Aprendizado Ativo para Classificadores de Fluxo de Dados Baseados em Agrupamento

Douglas Monteiro Cavalcanti

Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2021

Douglas Monteiro Cavalcanti

Aprendizado Ativo para Classificadores de Fluxo de Dados Baseados em Agrupamento

Dissertação de mestrado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientador: Elaine Ribeiro de Faria
Coorientador: Ricardo Cerri

Uberlândia
2021

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

C376
2021

Cavalcanti, Douglas Monteiro, 1997-
Aprendizado ativo para classificadores de fluxo de dados baseados em agrupamento [recurso eletrônico] / Douglas Monteiro Cavalcanti. - 2021.

Orientadora: Elaine Ribeiro de Faria.
Coorientador: Ricardo Cerri.
Dissertação (Mestrado) - Universidade Federal de Uberlândia, Pós-graduação em Ciência da Computação.
Modo de acesso: Internet.
Disponível em: http://doi.org/10.14393/ufu.di.2021.673
Inclui bibliografia.
Inclui ilustrações.

1. Computação. I. Faria, Elaine Ribeiro de,1980-, (Orient.). II. Cerri, Ricardo,1981-, (Coorient.). III. Universidade Federal de Uberlândia. Pós-graduação em Ciência da Computação. IV. Título.

CDU: 681.3

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:

Gizele Cristine Nunes do Couto - CRB6/2091

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Coordenação do Programa de Pós-Graduação em Ciência da Computação
Av. João Naves de Ávila, nº 2121, Bloco 1A, Sala 243 - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4470 - www.ppgco.facom.ufu.br - cpgfacom@ufu.br

ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Ciência da Computação				
Defesa de:	Mestrado Acadêmico, 27/2021, PPGCO				
Data:	25 de novembro de 2021	Hora de início:	14:00	Hora de encerramento:	16:20
Matrícula do Discente:	11912CCP006				
Nome do Discente	Douglas Monteiro Cavalcanti				
Título do Trabalho:	Active Learning Framework for Clustering-based Data Stream Classifiers				
Área de concentração:	Ciência da Computação				
Linha de pesquisa:	Inteligência Artificial				
Projeto de Pesquisa de vinculação:	-				

Reuniu-se, por videoconferência, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Ciência da Computação, assim composta: Professores Doutores: Fabíola Souza Fernandes Pereira - FACOM/UFU; Ronaldo Cristiano Prati - UFABC; Ricardo Cerri - UFSCAR (coorientador) e Elaine Ribeiro de Faria Paiva - FACOM/UFU, orientadora do candidato.

Os examinadores participaram desde as seguintes localidades: Ronaldo Cristiano Prati - Santo André/SP; Ricardo Cerri - São Carlos/SP; Fabíola Souza Fernandes Pereira e Elaine Ribeiro de Faria Paiva - Uberlândia/MG. O discente participou da cidade de Uberlândia/MG.

Iniciando os trabalhos a presidente da mesa, Prof.ª Dr.ª Elaine Ribeiro de Faria Paiva, apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público, e concedeu ao Discente a palavra para a exposição do seu trabalho. A duração da apresentação do Discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir a senhora presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir o candidato. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o candidato:

Aprovado.

Esta defesa faz parte dos requisitos necessários à obtenção do título de Mestre.

O competente diploma será expedido após cumprimento dos demais requisitos, conforme as normas do Programa, a legislação pertinente e a regulamentação interna da UFU.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.

Documento assinado eletronicamente por **Fabíola Souza Fernandes Pereira**, **Professor(a) do Magistério Superior**, em 26/11/2021, às 09:16, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Elaine Ribeiro de Faria Paiva**, **Professor(a) do Magistério Superior**, em 26/11/2021, às 09:23, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Ricardo Cerri**, **Usuário Externo**, em 26/11/2021, às 16:17, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **3198395** e o código CRC **D40BE9B0**.

Referência: Processo nº 23117.080222/2021-29 SEI nº 3198395

Agradecimentos

Agradeço primeiramente à minha Orientadora Prof.ª Dr.ª Elaine Ribeiro de Faria e ao meu Co-orientador Prof. Dr. Ricardo Cerri, que me ajudaram a formular as ideias por trás deste trabalho. Agradeço ao Programa de Pós-Graduação em Ciência da Computação da UFU pela oportunidade de participar do curso. Agradeço também aos meus amigos Tiago e Junício, que sempre acompanharam com otimismo o progresso deste trabalho.

"I may be wrong and you may be right, and by an effort,
we may get nearer to the truth."
(Karl Popper)

Resumo

O processo de atualização de classificadores de fluxo de dados baseados em agrupamento gera grupos a partir de instâncias de dados parcial ou totalmente não rotuladas. Cada grupo é então categorizado como a extensão de uma classe conhecida ou como o surgimento de uma nova, resumido e finalmente adicionado ao modelo de classificação. Considerando o custo de aquisição do rótulo, quando comparadas a abordagens exclusivamente supervisionadas, as estratégias baseadas em agrupamento apresentam a vantagem de permitir o uso de dados não rotulados para atualização do modelo de classificação. No entanto, o ganho de informações sobre a distribuição das classes de dados por meio de dados não rotulados está sujeito a suposições de como a distribuição dos atributos interage com a distribuição das classes de dados. Por causa disso, o processo de atualização de classificadores de fluxo de dados baseados em agrupamento está sujeito a falhar à medida que essa interação muda inesperadamente devido a característica não-estacionária do fluxo, levando a erros de inferência de classe e, consequentemente, à categorização incorreta de grupos, comprometendo a consistência do modelo de classificação. Considerando este problema, neste trabalho, propomos uma estratégia de aprendizagem ativa que seleciona os grupos para os quais a categorização é mais incerta e então, para cada grupo escolhido, consulta pelo rótulo das instâncias mais informativas no contexto da distribuição interna do grupo. Ao dividir a responsabilidade da consulta de aprendizagem ativa entre duas estratégias de consulta, uma para o nível dos grupos e outra para o nível das instâncias, a estratégia garante um uso eficiente e eficaz dos recursos de rótulo, adquirindo rótulos apenas para grupos com maior probabilidade de precisar deles. Para testar a estratégia de aprendizagem ativa proposta, ela foi aplicada a dois classificadores de fluxo de dados baseados em clustering da literatura: MINAS e ECHO. Nos resultados, a estratégia de aprendizagem ativa recuperou um número significativo de categorizações incorretas de cluster ao custo de poucas aquisições adicionais de rótulo.

Palavras-chave: Fluxo de Dados, Aprendizado Ativo, Agrupamento, Semi-supervisão.

Active Learning for Clustering-based Data Stream Classifiers

Douglas Monteiro Cavalcanti

Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2021

UNIVERSIDADE FEDERAL DE UBERLÂNDIA – UFU
FACULDADE DE COMPUTAÇÃO – FACOM
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO – PPGCO

The undersigned hereby certify they have read and recommend to the PPGCO for acceptance the dissertation entitled **"Active Learning for Clustering-based Data Stream Classifiers"** submitted by **Douglas Monteiro Cavalcanti** as part of the requirements for obtaining the **Master's degree in Computer Science**.

Uberlândia, ___ de _____ de _____

Supervisor: _____
Prof.ª Dr.ª Elaine Ribeiro de Faria
Universidade Federal de Uberlândia

Cosupervisor: _____
Prof. Dr. Ricardo Cerri
Universidade Federal de São Carlos

Examining Committee Members:

_____
Prof. Dr. Ronaldo Cristiano Prati
Universidade Federal do ABC

_____
Prof.ª Dr.ª Fabíola Souza Fernandes Pereira
Universidade Federal de Uberlândia

Abstract

The update process of clustering-based data stream classifiers generates clusters from partially or fully unlabeled data instances. Each cluster is then categorized as the extension of a known class or as the emergence of a new one, summarized, and finally added to the classification model. Considering the cost of label acquisition, when compared to exclusively supervised approaches, clustering-based strategies present the advantage of allowing the use of unlabeled data to update the classification model. However, the gain of information about the data classes' distribution through unlabeled data is subject to assumptions of how the distribution of the features interacts with the distribution of the data classes. Because of that, the updated process of clustering-based data stream classifiers is prone to fail as this interaction changes unexpectedly due to the stream's non-stationary characteristic, leading to class inference errors and consequently the miscategorization of clusters, compromising the consistency of the classification model. Considering this problem, in this work, we propose an active learning strategy that selects for the clusters for which the categorization is more uncertain and then, for each chosen cluster, queries for the label of the instances more informative in the context of the inner cluster distribution. By dividing the active learning query responsibility among two query strategies, one for the cluster-level and the other for the instance-level, the strategy guarantees an efficient and effective use of label resources by acquiring labels only for the clusters more likely to need it. To test the proposed active learning strategy, we applied it to two clustering-based data stream classifiers from the literature: MINAS and ECHO. In the results, the active learning strategy recovered a significant number of cluster miscategorizations at the cost of a few additional label acquisitions.

Keywords: Data Stream, Active Learning, Clustering, Semi-supervision.

List of Figures

List of Tables

Contents

I hereby certify that I have obtained all legal permissions from the owner(s) of each third-party copyrighted matter included in my thesis, and that their permissions allow availability such as being deposited in public digital libraries.

Douglas Monteiro Cavalcanti

Chapter 1

Introduction

Pattern Recognition concerns the discovery of regularities in data through computer algorithms and, ultimately, the use of those regularities to automate decision-making tasks (SVENSÉN; BISHOP, 2007). The process of classifier design is one of pattern recognition's primary goals (JAIN; DUIN; MAO, 2000). It aims to design a statistical model to automate a classification task that lacks a deterministic model (SVENSÉN; BISHOP, 2007; FUKUNAGA, 2013).

The process of classifier design involves using a computer algorithm to learn different classes of data from a set of samples (FUKUNAGA, 2013). This set is referred to as the training dataset. Each instance in the training dataset has its data class known in advance, typically by inspecting them individually and hand-labeling (SVENSÉN; BISHOP, 2007). The learning algorithm defines a classification model by generalizing the discriminant characteristics from the training data. In the application phase, the learned model is used to classify new data with some accuracy (SVENSÉN; BISHOP, 2007; FUKUNAGA, 2013).

From a statistical perspective, the learning or training of a classification model can be considered a problem of estimating the probability distribution of the data classes (JAIN; DUIN; MAO, 2000). In this context, the n attributes that characterize a data instance define a random vector, and a data class represents a distribution of this vector in the n-dimensional sample space. Thus, by estimating the distribution of each data class using the labeled data instances, the learning algorithm can define a discriminant function that divides the sample space into the regions of the data classes (FUKUNAGA, 2013).

The most traditional approach for classifier design supposes that the data classes distribution is stationary, and for that reason, it only needs to be estimated once (SVENSÉN; BISHOP, 2007; GAMA, 2010). In this setting, after processing the training dataset, the learning algorithm outputs a static classifier, i.e., a classifier whose discriminant function cannot be changed once defined. This approach is known as batch learning and has been the main focus of classifier design research and practice in the last decades (GAMA, 2010).

The assumption that the data distribution will not change holds significantly for many

real-world problems, as evidenced by the reported success of the practical application of many batch learning algorithms (SVENSÉN; BISHOP, 2007; GAMA, 2010). However, the recent advance in data generation has led to more dynamic classification problems, where the data distribution is non-stationary (GAMA, 2010).

In a non-stationary environment, given enough time for the distribution to change, a static classification model will get outdated and underperform (GAMA, 2010). Considering that training a static model usually takes a significant amount of time, building a new model may be unfeasible since the classification system would be offline during the period. Beyond that, it may be impossible for the new model to be available on time, as the distribution may suffer further changes during the model's training (GAMA, 2010; GAMA et al., 2014).

The issues faced by a traditional learning algorithm in non-stationary environments imply switching from batch to incremental learning algorithms (GAMA, 2010). Classification models based on incremental learning can incorporate new data by adjusting their discriminant function to changes in the data distribution. In this setting, the data is better described as a transient stream flowing at high speed, continuously generated by a non-stationary distribution (GAMA, 2010).

In a data stream environment, changes in the distribution of classes are referred to as concept drift (GAMA et al., 2014). Another phenomenon related to the non-stationarity of the data that may also be present in a data stream is the concept evolution, defined as the emergence of hitherto unknown data classes in the stream (FARIA et al., 2016a). A classification system for data stream must detect the occurrence of both phenomenons, so the learning algorithm can be called on time to update the classification model.

The training data used to detect changes in the distribution and update the classifier is collected over time during the entire classifier application lifetime (GAMA, 2010). The most traditional approaches usually suppose unrestricted access to label information (AGGARWAL et al., 2006; MASUD et al., 2010; LOSING; HAMMER; WERSING, 2016). Some more advanced ones selectively ask for the label of instances more likely to provide information about the changes in the data distribution (ZHU et al., 2007; CHU et al., 2011; LUGHOFER, 2012; ŽLIOBAITĖ et al., 2013; LUGHOFER; PRATAMA, 2017; MOHAMAD; SAYED-MOUCHAWEH; BOUCHACHIA, 2018; KSIENIEWICZ et al., 2019; PARREIRA; PRATI, 2019; ZYBLEWSKI; KSIENIEWICZ; WOŹNIAK, 2020). In any case, relying only on labeled data instances to update the classification model may not be suitable for real-world applications, where labeled data is usually scarce, and manual labeling is both costly and time-consuming (MASUD et al., 2008; KHEZRI et al., 2020). This problem motivates learning approaches to detect and incorporate distribution changes to the classification model by recognizing patterns in unlabeled or partially labeled data instances.

Clustering-based data stream classifiers (MASUD et al., 2008; FARIA et al., 2016b;

ABDALLAH et al., 2016; HAQUE; KHAN; BARON, 2016; HAQUE et al., 2016; GARCIA et al., 2019) are examples of methods that keep track of concept drift and evolution through unlabeled data. In these methods, the classification model is composed of a set of labeled cluster summaries, used to map the sample space into the regions of the data classes. A cluster refers to a group of samples expected to share the same data class. A cluster summary refers to a set of parameters calculated from a cluster by assuming a mathematical form for its inner distribution (AGGARWAL et al., 2003).

