**UNIVERSIDADE FEDERAL DE SÃO CARLOS**

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

# DETECÇÃO DE NOVIDADE EM FLUXOS CONTÍNUOS DE DADOS MULTIRRÓTULO

**JOEL DAVID COSTA JÚNIOR**

**ORIENTADOR: PROF. DR. RICARDO CERRI**

São Carlos – SP

Julho, 2019

**UNIVERSIDADE FEDERAL DE SÃO CARLOS**

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

# DETECÇÃO DE NOVIDADE EM FLUXOS CONTÍNUOS DE DADOS MULTIRRÓTULO

**JOEL DAVID COSTA JÚNIOR**

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação, área de concentração: Aprendizado de Máquina.

Orientador: Prof. Dr. Ricardo Cerri

São Carlos – SP

Julho, 2019

**UNIVERSIDADE FEDERAL DE SÃO CARLOS**

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

## Folha de Aprovação

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a Defesa de Dissertação de Mestrado do candidato Joel David Costa Júnior, realizada em 29/07/2019:

Prof. Dr. Murilo Coelho Naldi
UFSCar

Prof. Dr. Ricardo Cerri
UFSCar

Prof. Dr. Gustavo Enrique de Almeida Prado Alves Batista
UNSW

Certifico que a defesa realizou-se com a participação à distância do(s) membro(s) Ricardo Cerri, Gustavo Enrique de Almeida Prado Alves Batista e, depois das arguições e deliberações realizadas, o(s) participante(s) à distância está(ao) de acordo com o conteúdo do parecer da banca examinadora redigido neste relatório de defesa.

Prof. Dr. Murilo Coelho Naldi

*À minha amada mãe.*

*"A coragem não é apenas não se sentir assustado. É estar assustado mas mesmo assim fazer o que tem que ser feito."*

*— Doctor Who*

# AGRADECIMENTOS

Agradeço a Deus pela saúde, força e por guiar minhas decisões durantes esses últimos anos.

A pós-graduação pode ser um período difícil, mas foram os melhores anos de minha vida, pois tive pessoas maravilhosas ao meu lado e nesta seção eu busco agradecê-las.

Agradeço ao meu orientador Prof. Ricardo Cerri, pela excelente orientação durante todo o período de meu Mestrado. Obrigado pela confiança depositada em mim, paciência nos momentos difíceis e nas intermináveis correções dos textos, pelo incentivo, pela disponibilidade independente da hora e do dia, por todos os ensinamentos e conselhos compartilhados que foram fundamentais para realização deste trabalho.

Agradeço aos colaboradores deste trabalho, Profa. Elaine de Faria e Prof. Jonathan Silva, por toda a ajuda. Obrigado pelos ensinamentos, pelas leituras e correções dos textos, pela disponibilidade nas nossas longas reuniões e por acreditarem no meu trabalho.

Agradeço ao Prof. João Gama pela orientação durante o período de estágio em Portugal e por ter me recebido tão bem no LIAAD. Suas sugestões contribuíram muito para o desenvolvimento desta pesquisa.

Agradeço à Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) pelos auxílios financeiros indispensáveis à realização deste trabalho (Processo n. 2017/11513-0 e 2018/11321-6) e, também, à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo auxílio financeiro concedido no início desta pesquisa.

A todos os professores e funcionários do DC/UFSCar - São Carlos, que contribuíram direta ou indiretamente para a realização deste trabalho. Tudo seria mais difícil se não fossem os colegas e amigos do DC/UFSCar e do laboratório BIOMaL, que propiciaram momentos de descontração e também de discussões sobre diversas questões técnicas. Agradeço em especial meus amigos Tiago, Gean, Gerson, Eric, Igor, Frederico, Diogo, Diego, Bruno, Laís e Kenji. Sem vocês eu não teria aguentado nem do primeiro semestre.

Agradeço aos amigos Douglas Castilho e Lailiane que ajudaram a minimizar a saudade da família e do Brasil enquanto estive em Porto. Obrigado Douglas pela amizade e por toda a ajuda na pesquisa e nos momentos difíceis.

Agradeço a minha mãe Cleusa David e meu padrasto Antônio Assed por toda ajuda e incentivo durante esses anos e a meu irmão Gabriel Assed pela amizade, pelos conselhos e por me ouvir sempre.

Agradeço ao meu pai Joel Dornelas, minha madrasta Laila, meus irmãos João Paulo e Anna Luíza, meus sogros Sônia e Dênis, meus sobrinhos Arthur e Maria Clara e, toda minha família, pela torcida e orações.

Finalmente, agradeço à minha namorada Fabrícia Tavares, por acreditar em mim até quando nem mesmo eu acreditava, por vibrar comigo a cada conquista, pela paciência nos momentos de ausência e por ouvir minhas reclamações tão frequentes. Obrigado por ser minha amiga, confidente e companheira.

# RESUMO

**C**lassificação **M**ultirrótulo em **F**luxos **C**ontínuos de **D**ados (CMFCD) é a tarefa de classificar exemplos de fluxos de dados em várias classes simultaneamente. Essa é uma tarefa desafiadora pelas características presentes nos **F**luxo **C**ontínuo de **D**ados (FCD), especialmente em relação a potencial distribuição não-estacionária dos dados, em que novas classes podem surgir (Evoluções de Conceito) e classes conhecidas podem mudar ao longo do tempo (Mudanças de Conceito). Apesar dos diversos trabalhos propostos para CMFCD, a maioria assume a disponibilidade dos rótulos reais dos exemplos para atualização dos métodos. No entanto, esse é um cenário irreal, visto que muitas aplicações reais possuem Latência Extrema de Rótulos, problema no qual é inviável acessar os rótulos reais dos exemplos. Uma tarefa que têm se destacado ao lidar com esses problemas é a **D**etecção de **N**ovidade (DN). Em FCDs, a tarefa de DN consiste em identificar novos padrões em exemplos não rotulados dos fluxos de dados. Em alguns aspectos, esses padrões podem divergir dos exemplos observados e podem ser usados para atualizar os modelos de decisão. Apesar da gama de trabalhos que abordam o uso de DN em FCDs, essa é uma técnica pouco explorada para problemas de CM. Este trabalho propõem dois novos métodos: o ***Mult**I**-label** **learN**ing **A**lgorithm for data **S**treams with **L**abel **C**ombination-based methods* (MINAS-LC) que aplica DN exclusivamente para tratar mudanças de conceito; e o ***Mult**I**-label** **learN**ing **A**lgorithm for data **S**treams with **B**inary **R**elevance transformation* (MINAS-BR) que além de tratar mudanças de conceito, também é capaz de detectar novas classes ao longo dos fluxos de dados. Na Fase Offline, os métodos propostos constroem modelos de decisão baseados em microgrupos e, na Fase Online, novos exemplos são classificados ou rejeitados (marcados como desconhecidos) pelo modelo de decisão atual. Grupos de exemplos rejeitados podem formar novos padrões válidos e serem usados para atualizar o modelo de decisão. Essa atualização é feita ao longo do fluxo a fim de refletir mudanças e evoluções de conceito. Experimentos realizados com bases de dados reais e sintéticas mostraram que o MINAS-LC alcança resultados competitivos com os métodos da literatura e o MINAS-BR resultados superiores em bases de dados com evoluções de conceito, nas quais as classes emergentes são, na maioria das vezes, detectadas corretamente pelo método.

**Palavras-chave**: Classificação multirrótulo, detecção de novidade, fluxos contínuos de dados, latência extrema de rótulos.

# ABSTRACT

Multi-label Stream Classification (MLSC) aims at classifying streaming examples into multiple classes simultaneously. It is a very challenging task, since new classes may emerge (concept evolution), and known classes may change over the stream (concept drift. Most of classifier methods deal with the aforementioned characteristics updating their models when actuals labels of examples become available, it is a very optimistic scenario though, since for many data streams application these labels are never available, hence is unfeasible to update models in a supervised fashion. This is known as infinitely delayed labels problems and a task that has been stood out for dealing with it, is the Novelty Detection (ND). ND aims at detecting new patterns in arriving streaming examples that, in some aspect, might differ from the previously observed patterns and then used for updating the decision models. Despite different ND procedures have been proposed in the literature, it is still an open problem for MLSC. In this work we proposed two new methods: the ***Mult**I**-label learN**ing **A**lgorithm for data **S**treams with **L**abel **C**ombination-based methods* (MINAS-LC) which applies ND to deal exclusively with concept drifts and the ***Mult**I**-label learN**ing **A**lgorithm for data **S**treams with **B**inary **R**elevance transformation* (MINAS-BR) which applies ND to deal with both concept drifts and concept evolution either. The methods address problem transformation techniques to deal with multi-labelled data at the offline phase and build micro-clusters based decision models. At the online phase the arrival examples are classified or rejected (labelled as unknown) by the current decision models. Rejected examples are used in ND phase to shape new micro-clusters, maintaining the decision models updated. The ND phase is run from time to time to work around concept drifts and concept evolution. Experiments over synthetic and real-world data sets with different degrees of concept drift and concept evolution attested the potential of MINAS-BR and MINAS-LC, in which have obtained significant results when compared to baseline methods.

**Keywords**: Data stream, infinitely delayed labels, multilabel classification, novelty detection.

# SUMÁRIO

# LISTA DE ABREVIATURAS

**ADWIN** ***AD****aptive* ***WIN****dowing*

**BR** ***B****inary* ***R****elevance*

**CC** ***C****lassifier* ***C****hains*

**CFV** ***C****luster* ***F****eature* ***V****ector*

**CLAM** ***CLA****ss-based* ***M****icro classifier ensemble*

**CM** **C**lassificação **M**ultirrótulo

**CMFCD** **C**lassificação **M**ultirrótulo em **F**luxos **C**ontínuos de **D**ados

**DETECTNOD** ***D****iscr****ETE*** ***C****osine* ***T****ransform based* ***NO****velty and* ***D****rift detection*

**DDLGE** ***D****rift* ***D****etection based on* ***L****abel* ***G****rouping and* ***E****ntropy*

**DN** **D**etecção de **N**ovidade

**ECSMiner** ***E****nhanced* ***C****lassifier for data* ***S****treams with novel class* ***Miner***

**FCD** **F**luxo **C**ontínuo de **D**ados

**FCM-BR** ***F****uzzy* ***C****onfusion* ***M****atrix correction for* ***B****inary* ***R****elevance classifier*

**F1** *F-measure*

**F1M** *Macro F-measure*

**F1m** *Micro F-measure*

**MLHTcl** ***M****ulti****L****abel* ***H****oeffding* ***T****ree with* ***c****lass incremental* ***l****earning*

**MLHT** ***M****ulti****L****abel* ***H****oeffding* ***T****ree with pruned sets at the leaves*

**IBR** ***I****mproved* ***B****inary* ***R****elevance*

**iSOUPTree** ***i****ncremental* ***S****tructured* ***OU****tput* ***P****rediction* ***Tree***

**LBEF** ***L****abel-based* ***E****nsemble* ***F****ramework for multi-label stream classification*

**LP** ***L**abel **P**owerset*

**MCM** ***M**ulti **C**lass **M**iner in data streams*

**MINAS** ***M**ult**I**class lear**N**ing **A**lgorithm for data **S**treams*

**MINAS-BR** ***M**ult**I**-label lear**N**ing **A**lgorithm for data **S**treams with **B**inary **R**elevance transformation*

**MINAS-LC** ***M**ult**I**-label lear**N**ing **A**lgorithm for data **S**treams with **L**abel **C**ombination-based methods*

**MLDE** ***M**ulti-**L**abel **D**ynamic **E**nsemble*

**MLCC** ***M**ulti-**L**abel **C**luster-based **C**lassifiers*

**MuENL** ***Mu**lti-label learning with **E**merging **N**ew **L**abels*

**MWC** ***M**ultiple **W**indows **C**lassifier*

**OLINDDA** ***O**n**LI**ne **N**ovelty and **D**rfit **D**etection **A**lgorithm*

**OMLC** *incremental **O**nline **M**ulti-**L**abel **C**lassifier*

**ELM** ***E**xtreme **L**earning **M**achine*

**OS-ELM** ***O**nline **S**equential **E**xtreme **L**earning **M**achine*

**OSML-ELM** ***O**nline **S**equential **M**ulti-**L**abel **E**xtreme **L**earning **M**achine*

**PRO-EMLC** ***PRO**gressive-ELM **M**ulti-**L**abel **C**lassifier*

**PN** **P**adrão-**N**ovidade

**PS** ***P**runed **S**ets*

**SMART** ***S**treaming **M**ulti-l**A**bel **R**andom **T**rees*

**UnkRM** *Macro Unknown Rate*

# LISTA DE ALGORITMOS

# LISTA DE FIGURAS

# LISTA DE TABELAS

# Capítulo 1
# INTRODUÇÃO

## 1.1 Contextualização e Motivação

**F**luxos **C**ontínuos de **D**ados (FCDs), do inglês *Data Streams*, são sequências de dados de tamanho ilimitado, geradas de forma contínua e em ambientes dinâmicos. Além disso, esses dados não podem ser armazenados na memória devido ao fluxo de dados ser potencialmente infinito (GAMA, 2010). Uma das características mais desafiadoras dos FCDs é a potencial distribuição não estacionária dos dados, ou seja, podem ocorrer mudanças nos conceitos conhecidos e até mesmo novos conceitos podem surgir ao longo do tempo (FARIA et al., 2016a).

Com o advento da internet e de sistemas capazes de gerar grandes quantidades de dados em curtos espaços de tempo (*e.g.,* sistemas de segurança, redes de sensores usadas para agricultura ou em redes elétricas, mercado financeiro, redes de computadores, mineração de redes sociais e etc.), é notório a necessidade de desenvolver ferramentas capazes de extrair conhecimento relevante desses dados (GAMA; GABER, 2007; SILVA et al., 2013). A Mineração de Dados é um campo de pesquisa importante para auxiliar no desenvolvimento dessas ferramentas, que ajudam a entender e resolver os problemas encontrados em ambientes com grandes volumes de dados (GAMA, 2010). Na Mineração de Dados, algoritmos de Aprendizado de Máquina são utilizados para a aquisição automática de conhecimento. O Aprendizado de Máquina é uma subárea da Inteligência Artificial cujo intuito é estudar métodos para melhorar o desempenho de determinadas tarefas através de conhecimentos prévios, ou experiências anteriores (MITCHELL, 1997). Dentre as tarefas de Aprendizado de Máquina, a Classificação de dados destaca-se. Nela o objetivo é classificar novos exemplos (dados não rotulados) usando um modelo de decisão, construído em uma fase inicial de treinamento a partir de exemplos previamente rotulados (experiências passadas) (AGGARWAL, 2007). Nas tarefas de classificação, geralmente os exemplos analisados são monorrótulo, ou seja, cada exemplo é associado a um único rótulo $y$ de um conjunto disjunto de rótulos $L$. Se $|L| = 2$, então é um problema de classificação binária (*i.e.,* um problema envolvendo apenas duas classes distintas). Por outro lado, se $|L| > 2$, é um problema de classificação multiclasse. Entretanto, existem aplicações nas quais cada exemplo pode associar-se a mais de um rótulo simultaneamente (*i.e.,*$Y \subseteq L$). Nesse caso,

o problema é conhecido como **C**lassificação **M**ultirrótulo (CM) (TSOUMAKAS; KATAKIS, 2006). Na literatura é comum encontrar aplicações de Aprendizado Multirrótulo para solução de problemas como: Bioinformática (ZHANG; ZHOU, 2005; CLARE; KING, 2001; CERRI et al., 2009), Mineração na Web (KAZAWA et al., 2004; TANG et al., 2009), Recuperação da Informação (ZHU et al., 2005; YANG; GOPAL, 2012), *Tag Recommendation* (SONG et al., 2008; KATAKIS et al., 2008), e etc. Nos últimos anos pesquisas de Aprendizado Multirrótulo têm se focado em problemas dinâmicos que exigem análise em tempo real, como FCDs (TRAJDOS; KURZYNSKI, 2015; DAVE et al., 2016; VENKATESAN et al., 2017; OSOJNIK et al., 2017; AHMADI; KRAMER, 2018; ZHU et al., 2018; NGUYEN et al., 2019).

Para a construção de classificadores de FCDs, certas restrições devem ser seguidas (NGUYEN et al., 2015; GAMA, 2010):

- **resposta em tempo real:** o modelo de decisão deve ser rápido nas predições, pois o intervalo de chegada entre um exemplo e outro pode ser curto;
- **processar os dados uma única vez:** os exemplos devem ser processados e em seguida descartados, pois devido ao fluxo potencialmente infinito dos dados, o armazenamento dos exemplos se torna inviável;
- **detectar mudanças de conceito:** a distribuição dos dados é potencialmente não estacionária e tende a mudar com o passar do tempo. Portanto, o modelo de decisão deve ser atualizado para refletir tais mudanças;
- **memória de armazenamento limitada:** os fluxo de dados são potencialmente infinitos ou em larga escala, desse modo os dados não devem ser armazenados diretamente na memória. Geralmente, apenas uma sumarização dos dados é armazenada;

Aggarwal (2007) enfatiza que além dessas restrições, outras questões devem ser consideradas: tratamento de dados categóricos e dados complexos, desenvolvimento de técnicas de pré-processamento de baixo custo computacional e desenvolvimento de técnicas que lidem com a escassez de dados rotulados para atualização dos modelos de decisão.

A questão da escassez de dados rotulados é ainda mais importante. A maioria dos métodos de mineração de FCDs assume a disponibilidade dos rótulos reais dos exemplos sem nenhum atraso após a classificação. No entanto, esse cenário é irreal para maioria das aplicações, pois devido a fatores como altos custos envolvidos no processo de rotulagem, falhas no processo de transmissão dos rótulos, ou mesmo devido às características do processo gerador dos dados, as informações a respeito dos rótulos dos exemplos nem sempre estarão disponíveis. Problemas em que os rótulos reais dos exemplos nunca estarão disponíveis são conhecidos como *Latência Extrema de Rótulos* e a atualização dos modelos de decisão de forma supervisionada é inviável (SOUZA et al., 2015a; SOUZA et al., 2015b; KREMPL et al., 2014).

Uma característica presente em FCDs que é pouco explorado na literatura é a *Evolução de Conceito*. Esse fenômeno ocorre quando novas classes, não conhecidas pelo modelo de decisão, surgem ao longo do fluxo de dados (MASUD et al., 2011a). Adaptar modelos de decisão às Evoluções de Conceito é uma tarefa desafiadora, especialmente em problemas com Latência Extrema de Rótulos. Uma tarefa que têm se destacado ao lidar com Evoluções de Conceito e Latência Extrema de Rótulos é a **D**etecção de **N**ovidade (DN) (FARIA et al., 2016a).

A tarefa de DN consiste em identificar novos padrões em exemplos não rotulados. Em alguns aspectos, esses padrões podem ser divergentes dos observados previamente e, portanto, podem ser usados para atualizar os modelos de decisão (PERNER, 2009; FARIA et al., 2016a; GAMA, 2010).

Apesar das diferentes abordagens de DN para classificação em FCDs (SPINOSA et al., 2009; AL-KHATEEB et al., 2012a; MASUD et al., 2011a; FARIA et al., 2016b), a sua aplicação para **C**lassificação **M**ultirrótulo em **F**luxos **C**ontínuos de **D**ados (CMFCD) ainda é um problema pouco explorado na literatura, ainda para problemas como latência extrema de rótulos e evolução de conceito.

## 1.2 Hipótese e Objetivos

Este trabalho tem como objetivo principal desenvolver métodos de CMFCD com o auxilio de técnicas de DN, sendo esses métodos capazes de identificar mudanças e evoluções de conceito em ambientes de latência extrema de rótulos, uma suposição comum para problemas de mineração em FCDs.

Dado o objetivo principal desta pesquisa, a hipótese é apresentada da seguinte forma:

- Um método de CMFCD, que utiliza técnicas de DN para atualizar modelos de decisão em cenários de latência extrema de rótulos e para detectar evoluções de conceito sem perder desempenho na classificação das classes conhecidas e no tratamento de mudanças de conceito.

Formulada a hipótese desta pesquisa o objetivo principal foi concretizado por meio de uma adaptação do algoritmo ***Mult****I*class *learN*ing *A*lgorithm for data *S*treams (MINAS) (FARIA et al., 2016b) em conjunto com técnicas de *Transformação de Problemas* do aprendizado multirrótulo. Para isso, os seguintes objetivos específicos foram concluídos:

- Desenvolver, com auxilio de técnicas de aprendizado multirrótulo, uma fase de aprendizado supervisionada *offline* que receba uma pequena porção de exemplos multirrotulados e que seja capaz de construir modelos de decisão;

- Desenvolver uma fase de aplicação não supervisionada *online*, no qual o mecanismo de DN seja capaz de distinguir quando um exemplo é potencialmente pertencente às classes normais ou é potencialmente desconhecido. Ele também deve detectar *Padrões-Novidade* e atualizar o modelo de decisão sem os rótulos verdadeiros dos exemplos.
- Desenvolver uma metodologia de avaliação capaz de avaliar classificadores que utilizam DN para CMFCD.

O trabalho abordará alguns pontos importantes. São eles:

- Construção de modelos de decisão compostos por microgrupos rotulados. Devido à característica incremental e o fato de serem formados por agrupamento, com microgrupos é possível atualizar os modelos de decisão em cenários de latência extrema de rótulos;
- o uso da técnica ***B**inary **R**elevance* (BR) para a construção de múltiplos modelos de decisão, um para cada classe do problema;
- O uso de técnicas de combinação de rótulos, sendo elas a ***L**abel **P**owerset* (LP) e ***P**runed **S**ets* (PS), como alternativa para construir um único modelo de decisão composto por microgrupos multirrotulados;
- O uso de mineração de conjunto de itens frequentes para definir as combinações consideradas frequentes para o método PS;
- Tarefa de DN cujo objetivo é estender modelos de decisão para adaptá-los às mudanças de conceito, e criar novos modelos de decisão para adaptá-los às evoluções de conceito;
- Tarefa de DN alternativa cujo objetivo é criar novo microgrupos multirrotulados para adaptar o modelo de decisão somente às mudanças de conceito;
- Metodologia de avaliação para métodos de CMFCD através de DN, cujo objetivo principal é associar *padrões-novidade* às classes reais do problema.

## 1.3 Resumo dos Resultados

Neste trabalho foram desenvolvidos dois métodos ***Mult**I**-label** lear**N**ing **A**lgorithm for data **S**treams with **L**abel **C**ombination-based methods* (MINAS-LC) e ***Mult**I**-label** lear**N**ing **A**lgorithm for data **S**treams with **B**inary **R**elevance transformation* (MINAS-BR).

Considerando o MINAS-LC, duas variações foram propostas para construção do modelo de decisão: uma com o método LP para transformação de problema e outra com o método PS em conjunto com mineração de conjunto de itens frequentes. Através dessas técnicas, um único modelo de decisão, composto por microgrupos multirrotulados, foi construído.

Como o MINAS-LC trata apenas mudanças de conceito, foi discutido o seu desempenho em diferentes tipos de comportamentos de bases de dados com distribuição não estacionária com todas as classes são conhecidas na fase de treinamento. Experimentos com bases de dados reais, em que a distribuição dos dados é estacionária também foram executados. O MINAS-LC obteve resultados competitivos com os métodos da literatura, superando em alguns casos, até os métodos *upper bounds*.

Em relação ao MINAS-BR, o método utiliza o ***B**inary **R**elevance* (BR) para transformação de problema e constrói um modelo de decisão para representar cada classe do problema. Na DN, modelos de decisão são estendidos para contornar mudanças de conceito e, novos modelos de decisão são construídos para representar os *padrões-novidade*, tratando assim, evoluções de conceito.

O MINAS-BR trata mudanças de conceito e evoluções de conceito, portanto, os experimentos foram avaliados utilizando a metodologia proposta neste trabalho. Experimentos com bases de dados sintéticas e reais foram executados, mostrando o potencial do método na detecção de novas classes mantendo seu desempenho preditivo superior as dos métodos *lower bounds*.

## 1.4 Artigos Originados desta Pesquisa

Os artigos publicados são:

- Costa Junior, J. D., Faria, E. R., Silva, J. A., Cerri, R.: **Label powerset for multi-label data streams classification with concept drift**. In: 5th Symposium on Knowledge Discovery, Mining and Learning, Uberlândia, Brazil, p. 97–104, 2017.
- Costa Junior, J. D., Faria, E. R., Silva, J. A., Gama, J., Cerri, R.: **Pruned Sets for Multi-Label Stream Classification without True Labels**. In: International Joint Conference on Neural Networks, Budapest, Hungary, 14-19 July 2019. (Aceito para publicação).
- Costa Junior, J. D., Faria, E. R., Silva, J. A., Gama, J., Cerri, R.: **Novelty Detection for Multi-label Stream Classification**. In: 8th Brazilian Conference on Intelligent Systems, Salvador, Brazil, 15-18 October 2019. (Aceito para publicação).

Artigo em avaliação:

- Costa Junior, J. D., Faria, E. R., Silva, J. A., Gama, J., Cerri, R.: **Novelty Detection for Multi-label Stream Classification with Infinitely Delayed Labels**. In: IEEE International Conference on Data Mining (ICDM).

Além dessas publicações, um artigo descrevendo o método MINAS-LC e seus experimentos realizados neste trabalho está sendo redigido para ser submetido a um periódico internacional.

## 1.5 Organização do Documento

O restante do documento está organizado da seguinte forma:

- Capítulo 2 - Fluxos Contínuos de Dados: apresenta uma visão geral sobre FCDs. Também são mostrados os principais problemas envolvendo as tarefas de classificação e agrupamento em FCDs.
- Capítulo 3 - Classificação Multirrótulo: mostra uma visão geral da CM, principais desafios a serem tratados, diferentes abordagens para tratar o problema em cenários envolvendo FCDs e exemplos de aplicações.
- Capítulo 4 - Detecção de Novidade em FCDs: apresenta a tarefa de DN em FCDs, mostrando características, conceitos e os principais métodos encontrados na literatura.
- Capítulo 5 - Métodos de DN para CMFCD: descreve os métodos propostos neste trabalho – MINAS-LC e MINAS-BR – desenvolvidos para CMFCD em cenários com latência extrema de rótulos.
- Capítulo 6 - Experimentos: descreve os detalhes sobre os experimentos realizados para validar os métodos propostos neste trabalho.
- Capítulo 7 - Conclusões: discute os principais resultados obtidos durante este trabalho de pesquisa, as principais deficiências dos métodos propostos, bem como trabalhos futuros.

# Capítulo 2

## FLUXOS CONTÍNUOS DE DADOS

Aprendizado de Máquina é uma aplicação da Inteligência Artificial que fornece aos sistemas a capacidade de aprender automaticamente a partir de alguma experiência, ou seja, sem serem explicitamente programados (MITCHELL, 1997). Durante muito tempo o aprendizado de máquina manteve seu foco voltado a algoritmos de aprendizado *offline* (*batch learning*), em que os dados de treinamento estão sempre disponíveis e, se necessário, os dados podem ser reanalisados para a geração dos modelos de decisão. Esses modelos são estáticos e não é atualizado com novos dados (GAMA, 2010). No entanto, os avanços recentes de *hardware* e*software* têm possibilitado cada vez mais a aquisição de dados em larga escala e em alta velocidade, o que dificulta o uso de algoritmos de aprendizado *offline* (SILVA et al., 2013). Dados levantados pela plataforma *Domo*, trazem uma pequena amostra da gama de informações geradas a cada minuto do dia, no ano de 2019: cerca de 511 mil *tweets*; 694 mil horas de vídeos assistidos na *Netflix*; 4.5 milhões de vídeos assistidos no *Youtube*; e 277 mil *stories* postados no *Instagram*; mais de 4,5 milhões de Gigabytes de dados de internet são usados pelos americanos [1]. Além disso, há uma crescente popularização de sensores e dispositivos de medições capazes de gerar dados contínuos, em grande volume e em alta velocidade(SOUZA et al., 2015a). Lidar com esse tipo de informação é uma tarefa desafiadora, desse modo, é evidente o interesse em técnicas que lidam com sequências infinitas de dados geradas continuamente, conhecidas como **F**luxos **C**ontínuos de **D**ados (FCDs) (AGGARWAL, 2007; GAMA; GABER, 2007).

FCDs podem ser definidos como um conjunto $D$ contendo exemplos $X_1, X_2, \ldots, X_n, \ldots$, potencialmente infinito $(n \rightarrow \infty)$, cujos dados chegam nos marcadores de tempo $t_1$, $t_2$,..., $t_n$,.... e cada exemplo é representado por atributos $d$-dimensionais $X_i = \{x_1, x_2, \ldots, x_d\}$ (AGGARWAL et al., 2003).

As principais características dos FCDs são (BABCOCK et al., 2002; AGGARWAL, 2007; GAMA, 2010):

[1] Fonte: https://www.domo.com/learn/data-never-sleeps-7

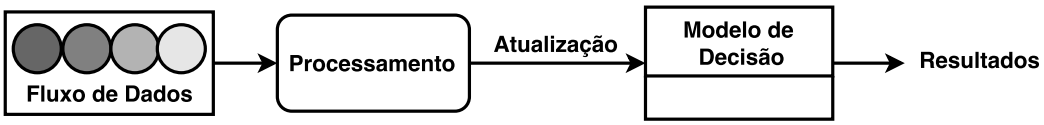
- dados potencialmente ilimitados;
- geração contínua de dados, e em geral, em alta velocidade;
- geração de dados sem um padrão de ordem de chegada dos exemplos;
- sistema para predição dos dados, deve ser apto a fazer predições a qualquer momento;
- inviável o armazenamento dos dados na memória, sendo assim, um exemplo deve ser processado uma única vez, ou apenas uma sumarização dos dados deve ser armazenada;
- distribuição de dados potencialmente não estacionária, ou seja, as características dos dados podem mudar com o passar do tempo.

Estas características impedem a aplicação de algoritmos tradicionais de aprendizado de máquina. Portanto, novas estruturas de dados e técnicas para mineração em FCDs são necessárias (MAHDIRAJI, 2009). Aggarwal (2007) destaca dois principais desafios da mineração em FCDs: processar os dados de forma rápida e incremental; lidar com mudanças na distribuição do dados. Em relação ao processamento rápido e incremental dos dados, algoritmos tradicionais como o k-Vizinhos mais próximo (KNN) e o *Naive Bayes* , são por natureza incrementais. No entanto, outros algoritmos como Árvores de Decisão e *Support Vector Machines* (SVM) devem ser adaptados, por não apresentarem essas características. Segundo Gama (2010), algoritmos incrementais são necessários, porém não são suficientes, isso por conta da distribuição não estacionária dos dados, também conhecida na literatura como *Mudanças de Conceito*. Assim, para lidar com dados não estacionários os algoritmos de mineração de dados devem ter mecanismos capazes detectar e adaptar seus modelos de decisão a tais mudanças, não apenas inserindo novas informações, mas também eliminando informações desatualizadas (NGUYEN et al., 2015).

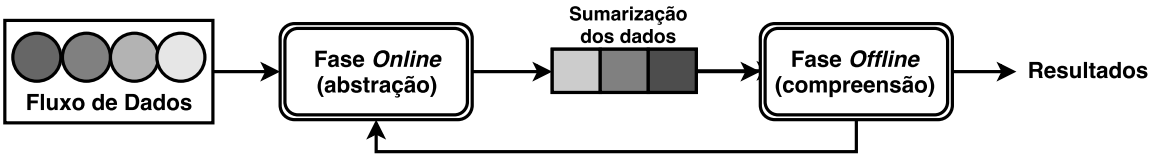
Na literatura, as abordagens mais comuns utilizadas para mineração em FCDs são o *Aprendizado em Uma Fase* e o *Aprendizado em Duas Fases*. No Aprendizado em Uma Fase, ilustrado na Figura 1a, todo o processo de mineração de dados (criação do modelo, processamento dos dados, atualização do modelo) ocorre exclusivamente *online* (NGUYEN et al., 2015).

O Aprendizado em Duas Fases, também conhecido como Aprendizado *Online-Offline* (Figura 1b), divide o processo de mineração de dados em duas fases: Fase *Online*, responsável pela abstração dos dados e Fase *Offline*, em que algoritmos tradicionais podem ser usados para extração de informações. Na fase *Online*, estruturas de sumarização são aplicadas nos exemplos que chegam através do fluxo de dados. Essas estruturas têm como objetivo preservar o significado dos exemplos originais sem armazená-los completamente na memória (SILVA, 2015). Mais informações sobre essas estruturas estão presentes na Seção 2.1. A fase *Offline*, na maior parte dos casos, é executada de acordo com a necessidade da aplicação, podendo ser através da solicitação do usuário ou de tempos em tempos seguindo algum parâmetro pré-definido, por exemplo. Nessa fase, as estruturas obtidas na fase *online* são utilizadas para atualização e/ou

Figura 1 – Visão geral das abordagens utilizadas na mineração em FCDs. Adaptado de (NGUYEN et al., 2015)




(a) Abordagem Uma Fase: O algoritmo recebe o próximo exemplo do fluxo, processa e atualiza o modelo seguindo as restrições impostas pelas características dos FCDs. O componente *Processamento*, diz respeito ao tratamento do exemplo realizado por cada método em particular, seja a criação ou atualização de uma estrutura de dados, ou um cálculo das probabilidades dos exemplos. Os resultados podem ser obtidos assim que o exemplo é processado.




(b) Abordagem Duas Fases (*online-offline*): Na fase *Online* à medida que os exemplos chegam, a estrutura de sumarização é atualizada. Na fase *Offline*, um algoritmo é executado usando a estrutura de sumarização. Com isso é possível a extração de informações.

construção de classificadores, definição de grupos, definição de superfícies de decisão, e etc. Técnicas de agrupamento (AGGARWAL et al., 2003; CAO et al., 2006; LOPES; CAMARGO, 2017) e/ou classificação (AGGARWAL, 2007; RAI et al., 2009) são comumente utilizadas para este propósito.

A atualização dos modelos de decisão também é uma característica importante para criar técnicas de mineração em FCDs. Existem duas abordagens para atualização dos modelos (READ et al., 2012b):

- *Bloco-incremental:* o fluxo de exemplos é divido em blocos (*chunks*). Os exemplos que chegam ao longo do fluxo são armazenados em um *buffer* e assim que esse *buffer* alcança seu limite os exemplos são utilizados na construção ou atualização do modelo. Suas principais desvantagens são: *i)* é dependente da definição do tamanho do bloco de exemplos; *ii)* em muitos casos, é forçado a eliminar modelos já treinados para dar lugar a novos modelos. Dessa maneira os métodos se adaptam as mudanças de conceito, porém ao eliminar esses modelos, informações importantes podem ser descartadas. Essa característica tem um impacto negativo na habilidade de aprender um conceito por completo; *iii)* não é capaz de aprender a partir de novos exemplos até que o bloco esteja completo, o que afeta a sua capacidade de responder a um novo conceito;

- *Exemplo-incremental:* os modelos são construídos e/ou atualizados de maneira puramente incremental, ou seja, são atualizados à medida que os exemplos aparecem ao longo do fluxo. Desvantagens da abordagem: *i)* poucos métodos foram desenvolvidos em relação aos

métodos que podem ser utilizados na abordagem bloco-incremental; *ii)* em sua maioria, são dependentes de métodos auxiliares para eliminar a influência de dados desatualizados ou adaptar-se a mudanças de conceito, como janelas de tempo ou métodos de monitoramento de mudanças.

Agrupamento e Classificação de Dados são duas das principais tarefas de mineração de dados. Para cada uma dessas, novos algoritmos foram desenvolvidos para trabalhar com FCDs. Neste capítulo serão apresentadas informações sobre essas tarefas, trazendo suas principais características em ambientes dinâmicos. A Seção 2.1 traz um resumo sobre os algoritmos de agrupamento em FCDs mais tradicionais da literatura, bem como a definição das estruturas de sumarização utilizadas por eles e a Seção 2.2 apresenta os desafios a serem considerados na classificação em FCDs, bem como as técnicas mais comuns da literatura.

## 2.1 Agrupamento em Fluxos Contínuos de Dados

Segundo Nguyen et al. (2015), Agrupamento de Dados é a tarefa de separar os dados em diferentes grupos, levando em consideração as suas características semelhantes. Gama (2010) destaca que o principal desafio do agrupamento em FCDs é manter continuamente grupos bons (de acordo com um critério de validação de agrupamento) e consistentes seguindo as restrições memória e tempo. O que traz essa dificuldade é a alta velocidade e a distribuição não estacionária dos dados.

Segundo Silva et al. (2013), para contornar o problema de memória e espaço limitado preservando os dados originais, estruturas de sumarização são utilizadas. As estruturas mais comuns da literatura são: Vetor de Características, Vetor de Protótipos, Árvore de Conjunto de Objetos Representativos, e Malha. Este capítulo aborda apenas os Vetores de Características, pois será a estrutura utilizada no trabalho. Uma descrição detalhada das demais estruturas de sumarização pode ser encontrada em (SILVA et al., 2013).

Vetores de Características, do inglês ***C**luster **F**eature **V**ectors* (CFVs), sumarizam os exemplos de um grupo mantendo uma tripla (ZHANG et al., 1996):

- $n$: número de exemplos no grupo;
- $LS$: soma linear dos $n$ exemplos do grupo;
- $SS$: soma quadrada dos $n$ exemplos do grupo.

Com esses três componentes é possível calcular o *centroide* e o *raio* dos grupos:

$$\textit{Centroide} = \frac{LS}{n} \qquad (2.1) \qquad \textit{Raio} = \sqrt{\left(\frac{SS}{n}\right) - \left(\frac{LS}{n}\right)^2} \qquad (2.2)$$

CFVs possuem propriedades de *Aditividade* e *Incrementabilidade* (GAMA, 2010):

- *Aditividade* - $CF1$ e $CF2$ são CFVs de dois grupos disjuntos.

$$CF1 + CF2 = (n_1 + n_2, LS_1 + LS_2, SS_1 + SS_2) \qquad (2.3)$$

- *Incrementalidade* - X é um exemplo adicionado ao CFV.

$$\begin{aligned} LS &\leftarrow LS + X \\ SS &\leftarrow SS + X^2 \\ n &\leftarrow n + 1 \end{aligned} \qquad (2.4)$$

CFVs para sumarizar grandes quantidades de dados foi introduzido no algoritmo BIRCH (*Balanced Iterative Reducing and Clustering using Hierarchies*) (ZHANG et al., 1996) e aplicado por diferentes algoritmos da literatura, como no *Scalable k-Means* (BRADLEY et al., 1998), que usa CFVs para aumentar a escalabilidade do algoritmo *k-Means* para grandes bases de dados, e no *Single-Pass k-Means* (FARNSTROM et al., 2000), uma simplificação do algoritmo anterior com o objetivo de reduzir sua complexidade computacional (SILVA et al., 2013).

Em Aggarwal et al. (2003), os autores apresentam o algoritmo *CluStream*, que usa uma extensão dos CFVs denominada *microgrupos*. Cada Microgrupo armazena, além das três informações do CFV, a soma dos marcadores de tempo ($CF1^t$) e a soma quadrática dos marcadores de tempo ($CF2^t$). Semelhante ao *CluStream*, o *SWClustering* (ZHOU et al., 2008) utiliza uma variante temporal do CFV, conhecida como *Temporal Cluster Features*, que além dos componentes originais do CFV, possui um marcador $t$ que registra o tempo do último exemplo inserido em um *Temporal Cluster Features*.

Outros dois algoritmos tradicionais de agrupamento em FCDs que utilizam o conceito de CFVs são o *DenStream* (CAO et al., 2006) e o *ClusTree* (KRANEN et al., 2011). No *DenStream*, para manter e distinguir entre microgrupos e ruídos, são construídas estruturas de *core-micro-cluster* e *outlier-micro-cluster*. Cada uma dessas estruturas possui um peso associado para indicar sua importância em relação ao tempo. O *ClusTree* também usa CFVs com pesos, o método mantém os CFVs em uma estrutura hierárquica (árvore da família *R-tree*).

Tabela 1 – Exemplos rótulos no formato atributo-valor. Adaptado de Alvares-Cherman et al. (2012)

| X | $x_1$ | $x_2$ | ... | $x_d$ | L |
|---|---|---|---|---|---|
| $X_1$ | $x_{11}$ | $x_{12}$ | ... | $x_{1d}$ | $y_1$ |
| $X_2$ | $x_{21}$ | $x_{22}$ | ... | $x_{2d}$ | $y_2$ |
| ⋮ | ⋮ | ⋮ | ⋱ | ⋮ | ⋮ |
| $X_n$ | $x_{n1}$ | $x_{n2}$ | ... | $x_{nd}$ | $y_n$ |

(a) Monorrótulo

| X | $x_1$ | $x_2$ | ... | $x_d$ | L |
|---|---|---|---|---|---|
| $X_1$ | $x_{11}$ | $x_{12}$ | ... | $x_{1d}$ | $Y_1$ |
| $X_2$ | $x_{21}$ | $x_{22}$ | ... | $x_{2d}$ | $Y_2$ |
| ⋮ | ⋮ | ⋮ | ⋱ | ⋮ | ⋮ |
| $X_n$ | $x_{n1}$ | $x_{n2}$ | ... | $x_{nd}$ | $Y_n$ |

(b) Multirrótulo

## 2.2 Classificação em Fluxos Contínuos de Dados

Classificação de Dados é uma técnica de aprendizado supervisionado cujo objetivo é criar modelos de decisão de acordo com exemplos rotulados disponíveis (conjunto de treinamento), de maneira que, esses modelos possam ser usados na predição de rótulos para novos exemplos (conjunto de teste) (HAN et al., 2011).

A classificação de dados pode ser definida da seguinte maneira: dado um conjunto de exemplos $D = \{X_1, \ldots, X_n\}$, com $X_i = \{x_1, \ldots, x_d\}$, tal que $x_j$ refere-se ao valor do $j$-ésimo atributo do exemplo $X_i$. Um problema de classificação monorrótulo consiste em atribuir um único rótulo $y_i$ a um exemplo $X_i$, em que $y_i$ pertencente ao conjunto de rótulos $L = \{y_1, \ldots, y_q\}$, sendo $q$ o número de rótulos de classes do problema. Quando $q = 2$ (ou $|L| = 2$), o problema é denominado classificação binária, ao passo que, se $q > 2$, o problema é denominado classificação multiclasse (Tabela 1a).Existe ainda a **C**lassificação **M**ultirrótulo (CM), problema no qual exemplos podem estar associados a mais de um rótulo simultaneamente, *i.e.*, um exemplo $X_i$ está associado a um conjunto de rótulos $Y_i$ e é representado na forma $(X_i, Y_i)$, sendo $Y_i \subseteq L$ (Tabela 1b) (ALVARES-CHERMAN et al., 2012; TSOUMAKAS et al., 2009).

