MARIANA ALVES PEREIRA

ARCABOUÇO PARA DETECÇÃO *ONLINE* DE *OUTLIERS* PARA ALGORITMOS DE AGRUPAMENTO EM FLUXOS CONTÍNUOS DE DADOS

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*.

VIÇOSA
MINAS GERAIS – BRASIL
2017

Ficha catalográfica preparada pela Biblioteca Central da Universidade Federal de Viçosa - Câmpus Viçosa

T
P436a
2017

Pereira, Mariana Alves, 1992-
Arcabouço para detecção *online* de *outliers* para algoritmos de agrupamento em fluxos contínuos de dados / Mariana Alves Pereira. – Viçosa, MG, 2017.
xi, 50f. : il. (algumas color.) ; 29 cm.

Orientador: Murilo Coelho Naldi.
Dissertação (mestrado) - Universidade Federal de Viçosa.
Referências bibliográficas: f.45-50.

1. Detecção de *outliers. 2.* Fluxos contínuos de dados. 3. Agrupamento. 4. Componente *online*. I. Universidade Federal de Viçosa. Departamento de Ciência da Computação. Programa de Pós-graduação em Ciência da Computação. II. Título.

CDD 22 ed. 005.1

MARIANA ALVES PEREIRA

ARCABOUÇO PARA DETECÇÃO ONLINE DE OUTLIERS PARA ALGORITMOS DE AGRUPAMENTO EM FLUXOS CONTÍNUOS DE DADOS

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*.

APROVADA: 31 de julho de 2017.

Jonathan de Andrade Silva

Alcione de Paiva Oliveira

Murilo Coelho Naldi
(Orientador)

Dedico este trabalho a Deus e a minha família.

"A persistência é o caminho do êxito."
(Charles Chaplin)

Agradecimentos

Agradeço primeiramente a Deus, pela dádiva da vida.

Aos meus pais, José Luiz e Maria Regina, e ao meu irmão Luiz Otavio, pelo apoio e incentivo incondicional em todos os momentos da minha vida.

Aos meus amigos Gilberto Oliveira, Rafael Sampaio, Geraldo Pessoa e Lillian Alves pelos conselhos e ajuda ao longo desse caminho.

Aos meus professores, pelo conhecimento adquirido e apoio durante o tempo que estive em Viçosa.

A professora Elaine Paiva pela dedicação e orientação constante ao longo dessa pesquisa.

Agradeço ao meu orientador Murilo Naldi, por sempre me auxiliar na solução dos problemas encontrados e por acreditar que seríamos capazes de realizar esta pesquisa mesmo em meio a tantas dificuldades.

Agradeço ao financiamento fornecido pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), ao qual foi essencial para realização e conclusão desse trabalho.

Agradeço a Universidade Federal de Viçosa e ao Departamento de Informática por fornecerem o suporte necessário para adquirir o título de mestre e ampliar meu conhecimento.

E finalmente, a todos aqueles que, direta ou indiretamente, contribuíram para a execução desse trabalho, os meus sinceros agradecimentos.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

PEREIRA, Mariana Alves, M.Sc., Universidade Federal de Viçosa, julho de 2017. **Arcabouço para detecção *online* de *outliers* para algoritmos de agrupamento em fluxos contínuos de dados**. Orientador: Murilo Coelho Naldi. Coorientadora: Elaine Ribeiro de Faria Paiva.

Avanços da tecnologia acarretam na geração rápida e contínua de massivas quantidades de dados. Tal cenário requer a criação de algoritmos de agrupamento incrementais para extração de conhecimento. Entre as restrições impostas a esses algoritmos, os mesmos devem ser capazes de detectar e tratar possíveis *outliers* que chegam ao fluxo. O arcabouço desenvolvido nesse trabalho apresenta uma estratégia para a restrição de tratamento e detecção de *outliers* na componente *online* dos algoritmos de agrupamento de fluxo de dados. A principal contribuição da proposta em estudo é a capacidade de validar possíveis *outliers* detectados previamente, com o intuito de manter um modelo sempre atualizado e com qualidade. Para isso, todos os potenciais *outliers* são armazenados em uma memória auxiliar que de tempos em tempos é verificada, agrupando seus objetos, validando os micro-grupos formados por *inliers* e inserindo-os no modelo. Todos os objetos restantes que não foram validados, são mantidos na memória auxiliar até que se tornem válidos ou obsoletos. Em seguida, objetos obsoletos são removidos. Este trabalho também propõe o *CluStreamOD*, uma extensão do algoritmo de agrupamento *CluStream*, que aplica a estratégia em estudo em sua componente *online*, para tratar *outliers*. Os experimentos realizados mostram a eficácia do *CluStreamOD* para detecção e tratamento *online* de *outliers* do fluxo em comparação com *CluStream*, e a potencialidade da abordagem proposta para ser aplicada em outros algoritmos de fluxo de dados baseados em micro-grupos.

Abstract

PEREIRA, Mariana Alves, M.Sc., Universidade Federal de Viçosa, July, 2017. **A framework for online detection of outliers in clusters of continuous data streaming**. Advisor: Murilo Coelho Naldi. Co-advisor: Elaine Ribeiro de Faria Paiva.

Advances in technology have led to the rapid and continuous generation of massive amounts of data. Such a scenario requires the creation of incremental clustering algorithms for knowledge extraction. Among the constraints imposed on these algorithms, they must be able to detect and treat possible outliers that arrive at the flow. The framework developed in this work presents a strategy for the restriction of treatment and detection of outliers in the online component of the clustering algorithms in data stream. The main contribution of the proposal under study is the ability to validate possible outliers previously detected, in order to maintain a model that is always updated and with quality. For this, all the potential outliers are stored in an auxiliary memory when for time to time is verified, clustering its objects, validating the formed micro-clusters by inserting them into the model. All remaining objects that have not been validated are held in auxiliary memory until they become valid or obsolete. Then obsolete objects are removed. This work also proposes the CluStreamOD, an extension of the CluStream clustering algorithm, which applies the strategy under study in its component online, to treat outliers. Experiments carried out show the efficacy of the CluStreamOD for online detection and treatment of the outliers in the data streams compared to CluStream, and the potentiality of the proposed approach to be applied in other algorithms in data stream based on micro-clusters.

Capítulo 1

INTRODUÇÃO

Nos últimos anos, avanços da tecnologia permitiram que diversos dispositivos, como por exemplo, redes de sensores, celulares e computadores fossem mais amplamente disseminados e utilizados para as mais diversas tarefas. Tais avanços acarretam na geração rápida e contínua de massivas quantidades de dados, denominados Fluxos Contínuos de Dados (do inglês, *data stream*) [Nguyen et al., 2015].

Atualmente, várias aplicações do mundo real geram FCDs. Dentre elas, pode-se citar: meteorologia, fluxos de dados comerciais, monitoramento em tempo real das atividades de um usuário por meio de celulares e redes sociais, entre outras [Muthukrishnan et al., 2005]. Analisar tais fluxos pode por exemplo, auxiliar nos estudos de como uma doença podem evoluir ao longo do tempo [Zhang et al., 2017]; ou mesmo descobrir tendências de mercado e explorar os diferentes perfis de usuários [Miglierina & Di Nitto, 2017],[Nguyen & Jung, 2017]. Também é possível inferir possíveis fraudes por meio da análise de transações bancárias de rotina executadas por um usuário [Mohammed et al., 2017].

Segundo Chen et al. [1996] o processo de extração não trivial de informações úteis e implícitas a partir dos dados é denominado Mineração de Dados (MD). Diversos algoritmos tradicionais de extração de informação aplicados a conjuntos de dados estáticos surgiram ao longo dos anos. Algumas das principais tarefas tratadas por eles são: agrupamento de dados, classificação, predição de padrões, entre outras [Tan et al., 2009]. De forma similiar, problemas envolvendo FCDs também requerem busca de informações úteis e sendo assim, a MD ganha destaque nesse cenário [Gama, 2010]. Entretanto, algoritmos tradicionais de MD não podem ser facilmente aplicados ao cenário de FCDs, pois faz-se necessário a implementação das seguintes restrições [Gama, 2010]:

- O algoritmo deve ser capaz de descobrir grupos com formatos diferentes ao longo do fluxo;
- Os objetos que chegam ao fluxo não são estacionários, ou seja, a distribuição de probabilidade pode mudar ao longo do tempo;
- Novos grupos podem surgir e grupos antigos ou obsoletos podem desaparecer, pois não representam mais o modelo do fluxo naquele dado momento;

- O fluxo pode ser considerado infinito e, sendo assim, não é possível armazenar todos os objetos em memória;
- Os objetos chegam de forma contínua e em muitos casos com alta velocidade, logo, o algoritmo deve ser capaz de analisá-los de imediato;
- Objetos *outliers* podem surgir ao longo do fluxo e devem ser detectados e tratados;
- A definição dos parâmetros de entrada é importante para a efetividade do algoritmo.

Uma das tarefas mais difundidas em MD, é o agrupamento de dados [Tan et al., 2009]. Essa técnica permite dividir os objetos do conjunto em grupos, sem a necessidade de um conhecimento prévio dos mesmos, ou seja, de forma não supervisionada [Jain & Dubes, 1988]. No cenário de FCDs, o agrupamento de dados também possui grande importância, sendo que vários algoritmos foram desenvolvidos para esta tarefa, como por exemplo, *CluStream* [Aggarwal et al., 2003], *DenStream* [Cao et al., 2006], *D-Stream* [Chen & Tu, 2007], *Stream LS* e *Birch-LS* [Guha et al., 2003].

Mesmo que muitos algoritmos tenham sido desenvolvidos para a tarefa de agrupamento em FCDs, a capacidade agrupar os dados e detectar *outliers* ao longo do FCDs ainda é uma questão a ser tratada. Considerando-se a tarefa de agrupamento, *outliers* são objetos com características discrepantes quando comparados aos demais objetos absorvidos pelos grupos que representam o modelo [Aggarwal, 2015].

Devido a natureza dinâmica dos FCDs, novos grupos podem surgir ao longo do tempo e para que o modelo que os representa se mantenha atualizado, grupos que não mais representam o fluxo na janela de tempo atual devem ser removidos. Esse processo de manutenção do modelo pode se tornar ainda mais complexo quando há a presença de *outliers* no fluxo e o algoritmo de agrupamento selecionado não apresenta nenhuma estratégia para detecção e tratamento de *outliers* [Cao et al., 2006].

Seguindo a linha de conhecimento descrita, o presente trabalho desenvolveu um arcabouço para detecção de *outliers* que pode ser aplicado na componente *online* de algoritmos de agrupamento em FCDs, com o objetivo de melhorar o seu desempenho. Nessa abordagem, potenciais *outliers*, ou seja, objetos que tendem a depreciar a qualidade do modelo são inseridos em uma memória auxiliar para futuras análises. Periodicamente, micro-grupos válidos obtidos a partir desta memória são inseridos no modelo. Dessa forma, essa abordagem tenta garantir que apenas inclusões seguras serão feitas no modelo, tanto no momento em que o objeto chega ao fluxo ou após o processo de validação da memória auxiliar, evitando assim a mitigação dos *outliers*. Inicialmente, o método foi incorporado ao *CluStream* [Aggarwal et al., 2003], com o intuito de verificar a eficiência da abordagem proposta, resultando no algoritmo *CluStreamOD*. No entanto, essa abordagem pode ser aplicada a outros algoritmos de FCDs baseados em micro-grupos.

1.1 Objetivos

O objetivo geral deste trabalho foi desenvolver um arcabouço para detecção de *oultiers* que pode ser aplicado na componente *online* de algoritmos de agrupamento em FCDs que se baseia em micro-grupos. Para possibilitar esse estudo, quatro passos foram seguidos e investigados:

- Criação de uma estratégia que possa identificar possíveis elementos *outliers* que chegam ao fluxo e os armazene em uma memória auxiliar, para validações futuras.
- Criação de um método que verifique a memória auxiliar na busca de micro-grupos válidos que devem ser inseridos no modelo.
- Desenvolvimento do algoritmo *CluStreamOD*, uma extensão do algoritmo *CluStream* [Aggarwal et al., 2003], que incorpora em sua componente *online*, o arcabouço desenvolvido nesse trabalho.
- Estudo dos parâmetros de entrada, buscando utilizar uma quantidade mínima desses, quando comparados aos demais algoritmos de agrupamento que realizam detecção de *outliers* na componente *online*.

1.2 Artigos Originados da Dissertação

O artigo publicado é:

Pereira, M. A., Paiva, E. R. F.; Naldi, M. C. Detecção online de outliers em agrupamento de fluxos contínuos de dados. Proceedings of the 4th Sysposium on Knowledge Discovery, Mining and Learning, Recife, p. 124-130, 2016.

O artigo aceito é:

Pereira, M. A., Paiva, E. R. F.; Naldi, M. C. Online detection of outliers in clusters of continuous data streaming. Brazilian Conference on Intelligent Systems.

1.3 Organização da Dissertação

O presente trabalho está organizado da seguinte maneira:

Capítulo 2: É apresentada uma revisão bibliográfica acerca do problema em estudo.

Capítulo 3: Os materiais e métodos utilizados para alcançar o objetivo desse trabalho são apresentados.

Capítulo 4: Resultados e discussões gerados pela aplicação da estratégia em estudo comparadas aos demais algoritmos selecionados.

Capítulo 5: Apresenta a conclusão desse trabalho e possíveis trabalhos futuros a serem desenvolvidos.

Capítulo 2

REVISÃO BIBLIOGRÁFICA

Este capítulo apresenta uma revisão bibliográfica realizada sobre Fluxos Contínuos de Dados e alguns principais métodos de agrupamento para FCDs. A revisão também apresenta estratégias para detecção de *outliers* aplicadas a algoritmos de agrupamento.

2.1 Fluxos Contínuos de Dados (FCDs)

Fluxos Contínuos de Dados são sequências de objetos, possivelmente infinitas, que chegam de forma contínua, em geral com alta velocidade. Adicionalmente, a distribuição de probabilidade que gera os dados pode mudar, fazendo com que objetos de novas categorias cheguem ao fluxo [Gama, 2010].

Definição 1 *Fluxos Contínuos de Dados (FCDs) são sequências ordenadas de objetos* $\chi = \mathbf{o_1}, \mathbf{o_2}, \mathbf{o_3}, ...,$ *onde* $\mathbf{o_i}$ *é um vetor que armazena um objeto de d dimensões, sendo que cada objeto tem um marcador de tempo t associado. Os objetos do fluxo são analisados em ordem crescente do índice i, que são potencialmente ilimitados e que podem evoluir ao longo do tempo [Silva et al., 2013].*

As características inerentes aos FCDs fazem com que algoritmos desenvolvidos nesse cenário sejam capazes de analisar os dados e montar um modelo que os representa, sem a necessidade de manter todos os objetos em memória. Adicionalmente, para que esse modelo mantenha-se consistente e válido ao longo do tempo, é importante que os mesmos considerem a ordem pré-determinada pelo fluxo [Gama et al., 2011].

Além das características previamente citadas, outras características relevantes as quais algoritmos aplicados a FCDs devem se preocupar são: i) a detecção de ruídos e *outliers*; ii) a velocidade na qual os objetos chegam ao fluxo; iii) o surgimento de novos grupos e o esquecimento dos obsoletos, iv) a exibição do modelo gerado continuamente, sem tornar esse processo custoso [Aggarwal, 2003]. Logo, considerar todas essas características em um mesmo algoritmo faz com que o processo de aprendizado em FCDs seja desafiador [Gaber et al., 2010].

2.2 Mineração de Dados e Fluxos Contínuos de Dados

Algoritmos desenvolvidos para a tarefa de Mineração de Dados (MD) ganharam destaque ao longo dos anos devido à capacidade de extração de padrões de interesse em um conjunto finito de objetos. Tais padrões são capazes de inferir o comportamento de futuros objetos desconhecidos pelo modelo [Tan et al., 2009]. O tipo de análise mais comum é conhecido como *batch*, ou seja, resolve problemas de aprendizado não incrementais. Dentre os algoritmos desenvolvidos para cenários em *batch* temos: k-médias [Lloyd, 1982], [MacQueen et al., 1967], DBSCAN [Ester et al., 1996], SVM [Steinwart & Christmann, 2008], árvores de decisão [Safavian & Landgrebe, 1991].

Contudo, ao analisar o mundo real e o processo de geração de dados, percebe-se que os mesmos possuem uma natureza contínua e dinâmica. Tais características mostram que, algoritmos tradicionais de MD, ou seja, desenvolvidos para cenários em *batch*, não podem ser facilmente convertidos e aplicados ao cenário descrito devido a natureza incremental dos dados [Gama, 2010].

Ao se considerar todas as características inerentes ao FCDs previamente apresentadas e realizando uma análise acerca dos algoritmos tradicionais de MD, conclui-se que a aplicação de algumas mudanças nos mesmos não são suficientes para lidar com FCDs. Por esse motivo, foram desenvolvidos algoritmos exclusivos para as tarefas de classificação e regressão em FCDs [Hulten et al., 2001], [Oza & Russell, 2001], [Pfahringer et al., 2007], [Bifet et al., 2009] e a tarefa de agrupamento em FCDs [Domingos & Hulten, 2000], [Aggarwal et al., 2003], [Chang & Lee, 2005], [Cao et al., 2006].

2.3 Agrupamento de Dados em FCDs

A MD tradicional apresenta algoritmos desenvolvidos para tarefas de aprendizado como agrupamento (do Inglês *clustering*), classificação, predição, regras de associação, entre outras [Tan et al., 2009]. Tais tarefas também foram exploradas no domínio de FCDs.

O agrupamento de dados é uma das tarefas mais difundidas da MD, tendo como exemplo algoritmos tradicionais importantes, como por exemplo: k-médias [MacQueen et al., 1967; Lloyd, 1982], [Jain, 2010], DBSCAN [Ester et al., 1996], PAM [Kaufman & Rousseeuw, 2009] e COBWEB [Fisher, 1987]. Uma das possíveis razões de tal sucesso é que o conceito de grupos nessa tarefa não é pré-determinado, ou seja, tais algoritmos podem ser aplicados a diversos conjuntos de dados com distribuições de probabilidades diferentes [Kaufman & Rousseeuw, 2009], [Tan et al., 2009].

O agrupamento é obtido pelo processo de análise e subdivisão dos objetos em grupos distintos, considerando a similaridade existente entre os objetos. Sendo assim, espera-se que os objetos de um mesmo grupo apresentem alta similaridade, quando comparado aos objetos pertencentes aos demais grupos [Jain & Dubes, 1988].

A subdivisão dos objetos em grupos, com relação a tarefa de agrupamento, em geral é realizada de forma não supervisionada. Isso implica que não é necessário conhecimento externo com relação ao comportamento e categoria de cada objeto por parte do algoritmo [Gonçalves et al., 2008].