The update process of these classifiers consists of adding new cluster summaries to the model and removing old ones as new clusters of data are learned (MASUD et al., 2008; FARIA et al., 2016b; ABDALLAH et al., 2016; HAQUE; KHAN; BARON, 2016; HAQUE et al., 2016; GARCIA et al., 2019). The underlying learning algorithm builds new data clusters from unlabeled or partially labeled data collected from the stream. The clustering process is performed online or else over a small data buffer, so the time required for the algorithm to execute does not significantly delay the model update. Once a cluster is made available by the underlying clustering algorithm, an arbitrary decision rule infers its class. In general, if the cluster is composed of partially labeled data, the labeled portion of the data is used to infer the cluster class (MASUD et al., 2008; HAQUE; KHAN; BARON, 2016; HAQUE et al., 2016). If the cluster is fully unlabeled, it is categorized as the extension of a known class or the emergence of a new one by being compared to the cluster summaries kept in the model (FARIA et al., 2016b; ABDALLAH et al., 2016; GARCIA et al., 2019). Once assimilated to a class or category, the cluster is summarized and incorporated into the classification model.

1.1 Motivation

A necessary condition for learning from unlabeled or partially labeled data is that the distribution of the random vector representing the n attributes of the data instances must contain information about the probability distribution of the data classes (ENGELEN; HOOS, 2020). If the data meets this condition, a learning algorithm can use unlabeled samples to gain information about the distribution of the attributes and thereby about the distribution of the classes.

Assumptions on how the distribution of the data attributes interacts with the distribution of the data classes are called semi-supervised assumptions (ENGELEN; HOOS, 2020). The most common semi-supervised assumptions are the following:

- ❑ Smoothness assumption: if two data instances are close in the sample space, they should belong to the same data class;
- ❑ Low-density assumption: the decision boundary between two different data classes should not pass through high-density areas of the sample space;

- ❏ Clustering assumption: instances belonging to the same cluster belong to the same class, where a cluster is a set of instances that are more similar to each other than to other instances from the feature space.

The updated process of clustering-based data stream classifiers is subject to at least one of these assumptions (MASUD et al., 2008; FARIA et al., 2016b; ABDALLAH et al., 2016; HAQUE; KHAN; BARON, 2016; HAQUE et al., 2016; GARCIA et al., 2019). Because of that, the updated process of clustering-based data stream classifier is prone to fail as the semi-supervised assumption loses validity in the constantly changing data distribution.

Consider, for example, a situation where the smoothness assumption fails, and consequently, data instances of different classes end up close to each other in the sample space. A clustering-based data stream classifier whose update process involves using a distance-based clustering algorithm may end up adding unlabeled instances from different classes to the same cluster. In this case, even if the cluster is partially labeled, the impurity cannot be solved unless the labels contemplate the diverging data instances.

Now consider, for example, a situation where the low-density assumption fails in a way that a cluster of unlabeled data maps to a region of the sample space densely populated by clusters of a different class. Without prior information about the data class of the instances inside the cluster, the update process will assimilate the cluster to the class dominant in the region, misclassifying all the data instances within the cluster.

Since the update process of clustering-based data stream uses the obtained clusters to update the classification model, the misclassification of the clusters will lead to an inconsistent model, compromising the accuracy of the classifier in explaining the data stream instances. In this context, it is of great importance to ensure the correct categorization of the clusters even in scenarios where the semi-supervised assumptions fail, making it indispensable to rely on label information at some point.

Regarding the efficient obtaining of label information in data streams, recent studies propose active learning strategies (ZHU et al., 2007; CHU et al., 2011; LUGHOFER, 2012; ŽLIOBAITĖ et al., 2013; LUGHOFER; PRATAMA, 2017; MOHAMAD; SAYED-MOUCHAWEH; BOUCHACHIA, 2018; KSIENIEWICZ et al., 2019; PARREIRA; PRATI, 2019; ZYBLEWSKI; KSIENIEWICZ; WOŹNIAK, 2020). Active learning refers to a set of algorithms that evaluates the information gain of unlabeled data in the context of classifier learning (AGGARWAL et al., 2014; SETTLES, 2009). The objective of these strategies is to provide the best training dataset for a classifier given a limited labeling resource. Clustering-based data stream classifiers use active learning strategies to select and label the instances that provide more information about the distribution changes (HAQUE; KHAN; BARON, 2016; HAQUE et al., 2016). In this way, the clustering performs by also considering the class purity of the labeled instances within each cluster.

In the context of the fail of the semi-supervised assumptions, clustering-based data stream classifiers that utilize an active learning and a semi-supervised clustering algorithm may present an advantage over approaches that generate clusters over buffers of exclusively unlabeled data. However, since the approach labels the instances before the clustering process, there's no guarantee that, after the clustering process, all the clusters requiring label information will receive enough labeled instances.

In this context, although clustering-based data stream classifiers represent a promising approach for handling classification over non-stationary environments, further study may be required to develop an update process that can actively ask for label information in situations where the semi-supervised assumptions fail while avoiding it for cases where the assumptions hold.

1.2 Objectives

The general objective of this work is to propose an active learning strategy to help in the update process of clustering-based data stream classifiers. Considering the computational and memory restrictions of the data stream scenario, it is also a general objective that any procedure used by the proposed strategy presents low computational and memory complexities.

The specific objectives of this work are:

- ❏ To propose a strategy to detect risky cluster categorizations by the update process of a clustering-based data stream classifier;
- ❏ To propose a strategy to select which instances inside a cluster are more representative in terms of the cluster inner distribution;
- ❏ To integrate the strategies mentioned above in an active learning strategy that first queries for clusters more likely to be miscategorized, and then queries for the instances inside the cluster that are more likely to help in the cluster categorization;
- ❏ To implement the active learning strategy proposed for the specific problem of distinguishing clusters representing concept drift from clusters representing concept evolution in the update process of clustering-based data stream classifiers;
- ❏ To apply the active learning strategy for the distinction of concept drift and evolution to two clustering-based data stream classifiers, where one presents an update process based on a buffer of fully unlabeled data, and the other presents an update process based on a buffer of partially labeled data;

- ❏ To develop a strategy to evaluate the performance of the active learning strategy in providing additional label information for all and only the clusters that the update process is unable to categorize as concept drift or concept evolution correctly;
- ❏ To develop evaluation measures to compute the percentage of clusters miscategorizations avoided by a cluster-based data stream classifier by incorporating the active learning strategy.

1.3 Hypothesis

The hypothesis of this work are the following:

- ❏ Clusters more likely to be miscategorized by the update process of a clustering-based data stream classifier can be detected without impacting the computational complexity of the update process.
- ❏ Once a cluster that would be miscategorized by the update process of a clustering-based data stream classifier is detected, it can be correctly categorized by labeling only a small portion of the cluster's data instances.

1.4 Contributions

The main contributions of this work are the following:

- ❏ Definition of the two-level active learning strategy for clustering-based data stream classifiers;
- ❏ Implementation of an active learning framework based on a low computational cost query-by-committee for distinguishing concept drift and evolution in clustering-based data stream classifiers;
- ❏ Definition of a set of decision rules for categorizing a cluster as concept drift or evolution;
- ❏ Definition of an evaluation strategy to analyze the impact of integrating the framework into existing clustering-based data stream classifiers.

1.5 Thesis Organization

This work is organized as follows:

- ❏ Chapter 2 details the fundamental concepts related to this work;

- ❏ Chapter 3 details the proposal of this work;
- ❏ Chapter 4 presents the experiments made for this work;
- ❏ Chapter 5 concludes the work.

Chapter 2

Fundamentals

In this chapter, we present the main definitions required to describe the proposed approach. We start by explaining the definition of the Bayes classifier, Bayes error, and the Grouped Error Estimate (FUKUNAGA; HOSTETLER, 1975; FUKUNAGA; MANTOCK, 1982). Those concepts form the base of the proposed meta-categorizers (modules of the proposed framework for the distinction of concept drift and evolution, to be described in Chapter 3) and also provide a formalization about classification probability and classes density function, key definitions for this work. In sequence, we formalize concept drift and concept evolution over data streams. After that, we present the definition of active learning, query-by-committee, and multiple-instance active learning. Finally, we describe the main clustering-based data stream classifiers from the literature.

2.1 Bayes classifier, Bayes error and Grouped Error Estimate

The input data of a classifier is an instance of a random vector X (FUKUNAGA, 2013). Each random variable of X represents one of the D features of the objects of classification. Many instances of X represent a distribution of the vector in the D-dimensional feature space. The data classes are different distributions of X, and each one is characterized by its density function. The probability density function that characterizes the distribution of a class c_i in the feature space is called the conditional density of class i and is expressed as $p(X|c_i)$. The density function that considers all M classes in the feature space is called the unconditional density function of X, and is given by (FUKUNAGA; KESSELL, 1973; FUKUNAGA, 2013):

$$\sum_{j=1}^{M} p(X|c_j) \cdot p(c_j),$$

where $p(c_j)$ is the prior probability of class c_j.

If we consider that the prior probability and the conditional density of all classes are known, the real probability $p(c_i|x)$ of an instance of x belonging to a class c_i can be calculated by (FUKUNAGA, 2013):

$$p(c_i|x) = \frac{p(x|c_i) \cdot p(c_i)}{\sum_{j=1}^{M} p(x|c_j) \cdot p(c_j)} \tag{1}$$

A discriminant function calculated from an inequation of the real probabilities of the different classes defines the Bayes classifier (FUKUNAGA, 2013).

The classification risk of an instance x can be calculated through $r(x) = 1 - p(c|x)$, where c is the class with the highest probability for the instance. Taking the expectation of the risk function r with respect to the random vector X gives the Bayes error, expressed by R (FUKUNAGA, 2013). Since in the Bayes classifier the density function of all classes is known, the Bayes error is the minimum achievable error for any classifier. It is higher than zero whenever there is a non-zero probability of an instance belonging to two or more classes (FUKUNAGA, 2013; FUKUNAGA; KESSELL, 1973). For that reason, the Bayes error is an optimum measure of system performance, and classification information of subregions of the feature space (FUKUNAGA; KESSELL, 1973).

Unfortunately, for many classification problems, the density function of the classes is unknown, and therefore, the Bayes error cannot be calculated. In this sense, several suggestions have been proposed to provide a estimate of the Bayes error (FUKUNAGA; KESSELL, 1973). One of the most basics and yet relevant in the context of this work is the Grouped Error Estimate (FUKUNAGA; HOSTETLER, 1975; FUKUNAGA; MANTOCK, 1982), proposed to estimate the classification information of a group of unlabeled instances in scenarios where the instance labeling is costly (FUKUNAGA; HOSTETLER, 1975; FUKUNAGA; MANTOCK, 1982).

The Grouped Error Estimate, denoted by $\hat{R}$, is calculated over a set of n unlabeled instances X' by taking the mean of the classification risk of all instances $x \in X'$ (FUKUNAGA; HOSTETLER, 1975). In scenarios where $p(c|X')$ is unknown, a non-parametric approximation $\hat{r}(X')$ of $r(X')$ is used (FUKUNAGA; HOSTETLER, 1975; FUKUNAGA; MANTOCK, 1982), resulting in the following formula for estimating $\hat{R}$ (FUKUNAGA; HOSTETLER, 1975):

$$\hat{R} = \frac{\sum_{i=1}^{n} \hat{r}(x_i)}{n} \tag{2}$$

In this case, the randomness of $\hat{R}$ comes from two sources: one from the estimate $\hat{r}(X')$ of $r(X')$, and the other from X'(FUKUNAGA, 2013).

2.2 Concept drift and evolution over data streams

A data stream is an infinite sequence of instances $x_1, x_2, \ldots, x_{now}, \ldots$ in the D-dimensional feature space (GAMA, 2010; FARIA et al., 2016a). x_1 is the very first instance in the stream, and x_{now} is the last data instance that just arrived. Each data instance x_i is associated with an arrival timestamp t_i. For simplicity, we consider $t_{i+1} = t_i + 1$ and $t_1 = 1$. Each instance x_i is also associated with a data class c. In the context of this work, the class of an instance is considered to be unknown, but can be obtained by querying an external agent, human or not, at a specific labeling cost (MASUD et al., 2010).

For dynamically changing environments, the data stream distribution is typically non-stationary, which means that its underlying distribution can change dynamically over time. The phenomenon where the distribution of the data changes is called concept drift, and the general assumption is that a concept drift can occur unexpectedly and take different forms, changing the input data characteristics or the relationship between the input data and their respective classes (GAMA, 2010; GAMA et al., 2014).

The concept drift between two timestamps t_i and t_{i+k}, with $k > 0$, can be formally defined as (GAMA et al., 2014; LOSING; HAMMER; WERSING, 2016)

$$\exists X : p_{t_i}(X) \cdot p_{t_i}(c|X) \neq p_{t_i+k}(X) \cdot p_{t_i+k}(c|X), \tag{3}$$

where $p_t(X)$ is the distribution of features in the timestamp t, and $p_t(c|X)$ is the posterior probability of the class c in the timestamp t.

Considering that the data is continuously generated, another phenomenon usually present in problems involving data streams is the concept evolution (FARIA et al., 2016a), defined as the emergence of classes unknown by the learner, where instances from a yet unseen class appear in the stream (MASUD et al., 2010; FARIA et al., 2016a).

2.3 Active learning

Learning problems involving a classification task require a sufficient quantity of labeled data for training purposes. However, it is known that for most classification problems, not all instances from the feature space are equally important to achieve the most effective training (AGGARWAL et al., 2014; SETTLES, 2009). For example, instances that clearly belong to a specific class may be helpful, but less so than instances that lie in boundaries between different classes (AGGARWAL et al., 2014). This implies that, besides the quantity, the quality of the labeled instances may also be of relevance when building a training set. In this context, and considering that for many real-world problems, there's a cost for label acquisition, the active learning strategies are proposed to reduce the amount of labeling required for an effective learning (SETTLES, 2009).

Every active learning system is composed of two primary components, the query system, which is responsible for selecting for which instances to query the label, and the oracle, responsible for responding to the underlying query (SETTLES, 2009). For most active learning algorithms, the oracle is treated as a black box that is used directly (AGGARWAL et al., 2014). The active learning strategies for different scenarios diverge the most in the query system when defining the framework of how the query is posed to the learner (AGGARWAL et al., 2014).

For this work, the most relevant active learning related topics are the query-by-committee query strategy (SEUNG; OPPER; SOMPOLINSKY, 1992; FREUND et al., 1997) and multiple-instance active learning (SETTLES; CRAVEN; RAY, 2007), described in sequence.