Na classificação em FCDs o conjunto de dados de treino e de teste podem chegar na forma de fluxos e o objetivo é predizer, em tempo real, a classe dos exemplos não rotulados que chegam constantemente (PAIVA, 2014). Por conta das características particulares dos FCDs, algoritmos comuns de aprendizado *offline* não são capazes de classificá-los adequadamente, principalmente por ser necessário atualizar o modelo de decisão. Segundo Aggarwal (2007), atualizar o modelo de decisão é o maior desafio da classificação em FCDs, pois mudanças de conceito podem ocorrer. Na **C**lassificação **M**ultirrótulo em **F**luxos **C**ontínuos de **D**ados (CMFCD), esse fenômeno é ainda mais desafiador. Com as mudanças na distribuição dos dados, outras características presentes nos dados multirrótulo, como a cardinalidade e a dependência de rótulos, também podem mudar. Essas são características fundamentais para a indução de modelos de decisão quando se trata de aprendizado multirrótulo (READ et al., 2009).

Técnicas tradicionais do cenário *batch* (aprendizado *offline*) para classificação de dados não são adequadas para FCDs. Porém, adaptações foram desenvolvidas na tentativa de adequá-las

as restrições impostas por esse cenário. As mais tradicionais são: árvores de decisão, KNN e redes neurais. Além disso, é comum usar Comitês de Classificadores, técnica que combina vários classificadores sob um esquema de votação para encontrar a predição final (NGUYEN et al., 2015). As próximas seções trazem uma revisão dos principais algoritmos encontrados na literatura para classificação monorrótulo em FCDs baseadas nessas técnicas.

### 2.2.1 Árvores de Decisão

Os métodos baseados em árvores de decisão são algoritmos estado-da-arte na classificação em FCDs. O primeiro algoritmo desenvolvido para este fim foi a *Very Fast Decision Tree* (DOMINGOS; HULTEN, 2000). Esse algoritmo usa *Árvores de Hoeffding* e seguem o princípio de que com uma pequena amostra dos dados é possível escolher atributos para expansão dos nós da árvore. Para isso, o *Limiar de Hoeffding* é definido para quantificar o número de exemplos necessários para estimar a relevância dos atributos. Assim, dadas $n$ observações independentes de uma variável aleatória $r$ com alcance $R$ e média $\bar{r}$, o *Limiar de Hoeffding* garante que a média real de $r$ será pelo menos $\bar{r} - \epsilon$ com probabilidade $1 - \delta$, sendo $\delta$ um valor especificado pelo usuário e $\epsilon$ é dado pela Equação 2.5. Assim, a cada novo exemplo, o algoritmo usa o *Limiar de Hoeffding* para checar se o atributo candidato à divisão possui confiança o suficiente para gerar um próximo nível na árvore.

$$\epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}} \tag{2.5}$$

Ainda que a Very Fast Decision Tree alcance resultados significativos com uma única varredura nos dados, o método não é capaz de se adaptar as mudanças de conceito, pois, uma vez criado, um nó não pode ser alterado. Em Hulten et al. (2001), os autores apresentam a *Concept-adapting Very Fast Decition Tree*, método que usa janelas deslizantes de exemplos, para minimizar o efeito de dados desatualizados. Quando uma subárvore se torna obsoleta, essa é substituída por outra subárvore que foi sendo construída à medida que a primeira perdia o desempenho preditivo. Outro método da família das Árvores de *Hoeffding*, é a *Hoeffding Adaptive Tree* (BIFET; GAVALDÀ, 2009), um algoritmo adaptativo, não paramétrico, capaz de adaptar-se as mudanças de conceito auxiliado pelo método ***AD**aptive **WIN**dowing* (ADWIN). O ADWIN mantém uma janela dinâmica de exemplos recém-chegados. Um fragmento mais antigo da janela é descartado se, e somente se, houver evidências suficientes de sua divergência entre o restante da janela. Esse método exige apenas um limiar de confiança para indicar o quão certo a saída do algoritmo é. Outra característica importante do ADWIN é usar uma variação da técnica de histograma exponencial. Com ela é possível manter uma janela de tamanho $w$ usando apenas $O(log\ w)$ de memória e $O(log\ w)$ de tempo de processamento por exemplo.

### 2.2.2 K-Vizinhos Mais Próximos

Xioufis et al. (2011) apresentam dois benefícios fundamentais do KNN aplicado à mineração em FCDs:

- KNN é incremental, assim é possível adicionar e remover exemplos sem a necessidade de reconstruir o modelo.
- KNN é um algoritmo que não faz suposições sobre a forma de distribuição dos dados, o que é útil em FCDs pois, na maioria das vezes, não se tem conhecimento da distribuição dos dados e essa distribuição sofre mudanças ao longo do tempo.

Nem sempre é possível controlar a taxa de chegada dos exemplos do conjunto de teste em um FCD. Desse modo, a classificação deve ser realizada de forma adaptável aos recursos disponíveis. Em particular, o fluxo precisa ser classificado usando uma abordagem de duas fases (*online-offline*) (AGGARWAL, 2014). Em Aggarwal et al. (2006), os autores propõem o *On-Demand-Stream*, algoritmo que usa KNN como classificador base, trazendo suas melhores características para um ambiente supervisionado (*i.e.* microgrupos e aprendizado *online-offline*). Nesse método, os microgrupos são formados separadamente para cada classe e, na fase *offline*, o processo de classificação se inicia com a busca da melhor janela de exemplos (também chamado de horizonte). Os microgrupos desse horizonte serão considerados como pseudopontos representados por seus centroides (Equação 2.1). A ideia de usar microgrupos ao invés de exemplos, é que existem menos microgrupos, portanto, a classificação poderá ser mais eficiente (em relação a tempo e memória utilizados). Para a classificação dos exemplos de teste, o método executa o KNN, que classifica exemplos de acordo com o rótulo dos $k$-microgrupos mais próximos.

Em Zhang et al. (2011) os autores apresentam uma estrutura de organização de micro-grupos (*exemplars* como é nomeado no artigo) denominada *Lazy-tree*. Inspirado na *M-tree* (CIACCIA et al., 1997), a *Lazy-tree* ajuda a obter uma grande redução de consumo de memória e uma complexidade do tempo sublinear para a classificação (de $O(m)$ para $O(\log m)$, sendo $m$ o número total de *exemplars* na *Lazy-tree*). A árvore possui três principais operações: *i) busca:* usada para classificar os exemplos do fluxo; *ii) inserção:* usada para adicionar um novo nó na árvore; *iii) remoção:* usada para eliminar um nó obsoleto da árvore. As duas últimas operações garantem o balanceamento da árvore e a adaptação natural às mudanças de conceito.

### 2.2.3 Redes Neurais

Leite et al. (2009) apresentam um algoritmo de classificação em FCDs baseado em redes neurais granulares evolutivas (eGNN), do inglês *evolving Granular Neural Network*. Este algoritmo é uma variante evolutiva de redes neuro-*fuzzy* granulares que trata dados com distribuições não estacionárias. O eGNN é composto por duas fases: Na primeira, ele usa

neurônios T-S para agregar informações dos exemplos do fluxo e construir grânulos. Na segunda fase, esses grânulos são usados na construção da uma rede neural, eliminando a necessidade de manter todos os exemplos do conjunto de treino (o número de exemplos é bem superior ao número de grânulos). A associação de um rótulo a um grânulo é definido por hiper-retângulos *fuzzy*, que posteriormente evoluem para acomodar novos dados. Dentre as características principais do eGNN estão: *i)* o ajuste de sua estrutura e parâmetros para aprender um novo conceito, enquanto esquece o que se torna obsoleto; detecta mudanças no ambiente e lida com incerteza nos dados; *ii)* a habilidade não linear de separação de classes; *iii)* o desenvolvimento do aprendizado ao longo do tempo usando mecanismos construtivos *bottom-up* e destrutivos *top-down*. Em Leite et al. (2010) os autores estendem este trabalho para uma abordagem semisupervisionada, visando trabalhar com bases de dados parcialmente rotuladas.

### 2.2.4 Comitês de Classificadores

Segundo Bifet et al. (2009), uma maneira de obter melhor performance preditiva, escalabilidade e paralelização é através de comitês de classificadores, técnica que reúne um grupo de classificadores sob um esquema de votação para encontrar a predição final. Muitas pesquisas têm sido desenvolvidas com o objetivo de adaptar esses métodos para ambientes de FCDs.

Wang et al. (2003) propõem um comitê de classificadores ponderados pela acurácia (*Accuracy-weighted-Ensemble*), cujo objetivo é melhorar a detecção de mudanças de conceito. O algoritmo constrói e mantém um número fixo de $k$ classificadores, que podem ser C4.5, RIPPER ou *Naive Bayes*. Este algoritmo processa os dados de maneira bloco-incremental e treina um novo classificador em cada bloco de exemplos. O comitê é formado pelos $k$ melhores classificadores e seus pesos são definidos pelos respectivos resultados da acurácia obtidos na classificação.

O método mais popular desta linha são os *Online Bagging & Boosting* proposto em Oza (2005). O *Online Bagging* atribui a cada exemplo um peso de acordo com uma distribuição $Poisson(1) = exp(-1)/k$, sendo $k$ o tamanho do conjunto de dados de treino. Essa técnica pode ser considerada uma substituição do método de amostragem. Na fase de *Online Boosting*, os pesos do exemplos do fluxo e dos classificadores do comitê são ajustados de acordo com a taxa de erro dos classificadores do comitê atual. Em Bifet et al. (2009), os autores estendem este trabalho com a combinação do *Online Bagging* com o ***AD**aptive **WIN**dowing* (ADWIN) (BIFET; GAVALDA, 2007).

## 2.3 Considerações Finais

O objetivo deste capítulo foi apresentar uma visão geral sobre FCDs. Os principais desafios foram discutidos, bem como os tipos de abordagens de aprendizado no qual se baseiam os algoritmos de mineração em FCDs. Basicamente, esses algoritmos seguem duas abordagens: Aprendizado em Uma Fase e Aprendizado *online-offline* (Duas Fases). Essas abordagens ainda

seguem dois tipos de atualização de modelos, Exemplo-Incremental e Bloco-Incremental. Nas seções deste capítulo foram expostos os principais desafios de Agrupamento e de Classificação em FCDs. Sobre classificação, foi apresentada uma visão geral sobre os principais algoritmos de classificação monorrótulo. Para a CMFCD, características particulares do aprendizado multirrótulo impõe novos desafios para o aprendizado, sendo eles relacionados às mudanças de conceito, evoluções de conceito, mudanças na dependência e cardinalidade de rótulos, alto desbalanceamento de classes e etc. O próximo capítulo mostra uma revisão da literatura dos principais trabalhos sobre CMFCD e suas principais vantagens e desvantagens.

# Capítulo 3

## CLASSIFICAÇÃO MULTIRRÓTULO

**C**lassificação **M**ultirrótulo (CM) é uma generalização da tarefa de classificação monorrótulo em que os exemplos, ao invés de serem associados a um único rótulo, são associados a vários simultaneamente (TSOUMAKAS et al., 2009). Trata-se de uma tarefa de classificação comum aplicada a diversos problemas reais. Por exemplo: um documento pode ser classificado como da área da Ciência da Computação e Física Aplicada; uma notícia sobre a reação da igreja em relação ao filme Código DaVinci, pode ser tanto relacionada à Religião, quanto à Cultura e Cinema; uma imagem pode ao mesmo tempo ser rotulada como Praia, Montanha e Florestas. Em todos esses casos, mais de um rótulo é vinculado a cada exemplo simultaneamente (CARVALHO; FREITAS, 2009; TSOUMAKAS et al., 2009).

De acordo com Tsoumakas et al. (2009), os métodos de CM podem ser divididos em dois grupos:

- *Transformação de Problema*: também conhecido como métodos *Independentes de Algoritmo*. Consiste em transformar um problema multirrótulo em vários monorrótulo, o que possibilita o uso de algoritmos de classificação monorrótulo para a solução do problema.
- *Adaptação de Algoritmo*: ou métodos *Dependentes de Algoritmos*. Nesses métodos, novos algoritmos são desenvolvidos ou adaptados com o intuito de lidar diretamente com dados multirrotulados. A vantagem de usar essa abordagem é que ao concentrar-se em um algoritmo próprio para o problema, esse pode apresentar um melhor desempenho em aplicações reais (CARVALHO; FREITAS, 2009).

Segundo Read et al. (2012a) o uso de métodos baseados em transformação do problema é vantajoso, pois usando algoritmos incrementais monorrótulo, como classificadores base, e detectores de mudanças de conceito, qualquer método pode ser aplicado à FCDs. Os métodos de transformação de problema são baseados em duas abordagens principais: abordagem baseada no método ***B**inary **R**elevance* (BR) e abordagem baseada no método ***L**abel **P**owerset* (LP).

Nos algoritmos baseados no BR (TSOUMAKAS et al., 2009) o problema multirrótulo é divido em vários problemas binários, um para cada classe do problema. Apesar de ser simples, o

BR apresenta resultados significantes (em relação a predição) em diversos casos e, para cenários de FCDs, a sua baixa complexidade computacional, fácil paralelismo e resistência a *overfitting*, tornam-no ainda mais atrativo (READ et al., 2012a). Sua desvantagem é não considerar a dependência entre rótulos, característica intrínseca do aprendizado multirrótulo (TSOUMAKAS et al., 2009). Estratégias como o uso de comitês de classificadores BR e o método ***C**lassifier **C**hains* (CC), foram desenvolvidas para tratar esse problema, levando em consideração as dependências de rótulos nas soluções dos problemas de classificação (READ et al., 2011).

Os métodos baseados no ***L**abel **P**owerset* (LP) (TSOUMAKAS et al., 2009), transformam o problema multirrótulo em um problema monorrótulo multiclasse, em que cada combinação de rótulos presente no conjunto de dados é transformada em uma classe diferente e única (metaclasses). Sua principal vantagem é considerar a dependência entre rótulos, porém pode ser o pior caso de complexidade computacional. A complexidade computacional do método LP depende da complexidade do classificador base, do número de classes do problema e do número de exemplos, pois ao final da transformação, teremos um número máximo de metaclasses (classes após a transformação do problema) definido por $\liminf n, 2^{|L|}$. Apesar de normalmente o número de metaclasses ser menor do que esse limite, ainda representa um importante problema de complexidade, especialmente para bases de dados com um alto número de exemplos e de classes. Em relação à aplicação desses métodos em FCD, a principal dificuldade é tratar devidamente Evoluções de Conceito (READ et al., 2012a).

Uma alternativa para o problema de complexidade computacional da abordagem LP foi proposta em Read et al. (2008). Neste trabalho, os autores apresentam o método ***P**runed **S**ets* (PS), no qual as combinações de rótulos menos frequentes são eliminadas (podadas), diminuindo a complexidade do problema. Em uma segunda etapa do algoritmo, as combinações de rótulos eliminadas são divididas em subconjuntos menores e mais frequentes, o que torna possível retornar alguns exemplos ao conjunto de dados, garantindo perda mínima de informação. Sua principal desvantagem é não conseguir predizer combinações de rótulos não vistas no conjunto de dados de treino. Para evitar o *overfitting* no processo de eliminação de combinações de rótulos e permitir a criação de novas combinações no momento de classificar novos exemplos, os autores usam um comitê de classificadores PS.

Bases de dados multirrótulo apresentam características particulares em relação às monorrótulo, tendo como principais: *Cardinalidade rótulos* (Equação 3.1), que representa número médio de rótulos por exemplo; e *densidade de rótulos* (Equação 3.2), que calcula a média do número de rótulos por exemplo, ponderado pela quantidade total de rótulos presente no conjunto. Esses são fatores que ajudam a compreender os resultados dos métodos e a quantificar o número de rótulos associados a um exemplo. O mesmo classificador pode comportar-se de forma diferente em base de dados cuja cardinalidade de rótulo é a mesma, mas a densidade de rótulo é distinta. Ainda, é importante conhecer o número de rótulos distintos associados aos exemplos, pois também influencia na predição dos classificadores (TSOUMAKAS et al., 2009; ZHANG;

ZHOU, 2014).

A cardinalidade (CR) e densidade (DR) de rótulos são dadas pelas equações abaixo, em que $D$ representa um conjunto de exemplos:

$$CR(D) = \frac{1}{|D|}\sum_{i=1}^{|D|}|Y_i| \qquad (3.1) \qquad\qquad DR(D) = \frac{1}{|D|}\sum_{i=1}^{|D|}\left|\frac{Y_i}{|L|}\right| \qquad (3.2)$$

Outra característica importante a ser analisada em problema de CM são as *Dependências de Rótulos*. Bases de dados multirrótulo apresentam classes com certo grau de dependência entre si (*i.e., tendem a coocorrer mais vezes*), assim é indispensável entender, identificar e explorar essa característica, para construir modelos de decisão eficazes (CHERMAN, 2013).

Segundo Dembszynski et al. (2010), dois tipos diferentes de dependências podem ser identificados: *i) dependência incondicional*: um tipo global de dependência, independente de qualquer observação e que pode ser interpretada como a correlação direta entre rótulos; *ii) dependência condicional:* captura a dependência entre rótulos condicionada a um exemplo.

O objetivo deste capítulo é apresentar os conceitos fundamentais da CM e como a literatura lida com este tema em cenários de FCDs. Assim, este capítulo é organizado da seguinte maneira: a Seção 3.1 traz diversas medidas propostas para avaliar predições dos modelo métodos de aprendizado multirrótulo e a Seção 3.2 apresenta os principais métodos desenvolvidos para a CMFCD. Nesta seção são abordadas as vantagens e desvantagens dos métodos e as lacunas existentes nos trabalhos da literatura.

## 3.1 Avaliação do Aprendizado Multirrótulo

Diferente da classificação monorrótulo, na qual os exemplos podem assumir somente dois tipos de classificação: certo ou errado; na CM, a classificação de um exemplo pode estar parcialmente correta, ou seja, pode acertar a classificação de uma de suas classes relevantes, mas errar na classificação das outras, e vice-versa. Dessa forma, são necessárias medidas de avaliação específicas ou a adaptação de medidas monorrótulo para uma avaliação justa (ZHANG; ZHOU, 2014). Para a definição dessas medidas, considere $H$ um classificador multirrótulo, com $Z_i = H(X_i)$ sendo o conjunto de rótulos preditos por $H$, dado o exemplo $X_i$; $Y_i$ o conjunto de rótulos reais do exemplo, em que $Y_i = \{y_1, y_2, \ldots, y_{|L|}\}$; e $D = \{(X_1, Y_1), \ldots, (X_i, Y_i)\}$ como um conjunto de dados.

Diversas medidas específicas foram propostas para a avaliação da CM. Segundo Gibaja e Ventura (2015) essas medidas podem ser divididas em dois grupos: medidas para avaliar bipartições e medidas para avaliar *rankings*. As medidas do primeiro grupo avaliam os classificadores que trazem como resultado um conjunto $Z_i = \{z_1, z_2, \ldots, z_{|L|}\}$, em que seus elementos repre-

sentam os rótulos relevantes para o exemplo. Por outro lado, as medidas que avaliam *rankings*, são usadas para algoritmos de aprendizado que fornecem, além da bipartição, uma função que retorna valores (*scores*) para a ordenação dos rótulos conforme sua relevância. Ao rótulo mais relevante é atribuída a primeira posição no *ranking*, e ao menos relevante a última posição no *ranking*.

### 3.1.1 Medidas para Avaliação de Bipartições

Essas medidas podem ser agrupadas em (GIBAJA; VENTURA, 2015):

- *Baseadas em exemplo:* as medidas calculam a eficácia do classificador em relação a cada exemplo multirrótulo de maneira individual para depois calcular a média sobre todos os exemplos.
- *Baseadas em rótulo:* medidas de avaliação monorrótulo binárias são aplicadas para cada rótulo e uma média geral é posteriormente calculada. Nesse caso, a média sobre os valores obtidos para cada rótulo pode ser calculada da maneira *Micro* ou *Macro*.

Cada um dos grupos pode ser subdividido em medidas que avaliam a qualidade da classificação e medidas que aferem a qualidade da ordenação (*ranking*) dos exemplos. A Seção 3.1.1.1 traz a definição das medidas baseadas em exemplo e a Seção 3.1.1.2 a definição das medidas baseadas em rótulo.

#### 3.1.1.1 Medidas Baseadas em Exemplos

Uma medida tradicional utilizada para a avaliação de classificadores Multirrótulo é a *Hamming-Loss* (Equação 3.3) (TSOUMAKAS; VLAHAVAS, 2007). Nessa medida, $\Delta$ representa a diferença simétrica entre dois conjuntos e quanto mais próxima de zero, melhor é a predição do classificador.

$$\textit{Hamming-Loss}(H, D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i \,\Delta\, Z_i|}{|L|} \tag{3.3}$$

A medida *Subset-Accuracy* é definida pela Equação 3.4, em que $I$ é uma função que mapeia verdadeiro em $1$ e falso em $0$. *Subset-Accuracy* não leva em consideração resultados parcialmente corretos, ou seja, o resultado só será $1$ se todos os rótulos relevantes do exemplo forem classificados corretamente.

$$\textit{Subset-Accuracy}(H, D) = \frac{1}{|D|} \sum_{i=1}^{|D|} I(Z_i = Y_i) \tag{3.4}$$

Tradicionais medidas monorrótulo foram adaptadas para o contexto multirrótulo, como a Acurácia (Equação 3.5), que representa os acertos obtidos da predição dos rótulos verdadeiros e considera o fato de não predizer um rótulo que deveria ser predito; Precisão (Equação 3.6), que considera apenas se os rótulos preditos (positivamente) realmente deveriam ser preditos e os rótulos não preditos não interferem no valor da medida; Revocação (Equação 3.7), que considera tanto os rótulos preditos, quanto os que não foram preditos; *F-measure* (F1) (Equação 3.8), que representa a média harmônica entre Precisão e Revocação.

$$Acuracia(H, D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|} \tag{3.5}$$

$$Precisao(H, D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i \cap Z_i|}{|Z_i|} \tag{3.6}$$

$$Revocacao(H, D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i \cap Z_i|}{|Y_i|} \tag{3.7}$$

$$\text{F1} = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{2 \times |Y_i \cap Z_i|}{|Y_i| + |Z_i|} \tag{3.8}$$

#### 3.1.1.2 Medidas Baseadas em Rótulos

Para essa abordagem qualquer medida de avaliação binária pode ser usada. A ideia é avaliar cada classe individualmente contanto os valores de Verdadeiro Positivo (VP), False Positivo (FP), Verdadeiro Negativo (VN) e Falso Negativo (FN), de um total de $N = VP + FP + FN + VN$. Em função desses valores, as medidas de avaliação binárias são calculadas seguindo as Equações abaixo Read et al. (2016):

$$Precisao(H, D) = \frac{VP}{VP + FP} \tag{3.9}$$

$$Revocacao(H, D) = \frac{VP}{VP + FN} \tag{3.10}$$

$$Acuracia(H, D) = \frac{VP + VN}{N} \tag{3.11}$$

$$F1(H, D) = \frac{VP}{VP + \frac{FN+FP}{2}} \tag{3.12}$$

A média final das medidas podem ser calculadas de forma macro (*macro-averaged*) e micro (*micro-averaged*) (TSOUMAKAS et al., 2009). A *macro* calcula as medidas de avaliação para cada rótulo separadamente e então retorna a média sobre todas elas. Já a abordagem *micro* calcula o operador binário para cada exemplo e então retorna a média sobre todos os valores calculados. Assim é possível avaliar os métodos ponderados pelos exemplos (*micro*) e ponderado pelos rótulos (*macro*) (CHERMAN, 2013).

Seja $B(VP, VN, FP, FN)$ uma medida de avaliação binária, calculada em função de uma matriz de confusão. Sendo $VP_{y_i}, FP_{y_i}, VN_{y_i}$ e $FN_{y_i}$ os valores para o rótulo $y_i$ do conjunto de rótulos $L$. As versões de *macro-averaged* e *micro-averaged* da medida $B$ são definidas da seguinte maneira:

$$B_{macro} = \frac{1}{|L|} \sum_{i=1}^{|L|} B(VP_{y_i}, FP_{y_i}, VN_{y_i}, FN_{y_i}) \tag{3.13}$$

$$B_{micro} = B(\sum_{i=1}^{|L|} VP_{y_i}, \sum_{i=1}^{|L|} FP_{y_i}, \sum_{i=1}^{|L|} VN_{y_i}, \sum_{i=1}^{|L|} FN_{y_i}) \tag{3.14}$$

### 3.1.2 Medidas para Avaliação de *Rankings*

Segundo Gibaja e Ventura (2015), todas as medidas para avaliação de *ranking* são baseadas em exemplo, pois elas são calculadas para cada exemplo de teste e, em seguida, a média é calculada para todo o conjunto. O *ranking* do exemplo $X_i$ predito para um rótulo $y \in L$ é denotado por $r_i(y)$. Ao rótulo mais relevante é atribuída a primeira posição no *ranking*, e ao menos relevante a última posição no ranking.

A medida *One-Error*, representa quantas vezes o rótulo ranqueado no topo não pertence ao conjunto de rótulos verdadeiros ($Y_i$) do exemplo considerado. Essa medida varia no intervalo $[0, 1]$. Quanto menor o valor obtido, melhor é a predição:

$$\textit{One-Error} = \frac{1}{|D|} \sum_{i=1}^{|D|} I(\arg\max_{y \in L} r_i(y)) \tag{3.15}$$

Na medida *Coverage*, a ideia é identificar a posição no *ranking* antes da qual estão todos os rótulos que deveriam ser preditos. Quanto mais perto do topo do ranking for essa posição, melhor será o valor dessa medida, pois há menos erros na ordem dos rótulos verdadeiros. Quanto menor o valor de *Coverage*, melhor é a predição:

$$\textit{Coverage} = \frac{1}{|D|} \sum_{i=1}^{|D|} \max_{y \in L} (r_i(y) \ - \ 1) \tag{3.16}$$

A medida *Ranking-Loss* conta quantas vezes um rótulo $y \in Y_i$ (rótulo relevante) tem um *ranking* pior que um rótulo $y' \in L - Y_i$ (rótulo irrelevante). Quanto menor o valor da medida, melhor a predição:

$$\textit{Ranking-Loss} = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{1}{|Y_i||\overline{Y_i}|} |(y_a, y_b) : r_i(y_a) < r_i(y_b), (y_a, y_b) \in Y_i \times (L - Y_i)| \tag{3.17}$$

Por fim, a medida *Average-Precision* representa, em média, a fração de rótulos em $Y_i$ ranqueados acima de um determinado rótulo $y' \in L - Y_i$. O desempenho é perfeito quando o valor é 1.

$$\textit{Average-Precision} = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{1}{|Y_i|}\sum_{y\in Y_i}\frac{|(y' \in Y_i : r_i(y') \leq r_i(y)|}{r_i(y)} \tag{3.18}$$

## 3.2 Métodos para Classificação Multirrótulo em Fluxos Contínuos de Dados

Esta seção apresenta uma revisão dos principais métodos desenvolvidos para CMFCD. A Figura 2 ilustra uma taxonomia desses métodos, separando-os em dois grupos: Transformação de Problema e Adaptação de Problema.

### 3.2.1 Métodos de transformação de problema

Na Tabela 2 são sumarizadas as características principais dos métodos de transformação de problema para CMFCD, bem como suas referências na literatura, classificador base utilizado e a abordagem de atualização do modelo.

O primeiro trabalho a apresentar um algoritmo para CM em FCDs foi proposto por Qu et al. (2009). Nesse trabalho, os autores desenvolveram um comitê dinâmico de classificadores (***I**mproved **B**inary **R**elevance* (IBR)) baseado no método SVM-HF (*Support Vector Machines with Heterogeneous Feature Kernels*) (GODBOLE; SARAWAGI, 2004), método que traz melhorias para o tradicional BR ao adicionar características com as quais é possível tratar dependências de rótulos na classificação.

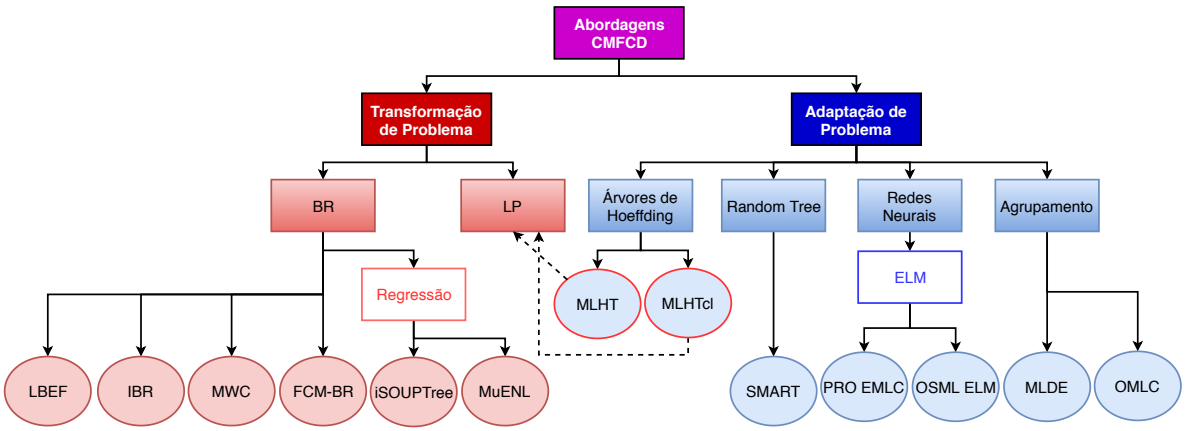



Figura 2 – Taxonomia dos algoritmos de CMFCD. Os retângulos representam as abordagens; as elipses representam os algoritmos; as setas com linhas contínuas representam especializações; e as setas com linhas tracejadas implementações, *i.e.*, o método faz parte de uma abordagem, mas em algum momento usa outra.

Tabela 2 – Métodos para CMFCD da abordagem de transformação de problema. Os acrônimos presentes no cabeçalho da tabela representam: **MC** - se o método detecta ou não **M**udanças de **C**onceito; **EC** - se o método detecta ou não **E**voluções de **C**onceito; **Dep** - se o método leva em consideração ou não as **Dep**endências entre rótulos; **RD** - se o método assume ou não que os **R**ótulos verdadeiros dos exemplos estarão sempre **D**isponíveis para a atualização do modelo

| Algoritmos | Referência | Classificador Base | Atualização | MC | EC | Dep. | RD |
|---|---|---|---|---|---|---|---|
| IBR | Qu et al. (2009) | *Naive Bayes* | bloco-incremental | sim | não | sim | sim |
| MWC | Xioufis et al. (2011) | KNN | exemplo-incremental | sim | sim | não | sim |
| LBEF | Wang et al. (2012) | Métodos Probabilísticos | bloco-incremental | sim | sim | não | não |
| FCM-BR | Trajdos e Kurzynski (2015) | Classificador Bayesiano | bloco-incremental | sim | não | sim | sim |
| iSOUPTree | Osojnik et al. (2017) | Redes Neurais | exemplo-incremental | não | não | sim | sim |
| MuENL | Zhu et al. (2018) | Regressão Linear | exemplo-incremental | não | sim | sim | não |

O IBR é bloco-incremental, assim, um fluxo de exemplos $D$, que possui um conjunto de classes $L = \{y_1, y_2, \ldots, y_m\}$, em que $m$ é o número total de classes do problema, é separado em blocos de mesmo tamanho $(S_1, S_2, \ldots, S_n)$, sendo $S_n$ o bloco de dados mais recente. Para cada bloco $S_i$, um conjunto $B$ de classificadores binários são treinados para cara classe do problema $(B_1, B_2, \ldots, B_{|L|})$, cada bloco $S_i$, tem como saída um bloco modificado $S'_i$, cuja saída dos classificadores $B$ são adicionados aos exemplos como atributos. Em seguida, o bloco $S'_i$ é usado para treinar um conjunto melhorado de classificadores $IB = \{IB_1, IB_2, \ldots, IB_{|L|}\}$.

Por ser bloco-incremental, qualquer classificador binário tradicional pode ser usado no método e o fato de substituir os classificadores antigos e possuir ponderação dinâmica faz o método eficiente na presença de mudanças de conceito, tanto abruptas quando graduais. Entretanto, além de possuir as desvantagens da abordagem bloco-incremental. Ainda, o método possui complexidade computacional alta, pois um conjunto de classificadores é construído e treinado em cada bloco de dados.

FCDs multirrótulo possuem múltiplos conceitos separados e é impraticável assumir que todos eles começam a mudar simultaneamente ou da mesma forma, ou seja, é provável que cada conceito apresente padrões de mudanças diferentes. Outro ponto importante relacionado à FCDs multirrótulo é a questão do desbalanceamento exacerbado de classes, pois, geralmente cada rótulo tem mais exemplos negativos do que positivos. Entretanto, isso não deve ser assumido como verdade para todos os rótulos do problema. Esses dois desafios são os que impulsionaram o desenvolvido do método ***M**ultiple **W**indows **C**lassifier* (MWC) (XIOUFIS et al., 2011). Neste trabalho os autores propõem o uso de múltiplas janelas de dados para manter um balanceamento entre exemplos positivos e negativos dos rótulos. Como o número de exemplos negativos geralmente é maior, a janela de exemplos negativos vai eliminando os exemplos mais antigos e os substituindo por exemplos mais recentes até que a janela de exemplos positivos esteja com exemplos o suficiente para o treinamento do classificador. Dessa maneira é possível reduzir a alta variância causada pelo número reduzido de exemplos positivos disponíveis para a classificação, construindo assim, modelos mais precisos.

A técnica de classificação utilizada neste método é baseada no BR, pois com ela é possível adequar as múltiplas janelas e as diversas mudanças de conceito entre os rótulos em momentos distintos. Além disso, como classificadores binários, os autores propõem uma melhoria no tradicional KNN: ao invés de encontrar o vizinho mais próximo dos exemplos de $Q^p_{y_j} \cup Q^n_{y_j}$ (janelas de exemplos positivos do rótulo $y_j$ e janelas de exemplos negativos do rótulos $y_j$, respectivamente), para cada rótulo $y_j$, eles calculam uma única vez a distância entre todos os exemplos em $S$ (*buffer* que armazena os exemplos compartilhados por todas as janelas) e os exemplos do conjunto de teste. A partir dessas distâncias, um ranking de rótulos é construído e o exemplo é classificado com os $t$ primeiros rótulos do ranking. O valor do limiar $t$ é dinâmico e varia de acordo com a cardinalidade de rótulo observada no último bloco de exemplos.

Apesar de ter trazido inúmeras contribuições e ter apresentado o primeiro método exemplo-incremental para CM em FCDs, esse método apresenta algumas restrições e desvantagens: *i)* é necessário certo conhecimento do fluxo de exemplos para definição do tamanho da janela de exemplos positivos, pois esse valor influência totalmente na adaptação dos classificadores às mudanças de conceito. Inclusive, em experimentos posteriores realizado em Read et al. (2012a), os autores mostraram que o algoritmo não obteve bons resultados com dados que apresentam tipos diferentes de mudanças de conceito; *ii)* é suscetível ao tamanho do bloco de exemplos para a definição do limiar $t$. Se o valor for pequeno pode não refletir a real cardinalidade de rótulo dos exemplos pertencentes aos conceitos atuais e se o valor for alto, consumirá muito espaço de memória para armazenar o bloco de exemplos; *iii)* não reconhece dependências entre rótulos; *iv)* assume que os rótulos verdadeiros dos exemplos estão sempre disponíveis após a classificação de um exemplo do conjunto de teste.

Em classificação de FCDs, a rotulagem dos dados para o treino do modelo é um problema desafiador, uma vez que o número de registros de um fluxo é sem precedentes e é impraticável rotular todos eles para o treinamento do modelo (WANG et al., 2012). Este problema é ainda mais agravante quando tratamos FCDs multirrótulo, pois o custo de rotulagem aumenta linearmente com o número rótulos. Pensando nisso e no problema do desbalanceamento de classes, em Wang et al. (2012) os autores apresentam o ***L**abel-based **E**nsemble **F**ramework for multi-label stream classification* (LBEF), um comitê de classificadores treinados com os dados individuais de cada classe e que usa Aprendizado Ativo (*Active Learning*) para selecionar exemplos para rotulagem.

A motivação dos autores na construção de classificadores baseados nos rótulos, vem do fato que os comitês baseados em blocos de exemplos possuem duas limitações que prejudicam a predição final do comitê: *i)* classificadores binários construídos do mesmo rótulo $y_n$ compartilham o mesmo peso, isso pode afetar negativamente a performance do comitê; *ii)* se um rótulo $y$ não estiver presente em um bloco $S$, não é possível construir um classificar binário para este rótulo. Formalmente um comitê é construído a partir de exemplos com rótulo $y$ e é representado por $E_y = \{E_j \mid j = 1, \ldots, |L|\}$, sendo $E_j = \{B^j_i \mid i = n - k + 1, \ldots, k\}$ e $B^j_i$ um classificador binário do $i$-ésimo rótulo no $j$-ésimo bloco de exemplos e seu peso é

representado por $w_{i_j}$. $E_y(x, w)$ é a saída resultante para um exemplo $X$. Além disso, a função de erro do modelo pode ser denotada por $\ell(E_y(x, w), y)$. Por conta das mudanças de conceito, uma das dificuldades em se calcular a função de erro é estimar a probabilidade. Como alternativa, os autores inferem a distribuição de $X$ usando todos os exemplos em um bloco de exemplos $S$. Assim, a perda esperada de um classificador do comitê pode ser dada pela formula:

$$\epsilon = \frac{1}{|D|} \sum^{x \in D} \ell_L(E_y(x, w), y)). \tag{3.19}$$

Para o cálculo de uma função de erro, são necessários os rótulos dos exemplos. Entretanto, como dito anteriormente, é inviável rotular todos os exemplos de um fluxo. Assim, os autores calculam aproximadamente qual é a função de erro esperada com o uso da função *Minimal Classifier Uncertainty*, que seleciona os exemplos mais informativos para refinar continuamente o limite das classes combinando informação tanto dos pesos dos modelos do comitê, quanto dos conceitos que estão mudando. Para isso, os autores ajustam o peso de um classificador através da incerteza de cada exemplo em relação aos diferentes classificadores base. Isso é equivalente a calcular a votação por entropia (DAGAN; ENGELSON, 1995), e quanto mais votos um exemplo tiver, mais informação ele carrega para ajuste de pesos do comitê. Com as mudanças de conceito, a acurácia dos classificadores base tende a cair significativamente, prejudicando o ajuste dos pesos dos classificadores do comitê. Em outras palavras, o valor da votação por entropia de um exemplo é inadequado para diferenciar um novo limite de decisão que não aparecera antes. Para contornar a situação, os autores usam uma ponderação *Maximum a Posteriori* para atualizar, de forma contínua, o peso dos classificadores do comitê.

Atualizar o comitê de cada rótulo independentemente pode preservar o melhor classificador para cada rótulo, e com isso se adaptar melhor às mudanças de conceito. Essa é uma das principais contribuições do método. Além disso, esse foi o primeiro trabalho a se preocupar com custo de rotulagem do problema e a usar aprendizado ativo na CM em FCDs. Suas desvantagens vêm da atualização bloco-incremental.

Em Trajdos e Kurzynski (2015), os autores abordam o uso de RRC (*Random Raference Classifier*) e matriz de confusão *fuzzy* local, com o objetivo de melhorar a qualidade de um comitê de classificadores BR. O método é divido em duas partes: Na primeira, é executado um classificador BR e no segundo é realizado um procedimento de correção que emprega medidas de competência e competência cruzada para ajustar a saída do classificador BR. As avaliações da saída do classificador é feito utilizando a matriz de confusão *fuzzy*.

Para treinar o classificador, os autores usaram um classificador Bayesiano. Quando a métrica *Hamming-Loss* é empregado neste classificador, o resultado é uma decomposição da tarefa em um conjunto de regras binárias independentes, essas regras poderão ser usadas na fase de correção. A fase de correção é acionada sempre que o modelo precisa ser atualizado e consiste em um modelo estatístico de competência, no qual o índice é proporcional a probabilidade de

classificação correta, e competência cruzada, que segue a probabilidade de classificação errada. Para obter essa probabilidade os autores criaram um procedimento de estimativa baseado em uma matriz de confusão *fuzzy* local. A matriz é construída com os exemplos vizinhos ao exemplo $X$. Eles definem a vizinhança de $X$ como uma rede *Fuzzy* $N(X)$, cuja associação é descrita usando uma função potencial Gaussiana:

$$\mu_{N(X)}(z) = exp(-\beta\delta(\mathbf{z}, X)^2) \tag{3.20}$$

Sendo $\beta \in \Re^+$ e $\beta(\mathbf{z}, X)$ a distância Euclidiana entre o exemplo $\mathbf{z}$ e o exemplo $X$ (os autores atribuíram $\beta = 1$).

O procedimento de aprendizado incremental é realizado usando bloco de exemplos. O bloco é divido em dois subconjunto, um com 30% dos exemplos usados para realizar a atualização incremental do comitê de classificadores BR, enquanto o restante dos exemplos (70%), são usados para substituir os exemplos do conjunto de teste mais antigos.

A principal desvantagem do método é, por conta do desbalanceamento, o classificador acaba enviesado pela classe majoritária. Além disso, o método ainda não conta com nenhum mecanismo eficaz para tratamento de evolução de conceito.

Muitos trabalhos usam a abordagem de transformação de problema, que transforma um problema de CM em vários monorrótulos, porém o método proposto em Osojnik et al. (2017), denominado ***i**ncremental **S**tructured **OU**tput **P**rediction **Tree*** (iSOUPTree), traz uma abordagem diferente: eles transformam o problema de CM em um problema de Regressão *Multi-Target*.

Um problema de regressão multi-target é a generalização do problema de regressão tradicional, cujo objetivo é predizer os valores múltiplas variáveis continuas atribuídas a cada exemplo (APPICE; DZEROSKI, 2007). Existem muitos pontos semelhantes entre CM e regressão multi-target, principalmente em relação as dependências entre rótulos, a motivação principal para o desenvolvimento do método proposto.

A transformação é realizada em dois passos: no primeiro, assume-se que as múltiplas variáveis alvo de um problema multirrótulo são compostas por várias variáveis alvo binárias $\mathbf{y} = (y_1, \ldots, y_n | y_i \in \{0, 1\})$. Porém, quando convertidos para um problema de regressão, cada uma dessas variáveis poderá assumir qualquer valor, *i.e.*, $\mathbf{y} = (y_1, \ldots, y_n | y_i \in \Re^n)$. O segundo passo, é quando a predição *multi-target* é convertida em multirrótulo. Essa conversão é feita utilizando um limiar. O iSOUPTree é a combinação desta metodologia de conversão e do algoritmo FIMT-MT (IKONOMOVSKA et al., 2011) com *perceptrons* adaptativas no nós folhas. Ainda, para buscar uma melhor performance os autores usam comitês de iSOUPTrees baseados na abordagem *online bagging* (OZA, 2005).

