge	Open Dishwasher	Close Dishwasher
S2	DT(%)	34	3.3	17.9	0	3.6	6.3	0	3.4	0.6
	SVM(%)	40	7.2	1	5.7	59.3	11.2	5.3	24	60.5
S3	DT(%)	51.3	11.1	2.2	8.7	67.4	6.8	43.3	45.9	10.7
	SVM(%)	40	18.5	7	0	66.1	1.1	1.1	28.3	0.7
S4	DT(%)	36.1	0	0	0	0	1.4	0	1.2	39.4
	SVM(%)	45	1.6	60.6	1.5	56.5	40.9	33.4	24.9	42.2

Open Drawer1	Close Drawer1	Open Drawer2	Close Drawer2	Open Drawer3	Close Drawer3	Clean Table	Drink Cup	Toggle Switch	ACA
13.3	0	0	27.5	75.1	0	27.8	64.3	0	18.1
9.8	0	25.8	69.5	72.8	7.6	24.6	61.1	5.5	26.5
57.6	65	62.2	17	77.8	0.3	59.7	78.7	92.3	37.8
12	16.7	0.2	3	92.9	0	64.2	67.1	93.8	27.8
39.5	0.1	0.5	0	0	0	59.4	88.7	59.5	17
0	36.3	36.7	59.9	62.2	46.5	6	63.6	94.6	39.3

the increase in the number of activities meaning that the same number of labels is distributed on larger number of classes. Thus, the risk of misclassifying the least frequent activities may increase. Noticeably, ACA values for SVM and DT decrease more significantly than that for CRBM-OSC. Such behaviour may be explained by the fact that OSC is a semi-supervised learning algorithm. That is, it uses both unlabelled and labelled data which curbs the consequence of scarce labels for certain activities.

4.6.3 Active learning performance

In this section, we evaluate the performance of the whole model including the active learning strategy. The evaluation of CRBM-OSC-BSAL is based on a prequential methodology: each time we get an instance, if its class label is revealed then it is used to train the learner. Otherwise, it is used for testing the classifier. The results are average over 30 runs. The parameters are set as in Sec. 4.6.2. Results obtained in the previous section (Tab. 4.6 and Tab. 4.9) are used for performance comparison.

4.6.3.1 Locomotion

We compare the results obtained in Sec. 4.6.2.1 (Tab. 4.6) to the ones of CRBM-OSC-BSAL with few queried data samples, around 5% for each subject. We also

Table. 4.9 Classification performance for gesture activities under *setting 2*

Subject	Method	AA	Open Door1	Open Door2	Close Door1	Close Door2	Open Fridge	Close Fridge	Open Dishwasher	Close Dishwasher
S2	CRBM-OSC(%)	**95**	**93.6**	**96**	**94.7**	**94.8**	93.6	**96.1**	**93.6**	90.1
	DT(%)	72.2	47.3	80.1	58.3	76.1	85.2	63.5	71.8	89.9
	SVM(%)	**95**	91.8	82.3	83	94.4	**95.9**	74	90.5	**99.2**
S3	CRBM-OSC(%)	**94.4**	**93.3**	**95.4**	**92.3**	**92.3**	**94.6**	**94.2**	**93.9**	94.2
	DT(%)	66.6	33.5	19.3	73.3	89.1	86.3	35.2	77.8	84.7
	SVM(%)	90.3	90.9	77.6	80.9	88.9	88.9	71.4	88.6	**98.2**
S4	CRBM-OSC(%)	**94.5**	**93.6**	**93.9**	**94**	**93.3**	**93.7**	**94.2**	**92.5**	92.3
	DT(%)	68.7	53.5	43.9	41.3	87.6	91.1	41.6	51	86.7
	SVM(%)	91.8	93.2	91.4	82.6	85.4	91.1	83.1	79.7	**98.5**
All	CRBM-OSC(%)	**94.9**	**92.5**	**93.9**	**94.6**	**94.1**	**94.6**	**94.7**	**93.3**	93.5
	DT(%)	57.1	24.2	18.9	61.6	81.8	86.6	46.4	46.4	43
	SVM(%)	84.5	79.3	74.1	64.9	82.2	82.9	66.2	75.8	**94.3**

Open Drawer1	Close Drawer1	Open Drawer2	Close Drawer2	Open Drawer3	Close Drawer3	Clean Table	Drink Cup	Toggle Switch	ACA
92.1	**94.9**	**93.6**	**91.6**	93.4	91.3	95	97.1	93.4	**93.8**
41.6	52.8	42.4	70.6	64.1	52	56.1	84	91.2	66.3
90.9	94.7	81.4	82.7	**98.4**	**93**	**97.6**	**99**	**97**	90.9
94	**95**	**93.2**	**92.5**	**93.9**	**95.2**	**92.5**	96.2	92.8	**93.9**
58.3	81	4.5	37.1	62.3	73.9	60.5	79	97.6	65.5
85.5	83.6	77.1	87.3	91.8	78.2	90	**98.7**	**99**	86.9
93.2	**95.5**	**93.9**	**93.4**	**94.8**	**92.7**	**93.4**	96.7	93.2	**93.8**
56.9	85.5	83.4	49	42.2	42.2	59.1	83.7	97.1	64.5
88.9	89.6	86.4	87.6	90.2	92.5	84.6	**99.5**	**99.5**	89.6
93.2	**94.7**	**94.7**	**93.7**	**94.8**	**95.2**	**93.6**	93.9	**97.2**	**94.3**
21.6	84.7	33.1	42.2	30	38.4	47.8	75.9	85.7	51.1
70.3	76.3	73.2	80.4	79.7	75.7	82.9	**97**	92.1	79.3

compare against the online method STAR [144] (already discussed in Sec. 4.2) which also queries 5% of the processed data. The results are shown in Tab. 4.10.

CRBM-OSC-BSAL shows better average accuracy (AA) than all competitors excluding CRBM-OSC. However, the AL strategy BSAL has reduced the number of labelled samples from 50% to around 5% while leading to an average (over all datasets) of arround 4.3% less accuracy compared to CRBM-OSC. CRBM-OSC-BSAL significantly outperforms STAR which also employs an AL strategy to query 5% of the data. Moreover, CRBM-OSC-BSAL outperforms SVM and DT even though the percentage of labels used for training is 45% higher. Similar to the results presented in Sec. 4.6.2, the performance of CRBM-OSC-BSAL is more significant when training is done on the datasets of all subjects. Hence, the proposed AL is capable of maintaining high performance when data evolves more substantially. Note that BSAL is able to maintain consistent performance across all activities. Indeed, the average class accuracy (ACA) is still high even though number of labels is 45% less than the one for CRBM-OSC. However, we can notice that ACA is not as better than AA as it is for CRBM-OSC. Though, this is normal

Table. 4.10 Classification performance for locomotion activities (5%)

Subject	Method	AA	Stand	Walk	Lie	Sit	ACA
S2	CRBM-OSC-BSAL(%)	93	88	90.1	93.2	99.3	92.7
	CRBM-OSC(%)	96	95.7	91.4	98.4	99.8	96.3
	STAR(%)	74.9	82.5	50.1	89.9	47.3	67.5
	DT(%)	89.5	89.8	78.9	89.7	96.5	88.7
	SVM(%)	91.2	90.1	79.6	99.3	99.6	92.2
S3	CRBM-OSC-BSAL(%)	86	88	81.5	98.3	84.2	88
	CRBM-OSC(%)	95.2	96.2	92.1	98.3	96.5	95.8
	STAR(%)	68.6	87	44.3	86.1	7.2	56.2
	DT(%)	83.6	86.1	74.8	97.9	89.3	87
	SVM(%)	85	93.7	60.4	99.9	96.4	88.1
S4	CRBM-OSC-BSAL(%)	91.4	91.8	85.4	92.9	97.9	92
	CRBM-OSC(%)	94.2	94.1	90.2	99.3	98.5	95.5
	STAR(%)	71.1	84.2	43.6	90	30.3	62
	DT(%)	87.2	87.8	81.1	96	91.9	89.2
	SVM(%)	88	93.4	67.1	99.9	98.3	89.7
All	CRBM-OSC-BSAL(%)	93.6	93.6	90.8	96.5	97.6	94.6
	CRBM-OSC(%)	96	95.7	94	98.7	98.4	96.7
	DT(%)	83.3	94	56.6	96.9	88.6	84
	SVM(%)	84.5	92	60.7	99.1	96.6	87.2

as the number of labels has dramatically decreased. Therefore, it is a strong point of BSAL to keep the ACA high (even higher than AA) with few labels.

In order to visualize the behaviour of BSAL, we draw the prequential labelling rate as the data streams pass though CRBM-OSC-BSAL (see Fig. 4.7, Fig. 4.8 and Fig. 4.9). To smooth the curve, a fading factor of 0.999 is used to compute the learning rate. Some observations on BSAL behaviour can be deduced from Fig. 4.7, Fig. 4.8 and Fig. 4.9. First, BSAL tends to query the activities appearing for the first time, then the labelling rate falls down. For instance, the labelling rate across *lying* and *sitting* activities is high when the activities first appear. In the second appearance of the same activities, the labelling rate is not triggered. Second, long lasting activities are infrequently queried which maintains the performance over evolving streams. Third, we notice that sometimes the labelling rate across *standing* activity suddenly grows. Such behaviour may be caused by the gesture activities which are usually performed while the user is standing. Hence, BSAL expects change or emergence in the activities leading to an increase in the labelling rate. *Walking* activity is peculiar because it is the most frequent and prone to change more than the others. However, when it lasts for considerable duration, labelling rate decreases.

