Evolução da Semissupervisão em Detecção Online de Agrupamentos

Guilherme Alves da Silva

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

Uberlândia
2017

Guilherme Alves da Silva

Evolução da Semissupervisão em Detecção Online de Agrupamentos

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

Área de concentração: Ciência da Computação

Orientador: Maria Camila Nardini Barioni

Uberlândia
2017

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU, MG, Brasil.

S586e 2017

Silva, Guilherme Alves da, 1993-
Evolução da semissupervisão em detecção online de agrupamentos / Guilherme Alves da Silva. - 2017.
112 f. : il.

Orientadora: Maria Camila Nardini Barioni.
Dissertação (mestrado) -- Universidade Federal de Uberlândia, Programa de Pós-Graduação em Ciência da Computação.
Disponível em: http://dx.doi.org/10.14393/ufu.di.2017.65
Inclui bibliografia.

1. Computação - Teses. 2. Mineração de dados (Computação) – Teses. I. Barioni, Maria Camila Nardini. II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU: 681.3

Aos meus amados pais.
E aos mestres que me inspiraram e permitiram que eu chegasse até aqui.

Agradecimentos

Muitas pessoas colaboraram direta ou indiretamente em minha jornada até a finalização desta dissertação. Gostaria de agradecer especialmente:

À Profª. Maria Camila,
por ter assumido prontamente o compromisso da minha orientação após o afastamento da Profª. Sandra, pela confiança e pelo esforço em me guiar rumo a um novo tópico de pesquisa. Certamente, o caminho teria sido muito mais árduo (e doloroso) sem a ajuda dela. Meu especial muito obrigado!

À Profª. Sandra de Amo,
por ter inicialmente aceitado a orientação, mas principalmente pelo papel fundamental nos anos iniciais de formação, preparando a base para o mestrado e para a posteridade. Obrigado pelo exemplo de dedicação ao trabalho e competência como pesquisadora!

Aos amigos de laboratório, Crícia, Fabíola e Klérisson,
pelos conselhos, discussões, divergências (muitas é verdade!), mas principalmente pelo estímulo que deram e ainda dão sobre o mestrado e sobre a carreia. E também por ouvirem os meus desabafos!

Ao Marcos Luiz,
que mesmo estando no “velho continente"me proporcionou conversas e discussões de valor imensurável sobre o desenvolvimento deste trabalho, sobre a pesquisa científica em geral e a carreira acadêmica.

Aos meus amados pais,
pelo amor e apoio incondicional que me dão forças para seguir em frente.

Aos mestres (nos sentidos restrito e amplo) e amigos,
que desde os tempos pré-mestrado e durante o desenvolvimento deste trabalho me deram valiosos conselhos e indicaram caminhos essenciais para minha carreira. Em

especial, à *Profª. Maria Adriana, Prof. Luis Ortellado, Profª. Maria Amélia, Profª. Elaine Cardoso, Prof. Osmar, Pe. Rogério e Ivo Nunes.*

À Profª. Elaine,
pelas valiosas recomendações e correções. Essas contribuições foram extremamente importantes para aparar as arestas deste trabalho.

Ao Prof. Marcelo Zanchetta,
pelas reuniões improvisadas onde aprendi um pouco sobre extração de características em imagens.

Ao Erisvaldo,
pelo empenho em diversos momentos para sanar as dúvidas e o auxílio nos processos administrativos e burocráticos.

Aos amigos da "velha guarda", Alana, Bruno, Camila, Guilherme, Vinícius e Vitor,
pelo apoio e amizade, por serem a brisa refrescante em muitos momentos e também por compreenderem minhas inúmeras ausências nos últimos anos.

Aos vários amigos e colegas da Pós-Graduação,
que por diversas vezes fizeram meus dias mais alegres e leves. Em especial, à *Lucimeire, Mateus, Maicon e Régis.*

Aos membros da banca examinadora, por aceitarem o convite.

Adicionalmente, deixo registrado de maneira particular os seguintes reconhecimentos.

O apoio financeiro da *CAPES* na pesquisa desenvolvida neste trabalho durante 13 meses.

A disponibilização de recursos e infraestrutura pela *FACOM/UFU* para o desenvolvimento deste trabalho.

Finalmente, agradeço *à Deus*, por ter colocado essas e outras pessoas na minha vida.

"There is too much of me in it. A good portrait is a portrait of the artist, not the sitter."
(Oscar Wilde. Adaptação de The Picture of Dorian Gray)

Resumo

A disponibilidade abundante de dados torna inviável a busca manual por informações relevantes. Os métodos automáticos para organizar os dados, como a detecção de agrupamentos, podem ser úteis para ajudar nesta tarefa propiciando o acesso à informação desejada em tempo hábil. As abordagens de detecção semissupervisionada de agrupamentos empregam alguma informação adicional para guiar o processo baseado nos atributos dos dados de forma a obter uma organização mais próxima da desejada pelo usuário. Todavia, a informação extra pode mudar ao longo do tempo impondo uma mudança na maneira como os dados devem ser organizados. Para ajudar a lidar com esse problema, propõe-se o framework CABESS (*Cluster Adaptation Based on Evolving Semi-Supervision*), para detecção online de agrupamentos semissupervisionada. O framework é capaz de lidar com a evolução da semissupervisão obtida a partir de feedbacks binários do usuário. Para validar a abordagem, os experimentos foram executados sobre sete conjuntos de dados com rótulos baseados em hierarquia considerando a especialização e generalização dos agrupamentos ao longo do tempo. Os resultados experimentais mostram o potencial do framework proposto para lidar com a evolução da semissupervisão. Além disso, eles também mostram que o framework é mais rápido que os tradicionais algoritmos de detecção de agrupamentos semissupervisionados, mesmo usando um tipo pobre de especificação da semissupervisão.

Palavras-chave: Detecção semissupervisionada de agrupamentos. Evolução da semissupervisão. Detecção online de agrupamentos. Transição de agrupamentos. Mineração de dados.

Abstract

The huge amount of currently available data puts considerable constraints on the task of information retrieval. Automatic methods to organize data, such as clustering, can be used to help with this task allowing timely access. Semi-supervised clustering approaches employ some additional information to guide the clustering performed based on data attributes to a more suitable data partition. However, this extra information may change over time imposing a shift in the manner by which data is organized. In order to help cope with this issue, this dissertation proposes the framework called CABESS (Cluster Adaptation Based on Evolving Semi-Supervision), for online clustering. This framework is able to deal with evolving semi-supervision obtained through user binary feedbacks. To validate the approach, the experiments were run over seven hierarchical labeled datasets considering clustering splits and merges over time. The experimental results show the potential of the proposed framework for dealing with evolving semi-supervision. Moreover, they also show that the framework is faster than traditional semi-supervised clustering algorithms using lower standard semi-supervision.

Keywords: Semi-supervised clustering. Evolving semi-supervision. Online clustering. Cluster transition. Data mining.

Lista de ilustrações

Lista de tabelas

Lista de siglas

ARI	*Adjusted Rand Index*
BIRCH	*Balanced Iterative Reducing and Clustering using Hierarchies*
CABESS	*Cluster Adaptation Based on Evolving Semi-Supervision*
DBSCAN	*Density-Based Spatial Clustering of Applications with Noise*
CL	*Cannot-link*
ML	*Must-link*
SSDBSCAN	*Semi-Supervised DBSCAN*

Lista de símbolos

$\mathcal{D}$	Conjunto ou sequência de instâncias de dados
x_i	i-ésima instância $\in \mathcal{D}$
k	Número de grupos (*clusters*)
n	Número de instâncias
Ω	Espaço de atributos
d	Número de atributos
Π	Uma partição dos dados
Π_i	i-ésimo agrupamento da partição Π
$\delta(\cdot, \cdot)$	Distância entre duas instâncias
C	Conjunto de restrições
$C_\in$	Conjunto de pares de restrições *must-link*
$C_\notin$	Conjunto de pares de restrições *cannot-link*
$c_\in(\cdot, \cdot)$	Par de restrição *must-link* entre duas instâncias
$c_\notin(\cdot, \cdot)$	Par de restrição *cannot-link* entre duas instâncias
$\mathcal{S}$	Um *stream*
$\mathcal{F}$	Um *stream* de feedbacks

Sumário

Capítulo 1

Introdução

A característica marcante da natureza e da sociedade humana é sua intrínseca capacidade de evoluir e se adaptar dinamicamente diante de cenários complexos. O desenvolvimento da tecnologia foi agente ativo nas transformações e na organização de grupos e informações [Castells 2007]. O aparecimento das redes sociais, da computação ubíqua e de sistemas focados na facilidade do usuário (assistentes virtuais inteligentes[1], por exemplo) são, também, responsáveis pelo aumento vertiginoso da geração de dados, que refletem em tempo real a sociedade no ambiente virtual. Assim, faz-se necessário usar a tecnologia como agente de apoio para entender, acompanhar e prever o comportamento dessa projeção da sociedade no meio digital.

O imenso volume de informações, no entanto, é um problema, pois gera o fenômeno conhecido como "afogamento nos dados" que dificulta a geração de conhecimentos e *insights*. Então, a análise de agrupamentos, uma área de pesquisa em mineração de dados, surge com o propósito de agrupar dados que compartilham alguma, ou mais de uma, característica objetivando detectar padrões, tendências ou resumir os dados. Em determinados cenários, quando detém-se um conhecimento de domínio ou um subconjunto de dados cuja variável resposta é conhecida, pode-se utilizar esse tipo de informação adicional para guiar uma técnica de aprendizado de máquina. Técnicas que suportam esse tipo de informação adicional posicionam-se entre o aprendizado supervisionado e o não supervisionado e, então, constituem uma área de pesquisa denominada de aprendizagem semissupervisionada. Não obstante, se a temporalidade da informação precisar ser considerada, os modelos e algoritmos de mineração de dados *online* ou em fluxo (*stream*) são utilizados de modo a considerar a característica não estacionária e contínua da informação ao longo do tempo.

[1] Microsoft Cortana, Google Now e Apple Siri.

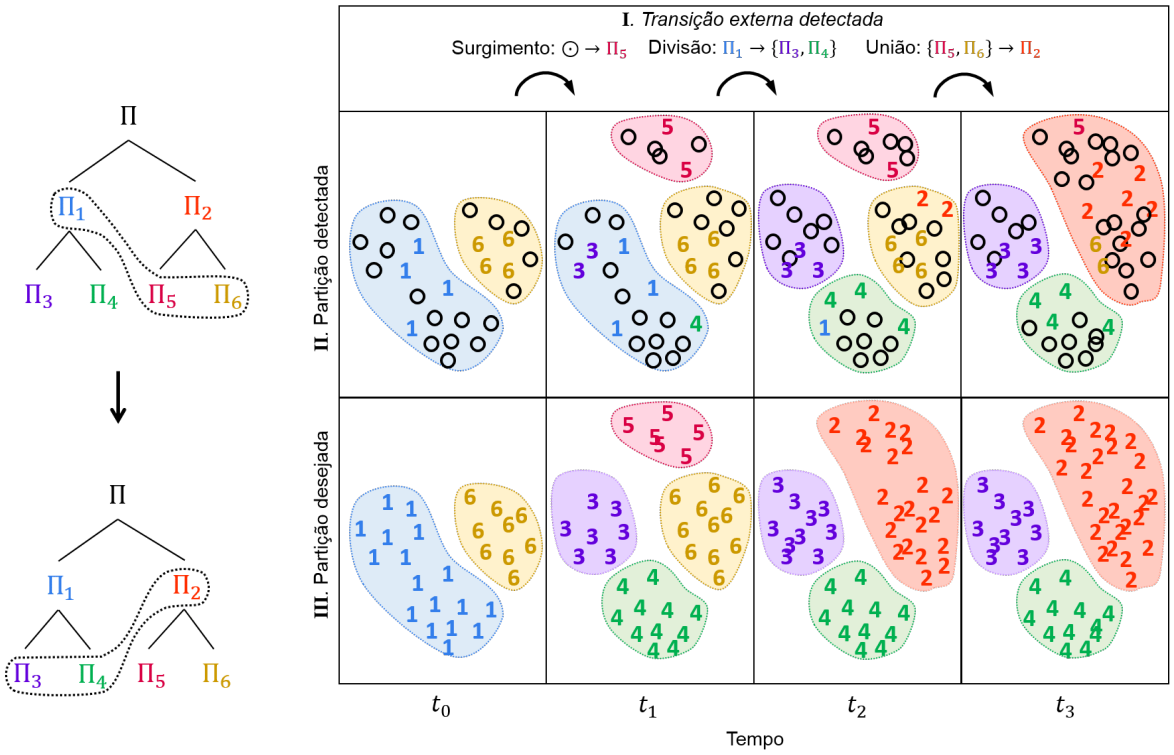
1.1 Motivação

Preferências e padrões podem mudar ao longo do tempo, ou seja, não são imutáveis. Considerando este cenário temporal e inconstante, trabalhos como [Spiliopoulou et al. 2006] e [Oliveira e Gama 2012] monitoram e analisam as mudanças em agrupamentos com o objetivo de entender a natureza da mudança e auxiliar a tomada de decisão. Além disso, a utilidade e qualidade da informação adicional usada para guiar o aprendizado semissupervisionado também tem sido investigada em alguns trabalhos. No entanto, há uma lacuna, ainda inexplorada, entre o rastreamento e a evolução de agrupamentos e as técnicas de semissupervisão.

A informação de semissupervisão pode auxiliar os algoritmos de detecção de agrupamentos a encontrar grupos que respeitem o conhecimento fornecido pelo usuário. Contudo, isso depende do conjunto utilizado para guiar o algoritmo, pois informações contraditórias podem, inclusive, degradar a qualidade do agrupamento [Davidson, Wagstaff e Basu 2006]. No entanto, as restrições também podem mudar ao longo do tempo, o que pode explicar mudanças nos agrupamentos: surgimento de grupos, alterações na forma espacial, entre outros. O trabalho descrito nesta dissertação é inspirado na ideia que a evolução do conjunto que fornece a informação de semissupervisão tem impacto nas alterações sofridas pelos agrupamentos ao longo do tempo.

A motivação para o trabalho descrito nesta dissertação é ilustrada a seguir. Suponha que uma empresa de marketing deseja segmentar os clientes em grupos e monitorar a evolução dos grupos ao longo do tempo. A fim de detectar esses grupos é possível usar um conjunto de características que representam cada cliente (como renda mensal, idade, etc.) e informação adicional fornecida pela empresa a partir do conhecimento de especialistas. Assim, um algoritmo de detecção semissupervisionada de agrupamentos é usado inicialmente. No entanto, a medida que o tempo passa, as necessidades da empresa podem mudar, implicando em um desejo diferente de visualizar a estrutura dos agrupamentos. Então, a empresa fornece algum feedback/retorno ao processo de agrupamento para indicar que um grupo de clientes mudou e os agrupamentos precisam refletir a mudança. Então, as restrições, derivadas a partir da informação adicional, impõem uma nova estrutura dos agrupamentos dividindo o grupo anterior em dois novos grupos. Note que mais clientes ainda podem aparecer, mas as características não mudam, ou seja, a representação dos clientes não muda ao longo do tempo, somente a informação adicional.

O cenário descrito previamente é uma instância do problema investigado nesta dissertação. Na Figura 1 é ilustrada a evolução da estrutura dos agrupamentos e das restrições ao longo do tempo. Na Figura 1a é retratada uma árvore de grupos, descrevendo os diferentes níveis possíveis da organização dos dados. A organização desejada é destacada pela região pontilhada. Assim, pode-se observar que há duas organizações dos dados possíveis. No lado esquerdo da Figura 1b são descritos os seguintes conjuntos ao longo do tempo: (I) as transições (alterações) de agrupamentos detectadas, (II) a partição obtida por um



(a) (b)

Figura 1 – Um exemplo ilustrativo. (a) Árvores de grupos mostrando a partição desejada destaca pela região pontilhada em dois diferentes instantes de tempo. (b) A evolução da estrutura dos agrupamentos quando novas instâncias de dados chegam e quando as restrições mudam ao longo do tempo. Os números representam os rótulos como informação de semissupervisão e os círculos sem preenchimento representam as instâncias de dados sem semissupervisão.

algoritmo de detecção de agrupamentos semissupervisionado, e (III) a estrutura dos agrupamentos desejada pelo usuário em cada instante de tempo. Analisando o resultado do agrupamento entre os instantes t_0 e t_3 é possível notar que o critério usado pelo usuário para particionar os dados mudou de acordo com o nível de hierarquia que ele gostaria de ver (Figura 1a). Os feedbacks definem restrições que guiam o processo de detecção de agrupamentos (ver os números como rótulos na Figura 1b).

Considerando que a informação de semissupervisão fornecida pelo usuário pode mudar ao longo do tempo, as seguintes transições de agrupamentos podem ocorrer: nascimento, divisão ou união. O trabalho descrito nesta dissertação está focado em divisões e uniões e assume que a evolução da semissupervisão pode causar impacto nas modificações da estrutura dos agrupamentos ao longo do tempo, implicando em novas divisões e/ou uniões. Então, o principal objetivo do trabalho descrito nesta dissertação é prover um framework que seja capaz de usar e manter a informação de semissupervisão apropriadamente para prover um processo online de detecção de agrupamentos eficiente e eficaz.

Nas seções seguintes são apresentados os objetivos, a hipótese que norteia o trabalho

descrito nesta dissertação e, finalmente, são listadas as principais contribuições.

1.2 Objetivos e Desafios da Pesquisa

O **objetivo geral** do trabalho realizado e descrito nesta dissertação consistiu em: estratégias eficientes para analisar, rastrear e otimizar o conjunto de informação de semissupervisão que guia os algoritmos de detecção e evolução de agrupamentos.

Os **objetivos específicos** são:

- ❑ Desenvolver um método genérico capaz de realizar a detecção semissupervisionada de agrupamentos no cenário online e que receba a informação de semissupervisão especificada em feedbacks.
- ❑ Desenvolver métodos capazes de extrair semissupervisão especificada em rótulos a partir de feedbacks simples do usuário (*positivo* e *deslocamento*) e deduzi-los do nível das instâncias dos dados para o nível do resumo dos dados.
- ❑ Utilizar a detecção de transições para guiar o processo de gerenciamento da informação de semissupervisão ao longo do tempo.

A fim de tratar cada objetivo, particionou-se a investigação em sete perguntas de pesquisa como segue:

***Q1*:** Quanto a semissupervisão auxilia na eficácia da detecção de agrupamentos quando ocorrem transições (alterações) nos agrupamentos?

***Q2*:** Quanto a variação do tamanho da janela deslizante de semissupervisão afeta a eficácia da detecção semissupervisionada dos agrupamentos?

***Q3*:** Qual é o impacto do uso do detector de transições na eficácia do framework CABESS?

***Q4*:** Quão eficaz é o framework CABESS comparado com as abordagens existentes de detecção de agrupamentos?

***Q5*:** Há grandes diferenças na eficácia da detecção de agrupamentos quando usa-se detecção semissupervisionada de agrupamentos baseada em diferentes tipos de semissupervisão?

***Q6*:** Há grandes diferenças na eficácia da detecção de agrupamentos quando ocorrem diferentes tipos de transições nos agrupamentos?

***Q7*:** Quão eficiente é o framework CABESS comparado com as abordagens existentes de detecção de agrupamentos?

1.3 Hipótese

A hipótese que conduziu o desenvolvimento do trabalho descrito nesta dissertação consiste em: “O rastreamento e monitoramento dos agrupamentos auxilia no gerenciamento e evolução da informação de semissupervisão no contexto de detecção online de agrupamentos semissupervisionado, proporcionando uma detecção mais eficaz".

1.4 Contribuições

As principais contribuições do trabalho descrito nesta dissertação são:

- A definição de um método genérico que permite fazer a detecção semissupervisionada de agrupamentos no contexto online. Este método genérico é referido aqui como o framework CABESS. O método recebe e lida com semissupervisão especificada em feedbacks e permite o uso de diferentes tipos de algoritmos de detecção de agrupamentos semissupervisionados internamente.

- A criação de um método que faz extração da informação de semissupervisão a patir de feedbacks e a especifica na forma de rótulos.

- O uso de uma estratégia que utiliza as técnicas de detecção de transição de agrupamento para fazer manutenção da semissupervisão ao longo do tempo. As transições detectadas são tratadas como gatilhos para a exclusão ou manutenção de um subconjunto da informação de semissupervisão, permitindo uma melhor gestão da semissupervisão ao longo tempo.

1.5 Organização da Dissertação

Esta dissertação está organizada em seis capítulos, a saber:

- **Capítulo 2.** Apresenta os conceitos fundamentais em detecção de agrupamentos não-supervisionados e semissupervisionados, necessários para uma compreensão do trabalho realizado.

- **Capítulo 3.** Traz e compara os trabalhos relacionados com o trabalho descrito nesta dissertação. Apresenta detalhes referentes a detecção semissupervisionada de agrupamentos nos cenários *online* e *data streams* e também explana sobre rastreamento e detecção de transições em agrupamentos.

- **Capítulo 4.** Descreve em detalhes a nova abordagem genérica de detecção online e semissupervisionada de detecção de agrupamentos.

- **Capítulo 5.** Traz os detalhes referentes ao projeto e execução dos experimentos realizados. Apresenta os resultados experimentais e as análises para cada conjunto de experimentos.
- **Capítulo 6.** Conclui esta dissertação e propõe possíveis trabalhos futuros. Além disso, apresenta a contribuição em termos de publicação bibliográfica.

Capítulo 2

Fundamentação Teórica

Neste capítulo são apresentados os fundamentos teóricos necessários para o entendimento da pesquisa descrita nesta dissertação. A organização deste capítulo é apresentada a seguir. Primeiramente, na Seção 2.1 são apresentados os conceitos básicos sobre detecção de agrupamentos considerando o cenário *batch*. Em seguida, na Seção 2.2, são abordadas as técnicas de semissupervisão, listando os principais tipos de informação de semissupervisão. Na Seção 2.3 são explanados os conceitos fundamentais dos algoritmos de detecção de agrupamentos em *data streams*. A última seção é dedicada para apresentação das considerações finais deste capítulo.

2.1 Detecção de Agrupamentos no Cenário *Batch*

Detecção de agrupamentos, ou análise de agrupamento (em inglês *clustering* ou *clustering analysis*), tem se mostrado uma tarefa importante em mineração de dados [Basu, Davidson e Wagstaff 2008]. Nesta seção é apresentada uma breve revisão sobre a detecção de agrupamentos no cenário *batch*, ou seja, quando o algoritmo possui *a priori* todos os dados na memória principal. Primeiramente, são definidos, a seguir, o conjunto de dados e o problema de detecção de agrupamentos no cenário *batch*.

Definição 2.1.1 (Conjunto de dados)**.** Um conjunto de dados $\mathcal{D}$ com n instâncias[1], onde cada instância é descrita por um vetor de características formado por d coordenadas (atributos), pode ser expresso como uma sequência com n instâncias como segue. Denota-se por x_i a i-ésima instância pertencente a $\mathcal{D}$. O valor para o j-ésimo atributo da instância x_i é denotado por x_i^j.

$$\mathcal{D} = \{x_1, x_2, ..., x_n\}$$

Definição 2.1.2 (Detecção de agrupamentos no cenário *batch*)**.** *Dado um conjunto de dados $\mathcal{D}$, o objetivo de um algoritmo de detecção de agrupamentos é dispor as instâncias*

[1] Também mencionado ao longo do texto como objetos ou exemplos.

em um conjunto de grupos (clusters), tal que os grupos satisfaçam alguma relação de similaridade. A ideia é maximizar a similaridade entre as instâncias que estão em um mesmo grupo e minimizar a similaridade entre instâncias de grupos distintos [Tan Pang-Ning; Steinbach e Kumar 2006].

A análise de agrupamentos é uma das principais tarefas em mineração de dados e pode ser organizada por seu propósito: (a) agrupamento para compreensão e (b) agrupamento por utilidade [Tan Pang-Ning; Steinbach e Kumar 2006]. Em agrupamento para compreensão o objetivo é identificar os grupos que são potenciais classes. Em outras palavras, automatizar a descoberta de classes. Essa abordagem é utilizada, por exemplo, em aplicações nas áreas da biologia, recuperação de informação, clima, psicologia, medicina e negócios. Já a ideia no agrupamento por utilidade é fornecer uma abstração das instâncias de um grupo, isto é, caracterizar cada grupo em termos de uma instância que seja representativa das outras instâncias do mesmo grupo. Técnicas e tarefas que objetivam resumir ou compactar os dados e encontrar eficientemente os vizinhos mais próximos são exemplos que abordam a detecção de agrupamentos como agrupamento por utilidade [Tan Pang-Ning; Steinbach e Kumar 2006].

Os algoritmos de detecção de agrupamentos podem, ainda, ser classificados conforme a técnica adotada nas seguintes categorias: hierárquicos (CURE [Guha, Rastogi e Shim 1998]), baseados em particionamento (KMEANS [MacQueen et al. 1967]), baseados em densidade (DBSCAN [Ester et al. 1996] e DENCLUE [Hinneburg e Keim 1998]), baseados em redes neurais (a rede de Kohonen [Kohonen 1982]) e baseados em *grid* (CLIQUE [Agrawal et al. 1998]). Outros autores dividem os algoritmos de mineração de dados, incluindo as técnicas de agrupamento, conforme o paradigma ao qual as abordagens utilizam, a saber: (a) combinatório (KMEANS, DBSCAN), (b) probabilístico (EM [Dempster, Laird e Rubin 1977]), (c) algébrico (*Spectral Clustering*) e (d) baseado em grafos (HSC [Hartuv e Shamir 2000] e CLICK [Sharan e Shamir 2000]) [Zaki e Jr 2014].

Há, ainda, algoritmos que combinam mais de uma técnica de agrupamento. Esses algoritmos são conhecidos como *ensembles* e objetivam superar as limitações das técnicas tradicionais. A ideia é inspirada nas técnicas que também usam múltiplos classificadores em uma tarefa de predição. Os algoritmos *ensembles* são divididos em: *ensembles* de agrupamentos, agrupamento multiobjetivo e *ensemble* multiobjetivo [Faceli et al. 2011].

Algoritmos de detecção de agrupamentos baseados em particionamento e densidade são adotados no trabalho descrito nesta dissertação. Portanto, a seguir são apresentadas as revisões sobre as técnicas de detecção de agrupamentos baseadas em particionamento e as técnicas baseadas em densidade no cenário *batch*.

2.1.1 Detecção de Agrupamentos por Particionamento

Diversos algoritmos baseados em particionamento têm sido desenvolvidos ao longo dos últimos anos, sendo o K-MEANS um dos algoritmos mais conhecidos [Jain 2010]. A ideia principal por trás desses algoritmos consiste em encontrar a melhor combinação que divide o conjunto de dados em um número específico de grupos (partição) [Barioni et al. 2014].

Definição 2.1.3 (Detecção de agrupamentos baseado em particionamento no cenário *batch*)**.** Dado um conjunto de dados com n instâncias $\mathcal{D} = \{x_i\}_{i=1}^{n}$, e dado um número desejado de grupos k, o objetivo de um algoritmo de detecção de agrupamentos por particionamento é dividir o conjunto de dados em k grupos, definindo assim uma partição $\Pi = \{\Pi_1, \Pi_2, ..., \Pi_k\}$ [Zaki e Jr 2014].

Note que, a quantidade de soluções possíveis (combinações) para n instâncias em k grupos distintos é $O\left(\frac{k^n}{k!}\right)$. Claramente, uma solução por força bruta não é viável [Zaki e Jr 2014]. Geralmente, as técnicas particionais refinam a busca pelo melhor particionamento ajustando as instâncias que representam os grupos a cada iteração e, então, melhoram a qualidade do particionamento iterativamente, até atingir um critério de convergência. No entanto, essas técnicas exigem que o usuário informe a quantidade k de grupos no início da execução do algoritmo. Consequentemente, o uso desses algoritmos pode ser inviabilizado quando não se sabe *a priori* a quantidade k de grupos que há no conjunto de dados. Então, visando minimizar esse problema, o algoritmo X-MEANS foi desenvolvido de forma a determinar o melhor k dentro de um intervalo de valores fornecidos pelo usuário, relaxando a exigência imposta pelo K-MEANS. O X-MEANS faz a busca do valor k por meio da divisão de cada agrupamento e usa a pontuação BIC (*Bayesian Information Criterion*) associada para decidir se deve manter a divisão ou não [Pelleg et al. 2000].