2.3.1 Query-by-committee strategy

Query-by-committee (SEUNG; OPPER; SOMPOLINSKY, 1992) is a theoretical approach to instance selection. In this approach, an ensemble of different hypotheses is learned, and instances that cause maximum disagreement among the committee regarding the predicted categorization are selected as the most informative ones. The approach was first proposed for the incremental query learning paradigm (FREUND et al., 1997), with a scope restricted to parametric learning models.

The general query-by-committee approach can be defined as an iterative process, where at each iteration, a committee of classifiers is trained based on the current training set and used to select a set of unlabeled data with high expected informational value (FREUND et al., 1997; AGGARWAL et al., 2014). The instances are selected according to the principle of maximal disagreement, where the instances that maximize the disagreement among the committee, with respect to the predicted label, are considered the most valuable. The selected instances are added to the training set at the end of the iteration, and then the algorithm is run again on the newly incremented training set, and so on (SEUNG; OPPER; SOMPOLINSKY, 1992).

2.3.2 Multiple-instance active learning

The multiple-instance active learning is an approach for learning at mixed levels of granularity in problems where instances are naturally organized into bags, and the bags, instead of the individual instances, are labeled for training (AGGARWAL et al., 2014; SETTLES; CRAVEN; RAY, 2007). In the target problems, referred to as multiple-instance learning problems, it is possible to obtain labels at the bag level and directly at the instance-level. However, getting labels at the instance-level is rather expensive. For that reason, it is easier to label the whole bag and infer its label to all the instances inside it. However, there's no guarantee that all the instances inside a bag belong to the same

class. Because of that, to label at the higher level of granularity may lead to inference errors in the lower level of granularity. Labeing all the instances to avoid inference errors is not a good solution either, given the cost for labeling at the instance-level. In this sense, the multiple-instance active learning is proposed to reduce the burden of labeling at the instance-level, by getting labels at the bag level of granularity, but also benefit from selectively labeling some part of the instances inside the bag at the finer level of granularity (SETTLES; CRAVEN; RAY, 2007).

2.4 Clustering-based Data Stream Classifiers

This section summarizes the main types of cluster categorization methods in clustering-based data stream classifiers. The objective is to highlight these methods' theoretical limitations and point out how the proposed two-level active learning strategy can cover these limitations by asking for additional label information.

We divide the clustering-based data stream classifiers between the ones that use a semi-supervised clustering algorithm in their update process (MASUD et al., 2008; HAQUE; KHAN; BARON, 2016; HAQUE et al., 2016), and the ones based on unsupervised novelty detection algorithms (FARIA et al., 2016b; ABDALLAH et al., 2016; GARCIA et al., 2019). Both approaches use labeled data in the initial training of the classifier, so the distinction between semi-supervised and unsupervised refers only to the process of detecting changes in the data stream distribution.

In the semi-supervised scenario, the clustering is performed over a buffer of partially labeled data by minimizing the impurity among the labeled instances inside the same cluster as well as the dispersion of the cluster's instances. If the clustering algorithm fails at building a complete pure cluster in terms of labeled data, the approach assigns multiple labels to it, weighting each label by its frequency within the cluster (MASUD et al., 2008; HAQUE; KHAN; BARON, 2016; HAQUE et al., 2016).

In this scenario, the miscategorization of a cluster happens when the labeled portion of the data does not represent the actual cluster distribution. If the clustering assumption fails and instances from different classes get clustered together, the strategy can only address the problem if each class in the cluster has at least one of its instances labeled. Since the approach labels the instances before the clustering process, there's no guarantee that each cluster will contain enough labeled data to represent its inner distribution.

This scenario requires additional label information in the context of the clusters more likely to be misrepresented. In this sense, the two-level active learning strategy might be used to analyze each resulting cluster, querying the ones more likely to be misrepresented and then querying for the label of instances more informative in the context of the inner cluster distribution.

In the unsupervised scenario, a novelty detection procedure detects patterns in a

buffer of unlabeled instances. Some of those approaches can only detect one pattern at a time, by using not a clustering algorithm but unsupervised indicators that search for a single cohesive cluster in a buffer of unlabeled data (ABDALLAH et al., 2016). Other approaches apply an unsupervised clustering algorithm over the buffer to detect multiple patterns at once (FARIA et al., 2016b; GARCIA et al., 2019). Once a cluster is made available, the strategy categorizes the cluster by comparing it to summaries kept by the learning model, using an arbitrary similarity measure. If the cluster is similar enough to one of the labeled cluster summaries, it is categorized as an extension of such summary. If the cluster differs enough from the cluster summaries, the strategy categorizes it as the emergence of a new class.

Without the label of the instances inside the clusters, the unsupervised strategies depend fundamentally on the clustering, and low-density assumptions (ENGELEN; HOOS, 2020). Two main situations can occur when one of these assumption fails: (i) If the clustering assumption fails, it results in impure clusters; (ii) if the low-density fails, it results in clusters of different classes getting close to each other, or clusters of the same class being apart by low-density regions of the feature space. In any of these situations, the similarity between clusters alone may not be enough to ensure that two clusters belong to the same class. In this sense, the two-level active learning strategy might be used to analyze the resulting clusters, detecting clusters more likely to be miscategorized by the similarity measures and providing label information in the context of their inner distribution.

To evaluated the proposed active learning strategy, we selected two different clustering-based data stream classifiers from the literature, MINAS (FARIA et al., 2016b) and ECHO (HAQUE et al., 2016). MINAS relies exclusively on unsupervised novelty detection algorithms to adjust its learning model. ECHO, on the other hand, utilizes a semi-supervised clustering algorithm to adapt its learning model. Even though ECHO uses labeled data, it also keeps a buffer of unlabeled data only, composed of instances that the learning model could not explain. When this buffer reaches a specific size, an unsupervised novelty detection algorithm is applied over it to detect and declare the emergence of a new class. In sequence, we defined in more detail the original approach for MINAS and ECHO.

2.4.1 MINAS

Proposed in (FARIA et al., 2016b), MINAS is a data stream classifier that relies on an unsupervised novelty detection procedure to detect not just new classes in the stream but also changes in the concept of known classes. To do so, MINAS does not require true label information. Instead, it maintains a buffer of unlabeled instances that the classifier could not explain, and whenever the buffer reaches its maximum size, an unsupervised clustering algorithm is applied. The resulting clusters are then categorized as the emergence of a new class or the extension of a class already known.

MINAS maintains a classification model composed of micro-clusters. When a new instance arrives in the stream, the algorithm computes its distance to the centroids of the micro-clusters. If the distance to the closest micro-cluster is lower than a threshold, the algorithm classifies the instance with the class assigned to the micro-cluster. If the distance is higher than the threshold, the algorithm sends the instance to a buffer. When the buffer reaches its maximum size, MINAS executes the novelty detection procedure.

The novelty detection procedure applies an unsupervised clustering algorithm over the buffer of unlabeled data. Once the clusters are available, MINAS categorizes each one as concept drift or evolution by computing the distance between the cluster's centroid to the centroids of the micro-clusters from the classification model. If the cluster's centroid is close enough to a micro-cluster of a known class, the cluster is assigned to that class. Otherwise, the cluster is categorized as belonging to a concept evolution and enumerated as a new pattern. Once categorized, the cluster is summarized as a micro-cluster and added to the classification model.

2.4.2 ECHO

Proposed in (HAQUE et al., 2016), ECHO is a semi-supervised approach for non-stationary data stream classification. The classifier maintains an ensemble of k-NN like classification models and uses change detection on classifier confidence to detect concept drifts. New instances are classified and stored in a dynamic window, along with the predicted class and a value representing its classification confidence. The algorithm expects the classification confidence to reduce consistently whenever a concept drift occurs. In this sense, the strategy searches over the window for any significant change in the confidence distribution at each new confidence value stored. If the change is detected, the algorithm uses the data in the window to train a new model and update the ensemble.

ECHO uses an active learning strategy to select which instances in the training dataset query for the label. If the classification confidence of an instance is below a parametrized threshold, the strategy queries for the instance's label. Otherwise, the algorithm labels the instance with the class predicted by the classifier. Once every instance in the training dataset is associated with a class, a semi-supervised clustering algorithm called MCI-KMeans is applied over the training dataset. By relying on the class associated with each instance, the MCI-KMeans minimizes the impurity among the labeled instances inside the same cluster. The resulting clusters are summarized in a data structure called pseudpoint—a pseudopoint stores the cluster centroid, the class frequency among the instance, and the radius. The radius is computed by calculating the distance between the centroid and the farthest instance. The obtained set of pseudopoints composes a new model. New models are added to the ensemble by replacing the oldest model.

To classify an instance with a single model, the approach finds the nearest neighbor among the model's pseudopoints by computing the distance between the instance and

the centroid of each pseudopoint. If the distance between the instance and the closest pseudopoint is lower than the pseudopoint's radius, the instance is classified by the class with high frequency in the pseudopoint. If the distance between the closest pseudopoint is higher than the pseudopoint's radius, the model cannot classify the instances. If no model in the ensemble can classify an instance, it is marked as an outlier and sent to a buffer. A novelty detection procedure is applied to the outliers buffer whenever it reaches the maximum size.

ECHO declares the emergence of a new class whenever its novelty detection procedure detects enough instances close to each other in the outlier buffer. It supposes that an instance should be close to instances of the same class and farther apart from instances from different classes. In this sense, to declare the emergence of a new class, the novelty detection verifies if there are at least Q instances in the outliers buffer that met a silhouette coefficient-based condition, where Q is a parameterized value. If the condition is met, the subset of instances is removed from the buffer and declared as belonging to a new class. ECHO assumes that only one novel class may appear at a time in the data stream.

Chapter 3

Two-level Active Learning for Clustering-based Data Stream Classifiers

In this chapter, we start by describing the two-level active learning strategy. After that, we describe the implementation of the strategy for distinguishing concept drift and evolution, the different modules of the resulting framework, and the time complexity of the related procedures.

3.1 Two-level active learning strategy

Clustering-based data stream classifiers use an arbitrary decision rule to categorize a cluster as the extension of a known class or the emergence of a new one. Even though this decision rule varies from one approach to another, it is a point in common among the clustering-based data stream classifiers, which we will refer to as the cluster categorization hypothesis.

Considering that the clusters generated by the underlying clustering algorithm are fully or partially composed by instances for which the true labels are unknown, to predict the category of a cluster, the categorization hypothesis must rely on a semi-supervised assumption to obtain information about the cluster category through the unlabeled data. However, as known classes change and new classes emerge (see Section 2.2), the interaction between the data distribution and the posterior probability of the classes also changes. In consequence of that, some semi-supervised assumptions may not always be valid.

Situations in which a semi-supervised assumption fails may lead to an inaccurate class inference for the unlabeled data, compromising the categorization hypothesis and, consequently, the performance of the data stream classifier. For these situations, label information may be necessary to provide more information about the interaction between the cluster distribution and the posterior probability of the data classes.

Given the importance of the correct categorization of the clusters, we consider the clustering-based data stream classifiers as a multiple-instance setting (see Sections 2.3 and 2.3.2). In this context, we propose an active learning technique that first queries the clusters for which the categorization is uncertain and then queries for the most informative instances in terms of the inner distribution of each cluster. This approach presents the advantage of focusing the labeling resource in the categorization of the cluster, whose summaries are the blocks that compose the learning model. The general flow of the proposed two-level active learning is presented in Fig. 1.

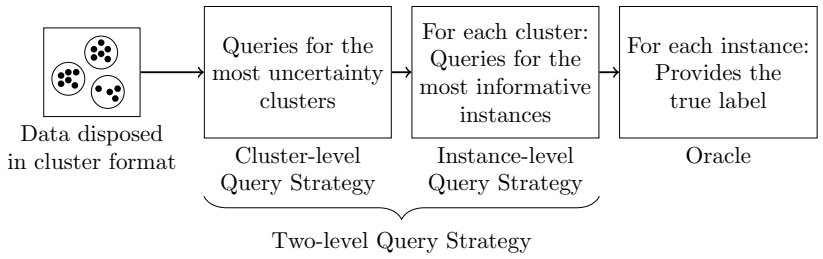


Figure 1 – The general flow of the proposed two-level active learning strategy.

As shown in Fig. 1, the proposed active learning structure queries the data first at the cluster-level, and then, for each cluster selected, it queries for the data at the instance-level. The combination of the two query strategies defines the two-level query strategy.

We summarized the main questions that need to be considered to implement the two-level active learning strategy in a clustering-based data stream classifier:

1. How to query for clusters for which the categorization hypothesis is more likely to fail?

2. How to query for the instances that would provide more information about the cluster distribution?

3. How to use the label information of the queried instances to improve the clusters' categorization?

The strategy is expected to improve the base classifier's cluster categorization performance by asking for the least possible number of labels.

3.2 The proposed framework

We propose a framework based on the two-level active learning strategy to improve the performance of clustering-based data stream classifiers in distinguishing concept drift and evolution. The integration of the framework to a compatible classifier is performed as follows:

1. We add a different categorization hypothesis, alongside the pre-existent one, forming a two-member committee (see Section 2.3.1). Whenever the committee disagrees about the category of a cluster, we mark the categorization as unreliable.

2. Once an unreliable categorization is detected, we evaluate each instance in the target cluster to find which one can provide more information about the cluster's true category. We query the oracle for the label of the K most informative instances.

3. After querying the labels of the most informative instances, we re-categorize the cluster by considering the new labels obtained and any previously available labeled instances.

A clustering-based data stream classifier is compatible with the framework if it presents the following characteristics:

- ❏ At some point of the flow of the classifier, a valid cluster of unlabeled or partially labeled data is made available.

- ❏ A cluster categorization hypothesis decides if the cluster belongs to a new or a known class. We refer to this categorization hypothesis as base-categorizer.

- ❏ The composition of the learning model of the classifier includes a set of cluster summaries representing the known classes. We refer to this set as the data classes summary.

When the cluster is assigned to a known class, we consider that the base-categorizer categorized it as "known", otherwise as "novelty". Clusters categorized as something else, like noise, are not considered. The category of a cluster indicates which distribution change phenomenon it represents. If the cluster is categorized as "known", the update procedure is assuming that the cluster represents the change in the concept of a class already known, a concept drift. If the cluster is categorized as "novelty", the update procedure is assuming that the cluster represents the emergence of a new class, a concept evolution. We refer to the moment when the base-categorizer categorizes a cluster as the interception point. The framework works as an interceptor, called at the interception point. It receives as input the cluster, the category predicted by the base-categorizer, and the data classes summary.