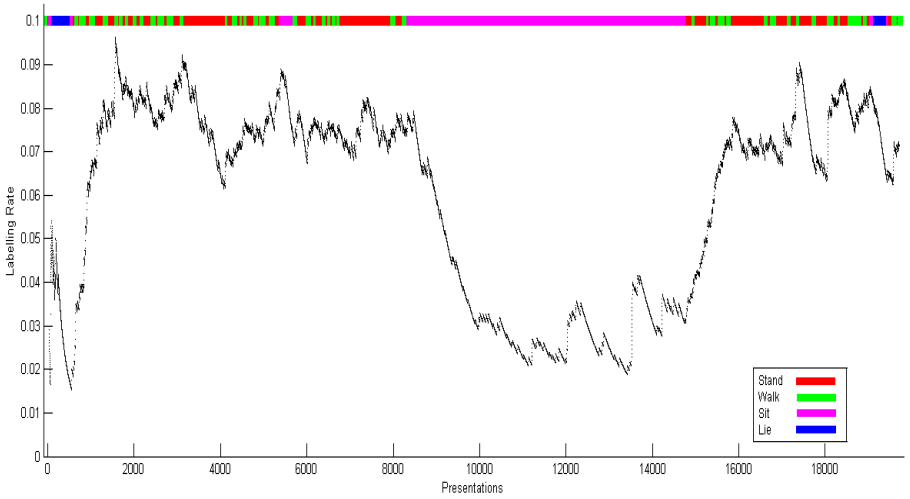


Figure. 4.7 Labelling rate along the stream of subject 2 (S2)

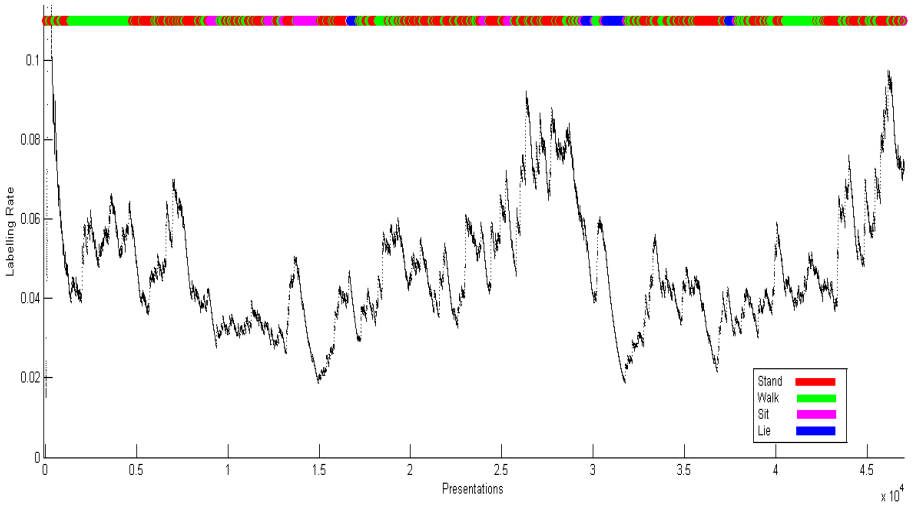


Figure. 4.8 Labelling rate along the stream of subject 3 (S3)

4.6.3.2 Gestures

We compare the results obtained in Sec. 4.6.2.2 to the ones of CRBM-OSC-BSAL with few queried data samples, around 5% for each subject (see Tab. 4.11).

Although, CRBM-OSC-BSAL uses only 5% labels for a relatively high number of activities, its AA for S3 is better than the one of SVM and DT which use 50% labels. SVM outperforms CRBM-OSC-BSAL for S2 and S4, but at the cost of

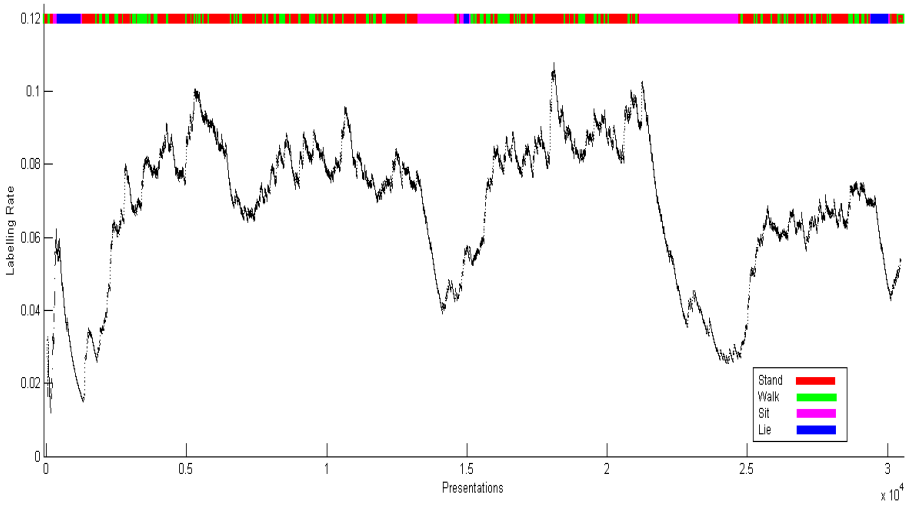


Figure. 4.9 Labelling rate along the stream of subject 4 (S4)

high labelling (45% more labels). Nevertheless, CRBM-OSC-BSAL shows better AA results when the datasets of all subjects are used. This highlights BSAL's ability to maintain high performance under more aggressive changes.

Predictably, CRBM-OSC-BSAL's ACA drops in comparison with CRBM-OSC's ACA as the number of labels has dramatically decreased. In addition, the relatively large number of activities makes it harder to maintain consistent high accuracy across all activities. Thus, we run another experiment with 10% of the data samples queried (see Tab. 4.12). It can be seen that CRBM-OSC-BSAL's ACA surpasses the one of SVM and DT that are trained with 40% more data. Furthermore, CRBM-OSC-BSAL's ACA becomes comparable to CRBM-OSC's. We can also notice that AA has improved and CRBM-OSC-BSAL outperforms SVM and DT for all subjects and has almost the same AA as CRBM-OSC trained with 40% more data.

4.7 Conclusion

In this chapter, we propose a novel online AL algorithm (BSAL) that can handle *concept evolution* in data stream involving classes of unbalanced proportions. BSAL is an information-based AL algorithm that selects data instances leading to more information gain. We also propose a new learning model equipped with BSAL to process data streams with sequential temporal dependency. This model composed of a feature extractor (CRBM), an online semi-supervised classifier (OSC)

Table. 4.11 Classification performance for gesture activities (5%)

Subject	Method	AA	Open Door1	Open Door2	Close Door1	Close Door2	Open Fridge	Close Fridge	Open Dishwasher	Close Dishwasher
S2	CRBM-OSC-BSAL(%)	85.2	81.3	55.8	77.1	96	96.8	76.5	89.7	90.9
	CRBM-OSC(%)	95	93.6	96	94.7	94.8	93.6	96.1	93.6	90.1
	DT(%)	72.2	47.3	80.1	58.3	76.1	85.2	63.5	71.8	89.9
	SVM(%)	95	91.8	82.3	83	94.4	95.9	74	90.5	99.2
S3	CRBM-OSC-BSAL(%)	91.1	87.6	84.2	72.7	95.8	94.1	80.1	88.6	92.3
	CRBM-OSC(%)	94.4	93.3	95.4	92.3	92.3	94.6	94.2	93.9	94.2
	DT(%)	66.6	33.5	19.3	73.3	89.1	86.3	35.2	77.8	84.7
	SVM(%)	90.3	90.9	77.6	80.9	88.9	88.9	71.4	88.6	98.2
S4	CRBM-OSC-BSAL(%)	85.6	77.7	53.9	57.5	91.7	92	53.5	85.1	87.5
	CRBM-OSC(%)	94.5	93.6	93.9	94	93.3	93.7	94.2	92.5	92.3
	DT(%)	68.7	53.5	43.9	41.3	87.6	91.1	41.6	51	86.7
	SVM(%)	91.8	93.2	91.4	82.6	85.4	91.1	83.1	79.7	98.5
All	CRBM-OSC-BSAL(%)	89.6	82.4	82.4	66	64.8	95.4	93.4	73.1	86.8
	CRBM-OSC(%)	94.9	92.5	93.9	94.6	94.1	94.6	94.7	93.3	93.5
	DT(%)	57.1	24.2	18.9	61.6	81.8	86.6	46.4	46.4	43
	SVM(%)	84.5	79.3	74.1	64.9	82.2	82.9	66.2	75.8	94.3

Open Drawer1	Close Drawer1	Open Drawer2	Close Drawer2	Open Drawer3	Close Drawer3	Clean Table	Drink Cup	Toggle Switch	ACA
95.7	86.7	95	96.8	96.6	95.4	94.4	77.9	97.9	82.6
92.1	94.9	93.6	91.6	93.4	91.3	95	97.1	93.4	93.8
41.6	52.8	42.4	70.6	64.1	52	56.1	84	91.2	66.3
90.9	94.7	81.4	82.7	98.4	93	97.6	99	97	90.9
93.7	88.7	84.5	93.9	95.4	88.2	95.1	93	94.7	89.5
94	95	93.2	92.5	93.9	95.2	92.5	96.2	92.8	93.9
58.3	81	4.5	37.1	62.3	73.9	60.5	79	97.6	65.6
85.5	83.6	77.1	87.3	91.8	78.2	90	98.7	99	86.9
84.3	73	62.7	88.9	83	84.8	91.9	97	90.5	79.7
93.2	95.5	93.9	93.4	94.8	92.7	93.4	96.7	93.2	93.8
56.9	85.5	83.4	49	42.2	42.2	59.1	83.7	97.1	64.5
88.9	89.6	86.4	87.6	90.2	92.5	84.6	99.5	99.5	89.6
87.9	91	84	80.5	90	92.3	87	93.9	98	80.9
93.2	94.7	94.7	93.7	94.8	95.2	93.6	93.9	97.2	94.3
21.6	84.7	33.1	42.2	30	38.4	47.8	75.9	85.7	51.1
70.3	76.3	73.2	80.4	79.7	75.7	82.9	97	92.1	79.3

and BSAL to cope the challenges of sequential data streams. CRBM helps overcome the weary features hand-crafting by learning generic features from unlabelled high-dimensional input. OSC online learns from stream of generic features. BSAL queries the samples that are expected to bring crucial information for OSC. Knowing that the proposed approach is generic and not dedicated to specific application, human activity recognition (HAR) is used as a real-world application.