Medidas de similaridade. Como a ideia base para muitos algoritmos de detecção de agrupamentos faz uso da similaridade, ou dissimilaridade, entre as instâncias do conjunto de dados, é fundamental definir e entender o comportamento dessa classe de funções. Esse tipo de função tem por objetivo quantificar o quão parecidos, ou próximos, são duas instâncias. Em geral, os algoritmos que usam a noção de similaridade constroem uma matriz de similaridade $n \times n$, tal que na posição ij da matriz é armazenada a distância entre a i-ésima instância em relação a j-ésima instância.

Seja x_i, x_j e x_k pertencentes ao conjunto de dados $\mathcal{D}$, uma função $\delta : \mathcal{D} \times \mathcal{D} \rightarrow \mathbb{R}_+$ é dita função de distância se satisfaz as seguintes condições [Faceli et al. 2011]:

- *Simetria*: $\delta(x_i, x_j) = \delta(x_j, x_i)$
- *Não-negatividade*: $0 \leq \delta(x_i, x_j) < \infty$
- *Desigualdade triangular*: $\delta(x_i, x_j) \leq \delta(x_i, x_k) + \delta(x_k, x_j)$

As medidas baseadas na métrica de Minkowski (Equação 1) são as mais utilizadas para atributos com valores contínuos ou racionais. As distâncias Euclidiana, Manhattan e *Supremum*, são exemplos de medidas baseadas na métrica de Minkowski [Faceli et al. 2011].

$$\delta(x_i, x_j) = \sqrt[p]{\sum_{l=1}^{d} |x_i^l - x_j^l|^p} \tag{1}$$

2.1.2 Detecção de Agrupamentos por Densidade

Neste tipo de técnica, o processo de detecção de agrupamentos é guiado pela densidade dos dados. Assim, uma região densa de instâncias de dados, separada de outras regiões de alta densidade por regiões de baixa densidade, pode ser considerada um agrupamento. O algoritmo DBSCAN é um conhecido algoritmo de detecção de agrupamentos baseado em densidade. Ele é capaz de produzir uma partição dos dados determinando automaticamente o número k de agrupamentos. No entanto, é necessário informar dois parâmetros: a distância mínima, ϵ, necessária para indicar se duas instâncias são acessíveis diretamente entre si e o número mínimo $minPoints$, de instâncias acessíveis, necessárias para caracterizar uma instância núcleo (*core point*). De maneira geral, o processo de detecção do algoritmo DBSCAN ocorre como descrito a seguir: (1) varre-se o conjunto de dados identificando as instâncias núcleo e, em seguida, (2) computa-se as componentes conexas dessas instâncias; (3) o algoritmo atribui as instâncias que não foram marcadas como instâncias núcleo ao agrupamento mais próximo obedecendo ao parâmetro ϵ; (4) finalmente, as demais instâncias são marcadas como ruído.

2.1.3 Avaliação da Detecção de Agrupamentos no Cenário *Batch*

Análise de agrupamentos é uma tarefa de aprendizado não-supervisionado, pois não se sabe *a priori* a qual grupo cada instância pertence. Devido a essa característica, os critérios de avaliação buscam mensurar, de maneira qualitativa, o valor das estruturas encontradas pelos algoritmos de detecção de agrupamentos. Então, pode-se dividir as métricas para avaliação dos agrupamentos em: critérios internos, critérios externos e critérios relativos.

A) Critérios internos. Esse tipo de critério avalia a qualidade do agrupamento com base no próprio conjunto de dados. Por exemplo, um índice que adota esse critério pode medir a qualidade de uma partição produzida por um algoritmo com base na matriz de similaridade que é uma informação inerente (interna) ao conjunto. O índice *Gap* é um exemplo que usa esse tipo de critério [Faceli et al. 2011].

B) Critério relativo. O emprego desse tipo de critério não mensura a qualidade do agrupamento com base em informações inerentes ao conjunto, mas compara duas

partições diferentes produzidas por algoritmos distintos ou partições obtidas por uma mesma técnica, porém com parâmetros diferentes. Assim, pode-se definir qual algoritmo melhor se ajusta aos dados ou a configuração mais adequada dos parâmetros. O índice Silhueta é um exemplo para esse tipo de critério [Faceli et al. 2011].

C) Critérios externos. Quando o pesquisador possui algum conhecimento *a priori* do domínio, ou do conjunto de dados, ou tiver alguma intuição sobre a estrutura do conjunto de dados, então pode-se utilizar esse conhecimento para comparar com o agrupamento obtido. Esse tipo de critério é conhecido como critério externo ou avaliação supervisionada. Os índices Pureza, *Rand*, *Rand* Corrigido, *Jaccard* e *F-measure* são exemplos dessa categoria de índices [Faceli et al. 2011].

No contexto do trabalho descrito nesta dissertação, ter-se-á informação adicional sobre os agrupamentos. Portanto, a adoção dos critérios externos é necessária e uma breve descrição e um exemplo são apresentados a seguir. Além disso, as medidas externas discutidas aqui necessitam das informações dos pares de instâncias entre os agrupamentos de uma partição. Geralmente, essas medidas são inspiradas na avaliação de técnicas de classificação. Por conseguinte, pares de instâncias da "mesma classe" devem pertencer ao mesmo agrupamento. A ideia é computar o tamanho desses conjuntos de pares para cada partição e, em seguida, comparar as duas partições (Π e Π') utilizando essas informações. A definição desses conjuntos é apresentada a seguir. Note que, esses conjuntos constituem a matriz de confusão.

$$S_{00} = \{(x_i, x_j) : \exists k(x_i, x_j \in \Pi_k) \wedge \exists l(x_i, x_j \in \Pi'_l)\} \tag{2}$$

$$S_{01} = \{(x_i, x_j) : \exists k(x_i, x_j \in \Pi_k) \wedge \exists l(x_i \in \Pi'_l) \wedge \exists m(x_j \in \Pi'_m)\} \tag{3}$$

$$S_{10} = \{(x_i, x_j) : \exists k(x_i \in \Pi_k) \wedge \exists l(x_j \in \Pi_l) \wedge \exists m(x_i, x_j \in \Pi'_m)\} \tag{4}$$

$$S_{11} = \{(x_i, x_j) : \exists k(x_i \in \Pi_k) \wedge \exists l(x_j \in \Pi_l) \wedge \exists m(x_i \in \Pi'_m) \wedge \exists n(x_j \in \Pi'_n)\} \tag{5}$$

Exemplo 2.1.4. Considere para este exemplo que o conjunto de dados $\mathcal{D}$ seja unidimensional, onde a única dimensão x tenha $dom(x) = \{\times, \circ, \diamond\}$. Deseja-se identificar os $k = 3$ grupos pertencentes a $\mathcal{D}$. Suponha que o resultado obtido (Π) é representado pelas colunas, e os grupos reais (Π^*) são representados pelas linhas, apresentados na Tabela 1. Para este cenário, a cardinalidade de cada conjunto de pares de instâncias é: $|S_{00}| = 20$, $|S_{01}| = 20$, $|S_{10}| = 24$ e $|S_{11}| = 72$.

Tabela 1 – Grupos reais (linhas) *versus* grupos detectados por um algoritmo (colunas).

	Π_1	Π_2	Π_3
$\Pi_1^* = \{\times\}$	5	1	2
$\Pi_2^* = \{\circ\}$	1	4	0
$\Pi_3^* = \{\diamond\}$	0	1	3

- Índice *Rand*. Esta medida foi proposta em [Rand 1971] e objetiva indicar a similaridade entre duas partições. O índice *Rand* foi inspirado na avaliação de algoritmos de classificação, em que sabe-se a classe correspondente a cada tupla. Na prática, não é necessário conhecer a partição ótima Π^*. Geralmente, avalia-se duas partições Π e Π', conforme a Equação 6. A computação dessa métrica gera um número real no intervalo [0,1]. Valores próximos de 1 indicam que as duas partições Π e Π' possuem agrupamentos semelhantes. Contudo, essa métrica apresenta o seguinte problema: ela não é capaz de indicar satisfatoriamente a aleatoriedade de duas partições. Para o Exemplo 2.1.4, o índice *Rand* com respeito a partição real Π^* e a partição computada por um algoritmo de detecção de agrupamentos Π é $\mathcal{R}(\Pi, \Pi^*) = \frac{20+72}{136} \approx 0.68$.

$$R(\Pi, \Pi') = \frac{|S_{00}| + |S_{11}|}{\binom{n}{2}} \tag{6}$$

- Índice *Rand* Ajustado (ARI - *Adjusted Rand Index*). Esta é a versão ajustada do índice *Rand*, proposta em [Hubert e Arabie 1985], também conhecida por índice *Rand* ajustado. A ideia é normalizar o índice *Rand* objetivando minimizar os problemas da métrica original. Sabe-se que o índice *Rand* não é capaz de expressar em um valor constante quando duas partições possuem agrupamentos construídos aleatoriamente. Hubert e Arabie, por exemplo, consideram que os agrupamentos em duas partições não precisam, necessariamente, ter o mesmo número de elementos. Em outras palavras, a normalização no índice *Rand* corrigido assume *a priori* uma distribuição hipergeométrica (hipótese nula). O índice *Rand* corrigido pode ser calculado de acordo com a Equação 7 e sua computação resulta em um valor no intervalo [-1,1]. Espera-se que o índice *Rand* corrigido entre duas partições aleatórias

seja zero (0) e entre duas partições idênticas seja um (1) [Wagner e Wagner 2007].

$$ARI(\Pi,\Pi') = \frac{\sum_{i=1}^{|\Pi|}\sum_{j=1}^{|\Pi'|}\binom{n_{ij}}{2} - t_3}{\frac{1}{2}(t_1+t_2) - t_3}, \text{ onde}$$
$$t_1 = \sum_{i=1}^{|\Pi|}\binom{n_i}{2} \qquad (7)$$
$$t_2 = \sum_{j=1}^{|\Pi'|}\binom{n_j}{2}$$
$$t_3 = \frac{2t_1t_2}{n(n-1)}$$

- Índice *F-measure*. A medida F, em inglês *F-measure*, é uma combinação das medidas: precisão, $precision(\Pi,\Pi') = \frac{S_{00}}{S_{00}+S_{10}}$, e revocação, $recall(\Pi,\Pi') = \frac{S_{00}}{S_{00}+S_{01}}$. Basicamente, este índice é a média harmônica das métricas apresentadas e é calculado conforme a Equação 8 [Tan Pang-Ning; Steinbach e Kumar 2006].

$$F(\Pi,\Pi') = \frac{2 \cdot precision(\Pi,\Pi') \cdot recall(\Pi,\Pi')}{precision(\Pi,\Pi') + recall(\Pi,\Pi')} \qquad (8)$$

2.2 Semissupervisão

Detecção de agrupamentos é uma tarefa inerentemente difícil em que objetiva-se obter o melhor conjunto de grupos, onde, geralmente, o número k de agrupamentos não é conhecido, e utiliza-se somente da informação intrínseca aos dados. A maioria dos problemas em agrupamento de dados são, em geral, NP (não determinístico polinomial). Os algoritmos propostos na literatura não obtém a solução ótima e são heurísticas que fornecem aproximações [Zeng et al. 2002].

O aprendizado semissupervisionado, proposto pioneiramente em [Scudder 1965], foi desenvolvido a partir da necessidade de propor e avaliar o desempenho das técnicas que utilizam conhecimento prévio, ou seja, quando têm-se alguma informação adicional ou conhecimento sobre o domínio. No contexto de agrupamento de dados, em geral, essa informação adicional é conhecida como restrições e é um conjunto pequeno usado para auxiliar o algoritmo no processo de detecção dos agrupamentos [Chapelle et al. 2006]. Espera-se um desempenho superior dos algoritmos quando faz-se uso da informação de semissupervisão em relação as técnicas não supervisionadas. Portanto, algumas questões sobre as restrições devem ser respondidas, a saber: (1) como especificar as restrições, (2) como obter essa informação na prática [Jain 2010] e (3) como avaliar a qualidade desse conjunto de restrições [Davidson, Wagstaff e Basu 2006].

Nesta seção são apresentados: (a) os tipos de informação de semissupervisão, ou seja, como as restrições são obtidas e especificadas nos trabalhos já publicados (respondendo as

questões 1 e 2), e (b) as métricas disponíveis na literatura que visam mensurar a qualidade das restrições (respondendo a questão 3).

2.2.1 Tipos de Restrições

Problemas distintos exigem diferentes formas de especificação de restrições, afinal, conforme o domínio, determinadas representações são mais adequadas. Portanto, diferentes maneiras de especificar restrições foram descritas na literatura. A seguir são listados as principais formas apresentadas na literatura para representar esse conhecimento.

Rótulos como Restrições. Este é o tipo de especificação de semissupervisão que traz informação adicional relacionada a cada instância do conjunto de semissupervisão individualmente, ou seja, é um tipo de informação adicional pontual. Geralmente, a classe da instância é usada como rótulo objetivando associar a classe a um agrupamento. Exemplos de algoritmos que usam restrições especificadas em rótulos são o SSDBscan [Lelis e Sander 2009] e o SSE-Stream [Treechalong, Rakthanmanon e Waiyamai 2015].

Restrições de Nível Instância. Este tipo de restrição pode estabelecer dois tipos de relação entre duas instâncias quaisquer do conjunto de dados, como definido a seguir.

Definição 2.2.1 (Pares de restrições de nível instância)**.** Dado um conjunto de dados $\mathcal{D} = \{x_1, ..., x_n\}$ e seja x_i e x_j duas instâncias em $\mathcal{D}$.

- Uma restrição ***must-link*** (ML) entre x_i e x_j é uma relação de equivalência $c_{\in}(x_i, x_j)$ que especifica que x_i e x_j devem estar no mesmo agrupamento. Logo, uma restrição *must-link* deve satisfazer as seguintes propriedades: (i) simetria $c_{\in}(x_i, x_i), \forall x_i \in \mathcal{D}$, (ii) reflexividade $c_{\in}(x_i, x_j) = c_{\in}(x_j, x_i), \forall x_i, x_j \in \mathcal{D}$ e (iii) transitividade $c_{\in}(x_i, x_j) \wedge c_{\in}(x_j, x_m) \Rightarrow c_{\in}(x_i, x_m), \forall x_i, x_j, x_m \in \mathcal{D}$.

- Uma restrição ***cannot-link*** (CL) entre x_i e x_j é uma relação binária $c_{\notin}(x_i, x_j)$ que especifica que x_i e x_j não podem estar no mesmo agrupamento. Portanto, uma restrição *cannot-link* deve satisfazer a propriedade reflexiva: $c_{\notin}(x_i, x_j) = c_{\notin}(x_j, x_i)$, $\forall x_i, x_j \in \mathcal{D}$. Uma restrição *cannot-link* pode ser originada da decorrência de várias restrições *cannot-link* [Davidson e Basu 2007].

Denota-se o conjunto de todos os pares de restrições *must-link* por $C_{\in}$ e o conjunto de todos os pares de restrições *cannot-link* por $C_{\notin}$. Na Figura 2 é apresentado um exemplo de conjunto de dados em que um subconjunto de instâncias constituem o conjunto de pares de restrições.

Essa especificação é uma das mais usadas para representar restrições [Aggarwal e Reddy 2013]. O primeiro algoritmo de detecção de agrupamentos a adotar esse tipo de representação foi o COP-Kmeans proposto em [Wagstaff et al. 2001]. Nesse trabalho, o algoritmo K-means é estendido para incorporar e detectar agrupamentos que respeitem o

conjunto de pares de restrições *must-link* e *cannot-link* informados. Não obstante, outros algoritmos que lidam com restrições desse tipo também foram propostos na literatura posteriormente: [Jiang et al. 2013], [Lai et al. 2014].

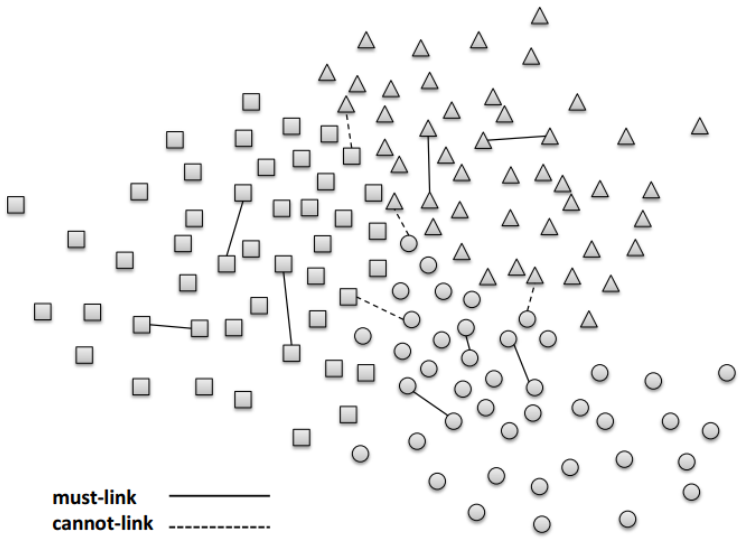


Figura 2 – Exemplo de pares de restrições *must-link* e *cannot-link* de nível instância (Obtido de [Silva 2015]).

Restrições de Nível Atributo. Restrições de nível atributo estabelecem uma relação entre os atributos do conjunto de dados. Em geral, dado um esquema relacional $R(A_1, ..., A_{n'})$, dois atributos A_i e A_j podem constituir uma nova forma de restrição *must-link* ou *cannot-link* de duas maneiras: juntamente com pares de restrições de nível instância [Pensa et al. 2010] ou somente baseando-se nos valores dos atributos ($\text{dom}(A_i)$ e $\text{dom}(A_j)$) [Schmidt, Brändle e Kramer 2011]. Esse tipo de representação é de grande valia, pois permite expressar que dois atributos: (a) compartilham características semelhantes, e então pode-se agrupar o conjunto de instâncias associado, ou (b) possuem características distintas, e então força-se a separação do conjunto de instâncias associado.

Restrições de Nível Grupo. Pares de restrições de nível instâncias são independentes e, então, exigem que o usuário especialista possa visualizar todo o espaço com os exemplos. No entanto, isto só é factível em um baixo número de dimensões (2 ou 3 dimensões), inviabilizando, portanto, os conjuntos que na prática possuem uma dimensionalidade maior. Considerando este cenário, em [Dubey, Bhattacharya e Godbole 2010] é apresentado uma abordagem interativa que possibilita ao usuário fornecer *feedback* durante o processo de atribuição das instâncias aos grupos. O *feedback* informado pelo usuário é especificado em restrições de nível grupo. De fato, o especialista pode mover uma instância de um dos grupos correntes para outro. Além disso, Dubey, Bhattacharya e Godbole 2010 permitem que o especialista altere o vetor de características dos representantes dos grupos detectados.

Restrições por *Ranking*. Autores como Hartigan 1975 afirmam que "*diferentes agrupamentos são corretos para diferentes propósitos, assim, não podemos dizer que um agrupamento é melhor*". Portanto, mais de uma possibilidade de particionamento pode ser satisfatório em um conjunto de dados. Não há algoritmo de detecção de agrupamentos capaz de descobrir toda a diversidade de estruturas que podem existir nos dados [Faceli et al. 2011]. Associado a esse problema, um usuário pode desejar informar uma ordem que expresse suas preferências no que se refere a quais e em que ordem as restrições devem ser aplicadas . Considerando esse cenário vertiginoso de possibilidades, alguns trabalhos especificam as restrições em *ranking* [Ben Ahmed, Nabli e Gargouri 2013].

Um *ranking* de restrições é uma relação entre um conjunto de pares de restrições de modo que, para quaisquer par de restrições ordenado, a primeira restrição é *'mais importante do que'* a segunda restrição. Em outras palavras, o ranqueamento de um conjunto de restrições estabelece uma ordem de prioridade entre as restrições. Formalmente, restrições por *ranking* estabelecem uma relação de ordem entre pares de restrições, conforme a Definição 2.2.2.

Definição 2.2.2 (*Ranking* de restrições)**.** Dado um conjunto de instâncias $\mathcal{D}$ e um conjunto de restrições *must-link* e *cannot-link* $C = C_{\in} \cup C_{\notin} = \{c_1, ..., c_p\}$, um *ranking* de restrições é uma relação de ordem parcial estrita $\mathcal{R} \subseteq C \times C$ que satisfaz as propriedades a seguir. Tipicamente, uma ordem parcial estrita é representada pelo símbolo $>$. A notação $c_i > c_j$ indica o fato que c_i é mais importante que c_j [Ben Ahmed, Nabli e Gargouri 2013].

- Irreflexividade: $\forall c_i \in C : c_i \not> c_i$
- Assimetria: $\forall c_i, c_j \in C : c_i > c_j \Rightarrow c_j \not> c_i$
- Transitividade: $\forall c_i, c_j, c_k \in C : c_i > c_j \wedge c_j > c_k \Rightarrow c_i > c_k$

Restrições Relativas. Esse tipo de restrição define uma relação de similaridade entre três ou mais instâncias de tal modo a estabelecer uma prioridade entre elas. O objetivo é, dado as instâncias x_i, x_j, x_k, definir qual o par mais próximo, ou mais relevante, entre as três instâncias. Em outras palavras, a ideia é representar o conhecimento na seguinte forma "x_i *está mais próximo de* x_j *do que de* x_k". Este exemplo é denominado restrição tripla e pode ser representado por $\tau_3 = (x_i, x_j, x_k)$. Não obstante, Liu, Zhang e Wang 2011 apresenta uma forma mais compacta de representar restrições triplas. Uma restrição tripla pode ser descrita na forma $x_i, x_j|x_k$ e determina que $\delta(x_i, x_j)$ é o par com menor distância em relação a qualquer outro par, ou seja, $\delta(x_i, x_j) < \delta(x_i, x_k) e \delta(x_i, x_j) < \delta(x_j, x_k)$, dado uma função de distância $\delta(\cdot, \cdot)$ por exemplo.

Restrições relativas tem a capacidade de expressar mais informação se comparado a pares de restrições de nível instância. Além disso, dado um número n de instâncias, a quantidade de restrições relativas possíveis de serem geradas é maior que a quantidade possível de combinações de pares de restrições.

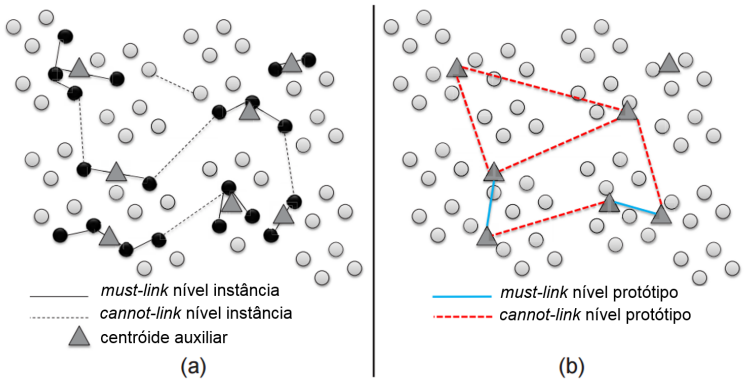


Figura 3 – Etapas do processo de definição dos pares de restrição de nível protótipo: (a) definição dos centróides (representantes) auxiliares sobre um conjunto de dados e restrições *must-link* e *cannot-link* de nível instância (b) definição dos pares de restrição de nível protótipo (Adaptado de [Silva et al. 2015]).

Restrições de Nível Protótipo. Esta especificação é proposta em [Silva et al. 2015] onde é apresentado o algoritmo MRS-KMEANS (*Multi-Representative Semi-supervised Kmeans*). O MRS-KMEANS usa os pares de restrições em nível instância para definir múltiplos representantes auxiliares para cada centróide do K-MEANS, conforme ilustrado na Figura 3. Os representantes auxiliares, por exemplo, os centróides auxiliares na Figura 3a, são usados para definirem os pares de restrições de nível protótipo que são os responsáveis por guiar a atribuição das instâncias aos grupos, juntamente com o apoio dos representantes auxiliares e do representante principal. O uso de restrições de nível protótipo permitiu ao MRS-KMEANS ser capaz de detectar agrupamentos de formatos diversos. Portanto, este tipo de restrição tornou possível uma aptidão que, em geral, é restrita a outros tipos de algoritmos, por exemplo, os algoritmos baseados em densidade.

2.2.2 Métricas de Qualidade de Restrições

A incorporação de semissupervisão objetiva melhorar a qualidade do resultado obtido em comparação com técnicas nativamente não-supervisionadas, isto é, no caso da detecção de agrupamentos, aperfeiçoar os agrupamentos descobertos. A ideia é que a medida que se aumenta o conjunto de restrições, obtém-se um resultado superior. Por outro lado, perde-se em complexidade computacional, pois o algoritmo precisará satisfazer as restrições informadas. No entanto, a falta de cuidado na amostragem da informação de semissupervisão pode não auxiliar de fato o algoritmo adotado e, ainda, degradar o resultado [Wagstaff, Basu e Davidson 2006]. De fato, trabalhos como [Wagstaff, Basu e Davidson 2006] e [Davidson, Wagstaff e Basu 2006] avaliaram o desempenho de um mesmo algoritmo de detecção de agrupamentos semissupervisionado sobre diferentes amostras de um conjunto de restrições. Esses trabalhos mostraram que um subconjunto de restrições de boa qualidade, de acordo com as métricas de avaliação, propicia resultados melhores

se comparado com o conjunto de restrições inteiro para um mesmo algoritmo.

Neste contexto, métricas para avaliar a qualidade do conjunto de restrições a ser usado podem ser usadas antes da execução do algoritmo de detecção de agrupamentos (ver Figura 4). A seguir são apresentadas as principais medidas que avaliam a qualidade da informação de semissupervisão, a saber: informatividade e coerência.

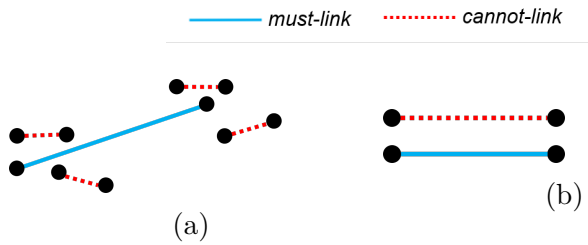


Figura 4 – Exemplos do impacto da configuração espacial dos pares de restrições ML e CL nas métricas de qualidade de semissupervisão: (a) um conjunto de restrições com alta informatividade para o algoritmo CKM [Wagstaff et al. 2001] e (b) um conjunto de restrições com alta incoerência (Adaptado de [Davidson, Wagstaff e Basu 2006]).

Informatividade. Refere-se a proporção de informação em um conjunto de restrições que um algoritmo não pode determinar por si próprio. Essa métrica depende de um algoritmo para ser calculada [Wagstaff, Basu e Davidson 2006].

Coerência. Esta métrica quantifica a proporção de concordância em um conjunto de restrições C utilizando uma função de distância δ. É importante destacar que, essa medida não depende de qualquer algoritmo [Wagstaff, Basu e Davidson 2006] e também não requer conhecimento da partição ótima Π^* [Davidson, Wagstaff e Basu 2006].

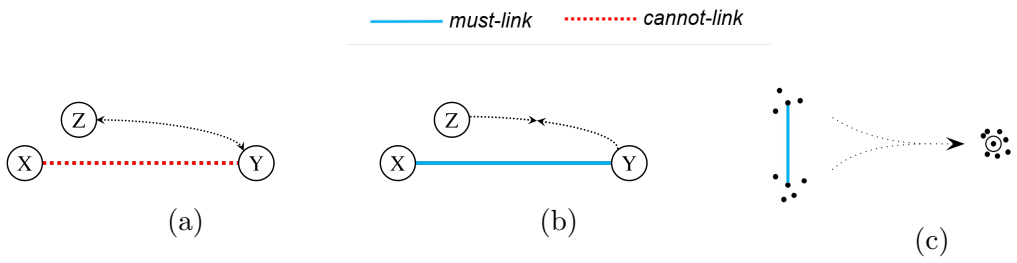


Figura 5 – Instâncias próximas a pares de restrições são afetadas: (a) uma restrição *cannot-link* entre X e Y atrai Z e Y, (b) uma restrição *must-link* entre X e Y afasta Z e Y, (c) agrupamentos que estão distantes no espaço de características (esquerda) podem ser unidos no espaço de similaridade (direita) com a propagação da restrição *must-link* (Adaptado de [Klein, Kamvar e Manning 2002]).