Two modules compose the framework: the meta-categorizer and the active-categorizer. The meta-categorizer is a categorization hypothesis that utilizes the data classes summary to predict the cluster's category. The meta-categorizer is the first module to be called. If the category predicted by the meta-categorizer differs from the one predicted by the base-categorizer, the cluster is queried. In the framework context, both the base-categorizer and the meta-categorizer compose a categorization committee. For that reason, the approach defines a query-by-committee strategy for querying the data at the cluster-level.

If the base-categorizer and meta-categorizer agree about the cluster's category, the framework finishes the interception. However, if the committee disagrees, the active-categorizer module is called.

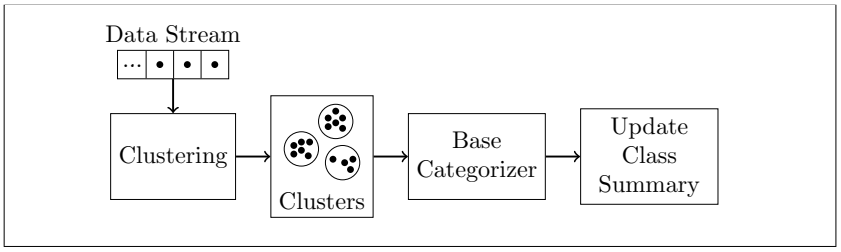
The active-categorizer is responsible for querying the data at the instance-level and categorizing the cluster using the obtained label information. The two-member committee and the active-categorizer's query strategy both compose the two-level query strategy.

In Table 1 we summarize the the main terms related to the proposed framework.

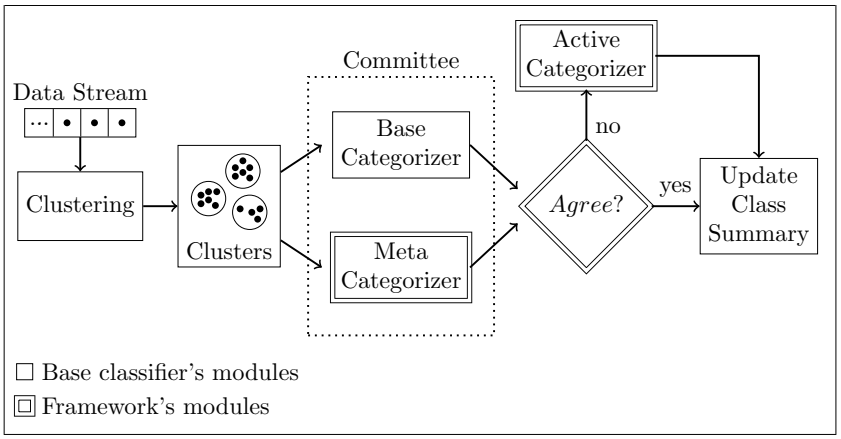
Table 1 – Description of the main terms related to the proposed framework.

Term	Description	Belongs to
Data class summary	Set of cluster summaries representing the classes known by the base classifier's learning model.	Base classifier
Base-categorizer	Cluster categorization hypothesis that decides if a cluster generated in the base classifier's update process belongs to a new or known class.	Base classifier
Meta-categorizer	Cluster categorization hypothesis that utilizes the base classifier's data classes summary to categorize a cluster as "known" or "novelty".	Proposed Framework
Active-categorizer	Cluster categorization hypothesis that categorizes a cluster as "known" or "novelty" by querying for the label of some instances inside the cluster.	Proposed Framework

Fig. 2 shows the base classifier's cluster categorization flow and how the modules of the proposed framework integrate with the base classifier's modules. As shown in Fig. 2 (a), without the framework, the clustering-based data stream classifier relies only on the base-categorizer to assimilate a category to the cluster and update the data classes summary. In Fig. 2 (b), we demonstrate how the meta-categorizer and active-categorizer are integrated into the base classifier's modules to improve the cluster categorization task.



(a) General cluster categorization flow.



(b) General cluster categorization flow after integration of the proposed framework.

Figure 2 – General cluster categorization flow of a clustering-based data stream classifier, first at its original form (a), and then after the integration of the proposed framework (b).

It's important to notice that if the cluster is not queried, it does not mean the categorization is correct. The base-categorizer may miscategorize a cluster, and the meta-categorizer may agree with it by also missing the correct category. In Table 2, we list all the possible situations that can arise from an interception. In the table, the three first columns represent the involved components. Each row represents a possible interception situation, where the "Hit" or "Miss" values for the component columns represent if the respective component correctly or wrongly categorized the cluster, respectively. The Accuracy Impact column shows the impact that each situation has on the categorization performance. The last column indicates which cases result in label consultation.

Table 2 – All possible scenarios considering the different possible outputs from each module involved in the cluster categorization.

Categorizer				
Base	Meta	Active	Accuracy Impact	Label Consultation
Hit	Miss	Hit	Neutral	Yes
Hit	Miss	Miss	Negative	Yes
Hit	Hit	-	Neutral	No
Miss	Miss	-	Neutral	No
Miss	Hit	Miss	Neutral	Yes
Miss	Hit	Hit	Positive	Yes

As shown in Table 2, the categorization accuracy of the base classifier is improved when a miscategorization by the base-categorizer is contested by the meta-categorizer and corrected by the active-categorizer. For the situations where the base-categorizer decision is kept, that is, when the base-categorizer and meta-categorizer agree, the accuracy isn't impacted. Of all the situations, the one that impacts the accuracy negatively is when the base-categorizer correctly predicts the category, and both the meta-categorizer and the active-categorizer get the category wrong.

To maximize the framework's positive impact over the base classifier, whenever the base-categorizer miscategorizes a cluster, the meta-categorizer must be able to contest the decision, and the active-categorizer must be able to correct it. To minimize the labels consultations, the meta-categorizer must only contest the base-categorizer's miscategorizations. And to avoid negatively impacting the base classifier, it is expected that, whenever the meta-categorizer contests the base-categorizer without necessity, the active-categorizer is able to confirm the base-categorizer decision as the correct one.

The framework's performance in improving the base classifier categorization performance depends fundamentally on the meta and active categorizers. In this sense, we propose different approaches for both modules. More information about the inner behavior and the different approaches of the modules are provided subsequently.

3.2.1 The meta-categorizer

The meta-categorizer is the module of the framework used to compose a two-member committee alongside the base-categorizer. It receives this name because it does not maintain a model of its own. Instead, it uses the base classifier's data classes summary to categorize clusters.

An implementation of the meta-categorizer receive as input the target cluster and the data classes summary and outputs the predicted category for the cluster. Any decision rule that is able to categorize a cluster as a concept evolution or as concept drift by utilizing the data classes summary can be used as a meta-categorizer. The basic requirement is that

the meta-categorizer does not utilize the same categorization hypothesis that constitutes the base-categorizer. Otherwise, the committee will always agree. In this sense, the more different the statistical approaches between the meta-categorizer and the base-categorizer are, the better.

The meta-categorizers implemented for this work are based on a new proposed assumption. The assumption states that, given $\hat{r}(X')$, an estimate of $r(X')$ (See Section 2.1), where X' represents the instances of a cluster, if the Grouped Error Estimate $\hat{R}$ is above a parameterized threshold, the cluster is more likely to belong to a class yet unknown by the current learning model than to a known class. The assumption comes from the fact that if the Bayes error of a region of the feature space is high, the instances sampled from this region will present less classification information than instances sampled from regions of the feature space where the Bayes error is low (FUKUNAGA; KESSELL, 1973; FUKUNAGA; HOSTETLER, 1975; FUKUNAGA; MANTOCK, 1982; FUKUNAGA, 2013). In this sense, if the data of a cluster does not carry enough classification information about any of the known classes, we may assume that the cluster belongs to an unseen class. This assumption allows us to categorize a cluster as "known" or "novelty" without requiring label information of its instances.

Beyond the scenario of the emergence of a new class, a different situation may result in a high value for the Grouped Error Estimate of a cluster, among which we can cite noise data, the natural intersection between different density functions, an imprecise estimate of the $r(X')$ function and a set X' with high variance. Another point to be considered is that, in a clustering-based data stream classifier, the estimate of the classification probability function is parametric as each cluster summary assumes a mathematical form for the respective cluster's instances. To the authors' knowledge, there are no studies on the variance of the Grouped Error Estimate through a parametric estimate of the density functions using cluster summaries. However, even in the face of these limitations, we suppose that a categorizer based on a parametric Grouped Error Estimate can be a good fit to compose the two-member committee as the categorization hypothesis of the meta-categorizer.

To implement a meta-categorizer utilizing the Grouped Error Estimate, a classification rule must be defined to provide an estimate of $r(X')$ (See Section 2.1). The classification rule used is the first nearest neighbor classifier (NN), where an instance is classified by the nearest instance in a set of labeled instances. In the context of the clustering-based data stream classifier, to calculate the classification probability for a given instance, we compute the nearest centroid in the data classes summary. Because of that, our estimate of the risk function is parametric. We consider only one centroid per class, in this sense, if more than one centroid is assigned to a class, only the one closest to the instance is considered (in the event of a tie, we choose the centroid respective to the cluster summary that was added to the decision model first). Following, we present the mathematical

development of the formula to calculate the Grouped Error Estimate of a cluster using the NN of cluster summaries.

In the context of the NN classification rule, the estimate $\hat{p}(x|c_i)$ of the probability $p(x|c_i)$ of an instance x given a class c_i is given by (FUKUNAGA; HOSTETLER, 1975; FUKUNAGA; MANTOCK, 1982):

$$\hat{p}(x|c_i) = \frac{1}{L_i(x)} \tag{4}$$

Where $L_i(x)$ is the volume of the hypersphere around the instance x, with radius equals to the distance between x and the closest centroid belonging to the class c_i.

Accordingly with (FUKUNAGA; HOSTETLER, 1975; FUKUNAGA; MANTOCK, 1982), $L_i(x)$ can be given by:

$$L_i(x) = \frac{2 \cdot d(x, \overline{x_i})^D \cdot \pi^{D/2}}{D \cdot \Gamma(D/2)} \tag{5}$$

Where d is the euclidean distance function, D is the dimensionality of the feature space, $\overline{x_i}$ is the centroid representing the class c_i, and Γ is the gamma function, defined as $\Gamma(n) = (n-1)!$, case $n \in \mathbb{Z}_+$, or else as $\Gamma(n) = \int_0^\infty x^{n-1} \cdot e^{-x} dx$, case $n \in \mathbb{R}_+$, where n is the input value for the function.

The estimate $\hat{p}(c_i)$ of $p(c_i)$ is given by (FUKUNAGA; HOSTETLER, 1975; FUKUNAGA; MANTOCK, 1982):

$$\hat{p}(c_i) = \frac{1}{M} \tag{6}$$

Where M is the number of classes known by the data classes summary.

From equations 4, 5 and 6, we can develop the equation for estimating the unconditional density function (See Section 2.1) by:

$$\begin{aligned}
\sum_{j=1}^{M} (\hat{p}(x|c_j) \cdot \hat{p}(c_j)) &= \sum_{j=1}^{M} \frac{1}{M \cdot L_j(x)} \\
&= \sum_{j=1}^{M} \frac{1}{\frac{M \cdot 2 \cdot d(x, \overline{x_j})^D \cdot \pi^{D/2}}{D \cdot \Gamma(D/2)}} \\
&= \sum_{j=1}^{M} \frac{D \cdot \Gamma(D/2)}{M \cdot 2 \cdot d(x, \overline{x_j})^D \cdot \pi^{D/2}} \\
&= \frac{D \cdot \Gamma(D/2)}{M \cdot 2 \cdot \pi^{D/2}} \cdot \sum_{j=1}^{M} \frac{1}{d(x, \overline{x_j})^D} \\
&= \frac{D \cdot \Gamma(D/2)}{M \cdot 2 \cdot \pi^{D/2}} \cdot \sum_{j=1}^{M} d^{-1}(x, \overline{x_j})^D
\end{aligned} \tag{7}$$

Where d^{-1} is the inverse Euclidean distance function.

Considering equations 1 and 7, the probability $p(c_i|x)$ of a instance belonging to a class c_i can be estimated by:

$$\hat{p}(c_i|x) = \frac{\hat{p}(x|c_i) \cdot \hat{p}(c_i)}{\sum_{j=1}^{M} \hat{p}(x|c_j) \cdot \hat{p}(c_j)}$$

$$= \frac{\frac{D \cdot \Gamma(D/2)}{M \cdot 2 \cdot \pi^{D/2}} \cdot d^{-1}(x, \overline{x_i})^D}{\frac{D \cdot \Gamma(D/2)}{M \cdot 2 \cdot \pi^{D/2}} \cdot \sum_{j=1}^{M} d^{-1}(x, \overline{x_j})^D}$$

$$= \frac{d^{-1}(x, \overline{x_i})^D}{\sum_{j=1}^{M} d^{-1}(x, \overline{x_j})^D} \tag{8}$$

Finally, the classification risk $r(x)$ of a instance x being classified as c_i can be estimated by:

$$\hat{r}(x) = 1 - \hat{p}(c_i|x) \tag{9}$$

From Equation 8, 9 and 2, we propose the first meta-categorizer, referred to as 1-NN Average Risk (NNAR).

The NNAR technique calculates the Grouped Error Estimate of the cluster by computing the distance of every instance in the cluster to every centroid in the data classes summary, and then calculating 8 and 9 to obtain the classification risky of every instance. The average risk is calculated using 2. If the estimate is above a parameterized threshold, the cluster is categorized as "novelty" otherwise, as "known".

To calculate the Grouped Error Estimate, NNAR needs to compute the distance between points $s \cdot n$ times, where s is the number of centroids in the data classes summary, and n is the number of instances in the cluster. Even though this value isn't enough to impact the computational complexity of the base classifier, we also propose an alternative method that requires fewer distance computations. The method configures a new meta-categorizer, referred to as 1-NN Centroid Risk (NNCR).

The NNCR technique calculates the Grouped Error Estimate of the target cluster by computing the classification risk of its centroid. In this sense, only equations 8 and 9 are necessary. This approach, of course, is far less precise as it assume $r(X') = r(\overline{x\prime})$, where $\overline{x\prime}$ represents the centroid of X'. The approach, however, presents a lower computation cost than NNAR, since it only computes the distance function s times.