O iSOUPTree obteve melhores resultados em métricas baseadas em *ranking*. No entanto, o método não trata mudanças de conceito e o seu custo computacional é alto, o que em aplicações do mundo real não é tão vantajoso quanto usar algoritmos mais simples para a classificação.

Em Zhu et al. (2018) os autores apresentam um método de detecção de anomalias para CMFCD, cujo objetivo principal é tratar evoluções de conceito em cenários de latência extrema de rótulos. O método, denominado ***Mu**lti-label learning with **E**merging **N**ew **L**abels* (MuENL), é divido em três procedimentos principais: *classificação*, realizada através regressão linear binária para cada rótulo, cuja função de erro minimiza a medida *Ranking-Loss*, mas levando em conta o ranking de pares de rótulo; *detecção*, em que uma extensão do método *Isolation Forest* em conjunto procedimentos de agrupamento é capaz de detectar exemplos anormais que possam representar o surgimento de uma nova classe; e *atualização*, um novo classificador é construído para cada classe emergente.

Apesar de ser o primeiro método de CMFCD a tratar evoluções de conceito em cenários de latência extrema de rótulos, o MuENL não trata mudanças de conceito. Quando ocorrem mudanças na distribuição das classes, o método *Isolation Forest* tende a considerar os exemplos com distribuição diferente como anomalia e, na fase de atualização, o método não é capaz de distinguir entre evoluções e mudanças de conceito.

### 3.2.2 Métodos de Adaptação de Problema

Na Tabela 3 são sumarizadas as características principais dos métodos de adaptação de problema para CMFCD, bem como suas referências na literatura, qual o classificador base utilizado e a abordagem de atualização do modelo.

Em Kong e Philip (2011), os autores propõem o ***S**treaming **M**ulti-l**A**bel **R**andom **T**rees* (SMART), um comitê de classificadores baseados em *Random Trees*. Resumidamente o SMART é divido em 3 partes: *i) Ranking*: no qual calcula-se a probabilidade de cada rótulo ser relevante ou não, assim como a probabilidade de correlação entre dois rótulos; *ii) Regressão*: no qual

Tabela 3 – Métodos para CM em FCD da abordagem de adaptação de problema. Os acrônimos presentes na primeira linha da tabela representam: **MC** - se o método detecta ou não **M**udanças de **C**onceito; **EC** - se o método detecta ou não **E**voluções de **C**onceito; **Dep** - se o método leva em consideração ou não as **Dep**endências entre rótulos; **RD** - se o método assume ou não que os **R**ótulos verdadeiros dos exemplos estarão sempre **D**isponíveis para a atualização do modelo

| Algoritmos | Referência | Classificador Base | Atualização | MC | EC | Dep. | RD |
|---|---|---|---|---|---|---|---|
| SMART | Kong e Philip (2011) | *Random Trees* | exemplo-incremental | sim | não | sim | sim |
| MLHT | Read et al. (2012a) | Árvores de *Hoeffding* | exemplo-incremental | sim | sim | sim | sim |
| MLHTcl | Shi et al. (2014) | Árvores de *Hoeffding* | exemplo-incremental | sim | sim | sim | sim |
| MLDE | Song e Ye (2014) | MLCC | bloco-incremental | sim | sim | sim | não |
| ELM | Dave et al. (2016) Venkatesan et al. (2017) | Redes Neurais | ambos | não | sim | sim | sim |
| OMLC | Nguyen et al. (2019) | Agrupamento | exemplo-incremental | sim | não | sim | não |

calula-se o número estimado de rótulos atribuídos a cada exemplo; *Classificação*: em que, a partir do ranking, os primeiros $t$ rótulos são atribuídos ao exemplo.

As *Random Trees* multirrótulo são construídas no inicio do fluxo de exemplos e o processo de atualização delas ocorre da seguinte maneira: a construção se inicia com os nós das árvores vazios. Quando um exemplo do conjunto de treino chega, cada nó atualiza duas estatísticas (*i)* as probabilidades de relevância de rótulos que atravessam esse nó, que é semelhante aos das árvores de decisão tradicionais; *ii)* número de rótulos estimados para aquele nó, semelhante as árvores de regressão.

Cada nó das árvores mantém as seguintes características:

- A contagem de cada rótulo $y_i$ presente no conjunto de rótulos dos exemplos do fluxo que passam pelo nó: $\mathbf{c} = (c_1, \ldots, c_m)$ e $c_j = \sum_{i=1}^{t} y_i^j$;
- O número de exemplos que passaram pelo nó: $n$;
- A soma do número de rótulos associados a cada exemplo: $\theta = \sum_{i=1}^{t} |Y_i|$;
- O último *timestamp* do exemplo que passou pelo nó: $t$.

$\mathbf{c}$, $\theta$ e $n$ são todas somas aditivas em diferentes pontos de dados, portanto, podem ser atualizados de forma eficiente ao longo dos fluxos de dados. Cada nó da árvore pode trazer dois tipos de saída para cada exemplo: *i)* a estimativa da relevância do rótulo: $Node.c/Node.n$; *ii)* estimativa do número de rótulos relevantes: $Node.\theta/Node.n$.

Para lidar com mudanças de conceito, os autores usaram um fator de esquecimento baseado no trabalho de Aggarwal et al. (2004), no qual o peso do exemplo $X_i$, no tempo $t$, é $w_i(t) = 2^{-(t-t_i)/\lambda}$. $\lambda$ é um parâmetro do modelo que indica a velocidade de esquecimento, quanto maior o valor de $\lambda$, mais lentamente o peso será decrementado.

O método é sensível a definição dos parâmetros $d$ (tamanho da árvore) e $n_T$ (quantidade de *Random Trees*). Entretanto, os autores notaram que os valores máximos necessários para esses parâmetros são $d = 20$ e $n_T = 10$. Em relação a detecção de mudanças de conceito, como a abordagem utilizada é sensível ao parâmetro $\lambda$, é necessário um conhecimento prévio do tipo de mudança que o fluxo sofrerá, *e.g.*, para mudanças abruptas o valor de $\lambda$ deverá ser menor, se for mudanças graduais o valor deverá ser maior. No entanto, se houver vários tipos de mudanças no mesmo fluxo, o algoritmo não se adaptará corretamente a todas elas. Outra desvantagem do método, é o fato do conjunto de rótulos do problema ter que ser conhecido previamente.

Em Read et al. (2010) e Read et al. (2012a), os autores apresentam o uso de Árvores de *Hoeffding* multirrótulo com classificadores base nos nós folhas (***Multi**Label **H**oeffding **T**ree with pruned sets at the leaves* (MLHT)). Os autores usaram a estratégia de cálculo de entropia proposta por Clare e King (2001) para construção da Árvores de *Hoeffding* multirrótulo, em

combinação com a técnica de transformação de problema PS, que foi usada nos nós folhas da Árvores de *Hoeffding*. A motivação dos autores para o uso do método PS, é o fato de poder ser usado em conjunto com classificadores incrementais, reconhecer dependências entre rótulos e não possuir complexidade computacional alta. Para isso, o processo de construção do modelo usa os primeiros $w$ exemplos, que são armazenados e usados para selecionar as $c$ combinações mais frequentes, que serão usada para construir o classificador. Esse classificador é continuamente atualizado até que o fluxo termine ou então uma mudança de conceito ocorra. Quando a mudança ocorre, o processo de construção do modelo se inicia novamente. Para evitar casos incomuns (como o pior caso de $2^w$ rótulos), os autores limitam o número de $c = 30$, ainda os autores definem o valor de $w = 1000$. Os autores salientam que armazenar combinações de rótulos com suas frequências não implica em armazenar os exemplos, satisfazendo assim, as restrições de memória da mineração em FCDs.

Cada novo nó da Árvores de *Hoeffding* multirrótulo é inicializado usando o algoritmo *majority-labelset*. Assim que os $w$ exemplos chegam aos nós folhas, o classificador PS é executado a fim de modelar as combinações observadas ao longo desses exemplos, para então iniciar o processo de construção do modelo com os novos exemplos.

Para obter um melhor desempenho, escalabilidade, paralelização do método e, ainda, melhorar a detecção de mudanças de conceito, os autores utilizam um comitê de MLHT baseados no ADWIN *Bagging* (BIFET et al., 2009). Com isso, é possível combinar vários métodos sob um esquema de votação para encontrar a predição final. Para detecção de mudanças de conceito, eles eliminam o classificador mais antigo do comitê quando uma mudança de conceito é detectada.

Apesar de ser considerado o algoritmo estado-da-arte na CMFCD, o MLHT apresenta algumas restrições como: é sensível ao valor de $w$ (quantidade de exemplos para iniciar a construção do classificador PS) e de $c$ (quantidade combinações de rótulos frequentes); só é possível detectar uma evolução de conceito se a classe aparecer entre os $w$ exemplos armazenados para o treinamento; como as tradicionais Árvores de *Hoeffding*, os rótulos dos exemplos devem estar sempre disponíveis para a atualização do modelo.

Em Shi et al. (2014), para contornar algumas restrições do algoritmo MLHT, os autores propõem uma estratégia de aprendizado classe incremental que reconhece dinamicamente novas combinações de rótulos frequentes para o PS, denominada ***M**ulti**L**abel **H**oeffding **T**ree with **c**lass incremental **l**earning* (MLHTcl).

O processo de aprendizagem é composto por duas fases: *i)* um número de exemplos com combinações de rótulos frequentes são coletados para inicializar o classificador utilizando o método PS; *ii)* para cada exemplo do FCD, se a combinação de seus rótulos não fizer parte do conjunto de rótulos frequentes, esse exemplo é armazenado, bem como o seu número de ocorrências da mesma combinação de rótulos associadas a ele. Quando o número de ocorrências de alguma combinação fora do conjunto de combinações frequência for maior do que alguma combinação presente no conjunto, os exemplos armazenados referentes a essa nova combinação

são usados para atualizar o modelo usando a estratégia de aprendizado classe incremental baseada no trabalho de Zhou e Chen (2002). Caso contrário, se a combinação de rótulos do exemplo pertencer ao conjunto de rótulos frequentes, o exemplo será usado para atualizar o modelo seguindo a estratégia de atualização exemplo-incremental também baseada no trabalho de Zhou e Chen (2002).

Apesar das melhorias relacionadas à definição dos parâmetros do método MLHT, o método MLHTcl exige uma quantidade de memória e tempo maior, visto que devem ser armazenados os exemplos das combinações de rótulos não pertencentes às combinações frequentes. Isso pode ser um problema se um FCD tiver muitas combinações distintas, ou uma frequência alta de mudanças e evoluções de conceito. Ainda, o problema da necessidade de rótulos reais disponíveis é presente.

Em Song e Ye (2014) os autores apresentam o ***M**ulti-**L**abel **D**ynamic **E**nsemble* (MLDE), método composto por inúmeros ***M**ulti-**L**abel **C**luster-based **C**lassifiers*s (MLCCs), algoritmo que combina árvores de decisão com algoritmos de agrupamento. O MLDE inclui um comitê adaptativo com votação ponderada pela *Subset-Accuracy* dos MLCCs e exemplos ponderados pela similaridade. O MLDE se diferencia dos outros métodos baseados em comitês pelo fato de usar agrupamento como classificador base. Assim, é possível usar o centroide dos grupos para ajustar os pesos em relação a similaridade dos exemplos e, ainda, torna possível a atualização do modelo sem a necessidade da disponibilidade dos rótulos verdadeiros dos exemplos. Outra vantagem é o fato dos classificadores base serem inseridos ou deletados de acordo com a presença ou não de mudanças de conceito, *i.e.*, se não houver mudanças o número de classificadores base aumenta, aumentando também a sua acurácia. Caso contrário, o número de classificados diminui para outros serem criados a partir do novo conceito.

O MLDE é divido em três partes: *i) treino*: são construídos os MLCCs que compõem o comitê. O processo de treinamento de um MLCC é realizado através do agrupamento dos exemplos do conjunto de treino de um bloco $S$. Se a pureza da classe não for maior do que um limiar, então o nó continua se dividindo. Um MLCC é construído para cada bloco de exemplos $S$; *ii) verificação*: quando um novo bloco de exemplos é completo, os classificadores que serão responsáveis pela classificação são selecionados. Todos os MLCCs do comitê são avaliados de acordo com a *Subset-Accuracy* obtida nos exemplos do conjunto de treino de $S$. Se a *Subset-Accuracy* de um MLCC for menor que um limiar, este é descartado. O valor do limiar é definido em relação os resultados da *subset-accuracy* dos MLCCs do comitê; *iii) teste*: para cada exemplo do conjunto de teste de $S$, verifica-se a similaridade entre o grupo mais próximo a ele de cada MLCC restante do comitê. O exemplo receberá os rótulos do grupo pertencente ao MLCC cujo valor de similaridade for o maior.

O MLDE reage muito bem a todo tipo de mudança de conceito, principalmente nas mudanças abruptas, nas quais o modelo é reconstruído rapidamente e se mantém estável em seguida. Ainda, em casos de mudanças de conceito que há pouca alteração na distribuição, o

modelo se mantém estável e não sofre quedas muito grandes de performance. Por outro lado, possui uma complexidade computacional alta, visto que um modelo é construído a cada bloco de exemplos, principalmente em se tratando de FCDs com poucas mudanças de conceito, pois não há um limite máximo de MLCCs para o comitê. Ainda, o MLDE é muito sensível a ruídos e é dependente do tamanho do bloco de exemplos.

Em Dave et al. (2016), os autores propõem o ***PRO****gressive-****E****LM* ***M****ulti-****L****abel* ***C****lassifier* (PRO-EMLC), método que faz uso de ***E****xtreme* ***L****earning* ***M****achine*s (ELMs) para o problema de evolução de conceito. ELM é uma generalização das redes neurais de camada única *feedfoward* (*Single Hidden Layer Feedforward Neural Network*). Esta abordagem ganhou muita atenção devido à sua característica de inicialização dos peso de entrada aleatória e por possuir uma alta velocidade de aprendizado. Em ELM, o pesos iniciais e os vieses das camadas ocultas podem ser selecionados de forma aleatória e a rede pode ser treinada com os pesos de saída para realizar a classificação (HUANG et al., 2011). O ***O****nline* ***S****equential* ***E****xtreme* ***L****earning* ***M****achine* (OS-ELM) é uma extensão do ELM para tratar dados em fluxos contínuos. Esse método mantém o conhecimento de treinamentos anteriores e continua aprendendo a partir de novos dados (LIANG et al., 2006).

O PRO-EMLC pode ser atualizado tanto por bloco-incremental, quanto por exemplo-incremental, basta definir o tamanho do bloco de exemplos. Se esse for igual a um, então a atualização ocorre por exemplo-incremental. Quando o algoritmo detecta a presença de um exemplo com um rótulo ainda não visto, a rede neural é recalibrada para se adaptar ao novo rótulo sem descartar o conhecimento adquirido anteriormente.

Em Venkatesan et al. (2017) os autores apresentam o método ***O****nline* ***S****equential* ***M****ulti-****L****abel* ***E****xtreme* ***L****earning* ***M****achine* (OSML-ELM), extensão do método OS-ELM para CM. Para a inicialização, o número de camadas ocultas é definido de acordo com a natureza e complexidade da base dados, enquanto a função de ativação é obtida através de uma função sigmoide. Para evitar o *overfitting*, os autores fazem uso de uma técnica de parada precoce.

Para o processamento das entradas, por se tratar de uma técnica multirrótulo, cada um dos exemplos de entrada terá o rótulo de saída associado como uma $m$-tupla com 0 ou 1 representando cada um dos rótulos do conjunto total de rótulos $L$. O processo de treino da ELM é o mesmo do método PRO-EMLC. O ponto principal do OSML-ELM é o pós-processamento e a definição do limiar. O valor de limiar é selecionado durante a fase de treinamento de tal forma que maximiza a separação entre o conjunto de rótulo associado ao exemplo ($Y_a$) e o conjunto de rótulos que não estão associados ao exemplo ($Y_b$). Baseado na distribuição de $Y_a$ e $Y_b$, o valor do limiar é definido de acordo com a fórmula: $(\min Y_a + \max Y_b)/2$.

Os métodos baseado em ELM para CM em FCDs, ainda não foram muito explorados e os trabalhos encontrados na literatura não possuem experimentos baseados em metodologias de avaliação de FCDs, nem a comparação com métodos específicos da abordagem. Ainda, em nenhum deles os autores abordam mecanismos de detecção de mudanças de conceito e simulam

tais mudanças com geradores de dados sintéticos.

Vale ressaltar que as ELMs não são bem vistas pela comunidade de pesquisa de aprendizado de máquina devido a sua originalidade contraditória [1]. Segundo Wang e Wan (2008), o autor das ELMs desenvolveu o seu trabalho (HUANG; SIEW, 2004) excluindo referências anteriores (ou seja, sem comparações qualitativas ou quantitativas com métodos anteriores), e ainda, em seus trabalhos posteriores ele cita trabalhos de forma incorreta e continua negando aos autores originais seus créditos.

Em (NGUYEN et al., 2019), os autores desenvolveram um método exemplo-incremental para cenários com latência extrema de rótulos (denominado neste trabalho como *incremental **O**nline **M**ulti-**L**abel **C**lassifier* (OMLC)), cujo modelo de decisão baseado em microgrupos ponderados pelo tempo, em que uma maior atenção é dada aos exemplos mais recentes através de um mecanismo de decaimento de peso baseado na ideia proposta em Cao et al. (2006). O modelo de decisão proposto é formado por dois tipos de microgrupos: *maturos* e *imaturos*. Eles são diferenciados levando em conta os seus pesos e raios. Para classificação, cada microgrupo possui a distribuição de rótulo medida à partir dos exemplos usados para formá-los. Exemplos são classificados medindo a distância entre eles e o seus microgrupos maturos mais próximo. Então, a probabilidade *posteriori* de o exemplo pertencer a uma classe é calculada usando a distribuição do microgrupo mais próximo e os $h$ rótulos com maior probabilidade são associados ao exemplo. O valor de $h$ é calculado através da cardinalidade de rótulos e *Desigualdade de Hoeffding*.

Apesar das vantagens de usar modelos de decisão baseados em microgrupos ponderados e com distribuição de rótulos, podendo assim, lidar com problemas de latência extrema de rótulos, os experimentos executados pelos autores não levam isso em conta. Além disso, o método não lida com evoluções de conceito.

### 3.2.3 Outros Aspectos

Como visto anteriormente, o tratamento de mudanças de conceito é um requisito fundamental para classificação em FCDs e o problema é ainda mais desafiador quando os dados são multirrótulo. Porém, poucas pesquisas têm investido em mecanismos próprios para mudanças de conceito em CMFCD. Pensando nisso, Shi et al. (2014) propõem um mecanismo para detecção de mudanças de conceito baseados em agrupamento de rótulos e entropia.

O método separa os conjuntos de rótulos em dois grupos, um para representar os rótulos correlacionados e outro os interdependentes. Em seguida, outros subconjuntos de rótulos são atribuídos aos exemplos de acordo com seus rótulos originais. Para encontrar as dependências entre rótulos, os autores usam o algoritmo *Apriori* (MA; LIU, 1998) e, para separar o conjunto de rótulos, o algoritmo EM (*Expectation Maximization*) (DEMPSTER et al., 1977).

[1] Fonte disponível em: https://elmorigin.weebly.com/. Acesso em 3 jan 2018

Métodos baseados em entropia já foram utilizados em FCDs monorrótulo para detecção de mudanças de conceito, como é o caso do trabalho proposto em Vorburger e Bernstein (2006). Inspirados neste trabalho os autores adaptaram a versão multirrótulo do cálculo de entropia proposto por Clare e King (2001) para FCDs da seguinte maneira:

$$H(y) = -\sum i[P(y_i)log_2(P(y_i)) + (1 - P(y_i))log_2(1 - P(y_i))] \tag{3.21}$$

O que difere do cálculo proposto por Clare e King (2001) é o fato de eles contarem todos os exemplos em relação aos seus valores de atributos e de cada subconjunto dos conjuntos de rótulos que foram agrupados anteriormente.

Por fim, para detectar as mudanças de conceito, eles empregam uma estratégia de duas janelas deslizantes, uma para representar os exemplos passados e outros os exemplos recentes. Então, é calculado a entropia das duas janelas e, em seguida, a diferença entre elas. Se essa diferença exceder um limiar, então houve mudança de conceito e o tamanho da janela é reduzido para o valor mínimo estipulado pelo usuário.

Apesar de todos os algoritmos citados neste capítulo trazerem resultados significativos para CMFCD, eles possuem certas restrições pouco abordadas na literatura. Uma delas é a suposição de que os rótulos verdadeiros estejam disponíveis para atualização do classificador. Para algumas aplicações essa suposição é perfeitamente factível, entretanto, devido a fatores como altos custos envolvidos no processo de rotulagem, falhas no processo de transmissão dos rótulos, ou mesmo devido às características do processo gerador dos dados, as informações a respeito dos rótulos podem nunca serem obtidas (*Latência Extrema de Rótulos*). Sensores responsáveis por classificar espécies de insetos que cruzam por uma fonte de luz, prevenção e detecção de fraudes em transações financeiras e sistemas de automação de veículos não tripulados, são exemplos de aplicações cujos rótulos dos exemplos não estão imediatamente disponíveis (SOUZA et al., 2015a).

Outro problema que é ignorado por muitos métodos é o fenômeno da evolução de conceito, ou seja, o número de classes pode sofrer alterações ao longo do fluxo, e novas classes podem surgir ou desaparecer (MASUD et al., 2011a). Exemplos disso são tarefas de detecção de intrusos em redes de computadores, em que novos tipos de intrusões podem ser detectados, ou o aparecimento de uma nova categoria de texto em um fluxo de dados vindo do Twitter. A maioria das técnicas tradicionais de classificação em FCDs não são capazes de classificar essas classes emergentes, pois elas não fazem parte do conjunto de classes conhecido pelo classificador (MASUD et al., 2011b).

Uma técnica que vem sendo utilizada nos dois problemas citados acima (latência extrema de rótulos e evolução de conceito) e que ainda não foi bem explorada para CM, é a **D**etecção de **N**ovidade (DN). Essa técnica é uma tarefa de classificação que consiste em identificar se um exemplo, ou conjunto de exemplos, difere significativamente de conceitos conhecidos (FARIA et

al., 2016a). O uso de DN é útil para esses problemas, pois com ela é possível identificar conceitos novidade, que podem indicar o surgimento de um novo conceito, uma mudança ocorrida nos conceitos conhecidos, ou a presença de ruído (GAMA, 2010).

## 3.3 Considerações Finais

Esse capítulo apresentou uma revisão de literatura sobre **C**lassificação **M**ultirrótulo em **F**luxos **C**ontínuos de **D**ados (CMFCD). Foram abordadas características de bases de dados multirrótulo, como cardinalidade de rótulos, densidade de rótulos e dependências de rótulo. Também foram apresentadas as medidas de avaliação multirrótulo. O métodos para CMFCD revisados nesse capítulo, foram divididos em dois grupos (Transformação de Problema e Adaptação de Problema) e suas principais vantagens e desvantagens foram destacadas. Além disso, problemas como tratamento de Evolução de Conceito e a atualização do modelo sem a necessidade em cenários de latência extrema de rótulos, foram apresentados mostrando que poucos trabalhos da literatura tratam esses problemas.

O próximo capítulo deste trabalho apresenta a tarefa de **D**etecção de **N**ovidade (DN) em **F**luxos **C**ontínuos de **D**ados (FCDs), mostrando características, conceitos e os principais métodos encontrados na literatura.

# Capítulo 4

## DETECÇÃO DE NOVIDADE EM FLUXOS CONTÍNUOS DE DADOS

Com os avanços das técnicas de aprendizado de máquina, pesquisas identificaram a necessidade de criar modelos que são capazes de extrair informações de dados imprecisos com presença de ruídos, dados inconsistentes por conta de anomalias, informações com padrões escassos, e dados que tendem a mudar suas características. A criação desses modelos motivou estudos em **D**etecção de **N**ovidade (DN), que têm como objetivo a detecção de dados raros e desconhecidos (ALBERTINI; MELLO, 2010).

Segundo Faria et al. (2016a), DN consiste em identificar se um exemplo (ou conjunto de exemplos) não rotulado difere significativamente dos conceitos já aprendidos. Gama (2010) destaca que quando a tarefa de DN é aplicada a FCDs, surgem alguns desafios que incluem a presença de:

- *Mudanças de Conceito*, que dificulta distinguir novos conceitos dos conceitos conhecidos;
- *Ruídos e Outliers*, que podem ser confundidos com a ocorrência de novos conceitos;
- *Conceitos Recorrentes*, que são fáceis de serem confundidos com a descoberta de novos conceitos;
- *Evoluções de Conceito*, que são descobertas de novas classes ao decorrer do tempo e que devem ser incorporadas ao modelo de decisão.

Dentre as diversas aplicações de DN em FCDs encontradas na literatura, destacam-se Detecção de Intrusos (COULL et al., 2003; SPINOSA et al., 2008), Detecção de Falhas (ZHANG et al., 2006), Diagnósticos Médicos (PERNER, 2009; SPINOSA; CARVALHO, 2004), Detecção de Regiões de Interesse em Imagens (SINGH; MARKOU, 2004), Detecção de Fraudes (WANG et al., 2003), Detecção do Tipo de Cobertura Florestal (MASUD et al., 2011a), Filtros de Spam (HAYAT; HASHEMI, 2010), Sistemas Biométricos (PISANI; LORENA, 2011; PISANI et al.,

2015; PISANI et al., 2017), Detecção de Variações Comportamentais em um Jogador (VALLIM et al., 2013) e Classificação de Texto (LI; CROFT, 2006).

Em geral, as técnicas desenvolvidas para DN possuem uma abordagem de aprendizado *Online-Offline*. Na fase *offline* é construído um modelo de decisão e na fase *online*, sempre que um exemplo chega, ele é classificado ou é rejeitado pelo modelo de decisão. Essa rejeição de exemplos da fase *online* torna possível reconhecer um conceito novidade, que pode indicar o surgimento de novos conceitos, uma mudança ocorrida nos conceitos conhecidos ou a presença de ruído (GAMA, 2010). Assim, o uso desta técnica para classificação em FCDs é fundamental, pois auxilia na atualização do modelo de decisão.

Este capítulo tem como objetivo apresentar a tarefa de DN e definir os principais conceitos do tema. Além disso, o capítulo traz uma visão geral dos principais algoritmos de DN em FCDs e quais suas contribuições para literatura, destacando o algoritmo ***Mult**Iclass learNing **A**lgorithm for data **S**treams* (MINAS) (FARIA et al., 2016b). A organização do capítulo é a seguinte: a Seção 4.1 apresenta as definições de Mudança e Evolução de Conceito, dois fenômenos presentes em problemas envolvendo dados em fluxos contínuos, nos quais a DN pode ser usada como auxílio para o tratamento desses fenômenos; a Seção 4.2 traz os principais conceitos e formalização da tarefa de DN em FCDs; a Seção 4.3 aborda os principais métodos de DN em FCDs encontrados na literatura; o método MINAS, que será usado como base para o desenvolvimento do método proposto neste trabalho, é apresentado na Seção 4.4; por fim, a Seção 4.5, traz as principais metodologias de avaliação usadas para validar os algoritmos de DN em FCDs.

## 4.1 Mudança de Conceito e Evolução de Conceito

Grande parte dos algoritmos de mineração de dados assume que um exemplo é gerado de acordo com uma distribuição de probabilidade estacionária. Em FCDs, em contrapartida, um FCD possui uma distribuição não estacionária, ou seja, os conceitos não são estáticos mas mudam ao longo do tempo. Com isso, dois fenômenos podem ocorrer: *Mudança de Conceito* e *Evolução de Conceito* (GAMA, 2010; FARIA et al., 2016a).

Na literatura, podemos encontrar várias definições de Mudança de Conceito:

- Dries e Rückert (2009): Mudança de Conceito é um importante problema em aprendizado de máquina e pode ser definido como uma mudança significativa na distribuição dos dados;
- Elwell e Polikar (2011): Mudança de Conceito refere-se a uma mudança nas definições das classes (conceitos) ao longo do tempo e, portanto, uma mudança na distribuição a partir da qual os dados são gerados;

- Farid et al. (2013): todo exemplo $X_t$ é gerado a partir uma distribuição $D_t$. Se para quaisquer dois exemplos $X_1$ e $X_2$, com marcadores de tempo $t_1$ e $t_2$, $D_1 \neq D_2$, então ocorre uma Mudança de Conceito;

- Souza et al. (2015a): um conceito pode ser definido como um conjunto de exemplos geradas pela mesma função subjacente. Quando essa função muda por alguma razão, ocorre o fenômeno da Mudança de Conceito.

De acordo com (TSYMBAL, 2004), um grande desafio ao se tratar Mudança de Conceito é distinguir entre uma mudança real e um ruído. Alguns exemplos de Mudança de Conceito são: mudanças no padrão de uma doença, mudanças de temperatura, mudanças no perfil de compras de um cliente ao longo dos anos, etc (FARIA et al., 2016a).

Segundo Gama et al. (2013), duas abordagens são usadas para adaptar o modelo de decisão com o objetivo de tratar Mudanças de Conceito: *Blind* e *Informed*. A primeira atualiza o modelo em intervalos regulares de tempo sem verificar se realmente houve mudanças. Geralmente o modelo é atualizado através dos últimos exemplos rotulados. A segunda modifica o modelo de decisão no momento em que uma Mudança de Conceito é detectada.

Em FCDs, nem sempre o número de classes é previamente definido, uma vez que nem todas as classes são conhecidas na fase de treinamento e exemplos de novas classes podem aparecer ao longo do tempo (FARIA et al., 2016a). Assim, os modelos devem estar aptos a reconhecer essas novas classes assim que elas aparecerem no fluxo (ABDALLAH et al., 2016). Esse fenômeno é chamado de Evolução de Conceito e pode ocorrer em problemas como detecção de intrusão em redes de computadores (SPINOSA et al., 2008), filtro de spam (HAYAT; HASHEMI, 2010) e classificação de texto (LI; CROFT, 2006).

Na área de Reconhecimento de Padrões, um cenário semelhante à Evolução de Conceito é estudado. Este recebe o nome de *Reconhecimento em Cenário Aberto*, do inglês *Open-Set Recognition*, no qual não é necessário, ou não é conhecido, o conjunto completo das possíveis classes do problema. Isso torna indispensável o uso de modelos que sejam capazes de lidar com um vasto número de classes que ainda não foram vistas na fase de treinamento (SCHEIRER et al., 2013; SCHEIRER et al., 2014; JAIN et al., 2014). Nesse cenário, quando os exemplos de teste não pertencem a nenhuma classe conhecida, o classificador deve rejeitá-los adequadamente, classificando-os como *desconhecidos* (JÚNIOR et al., 2016). Uma abordagem comum para o tratamento parcial desse problema depende do uso de esquemas de classificação baseados em limiar (PHILLIPS et al., 2011). Basicamente, esses métodos verificam se um *score* correspondente é maior ou igual a um limiar previamente definido. Outra abordagem que também é bastante utilizada (SCHEIRER et al., 2013; COSTA et al., 2012; COSTA et al., 2014) baseia-se na modificação do motor de classificação ou função objetivo das SVMs. Como exemplos de aplicações nesse cenário temos: atribuição de imagem a uma determinada câmera para identificar a autenticidade de uma imagem sob investigação (COSTA et al., 2012; COSTA et al., 2014);

detecção de impressão digital falsa em biometria (RATTANI et al., 2015); identificação de faces (JUNIOR; SCHWARTZ, 2014), etc.

Apesar de tratar problemas semelhantes ao de Evolução de Conceito, os modelos de Reconhecimento em Cenário Aberto não são dinâmicos, uma característica fundamental para algoritmos de mineração em FCDs. No entanto, Bendale e Boult (2015) apresentam outro tipo de problema denominado Reconhecimento em Mundo Aberto (*Open World Recognition*), cuja ideia é fazer com que o sistema rotule explicitamente as novas entradas como *desconhecidas*. Essas novas entradas desconhecidas devem ser então coletadas e rotuladas (por um especialista humano, por exemplo). Quando há um número suficiente de exemplos *desconhecidos* para o aprendizado de novas classes, o sistema deve atualizar o classificador, tornando cada uma dessas novas classes *conhecidas* para o sistema. Para isso, os autores estendem o algoritmo *Nearest Class Mean* (MENSINK et al., 2013; RISTIN et al., 2014) para o *Nearest Non-Outlier* (BENDALE; BOULT, 2015).

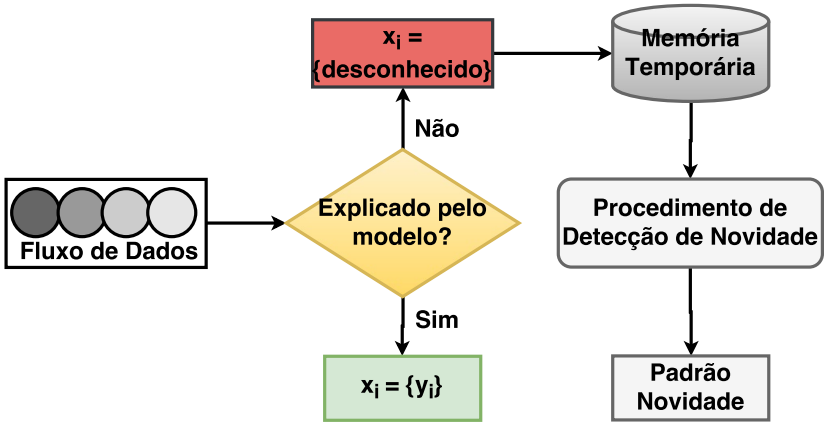
O trabalho de Bendale e Boult (2015), apesar de ser viável para alguns casos, utiliza uma abordagem que mantém fixo o limiar de detecção de novas classes à medida que o problema evolui. Segundo Rosa et al. (2016), isso entra em conflito com a definição de Reconhecimento em Mundo Aberto, no qual a estrutura do problema é progressivamente revelada à medida que mais dados são observados e os parâmetros tendem a sofrer mudanças ao longo do tempo. Com isso, eles mostram que é necessário usar limiares dinâmicos que mudam à medida que novos exemplos e novas classes chegam, ao invés de estimá-los a partir de um conjunto inicial e fechado de classes, como feito até agora (MENSINK et al., 2013; RISTIN et al., 2014; BENDALE; BOULT, 2015).

Os trabalhos citados anteriormente são fundamentais para este projeto, principalmente aqueles que fazem uso de mecanismo de DN, habilidade de reconhecer se uma entrada difere significativamente das anteriores (PERNER, 2009). Esse mecanismo, em conjunto com o Aprendizado *Online-Offline*, facilita a detecção de Mudanças de Conceito e serve como base para o desenvolvimento de algoritmos que tratam o problema da Evolução de Conceito. Alguns deste algoritmos serão apresentados na Seção 4.3.

## 4.2 Formalização do Problema

A maioria dos método de DN trabalham com a abordagem *online-offline*. Na fase *offline* um conjunto de exemplos rotulados são usados para construção de um modelo de decisão (Figura 3a). Esses exemplos, geralmente, fazem parte de apenas uma classe, chamada *Classe Normal*. Na fase *online*, os exemplos que chegam ao longo do fluxo são classificados como pertencentes a *classe normal* ou são rejeitos pelo modelo, o que pode indicar uma *Anormalidade*, *Anomalia* ou *Novidade* (FARIA et al., 2016a) (Figura 3b). Esse é um caso de DN conhecido como classificação com uma classe e pode ser encontrados em trabalhos como: Spinosa et al.


(a) *Fase Offline*: Criação do modelo de decisão a partir de exemplos rotulados




(b) *Fase Online*: Classificação de exemplos que chegam ao longo do fluxo. Caso o modelo de decisão não consiga classificar o exemplo ele é marcado como *desconhecido* e enviado para a *Memória Temporária*. Futuramente os exemplos *desconhecidos* passam pelo *processo de detecção de novidade* para verificar se são PNs, Extensões ou *Outliers*

Figura 3 – Visão geral da tarefa de DN. Adaptada de Faria et al. (2016b)

(2009), Hayat e Hashemi (2010), Rusiecki (2012), Krawczyk e Woźniak (2013). Porém, DN pode ser considerada uma tarefa de classificação multiclasse, pois o conjunto de exemplos de treino pode conter mais uma classe e novas classes podem surgir ao longo do tempo. Trabalhos como os de Faria et al. (2016a), Masud et al. (2010), Masud et al. (2011a), Al-Khateeb et al. (2012a), Al-Khateeb et al. (2012b) tratam DN como uma tarefa multiclasse.

De uma maneira geral, Faria et al. (2016a) formalizam a tarefa de DN multiclasse da seguinte maneira: na fase *offline*, cada exemplo $X_i$ de um conjunto de treinamento é associado a um rótulo $y_i$, sendo $y_i \in Y^{tr}$, com $Y^{tr} = \{y_{con_1}, y_{con_2}, \ldots, y_{con_q}\}$, em que $y_{con_i}$ representa a $i$-ésima classe conhecida do problema e $q$ é o número de classes conhecidas. Na fase *online*, assim que um novo exemplo chega, novas *Classes Novidade* podem ser detectadas, expandindo o conjunto de classes rótulo para $Y^{all} = \{y_{con_1}, y_{con_2}, \ldots, y_{con_q}, y_{nov_1}, \ldots, y_{nov_k}\}$, em que $y_{nov_i}$ representa a $i$-ésima *Classe Novidade* e $k$ é o número de classes novidade que eram, até então, desconhecidas.

Abaixo segue a definição de alguns conceitos importantes sobre DN e que serão usadas neste trabalho (FARIA et al., 2016a):

- ***Outlier:*** exemplos isolados, grupos de exemplos não representativos ou não coesos, que não são explicados pelo modelo normal, que devem ser identificados, mas não devem ser usados para atualizar o modelo de decisão.

- **Classe Novidade:** classe que não está presente na fase de treinamento (*offline*), mas que aparece ao longo do fluxo na fase *online*. O aparecimento de novas classes também é conhecido como Evolução de Conceito (MASUD et al., 2011a).
- **Desconhecido:** exemplo não explicado pelo modelo de decisão atual. Em problemas multiclasse, um grupo de exemplos desconhecidos pode ser usado para modelar novos conceitos, detectar ruídos e *outliers*, e atualizar o modelo de decisão (FARIA et al., 2016b).
- **Padrão-Novidade (PN):** padrão identificado em exemplos não rotulados, previamente considerados como desconhecidos pelo sistema de classificação. *Outliers* e ruídos não se encaixam como PNs, uma vez que eles representam exemplos isolados ou grupos de exemplos não-coesos. Um PN pode indicar o possível aparecimento de um novo conceito (Evolução de Conceito).
- **Extensão:** grupo de exemplos não rotulados pelo sistema, previamente marcados como desconhecidos e que representa uma extensão de um conceito conhecido (mudança de conceito).

## 4.3 Métodos para Detecção de Novidade em Fluxos Contínuos de Dados

Os algoritmos que serão apresentados nesta seção são baseados em três abordagens principais (FARIA et al., 2016b):

1. Trabalhos que tratam DN como uma tarefa de classificação com uma classe;
2. Trabalhos que trazem melhorias para a abordagem de classificação com uma classe;
3. Trabalhos que tratam DN como uma tarefa de classificação multiclasse, porém detectam apenas uma classe novidade por vez.

A Tabela 4 apresenta os trabalhos que serão discutidos, bem como suas principais características e referências na literatura.

Os algoritmos de Redes Neurais para **D**etecção de **N**ovidade (DN) (RUSIECKI, 2012) e o WOCSVM (*Wighted One-Class Support Vector Machine*) (KRAWCZYK; WOŹNIAK, 2013) criam na fase *offline* dos algoritmos, um modelo de decisão com base apenas na classe normal e, na fase *online*, os exemplos são classificados apenas como normal ou novidade. O fato desses métodos não serem multiclasse torna seus modelos de decisão incapazes de distinguir diferentes PNs que podem surgir. Além disso, os métodos necessitam dos rótulo reais dos exemplos para a atualização do modelo.

Tabela 4 – Métodos de DN em FCDs. A descrição das abordagens referentes aos números informados em sua coluna encontram-se no texto da seção 4.3. A coluna com o acrônimo **RD**, informa se o método assume ou não que os **R**ótulos verdadeiros dos exemplos estarão sempre **D**isponíveis para a atualização do modelo. A coluna ***Outliers*** informa se o método é capaz de tratar ou não *outliers*

| **Algoritmos** | **Referência** | **Abordagem** | **RD** | ***Outliers*** |
|---|---|---|---|---|
| OLINDDA | (SPINOSA et al., 2009) | 2 | não | sim |
| DETECTNOD | (HAYAT; HASHEMI, 2010) | 2 | não | não |
| ECSMiner | (MASUD et al., 2011a) | 3 | sim | sim |
| MCM | (MASUD et al., 2010) | 3 | sim | sim |
| Redes Neurais para DN | (RUSIECKI, 2012) | 1 | sim | não |
| CLAM | (AL-KHATEEB et al., 2012a) | 3 | sim | sim |
| WOCSVM Adaptativo | (KRAWCZYK; WOŹNIAK, 2013) | 1 | sim | não |

O ***OnLI**ne **N**ovelty and **D**rfit **D**etection **A**lgorithm* (OLINDDA) (SPINOSA et al., 2009) e o ***D**iscr**ETE** **C**osine **T**ransform based **NO**velty and **D**rift detection* (DETECTNOD) (HAYAT; HASHEMI, 2010) tratam alguns desses problemas. Eles transferem os exemplos classificados como *desconhecidos* para uma memória temporária para que futuramente sejam analisados. No OLINDDA, cada vez que um novo exemplo é marcado como *desconhecido*, o algoritmo *k-Means* é executado produzindo $k$ Microgrupos. O DETECTNOD também usa agrupamento nos exemplos *desconhecidos* e, em seguida, interpola os grupos e produz uma representação baseada em *Discrete Cosine Transform*. Nos dois métodos, os grupos considerados válidos podem representar *Extensões* do conceito normal (*i.e.* Mudanças de Conceito) ou um PN e, ambos, são usados para atualizar o modelo de decisão. A desvantagem do DETECTNOD é a sensibilidade a ruídos e *outliers* e o fato de não considerar a tarefa de DN como multiclasse. É importante destacar que ambos os métodos atualizam o modelo de decisão sem usar os rótulos reais dos exemplos.