As restrições *must-link* e *cannot-link* impõem uma força de atração, ou repulsão, respectivamente em um espaço de características (ver Figura 5) [Klein, Kamvar e Manning

2002] [Davidson, Wagstaff e Basu 2006]. Portanto, duas restrições não são coerentes se elas exercem forças contraditórias sobre a mesma vizinhança, por exemplo na Figura 4b [Davidson, Wagstaff e Basu 2006].

2.3 Detecção de Agrupamentos em *Data Streams*

Nos últimos anos, o avanço tecnológico e o desenvolvimento do hardware e das redes de comunicação proporcionaram uma capacidade de geração de dados vertiginosa. Explorar esse tipo de informação que é produzida e invalidada rapidamente tornou-se uma nova linha de pesquisa dentro de mineração de dados. Esse cenário, em que não se tem todos os dados *a priori* antes da execução de um algoritmo, mas os dados chegam em fluxo é conhecido como *data streams*. A seguir são apresentados formalmente a definição de um *data stream* e a extensão da definição de detecção de agrupamentos no respectivo cenário.

Definição 2.3.1 (*Data streams*)**.** Um data stream $\mathcal{S}$ é uma sequência maciça de dados $x_1, x_2, ..., x_N$, ou seja, $\mathcal{S} = \{x_i\}_{i=1}^{n}$, potencialmente sem limite ($n \to \infty$). Cada objeto de dado é descrito por um vetor de atributos d-dimensional $x_i = [x_i^j]_{j=1}^{d}$ pertencente a um espaço de atributos Ω que pode ser contínuo, categórico ou misto [Silva et al. 2013].

Definição 2.3.2 (Detecção de agrupamentos em *data streams*)**.** Dado um *data stream* $\mathcal{S}$, um algoritmo de detecção de agrupamentos sobre $\mathcal{S}$ objetiva manter uma boa consistência dos agrupamentos continuamente, a medida que analisa-se os dados, usando uma pequena quantidade de memória em um espaço limitado de tempo [Gama 2010].

As características que distinguem o cenário de *data streams* em relação ao cenário *batch* são apresentados em [Silva et al. 2013] e listadas a seguir. Essas particularidades tornam o processo de detecção de agrupamentos em *data streams*, e de modo geral o de mineração de dados, mais desafiador.

1. Os dados chegam continuamente.

2. Não há controle sobre a ordem que os dados devem ser processados.

3. O tamanho do fluxo de dados é (potencialmente) infinito.

4. Os dados são descartados após serem processados. Na prática, uma porção dos dados pode ser armazenada por um período de tempo determinado, usando uma técnica de ***esquecimento*** para descartá-los mais tarde ou uma ***janela deslizante*** que define quais os dados são recentes, ou válidos.

5. A distribuição de probabilidade dos dados pode mudar ao longo do tempo.

Diversos algoritmos foram desenvolvidos para lidar com o problema de detecção de agrupamentos em fluxo de dados. Além disso, as peculiaridades do cenário de *data streams* impõem os seguintes requisitos aos algoritmos de agrupamento: (a) devem fornecer resultados rapidamente, ou seja, detectar os agrupamentos em pouco tempo, (b) adaptar-se com rapidez às mudanças dos dados, isto é, identificar quando um novo agrupamento surge ou desaparece, por exemplo, (c) adaptar-se a taxa de chegada dos dados, (d) fornecer um modelo de representação compacto, mas que não cresça a uma taxa linear proporcionalmente ao número de dados que chegam, (e) detectar rapidamente os *outliers* e (f) lidar com diferentes tipos de dados. No entanto, nem todos os requerimentos são cumpridos na prática [Silva et al. 2013].

Em geral, alguns algoritmos de detecção de agrupamentos em *data streams* são inspirados nas ideias subjacentes às técnicas adotadas no cenário *batch*: BIRCH (hierárquico) [Zhang, Ramakrishnan e Livny 1996], SINGLE-PASS K-MEANS (particional) [Farnstrom, Lewis e Elkan 2000] , DENSTREAM (densidade) [Cao et al. 2006]. Outras técnicas como o E-STREAM [Udommanetanakit, Rakthanmanon e Waiyamai 2007] e o SED-STREAM [Waiyamai et al. 2014] são guiados pela evolução dos próprios agrupamentos, por exemplo: a fusão de dois agrupamentos em um só, ou a divisão de um agrupamento em dois.

2.3.1 O algoritmo BIRCH

O BIRCH (*Balanced Iterative Reducing and Clustering using Hierarchies*) é um algoritmo de detecção de agrupamentos proposto em [Zhang, Ramakrishnan e Livny 1996]. Esse algoritmo foi projetado para lidar com grandes volumes de dados, sobretudo quando os dados não cabe inteiramente na memória principal, necessitando, em muitos casos, de uma única análise sobre cada instância de dado do conjunto (*single-pass*). Ele é capaz de lidar com grandes conjuntos de dados, pois faz uso de vetores-CF (*Clustering Feature Vectors*). Um vetor-CF sumariza os dados de um conjunto de instâncias em um tripla $CF = (N, \overrightarrow{LS}, SS)$ onde $\overrightarrow{LS}$ é a soma linear das instâncias de dados e SS é a soma quadrática das instâncias. A dimensão de $\overrightarrow{LS}$ e SS é a mesma da quantidade de atributos que descrevem as instâncias de dados. Vetores-CF possuem importantes propriedades que são úteis para o BIRCH e também para o framework proposto nesta dissertação. Essas propriedades permitem, calcular o centróide, o raio e também atualizar a informação sumarizada usando somente estatísticas suficientes, por exemplo: adicionar uma nova instância de dado a um vetor-CF, unir dois vetores-CF, e subtrair uma instância de dado.

O algoritmo BIRCH constrói uma CF-Tree (*Clustering Feature Tree*), uma árvore balanceada em que cada nodo interno possui pelo menos uma entrada na forma $\langle CF_i, child_i \rangle$, onde $child_i$ é uma referência para o i-ésimo nodo filho, $1 < i \leq B$. O número de entradas que um nodo não folha pode conter é restringido pelo parâmetro B. Cada nodo folha é uma sequência de vetores-CF juntamente com dois ponteiros indicando o nodo folha

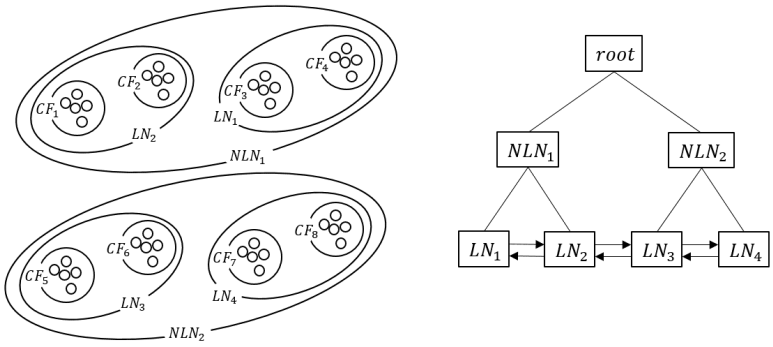


Figura 6 – Um conjunto de dados resumido pelo BIRCH (direita) e a CF-Tree construída a partir do conjunto de dados (esquerda). NLN_i são nodos não folha e LN_i são nodos folha.

anterior e o próximo novo folha $\langle prev, CF_1, ..., CF_j, next \rangle$, $1 < j \leq L$. De maneira análoga ao caso dos nodos não-folha, os nodos folha são limitados pelo parâmetro L. Cada vetor-CF de um nodo folha representa um determinado número de instâncias de dados, ou seja, o vetor-CF é um resumo desse subconjunto de instâncias. O número máximo de instâncias que um vetor-CF de um nodo folha pode representar também é limitado por um parâmetro. Ademais, as instâncias de dados representadas em um nodo folha devem conferi-lo um raio menor que um limiar T, tendo a média como centro do vetor-CF.

A construção da CF-Tree é feita por meio de uma abordagem *top-down*. O conjunto de dados é varrido uma única vez. Para cada instância de dado, o algoritmo percorre a árvore, a partir do nodo raiz, escolhendo o nodo mais próximo em cada nível. Quando atinge-se um nodo folha, um teste é realizado para verificar se a inserção da instância não viola a restrição imposta pelo parâmetro T. Se não ocorrer violação, então a instância é inserida. Caso contrário, o nodo folha precisa ser dividido se estiver cheio (parâmetro L) ou, então, basta criar um novo vetor-CF e a instância é inserida. No processo de divisão, os dois vetores-CF que estão mais distantes são utilizados como sementes (um para cada novo nodo folha) e funcionam como polos, auxiliando no processo de re-distribuição dos vetores-CF restantes. Após o processo de divisão, outro teste deve ser realizado. Esse teste avalia se o nodo pai também deve ser dividido, pois ao dividir nodos filhos, o parâmetro B pode ser violado. Caso o parâmetro seja de fato violado, o nodo pai é dividido e os vetores-CF são re-distribuídos em um processo análogo ao descrito anteriormente. O teste é realizado em cada nível da árvore podendo continuar até o nodo raiz.

O BIRCH possui ainda outras fases que podem, inclusive, reconstruir a CF-Tree ou unir nodos da árvore objetivando melhorar a utilização de espaço. No entanto, no trabalho descrito nesta dissertação, somente a fase inicial que constrói a CF-Tree é utilizada. Assim, as demais fases não são apresentadas neste texto.

2.4 Considerações Finais

Neste capítulo foram apresentados os conceitos fundamentais com relação às técnicas de detecção de agrupamento em vários cenários: *batch*, semissupervisionado e *data streams*. As diferentes maneiras de se especificar a semissupervisão também foram apresentadas.

No próximo capítulo são apresentados os trabalhos relacionados com o trabalho desenvolvido nesta dissertação. Nele são descritos as técnicas relacionadas que fazem detecção semissupervisionada de agrupamentos nos cenários de *data streams* e *online*. Além disso, o rastreamento de agrupamentos também é descrito no próximo capítulo.

Capítulo 3

Trabalhos Relacionados

Neste capítulo são apresentados os trabalhos relacionados à pesquisa desenvolvida nesta dissertação. O objetivo é posicionar esta dissertação em relação aos demais trabalhos já desenvolvidos e publicados. As estratégias empregadas no framework CABESS estão relacionadas com as pesquisas em detecção online e semissupervisionada de agrupamentos e também com as estratégias de monitoramento e evolução de agrupamentos. Assim, este capítulo está organizado como segue. Na Seção 2.3 são explanados os algoritmos de detecção de agrupamentos em *data streams* que suportam semissupervisão. O processo de detecção de agrupamentos online e a principal referência relacionada a esta dissertação são descritos em seguida, na Seção 3.3. Na Seção 3.4, uma análise comparativa, entre as técnicas da literatura e a proposta nesta dissertação, é apresentada. Finalmente, na Seção 3.5, os trabalhos referentes a rastreamento e evolução de agrupamentos são mencionados. A última seção é dedicada para apresentação das considerações finais deste capítulo.

3.1 Detecção Semissupervisionada de Agrupamentos

As técnicas semissupervisionadas de detecção de agrupamentos descritas na literatura científica usam diferentes abordagens a fim de prover o auxílio no processo de agrupamento objetivando obter grupos significativos [Barioni et al. 2014]. De maneira geral, algoritmos de detecção semissupervisionada de agrupamentos são divididos em duas categorias: (1) métodos de adaptação de similaridade que adaptam a função de similaridade empregada no processo a fim de satisfazer os rótulos ou restrições nos dados e (2) métodos que empregam rótulos ou restrições providas pelo usuário para modificar a etapa de atribuição aos agrupamentos [Basu, Davidson e Wagstaff 2008].

Há também uma variedade de maneiras de expressar e obter as restrições, como apresentado no Capítulo 2. Uma parte substancial dos trabalhos descritos na literatura científica exploram a especificação dessas restrições na forma de restrições em nível instância [Bilenko, Basu e Mooney 2004], que podem ser do tipo ML e CL. Muitos trabalhos estendem métodos clássicos de detecção de agrupamentos para torná-los capazes de lidar

com informação adicional. O MPCK-Kmeans [Bilenko, Basu e Mooney 2004], por exemplo, é uma extensão do largamente usado algoritmo K-MEANS que incorpora aprendizado de métrica e semissupervisão na forma de ML e CL. O C-DBScan [Ruiz, Spiliopoulou e Menasalvas 2007] e o SSDBSCAN [Lelis e Sander 2009] são extensões do algoritmo DBSCAN [Ester et al. 1996]. Enquanto o primeiro lida com pares de restrições, o último recebe um conjunto de instâncias rotuladas como semissupervisão. Esses métodos requerem que todo o conjunto de dados e toda a informação de semissupervisão estejam disponíveis no início do processo de detecção de agrupamentos.

De maneira geral, o algoritmo SSDBSCAN faz uso das árvores de extensão mínima construídas a partir das instâncias de dados do subconjunto de instâncias rotuladas de forma a eliminar a necessidade de informar os parâmetros ϵ e *minPoints*, que são exigidos pelo DBSCAN. As árvores construídas na primeira etapa da execução do algoritmo o auxiliam no processo de atribuição das instâncias de dados que já estão rotuladas e também das que não estão rotuladas.

No entanto, esses métodos não conseguem lidar com o cenário em que eles não possuem todo o conjunto de dados ou toda a informação de semissupervisão em memória antes da execução.

3.2 Detecção de Agrupamentos Semissupervisionada em *Data Streams*

Diversos algoritmos de detecção de agrupamentos em *data streams* foram estendidos com o objetivo de permitir a semissupervisão por meio da utilização de restrições. Os algoritmos DENSTREAM, E-STREAM e SED-STREAM, citados na seção 2.3, são exemplos de técnicas que foram modificadas e deram origem aos algoritmos C-DENSTREAM [Ruiz, Menasalvas e Spiliopoulou 2009], CE-STREAM [Sirampuj, Kangkachit e Waiyamai 2013] e SSE-STREAM [Treechalong, Rakthanmanon e Waiyamai 2015], respectivamente.

O SSE-STREAM lida com restrições de instâncias rotuladas, isto é, cada instância que faz parte do conjunto de restrições é rotulada. Assim, o rótulo determina o *cluster* ao qual a instância deve pertencer. Instâncias rotuladas podem ser utilizadas imediatamente, a medida que são analisadas, pois é fácil determinar a classe dos *clusters*, utilizando uma estrutura para representar os agrupamentos [Treechalong, Rakthanmanon e Waiyamai 2015].

Alguns algoritmos usam uma estrutura denominada FCH (*Fading Cluster Structure with Histogram*) para representar os agrupamentos. Essa estrutura é a responsável por auxiliar o algoritmo a detectar quando se deve unir um ou mais agrupamentos em um grande agrupamento ou dividir um grande agrupamento em dois ou mais agrupamentos. Os algoritmos SSE-STREAM e CE-STREAM são exemplos de abordagens que fazem uso desse tipo de estrutura. Não obstante, os algoritmos mencionados nesta seção fazem uso

de uma função de decaimento que pondera a importância de instâncias e restrições a medida que o tempo passa.

Os algoritmos C-DenStream e CE-Stream exploram restrições em pares de nível instância. No entanto, o algoritmo C-DenStream, extensão do DenStream que é baseado em densidade, adota pares de restrições definidos entre *micro-clusters*, fundamentado no fato de que os elementos envolvidos em uma restrição são geralmente representativos da sua vizinhança local. O algoritmo CE-Stream lida, somente, com pares de restrições do tipo *must-link*, sendo, portanto, uma limitação do algoritmo, sobretudo considerando as propriedades e a riqueza de informação inutilizadas ao não tratar as restrições *cannot-link*.

3.3 Detecção Online de Agrupamentos Semissupervisionada

Uma característica importante dos algoritmos de detecção de agrupamentos em *data streams* é sua capacidade de avaliar uma instância de dado uma única vez. Isso significa que após ler uma instância de dado, esse tipo de algoritmo nunca mais avaliará a instância novamente, mas trabalhará somente com o resumo dela. No entanto, no cenário em que o usuário fornece a semissupervisão ao longo do tempo para guiar o processo de detecção, o algoritmo pode precisar acessar uma instância de dado mais de uma vez. Note que, inclusive, o usuário pode fornecer semissupervisão diferente para uma mesma instância ao longo do tempo. Esse cenário viola uma premissa do cenário descrito sobre *data streams*. Então, o processo de detecção de agrupamentos na qual os algoritmos podem acessar as instâncias de dados mais de uma vez é dito online. Nesta seção é explanado em detalhes a técnica apresenta em [Lai et al. 2014] que faz a detecção de agrupamentos semissupervisionado de maneira online usando feedbacks do usuário.

Exemplos de abordagens online de detecção de agrupamentos são [Castellano, Fanelli e Torsello 2013] e [Lai et al. 2014]. O primeiro é uma extensão do SSFCM para anotação automática em imagens sobre agrupamento semissupervisionado. O algoritmo foi desenvolvido assumindo que os conjuntos de instâncias pertencentes a k agrupamentos são disponibilizados ao longo do tempo e processados em lotes, ou seja, n_1 instâncias são disponibilizadas no instante t_1, n_2 são disponibilizadas em t_2 e assim por diante. O segundo é a principal referência desta dissertação, pois o framework proposto foi inspirado a partir do referido trabalho. A seção seguinte é dedicada a apresentar em detalhes a técnica base do framework CABESS.

3.3.1 O Processo de Detecção Online Proposto em [Lai et al. 2014]

Lai et al. 2014 propõe uma técnica de detecção semissupervisionada de agrupamentos

que melhora a qualidade da partição iterativamente por meio da obtenção de feedbacks do usuário usando uma interface gráfica. A interface gráfica permite que o usuário forneça os feedbacks de maneira apropriada, pois facilita a visualização da estrutura dos agrupamentos e da organização das instâncias de dados na estrutura. Os feedbacks podem ser do tipo *positivo*, *negativo* e *deslocamento*. Um feedback *positivo* significa que a instância de dado foi corretamente atribuída a um agrupamento. Um feedback *negativo* significa que a instância de dado foi incorretamente atribuída a um agrupamento. Um feedback do tipo *deslocamento* significa que a instância não deve pertencem ao agrupamento corrente e indica qual é o agrupamento apropriado.

A técnica proposta permite que o usuário forneça os feedbacks em cada iteração do processo de detecção. De maneira geral, o método proposto possui duas fases, a saber: (1) faz-se uso de uma técnica para sumarizar os dados e obter uma primeira partição e, em seguida, (2) faz-se um refinamento a partir do agrupamento inicial por meio da obtenção dos feedbacks do usuário interativamente em várias iterações. A seguir é descrito os detalhes de cada fase do processo de detecção.

1. Micro-agrupamento. Como técnica para sumarizar os dados, o BIRCH é usado na primeira etapa. Nesta etapa, todo o conjunto de dados é sumarizado em uma única passada. A ideia por trás do uso do BIRCH é fazer um micro-agrupamento dos dados, com o objetivo de reduzir a quantidade de informação a ser processada na próxima etapa.

2. Macro-agrupamento. Após a execução do BIRCH, o método trabalha somente sobre o conjunto dos vetores-CF das folhas da árvore construída pelo BIRCH, a CF-Tree. Nesta etapa, um algoritmo de detecção de agrupamentos não supervisionado é usado para fazer um macro-agrupamento sobre os vetores-CF das folhas da CF-Tree. Logo, o método obtém uma primeira configuração da partição dos dados.

3. Refinamento. O objetivo da etapa final é refinar o partição obtida na etapa anterior utilizando a semissupervisão. A Equação 9 expressa a função objetivo que é minimizada no processo interativo de refinamento da detecção. O primeiro termo expressa a distorção/dissimilaridade entre cada vetor-CF CF_i e o centro do agrupamento associado μ_{l_i} (l_i refere-se ao rótulo do agrupamento CF_i). O segundo e o terceiro termos expressam, respectivamente, o custo de penalização quando uma restrição ML e CL é violada: w e $\bar{w}$ são constantes que especificam o custo da violação das restrições ML e CL, respectivamente. $D(CF_i, CF_j)$ é a função de distância entre dois vetores-CF e N_{CF_i} é o número de instâncias representadas por CF_i. A função de distância adotada é a Euclidiana e é calculada conforme a Equação 10.

$$\begin{aligned} J_{obj} = \sum_{CF_i \in S_{CF}} D(CF_i, \mu_i) + \sum_{\substack{(CF_i, CF_j) \in C_{CF} \\ I_i \neq I_j}} w N_{CF_i} N_{CF_j} D(CF_i, CF_j) \\ + \sum_{\substack{(CF_i, CF_j) \in C_{CF} \\ I_i = I_j}} \bar{w} N_{CF_i} N_{CF_j} (D_{max} - D(CF_i, CF_j)) \end{aligned} \tag{9}$$

$$D(CF_i, CF_j) = \sum_{p=1}^{d} \left(\frac{LS_{CF_i}(p)}{N_{CF_i}} - \frac{LS_{CF_j}(p)}{N_{CF_j}} \right)^2 \tag{10}$$

No entanto, é importante destacar que antes de executar o processo de refinamento, é preciso extrair as restrições ML e CL entre os vetores-CF a partir dos feedbacks fornecidos pelo usuário. Assim, é proposto uma estratégia para deduzir restrições a partir dos feedbacks. Essa estratégia é baseada no conceito de vizinhança. Uma vizinhança é definida como um conjunto de instâncias de dados que devem estar em um mesmo agrupamento. A definição de vizinhança segue duas regras: (1) todas as instâncias de um mesmo agrupamento que receberam feedback positivo são atribuídos para a mesma vizinhança e (2) todas as instâncias de diferentes agrupamentos que receberam feedback do tipo deslocamento para um mesmo agrupamento de destino são atribuídos para a mesma vizinhança, no entanto, instâncias com destino para diferentes agrupamentos são atribuídas para as respectivas vizinhanças (se a vizinhança não existe, então é criada).

Além disso, se um vetor-CF representa instâncias que receberam feedbacks diferentes (ou receberam feedback negativo), então ele é dividido de forma a garantir a consistência da informação de semissupervisão nas instâncias representadas. Esse processo é denominado de purificação dos vetores-CF. Após a purificação, a árvore construída pelo BIRCH é atualizada. Finalmente, pode-se inferir as restrições ML e CL entre os vetores-CF (restrições em nível sumário). Várias estratégias para inferir e manter as restrições são apresentadas. Uma estratégia possível[1] gera todos os pares ML possíveis entre: (a) vetores-CF que possuem instâncias com feedbacks positivos e que pertencem a um mesmo agrupamento ou (b) vetores-CF que possuem instâncias com feedback do tipo deslocamento e que apontam para um mesmo agrupamento de destino. Para geração de restrições CL uma matriz é mantida informando se há algum feedback (negativo ou deslocamento) em cada agrupamento que indica que a instância e seu respectivo vetor-CF não devem pertencer ao agrupamento corrente. Assim, usa-se essa informação para gerar pares CL entre o vetor-CF alvo e os vetores-CF que pertencem ao agrupamento corrente.

O processo de refinamento da estrutura dos agrupamentos é feito conforme o Algoritmo 1. As estruturas usadas no processo descrito aqui são apresentadas na Figura 7. Note que, o número k de grupos permanece inalterado durante todo o processo e deve ser conhecido e informado no início da execução do método. O refinamento altera algumas instâncias dos agrupamentos a fim de ajustar a organização dos dados conforme o desejo do usuário. Nesta etapa, cada vetor-CF é reatribuído a um agrupamento que contribui para minimizar a função objetivo expressa pela Equação 11. Após a reatribuição, o centróide de cada agrupamento μ_i é recalculado. Como o método faz uso da distância euclidiana

[1] Outras estratégias eliminam feedbacks em iterações anteriores, diminuindo a quantidade possível de restrições que podem ser geradas, ou usam técnicas mais robustas para inferir os pares ML e CL objetivando lidar com o menor número de restrições deixar de perder muita informação de semi-supvervisão.

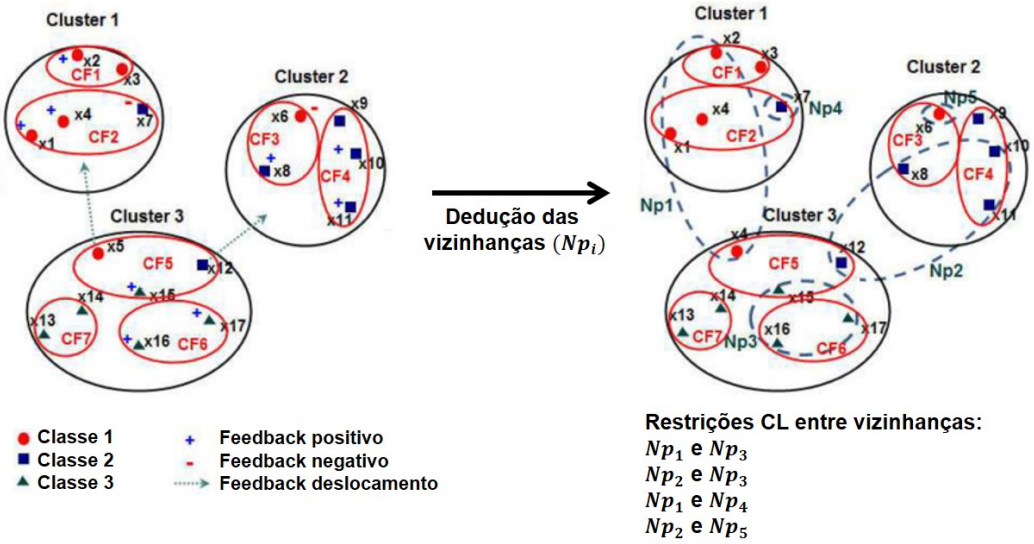


Figura 7 – Estruturas usadas para realizar o refinamento da detecção de agrupamentos usando semissupervisão. Os círculos vermelhos representam os vetores-CF. Os círculos pontilhados representam as vizinhanças deduzidas por meio dos feedbacks (Adaptado de [Lai et al. 2014]).

e cada agrupamento é representado pelo seu centróide especificado em um vetor-CF, os valores N_{μ_h}, $\overrightarrow{LS}_{\mu_h}$ e $\overrightarrow{SS}_{\mu_h}$ são calculados conforme a Equação 12.

$$\begin{aligned} J_{obj}(CF_i, \mu_h) = D(CF_i, \mu_i) + & \sum_{\substack{(CF_i, CF_j) \in C_{CF} \\ I_i \neq I_j}} w N_{CF_i} N_{CF_j} D(CF_i, CF_j) \\ + & \sum_{\substack{(CF_i, CF_j) \in C_{CF} \\ I_i = I_j}} \bar{w} N_{CF_i} N_{CF_j} (D_{max} - D(CF_i, CF_j)) \end{aligned} \tag{11}$$

$$\begin{aligned} N_{\mu_h} &= \sum_{l_i = h} N_{CF_i} \\ \overrightarrow{LS}_{\mu_h} &= \sum_{l_i = h} \overrightarrow{LS}_{CF_i} \\ \overrightarrow{SS}_{\mu_h} &= \sum_{l_i = h} \overrightarrow{SS}_{CF_i} \end{aligned} \tag{12}$$

Após a execução da etapa de refinamento, novos feedbacks podem ser fornecidos pelo usuário ao sistema por meio da interface gráfica. E o processo de refinamento continua até que o usuário esteja satisfeito com a partição retornada pelo método. Além disso, dependendo da estratégia utilizada para extrair restrições ML e CL a partir dos feedbacks, os feedbacks desta iteração e/ou das anteriores podem ser descartados. Não obstante, a abordagem assume que o feedback do usuário é sempre coerente ao longo das iterações executadas pelo algoritmo [Lai et al. 2014].

Algoritmo 1: Método de agrupamento semissupervisionado proposto em [Lai et al. 2014].

Entrada: $\{CF_i\}_{i=1}^{m}, \{\mu_h\}_{h=1}^{k}$: o conjunto de vetores-CF das folhas que foram agrupados em k grupos e seus respectivos centróides
Saída: Π: Uma partição com k grupos disjuntos.

```
1 início
2     enquanto usuário não estiver satisfeito com a organização dos agrupamentos faça
3         Receber os feedbacks do usuário e deduzir as restrições ML e CL;
4         t ← 0;
5         enquanto não convergir faça
6             Re-atribuir os rótulos dos vetores-CF para minimizar a função objetivo J_obj;
7             Calcular e atualizar o centróide dos agrupamentos;
8             t ← t + 1;
9 fim
```

3.4 Análise Comparativa

Na Tabela 2 são apresentados as principais características dos trabalhos descritos aqui. Note que os trabalhos que investigam semissupervisão não abordam o rastreamento de agrupamentos e as técnicas que avaliam a evolução e rastreamento dos agrupamentos não permitem o uso de restrições.