A variation for each, the NNAR and NNCR meta-categorizer, are also proposed. In these variations, we change Equation 7 to consider not only the cluster summaries' centroids but also the standard deviation of the clusters (the standard deviation is expected to be stored in the cluster summary). To do so, we add the standard deviation respective to the centroid multiplying the distance between the centroid and the target instance. The resulting classification probability can be calculated as follows:

Table 3 – Proposed meta-categorizers.

Meta-Categorizer	Description	Equations
NNAR	Computes the Grouped Error Estimate by calculating the NN classification risk of every instance inside the cluster.	8, 9 and 2
NNCR	Computes the Grouped Error Estimate by calculating the NN classification risk of the cluster's centroid.	8 and 9
NDNAR	Computes the Grouped Error Estimate by calculating the NDN classification risk of every instance inside the cluster.	10, 9 and 2
NDNCR	Computes the Grouped Error Estimate by calculating the NDN classification risk of the cluster's centroid.	10 and 9

$$\hat{p}(c_i|x) = \frac{[d^{-1}(x, \overline{x_i}) * \sigma_i^{-1}]^D}{\sum_{j=1}^{M}[d^{-1}(x, \overline{x_j}) * \sigma_j^{-1}]^D} \tag{10}$$

Where σ_i^{-1} is the inverse of the standard deviation of the class c_i.

The classifier represented by Equation 10 is referred to as First Denser Nearest Neighbor (NDN), while the variations of NNAR and NNCR, are referred to NDN Average Risk (NDNAR) and NDN Centroid Risk (NDNAR), respectively. The motivation behind the utilization of the NDN is the low-density assumption. From this assumption, we may expect clusters denser to be more likely to be far from the region where the decision boundary between classes lies. Because of that, we expected clusters denser to better characterize the class to which they were assigned before being summarized and added to the data classes summary. The proposed meta-categorizers are summarized in Table 3.

3.2.1.1 The threshold parameter

The classification risk of an instance is a value in the interval $(0, 1 - \frac{1}{M})$. Since the Grouped Error Estimate of a cluster is calculated by the mean of the classification risk of its instances, the estimate will also be a value inside this interval. In this context, the categorization threshold t can be defined as $t = (1 - \frac{1}{M}) * factor$, where $factor$ is a parametrized value between 0 and 1 that defines the threshold.

An important point to consider is the behavior of the estimate of the classification probability (Eq. 8) in high-dimensional feature spaces. In the situations where an instance is successfully classified by a single class c_i, the inverse distance $d^{-1}(x, \overline{x_i})$ of x to the closest centroid of c_i is the highest value among $d^{-1}(x, \overline{x_j})$ for $j \in |M|$. From this fact, it can be shown that in these situations, $\lim_{D\to\infty} \frac{d^{-1}(x,\overline{x_i})^D}{\sum_{j=1}^{M} d^{-1}(x,\overline{x_j})^D} = 1$, meaning that, as D increases,

the estimate of the classification probability tends to 1, and consequently, considering Eq. 8, the classification risk tends to 0. Since the the Grouped Error Estimate is given by the mean of the estimate classification risk of the instances in a cluster (Eq. 8), for high-dimensional datasets, this estimate will be compressed close to zero, reducing the interval where the threshold must be defined in order to effectively separate low and high values of the estimate.

To avoid this problem, when using the equations 8 and 10, we reduce the effect of the dimensionality over the estimate of the classification probability by replacing D by 1. This allows us to consider the same range threshold values for datasets of different dimensionality.

3.2.2 The active-categorizer

Whenever the base-categorizer and the meta-categorizer diverge about the category of a cluster, the active-categorizer is called. The active-categorizer is responsible for querying the data at the instance-level. The module is also responsible for categorizing the cluster using the label information obtained.

We propose two different implementations for the active-categorizer. Both implementations select K instances of the cluster to be labeled, where K is a parameterized value. After the instances are selected, the cluster is categorized by considering the majority class among the labeled instances. If the majority belong to a known class, the cluster is categorized as "known", otherwise as "novelty". The implementations are referred to as the Majority of the K Centroid Neighbors (MKCN) and the Majority of the K Random (MKR). The implementations differ from each other by the way each one selects the instances to be labeled. MKCN active-categorizer selects from the cluster the K instances more close to the centroid to be labeled. MKR active-categorizer selects K random instances in the cluster to be labeled.

The hypothesis for the MKCN active-categorizer is that instances more close to the centroid of the cluster provide more information about the cluster distribution, and consequently, are adequate for indicating the category that results in less inference error. The hypothesis for the MKR is that all instances inside the cluster have the same informational value.

3.2.3 Time complexity

To calculate the Grouped Error Estimate for a given cluster, the NNAR and NDNAR meta-categorizers compute the euclidean distance of every cluster's instance to every centroid obtained from the data classes summary. So, the time complexity of NNAR and NDNAR is $O(n \times s \times D)$, where n is the number of instances inside the cluster, s is the number of centroids in the classes summaries, and D is the dimensionality of the data.

The NNCR and NDNCR meta-categorizers calculate the Grouped Error Estimate of a given cluster by first computing the cluster's centroid and then computing the distance of the obtained centroid to every centroid in the classes summaries. The time complexity to compute the cluster's centroid is $O(n \times D)$, while the time complexity to compute the distance of the obtained centroid to every centroid in the classes summaries is $O(s \times D)$. So, the final time complexity of NNCR and NDNCR is $O(D \times (s + n))$.

To select which cluster's instances to ask the oracle for the label, the MKCN active-categorizer starts by computing the cluster's centroid. This procedure presents a time complexity of $O(n \times D)$. After the centroid is computed, the method calculates the distance of the centroid to every cluster's instance while keeping track of the K closest instances. The distances computation and the selection of the K instances closer to the centroid has a time complexity of $O(n \times (K + D))$. After the K instances are selected, each one is labeled by the oracle. Considering that an instance labeling by the oracle has the constant cost of $O(1)$, the time complexity of labeling the K selected instance is $O(K)$. So, the final time complexity of the MKCN active-categorizer is $O(n \times (K + D))$

The MKR active-categorizer chooses at random K cluster's instances to be labeled by the oracle. Considering that choosing at random an instance has a constant cost $O(1)$, the time complexity of the MKR active-categorizer is $O(K)$.

Chapter 4

Experimental Results

In this chapter, we describe the evaluation strategy proposed for evaluating the framework and its modules. We start by defining two different ways to integrate the framework to the base classifier: the tight integration, which refers to the standard way of integrating the framework to the classifier, and the loose integration, a non-intrusive way of integrating the framework, intended to evaluate the meta and active categorizers in the context of the base classifier original behavior. Once both integrations settings are defined, we present the evaluation measures used to evaluate the framework.

4.1 Framework integration with clustering-based classifiers

The framework can be integrated with a clustering-based data stream classifier in two different manners: tight integration or loose integration.

4.1.1 Tight integration

Tight integration is the standard way to integrate the framework into a base classifier, as shown in Figure 2. In this integration, the original behavior of the baseline classifier is changed. Whenever the active-categorizer is called, the base-categorizer decision is ignored, and the base classifier's behavior is adjusted, so the cluster is processed by taking into account the category predicted by the active-categorizer. The objective of this integration is to apply the framework to a compatible data stream classifier to improve its performance in distinguishing concept drift and concept evolution. We assume that tight integration with a compatible data stream classifier is justified if good results are obtained in a loose integration.

4.1.2 Loose integration

In the loose integration, the original behavior of the base classifier is not changed. The results outputted by the active-categorizer are used only for evaluation purposes. The correct category is calculated and compared with the categories predicted by the base-categorizer, meta-categorizer and active-categorizer. The objective of the loose integration is to evaluate and analyze the framework in the context of a classifier without changing the classifier's original behavior or results. This analysis is done by counting how many miscategorizations made by the base-categorizer can be detected and for how many of them the active-categorizer can output the correct category.

After a cluster is made available, it is categorized by the base-classifier and meta-categorizer. If the two modules disagree about the cluster category, the cluster is also categorized by the active-categorizer. The categorizations are then sent to the module responsible for calculating the evaluation measures. The loose integration allows us to analyze the framework considering the base classifier original behavior.

4.2 Evaluation Measures

4.2.1 Categorization Confusion Matrix

We define the cluster categorization as a binary classification task, where the "novelty" category is considered the positive class and the "known" category is considered the negative class. The categorization confusion matrix (Table 4) can be calculated for the base-categorizer, meta-categorizer and active-categorizer.

The cluster's true category is defined in the context of the state of the learning model at the interception point. If the majority of the cluster instances belong to one or more classes known by the learning model, we state that the cluster's true category is "known". If most of the instances belong to one or more classes unknown by the current learning model, we state that the cluster's true category is "novelty".

Given the confusion matrix of predictions made by a categorizer over a set of clusters, we calculate the categorization sensitivity (CSe) and categorization specificity (CSp) as shown in the equations 11 and 12, respectively.

Table 4 – Categorization confusion matrix

		Predicted Category	
		Novelty	**Known**
True Category	**Novelty**	$TrueNovelty$	$FalseKnown$
	Known	$FalseNovelty$	$TrueKnown$

$$CSe = \frac{TrueNovelty}{TrueNovelty + FalseKnown} \tag{11}$$

$$CSp = \frac{TrueKnown}{TrueKnown + FalseNovelty} \tag{12}$$

The CSe and CSp measures allow us to evaluate the performance of the base-categorizer, and also the different approaches for the meta-categorizer and active-categorizer modules.

4.2.2 Cluster Querying Confusion Matrix

As described in Section 3, the query-by-committee strategy queries a cluster whenever the meta-categorizer disagrees with the base-categorizer. In this sense, and considering that there are only two possible categories, "known" or "novelty", if the strategy queries a cluster, we may conclude that one of the two members of the committee missed the cluster's true category.

Both the meta and base categorizers are expected to fail at some point. However, to maximize the committee's ability to detect miscategorizations, they must never fail at the same time. Otherwise, the categorization error will pass unnoticed, negatively impacting the framework's efficacy in detecting miscategorizations.

We also don't want the committee to always disagree. Otherwise, the active-categorizer will be called for situations where, without additional label information, the base-categorizer could correctly categorize the cluster, representing a waste of labeling resources.

In this sense, to evaluate the meta-categorizer, it is important to analyze in which situations it has failed. For that reason, we define the query-by-committee in terms of a binary classification task, where an agreement is the positive class and a disagreement the negative one. In this context, an *GoodAgreement* represents an agreement where both the meta and base-categorizer correctly predicted the cluster's true category and a *BadAgreement* represents an agreement where both categorizers missed the cluster's true category. While a *GoodDisagreement* represents a disagreement where the meta-categorizer was the one to predict the cluster's category correctly, and a *BadDisagreement* represents a disagreement where the meta-categorizer missed the true category. The cluster querying confusion matrix is represented at Table 5.

Table 5 – Committee agreement confusion matrix

		Meta Categorizer	
		Hit	**Miss**
Base Categorizer	**Hit**	*GoodAgreement*	*BadDisagreement*
	Miss	*GoodDisagreement*	*BadAgreement*

Once the matrix is available, we use the summarized outcomes to calculate the querying precision (QPr) and querying sensitivity (QSe) measures. The formula for theses measures are presented in equations 14 and 13.

$$QPr = \frac{GoodDisagreement}{GoodDisagreement + BadDisagreement} \tag{13}$$

$$QSe = \frac{GoodDisagreement}{GoodDisagreement + BadAgreement} \tag{14}$$

The QPr and QSe measures allow the evaluation of a meta-categorizer in the context of the query-by-committee. The QPr indicates how many of the queried clusters were miscategorized by the base-categorizer. When this measure reaches its maximum value, it means that the framework was able to only send to the active-categorizer the clusters that the base-categorizers could not categorize. The QSe indicates how many of the clusters miscategorized by the base-categorizer were queried. When this measure reaches its maximum value, it means that the framework was able to detect all miscategorizations and send the respective clusters to the active-categorizer.

4.2.3 Framework Impact Confusion Matrix

If the query-by-committee queries all and only the clusters for which the base-categorizer missed the correct category, and if the active-categorizer provides the correct category for all the queried clusters, the framework reaches its maximum performance. To measure the performance of the framework as a whole, we define the framework in terms of a binary classification task, where a *GoodDisagreement* represents a *Recovered* categorization if the active-categorizer outputs the correct category for the respective cluster and an *Unrecovered* categorization if the active-categorizer miscategorize the cluster. While a *BadDisagreement* represents a *Corrupted* categorization if the active-categorizer miscategorizes the respective cluster and an *Uncorrupted* categorization if the active-categorizer outputs the correct category. The respective confusion matrix is represented at Table 6.

Table 6 – Confusion matrix for the impact over the categorization of the queried clusters

		GoodDisagreement	**BadDisagreement**
Active Categorizer	**Hit**	*Recovered*	*Uncorrupted*
	Miss	*Unrecovered*	*Corrupted*

Once the matrix is available, we use the summarized outcomes to calculate the error recovery rate ($RecR$) and the corruption rate ($CorrR$) measures. The formula for calculating the measures given a framework impact confusion matrix is presented in equations 15 and 16, where the index BC indicates that the outcome refers to the categorization confusion matrix calculated for the base-categorizer.

$$RecR = \frac{Recovered}{FalseNovelty_{BC} + FalseKnown_{BC}} \tag{15}$$

$$CorrR = \frac{Corrupted}{TrueNovelty_{BC} + TrueKnown_{BC}} \tag{16}$$

Although it can be calculated for both tight and loose integration, in the loose integration setting, the $RecR$ and $CorrR$ are only virtual measures, meaning that the errors are not actually recovered or introduced since the base classifier does not consider the decisions made by the framework in this setting.

In the tight integration, beyond calculating the $RecR$ and $CorrR$ measures, we are also interested in calculating the final categorization sensitivity and specificity considering the cluster miscategorizations recovered by the framework and also any eventual categorization corruption. These measures allow us to compare, in terms of cluster categorization performance, the original base classifier and the approach resultant of integrating the framework to the base classifier. The equation for both measures, referred to as $FinalCSe$ and $FinalCSp$ are presented in equations 17 and 18 respectively, where the indexes N and K indicate when the $Recovered$ and $Corrupted$ outcomes refer to the actual "known" clusters, or the actual "novelty" cluster, respectively.

$$FinalCSe = \frac{TrueNovelty_{BC} + (Recovered_N - Corrupted_N)}{TrueNovelty_{BC} + FalseKnown_{BC}} \tag{17}$$

$$FinalCSp = \frac{TrueKnown_{BC} + (Recovered_K - Corrupted_K)}{TrueKnown_{BC} + FalseNovelty_{BC}} \tag{18}$$

The $FinalCSe$ and $FinalCSp$ are only calculated for the tight integration setting as it measures the final categorization performance of the base classifier after incorporating the framework.