Segundo Aggarwal [2003], considerando-se a tarefa de agrupamento em FCDs, uma sub-área de MD contida no Aprendizado de Máquina (AM), o maior problema enfrentado é a capacidade de manter um modelo coerente com o fluxo recebido. Ou seja, os grupos gerados devem ser capazes de acompanhar a dinamicidade do fluxo por meio de remodelagem contínua do mesmo, sempre que novos objetos sejam recebidos e analisados. Tal processo descrito anteriormente induz a necessidade de criação de algoritmos incrementais e uso de estratégias menos custosas para atualização contínua do modelo.

No domínio de FCDs, os grupos presentes no FCD devem respeitar a ordem e tempo de chegada de seus objetos e o tamanho da memória. Dado que um FCD é possivelmente ilimitado, armazenar todos os objetos que chegam para agrupá-los posteriormente é inviável. Também, é importante que os objetos sejam analisados no momento de sua chegada, evitando a inclusão de objetos *outliers* nas estruturas de sumarização consideradas pelo modelo. O algoritmo também deve ser capaz de remover grupos obsoletos do modelo [Silva et al., 2013]. Todas essas características não podem ser tratadas pelos algoritmos de agrupamento tradicionais.

Aggarwal [2003] considera que o processo de agrupamento em FCDs é desafiador, pois faz-se necessário a busca de estratégias incrementais, como por exemplo micro-grupos (ver Subseção 2.3.1), que são capazes de considerar uma quantidade limitada de informação acerca dos objetos, porém suficiente para representar o modelo. Cao et al. [2006] acreditam que uma atenção especial também deve ser dada a objetos ruidosos e *outliers* presentes no fluxo (ver Seção 2.4). Os mesmos devem ser detectados e tratados, pois a inserção desses no modelo culminam em uma depreciação do mesmo ao longo do tempo. Adicionalmente, é importante diferenciar objetos novidades (que darão início a um novo grupo) de objetos *outliers*.

Dadas as restrições dos FCDs previamente apresentadas nesse capítulo, percebemos que uma simples varredura sobre os dados não é capaz de demonstrar com efetividade o desenvolvimento do fluxo. Isso porque, muitas características relevantes podem ser ignoradas, como por exemplo: i) objetos *outliers* que poderiam ser inseridos nos grupos do modelo, ii) o surgimento de novos grupos, iii) o desaparecimento de grupos antigos. Tais resultados implicam em um modelo de baixa qualidade e que não permite ao usuário acompanhar o comportamento do fluxo como um todo [Gaber et al., 2010].

Muitos algoritmos foram desenvolvidos para o cenário de agrupamento em FCDs. Entre os mais relevantes e precursores dessa área podemos citar: CluStream [Aggarwal, 2003], DenStream [Cao et al., 2006], StreamKM++ [Ackermann et al., 2012], D-Stream [Chen & Tu, 2007] e ClusTree [Kranen et al., 2011]. Uma descrição detalhada sobre os algoritmos de agrupamento mais relevantes para esse trabalho será apresentada na Seção 2.5.

2.3.1 Vetor de Características (*CF-Vector*)

Muitos dos algoritmos de agrupamento em FCDs utilizam a estrutura sumária de dados chamada vetor de características (do Inglês, *Cluster Feature Vector - CF-Vector*), ou micro-grupo, para armazenar as principais informações referentes aos objetos do FCDs. Sendo assim, um micro-grupo permite representar os dados de forma sumária. Assim, não é preciso armazenar todos os objetos Zhang et al. [1997].

Um micro-grupo pode ser definido como MG=(N, $\vec{LS}$, $\vec{SS}$), onde $\vec{LS}$ e $\vec{SS}$ são vetores d-dimensionais. Os objetos de um micro-grupo são sumarizados mantendo:

- *N:* número de objetos presentes no micro-grupo;
- $\vec{LS}$: soma linear dos N objetos do micro-grupo;
- $\vec{SS}$: soma dos quadrados dos valores dos N objetos do micro-grupo.

As propriedades de um micro-grupo são [Gama, 2010]:

- Aditividade: Sejam *MG1* e *MG2* dois micro-grupos disjuntos, então o micro-grupo da união de *MG1* e *MG2* é dado por:

$$MG1 + MG2 = (N_1 + N_2 + L\vec{S}_1 + L\vec{S}_2 + S\vec{S}_1 + S\vec{S}_2)$$

- Incrementabilidade: Se um objeto $\mathbf{o_t}$ é adicionado ao micro-grupo, o mesmo deve ser atualizado de forma que:

$$\begin{gathered} N = N + 1 \\ \vec{LS} = \vec{LS} + \mathbf{o_t} \\ \vec{SS} = \vec{SS} + \mathbf{o_t}^2 \end{gathered}$$

Adicionalmente, dado um micro-grupo MG, o raio e o centróide desse micro-grupo podem ser calculados por [Zhang et al., 1997]:

- Centróide:

$$\vec{MG}0 = \frac{\sum_{i=1}^{N} \vec{MG}_i}{N} \qquad \vec{MG}0 = \frac{\vec{LS}}{N} \tag{2.1}$$

- Raio:

$$R = \left(\frac{\sum_{i=1}^{N} \left(\vec{MG}_i - \vec{MG}0 \right)^2}{N} \right)^{\frac{1}{2}} \qquad R = \left(\frac{\vec{SS}}{N} - \left(\frac{\vec{LS}}{N} \right)^2 \right)^{\frac{1}{2}} \tag{2.2}$$

Outras estratégias de sumarização dos objetos do fluxo podem ser aplicadas ao problema de agrupamento em FCDs, porém não foram utilizadas nesse trabalho devido ao mesmo apresentar inicialmente uma extensão do algoritmo CluStream [Aggarwal, 2003]. Uma descrição detalhada das demais estratégias pode ser encontrada em [Silva et al., 2013].

2.3.2 Janelas de Tempo em FCDs

Lidar com FCDs remete à necessidade de acompanhar a chegada de novos objetos ao longo do tempo, preservando a evolução do mesmo. Para que isso seja possível, o modelo deve ser estruturado de forma que os novos objetos do fluxo se tornem mais relevantes que os objetos mais antigos [Babcock et al., 2002], [Chen & Tu, 2007], [Gama, 2010].

Com relação a FCDs, a problemática acima descrita pode ser resolvida com a aplicação de janelas de tempo. Na literatura, três modelos de janelas de tempo foram comumente estudados e aplicados em algoritmos de agrupamento em FCDs. As mesmas são: i) janela deslizante (do inglês *sliding windows*); ii) janela amortizada (do inglês *damped windows*); e iii) janela de referência (do inglês *landmark windows*) [Silva et al., 2013].

Em uma janela deslizante, apenas as informações acerca dos objetos mais recentes que chegam ao FCDs são armazenadas. As mesmas podem ser armazenadas em estruturas de tamanho fixo ou variável. Em geral, essa estrutura é do tipo FIFO (primeiro que entra é o primeiro que sai, do inglês, *first in, first out*), que considera objetos do período de tempo atual até um certo período de tempo do passado. A organização e manipulação dos objetos são de acordo com os princípios de uma fila. Alguns dos algoritmos que utilizam essa estrutura são [Babcock et al., 2003], [Chi et al., 2006], [Liu et al., 2009], [Ren & Ma, 2009]. Na Figura 2.1 foi apresentado um exemplo de janela deslizante.

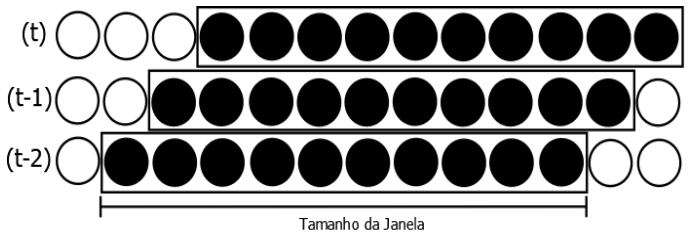


Figura 2.1: Exemplo de janela deslizante - adaptado de [Silva et al., 2013].

Na janela amortizada, também conhecida como modelo de desvanecimento, são atribuídos pesos aos objetos que chegam ao FCDs. Objetos mais recentes recebem pesos maiores que objetos mais antigos, e o peso dos objetos decrementa com o passar do tempo. Essa estrutura foi aplicada em algoritmos como: [Giannella et al., 2003], [Chang & Lee, 2004], [Cao et al., 2006], [Chen & Tu, 2007], [Isaksson et al., 2012]. A Figura 2.2 apresenta um exemplo de janela amortizada, onde o peso dos objetos decai exponencialmente dos mais recentes (preto) para os esquecidos (branco).

Janela de referência requerem a manipulação de porções disjuntas do FCDs, dadas como *chunks*, separadas por pontos de referência. Pontos de referência podem ser definidos em termos de tempo ou na quantidade de objetos observados desde o último ponto de referência. Todos os objetos que chegam ao FCDs após um ponto de referência são mantidos ou sumarizados na janela de tempo atual. Quando um novo ponto de referência é alcançado, todos os objetos analisados na janela são removidos e os novos objetos são guardados. Tal estrutura pode ser encontrada nos algoritmos

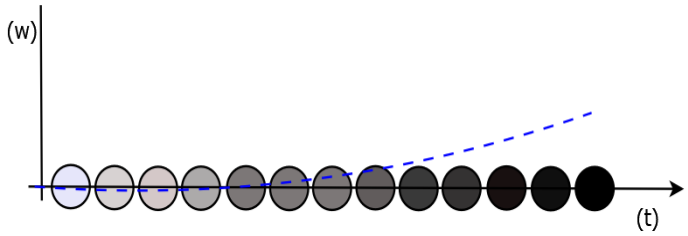


Figura 2.2: Exemplo de janela amortizada - adaptado de [Silva et al., 2013].

[O'callaghan et al., 2002], [Manku & Motwani, 2002], [Aggarwal et al., 2003], [Liu et al., 2009], [Ackermann et al., 2012]. A Figura 2.3 apresenta um exemplo de janela de referência.

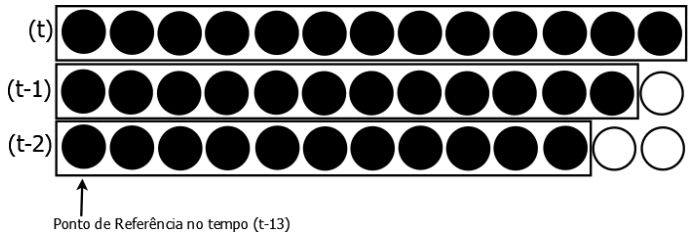


Figura 2.3: Exemplo de janela de referência para um intervalo de tempo de tamanho 13 - adaptado de [Silva et al., 2013].

Definir um tamanho de janela fixo é desafiador. Pois tamanhos pequenos são capazes de descobrir mudanças de conceitos rapidamente, mas afetam o desempenho de aprendizado do algoritmo. Em janelas maiores, o aprendizado não é afetado frequentemente como ocorre em janela pesquenas, mas mudanças de conceito podem não ser rapidamente detectadas [Gama et al., 2004]. Devido a natureza do algoritmo utilizado no arcabouço proposto neste trabalho, ser uma extensão do algoritmo *CluStream* [Aggarwal et al., 2003], a janela de referência foi adotada nesse trabalho. Uma descrição detalhada de cada uma dessas janelas pode ser obtida em [Silva et al., 2013].

2.3.3 Algoritmos de Agrupamento em FCDs baseados em objetos

Em sua maioria, algoritmos de agrupamento em FCDs baseados em objetos requerem que o processo de agrupamento dos mesmos seja realizado por duas componentes distintas: a componente *online* (também conhecida como micro-agrupamento ou abstração dos dados) e a componente *offline* (também conhecida como macro-agrupamento ou agrupamento), conforme apresentado na Figura 2.4. [Silva et al., 2013].

A componente *online* é responsável por sumarizar os objetos que chegam ao fluxo com o apoio de estratégias adicionais capazes de absorver as principais características dos objetos do fluxo, como por exemplo, os micro-grupos (apresentados na Subseção 2.3.1). Tais estratégias são capazes de manter um conhecimento sobre

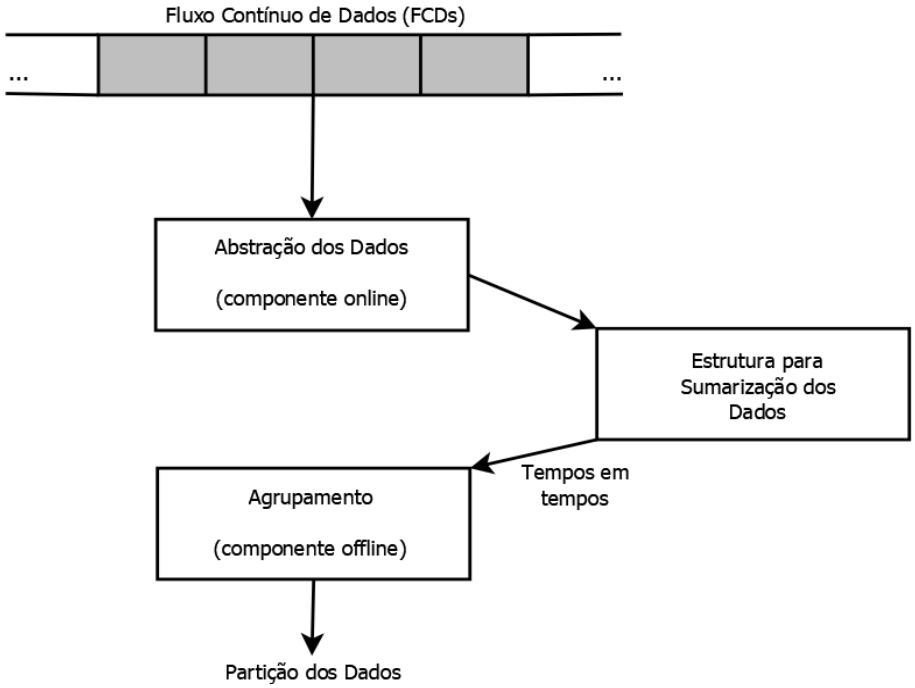


Figura 2.4: Componentes *online* e *offline* aplicadas ao processo de agrupamento em FCDs - adaptado de [Silva et al., 2013].

os objetos originais, sem a necessidade de armazená-los por completo em memória, lidando com a restrição de espaço e memória imposta pelos FCDs [Aggarwal, 2003].

Além de sumarizar os objetos do fluxo, é importante que esse processo seja continuamente executado e principalmente, que os novos objetos sejam diferenciados dos antigos ao longo do tempo. O uso de estruturas como as janelas de tempo (apresentadas na Subseção 2.3.2) são então aplicadas gerenciar que o modelo seja atualizado incrementalmente. Dessa forma, mudanças no FCDs podem ser acompanhadas e analisadas pelo usuário ao longo do tempo e demonstram que o modelo gerado se mantém constantemente atualizado e consistente [Silva et al., 2013].

Além das restrições apresentadas anteriormente, ainda na componente *online*, espera-se que os algoritmos desenvolvidos para um cenário de FCDs sejam capazes de detectar e tratar objetos *outliers*. Porém, é importante que o mesmo seja capaz de diferenciar uma evolução presente no fluxo (por exemplo, o surgimento de um novo grupo) desses objetos. Tais medidas garantem que o modelo gerado seja consistente ao longo do tempo [Cao et al., 2006]. Uma estratégia adotada por Cao et al. [2006] e [Paiva, 2014] para tratar a restrição previamente descrita é o uso de uma memória auxiliar para armazenar os objetos considerados possíveis *outliers*. Nesse trabalho, o arcabouço desenvolvido também utiliza-se de uma memória auxiliar para armazenamento de possíveis *outliers*.

O processo de relevância de um objeto na componente *online* é dado então pela junção da informação referente ao tempo de chegada t de um objeto no fluxo e o tamanho de janela h definido para o algoritmo (usualmente dado pelo usuário). Considerando-se a estrutura de micro-grupos (apresentada na Subseção 2.3.1) e um

tempo de janela h definido, se um micro-grupo está no modelo pelo menos a h tempos sem incorporar novos objetos (ver Definição Incrementabilidade na Subseção 2.3.1) o mesmo é dado como obsoleto e deve ser removido. Entretanto, considerando o valor de h definido, os micro-grupos do modelo que foram capazes de absorver novos objetos do fluxo, devem se manter relevantes e não devem ser removidos [Aggarwal et al., 2003].

Com relação a componente *offline*, a mesma é responsável por agrupar os sumários estatísticos obtidos na componente *online* por meio de algoritmos de agrupamento como por exemplo k-médias [MacQueen et al., 1967], [Lloyd, 1982] e DBSCAN [Ester et al., 1996], sempre que solicitado pelo usuário. Dessa forma, uma visão geral geral do agrupamento do Fluxo Contínuo de Dados é apresentada ao usuário para uma dada janela de tempo h. Como entrada para essa componente, serão considerados os sumários estatísticos em conjunto com demais parâmetros de entrada definidos pelo usuário, como por exemplo, número de grupos, tamanho da janela de tempo, entre outros [Aggarwal, 2003].

2.4 Detecção de *Outliers* em FCDs

Entre as restrições impostas ao desenvolvimento de um algoritmo de agrupamento em FCDs, pode-se citar a necessidade de detectar e tratar *outliers* presentes no fluxo [Cao et al., 2006], [Aggarwal, 2015]. *Outliers* são objetos com características discrepantes quando comparados aos demais objetos que representam o modelo do FCDs. Adicionalmente, esses objetos são capazes de interferir negativamente na qualidade do modelo apresentado ao usuário final [Gama, 2010], [Aggarwal, 2015].

Na maioria dos algoritmos de agrupamento em FCDs que realizam o processo de detecção de *outliers*, os objetos classificados como tal devem ser removidos do fluxo e não devem pertencer ao modelo. Essa estratégia garante que o modelo dos dados gerado se mantenha continuamente consistente e válido, sem a perda de informações relevantes sob influência de objetos *outliers* [Cao et al., 2006], [Chen & Tu, 2007].

Objetos que representam padrões novidades presentes em um FCDs podem ser considerados à priori como *outliers* por se enquadrarem na definição previamente citada no que se refere a caracterização de um objeto *outlier*, como por exemplo, o surgimento de um novo grupo. Porém, tais padrões são importantes e demonstram possíveis evoluções do fluxo, logo não devem ser descartados. Sendo assim, os algoritmos de agrupamento em FCDs precisam detectar os reais *outliers* e serem capazes de perceber a diferença entre esse real *outlier* e um padrão novidade que está surgindo no fluxo [Silva et al., 2013].

No algoritmo *CluStream* [Aggarwal et al., 2003], uma quantidade fixa de micro-grupos é mantida pelo modelo. Tal abordagem é arriscada considerando-se FCDs que contém *outliers*. Pois, muitos novos micro-grupos serão criados para os *outliers*, enquanto que muitos micro-grupos existentes serão excluídos ou unidos. Idealmente, o algoritmo de agrupamento deve possuir alguma estratégia capaz de distinguir entre *outliers* e padrões novidades [Cao et al., 2006].