Uma lacuna pode ser observada na interseção entre as pesquisas sobre evolução e rastreamento de agrupamentos e análise de agrupamentos semissupervisionado. Além disso, as técnicas anteriores não levam em consideração a natureza dinâmica das restrições que podem mudar ao longo do tempo, principal aspecto investigado nesta dissertação.

Tabela 2 – Algoritmos de Detecção Semissupervisionada de Agrupamentos.

Algoritmo	**Abordagem**	**Restrições**	**Incoerência**	**Cenário**	**Referência**
MPCK-Means	Particional	ML & CL	Não	Batch	[Bilenko, Basu e Mooney 2004]
C-DBScan	Densidade	ML & CL	Não	Batch	[Ruiz, Spiliopoulou e Menasalvas 2007]
SSDBScan	Densidade	Rótulo	Não	Batch	[Lelis e Sander 2009]
C-DenStream	Densidade	Micro	Não	Stream	[Ruiz, Menasalvas e Spiliopoulou 2009]
SemiStream	Particional	ML & CL	Não	Stream	[Halkidi, Spiliopoulou e Pavlou 2012]
CE-Stream	FCH	ML	Não	Stream	[Sirampuj, Kangkachit e Waiyamai 2013]
SSE-Stream	FCH	Rótulo	Não	Stream	[Treechalong, Rakthanmanon e Waiyamai 2015]
Incremental SSFCM	Fuzzy	Rótulo	Não	Online	[Castellano, Fanelli e Torsello 2013]
[Lai et al. 2014]	Particional	Feedback	Não	Online	[Lai et al. 2014]
CABESS	Genérico	Feedback	Sim	Online	-

3.5 Evolução dos Agrupamentos

Os conceitos e limites de decisão definidos em um instante de tempo t_i, por exemplo, podem tornar-se obsoletos no instante de tempo t_{i+1}. Portanto, os conceitos no mundo real não são estáveis e isso deve ser considerado. Esse problema é conhecido na literatura de *data streams* como *concept drift* [Brzeziński 2010]. A seguir são apresentados a definição para *concept drift* e os efeitos causados na evolução dos agrupamentos.

Definição 3.5.1 (*Concept drift*)**.** *Concept drift* é a substituição não prevista de uma fonte $\mathcal{S}_1$ de um *stream* (com distribuição de probabilidade $\coprod_{\mathcal{S}_1}$ subjacente) por outra fonte $\mathcal{S}_2$ (com distribuição de probabilidade $\coprod_{\mathcal{S}_2}$ subjacente) [Brzeziński 2010]. Assim, o *concept drift* refere-se a mudança na distribuição de probabilidade da variável alvo, ou seja, quando a distribuição que gera os dados muda em um momento no tempo [Gama et al. 2014].

Entender o comportamento dos agrupamentos ao longo do tempo, sobretudo após alterações no conceito subjacente à estrutura dos agrupamentos, tem sido um importante problema de pesquisa explorado recentemente. O tempo é um fator crucial no contexto de mineração em *data streams*, e quanto o tempo é considerado em agrupamento de dados não é diferente. Diversos trabalhos exploram o impacto da temporalidade na detecção de agrupamentos. Em [Silva et al. 2013] são apresentados alguns aspectos que devem ser considerados quando do projeto de um novo algoritmo, a saber: (a) detecção de agrupamentos sensível ao tempo, (b) evolução de *outliers* e (c) rastreamento de agrupamentos. A seguir é apresentado os aspectos relevantes para esta dissertação referentes ao rastreamento de agrupamentos.

Rastreamento de agrupamentos. Explorar o comportamento e as mudanças ao longo do tempo em janelas pode fornecer uma melhor compreensão sobre o comportamento dinâmico dos agrupamentos. Assim, algoritmos de detecção de agrupamentos devem fornecer uma maneira de examinar as mudanças nos agrupamentos em diferentes granularidades de tempo (por exemplo, diária, mensal ou anual). Observe que, o foco das abordagens que exploram este aspecto não é dedicar esforços adaptando os agrupamentos a medida que os dados evoluem, mas rastrear e compreender a evolução dos agrupamentos em si (*cluster tracking*). A ideia é obter *insights* sobre os dados para apoiar decisões estratégicas, ao fazer uso do rastreamento da evolução dos agrupamentos. Portanto, o objetivo é fornecer *insights* sobre a natureza da mudança em um ou mais grupos e responder a questões como: (1) um grupo desapareceu ou as instâncias migraram para outros grupos? (2) um novo grupo está emergindo de um novo perfil de instâncias ou de instâncias antigas cujas características evoluíram?

Os frameworks MONIC [Spiliopoulou et al. 2006] e MClusT [Oliveira e Gama 2010] são exemplos de trabalhos bem conhecidos que lidam com esse problema. O MONIC modela e rastreia as mudanças nos agrupamentos a fim de auxiliar no entendimento da

natureza da mudança. As transições de agrupamentos são formalizadas e um algoritmo de detecção de transições é proposto. O MClusT constrói um grafo bipartido para modelar as transições dos agrupamentos, onde os vértices representam os agrupamentos e as arestas representam as relações entre pares de agrupamentos em dois períodos distintos no tempo. O uso do grafo objetiva facilitar a visualização da transição dos agrupamentos. Além disso, essa estrutura é usada para calcular a probabilidade condicional associada a cada par que conecta duas arestas (agrupamentos) no grafo bipartido. A ideia é estimar a probabilidade de mudança, ou permanência, das instâncias de cada agrupamento de um instante t_i para um instante t_{i+1}. É importante notar aqui que nenhum desses trabalhos explora a análise da evolução da semissupervisão ao longo do tempo.

A abordagem usada no *framework* MONIC assume que um agrupamento é um objeto em um espaço geométrico. O MONIC usa uma técnica de rastreamento de transição baseada no grau de sobreposição entre dois agrupamentos. O conceito de sobreposição, *overlap*, entre dois agrupamentos é definido como a proporção de instâncias em comum ponderada pela idade das respectivas instâncias, $age : \mathcal{D} \times \mathbb{N} \rightarrow [0, 1]$. Dado dois agrupamentos $\Pi_a^{t_i}$ e $\Pi_b^{t_j}$, o grau de sobreposição entre eles é calculado conforme a Equação 13, onde $\Pi_a^{t_i}$ foi obtido no instance de tempo t_i e $\Pi_b^{t_j}$ no instance t_j, $t_j > t_i$.

$$overlap(\Pi_a^{t_i}, \Pi_b^{t_j}) = \frac{\sum_{a \in \Pi_a^{t_i} \cap \Pi_b^{t_j}} age(a, t_j)}{\sum_{x \in \Pi_b^{t_j}} age(x, t_j)} \tag{13}$$

Assim, uma ***transição de um agrupamento*** no instante de tempo t_{i+1} é uma mudança em um agrupamento no instante de tempo anterior t_i. No entanto, antes de definir e indicar qual tipo de transição pode ter sofrido um agrupamento, é preciso definir qual agrupamento em t_{i+1} corresponde ao agrupamento no instante de tempo anterior t_i, se ele não for um agrupamento novo. Portanto, o MONIC usa uma função $match_\tau$ para obter um mapeamento entre duas partições obtidas em instantes distintos no tempo. O cálculo da função $match_\tau$ é apresentado na Equação 14. Nota-se que dado um agrupamento $\Pi_a^{t_i} \in \Pi^{t_i}$ do instante de tempo anterior, a função $match_\tau$ calcula o grau de sobreposição de $\Pi_a^{t_i}$ sobre cada agrupamento $\Pi_b^{t_i+1} \in \Pi^{t_i+1}$. Assim, a função $match_\tau$ retorna um subconjunto de agrupamentos com grau de sobreposição que respeitem o limiar mínimo τ. Inobstante, o framework MONIC considera dois tipos de transições: internas e externas, que são apresentadas a seguir.

$$match_\tau(\Pi_a^{t_i}, \Pi^{t_i+1}) = \left\{ \Pi_b^{t_i+1} \in \Pi^{t_i+1} : overlap(\Pi_a^{t_i}, \Pi_b^{t_i+1}) \geq \tau \right\} \tag{14}$$

Transições externas. Quando a transição diz respeito a um agrupamento em relação ao restante da partição Π, diz-se que essa transição é do tipo externa. O framework MONIC detecta e explora as seguintes transições externas na evolução dos agrupamentos: nascimento, sobrevivência, morte, divisão e absorção. Na Figura 8 é ilustrado um exemplo indicando esse tipo de transição. De maneira geral, as transições externas são descritas

a seguir com o uso das funções $match_\tau$ e *overlap* juntamente com dois parâmetros τ e $\tau_s split$. O primeiro parâmetro estabelece um limiar mínimo para que o detector considere que um agrupamento sobreviveu ao longo do tempo. O segundo parâmetro estabelece um limiar mínimo necessário para auxiliar o detector a identificar agrupamentos que foram divididos ao longo do tempo, ou seja, o agrupamento não satisfaz o parâmetro τ, mas ainda está em uma faixa intermediária de sobreposição . Assim, supõe-se que $\tau > \tau_{split}$.

- **Sobrevivência.** Considera-se que Π_j^t é um agrupamento sobrevivente a partir de Π_i^{t-1}, se Π_j^t satisfaz o limiar mínimo τ e a função $match_\tau$ não retorna nenhum outro agrupamento, somente Π_j^t.

- **Divisão.** Para que um subconjunto de agrupamentos no instante corrente seja considerado resultante da divisão Π_i^{t-1} deve satisfazer os seguintes critérios: (1) cada agrupamento do subconjunto deve possuir grau de sobreposição superior ao parâmetro τ_{split} com respeito a Π_i^{t-1} e (2) a união dos agrupamentos desse subconjunto deve satisfazer o limiar estabelecido por τ.

- **Absorção.** Considera-se que Π_j^t é o agrupamento Π_i^{t-1} absorvido por outro quando tem-se o mesmo caso da transição de sobrevivência, mas a função $match_\tau$ retorna mais um agrupamento além de Π_j^t.

- **Morte.** Quando não há correspondência para um agrupamento que existia no passado e no instante atual não é mais detectado, então considera-se que o agrupamento morreu (ou desapareceu).

- **Nascimento.** Quando não há correspondência para um agrupamento atual no instante de tempo anterior, então considera-se que o agrupamento atual nasceu (ou surgiu) no tempo corrente.

No Algoritmo 2 é descrito em detalhes a rotina executada pelo framework MONIC para realizar a detecção de transições do tipo externas. Internamente, o Algoritmo 2 calcula e utiliza os indicadores descritos na Tabela 3 para detectar cada tipo de transição.

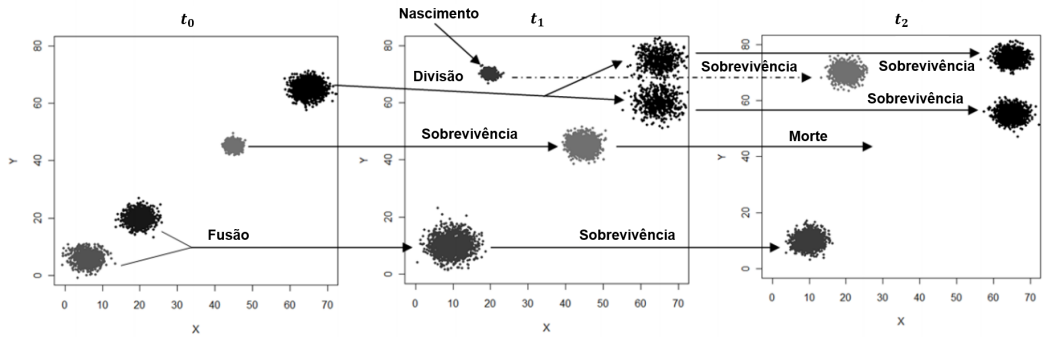


Figura 8 – Exemplo de transições externas (Adaptado de [Oliveira e Gama 2012]).

Algoritmo 2: O detector de transições externas do MONIC. Adaptado de [Spiliopoulou et al. 2006].

Entrada: Π^{t-1}, Π^{t}_{j}: a partição prévia e a partição atual respectivamente.
Saída: Lista com as transições externas detectadas entre Π^{t-1} e Π^{t}_{j}.

```
1  início
2      para Π_i^{t-1} ∈ Π^{t-1} faça
3          splitCandidates ← {};
4          splitUnion ← {};
5          survivalCandidate ← NULL;
6          para Π_j^t ∈ Π^t faça
7              Mcell ← overlap(Π_i^{t-1}, Π_j^t)
8              se Mcell ≥ τ então
9                  se g(Π_i^{t-1}, Π_j^t) > g(Π_i^{t-1}, survivalCandidate) então
10                     survival ← Π_j^t;
11                 se Mcell ≥ τ_split então
12                     splitCandidates ← splitCandidates ∪ Π_j^t;
13                     splitUnion ← splitUnion ∪ Π_j^t;
14         se survivalCandidate = NULL ∨ splitCandidates = {} então
15             deadList ← deadList ∪ Π_i^{t-1};
16         senão se splitCandidates! = {} então
17             se overlap(Π_i^{t-1}, splitUnion) ≥ τ então
18                 para Π_j^t ∈ splitCandidates faça
19                     splitList ← splitList ∪ (Π_i^{t-1}, Π_j^t);
20         senão
21             deadList ← deadList ∪ Π_i^{t-1};
22     para Π_j^t ∈ Π^t faça
23         absorptionCandidates ← makeList(Absorptions'Survivals, Π_j^t);
24         se cardinality(absorptionCandidates) > 1 então
25             for Π_i^{t-1} ∈ absorptionCandidates do
26                 absorbtionList ← absorbtionList ∪ (Π_i^{t-1}, Π_j^t);
27                 Absorptions'Survivals ← Absorptions'Survivals − (Π_i^{t-1}, Π_j^t);
28         senão se absorptionCandidates = Π_i^{t-1} então
29             survivalList ← survivalList ∪ (Π_i^{t-1}, Π_j^t);
30             Absorptions'Survivals ← Absorptions'Survivals − (Π_i^{t-1}, Π_j^t);
31 fim
```

Tabela 3 – Algoritmos de Detecção Semissupervisionada de Agrupamentos. Adaptado de [Spiliopoulou et al. 2006].

Transição	**Notação**	**Indicador**
Sobrevivência	$\Pi_i^{t-1} \rightarrow \Pi_j^t$	$\Pi_j^t = match_\tau(\Pi_i^{t-1}, \Pi^t) \wedge \nexists \Pi' \in \Pi^{t-1} - \Pi_i^{t-1} : \Pi_j^t = match_\tau(\Pi', \Pi^t)$
Divisão	$\Pi_i^{t-1} \overset{\subseteq}{\rightarrow} \{\Pi_1^t, ..., \Pi_p^t\}$	$(\forall u = 1, ..., p : overlap(\Pi_i^{t-1}, \Pi_u^t) \geq \tau_{split}) \wedge$ $overlap(\Pi_i^{t-1}, \cup_{u=1}^p \Pi_u^t) \geq \tau \wedge$ $(\nexists \Pi_j^t \in \zeta - \{\Pi_1^t, ..., \Pi_p^t\} : overlap(\Pi_i^{t-1}, \Pi_j^t) \geq \tau_{split})$
Absorção	$\Pi_i^{t-1} \overset{\subseteq}{\rightarrow} \Pi_j^t$	$\Pi_j^t = match_\tau(\Pi_i^{t-1}, \Pi^t) \wedge \exists \Pi' \in \Pi^{t-1} - \{\Pi_i^{t-1}\} : \Pi_j^t = match_\tau(\Pi', \Pi^t)$
Morte	$\Pi_i^{t-1} \rightarrow \odot$	nenhum dos casos anteriores é valido para Π_i^{t-1}
Nascimento	$\odot \rightarrow \Pi_j^t$	

Transições internas. Transições do tipo interna dizem respeito ao conteúdo e a forma de um agrupamento especificamente, ou seja, esse tipo de transição indica uma mudança no agrupamento por si próprio. São exemplos de transições internas: expansão e encolhimento (transições de tamanho), tornar-se compacto, tornar-se difuso (transições de compacidade) e mudança do centro, mudança de simetria (transições de localidade).

3.6 Considerações Finais

Neste capítulo foram descritos os principais conceitos e trabalhos em rastreamento e evolução de agrupamentos. Além disso, evidenciou-se as diferenças entre as principais pesquisas relacionadas com o trabalho desenvolvido nesta dissertação. Inobstante, enfatizou-se o fato da literatura científica ainda não ter explorado a evolução temporal da semissupervisão e considerar que o usuário é coerente ao longo do tempo.

No próximo capítulo é descrito o trabalho desenvolvido nesta dissertação, isto é, o framework genérico capaz de fazer detecção online de agrupamentos semissupervisionado usando feedbacks.

Capítulo 4

O Framework CABESS

Neste capítulo é apresentado o framework de detecção online de agrupamentos semissupervisionado proposto nesta dissertação, denominado CABESS (*Cluster Adaptation Based on Evolving Semi-Supervision*). Diferentemente da principal referência utilizada para o desenvolvimento do framework e das demais técnicas de detecção de agrupamentos semissupervisionados da literatura, o framework proposto parte do pressuposto que a semissupervisão informada pelo usuário pode não ser coerente ao longo do tempo.

O capítulo está organizado como segue. Inicialmente, na Seção 4.1, o problema é formalizado. Na Seção 4.2 é apresentado o framework e os módulos. Nas Seções 4.3 e 4.4 são apresentadas as duas instanciações do framework, Pointwise CABESS e Pairwise CABESS. Finalmente, na Seção 4.5 são apresentadas as considerações finais sobre este capítulo.

4.1 Formalização do problema

Dada uma sequência $\mathcal{D}$ e um *stream* $\mathcal{F}$, onde $\mathcal{F}$ pode não ser coerente ao longo do tempo, o principal objetivo da detecção online de agrupamentos semissupervisionado é ser capaz de manter e melhorar a qualidade dos agrupamentos ao longo do tempo. Assim, uma técnica de detecção online de agrupamentos semissupervisionada deve detectar e adaptar a partição Π^t, periodicamente retornada ao usuário durante o processo de agrupamentos, usando a informação de semissupervisão a partir de $\mathcal{F}$. Quando uma nova partição Π^{t+1} está disponível para o usuário, ele pode fornecer mais feedbacks que podem não ser coerentes com os feedbacks fornecidos anteriormente ou indicar uma mudança na partição dos dados a ser identificada pelo processo de detecção de agrupamentos.

4.2 Descrição do framework

Para propor o CABESS, um framework genérico que objetiva usar uma quantidade limitada de informação adicional para prover uma detecção online eficaz e eficiente,

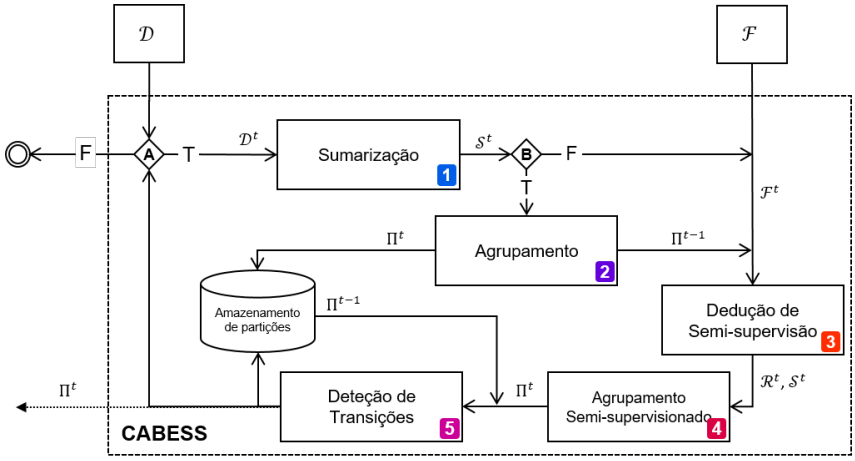


Figura 9 – Esquema geral do framework CABESS. O teste A verifica se novas instâncias estão sendo geradas ou se o usuário está satisfeito com a qualidade do agrupamento detectado (geração de feedbacks). O teste B verifica se é a primeira execução da detecção de agrupamentos.

revisitou-se a técnica de detecção de agrupamentos descrita em [Lai et al. 2014]. Assim como a principal referência desta dissertação, o framework CABESS também recebe como entrada uma sequência maciça de instâncias de dados $(x_1, ..., x_n)$, $\mathcal{D} = \{x_i\}_{i=1}^n$ e uma sequência limitada de feedbacks $\mathcal{F} = \{f_i^j\}_{j=1}^m$, $|\mathcal{F}| \ll |\mathcal{D}|$. Cada instância é descrita por um vetor de características d-dimensional $x_i = [x_i^j]_{j=1}^d$ que pertence a uma espaço de características contínuo Ω. Cada feedback do usuário f_i^j relativo a uma instância $x_i \in \mathcal{D}$ pode ser expresso em dois tipos de feedbacks: **positivo** ou **deslocamento**. No entanto, diferentemente da técnica descrita em [Lai et al. 2014], o framework CABESS não assume que o usuário é coerente no fornecimento de feedbacks ao longo do tempo, pelo contrário, o CABESS foi projetado considerando a mudança de opinião do usuário. Por isso, ele também usa uma abordagem que detecta transições externas em agrupamentos a fim de gerenciar a evolução da semissupervisão ao longo do tempo. O processo de detecção do framework CABESS se encerra quando a partição resultante satisfaz o usuário e quando não há mais instâncias geradas em $\mathcal{D}$.

O processo de detecção de agrupamentos executado pelo CABESS é feito por meio de cinco módulos, a saber: (1) módulo de Sumarização, (2) módulo de Agrupamento Não-supervisionado, (3) módulo de Dedução da Semissupervisão, (4) módulo de Agrupamento Semissupervisionado e (5) módulo de Detecção de Transições. Os módulos foram inspirados nas etapas do processo de detecção semissupervisionada proposto em [Lai et al. 2014]. Cada módulo permite o uso de mais de um algoritmo. Assim, eles fornecem uma abstração que prove uma funcionalidade genérica. Na Figura 9 é apresentado o esquema com o fluxo geral do processo de detecção do framework CABESS. A seguir cada módulo é descrito em detalhes e são apresentadas suas entradas e saídas.

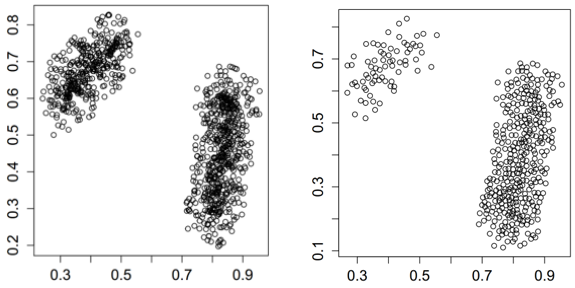


Figura 10 – Resultado obtido sobre um conjunto de dados sintético (direita) após a execução de uma técnica de sumarização (esquerda).

4.2.1 Módulo 1: Sumarização

O processo de detecção inicia-se sumarizando as instâncias usando uma abordagem de detecção de micro-clustering baseada em CF-Vectors no módulo 1. Entre as abordagens que podem ser usadas nesta etapa, é possível mencionar o BIRCH [Zhang, Ramakrishnan e Livny 1996], CluStream [Aggarwal et al. 2003] e DenStream [Cao et al. 2006], por exemplo.

O objetivo deste módulo é reduzir o custo computacional da tarefa de detecção de agrupamentos. Por meio deste módulo é possível diminuir o número n de instâncias em nível sumário que serão processadas nos módulos seguintes. Na Figura 10 é ilustrado o efeito do uso deste módulo sobre um conjunto de dados artificial. Note que, a densidade e o número de instâncias é reduzido. Na Tabela 4 são descritos a entrada e a saída deste módulo.

Tabela 4 – Descrição da entrada e saída do primeiro módulo do CABESS

Módulo 1 - Sumarização das instâncias de dados	
Entrada	Um conjunto de instâncias de dados $\mathcal{D}^t$
Saída	Um conjunto de instâncias em nível sumário $\mathcal{S}^t$

4.2.2 Módulo 2: Agrupamento Não-supervisionado

Considerando que nenhuma informação de feedback está disponível no início da execução, no módulo 2, o framework CABESS faz a detecção de macro-agrupamentos sobre a representação resumida dos dados $\mathcal{S}^t$ usando um algoritmo de detecção de agrupamentos não-supervisionado. Nesta etapa, pode-se empregar os algoritmos DBSCAN ou K-MEANS, por exemplo. É importante ressaltar que este módulo é usado somente uma vez, ou seja, no início da execução, pois o usuário ainda não visualizou uma partição dos dados para fornecer feedbacks a respeito. Na Tabela 5 são descritos a entrada e a saída deste módulo.

Tabela 5 – Descrição da entrada e saída do módulo 2 do CABESS

Módulo 2 - Detecção de agrupamentos não supervisionado	
Entrada	Um conjunto de instâncias em nível sumário $\mathcal{S}^t$
Saída	Uma partição $\Pi^t = \{\Pi_1^t, ..., \Pi_k^t\}$

4.2.3 Módulo 3: Dedução da Semissupervisão

Este módulo é o responsável pelo CABESS permitir ao usuário fornecer feedbacks em nível instância com respeito a partição prévia dos dados Π^{t-1} e gerar informação de semissupervisão em nível sumário para ser usado pela etapa seguinte (módulo 4). Assim, os feedbacks serão usados para ajustar a execução do processo de detecção no próximo instante de tempo $(t+1)$. A fim de fazê-lo, os feedbacks em nível instância são usados para deduzir a informação de semissupervisão em nível sumário $\mathcal{R}^t$. Repare que o feedback fornecido pelo usuário é em nível instância, mas o módulo seguinte processará dados em nível sumário, inclusive a semissupervisão. Assim, faz-se necessário obter a informação de semissupervisão do nível instância para o nível sumário. Como o processo de dedução é uma contribuição desta dissertação, ele será detalhado nas Seções 4.3.1 e 4.3.2. Na Tabela 6 são descritos a entrada e a saída deste módulo.

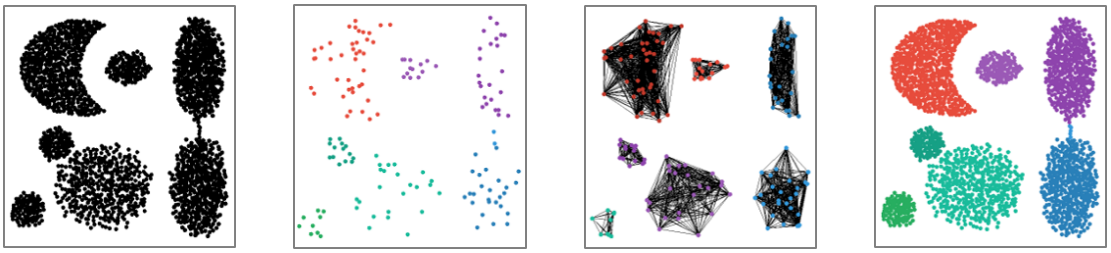
Tabela 6 – Descrição da entrada e saída do módulo 3 do CABESS

Módulo 3 - Dedução da Semissupervisão	
Entrada	A partição prévia $\Pi^{t-1} = \{\Pi_1^{t-1}, ..., \Pi_k^{t-1}\}$ O conjunto corrente de instâncias em nível sumário $\mathcal{S}^t$
Saída	Um mapeamento $\mathcal{R}^t$ que associa a informação de semissupervisão ao subconjunto de $\mathcal{S}^t$

4.2.4 Módulo 4: Agrupamento Semissupervisionado

Após a dedução da semissupervisão, no módulo 4, um algoritmo de detecção de agrupamentos semissupervisionado é usado para re-organizar a partição dos dados. Exemplos de algoritmos elegíveis para serem usados neste módulo são o SSDBScan e o MPCK-Means. Deve-se enfatizar que a semissupervisão requerida pelo algoritmo de detecção de agrupamentos deve ser do mesmo tipo da semissupervisão deduzida no módulo anterior (Dedução da Semissupervisão). Assim, se for escolhido o SSDBScan, no módulo antecessor deve deduzir semissupervisão na forma de rótulos, pois o SSDBScan lida com semissupervisão especificada em rótulos. Na Figura 11 são ilustradas duas possíveis entradas para esse módulo: uma entrada com semissupervisão especificada em rótulos e outra entrada com semissupervisão especificada em restrições ML.