Os métodos ***E**nhanced **C**lassifier for data **S**treams with novel class **Miner*** (ECSMiner) (MASUD et al., 2011a), ***M**ulti **C**lass **M**iner in data streams* (MCM) (MASUD et al., 2010) e ***CLA**ss-based **M**icro classifier ensemble* (CLAM) (AL-KHATEEB et al., 2012a; AL-KHATEEB et al., 2012b) consideram DN como uma tarefa multiclasse. Na fase *offline*, comitês de classificadores são treinados a partir de exemplos rotulados de diferentes classes. Na fase *online*, a classificação é dada pela regra da votação da maioria. Novidades são reconhecidas somente com a chegada de exemplos semelhantes e para a atualização do modelo de decisão é utilizada uma abordagem bloco-incremental. No entanto, o modelo de decisão só é atualizado quando todos os exemplos do bloco forem rotulados, substituindo o classificador com o maior erro por um novo. O erro é computado usando o conjunto de exemplos rotulados do último bloco de exemplos. Suas principais desvantagens são o fato de assumir que os rótulos de todos os exemplos estarão sempre disponíveis e a capacidade de identificar apenas uma classe Novidade em cada bloco de exemplos. Sendo assim, quando exemplos de classes distintas aparecem, o

modelo às classifica como Classes Novidade, mas não consegue distingui-las. A distinção só acontece quando os rótulos verdadeiros são fornecidos e novos classificadores são treinados.

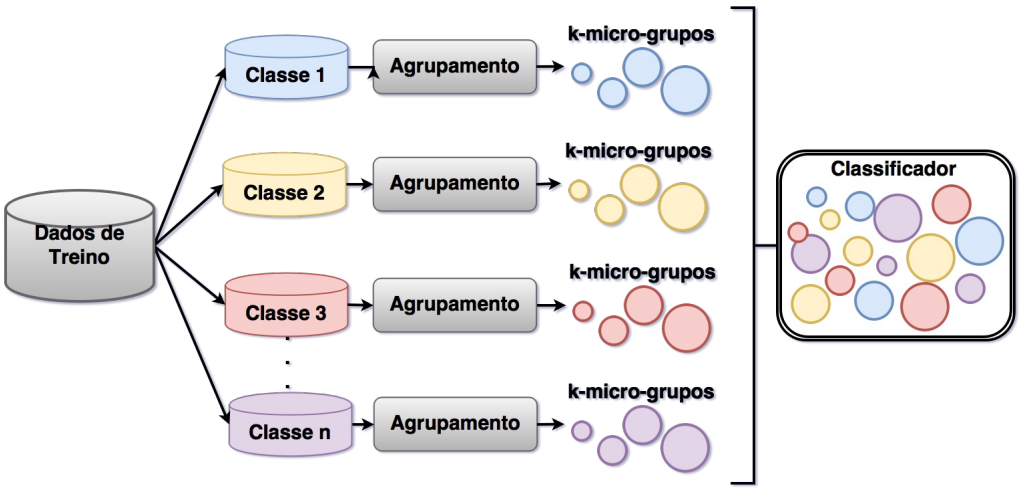
Com o intuito de minimizar os problemas dos métodos citados acima, em Faria et al. (2016b) os autores apresentaram um novo algoritmo denominado ***M**ult**I**class learN**i**ng **A**lgorithm for data **S**treams* (MINAS). A seção 4.4 apresenta detalhes desse algoritmo.

## 4.4 MINAS

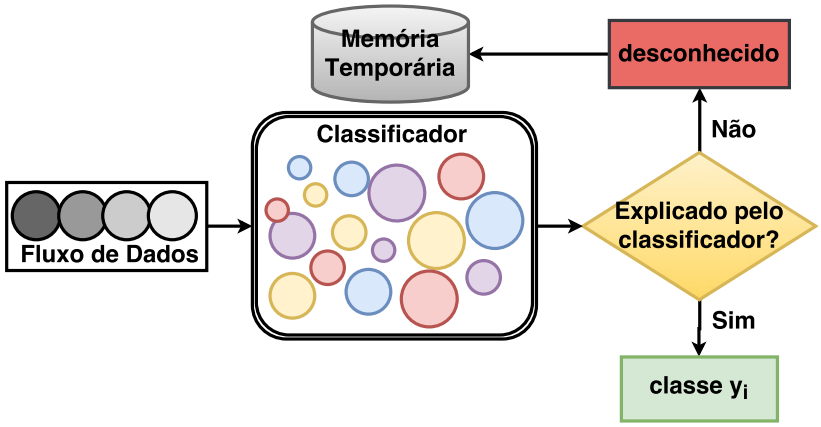
Esta seção tem o objetivo de introduzir o ***M**ult**I**class learN**i**ng **A**lgorithm for data **S**treams* (MINAS) (FARIA et al., 2016b), algoritmo de DN em FCDs que, considera a tarefa de DN como multiclasse, é capaz de identificar várias Classes Novidade simultaneamente e não assume que o rótulo verdadeiro dos exemplos estará disponível para atualização. Suas principais características são (FARIA et al., 2016b):

- O uso de apenas um modelo de decisão para representar as classes conhecidas representadas por exemplos rotulados e PNs encontrados a partir de exemplos não rotulados;
- Um procedimento de DN que identifica e distingue diferentes PNs durante o FCD;
- Um procedimento de DN que identifica extensões de conceitos conhecidos, ao invés de identificar apenas extensões da classe normal e, ainda, distingue entre extensões e PNs;
- Remoção de exemplos antigos da memória temporária com o auxílio de *timestaps*;
- Cálculo automático de limiares para distinguir entre extensões e PNs;
- Classificação de novos exemplos em uma das classes conhecidas ou em um dos PNs detectados durante o FCD;
- O uso de memória *sleep* para detecção de contextos recorrentes (tipo especial de mudança de conceito, no qual os conceitos que apareceram no passado voltam a aparecer no futuro e podem ser confundidos com novos conceitos (FARIA et al., 2016a)).

Na fase *offline* do MINAS, um modelo de decisão é criado utilizando dados rotulados. Nessa fase, um conjunto de dados contendo exemplos de diferentes classes é dividido em subconjuntos, cada um representando uma das classes do problema. Em seguida, um algoritmo de agrupamento ($k$-*means* ou *CluStream*) é executado em cada um desses subconjuntos produzindo $k$ Microgrupos. Cada um desses microgrupos, além de suas características tradicionais ($n$, $SS$, $LS$, $CF1^t$ e $CF2^t$), contém o rótulo da classe representada por eles. O conjunto desses microgrupos formam o modelo de decisão. A Figura 4a ilustra a visão geral da fase *offline* do algoritmo.




(a) Fase *offline* do algoritmo MINAS.




(b) Fase *online* do algoritmo MINAS.

Figura 4 – Visão geral do algoritmo MINAS. Adaptado de Faria et al. (2016b)

Na fase *online* do MINAS, os exemplos que chegam ao longo do fluxo são rotulados usando o modelo de decisão atual. Essa fase é composta de três operações: *i)* classificar novos exemplos; *ii)* detectar novidades; *iii)* atualizar o modelo de decisão.

Na operação de classificação, para cada novo exemplo é calculada a distância Euclidiana entre ele e o centroide do microgrupo mais próximo. Se essa distância for menor que o raio do microgrupo, o exemplo recebe o rótulo desse microgrupo, caso contrário o exemplo é considerado *desconhecido* e é enviado para a memória temporária. A Figura 4b ilustra uma visão geral da operação de classificação da fase *online* do algoritmo.

Para a DN, periodicamente, um agrupamento é executado na memória temporária produzindo um conjunto de $k$ novos microgrupos. Os microgrupos obtidos são validados a fim de descartar os não coesos ou não representativos. Os válidos são então analisados a fim de decidir se eles representam Extensões dos conceitos já aprendidos ou PNs. Para distinção entre Extensões e PNs, é calculada a distância Euclidiana entre o centroide da Novidade e de seu

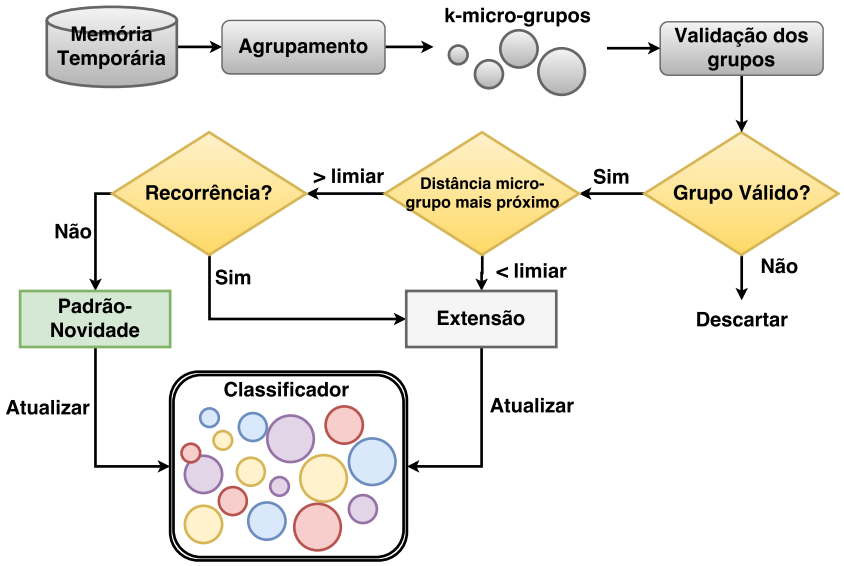



Figura 5 – Procedimento de DN do algoritmo MINAS. Adaptado de Faria et al. (2016b)

microgrupo mais próximo. Se essa distância for menor que um limiar, então essa Novidade é uma Extensão e seus microgrupos recebem o rótulo do microgrupo mais próximo, caso contrário é um PN, e os microgrupos recebem o rótulo $PN_i$, sendo $i$ o $i$-ésimo PN encontrado.

Vale destacar que os PNs encontrados sem *feedback* externo (*e.g., especialista humano*) não estão associados diretamente às novas classes do problema. Uma classe pode ser composta por vários PNs. Assim, somente um especialista de domínio poderá dizer se os PNs encontrados pelo método representam, de fato, uma nova classe do problema e se diferentes PNs podem estar relacionados a uma mesma classe novidade. O MINAS possui, também, uma versão que atualiza o modelo de decisão com base em um pequeno conjunto de exemplos, rotulados pelo especialista, selecionados por uma técnica de aprendizado ativo.

A operação de atualização do modelo de decisão ocorre no procedimento de DN. Nela, tanto os microgrupos das Extensões, quanto os dos PNs são adicionados ao modelo de decisão e usados para classificar novos exemplos.

A Figura 5, ilustra uma visão geral do processo de DN, em que é possível visualizar as operações de DN e de atualização do modelo de decisão.

Nota-se que, apesar de existirem muitos trabalhos sobre DN em FCDs, não foi encontrado nenhum algoritmo que usa DN para CMFCD. Assim, este trabalho apresenta dois métodos: ***Mult**I*-label *learN*ing *A*lgorithm for data *S*treams with *L*abel *C*ombination-based methods (MINAS-LC) e ***Mult**I*-label *learN*ing *A*lgorithm for data *S*treams with *B*inary *R*elevance transformation (MINAS-BR). Os detalhes destes métodos são apresentados no chapter 5.

## 4.5 Avaliação em Detecção de Novidade

Apesar de muitos algoritmos terem sido propostos para lidar com DN em cenários envolvendo FCDs, pouca atenção foi dedicada à avaliação do desempenho preditivo dos mesmos (FARIA et al., 2016a).

Dois métodos de amostragem têm sido utilizados para avaliar a tarefa de classificação em FCDs: *i*) *holdout*, no qual os dados são divididos em conjunto de treino e conjunto de teste e de tempos em tempos o modelo de decisão atual é aplicado ao conjunto de treino; *ii) prequential*, em que cada exemplo individual pode ser usado para testar o modelo antes de ser usado para treinamento e as medidas de avaliação podem ser atualizadas de forma incremental. Segundo Bifet et al. (2010), a metodologia *prequential* possui a vantagem de que nenhum conjunto é necessário para testes, aproveitando ao máximo os dados disponíveis.

Geralmente, os trabalhos da literatura para DN em FCDs não discutem sobre a metodologia de avaliação usada. Alguns deles usam metodologias de cenários *batch*, enquanto outros usam metodologias novas, mas sem a justificativa do seu uso.

Nos trabalhos de Spinosa et al. (2009) e Hayat e Hashemi (2010), os autores usam metodologias similares. Eles usam validação cruzada com 10 partições, as partições são compostas apenas por exemplos da classe normal e o conjunto de teste é composto por exemplos das classes novidades mais os exemplos restantes da classe normal. As medidas de avaliação usadas pelos autores são: *Mnew* (percentual de exemplos pertencentes a classes novidade classificados como classe normal. Equação 4.1); *Fnew* (percentual de exemplos pertencentes a classe normal classificados como classe novidade ou extensão. Equação 4.2); *Err* (percentual de classificações incorretas. Equação 4.3). Para estas equações, *FP* é o número total de exemplos pertencentes a classe normal classificados como classe novidade, extensão ou desconhecido, *FN* é o número total de exemplos pertencentes a classes novidade classificados como classe normal, *FE* é o total de exemplos das classes existentes classificados incorretamente, *Nc* é o número total de exemplos pertencentes a classes novidade e *N* é o número total de exemplos do FCD:

$$Mnew = \frac{FN * 100}{Nc} \tag{4.1}$$

$$Fnew = \frac{FP * 100}{N - Nc} \tag{4.2}$$

$$Err = \frac{(FP + FN + FE) * 100}{N} \tag{4.3}$$

Em Masud et al. (2010), Masud et al. (2011a), Farid et al. (2013), Al-Khateeb et al. (2012a), Al-Khateeb et al. (2012b), mesmo os métodos considerando DN como uma tarefa multiclasse, eles usam medidas de avaliação binárias. Entretanto, com essas medidas não é possível avaliar corretamente os métodos, pois não é suficiente classificar um exemplo de uma

classe novidade apenas como novidade. Um erro deve ser computado quando exemplos de diferentes classes novidade são classificados no mesmo PN.

Muitos trabalhos usam medidas clássicas para avaliação dos métodos, como a medida AUC (*Area Under the ROC Curve*), usada em Tan et al. (2011). Nesse trabalho, os exemplos do conjunto de teste são ranqueados de acordo com um *score* de anomalia. A Acurácia é usada por Krawczyk e Woźniak (2013), mas os autores não deixam claro como as medidas são computadas na DN. As tradicionais Precisão, Revocação e F1 (Equações 4.4, 4.5 e 4.6, respectivamente) foram adaptadas por Albertini e Mello (2007) para avaliação de DN. Nas equações 4.4, 4.5 e 4.6, *TNov* é a contagem de novidades verdadeiras detectadas, *DNov* é o total de exemplos detectados como novidade, e *Nc* é o número total de exemplos que pertencem a classes novidade no FCD:

$$Precisao = \frac{TNov}{DNov} \tag{4.4}$$

$$Revocacao = \frac{TNov}{Nc} \tag{4.5}$$

$$\text{F1} = 2 * \frac{Precisao * Revocacao}{Precisao + Revocacao} \tag{4.6}$$

Em Rusiecki (2012), os autores realizam os experimentos com conjuntos de dados 2D, e ao invés de usarem medidas de avaliação, eles usam um gráfico 2D com duas curvas: uma representando os exemplos e a outra representando os *timestamps* dos momentos em que são detectadas as novidades.

Outra limitação que é encontrada para avaliar algoritmos de DN em FCDs, é a questão dos exemplos que não são explicados pelo modelo e que são marcados como *desconhecidos* (FARIA et al., 2016a). Em Faria et al. (2015), os autores apresentam uma metodologia de avaliação que considera os exemplos *desconhecidos*. Nessa metodologia uma medida de avaliação específica para exemplos *desconhecidos* é calculada. Além disso, com ela é possível associar PNs com as classes do problema e, posteriormente, avaliá-las usando medidas de avaliação multiclasse.

Para isso, os autores assumem que a matriz de confusão gerada pelos algoritmos de DN não é quadrada, pois o número de colunas aumenta sempre que um novo PN é detectado. Cada linha da matriz representa uma classe do problema, sendo classes conhecidas ou classes novidade e cada coluna representa uma das classes preditas pelo algoritmo de DN. Ainda, a última coluna da matriz representa os exemplos marcados como *desconhecidos*. É válido destacar que os PNs detectados pelo algoritmo não possuem uma relação direta com as classes do problema (FARIA et al., 2015; FARIA et al., 2016a).

A fim de avaliar a matriz de confusão, cinco requisitos devem ser considerados (FARIA et al., 2016a):

1. Uma classe pode ser representada por mais de um PN, assim é possível existir mais PNs do que classes do problema;

2. O algoritmo pode detectar menos PNs do que o número de classes novidade. Isso acontece se o algoritmo não conseguiu distinguir os exemplos de todas as classes novidade;

3. Exemplos não explicados pelo modelo de decisão são marcados como *desconhecidos*;

4. Por se tratar de uma tarefa multiclasse, os cálculos das medidas de acurácia e erro devem considerar as diferentes classes aprendidas nas fases *offline* e *online*, sendo que esse cenário é mais difícil de ser tratado do que simplesmente distinguir entre conceitos normal e novidade;

5. Para representar as mudanças ao longo do tempo é necessário realizar uma avaliação periódica da matriz de confusão.

## 4.6 Considerações Finais

Este capítulo apresentou a tarefa de DN em FCDs, mostrando os principais conceitos, algoritmos e metodologias de avaliação. Foram discutidas as vantagens e desvantagens dos algoritmos e foi apresentado o algoritmo MINAS que servirá como base para o desenvolvimento deste trabalho.

O próximo capítulo apresenta os métodos MINAS-LC e MINAS-BR, desenvolvidos para problemas de CMFCD em cenários de latência extrema de rótulos, usando técnicas de DN e transformação de problemas.

# Capítulo 5

# MÉTODOS DE DETECÇÃO DE NOVIDADES PARA CLASSIFICAÇÃO MULTIRRÓTULO EM FLUXOS CONTÍNUOS DE DADOS

Este capítulo apresenta a descrição dos dois métodos de **C**lassificação **M**ultirrótulo em **F**luxos **C**ontínuos de **D**ados (CMFCD) propostos neste trabalho. O primeiro, denominado ***Mult**I**-label** **learN**ing **A**lgorithm for data **S**treams with **L**abel **C**ombination-based methods* (MINAS-LC), detalhado na Seção 5.1.2, lida com problemas de CMFCD em ambientes de latência extrema de rótulos apenas com ocorrências mudanças de conceito. Portanto, seu procedimento de DN reconhece apenas extensões de conceitos conhecidos. Na Seção 5.2, será apresentado ***Mult**I**-label** **learN**ing **A**lgorithm for data **S**treams with **B**inary **R**elevance transformation* (MINAS-BR), método de CMFCD construído para tratar tanto mudanças de conceito, quanto evoluções de conceito em ambientes de latência extrema de rótulos. Esse método possui um conjunto de modelos de decisão, um para cada classe do problema, e seu procedimento de DN reconhece extensões de classes conhecidas e **P**adrões-**N**ovidade (PNs), que podem representar novas classes emergentes.

## 5.1 MINAS-LC

### 5.1.1 Visão geral do MINAS-LC

**C**lassificação **M**ultirrótulo (CM) consiste em associar exemplos a múltiplas classes simultaneamente (TSOUMAKAS et al., 2009). Com a popularização da mineração em **F**luxos **C**ontínuos de **D**ados (FCDs), a tarefa de CM tornou-se ainda mais desafiadora, pois métodos convencionais não são capazes de lidar diretamente com dados não estacionários e potencialmente infinitos. Portanto, são necessários novos métodos de **C**lassificação **M**ultirrótulo em **F**luxos **C**ontínuos de **D**ados (CMFCD) com procedimentos adaptativos para lidar com Mudanças de Conceito (mudanças na distribuição das classes) e, que processem os dados sem armazená-los diretamente na memória (pois o fluxo de dados pode ser infinito)(GAMA, 2010; SILVA et al.,

2013).

A maioria dos métodos de CMFCD assumem a disponibilidade constante dos rótulos reais dos exemplos para atualização de seus modelos. Em outras palavras, eles assumem que após a classificação de um exemplo, seu rótulo estará disponível sem nenhum atraso. Esse é um cenário irreal para a maioria dos problemas de FCDs, pois os rótulos reais dos exemplos podem nunca ser disponibilizados (*Latência Extrema de Rótulos*) ou, só podem ser acessados após certo tempo. Isso ocorre por conta do alto custo do processo de rotulagem, por falhas na transmissão dos rótulos, pela necessidade de especialistas de domínio e etc. Esse tipo de restrição impossibilita os métodos de atualizarem seus modelos de decisão de forma totalmente supervisionada (KREMPL et al., 2014; SOUZA, 2016).

Uma tarefa que tem se destacado ao lidar com problemas de latência de rótulos é a **D**etecção de **N**ovidade (DN) (MASUD et al., 2011a; MASUD et al., 2011b; SOUZA et al., 2015b; FARIA et al., 2016b). Na classificação em FCDs, a tarefa de DN consiste em identificar novos padrões em exemplos do fluxo de dados. Em alguns aspectos, esses padrões podem ser diferentes dos exemplos observados previamente e, portanto, podem ser usados para atualizar os modelos de decisão (classificadores) (FARIA et al., 2016a).

A DN aplicada à classificação em FCDs geralmente trabalha em duas fases: *fase offline*, em que a partir de uma porção disponível de dados rotulados, o método constrói modelos de decisão (fase executada apenas uma vez); e *fase online*, em que os métodos usam modelos de decisão construídos na fase *offline* para classificar exemplos. Essa fase também é responsável pela atualização dos modelos de decisão (FARIA et al., 2016a; FARIA et al., 2016b; FARID et al., 2013; AL-KHATEEB et al., 2012a; AL-KHATEEB et al., 2012b; MASUD et al., 2010; MASUD et al., 2011a; MASUD et al., 2011b).

Embora diferentes trabalhos de DN tenham sido propostos para classificação de FCDs, esse ainda é um problema em aberto para CMFCD. Além disso, poucos trabalhos relacionados à CMFCD se preocupam em lidar com características importantes de FCDs como a latência extrema de rótulos. Motivados por esses problemas, esta seção apresenta um método de DN para CMFCD chamado ***Mult**I**-label** lear**N**ing **A**lgorithm for data **S**treams with **L**abel **C**ombination-based methods* (MINAS-LC), cujo objetivo principal é adaptar-se as mudanças de conceito de maneira não supervisionado considerando o problema de latência extrema de rótulos.

Está seção apresenta as seguintes contribuições:

- um novo método para CMFCD que é capaz de atualizar seu modelo de decisão sem a necessidade dos rótulos reais dos exemplos e sem nenhum *feedback* externo;
- um novo procedimento de DN para CMFCD capaz de detectar e aprender extensões de classes após mudanças de conceito;
- uma fase *online* que classifica exemplos usando os rótulos frequentes presentes nos

conjunto de rótulos dos microgrupos mais próximos aos exemplos. Com esse processo o modelo não fica restrito a classificar exemplos apenas com as combinações de rótulos vistas na fase *offline* e é uma tentativa adaptação às mudanças nas dependências das classes;

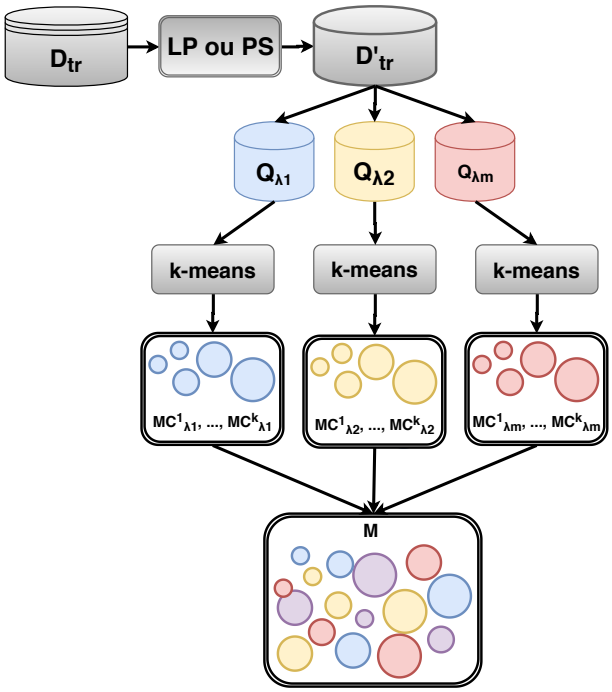
- a aplicação do método PS junto com mineração de *maximal itemsets* para reduzir o processamento de informações irrelevantes de meta-rótulos com poucos exemplos positivos.

### 5.1.2 Descrição do método MINAS-LC

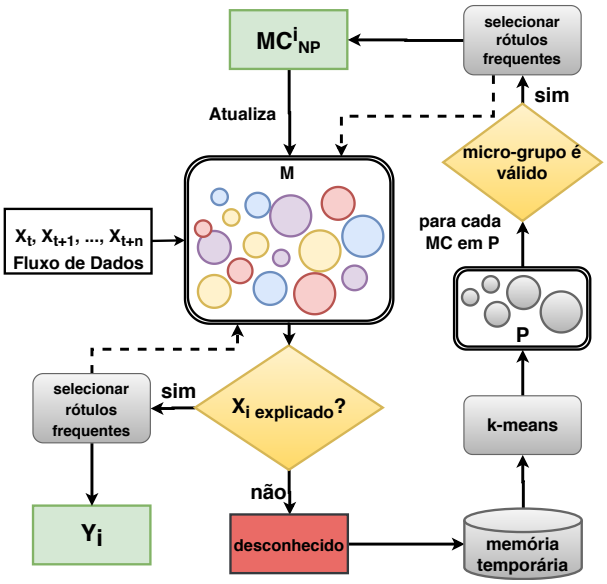
Está seção detalha o desenvolvimento do MINAS-LC, método de CMFCD que aplica DN para tratar mudança de conceito em cenários com latência extrema de rótulos. As seguintes notações serão usadas ao longo deste capítulo:

- $L = \{y_1, \ldots, y_q\}$ é o conjunto fechado com todos os $q$ rótulos representando as classes do problema;
- $X = \{x_1, \ldots, x_d\}$ representa um exemplo $d$-dimensional com $x_i$ sendo seu $i$-ésimo atributo;
- $Y = \{y_1, \ldots, y_p\}$ representa um conjunto de rótulos de um exemplo, com $p$ sendo o número de classes associadas ao exemplo e $Y \subset L$;
- $\lambda$ representa uma metaclasse, formada a partir da transformação de um conjunto de rótulos;
- $D_{tr} = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ representa um conjunto de treinamento de tamanho $n$, com $(X_i, Y_i)$ representando seu $i$-ésimo exemplo;
- $D'_{tr} = \{(X_1, \lambda_1), \ldots, (X_n, \lambda_n)\}$ representa o conjunto de treinamento multiclasse resultante da transformação de $D_{tr}$;
- $MC = \{LS, SS, n, t, \lambda\}$ representa um microgrupo, com $LS$ sendo a soma linear dos exemplos do grupo, $SS$ soma quadrada dos exemplos do grupo, $n$ número de exemplos no grupo, $t$ marcador de tempo (*timestamp*), e $\lambda$ a metaclasse associada ao grupo;
- $Q_\lambda = \{Q_{\lambda_1}, \ldots, Q_{\lambda_m}\}$ representa um conjunto formado por $m$ subconjunto $Q_{\lambda_i}$, cada um contendo exemplos de treinamento da metaclasse $\lambda_i$ ($Q_{\lambda_i} \subset D'_{tr}$);
- $M = \{MC^1_{\lambda_1}, \ldots, MC^k_{\lambda_1}, \ldots, MC^k_{\lambda_m}\}$ representa um modelo de decisão composto por microgrupos, sendo $MC^i_{\lambda_j}$ o $i$-ésimo microgrupo da $j$-ésima metaclasse;
- $P = \{MC^1_{NP}, \ldots, MC^b_{NP}\}$ representa um conjunto de PNs de tamanho $b$ formado pelo procedimento de DN, com cada PN sendo um microgrupo não rotulado $MC_{NP}$.

Figura 6 – Visão geral MINAS-LC




(a) *Fase offline*: fase inicial de treinamento. Primeiro, o MINAS-LC transforma a base de dados de treino multirrótulo $D_{tr}$ em uma multiclasse $D'_{tr}$ aplicando os método de transformação de problema LP ou o PS. Em seguida, os exemplos são divididos em subconjuntos $Q_{\lambda_j}$, cada subconjunto contendo apenas exemplos de uma metaclasses. Em cada um dos subconjuntos o MINAS-LC executa o *k-means* para formar $k$ microgrupos $MC_{\lambda_j}$ e rotula-os de acordo com a correspondente metaclasse do subconjunto. Como resultado, vários microgrupos $MC^i_{\lambda_j}$ são criados representando cada uma das metaclasses $\lambda_j$. Finalmente, a composição de todos os microgrupos formam o modelo de decisão $M$




(b) *Fase online:* fase de classificação e atualização. O MINAS-LC usa o modelo de decisão $M$ para classificar ou rejeitar exemplos $X_i$ do fluxo de dados. Se $M$ é capaz de explicar (classificar) $X_i$, o exemplo é rotulado com o rótulos frequentes $Y_i$ dentre os rótulos dos $\gamma$ microgrupos mais próximos de $X_i$. Caso contrário, $M$ rejeita $X_i$ e o exemplo é, então, rotulado como $desconhecido$ e enviado para memória temporária. Para atualizar $M$, o MINAS-LC executa o algoritmo *k-means* na memória temporária formando um novo conjunto de PNs $P$ (representados pelos microgrupos recém-formados). Em seguida, o MINAS-LC verifica a coesão e representatividade de cada novo microgrupo $MC^i_{NP} \in P$. Se um $MC^i_{NP}$ é válido, ele recebe os rótulos mais frequentes dentre os rótulos de seus $\gamma$ microgrupos mais próximos e é adicionado a $M$.

O MINAS-LC é formado por duas fases: a fase *offline* (Figura 6a), em que a partir de exemplos rotulados o MINAS-LC constrói o modelo de decisão; e a fase *online*, em que o MINAS-LC classifica os exemplos do fluxo de dados e atualiza o modelo de decisão para lidar com as mudanças de conceito.

#### 5.1.2.1 Fase *offline* do MINAS-LC

A fase *offline* é detalhada no Algoritmo 1. Essa fase é responsável pela construção do modelo de decisão considerando a disponibilidade de um conjunto inicial de dados multirrotulados $D_{tr}$. O MINAS-LC transforma esse conjunto de dados em um conjunto multiclasse $D'_{tr}$ aplicando os métodos de transformação de problema LP ou o PS.

**Algoritmo 1:** Fase *offline* do MINAS-LC

**Entrada:** $D_{tr}$: conjunto de dados de treinamento
$k_{ini}$: parâmetro para definir o valor de $k$ do k-means
$LCmethod$: método para transformação do problema (LP ou PS)
$minSup$: suporte mínimo para mineração de *item sets* frequentes

1 **retorna** $M$*: conjunto de modelos de decisão atualizado*
2 **início**
3 $M \leftarrow \emptyset$
4 **se** $LCmethod$ = *"LP"* **então**
5 $D'_{tr} \leftarrow \emptyset$ `// conjunto de dados multiclasse`
6 **para cada** *exemplo* $(X_i, Y_i)$ *em* $D_{tr}$ **faça**
7 $\lambda_i \leftarrow$ transformaEmMetaClasse($Y_i$)
8 $D'_{tr} \leftarrow D'_{tr} \cup (X_i, \lambda_i)$
9 **senão**
10 $D'_{tr} \leftarrow \emptyset$
11 $MI \leftarrow$maximalItemSetMining($D_{tr}, minSup$)
12 **para cada** *exemplo* $(X_i, Y_i)$ *em* $D_{tr}$ **faça**
13 **para cada** *maximal itemset* $Z_j$ *em* $MI$ **faça**
14 **se** $Z_j \subseteq Y_i$ **então**
15 $\lambda_j \leftarrow$ transformaEmMetaClasse($Z_j$)
16 $D'_{tr} \leftarrow D'_{tr} \cup (X_i, \lambda_j)$
17 $Q_\lambda \leftarrow \{\emptyset_{\lambda_1}, \ldots, \emptyset_{\lambda_m}\}$
18 **para cada** *exemplo* $(X_i, \lambda_i)$ *em* $D'_{tr}$ **faça**
19 $Q_{\lambda_i} \leftarrow Q_{\lambda_i} \cup X_i$
20 **para cada** *subconjunto* $Q_{\lambda_i}$ *em* $Q_\lambda$ **faça**
21 $k = \lceil |Q_{\lambda_i}| * k_{ini} \rceil$ `// definindo o valor de` $k$
22 $Grupos \leftarrow$ kMeans($Q_{\lambda_i}, k$)
23 **para cada** *grupo* $clus_j$ *em* $Grupos$ **faça**
`// calcula as propriedade dos microgrupos`
24 $MC^j_{\lambda_i} \leftarrow$criarMicroGrupo($clus_j$)
25 $M \cup MC^j_{\lambda_i}$
26 **retorna** $M, z$

Considerando a transformação por LP, o MINAS-LC simplesmente considera cada conjunto de rótulo distinto $Y_i$ como um único rótulo de uma nova classe (denominada como metaclasse $\lambda_i$) (passo 7). Portanto, cada exemplo $(X_i, Y_i) \in D_{tr}$, terá um representante $(X_i, \lambda_i)$ em $D'_{tr}$ (passo 8).

Considerando a transformação por PS, é necessário podar (*i.e.,* eliminar) conjunto de rótulos infrequentes, ou seja, combinações distintas de rótulos que são raras na base de dados. No entanto, ao invés de eliminar os exemplos rotulados com conjuntos infrequentes de rótulos, o PS os mantêm, mas considerando-os como exemplos rotulados com possíveis subconjuntos dos conjuntos infrequentes de rótulos (desde que esses subconjuntos sejam frequentes). Por exemplo,

considere o exemplo $(X_i, Y_i)$, o conjunto $FI = \{Y_{freq_1}, \ldots, Y_{freq_n}\}$ contendo $n$ conjunto de rótulos frequentes e, o conjunto $I = \{Y_{infreq_1}, \ldots, Y_{infreq_m}\}$ contendo $m$ conjunto de rótulos infrequentes. Se $Y_i \in I$, então ele deve ser podado pelo método PS, mas o conjunto de atributos $X_i$ deve ser reaproveitado. Para isso, $X_i$ é associado a subconjuntos de rótulos $Z_j$, sendo $Z_j \subset Y_i$, desde que $Z_j$ seja frequente ($Z_j \in FI$). Após isso o exemplo assumirá os valores de $(X_i, Z_j)$ na base de dados.

De acordo com as características do PS citadas acima, em conjunto com as características de mineração de conjuntos de itens frequentes (*Frequent Itemsets Mining*) (*e.g.,* encontrar coocorrências frequentes correlacionados de itens em um banco de dados (AGRAWAL et al., 1994) e reduzir o número de conjunto de itens sem perda de representatividade (UNO et al., 2004)). Para definir o conjunto de rótulos frequentes do PS, o MINAS-LC aplica um processo de mineração de item frequentes em $D_{tr}$ com o intuito de encontrar os *maximal itemsets* (UNO et al., 2004) $MI$ desse conjunto de dados. Desse modo, o MINAS-LC considera como conjunto de rótulos frequentes somente os conjuntos que são *maximal itemsets*.

Fournier-Viger et al. (2017) define *maximal itemsets* como conjuntos de itens frequentes, no qual nenhum de seus superconjuntos também sejam frequentes. *i.e., maximal itemsets* são os conjuntos de itens com o maior número de elementos. Nessa proposta, cada rótulo é considerado um item e, da mesma forma, um conjunto de rótulos é considerado um conjunto de itens (*itemset*). Portanto, considere um *maximal itemset* sendo $MI = \{Y | Y \in FI \wedge \nexists\, Y' \in FI$ sendo $Y \subset Y'\}$, com $FI$ representando os conjuntos de rótulos frequentes e $Y'$ um superconjuntos de $Y$. Usando o método PS com os *maximal itemsets* é possível reduzir o número de combinações possíveis de rótulos - logo o número de metaclasses que serão criadas - e, por aplicar a mineração de conjuntos de itens frequentes, os exemplos são reaproveitados de forma mais efetiva. No Algoritmo 1, do passo 9 ao passo 16, encontra-se os detalhes desse procedimento.

No passo 11, o MINAS-LC encontra os *maximal itemsets* $MI$ de $D_{tr}$. Qualquer algoritmo de mineração de *maximal itemset* pode ser usado, mas é preciso definir o suporte mínimo (no caso, o número de exemplos associados ao conjunto de rótulos em questão) para um conjunto de itens ser considerado frequente. Esse suporte mínimo é definido de acordo com o parâmetro $minSup$ (definido pelo usuário).

Para criar o novo conjunto de dados $D'_{tr}$, seguindo o método PS, o MINAS-LC forma metaclasses apenas a partir dos conjuntos de rótulos presente no conjunto $MI$ (passo 13). Então, para cada exemplo $(X_i, Y_i)$ em $D_{tr}$, o MINAS-LC cria um novo exemplo contendo o conjunto de atributos $X_i$ e o rótulo da metaclasse $\lambda_j$ (formada a partir do conjunto de rótulos $Z_j$, sendo $Z_j \subset Y_i$). Por fim, o novo exemplo $(X_i, \lambda_j)$ é adicionado a $D'_{tr}$ (do passo 13 ao passo 16). É importante destacar que, se um exemplo for rotulado com um conjunto de rótulos que tenha mais de um subconjuntos presentes no conjunto de *maximal itemsets*, esse exemplo deverá ser replicado para cada um desses subconjuntos. Por exemplo, considere o exemplo $(X, Y)$ e dois conjuntos de rótulos que fazem parte dos *maximal itemsets* $Z_1$ e $Z_2$ ($Z_1 \in MI$ e $Z_2 \in MI$). Se

$Z_1$ e $Z_2$ forem subconjuntos de $Y$ (*i.e.,* $Z_1 \subset Y$ e $Z_2 \subset Y$), então o conjunto de atributos $X$ deve ser replicado duas vezes, uma associada à metaclasse $\lambda_1$, formada a partir de $Z_1$ $(X, \lambda_1)$; e uma segunda vez, em que $X$ deve ser associada à metaclasse $\lambda_2$, formada a partir de $Z_2$ $(X, \lambda_2)$. Com isso, o MINAS-LC reaproveita os exemplos de forma mais efetiva ajudando a criar microgrupos mais representativos para cada metaclasse.

Após o procedimento de transformação, o MINAS-LC divide o novo conjunto de dados $D'_{tr}$ em subconjuntos $Q_{\lambda_j}$, um para cada metaclasse $\lambda_j$ (passo 19). Como visto anteriormente, o conjunto de rótulos associado a um exemplo pode ser composto por mais de um *maximal itemset*. Portanto, um exemplo poderá pertencer a mais de um subconjunto $Q_{\lambda_i}$ simultaneamente. Vale lembrar que quando isso ocorre, o exemplo é replicado uma vez para cada um dos *maximal itemsets*.

Para o agrupamento dos exemplos o MINAS-LC aplica o algoritmo *k-means*. O agrupamento é realizado em cada um dos subconjunto $Q_{\lambda_j}$, formando assim, $k$ microgrupos $MC^i$, com $i = \{1, \ldots, k\}$. Para cada $Q_{\lambda_j}$, é definido o valor de $k$ do *k-means* como $k_{ini}$-por cento do número de exemplos em um subconjunto $Q_{\lambda_j}$ e três como o número mínimo de exemplos para um microgrupo ser considerado válido na fase *offline*. A definição desse valor mínimo de exemplos foi baseada em outros trabalhos de DN da literatura (SPINOSA et al., 2009; MASUD et al., 2011a; AL-KHATEEB et al., 2012a; FARIA et al., 2016b). Na fase *online* do MINAS-LC, esse valor mínimo de exemplo é definido seguindo outras abordagens que serão discutidas na Seção 6.4.1.

Os microgrupos criados são rotulados de acordo com os rótulos dos exemplos usados para formá-los. Em outras palavras, microgrupos criados a partir dos exemplos do subconjunto $Q_{\lambda_j}$, serão rotulados com o rótulo da metaclasse $\lambda_j$. Lembrando que cada microgrupo $MC$ é formado por uma quíntupla, sendo $\lambda$ é um de seus elementos e representa o seu respectivo rótulo. O rótulo de cada microgrupo será usado na fase *online* para rotular exemplos do fluxo de dados. O limite de decisão de cada metaclasse $\lambda_j$ é definido pela união de seus $k$ microgrupos $MC_{\lambda_j}$. Por fim, o MINAS-LC retorna modelo de decisão inicial $M$, que é composto pelos $k$ microgrupos de cada subconjunto $Q_{\lambda_j}$ (passo 25).

Nós desenvolvemos essa fase baseada na fase *offline* do ***M**ult**I**class learN**i**ng **A**lgorithm for data **S**treams* (MINAS) (FARIA et al., 2016b), com diferença dos procedimento de transformação de problema, aplicados para lidar com dados multirrótulo. No mesmo trabalho, Faria et al. (2016b) destacam a importância de construir modelos de decisão baseados em microgrupos, já que podem evoluir ao longo do tempo através da adição novos microgrupos, remoção de microgrupos desatualizado ou atualização dos microgrupos existentes. Isso permite a atualização de modelos de decisão com baixo custo computacional.

#### 5.1.2.2 Fase *online* do MINAS-LC

A fase *online* do MINAS-LC, detalhada no Algoritmo 2, possui duas operações principais: *i) classificação*, em que o modelo de decisão classifica ou rejeita exemplos do fluxo de dados; e *ii) atualização*, em que através do procedimento de DN, exemplos rejeitados formam novos microgrupos para atualizar o modelo de decisão.

A fase *online* do MINAS-LC inicia-se verificando, para cada novo exemplo do fluxo de dados, se o modelo de decisão $M$ consegue classifica-los. Para isso, o MINAS-LC testa se o exemplo encontra-se dentro do raio de ao menos um de seus $\gamma$-microgrupos mais próximos ($\gamma$ é um parâmetro, cujo valor é definido pelo usuário). Um exemplo $X_i$ está dentro do raio de um microgrupo $MC$, quando a distância Euclidiana ($dist$, calculada no passo 12) entre $X_i$ e o centroide de $MC$ é menor ou igual ao raio de $MC$ (passo 13). Se o teste anterior for verdadeiro, o modelo de decisão é considerado apto a classificar o exemplo. O exemplo $X_i$ é rejeitado pelo modelo de decisão quando $X_i$ está fora do raio de todos seus $\gamma$-microgrupos mais próximos. Quando um exemplo é rejeitado ele recebe o rótulo $desconhecido$ e é enviado para a memória temporária ($ShortMem$) (procedimento é detalhado do passo 16 ao passo 18).

Uma vez o modelo de decisão sendo apto para classificar um exemplo $X_i$, o MINAS-LC atualiza o *timestamp* dos $\gamma$-microgrupos mais próximos de $X_i$ (passo 23) e transforma de volta os rótulos de suas correspondentes metaclasses $\lambda_j$ em conjunto de rótulos $Z_j$ (passo 25). O MINAS-LC armazena esses conjuntos $Z_j$ em $Voting$ para, posteriormente, calcular a frequência relativa de cada um de seus elementos (rótulos).