Algumas das estratégicas aplicada para realizar o processo de detecção de *outliers* em FCDs são o uso de estruturas auxiliares, como por exemplo: i) uso de uma memória auxiliar para armazenar e tratar possíveis *outliers* [Cao et al., 2006], [Paiva, 2014], ii) uso de grades para analisar as densidades dos objetos do fluxo e então detectar os reais *outliers* [Chen & Tu, 2007], iii) utilização de árvores para armazenar os objetos do fluxo, de forma que os objetos com menor densidade de concentrem nas folhas [Zhang et al., 1997].

O processo de detecção de *outliers* em FCDs é considerado desafiador, pois conforme discutido previamente, faz-se necessário aplicar um conjunto de estruturas capazes de diferenciar e tratar os reais *outliers* do fluxo continuamente (na componente *online*), sem depreciar a eficiência do processamento dos demais objetos do FCDs [Silva et al., 2013]. Os principais algoritmos considerados para esse cenário requerem para o funcionamento interno das estruturas aplicadas, diversos parâmetros de entrada. Sabe-se a dificuldade existente em adaptar tais parâmetros, de forma a obter a melhor acurácia por parte do algoritmo para os FCDs analisados [Gama et al., 2011].

Uma descrição detalhada sobre os algoritmos para FCDs de interesse desse trabalho é apresentada na Seção 2.5.

2.5 Detecção de *outliers* no agrupamento de FCDs

Em geral, o agrupamento de FCDs tem duas componentes que compõem sua estrutura básica: i) componente *online*, que sumariza os objetos, e ii) componente *offline*, que utiliza o sumário dos objetos em conjunto com outras entradas do usuários para fornecer o agrupamento do modelo quando solicitado pelo usuário [Silva et al., 2013]. Uma das estruturas mais utilizadas para sumarizar objetos é chamada de micro-grupo (ver Subseção 2.3.1).

O *CluStream* Aggarwal et al. [2003] é um dos algoritmos precursores na área de agrupamento em FCDs. Sua componente *online* mantém uma quantidade fixa de micro-grupos na memória e para cada novo objeto que chega ao FCDs é calculada a distância deste ao micro-grupo mais próximo do modelo. Caso sua distância esteja dentro do limite máximo desse micro-grupo, o objeto é inserido no mesmo. Caso contrário, um novo micro-grupo é criado para absorver o novo objeto. Se a quantidade limite de micro-grupos estiver em seu máximo, a criação de um novo micro-grupo resulta na necessidade da exclusão do mais antigo ou, caso não seja possível, na união dos dois micro-grupos mais próximos. Para a criação dos micro-grupos iniciais, uma versão adaptada do algoritmo k-médias é aplicada sobre os primeiros objetos do FCDs. Na componente *offline*, um algoritmo de agrupamento, como por exemplo o k-médias, é executado sobre os micro-grupos do modelo para encontrar os grupos. O único tratamento de *outliers* do algoritmo *CluStream* ocorre no momento da exclusão de um micro-grupo antigo, onde o mesmo considera que um micro-grupo formado por um objeto *outlier* não absorve novos objetos do FCDs ao longo do tempo e portanto torna-se obsoleto.

Outros algoritmos também utilizam-se da estrutura de micro-grupos e possuem rotinas para tratamento de *outliers*. O *DenStream* Cao et al. [2006] é um algoritmo baseado em densidade que trata *outliers*. Para isso, a componente *online* utiliza dois *buffers* denominados potenciais micro-grupos *(p-micro-grupos)* e *outliers* micro-grupos *(o-micro-grupos)*. Quando um novo objeto não consegue ser alocado no *p-micro-grupos*, ele é armazenado em *o-micro-grupos* até que seja promovido ou removido, que de fato o caracteriza como um real *outlier*. Periodicamente uma análise sobre os *buffers* é realizada, buscando micro-grupos que devem ser promovidos a *p-micro-grupos* ou removidos dos *buffers*. Essa análise depende de parâmetros de entrada fornecidos pelo usuário, os quais são difíceis de serem identificados, especialmente em conjuntos de dados reais.

De maneira semelhante ao algoritmo *DenStream* [Cao et al., 2006], o algoritmo *ClusTree* [Kranen et al., 2011] também propõe o uso de micro-grupos ponderados, os quais são mantidos em uma árvore hierárquica. Dois parâmetros de entrada são necessários para a construção desta árvore: o número de entradas em um nó folha e o número de entrada em nós não-folhas. *ClusTree* fornece estratégias para lidar com restrições de tempo para a tarefa de agrupamento em qualquer momento, ou seja, possibilita a interrupção na inserção de novos objetos na árvore sempre que necessário. Nesse algoritmo, não há um pressuposto a respeito do tamanho do modelo do agrupamento, já que suas operações de agregação e divisão ajustam automaticamente o tamanho do mesmo. Os objetos que não foram inseridos na árvore devido a uma interrupção, são armazenados em um *buffer* que posteriormente é inserido na árvore. Esse algoritmo se adapta bem a FCDs lentos e rápidos, de forma que em fluxos rápidos o mesmo agrupa objetos semelhantes para acelerar o processo de inserção e em fluxos lentos, o tempo ocioso é usado para melhorar a qualidade do agrupamento. Entretanto, esse algoritmo não apresenta nenhum tipo de estratégia para detecção e tratamento de *outliers* que chegam ao FCDs, ou seja, todos os objetos são inseridos na árvore.

D-Stream Chen & Tu [2007] é um algoritmo de agrupamento baseado na densidade da grade. Possui a estrutura básica dos algoritmos de agrupamento, de forma que na componente *online* cada objeto de entrada é mapeado em uma grade correspondente. Na componente *offline* são computadas as densidades de cada grade e as mesmas são agrupadas em relação a esses valores. Uma função de desvanecimento é utilizada para diminuir a densidade das grades com o tempo, se ela estiver abaixo de limiar definido pelo usuário e nenhum objeto foi adicionado desde a última verificação da densidade da grade, a mesma é descartada. O tratamento de *outliers* nesse algoritmo é realizado periodicamente por meio da remoção das grades que permaneceram marcadas como esporádicas, ou seja, possuem poucos objetos ou se tornaram muito antigas em relação ao *timestamp* atual, desde a última verificação. Caso essa verificação siga os parâmetros definidos pelos autores, o processo de análise das mesmas pode tornar-se computacionalmente custoso.

É importante notar que a definição de densidade (e, portanto, *outlier*) nos algoritmos *DenStream* e *D-Stream* é baseada em parâmetros definidos pelo usuário, para os quais os valores ótimos podem ser difíceis de estimar, especialmente quando associados a conjuntos de dados reais ou FCDs com variabilidade da densidade.

Além disso, os algoritmos baseados em densidade são computacionalmente mais onerosos quando comparados ao agrupamento baseado em k-médias.

O algoritmo MINAS [Paiva, 2014] também apresenta a estrutura de subdivisão em duas componentes *online* e *offline*. Na componente *online*, um modelo de decisão é criado com base nos conceitos conhecidos advindos de um conjunto de dados rotulado com uma ou mais classes. Para representar as classes do problema em estudo, as mesmas são representadas por conjuntos de micro-grupos. Cada novo objeto não rotulado é classificado de acordo com o modelo de decisão previamente criado. Objetos que não são explicados pelo modelo de decisão são marcados com o perfil *desconhecido* e armazenados em uma memória auxiliar. De tempos em tempos, os objetos dessa memória são agrupados, gerando novos micro-grupos. Todos os micro-grupos são validados e apenas os coesos e representativos são adicionados no modelo de decisão. O algoritmo MINAS também apresenta um mecanismo de verificação dos micro-grupos obsoletos e remoção desses do modelo de decisão e uma estratégia para limpeza da memória auxiliar, a fim de eliminar os objetos *desconhecidos* que não foram utilizados para criação de novos micro-grupos. A componente *offline* é supervisionada e executada apenas uma vez. Porém, este algoritmo foi desenvolvido para a tarefa de classificação em FCDs, onde um conhecimento externo sobre os objetos se faz necessário.

Alguns exemplos de algoritmos específicos para detecção de *outliers* em FCDs propostos na literatura são: *iLOF* Salehi et al. [2015], *CORM* Elahi et al. [2008], *Abstract-C* Yang et al. [2009], *STORM* Angiulli & Fassetti [2010], *MCOD* Kontaki et al. [2011]. Em geral, esses empregam um critério para determinar se um objeto é um *outlier* baseado na distância Aggarwal [2015]. Considerando que os objetos pertencem a um mesmo espaço, se existem menos que *minPts* objetos na vizinhança do objeto, o mesmo é denominado *outlier*. Entretanto, esses algoritmos não são empregados em conjunto com um algoritmo de agrupamento em FCDs, conforme é realizado na proposta em estudo.

Entre os algoritmos específicos para detecção de *outliers* em FCDs propostos na literatura, o MCOD [Kontaki et al., 2011] baseia-se na estrutura de micro-grupos evolutivos para detectar *outliers*. A mesma é utilizada para suprir a deficiência apresentada pelos demais algoritmos quanto à comparação do novo objeto aos demais objetos ativos. Um novo objeto do FCDs é inserido em um micro-grupo se está dentro do raio do micro-grupo mais próximo. Caso contrário, verifica-se na lista *PD* se existe pelo menos uma quantidade mínima de vizinhos, definida pelo usuário, para o novo objeto. Caso exista, um novo micro-grupo é formado. Caso contrário, o objeto é inserido em *PD* e na fila de eventos se for um *inliers* inseguros, ou seja, que não foi absorvido por nenhum micro-grupo. Quando a janela de tempo desliza, objetos expirados são removidos dos micro-grupos e de *PD*. Após essa verificação, a fila de eventos é atualizada analisando se lista de vizinhos dos *inliers* inseguros foi alterada. Após uma nova janela e o processo de verificação ser aplicado, todos os objetos em *PD* que possuirem menos que uma quantidade mínima de vizinhos são reportados como *outliers*. Entretanto, nesse trabalho a componente *online* possui a finalidade exclusiva de detectar objetos *outliers* e os micro-grupos são adaptados totalmente a essa componente.

Capítulo 3

MATERIAIS E MÉTODOS

Este capítulo apresenta de forma detalhada o arcabouço proposto nesse trabalho para ser aplicado na componente *online* de algoritmos de agrupamento em FCDs. Inicialmente, o mesmo foi aplicado no algoritmo *CluStream* [Aggarwal et al., 2003], um dos algoritmos precursores da área de agrupamento em FCDs, com o intuito de validar seu desempenho quanto à problemática de detecção e tratamento de *outliers*.

3.1 Arcabouço proposto

O principal foco do arcabouço em estudo foi desenvolver uma estratégia capaz de lidar com a restrição de detecção de ruídos e *outliers* presentes em FCDs, que pode ser aplicada à componente *online* de algoritmos de agrupamento. Pois, muitos algoritmos de agrupamento não lidam com essa restrição, como por exemplo o *CluStream* [Aggarwal et al., 2003]. Outros algoritmos o fazem mas, em geral, dependem de muitos parâmetros de entrada difíceis de serem definidos, como por exemplo o *DenStream* [Cao et al., 2006].

A componente *online* de um algoritmo de agrupamento baseado em objetos é responsável por receber, analisar e sumarizar os objetos presentes no FCDs, em tempo viável. O mesmo tem como resultado a criação de um modelo capaz de representar as características do FCDs ao usuário, sempre que solicitado pela componente *offline* [Silva et al., 2013].

Uma apresentação abstrata e detalhada do arcabouço proposto pode ser encontrada na Figura 3.1.

Também é importante que o arcabouço proposto seja capaz de manter a memória auxiliar utilizada sempre atualizada, para isso, uma estratégia de remoção de objetos antigos é adotada. Uma apresentação abstrata deste processo é apresentada na Figura 3.2.

Cada subprocesso aplicado pelo arcabouço proposto neste trabalho é explicado de forma detalhada nas Subseções seguintes.

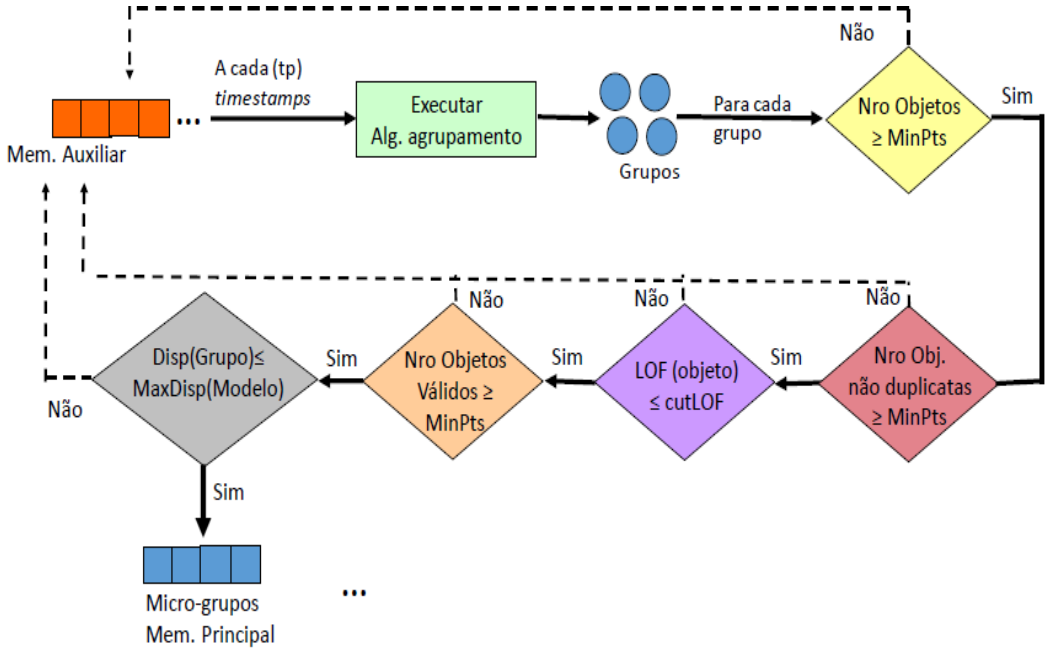


Figura 3.1: Arcabouço proposto.

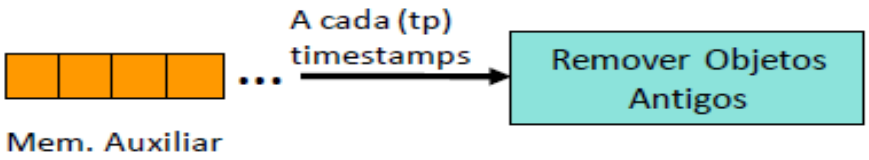


Figura 3.2: Tratamento de objetos antigos no arcabouço proposto.

3.1.1 Ativação da Memória Auxiliar

A Figura 3.3 apresenta de forma geral, a ativação do arcabouço proposto.

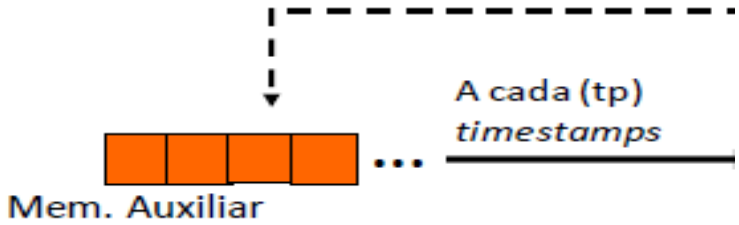


Figura 3.3: Ativação do processo de Detecção de *Outliers*.

O arcabouço proposto nesse trabalho é ativado somente se ambas as condições impostas ao processo forem satisfeitas. Tal que: i) o processo é executado apenas nos *timestamps* correspondentes a *tp* definidos pelo usuário e, ii) a memória auxiliar M_{aux} deve possuir mais que uma quantidade mínima de objetos definida pelo usuário ($minPts$). Caso ambas as condições sejam satisfeitas, o processo para Detecção de *Outliers* proposto é ativado e o próximo subprocesso é analisado.

Espera-se que valores de tp sejam menores que o tamanho de janela h definido, pois dessa forma é possível garantir que os objetos armazenados em M_{aux} serão analisados enquanto ainda forem válidos dada a janela de tempo em questão.

Caso algumas das condições mencionadas previamente não forem satisfeitas, o processo de Detecção de *Outliers* não é ativado.

Caso nenhuma das premissas anteriores sejam atendidas, um novo objeto $\mathbf{o_t}$ recebido pelo fluxo χ deverá ser analisado.

3.1.2 Agrupamento dos objetos da Memória Auxiliar

A Figura 3.4 apresenta o primeiro subprocesso aplicado após a ativação de M_{aux}, o agrupamento dos objetos presentes nessa memória .

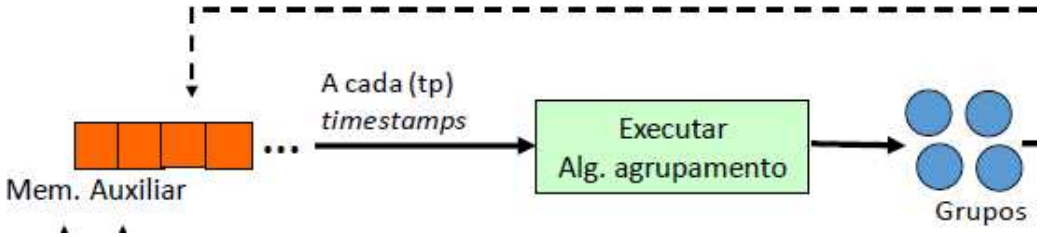


Figura 3.4: Agrupamento da Memória Auxiliar.

Nesse subprocesso uma técnica de agrupamento é selecionada e aplicada sobre os objetos armazenados em M_{aux} com o intuito de verificar se os grupos gerados são válidos e devem ser inseridos no Modelo M.

3.1.3 Validação dos grupos resultantes quanto a presença de um número mínimo de objetos

A Figura 3.5 apresenta o próximo subprocesso aplicado a cada grupo resultante da etapa anterior.

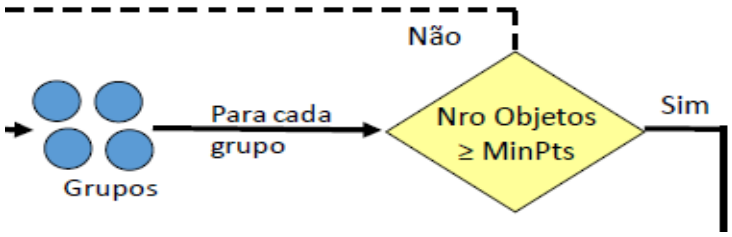


Figura 3.5: Validação dos grupos resultantes quanto a $minPts$.

Para cada grupo é necessário que o mesmo seja verificado quanto a presença de pelo menos $minPts$ objetos. Todos os grupos que possuem pelo menos essa quantidade mínima de pontos necessária prosseguem para a próxima etapa de validação do arcabouço proposto. Os grupos que não satisfazem essa restrição tem seus objetos mantidos em M_{aux} para futuras análises.

3.1.4 Validação dos grupos quanto à presença de duplicatas

A Figura 3.6 apresenta o próximo subprocesso aplicado a cada grupo resultante da etapa anterior quanto à existência de duplicatas.