É importante notar que os algoritmos de detecção de agrupamentos executados nos módulos 2 e 3 consideram informação sumarizada obtida a partir dos dados originais. Após o processo de detecção de agrupamentos, o usuário pode fornecer os feedbacks dele. Assim, uma instância de dado sumarizada $s_1 \in \mathcal{S}^t$ em um grupo, na verdade, pode representar mais de uma instância. Consequentemente, uma informação de semissupervisão



(a) (b) (c) (d)

Figura 11 – Entradas possíveis (a+b ou a+c) e uma saída desejada (d) do módulo de detecção de agrupamentos usando semissupervisão. (a+b) Entrada para execução de agrupamento semissupervisionado usando rótulos como semissupervisão. (a+c) Entrada para execução de agrupamento semissupervisionado usando alta taxa de restrições em pares ML como semissupervisão. (d) Uma partição obtida como resultado da execução do módulo.

envolvendo s_1 impõe restrições sobre todas as instâncias representadas por s_1. Na Tabela 7 são descritos a entrada e a saída deste módulo.

Tabela 7 – Descrição da entrada e saída do módulo 4 do CABESS

Módulo 4 - Detecção de agrupamento semissupervisionado	
Entrada	O conjunto corrente de instâncias em nível sumário $\mathcal{S}^t$ O mapeamento corrente $\mathcal{R}^t$ que associa a informação de semissupervisão ao subconjunto de $\mathcal{S}^t$
Saída	A partição corrente $\Pi^t = \{\Pi_1^t, ..., \Pi_k^t\}$

4.2.5 Módulo 5: Detecção de Transições

O último módulo é responsável por detectar transições entre a partição corrente Π^t e a partição anterior Π^{t-1}. O MONIC e o MClust podem ser adotados para serem usados nesta etapa. A ideia por trás do uso deste módulo no CABESS está relacionada com o gerenciamento de semissupervisão em nível sumário. Considerando que os agrupamentos são dinâmicos, este módulo auxilia no gerenciamento do ciclo de vida da semissupervisão, pois da mesma forma que os agrupamentos podem surgir, dividir-se, unir-se e desaparecer ao longo do tempo, a semissupervisão associada também pode apresentar estes comportamentos. O uso deste módulo como gestor da semissupervisão é uma contribuição desta dissertação e é detalhado nas Seções 4.3.2 e 4.4.2. Na Tabela 8 são descritos a entrada e a saída deste módulo.

Tabela 8 – Descrição da entrada e saída do módulo 5 do CABESS

Módulo 5 - Detecção de Transições	
Entrada	A partição corrente $\Pi^t = \{\Pi_1^t, ..., \Pi_k^t\}$ A partição prévia $\Pi^{t-1} = \{\Pi_1^{t-1}, ..., \Pi_k^{t-1}\}$
Saída	Um conjunto $\mathcal{T}$ onde cada elemento é uma dupla representando o tipo de transição e a lista de agrupamentos afetados

4.3 Pointwise CABESS

Para ilustrar uma instanciação do framework CABESS, nesta seção é apresentado o Pointwise CABESS. Assim como a principal referência desta dissertação, esta instanciação do CABESS também emprega o algoritmo BIRCH para sumarizar os dados (módulo 1). O Pointwise CABESS usa o BIRCH para obter instâncias em nível sumário. Após a execução do BIRCH, essa instanciação do framework trabalha sobre os elementos dos nodos folhas da CF-Tree. Cada folha é definida por um vetor-CF. Os demais algoritmos usados no Pointwise CABESS são: no módulo 2, usa-se o DBScan para executar a primeira detecção de agrupamentos sem qualquer semissupervisão; o SSDBScan é adotado para detectar agrupamentos com rótulos em nível sumário na etapa 4; e na última etapa, o framework MONIC é usado para fazer a detecção das transições externas. Essa instanciação do framework CABESS utiliza abordagens de detecção de agrupamentos baseados em densidade, pois são abordagens que não se exigem a priori o número k de agrupamentos, tanto o DBScan quanto o SSDBScan não exigem o número k. Além disso, outra importante questão que justifica a escolha desses algoritmos nesta instanciação é o fato do número k de agrupamentos poder mudar ao longo do tempo. O framework MONIC é adotado pois trata-se de uma estratégias bastante conhecida na literatura. Técnicas mais recentes para detectar transições externas não foram adotadas no trabalho descrito aqui, pois partem do pressuposto de que os agrupamentos possuem formatos bem definidos objetivando melhor a eficácia da detecção. Esse pressuposto não pode ser assumido neste trabalho devido a generalidade com que o problema é tratado nesta dissertação.

O Pointwise CABESS lida internamente com rótulos como informação de semissupervisão. Assim, na próxima seção será explanado como o Pointwise CABESS extrai rótulos em nível instância e como ele deduz rótulos em nível sumário.

4.3.1 Extração de rótulos a partir de feedbacks

Rótulos em nível sumário são necessários para o módulo de detecção semissupervisionada de agrupamentos. No entanto, o usuário fornece a informação de semissupervisão na forma de feedbacks em nível instância. Logo, é preciso transformar a informação em rótulos e sumarizá-la. Rótulos em nível sumário são obtidos em duas fases: primeiro os rótulos em nível instância são extraídos a partir dos feedbacks e então os rótulos em nível sumário são deduzidos a partir dos rótulos em nível instância. Além disso, pode-se deparar, no entanto, com vários problemas ao tentar executar esse processo. (1) Como gerenciar essa informação ao longo do tempo, isto é, deve-se descartar rótulos e/ou feedbacks? Se sim, quando deve-se descartar a semissupervisão? (2) Uma instância sumarizada pode representar duas ou mais instâncias de dados que receberam feedbacks distintos. Então, como deve-se extrair os rótulos neste contexto?

O algoritmo que faz a extração dos rótulos a partir dos feedbacks no Pointwise CA-

BESS descarta os feedbacks após a extração dos rótulos. O gerenciamento dos rótulos é feito usando uma janela deslizante no módulo de dedução juntamente com o módulo de detecção de transições. Isso quer dizer que se um rótulo torna-se antigo, então ele ficará fora dos limites da janela deslizante e será removido independentemente da utilidade para o processo de agrupamento, minimizando o custo computacional em processar toda a informação de semissupervisão acumulada. O gerenciamento dos rótulos feito no módulo de detecção de transições será detalhado na Seção 4.3.2, pois trata-se de uma contribuição do trabalho descrito nesta dissertação.

1. **Rótulos em nível instância**. A principal ideia por trás do processo de extração está relacionado com o conceito de vizinhança apresentado no capítulo anterior, Seção 3.3.1. Para cada vizinhança computada é atribuído um rótulo único a todas as instâncias de dados que compõem essa vizinhança. Se for uma vizinhança nova, então um novo rótulo é gerado e atribuído. Caso contrário, se for uma vizinhança que possui instâncias previamente rotuladas no passado, então o rótulo é propagado para as instâncias novas na vizinhança. Pode-se notar na Figura 12b que o rótulo 1 será atribuído à vizinhança associada ao agrupamento Π_1 no final do processo (Figura 12d).

2. **Rótulos em nível sumário**. A dedução de rótulos em nível sumário é executada como uma tarefa de propagação. Para cada instância sumarizada que contém instâncias rotuladas, atribui-se o rótulo da instância de dado para o seu sumário, ou seja, a instância sumarizada. Esse rótulo é agora chamado de rótulo em nível sumário. Note que na Figura 12d os vetores-CF CF_1 e CF_2 recebem o rótulo 1, pois eles resumem instâncias que estão na vizinhança N_1.

   Nesta etapa, pode-se observar o segundo problema apontado no início desta seção: se uma instância em nível sumário pode representar mais de uma instância de dado, então ela pode representar instâncias que receberam feedbacks distintos no instante de tempo corrente ou instâncias que foram rotuladas no passado cujo rótulo é oriundo de uma segunda vizinhança. Logo, diz-se que essas instâncias sumarizadas não estão puras. Para deduzir ou atualizar o rótulo em nível sumário é necessário dividir a instância sumarizada a fim de obter instâncias em nível sumário puras, isto é, instâncias sumarizadas que contenham instâncias de dados rotuladas igualmente. O processo de divisão consiste em inserir um novo vetor-CF na mesma folha da CF-Tree que contém o vetor-CF original usando uma abordagem *bottom-up*. O novo vetor-CF recebe as instâncias de dados do vetor-CF original que receberam feedback e/ou pertencem a uma segunda vizinhança. Uma função de distância[1] é usada para determinar quais instâncias de dados que não receberam feedbacks e

[1] Nas instanciações propostas nesta dissertação a distância Euclidiana é adotada na processo de purificação dos vetores-CF.

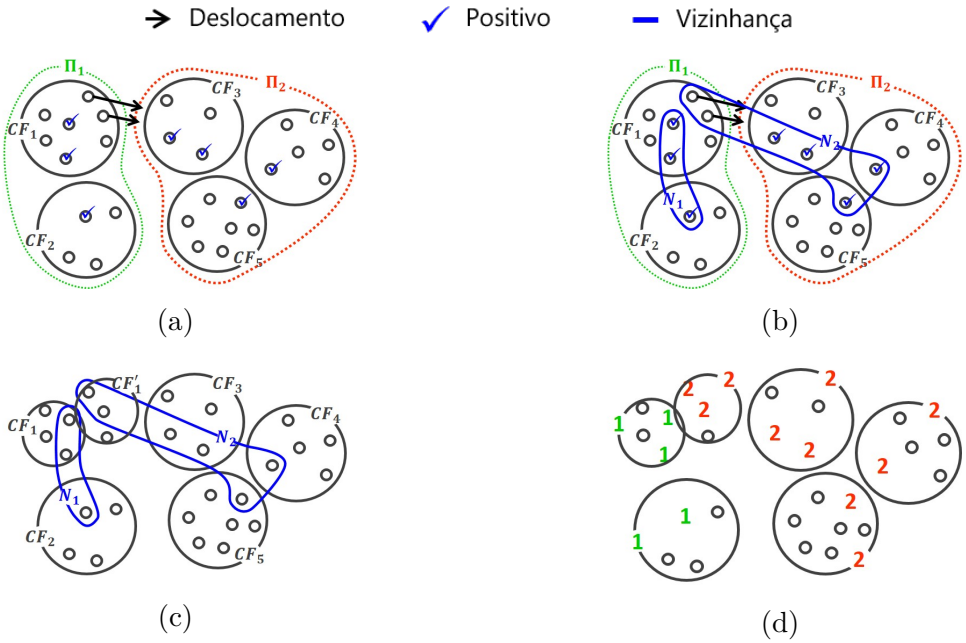


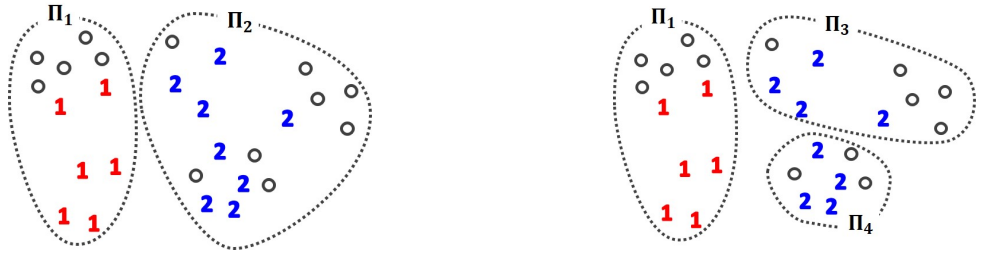
Figura 12 – Exemplo da extração de rótulos a partir dos feedbacks (a,b) e a dedução de rótulos a partir do nível instância para o nível sumário (c,d).

não foram rotuladas no passado devem permanecer no vetor-CF original e quais devem ir para o novo vetor-CF. A ideia é que as instâncias dos dados que receberam feedback ou rótulo que são incompatíveis funcionem como polos que atraem as demais instâncias de dados no processo de divisão do vetor-CF. Somente após o processo de divisão da instância sumarizada e a atualização da CF-Tree construída pelo BIRCH, pode dar prosseguimento ao processo de dedução de rótulos em nível sumário. É possível notar nas Figuras 12b e 12c que antes de atribuir os rótulos, o vetor-CF CF_1 é dividido em dois vetores-CF, CF_1 e CF_1'. Isso acontece, pois o vetor-CF CF_1 original representava instâncias que estavam nas vizinhanças N_1 e também na vizinhança N_2.

4.3.2 Lidando com rótulos obsoletos

Rótulos obsoletos são rótulos atribuídos a instâncias cujos agrupamentos não existem mais. As instâncias de dados ainda permanecem ativas, mas elas são atribuídas a outros agrupamentos. Esses rótulos tornam-se obsoletos quando uma transição de agrupamento externa acontece (Figura 13). Por exemplo, suponha que exista um agrupamento Π_2^0 em um instante de tempo $t = 0$ e suponha que, no próximo instante de tempo, Π_2^0 divida-se nos dois novos agrupamentos Π_3^1 e Π_4^1. Então, o rótulo associado a Π_2^0 não é válido em $t = 1$, pois ele não ajuda o processo semissupervisionado de detecção de agrupamentos.

A fim de minimizar o problema de rótulos obsoletos, o CABESS adota um detector de transições. Essa abordagem objetiva permitir um melhor gerenciamento das vizinhanças. Vizinhanças são responsáveis por gerar novos rótulos e remover rótulos obsoletos. Assim,



(a) (b)

Figura 13 – Exemplo do impacto de uma transição sobre a utilidade da semissupervisão especificada em rótulos. (a) No instante $t = 0$, a partição corrente é $\Pi^0 = \{\Pi_1^0, \Pi_2^0\}$ e todos os rótulos são úteis para o processo de detecção de agrupamentos. (b) No instante de tempo seguinte, $t = 1$, Π_2^0 se divide em dois agrupamentos Π_3^1, Π_4^1 e os rótulos "2" se tornam obsoletos.

quando detecta-se que um agrupamento sobreviveu com o passar do tempo, a vizinhança e os rótulos associados são preservados. Quando detecta-se uma transição do tipo divisão, isto é, um agrupamento se divide ao longo do tempo, os rótulos associados com o agrupamento anterior são removidos e a vizinhança é divida para criar duas ou mais vizinhanças novas e, então, novos rótulos são gerados de acordo. Este é um ponto importante, pois nesta dissertação faz-se um novo uso do detector de transições. Diferentemente da literatura, que detecta as transições para fazer uma análise global da evolução temporal dos agrupamentos, aqui as transições são tratadas como evidências que auxiliam o processo de manutenção da semissupervisão.

4.4 Pairwise CABESS

A fim de ilustrar uma segunda instanciação do framework CABESS, nesta seção é apresentado o Pairwise CABESS. Esta instanciação do CABESS também emprega o algoritmo BIRCH para sumarizar os dados (módulo 1) e o framework MONIC é usado para fazer a detecção das transições externas (módulo 5). No entanto, diferentemente do Pointwise que lida internamente com semissupervisão especificada em rótulos, o Pairwise CABESS faz uso de semissupervisão especificada em restrições ML e CL. É importante ressaltar que o *input* de semissupervisão do framework são os feedbacks do usuário, no entanto, cada instanciação pode lidar com a semissupervisão especificada em um tipo distinto por meio do módulo de dedução de semissupervisão. Portanto, a segunda instanciação do CABESS faz a dedução e lida internamente com restrições ML e CL. Assim, os demais algoritmos usados no Pairwise CABESS são: no módulo 2, usa-se o XMeans para executar a primeira detecção de agrupamentos sem qualquer semissupervisão e o MPC-KMeans é adotado para detectar agrupamentos usando pares ML e CL em nível sumário no módulo 4. Como apresentado anteriormente, o algoritmo XMeans não necessita do conhecimento a priori do número k de agrupamentos, ele exige um intervalo de

valores possíveis para o número k, o que relaxa a exigência para o *input* e permite a sua adoção na primeira etapa do processo de detecção. O algoritmo MPC-KMeans é usado nesta instanciação pois, assim como o XMeans, ele também é um algoritmo de detecção baseado em particionamento e apresenta desempenho superior ao clássico algoritmo COP-Kmeans, incorporando o aprendizado da função de distância juntamente com a semissupervisão.

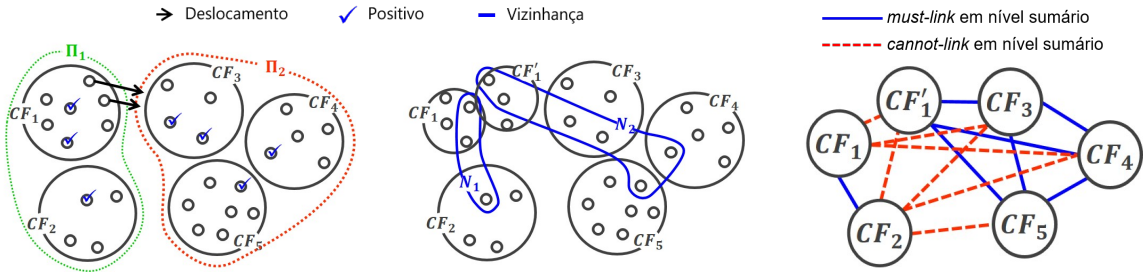
O Pairwise CABESS lida internamente com restrições especificadas em pares ML e CL como informação de semissupervisão. Na próxima seção será explanado como o Pairwise CABESS extrai restrições em nível instância e como ele deduz restrições em nível sumário.

4.4.1 Extração de restrições a partir de feedbacks

Assim como na primeira instanciação, as restrições em nível sumário são obtidos em duas fases: primeiramente, as vizinhanças são computadas e atualizadas a partir dos feedbacks e, então, as restrições em nível sumário são deduzidas a partir das vizinhanças.

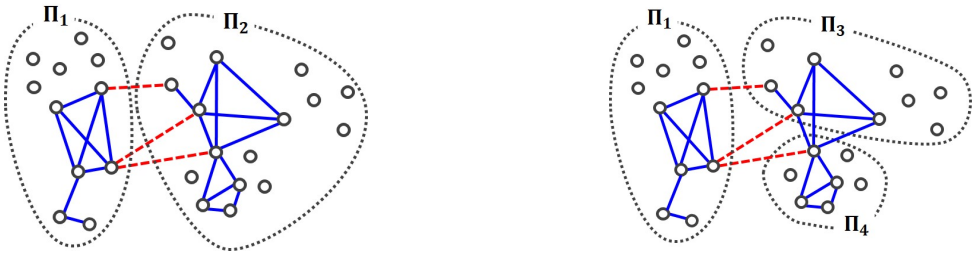
1. **Restrições em nível instância**. Na primeira fase, as vizinhanças são computadas e atualizadas como no Pointwise CABESS (Figura 14b). No entanto, diferentemente de atribuir rótulos para as instâncias de dados, nesta instanciação, os pares ML são gerados entre todas as instâncias de dados que compõem a mesma vizinhança e os pares CL são gerados entre todas as instâncias de dados que pertencem a vizinhanças distintas . A ideia é que cada vizinhança dará origem a uma componente conexa de pares ML e os pares CL serão atribuídos entre instâncias que pertencem a diferentes vizinhanças.

2. **Restrições em nível sumário**. Nesta etapa os pares ML são gerados entre todas as instâncias em nível sumário que pertencem a uma mesma vizinhança, isto é, se existe um par ML entre uma instância de dado em um vetor-CF CF_i e outra instância de dado em CF_j, então uma restrição ML é atribuída entre os vetores-CF CF_i e CF_j. De maneira análoga, os pares CL são gerados entre instâncias em nível sumário que pertencem a vizinhanças distintas (Figura 14c). É importante ressaltar que neste cenário também pode ser necessário dividir, de maneira análoga ao cenário com rótulos, os vetores-CF que representem instâncias de dados com restrições incompatíveis. Assim, se um vetor-CF representa instâncias de dados que pertencem a diferentes vizinhanças, então, na primeira etapa, uma restrição CL será atribuída entre as instâncias. Consequentemente, é necessário dividir o vetor-CF evitando a existência de restrições CL entre instâncias de dados representadas por uma mesma instância sumarizada. Em seguida, o processo de dedução ocorre conforme a regra definida no início deste parágrafo.



(a) (b) (c)

Figura 14 – Exemplo da extração e dedução de pares em nível sumário a partir dos feedbacks. (a) Configuração da partição e os feedbacks fornecidos pelo usuário. (b) Vizinhanças computadas a partir dos feedbacks. (c) Pares ML e CL em nível sumário deduzidos.



(a) (b)

Figura 15 – Exemplo do impacto de uma transição sobre a utilidade da semissupervisão especificam e restrições CL e ML. (a) No instante $t = 0$, a partição corrente é $\Pi^0 = \{\Pi_1^0, \Pi_2^0\}$ e todos os rótulos são úteis para o processo de detecção de agrupamentos. (b) No instante de tempo seguinte, $t = 1$, Π_2^0 se divide em dois agrupamentos Π_3^1, Π_4^1 e os pares ML entre Π_3^1 e Π_4^1 se tornam obsoletos.

4.4.2 Lidando com restrições obsoletas

Restrições ML e CL também podem tornar-se obsoletas quando uma transição do tipo divisão ou união ocorre. Por exemplo, suponha que o agrupamento Π_1^0 em $t = 0$ se divide em dois agrupamentos Π_{10}^1 e Π_{11}^1 no próximo instante de tempo. Então, os pares ML que envolvem instâncias entre Π_{10}^1 e Π_{11}^1 são incoerentes em $t = 1$. De maneira análoga, pode-se notar o mesmo problema envolvendo pares CL após uma transição de união. Portanto, o uso do detector de transições também é importante na segunda instanciação.

É importante ressaltar que no caso de restrições ML e CL, após uma transição de divisão ou união, alguns pares podem continuar sendo úteis para o processo de detecção de agrupamentos. Note que, isso não ocorre no caso da semissupervisão em rótulos. Após uma divisão de um agrupamento em dois ou mais, todos os rótulos referentes ao agrupamento prévio são invalidados. Analisando as Figuras 13 e 15 é possível notar a diferença: no cenário com pares ML e CL, somente um subconjunto de restrições torna-se obsoleto, enquanto as demais restrições ainda auxiliam no processo de agrupamento. No entanto, no cenário com rótulos todos os rótulos referente ao agrupamento Π_2 tornam-se inválidos após a divisão de Π_2 em Π_3 e Π_4.

4.5 Considerações Finais

Este capítulo descreveu os detalhes referentes ao framework CABESS e suas duas instanciações, Pointwise CABESS e Pairwise CABESS, propostos nesta dissertação para execução online de detecção de agrupamentos semissupervisionados. O framework considera que, diferentemente da literatura, o usuário pode ser incoerente ao longo do tempo ao fornecer seus feedbacks. Além disso, um ponto importante, é que o framework permite explorar diferentes tipos de semissupervisão. E, finalmente, o CABESS faz o uso inédito do detector de transições para gerenciar a evolução da semissupervisão, e não somente usar a detecção de transições para analisar o comportamento geral dos agrupamentos. No próximo capítulo são apresentados os experimentos realizados e análise sobre os resultados obtidos.

Capítulo 5

Experimentos e Análise dos Resultados

Neste capítulo são apresentados os experimentos realizados com duas instanciações do framework CABESS, Pointwise CABESS e Pairwise CABESS, e mais seis abordagens de detecção de agrupamentos. O capítulo está organizado como segue. Na Seção 5.1 são descritos os métodos utilizados para a realização dos experimentos. Na Seção 5.2 são apresentados os resultados obtidos e análises referentes a eficácia e eficiência do framework proposto e das demais abordagens. Finalmente, a Seção 5.3 traz as considerações finais deste capítulo.

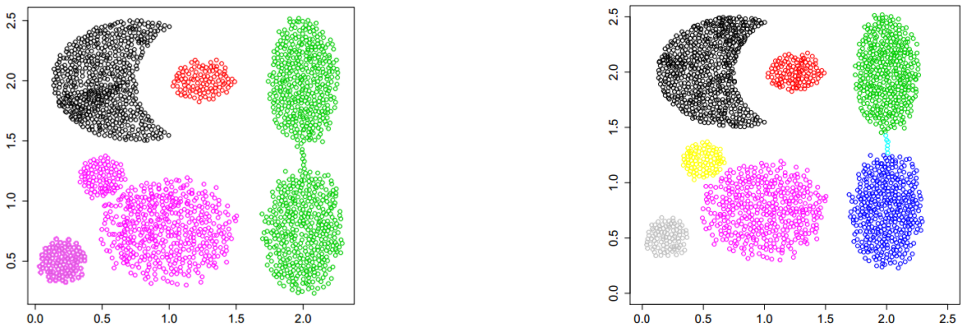
5.1 Método de Avaliação

Conjuntos de dados. Os experimentos foram conduzidos sobre 7 conjuntos de dados[1]: 4 contendo dados reais e 3 contendo dados sintéticos. Os detalhes sobre cada um são sumarizados na Tabela 9. Todos os conjuntos de dados compartilham uma característica em comum que permite executar os experimentos, isto é, cada instância é multi-rotulada conforme uma estrutura hierárquica. As particularidades de cada conjunto de dados são descritas a seguir.

- *SYN3 e SYN4.* Entre os conjuntos de dados sintéticos, com distribuição Gaussiana, SYN3 e SYN4 foram gerados usando o gerador `RandomRBFGenerator` disponível no `streamMOA`[2], uma interface do MOA[3] (*Massive Online Analysis*) desenvolvido para a linguagem de programação R.

- *DB7.* Considerando outra distribuição para gerar dados sintéticos, este é um conjunto de dados sintéticos 2D construído por grupos de diferentes distribuições espaciais. As hierarquias dos agrupamentos foram simuladas nos dados sintéticos de

[1] Disponíveis em: http://guilhermealves.eti.br/research/data/
[2] Documentação disponível em: https://cran.r-project.org/web/packages/streamMOA/streamMOA.pdf.
[3] Framework para mineração de dados em fluxo contínuo: http://moa.cms.waikato.ac.nz/



(a) (b)

Figura 16 – As possíveis estruturas de agrupamentos consideradas nos experimentos envolvendo o conjunto DB7: (a) estrutura generalizada e (b) estrutura especializada.

acordo com a distância entre os grupos. Assim, considerou-se que dois ou mais agrupamentos pertencem a um agrupamento maior se eles estão mais próximos entre si do que outros agrupamentos do conjunto.

- *IPEA.* Neste conjunto[4], cada tupla corresponde a uma das 5564 cidades brasileiras associada com o rótulo da unidade da federação e a região que ela faz parte. Cada cidade é descrita por 5 características como informação de localização (latitude e longitude) e os índices de desenvolvimento: IDHM-Longevidade, IDHM-Educação e IDHM-Renda.

- *FROGS.* Cada tupla neste conjunto de dados corresponde a uma gravação de áudio de um *Anura*, uma ordem da classe dos anfíbios, e é composta por informação sobre espécie, gênero e família.

- *KDD'99*$_5$. Este conjunto de dados reais é uma amostra gerada a partir do conjunto KDD Cup 99[5]. Uma amostragem de 5% foi feita sobre o conjunto que fora construído dos 10% das instâncias de dados do conjunto original, mantendo a proporção das instâncias em cada classe. Além disso, o conjunto amostrado foi processado usando a técnica PCA para reduzir a quantidade de atributos pois trata-se do conjunto com o maior número de dimensões. Considerou-se dois níveis de rótulos para o KDD'99$_5$: (1) se o acesso é uma intrusão ou não e (2) se for uma intrusão, que tipo de intrusão se trata.

- *YEAST.* Neste conjunto de dados reais, cada tupla descreve uma célula de levedura. São ao todo 10 tipos de células. Em [Horton e Nakai 1996] é apresentado uma hierarquia subjacente as classes desse conjunto de dados. Os experimentos descritos nesta dissertação consideraram uma subárvore da hierarquia original a fim de garantir um número considerável de instâncias de dados por agrupamento.

[4] Instituto de Pesquisa Econômica Aplicada, IPEA: http://www.ipeadata.gov.br
[5] UCI KDD archive: http://kdd.ics.uci.edu/

Tabela 9 – Conjunto de dados empregados nos experimentos.