Experimental results on real-world activity recognition datasets show the effectiveness of the proposed model. Although, the classifier and the active learning work online, the feature extractor pre-training step needs further attention in order to accommodate it into the online setting. Another future work is to compare SAL and BSAL on the same datasets that show multiple classes of severely unbalanced proportions.

Table. 4.12 Classification performance for gesture activities (10%)

Subject	Method	AA	Open Door1	Open Door2	Close Door1	Close Door2	Open Fridge	Close Fridge	Open Dishwasher	Close Dishwasher
S2	CRBM-OSC-BSAL(%)	97.2	86.5	81.1	87.8	98.8	98.1	91.9	95.3	96.2
	CRBM-OSC(%)	95	93.6	96	94.7	94.8	93.6	96.1	93.6	90.1
	DT(%)	72.2	47.3	80.1	58.3	76.1	85.2	63.5	71.8	89.9
	SVM(%)	95	91.8	82.3	83	94.4	95.9	74	90.5	99.2
S3	CRBM-OSC-BSAL(%)	94	90.2	86.8	75.6	95.7	95.5	82.4	91	94
	CRBM-OSC(%)	94.4	93.3	95.4	92.3	92.3	94.6	94.2	93.9	94.2
	DT(%)	66.6	33.5	19.3	73.3	89.1	86.3	35.2	77.8	84.7
	SVM(%)	90.3	90.9	77.6	80.9	88.9	88.9	71.4	88.6	98.2
S4	CRBM-OSC-BSAL(%)	94.2	88.2	86.9	82.3	96.3	95.1	84.8	93.9	93.9
	CRBM-OSC(%)	94.5	93.6	93.9	94	93.3	93.7	94.2	92.5	92.3
	DT(%)	68.7	53.5	43.9	41.3	87.6	91.1	41.6	51	86.7
	SVM(%)	91.8	93.2	91.4	82.6	85.4	91.1	83.1	79.7	98.5
All	CRBM-OSC-BSAL(%)	91.4	84.6	87.6	69	74	96	95.4	76.7	90.5
	CRBM-OSC(%)	94.9	92.5	93.9	94.6	94.1	94.6	94.7	93.3	93.5
	DT(%)	57.1	24.2	18.9	61.6	81.8	86.6	46.4	46.4	43
	SVM(%)	84.5	79.3	74.1	64.9	82.2	82.9	66.2	75.8	94.3

Open Drawer1	Close Drawer1	Open Drawer2	Close Drawer2	Open Drawer3	Close Drawer3	Clean Table	Drink Cup	Toggle Switch	ACA
97.8	97.9	98.2	98.7	98.5	95.4	97.7	99.6	100	95.3
92.1	94.9	93.6	91.6	93.4	91.3	95	97.1	93.4	93.8
41.6	52.8	42.4	70.6	64.1	52	56.1	84	91.2	66.3
90.9	94.7	81.4	82.7	98.4	93	97.6	99	97	90.9
94.5	92.1	83.4	95.1	96.5	91	95.4	98.8	94.9	91.4
94	95	93.2	92.5	93.9	95.2	92.5	96.2	92.8	93.9
58.3	81	4.5	37.1	62.3	73.9	60.5	79	97.6	65.6
85.5	83.6	77.1	87.3	91.8	78.2	90	98.7	99	86.9
93	89	85.7	96.1	94.5	96.2	94.8	98.9	94.4	92
93.2	95.5	93.9	93.4	94.8	92.7	93.4	96.7	93.2	93.8
56.9	85.5	83.4	49	42.2	42.2	59.1	83.7	97.1	64.5
88.9	89.6	86.4	87.6	90.2	92.5	84.6	99.5	99.5	89.6
92.4	93.4	83.2	83.9	91.8	94.3	87.7	94.6	98.4	88
93.2	94.7	94.7	93.7	94.8	95.2	93.6	93.9	97.2	94.3
21.6	84.7	33.1	42.2	30	38.4	47.8	75.9	85.7	51.1
70.3	76.3	73.2	80.4	79.7	75.7	82.9	97	92.1	79.3

Chapter 5

Conclusions and Future Work

This chapter summarizes the contributions of this dissertation and discusses the future directions.

5.1 Conclusion

The research questions that this thesis revolves around whether online AL can improve performance of classification with less resources, compared to commonly used online passive learning (classification). Many similar studies have been done in offline setting, where AL has been shown to reduce the amount of data that needs to be actively labelled, while at the same time, increasing the quality of the resulting classifiers. Moving from offline to online setting is not straightforward. In online setting, data comes as an unbounded stream, where *concept drift* and *concept evolution* can occur at any time. Furthermore, the *sampling bias* problem emerges as a side effect of AL which is more severe in online setting under the presence of *concept drift* and *concept evolution*. In this thesis, we have demonstrated that AL can improve learning from data streams. We have also shown that AL can be harnessed to handle *concept drift* and *concept evolution*. Furthermore, *sampling bias* has been attenuated, while maintaining AL advantages. Finally, we have shown that AL can also improve learning even for sequential data steams in the presence of *concept drift* and *concept evolution*.

These contributions have been the result of three algorithms proposed in three chapters. The core idea of these algorithms is to query samples that give the largest

reduction of an expected loss function that measures the learning performance. Two types of AL have been proposed: *decision theory* based AL whose losses involve the prediction error and *information theory* based AL whose losses involve the model parameters. The summary of each chapter contribution and limitation can be presented as follow:

- *Chapter 2* proposes an active learning algorithm, BAL, for data streams capable of dealing with changes of the data distribution. The algorithm labels the samples with high uncertainty and representativeness in a completely online scenario. It also tackles the *sampling bias* of active learning with potentially adversarial *concept drift*.

  Experimental results on real-world data showed the limitation of the proposed approach when the budget is high or the drift occurrence is rare and smooth. However, the main goal of reducing the labelling cost in presence of *concept drift*, while maintaining good accuracy, has been achieved.

  To sum up, this chapter shows that AL can improve binary classification of drifting data streams coping well with the problem of *sampling bias*.

- *Chapter 3* proposes an active learning algorithm, SAL, for data streams to deal 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 are very good in general. Unfortunately, SAL shows some limitations for class discovery when applied to highly unbalanced data. However, the main goal of the proposed algorithm is for classification in presence of unknown number of classes not specifically for unbalanced data. Furthermore, the performance of SAL with respect to class discovery is comparable to the state-of-the-art and is even better when the classes are not severely unbalanced.

  To sum up, this chapter shows that AL is capable of dealing with both *concept drift* and *concept evolution* together despite the problem of *sampling bias*.

- *Chapter 4* proposes a novel online AL algorithm (BSAL) that can handle *concept evolution* in data stream involving classes of unbalanced proportions. BSAL is an *information theory* based AL algorithm that selects data

instances leading to more information gain. This chapter also proposes a new learning model equipped with BSAL to process data streams with sequential temporal dependency. This model composed of a feature extractor (CRBM), an online semi-supervised classifier (OSC) and BSAL to cope the challenges of sequential data streams. CRBM helps overcome the weary features handcrafting by learning generic features from unlabelled high-dimensional input. OSC online learns from stream of generic features. BSAL queries the samples that are expected to bring crucial information for OSC. Knowing that the proposed approach is generic and not dedicated to specific application, human activity recognition (HAR) is used as a real-world application.

Experimental results on real-world activity recognition datasets show the effectiveness of the proposed model. Although, the classifier and the active learning work online, the feature extractor pre-training step needs further attention in order to accommodate it into the online setting.

To sum up, this chapter shows that AL can deal with sequential data streams in presence of *concept drift* and *concept evolution*, even when data streams involve classes of unbalanced proportions. It also presents a real-world application of AL, namely human activity recognition (HAR).

5.2 Future Work

This thesis opens up many avenues for future research, some of which are listed below. In short terms, we foresee several directions for research on dealing with the shortcomings of the present version of the proposed algorithms.