Para o processo de rotulagem de um exemplo $X_i$, o MINAS-LC calcula a frequência relativa $freq(y_j)$ de cada um dos rótulos $y_j$ do problema ($y_j \in L$). Esse cálculo é feito em relação à quantidade de vezes em que cada rótulo $y_j$ está contido nos conjuntos de rótulos $Z_k$ armazenados em $Voting$ (*i.e.,* conjunto de rótulos dos $\gamma$-microgrupos mais próximos de $X_i$) (passo 28). As frequências relativas dos um rótulos ($freq(y_j)$), são armazenadas em $FR$ (*i.e.,* $freq(y_j) \in FR$) e calculadas de acordo com a Equação 5.1, em que $\| \cdot \|$ retorna 1 se o predicado for verdadeiro e 0 caso contrário. Um rótulo $y_j$ é associado a um exemplo se a sua frequência relativa $freq(y_j)$ for maior ou igual a uma frequência mínima definida por $minFreq$ (parâmetro definido pelo usuário) (do passo 29 ao passo 31). Na Seção 6.4.1, será apresentada outra abordagem que ao invés de usar o parâmetro $minFreq$ para selecionar os rótulos, seleciona os $\lceil z \rceil$ primeiros rótulos mais frequentes.

$$freq(y_j) = \frac{1}{|Voting|} \sum_{Z_k \in Voting} \| \, y_j \in Z_k \, \| \tag{5.1}$$

Geralmente, os métodos com modelos de decisão baseados em microgrupos usam apenas o microgrupo mais próximo a um exemplo para classificá-lo (AGGARWAL et al., 2006; SPINOSA et al., 2009; FARIA et al., 2016b; NGUYEN et al., 2019). No caso do MINAS-LC, por se tratar de um método para classificação multirrótulo, são selecionados vários microgrupos

**Algoritmo 2:** Fase *online* do MINAS-LC

**Entrada:** $M$: modelo de decisão
$DS$: fluxo de dados multirrótulo
$L = \{y_1, \ldots, y_q\}$: conjunto dos $q$ rótulos de todas as classes do problema
$\Theta$: limite da memória temporária
$\omega$: tamanho da janela
$minFreq$: frequência mínima para um rótulo ser considerado frequente na classificação
$\gamma$: número de microgrupos mais próximos para rotular um exemplo
$minExClu$: número mínimo de exemplos em um microgrupo válido

**retorna** $M$*: modelo de decisão atualizado*
**início**
  $ShortMem \leftarrow \emptyset$ // memória temporária
  $t \leftarrow 0$ // timestamp do exemplo atual
  $Voting \leftarrow \emptyset$ // armazena conjuntos de rótulo para classificação
  **para cada** *exemplo* $X_i$ *em* $DS$ **faça**
    $Y_i \leftarrow \emptyset$ // rótulos do novo exemplo
    $t \leftarrow t + 1$
    // $\gamma$-microgrupos mais próximos
    $M_{closest} \leftarrow$microGruposMaisProximos$(M, X_i, \gamma)$
    **para cada** *microgrupo* $MC_j$ *em* $M_{closest}$ **faça**
      // marca se o exemplo será rejeitado ou não
      $unk \leftarrow$ VERDADEIRO
      $dist \leftarrow$distanciaEuclidiana$(X_i, MC_j)$
      **se** $dist \leq raio(MC_j)$ **então**
        $unk \leftarrow$ FALSO
        PARAR
    **se** $unk$ **então**
      // modelo de decisão rejeita o exemplo
      $Y_i \leftarrow desconhecido$
      $ShortMem \leftarrow ShortMem \cup (X_i, Y_i)$
      **se** $|ShortMem| \geq \Theta$ **então**
        procedimentoDN$(M, ShortMem, \gamma, minExClu, t)$
    **senão**
      // modelo de decisão classifica o exemplo
      **para cada** *microgrupo* $MC_j$ *em* $M_{closest}$ **faça**
        $MC_j.t \leftarrow t$ // atualiza o timestamp de $MC$
        $\lambda_j \leftarrow MC_j.\lambda$ // rótulo da metaclasse do microgrupo
        // transforma metaclasses em conjunto de rótulos
        $Z_j \leftarrow$transformarEmConjuntoRotulos$(\lambda_j)$
        $Voting \leftarrow Voting \cup Z_j$
      // armazena a frequência relativa de cada rótulo
      $FR \leftarrow \{freq(y_1), \ldots, freq(y_q)\}$
      calcularFrequenciaRelativa$(FR, |Voting|)$
      **para cada** *frequência relativa de rótulo* $freq(y_j)$ *em* $FR$ **faça**
        **se** $freq(y_j) \geq minFreq$ **então**
          // atribui apenas rótulos frequentes
          $Y_i \leftarrow Y_i \cup L.y_j$
    **se** $(timeStamp \mod \omega) = 0$ **então**
      classificacaoForcada$(M, ShortMem, \omega)$
      removeMicroGruposObsoletos$(M, \omega)$
  **se** $ShortMem \neq \emptyset$ **então**
    classificacaoForcada$(M, ShortMem)$
  **retorna** $M$

ao entorno dos exemplos - os $\gamma$-mais próximos - para classificação. Com isso, é possível diminuir a propagação de erros referentes ao processo de rotulagem dos microgrupos, pois ao selecionar os rótulos mais frequências entre os rótulos de vários microgrupos, a classificação tende a ser mais precisa. Além disso, dado a alta sobreposição das classes em problemas multirrótulo, usar apenas um microgrupo na classificação é insuficiente para determinar se um exemplo é *desconhecido* ou não. Portanto, é possível evitar por exemplo, que um exemplo seja rejeitado desnecessariamente por estar fora do raio de seu microgrupo mais próximo (pois o mesmo pode ter um raio de pouco alcance), estando dentro do raio de outros microgrupos que não são os mais próximos, mas possuem raios maiores.

Quando o modelo de decisão rejeita um exemplo $X_i$, o mesmo recebe o rótulo de $desconhecido$ e é enviado para memória temporária. Exemplos $desconhecidos$ podem indicar ocorrências de mudanças de conceito, portanto, o modelo de decisão deve ser atualizado. Para atualização do modelo de decisão, o MINAS-LC executa seu procedimento de DN sempre que a memória temporária atinge seu limite $\Theta$ (parâmetro definido pelo usuário) (Algoritmo 2, step 20). Esse procedimento consiste em criar microgrupos a partir de exemplos $desconhecidos$, sendo esses microgrupos coesos e representativos o suficiente para representar **P**adrões-**N**ovidade (PNs) e atualizar o modelo de decisão.

O procedimento de DN (Algoritmo 3) cria novos microgrupos executando o algoritmo *k-means* na memória temporária. Para evitar a definição manual dos centroides iniciais do *k-means*, o MINAS-LC executa o algoritmo *Leader* (SPATH, 1980), que define os exemplos propensos a serem os centroides do agrupamento de acordo com uma distância mínima entre eles (aqui definido pelo maior raio $r$ dos microgrupos do modelo de decisão). O algoritmo *Leader* defini os centroides calculando a distância entre um novo exemplo e os centroides definidos até então. Se todas essas distâncias forem superiores a $r$, o novo exemplo é considerado um novo centroide. Os exemplos são processados na ordem que foram adicionados na memória temporária (passo 4).

Os microgrupos criados são validados de acordo com a metodologia proposta em Faria et al. (2016b), na qual a largura da Silhueta do microgrupo é usada para medir sua coesão e a quantidade de exemplos do microgrupo é usada para medir sua representatividade. Dessa maneira, um microgrupo será válido se sua Silhueta for maior do que zero e se for composto por um valor mínimo de exemplos, definido pelo parâmetro $minEx$ (parâmetro definido pelo usuário) (passo 7).

Após o processo de validação, o MINAS-LC remove da memória temporária os exemplos usados para formar os microgrupos válidos (passo 8). O próximo passo é rotular os novos microgrupos $MC^i_{NP}$. Para isso, o MINAS-LC aplica um processo semelhando ao usado para classificar exemplos: seleciona os conjuntos de rótulos dos $\gamma$-microgrupos mais próximos ao $MC^i_{NP}$ (passo 9 até o passo 14), seleciona os rótulos cuja frequência relativa é maior ou igual ao parâmetro $minFreq$ e atribui os rótulos selecionados ao novo microgrupo $MC^i_{NP}$ (passo 15 até o passo 21). Ainda, é necessário marcar os novos microgrupos com o *timestamp* atual (passo 22),

**Algoritmo 3:** procedimentoDN - atualiza o modelo de decisão

**Entrada:** $M$: modelo de decisão
$ShortMem$: memória temporária
$\gamma$: número de microgrupos mais próximos para rotular um microgrupo
$minExClu$: número mínimo de exemplo para um microgrupo ser válido
$t$: timestamp para atribuir ao novo microgrupo

1 **início**
2 $Voting \leftarrow \emptyset$ // armazena os conjuntos de rótulos
3 $r \leftarrow$maiorRaio($M$)// maior raio dos microgrupos
4 $Grupos \leftarrow$ kMeansComLeader($ShortMem, r$)
5 **para cada** *grupo* $clus_i$ *em* $Grupos$ **faça**
6 $MC^i_{NP} \leftarrow$criarMicroGrupo($clus_i$)
7 **se** *numeroExemplos(*$MC^i_{NP}$*)* $> minEx$ *E silhueta(*$MC^i_{NP}$*)* $> 0$ **então**
8 removerExemplos($ShortMem, MC^i_{NP}$)
9 $centroide \leftarrow$centroide($MC^i_{NP}$)
// $\gamma$-microgrupos mais próximos
10 $M_{closest} \leftarrow$ microGruposMaisProximos($M, centroide, \gamma$)
11 **para cada** *microgrupo* $MC_j$ *em* $M_{closest}$ **faça**
12 $\lambda_j \leftarrow MC_j.\lambda$ // rótulo da metaclasse do microgrupo
// metaclasse para conjunto de rótulos
13 $Z_j \leftarrow$transformarEmConjuntoRotulos($\lambda_j$)
14 $Voting \leftarrow Voting \cup Z_j$
// armazena a frequências relativas
15 $FR \leftarrow \{freq(y_1), \ldots, freq(y_q)\}$
16 $FR \leftarrow$calcularFrequenciaRelativa($F, |Voting|$)
17 **para cada** *frequencia relativa de rótulo* $freq(y_j)$ *em* $FR$ **faça**
18 **se** $freq(y_j) \geq minFreq$ **então**
19 $Y \leftarrow Y \cup L.y_j$
// rotula o novo microgrupo $MC^i_{NP}$
20 $\lambda \leftarrow$ transformaEmMetaClasse($Y$)
21 $MC^i_{NP}.\lambda \leftarrow \lambda$
22 $MC^i_{NP}.t \leftarrow t$// atribui timestamp atual
23 $M \leftarrow M \cup MC^i_{NP}$// atualiza o modelo de decisão

pois no futuro eles podem se tornar obsoletos e deverão ser removidos do modelo de decisão. O MINAS-LC usa os *timestamps* dos microgrupos para verificar se são obsoletos ou não. Por fim, o modelo de decisão é atualizado recebendo os novos microgrupos (passo 23).

Para evitar a influência de informações obsoletas, de tempos em tempos o MINAS-LC remove os microgrupos que não foram usados na classificação de exemplos durante a última janela de tempo $\omega$ (parâmetro definido pelo usuário) (Algoritmo 2, passo 34). Faria et al. (2016b) aplicam um procedimento semelhante, mas enviando os microgrupos removidos para uma memória de longo prazo para futuras análises relacionadas a Contextos Recorrentes. Esse tema

será tratado em trabalhos futuros, pois problemas de FCDs multirrótulo trazem novos desafios no tratamento de contextos recorrentes.

O procedimento DN também deve tratar ruídos e *outliers*, pois podem ser confundidos com PNs. Uma forma de tratar esses problemas é limpar a memória temporária removendo exemplos $desconhecidos$ remanescentes que não formaram nenhum microgrupos (FARIA et al., 2016b). Para efeito de comparações mais justas entre o MINAS-LC e os métodos da literatura que não tratam ruídos e *outliers*, o MINAS-LC classifica de forma forçada os exemplos remanescentes na memória temporária ao invés de remove-los. Masud et al. (2011a) aplicam um processo semelhante. Dessa forma, se um exemplo permanece na memória temporária por mais de $\omega$-*timestamps*, ele é forçado à classificação (Algoritmo 2, passo 33). Da mesma forma, se ao fim do fluxo de dados existirem exemplos na memória temporária, esses exemplos também serão forçados à classificação. A classificação forçada acontece da mesma que a classificação normal, usando os rótulos frequentes dentre os rótulos dos $\gamma$-microgrupos mais próximos ao exemplo.

O MINAS-LC tem como principais características a utilização de um modelo unificado de decisão composto por microgrupos, que podem representar mais de uma classe simultaneamente. Dessa forma, é possível classificar exemplos de FCDs multirrótulo usando os microgrupos formados na fase *offline* ou usando suas extensão formadas na fase *online* a partir de exemplos não rotulados. Por conta dessas características o MINAS-LC pode ser atualizado sem os rótulos reais dos exemplos e sem qualquer *feedback* externo.

## 5.2 MINAS-BR

### 5.2.1 Visão geral do MINAS-BR

Apesar das diferentes abordagens de **D**etecção de **N**ovidade (DN) para classificação multiclasse em FCDs (SPINOSA et al., 2009; AL-KHATEEB et al., 2012a; MASUD et al., 2011a; FARIA et al., 2016b), a sua aplicação para **C**lassificação **M**ultirrótulo em **F**luxos **C**ontínuos de **D**ados (CMFCD) é pouco explorada na literatura. A DN é uma tarefa importante para tal problema, especialmente porque poucos autores têm dado atenção para características importantes da CMFCD como: *Latência Extrema de Rótulos*, cenário em que os rótulos reais dos exemplos nunca são disponibilizados para a atualização dos modelos de decisão dos métodos (SOUZA et al., 2015a); e *Evolução de Conceito*, fenômeno que caracteriza o surgimento de novas classes ao longo do fluxo de dados, que não são conhecidas pelo modelo de decisão até o momento (MASUD et al., 2011a).

Motivados pelos problemas em aberto discutidos anteriormente, esta seção apresenta o ***Mult**I**-label** learNing **A**lgorithm for data **S**treams with **B**inary **R**elevance transformation* (MINAS-BR), novo método capaz de detectar evoluções e mudanças de conceitos em problemas de CMFCD. O MINAS-BR é uma extensão avançado do ***Mult**I**class** learNing **A**lgorithm for data*

***S**treams* (MINAS) (FARIA et al., 2016b), método que aplicada DN na classificação multiclasse de FCDs para cenários de latência extrema de rótulos (mais detalhes podem ser encontrados na Seção 4.4 do Capítulo 4).

Apesar dos resultados satisfatórios e das novas perspectivas para DN em FCDs que o MINAS apresentara, o método não é apto para CM. O MINAS-BR vêm para tratar tais problemas. O método aplica estratégias baseadas na tradicional técnica de transformação de problema ***B**inary **R**elevance* (BR) (TSOUMAKAS et al., 2009), constrói diversos modelos de decisão, um para cada classe do problema. Esses modelos de decisão são usados na fase *online* do método, em que cada exemplo do fluxo de dados é testado em um dos modelos de decisão. Dessa forma é possível classificar exemplos em várias classes simultaneamente. No procedimento de DN do MINAS-BR, os modelos de decisão que representam as classes conhecidas do problema podem ser estendidos (para adaptar às mudanças de conceito) ou novos modelos de decisão podem ser construídos (para adaptar às evoluções de conceito).

As principais contribuições desta seção são:

- um novo método para CMFCD que atualiza seus modelos de decisão sem a necessidade dos rótulos reais do exemplos e sem nenhum tipo de *feedback* externo;
- um novo procedimento de DN capaz de detectar e adaptar modelos de decisão tanto às mudanças de conceito, quanto às evoluções de conceito;

### 5.2.2 Descrição do método MINAS-BR

Está seção detalha o desenvolvimento do ***Mult**I**-label** lear**N**ing **A**lgorithm for data **S**treams with **B**inary **R**elevance transformation* (MINAS-BR), método de CMFCD que trata mudança de conceito e evolução de conceito em cenários com latência extrema de rótulos. As seguintes notações serão usadas ao longo desta seção:

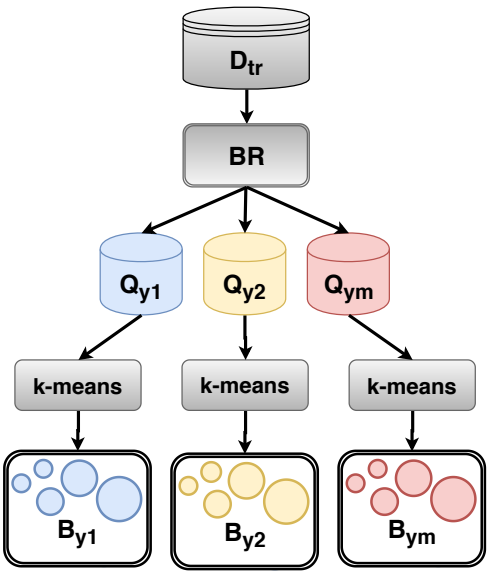
- $X = \{x_1, \ldots, x_d\}$ representa um exemplo $d$-dimensional com $x_i$ sendo seu $i$-ésimo atributo;
- $Y = \{y_1, \ldots, y_p\}$ representa um conjunto de rótulos de um exemplo, com $p$ sendo o número de classes associadas ao exemplo;
- $L_{tr} = \{y_1, \ldots, y_m\}$ é o conjunto de $m$ classes aprendidas na fase de treinamento (*i.e.,* conjunto de classes conhecidas);
- $D_{tr} = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ representa o conjunto de exemplos de treinamento, com $n$ exemplos, sendo $(X_i, Y_i)$ o $i$-ésimo exemplo e $Y_i \subset L_{tr}$;
- $MC = \{LS, SS, n, t, y\}$ representa um microgrupo, com $LS$ sendo a soma linear dos exemplos do grupo, $SS$ a soma quadrada dos exemplos do grupo, $n$ número de exemplos

do grupo, $t$ marcador de tempo do grupo (*timestamp*), e $y$ rótulo da classe associada ao grupo;

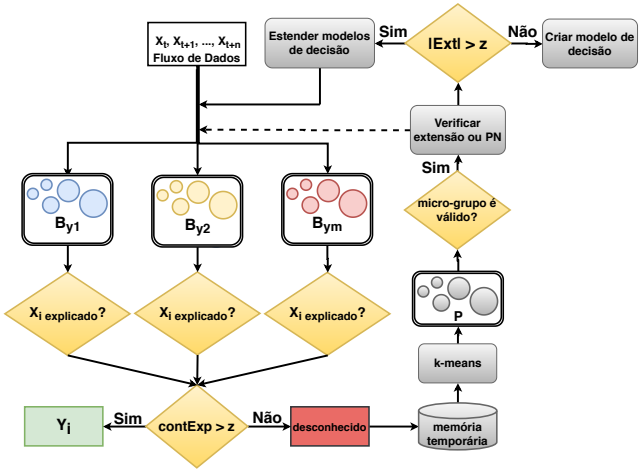
- $Q_y = \{Q_{y_1}, \ldots, Q_{y_m}\}$ representa o conjunto formado por $m$ subconjuntos, em que cada subconjunto $Q_{y_j}$ contém exemplos de treinamento associados à classe $y_j$ ($Q_y \subset D_{tr}$);
- $B = \{MC_1, \ldots, MC_k\}$ representa um modelo decisão com $k$ microgrupos;
- $M = \{B_{y_1}, \ldots, B_{y_m}\}$ é um conjunto de $m$ modelos de decisão, em que cada modelo de decisão $B_{y_j}$ contém $k$ microgrupos formados por exemplos associados à classe $y_j$;
- $DS$ representa um fluxo contínuo de dados.

O MINAS-BR, ilustrado na Figura 7, possui duas fases: Fase *Offline* (Figura 7a), em que através de uma porção disponível de dados rotulados ($D_{tr}$), o método constrói $m$ modelos de decisão, cada um representando uma classe conhecida do problema; e a Fase *Online* (Figura 7b), responsável por classificar novos exemplos do fluxo de dados, atualiza seus modelos decisão para adaptar-se às mudanças de conceito, e constrói novos modelos de decisão para adaptar-se às evoluções de conceito.

Figura 7 – Visão geral do MINAS-BR




(a) *Fase offline*: fase inicial de treinamento. Primeiro, o MINAS-BR divide $D_{tr}$ em subconjuntos $Q_{y_i}$. Em cada um dos subconjunto o MINAS-BR executa o *k-means* para formar $k$ microgrupos $MC_{y_i}$ e rotula-os de acordo com a correspondente classe do subconjunto. Como resultado, vários modelos de decisão $B_{y_i}$ são formados, um para cada classe do problema.




(b) *Fase online:* fase de classificação e atualização. O MINAS-BR usa os modelos de decisão $B_y$ para classificar ou rejeitar exemplos $X_i$ do fluxo de dados. Se $X_i$ for classificado, ele recebe o rótulos dos modelos de decisão que o explicaram. Caso contrário, é enviado para memória temporária. O procedimento de DN é executado na memória temporária, formando um conjunto de novos microgrupos $P$. Os microgrupos válidos poderão ser extensões dos modelos de decisão, ou PNs e serem usados para criar novos modelos de decisão.

#### 5.2.2.1 fase *offline* do MINAS-BR

A fase *offline* do MINAS-BR (detalhada no Algoritmo 4) considera disponível um conjunto de dados rotulados para treinamento $D_{tr}$. O método então, divide esse conjunto de dados em subconjuntos $Q_{y_j}$, cada um contendo exemplos associados à classe $y_j$, com $y_j \in L_{tr}$ (passo 5). Essa fase diferencia-se da fase *offline* do método MINAS, pelo fato de que nos subconjuntos do MINAS-BR, cada exemplo pode estar contido em mais de um subconjunto simultaneamente, visto que os exemplos multirrótulo podem estar associados a mais de uma classe.

Após dividir o conjunto inicial de dados em subconjuntos $Q_y$, o método executa o algoritmo *k-means* em cada um dos subconjuntos, criando assim, $k$ microgrupos $MC_i$, com $i = 1, \ldots, k$, para cada $Q_y$. O valor de $k$ do *k-means* de cada subconjunto, foi definido como um porcentagem $k_{ini}$ dos números de exemplos do subconjunto e foi definido três como número mínimo de exemplos para formar um microgrupo. Esses valores foram definidos baseando-se em métodos de DN da literatura (SPINOSA et al., 2009; FARIA et al., 2016b; MASUD et al., 2011a; AL-KHATEEB et al., 2012a).

Após o procedimento de agrupamento e criação de microgrupos, cada conjunto de microgrupos correspondente $B_{y_j}$ é usado como modelo de decisão para a classe $y_j$ (passo 11). O algoritmo retorna então, um conjunto $M$ contendo $m$ modelos de decisão, cada um representando uma classe $y_j$ ($y_j \in L_{tr}$). O MINAS-BR também calcula a cardinalidade de rótulo $z$ do conjunto de treinamento, um parâmetro importante para a fase *online* do método.

#### 5.2.2.2 fase *online* do MINAS-BR

A fase *online* do MINAS-BR, detalhada pelo Algoritmo 5, inicia-se verificando qual modelo de decisão $B_{y_j}$ ($B_{y_j} \in M$) é capaz de classificar cada um dos exemplos $X_i$ ($X_i \in DS$). Para isso, em cada modelo de decisão $B_{y_j}$, o MINAS-BR calcula a distância Euclidiana ($dist$) entre o novo exemplo e o centroide de seu microgrupo mais próximo ($MC_{closest}$). Se essa distância $dist$ for menor ou igual ao raio do microgrupo $MC_{closest}$ (passo 13), o modelo de decisão $B_{y_j}$ é considerado capaz de classificar o novo exemplo $X_i$. Para todo modelo de decisão $B_{y_j}$ capaz de classificar o exemplo $X_i$, o método armazena a distância $dist$ e o rótulo $y_j$ correspondentes em um conjunto de candidatos ($Voting$) para a decisão final do método referente a classificação (passo 15).

Um dos maiores desafios na CMFCD é a DN, visto que novos exemplos podem pertencer tanto à classes já conhecidas pelo método, como também à possíveis classes emergentes ao longo do fluxo de dados. O MINAS-BR usa a cardinalidade de rótulo $z$ do conjunto de treinamento como um parâmetro para decidir se um novo exemplo deverá ser classificado nas classes conhecidas pelo método, ou se ele é um candidato forte para representar uma classe novidade. Esse parâmetros é usado porque, se um exemplo pertencer apenas as classes conhecidas do

**Algoritmo 4:** Fase *offline* do MINAS-BR

**Entrada:** $D_{tr}$: conjunto de dados para treinamento
$k_{ini}$: parâmetro para definir o valor de $k$ do k-means
**Saída:** $M$: conjunto com $m$ modelos de decisão
$z$: cardinalidade de rótulo do conjunto de treinamento $D_{tr}$

```
1  início
2      M ← ∅
3      z ← calcularCardinalidadeRotulo(D_tr)
4      Q_y ← {∅_{y_1}, ..., ∅_{y_m}}
5      para cada exemplo (X_i, Y_i) em D_tr faça
6          para cada classe y_j em Y_i faça
7              Q_{y_j} ← Q_{y_j} ∪ X_i
8      para cada subconjunto Q_{y_j} em Q_y faça
9          k = ⌈|Q_{y_j}| * k_ini⌉
10         Grupos ← kMeans(Q_{y_j}, k)
11         para cada grupo grupo em Grupos faça
               // Calcula as propriedades de um microgrupo
12             MC ←criarMicroGrupo(grupo)
13             B_{y_j} ← B_{y_j} ∪ MC
           // Armazena o modelo de decisão que representa a
              classe y_j
14         M ← M ∪ B_{y_j}
15     retorna M, z
```

problema, é provável que esse exemplo seja explicado (classificados) por um número de modelos de decisão aproximadamente igual a cardinalidade de rótulos do conjunto de treinamento. Caso contrário, é possível que esse exemplos pertença a ao menos uma classe novidade. Desse modo, após um exemplo ser testado em todos os modelos de decisão $B_{y_j}$, verifica-se quantos modelos de decisão ($countExp$, incrementado no passo 14) podem explicar o exemplo. Se $countExp$ é maior do que a cardinalidade de rótulos do conjunto de treinamento $z$ (passo 16), então o exemplo é rotulado de acordo com as classes representadas pelos $\lceil z \rceil$-microgrupos com menor $dist$ dentre todos os microgrupos capazes de classificar esse exemplo (passo 17). Caso contrário, o exemplo é rotulado como *desconhecido* e enviado para a memória temporária ($ShortMem$) para análises futuras (passo 23). É importante destacar a necessidade de atualizar os *timestamps* $t$ dos microgrupos selecionados para rotular os exemplos. Com isso, evita-se que esses microgrupos sejam considerados obsoletos e removidos do modelo de decisão.

O MINAS-BR considera os $\lceil z \rceil$-microgrupos mais próximos (logo os $\lceil z \rceil$ modelos de decisão mais próximos) para evitar classificar o exemplo em muitas classes, situação comum em problemas em que as classes conhecidas estão muito próximas umas das outras. Portanto, se um exemplo não for considerado *desconhecido*, o mesmo é classificado com as classes de seus $\lceil z \rceil$ modelos de decisão mais similares (mais próximos, considerando distância Euclidiana de seus

**Algoritmo 5:** Fase *online* do MINAS-BR

**Entrada:** $M$: conjunto com $m$ modelos de decisão
$DS$: fluxo contínuo de dados
$\Theta$: limite da memória temporária
$\omega$: tamanho da janela
$\mathcal{F}$: fator para calcular o limiar dos PNs
$z$: cardinalidade de rótulos do conjunto de treinamento

1 **retorna** $M$*: conjunto de modelos de decisão atualizado*
2 **início**
3 $ShortMem \leftarrow \emptyset$
4 $t \leftarrow 0$ // marcador de tempo dos exemplos (*timestamp*)
5 $Voting \leftarrow \emptyset$ // armazena as classes para classificar novos exemplos
6 $countExp \leftarrow 0$
7 $nPN \leftarrow 0$ // Número de PNs
8 **para cada** *exemplo* $X_i$ *em* $DS$ **faça**
9 $Y_i \leftarrow \emptyset$ // Rótulos do novo exemplo
10 $t \leftarrow t + 1$
11 **para cada** *modelo de decisão* $B_{y_j}$ *em* $M$ **faça**
12 $\{dist, MC_{closest}\} \leftarrow$ microGrupoMaisProximo$(X_i, B_{y_j})$
13 **se** $dist \leq$ *calcularRaio(*$MC_{closest}$*)* **então**
14 $countExp \leftarrow countExp + 1$
15 $Voting \leftarrow Voting \cup \{dist, MC_{closest}\}$
16 **se** $contExp > z$ **então**
17 $Closest \leftarrow$ microGruposMaisProximoClassificar$(Voting, z)$
18 **para cada** *microgrupo* $MC$ *em* $Closest$ **faça**
19 $Y_i \leftarrow Y_i \cup MC.y$ // pega os rótulos para classificar o exemplo
20 $MC.t \leftarrow t$ // atualiza o *timestamp* de $MC$
21 **senão**
22 $Y_i \leftarrow desconhecido$
23 $ShortMem \leftarrow ShortMem \cup (X_i, Y_i)$
24 **se** $|ShortMem| \geq \Theta$ **então**
25 procedimentoDN$(M, ShortMem, \mathcal{F}, z, t, nPN)$
26 **se** $(t \mod \omega) = 0$ **então**
27 removerExemplosObsoletos$(ShortMem, \omega)$
28 removerMicroGruposObsoletos$(M, \omega)$
29 **retorna** $M$

microgrupos). Assim é possível manter a cardinalidade média dos rótulos dos dados que vão sendo explicados pelo conjunto de modelos de decisão atual.

Os exemplos *desconhecidos* armazenados na memória temporária podem representar mudanças de conceito, evoluções de conceito e ruídos. Para diferenciar os exemplos entre esses três diferentes fenômenos, um procedimento de DN é necessário. O procedimento de DN do MINAS-BR inicia-se quando a memória temporária alcança seu limite $\Theta$ (parâmetro definido pelo usuário) (Algoritmo 5, passo 25). Esse procedimento consiste em formar novos microgrupos com exemplos *desconhecidos* armazenados na memória temporária. O MINAS-BR então verifica se esses novos microgrupos são *extensões* das classes conhecidas - nesse caso os modelos de decisão existentes devem ser atualizados - ou se eles representam um **P**adrão-**N**ovidade (PN) - nesse caso um novo modelo de decisão deve ser criado. O MINAS-BR considera *extensões* como mudanças nos conceitos das classes conhecidas, por isso atualiza os modelos de decisão atuais e, considera os PNs como indícios do surgimento de classes novidades, por isso cria novos modelos de decisão para adaptar a essas possíveis evoluções de conceito.

O procedimento de DN, detalhado no Algoritmo 6, forma novos microgrupos executando o algoritmo *k-means* na memória temporária. Para definir a quantidade e quais exemplos são melhores centroides iniciais do *k-means*, o MINAS-BR executa o algoritmo *Leader* (SPATH, 1980). Dessa forma, não é necessário a definição manual do valor de $k$. O algoritmo *Leader* exige um parâmetro inicial que define a distância mínima entre um centroide e outro. Esse valor é definido de acordo como o maior raio ($r$) de microgrupos dentre todos os microgrupos de todos modelos de decisão atuais.

Após a criação dos novos microgrupos, o MINAS-BR avalia se são coesivos e representativos o suficiente para atualização. Um microgrupo é coeso se o mesmo é formado por um número de exemplos maior ou igual a o parâmetro $minEx$ (parâmetro definido pelo usuário) e, um microgrupo é representativo se sua silhueta é maior do que zero (Algoritmo 6 passo 8). Esse procedimento é o mesmo utilizado por em Faria et al. (2016b) e é executado para cada um dos modelos de decisão.

Quando um microgrupo $MC$ é considerado valido, os exemplos utilizados para formá-lo são removidos da memória temporária (Algoritmo 6 passo 9). Agora, é necessário decidir se os novo microgrupos válidos representarão *extensões* ou PNs.

Para cada modelo de decisão $B_{y_j}$ em $M$, o MINAS-BR calcula a distância Euclidiana $dist$ entre o centroide de um novo microgrupo $MC$ e o centroide do microgrupo mais próximo de $MC$ em $B_{y_j}$ (denominado $MC_{closest}$) (Algoritmo 6 passo 11). Se $dist$ for menor ou igual ao desvio padrão $sd$ das distâncias entre os exemplos de $MC_{closest}$ e seu centroide, multiplicado pelo fator $\mathcal{F}$ (parâmetro definido pelo usuário) (Algoritmo 6 passo 11), então $MC$ é considerado *extensão* e o modelo de decisão $B_{y_j}$ é armazenado no conjunto ($Ext$) de potenciais candidatos a serem estendidos por $MC$. Caso contrário, $MC$ é considerado um PN.

**Algoritmo 6:** procedimentoDN - atualiza os modelos de decisão

```
Data: M: conjunto com m modelos de decisão
      ShortMem: memória temporária
      F: fator para calcular o limiar de PNs
      z: cardinalidade de rótulo do conjunto de treinamento
      t: timestamp // marcador de tempo para os novos
      microgrupos
1     nPN: número de PNs
2  início
3     r ← maiorRaio(M)
4     Grupos ← kMeansComLeader(ShortMem, r)
5     para cada grupo em Grupos faça
6        MC ←criarMicroGrupo(grupo)
7        Ext ← ∅ // Modelos a serem estendidos
8        if numeroExemplos(MC) > minEx E silhueta(MC) > 0 then
9           removerExemplos(ShortMem, MC)
10          para cada modelo de decisão B_yj em M faça
11             (dist, MC_closest) ←microGrupoMaisProximo(MC, B_yj)
12             sd ← calcularDesvioPadrao(MC_closest)*F
13             se dist ≤ sd então
14                Ext ← Ext ∪ B_yj
15          se |Ext| > z então
16             MC.y ← y_j // MC agora representa a classe y_j
17             para cada B_yj em Ext faça
18                B_yj ← B_yj ∪ MC
19          senão
20             nPN ← nPN + criarModelo(Ext, MC, M, nNP)
```

Pensando na sobreposição de classes originada de problemas multirrótulo, usar apenas a distância entre os microgrupos pode ser insuficiente para diferenciar entre *extensões* e PNs. O MINAS-BR usa, não só a distância entre os microgrupos, mas também a cardinalidade de rótulos $z$ do conjunto de treinamento como parâmetro para decidir se um novo microgrupo difere significativamente dos microgrupos presentes nos modelos de decisão. Portanto, a decisão final é tomada considerando a quantidade de elementos em $Ext$ (*i.e.,* $|Ext|$, representando quantos modelos de decisão consideraram o novo microgrupo como extensão). Se $|Ext|$ for maior que $z$ (Algoritmo 6 passo 15), é provável que mudanças de conceito nas classes representadas pelos modelos de decisão presentes em $Ext$ estão acontecendo. Assim, com o intuito de adaptar os modelos de decisão presentes em $Ext$ às mudanças de conceito, $MC$ é adicionado em cada um dos modelos de decisão presentes em $Ext$ (Algoritmo 6 passo 17 e 18). Por outro lado, se $|Ext|$ for menor ou igual a $z$, $MC$ é considerado um PN, indicando uma provável evolução de conceito. Então, um novo modelo de decisão deve ser criado para representar essa classe emergente. (Algoritmo 6 passo 20).

**Algoritmo 7:** criarModelo - cria um novo modelo de decisão para representar um novo PN

**Entrada:** $Ext$: modelos de decisão que consideraram $MC$ como extensão
$MC$: microgrupo considerado PN
$M$: conjunto de $m$ modelos de decisão
$nPN$: número de PNs

1 **retorna** $numNovoModelo$: *apenas um modelo de decisão*
2 **início**
// $MC$ agora representa a classe $PN_{nPN+1}$
3 $MC.y \leftarrow PN_{nPN+1}$
4 $B_{PN} \leftarrow \emptyset$ // novo modelo
5 $B_{PN} \leftarrow B_{PN} \cup MC$ // insere $MC$ em $B_{NP}$
6 **para cada** $B_{y_j}$ *em* $Ext$ **faça**
7 $MC_{closest} \leftarrow$ copiarMicroGrupoMaisProximo($MC, B_{y_j}$)
// $MC_{closest}$ agora representa a classe $PN_{nPN+1}$
8 $MC_{closest}.y \leftarrow PN_{nPN+1}$
9 $B_{PN} \leftarrow B_{PN} \cup MC_{closest}$
10 $M \leftarrow M \cup B_{PN}$
11 **retorna** *1*

O Algoritmo 7 detalha o procedimento de criação de um novo modelo de decisão. Um novo modelo $B_{PN}$ é formado pelo novo microgrupo $MC$ e pelas cópias dos microgrupos mais próximos a $MC$ presentes em cada modelo de decisão contido em $Ext$ (Algoritmo 7 passo 6). Os microgrupos que formam o novo modelo de decisão recebem um novo rótulo sequencial ($PN_1, PN_2, \dots$) indicando qual PN ele representa. Portanto, se o conjunto atual de modelos de decisão $M$ tem $nPN$ modelos de decisão criados para representar PNs, o próximo modelo de decisão a ser criado receberá o rótulo $PN_{nPN+1}$. Após o novo modelo de decisão ser criado, ele é adicionado ao conjunto de modelos de decisão $M$. A partir dessa atualização, o conjunto de modelos de decisão $M$ será capaz de classificar novos exemplos com o rótulo $PN_i$ e, novos microgrupos formados nos próximos procedimentos de DN, poderão ser extensões desse novo modelo de decisão representado por $PN_i$.

O MINAS-BR usa cópias de microgrupos próximos de $MC$ para considerar potenciais sobreposições entre as classes novidade e as classes conhecidas. Quando isso ocorre, os exemplos *desconhecidos* que formaram o novo microgrupo podem pertencer não só a uma classe novidade, como também a outras classes conhecidas. Isso explica o motivo do MINAS-BR usar o parâmetro $z$ para definir se o microgrupo será uma extensão ou um PN, possibilitando assim, alguns modelos de decisão considerarem esse novo microgrupo como *extensão* (provavelmente pois seus microgrupos estão próximos ou sobrepostos ao novo microgrupo), mas como decisão final ele ser escolhido para representar um PN. Assim, ao usar as cópias dos microgrupos próximos de $MC$, sempre que um novo exemplo pertencente à uma classe conhecida $y_j$ (*i.e.*, $y_j \in L_{tr}$) e também à uma classe novidade $y_{new_j}$, ele poderá ser classificado em ambas as classes. Uma vez

que é provável que esse novo exemplo esteja dentro do raio do microgrupo da classe $y_j$ copiado para o modelo de decisão da classe $y_{new_j}$, fazendo parte de ambos os modelos de decisão. Por outro lado, se o MINAS-BR não usar as cópias dos microgrupos para formar o novo modelo de decisão, só seria possível classificar o exemplo na classe conhecida $y_j$. Esse procedimento é uma forma do MINAS-BR lidar com dependências entre classes novidade e classes conhecidas, pois em cenários reais é raro uma classe ser totalmente independente.

É importante destacar que, como os PNs são formados de forma não supervisionada e sem *feedback* externo, eles não estão associados diretamente às classes do problema. Assim, quando uma nova classe aparece no fluxo de dados, ela pode ser representada por um ou mais PNs. Somente um especialista de domínio poderá dizer se os PNs encontrados pelo algoritmo representam de fato uma nova classe do problema e se diferentes PNs podem estar relacionados a uma mesma classe (FARIA et al., 2016b).

Nos Algoritmos 5, 6 e 7, todos os componentes de um microgrupo ($MC = \{LS, SS, n, t, y\}$) são criados sempre que ocorre a criação de um microgrupo, mas para facilitar a visualização, só são mostrados a classe $y$ e o *timestamp* $t$ dos microgrupos criados ou atualizados.

Para evitar a influência de informações passadas, de tempos em tempos o MINAS-BR remove os microgrupos obsoletos, ou seja, microgrupos que não classificaram exemplos nos últimos $\omega$ *timestamps* (parâmetro definido pelo usuário) (Algoritmo 5 passo 26 e 28). O MINAS (FARIA et al., 2016b) aplica um processo semelhante, mas armazenando os microgrupos removidos em uma memória de longo prazo para analisar posteriormente, a ocorrência de Conceitos Recorrentes. Neste trabalho, a detecção de conceitos recorrentes é deixado para trabalhos futuros, pois lidar com esse fenômeno em problemas de CMFCD é ainda mais desafiador e não é bem explorado na literatura.

Um procedimento de DN deve tratar ruídos e *outliers*, pois esses fenômenos podem ser confundidos com mudanças ou evoluções de conceito. Para lidar com esse problema, o MINAS-BR limpa a memória temporária removendo exemplos *desconhecidos* que não formaram microgrupos nos últimos $\omega$ *timestamp* (Algoritmo 5 passo 26 e 27). O fato dos exemplos permanecerem na memória temporária por um longo período, é um forte indicio de esses exemplos serem ruídos.

É importante destacar as principais característica do MINAS-BR que possibilitam a atualização de forma totalmente não supervisionada, lidando assim com problemas de latência infinita de rótulos. Portanto destacamos: *i)* o fato dos modelos de decisão serem formados por microgrupos, sendo possível atualiza-los sem os rótulos reais dos exemplos. Isso ocorre por conta das características dos microgrupos de aditividade, incrementalidade e por serem criados à partir de agrupamento; *ii)* o fato de existirem vários modelos de decisão, um para cada classe do problema. Com isso é possível criar novos modelos de decisão à medida que novas classes surgem, estando elas sobrepostas ou não.

## 5.3 Considerações Finais

Este capítulo apresentou o método ***Mult**I-label learN**ing** **A**lgorithm for data **S**treams with **L**abel **C**ombination-based methods* (MINAS-LC) e o ***Mult**I-label learN**ing** **A**lgorithm for data **S**treams with **B**inary **R**elevance transformation* (MINAS-BR) para CMFCD com latência extrema de rótulos.

O MINAS-LC lida com problemas apenas com mudanças de conceito. O seu modelo de decisão e composto por microgrupos multirrotulados sendo capaz de classificar exemplos em várias classes simultaneamente e evoluir ao longo do fluxo de dados. Foram propostas duas variações do método: utilizando o método de transformação de problema ***L**abel **P**owerset* (LP) e, utilizando o método ***P**runed **S**ets* (PS) com mineração de conjunto de itens frequentes.

O MINAS-BR lida com problemas tanto com mudanças de conceito, como com evoluções de conceito. Ele possui um conjunto de modelos de decisão, um para cada classe do problema. Esses modelos de decisão podem ser entendidos adaptando-se às mudanças de conceito, ou novos modelos de decisão podem ser criados, adaptando-se às evoluções de conceito.

O próximo capítulo apresenta os experimentos realizados envolvendo os dois métodos propostos neste trabalho.