4.3 Experiments

In this section we analyze the framework in the context of two different clustering-based data stream classifiers from the literature: MINAS and ECHO. We start by describing the benchmark datasets used in the experiments. Then, we describe the base-categorizer, data classes summary, interception point, and baseline results of both MINAS and ECHO. After that, we present an analysis of the NNAR, NNCR, NDNAR, and NDNCR threshold factor parameter. In sequence, we evaluate the framework by setting up a loose integration experiment for each base classifier. Finally, we tightly integrate the framework to MINAS and show the difference in performance between the original classifier and the tight integration approach in terms of cluster categorization performance and the number of labels required.

4.3.1 Datasets

To execute the experiments, we used a set of four benchmark datasets, listed in Table 7. MOA and SynEDC are synthetic datasets, while KDD99 and covtype are real-world datasets. In the table, the Training and Test columns indicate the portion of the data used in the base classifier's initial training and testing phase, respectively. The *#Instances* columns indicate the number of instances used in the training and test. The total number of classes is indicated in the test's *#Classes* column, while the number of classes present in the training is indicated in the training's *#Classes* column.

Table 7 – Benchmark datasets

		Training		Test	
Dataset	Dimensionality	*#Instances*	*#Classes*	*#Instances*	*#Classes*
MOA3	4	10000	2	90000	4
SynEDC	40	40000	7	360000	20
KDD99	34	48993	2	444619	5
covtype	54	47045	2	522911	7

The portion of the data used in the training is respective to 10% of the dataset's total (FARIA et al., 2016b). Futher details about the datasets are provided following.

4.3.1.1 MOA3

This synthetic dataset was created using the MOA (Massive Online Analysis) software (BIFET et al., 2010), and used by (FARIA et al., 2016a). Normal-like distributions generated the dataset's instances. Each distribution is represented by a 4-dimensional hypersphere that moves in the hyperspace, generating changes in the classes' distributions. The stream starts with two classes, and two more appears during the stream, totalizing four classes in the dataset. The attribute values are between 0 and 1. The dataset contains 100.000 instances, of which the first 10% were used as training data.

4.3.1.2 SynEDC

This synthetic dataset was used by (MASUD et al., 2010), (AL-KHATEEB et al., 2012a), (AL-KHATEEB et al., 2012b) and (FARIA et al., 2016a). It contains both concept drift and concept evolution. The instances are generated by Gaussian distributions in the 40-dimensional space, with different means (-5.0 to +5.0) and variances (0.5 to 6) per class. Beyond that, some classes appear and disappear during the stream. The attribute values are between 0 and 1. The dataset contains 400.000 instances, of which the first 10% were used as training data.

4.3.1.3 KDD99

This dataset in its original format can be found at the UCI dataset repository (FRANK; ASUNCION et al., 2011). It refers to the real problem of real-time automatic detection of cyberattacks. Each instance of the dataset represents the information about a network connection, classified as normal, or one of the 22 different types of attack, that can be condensed into five classes. The version of the dataset used in this work corresponds to the 10% version of the original dataset. Only the numeric attributes of the original dataset were considered, totalizing 34 attributes normalized to the range [0, 1]. The attribute values are between 0 and 1. The resulting dataset contains 494.021 instances, of which the first 10% corresponds to the training data. In order to keep only two classes in the training (classes "dos" and "normal"), we removed 409 instances from the training data.

4.3.1.4 covtype

This dataset in its original format can be found at the UCI dataset repository (FRANK; ASUNCION et al., 2011). It refers to a real problem of predicting the forest cover type from cartographic information such as elevation, slope, soil type, etc. The dataset contains instances with a total of 7 different types of forest cover. Each sample has 54 numeric attributes. All attributes were used and normalized to the [0, 1] interval. The dataset contains 581.012 instances, of which the first 10% corresponds to the training data. In order to keep only two classes in the training, we removed 11.056 instances from the training data.

4.3.2 Base classifiers

The base-categorizer, data classes summary, and interception point of both MINAS and ECHO are described in more detail in sequence. We also present the classifiers' parameter configuration used in the experiments. The parameters configuration were defined by taking into account the experiments made in the papers where MINAS (FARIA et al., 2016a), and ECHO (HAQUE et al., 2016) were proposed as well empirical evaluations of the performance of the classifiers in the implementations made for this work.

4.3.2.1 MINAS

The data classes summary of MINAS is defined as the set of all micro-cluster assigned to a known class in its classification model in a given moment. Its base-categorizer is defined as follows: If a cluster is assigned to a known class by the novelty detection procedure, it is categorized as "known", otherwise, as "novelty". Finally, we define the interception point as the moment after the cluster is categorized.

In the executed experiments, the parameters of MINAS were configured as follows. For all datasets, the number of instances to execute the novelty detection procedure

($MaxBufferSize$) was defined as 2000, the number of clusters generated in the novelty detection procedure ($NumClusters$) was defined as 100, the minimum number of instances in a cluster ($MinClusterSize$) was defined as 20 and the window size to forget outdated instances ($WindowToForget$) was defined as 4000. The window size to send micro-clusters from the decision model to the sleeping memory ($WindowToSleep$) was defined as 4000 for MOA3, SynEDC, and covtype, and as 1000 for KDD99. We choose not to update the micro-clusters at each new instance explained by the model for all datasets. The clustering algorithm used in the offline phase was CluStream (AGGARWAL et al., 2003; FARIA et al., 2016b), and the K-Means++ for the online phase. The seed used by the random procedures of the K-Means++ algorithm was fixed for all runs. The parameters are listed in Table 8.

Table 8 – MINAS' parameters

Parameter	Value
$MaxBufferSize$	2000
$NumClusters$	100
$MinClusterSize$	20
$WindowToForget$	4000
$WindowToSleep$	4000 (1000 for KDD99)
$UpdateMicro$	False
$OfflineClustering$	CluStream
$OnlineClustering$	KMeans++

The decision rule to verify if a micro-cluster explains an instance was defined as follows: if the distance between the instance x and the centroid $\overline{x_i}$ of the micro-cluster i is lower than two times the standard deviation σ_i of the micro-cluster i, the instance is classified. In order words, if the condition $d(x, \overline{x_i}) < 2 \cdot \sigma_i$ is true, the instance x is classified with the class of the micro-cluster i, where d is the euclidean distance function.

The decision rule to verify if a target micro-cluster i is the extension of another micro-cluster j was defined as follows: if the distance between the centroids $\overline{x_i}$ and $\overline{x_j}$ is lower than the sum of the standard deviations σ_i and σ_j, the target micro-cluster i is said to be an extension of the micro-cluster j. In order words, if the condition $d(\overline{x_i}, \overline{x_j}) < \sigma_i + \sigma_j$ is true, the micro-cluster i is considered an extension of the micro-cluster j.

4.3.2.2 ECHO

The data classes summary of ECHO is defined as the set of all pseudopoints of all models kept in the ensemble in the current timestamp. Its base-categorizer is defined as follows: If a cluster results from the outlier buffer, it is categorized as "novelty". If a cluster results from the partially labeled window, it is categorized as "known" if the class with the highest frequency among the cluster's labeled instances is known by the data classes summary in its current state. If a cluster results from the partially labeled

window, it is categorized as "novelty" if the class with the highest frequency among the cluster's labeled instances isn't known by the data classes summary in its current state. A class is known by the data classes summary if at least one pseudopoint where the class appears with the highest frequency.

ECHO presents two interception points. The first interception is defined as the moment soon before a cohesive cluster is detected in the outlier buffer. The second interception point is defined as the moment soon before a cluster resultant from the partially labeled window is summarized in a pseudopoint and used to build a new model. For both interception points, the cluster and its instance, the data classes summary in its current state, and the category defined by the base-categorizer compose the information to be intercepted by the framework.

Since originally ECHO could only detect one new class at a time, we needed to adapt the ECHO novelty detection algorithm to utilize the same benchmark datasets for MINAS and ECHO. When ECHO's outlier buffer reaches its maximum size, instead of finding one cohesive cluster of unlabeled instances, we modified the procedure to apply a K-Means++ clustering algorithm over the buffer. Each of the resulting clusters is then categorized as a different novelty pattern added to the separated decision model. If an instance cannot be explained by ECHO's ensemble, before sending it to the outlier buffer, we try to classify the instance using the novelty pattern decision mode, using the same nearest neighbor approach used by the pre-existing decision rule for classification using the ensemble.

In the executed experiments, the parameters of ECHO were configured as follows. The radius of a pseudopoint ($ClusterRadius$) was set as two times the standard deviation of the respective cluster. For all datasets, the maximum size of the outlier buffer ($MaxBufferSize$) was defined as 2000, the number of clusters generated over the buffer ($NumClusters$) was defined as 25, the value of the gamma parameter ($Gamma$) was defined as 0.5, the active learning threshold ($ALThreshold$) was defined as 0.4, the value of the sensitivity parameter ($Sensitivity$) was defined as 0.001, the confidence threshold ($ConfThreshold$) was set to 0.7, the size of the ensemble ($EnsembleSize$) was defined as 5. The chunk ($ChunkSize$) and window size ($WindowSize$) were defined as the number of training instance ($\#Training$) divided by the ensemble size for each dataset. The seed used by the random procedures of the K-Means++ and MCIKmeans algorithms was fixed for all executions. The parameters are listed in Table 9.

Table 9 – ECHO's paramaters

Paramaters	Value
$ClusterRadius$	$2 \cdot \sigma$
$MaxBufferSize$	2000
$NumClusters$	25
$Gamma$	0.5
$ALThreshold$	0.4
$ConfThreshold$	0.7
$Sensitivity$	0.001
$EnsembleSize$	5
$ChunkSize$	$\frac{\#Training}{EnsembleSize}$
$WindowSize$	$\frac{\#Training}{EnsembleSize}$

4.3.3 Source code

The source code developed for the experiments can be found at the repositories listed in Table 10. The commit short hash indicate the state of the respective repository at the point in history where the experiments were performed. New changes may be done in the repositories in order to improve the code's readability and also the output's and variable's nomenclature, so it matches better the terms used in this work. More details about each repository can be found at the README.md file at the repository root. The pcf-workspace repository contains the source code of all other repositories already compiled, as well the datasets used in the experiments and files containing the experiment's configurations. Suggestions, doubts, or bugs can be reported in the "Issues" section of each repository.

Table 10 – Source code repositories

Repository	URL	Commit Short Hash
pcf	https://github.com/douglas444/pcf	85aeba1
minas	https://github.com/douglas444/minas	d2235cc
echo	https://github.com/douglas444/echo	7773687
pcf-categorizers	https://github.com/douglas444/pcf-categorizers	5ee3a1f
streams	https://github.com/douglas444/streams	72600cb
pcf-workspace	https://github.com/douglas444/pcf-workspace	df30211

4.3.4 Baseline results

In this section, we provide the baseline results for both ECHO and MINAS. These results refer to the execution of both algorithms over the benchmark datasets without the intervention of the framework.

Table 11 summarizes MINAS' baseline results, and Table 12 summarizes ECHO's baseline results. The $\#Novelty$ column indicates the number of true novelty clusters

generated by the underlying clustering strategy. Analogously, the $\#Known$ column indicates the amount of true known clusters generated. The $\#LC$ column indicates the number of labels consultations made by the classifier's update process (the counting does not include labels used in the initial training phase). The CSe and CSp columns indicate the performance of the base-categorizer in categorizing the clusters generated during the classifier execution.

Table 11 – MINAS' baseline results

	$\#Novelty$	$\#Known$	$\#LC$	CSe	CSp
MOA3	108	367	0	100%	0.54%
SynEDC	25	9	0	56%	100%
KDD99	43	926	0	90.69%	41.46%
covtype	203	1040	0	88.66%	29.61%

Table 12 – ECHO's baseline results

	$\#Novelty$	$\#Known$	$\#LC$	CSe	CSp
MOA3	44	9411	10435	100%	100%
SynEDC	6	4810	5456	100%	99.95%
KDD99	1	483	19847	100%	96.27
covtype	13	833	83755	100%	87.99%

4.3.5 Meta-categorizers' threshold factor parameter analysis

In this section, we analyze the meta-categorizers' threshold factor parameter to select a suitable threshold factor configuration for use in all the experiments. To do so, we execute both ECHO and MINAS in a loose integration scenario with two benchmark datasets: MOA3 and covtype. For each meta-categorizer, each classifier is executed once for a different threshold factor value. The threshold factor varies in the interval between 0.1 and 1 in steps of 0.1. For each execution, we register the meta-categorizer CSp and CSe. The obtained results are plotted in the Fig. 3, for MINAS, and Fig. 4, for ECHO. For both figures, the subfigures (a), (b), (c) and (d) refer to the application of the base classifier to the MOA3 dataset, and the subfigures (e), (f), (g) and (h) refer to the application of the base classifier to the covtype dataset.

The results obtained for MINAS over the MOA3 datataset (Fig. 3 (a), (b), (c) and (d)) show that for all the meta-categorizers, the CSe measure kept the performance of 1 from the threshold value of 0 to 0.8, and then started to decrease at the threshold value of 0.9 until reaching the performance of 0 at the threshold value of 1. This result implies that all the meta-categorizers can correctly categorize all the true novelty clusters generated by MINAS for the MOA3 dataset if the threshold factor is set to 0.8 or lower value. The graphics also show that the CSp grows almost linearly, meaning that the

Grouped Error Estimate of the true known clusters is almost equally distributed over the range of possible values. In consequence of that, the greater the value of the threshold, the better the CSp performance.

The results obtained for MINAS over the covtype dataset (Fig. 3 (e), (f), (g) and (h)) show that for all the meta-categorizers, the CSe continuously decreased as the threshold factor increased. The decrease of CSe got more significant after the threshold value of 0.8 for the NNCR and NNAR, and 0.9 for the NDNCR and NDNAR. On the other hand, the CSp increased in an accelerated way as the threshold value increased.