- **Extending BAL**: the two criteria of BAL can be dynamically combined yielding better result under any kind of drift over the whole budget line.
- **Extending SAL**: the performance of class discovery from severely unbalanced datasets can be improved by increasing the importance of rare classes resulting in sampling bias towards rare classes. Developing online AL-based novelty detection algorithm inspired by SAL can be another future work.
- **Extending CRBM-OSC-BSAL**: A potential improvement of this algorithm is to upgrade it to cope with *sampling bias*. Exploiting the model in other applications such as industrial maintenance can also be a potential

future investigation. Another future work is to eliminate the pre-training step by accommodating the feature extractor into the online setting.

- **Scalability**: An interesting and novel development of the proposed algorithms is to make them scalable by distributing them on many machines. Performing machine learning algorithms in a distributed environment first involves conceptually converting single-threaded algorithms to parallel algorithms. The second step involves implementing the parallel algorithms.

  The distributed architecture which we proposed in [158] could be employed to convert the proposed algorithms to parallel ones. Integrating these parallel algorithms into SOLMA[1], which is implemented on top of Apache Flink, allows sort of high level distributed implementation with no need to understand the system configurations.

In long terms, many open questions regarding online active learning need be addressed to make it possible to apply AL for data streams in more realistic scenarios, in particular, relaxing the assumptions related to the annotators (oracles):

- **Labelling delay**: An assumption in this thesis is that the oracle provides the label of the queried samples in no time. In reality, delayed labelling occurs and does affect the performance when the stream velocity is higher than the labelling speed. This makes the adaptation of the learning algorithm more challenging. However, for some real-world applications, labelling delay and its effect can be diminished. In these applications, the oracle feeding the labels is the same as that controls the data. To curb the delay more, offering label choices instead of asking about the labels could be easily done. Furthermore, choices could be limited to specific set of the labels according to some criteria.

- **Noisy oracles**: Another strong assumption taken in this thesis is that the oracle always gives correct labels. However, uncertainty arises whether the labels come from human experts (one or multiple experts) or as a result of an empirical experiment. The instrumentation of experimental setting may produce some noise. Human experts may not always be reliable, for several

[1]Solma is a scalable online machine learning algorithms (including classification, clustering, regression, ensemble algorithms, graph oriented algorithms, linear algebra operations, and anomaly and novelty detection. https://github.com/proteus-h2020/proteus-solma) implemented on top of Apache Flink using the hybrid processing capabilities.

reasons. First, some instances are implicitly difficult for people and machines and second, people can become distracted or tired over time. The question remains about how to use noisy oracles in active learning. In particular, when should the learner decide to query for the (potentially noisy) label of a new unlabelled instance? or querying for repeated labels to de-noise an existing training instance that seems a bit off?

- **Multiplicity of oracles**: The recent introduction of Internet-based "crowdsourcing" tools such as Amazons Mechanical Turk[2] and the clever use of online annotation games[3] have enabled some researchers to attempt to "average out" some of this noise by cheaply obtaining labels from multiple non-experts. Such approaches have been used to produce gold-standard quality training sets and also to evaluate learning algorithms on data for which no gold-standard labelling exists.

- **Labelling costs**: In many applications there is variance in the cost of obtaining labels. If our goal in active learning is to minimize the overall cost of training an accurate model, then simply reducing the number of labelled instances does not necessarily guarantee a reduction in overall labelling cost. Active learning approaches which ignore cost may perform no better than random selection (i.e., passive learning). The cost of labelling data samples could be expressed as different weights attached to each sample to represent its labelling cost. Then, existing AL approaches could be simply applied.

One of the future directions is to develop online AL for other machine learning problems such as parameter estimation and regression. In such problems, queries do not have to be about revealing the labels, but rather other type of information feedback such as group assignment and relation between instances.

Finally, finding theoretical guarantees for online AL is an important and challenging future work. Online AL relation to bandit problems and reinforcement learning can be key for such a direction.

[2]https://www.mturk.com/mturk/welcome
[3]http://www.gwap.com/

Appendix A

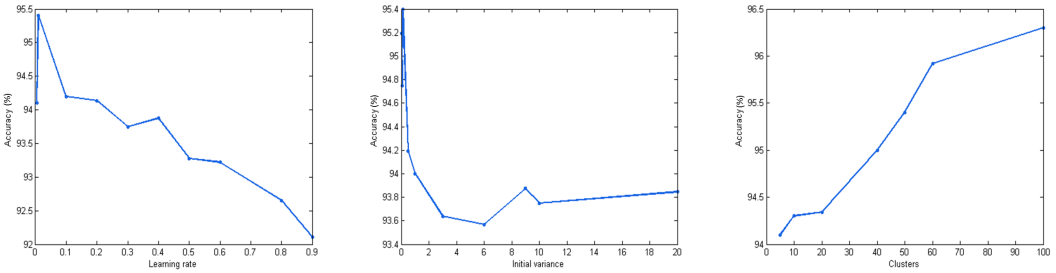
A.1 Parameters setting of Growing Gaussian Mixture Model

To find the parameters giving the best performance, we need to maximize the accuracy of BAL for each dataset. Here, we have three variables: variance, learning rate, and number of clusters. Obviously, carrying out many experiments with three degrees of freedom is time costly. In order to estimate these parameters with less time, we use prior knowledge about the dynamicity of the model. We already know from the previous experiments that the accuracy increases as the number of clusters increases (Fig. 2.11).

Thus, we can set the maximum number of clusters to 50 and maximize the accuracy of BAL with respect to variance, and learning rate. The removal and creation of clusters depend on the coverage of the input by the existing clusters. Thus, small variance with insufficient number of clusters results in instability and inaccurate density estimation. On the other hand, small variance covers thoroughly the input space as long as the number of clusters is enough.

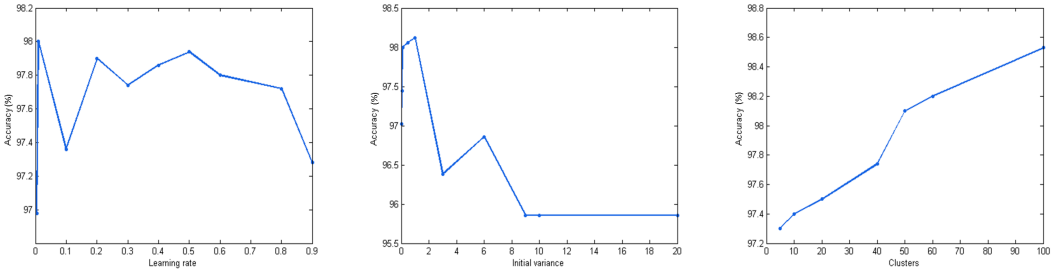
We empirically set the variance to 0.1. We can now estimate the maximum accuracy of BAL for each variable separately. Figures A.1a, A.2a, A.3a, and A.4a show the possible optimal values of the learning rate for each dataset. Figures A.1b, A.2b, A.3b, and A.4b show the effect of the variance on the accuracy while keeping the learning rate set to the optimal values obtained from the previous experiment. Finally, Fig. A.1c, A.2c, A.3c, and A.4c depict the accuracy when

the number of clusters increases. As the number of clusters increases, the computational time increases also, while the accuracy does not change substantially. Thus we set the number of clusters to 50 for all datasets. Table 2.1 summarizes the obtained optimal values of the parameters.



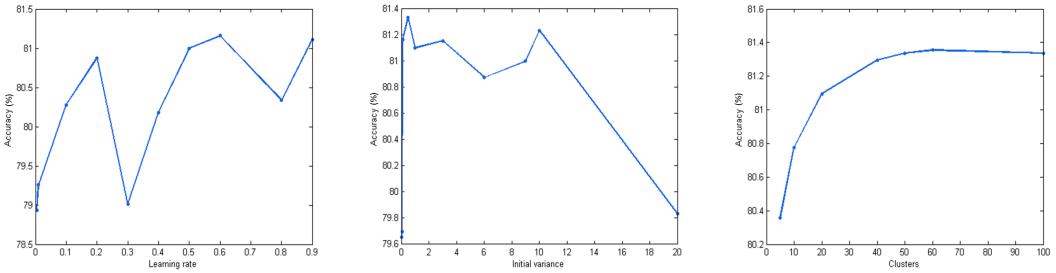
(a) Learning rate (b) Variance (c) Number of clusters

Figure. A.1 Gradual drift



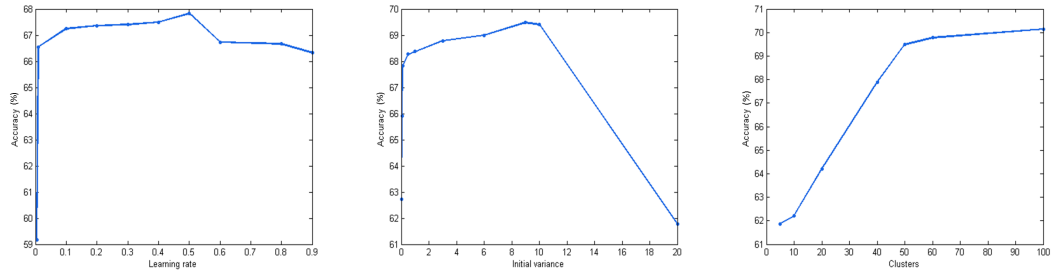
(a) Learning rate (b) Variance (c) Number of clusters

Figure. A.2 Mixture drift



(a) Learning rate (b) Variance (c) Number of clusters

Figure. A.3 Electricity



(a) Learning rate (b) Variance (c) Number of clusters

Figure. A.4 Airlines

Appendix B

B.1 Compute Eq. (3.22):

- If z_t refers to a new component:

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

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 $\boldsymbol{\theta}$.