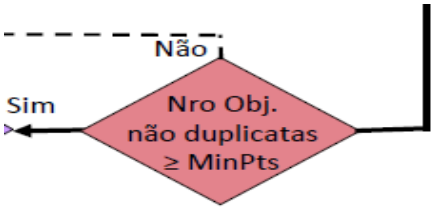


Figura 3.6: Validação dos grupos resultantes.

A proposta apresentada nesse trabalho considera que grupos formados por objetos duplicados devem ser automaticamente inseridos em M por possuírem alta coesão. Dessa forma, a verificação quanto a duplicatas implementada nesse arcabouço (*checkDup*) calcula a distância entre os objetos do grupo G_l em análise e caso existam objetos duplicados, ou seja, com distância entre si nula, os mesmos são removidos de G_l e adicionados em um micro-grupo que é inserido em M. O grupo G_l resultante desse processo é então analisado quanto a sua cardinalidade. Caso a cardinalidade de G_l ($|G_l|$) seja maior que $minPts$, então o grupo está apto a ser analisado no subprocesso seguinte.

Caso o grupo G_l resultante não esteja apto a ser analisado no subprocesso seguinte, os objetos presentes no grupo são mantidos em M_{aux} para análises futuras.

3.1.5 Aplicação da Técnica *LOF*

A Figura 3.7 apresenta o terceiro subprocesso a ser executado após a ativação do processo de Detecção de *Outliers*. Nesse subprocesso, uma técnica de Detecção de *Outliers* tradicional foi selecionada para ser aplicada sobre cada objeto $\mathbf{o}_y$ pertencente ao grupo G_l validado no subprocesso anterior.

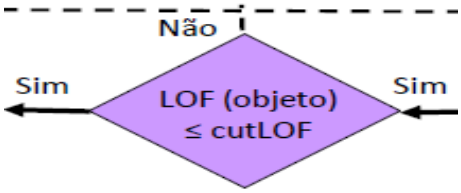


Figura 3.7: Aplicação da técnica LOF.

Para cada objeto $\mathbf{o}_y$ pertencente a G_l, aplica-se a técnica de Detecção de *Outliers* LOF (*Local Outlier Factor*) [Breunig et al., 2000], selecionada nesse estudo e apresentada brevemente na Subsubseção 3.1.5.1. Tal subprocesso tem o intuito de verificar quais objetos são considerados pela técnica como *inliers* ou *outliers*. A técnica LOF foi selecionada nesse estudo devido a apresentar bom desempenho

comparada com outras técnicas de Detecção de *Outliers* existentes, conforme apresentado no estudo experimental desenvolvido em [Campos et al., 2016].

O resultado dado pela técnica LOF para cada objeto $\mathbf{o}_y$ pertencente a G_l é comparado a um valor de corte $cutLOF$ definido pelo usuário. Esse valor é responsável por classificar cada objeto $\mathbf{o}_y$ como *inlier* ou *outlier*. Objetos com valores de LOF menores ou iguais ao valor de $cutLOF$ definido, são classificados como *inliers* e devem ser mantidos no grupo para serem analisados pelo próximo subprocesso. Os objetos classificados como *outliers*, segundo $cutLOF$, permanecem em M_{aux} para validações futuras ou até que sejam removidos.

3.1.5.1 *Local Outlier Factor (LOF)*

A técnica LOF busca por *outliers* baseando-se na densidade de um dado objeto o_y e na densidade de seus $minPts$ vizinhos, sendo $minPts$ definido pelo usuário. Essa técnica é pioneira no processo de Detecção de *Outliers* local e é baseada em *score* [Breunig et al., 2000]. A geração de um *score* para cada objeto o_y em estudo, faz necessário algumas definições.

Definição 2 *Seja D um conjunto de dados, para qualquer inteiro positivo $minPts$, a $minPts\_distance$ de um objeto o_y, definida por $minPts\_distance(o_y)$, é a distância $d(o_y, o_x)$ entre o_y e um objeto $o_x \in D$. Tal que: (i) no mínimo $minPts$ objetos $o' \in D \setminus \{o_y\}$ têm-se $d(o_y, o') \leq d(o_y, o_x)$ e (ii) no máximo $minPts - 1$ objetos $o' \in D \setminus \{o_y\}$ têm-se $d(o_y, o') < d(o_y, o_x)$.*

Definição 3 *Dada a $minPts\_distance(o_y)$, a vizinhança $minPts\_distance$ de o_y contém cada objeto cuja distância para o_y não é maior que $minPts\_distance(o_y)$. Formalmente, $N_{minPts\_dist}(o_y) = \{q \in D \setminus \{o_y\} | d(o_y, q) \leq minPts\_distance(o_y)\}$.*

Definição 4 *Seja $minPts$ um número natural, a distância de alcançabilidade (ou "reachability distance") de um objeto o_y para um objeto o_x é definida como $reach\_dist_{minPts}(o_y, o_x) = max\{minPts\_distance(o_x), d(o_y, o_x)\}$.*

A distância de alcançabilidade apresentada na Definição 4 foi utilizada para verificar qual é o limite da vizinhança de um ponto o_y, com a ideia de eliminar a influência de outros *outliers*.

Com as Definições apresentadas, o próximo passo é calcular a densidade da vizinhança de um ponto o_y. A Equação 3.1 é definida pelo inverso da média das distâncias de alcançabilidade entre o_y e seus vizinhos. Quanto maior o valor de $lrd_{minPts}(o_y)$, mais densa é a região onde o_y está. O *score* final da técnica LOF é dado pela Equação 3.2, a qual calcula uma média das razões entre as densidades de cada vizinho mais próximo de o_y em relação à densidade de o_y. Este cálculo possibilita verificar se a vizinhança mais próxima de o_y é relativamente mais densa que o próprio o_y ou não, evitando os problemas encontrados por algoritmos globais Campos et al. [2016].

$$lrd_{minPts}(o_y) = 1/\left(\frac{\sum_{o_x \in N_{minPts\_dist}(o_y)} reach\_dist_{minPts}(o_y, o_x)}{|N_{minPts\_dist}(o_y)|}\right) \tag{3.1}$$

$$lof_{minPts}(o_y) = \frac{\sum_{o_x \in N_{minPts\_dist}(o_y)} \frac{lrd_M inPts(o_x)}{lrd_M inPts(o_y)}}{|N_{minPts\_dist}(o_y)|} \tag{3.2}$$

O valor de *minPts* utilizado internamente pela técnica LOF deve ser definido pelo usuário. O *score* produzido pelo LOF possui valor esperado em torno de 1, quando a densidade do objeto e de seus vizinhos é aproximadamente a mesma, o que ocorre com *inliers* internos. Valores cada vez maiores que 1 caracterizam o ponto como um *outlier*.

3.1.6 Validação dos grupos resultantes da técnica LOF quanto a presença de um número mínimo de objetos

A Figura 3.8 apresenta o próximo subprocesso aplicado a cada grupo resultante da etapa anterior.

Figura 3.8: Validação dos grupos resultantes quanto a $minPts$.

Para cada grupo composto por objetos classificados pela técnica LOF no subprocesso anterie é verificado quanto a presença de pelo menos $minPts$ objetos. Todos os grupos que possuem pelo menos essa quantidade mínima de pontos necessária prosseguem para a próxima etapa de validação do arcabouço proposto. Os grupos que não satisfazem essa restrição tem seus objetos mantidos em M_{aux} para futuras análises.

3.1.7 Cálculo da dispersão de um potencial micro-grupo

A Figura 3.9 apresenta o último subprocesso aplicado pelo processo de Detecção de *Outliers* do arcabouço proposto. O grupo resultante dos subprocessos anteriores tem sua dispersão calculada.

A dispersão de cada micro-grupos do modelo M também é calculada e a máxima dispersão é armazenada. A dispersão do grupo em análise pelo arcabouço proposto é então comparada à máxima dispersão de M. Caso a mesma seja menor ou igual a máxima dispersão, o grupo em análise é considerado válido e apto a ser inserido no modelo M.

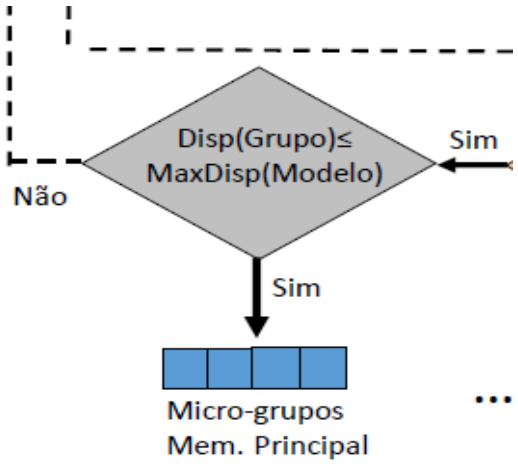


Figura 3.9: Cálculo da dispersão do grupo em análise.

A dispersão utilizada nesse arcabouço é dada pela Equação 3.3 e quanto maior o valor resultante, mais esparso é o micro-grupo em análise [Spinosa, 2008].

A Equação 3.3 calcula a dispersão de um potencial micro-grupo (m_{pot}). A mesma é dada pela subtração entre a soma quadrada de todos os objetos pertencentes ao micro-grupo ($m_{pot}.\vec{SS}$) pelo tamanho do micro-grupo ($m_{pot}.N$) e o quadrado da soma de todos os objetos pertencentes micro-grupo ($m_{pot}.\vec{LS}$) pelo tamanho do micro-grupo ($m_{pot}.N$).

$$disp(m_{pot}) = \left(\frac{m_{pot}.\vec{SS}}{m_{pot}.N} - \left(\frac{m_{pot}.\vec{LS}}{m_{pot}.N} \right)^2 \right) \tag{3.3}$$

Caso o grupo não satisfaça a condição previamente apresentada, os objetos que o compõe permanecem em M_{aux} para validações futuras.

O processo de inserção do grupo considerado válido pelo arcabouço proposto é semelhante ao processo de inserção apresentado pelo algoritmo de agrupamento selecionado, no qual o arcabouço está aplicado em sua componente *online*.

3.1.8 Remoção de objetos obsoletos da Memória Auxiliar

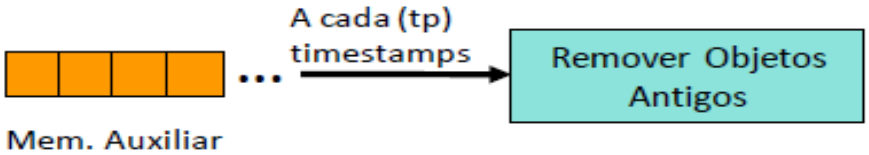


Figura 3.10: Remoção de objetos antigos em M_{aux}.

A Figura 3.10 apresenta o processo de verificação de M_{aux} para remoção dos objetos *inliers* obsoletos e *outliers*. A remoção de objetos menos relevantes ou

antigos de M_{aux} ocorre na mesma periodicidade que a validação de M_{aux} e de forma semelhante ao processo de remoção adotado pelo algoritmo de agrupamento em FCDs, selecionado para estudo. O mesmo tem como objetivo buscar objetos que estão na memória auxiliar a mais que h *timestamps* e não foram absorvidos pelo modelo até o presente momento.

Esse processo é importante para evitar que objetos muito antigos que não representam mais o modelo M, sobre o horizonte h, sejam em algum momento incluídos pelo processo de validação de M_{aux} em M, depreciando sua qualidade.

3.2 Pseudo-código referente ao arcabouço proposto

O Algoritmo 1 apresenta o pseudo-código referente ao arcabouço proposto nesse trabalho e apresentado nas Figuras 3.1 e 3.2.

Algorithm 1 *analiseMemAuxiliar*

```
1: Entrada: M_aux, h, tp, minPts, cutLOF
2: se (t % tp == 0) então
3:     se (|M_aux| > minPts) então
4:         k_aux ← |M_aux| / minPts // |M_aux| é cardinalidade de M_aux
5:         G_aux ← k-médias++(k_aux, M_aux)
6:         para G_l ∈ G_aux faça
7:             se (¬(checkDup(G_l))) e (|G_l| > minPts) então
8:                 para o_y ∈ G_l faça
9:                     result_lof ← lof(o_y)
10:                    se result_lof ≤ cutLOF então
11:                        insere(o_y, m_pot)
12:                    fim se
13:                fim para
14:                se (m_pot.N > minPts) e (disp(m_pot) ≤
15:                                    (max_{m_i∈M} (disp(m_i)))) então
16:                    insereM(m_pot, M) // ver Algoritmo 3
17:                fim se
18:            fim se
19:        fim para
20:    fim se
21:    // Checagem para remoção de objetos obsoletos em M_aux
22:    para (o_y ∈ M_aux) faça
23:        se o_y < (t − h) então
24:            remove(o_y, M_aux)
25:        fim se
26:    fim para
27: fim se
```

Conforme apresentado na Linha 1 do Algoritmo 1, o arcabouço proposto possui como entrada os seguintes parâmetros:

- M_{aux}: memória auxiliar utilizada no arcabouço proposto para armazenar todos os objetos que não foram absorvidos pelo modelo M.
- h: tamanho da janela de tempo (Ver Subseção 2.3.2).
- tp: intervalo de tempo definido para aplicação da validação interna na Memória Auxiliar e remoção de objetos antigos na mesma.
- $minPts$: quantidade mínima de pontos necessária e aplicada nos subprocessos de validação interna da Memória Auxiliar.
- $cutLOF$: valor limiar de corte necessário pela técnica de Detecção de *Outliers* selecionada, para definir os objetos como *inliers* ou *outliers*.

3.3 *CluStreamOD*: Extensão *CluStream* com Detecção de *Outliers*

O algoritmo *CluStreamOD* apresentado nesse trabalho é uma extensão do algoritmo *CluStream* [Aggarwal et al., 2003] que aplica em sua componente *online* o arcabouço proposto nesse trabalho para detecção e tratamento de *outliers*.

O pseudo-código referente a componente *online* do *CluStreamOD* é apresentado no Algoritmo 2.

Algorithm 2 Componente *online* do *CluStreamOD*

1: Entrada: $q, M_{init}, h, \chi, minPts, cutLOF, tp$
2: $M \leftarrow k$-médias++(q, M_{init})
3: // t é o *timestamp* (tempo de chegada) de $\mathbf{o_t}$ no fluxo χ
4: **para** $\mathbf{o_t} \in \chi$ **faça**
5: // encontra o micro-grupos mais próximo m_j
6: $j \leftarrow \arg\min_{m_j \in M}(dist(\mathbf{o_t}, m_j))$
7: **se** $dist(\mathbf{o_t}, m_j) < limiteMaximo(m_j)$ **então**
8: insere$(\mathbf{o_t}, m_j)$
9: **senão**
10: adiciona$(\mathbf{o_t}, M_{aux})$ // added $\mathbf{o_t}$ in M_{aux}
11: **fim se**
12: $analiseMemAuxiliar(M_{aux}, h, tp, minPts, cutLOF)$ // ver Algoritmo 1
13: **fim para**

Conforme apresentado na Linha 1 do Algoritmo 2, a componente *online* do arcabouço proposto possui como entrada os seguintes parâmetros:

- q: quantidade fixa de micro-grupos que representam o Modelo. (Parâmetro herdado do *CluStream* [Aggarwal et al., 2003].)
- M_{init}: quantidade de objetos iniciais definidos pelo usuário para geração dos q primeiros micro-grupos do modelo. (Parâmetro herdado do *CluStream* [Aggarwal et al., 2003].)

- χ: representa o Fluxo Contínuo de Dados (FCDs).

Com os parâmetros de entrada definidos, conforme encontrado na maioria dos algoritmos de agrupamento em FCDs e verificado na Linha 2 do Algoritmo 2, os micro-grupos iniciais que compõe o modelo M de χ precisam ser gerados.

Para que o modelo inicial seja gerado, o algoritmo *CluStreamOD* seleciona e armazena os primeiros objetos (dado por M_{init}) que chegam ao fluxo χ. Posteriormente, um algoritmo de agrupamento de dados adaptado ao cenário de FCDs é executado sobre os M_{init} objetos armazenados (ver Algoritmo 2, Linha 2), gerando os q primeiros micro-grupos do modelo.

No arcabouço em estudo, é importante ressaltar que os M_{init} objetos armazenados são livres de *outliers*. O objetivo dessa estratégia é partir de um Modelo Inicial formado por micro-grupos de qualidade para prosseguir a análise de novos objetos que chegam ao FCDs.

Para que M_{init} gere os primeiros q micro-grupos de M, faz-se necessário a aplicação de um algoritmo de agrupamento adaptado ao cenário de FCDs. O *CluStreamOD*, assim como o algoritmo *CluStream* [Aggarwal et al., 2003], utilizou-se do algoritmo k-médias++ [Arthur & Vassilvitskii, 2007] para agrupar os objetos iniciais do fluxo. Nesse algoritmo, o primeiro centróide é escolhido de forma aleatória e os demais são escolhidos a partir das probabilidades proporcionais aos objetos restantes. Essa inicialização é responsável pelo desempenho superior desse algoritmo em relação k-médias [Arthur & Vassilvitskii, 2007].

Após a geração dos micro-grupos iniciais, os novos objetos chegam ao FCDs precisam ser absorvidos por M (Linha 4 à Linha 11 no Algoritmo 2).

Na componente *online*, novos objetos $\mathbf{o_t}$ chegam ao fluxo χ com tempo de chegada t e precisam ser analisados (ver Algoritmo 2, Linha 4).

Os q micro-grupos gerados inicialmente são então analisados a fim de verificar se o novo objeto do fluxo $\mathbf{o_t}$ será absorvido por algum micro-grupo de M. Um novo objeto é absorvido pelo modelo se apresenta características similares a algum micro-grupo já presente em M.

Quando um novo objeto $\mathbf{o_t}$ chega ao fluxo χ, a distância euclidiana entre $\mathbf{o_t}$ e os centróides dos q micro-grupos de M é calculada, a fim de encontrar o micro-grupo mais próximo de $\mathbf{o_t}$, dado por m_j (ver Algoritmo 2, Linha 6). O cálculo do centróide de um micro-grupo é realizado conforme apresentado na Subseção 2.3.1.

Caso a distância entre $\mathbf{o_t}$ e m_j previamente calculada seja menor que o máximo limite (ver Definição 5) do micro-grupo m_j (ver Algoritmo 2, Linha 7), $\mathbf{o_t}$ deve ser absorvido/inserido em m_j, pois possui características similares aos objetos que compõe esse micro-grupo (ver Algoritmo 2, Linha 8). O processo de inserção em um micro-grupo é facilmente realizado pela propriedade de Aditividade (ver Subseção 2.3.1).