# Capítulo 6
# EXPERIMENTOS

No capítulo anterior foram apresentados os métodos MINAS-LC e MINAS-BR. Este capítulo descreve os detalhes sobre os experimentos realizados para validar esses métodos. Na Seção 6.1 serão apresentadas as bases de dados sintéticas e reais usadas. Na Seção 6.2, serão mostrados os métodos da literatura para comparação com os métodos propostos. Na Seção 6.3 será discutida a metodologia de avaliação empregada nos experimentos do MINAS-LC, e será apresentada uma nova metodologia desenvolvida para avaliar o MINAS-BR e métodos de DN para CM com latência extrema de rótulos. Por fim, serão detalhados os experimentos, bem como uma discussão sobre os resultados obtidos do MINAS-LC na Seção 6.4, e do MINAS-BR na Seção 6.5.

## 6.1 Base de Dados

Esta seção descreve as bases de dados utilizadas nos experimentos. Como foram propostos dois métodos, um que considera Evoluções de Conceito (MINAS-BR) e outra que não considera (MINAS-LC). Foram utilizadas praticamente as mesmas bases de dados, mas com adaptações para cada um dos métodos.

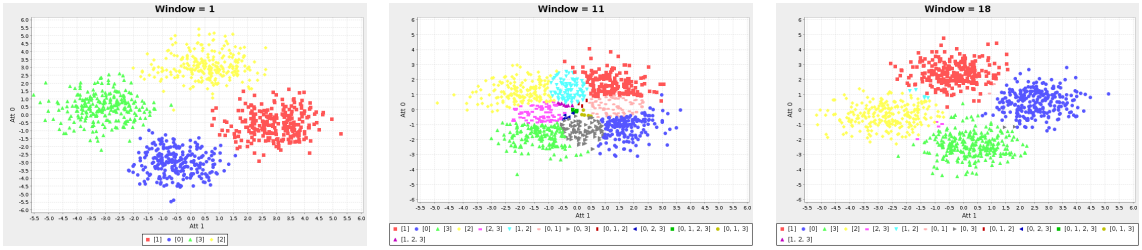
### 6.1.1 Bases de dados sem evolução de conceito

As bases de dados descrita nesta seção serão usadas para experimentos com o MINAS-LC, em que não há ocorrências de evoluções de conceitos.

Nós usamos seis bases de dados sintéticas e cinco reais, todas formadas apenas por atributos numéricos. As bases de dados sintética *MOA-3C-2D*[1] e *MOA-5C-2D*[2] foram criadas no *framework* MOA (*Massive Online Analisys*) (BIFET et al., 2010) e as *4CRE-V1*[3], *4CRE-V2*[4] e

[1] gerada no MOA com três classes dois atributos
[2] gerada no MOA com cinco classes dois atributos
[3] Four Classes Rotating with Expansion V1
[4] Four Classes Rotating with Expansion V2

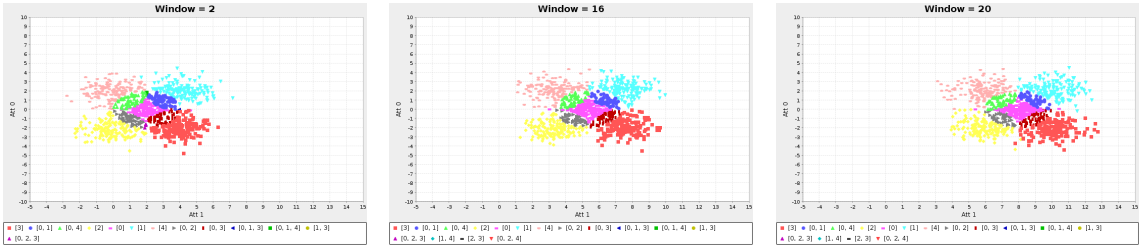
Figura 8 – Seis diferentes momentos da base de dados 4CRE-V2. Cada janela representa 1000 *timestamps*




(a) Janela 1 (b) Janela 11 (c) Janela 18

(d) Janela 81 (e) Janela 101 (f) Janela 120

Figura 9 – Três diferentes momentos da base de dados 5CVT. Cada janela representa 1000 *timestamps*




(a) Janela 2 (b) Janela 16 (c) Janela 20

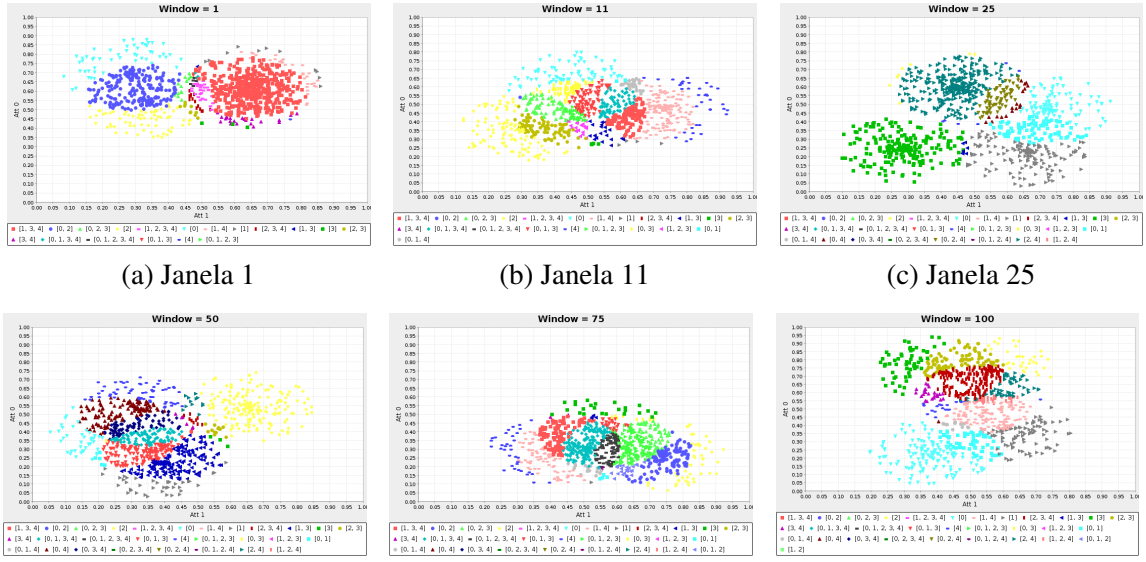
*5CVT*[5] [6] foram geradas originalmente em Souza et al. (2015b). Essas as bases de dados foram criadas a partir da função *Radial Basis Function* (RBF). Essa função permite apenas a criação de bases de dados multiclasse, em que cada classe é representada por um grupo de exemplos. Portanto, para adapta-las para cenários multirrótulo, consideramos cada sobreposição de grupo como uma assinatura multirrótulo.

Nas Figuras 8, 9 e 10, são apresentados respectivamente os comportamentos das bases de dados *4CRE-V2*, *5CVT* e *MOA-5C-2D*. Nas figuras são apresentados diferentes momentos das bases de dados, em que os exemplos são representados por diferentes símbolos e cores para cada combinação de rótulos distinta. Dessa forma, é possível diferenciar quando há sobreposição de classes, ou seja, quando há assinaturas multirrótulo.

5 Five Classes Vertical Translation

6 versões original disponíveis em: <https://sites.google.com/site/nonstationaryarchive/datasets>

Figura 10 – Seis diferentes momentos da base de dados MOA-5C-2D. Cada janela representa 1000 *timestamps*




(a) Janela 1 (b) Janela 11 (c) Janela 25

(d) Janela 50 (e) Janela 75 (f) Janela 100

Foram utilizadas bases de dados sintéticas por conta da falta de bases de dados reais para problemas de FCDs multirrótulo. Através de bases de dados sintéticas, também é possível construir ambientes controlados, em que certas restrições são impostas com o intuito de facilitar a avaliação e a demonstração dos experimentos. As restrições impostas foram principalmente: número de classes, velocidade das mudanças de conceito, cardinalidade de rótulos, dependências de rótulo e distribuição de classes.

Utilizamos os primeiros 10% do fluxo de dados como base de dados de treinamento e, os 90% restante dos dados foram usando para teste. Garantimos a presença de todas as classes do problema na base de dados de treinamento. Dessa forma, simulamos problemas apenas com mudanças de conceito, evitando o surgimento de novas classes ao longo do fluxo de dados (Evoluções de Conceito). Por fim, pré-processamos as bases de dados reais para evitar problemas como classes infrequentes e alto desbalanceamento. Então, consideramos apenas as classes associadas mais de 5% dos exemplos e, construímos os conjuntos de dados de treinamento selecionando - sempre que possível - o mesmo número de exemplos por classe.

Na Tabela 5 apresentamos as características das, já pré-processadas, bases de dados usadas nos experimentos. Para cada base de dados, mostramos o número total de exemplos (representado pela coluna $|DS|$, nesse caso consideramos os exemplos do conjunto de treinamento e do conjunto de teste), o número de atributos ($|X|$), o número de classes do problema ($|L|$) e a cardinalidade de rótulo ($z$, nesse caso consideramos a cardinalidade de rótulos de todo a base de dados). Para as bases de dados sintéticas, também apresentamos o intervalo de exemplos ($speed$) entre deslocamentos consecutivos dos grupos - que representam as classes no espaço. Como as bases de dados reais são originalmente estacionárias, suas classes não se movimentam no espaço.

Tabela 5 – Características das bases de dados sem Evoluções de Conceito.

| Nome | $\|DS\|$ | $\|X\|$ | $\|L\|$ | $z$ | $speed$ |
|---|---|---|---|---|---|
| Mediamill | 41,701 | 120 | 18 | 3.82 | - |
| NUS-WIDE | 186,290 | 128 | 9 | 1.71 | - |
| Scene | 2,407 | 294 | 6 | 1.07 | - |
| Yeast | 2,417 | 103 | 12 | 4.15 | - |
| Reuters | 5,694 | 500 | 40 | 1.36 | - |
| 4CRE-V1 | 125,000 | 2 | 4 | 1.01 | 1,000 |
| 4CRE-V2 | 183,000 | 2 | 4 | 1.22 | 1,000 |
| 5CVT | 24,000 | 2 | 5 | 1.23 | 1,000 |
| MOA-3C-2D | 100,000 | 2 | 3 | 1.48 | 1,000 |
| MOA-5C-2D | 100,000 | 2 | 5 | 2.02 | 1,000 |

### 6.1.2 Base de dados com evoluções de conceito

As bases de dados descrita nesta seção serão usadas nos experimentos com o MINAS-BR. Nesses experimentos há ocorrências de Evoluções de Conceito.

Utilizamos duas base de dados sintéticas e cinco reais, formadas apenas por atributos numéricos. A base de dados sintética *MOA-EC*[7] foi criada no *framework* MOA (BIFET et al., 2010) e a *4CRE-V2-EC*[8] foi gerada originalmente por Souza et al. (2015b). A base de dados *4CRE-V2-EC* é a mesma da Tabela 5 (*4CRE-V2*), mas adaptada para duas das suas quatro classes aparecerem como novidade ao longo do fluxo de dados. A mesma abordagem descrita na Seção 6.1.1 para adaptar, com assinaturas multirrótulo, as bases de dados geradas pela função RBF foram empregadas aqui.

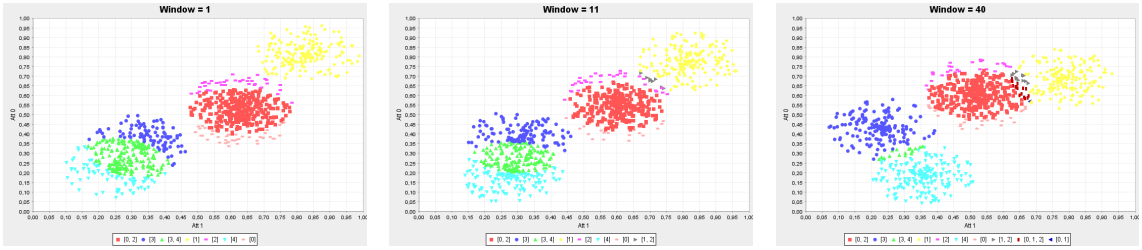
Nas Figuras 11 e 12, são apresentados respectivamente os comportamentos das bases de dados *4CRE-V2-EC* e *MOA-EC*. Destacamos os momentos das bases de dados para ilustramos diferentes comportamentos dos dados e também, os momentos em que novas classes surgem.

Novamente utilizamos bases de dados sintéticas por conta da falta de bases de dados reais para problemas de FCDs multirrótulo com evoluções de conceito e, também, porque é possível construir ambientes controlados, em que certas restrições são impostas com o intuito de facilitar a avaliação e a demonstração dos experimentos. Por exemplo, em ambientes reais sem nenhum tipo de controle, os exemplos recém-chegados podem pertencer simultaneamente a número muito alto de classes novidade. Um cenário desafiador para classificação multirrótulo, visto que não se tem o conhecimento da maioria dessas classes.

Na fase *offline* do MINAS-BR, usamos os primeiros 10% do fluxo para criar os modelos de decisão iniciais. O restante dos exemplos usamos para teste durante a fase *online*. É importante destacar que, o conjunto de classes conhecidas do problema $L_{tr}$ é formado apenas pelas classes dos exemplos presentes nos primeiros 10% do fluxo. Qualquer exemplo rotulado com classes

[7] gerada no MOA com Evoluções de Conceito
[8] *Four Classes Rotating with Expansion V2* com Evolução de Conceito

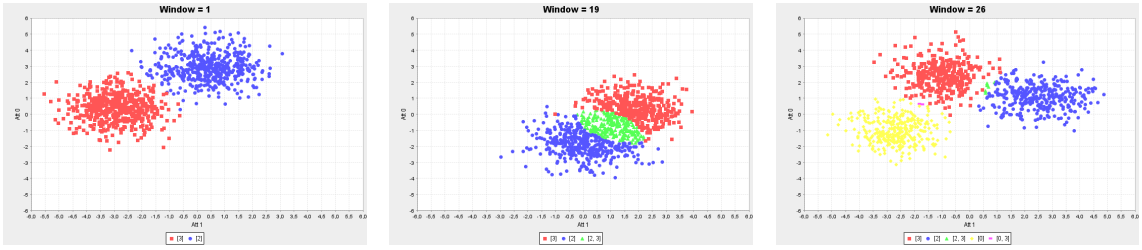
Figura 11 – Seis diferentes momentos da base de dados MOA-EC. Cada janela representa 1000 *timestamps*




(a) Janela 1 (b) Janela 11 (c) Janela 40

(d) Janela 41 (e) Janela 80 (f) Janela 81

Figura 12 – Seis diferentes momentos da base de dados 4CRE-V2-EC. Cada janela representa 1000 *timestamps*




(a) Janela 1 (b) Janela 19 (c) Janela 26

(d) Janela 63 (e) Janela 73 (f) Janela 96

diferentes das existente no conjunto $L_{tr}$, ao longo dos 90% dos dados restantes, são consideradas classes novidade.

Como CMFCD é um cenário desafiador, nós restringimos as evoluções de conceito em todos as bases de dados, garantindo o surgimento de apenas uma classe novidade por janela de avaliação (exceto para a base de dados *NUS-WIDE*, pois as duas classes novidade são altamente dependentes). Além dessa restrição, nós realizamos outros pré-processamentos nas bases de dados reais, na tentativa de controlar melhor o ambiente de validação do método:

Tabela 6 – Características das bases de dados com Evolução de Conceito

| Base de dados | $\|DS\|$ | $\|X\|$ | $\|L_{tr}\|$ | $\|L_{new}\|$ | $z$ | $speed$ |
|---|---|---|---|---|---|---|
| Mediamill-EC | 41,701 | 120 | 15 | 3 | 3.78 | - |
| NUS-WIDE-EC | 182,920 | 128 | 7 | 2 | 1.71 | - |
| Scene-EC | 2,407 | 294 | 4 | 2 | 1.07 | - |
| Yeast-EC | 2,417 | 103 | 9 | 3 | 4.15 | - |
| Reuters-EC | 5694 | 500 | 35 | 5 | 1.36 | - |
| MOA-EC | 100,000 | 2 | 5 | 2 | 1.54 | 1500 |
| 4CRE-V2-EC | 146,000 | 2 | 2 | 2 | 1.19 | 1,000 |

1. rótulos infrequentes foram ignorados (um rótulo é frequente se é atribuído a mais de 5% dos exemplos da base de dados);

2. as classes com menor coocorrência foram selecionadas como classe novidade do problema. Para isso, calculamos o grau de dependência de todos os pares de classe. Para medir o grau de dependência de um par de classes $(y_i, y_j)$, nós calculamos o *Jaccard Index* (Equação 6.1), sendo $a$ é o número de ocorrências de $y_i$ sem $y_j$, $b$ é o número de ocorrências de $y_j$ sem $y_i$, e $ab$ é o número de ocorrências simultâneas de $y_i$ e $y_j$ (quanto maior o *Jaccard Index* maior o grau de dependência). Após medir as dependências de todos os pares, esses são ranqueados em ordem crescente em relação ao grau de dependência (o primeiro par do ranking será o com menor grau de dependência). As classes novidade candidatas devem pertencer aos primeiros pares dos rankings e não pertencerem a nenhum dos últimos pares do ranking. Como resultado, é possível garantir, na maioria das vezes, que classes novidades tenha baixa dependência com outras classes;

3. o conjunto de treinamento foi formado selecionando (sempre que possível) o mesmo número de exemplos para cada classe, ignorando os exemplos das classes novidades.

$$JaccardIndex = \frac{ab}{ab + a + b} \tag{6.1}$$

Na Tabela 6 apresentamos as características das, já pré-processadas, bases de dados usadas nos experimentos. As colunas $|DS|$, $z$ e $speed$ representam as mesmas características da Tabela 5, já as coluna $|L_{tr}|$ representa o número de classes conhecidas e a $|L_{new}|$ o número de classes novidade.

## 6.2 Métodos de Comparação

Nós comparamos o MINAS-BR e o MINAS-LC com os métodos mostrados na Tabela 7. Os métodos estão divididos em *upper bounds* ($E_aBR, E_aCC$ e $E_aMLHT$) e *lower bounds* ($CC, PS, MLHT$ e $iSOUPtree$). Todos eles estão disponíveis no *framework* MOA.

Tabela 7 – Métodos de CMFCD da literatura e suas respectivas referências. A notação $E$ representa o uso de comitês (*Ensembles*) de classificadores e a notação $_a$ representa a aplicação da técnica ADWIN pelo método.

| Acrônimo | Método | Referência |
|---|---|---|
| $CC$ | Classifiers Chains | (READ et al., 2011) |
| $PS$ | Pruned Sets | (READ et al., 2008) |
| $MLHT$ | Multilabel Hoeffding Tree with PS at leaves | (READ et al., 2012a) |
| $iSOUPtree$ | incremental Structured OUtput Prediction tree | (OSOJNIK et al., 2017) |
| $E_aBR$ | Comitês de BR | (READ et al., 2012a) |
| $E_aCC$ | Comitês de CC | (READ et al., 2012a) |
| $E_aMLHT$ | Comitês de MLHT | (READ et al., 2012a) |

Os métodos *lower bounds* constroem seus modelos de decisão com os exemplos do conjunto de treinamento (os primeiros 10% da base de dados) e, não atualizam seus modelos de decisão após sua construção. Por outro lado, os métodos *upper bounds*, são comitês de classificadores e sempre terão acesso aos rótulos reais dos exemplos para atualizar seus modelos de decisão. Além disso, esses métodos usam a técnica ***AD****aptive* ***WIN****dowing* (ADWIN) para detectar mudanças de conceito e, sempre que essas mudanças são detectadas, os métodos reconstroem o pior modelo de decisão de seus comitês.

## 6.3 Avaliação

Está seção apresenta as metodologias de avaliação empregadas nos experimentos do MINAS-LC e do MINAS-BR. Pelo fato do MINAS-BR lidar com Evoluções de Conceito através de procedimentos de DN em ambientes com latência extrema de rótulos, desenvolvemos uma nova metodologia de avaliação que será detalhada na Seção 6.3.2.

### 6.3.1 Avaliação do MINAS-LC

Para avaliação dos experimentos usamos a metodologia *Prequetial* (GAMA et al., 2009) com janelas de avaliação ($w$) de tamanho $w_{tam} = \frac{|DS|}{50}$ (o fluxo de dados completo, com exceção dos primeiros 10% dos exemplos usados para construir do modelo de decisão, dividido em cinquenta janelas). Não definimos esse valor com um tamanho fixo, pois as bases de dados são de tamanhos diferentes e com cinquenta janelas é possível visualizar melhor os gráficos e o comportamento dos métodos ao longo do fluxo de dados.

Geralmente o MINAS-LC não classifica todos os exemplos em uma janela de avaliação. Isso acontece por conta da opção de rejeitar exemplos e envia-los para memória temporária. Assim, ao final da janela de avaliação pode existir exemplos que não foram explorados para formar PNs e que ainda não foram forçados à classificação. Portanto, ao invés de calcular a média das medidas de avaliação com base no tamanho total de uma janela de avaliação, nós

as calculamos com base no número de exemplos classificados pelo MINAS-LC. Por exemplo, dada às janelas de avaliação de tamanho $w_{tam} = 1000$, se na janela $w_t$ o método classificar $800$ exemplos e $200$ exemplos permanecerem na memória temporária, então devemos calcular as medidas de avaliação de $w_t$ com base nesses $800$ exemplos e não em $1000$. Os $200$ exemplos remanescentes serão avaliados na próxima janela $w_{t+1}$, pois lá que serão usados para formar novos microgrupos ou forçados à classificação. Como resultado disso, conseguimos comparar de forma justa o MINAS-LC contra os outros métodos que não rejeitam exemplos no processo de classificação.

### 6.3.2 Nova metodologia de avaliação para CMFCD com evolução de conceito e latência extrema de rótulos

O MINAS-BR lida com FCDs multirrótulo em cenários com latência extrema de rótulo, no qual é impossível atualizar modelos de decisão de forma supervisionada quando novas classes emergem ao longo do fluxo de dados. Isso porque não é possível definir o significado de novas classes sem conhecimento prévio, *i.e.,* os PNs detectados surgem sem rótulo.

No procedimento de DN do MINAS-BR (Algoritmo 6), são criados PNs para representar o surgimento de uma nova classe. Porém, não é possível definir qual classe específica esses PNs representam. Ainda, cada classe pode ser representada por vários PNs, mas também não é possível saber o número exato. Portanto, é impossível aplicar medidas de avaliação tradicionais como as apresentadas na Seção 3.1 (Capítulo 3). Restrições como essas foram investigadas em Faria et al. (2015), mas a metodologia proposta pelos autores trata apenas métodos de classificação multiclasse. Baseado nessa metodologia, nós desenvolvemos uma nova e específica para CMFCD, cujo objetivo é associar os PNs às classes novidade medindo seus graus de dependências.

Para avaliar o MINAS-BR, primeiro é necessário associar seus PNs às classes do problema. Esse procedimento é realizado apenas para avaliação. Portanto, os rótulos reais dos exemplos são acessados apenas para avaliação do método. O procedimento de associação de PNs ocorre antes da avaliação e é realizado periodicamente de acordo com a necessidade do usuário. Para associar um dado PN $PN_i$ a uma classe novidade $y_{new_j}$, primeiro nós calculamos os valores da tabela de contingência representada na Tabela 8. Cada célula da tabela representam os seguintes valores:

- $a$: o número de ocorrências simultâneas entre $PN_i$ e $y_{new_j}$;
- $b$: o número de ocorrências de $y_{new_j}$ sem $PN_i$;
- $c$: o número de ocorrências de $PN_i$ sem $y_{new_j}$;
- $d$: o número de ocorrências sem $y_{new_j}$ e sem $PN_i$.

Tabela 8 – Tabela de contingência para $PN_i$ e classe $y_{new_j}$.

| | $NP_i$ | $\neg NP_i$ |
|---|---|---|
| $y_{new_j}$ | $a$ | $b$ |
| $\neg y_{new_j}$ | $c$ | $d$ |

Utilizando os valores da Tabela 8, nós calculamos o grau de dependência entre $PN_i$ e $y_{new_j}$ através da F1 (Equação 6.2). Nós usamos essa medida pois ignora o valor da célula $d$ da tabela de contingência, que para esse caso não é interessante.

$$F1 = \frac{a}{a + \frac{b+c}{2}} \tag{6.2}$$

Nós construímos a matriz de contingência e calculamos o grau de dependência de cada $PN_i$ sem associação, a cada classe novidade $y_{new_j}$ descoberta até o momento. Um PN $PN_i$ é associada a classe novidade $y_{new_j}$, que obtiver o maior valor, *i.e.,* a classe novidade que o $PN_i$ é mais dependente. Caso o valor de F1 de $PN_i$ para todas as classes novidades for igual a zero, então consideramos que o MINAS-BR não tem informação suficiente para fazer a associação e o $PN_i$ continua sem representar nenhuma classe novidade (isso ocorre quando o MINAS-BR erra na detecção do PN).

Após o procedimento de associação, todos os exemplos preditos - ou que posteriormente serão preditos - como $PN_i$ serão classificados na classe $y_{new_j}$. Agora as medidas de avaliação convencionais podem ser aplicadas.

Durante o processo de avaliação, devemos considerar também os exemplos *desconhecidos*. Esses são exemplos que não são usados para formação de PNs ou extensões. Portanto, na metodologia proposta, nós medimos a média macro da taxa de exemplos desconhecidos em cada janela de avaliação (*Macro Unknown Rate* (UnkRM) Equação 6.3). Então, a avaliação completa da predição do MINAS-BR é dada pela F1M juntamente com a UnkRM.

$$UnkRM = \frac{1}{|L|} \sum_{i=1}^{|L|} \frac{desc_i}{n_i} \tag{6.3}$$

Na Equação 6.3, $desc_i$ representa o número de exemplos *desconhecidos* da classe $y_i$, $n_i$ o número total de exemplos da classe $y_i$ e $|L|$ o número total de classes até o momento da avaliação. Nós calculamos as médias das medidas de avaliação baseadas em rótulos de forma macro, pois refletem melhor o comportamento da predição dos métodos perante evoluções de conceito ao longo do fluxo de dados. Assim, também facilita a visualização e avaliação do procedimento de DN nos gráficos.

## 6.4 Análise dos Resultados do MINAS-LC

Esta seção apresenta os experimentos executados e as discussões dos resultados obtidos pelo método MINAS-LC. Os experimentos têm o objetivo de avaliar a performance preditiva do MINAS-LC em comparação com os métodos *lower bounds* em cenários com latência extrema de rótulos. Além disso, nós avaliamos o MINAS-LC em comparação com os métodos *upper bounds*, que apesar de ser uma comparação injusta, visto que esses métodos têm acesso aos rótulos reais dos exemplos, esperamos ter resultados competitivos em relação aos deles.

### 6.4.1 Sensibilidade dos parâmetros

Antes de apresentar os resultados do MINAS-LC em comparação com os métodos da literatura, nós analisamos como diferentes configurações de parâmetros influenciam o desempenho preditivo do MINAS-LC.

Na Tabela 9 nós apresentamos os parâmetros do MINAS-LC e suas respectivas descrições e, na Tabela 10, apresentamos diferentes configurações de valores dos parâmetros. Embora na Tabela 10 não tenhamos apresentado valores para o parâmetro *minSup*, definimos seu valor como 1% do conjunto de dados de treinamento (*i.e.,* para um conjunto de rótulos ser considerado frequente ele deverá estar associado a pelo menos 1% dos exemplos na base de dados). Com um valor menor atribuído ao parâmetro *minSup*, garantimos que todos os rótulos estarão presentes entre os *maximal itemsets*.

Cada uma das configurações presentes na Tabela 10 assume valores definidos para tratar comportamentos observados nas bases de dados (*e.g.,* velocidade de deslocamento dos grupos, alta dependência de classes, cardinalidade de rótulos, etc.). Além disso, tentamos tornar a definição de alguns valores dos parâmetros a mais automática possível. Os intervalos desses valores foram encontrados através de experimentos empíricos, executando o MINAS-LC com valores baseados nos trabalhos de DN propostos na literatura (FARIA et al., 2016b; SPINOSA et al., 2009; AL-KHATEEB et al., 2012a; MASUD et al., 2011a).

- **Configuração 1 (C1)**: definida para lidar com bases de dados cujas classes são próximas no espaço e deslocam-se em menor velocidade. A seguir descrevemos os valores dos parâmetros e suas motivações de acordo com a C1:
  - definimos o $\Theta = 30$, pois quando as classes são próximas, poucos exemplos são rejeitados pelo modelo de decisão e enviados para memória temporária. Assim, com um valor baixo de $\Theta$ é possível executar com maior frequência o procedimento de DN e detectar as sutis mudanças dos microgrupos;
  - definimos $minEx = 3$ para facilitar a validação de representatividade dos novos microgrupos criados. Esperamos que assim seja possível criar mais microgrupos com poucos exemplos na memória temporária;

Tabela 9 – Descrição dos parâmetros do MINAS-LC

| **Parâmetros** | **Descrição** |
|---|---|
| $k_{ini}$ | proporção do número de exemplos do conjunto de dados para definir o valor de $k$ do *k-means* na fase *offline* |
| *minSup* | suporte mínimo para um conjunto de rótulos ser considerado frequente para algoritmos de mineração de itens frequentes |
| $\gamma$ | número de microgrupos mais próximos para rotular exemplos e microgrupos |
| $minFreq$ | frequência mínima de rótulos para atribuir a exemplos e microgrupos. Quando esse valor é zero, o MINAS-LC atribui os $\lceil z \rceil$ primeiros rótulos do ranking |
| Atualizar $MC$ | toda vez que microgrupo são usados para classificar um exemplo, suas estatísticas são atualizadas com as informações do exemplo (Incrementalidade) |
| $\omega$ | janela de classificação, tempo máximo que um microgrupo pode permanecer sem classificar exemplos e que um exemplo pode permanecer na memória temporária |
| $\Theta$ | limite máximo de armazenamento da memória temporária |
| $minEx$ | número mínimo de exemplos para um microgrupo ser válido |

  - definimos $k_{ini} = 0,1$ para que cada metaclasse seja representada por mais microgrupos (em relação as outras configurações). Dessa forma, esperamos melhorar a representação das classes com distribuição de exemplos não esféricas;
  - definimos $minFreq = 0,3$ para o MINAS-LC classificar cada exemplo em um menor número de classes, favorecendo bases de dados com baixa cardinalidade de rótulo. É importante destacar que quando nenhuma classe tem frequência relativa superior a $minFreq$, o exemplo é rotulado de acordo com o conjunto de rótulos de seu microgrupo mais próximo;
  - definimos $\omega = \Theta * 10$ para que seja possível ocorrer vários procedimentos de DN antes que os exemplos sejam removidos da memória temporária. Além disso, é importante definir $\omega$ de acordo com o valor de $\Theta$, pois eles são relacionados;
  - definimos $\gamma = \lceil |M| * 0.1 \rceil$, ou seja, 10% do tamanho total do modelo de decisão. O tamanho do modelo de decisão pode variar ao longo do tempo, portanto o número de microgrupos mais próximos ao exemplo para o processo de rotulagem deve ser proporcional ao tamanho do modelo de decisão;
  - nessa configuração os microgrupos não são atualizados depois de classificar os exemplos. A atualização do modelo de decisão é feita ao remover microgrupos obsoletos e adicionar novos microgrupos.

- **Configuração 2 (C2)**: definida para lidar com bases de dados cujas classes são bem separadas e se deslocam no espaço com velocidade, sobrepondo-se umas as outras em diferentes momentos ao longo do fluxo de dados. Essa configuração também favorece bases de dados com velocidade de deslocamento alta:

  - definimos o $\Theta = 2000$, pois quando as classes deslocam-se rapidamente no espaço, o MINAS-LC tende a enviar mais exemplos para a memória temporária. Assim, para evitar a alta frequência de procedimento de DN e para criar microgrupos mais representativos, definimos um valor alto para $\Theta$;
  - definimos $minEx = \frac{\Theta}{k}$, sendo $k$ o número de centroides definidos pelo algoritmo *Leader* no procedimento de DN. Assim, se os exemplos da memória temporária estiverem em grupos mais separados, microgrupos formados por poucos exemplos poderão ser validos (a validação da silhueta garantirá a coesão desses microgrupos). Por outro lado, se os exemplos estiverem mais próximos entre si, o processo de validação de representatividade será mais severo, validando apenas microgrupos formados por muitos exemplos;
  - definimos $\omega = \Theta * 2$. Como o valor de $\Theta$ é alto, não são necessários muitos procedimentos de DN para adaptar o modelo de decisão às mudanças de conceito, portanto consideramos $\omega$ como o dobro de $\Theta$. Além disso, com valores altos atribuídos a $\omega$, a influência de informações obsoletas no modelo de decisão é maior, podendo assim, prejudicar o tratamento das mudanças de conceito;
  - definimos $k_{ini} = 0,01$ para que menos microgrupos por metaclasse sejam formados. Dessa forma, os microgrupos terão raios maiores o que percebemos ser uma vantagem para bases de dados com classes mais disjuntas, especialmente as sintéticas construídas a partir da função RBF.

- **Configuração 3 (C3)**: a ideia principal dessa configuração é idêntica a C1, mas cada vez que microgrupos são usados para rotular exemplos, suas estatísticas são atualizadas. Além disso, definimos o $\Theta = 50$, e agora ao invés de usarmos a frequência mínima $minFreq$ para associar os rótulos aos exemplos, usamos os $\lceil z \rceil$ primeiros rótulos mais frequentes. Lembrando que $z$ é a cardinalidade de rótulos da base de treino.

- **Configuração 4 (C4)**: definimos essa configuração de forma mais automática possível. O valor de $\Theta$ foi definido primeiramente como o valor da média dos tamanhos dos subconjuntos $Q_{\lambda_j}$. Lembrando que o MINAS-LC divide o conjunto de treinamento em subconjuntos, cada um formado por exemplos de uma metaclasse $\lambda_j$. Após o primeiro processo de DN, o valor de $\Theta$ é dado por $minEx * |P|$, sendo $|P|$ o número de novos microgrupos criados no agrupamento na memória temporária.

- **Configuração 5 (C5)**: definimos essa configuração como um meio termo entre as configurações C1 e C2. No entanto, parâmetros como $\gamma, k_{ini}$ e $minEx$ foram definidos empiricamente através de execuções do MINAS-LC com diferentes valores atribuídos a esses parâmetros.

Tabela 10 – Configurações de parâmetros do MINAS-LC

| **Parâmetros** | **C1** | **C2** | **C3** | **C4** | **C5** |
|---|---|---|---|---|---|
| $k_{ini}$ | 0.1 | 0.01 | 0.1 | 0.1 | 0.01 |
| $\gamma$ | $\lceil \lvert M \rvert * 0.1 \rceil$ | $\lceil \lvert M \rvert * 0.1 \rceil$ | $\lceil \lvert M \rvert * 0.1 \rceil$ | $\lceil \lvert M \rvert * 0.1 \rceil$ | 10 |
| $minFreq$: | 0.3 | 0.1 | - | - | 0.2 |
| Atualizar $MC$ | não | não | sim | não | não |
| $\omega$ | $\Theta * 10$ | $\Theta * 2$ | $\Theta * 10$ | $\Theta * 10$ | 2000 |
| $\Theta$ | 30 | 2000 | 50 | media($\lvert Q_{\lambda_j} \rvert$) e $minEx * \lvert P \rvert$ | 200 |
| $minEx$ | 3 | $\Theta/k$ | 3 | $\Theta/k$ | 3 |

É possível executar o MINAS-LC com dois métodos de transformação de problema, LP e PS. Para melhor visualização, chamaremos de MINAS-LP quando o MINAS-LC for executado com o método LP e, da mesma forma o MINAS-PS, para execuções com o método PS. Executamos o MINAS-LC com os dois métodos de transformação de problema, com todas as cinco configurações de parâmetros da Tabela 10, em todas as bases de dados. Os resultados das medidas F1m e F1M das execuções nas bases de dados *4CRE-V2, MOA-3C-2D, Mediamill, Scene* e *Yeast* (bases de dados com comportamentos mais distintos) são apresentados na Tabela 11. Sublinhamos os resultados que superam todos os métodos *lower bounds* e destacamos em negrito os melhores resultados obtidos pelo MINAS-LC em cada base de dados.

Na base de dados *4CRE-V2* o MINAS-LC superou os métodos *lower bounds* com as configurações 1, 2 e 5. Visto que essa base de dados possui classes com deslocamento mais rápido dentre as demais, os melhores resultados foram obtidos com a C2 e C5. A C5 se saiu melhor por conta de dois fatores: o primeiro é o $\omega$, o deslocamento rápido das classes da base *4CRE-V2* faz os microgrupos tornarem-se obsoletos em um curto espaço de tempo. Assim, com um valor menor para $\omega$, evitamos que esses microgrupos interfiram na evolução do modelo de decisão. Além disso, o MINAS-LC limpa a memória temporária com mais frequência, também ajudando a evitar informações obsoletas. O outro fator é o valor de $\gamma$. O tamanho do modelo de decisão nas execuções giram em torno de 60 e 80 microgrupos. Portanto, para C2 eram selecionados de 6 a 8 microgrupos mais próximos para classificação. Já na C5 esse valor é 10 ($\gamma = 10$), independente do tamanho do modelo de decisão. Esses de 2 a 4 microgrupos a mais fizeram essa pequena diferença nos resultados. Notamos isso pelos valores das revocações que subiram de $0,637$ e $0,641$ (micro e macro respectivamente) na C2 para $0,957$ e $0.956$ (micro e macro respectivamente).

Como a base de dados *MOA-3C-2D* é simples, o MINAS-LC superou os métodos *lower bounds* em quase todas as configurações, exceto na C3 e C5 com o método PS. Essa é uma das desvantagem do método PS com *maximal itemsets* em bases de dados com poucas combinações diferentes de rótulos. Nesse caso, o método considerou como *maximal itemsets* apenas o conjunto

Tabela 11 – Resultados do MINAS-LC com diferentes configurações de parâmetros

| Base de Dados | Configuração | F1m | | F1M | |
|---|---|---|---|---|---|
| | | MINAS-LP | MINAS-PS | MINAS-LP | MINAS-PS |
| 4CRE-V2 | 1 | 0,300 | 0,378 | 0,314 | 0,383 |
| | 2 | 0,360 | 0,405 | 0,369 | 0,410 |
| | 3 | 0,319 | 0,317 | 0,326 | 0,325 |
| | 4 | 0,311 | 0,317 | 0,322 | 0,325 |
| | 5 | 0,388 | **0,451** | 0,395 | **0,455** |
| MOA-3C-2D | 1 | 0,713 | 0,703 | 0,696 | 0,689 |
| | 2 | **0,762** | 0,704 | **0,730** | 0,689 |
| | 3 | 0,735 | 0,603 | 0,691 | 0,560 |
| | 4 | 0,752 | 0,605 | 0,712 | 0,605 |
| | 5 | 0,749 | 0,704 | 0,723 | 0,689 |
| Mediamill | 1 | 0,585 | 0,551 | 0,331 | 0,337 |
| | 2 | 0,496 | 0,522 | **0,380** | 0,346 |
| | 3 | 0,575 | 0,527 | 0,200 | 0,212 |
| | 4 | **0,610** | 0,566 | 0,170 | 0,200 |
| | 5 | 0,431 | 0,404 | 0,367 | 0,362 |
| Scene | 1 | 0,551 | 0,423 | 0,508 | 0,447 |
| | 2 | **0,689** | 0,545 | **0,566** | 0,514 |
| | 3 | 0,642 | 0,477 | 0,556 | 0,506 |
| | 4 | 0,362 | 0,278 | 0,157 | 0,198 |
| | 5 | **0,689** | 0,545 | **0,566** | 0,514 |
| Yeast | 1 | 0,422 | 0,564 | 0,401 | 0,527 |
| | 2 | 0,398 | **0,551** | 0,361 | 0,503 |
| | 3 | 0,366 | 0,423 | 0,344 | 0,355 |
| | 4 | 0,132 | 0,137 | 0,224 | 0,289 |
| | 5 | 0,542 | 0,545 | 0,537 | **0,538** |

de rótulos $Y = 0, 1, 2$. Portanto, o método sempre classificou exemplos nessas classes.

Nas bases de dados reais *Mediamill* e *Scene*, o MINAS-LC obteve melhores resultados que os métodos *lower bounds* em quase todas as configurações. No entanto, para a base *Yeast*, o MINAS-LC só superou-os com a C5 e apenas em relação a F1M. Para a *Yeast*, nenhuma das configurações levou o MINAS-LC a executar processos de DN, pois o modelo de decisão foi capaz de classificar todos os exemplos sem rejeitá-los. Isso acontece devido à distribuição estacionaria das bases de dados reais e, também, pela *Yeast* possuir alta dependência de classes e alta cardinalidade.

A partir desses experimentos, selecionamos as melhores configurações para cada base de dados (Tabela 11). Com essas configurações no MINAS-LC, executamos os experimentos para compara-lo com todos os métodos da literatura, e apresentamos os resultados graficamente ao longo do tempo. Os resultados desses experimentos são apresentados na próxima seção.

### 6.4.2 Resultados

Nesta seção apresentaremos os resultados obtidos pelo MINAS-LC e por todos os métodos da literatura apresentados na Tabela 7 (Seção 6.2). As bases de dados usadas nos experimentos foram apresentadas na Tabela 5 (Seção 6.1.1) avaliadas seguindo a metodologia da Seção 6.3.1.

Nas Tabelas 12 e 13, apresentamos as médias das medidas F1M e F1m das bases de dados. Nessas tabelas, também são mostradas a configuração de parâmetros utilizada pelo MINAS-LC em cada uma das bases de dados. Escolhemos a F1 para mostrar os resultados pois é uma estratégia simples e comumente usada em trabalhos da literatura para comparar métodos. Ainda, através dos cálculos das média micro e macro, é possível calcular os resultados dando pesos iguais para os exemplos (média micro) e para os rótulos (macro), obtendo assim, uma comparação mais justa.

Como o objetivo dos experimentos é validar o MINAS-LC em relação os métodos *lower bounds*, nas Tabelas 12 e 13, destacamos em sublinhado os melhores resultados dentre esses métodos e marcamos em negrito os melhores resultados dentre todos os métodos, incluindo os *upper bounds*.

O MINAS-LC obteve resultados superiores aos métodos *lower bounds* em todas as bases de dados, exceto na *5CVT* (quando não os superou na F1M, superou-os na F1m, mesmo com pequenas diferenças entre os resultados). É importante destacar que nas bases de dados reais o MINAS-LC obteve resultados competitivos até com os métodos *upper bounds*, superando-os nas bases *NUS-WIDE*, *Yeast* e *Reuters*. Isso indica que, mesmo em ambientes estacionários e com poucos exemplos disponíveis para treinamento, o MINAS-LC é capaz de construir modelos de decisão eficazes.