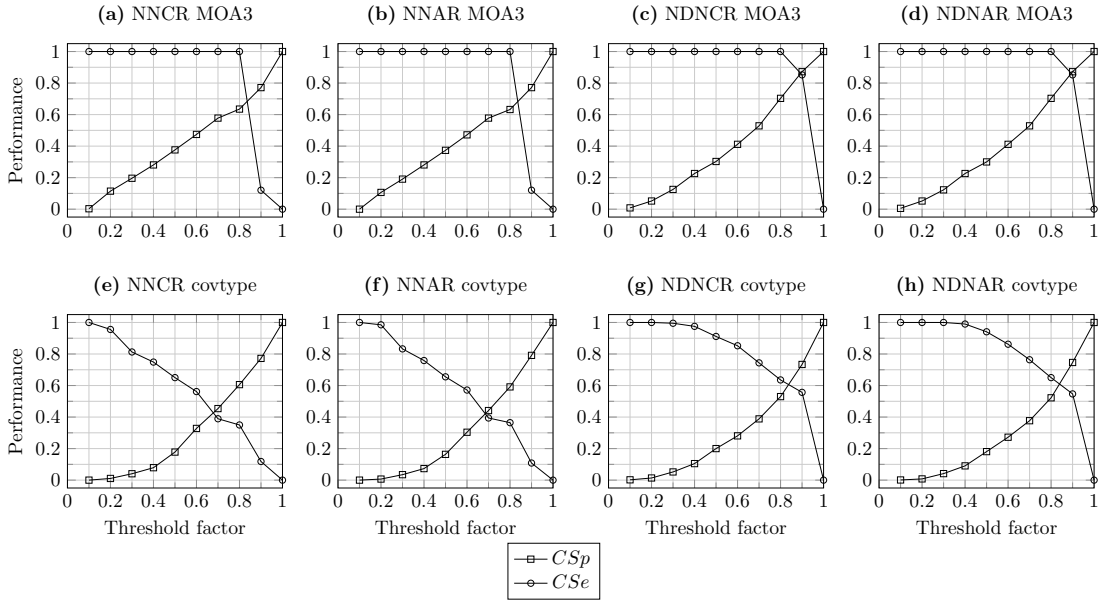


Figure 3 – Meta-categorizers' threshold factor parameter analysis for MINAS

The results obtained for ECHO over the MOA3 datataset (Fig. 4 (a), (b), (c) and (d)) shows that in the interval between 0.4 and 0.8, for all meta-categorizers, both CSe and CSp reached perfect performance. For the NNCR and NNAR applied to ECHO over the covtype dataset (Fig. 4 (e) and (f)), any threshold factor value in the interval between 0.4 and 0.8 is also suitable, although only CSe reached maximum performance, with CSp variating between the performance values of 0.8 and 0.9. For the NDNCR and NDNAR applied to ECHO over the covtype dataset (Fig. 4 (g) and (h)), the best threshold factor would be 0.3 since, after this point, the CSe falls drastically.

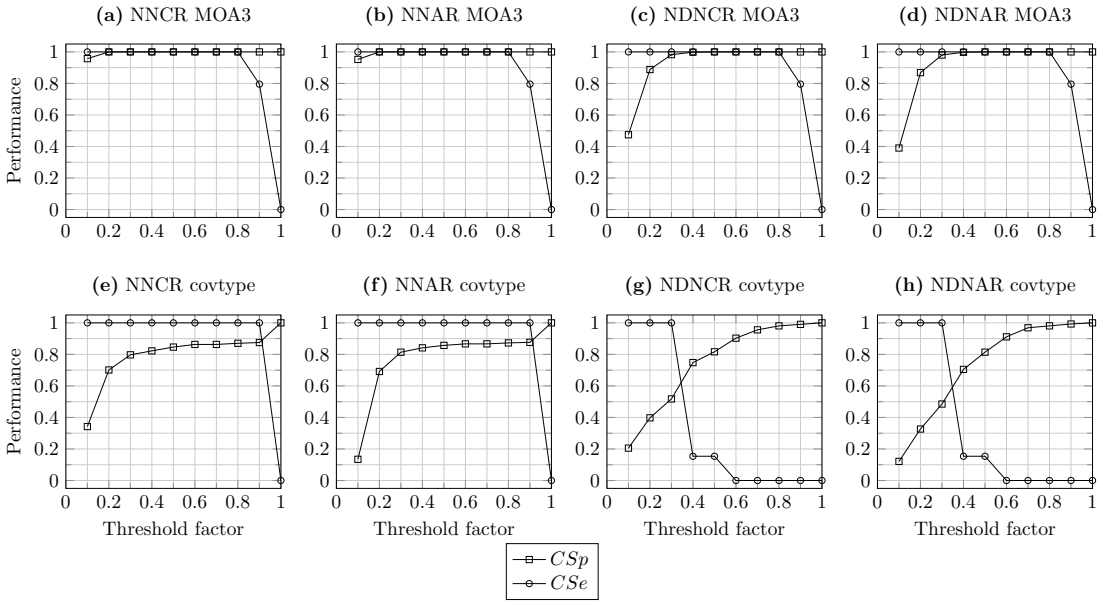


Figure 4 – Meta-categorizers' threshold factor parameter analysis for ECHO

From the analysis of the threshold factor parameter in the context of both MINAS and ECHO, we may conclude that the threshold factor value of 0.8 is a suitable setting for the NNCR and NNAR. As for the NDNCR and NDNAR, a better value for the threshold factor is 0.9, even though the experiments with ECHO and the covtype dataset point out an exception. The main conclusions used to base this configuration were two: (i) the higher the value for the threshold factor, the better the CSp performance; (ii) in general, the biggest decrease in the CSe performance occurs after the threshold factor of 0.8 for the NNCR and NNAR meta-categorizer, and after the threshold factor of 0.9 for the NDNCR and NDNAR meta-categorizers.

In the remaining experiments, we will configure the threshold factor of NNCR and NNAR to 0.8, and the threshold factor of NDNCR and NDNAR to 0.9.

4.3.6 Results of Loose Integration with MINAS

This section presents the results obtained for the loose integration with MINAS for each benchmark dataset. Table 13 shows the querying performance (See Section 4.2.2) of the query-by-committee strategy by meta-categorizer. The cells in bold indicate the best performance value for the dataset.

The table shows that the NDNAR presented the highest values for the QPr and QSe measures except for the QPr in the MOA3 and KDD99 datasets. From this result, we may conclude that the NDNAR is the best meta-categorizer to compose the committee together with the base-categorizer.

Even though the NDNAR presented the best results, it is important to consider that the meta-categorizer presents higher computational complexity than the NDNCR and NNCR meta-categorizers. In this sense, the better results obtained by the NDNAR may not justify its use as the NNCR and NDNCR were able to reach a similar performance presenting a lower computational complexity. Compared to the NDNCR, the NNCR performed better for the QPr in MOA3 and KDD99 datasets and the QSe in the SynEDC dataset. Other than that, NDNCR performed better than NNCR. Because of that, we chose NDNCR as the default meta-categorizer for the following experiments involving MINAS.

Table 13 – Meta-Categorizer's Cluster Query Performance for MINAS

	NNCR		NNAR		NDNCR		NDNAR	
	QSe	QPr	QSe	QPr	QSe	QPr	QSe	QPr
MOA3	63.28%	**100%**	63.01%	**100**%	**87.12**%	95.20%	**87.12**%	95.20%
SynEDC	90.90%	**100%**	**100%**	55%	45.45%	**100%**	90.90%	**100%**
KDD99	93.39%	91.61%	93.39%	**92.5%**	**98.05%**	89.53%	**98.44%**	91.35%
covtype	54.03%	67.66%	53.50%	66.55%	68.6%	79.93%	**69.80%**	**80.45%**

Tables 14 and 15 presents the comparison between both, the MKCN and MKR active-categorizer, where Table 14 presents the categorization measures (See Section 4.2.1), and Table 15 presents framework impact measures (See Section 4.2.3). The cells in bold indicate the best performance value for the dataset. The experiments were performed using NDNCR as meta-categorizer. In the Table 14, some cells are marked with "–", which indicates that the fraction's denominator of the equation for calculating the respective measure assumed the zero value. In the case of the CSp measure, a zero denominator means that no known cluster was presented to the categorizer, so it is not possible to evaluate the categorizer's performance in detecting known clusters.

The tables indicate that the MKCN and MKR active-categorizer presented similar performance, but the MKCN performed slightly better for the KDD99 and covtype datasets. Both active-categorizers presented a $CorrR$ higher than 0 for the covtype dataset. Both active-categorizer obtained some improvement by going from K = 1 to K = 3. The performance gain, however, may not justify triplicating the number of labeled data instances required.

Table 14 – Active-Categorizer's Categorization Performance for MINAS and NDNCR

	MKCN				MKR			
	$K = 1$		$K = 3$		$K = 1$		$K = 3$	
	CSe	CSp	CSe	CSp	CSe	CSp	CSe	CSp
MOA3	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**
SynEDC	**100%**	–	**100%**	–	**100%**	–	**100%**	–
KDD99	**100%**	**100%**	**100%**	**100%**	97.43%	**100%**	**100%**	**100%**
covtype	91.35%	98.41%	**93.82%**	98.58%	91.35%	98.41%	**93.82%**	**99.29%**

Table 15 – Active-Categorizer's Framework Impact for MINAS and NDNCR

	MKCN				MKR			
	$K = 1$		$K = 3$		$K = 1$		$K = 3$	
	$RecR$	$CorrR$	$RecR$	$CorrR$	$RecR$	$CorrR$	$RecR$	$CorrR$
MOA3	**87.12%**	**0%**	**87.12%**	**0%**	**87.12%**	**0%**	**87.12%**	**0%**
SynEDC	**45.45%**	**0%**	**45.45%**	**0%**	**45.45%**	**0%**	**45.45%**	**0%**
KDD99	**96.70%**	**0%**	**96.70%**	**0%**	**96.70%**	0.23%	**96.70%**	**0%**
covtype	67.28%	0.79%	67.41%	**0.52%**	67.28%	0.79%	**67.94%**	**0.52%**

4.3.7 Results of Loose Integration with ECHO

This section presents the results obtained for the loose integration with ECHO for each benchmark dataset. Table 16 shows the querying performance of the query-by-committee strategy by meta-categorizer. The cells in bold indicate the best performance value for the dataset.

In the table, some cells are marked with "–". In the case where QPr is marked with "–", it means that the committee agreed on all cluster categorizations. While in the case where QSe is marked with "–", it means that the base-categorizer correctly categorized all clusters. In the cases where both the QPr and QSe are marked with "–", it means that the base-categorizer and meta-categorizer both agreed for all categorization and correctly categorized all clusters, so no clusters were queried by the committee.

From this table, we may conclude that the NDNAR is the best meta-categorizer to compose the committee together with the base-categorizer since the NDNAR presented the highest values for the QPr and QSe measures except for the QPr in the SynEDC dataset. However, NDNCR reached a very similar performance, and since it has lower computational complexity, we choose NDNCR as the default meta-categorizer for the following experiments involving ECHO.

Table 16 – Meta-Categorizer's Cluster Query Performance for ECHO

	NNCR		NNAR		NDNCR		NDNAR	
	QSe	QPr	QSe	QPr	QSe	QPr	QSe	QPr
MOA3	–	–	–	–	–	0%	–	0%
SynEDC	50%	1.33%	0%	0%	50%	**3.57%**	**100%**	1.29%
KDD99	0%	0%	0%	0%	**100%**	**66.66%**	**100%**	**66.66%**
covtype	0%	0%	0%	0%	**100%**	82.64%	**100%**	**84.03%**

Tables 17 and 18, presents the comparison between both, the MKCN and MKR active-categorizer, where Table 17 presents the categorization measures, and Table 18 presents framework impact measures. The cells in bold indicate the best performance value for the dataset. The experiments were performed using NDNCR as meta-categorizer. In the tables, some cells are marked with "–". In the case of the $RecR$ measure, a cell marked with "–" indicates that the base-categorizer categorized all clusters correctly, so there's no error to be recovered.

As indicated in the tables, the MKCN and MKR active-categorizer presented similar performance, but the MKCN performed slightly better for the SynEDC, KDD99 and covtype datasets. Both active-categorizers presented a $CorrR$ higher than 0 for the covtype dataset. The performance of MKCN decreased from K = 1 to K = 3, while the performance of MKR increased.

Table 17 – Active-Categorizer's Categorization Performance for MINAS and NDNCR

	MKCN				MKR			
	$K = 1$		$K = 3$		$K = 1$		$K = 3$	
	CSe	CSp	CSe	CSp	CSe	CSp	CSe	CSp
MOA3	**100%**	–	**100%**	–	**100%**	–	**100%**	–
SynEDC	**100%**	**98.70%**	**100%**	96.29%	**100%**	96.29%	**100%**	96.29%
KDD99	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**
covtype	**100%**	**99.07%**	92.30%	98.14%	84.61%	98.14%	92.30%	96.29%

Table 18 – Active-Categorizer's Framework Impact for MINAS and NDNCR

	MKCN				MKR			
	$K = 1$		$K = 3$		$K = 1$		$K = 3$	
	$RecR$	$CorrR$	$RecR$	$CorrR$	$RecR$	$CorrR$	$RecR$	$CorrR$
MOA3	–	**0%**	–	**0%**	–	**0%**	–	**0%**
SynEDC	**0%**	**0%**	**0%**	**0%**	**0%**	**0%**	**0%**	**0%**
KDD99	**100%**	**0%**	**100%**	**0%**	**100%**	**0%**	**100%**	**0%**
covtype	**99%**	**0%**	98%	1%	98%	2%	98%	**0%**

4.3.8 Results of Tight Integration with MINAS

In this section, we present the results of the tight integration of the framework with MINAS. Table 19 present the results of the tight integration using the NDNCR meta-categorizer and MKR active-categorizer. Table 20 present the results of the tight integration using the NDNCR meta-categorizer and MKR active-categorizer.

The MKCN and MKR setting for the framework presented almost identical results, differing only for the covtype dataset, where the MKCN setting performed slightly better. The integration of the framework into MINAS presented excellent results, with a $RecR$ higher than 90% for all datasets, by the cost of acquiring very few labels considering the total of instances in the test phase.