Definição 5 *O limite máximo de um micro-grupo m_q é definido como um fator de t (dado que t corresponda ao timestamp do objeto) do desvio RMS dos objetos em m_q para o seu centróide. O desvio RMS apenas pode ser definido para micro-grupos com cardinalidade superior a 1. Caso o micro-grupo possua cardinalidade 1, a distância para o micro-grupo mais próximo é considerada [Aggarwal et al., 2003].*

Quando um objeto $\mathbf{o_t}$ não pode ser absorvido por um micro-grupo de M, pode-se inferir ou que o mesmo corresponde a um objeto *outlier* ou pertence a um padrão novidade que está surgindo no Fluxo Contínuo de Dados. Porém, diferentemente do algoritmo *CluStream*, quando o objeto $\mathbf{o_t}$ não é absorvido por m_j, o *CluStreamOD* insere este objeto em uma memória auxiliar, dada por M_{aux}, para análise futura (ver Algoritmo 2, Linha 10). Todos os objetos inseridos em M_{aux} são determinados como possíveis *outliers* por não possuírem características similares aos micro-grupos que representam M.

Para diferenciar os reais *outliers* dos padrões novidades presentes em M_{aux}, o *CluStreamOD* utiliza-se de um processo interno de Detecção de *Outliers* aplicado em M_{aux}. Dessa forma, o mesmo é capaz de atender as restrições impostas aos algoritmos de agrupamento de FCDs, quanto à capacidade de diferenciar padrões novidades e *outliers*.

A Linha 12 do Algoritmo 2 demonstra a chamada ao processo de Detecção de *Outliers* desenvolvido nesse trabalho e aplicado à componente *online* dos algoritmos de agrupamento em FCDs. É importante ressaltar que o algoritmo de agrupamento selecionado e aplicado internamente no arcabouço proposto é o k-médias++ [Arthur & Vassilvitskii, 2007].

Todos os grupos considerados válidos pelo arcabouço proposto devem ser inseridos em M. Para isso, o mesmo é sumarizado em um micro-grupo e o pseudo-código referente ao processo de inserção de um novo micro-grupo em M é dado pelo Algoritmo 3 e derivado do algoritmo *CluStream* [Aggarwal et al., 2003].

Algorithm 3 *insereM*

1: Entrada: m_{pot}, M
2: $e \leftarrow \arg\min_{m_y \in M} (\frac{m_y.LST}{m_y.N})$ //micro-grupo mais antigo
3: **se** $(\frac{m_e.LST}{m_e.N} < (t - h))$ **então**
4: remove(m_e, M) // remove micro-grupo menos relevante
5: **senão**
6: merge(M) // dois micro-grupos mais próximos de M são unidos
7: **fim se**
8: insere(m_{pot}, M)

Devido a restrição de se manter uma quantidade fixa q de micro-grupos no Modelo M, a inserção de um novo micro-grupo resulta na necessidade de exclusão de um micro-grupo do Modelo ou união de dois micro-grupos mais próximos.

Inicialmente, conforme apresentado nas Linhas 2 a 4 do Algoritmo 3, o micro-grupo mais antigo do Modelo é encontrado e verifica-se se o mesmo já está no Modelo a mais tempo que a janela de tempo h definida pelo usuário. Caso sim, o micro-grupo é removido e o novo micro-grupo é então inserido em M.

Caso o micro-grupo mais antigo ainda não tenha se tornado obsoleto, considerando-se a janela de tempo h definida pelo usuário, dois micro-grupos mais próximos são encontrados e unidos, conforme apresentado na Linha 6 do Algoritmo 3. Novamente, após a união desses micro-grupos, o novo micro-grupo pode ser in-

serido no modelo, conforme Linha 8 do Algoritmo 3, respeitando a quantidade fixa de q micro-grupos em M.

3.4 Componente *offline* dos Algoritmos de Agrupamento em FCDs

A componente *offline* dos algoritmos de agrupamento em FCDs é responsável por exibir ao usuário final o comportamento do FCDs ao longo do tempo, com base nos micro-grupos que compõe M, em conjunto com os parâmetros de entrada determinados.

Essa componente é responsável por apresentar a evolução do FCDs sempre que solicitado pelo usuário. Dessa forma, esse pode acompanhar o comportamento dos grupos dada uma janela de tempo h definida. Ao longo de um FCDs, novos grupos podem surgir, expandir ou desaparecer, de acordo com a distribuição aplicada aos objetos que chegam ao FCDs.

O algoritmo *CluStreamOD* apresentado nesse trabalho, possibilita que o usuário escolha *timestamps* aleatórios para acompanhamento do FCDs. Para isso, aplica em sua componente *offline* o algoritmo k-médias++ sobre os micro-grupos do modelo M, sempre que solicitado pelo usuário. O formato dos grupos exibidos por esse processo está diretamente ligado ao algoritmo de agrupamento selecionado, nesse caso, grupos hiper-esféricos são formados [Silva et al., 2013].

Métricas de validação podem ser aplicadas sobre os grupos resultantes da requisição dessa componente, a fim de avaliar a disposição dos grupos no FCDs analisado, ou mesmo a qualidade interna dos mesmos.

Capítulo 4

RESULTADOS e DISCUSSÕES

O arcabouço proposto nesse trabalho foi aplicado à componente *online* do algoritmo de agrupamento *CluStream*, para tratamento e detecção de *outliers*. A mesma resultou no algoritmo *CluStreamOD (CluStream with Outlier Detection)*.

Este capítulo apresenta uma comparação entre o *CluStreamOD* e o algoritmo *CluStream* original [Aggarwal et al., 2003], com a finalidade de medir o desempenho dos mesmos quanto à detecção de *outliers* em FCDs.

Ambos os algoritmos, *CluStream* e *CluStreamOD*, foram desenvolvidos em Java e comparados utilizando o arcabouço *MOA (Massive Online Analysis)* [Bifet & Kirkby, 2009]. Para simulação de um fluxo, conjuntos de dados aplicados ao cenário de detecção de *outliers* e FCDs foram selecionados. Uma breve apresentação sobre os conjuntos de dados reais e artificiais aplicados a esse estudo se encontra na Seção 4.1.

4.1 Conjuntos de Dados em estudo

Os conjuntos de dados utilizados nesse trabalho foram selecionadas devido à sua relevância para validação de propostas para FCDs e cenários de detecção de *outliers*. Os conjuntos foram coletados do repositório *UCI Machine Learning Repository* Lichman [2013] e do repositório gerado por Campos et al. [2016]. Em sua maioria, os conjuntos de dados reais selecionados não foram gerados para o cenário de FCDs. Dessa forma, um pré–processamento foi aplicado sobre os mesmos de modo que os dados foram reorganizados simulando um fluxo.

Todos os conjuntos de dados reais selecionados nesse estudo, com a presença de *outliers*, tiveram seus objetos reorganizados de forma simular um FCDs. Essa reorganização é dada por meio da seleção aleatória dos primeiros objetos de cada grupo e posteriormente os demais objetos são selecionados com base na proporção de objetos existentes em cada grupo do conjunto[1]. Vale ressaltar que todos os conjuntos de dados com *outliers* também foram estruturados de forma a manter os primeiros objetos do FCDs, utilizados para a criação dos micro-grupos iniciais do modelo, livres de *outliers* [Aggarwal et al., 2003].

[1]Disponível em: `www.facom.ufu.br/~elaine/projetos/bases/basesOutlier`

Também foram gerados conjuntos de dados artificiais (utilizando o arcabouço MOA [Bifet & Kirkby, 2009]) com quantidades de grupos e atributos diferentes, de forma a verificar o impacto desses parâmetros no comportamento do algoritmo *CluStreamOD*. Uma descrição detalhada sobre cada conjunto em estudo é mostrada na Tabela 4.1.

Quanto à comparação de ambos os algoritmos, a mesma se dá pelo uso de métricas de validação internas e externas, implementadas no *MOA* e selecionadas para esse estudo. Um breve discussão acerca das métricas selecionadas é apresentada na Seção 4.3.

Tabela 4.1: Principais características dos conjuntos de dados em estudo.

Nome	*Inliers*	*Outliers*	*Atributos*	*Grupos*	*Tipo*
Covertype	480.000	-	54	7	Real
$KDDCUP99_{10\%}$	494.021	-	34	5	Real
PenDigits	9.848	20	16	10	Real
Shuttle	1.000	13	9	7	Real
WaveForm	3.343	100	21	3	Real
Art_2:15att_5c	90.000	10.000	2, 5 e 15	5	Artificial
Art_2:15att_15c	90.000	10.000	2, 5 e 15	15	Artificial

A Tabela 4.1 apresenta as principais características dos conjuntos de dados selecionados nesse estudo. É importante ressaltar que todos os conjuntos de dados selecionados do repositório gerado por Campos et al. [2016] estavam subdivididos em apenas duas classes, *inliers* e *outliers*. Dessa forma, devido a relevância das classes reais de cada conjunto para a problemática em estudo, estas foram aplicadas aos seus respectivos objetos em cada conjunto de dados.

- **Conjunto Covertype:** Esse conjunto diz respeito à predição do tipo de cobertura de uma floresta, por meio de informações cartográficas como elevação, tipo do solo, etc. Está subdividido em 7 grupos distintos, correspondentes aos tipos de cobertura analisados. O conjunto em questão foi utilizado por [Masud et al., 2011] e analisa 480.000 objetos *inliers*, considerando 54 atributos. Os objetos foram normalizados e não há a presença de objetos *outliers* e duplicatas.

- **Conjunto $KDDCUP99_{10\%}$: (*Network Intrusion Detection)*:** Esse conjunto possui registros relacionados a intrusões/ataques simulados em uma rede de computadores. A versão utilizada nesse artigo compreende uma subamostra de 10% do conjunto original, conforme apresentado em Lichman [2013]. Tal conjunto é amplamente utilizado em problemáticas envolvendo cenários de FCDs, pois seu comportamento pode ser analisado como um fluxo. O conjunto possui 5 grupos (dos, u2r, probe, normal e r2l) responsáveis por agrupar os 22 tipos de ataques diferentes, sem a presença de objetos *outliers*. A versão utilizada nesse trabalho possui 494.021 objetos e apenas os 34 atributos numéricos foram utilizados. O mesmo foi normalizado e possui duplicatas.

- **Conjunto PenDigits (*Pen-Based Recognition of Handwritten Digits)*:** O mesmo possui 10 grupos que correspondem aos dígitos de 0 a 9

representados por meio de escrita manual. A definição dos *outliers* presentes no conjunto segue a abordagem utilizada em Campos et al. [2016], onde o grupo 4 é definido como *outlier* e subamostrado para apenas 20 objetos. O mesmo apresenta 16 atributos numéricos e 9868 objetos, divididos em 20 *outliers* (0.2%) e 9848 *inliers* (99.8%). O conjunto foi normalizado e não possui duplicatas.

- **Conjunto WaveForm**: O conjunto possui 3 grupos de ondas. O grupo 0 foi definido como grupo *outlier* e subamostrado para 100 objetos, assim como realizado em Campos et al. [2016]. Após o pré-processamento, o mesmo apresenta 21 atributos numéricos e 3443 objetos, divididos em 100 *outliers* (2.9%) e 3343 *inliers* (97.1%). Uma normalização foi aplicada aos dados e não há a presença de duplicatas.

- **Conjunto Shuttle:** Este conjunto apresenta a leitura de um dispositivo para controle de fluxo de fluídos em aeronaves. Conforme apresentado em Campos et al. [2016], os grupos 1,3,4,5,4 e 7 são definidos como *inliers* e o grupo 2 foi subamostrado como *outlier*. Entre os grupos *inliers*, uma seleção de 1000 objetos foi realizada para se unir a subamostra de 13 *outliers*. O conjunto possui 9 atributos numéricos e 1013 objetos, sendo 1000 *inliers* (98.72%) e 13 *outliers* (1.28%), normalizados e sem a presença de duplicatas.

- **Conjunto Art_2:15att_5c:** Foram gerados três conjuntos de dados com 2, 5 e 15 atributos por meio do arcabouço MOA Bifet & Kirkby [2009]. O mesmo permite que a evolução dos grupos ao longo do tempo seja configurada de acordo com a necessidade do usuário. Sendo assim, foi gerado um fluxo contendo 100.000 objetos divididos em 99.000 *inliers* (99%) e 1.000 *outliers* (1%), sem a presença de objetos duplicatas. A quantidade de grupos foi definida como 5, possibilitando que surjam ou desapareçam novos grupos dentro de um limiar de ±3 (mínimo de 2 e máximo de 8 grupos).

- **Base Art_2:15att_15c:** Foram gerados três conjuntos de dados, com 2, 5 e 15 atributos por meio do arcabouço MOA Bifet & Kirkby [2009]. O mesmo permite que a evolução dos grupos ao longo do tempo seja configurada de acordo com a necessidade do usuário. Sendo assim, foi gerado um fluxo contendo 100.000 objetos divididos em 99.000 *inliers* (99%) e 1.000 *outliers* (1%), sem a presença de objetos duplicatas. A quantidade de grupos foi definida como 15, possibilitando que surjam ou desapareçam novos grupos dentro de um limiar de ±3 (mínimo de 12 e máximo de 18 grupos).

Dois conjuntos de dados reais, sem *outliers*, também foram selecionados nesse estudo a fim de demonstrar que as estruturas auxiliares e subprocessos aplicados ao *CluStreamOD* não influenciaram negativamente no desempenho do algoritmo. Uma descrição acerca da parametrização de ambos os algoritmos é apresentada na Seção 4.2.

4.2 Parametrização dos algoritmos em estudo

Os experimentos foram conduzidos em um PC com Intel Core i7-4510U CPU 2.00GHz com 8GB de memória com Sistema Operacional Windows 10 Home Single Language. Os parâmetros de entrada utilizados para ambos os algoritmos foram definidos a partir dos dos parâmetros padrão do *CluStream* [Aggarwal et al., 2003], implementado no *MOA*. Devido aos conjuntos de dados possuírem quantidades diferentes de objetos, os mesmos foram calculados de forma proporcional para cada conjunto. Os parâmetros específicos do *CluStreamOD* foram determinados de forma empírica, baseados em estudos desenvolvidos em Breunig et al. [2000] e Campos et al. [2016]. Uma apresentação detalhada dos parâmetros de entrada foi demonstrada na Tabela 4.2.

É importante ressaltar que o algoritmo de agrupamento utilizado na componente *offline* e na Geração Modelo Inicial (ver Subseções 3.4 e **??**, respectivamente) dos algoritmos *CluStream* e *CluStreamOD* foi o k-médias++ [Arthur & Vassilvitskii, 2007].

Tabela 4.2: Parâmetros de Entrada: Algoritmos *CluStream* e *CluStreamOD*

Conjunto (χ)	h	q	M_{init}	$minPts$	$cutLOF$	tp
Covertype	1000	100	1000	3	1.5	100
$KDDCUP99_{10\%}$	1000	100	1000	3	1.5	100
PenDigits	500	50	500	3	1.5	50
Shuttle	100	10	100	3	1.5	10
WaveForm	500	50	500	3	1.5	50
Art_2:15att_5g	1000	100	1000	3	1.5	100
Art_2:15att_15g	1000	100	1000	3	1.5	100

4.3 Métricas de Validação selecionadas

De maneira a garantir maior confiabilidade aos resultados apresentados, todos os experimentos foram executados 10 vezes para cada conjunto de dados, dado o *tp* selecionado (ver Tabela 4.2). As métricas de validação selecionadas nesse trabalho foram: i) *CMM (Cluster Mapping Measure)* Kremer et al. [2011], ii) o Índice Silhueta Rousseeuw [1987] e iii) Índice Rand Ajustado [Rand, 1971]. Os valores médios de cada métrica em estudo serão apresentados nas subseções a seguir, para cada conjunto de dados.

A métrica de avaliação externa *CMM (Cluster Mapping Measure)* [Kremer et al., 2011], é formada pelas componentes *Missed*, *Misplaced* e *Noise*, que indicam de forma efetiva diferentes tipos de erros que devem ser considerados importantes no cenário de FCDs. Um objeto é penalizado como *missed* se não é um *noise* e não foi atribuído a algum grupo/micro-grupo. Objetos *misplaced* são objetos atribuídos em grupos que possuem objetos de uma classe diferente da sua. Por fim, o erro *noise* é dado pela atribuição de um objeto *noise/outlier* a um micro-grupo/grupo.

O cálculo do valor de *CMM* se dá pela multiplicação de suas componentes. Porém, segundo Kremer et al. [2011], uma melhor comparação dessa métrica é dada pela análise de suas componentes em separado. A métrica *CMM* e suas componentes resultam em valores entre 0 e 1, de forma que quanto mais próximo de 1, melhor.

A métrica de avaliação interna Silhueta (*Silhouette Index*) [Rousseeuw, 1987] foi selecionada para avaliar quão bem dispostos estão os objetos dentro de seu grupo. A mesma apresenta bom desempenho para cenários de agrupamento de dados [Vendramin et al., 2010]. Os valores resultantes dessa métrica variam entre -1 e 1, de forma que o valor 1 (melhor valor de Silhueta) é dado quando a distância intra-grupo é minimizada e a distância inter-grupos é maximizada.

A métrica de avaliação externa, o índice Rand Ajustado (*Adjusted Rand Index*) [Yeung & Ruzzo, 2001] é capaz de analisar a similaridade existente entre dois agrupamentos. A mesma apresenta uma normalização da métrica Rand [Rand, 1971], de forma um valor resultante 0 é esperado quando dois agrupamentos aleatórios são comparados. A mesma apresenta desempenho superior quando comparada a outras métricas de avaliação externa [Vendramin et al., 2010]. Não há valores mínimos para esse índice, mas o valor máximo 1 é obtido quando os dois agrupamento comparados são equivalentes.

Ambas as métricas e conjuntos de dados em estudo possuem uma análise a nível macro-agrupamento e micro-agrupamento, visto que a taxa de *outliers* presentes principalmente nos conjuntos reais em estudo é muito baixa, tornando-se em alguns casos imperceptível o desempenho do algoritmo *CluStreamOD* a nível macro.

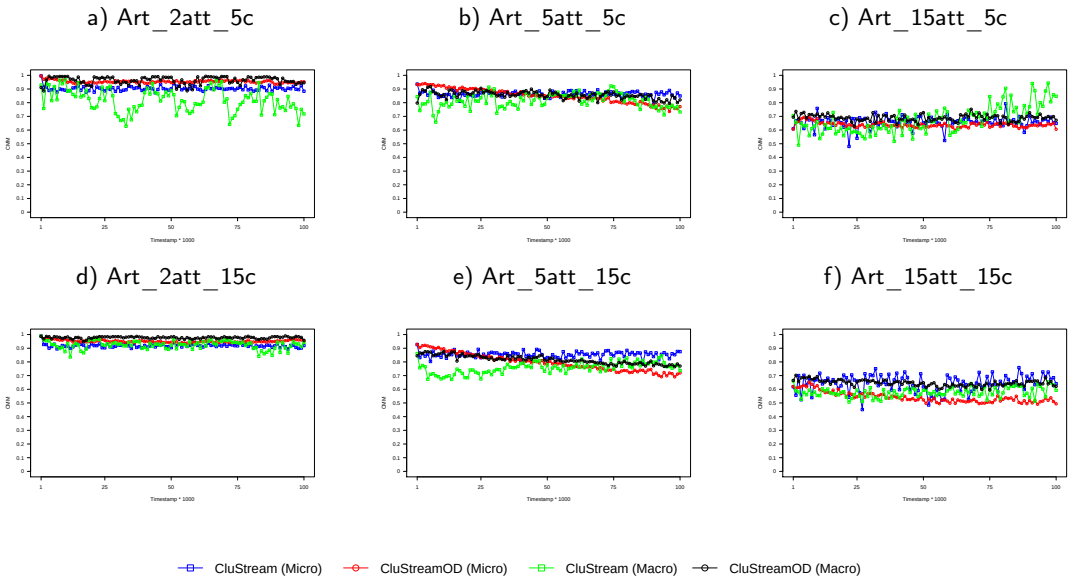
4.4 Avalição dos Conjuntos de Dados Artificiais: Nível Macro e Micro agrupamento

Uma avaliação a nível macro e micro-agrupamento dos conjuntos artificiais selecionados nesse estudo será apresentada a seguir para cada métrica de avaliação.