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

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

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

where d is the dimension of the data.

- If z_t refers to an already seen component:

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

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

$$\boldsymbol{\Sigma}_2 = \frac{1}{(k_0 + n_{z_t,t-1})(v_0 + n_{z_t,t-1} - d + 1)} \Big(\boldsymbol{\Sigma}_0 + \boldsymbol{sc}_{z_t,t-1} + \frac{k_0 n_{z_t,t-1}}{k_0 + n_{z_t,t-1}} (\boldsymbol{su}_{z_t,t-1} - \boldsymbol{\mu}_0) (\boldsymbol{su}_{z_t,t-1} - \boldsymbol{\mu}_0)^T\Big)(k_0 + n_{z_t,t-1} + 1) \tag{B.7}$$

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

B.2 Compute the first term of Eq. (3.31):

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

$$p(y_t|z_t, D_{t-1}) = p(y_t|z_t, y_{t-1}) \tag{B.9}$$

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. (3.19), we end up with the following equations:

$$p(y_t|z_t, D_{t-1}) = \begin{cases} \frac{n_{z_t,y_t,t}}{\alpha_1 + n'_{z_t,t} - 1} \propto n'_{z_t,y_t,t} & y_t \text{ is an existing class} \\ \frac{\alpha_1}{\alpha_1 + n'_{z_t,t} - 1} \propto \alpha_1 & y_t \text{ is a new class} \end{cases} \tag{B.10}$$

Appendix C

C.1 Computation of Eq. (4.28)

After marginalizing out the Gaussian and the stick-breaking components, $\boldsymbol{\psi}_t$ becomes equivalent to $\boldsymbol{z}_{1:t}$. Thus, the discrepancy between the current risk and the current expected risk can be written as follows:

$$\hat{\Delta}(\boldsymbol{x}_t, y_t | D_{t-1}, q_t) = R(p(z_{1:t}|D_{t-1}, \boldsymbol{x}_t), \hat{z}_{1:t}) - \hat{R}(p(z_{1:t}|D_{t-1}, \boldsymbol{x}_t, y_t), \hat{z}_{1:t}; q_t) \quad \text{(C.1)}$$

$$\begin{aligned} R(p(z_{1:t}|D_{t-1}, \boldsymbol{x}_t), \hat{z}_{1:t}) &= E_{z_{1:t} \sim p(z_{1:t}|D_{t-1}, \boldsymbol{x}_t)}[L(z_{1:t}, \hat{z}_{1:t})] \\ &= \sum_{z_{1:t}} p(z_{1:t}|D_{t-1}, \boldsymbol{x}_t) L(z_{1:t}, \hat{z}_{1:t}) \end{aligned} \quad \text{(C.2)}$$

Given that the data instance at time t is queried ($q_t = 1$), the current expected risk can presented as follows:

$$\hat{R}(p(z_{1:t}|D_{t-1}, \boldsymbol{x}_t, y_t), \hat{z}_{1:t}; q_t = 1) = \sum_{y_t} p(y_t|D_{t-1}, \boldsymbol{x}_t) R(p(z_{1:t}|D_{t-1}, \boldsymbol{x}_t, y_t), \hat{z}_{1:t}) \quad \text{(C.3)}$$

To compute Eq. (C.1), both Eq. (C.2) and Eq. (C.3) must be solved. Given that the loss function in Eq. (C.2) is solved, the computation of Eq. (C.2) and Eq. (C.3) is straightforward. Thus, we start by the loss function which can be written as follows:

$$L(z_{1:t}, \hat{z}_{1:t}) = A - B \quad \text{(C.4)}$$

where:

$$A = \sum_y \int_{\boldsymbol{x}} p(\boldsymbol{x}, y|D_t, z_{1:t}) \log\big(p(\boldsymbol{x}, y|D_t, z_{1:t})\big) d\boldsymbol{x} \tag{C.5}$$

$$B = \sum_y \int_{\boldsymbol{x}} p(\boldsymbol{x}, y|D_t, z_{1:t}) \log\big(p(\boldsymbol{x}, y|D_t, \hat{z}_{1:t})\big) d\boldsymbol{x} \tag{C.6}$$

$$A = A_1 + A_2 \tag{C.7}$$

where

$$A_1 = \sum_y \log\big(p(y|D_t, z_{1:t})\big) p(y|D_t, z_{1:t}) \sum_z p(z|y, z_{1:t}, D_t) \int_{\boldsymbol{x}} \int_{\boldsymbol{\theta}} p(\boldsymbol{x}|z, \boldsymbol{\theta}) p(\boldsymbol{\theta}|z, D_t, z_{1:t}) d\boldsymbol{\theta} d\boldsymbol{x} \tag{C.8}$$

The integral over $\boldsymbol{\theta}$ leads to a t-student distribution as shown in Eq. (4.7). Hence,

$$A_1 = \sum_y \log\big(p(y|D_t, z_{1:t})\big) p(y|D_t, z_{1:t}) \tag{C.9}$$

where the terms of Eq. (C.9) are already computed in Eq. (4.3).

$$A_2 = \sum_y p(y|D_t, z_{1:t}) \sum_z p(z|y, z_{1:t}, D_t) \int_{\boldsymbol{x}} \int_{\boldsymbol{\theta}} p(\boldsymbol{x}|z, \boldsymbol{\theta}) p(\boldsymbol{\theta}|z, D_t, z_{1:t}) d\boldsymbol{\theta} \log\Bigg(\sum_z p(z|y, z_{1:t}, D_t) \int_{\boldsymbol{\theta}} p(\boldsymbol{x}|z, \boldsymbol{\theta}) p(\boldsymbol{\theta}|z, D_t, z_{1:t}) d\boldsymbol{\theta} \Bigg) d\boldsymbol{x} \tag{C.10}$$

Similarly, we compute B:

$$B = B_1 + B_2 \tag{C.11}$$

$$B_1 = \sum_y \log\big(p(y|D_t, \hat{z}_{1:t})\big) p(y|D_t, z_{1:t}) \sum_z p(z|y, z_{1:t}, D_t) \int_{\boldsymbol{x}} \int_{\boldsymbol{\theta}} p(\boldsymbol{x}|z, \boldsymbol{\theta}) p(\boldsymbol{\theta}|z, D_t, z_{1:t}) d\boldsymbol{\theta} d\boldsymbol{x} \tag{C.12}$$

$$B_1 = \sum_y \log\big(p(y|D_t, \hat{z}_{1:t})\big) p(y|D_t, z_{1:t}) \tag{C.13}$$

$$B_2 = \sum_y p(y|D_t, z_{1:t}) \sum_z p(z|y, z_{1:t}, D_t) \int_{\boldsymbol{x}} \int_{\boldsymbol{\theta}} p(\boldsymbol{x}|z, \boldsymbol{\theta}) p(\boldsymbol{\theta}|z, D_t, z_{1:t}) d\boldsymbol{\theta} \log \Bigg(\sum_z p(z|y, \hat{z}_{1:t}, D_t) \int_{\boldsymbol{\theta}} p(\boldsymbol{x}|z, \boldsymbol{\theta}) p(\boldsymbol{\theta}|z, D_t, \hat{z}_{1:t}) d\boldsymbol{\theta} \Bigg) d\boldsymbol{x} \tag{C.14}$$

Given that

$$\begin{aligned} Q_1(\boldsymbol{x}) &= \sum_z p(z|y, z_{1:t}, D_t) \int_{\boldsymbol{\theta}} p(\boldsymbol{x}|z, \boldsymbol{\theta}) p(\boldsymbol{\theta}|z, D_t, z_{1:t}) d\boldsymbol{\theta} \\ Q_2(\boldsymbol{x}) &= \sum_z p(z|y, \hat{z}_{1:t}, D_t) \int_{\boldsymbol{\theta}} p(\boldsymbol{x}|z, \boldsymbol{\theta}) p(\boldsymbol{\theta}|z, D_t, \hat{z}_{1:t}) d\boldsymbol{\theta} \\ g(\boldsymbol{x}) &= \log \frac{Q_1(\boldsymbol{x})}{Q_2(\boldsymbol{x})} \end{aligned} \tag{C.15}$$

$A_2 - B_2$ can be written as follows:

$$\begin{aligned} A_2 - B_2 &= \sum_y p(y|D_t, z_{1:t}) \int_{\boldsymbol{x}} Q_1(\boldsymbol{x}) g(\boldsymbol{x}) d\boldsymbol{x} \\ &= \sum_y p(y|D_t, z_{1:t}) \sum_z \int_{\boldsymbol{x}} p(z|y, z_{1:t}, D_t) t(\boldsymbol{x}|z, D_t, z_{1:t}) g(\boldsymbol{x}) d\boldsymbol{x} \end{aligned} \tag{C.16}$$

where $t(\boldsymbol{x}|.)$ refers to student's t-distribution. We can approximate each tern in this sum with a second order Taylor series expansion of $g(\boldsymbol{x})$ around the means $\boldsymbol{\mu}_z \equiv \boldsymbol{\mu}_{(z,z_1:t,D_t)}$ of the student's t-distribution's components:

$$g(\boldsymbol{x}) \approx \hat{g}_{\boldsymbol{\mu}_z}(\boldsymbol{x}) = g(\boldsymbol{\mu}_z) + \nabla g(\boldsymbol{\mu}_z)(\boldsymbol{x} - \boldsymbol{\mu}_z) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_z)^T \nabla^2 g(\boldsymbol{\mu}_z)(\boldsymbol{x} - \boldsymbol{\mu}_z) \tag{C.17}$$

where ∇g and $\nabla^2 g$ are the gradient and the Hessian matrix of the second derivatives. Hence, Eq. (C.16) can be written as follows:

$$A_2 - B_2 = \sum_y p(y|D_t, z_{1:t}) \sum_z p(z|y, z_{1:t}, D_t) \Bigg(g(\boldsymbol{\mu}_z) + \frac{1}{2} tr\big(\nabla^2 g(\boldsymbol{\mu}_z) \frac{v_z}{v_z - 2} \boldsymbol{\Sigma}_z\big) \Bigg) \tag{C.18}$$

where v_z and $\boldsymbol{\Sigma}_z$ are the degree of freedom and the covariance matrix of the student's t-distribution's component determined by z. This approximation is known as the multivariate delta method for moments [159]. Finally, the loss function in