Table 19 – Tight Integration with MINAS using NDNCR and MKCN

	$\#Novelty$	$\#Known$	$\#LC$	$RecR$	$CorrR$	$FinalCSe$	$FinalCSp$
MOA3	2	473	157	100%	0%	100%	100%
SynEDC	19	15	7	100%	0%	100%	100%
KDD99	3	966	169	99.40%	0%	66.66%	100%
covtype	3	1512	613	93.59%	0%	100%	97.28%

Table 20 – Tight Integration with MINAS using NDNCR and MKR

	$\#Novelty$	$\#Known$	$\#LC$	$RecR$	$CorrR$	$FinalCSe$	$FinalCSp$
MOA3	2	473	157	100%	0%	100%	100%
SynEDC	19	15	7	100%	0%	100%	100%
KDD99	3	966	169	99.40%	0%	66.66%	100%
covtype	5	1510	611	93.40%	0.22%	60%	97.21%

Tables 21 and 22 presents a comparasion between the baseline results of MINAS and the results obtained by tightly integrating the framework in the MKCN setting into MINAS, where Table 21 present the number of true novelty ($\#Novelty$) and true known ($\#Known$) clusters, and Table 22 present the results in the categorization of theses clusters (CSe and CSp for original MINAS and $FinalCSe$ and $FinalCSp$ for MINAS + Framwork) as well the number of labels queried ($\#LC$).

Table 21 – Comparasion between MINAS baseline results and the results obtained with the MINAS + Framwork approach: numbers of clusters

	Original MINAS		MINAS + Framwork	
	$\#Novelty$	$\#Known$	$\#Novelty$	$\#Known$
MOA3	108	367	2	473
SynEDC	25	9	19	15
KDD99	43	926	3	966
covtype	203	1040	3	1512

Table 22 – Comparasion between MINAS baseline results and the results obtained with the MINAS + Framwork approach: labels and categorization performance

	Original MINAS			MINAS + Framework		
	$\#LC$	CSe	CSp	$\#LC$	$FinalCSe$	$FinalCSp$
MOA3	0	100%	0.54%	157	100%	100%
SynEDC	0	56%	100%	7	100%	100%
KDD99	0	60.69%	41.46%	169	66.66%	100%
covtype	0	88.66%	29.61%	613	100%	97.28%

The results in Table 22 show the improvement of categorization performance obtained by MINAS by incorporating the framework. It is also interesting to notice that there are fewer novelty clusters in the approach that includes the framework. This result is expected since, by labeling instances, we give MINAS information about classes that it didn't know yet.

Chapter 5

Conclusion

In this work, we proposed an active learning strategy that queries the data first at the cluster-level and then at the instance-level to provide label information to a clustering-based data stream classifier in the context of the clusters used to update its classification model. We also proposed the implementation of this strategy for the task of distinguishing clusters representing concept drift from the ones representing concept evolution. This implementation was presented in the format of a framework. We described how to integrate the framework into a compatible clustering-based data stream classifier.

The framework relies on a query-by-committee approach to query for clusters, where a two-member committee is formed by adding a second cluster categorizer alongside the one already present in the base classifier. Different categorizers were proposed to compose the committee; none of them required an additional learning model. Instead, they use the cluster summaries already kept by the base classification model. By doing this, the framework avoids the computational cost of maintaining an extra learning model. The categorizer proposed was based on the assumption that relates the Group Error Estimate of a cluster to the concept change it is more likely to be representing, drift or evolution.

To evaluate the framework, we proposed an evaluation strategy capable of measuring the performance of each framework's module in the context of the base classifier's original behavior and the context of the behavior resultant of the full integration of the framework to it. In the experiments, we evaluate the framework considering two different clustering-based data stream classifiers from the literature: MINAS, which relies exclusively on unsupervised novelty detection algorithms to adjust its learning model, and ECHO, which utilizes a semi-supervised clustering algorithm to adapt its learning model. Four datasets were used in the experiments, being two of them real-world datasets. The analysis of the framework over the clusters generated by MINAS and ECHO during their application phase shows that the framework can effectively detect cluster miscategorization and categorize it correctly by asking for just one label per cluster.

After testing the framework without intervening in the original results of the base classifiers, we proposed a full integration of the framework to MINAS. We then com-

pared the resulting approach with the original MINAS in terms of cluster categorization performance. In the results, the approach integrating the framework to MINAS reached superior cluster categorization performance by asking for a few labels.

Future works include the following topics:

- ❏ Analyze the framework's impact over the accuracy of the base classifier in classifying the instances of the stream;
- ❏ Test the performance of the framework using datasets with more instances, so more clusters are generated, increasing the statistic relevance of the results;
- ❏ Evaluate the variance of the Group Error Estimate through the clustering-based estimate of the data classes distributions towards the actual Bayes error in simulation datasets where at any timestamp of the stream, the Bayes error is known;
- ❏ Analyze the use of the Group Error Estimate for the categorization of clusters as concept drift or evolution as the base decision rule for a clustering-based data stream classifier;
- ❏ Further investigate other settings for the framework by proposing and testing meta-categorizers other than the ones based on the Group Error Estimate, proposing and testing more active-categorizers, testing a query-by-committee with more than two members, and integrating the proposed framework into more clustering-based data stream classifiers, including approaches that use cluster algorithms other than partitional clustering;
- ❏ Implement the Group Error Estimate as an incremental measure to turn the proposed framework compatible with clustering-based data stream classifier that clusters the data online;
- ❏ Implement the two-level active learning strategy for distinguishing clusters between different classes, rather than between the known and novelty category only;
- ❏ Study the development of a two-level active learning strategy for parametric data stream classifier where the blocks of the incremental learning model is a structure other than a cluster.

Bibliography

ABDALLAH, Z. S. et al. Anynovel: detection of novel concepts in evolving data streams. **Evolving Systems**, Springer, v. 7, n. 2, p. 73–93, 2016. Disponível em: <https://doi.org/10.1007/s12530-016-9147-7>.

AGGARWAL, C. C. et al. A framework for on-demand classification of evolving data streams. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 18, n. 5, p. 577–589, 2006. Disponível em: <https://doi.org/10.1109/TKDE.2006.69>.

_____. Active learning: A survey. In: **Data Classification: Algorithms and Applications**. [S.l.]: CRC Press, 2014. p. 571–605.

_____. A framework for clustering evolving data streams. In: ELSEVIER. **Proceedings 2003 VLDB conference**. 2003. p. 81–92. Disponível em: <https://doi.org/10.1016/B978-012722442-8/50016-1>.

AL-KHATEEB, T. et al. Stream classification with recurring and novel class detection using class-based ensemble. In: IEEE. **2012 IEEE 12th International Conference on Data Mining**. 2012. p. 31–40. Disponível em: <https://doi.org/10.1109/ICDM.2012.125>.

AL-KHATEEB, T. M. et al. Cloud guided stream classification using class-based ensemble. In: IEEE. **2012 IEEE Fifth International Conference on Cloud Computing**. 2012. p. 694–701. Disponível em: <https://doi.org/10.1109/CLOUD.2012.127>.

BIFET, A. et al. Moa: Massive online analysis, a framework for stream classification and clustering. In: PMLR. **Proceedings of the First Workshop on Applications of Pattern Analysis**. [S.l.], 2010. p. 44–50.

CHU, W. et al. Unbiased online active learning in data streams. In: **Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining**. [s.n.], 2011. p. 195–203. Disponível em: <https://doi.org/10.1145/2020408.2020444>.

ENGELEN, J. E. V.; HOOS, H. H. A survey on semi-supervised learning. **Machine Learning**, Springer, v. 109, n. 2, p. 373–440, 2020. Disponível em: <https://doi.org/10.1007/s10994-019-05855-6>.

FARIA, E. R. et al. Novelty detection in data streams. **Artificial Intelligence Review**, Springer, v. 45, n. 2, p. 235–269, 2016. Disponível em: <https://doi.org/10.1007/s10462-015-9444-8>.

FARIA, E. R. de et al. Minas: multiclass learning algorithm for novelty detection in data streams. **Data mining and knowledge discovery**, Springer, v. 30, n. 3, p. 640–680, 2016. Disponível em: <https://doi.org/10.1007/s10618-015-0433-y>.

FRANK, A.; ASUNCION, A. et al. Uci machine learning repository, 2010. **URL http://archive. ics. uci. edu/ml**, v. 15, p. 22, 2011.

FREUND, Y. et al. Selective sampling using the query by committee algorithm. **Machine learning**, Springer, v. 28, n. 2, p. 133–168, 1997. Disponível em: <https://doi.org/10.1023/A:1007330508534>.

FUKUNAGA, K. **Introduction to statistical pattern recognition**. [S.l.]: Elsevier, 2013.

FUKUNAGA, K.; HOSTETLER, L. K-nearest-neighbor bayes-risk estimation. **IEEE Transactions on Information Theory**, IEEE, v. 21, n. 3, p. 285–293, 1975. Disponível em: <https://doi.org/10.1109/TIT.1975.1055373>.

FUKUNAGA, K.; KESSELL, D. Nonparametric bayes error estimation using unclassified samples. **IEEE Transactions on Information Theory**, IEEE, v. 19, n. 4, p. 434–440, 1973. Disponível em: <https://doi.org/10.1109/TIT.1973.1055049>.

FUKUNAGA, K.; MANTOCK, J. M. A nonparametric two-dimensional display for classification. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, n. 4, p. 427–436, 1982. Disponível em: <https://doi.org/10.1109/TPAMI.1982.4767276>.

GAMA, J. **Knowledge discovery from data streams**. CRC Press, 2010. Disponível em: <https://doi.org/10.1201/EBK1439826119>.

GAMA, J. et al. A survey on concept drift adaptation. **ACM computing surveys (CSUR)**, ACM New York, NY, USA, v. 46, n. 4, p. 1–37, 2014. Disponível em: <https://doi.org/10.1145/2523813>.

GARCIA, K. D. et al. Online clustering for novelty detection and concept drift in data streams. In: SPRINGER. **EPIA Conference on Artificial Intelligence**. 2019. p. 448–459. Disponível em: <https://doi.org/10.1007/978-3-030-30244-3_37>.

HAQUE, A.; KHAN, L.; BARON, M. Sand: Semi-supervised adaptive novel class detection and classification over data stream. In: **Proceedings of the AAAI Conference on Artificial Intelligence**. [S.l.: s.n.], 2016. v. 30, n. 1.

HAQUE, A. et al. Efficient handling of concept drift and concept evolution over stream data. In: IEEE. **2016 IEEE 32nd International Conference on Data Engineering (ICDE)**. 2016. p. 481–492. Disponível em: <https://doi.org/10.1109/ICDE.2016.7498264>.

JAIN, A. K.; DUIN, R. P. W.; MAO, J. Statistical pattern recognition: A review. **IEEE Transactions on pattern analysis and machine intelligence**, Ieee, v. 22, n. 1, p. 4–37, 2000. Disponível em: <https://doi.org/10.1109/34.824819>.

KHEZRI, S. et al. Stds: self-training data streams for mining limited labeled data in non-stationary environment. **Applied Intelligence**, Springer, p. 1–20, 2020. Disponível em: <https://doi.org/10.1007/s10489-019-01585-3>.

KSIENIEWICZ, P. et al. Data stream classification using active learned neural networks. **Neurocomputing**, Elsevier, v. 353, p. 74–82, 2019. Disponível em: <https://doi.org/10.1016/j.neucom.2018.05.130>.

LOSING, V.; HAMMER, B.; WERSING, H. Knn classifier with self adjusting memory for heterogeneous concept drift. In: IEEE. **2016 IEEE 16th international conference on data mining (ICDM)**. 2016. p. 291–300. Disponível em: <https://doi.org/10.1109/ICDM.2016.0040>.

LUGHOFER, E. Single-pass active learning with conflict and ignorance. **Evolving Systems**, Springer, v. 3, n. 4, p. 251–271, 2012. Disponível em: <https://doi.org/10.1007/s12530-012-9060-7>.

LUGHOFER, E.; PRATAMA, M. Online active learning in data stream regression using uncertainty sampling based on evolving generalized fuzzy models. **IEEE Transactions on fuzzy systems**, IEEE, v. 26, n. 1, p. 292–309, 2017. Disponível em: <https://doi.org/10.1109/TFUZZ.2017.2654504>.

MASUD, M. et al. Classification and novel class detection in concept-drifting data streams under time constraints. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 23, n. 6, p. 859–874, 2010. Disponível em: <https://doi.org/10.1109/TKDE.2010.61>.

MASUD, M. M. et al. A practical approach to classify evolving data streams: Training with limited amount of labeled data. In: IEEE. **2008 Eighth IEEE International Conference on Data Mining**. 2008. p. 929–934. Disponível em: <https://doi.org/10.1109/ICDM.2008.152>.

MOHAMAD, S.; SAYED-MOUCHAWEH, M.; BOUCHACHIA, A. Active learning for classifying data streams with unknown number of classes. **Neural Networks**, Elsevier, v. 98, p. 1–15, 2018. Disponível em: <https://doi.org/10.1016/j.neunet.2017.10.004>.

PARREIRA, P.; PRATI, R. Aprendizagem ativa em fluxo de dados com latência intermediária. In: SBC. **Anais do XVI Encontro Nacional de Inteligência Artificial e Computacional**. 2019. p. 365–376. Disponível em: <https://doi.org/10.5753/eniac.2019.9298>.

SETTLES, B. Active learning literature survey. University of Wisconsin-Madison Department of Computer Sciences, 2009.

SETTLES, B.; CRAVEN, M.; RAY, S. Multiple-instance active learning. **Advances in neural information processing systems**, v. 20, p. 1289–1296, 2007.

SEUNG, H. S.; OPPER, M.; SOMPOLINSKY, H. Query by committee. In: **Proceedings of the fifth annual workshop on Computational learning theory**. [s.n.], 1992. p. 287–294. Disponível em: <https://doi.org/10.1145/130385.130417>.

SVENSÉN, M.; BISHOP, C. M. **Pattern recognition and machine learning**. [S.l.]: Springer Cham, Switzerland, 2007.

ZHU, X. et al. Active learning from data streams. In: IEEE. **Seventh IEEE International Conference on Data Mining (ICDM 2007)**. 2007. p. 757–762. Disponível em: <https://doi.org/10.1109/ICDM.2007.101>.

ŽLIOBAITĖ, I. et al. Active learning with drifting streaming data. **IEEE transactions on neural networks and learning systems**, IEEE, v. 25, n. 1, p. 27–39, 2013. Disponível em: <https://doi.org/10.1109/TNNLS.2012.2236570>.

ZYBLEWSKI, P.; KSIENIEWICZ, P.; WOŹNIAK, M. Combination of active and random labeling strategy in the non-stationary data stream classification. In: SPRINGER. **International Conference on Artificial Intelligence and Soft Computing**. 2020. p. 576–585. Disponível em: <https://doi.org/10.1007/978-3-030-61401-0_54>.