Tabela 12 – Médias F1M sobre todas a janelas de avaliação nos experimentos do MINAS-LC. A coluna *Config.* representa a configuração de parâmetros usada pelo MINAS-LC na base de dados. Os melhores resultados entre o MINAS-LC e os métodos *lower bounds* estão sublinhados e os melhores resultados entre todos os métodos estão em negrito. representa a configuração de parâmetros usada pelo MINAS-LC na base de dados.

| **Bases de Dados** | Config. | MINAS-LP | MINAS-PS | CC | PS | MLHT | iSOUPtree | $E_aMLHT$ | $E_aCC$ | $E_aBR$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Mediamill | 4 | 0.170 | 0,205 | 0,208 | 0,126 | 0,126 | 0,355 | 0,248 | 0,350 | **0,370** |
| NUS-WIDE | 2 | **0,443** | 0,391 | 0,354 | 0,301 | 0,301 | 0,323 | 0,344 | 0,420 | 0,434 |
| Scene | 5 | 0,566 | 0,514 | 0,137 | 0,025 | 0,025 | 0,174 | 0,331 | 0,643 | **0,652** |
| Yeast | 5 | 0,537 | **0,538** | 0,358 | 0,244 | 0,244 | 0,531 | 0,320 | 0,430 | 0,399 |
| Reuters | 1 | 0,400 | **0,407** | 0,124 | 0,004 | 0,004 | 0,067 | 0,078 | 0,099 | 0,098 |
| 5CVT | 3 | 0,586 | 0,563 | 0,651 | 0,645 | 0,655 | 0,499 | 0,755 | 0,869 | **0,888** |
| 4CRE-V1 | 5 | 0,357 | 0,352 | 0,271 | 0,233 | 0,233 | 0,328 | 0,725 | 0,966 | **0,968** |
| 4CRE-V2 | 5 | 0,399 | 0,455 | 0,351 | 0,295 | 0,325 | 0,440 | 0,836 | 0,942 | **0,955** |
| MOA-3C-2D | 2 | 0,729 | 0,689 | 0,542 | 0,548 | 0,520 | 0,649 | 0,928 | 0,964 | **0,973** |
| MOA-5C-2D | 5 | 0,567 | 0,569 | 0,554 | 0,350 | 0,363 | 0,562 | 0,861 | 0,950 | **0,956** |

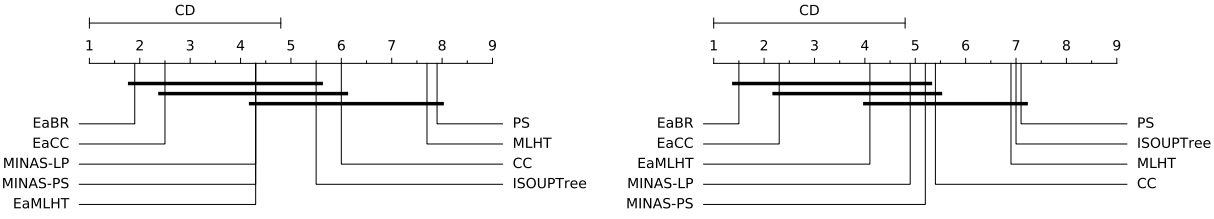
Tabela 13 – Médias F1m sobre todas a janelas de avaliação nos experimentos do MINAS-LC. A coluna *Config.* representa a configuração de parâmetros usada pelo MINAS-LC na base de dados. Os melhores resultados entre o MINAS-LC e os métodos *lower bounds* estão sublinhados e os melhores resultados entre todos os métodos estão em negrito.

| **Bases de Dados** | Config. | MINAS-LP | MINAS-PS | CC | PS | MLHT | iSOUPtree | $E_aMLHT$ | $E_aCC$ | $E_aBR$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Mediamill | 4 | 0,610 | 0,589 | 0,538 | 0,552 | 0,552 | 0,368 | 0,601 | 0,647 | **0,656** |
| NUS-WIDE | 2 | 0,503 | 0,426 | 0,530 | 0,463 | 0,463 | 0,338 | 0,424 | 0,550 | **0,559** |
| Scene | 5 | 0,689 | 0,545 | 0,306 | 0,059 | 0,059 | 0,242 | 0,213 | 0,775 | **0,789** |
| Yeast | 5 | 0,542 | 0,545 | 0,669 | 0,565 | 0,565 | 0,539 | 0,584 | 0,670 | **0,678** |
| Reuters | 1 | 0,380 | **0,399** | 0,110 | 0,069 | 0,069 | 0,071 | 0,136 | 0,105 | 0,105 |
| 5CVT | 3 | 0,480 | 0,528 | 0,617 | 0,620 | 0,639 | 0,484 | 0,755 | 0,857 | **0,877** |
| 4CRE-V1 | 5 | 0,345 | 0,340 | 0,236 | 0,193 | 0,193 | 0,321 | **0,696** | 0,964 | 0,967 |
| 4CRE-V2 | 5 | 0,392 | 0,451 | 0,374 | 0,292 | 0,332 | 0,461 | 0,832 | 0,940 | **0,954** |
| MOA-3C-2D | 2 | 0,760 | 0,704 | 0,613 | 0,605 | 0,581 | 0,654 | 0,928 | 0,928 | **0,972** |
| MOA-5C-2D | 5 | 0,603 | 0,578 | 0,618 | 0,361 | 0,388 | 0,561 | 0,867 | 0,950 | **0,956** |

Dados os resultados apresentados nas Tabelas 12 e 13, para analisar se há diferenças estatísticas entre os algoritmos, realizamos o teste de *Friedman* assumindo a equivalência entre os algoritmos como hipótese nula. Se essa hipótese nula é rejeitada (com confiança de 95%) o pós-teste de *Nemenyi* é executado para detectar quais métodos possuem diferenças significativas entre si (DEMŠAR, 2006). O desempenho de dois métodos é significativamente diferente se a média de seus correspondentes rankings for maior do que o valor da diferença crítica (CD) definido no pós-teste de *Nemenyi*.

Na Figura 13, mostramos o gráfico de diferenças críticas das medidas F1M e F1m. Nas figuras, os métodos são dispostos em rankings de acordo com seu desempenho nas bases de dados de maneira que os métodos melhores ranqueados estão a esquerda. A diferença crítica necessária para diferenciar dois métodos é mostrada no topo dos gráficos.É possível notar que, mesmo o MINAS-LC não obtendo resultados significativamente diferentes dos métodos *lower bounds*, ele sempre está mais bem ranqueado do que eles. Além disso, os resultados do MINAS-LC também não apresentam diferenças significativas dos resultados dos métodos *upper bounds* e, em relação a F1M, o MINAS-LC está na mesma posição do ranking que o $EaMLHT$.

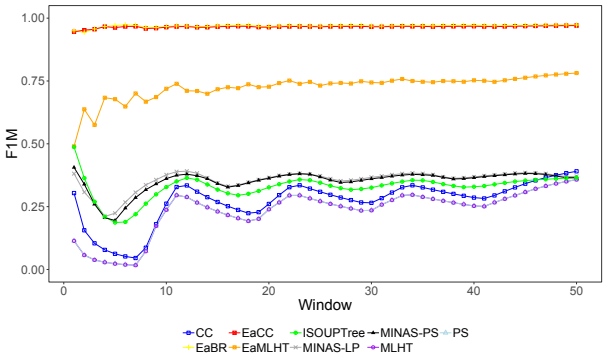
Figura 13 – Diagramas de diferença crítica obtidos a partir do teste de *Friedman* com 95% de significância e pós-teste de *Nemenyi*.



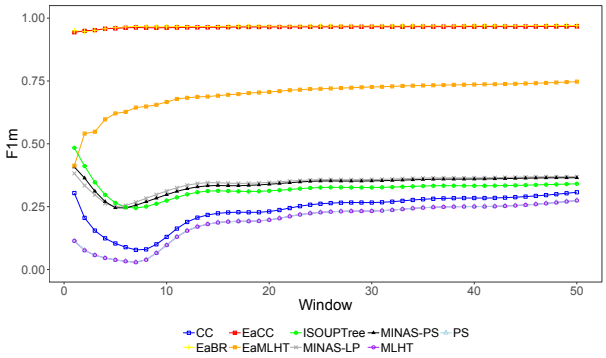

(a) Diagrama de diferença crítica da F1M

(b) Diagrama de diferença crítica da F1m

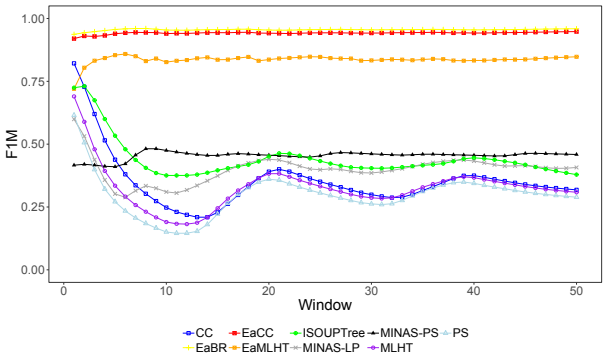
Figura 14 – Resultados F1m e F1M do MINAS-LC nas bases de dados sintéticas



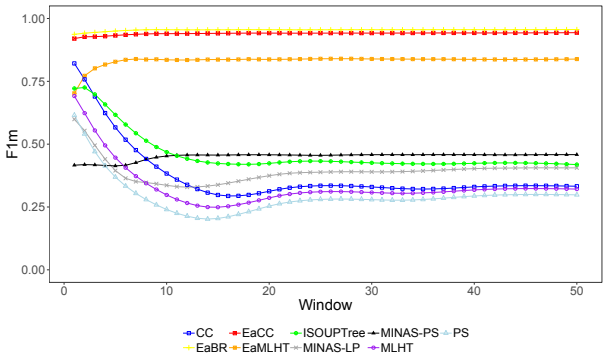

(a) F1M da *4CRE-V1* com configuração 5 de parâmetros



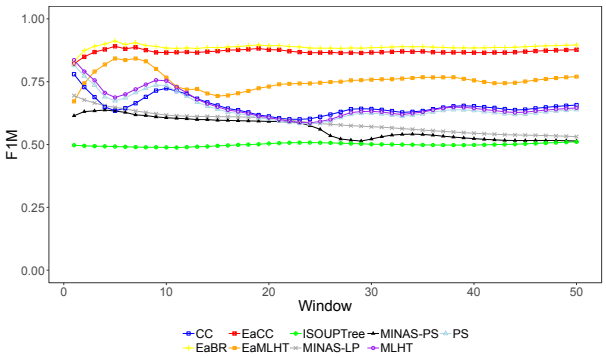

(b) F1m da *4CRE-V1* com configuração 5 de parâmetros



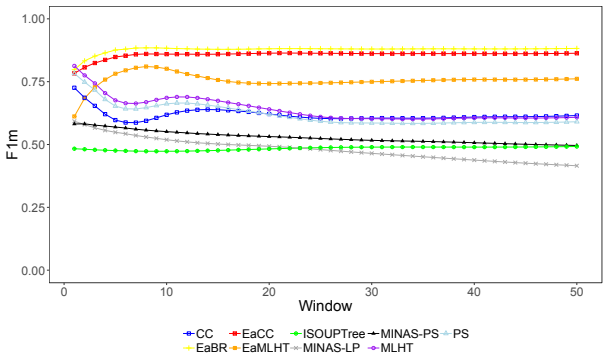

(c) F1M da *4CRE-V2* com configuração 5 de parâmetros




(d) F1m da *4CRE-V2* com configuração 5 de parâmetros




(e) F1M da *5CVT* com configuração 3 de parâmetros

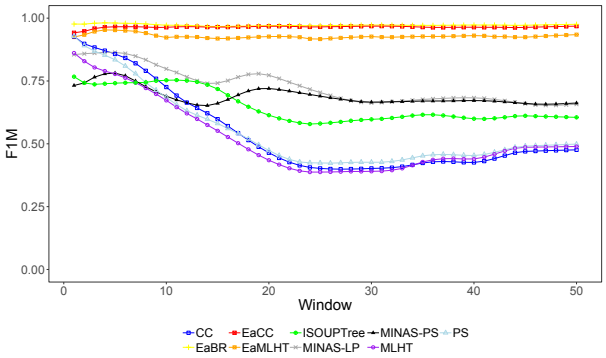



(f) F1m da *5CVT* com configuração 3 de parâmetros

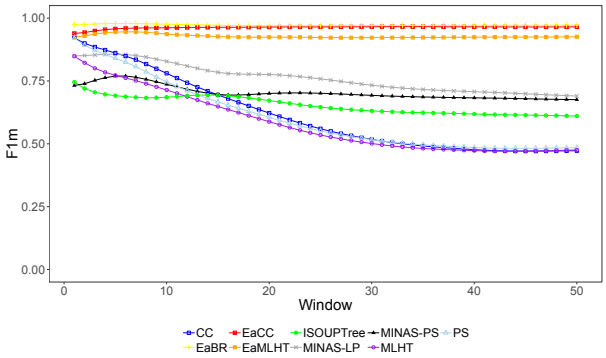
Na mineração de FCD mostrar a média dos resultados dos métodos em bases de dados cuja distribuição é não estacionárias, não é suficiente. É importante mostrar os resultados do método ao longo do tempo, possibilitando assim, observarmos seus comportamentos perante mudanças de conceito. Portanto, apresentamos nas Figuras 14 e 15 os resultados de F1M e F1m dos métodos ao longo do tempo.

Na base de dados *4CRE-V1* (Figura 14a e 14b) ambas as variações do MINAS-LC (com métodos PS e LP), mesmo encontrando dificuldades em superar o $iSOUPTree$, estão quase sempre acima dos métodos *lower bounds*. Nessa base de dados, nota-se a grande diferença entre

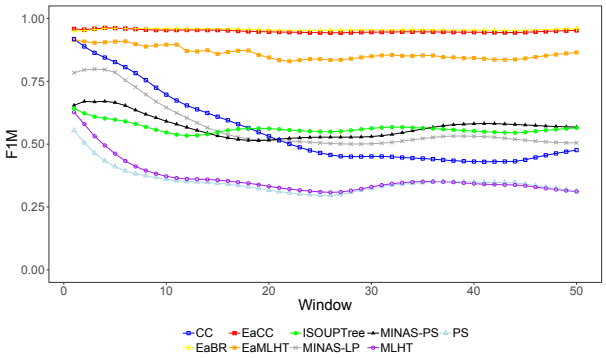
Figura 15 – Resultados F1m e F1M do MINAS-LC nas bases de dados sintéticas



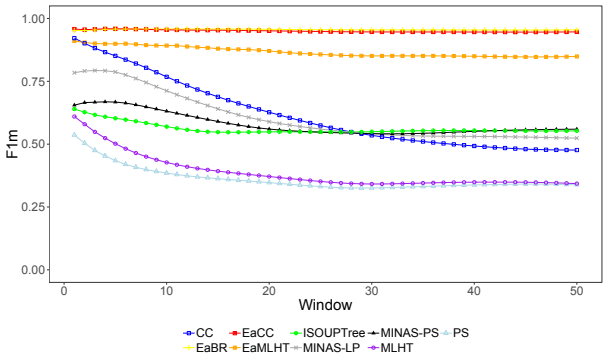

(a) F1M da *MOA-3C-2D* com configuração 2 de parâmetros




(b) F1m da *MOA-3C-2D* com configuração 2 de parâmetros




(c) F1M da *MOA-5C-2D* com configuração 5 de parâmetros




(d) F1m da *MOA-5C-2D* com configuração 5 de parâmetros

os resultados dos métodos *upper bounds*, cujos modelos de decisão são atualizados com os rótulos reais de exemplos, e os resultados do MINAS-LC e dos métodos *lower bounds*, cujos modelos de decisão não atualizam ou são atualizados de forma não supervisionada. O $EaCC$ e $EaBR$ quase sempre mantêm seus resultados em 100% de F1M e F1m, mesmo passando por várias mudanças de conceito ao longo do tempo.

Um característica importante do MINAS-LC é manter sua estabilidade após às mudanças de conceito. Essa característica é observada em todos as bases de dados não estacionária, destacando-se na *4CRE-V2* (Figura 14c e 14d) e *MOA-3C-2D* (Figura 15a e 15b). Nessas bases de dados, o desempenho preditivo dos métodos *lower bounds* sofrem queda contínua e significativa (entre 40% a 80% aproximadamente) até a janela 10 (no caso da *4CRE-V2*) e janela 20 (no caso da *MOA-3C-2D*), enquanto o MINAS-LC se mantém sempre estável ou, após uma queda de desempenho, se estabiliza e até evolui (como observado em sua variação MINAS-PS na Figura 14c).

A base de dados *MOA-5C-2D* é composta por classes com deslocamento rápido no espaço de forma a trocarem de posição umas com as outras com frequência (isso pode ser observado na Figura 10). Esse tipo de comportamento é desafiador para métodos com modelos de decisão baseados em agrupamento. No caso do MINAS-LC esse tipo de mudança de conceito,

impossibilita a atualização de seu modelo de decisão a tempo de adaptar-se de forma precisa. Como as classes trocam de lugar com frequência, os microgrupos não são removidos, ou são atualizados erroneamente, pois continuam classificando exemplos da nova classe que assumiu o lugar da classe usada na construção do microgrupo. Portanto, nos experimentos executados nessa bases de dados, o MINAS-LC superou com dificuldades (em média F1M) os métodos *lower bounds*. No entanto, ao observar o desempenho ao longo do tempo (Figuras 15c e 15d), percebemos a superioridade dos resultados do método $CC$ em relação ao MINAS-LC até a metade do fluxo de dados. Ainda, o método iSOUPTree manteve-se com resultados semelhantes aos do MINAS-LC por todo o fluxo de dados.

Por fim, na base de dados *5CVT*, acontece um problema semelhante ao da base de dados *MOA-5C-2D*. As classes não trocam de lugar, mas se movem em conjunto de um lado para outro rapidamente (ver Figura 9). Isso faz com que o MINAS-LC, até mesmo com configurações de parâmetros que possibilitam a eliminação e criação de microgrupos com frequência (C1 e C3, por assumirem valores de $\Theta$ e $\omega$ baixos), não seja apto a adaptar-se a tais mudanças de conceito de modo a superar os métodos *lower bounds*.

## 6.5 Análise dos Resultados do MINAS-BR

Esta seção apresenta os experimentos executados e as discussões dos resultados obtidos pelo MINAS-BR. Os experimentos têm dois principais objetivos: *i)* avaliar o procedimento de DN do MINAS-BR, medindo a qualidade e o tempo de detecção de evoluções de conceito (medido a partir da janela em que surgiu o primeiro exemplo pertencente a uma determinada classe novidade, até a detecção do PN associado à ela); *ii)* avaliar a performance preditiva do MINAS-BR em comparação com os métodos *lower bounds* em cenários com latência extrema de rótulos. Além desses objetivos, nós avaliamos o MINAS-BR em comparação com os métodos *upper bounds*, que apesar de ser uma comparação injusta, visto que esses métodos têm acesso aos rótulos reais dos exemplos, esperamos ter resultados competitivos.

### 6.5.1 Sensibilidade dos parâmetros

Antes de apresentar os resultados do MINAS-BR em comparação com os métodos da literatura, nós analisamos como diferentes configurações de parâmetros influenciam o desempenho preditivo do MINAS-BR.

Na Tabela 14 nós apresentamos s parâmetros do MINAS-BR e suas respectivas configurações usadas nos experimentos. Definimos esses valores através de experimentos empíricos, executando o MINAS-BR com diferentes combinações de valores nos baseando em trabalhos de DN propostos na literatura (FARIA et al., 2016b; SPINOSA et al., 2009; AL-KHATEEB et al., 2012a; MASUD et al., 2011a).

Tabela 14 – Configurações de parâmetros e abordagens

| **Parâmetros e Abordagens** | **Valores** |
|---|---|
| $k_{ini}$: valor para definição do valor de $k$ do *k-means* na fase *offline* | proporção 0,01, 0,1 e 0,3 em relação ao número de exemplos a serem agrupados |
| $minEx$: número mínimo de exemplos para formar um microgrupo válido | 3 |
| número de rótulos a serem atribuídos a um exemplo na classificação | rótulos dos $\lceil z \rceil$ microgrupos mais próximos entre todos os modelos de decisão |
| tamanho da janela ($\omega$) para eliminar exemplos e microgrupos obsoletos | 100, 500, 1000, 2000 |
| limite de armazenamento da memória temporária ($\Theta$) | 20, 50, 100, 1000, 2000 |
| fator $\mathcal{F}$ para calcular o limiar de PNs $\mathcal{T}$ | 0,5, 0,7, 1,1,1,3,1,3 |

Durante os experimentos, notamos que determinadas atribuições de valores não melhoram a performance preditiva do MINAS-BR. Como por exemplo, se o valor $\Theta$ é maior do que os presentes na Tabela 14, o MINAS-BR nunca cria novos microgrupos e, os exemplos *desconhecidos* enviados para memória temporária, são sempre eliminados. A seguir, apresentamos a análise de sensibilidade de parâmetros do MINAS-BR e mostramos os resultados obtidos com suas diferentes configurações.

A partir dos experimento, nós percebemos que os valores de $\omega$ e $\Theta$ devem ser definidos de acordo com cada base de dado. Por exemplo, se uma determinada base de dados possui mudanças de conceitos abruptas, os valores de $\omega$ e $\Theta$ devem ser menores para que o procedimento de DN possa detectar essas mudanças com maior precisão.

Considerando o valor do parâmetro $\mathcal{F}$, atribuir a ele um valor único não é adequado para distinguir entre PNs e extensões, pois geralmente, valores menores produzem um número maior de PNs e poucas extensões. Por outro lado, valores maiores produzem alto número de extensões e poucos PNs (FARIA et al., 2016b). Do mesmo modo, o valor de $k_{ini}$ também pode influenciar no processo de DN. Se as classes de uma base de dados forem muito próximas entre si no espaço, um valor alto de $k_{ini}$ é mais apropriado, pois o método criará microgrupos com raios menores e poucos exemplos. Como resultado, é possível detectar PNs mesmo eles estando próximos dos microgrupos das classes conhecidas, porque o desvio padrão desses microgrupos tende a ser menor (é importante notar que o valor de $\mathcal{F}$ também influencia nessa parte). Por outro lado, se a base de dados possuir classes mais disjuntas (*e.g.,* base de dados sintéticas), valores menores de $k_{ini}$ são melhores opções.

Para mostrar a influência dos parâmetros, nós escolhemos uma base de dados sintética e outra real e apresentamos os resultados das execuções do MINAS-BR com todas as possíveis configurações dos parâmetros $\omega, \Theta, \mathcal{F}$ e $k_{ini}$. Na Tabela 15 apresentamos os resultados da

Tabela 15 – Médias F1M para as bases de dados *Mediamill-EC* e *MOA-EC* variando os parâmetros $\omega$, $\Theta$ e $k_{ini}$. Atribuímos o valor 1.1 ao parâmetro $\mathcal{F}$ visto que diferentes valores nesse parâmetro não alteram os resultados.

| $\Theta$ | $\omega$ | Mediamill-EC | | | MOA-EC | | |
|---|---|---|---|---|---|---|---|
| 20 | 100 | 0,252 | 0,208 | 0,185 | 0,319 | 0,211 | 0,195 |
| | 500 | 0,265 | 0,269 | 0,237 | 0,802 | 0,393 | 0,277 |
| | 1000 | 0,269 | **0,273** | 0,245 | 0,829 | 0,615 | 0,375 |
| | 2000 | 0,254 | 0,272 | 0,254 | 0,830 | 0,765 | 0,562 |
| 50 | 100 | 0,244 | 0,209 | 0,205 | 0,350 | 0,208 | 0,148 |
| | 500 | 0,250 | 0,265 | 0,234 | 0,813 | 0,397 | 0,275 |
| | 1000 | 0,253 | 0,259 | 0,250 | 0,828 | 0,615 | 0,370 |
| | 2000 | 0,254 | 0,262 | 0,247 | 0,830 | 0,770 | 0,564 |
| 100 | 100 | 0,236 | 0,212 | 0,206 | 0,384 | 0,200 | 0,055 |
| | 500 | 0,250 | 0,249 | 0,250 | 0,823 | 0,393 | 0,274 |
| | 1000 | 0,253 | 0,259 | 0,243 | 0,831 | 0,629 | 0,371 |
| | 2000 | 0,254 | 0,262 | 0,246 | 0,833 | 0,765 | 0,564 |
| 1000 | 1000 | 0,253 | 0,259 | 0,240 | 0,815 | 0,653 | 0,408 |
| | 2000 | 0,254 | 0,262 | 0,242 | **0,840** | 0,792 | 0,578 |
| 2000 | 2000 | 0,254 | 0,262 | 0,242 | 0,814 | 0,826 | 0,636 |
| $k_{ini}$ | | 0,01 | 0,1 | 0,3 | 0,01 | 0,1 | 0,3 |

média F1M em todas as janelas de avaliação das bases de dados *MOA-EC* e *Mediamill-EC*. Nós destacamos (sublinhado) todos os casos nos quais o MINAS-BR superou a performance preditiva dos métodos *lower bounds*. As melhores performances preditivas do MINAS-BR, dentre todas as execuções, para cada base de dados, estão destacadas em negrito.

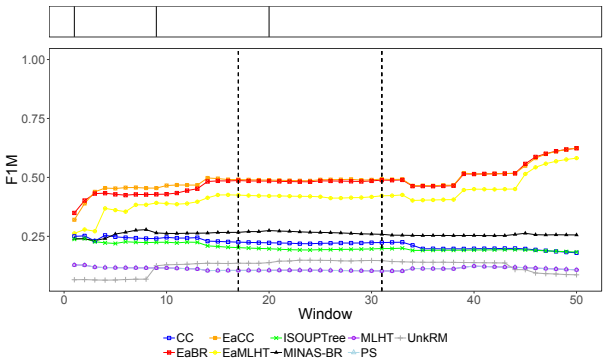
Apesar da variação dos resultados com diferentes configurações de parâmetros, o MINAS-BR obteve melhor performance do que os métodos *lower bounds* na maioria das vezes, independente dos valores parâmetros. Chegamos à mesma conclusão ao realizar esta análise nas outras bases de dados.

### 6.5.2 Resultados

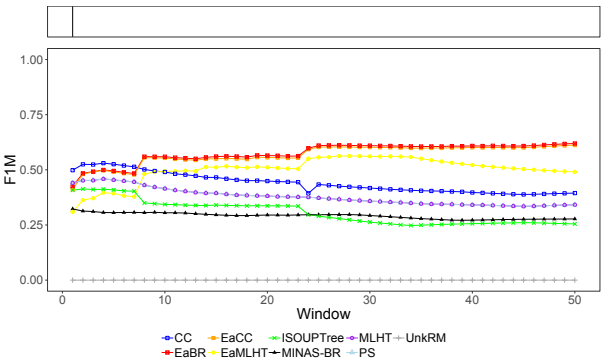
Comparamos o MINAS-BR com os métodos apresentados na Tabela 7 (Seção 6.2). Para cada base de dados, nós mostramos os melhores resultados do MINAS-BR obtidos após usar todas as diferentes configurações de parâmetros apresentados na Tabela 14. Apresentamos na Tabela 16 as médias da F1M em todas as janelas de avaliação. Nós destacamos com sublinhado os melhores resultados dentre os do MINAS-BR e dos métodos *lower bounds* e, com negrito, destacamos os melhores resultados dentre os de todos os métodos.

Na Figuras 16 (bases de dados reais) e na Figura 17 (bases de dados sintéticas), nós mostramos os valores da F1M e UnkRM em todas as janelas de avaliação. O tamanho das janelas

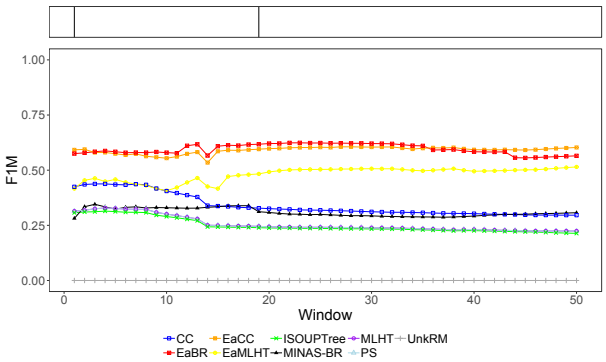
Figura 16 – Valores da F1M das bases de dados reais



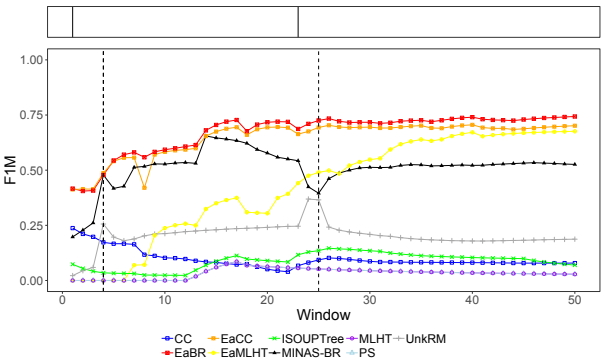

(a) Mediamill-EC - $k_{ini} = 0.1$, $\mathcal{F} = 1.1$, $\Theta = 50$, $\omega = 1000$



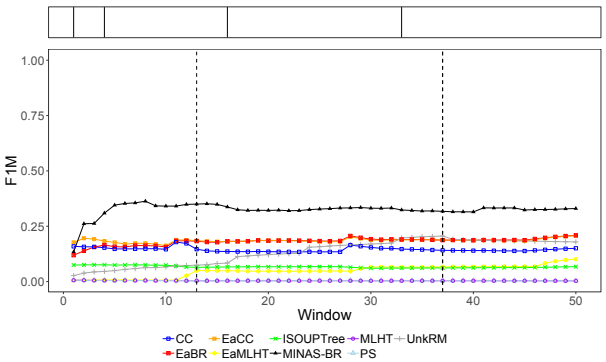

(b) NUS-WIDE-EC - $k_{ini} = 0.1$, $\mathcal{F} = 1.1$, $\Theta = 50$, $\omega = 1000$




(c) Yeast-EC - $k_{ini} = 0.3$, $\mathcal{F} = 1.1$, $\Theta = 20$, $\omega = 2000$




(d) Scene-EC - $k_{ini} = 0.3$, $\mathcal{F} = 1.1$, $\Theta = 50$, $\omega = 1000$

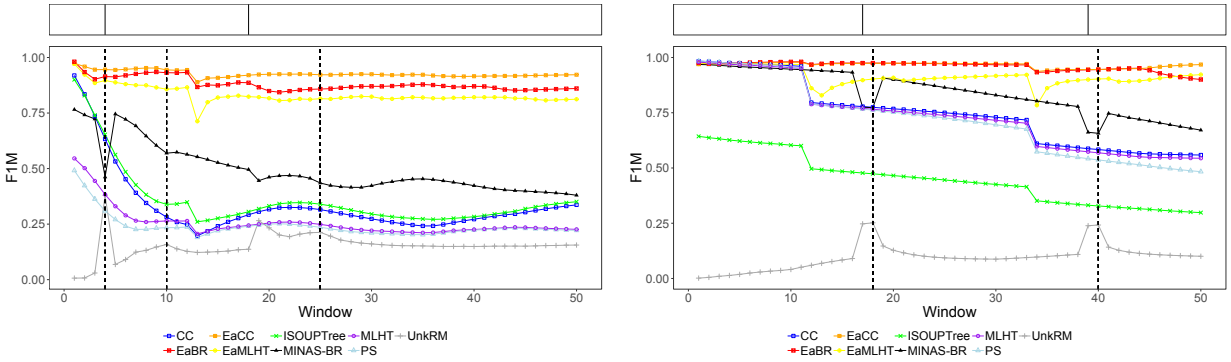



(e) Reuters-EC - $k_{ini} = 0.3$, $\mathcal{F} = 1.1$, $\Theta = 20$, $\omega = 2000$

Tabela 16 – Médias F1M sobre todas a janelas de avaliação nos experimentos do MINAS-BR.

| **Bases de Dados** | MINAS-BR | CC | PS | MLHT | iSOUPtree | $E_aMLHT$ | $E_aCC$ | $E_aBR$ |
|---|---|---|---|---|---|---|---|---|
| Mediamill-EC | 0.26 | 0.22 | 0.12 | 0.12 | 0.20 | 0.42 | **0.50** | 0.49 |
| NUS-WIDE-EC | 0.29 | 0.44 | 0.38 | 0.38 | 0.31 | 0.50 | 0.57 | **0.58** |
| Scene-EC | 0.51 | 0.10 | 0.03 | 0.03 | 0.09 | 0.43 | 0.65 | **0.67** |
| Yeast-EC | 0.31 | 0.34 | 0.25 | 0.25 | 0.25 | 0.48 | 0.59 | **0.59** |
| Reuters-EC | **0.32** | 0.14 | 0.01 | 0.01 | 0.07 | 0.05 | 0.18 | 0.18 |
| MOA-EC | 0.84 | 0.74 | 0.72 | 0.73 | 0.45 | 0.91 | **0.96** | 0.96 |
| 4CRE-V2-EC | 0.54 | 0.33 | 0.24 | 0.26 | 0.36 | 0.83 | **0.93** | 0.88 |

Figura 17 – Valores de F1M das bases de dados sintéticas.




(a) 4CRE-V2-EC - $k_{ini} = 0.01$, $\mathcal{F} = 1.1$, $\Theta = 100$, $\omega = 100$

(b) MOA-EC - $k_{ini} = 0.01$, $\mathcal{F} = 1.1$, $\Theta = 1000$, $\omega = 2000$

foi definido como $w = \frac{|DS|}{50}$ (*i.e.,* todo fluxo de dados, exceto os primeiros 10%, que foram usados no treinamento, é divido em cinquenta janelas). O eixo y dos gráficos representam os valores da F1M e UnkRM (para o MINAS-BR), enquanto o eixo x representa todas as cinquenta janelas. As linhas verticais nós retângulos acima dos gráficos, representam as janelas em que uma nova classe surgiu e, as linhas verticais tracejadas cruzando o eixo x do gráfico representam as janelas que o MINAS-BR criou novos modelos de decisão para representar os PNs. Os valores da UnkRM do MINAS-BR são representados pelas linhas cinza.

Os picos da UnkRM indicam o surgimento de novas classes. É esperado que o MINAS-BR use os correspondes exemplos *desconhecidos* para formar modelos de decisão que representem PNs (linhas verticais tracejadas). Se isso ocorrer, o valor da UnkRM deve diminuir, indicando que o método se adaptou bem às evoluções de conceito. Da Figura 16d até a 17b, nota-se que o MINAS-BR comportou-se como esperado, destacando-se na base de dados *MOA-EC*. No entanto, para as outras bases de dados, o MINAS-BR atrasou-se ao detectar novos PNs ou não os detectou.

Em relação a F1M, o MINAS-BR obteve melhores resultados que os métodos *lower bounds* para todos as bases de dados, exceto as *Yeast-EC* e *NUS-WIDE-EC*. Nos experimentos nessas duas bases de dados, o MINAS-BR não detectou classes novidade, ocasionando assim, uma performance baixa em relação aos outros métodos, cujos seus resultados foram competitivos

apenas com os do método $CC$ na *Yeast-EC*. O MINAS-BR encontrou dificuldades na detecção de classes novidade, pois as classes dessas duas bases de dados possuem alto grau de dependência e, no caso da *Yeast-EC*, além da dependência, a cardinalidade de rótulos é alta. Isso torna a tarefa de DN com modelos de decisão baseados em agrupamento ainda mais desafiadora, pois como as classes são sempre sobrepostas o método é incapaz de detectar o surgimento de uma classe novidade dentre tanta sobreposição, mesmo as classes novidades sendo escolhidas com menor grau de dependência (durante o pré-processamento). Constatamos isso observando que o MINAS-BR, nos experimentos envolvendo essas duas bases de dados e, independe das configurações de parâmetros, nunca envia exemplos para a memória temporária. Esse fato pode ser percebido analisando a Figura 16c e 16b, cujos valores da UnkRM são sempre zero.

Na base de dados *Mediamill-EC* (Figura 16a), o MINAS-BR não detectou a primeira classe novidade e as duas classes novidades posteriores foram detectas com atraso. Apesar da alta cardinalidade de rótulos e grau de dependência entre as classes do *Mediamill-EC*, foi possível detectar as classes novidades ao atribuir um valor baixo no parâmetro $\Theta$. Dessa maneira, o procedimento de DN foi executado em curtos espaços de tempo, fazendo com que os poucos exemplos enviados para memória temporária fossem suficientes para criar extensões e PNs.

Em relação a base de dados *Reuters-EC*, o MINAS-BR foi melhor que todos os métodos, inclusive os *upper bounds*. Isso aconteceu porque o conjunto de treinamento dessa base de dados foi construído com poucos exemplos positivos para cada classe (máximo de vinte, para evitar o desbalanceamento). Isso, somado com o fato da *Reuters-EC* ter um alto número de classes ($|L_{tr}| = 35$) e de 10% da base de dados representar poucos exemplos (apenas 569 exemplos), torna-se difícil para qualquer método construir de modelos de decisão eficazes na classificação. Sobre esse ponto, é importante destacar a capacidade do MINAS-BR em criar modelos de decisão eficazes (em relação aos resultados dos outros métodos) com poucos exemplos. Uma característica importante em mineração de FCD. No entanto, o MINAS-BR detectou três das cinco classes novidade, mas percebemos que os PNs não são tão representativos. Os PNs foram sempre formados pelo número de mínimo de exemplos ($minEx = 3$). Isso explica o motivo da UnkRM e F1M permanecerem estáveis após a detecção das classes novidade. Em relação ao valor de $minEx$, se atribuíssemos um valor maior a esse parâmetro, nenhum PN seria criado. Portanto, escolhemos deixar o valor igual a três.

Na *4CRE-V2-EC*, os grupos que representam as classes deslocam-se rápido e com movimentos de união e separação uns dos outros (ver Figura 12). Apesar do cenário desafiador dessa base de dados, o MINAS-BR detectou precisamente a primeira classe novidade. Por outro lado, a alta velocidade do deslocamento dos grupos aumentou rápido o número de exemplos *desconhecidos*, fazendo com que o MINAS-BR perdesse desempenho na F1M e criando PNs desnecessários na janela 10. Em relação a segunda classe novidade, que aparece sobreposta a outras classes (ilustrado na Figura 12d), o MINAS-BR detectou-a com atraso, pois só foi possível criar PNs quando seu grupo correspondente se afastou dos demais.

## 6.6 Considerações Finais

Neste capítulo foram apresentados os experimentos realizados com os métodos ***MultI****-label learNing* ***A****lgorithm for data* ***S****treams with* ***L****abel* ***C****ombination-based methods* (MINAS-LC) e ***MultI****-label learNing* ***A****lgorithm for data* ***S****treams with* ***B****inary* ***R****elevance transformation* (MINAS-BR). Como o MINAS-LC trata apenas mudanças de conceito, foi discutido o seu desempenho em diferentes tipos de comportamentos de bases de dados com distribuição de dados não estacionária, onde todas as classes são conhecidas na fase de treinamento. Experimentos com bases de dados reais, onde a distribuição dos dados é estacionária também foram executados. O MINAS-LC obteve resultados competitivos com os métodos da literatura, superando em alguns casos, até os métodos *upper bounds*.

Em relação ao MINAS-BR, como o método lida com evoluções de conceito, os experimentos foram avaliados utilizando a metodologia proposta neste trabalho. Experimentos com bases de dados sintéticas e reais foram executados, mostrando o potencial do método na detecção de novas classes mantendo sua performance preditiva superior as dos métodos *lower bounds*. O próximo capítulo apresenta as principais contribuições deste trabalho, discute as limitações das técnicas propostas e os principais trabalhos futuros a serem desenvolvidos.

# Capítulo 7
## CONCLUSÕES

Neste capítulo, são apresentadas as conclusões deste trabalho. Na Seção 7.1, são apresentadas as principais contribuições; na Seção 7.2, são discutidas algumas limitações das propostas realizadas neste trabalho; e, por fim, na Seção 7.3 são apresentados direcionamentos para trabalhos futuros.

## 7.1 Principais Contribuições

Apesar dos diversos trabalhos de **C**lassificação **M**ultirrótulo em **F**luxos **C**ontínuos de **D**ados (CMFCD) propostos na literatura, características importantes ainda são pouco exploradas, sendo elas: *Latência Extrema de Rótulos*, em que os rótulos reais dos exemplos nunca estarão disponíveis para atualização dos modelos de decisão; e *Evoluções de Conceito*, em que novas classes, não conhecidas na fase inicial de treinamento, surgem ao longo do fluxo de dados.

Técnicas de **D**etecção de **N**ovidade (DN) são utilizadas para lidar com esses problemas, pois com ela é possível encontrar padrões em exemplos não rotulados, que distinguem significativamente dos padrões conhecidos. Através desses novos padrões, é possível atualizar de forma não supervisionada os modelos de decisão dos métodos. No entanto, apesar dos resultados significativos para classificação multiclasse, a DN não foi bem explorada para CMFCD.

Motivados por problemas envolvendo DN e CMFCD em cenários de latência extrema de rótulos com mudanças e evoluções de conceito, a principal contribuição deste trabalho foi o desenvolvimento de dois métodos denominados ***Mult**I**-label** **learN**ing **A**lgorithm for data **S**treams with **L**abel **C**ombination-based methods* (MINAS-LC) e ***Mult**I**-label** **learN**ing **A**lgorithm for data **S**treams with **B**inary **R**elevance transformation* (MINAS-BR).

O MINAS-LC apresenta as seguintes contribuições:

- o uso de dois métodos de transformação de problemas, ***L**abel **P**owerset* (LP) e ***P**runed **S**ets* (PS), para transformar o conjunto de dados em multiclasse na fase de treinamento do método;

- O uso do método PS em conjunto com algoritmos de mineração de conjunto de itens frequentes, em que são considerados frequentes apenas os conjuntos de rótulos que fazem parte dos *maximal itemsets* do conjunto de dados de treinamento;
- o uso de um modelo de decisão formado a partir de microgrupos associados a múltiplas classes, podendo assim, rotular em exemplos em vários rótulos simultaneamente;
- o uso de técnicas de DN para tratar mudanças de conceito em problemas de CMFCD;
- a adaptação de bases de dados sintéticas multiclasse para problemas multirrótulo com mudanças de conceito;

O MINAS-BR apresenta as seguintes contribuições:

- o uso do método ***B**inary **R**elevance* (BR) para dividir o conjunto de dados multirrótulo e criar múltiplos modelos de decisão, um para cada classe do problema;
- o uso de um modelo de decisão para cada classe, formados por microgrupos rotulados, em que os exemplos são testados em cada um desses modelos de decisão, podendo assim ser classificados em múltiplas classes simultaneamente;
- o uso de DN para gerar extensões dos modelos de decisão, quando ocorrer mudanças de conceitos, ou para gerar novos modelos de decisão, quando ocorrer evoluções de conceitos;
- uma nova metodologia para avaliar a tarefa de DN em CMFCD que, através de correlação é possível associar os padrões-novidade às classes do problema;
- pré-processamento das bases de dados reais para selecionar melhores classes novidade, ou seja, as com menor grau de correlação com outras classes, para validação dos métodos;
- a adaptação de bases de dados sintéticas multiclasse para problemas multirrótulo com evolução de conceito.