4.4.1 Métrica CMM - Nível Macro e Micro

Analisando-se a Figura 4.1 é possível perceber os altos valores apresentados pela métrica CMM quando comparados aos conjuntos de dados reais. A medida que o número de atributos aumenta (Figuras 4.1c e 4.1f), mais erros são identificados pelo fluxo, conforme apresentado pelos valores de CMM. O algoritmo *CluStreamOD* apresenta, segundo a métrica em estudo, os melhores valores para os conjuntos com menor quantidade de atributos (Figuras 4.1a e 4.1d), quando comparado ao *CluStream*. Porém, conforme mencionado previamente, uma melhor avaliação da métrica CMM é obtida pelas suas componentes em separado. Sendo assim, uma análise individual das mesmas é apresentada a seguir.

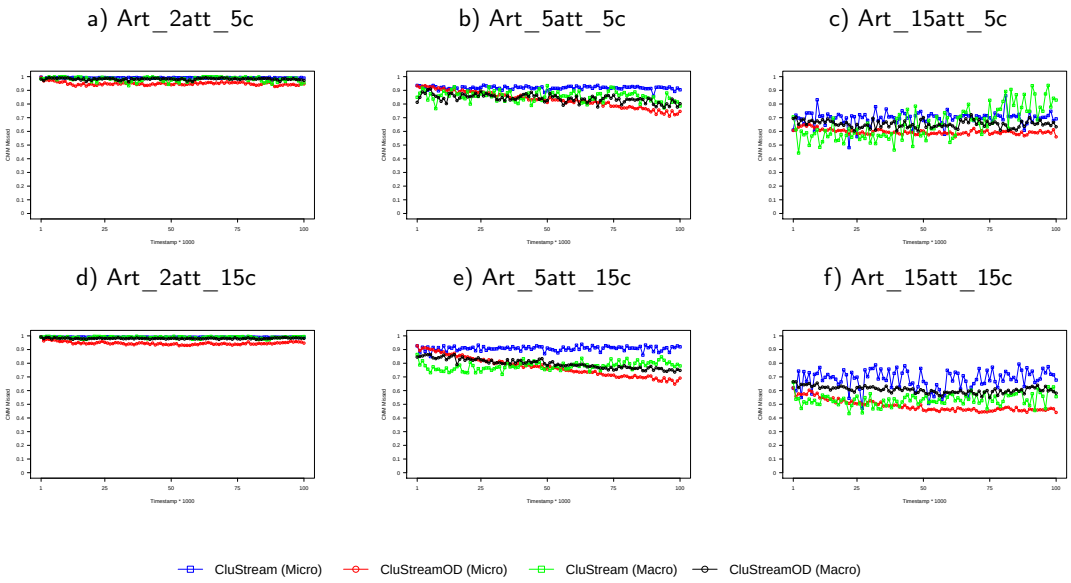
Figura 4.1: Valores médios da métrica CMM nos conjuntos de dados artificiais com *outliers* - Nível Macro e Micro



4.4.2 Métrica CMM Missed - Nível Macro e Micro

Analisando-se a Figura 4.2 pode-se perceber a superioridade do algoritmo *CluStream* nos conjuntos em questão para uma análise a nível micro. As análise a nível macro apresentam valores quase que equivalentes para ambos os algoritmos.

Figura 4.2: Valores médios da métrica CMM Missed nos conjuntos de dados artificiais com *outliers* - Nível Macro e Micro



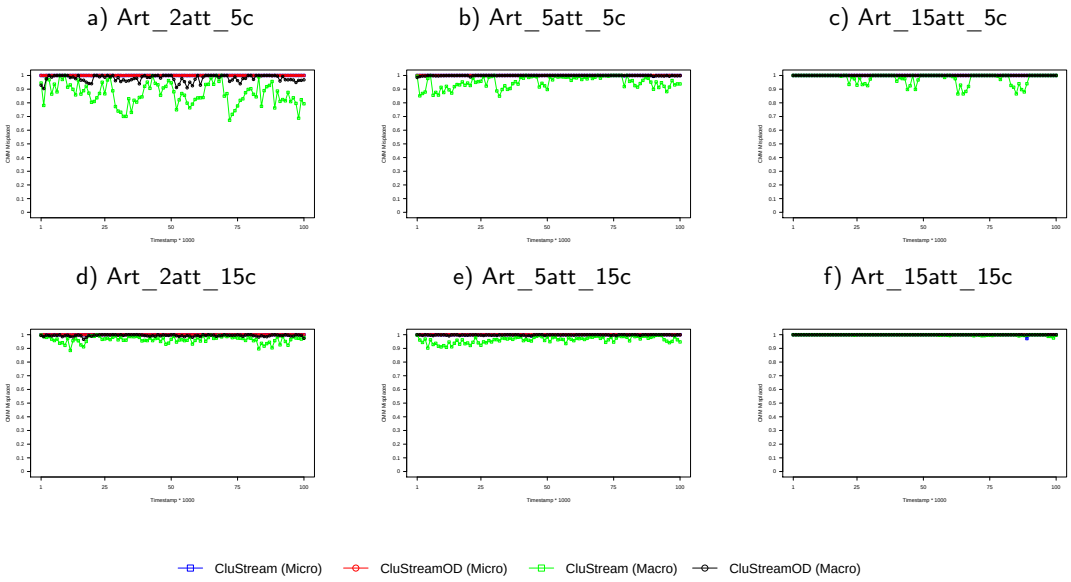
Considerando-se o algoritmo *CluStreamOD*, uma explicação para os valores inferiores da componente *Missed* a nível micro-agrupamento é que no momento da

verificação, alguns objetos *inliers* inseridos na memória auxiliar ainda não passaram pelo processo de validação interna para serem inseridos ao modelo, sendo marcados como *missed*. A inferioridade do *CluStreamOD* a nível micro pode ser percebida em todos os cenários em estudo ao longo do FCDs.

4.4.3 Métrica CMM Misplaced - Nível Macro e Micro

A Figura 4.3 apresenta a superioridade do algoritmo *CluStreamOD* para todos os conjuntos em estudo, a nível macro e micro, com relação ao erro descrito pela componente *Misplaced*. Objetos marcados como *misplaced* foram atribuídos em grupos que possuem objetos de uma classe diferente da sua.

Figura 4.3: Valores médios da métrica CMM Misplaced nos conjuntos de dados artificiais com *outliers* - Nível Macro e Micro



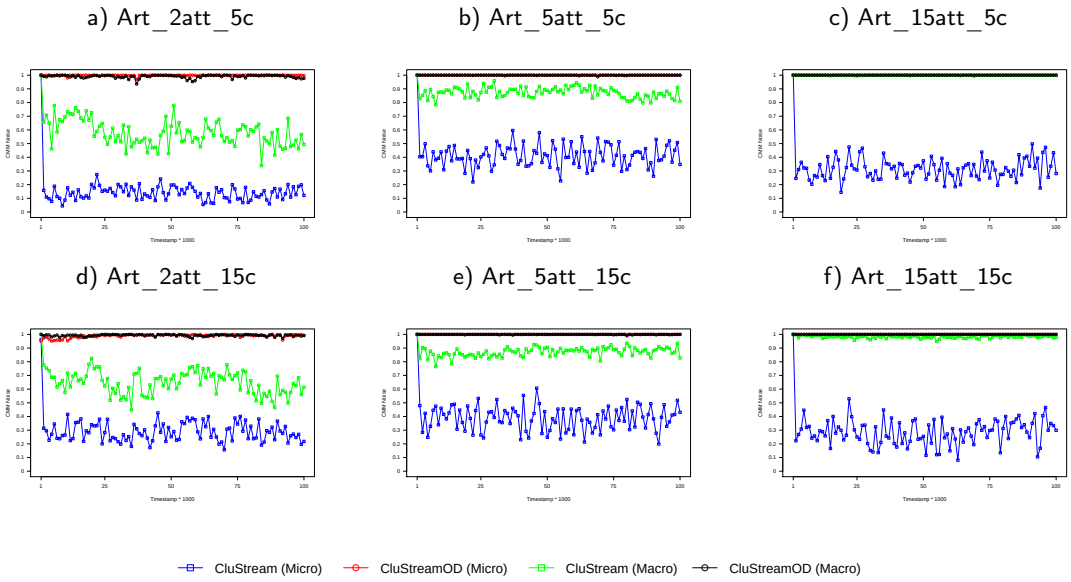
Percebe-se que quanto menor o número de grupos e atributos (Figura 4.3a), maior o índice desse erro no *CluStream*. O aumento do número de atributos e grupos leva a uma equivalência quase que completa entre ambos os algoritmos (Figura 4.3f), pois o aumento no número de atributos dos objetos do fluxo tornam-os mais dispersos, em especial quando considerado o espaço de característica dos mesmos.

4.4.4 Métrica CMM Noise - Nível Macro e Micro

A análise da componente *Noise* é a mais importante para validar a proposta apresentada nesse estudo. Analisando-se a Figura 4.4, pode-se perceber a superioridade, ou seja, os altos valores apresentados pelo algoritmo *CluStreamOD* para todos os conjuntos em estudo, a nível macro e micro. O erro *noise* evidencia a atribuição de um objeto *noise/outlier* a um micro-grupo/grupo. Tal diferença é mais evidente nos conjuntos com menor quantidade de atributos (Figuras 4.4a e 4.4d). A medida que o número de atributos aumenta, os objetos se tornam mais dispersos em seu espaço

de característica, facilitando a deteção dos mesmos no fluxo, conforme apresentado pela Figura 4.4f.

Figura 4.4: Valores médios da métrica CMM Noise nos conjuntos de dados artificiais com *outliers* - Nível Macro e Micro



4.4.5 Métrica Rand Ajustado - Nível Macro e Micro

Analisando-se a Figura 4.5 percebe-se uma menor diferença entre os agrupamentos/micro-grupos comparados para os conjuntos artificiais com menor quantidade de atributos (Figuras 4.5a e 4.5d). A medida que o valor de atributos aumenta, os valores de Rand apresentam maior variação e valores resultantes menores, demonstrando que os agrupamentos ou micro-grupos gerados por ambos os algoritmos se tornam mais discrepantes.

O algoritmo *CluStream* apresenta melhores resultados nos conjuntos Art_2att_5c e Art_2att_15c (Figuras 4.5a e 4.5d, respectivamente), onde o mesmo apresenta valores superiores em alguns *timestamps* e equivalentes em outros. A presença de objetos *inliers* na memória auxiliar que ainda não foram validados e inseridos no modelo, influenciam nos resultados gerados pelo índice Rand Ajustado também.

4.4.6 Métrica Silhueta - Nível Macro e Micro

Analisando-se a Figura 4.6 percebe-se que para quase todos os cenários em estudo, os melhores valores do índice Silhueta estão relacionados ao algoritmo *CluStreamOD* a nível macro.

Quando analisado o comportamento do algoritmo *CluStreamOD* a nível micro, que apresenta maior riqueza de detalhes sobre os micro-grupos do modelo, percebe-se

Figura 4.5: Valores médios da métrica Rand Ajustado nos conjuntos de dados artificiais com *outliers* - Nível Macro e Micro

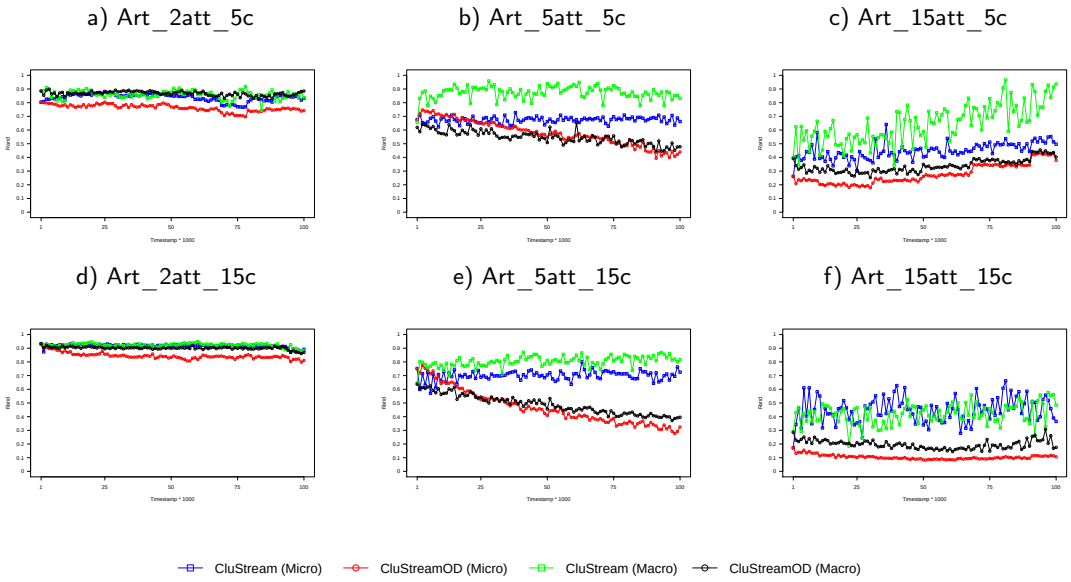
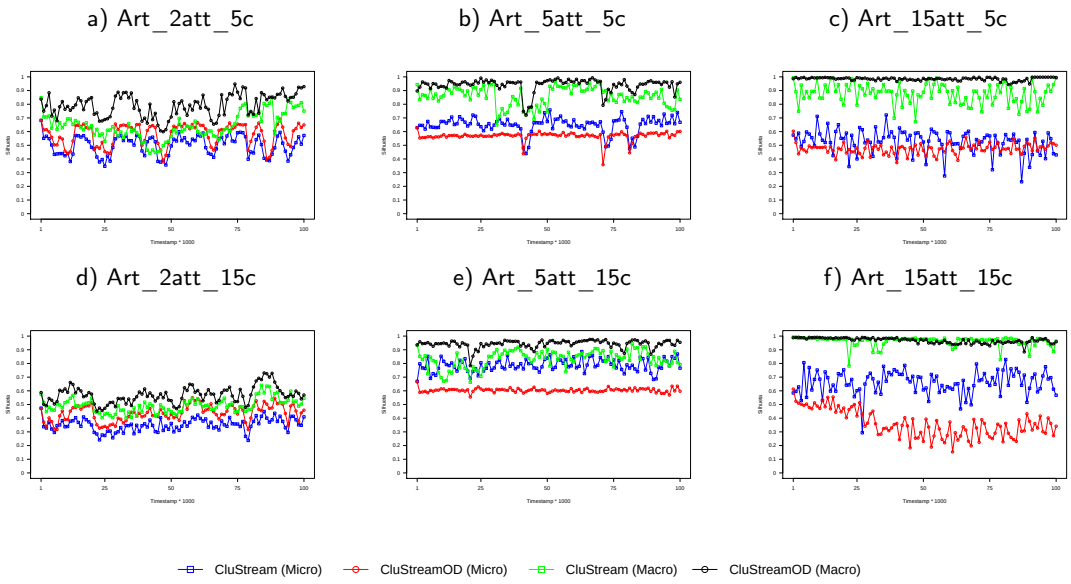


Figura 4.6: Valores médios da métrica Silhueta nos conjuntos de dados artificiais com *outliers* - Nível Macro e Micro



valores em alguns conjuntos de dados em estudo inferiores ao *CluStream* (ver Figuras 4.6b, 4.6c, 4.6e e 4.6f). A metodologia abordada pelo arcabouço proposto visa a criação de micro-grupos que tenham pelo menos uma quantidade mínima de pontos, através do processo de validação interna de M_{aux}, resultando em micro-grupos possivelmente maiores. Logo, em cenários que apresentam aumento da quantidade de atributos, os objetos são mais dispersos no espaço de características e micro-grupos maiores apresentam baixa coesão e dissimilaridade.

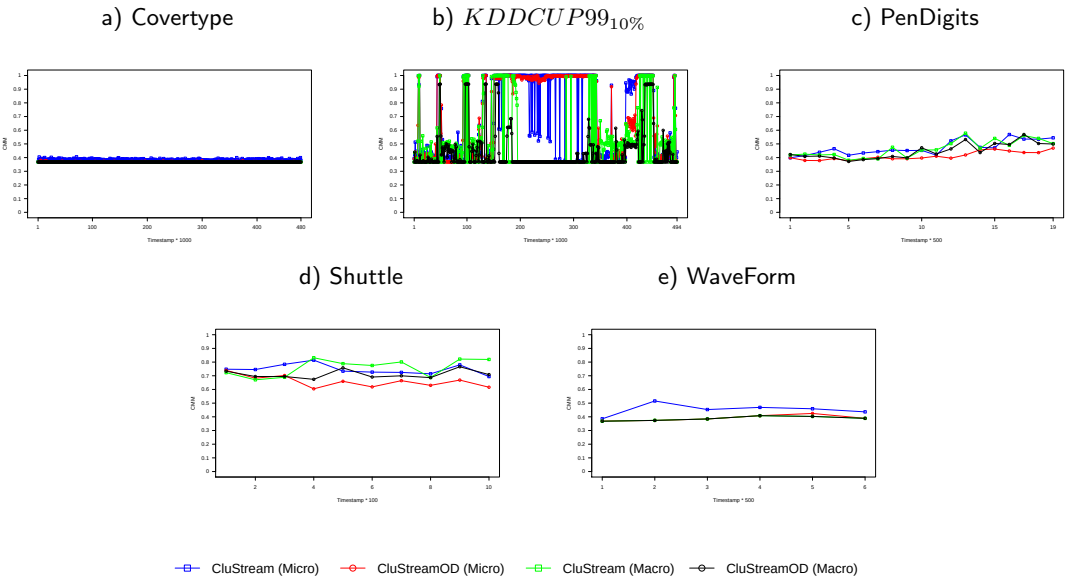
4.5 Avalição dos Conjuntos de Dados Reais: Nível Macro e Micro agrupamento

Uma avaliação a nível macro e micro-agrupamento dos conjuntos reais selecionados nesse estudo será apresentada a seguir para cada métrica selecionada.