Eq. (C.4) can be written as follows:

$$L(z_{1:t}, \hat{z}_{1:t}) = \sum_{y} p(y|D_t, z_{1:t}) \Bigg[\log \frac{p(y|D_t, z_{1:t})}{p(y|D_t, \hat{z}_{1:t})} + \sum_{z} p(z|y, z_{1:t}, D_t) \Big(g(\boldsymbol{\mu}_z) + \frac{1}{2} tr\big(\nabla^2 g(\boldsymbol{\mu}_z) \frac{v_z}{v_z - 2} \boldsymbol{\Sigma}_z\big) \Big) \Bigg]. \tag{C.19}$$

The current risk in Eq. (C.2) can be easily computed by replacing the loss function with its solution in Eq. (C.19). Thus, the current expected risk in Eq. (C.3) can be computed by replacing the current risk with its solution. Hence, the discrepancy between the current risk and the current expected risk in Eq. (C.1) is solved.

C.2 Sampling precision hyper-parameters

Algorithm 4 Precision Parameters Sampling

1: $HypSamp(\{H_t^i, w_t^i\}_{i=1}^P, \{\alpha_2, \alpha_{1i}\}_{i=1}^{m_2} a, b)$
2: Sample a particle: $h \sim \sum_{i=1}^P w_t^{(i)} \delta(H_t - H_t^{(i)})$
3: Derive $\{m_2, m_{11}, ..., m_{1m_2}\}$ from h
4: Sample $\{\eta_2, \eta_{11}, ..., \eta_{1m_2}\}$ (Eq. (C.23))
5: Sample $\{\alpha_2^{new}, \alpha_{11}^{new}, ..., \alpha_{1m_2}^{new}\}$ (Eq. (C.22))
6: Return $\{\alpha_2^{new}, \alpha_{11}^{new}, ..., \alpha_{1m_2}^{new}\}$

Authors in [153] show that the precision parameter in a DP mixture model α is conditionally independent of the data given the number of distinct components m and the size of the data, $n = \boldsymbol{e}^T \boldsymbol{n} \boldsymbol{e}$. Let $\alpha \sim G(a, b)$, a gamma prior with shape a and scale b which are both fixed to 1. The posterior distribution of α can be written as follows:

$$p(\alpha|m, n) \propto p(\alpha) p(m|\alpha, n) \tag{C.20}$$

According to [160], the likelihood in Eq. (C.20) may be written as:

$$p(m|\alpha, n) = c_n(m) n! \alpha^k \frac{\Gamma(\alpha)}{\Gamma(\alpha + n)} \tag{C.21}$$

where $c_n(m) = p(m|\alpha = 1, n)$ does not involve α. In this case, Eq. (C.20) can be expressed as mixture of two gamma posteriors [153].

$$(\alpha|\eta, m) \sim \pi_\eta G(a + m, b - \log(\eta)) + (1 - \pi_\eta)G(a + m - 1, b - \log(\eta)) \quad \text{(C.22)}$$

$$(\eta|\alpha, m) \sim B(\alpha + 1, n) \quad \text{(C.23)}$$

$$\frac{\pi_\eta}{1 - \pi_\eta} = \frac{a + m - 1}{n(b - \log(\eta))}$$

where B denotes the beta distribution, η is an auxiliary variable used in the sampling. To infer the distribution over α, Gibbs sampling iterations go as follows; First η is sampled from Eq. (C.23) conditional on the most recent value m and α. Second, α is sampled from Eq. (C.22) conditional on the already sampled η and the same m. The number of components can be deduced from the configuration variables.

To apply this sampling approach online on OSC model, we must sample for each class-specific mixture model, c, its corresponding, precision parameter α_{1c}, number of components m_{1c} and auxiliary variable η_{1c}. The set of parameters related to the class distribution, α_2, m_2 and η_2 must be sampled too. Furthermore, sampling must be done online. That is, once the data is processed, it is discarded. Thus, unlike offline Gibbs sampling, one set of precision parameters $\{\alpha_2^{new}, \alpha_{11}^{new}, ..., \alpha_{1m_2}^{new}\}$ is sampled in each iteration. The precision parameters are independent given the class label. Hence, they can be sampled independently using the same sampling routine described in [153].

Bibliography

[1] Indre Zliobaite, Albert Bifet, Bernhard Pfahringer, and Geoffrey Holmes. Active learning with drifting streaming data. *IEEE Transactions on Neural Networks and Learning Systems*, 2014.

[2] Chen Change Loy, Tao Xiang, and Shaogang Gong. Stream-based active unusual event detection. In *Asian Conference on Computer Vision*. Springer, 2010.

[3] Chen Change Loy, Timothy M Hospedales, Tao Xiang, and Shaogang Gong. Stream-based joint exploration-exploitation active learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[4] John Gantz and David Reinsel. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. *IDC iView: IDC Analyze the Future*, 2012.

[5] Michael Jordan. statistics and machine learning. URL https://www.reddit.com/r/MachineLearning/comments/2fxi6v.

[6] Simon Tong and Daphne Koller. Active learning for parameter estimation in Bayesian networks. In *NIPS*, 2000.

[7] Yi Yang, Shimei Pan, Jie Lu, Mercan Topkara, and Doug Downey. Incorporating user input with topic modeling. In *CIKM Workshop on Interactive Mining for Big Data (ImBig14)*, 2014.

[8] Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.

[9] Shai Shalev-Shwartz and Yoram Singer. Online learning: Theory, algorithms, and applications. 2007.

[10] Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *Proceedings of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001.

[11] Charu Aggarwal. *Data streams: models and algorithms*. Springer Science & Business Media, 2007.

[12] Mohamed Medhat Gaber, Arkady Zaslavsky, and Shonali Krishnaswamy. Mining data streams: a review. *ACM Sigmod Record*, 2005.

[13] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000.

[14] David Rosgen et al. A stream classification system. In *Riparian Ecosystems and Their Management. First North American Riparian Conference. Rocky Mountain Forest and Range Experiment Station, RM-120*, 1985.

[15] Léon Bottou. Online learning and stochastic approximations. *On-line Learning in Neural Networks*, 1998.

[16] Robi Polikar, Lalita Upda, Satish Upda, and Vasant Honavar. Learn++: An incremental learning algorithm for supervised neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, part C (applications and reviews)*, 2001.

[17] Gert Cauwenberghs and Tomaso Poggio. Incremental and decremental support vector machine learning. In *NIPS*, 2000.

[18] Gail Carpenter, Stephen Grossberg, Natalya Markuzon, John Reynolds, and David Rosen. Fuzzy artmap: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks*, 1992.

[19] Shai Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 2012.

[20] Joao Gama. *Knowledge discovery from data streams*. CRC Press, 2010.

[21] Joao Gama, Indre Zliobaite, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, 2014.

[22] Edwin Lughofer and Plamen Angelov. Handling drifts and shifts in on-line data streams with evolving fuzzy systems. *Applied Soft Computing*, 2011.

[23] Shanmugavelayutham Muthukrishnan et al. Data streams: Algorithms and applications. *Foundations and Trends in Theoretical Computer Science*, 2005.

[24] Albert Bifet and Ricard Gavalda. Adaptive learning from evolving data streams. In *International Symposium on Intelligent Data Analysis*. Springer, 2009.

[25] Gregory Ditzler, Manuel Roveri, Cesare Alippi, and Robi Polikar. Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, 2015.

[26] Antti Honkela and Harri Valpola. On-line variational bayesian learning. In *4th International Symposium on Independent Component Analysis and Blind Signal Separation*, 2003.

[27] Tamara Broderick, Nicholas Boyd, Andre Wibisono, Ashia Wilson, and Michael Jordan. Streaming variational bayes. In *Advances in Neural Information Processing Systems*, 2013.

[28] William Penny and Stephen Roberts. Dynamic logistic regression. In *International Joint Conference on Neural Networks*, 1999.

[29] Carlos Carvalho, Hedibert Lopes, Nicholas Polson, Matt Taddy, et al. Particle learning for general mixtures. *Bayesian Analysis*, 2010.

[30] Joao Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In *Brazilian Symposium on Artificial Intelligence*. Springer, 2004.

[31] Mohammad Masud, Qing Chen, Latifur Khan, Charu Aggarwal, Jing Gao, Jiawei Han, and Bhavani Thuraisingham. Addressing concept-evolution in concept-drifting data streams. In *IEEE 10th International Conference on Data Mining*. IEEE, 2010.