O MINAS-LC obteve resultados competitivos com os métodos da literatura, superando em alguns casos, até os métodos *upper bounds*. Seu desempenho foi avaliado em diferentes tipos de mudanças de conceito, em que todas as classes são conhecidas na fase de treinamento. Experimentos com bases de dados reais em que a distribuição dos dados é estacionária também foram analisados, comprovando a eficiência do MINAS-LC na construção de modelos de decisão com poucos exemplos rotulados disponíveis.

Em relação ao MINAS-BR, como o método lida com evoluções de conceito, os experimentos foram analisados utilizando a metodologia de avaliação proposta neste trabalho. Experimentos com bases de dados sintéticas e reais foram executados, mostrando o potencial do método na detecção de novas classes mantendo sua performance preditiva superior as dos métodos *lower bounds*.

## 7.2 Limitações

As principais limitações dos métodos propostos nesse trabalho são:

- **Parametrização:** como a maioria dos trabalhos propostos de mineração em FCD, os métodos desenvolvidos neste trabalho possuem muitos parâmetros a serem configurados, tais como: número de microgrupos usado para representar cada classe, número de modelos de decisão para classificar ou rejeitar um exemplo, número de rótulos atribuídos a cada exemplo, quando realizar o processo de DN, critérios relativos à validação de um novo grupo, limiar para separar um novo PN de uma *extensão*, número de modelos de decisão para considerar o novo microgrupo extensão ou PN, número de microgrupos para classificar um exemplo e tamanho da janela de dados atual. Em geral, os valores dos parâmetros podem variar de uma base para outra. Encontrar os melhores valores de parâmetros para uma base de dados não é uma tarefa fácil, especialmente em ambientes não estacionários, cuja definição de um valor único pode trazer bons resultados em certos momentos, mas após mudanças de conceito, esse valor pode se tornar obsoleto. Com os experimentos foi possível notar os parâmetros que mais influenciam os resultados dos métodos, sendo eles: o número de rótulos atribuídos aos exemplos na classificação e o liminar para distinguir entre extensões e PNs, tanto relacionado a distância, quanto ao número de modelos de decisão.

- **Modelo de decisão baseado em hiperesferas**: o modelo de decisão é formado por microgrupos para sua atualização constante, mesmo sem ter o rótulo verdadeiro dos exemplos. Dessa forma, o método pode trazer resultados ruins para bases de dados cuja distribuição das classes não é esférica. Foram realizadas tentativas de minimizar o problema gerando um número maior de microgrupos para representar as classes, mas ainda assim essa é uma limitação para os métodos desenvolvidos neste trabalho. Outro ponto a ser destacado é a dificuldade encontrada nos procedimento de DN usando modelos de decisão baseados em hiperesferas. Essa dificuldade vem da alta sobreposição de classes característica de problemas multirrótulo.

- **Número de rótulos atribuído aos exemplos:** para classificação de um exemplo, é necessário decidir quantas classes devem ser associadas ao exemplo. Neste trabalho, esse número é definido de acordo com a cardinalidade de rótulos da base de treinamento. No entanto, essa abordagem não é a ideal, visto que a cardinalidade de rótulos é uma média calculada em uma pequena porção dos dados disponíveis para o treinamento e, por conta das mudanças de conceito, essa média pode variar ao longo do fluxo de dados.

- **Técnica para tratamento de evoluções de conceito**: Uma das limitações do trabalho diz respeito às técnicas para tratamento de evoluções de conceito. As evoluções de conceito são tratadas criando um novo modelo de decisão com novos microgrupos e cópias dos

microgrupos dos modelos de decisão próximos. Assim, quando novas classes surgem parcialmente sobrepostas as outras, o método consegue identificá-las. No entanto, quando uma nova classe surge totalmente sobreposta as outras, essa classe só é identificada corretamente pelo método se ela se afastar das demais em algum momento do fluxo de dados.

- **Metodologia de avaliação**: A metodologia de avaliação apresentada neste trabalho representa uma contribuição na busca por metodologias de avaliação adequadas para os algoritmos que tratam CMFCD através de DN. Ela associa padrões-novidade, formados por exemplos não rotulados, às classes conhecidas do problema. Após essa associação é possível calcular as medidas de avaliação multirrótulo tradicionais. No entanto, a metodologia de avaliação proposta precisa ser refinada. Um dos problemas já identificados é que ela não penaliza um classificador quando muitos padrões-novidades são identificados para representar uma classe. Quanto mais padrões-novidades um classificador cria, mais chances há de que ele consiga separar bem as classes novidades. No entanto, é preciso ponderar entre a taxa de erro versus o número de novidades criadas. Ainda, por se tratar de um problema multirrótulo, os exemplos usados para formar os padrões-novidade podem estar associados a mais de uma classe. Portanto, os padrões-novidade devem, também, ser associados a mais de uma classe, e isso ainda não é tratado na metodologia proposta.

- **Dados não estacionários sintéticos:** As bases de dados não estacionárias usadas nos experimentos são todas sintéticas e adaptadas de bases multiclasse com sobreposições de grupos. Isso nem sempre pode refletir comportamentos de problemas de CMFCD reais, o que dificulta a avaliação dos métodos em relação às mudanças de conceito e evoluções de conceito.

## 7.3 Trabalhos Futuros

Alguns dos possíveis trabalhos futuros são descritos a seguir:

- **Desenvolvimento de novas estratégias para diferenciar entre padrões-novidade e extensões**: a escolha do limiar que separa extensões das novidades é um fator importante no desempenho do MINAS-BR. Novas estratégias para calcular automaticamente este valor devem ser estudadas.

- **Uso de aprendizado ativo**: O uso de aprendizado ativo em tarefas de DN em FCD parece ser uma boa alternativa. Solicitar o rótulo verdadeiro de apenas alguns exemplos do fluxo pode aumentar o desempenho preditivo do classificador sem exigir muito esforço na rotulação. Ainda, é interessante explorar a rotulagem dos novos microgrupos formados pelo processo de DN.

- **Tratamento de Contextos Recorrentes**: Contextos recorrentes são uma característica comum aos cenários envolvendo FCD, mas que ainda não foi explorada na CM. Portanto, o tratamento de contextos recorrentes na CMFCD deve ser explorado.

- **Bases de dados experimentais**: Poucas bases de dados reais e artificiais estão disponíveis para serem usadas em experimentos envolvendo FCDs multirrótulo. Faltam bases que simulem a presença de ruídos e mudança de conceito específicas como cardinalidade e dependência de rótulos, mudança no balanceamento das classes e surgimento de novas classes. A criação de um repositório de dados que possa ser usado na validação de experimentos com algoritmos para FCDs pode trazer importantes benefícios para a pesquisa na área.

- **Métodos de classificação multirrótulo globais**: os métodos de transformação de problemas, e alguns métodos de adaptação de problemas, lidam com a CM de forma local, em que um conjunto de classificadores é induzido, sendo cada classificador responsável pela predição de uma classe, ou de um conjunto de classes e, a classificação de exemplos ocorrem em vários passos, combinando as predições de vários classificadores previamente treinados. O uso de abordagens globais para CMFCD e DN pode trazer vantagens. Nesta abordagem um único classificador é induzido considerando todas as classes ao mesmo tempo, sem passá-las por nenhum tipo de transformação e a classificação ocorre em apenas um passo. Portanto, técnicas dessa abordagem deverão ser estudadas.

# REFERÊNCIAS


ABDALLAH, Z. S.; GABER, M. M.; SRINIVASAN, B.; KRISHNASWAMY, S. Anynovel: detection of novel concepts in evolving data streams. *Evolving Systems*, Springer, v. 7, n. 2, p. 73–93, 2016. Citado na página 53.

AGGARWAL, C. C. *Data streams: models and algorithms*. [S.l.]: Springer Science & Business Media, 2007. v. 31. Citado 6 vezes nas páginas 16, 17, 22, 23, 24 e 27.

AGGARWAL, C. C. A survey of stream classification algorithms. In: ____. [S.l.]: CRC Press, 2014. cap. 9, p. 245–274. Citado na página 29.

AGGARWAL, C. C.; HAN, J.; WANG, J.; YU, P. S. A framework for clustering evolving data streams. In: VLDB ENDOWMENT. *Proceedings of the 29th international conference on Very large data bases-Volume 29*. [S.l.], 2003. p. 81–92. Citado 3 vezes nas páginas 22, 24 e 26.

AGGARWAL, C. C.; HAN, J.; WANG, J.; YU, P. S. A framework for projected clustering of high dimensional data streams. In: VLDB ENDOWMENT. *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. [S.l.], 2004. p. 852–863. Citado na página 44.

AGGARWAL, C. C.; HAN, J.; WANG, J.; YU, P. S. 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. Citado 2 vezes nas páginas 29 e 71.

AGRAWAL, R.; SRIKANT, R. et al. Fast algorithms for mining association rules. In: *Proc. 20th int. conf. very large data bases, VLDB*. [S.l.: s.n.], 1994. v. 1215, p. 487–499. Citado na página 69.

AHMADI, Z.; KRAMER, S. A label compression method for online multi-label classification. *Pattern Recognition Letters*, Elsevier, v. 111, p. 64–71, 2018. Citado na página 17.

AL-KHATEEB, T.; MASUD, M. M.; KHAN, L.; AGGARWAL, C.; HAN, J.; THURAISINGHAM, B. Stream classification with recurring and novel class detection using class-based ensemble. In: IEEE. *Data Mining (ICDM), 2012 IEEE 12th International Conference on*. [S.l.], 2012. p. 31–40. Citado 10 vezes nas páginas 18, 55, 57, 61, 65, 70, 75, 78, 95 e 104.

AL-KHATEEB, T. M.; MASUD, M. M.; KHAN, L.; THURAISINGHAM, B. Cloud guided stream classification using class-based ensemble. In: IEEE. *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*. [S.l.], 2012. p. 694–701. Citado 4 vezes nas páginas 55, 57, 61 e 65.

ALBERTINI, M. K.; MELLO, R. F. de. A self-organizing neural network for detecting novelties. In: ACM. *Proceedings of the 2007 ACM symposium on Applied computing*. [S.l.], 2007. p. 462–466. Citado na página 62.

ALBERTINI, M. K.; MELLO, R. F. de. *A Self-Organizing Neural Network Approach Novelty Detection*. [S.l.]: IGI GLOBAL, 2010. Citado na página 51.

ALVARES-CHERMAN, E.; METZ, J.; MONARD, M. C. Incorporating label dependency into the binary relevance framework for multi-label classification. *Expert Systems with Applications*, Elsevier, v. 39, n. 2, p. 1647–1655, 2012. Citado na página 27.

APPICE, A.; DZEROSKI, S. Stepwise induction of multi-target model trees. In: SPRINGER. *18th European Conference on Machine Learning, Warsaw, Poland, September 17-21, 2007. Proceedings*. [S.l.], 2007. v. 7, p. 502–509. Citado na página 42.

BABCOCK, B.; BABU, S.; DATAR, M.; MOTWANI, R.; WIDOM, J. Models and issues in data stream systems. In: ACM. *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. [S.l.], 2002. p. 1–16. Citado na página 22.

BENDALE, A.; BOULT, T. Towards open world recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2015. p. 1893–1902. Citado na página 54.

BIFET, A.; GAVALDA, R. Learning from time-changing data with adaptive windowing. In: SIAM. *Proceedings of the 2007 SIAM International Conference on Data Mining*. [S.l.], 2007. p. 443–448. Citado na página 30.

BIFET, A.; GAVALDÀ, R. Adaptive learning from evolving data streams. In: SPRINGER. *International Symposium on Intelligent Data Analysis*. [S.l.], 2009. p. 249–260. Citado na página 28.

BIFET, A.; HOLMES, G.; KIRKBY, R.; PFAHRINGER, B. Moa: Massive online analysis. *Journal of Machine Learning Research*, v. 11, p. 1601–1604, 2010. Citado 3 vezes nas páginas 61, 86 e 89.

BIFET, A.; HOLMES, G.; PFAHRINGER, B.; KIRKBY, R.; GAVALDÀ, R. New ensemble methods for evolving data streams. In: ACM. *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.], 2009. p. 139–148. Citado 2 vezes nas páginas 30 e 45.

BRADLEY, P. S.; FAYYAD, U. M.; REINA, C. et al. Scaling clustering algorithms to large databases. In: *KDD*. [S.l.: s.n.], 1998. p. 9–15. Citado na página 26.

CAO, F.; ESTERT, M.; QIAN, W.; ZHOU, A. Density-based clustering over an evolving data stream with noise. In: SIAM. *Proceedings of the 2006 SIAM international conference on data mining*. [S.l.], 2006. p. 328–339. Citado 3 vezes nas páginas 24, 26 e 48.

CARVALHO, A. C. de; FREITAS, A. A. A tutorial on multi-label classification techniques. In: *Foundations of Computational Intelligence Volume 5*. [S.l.]: Springer, 2009. p. 177–195. Citado na página 32.

CERRI, R.; SILVA, R. R. da; CARVALHO, A. C. de. Comparing methods for multilabel classification of proteins using machine learning techniques. In: SPRINGER. *Brazilian Symposium on Bioinformatics*. [S.l.], 2009. p. 109–120. Citado na página 17.

CHERMAN, E. A. *Aprendizado de máquina multirrótulo: explorando a dependência de rótulos e o aprendizado ativo*. Tese (Doutorado) — Universidade de São Paulo, 2013. Citado 2 vezes nas páginas 34 e 36.

CIACCIA, P.; PATELLA, M.; ZEZULA, P. M-tree: An eæcient access method for similarity search in metric spaces. In: *Proceedings of the 23rd International Conference on Very Large Databases, Athens, Greece*. [S.l.: s.n.], 1997. p. 426–435. Citado na página 29.

CLARE, A.; KING, R. D. Knowledge discovery in multi-label phenotype data. In: SPRINGER. *European Conference on Principles of Data Mining and Knowledge Discovery*. [S.l.], 2001. p. 42–53. Citado 3 vezes nas páginas 17, 44 e 49.

COSTA, F. d. O.; ECKMANN, M.; SCHEIRER, W. J.; ROCHA, A. Open set source camera attribution. In: IEEE. *Graphics, Patterns and Images (SIBGRAPI), 2012 25th SIBGRAPI Conference on*. [S.l.], 2012. p. 71–78. Citado na página 53.

COSTA, F. d. O.; SILVA, E.; ECKMANN, M.; SCHEIRER, W. J.; ROCHA, A. Open set source camera attribution and device linking. *Pattern Recognition Letters*, Elsevier, v. 39, p. 92–101, 2014. Citado na página 53.

COULL, S.; BRANCH, J.; SZYMANSKI, B.; BREIMER, E. Intrusion detection: A bioinformatics approach. In: IEEE. *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*. [S.l.], 2003. p. 24–33. Citado na página 51.

DAGAN, I.; ENGELSON, S. P. Committee-based sampling for training probabilistic classifiers. In: THE MORGAN KAUFMANN SERIES IN MACHINE LEARNING,(SAN FRANCISCO, CA, USA). *Proceedings of the Twelfth International Conference on Machine Learning*. [S.l.], 1995. p. 150–157. Citado na página 41.

DAVE, M.; TAPIAWALA, S.; ER, M. J.; VENKATESAN, R. A novel progressive multi-label classifier for class-incremental data. In: IEEE. *Systems, Man, and Cybernetics (SMC), 2016 IEEE International Conference on*. [S.l.], 2016. p. 003589–003593. Citado 3 vezes nas páginas 17, 43 e 47.

DEMBSZYNSKI, K.; WAEGEMAN, W.; CHENG, W.; HÜLLERMEIER, E. On label dependence in multilabel classification. In: GHENT UNIVERSITY, KERMIT, DEPARTMENT OF APPLIED MATHEMATICS, BIOMETRICS AND PROCESS CONTROL. *LastCFP: ICML Workshop on Learning from Multi-label data*. [S.l.], 2010. Citado na página 34.

DEMPSTER, A. P.; LAIRD, N. M.; RUBIN, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, JSTOR, p. 1–38, 1977. Citado na página 48.

DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, v. 7, n. Jan, p. 1–30, 2006. Citado na página 101.

DOMINGOS, P.; HULTEN, G. Mining high-speed data streams. In: ACM. *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.], 2000. p. 71–80. Citado na página 28.

DRIES, A.; RÜCKERT, U. Adaptive concept drift detection. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, Wiley Subscription Services, Inc., A Wiley Company, v. 2, n. 5-6, p. 311–327, 2009. Citado na página 52.

ELWELL, R.; POLIKAR, R. Incremental learning of concept drift in nonstationary environments. *IEEE Transactions on Neural Networks*, IEEE, v. 22, n. 10, p. 1517–1531, 2011. Citado na página 52.

FARIA, E. R.; GONÇALVES, I. J.; CARVALHO, A. C. de; GAMA, J. Novelty detection in data streams. *Artificial Intelligence Review*, Springer, v. 45, n. 2, p. 235–269, 2016. Citado 12 vezes nas páginas 16, 18, 50, 51, 52, 53, 54, 55, 58, 61, 62 e 65.

FARIA, E. R. de; FERREIRA, A. C. P. de L.; GAMA, J. 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. Citado 20 vezes nas páginas 18, 52, 55, 56, 58, 59, 60, 65, 70, 71, 73, 74, 75, 76, 78, 81, 84, 95, 104 e 105.

FARIA, E. R. de; GONCALVES, I. R.; GAMA, J.; FERREIRA, A. C. P. de L. et al. Evaluation of multiclass novelty detection algorithms for data streams. *Transactions on Knowledge and Data Engineering*, IEEE, v. 27, n. 11, p. 2961–2973, 2015. Citado 2 vezes nas páginas 62 e 93.

FARID, D. M.; ZHANG, L.; HOSSAIN, A.; RAHMAN, C. M.; STRACHAN, R.; SEXTON, G.; DAHAL, K. An adaptive ensemble classifier for mining concept drifting data streams. *Expert Systems with Applications*, Elsevier, v. 40, n. 15, p. 5895–5906, 2013. Citado 3 vezes nas páginas 53, 61 e 65.

FARNSTROM, F.; LEWIS, J.; ELKAN, C. Scalability for clustering algorithms revisited. *ACM SIGKDD Explorations Newsletter*, ACM, v. 2, n. 1, p. 51–57, 2000. Citado na página 26.

FOURNIER-VIGER, P.; LIN, J. C.-W.; VO, B.; CHI, T. T.; ZHANG, J.; LE, H. B. A survey of itemset mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, Wiley Online Library, v. 7, n. 4, p. e1207, 2017. Citado na página 69.

GAMA, J. *Knowledge discovery from data streams*. [S.l.]: Chapman & Hall/CRC Data Mining and Knowledge Discovery Series, 2010. Citado 12 vezes nas páginas 16, 17, 18, 22, 23, 25, 26, 50, 51, 52, 64 e 65.

GAMA, J.; GABER, M. M. *Learning from data streams*. [S.l.]: Springer, 2007. Citado 2 vezes nas páginas 16 e 22.

GAMA, J.; SEBASTIÃO, R.; RODRIGUES, P. P. Issues in evaluation of stream learning algorithms. In: ACM. *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.], 2009. p. 329–338. Citado na página 92.

GAMA, J.; SEBASTIÃO, R.; RODRIGUES, P. P. On evaluating stream learning algorithms. *Machine learning*, Springer, v. 90, n. 3, p. 317–346, 2013. Citado na página 53.

GIBAJA, E.; VENTURA, S. A tutorial on multilabel learning. *ACM Computing Surveys (CSUR)*, ACM, v. 47, n. 3, p. 52, 2015. Citado 3 vezes nas páginas 34, 35 e 37.

GODBOLE, S.; SARAWAGI, S. Discriminative methods for multi-labeled classification. In: SPRINGER. *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. [S.l.], 2004. p. 22–30. Citado na página 38.

HAN, J.; PEI, J.; KAMBER, M. *Data mining: concepts and techniques*. [S.l.]: Elsevier, 2011. Citado na página 27.

HAYAT, M. Z.; HASHEMI, M. R. A dct based approach for detecting novelty and concept drift in data streams. In: IEEE. *Soft Computing and Pattern Recognition (SoCPaR), 2010 International Conference of*. [S.l.], 2010. p. 373–378. Citado 5 vezes nas páginas 51, 53, 55, 57 e 61.

HUANG, G.-B.; SIEW, C.-K. Extreme learning machine: Rbf network case. In: IEEE. *Control, Automation, Robotics and Vision Conference, 2004. ICARCV 2004 8th*. [S.l.], 2004. v. 2, p. 1029–1036. Citado na página 48.

HUANG, G.-B.; WANG, D. H.; LAN, Y. Extreme learning machines: a survey. *International journal of machine learning and cybernetics*, Springer, v. 2, n. 2, p. 107–122, 2011. Citado na página 47.

HULTEN, G.; SPENCER, L.; DOMINGOS, P. Mining time-changing data streams. In: ACM. *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.], 2001. p. 97–106. Citado na página 28.

IKONOMOVSKA, E.; GAMA, J.; DŽEROSKI, S. Incremental multi-target model trees for data streams. In: ACM. *Proceedings of the 2011 ACM symposium on applied computing*. [S.l.], 2011. p. 988–993. Citado na página 42.

JAIN, L. P.; SCHEIRER, W. J.; BOULT, T. E. Multi-class open set recognition using probability of inclusion. In: SPRINGER. *European Conference on Computer Vision*. [S.l.], 2014. p. 393–409. Citado na página 53.

JUNIOR, C. E. dos S.; SCHWARTZ, W. R. Extending face identification to open-set face recognition. In: IEEE. *Graphics, Patterns and Images (SIBGRAPI), 2014 27th SIBGRAPI Conference on*. [S.l.], 2014. p. 188–195. Citado na página 54.

JÚNIOR, P. R. M.; BOULT, T. E.; WAINER, J.; ROCHA, A. Specialized support vector machines for open-set recognition. *arXiv preprint arXiv:1606.03802*, 2016. Citado na página 53.

KATAKIS, I.; TSOUMAKAS, G.; VLAHAVAS, I. Multilabel text classification for automated tag suggestion. *Proceedings of the ECML/PKDD 2008 Discovery Challenge*, v. 75, 2008. Citado na página 17.

KAZAWA, H.; IZUMITANI, T.; TAIRA, H.; MAEDA, E. Maximal margin labeling for multi-topic text categorization. In: MIT PRESS. *Proceedings of the 17th International Conference on Neural Information Processing Systems*. [S.l.], 2004. p. 649–656. Citado na página 17.

KONG, X.; PHILIP, S. Y. An ensemble-based approach to fast classification of multi-label data streams. In: IEEE. *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2011 7th International Conference On*. [S.l.], 2011. p. 95–104. Citado na página 43.

KRANEN, P.; ASSENT, I.; BALDAUF, C.; SEIDL, T. The clustree: indexing micro-clusters for anytime stream mining. *Knowledge and information systems*, Springer, v. 29, n. 2, p. 249–272, 2011. Citado na página 26.

KRAWCZYK, B.; WOŹNIAK, M. Incremental learning and forgetting in one-class classifiers for data streams. In: SPRINGER. *Proceedings of the 8th International Conference on Computer Recognition Systems CORES 2013*. [S.l.], 2013. p. 319–328. Citado 4 vezes nas páginas 55, 56, 57 e 62.

KREMPL, G.; ŽLIOBAITE, I.; BRZEZIŃSKI, D.; HÜLLERMEIER, E.; LAST, M.; LEMAIRE, V.; NOACK, T.; SHAKER, A.; SIEVI, S.; SPILIOPOULOU, M. et al. Open challenges for data stream mining research. *ACM SIGKDD explorations newsletter*, ACM, v. 16, n. 1, p. 1–10, 2014. Citado 2 vezes nas páginas 17 e 65.

LEITE, D.; COSTA, P.; GOMIDE, F. Evolving granular neural network for semi-supervised data stream classification. In: IEEE. *Neural Networks (IJCNN), The 2010 International Joint Conference on*. [S.l.], 2010. p. 1–8. Citado na página 30.

LEITE, D. F.; COSTA, P.; GOMIDE, F. Evolving granular classification neural networks. In: IEEE. *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*. [S.l.], 2009. p. 1736–1743. Citado na página 29.

LI, X.; CROFT, W. B. Improving novelty detection for general topics using sentence level information patterns. In: ACM. *Proceedings of the 15th ACM international conference on Information and knowledge management*. [S.l.], 2006. p. 238–247. Citado 2 vezes nas páginas 52 e 53.

LIANG, N.-Y.; HUANG, G.-B.; SARATCHANDRAN, P.; SUNDARARAJAN, N. A fast and accurate online sequential learning algorithm for feedforward networks. *IEEE Transactions on neural networks*, IEEE, v. 17, n. 6, p. 1411–1423, 2006. Citado na página 47.

LOPES, P. d. A.; CAMARGO, H. d. A. Fuzzstream: Fuzzy data stream clustering based on the online-offline framework. In: IEEE. *Fuzzy Systems (FUZZ-IEEE), 2017 IEEE International Conference on*. [S.l.], 2017. p. 1–6. Citado na página 24.

MA, B. L. W. H. Y.; LIU, B. Integrating classification and association rule mining. In: *Proceedings of the fourth international conference on knowledge discovery and data mining*. [S.l.: s.n.], 1998. Citado na página 48.

MAHDIRAJI, A. R. Clustering data stream: A survey of algorithms. *International Journal of Knowledge-based and Intelligent Engineering Systems*, IOS Press, v. 13, n. 2, p. 39–44, 2009. Citado na página 23.

MASUD, M.; GAO, J.; KHAN, L.; HAN, J.; THURAISINGHAM, B. M. 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, 2011. Citado 13 vezes nas páginas 18, 49, 51, 55, 56, 57, 61, 65, 70, 75, 78, 95 e 104.

MASUD, M. M.; AL-KHATEEB, T. M.; KHAN, L.; AGGARWAL, C.; GAO, J.; HAN, J.; THURAISINGHAM, B. Detecting recurring and novel classes in concept-drifting data streams. In: IEEE. *Data Mining (ICDM), 2011 IEEE 11th International Conference on*. [S.l.], 2011. p. 1176–1181. Citado 2 vezes nas páginas 49 e 65.

MASUD, M. M.; CHEN, Q.; KHAN, L.; AGGARWAL, C.; GAO, J.; HAN, J.; THURAISINGHAM, B. Addressing concept-evolution in concept-drifting data streams. In: IEEE. *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. [S.l.], 2010. p. 929–934. Citado 4 vezes nas páginas 55, 57, 61 e 65.

MENSINK, T.; VERBEEK, J.; PERRONNIN, F.; CSURKA, G. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 35, n. 11, p. 2624–2637, 2013. Citado na página 54.

MITCHELL, T. *Machine Learning*. [S.l.]: McGraw Hill, 1997. Citado 2 vezes nas páginas 16 e 22.

NGUYEN, H.-L.; WOON, Y.-K.; NG, W.-K. A survey on data stream clustering and classification. *Knowledge and information systems*, Springer, v. 45, n. 3, p. 535–569, 2015. Citado 5 vezes nas páginas 17, 23, 24, 25 e 28.

NGUYEN, T. T.; DANG, M. T.; LUONG, A. V.; LIEW, A. W.-C.; LIANG, T.; MCCALL, J. Multi-label classification via incremental clustering on evolving data stream. *Pattern Recognition*, Elsevier, 2019. Citado 4 vezes nas páginas 17, 43, 48 e 71.

OSOJNIK, A.; PANOV, P.; DŽEROSKI, S. Multi-label classification via multi-target regression on data streams. *Machine Learning*, Springer, v. 106, n. 6, p. 745–770, 2017. Citado 4 vezes nas páginas 17, 39, 42 e 92.

OZA, N. C. Online bagging and boosting. In: IEEE. *Systems, man and cybernetics, 2005 IEEE international conference on*. [S.l.], 2005. v. 3, p. 2340–2345. Citado 2 vezes nas páginas 30 e 42.

PAIVA, E. R. d. F. *Detecção de novidade em fluxos contínuos de dados multiclasse*. Tese (Doutorado) — Universidade de São Paulo, 2014. Citado na página 27.

PERNER, P. Concepts for novelty detection and handling based on a case-based reasoning process scheme. *Engineering Applications of Artificial Intelligence*, Elsevier, v. 22, n. 1, p. 86–91, 2009. Citado 3 vezes nas páginas 18, 51 e 54.

PHILLIPS, P. J.; GROTHER, P.; MICHEALS, R. Evaluation methods in face recognition. In: *Handbook of face recognition*. [S.l.]: Springer, 2011. p. 551–574. Citado na página 53.

PISANI, P. H.; LORENA, A. C. Detecção de intrusões com dinâmica da digitação: uma revisão sistemática. *Universidade Federal do ABC, Santo André, Brasil, Technical Report*, v. 6, p. 2011, 2011. Citado 2 vezes nas páginas 51 e 52.

PISANI, P. H.; LORENA, A. C.; ANDRÉ, C.; CARVALHO, F. de et al. Adaptive positive selection for keystroke dynamics. *Journal of Intelligent & Robotic Systems*, Springer Science & Business Media, v. 80, p. 277, 2015. Citado 2 vezes nas páginas 51 e 52.

PISANI, P. H.; LORENA, A. C.; CARVALHO, A. C. de. Adaptive algorithms applied to accelerometer biometrics in a data stream context. *Intelligent Data Analysis*, IOS Press, v. 21, n. 2, p. 353–370, 2017. Citado 2 vezes nas páginas 51 e 52.

QU, W.; ZHANG, Y.; ZHU, J.; QIU, Q. Mining multi-label concept-drifting data streams using dynamic classifier ensemble. In: SPRINGER. *First Asian Conference on Machine Learning, ACML 2009, Nanjing, China, November 2-4, 2009. Proceedings*. [S.l.], 2009. p. 308–321. Citado 2 vezes nas páginas 38 e 39.

RAI, P.; III, H. D.; VENKATASUBRAMANIAN, S. Streamed learning: one-pass svms. *IJCAI'09: Proceedings of the 21st International Jont Conference on Artifical Intelligence, Morgan Kaufmann Publishers.*, p. 1211–1216, 2009. Citado na página 24.

RATTANI, A.; SCHEIRER, W. J.; ROSS, A. Open set fingerprint spoof detection across novel fabrication materials. *IEEE Transactions on Information Forensics and Security*, IEEE, v. 10, n. 11, p. 2447–2460, 2015. Citado na página 54.

READ, J.; BIFET, A.; HOLMES, G.; PFAHRINGER, B. Scalable and efficient multi-label classification for evolving data streams. *Machine Learning*, Springer, v. 88, n. 1-2, p. 243–272, 2012. Citado 6 vezes nas páginas 32, 33, 40, 43, 44 e 92.

READ, J.; BIFET, A.; PFAHRINGER, B.; HOLMES, G. Batch-incremental versus instance-incremental learning in dynamic and evolving data. *Advances in Intelligent Data Analysis XI*, Springer, p. 313–323, 2012. Citado na página 24.

READ, J.; BOUCKAERT, R. R.; FRANK, E.; HALL, M. A.; HOLMES, G.; PFAHRINGER, B.; REUTEMANN, P.; WITTEN, I. H.; BIFET, A.; FRANK, E. et al. Efficient multi-label classification for evolving data streams. *Journal of Machine Learning Research*, JMLR, v. 21, p. 1141–1142, 2010. Citado na página 44.

READ, J.; PFAHRINGER, B.; HOLMES, G. Multi-label classification using ensembles of pruned sets. In: IEEE. *International Conference on Data Mining*. [S.l.], 2008. p. 995–1000. Citado 2 vezes nas páginas 33 e 92.

READ, J.; PFAHRINGER, B.; HOLMES, G. Generating synthetic multi-label data streams. In: *ECML/PKKD 2009 Workshop on Learning from Multi-label Data (MLD'09)*. [S.l.: s.n.], 2009. p. 69–84. Citado na página 27.

READ, J.; PFAHRINGER, B.; HOLMES, G.; FRANK, E. Classifier chains for multi-label classification. *Machine learning*, Springer, v. 85, n. 3, p. 333–359, 2011. Citado 2 vezes nas páginas 33 e 92.

READ, J.; REUTEMANN, P.; PFAHRINGER, B.; HOLMES, G. MEKA: A multi-label/multi-target extension to Weka. *Journal of Machine Learning Research*, v. 17, n. 21, p. 1–5, 2016. Disponível em: <http://jmlr.org/papers/v17/12-164.html>. Citado na página 36.

RISTIN, M.; GUILLAUMIN, M.; GALL, J.; GOOL, L. V. Incremental learning of ncm forests for large-scale image classification. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2014. p. 3654–3661. Citado na página 54.

ROSA, R. D.; MENSINK, T.; CAPUTO, B. Online open world recognition. *arXiv preprint arXiv:1604.02275*, 2016. Citado na página 54.

RUSIECKI, A. Robust neural network for novelty detection on data streams. In: SPRINGER. *International Conference on Artificial Intelligence and Soft Computing*. [S.l.], 2012. p. 178–186. Citado 4 vezes nas páginas 55, 56, 57 e 62.

SCHEIRER, W. J.; JAIN, L. P.; BOULT, T. E. Probability models for open set recognition. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 36, n. 11, p. 2317–2324, 2014. Citado na página 53.

SCHEIRER, W. J.; ROCHA, A. de R.; SAPKOTA, A.; BOULT, T. E. Toward open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, v. 35, n. 7, p. 1757–1772, 2013. Citado na página 53.

SHI, Z.; WEN, Y.; XUE, Y.; CAI, G. Efficient class incremental learning for multi-label classification of evolving data streams. In: IEEE. *International Joint Conference on Neural Networks*. [S.l.], 2014. p. 2093–2099. Citado 3 vezes nas páginas 43, 45 e 48.

SILVA, J. A.; FARIA, E. R.; BARROS, R. C.; HRUSCHKA, E. R.; CARVALHO, A. C. de; GAMA, J. Data stream clustering: A survey. *ACM Computing Surveys (CSUR)*, ACM, v. 46, n. 1, p. 13, 2013. Citado 6 vezes nas páginas 16, 22, 25, 26, 64 e 65.

SILVA, J. d. A. *Agrupamento de dados em fluxos contínuos com estimativa automática do número de grupos*. Tese (Doutorado) — Universidade de São Paulo, 2015. Citado na página 23.

SINGH, S.; MARKOU, M. An approach to novelty detection applied to the classification of image regions. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, v. 16, n. 4, p. 396–407, 2004. Citado na página 51.

SONG, G.; YE, Y. A new ensemble method for multi-label data stream classification in non-stationary environment. In: IEEE. *International Joint Conference on Neural Networks*. [S.l.], 2014. p. 1776–1783. Citado 2 vezes nas páginas 43 e 46.

SONG, Y.; ZHANG, L.; GILES, C. L. A sparse gaussian processes classification framework for fast tag suggestions. In: ACM. *Proceedings of the 17th ACM conference on Information and knowledge management*. [S.l.], 2008. p. 93–102. Citado na página 17.

SOUZA, V. M.; SILVA, D. F.; BATISTA, G. E.; GAMA, J. Classification of evolving data streams with infinitely delayed labels. In: IEEE. *International Conference on Machine Learning and Applications*. [S.l.], 2015. p. 214–219. Citado 5 vezes nas páginas 17, 22, 49, 53 e 75.

SOUZA, V. M. A.; SILVA, D. F.; GAMA, J.; BATISTA, G. E. A. P. A. Data stream classification guided by clustering on nonstationary environments and extreme verification latency. In: *Proceedings of SIAM International Conference on Data Mining (SDM)*. [S.l.: s.n.], 2015. p. 873–881. Citado 4 vezes nas páginas 17, 65, 87 e 89.

SOUZA, V. M. A. d. *Classificação de fluxo de dados não estacionários com aplicação em sensores identificadores de insetos*. Tese (Doutorado) — Universidade de São Paulo, 2016. Citado na página 65.

SPATH, H. *Cluster analysis algorithms for data reduction and classification of objects*. [S.l.]: Ellis Horwood Chichester, 1980. Citado 2 vezes nas páginas 73 e 81.

SPINOSA, E. J.; CARVALHO, A. C. P. L. F. de. Svms for novel class detection in bioinformatics. *Brazilian Workshop on Bioinformatics*, v. 2004, p. 81–88, 2004. Citado na página 51.

SPINOSA, E. J.; CARVALHO, A. P. de Leon F de; GAMA, J. Cluster-based novel concept detection in data streams applied to intrusion detection in computer networks. In: ACM. *Proceedings of the 2008 ACM Symposium on Applied computing*. [S.l.], 2008. p. 976–980. Citado 2 vezes nas páginas 51 e 53.

SPINOSA, E. J.; LEON, F. de; PONCE, A.; GAMA, J. Novelty detection with application to data streams. *Intelligent Data Analysis*, IOS Press, v. 13, n. 3, p. 405–422, 2009. Citado 10 vezes nas páginas 18, 55, 57, 61, 70, 71, 75, 78, 95 e 104.

TAN, S. C.; TING, K. M.; LIU, T. F. Fast anomaly detection for streaming data. In: *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*. [S.l.: s.n.], 2011. v. 22, p. 1511. Citado na página 62.

TANG, L.; RAJAN, S.; NARAYANAN, V. K. Large scale multi-label classification via metalabeler. In: ACM. *Proceedings of the 18th international conference on World wide web*. [S.l.], 2009. p. 211–220. Citado na página 17.

TRAJDOS, P.; KURZYNSKI, M. Multi-label stream classification using extended binary relevance model. In: IEEE. *Trustcom/BigDataSE/ISPA*. [S.l.], 2015. v. 2, p. 205–210. Citado 3 vezes nas páginas 17, 39 e 41.

TSOUMAKAS, G.; KATAKIS, I. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, v. 3, n. 3, 2006. Citado na página 17.

TSOUMAKAS, G.; KATAKIS, I.; VLAHAVAS, I. Mining multi-label data. In: *Data mining and knowledge discovery handbook*. [S.l.]: Springer, 2009. p. 667–685. Citado 7 vezes nas páginas 27, 32, 33, 34, 36, 64 e 76.

TSOUMAKAS, G.; VLAHAVAS, I. Random k-labelsets: An ensemble method for multilabel classification. In: SPRINGER. *European Conference on Machine Learning*. [S.l.], 2007. p. 406–417. Citado na página 35.

TSYMBAL, A. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*, v. 106, n. 2, 2004. Citado na página 53.

UNO, T.; KIYOMI, M.; ARIMURA, H. Lcm ver. 2: Efficient mining algorithms for frequent/-closed/maximal itemsets. In: *Fimi*. [S.l.: s.n.], 2004. v. 126. Citado na página 69.

VALLIM, R. M.; FILHO, J. A. A.; MELLO, R. F. D.; CARVALHO, A. C. D. Online behavior change detection in computer games. *Expert Systems with Applications*, Elsevier, v. 40, n. 16, p. 6258–6265, 2013. Citado na página 52.

VENKATESAN, R.; ER, M. J.; DAVE, M.; PRATAMA, M.; WU, S. A novel online multi-label classifier for high-speed streaming data applications. *Evolving Systems*, Springer, v. 8, n. 4, p. 303–315, 2017. Citado 3 vezes nas páginas 17, 43 e 47.

VORBURGER, P.; BERNSTEIN, A. Entropy-based concept shift detection. In: IEEE. *Data Mining, 2006. ICDM'06. Sixth International Conference on*. [S.l.], 2006. p. 1113–1118. Citado na página 49.

WANG, H.; FAN, W.; YU, P. S.; HAN, J. Mining concept-drifting data streams using ensemble classifiers. In: ACM. *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.], 2003. p. 226–235. Citado 2 vezes nas páginas 30 e 51.

WANG, L. P.; WAN, C. R. *Comments and Replies Comments on "The Extreme Learning Machine"*. 2008. Citado na página 48.

WANG, P.; ZHANG, P.; GUO, L. Mining multi-label data streams using ensemble-based active learning. In: SIAM. *international conference on data mining*. [S.l.], 2012. p. 1131–1140. Citado 2 vezes nas páginas 39 e 40.

XIOUFIS, E.; SPILIOPOULOU, M.; TSOUMAKAS, G.; VLAHAVAS, I. Dealing with concept drift and class imbalance in multi-label stream classification. In: *Twenty-Second International Joint Conference on Artificial Intelligence*. [S.l.: s.n.], 2011. Citado 2 vezes nas páginas 29 e 39.

YANG, Y.; GOPAL, S. Multilabel classification with meta-level features in a learning-to-rank framework. *Machine Learning*, Springer, v. 88, n. 1, p. 47–68, 2012. Citado na página 17.

ZHANG, J.; YAN, Q.; ZHANG, Y.; HUANG, Z. Novel fault class detection based on novelty detection methods. In: *Intelligent Computing in Signal Processing and Pattern Recognition*. [S.l.]: Springer, 2006. p. 982–987. Citado na página 51.

ZHANG, M.-L.; ZHOU, Z.-H. A k-nearest neighbor based algorithm for multi-label classification. In: IEEE. *Granular Computing, 2005 IEEE International Conference on*. [S.l.], 2005. v. 2, p. 718–721. Citado na página 17.

ZHANG, M.-L.; ZHOU, Z.-H. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, IEEE, v. 26, n. 8, p. 1819–1837, 2014. Citado 2 vezes nas páginas 33 e 34.

ZHANG, P.; GAO, B. J.; ZHU, X.; GUO, L. Enabling fast lazy learning for data streams. In: IEEE. *Data Mining (ICDM), 2011 IEEE 11th International Conference on*. [S.l.], 2011. p. 932–941. Citado na página 29.

ZHANG, T.; RAMAKRISHNAN, R.; LIVNY, M. Birch: an efficient data clustering method for very large databases. In: ACM. *ACM Sigmod Record*. [S.l.], 1996. v. 25, p. 103–114. Citado 2 vezes nas páginas 25 e 26.

ZHOU, A.; CAO, F.; QIAN, W.; JIN, C. Tracking clusters in evolving data streams over sliding windows. *Knowledge and Information Systems*, Springer, v. 15, n. 2, p. 181–214, 2008. Citado na página 26.

ZHOU, Z.-H.; CHEN, Z.-Q. Hybrid decision tree. *Knowledge-based systems*, Elsevier, v. 15, n. 8, p. 515–528, 2002. Citado na página 46.

ZHU, S.; JI, X.; XU, W.; GONG, Y. Multi-labelled classification using maximum entropy method. In: ACM. *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*. [S.l.], 2005. p. 274–281. Citado na página 17.

ZHU, Y.; TING, K. M.; ZHOU, Z.-H. Multi-label learning with emerging new labels. *Transactions on Knowledge and Data Engineering*, IEEE, 2018. Citado 3 vezes nas páginas 17, 39 e 43.