Nome	# instâncias	d	# classes	Referência	Tipo
DB_7	9,050	2	8	[Silva 2015]	Sintético
SYN_3	5,000	2	3	streamMOA	
SYN_4	10,000	3	5	streamMOA	
FROGS	1,484	8	4	[Colonna, Gama e Nakamura 2016]	Real
IPEA	5,564	5	27	IPEA	
$KDD'99_5$	24,692	19	11	UCI	
YEAST	1484	8	10	UCI	

Métodos de comparação. Nos experimentos, compara-se as duas instanciações do framework CABESS com três abordagens de detecção de agrupamentos: uma abordagem não supervisionada e duas abordagens semissupervisionadas. Cada abordagem é detalhada a seguir. A escolha delas leva em consideração o tipo de algoritmo e de semissupervisão, no caso das abordagens semissupervisionadas, usado pelas instanciações do framework CABESS. Em outras palavras, compara-se o Pointwise CABESS com estratégias de agrupamento por densidade e que lidem com semissupervisão especificada em rótulos e, de maneira análoga, compara-se o Pairwise CABESS com estratégias de agrupamento por particionamento e que lidem com restrições ML e CL. Além disso, as abordagens semissupervisionadas foram executadas considerando duas estratégias temporais para a semissupervisão: estática e baseada em janela.

- *Abordagem não supervisionada.* A abordagem não supervisionada consiste na execução periódica de um algoritmo de detecção de agrupamentos de dados periodicamente. Nos experimentos, adotou-se os algoritmos: DBScan (agrupamento por densidade) e X-Means (agrupamento por particionamento).

- *Abordagem estática.* Consiste em executar periodicamente um algoritmo de detecção de agrupamentos semissupervisionado. Nos experimentos, adotou-se o algoritmo SSDBScan (agrupamento por densidade) e MPC-KMeans (agrupamento por particionamento) provendo os rótulos verdadeiros e os pares ML e CL como semissupervisão respectivamente. Note que esta abordagem não descarta qualquer rótulo ao longo do tempo.

- *Abordagem baseada em janela.* É uma variação da abordagem anteriormente apresentada, em que ao invés de executar o algoritmo de detecção de agrupamentos semissupervisionado sobre todo o conjunto de semissupervisão, a abordagem remove a informação de semissupervisão (rótulos e pares ML e CL) antiga de acordo com uma janela (*window-based*).

Métrica de avaliação. A fim de avaliar a eficácia das abordagens, comparou-se os resultados dos agrupamentos obtidos com a partição ótima usando o Índice Rand Ajustado (ARI), descrito na Seção 2.1.3. ARI é uma métrica de critério externo indicado quando

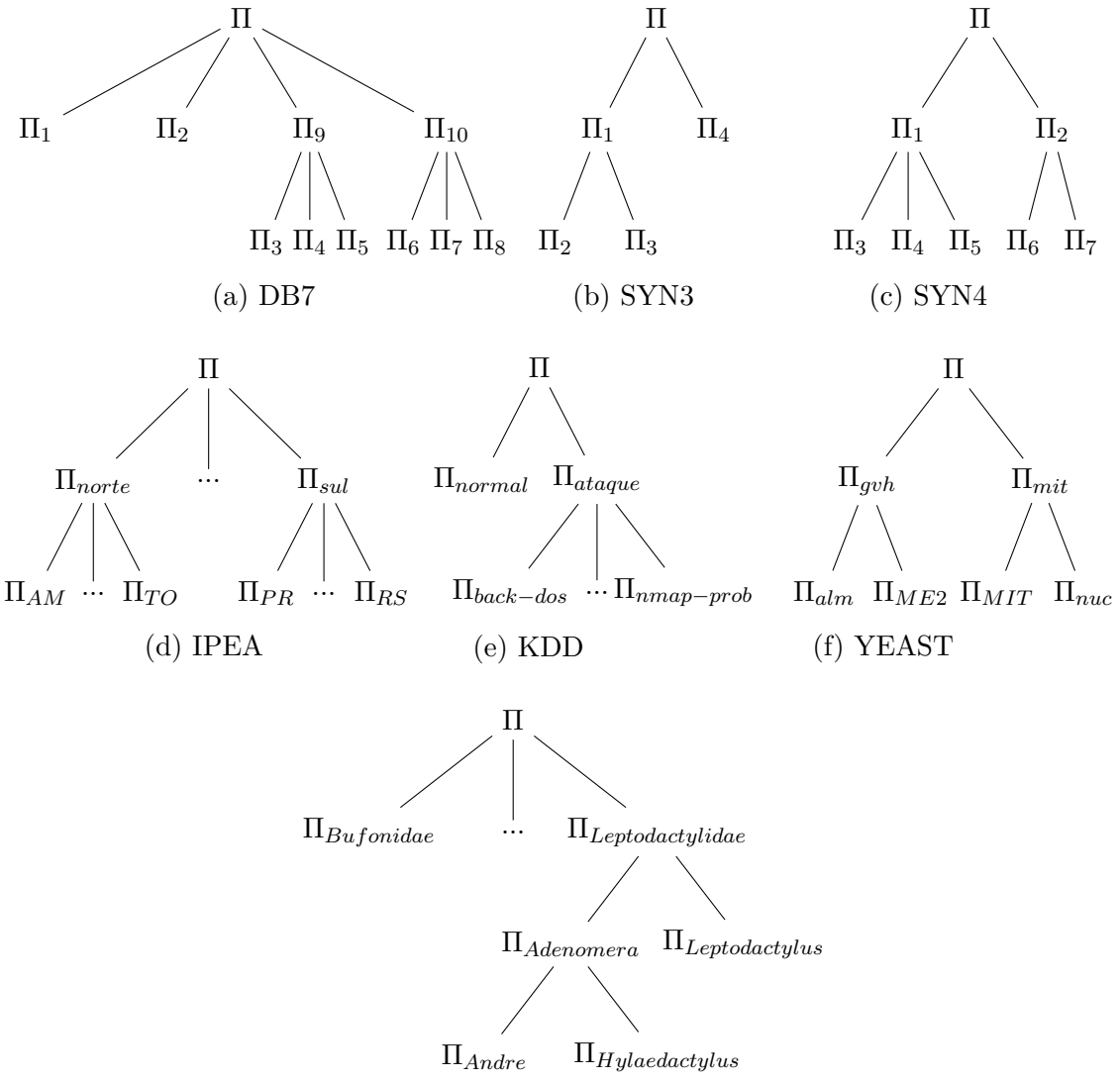
têm-se *a priori* informações sobre a partição desejada. Nos experimentos, a partição desejada é obtida a partir dos rótulos reais dos conjuntos de dados. Para cada timestamp t somente um rótulo é considerado válido para uma instância de acordo com a árvore de agrupamentos.

Protocolo experimental. Cada experimento sobre o Pointwise CABESS e o Pariwise CABESS foi realizado baseado no protocolo *Prequential*. Consequentemente, a eficácia do Pointwise CABESS foi avaliada antes do usuário dar seus feedbacks correntes. As outras abordagens foram avaliadas periodicamente, sem atraso entre a nova partição e os novos rótulos ou restrições.

O protocolo *Prequential* foi desenvolvido para testar os modelos sobre dados ainda não vistos no treinamento, ou seja, é mais adequado para avaliar *streams* que possuem ocorrência de *concept drift*. O *Prequential* usa a estratégia de testar o modelo primeiro e, só então, treinar o modelo com a nova instância do *stream*, tornando-o mais pessimista [Gama 2010].

Considerando que os conjuntos de dados não possuem informação da chegada das instâncias e feedbacks, simulou-se o aspecto temporal das instâncias e da fase de interação do usuário para obter a semissupervisão, assim como a principal referência do trabalho descrito nesta dissertação [Lai et al. 2014]. Consequentemente, a chegada das instâncias e dos feedbacks do usuário seguem uma distribuição uniforme. Além disso, inseriu-se um tipo de transição externa, conforme a árvore de agrupamentos (ver Figura 17), no instante $t = 5$ a fim de avaliar a habilidade das abordagens de se adaptarem ao(s) novo(s) agrupamento(s). Para cada conjunto de dados: (1) um ou mais grupos são especializados (transição de divisão) em dois ou mais novos grupos, ou (2) dois ou mais grupos são generalizados (transição de absorção) em um grande grupo, e os feedbacks são gerados de acordo com os novos grupos. Por exemplo, a especialização dos grupos implica que os feedbacks são gerados considerando que as instâncias que no instante de tempo anterior pertenciam ao agrupamento Π_9 da Figura 17a, pertencem no instante de tempo corrente a um dos novos agrupamentos: Π_3, Π_4 ou Π_5. Com efeito, a árvore de agrupamentos é expandida de forma que a antiga estrutura baseada somente no agrupamento Π_9 não é mais considerada adequada. De forma análoga, a simulação dos feedbacks a partir da generalização dos agrupamentos implica em uma contração da árvore de agrupamentos. Assim, o usuário vai avaliar a partição obtida pelo processo de detecção de agrupamentos considerando o agrupamento mais geral, penalizando o algoritmo por instâncias que deveriam ter sido atribuídas ao mesmo grupo, mas pertencem a grupos distintos.

Configuração dos parâmetros. Na Tabela 10 são apresentados os valores usados nos parâmetros dos algoritmos BIRCH, DBScan e X-Means para cada conjunto de dados. É importante ressaltar que como o algoritmo X-Means não requer o valor fixo k de agrupamentos, ele exige, no entanto, um intervalo de valores possíveis para k e determina o melhor valor dentro desse intervalo. Assim, a Tabela 10 contém os intervalos



(a) DB7 (b) SYN3 (c) SYN4

(d) IPEA (e) KDD (f) YEAST

(g) FROGS

Figura 17 – Árvore de agrupamentos associadas a cada conjunto de dados. Alguns ramos foram suprimidos para facilitar a visualização. As árvores dos conjuntos KDD, YEAST e FROGS são subárvores dos conjuntos originais. Os conjuntos originais possuem agrupamentos com baixos números de instâncias de dados inviabilizando o uso de técnicas de detecção semissupervisionada de agrupamentos, devido ao desbalanceamento das classes.

Tabela 10 – Configuração dos parâmetros do BIRCH para cada conjunto de dados.

Algoritmo	Parâmetro	IPEA	KDD'99$_5$	FROGS	DB$_7$	SYN$_3$	SYN$_4$	YEAST
BIRCH	B e L	500	10k	500	500	50	50	500
	T	0.07	0.01	0.25	0.01	0.0075	0.0075	0.025
DBSCAN	*eps*	0.075	0.02	0.5	0.075	0.075	0.075	0.25
X-MEANS	# grupos	[5,27]	[2,11]	[4,8]	[4,8]	[2,3]	[2,5]	[2,10]

usados para execução do X-MEANS. Para os outros algoritmos, os parâmetros adotados são descritos a seguir. No módulo 4, o SSDBSCAN é executado usando-se o mesmo valor do DBSCAN para $minPts$, i.e., $minPts = 2$. O valor usado para $minPts$ é baixo, pois como cada instância sumarizada pode resumir a informação de várias instâncias de dados. Consequentemente, poucas instâncias sumarizadas são suficientes para caracterizar um agrupamento. No último módulo, o MONIC é utilizado com a seguinte configuração: $\tau = 0.7$ e $\tau_{split} = 0.1$. Os valores para os limiares do detector de transições do MONIC são variados em um conjunto de experimentos especificamente projetado para avaliar o impacto desses limiares na eficácia da detecção de agrupamentos. No entanto, baseando-se nos resultados experimentais apresentados no artigo que apresenta o MONIC [Spiliopoulou et al. 2006], adotou-se os valores anteriormente mencionados por apresentarem resultados satisfatórios na literatura.

Todos os experimentos foram implementados dentro da mesma plataforma usando a linguagem de programação Java. Os experimentos foram executados em uma máquina equipada com processador Intel Core i7 (3.4 GHz) com 12 GB de RAM, um HD SATA3 de 1.31 TB (7200 rpm) rodando Windows 7 x64.

5.2 Experimentos

Nesta seção são descritos os experimentos e apresentadas as análises dos resultados obtidos por meio deles. De maneira geral, o objetivo dos experimentos foi explorar e avaliar a eficácia e eficiência das instanciações do framework proposto, Pointwise CABESS e Pairwise CABESS, em comparação com as demais abordagens, referidas aqui como *baselines*. Além disso, os experimentos foram divididos em cinco grupos de forma a estressar as técnicas de detecção de agrupamentos em diferentes situações e auxiliar a análise. Especificamente, os conjuntos de experimentos foram projetados e realizados objetivando responder as seis perguntas de pesquisa apresentadas na Capítulo 1. Assim, cada conjunto de experimentos responde a pelo menos uma pergunta. É importante ressaltar que todos os conjuntos de experimentos realizados guiam as análises para responder as perguntas ***Q5*** e ***Q6***. Portanto, ***Q5*** e ***Q6*** são as únicas perguntas de pesquisa que não possuem um conjunto de experimentos específicos projetado para respondê-las.

5.2.1 Avaliação da utilidade da semissupervisão

O primeiro conjunto de experimentos objetivou avaliar se o uso de pouca semissupervisão auxilia de fato a detecção dos agrupamentos no cenário investigado nesta dissertação. Ou seja, esses experimentos buscam responder a pergunta ***Q1***: *quanto a semissupervisão auxilia na eficácia da detecção de agrupamentos quando ocorrem transições?*. Assim, os experimentos desta seção foram executados com o propósito de comparar a eficácia da detecção entre um algoritmo não-supervisionado e um algoritmo semissupervisionado usando 1% do total das instâncias dos conjuntos de dados como conjunto de semissupervisão (baixa taxa de semissupervisão) e considerando um tipo de transição externa: divisão ou união. Avaliar a utilidade da semissupervisão é uma questão importante para determinar se a detecção de agrupamentos pode se beneficiar da semissupervisão. Os resultados desses experimentos são retratados nos gráficos das Figuras 18, 19, 20 e 21 e analisados a seguir.

Nas Figuras 18 e 19 são apresentados os valores de ARI considerando um cenário onde os agrupamentos se especializaram ao longo do tempo (transições do tipo divisão) e nas Figuras 20 e 21 os agrupamentos se generalizaram ao longo do tempo (transições do tipo absorção).

De maneira geral, com uma baixa taxa de semissupervisão, é possível perceber que a semissupervisão consegue ajudar de maneira satisfatória no cenário onde os algoritmos de detecção de agrupamentos são baseados em densidade (Figuras 18 e 20). Há uma exceção que é o conjunto KDD'99_5, onde a semissupervisão melhorou a eficácia do processo de agrupamento antes da divisão em $t = 5$. No entanto, quando considera-se somente os algoritmos baseados em particionamento (Figuras 19 e 21) não há unanimidade com relação ao auxílio da semissupervisão no processo de detecção de agrupamentos. Analisando detalhadamente é possível notar que uma baixa taxa de semissupervisão não consegue auxiliar de maneira satisfatória o processo de detecção de agrupamentos quando o conjunto de dados é pequeno. Essa situação, onde o algoritmo semissupervisionado não foi nem equivalente ao algoritmo não-supervisionado, é visto nos conjuntos IPEA, YEAST e FROGS (Figuras 19(a,c,f) e 21(a,c,f)).

Então, os resultados obtidos corroboram que uma baixa taxa de semissupervisão só é suficiente para uma obtenção mais eficaz da detecção de agrupamentos quando se tem conjuntos que não possuem uma pequena quantidade de instâncias. Conjuntos com poucas instâncias implicam em um baixo número de instâncias com informação de semissupervisão. Agrupamentos formados por poucas instâncias, por exemplo, podem inclusive não ter nenhuma instância com informação de semissupervisão. Portanto, a fim de estressar as técnicas semissupervisionadas de agrupamento, a taxa de semissupervisão adotada será incrementada nos próximos conjuntos de experimentos.

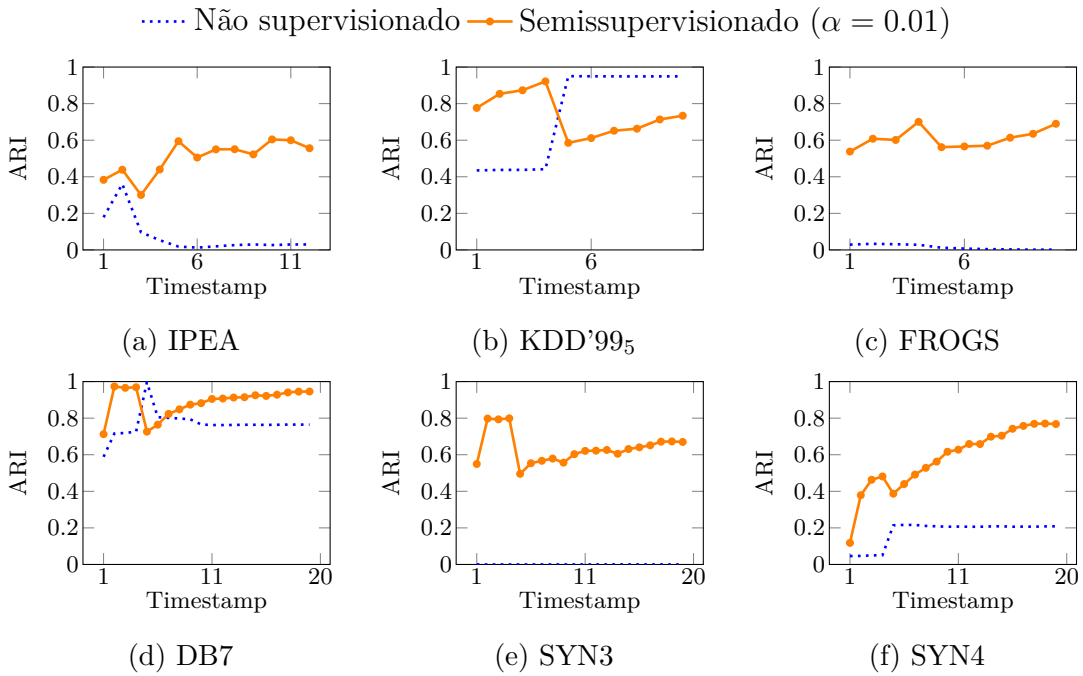


Figura 18 – Avaliação da eficácia: semissupervisão em rótulos e especialização de grupos.

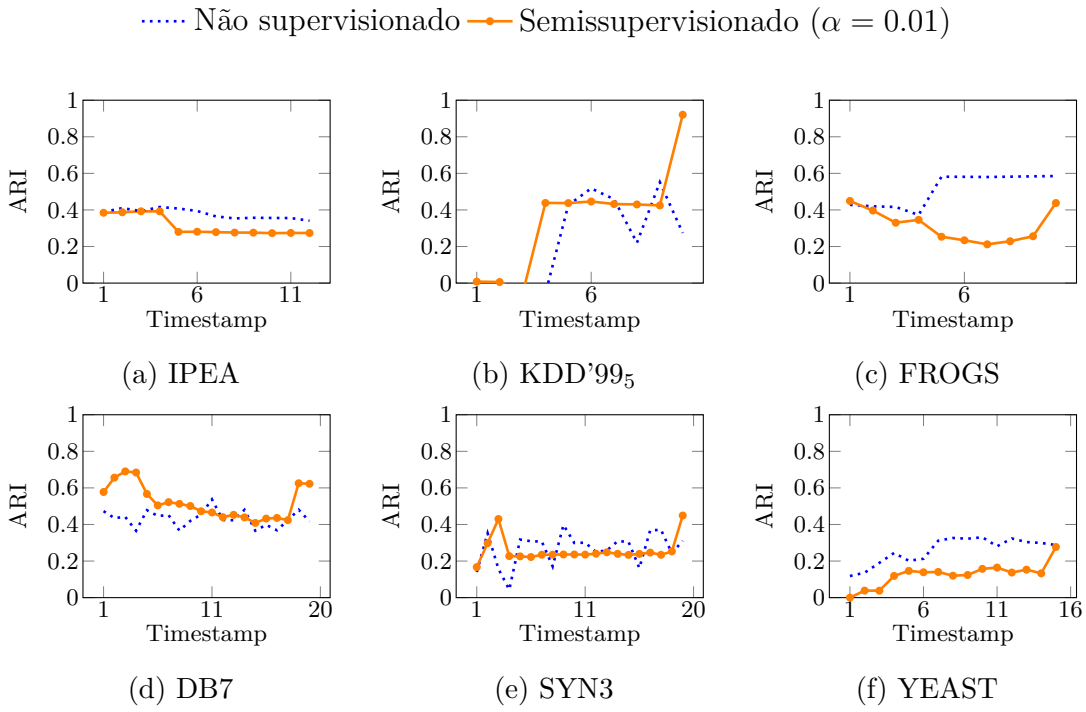


Figura 19 – Avaliação da eficácia: restrições ML e CL e especialização de grupos.

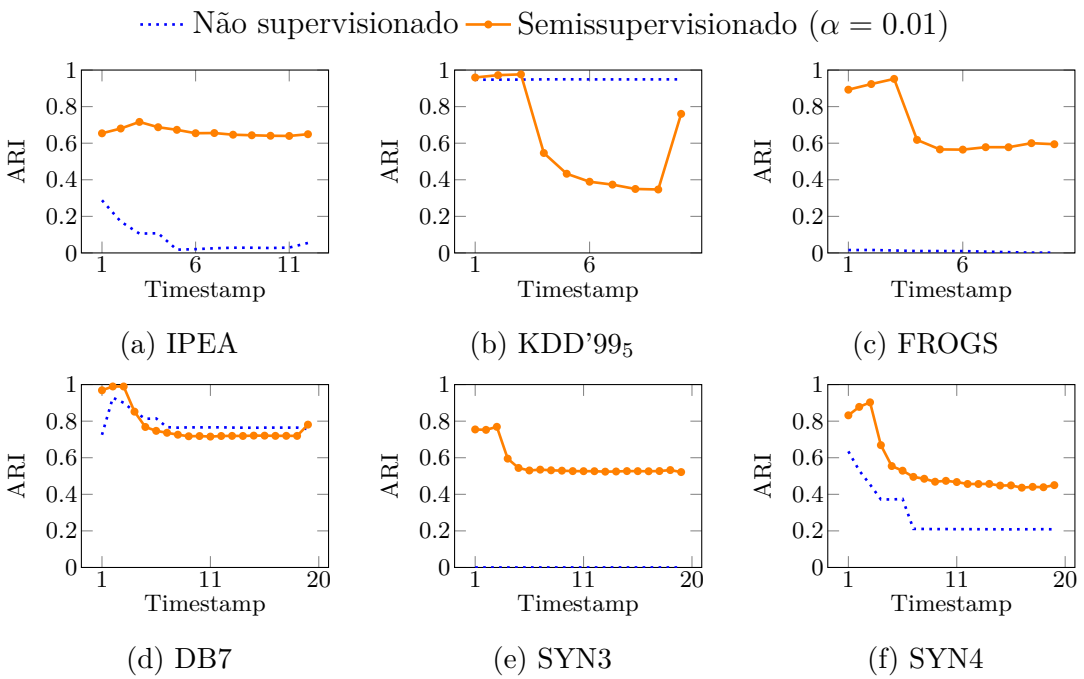


Figura 20 – Avaliação da eficácia: semissupervisão em rótulos e generalização dos grupos.

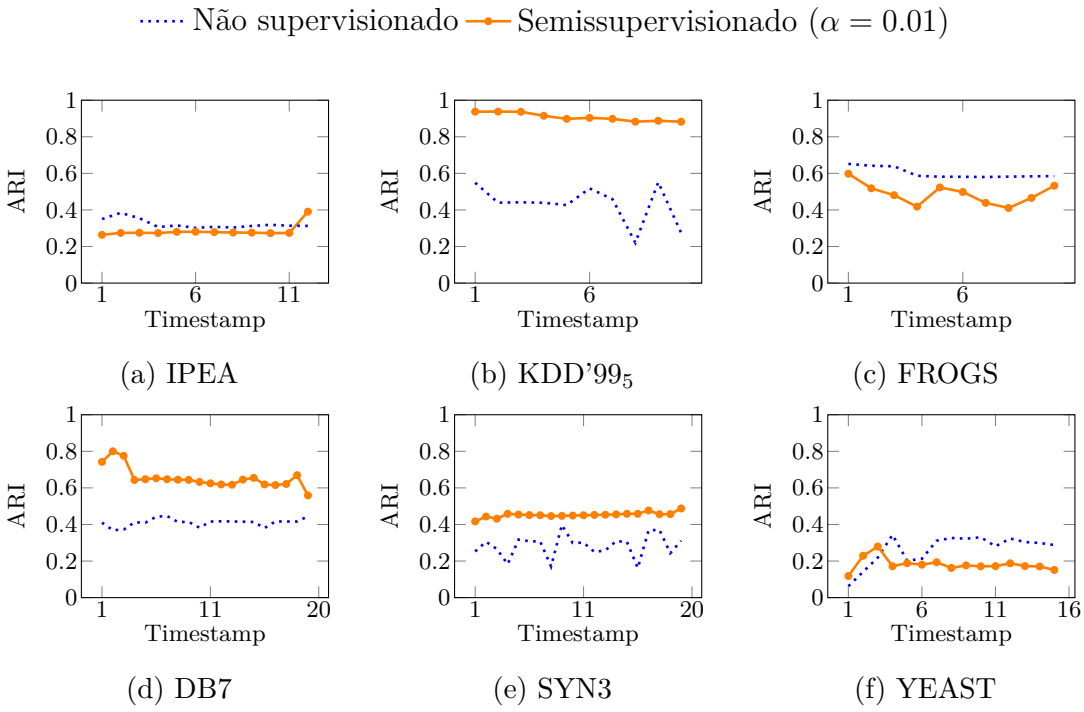


Figura 21 – Avaliação da eficácia: restrições ML e CL e generalização dos grupos..

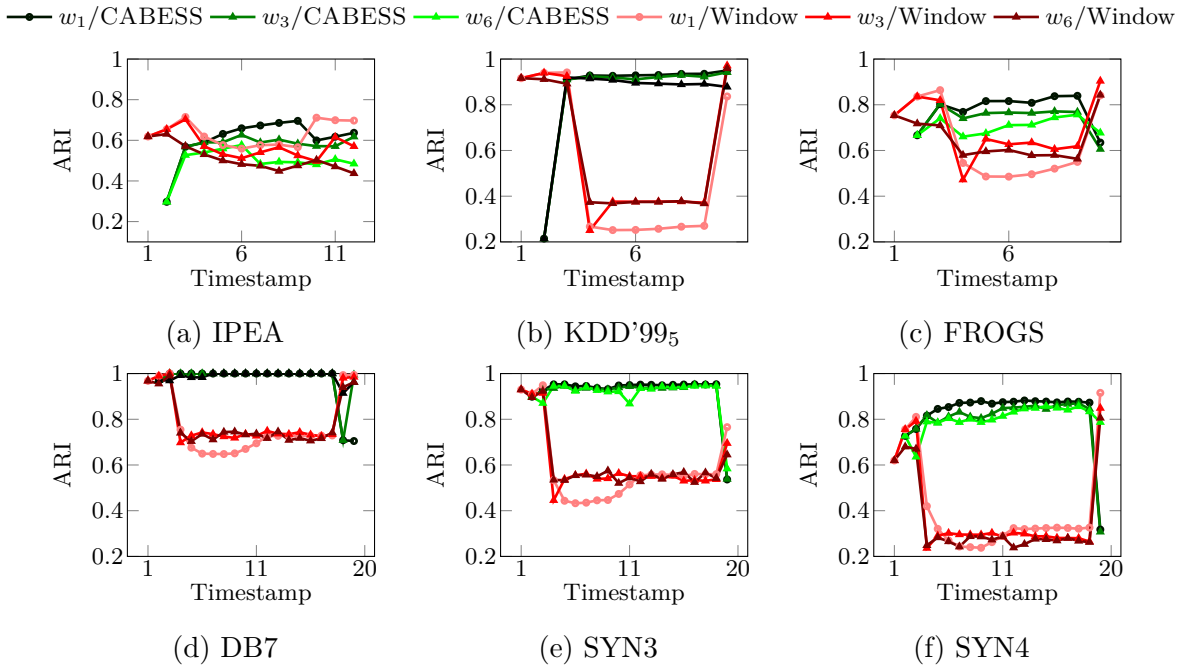
5.2.2 Avaliação da eficácia usando janelas de semissupervisão de tamanhos distintos

Na literatura sobre mineração em fluxo contínuo de dados há uma suposição comum que janelas pequenas mitigam o tempo de adaptação do processo de agrupamento e de aprendizagem [Gama 2010]. A fim de investigar esse cenário, neste conjunto de experimentos foram adotados três diferentes tamanhos de janelas de semissupervisão: $w_6 < w_3 < w_1$. A ideia é avaliar o impacto no tamanho da janela de semissupervisão sobre a eficácia das instanciações do framework, Pointwise CABESS e Pairwise CABESS, e do *baseline* semissupervisionado baseado em janela. Assim, pode-se responder a pergunta ***Q2*****:** *quanto a variação do tamanho da janela de semissupervisão afeta a eficácia da detecção semissupervisionada dos agrupamentos?* Além disso, variou-se a taxa de semissupervisão (α) em 5% e 10%, permitindo determinar o nível de semissupervisão adequado que propicie resultados satisfatórios. Considerando os resultados obtidos na Seção 5.2.1, os experimentos com a taxa de semissupervisão em 1% foram descartados neste conjunto. De maneira análoga ao conjunto de experimentos anterior, também há aqui uma subdivisão em: resultados especificados no índice ARI envolvendo técnicas de detecção baseadas em densidade (Figuras 22, 23, 26 e 27) e técnicas baseadas em particionamento (Figuras 24, 25, 28 e 29).