4.5.1 Métrica CMM - Nível Macro e Micro

A Figura 4.7 apresenta uma análise a respeito dos valores médios da métrica CMM a nível macro e micro para os conjuntos de dados reais em estudo. Conforme apresentado na Seção 4.3 e descrito por Kremer et al. [2011], uma análise sobre a métrica CMM é mais eficaz em uma de suas componentes. Porém, pode-se perceber que a maioria dos conjuntos em análise, como por exemplo o *Covertype* (Figura 4.7a) apresenta desempenho quase que equivalente ao longo do tempo para ambos os algoritmos. Já o conjunto $KDDCUP99_{10\%}$ (Figura 4.7b) apresenta grande variação de desempenho ao longo do tempo para ambos os algoritmos, tanto a nível macro quanto a micro-agrupamento. O conjunto *Shuttle* (Figura 4.7d) apresenta desempenho superior do *CluStream* a nível macro e micro agrupamento. O conjunto *WaveForm* (Figura 4.7e) e *PenDigits* (Figura 4.7c) apresenta um desempenho superior por parte do algoritmo *CluStream* a nível macro agrupamento e a nível micro, ambos os algoritmos apresentam desempenho quase que equivalente. Tais variações serão verificadas e identificadas na análise sobre as componentes da métrica *CMM* apresentadas a seguir.

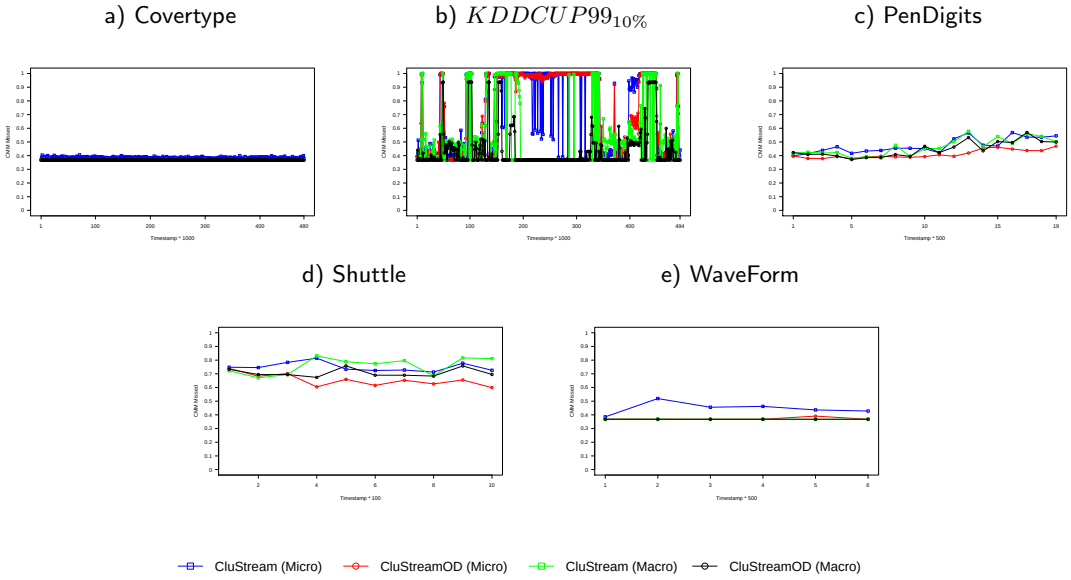
Figura 4.7: Valores médios da métrica CMM nos conjuntos de dados reais com e sem *outliers* - Nível Macro e Micro



4.5.2 Métrica CMM Missed - Nível Macro e Micro

A Figura 4.8 apresenta uma análise acerca dos valores médios da métrica CMM *Missed* a nível macro e micro para os conjuntos de dados reais. Uma análise sobre a componente CMM *Missed* permite verificar que muitos dos micro-grupos e grupos dos conjuntos de dados em estudo tiveram objetos marcados como *missed* ao longo do tempo, acarretando em uma alta taxa de erro para essa medida. No *CluStreamOD*, os objetos marcados como *missed* estão na memória auxiliar, esperando tornar-se válidos e serem inseridos no modelo. Porém, no *CluStream*, esses objetos foram removidos do modelo para dar espaço a novos micro-grupos.

Figura 4.8: Valores médios da métrica CMM *Missed* nos conjuntos de dados reais com e sem *outliers* - Nível Macro e Micro



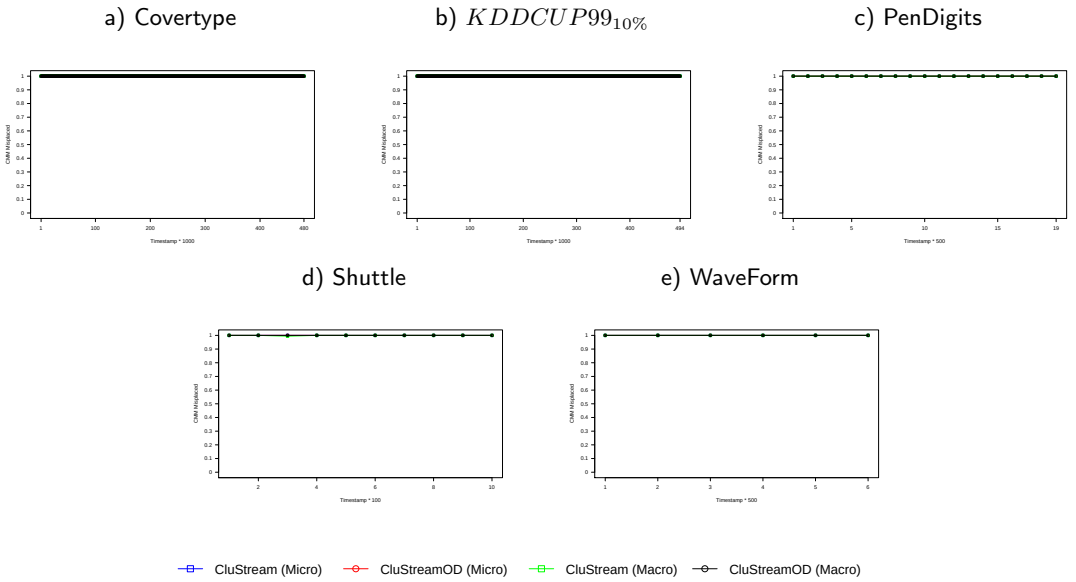
O conjunto $KDDCUP99_{10\%}$ (Figura 4.8b) apresenta uma grande variação de desempenho ao longo do tempo para ambos os algoritmos, tanto a nível macro quanto a micro-agrupamento, devido a distribuição esparsa de seus objetos [Kremer et al., 2011]. O *Covertype* apresenta valores quase que equivalentes ao *CluStream* (Figura 4.8a) ao longo do tempo, sendo a nível micro o *CluStream* apresenta uma sutil superioridade e ao nível micro, o *CluStreamOD* apresenta ser superior. Tal característica também é apresentada pelo *PenDigits* (Figura 4.8c) e pelo nível micro do *WaveForm* (Figura 4.8e). Para o conjunto *Shuttle* (Figura 4.8d), em ambas as análises, o algoritmo *CluStream* apresenta desempenho quase sempre superior ao *CluStreamOD*.

Por fim, a análise da Figura 4.7 permite inferir a dificuldade apresentada por ambos os algoritmos para absorver os objetos do fluxo pelos micro-grupos do modelo. Isso se deve as características dos objetos do conjunto possuírem uma alta variação, mesmo para objetos de uma mesma classe.

4.5.3 Métrica CMM Misplaced - Nível Macro e Micro

Analisando os valores médios apresentados para a componente CMM *Misplaced* na Figura 4.9, pode-se perceber que nenhum objeto dos conjuntos reais em estudo foi marcado como *misplaced*. Dessa forma, todos os conjuntos de dados apresentados possuem valor máximo (que é 1) para essa componente, evidenciando erro zero.

Figura 4.9: Valores médios da métrica CMM *Misplaced* nos conjuntos de dados reais com e sem *outliers* - Nível Macro e Micro

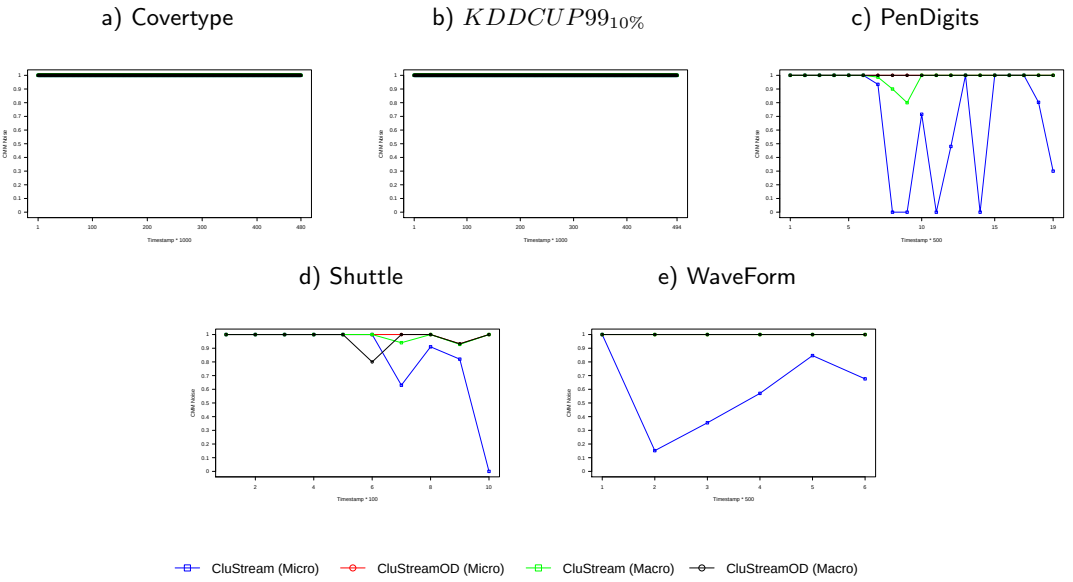


4.5.4 Métrica CMM Noise - Nível Macro e Micro

A Figura 4.10 apresenta os valores médios da métrica CMM *Noise* a nível macro e micro para os conjuntos de dados reais. A componente *noise* é a mais importante para o trabalho em questão, visto que a mesma é capaz de quantificar o impacto da inclusão de objetos *outliers* em micro-grupos do modelo. Dessa forma, a componente em análise nos permite verificar o desempenho da estratégia proposta nesse estudo, na detecção e tratamento desses objetos.

Como apresentado na Figura 4.10 pelos conjuntos reais com *outliers*, o *CluStreamOD* mantém desempenho superior ao *CluStream* ao longo do tempo para os conjuntos *PenDigits* e *WaveForm* (Figuras 4.10c e 4.10e respectivamente). Isso demonstra que o uso da estratégia proposta é válida para detectar e tratar *outiers* presentes no fluxo. O conjunto *Shuttle* (Figura 4.10d) apresenta uma pequena perda de desempenho por parte da proposta em estudo. Verificando-se a taxa de *Missed* apresentada pelo mesmo conjunto, pode-se perceber que os objetos que o compõe apresentam mais similares entre si, o que evita a perca desses, mas dificulta o processo de deteção e tratamento dos *outliers*.

Figura 4.10: Valores médios da métrica CMM *Noise* nos conjuntos de dados reais com e sem *outliers* - Nível Macro e Micro



4.5.5 Métrica Rand Ajustado - Nível Macro e Micro

Os valores médios de Rand Ajustado apresentados na Figura 4.11 são muito próximos em vários momentos do fluxo para os conjuntos de dados em análise *Covertype* (Figura 4.11a) e *WaveForm* (Figura 4.11e). O conjunto $KDDCUP99_{10\%}$ (Figura 4.11b) apresenta alta variabilidade a nível macro e micro agrupamento para essa métrica. O conjunto *PenDigits* (Figura 4.11c) apresenta uma sutil superioridade do *CluStream* a nível macro e o conjunto *Shuttle* (Figura 4.11d) apresentam uma sutil diferença de superioridade do algoritmo *CluStreamOD* a nível macro e micro agrupamento. Tais resultados demonstram que a validação interna aplicada sobre a memória auxiliar é dificultada pela característica esparsa dos objetos desse conjunto, fazendo com que poucos objetos *inliers* presentes na mesma retornem ao modelo.

4.5.6 Métrica Silhueta - Nível Macro e Micro

A Figura 4.12 apresenta os valores médios do índice Silhueta a nível macro e micro para os conjuntos de dados reais. Analisando-se os valores médios apresentados pela métrica Silhueta, pode-se novamente perceber a superioridade parcial, do *CluStreamOD* ao longo dos cenários em estudo. De forma semelhante a análise da métrica Rand Ajustado, também nesse caso o $KDDCUP99_{10\%}$ (Figura 4.12b) apresenta altas taxas de variação e valores inferiores ao *CluStreamOD* em alguns *timestamps*. Esse cenário também pode ser observado em dois *timestamps* do *Shuttle* (Figura 4.12d). Os resultados apresentados demonstram a capacidade do algoritmo *CluStreamOD* em gerar micro-grupos com maior coesão interna e maior dissimilaridade inter-grupos, características desejáveis para o cenário de agrupamento.

Figura 4.11: Valores médios do índice Rand Ajustado nos conjuntos de dados reais com e sem *outliers* - Nível Macro e Micro

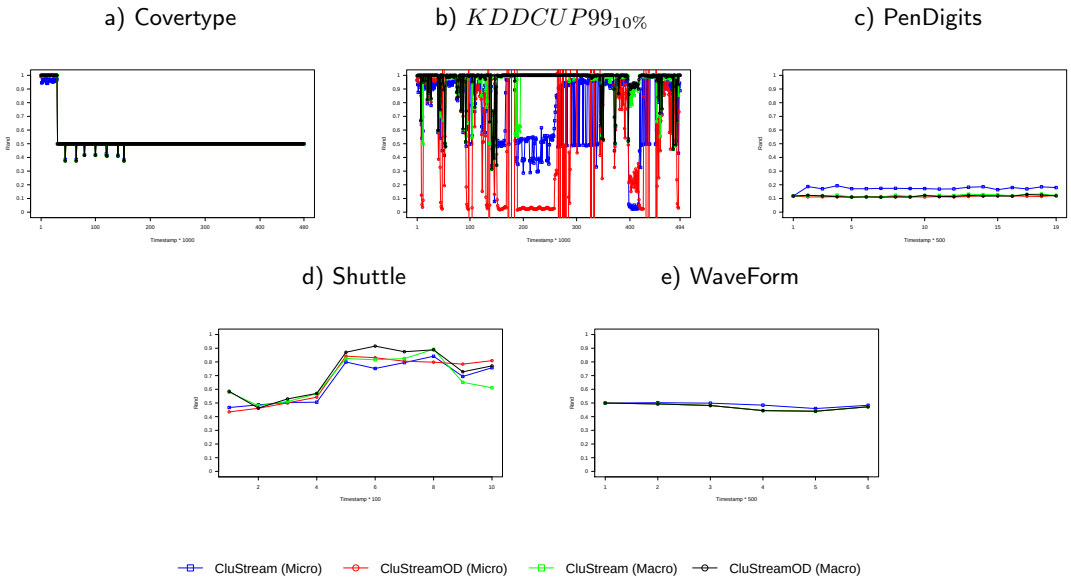
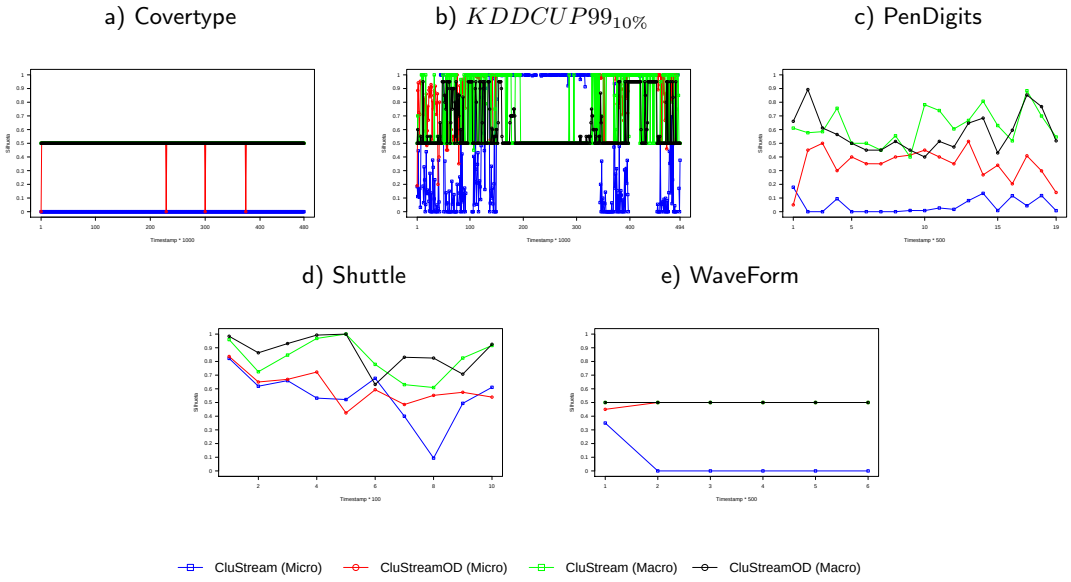


Figura 4.12: Valores médios do índice Silhueta nos conjuntos de dados reais com e sem *outliers* - Nível Macro e Micro



4.6 Proporções de objetos inseridos em M_{aux} nos conjuntos de dados com *outliers*

Objetos *inliers* que representam padrões novidades presentes no fluxo, tão bem como objetos *outliers*, são inseridos na memória auxiliar M_{aux} para serem avaliados. Dessa forma, os *inliers* devem ser movidos para o modelo M e os *outliers* devem ser mantidos em M_{aux} até expirarem e serem removidos. De forma a revalidar a

efetividade do mecanismo proposto nesse estudo, a Tabela 4.3 apresenta a proporção de objetos *inliers* e *outliers* que são inseridos e movidos de M_{aux}. Os valores das médias e desvio padrão (em parênteses) sobre as 10 execuções, variam em um intervalo de [0,1], onde 0 significa nenhum e 1 significa todos.

Considerando a Tabela 4.3, a segunda e quarta coluna (*Inliers em* M_{aux}) e (*Outliers em* M_{aux}), respectivamente, correspondem a proporção de objetos *inliers* e *outliers* que foram inseridos em M_{aux} ao longo do FCDs. A terceira e quarta coluna (*Inliers fora* M_{aux}) e (*Outliers fora* M_{aux}), respectivamente, correspondem a proporção de objetos *inliers* e *outliers* que foram inseridos em M, no formato de micro-grupo, por terem sido considerados válidos pelos quatro subprocessos de análise de M_{aux}.

Tabela 4.3: Valores proporcionais de média e desvio padrão dos objetos inseridos e movidos de M_{aux}.