[32] Plamen Angelov and Xiaowei Zhou. Evolving fuzzy-rule-based classifiers from data streams. *IEEE Transactions on Fuzzy Systems*, 2008.

[33] Plamen Angelov. An approach for fuzzy rule-base adaptation using on-line clustering. *International Journal of Approximate Reasoning*, 2004.

[34] Shmuel Agmon. The relaxation method for linear inequalities. *Canadian Journal of Mathematics*, 1954.

[35] Albert Novikoff. On convergence proofs for perceptrons. Technical report, DTIC Document, 1963.

[36] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 1958.

[37] Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 2010.

[38] Eric Baum and Kenneth Lang. Query learning can work poorly when a human oracle is used. In *International Joint Conference on Neural Networks*, 1992.

[39] David Lewis and William Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Springer-Verlag New York, Inc., 1994.

[40] Aron Culotta and Andrew McCallum. Reducing labeling effort for structured prediction tasks. In *AAAI*, 2005.

[41] David Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, 1994.

[42] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active hidden markov models for information extraction. In *Advances in Intelligent Data Analysis*. Springer, 2001.

[43] Simon Tong and Edward Chang. Support vector machine active learning for image retrieval. In *Proceedings of the Ninth ACM International Conference on Multimedia*. ACM, 2001.

[44] Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2008.

[45] Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. ACM, 1992.

[46] Ido Dagan and Sean Engelson. Committee-based sampling for training probabilistic classifiers. In *Proceedings of the Twelfth International Conference on Machine Learning*, 1995.

[47] Prem Melville and Raymond Mooney. Diverse ensembles for active learning. In *Proceedings of the twenty-first International Conference on Machine learning*. ACM, 2004.

[48] Naoki Abe Hiroshi Mamitsuka. Query learning strategies using boosting and bagging. In *Proceedings of the Fifteenth International Conference Machine Learning*. Morgan Kaufmann Pub, 1998.

[49] Hieu Nguyen and Arnold Smeulders. Active learning using pre-clustering. In *Proceedings of the twenty-first International Conference on Machine Learning*. ACM, 2004.

[50] Kamal Nigam and Andrew McCallum. Employing em in pool-based active learning for text classification. In *Proceedings of of the Fifteenth International Conference on Machine Learning*, 1998.

[51] Yoram Baram, Ran El-Yaniv, and Kobi Luz. Online choice of active learning algorithms. *The Journal of Machine Learning Research*, 2004.

[52] Leslie Pack Kaelbling, Michael Littman, and Andrew Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 1996.

[53] Simon Tong. *Active learning: theory and applications*. PhD thesis, Citeseer, 2001.

[54] Ravi Ganti and Alexander Gray. Building bridges: Viewing active learning from the multi-armed bandit lens. *arXiv preprint arXiv:1309.6830*, 2013.

[55] Djallel Bouneffouf, Romain Laroche, Tanguy Urvoy, Raphael Féraud, and Robin Allesiardo. Contextual bandit for active learning: Active thompson sampling. In *Neural Information Processing*. Springer, 2014.

[56] Dwi Widyantoro and John Yen. Relevant data expansion for learning concept drift from sparsely labeled data. *IEEE Transactions on Knowledge and Data Engineering*, 2005.

[57] Mohammad Masud, Clay Woolam, Jing Gao, Latifur Khan, Jiawei Han, Kevin Hamlen, and Nikunj Oza. Facing the reality of data stream classification: coping with scarcity of labeled data. *Knowledge and Information Systems*, 2012.

[58] Patrick Lindstrom, Sarah Jane Delany, and Brian Mac Namee. Handling concept drift in text data stream constrained by high labelling cost. 2010.

[59] Jesse Read, Albert Bifet, Bernhard Pfahringer, and Geoff Holmes. Batch-incremental versus instance-incremental learning in dynamic and evolving data. In *Advances in Intelligent Data Analysis XI*. Springer, 2012.

[60] Josh Attenberg and Foster Provost. Online active inference and learning. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2011.

[61] Sculley. Online active learning methods for fast label-efficient spam filtering. In *CEAS*, 2007.

[62] Wei Chu, Martin Zinkevich, Lihong Li, Achint Thomas, and Belle Tseng. Unbiased online active learning in data streams. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2011.

[63] Sanjoy Dasgupta. Two faces of active learning. *Theoretical Computer Science*, 2011.

[64] Jim Pitman and Marc Yor. The two-parameter poisson-dirichlet distribution derived from a stable subordinator. *The Annals of Probability*, 1997.

[65] Mohammad Masud, Jing Gao, Latifur Khan, Jiawei Han, and Bhavani Thuraisingham. Classification and novel class detection in data streams with active mining. In *Advances in Knowledge Discovery and Data Mining*. Springer, 2010.

[66] Alina Beygelzimer, Sanjoy Dasgupta, and John Langford. Importance weighted active learning. In *Proceedings of the 26th annual International Conference on Machine Learning*. ACM, 2009.

[67] Manuel Baena-Garcıa, José del Campo-Ávila, Raúl Fidalgo, Albert Bifet, Gavalda, and Morales-Bueno. Early drift detection method. In *Fourth International Workshop on Knowledge Discovery from Data Streams*, 2006.

[68] Qing Da, Yang Yu, and Zhi-Hua Zhou. Learning with augmented class by exploiting unlabeled data. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. AAAI Press, 2014.

[69] Abdelhamid Bouchachia and Charlie Vanaret. GT2FC: An online growing interval type-2 self-learning fuzzy classifier. *IEEE Transactions on Fuzzy Systems*, 2014.

[70] Abdelhamid Bouchachia. Incremental learning with multi-level adaptation. *Neurocomputing*, 2011.

[71] Abdelhamid Bouchachia. On the scarcity of labeled data. In *Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce*. IEEE, 2005.

[72] Zhao Xu, Kai Yu, Volker Tresp, Xiaowei Xu, and Jizhi Wang. *Representative sampling for text classification using support vector machines*. Springer, 2003.

[73] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2002.

[74] Pinar Donmez, Jaime Carbonell, and Paul Bennett. Dual strategy active learning. In *European Conference on Machine Learning*. Springer, 2007.

[75] Alina Beygelzimer, Sanjoy Dasgupta, and John Langford. Importance weighted active learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009.

[76] David Cohn, Zoubin Ghahramani, and Michael Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 1996.

[77] Christopher Bishop et al. Neural networks for pattern recognition. Oxford: Clarendon Press, 1995.

[78] George Box and George Tiao. *Bayesian inference in statistical analysis*. John Wiley & Sons, 2011.

[79] Arthur Dempster, Nan Laird, and Donald Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (methodological)*, 1977.

[80] Jeffrey Banfield and Adrian Raftery. Model-based gaussian and non-gaussian clustering. *Biometrics*, 1993.

[81] Chris Stauffer and Eric Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000.

[82] Kevin Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[83] David Spiegelhalter and Steffen Lauritzen. Sequential updating of conditional probabilities on directed graphical structures. *Networks*, 1990.

[84] de Freitas, Mahesan Niranjan, and Andrew Gee. Hierarchical bayesian models for regularization in sequential learning. *Neural computation*, 2000.

[85] Joao Gama, Raquel Sebastiao, and Pedro Pereira Rodrigues. On evaluating stream learning algorithms. *Machine Learning*, 2013.

[86] Jun Liu. Monte carlo strategies in scientific computing. 2001. *NY: Springer*.

[87] Anand Narasimhamurthy and Ludmila Kuncheva. A framework for generating data to simulate changing environments. In *Artificial Intelligence and Applications*, 2007.

[88] Michael Bonnell Harries, Claude Sammut, and Kim Horn. Extracting hidden context. *Machine Learning*, 1998.

[89] Elena Ikonomovska, Joao Gama, and Saso Dzeroski. Learning model trees from evolving data streams. *Data Mining and Knowledge Discovery*, 2011.

[90] Houari Toubakh and Moamar Sayed-Mouchaweh. Hybrid dynamic classifier for drift-like fault diagnosis in a class of hybrid dynamic systems: Application to wind turbine converters. *Neurocomputing*, 2015.

[91] Yu Sun, Ke Tang, Leandro Minku, Shuo Wang, and Xin Yao. Online ensemble learning of data streams with gradually evolved classes. *IEEE Transactions on Knowledge and Data Engineering*, 2016.

[92] Mohammad Masud, Jing Gao, Latifur Khan, Jiawei Han, and Bhavani Thuraisingham. Classification and novel class detection in concept-drifting data streams under time constraints. *IEEE Transactions on Knowledge and Data Engineering*, 2011.

[93] Tahseen Al-Khateeb, Mohammad Masud, Latifur Khan, Charu Aggarwal, Jiawei Han, and Bhavani Thuraisingham. Stream classification with recurring and novel class detection using class-based ensemble. In *IEEE 12th International Conference on Data Mining*. IEEE, 2012.

[94] Marco AF Pimentel, David Clifton, Lei Clifton, and Lionel Tarassenko. A review of novelty detection. *Signal Processing*, 2014.

[95] Ryan Prescott Adams and David MacKay. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*, 2007.

[96] Nicholas Roy and Andrew McCallum. Toward optimal active learning through monte carlo estimation of error reduction. *ICML, Williamstown*, 2001.

[97] Sudheendra Vijayanarasimhan and Kristen Grauman. Multi-level active prediction of useful image annotations for recognition. In *Advances in Neural Information Processing Systems*, 2009.