Analisando os resultados obtidos pode-se notar, especificamente nos experimentos envolvendo especialização dos agrupamentos (Figura 23), que a menor janela w_6 apresentou uma rápida adaptação após a transição de divisão. No entanto, nestes casos, a eficácia da detecção dos agrupamentos foi menos estável, ou seja, nota-se uma variação maior do índice ARI quando usa-se w_6 do que quando adota-se a janela w_1 (Figuras 23e, 23f, por exemplo). Isto ocorre porque algumas instâncias mudam de agrupamento em cada iteração. Inobstante, quando tem-se a janela grande, w_1, é possível observar uma adaptação lenta, porém com eficácia mais estável. O uso do detector de transições é um dos responsáveis por esse comportamento.

Além disso, a execução do Pointwise CABESS e Pairwise CABESS sobre uma janela de rótulos pequena apresentou valores de ARI piores em comparação com os resultados usando w_1, pois a adoção de janelas pequenas no CABESS mitiga o efeito benéfico de usar um detector de transições (Figuras 22(a,b), 24(a,c,e,f) e 25(a,c,e,f)).

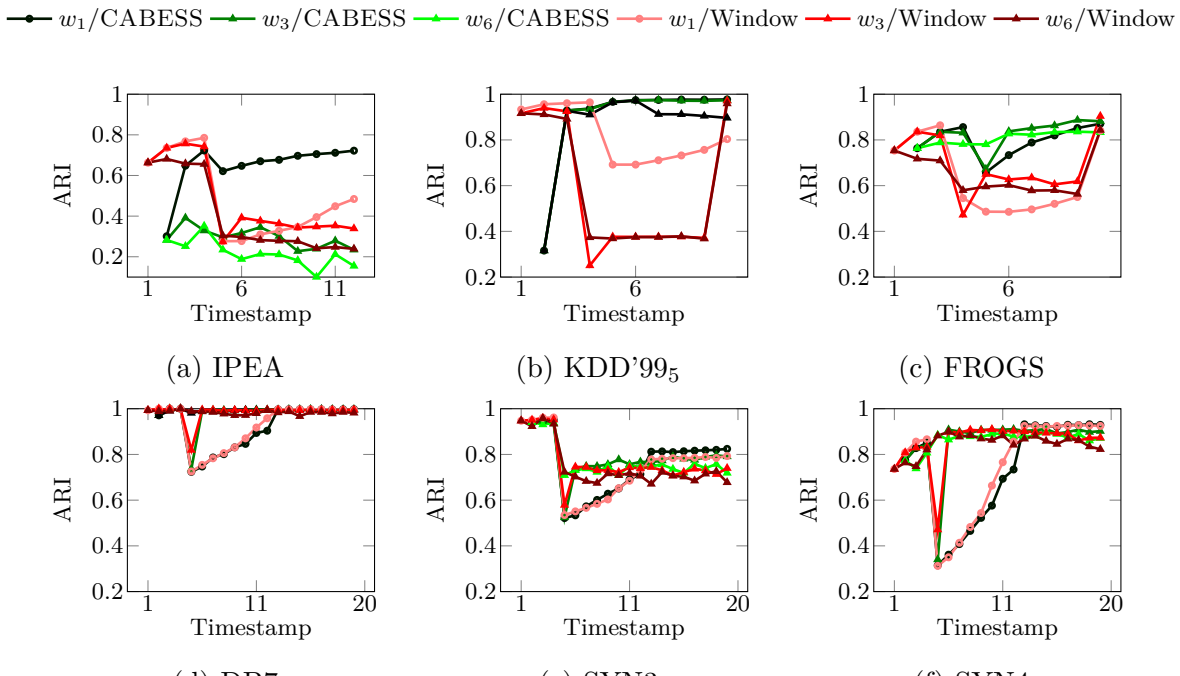
De maneira geral, é possível dizer que com uma taxa de semissupervisão de 10% e uma janela de semissupervisão de tamanho grande (w_1) obteve-se os melhores resultados para ambas as instanciações do framework CABESS. A curva referente a w_1 é, na maioria dos casos, superior as outras curvas, tanto as demais curvas das instanciações do CABESS quanto das abordagens baseadas em janela. Considerando somente as curvas referentes ao CABESS, é possível notar ainda que o uso da janela w_1 nas instanciações do CABESS é, nos piores casos, equivalente a curva da janela w_6 (Figuras 25b e 27d).



(a) IPEA (b) KDD'99$_5$ (c) FROGS

(d) DB7 (e) SYN3 (f) SYN4

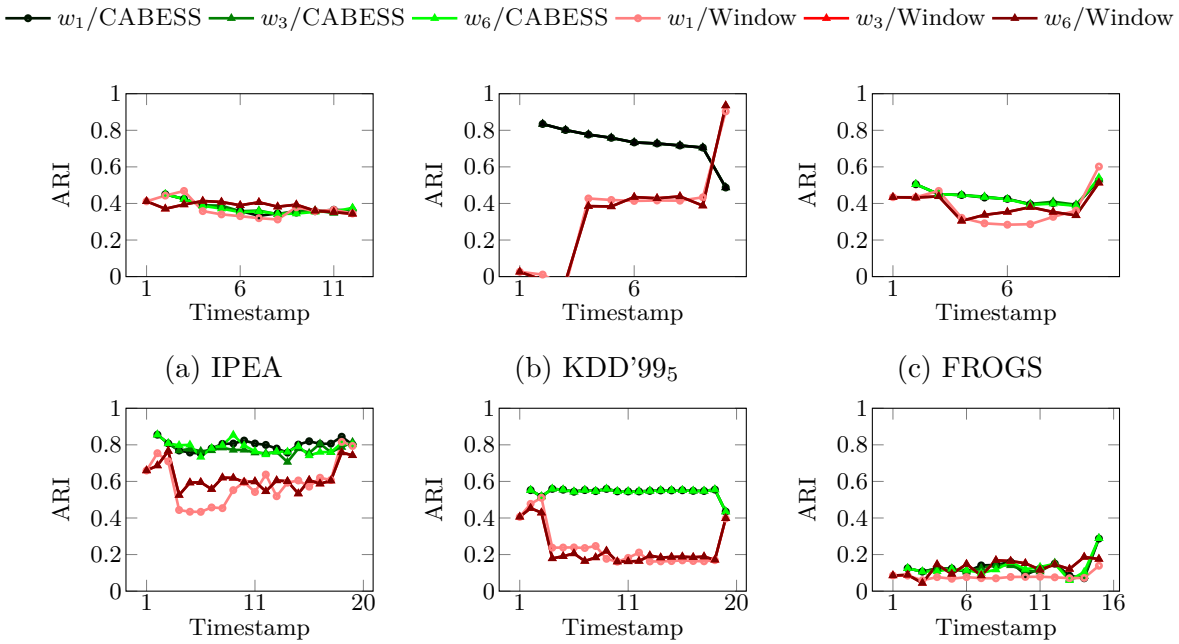
Figura 22 – Avaliação da eficácia do **Pointwise CABESS** e da abordagem baseada em janela considerando a **especialização dos grupos**, diferentes tamanhos de janela e usando $\alpha = 0.05$.



(a) IPEA (b) KDD'99$_5$ (c) FROGS

(d) DB7 (e) SYN3 (f) SYN4

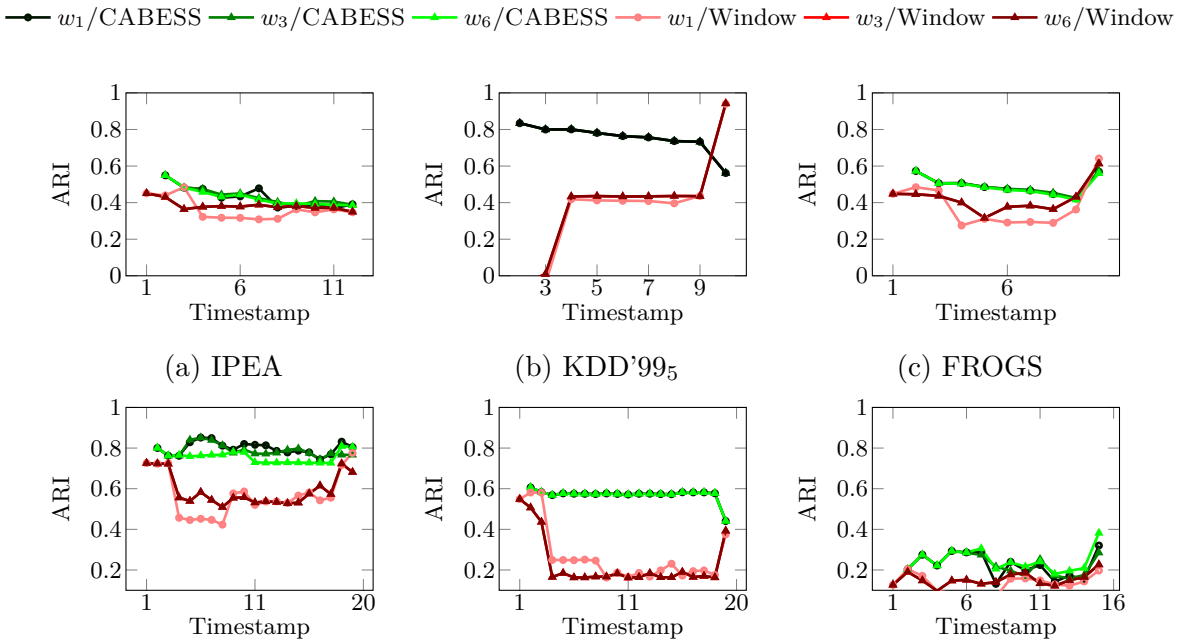
Figura 23 – Avaliação da eficácia do **Pointwise CABESS** e da abordagem baseada em janela considerando a **especialização dos grupos**, diferentes tamanhos de janela e usando $\alpha = 0.1$.



(a) IPEA (b) KDD'99$_5$ (c) FROGS

(d) DB7 (e) SYN3 (f) YEAST

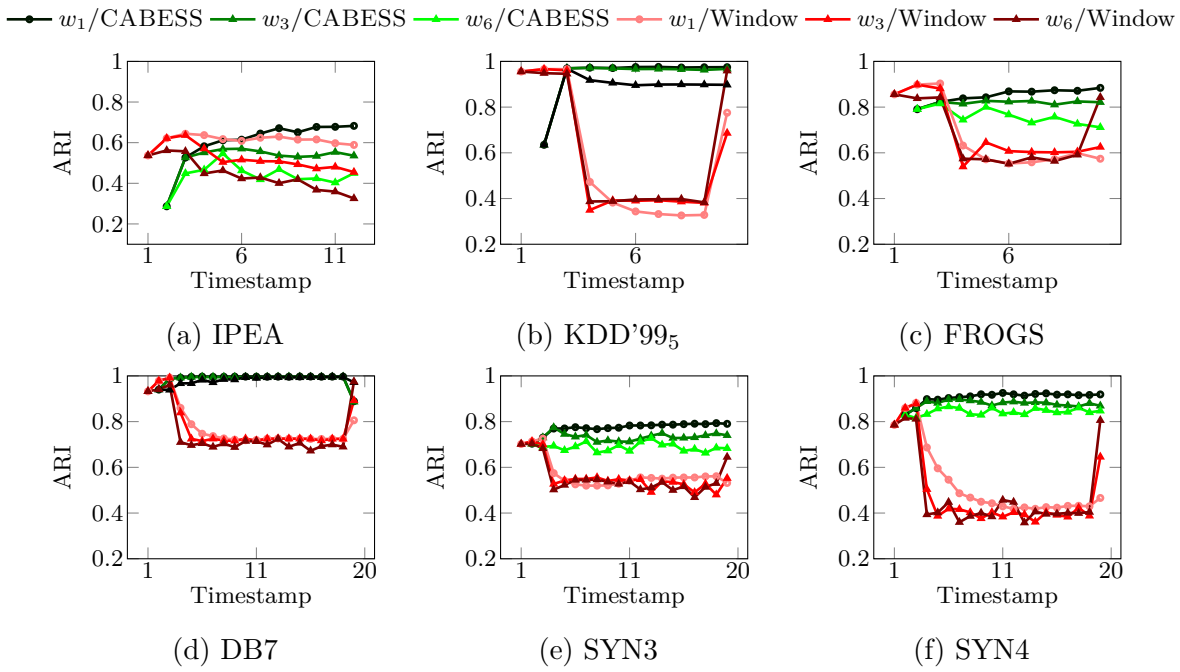
Figura 24 – Avaliação da eficácia do **Pairwise CABESS** e da abordagem baseada em janela considerando a **especialização dos grupos**, diferentes tamanhos de janela e usando $\alpha = 0.05$.



(a) IPEA (b) KDD'99$_5$ (c) FROGS

(d) DB7 (e) SYN3 (f) YEAST

Figura 25 – Avaliação da eficácia do **Pairwise CABESS** e da abordagem baseada em janela considerando a **especialização dos grupos**, diferentes tamanhos de janela e usando $\alpha = 0.1$.



(a) IPEA (b) KDD'99$_5$ (c) FROGS

(d) DB7 (e) SYN3 (f) SYN4

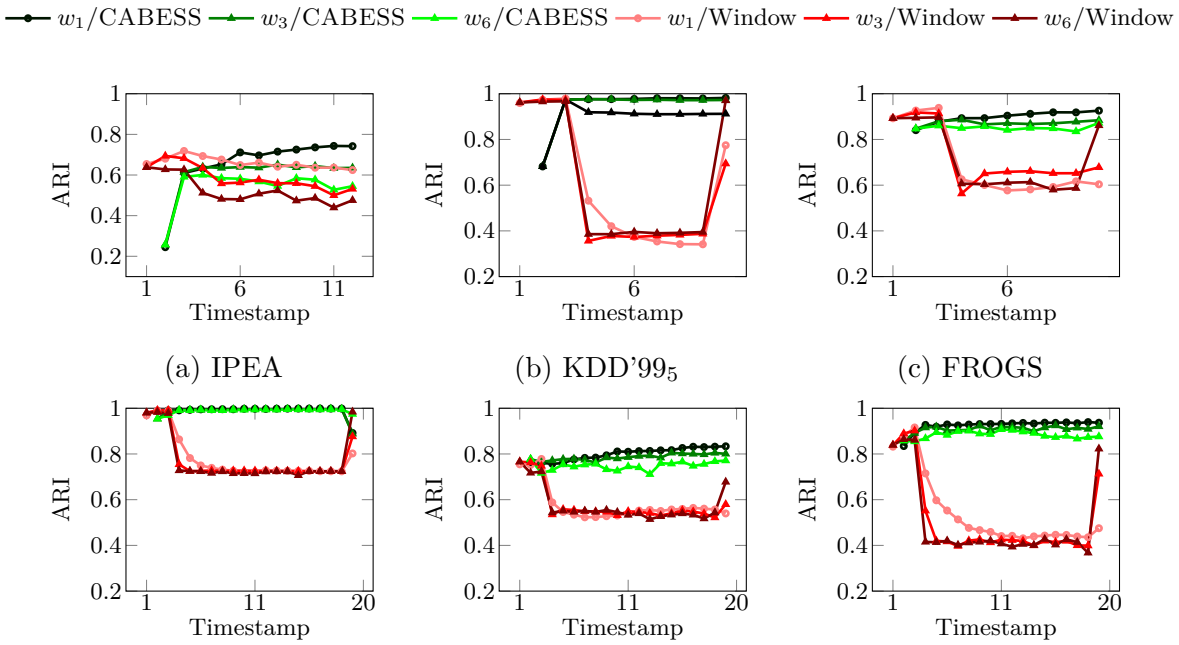
Figura 26 – Avaliação da eficácia do **Pointwise CABESS** e da abordagem baseada em janela considerando a **generalização dos grupos**, diferentes tamanhos de janela e usando $\alpha = 0.05$.

5.2.3 Impacto do detector de transições na eficácia do framework

Uma das principais razões para a relevância do framework CABESS é o uso de um detector de transições externas para auxiliar no gerenciamento da evolução da semissupervisão. Como o conjunto de experimentos anterior foi executado com foco no tamanho da janela e na taxa de semissupervisão, os resultados apresentados nesta subseção foram obtidos por meio de um conjunto de experimentos focados na variação dos valores dos parâmetros do detector de transições, usado nas instanciações do CABESS. Por isso, neste conjunto de experimentos, fixou-se a taxa de semissupervisão em 10% e adotou-se a janela de semissupervisão grande w_6. Assim, busca-se responder a seguinte pergunta de pesquisa ***Q3***: *qual é o impacto do uso do detector de transições na eficácia do framework* CABESS*?*

Neste conjunto de experimentos os valores para os parâmetros τ e τ_{split} foram variados com o objetivo de avaliar se o detector de transições externas seria sensível o suficiente para impactar a eficácia do processo de agrupamento. De acordo com o artigo original que propõe o MONIC [Spiliopoulou et al. 2006] definiu-se os seguintes valores para o parâmetro τ: $0.5, 0.6$ e 0.7, e os seguintes valores para o parâmetro τ_{split} em $0.1, 0.2$ e 0.3.

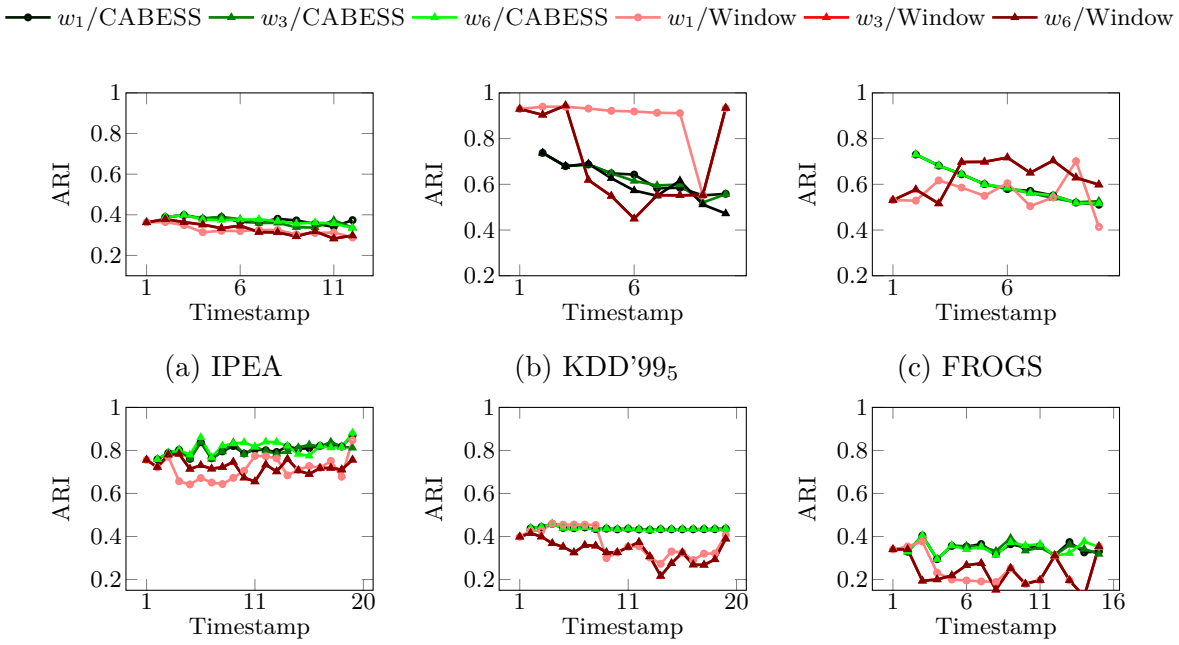
Para os conjuntos de dados considerados, não é possível notar grandes diferenças no índice ARI ao longo dos experimentos variando os parâmetros τ e τ_{split}. Na Figura 30 são apresentados os resultados obtidos por meio da variação dos valores dos parâmetros usando o Pairwise CABESS no cenário onde os agrupamentos são especializados. Neste cenário, em alguns conjuntos, uma pequena variação é notada, por exemplo, nos conjuntos



(a) IPEA (b) KDD'99$_5$ (c) FROGS

(d) DB7 (e) SYN3 (f) SYN4

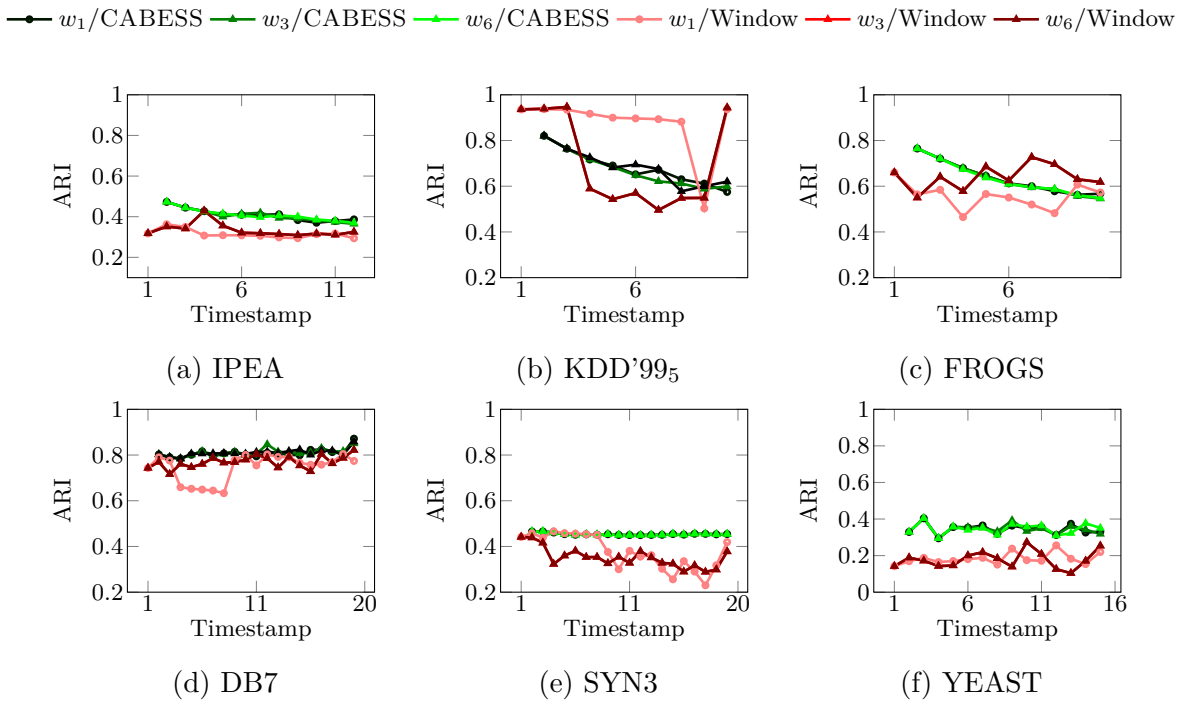
Figura 27 – Avaliação da eficácia do **Pointwise CABESS** e da abordagem baseada em janela considerando a **generalização dos grupos**, diferentes tamanhos de janela e usando $\alpha = 0.1$.



(a) IPEA (b) KDD'99$_5$ (c) FROGS

(d) DB7 (e) SYN3 (f) YEAST

Figura 28 – Avaliação da eficácia do **Pairwise CABESS** e da abordagem baseada em janela considerando a **generalização dos grupos**, diferentes tamanhos de janela e usando $\alpha = 0.05$.



(a) IPEA (b) KDD'99$_5$ (c) FROGS

(d) DB7 (e) SYN3 (f) YEAST

Figura 29 – Avaliação da eficácia do **Pairwise CABESS** e da abordagem baseada em janela considerando a **generalização dos grupos**, diferentes tamanhos de janela e usando $\alpha = 0.1$.

DB7 e YEAST (Figuras 30d e 30f). No entanto, na maioria dos cenários e na maioria dos conjuntos de dados as curvas de ARI se sobrepõem. Esse comportamento ocorre porque o detector não foi capaz de perceber nenhuma transição diferente, inclusive transições que não aconteceram de fato, quando variou-se os parâmetros nos valores anteriormente mencionados. Além disso, cada experimento simula uma única transição ao longo do tempo. Assim, a evolução foi controlada e os valores usados nos limiares não deixaram o detector sensível o suficiente para identificar transições precocemente ou tardiamente que pudessem afetar o conjunto de semissupervisão e posteriormente o índice ARI.

Portanto, como o comportamento é semelhante, suprimiu-se os valores obtidos nos demais experimentos envolvendo o Pointwise CABESS e os resultados do Pairwise CABESS deste capítulo. O leitor consegue checar os valores de ARI dos demais experimentos olhando o Apêndice A.

5.2.4 Comparação entre o framework e as demais abordagens semissupervisionadas

Uma das principais contribuições desta dissertação é a habilidade do framework proposto lidar com semissupervisão especificada em feedbacks e extrair rótulos e restrições a partir dos feedbacks. Os conjuntos de experimentos anteriores foram executados com foco em entender se a detecção online de agrupamentos poderia se beneficiar da semissupervisão e do detector de transições. Neste conjunto de experimentos, avaliou-se a performance

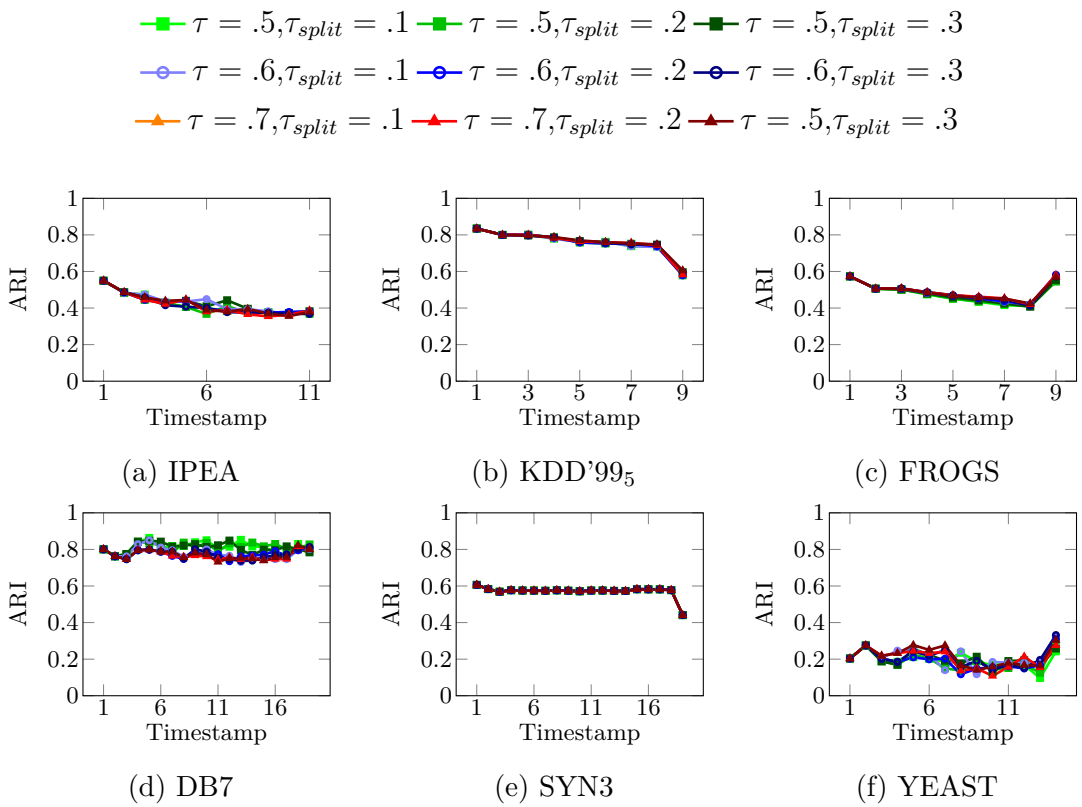


Figura 30 – Efeito da variação dos parâmetros τ e τ_{split} na eficácia da detecção de agrupamentos do **Pairwise CABESS** no cenário de **especialização dos grupos**.

das abordagens semissupervisionadas tradicionais contra o framework proposto. Assim, pode-se responder a pergunta ***Q4***: *quão eficaz é o framework* CABESS *comparado com as abordagens existentes de detecção de agrupamentos?*

Nas Figuras 31 e 32 são apresentados os valores de ARI para todas as abordagens *baselines* consideradas e o Pointwise CABESS. Da mesma forma, nas Figuras 33 e 34, são apresentados os resultados experimentais comparando as demais abordagens e o Pairwise CABESS. Nestes experimentos, adotou-se $\alpha = 0.1$ (10% de taxa de semissupervisão), a maior janela de semissupervisão (w_1) e a seguinte configuração dos parâmetros do detector de transições $\tau = 0.7$ e $\tau_{split} = 0.1$, baseando-se nas análises obtidas anteriormente.