Conjuntos de Dados	**Inliers em** M_{aux}	**Inliers fora** M_{aux}	**Outliers em** M_{aux}	**Outliers fora** M_{aux}
PenDigits	0,842 (0,15)	0,9305 (0,05)	0,7112 (0,45)	0 (0)
Shuttle	0,7414 (0,17)	0,9160 (0,04)	0,4945 (0,50)	0,0901 (0,07)
WaveForm	0,6735 (0,18)	0,9342 (0,09)	0,9655 (0,18)	0,7309 (0,28)
Art_2att_5c	0,3994 (0,05)	0,9275 (0,05)	0,9528 (0,00)	0,0685 (0,00)
Art_5att_5c	0,6278 (0,09)	0,8791 (0,05)	1 (0)	0 (0)
Art_15att_5c	0,9287 (0,09)	0,9664 (0,01)	1 (0)	0 (0)
Art_2att_15c	0,4064 (0,05)	0,9240 (0,05)	0,8913 (0,01)	0,2056 (0,01)
Art_5att_15c	0,7346 (0,10)	0,8290 (0,06)	1 (0)	0 (0)
Art_15att_15c	0,9415 (0,09)	0,9303 (0,02)	1 (0)	0 (0)

No melhor cenário, 39% dos objetos *inliers* do conjunto Art_2att_5c são inseridos em M_{aux} e 92% desses objetos são movidos para o modelo M pelo processo de validação interna de M_{aux}. No pior caso, o conjunto Art_15att_15c inclui em M_{aux} cerca de 94% de objetos *inliers*, que não se torna um problema pois 93% desses objetos são em um segundo momento movidos para M. Esse comportamento é observado também nos conjuntos de dados restantes, indicando que a informação de muitos dos *inliers* que compõe o fluxo é ainda usada para induzir o modelo.

Em relação aos objetos *outliers* do FCDs, os mesmos não deveriam se assemelhar com nenhum dos micro-grupos do modelo e, dessa forma, serem inseridos em M_{aux}. Analisando-se se a Tabela 4.3, mais que 90% dos *outliers* são inseridos em M_{aux} em seis dos 9 conjuntos de dados analisados nesse trabalho que possuem *outliers*. Nos piores cenários, 49% dos *outliers* do conjunto *Shuttle* são selecionados e inseridos em M_{aux}. Considerando-se um exame mais detalhado acerca dos objetos *outliers* do conjunto *Shuttle*, verificou-se que os mesmos se assemelham aos objetos *inliers*. Tal característica permite que esse objetos sejam então absorvidos pelos micro-grupos do modelo. Considerando-se os objetos *outliers* em M_{aux}, 9% ou menos desses são movidos para o modelo M em sete dos nove conjuntos de dados analisados. O pior cenário é apresentado pelo conjunto $WaveForm$, no qual os objetos são tão dissimilares uns dos outros que 96% dos *outliers* são inseridos em M_{aux}, mas 73% são movidos posteriormente ao modelo M.

Com base nos experimentos apresentados, foi possível verificar o arcabouço proposto para ser aplicado a componente *online* de algoritmos de agrupamento em

FCDs, para detecção de *outliers* foi eficaz para a maioria dos cenários em estudo. Tal afirmação é fundamentada nos resultados apresentados principalmente pela componente *CMM Noise* e a Tabela 4.3 que apresenta a proporção média de retorno dos objetos *inliers* ao Modelo M, no formato de micro-grupos.

4.7 Análise de complexidade do algoritmo *CluStreamOD*

A análise de complexidade do algoritmo *CluStreamOD* foi realizada para ambas as componentes do algoritmo, *online* e *offline*, baseada no tempo.

Segundo Silva et al. [2013], considerando-se a componente *online* do *CluStream* [Aggarwal et al., 2003], o custo para criação de micro-grupos é de $O(q \cdot N_{first} \cdot n \cdot v)$, onde q é o número de micro-grupos, N_{fist} é o número de objetos usados para criar os micro-grupos iniciais, n é a dimensionalidade dos objetos e v é o número de iterações do k-médias. Em seguida, para cada novo objeto que chega ao fluxo, a atualização dos micro-grupos do modelo requerem três passos: i) encontrar o micro-grupo mais próximo, que leva $O(q \cdot n)$ tempos; ii) possivelmente descartar o micro-grupo mais antigo, que requer $O(q)$ tempos; e iii) possivelmente unir os dois micro-grupos mais próximos, que requer $O(q^2 \cdot n)$. A componente *offline* executa o tradicional algoritmo de agrupamento k-médias sobre os micro-grupos do modelo, que requer $O(q \cdot n \cdot k \cdot v)$ tempos, onde k é o número de grupos.

O algoritmo *CluStreamOD* apresenta complexidade em função do tempo para as componentes *online* e *offline* similar ao *CluStream*, considerando que o algoritmo de agrupamento utilizado é o k-médias++. Porém, o *CluStreamOD* gerencia a memória auxiliar, no qual o custo assintótico desse processo é $O(|M_{aux}| \cdot k \cdot n \cdot v)$, onde $|M_{aux}|$ é a cardinalidade da memória M_{aux}.

Capítulo 5

CONCLUSÕES E TRABALHOS FUTUROS

Com base nos experimentos apresentados neste trabalho, foi possível verificar que o arcabouço proposto e aplicado a componente *online* do algoritmo *CluStream* para detecção e validação de *outliers* é eficaz para a maioria dos cenários em estudo. Nesse estudo, valores padrões foram aplicados a todos os conjuntos de dados, a fim de permitir a validação da proposta em estudo com entradas padronizadas.

Os micro-grupos resultantes do *CluStreamOD* apresentam significativamente menos *outliers* que o algoritmo *CluStream* original, sendo mais coesos e melhor definidos ao longo do fluxo. O impacto do método proposto resulta, nos melhores casos, em uma melhoria de 100% sobre o *CluStream* para alguns conjuntos de dados artificiais e, no pior cenário, em uma melhoria de aproximadamente 49% na detecção de *outliers*. Adicionalmente, *CluStreamOD* foi capaz de retornar para o modelo a maioria dos objetos *inliers* que foram previamente considerados *outliers* pela análise e inseridos na memória auxiliar. Tal característica é especialmente importante para conjuntos de dados sem *outliers* (como *Covertype* e $KDDCUP99_{10\%}$) nos quais o algoritmo apresenta performance superior ou equivalente ao *CluStream*.

Considerando a complexidade assintótica apresentada pelo arcabouço desenvolvido e os resultados apresentados pelas métricas de validação internas e externas, conclui-se que o método proposto possui potencial para ser aplicado em novos algoritmos de agrupamento. O mesmo possui efetividade tanto em conjuntos de dados sem *outliers*, como em conjuntos com *outliers*. Com relação a complexidade do algoritmo, o mesmo apresenta complexidade inferior ou equivalente a outros algoritmos que possuem um processo de detecção de *outliers* incorporado à componente *online*, como por exemplo, o *DenStream* [Cao et al., 2006], [Silva et al., 2013].

O tratamento de *outliers* em conjunto com a componente *online* no algoritmo *CluStream*, garante que o modelo gerado se mantenha constantemente válido e elimina o problema citado por Cao et al. [2006]. No qual, os autores descrevem como sendo uma ação arriscada em FCDs com a presença de *outliers*, a forma como o algoritmo realiza a manutenção dos micro-grupos do modelo.

Tais resultados validam os objetos desse estudo e levam ao desenvolvimento de futuros trabalhos como por exemplo: i) a verificação de alternativas para a redu-

ção do número de parâmetros de entrada e eliminação dos não críticos; ii) executar uma análise de parametrização, com a intenção de verificar os melhores intervalos de valores para os conjuntos de dados em estudo (como por exemplo o conjunto $WaveFomr$); iii) aplicação do arcabouço proposto em outros algoritmos para agrupamento de FCDs, a fim de verificar a performance dos mesmos quanto à detecção de *outliers*; iv) comparar a abordagem proposta com algoritmos de agrupamento em FCDs que já realizam um processo de detecção de *outliers* (como por exemplo o *DenStream*); e v) verificar estruturas capazes de otimizar a complexidade da proposta em estudo em função do tempo.

Referências Bibliográficas

Ackermann, M. R.; Märtens, M.; Raupach, C.; Swierkot, K.; Lammersen, C. & Sohler, C. (2012). Streamkm++: A clustering algorithm for data streams. *J. Exp. Algorithmics*, 17:2.4:2.1--2.4:2.30.

Aggarwal, C. C. (2003). A framework for diagnosing changes in evolving data streams. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, SIGMOD '03, pp. 575--586, New York, NY, USA. ACM.

Aggarwal, C. C. (2015). Outlier analysis. In *Data Mining*, pp. 237--263. Springer, Springer.

Aggarwal, C. C.; Han, J.; Wang, J. & Yu, P. S. (2003). A framework for clustering evolving data streams. In *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*, VLDB '03, pp. 81--92. VLDB Endowment.

Angiulli, F. & Fassetti, F. (2010). Distance-based outlier queries in data streams: the novel task and algorithms. *Data Mining and Knowledge Discovery*, 20(2):290--324.

Arthur, D. & Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1027--1035. Society for Industrial and Applied Mathematics, Society for Industrial and Applied Mathematics.

Babcock, B.; Babu, S.; Datar, M.; Motwani, R. & Widom, J. (2002). Models and issues in data stream systems. In *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '02, pp. 1--16, New York, NY, USA. ACM.

Babcock, B.; Datar, M.; Motwani, R. & O'Callaghan, L. (2003). Maintaining variance and k-medians over data stream windows. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 234--243. ACM.

Bifet, A.; Holmes, G.; Pfahringer, B.; Kirkby, R. & Gavaldà, R. (2009). New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pp. 139--148, New York, NY, USA. ACM.

Bifet, A. & Kirkby, R. (2009). Data stream mining a practical approach.

Breunig, M. M.; Kriegel, H.-P.; Ng, R. T. & Sander, J. (2000). Lof: identifying density-based local outliers. In *ACM sigmod record*. ACM, ACM.

Campos, G. O.; Zimek, A.; Sander, J.; Campello, R. J.; Micenková, B.; Schubert, E.; Assent, I. & Houle, M. E. (2016). On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. *Data Mining and Knowledge Discovery*, 30(4):891--927.

Cao, F.; Ester, M.; Qian, W. & Zhou, A. (2006). Density-based clustering over an evolving data stream with noise. In *In 2006 SIAM Conference on Data Mining*, pp. 328--339. SIAM.

Chang, J. H. & Lee, W. S. (2004). Decaying obsolete information in finding recent frequent itemsets over data streams. *IEICE transactions on information and systems*, 87(6):1588--1592.

Chang, J. H. & Lee, W. S. (2005). estwin: Online data stream mining of recent frequent itemsets by sliding window method. *Journal of Information Science*, 31(2):76--90.

Chen, M.-S.; Han, J. & Yu, P. S. (1996). Data mining: An overview from a database perspective. *IEEE Trans. on Knowl. and Data Eng.*, 8(6):866--883.

Chen, Y. & Tu, L. (2007). Density-based clustering for real-time stream data. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 133--142. ACM.

Chi, Y.; Wang, H.; Philip, S. Y. & Muntz, R. R. (2006). Catch the moment: maintaining closed frequent itemsets over a data stream sliding window. *Knowledge and Information Systems*, 10(3):265--294.

Domingos, P. & Hulten, G. (2000). Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 71--80. ACM.

Elahi, M.; Li, K.; Nisar, W.; Lv, X. & Wang, H. (2008). Efficient clustering-based outlier detection algorithm for dynamic data stream. In *Fuzzy Systems and Knowledge Discovery, 2008. FSKD'08. Fifth International Conference on*, volume 5, pp. 298--304. IEEE, IEEE.

Ester, M.; Kriegel, H.-P.; Sander, J.; Xu, X. et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pp. 226--231.

Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine learning*, 2(2):139--172.

Gaber, M. M.; Zaslavsky, A. & Krishnaswamy, S. (2010). Data stream mining. *Data Mining and Knowledge Discovery Handbook*, pp. 759--787.

Gama, J. (2010). *Knowledge discovery from data streams*. CRC Press.

Gama, J.; Medas, P.; Castillo, G. & Rodrigues, P. (2004). Learning with drift detection. In *Brazilian Symposium on Artificial Intelligence*, pp. 286--295. Springer.

Gama, J.; Rodrigues, P. P. & Lopes, L. (2011). Clustering distributed sensor data streams using local processing and reduced communication. *Intelligent Data Analysis*, 15(1):3--28.

Giannella, C.; Han, J.; Pei, J.; Yan, X. & Yu, P. S. (2003). Mining frequent patterns in data streams at multiple time granularities. *Next generation data mining*, 212:191--212.

Gonçalves, M. L.; Netto, M.; Zullo Jr, J. & Costa, J. A. F. (2008). Classificação não-supervisionada de imagens de sensores remotos utilizando redes neurais auto-organizáveis e métodos de agrupamentos hierárquicos. *Revista Brasileira de Cartografia*, 60(1):17--29.

Guha, S.; Meyerson, A.; Mishra, N.; Motwani, R. & O'Callaghan, L. (2003). Clustering data streams: Theory and practice. *IEEE Trans. on Knowl. and Data Eng.*, 15(3):515--528.

Hulten, G.; Spencer, L. & Domingos, P. (2001). Mining time-changing data streams. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pp. 97--106, New York, NY, USA. ACM.

Isaksson, C.; Dunham, M. H. & Hahsler, M. (2012). Sostream: Self organizing density-based clustering over data stream. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pp. 264--278. Springer.

Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651--666.

Jain, A. K. & Dubes, R. C. (1988). *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Kaufman, L. & Rousseeuw, P. J. (2009). *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons.

Kontaki, M.; Gounaris, A.; Papadopoulos, A. N.; Tsichlas, K. & Manolopoulos, Y. (2011). Continuous monitoring of distance-based outliers over data streams. In *2011 IEEE 27th International Conference on Data Engineering*, pp. 135--146. IEEE, IEEE.

Kranen, P.; Assent, I.; Baldauf, C. & Seidl, T. (2011). The clustree: indexing micro-clusters for anytime stream mining. *Knowledge and information systems*, 29(2):249--272.

Kremer, H.; Kranen, P.; Jansen, T.; Seidl, T.; Bifet, A.; Holmes, G. & Pfahringer, B. (2011). An effective evaluation measure for clustering on evolving data streams. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 868--876. ACM, ACM.

Lichman, M. (2013). UCI machine learning repository.

Liu, X.; Guan, J. & Hu, P. (2009). Mining frequent closed itemsets from a landmark window over online data streams. *Computers & Mathematics with Applications*, 57(6):927--936.

Lloyd, S. (1982). Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129--137.

MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pp. 281--297. Oakland, CA, USA.

Manku, G. S. & Motwani, R. (2002). Approximate frequency counts over data streams. In *Proceedings of the 28th international conference on Very Large Data Bases*, pp. 346--357. VLDB Endowment.

Masud, M.; Gao, J.; Khan, L.; Han, J. & Thuraisingham, B. M. (2011). Classification and novel class detection in concept-drifting data streams under time constraints. *IEEE Transactions on Knowledge and Data Engineering*, 23(6):859--874.

Miglierina, M. & Di Nitto, E. (2017). *Monitoring in a Multi-cloud Environment*, pp. 47--52. Springer International Publishing, Cham.

Mohammed, S. A. S. A.-N.; Muhammed, D. J. et al. (2017). Financial crisis, legal origin, economic status and multi-bank performance indicators. *Journal of Applied Accounting Research*, 18:208--222.

Muthukrishnan, S. et al. (2005). Data streams: Algorithms and applications. *Foundations and Trends® in Theoretical Computer Science*, 1(2):117--236.

Nguyen, D. T. & Jung, J. E. (2017). Real-time event detection for online behavioral analysis of big social data. *Future Generation Computer Systems*, 66:137 – 145.

Nguyen, H.-L.; Woon, Y.-K. & Ng, W.-K. (2015). A survey on data stream clustering and classification. *Knowledge and Information Systems*, 45(3):535--569.

O'callaghan, L.; Mishra, N.; Meyerson, A.; Guha, S. & Motwani, R. (2002). Streaming-data algorithms for high-quality clustering. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pp. 685--694. IEEE.

Oza, N. C. & Russell, S. (2001). Online bagging and boosting. In *In Artificial Intelligence and Statistics 2001*, pp. 105--112. Morgan Kaufmann.

Paiva, E. R. d. F. (2014). *Detecção de novidade em fluxos contínuos de dados multiclasse*. PhD thesis, Universidade de São Paulo.

Pfahringer, B.; Holmes, G. & Kirkby, R. (2007). New options for hoeffding trees. In *Proceedings of the 20th Australian Joint Conference on Advances in Artificial Intelligence*, AI'07, pp. 90--99, Berlin, Heidelberg. Springer-Verlag.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846--850.

Ren, J. & Ma, R. (2009). Density-based data streams clustering over sliding windows. In *Fuzzy Systems and Knowledge Discovery, 2009. FSKD'09. Sixth International Conference on*, volume 5, pp. 248--252. IEEE.

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53--65.

Safavian, S. R. & Landgrebe, D. (1991). A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660--674.

Salehi, M.; Leckie, C.; Bezdek, J. C. & Vaithianathan, T. (2015). Local outlier detection for data streams in sensor networks: Revisiting the utility problem invited paper. In *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2015 IEEE Tenth International Conference on*, pp. 1--6. IEEE, IEEE.

Silva, J. A.; Faria, E. R.; Barros, R. C.; Hruschka, E. R.; Carvalho, A. C. P. L. F. d. & Gama, J. a. (2013). Data stream clustering: A survey. *ACM Comput. Surv.*, 46(1):13:1--13:31.

Spinosa, E. J. (2008). *Detecção de novidade com aplicação a fluxos contínuos de dados*. PhD thesis, Universidade de São Paulo.

Steinwart, I. & Christmann, A. (2008). *Support vector machines*. Springer Science & Business Media.

Tan, P.-N.; Steinbach, M. & Kumar, V. (2009). *Introdução ao datamining: mineração de dados*. Ciência Moderna.

Vendramin, L.; Campello, R. J. G. B. & Hruschka, E. R. (2010). Relative clustering validity criteria: A comparative overview. *Statistical Analysis and Data Mining*, 3(4):209--235.

Yang, D.; Rundensteiner, E. A. & Ward, M. O. (2009). Neighbor-based pattern detection for windows over streaming data. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '09, pp. 529--540, New York, NY, USA. ACM.

Yeung, K. Y. & Ruzzo, W. L. (2001). Details of the adjusted rand index and clustering algorithms, supplement to the paper an empirical study on principal component analysis for clustering gene expression data. *Bioinformatics*, 17(9):763--774.

Zhang, Q.; Perra, N.; Perrotta, D.; Tizzoni, M.; Paolotti, D. & Vespignani, A. (2017). Forecasting seasonal influenza fusing digital indicators and a mechanistic disease model. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pp. 311--319, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Zhang, T.; Ramakrishnan, R. & Livny, M. (1997). Birch: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1(2):141--182.