[98] Yee Whye Teh. Dirichlet process. In *Encyclopedia of Machine Learning*. Springer, 2011.

[99] Jayaram Sethuraman. A constructive definition of Dirichlet priors. *Statistica Sinica*, 1994.

[100] Paul Fearnhead. Particle filters for mixture models with an unknown number of components. *Statistics and Computing*, 2004.

[101] Jayaram Sethuraman. A constructive definition of dirichlet priors. Technical report, DTIC Document, 1991.

[102] Radford Neal. Bayesian mixture modeling. In *Maximum Entropy and Bayesian Methods*. Springer, 1992.

[103] Carl Edward Rasmussen. The infinite gaussian mixture model. In *NIPS*, 1999.

[104] Thomas Ferguson. A bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1973.

[105] David Blackwell and James MacQueen. Ferguson distributions via pólya urn schemes. *The Annals of Statistics*, 1973.

[106] Yee Whye Teh. Dirichlet process. In *Encyclopedia of Machine Learning*. Springer, 2010.

[107] Radford Neal. Markov chain sampling methods for dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 2000.

[108] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. Moa: Massive online analysis. *The Journal of Machine Learning Research*, 2010.

[109] Arthur Asuncion and David Newman. UCI machine learning repository, 2007.

[110] Timothy Hospedales, Shaogang Gong, and Tao Xiang. Finding rare classes: Active learning with generative and discriminative models. *IEEE Transactions on Knowledge and Data Engineering*, 2013.

[111] Brent Longstaff, Sasank Reddy, and Deborah Estrin. Improving activity classification for health applications on mobile devices using active and semi-supervised learning. In *2010 4th International Conference on Pervasive Computing Technologies for Healthcare*. IEEE, 2010.

[112] Jun Yang. Toward physical activity diary: motion recognition using simple acceleration features with mobile phones. In *Proceedings of the 1st International Workshop on Interactive Multimedia for Consumer Electronics*. ACM, 2009.

[113] Geetika Singla, Diane Cook, and Maureen Schmitter-Edgecombe. Recognizing independent and joint activities among multiple residents in smart environments. *Journal of Ambient Intelligence and Humanized Computing*, 2010.

[114] Diane Cook and Maureen Schmitter-Edgecombe. Assessing the quality of activities in a smart environment. *Methods of Information in Medicine*, 2009.

[115] Abdelhamid Bouchachia. An evolving classification cascade with self-learning. *Evolving Systems*, 2010.

[116] Abdelhamid Bouchachia. Fuzzy classification in dynamic environments. *Soft Computing*, 2011.

[117] Ronald Poppe. A survey on vision-based human action recognition. *Image and Vision Computing*, 2010.

[118] Thomas Moeslund, Adrian Hilton, and Volker Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 2006.

[119] Daniel Weinland, Remi Ronfard, and Edmond Boyer. A survey of vision-based methods for action representation, segmentation and recognition. *Computer Vision and Image Understanding*, 2011.

[120] Andreas Bulling, Ulf Blanke, and Bernt Schiele. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys (CSUR)*, 2014.

[121] Shuangquan Wang and Gang Zhou. A review on radio based activity recognition. *Digital Communications and Networks*, 2015.

[122] Li Deng. A tutorial survey of architectures, algorithms, and applications for deep learning. *APSIPA Transactions on Signal and Information Processing*, 2014.

[123] Yoshua Bengio. Learning deep architectures for ai. *Foundations and Trends in Machine Learning*, 2009.

[124] Graham Taylor, Geoffrey Hinton, and Sam Roweis. Modeling human motion using binary latent variables. In *Advances in Neural Information Processing Systems*, 2006.

[125] Matthew Zeiler, Graham Taylor, Nikolaus Troje, and Geoffrey Hinton. Modeling pigeon behavior using a Conditional Restricted Boltzmann Machine. In *ESANN*, 2009.

[126] Abdel-rahman Mohamed and Geoffrey Hinton. Phone recognition using Restricted Boltzmann Machines. In *International Conference on Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE*. IEEE, 2010.

[127] Zhizheng Wu, Eng Siong Chng, and Haizhou Li. Conditional Restricted Boltzmann Machines for voice conversion. In *IEEE China Summit International Conference on Signal and Information Processing (ChinaSIP)*. IEEE, 2013.

[128] Graham Taylor, Leonid Sigal, David Fleet, and Geoffrey Hinton. Dynamical binary latent variable models for 3d human pose tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2010.

[129] Volodymyr Mnih, Hugo Larochelle, and Geoffrey Hinton. Conditional Restricted Boltzmann Machines for structured output prediction. *arXiv preprint arXiv:1202.3748*, 2012.

[130] Jiankou Li and Wei Zhang. Conditional Restricted Boltzmann Machines for cold start recommendations. *arXiv preprint arXiv:1408.0096*, 2014.

[131] Liming Chen, Jesse Hoey, Chris Nugent, Diane Cook, and Zhiwen Yu. Sensor-based activity recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 2012.

[132] Oscar Lara and Miguel Labrador. A survey on human activity recognition using wearable sensors. *IEEE Communications Surveys & Tutorials*, 2013.

[133] Kerem Altun and Billur Barshan. Human activity recognition using inertial/magnetic sensor units. In *International Workshop on Human Behavior Understanding*. Springer, 2010.

[134] Martin Berchtold, Matthias Budde, Dawud Gordon, Hedda Schmidtke, and Michael Beigl. Actiserv: Activity recognition service for mobile phones. In *International Symposium on Wearable Computers (ISWC) 2010*. IEEE, 2010.

[135] Jennifer Kwapisz, Gary Weiss, and Samuel Moore. Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter*, 2011.

[136] Wenyao Xu, Mi Zhang, Alexander Sawchuk, and Majid Sarrafzadeh. Robust human activity and sensor location corecognition via sparse signal representation. *IEEE Transactions on Biomedical Engineering*, 2012.

[137] Cagatay Catal, Selin Tufekci, Elif Pirmit, and Guner Kocabag. On the use of ensemble of classifiers for accelerometer-based activity recognition. *Applied Soft Computing*, 2015.

[138] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 2015.

[139] Jian Bo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiao Li Li, and Shonali Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI), Buenos Aires, Argentina*, 2015.

[140] Yong Du, Wei Wang, and Liang Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[141] Natalia Neverova, Christian Wolf, Griffin Lacey, Lex Fridman, Deepak Chandra, Brandon Barbello, and Graham Taylor. Learning human identity from motion patterns. *IEEE Access*, 2016.

[142] Nils Hammerla, Shane Halloran, and Thomas Ploetz. Deep, convolutional, and recurrent models for human activity recognition using wearables. *arXiv preprint arXiv:1604.08880*, 2016.

[143] Thomas Plötz, Nils Y Hammerla, and Patrick Olivier. Feature learning for activity recognition in ubiquitous computing. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, 2011.

[144] Zahraa Said Abdallah, Mohamed Medhat Gaber, Bala Srinivasan, and Shonali Krishnaswamy. Adaptive mobile activity recognition system with evolving data streams. *Neurocomputing*, 2015.

[145] Niall Twomey, Tom Diethe, and Peter Flach. Bayesian active learning with evidence-based instance selection. In *Workshop on Learning over Multiple Contexts, European Conference on Machine Learning (ECML15)*, 2015.

[146] Ruslan Salakhutdinov, Joshua Tenenbaum, and Antonio Torralba. Learning with hierarchical-deep models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.

[147] Geoffrey Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 2006.

[148] Saad Mohamad, Abdelhamid Bouchachia, and Moamar Sayed-Mouchaweh. A bi-criteria active learning algorithm for dynamic data streams. *IEEE Transactions on Neural Networks and Learning Systems*, 2016.

[149] Saad Mohamad, Moamar Sayed-Mouchaweh, and Abdelhamid Bouchachia. Active learning for classifying data streams with unknown number of classes. *Neural Networks (accepted with minor revision, 2017)*, .

[150] David MacKay. Information-based objective functions for active data selection. *Neural Computation*, 1992.

[151] Yoav Freund, Sebastian Seung, Eli Shamir, and Naftali Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 1997.

[152] Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

[153] Mike West. *Hyperparameter estimation in Dirichlet process mixture models.* Duke University ISDS Discussion Paper# 92-A03, 1992.

[154] Ricardo Chavarriaga, Hesam Sagha, Alberto Calatroni, Sundara Tejaswi Digumarti, Gerhard Tröster, José del Millán, and Daniel Roggen. The opportunity challenge: A benchmark database for on-body sensor-based activity recognition. *Pattern Recognition Letters*, 2013.

[155] Saad Mohamad, Abdelhamid Bouchachia, and Moamar Sayed-Mouchaweh. A non-parametric hierarchical clustering model. In *International Conference on Evolving and Adaptive Intelligent Systems (EAIS), year=2015, organization=IEEE*, .

[156] Jon McAuliffe, David Blei, and Michael Jordan. Nonparametric empirical bayes for the dirichlet process mixture model. *Statistics and Computing*, 2006.

[157] Ian Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, 2005.

[158] Saad Mohamad, Abdelhamid Bouchachia, and Moamar Sayed-Mouchaweh. Asynchronous stochastic variational inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence (Under review, 2017)*, .

[159] Peter Bickel and Kjell Doksum. *Mathematical statistics: basic ideas and selected topics*. CRC Press, 2015.

[160] Charles Antoniak. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The Annals of Statistics*, 1974.