Analisando os resultados obtidos, no cenário em que a semissupervisão é especificada em rótulos (Figuras 31 e 32), é possível notar que o Pointwise CABESS obteve melhores resultados que as outras abordagens, sobretudo após as transições externas de divisão nos conjuntos KDD'99$_5$ e FROGS (Figura 31(b,c)). Para os outros conjuntos de dados a eficácia obtida para as abordagens *baseline* foi equivalente. É importante enfatizar que a informação de semissupervisão utilizada pelas instanciações do framework CABESS é pobre quando comparada com a semissupervisão usada pelas demais abordagens. Note que o framework proposto recebe feeedbacks ao invés de rótulos. Assim, é necessário inferir e manter os rótulos corretamente ao longo do tempo. Os resultados experimentais mostram que não há perda de eficácia na detecção dos agrupamentos quando é preciso inferir os

rótulos a partir de feedbacks, como é feito pelo framework CABESS, em comparação com o uso direto dos rótulos pelas demais abordagens semissupervisionadas.

Observando os resultados referentes ao cenário com restrições ML e CL (Figuras 33 e 34), percebe-se que o framework CABESS, instanciado como Pairwise CABESS, apresenta os melhores resultados nos conjuntos IPEA, DB7 e SYN3 independentemente do tipo de transição (Figuras 33(a,d,e) e 34(a,d,e)). No cenário em que os grupos são especializados, o CABESS apresenta bom resultado no conjunto KDD'99_5. No entanto, nos demais conjuntos, o Pairwise CABESS perde em eficácia para outras abordagens. Nos conjuntos de dados FROGS e YEAST, por exemplo, os resultados foram inclusive piores que a estratégia não supervisionada: a partir de um determinado instante do tempo a curva do método não supervisionado fica acima da curva do Pairwise CABESS (Figuras 33(c,f) e 33(c)). Esse comportamento pode ser um indicativo de que a semissupervisão, ou a taxa de semissupervisão adotada[6], não consegue auxiliar o algoritmo a ajustar a estrutura dos agrupamentos de maneira a satisfazer o desejo do usuário.

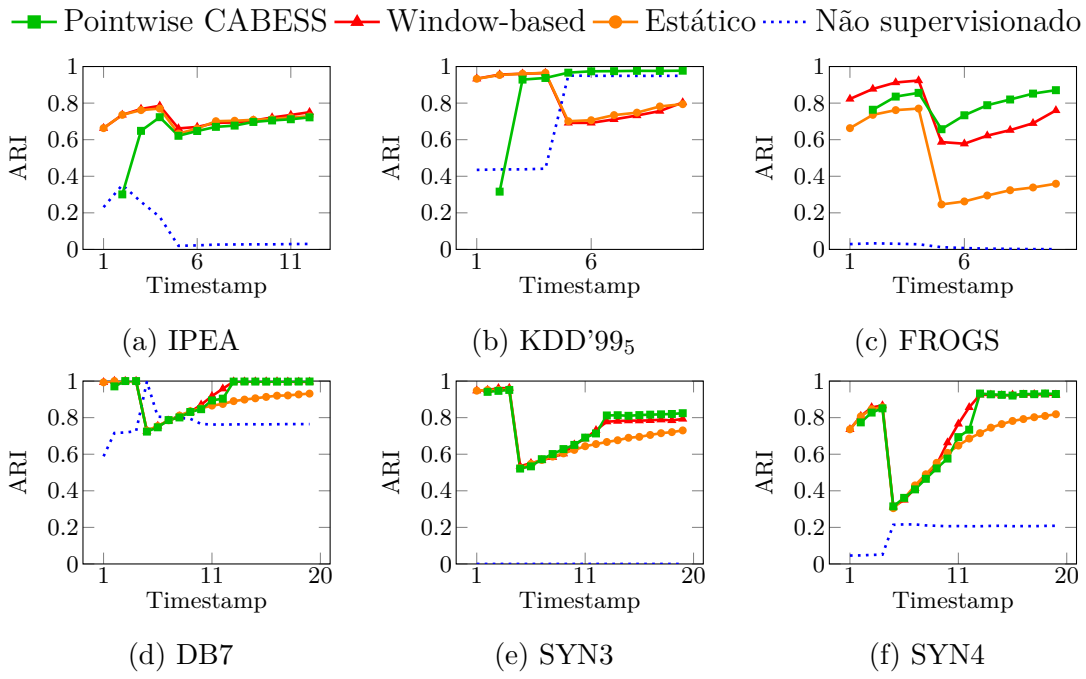
Finalmente, é importante destacar que, de maneira geral, as curvas obtidas com o Pointwise CABESS (Figuras 31 e 32) foram melhores que as curvas referentes ao Pairwise CABESS (Figuras 33 e 34), independentemente do tipo de transição. No entanto, é preciso levar em consideração que os conjuntos de dados sintéticos foram projetados de forma a permitir um melhor agrupamento por abordagens baseadas em densidade, que é o caso do Pointwise CABESS e das demais abordagens avaliadas usando semissupervisão em rótulos.

5.2.5 Avaliação do impacto do tipo de semissupervisão na eficácia do agrupamento

Esta subseção é destinada a responder a pergunta ***Q5*****:** *há grandes diferenças na eficácia da detecção de agrupamentos quando usa-se detecção semissupervisionada de agrupamentos baseada em feedbacks, rótulos e restrições ML e CL?*

Nos experimentos apresentados neste capítulo percebe-se que o tipo de especificação de semissupervisão impacta na eficácia do processo de detecção de agrupamentos independentemente do tipo de transição considerada. É notável que quando a semissupervisão é fornecida em forma de rótulos ou ela é internamente deduzida pelo processo de detecção (caso do Pointwise CABESS), a qualidade da partição obtida pela técnica é sensível às transições dos agrupamentos. Por outro lado, quando considera-se a semissupervisão especificada em restrições ML e CL é notável a percepção de que o processo de detecção de agrupamentos torna-se resiliente ao longo do tempo, principalmente quando considera-se

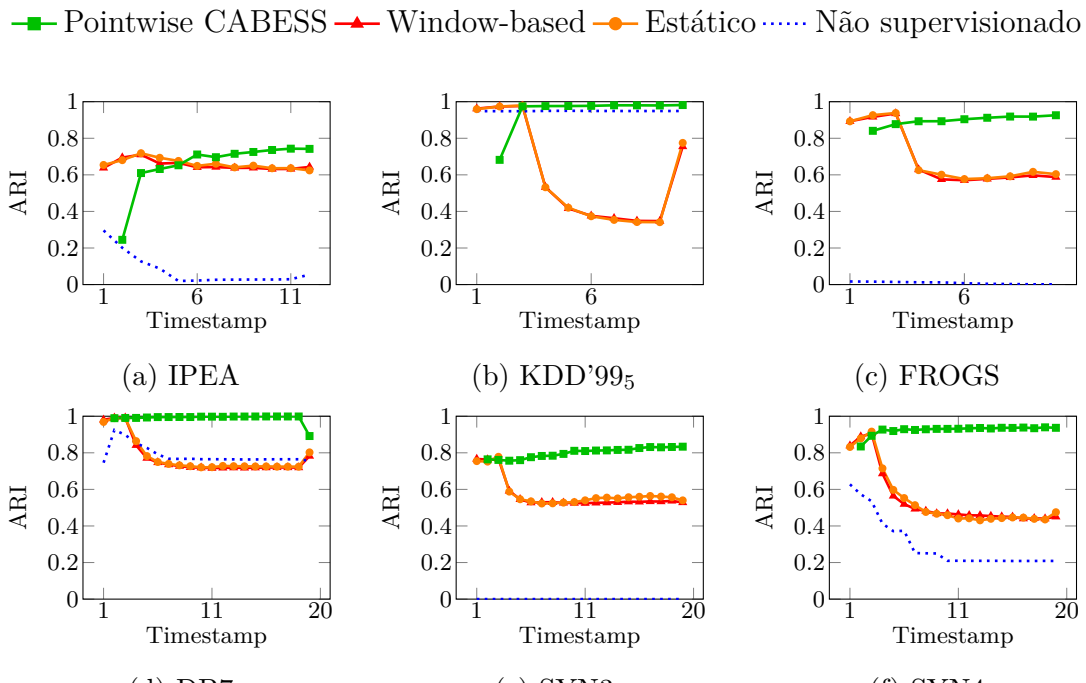
[6] No cenário online, taxas muitas altas de semissupervisão não são adequadas, pois exigem muito esforço do usuário para cumpri-las.



(a) IPEA (b) KDD'99$_5$ (c) FROGS

(d) DB7 (e) SYN3 (f) SYN4

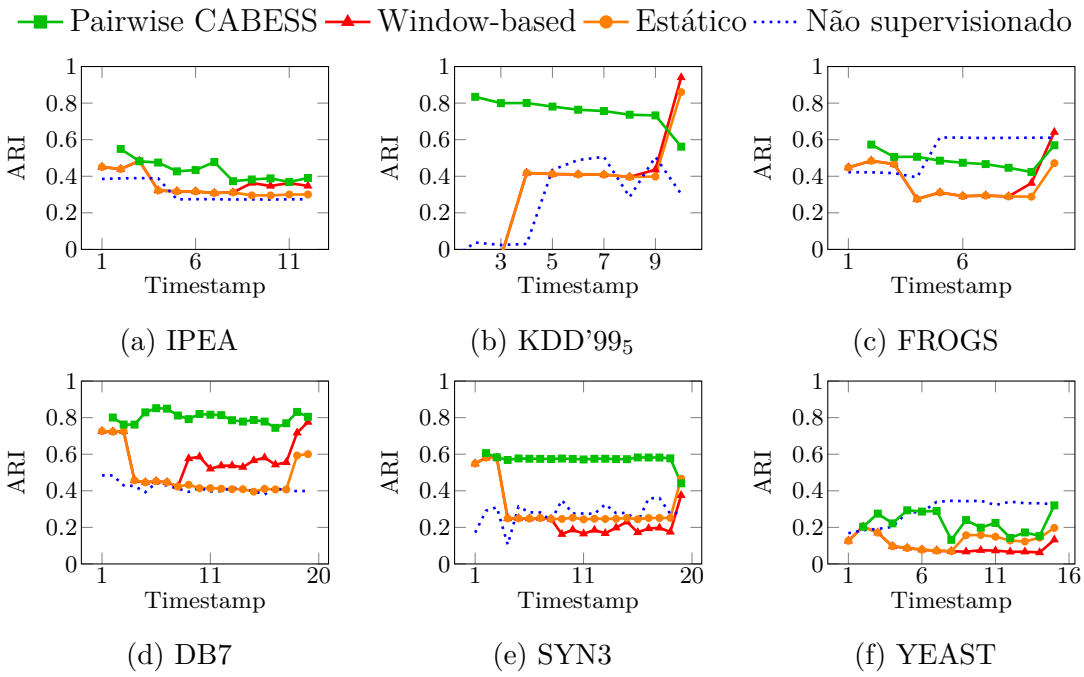
Figura 31 – Comparação de eficácia entre o **Pointwise CABESS** e as abordagens baseadas em rótulos no cenário com a **especialização dos grupos** ($\alpha = 0.1$ e w_1).



(a) IPEA (b) KDD'99$_5$ (c) FROGS

(d) DB7 (e) SYN3 (f) SYN4

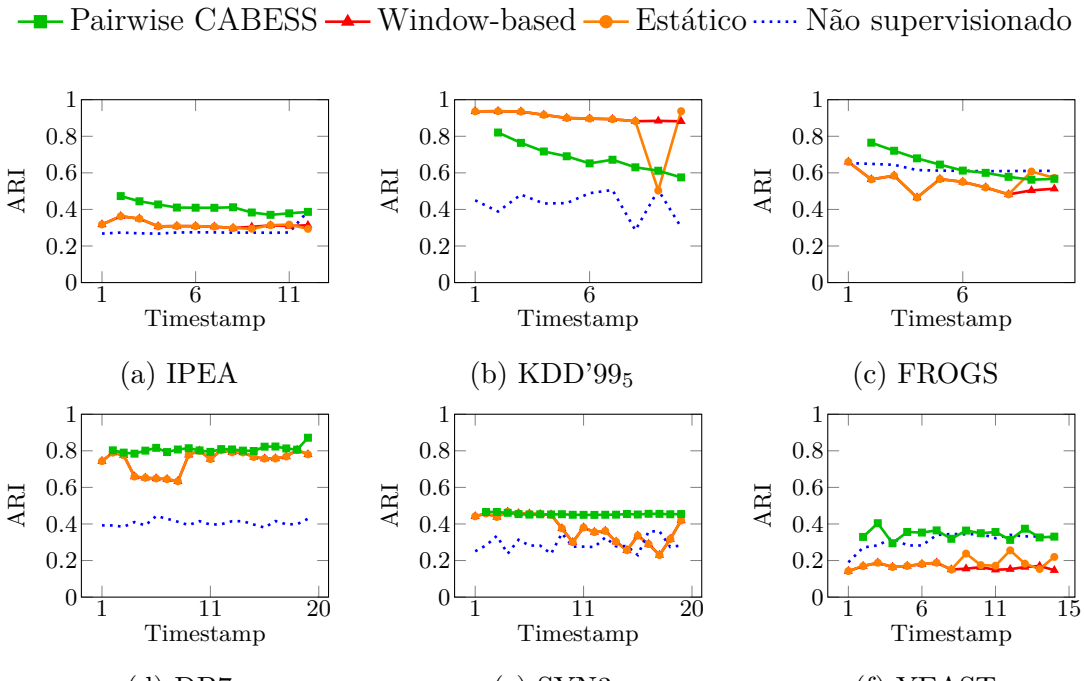
Figura 32 – Comparação de eficácia entre o **Pointwise CABESS** e as abordagens baseadas em rótulos no cenário com a **generalização dos grupos** ($\alpha = 0.1$ e w_1).



(a) IPEA (b) KDD'99$_5$ (c) FROGS

(d) DB7 (e) SYN3 (f) YEAST

Figura 33 – Comparação de eficácia entre o **Pairwise CABESS** e as abordagens baseadas em rótulos no cenário com a **especialização dos grupos** ($\alpha = 0.1$ e w_1).



(a) IPEA (b) KDD'99$_5$ (c) FROGS

(d) DB7 (e) SYN3 (f) YEAST

Figura 34 – Comparação de eficácia entre o **Pairwise CABESS** e as abordagens baseadas em rótulos no cenário com a **generalização dos grupos** ($\alpha = 0.1$ e w_1).

as curvas do Pairwise CABESS após a ocorrência de mudanças na estrutura dos agrupamentos (Figuras 33 e 34). Esse comportamento corrobora o fato, discutido na Seção 4.4.2, de que após uma transição nem todo o conjunto de restrições ML e CL torna-se obsoleto. Assim, a inutilização de parte do conjunto de restrições ML e CL minimiza o impacto de uma transição na eficácia do processo de agrupamento, principalmente no Pairwise CABESS (Figuras 33 e 34).

Portanto, pode-se afirmar que há diferenças na eficácia da detecção de agrupamentos quando utiliza-se diferentes tipos de especificação de semissupervisão. A diferença também é notada na eficácia do CABESS, pois mesmo recebendo feedbacks como *input*, os algoritmos usados nas instanciações do framework não lidam com feedbacks e sim com restrições em rótulos ou em pares ML e CL. Assim, pode-se notar a diferença de resiliência do processo de agrupamento entre as duas instanciações do framework, Pointwise CABESS e Pairwise CABESS (Figuras 31, 32, 33 e 34).

5.2.6 Avaliação do impacto do tipo de transição na eficácia do agrupamento

Nesta subseção confronta-se os resultados experimentais com a pergunta de pesquisa ***Q6*****:** *há grandes diferenças na eficácia da detecção de agrupamentos quando ocorrem transições de divisão e quando ocorrem união dos agrupamentos?*

Analisando novamente os resultados experimentais obtidos por meio da variação do tamanho das janelas e da taxa de semissupervisão, pode-se perceber que, no cenário onde a semissupervisão é especificada em rótulos e com $\alpha = 0.1$, o processo de detecção de agrupamentos é mais sensível quando a transição é do tipo divisão do que quando é do tipo união (Figuras 31 e 32). Por outro lado, quando a semissupervisão é especificada em restrições ML e CL, não é possível notar comportamentos diferentes nos valores de ARI obtidos entre os dois tipos de transição (Figuras 33 e 34). Esse comportamento pode ter relação com o tipo de semissupervisão e não o tipo de transição. Assim, com os conjuntos de dados e os parâmetros usados nos experimentos descritos nesta dissertação não é possível afirmar que há diferenças na eficácia do processo de agrupamento entre os tipos de transição analisados.

5.2.7 Avaliação do tempo de execução

Outra contribuição do framework CABESS é sua habilidade de sumarizar as instâncias de dados e a informação de semissupervisão. Nesta seção quantificou-se a eficiência do framework proposto e das demais abordagens executadas a fim de responder a pergunta ***Q7*****:** *quão eficiente é o framework* CABESS *comparado com as abordagens existentes de detecção de agrupamentos?*. Nas Figuras 35 e 36 são apresentados o tempo médio de execução para cada uma das instanciações do framework CABESS e das demais aborda-

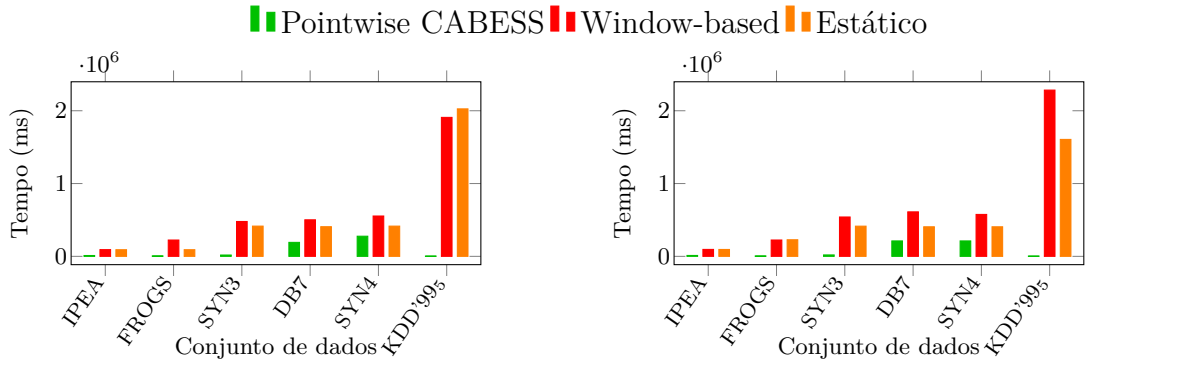
gens. Observando os gráficos pode-se notar que, de maneira geral, o CABESS apresentou um tempo de processamento inferior em relação as demais abordagens. Somente no conjunto DB7 e executando o Pairwise CABESS, o framework precisou de mais tempo para completar o processo de detecção de agrupamentos do que as outras abordagens. É importante ressaltar que o conjunto DB7 foi projetado para ser facilmente agrupado por algoritmos baseados em densidade e não por algoritmos baseados em particionamento, usados no Pairwise CABESS. A principal razão para o bom desempenho do framework é o módulo de sumarização do CABESS. Consequentemente, o algoritmo de detecção de agrupamentos lida com um pequeno número de instâncias sumarizadas comparado com o número real de instâncias, com o qual as demais abordagens precisam processar.

Os gráficos das Figuras 35 e 36 apresentam o tempo médio de execução considerando dois cenários: (I) com alta taxa de semissupervisão, $\alpha = 0.1$, e janela grande, $w1$, e (II) com baixa taxa de semissupervisão, $\alpha = 0.01$, e janela pequena, w_6. Analisando o cenário com restrição ML e CL e comparando os gráficos da Figura 36(a,c) com os da Figura 36(b,d), observa-se uma notável diferença no tempo de execução das abordagens entre (I) e (II). Nos experimentos com um volume maior de semissupervisão e onde essa informação permanece em memória principal por um período amplo, as abordagens gastaram mais tempo para detectar os agrupamentos. Esse comportamento reflete o custo computacional causado pelo aumento da informação de semissupervisão com que as abordagens e o CABESS precisam lidar ao longo do tempo. No entanto, este mesmo comportamento não é observável no cenário com rótulos (Figura 35).

Outra importante observação com relação a eficiência é a comparação entre os tempos de execução dos experimentos envolvendo o cenário com rótulos e o cenário com restrições ML e CL. Observando os gráficos das Figuras 35 e 36, percebe-se que no cenário com rótulos as estratégias gastaram mais tempo do que no cenário com restrições. Este comportamento pode não estar relacionado com o tipo de semissupervisão e com a abordagem usada para detectar os agrupamentos (particionamento ou densidade), mas trata-se de uma diferença causada pela implementação dos algoritmos SSDBScan e MPCK-Means. Inobstante, o framework CABESS foi capaz de reduzir o tempo médio de execução nos dois casos, evidenciando a capacidade dele de ser eficiente em cenários distintos.

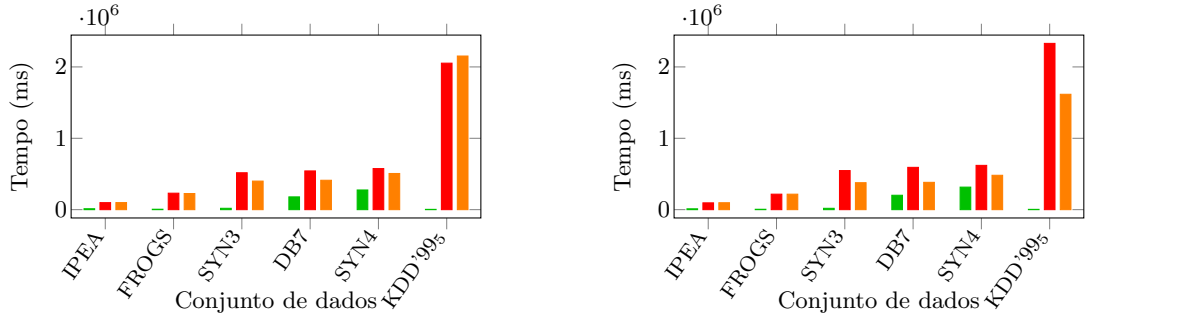
5.3 Considerações Finais

Neste capítulo foram descritos os conjuntos de experimentos e apresentados os resultados obtidos que guiaram o processo de análise e resposta das questões de pesquisa colocadas no início desta dissertação. Analisando os resultados experimentais, percebeu-se o potencial do framework proposto nesta dissertação para realizar a detecção online e



(a) Especialização dos grupos, $\alpha = 0.1$ e w_1

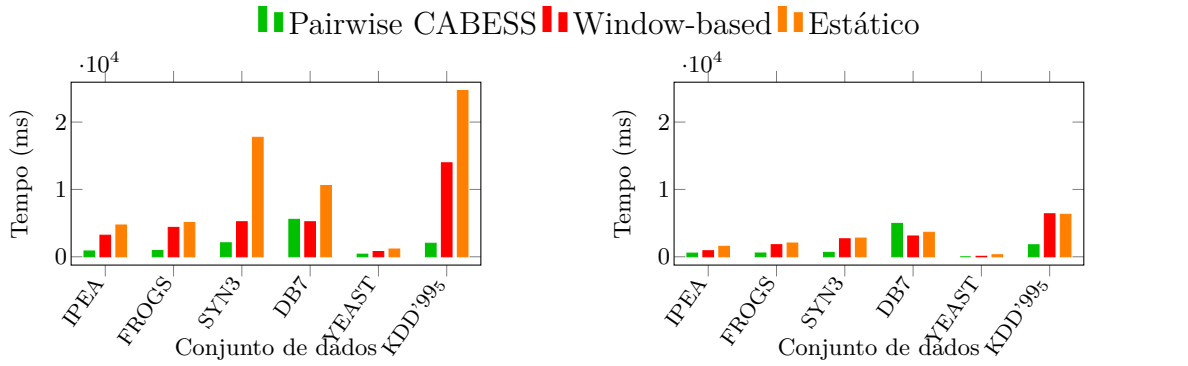
(b) Especialização dos grupos, $\alpha = 0.01$ e w_6



(c) Generalização dos grupos, $\alpha = 0.1$ e w_1

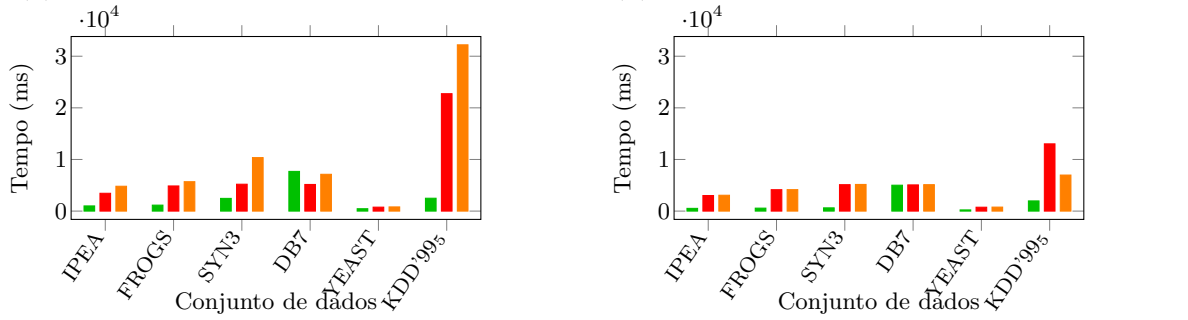
(d) Generalização dos grupos, $\alpha = 0.01$ e w_6

Figura 35 – Tempo médio de execução para agrupar diferentes conjuntos de dados usando taxas de semissupervisão alta e baixa no cenário com rótulos como semissupervisão.



(a) Especialização dos grupos, $\alpha = 0.1$ e w_1

(b) Especialização dos grupos, $\alpha = 0.01$ e w_6



(c) Generalização dos grupos, $\alpha = 0.1$ e w_1

(d) Generalização dos grupos, $\alpha = 0.01$ e w_6.

Figura 36 – Tempo médio de execução para agrupar diferentes conjuntos de dados usando taxas de semissupervisão alta e baixa no cenário com restrições ML e CL.

semissupervisionada de agrupamentos. As instanciações do framework CABESS obtiveram uma eficiência superior em comparação com as abordagens tradicionais e, de maneira geral, apresentaram boa eficácia com uma taxa de 10% de semissupervisão juntamente com o uso de uma janela de semissupervisão de tamanho grande. Assim, pode-se afirmar que ele é um bom *threshold* entre eficácia e eficiência na maioria dos cenários e ainda tem grande vantagem em cenários específicos: em conjuntos de dados grandes e em casos com taxas de semissupervisão a partir de 10%. Além disso, notou-se que as restrições ML e CL conferem resiliência ao processo online de detecção de agrupamentos independentemente do tipo de transição.

O capítulo seguinte conclui esta dissertação por meio da revisão dos objetivos apresentados no Capítulo 1 e também faz propostas de trabalhos futuros.

Capítulo 6

Conclusão

O principal objetivo do framework proposto desta dissertação, CABESS, foi auxiliar a detecção online de agrupamentos semissupervisionada a lidar com transições externas. Os resultados experimentais e as análises foram realizadas seguindo sete perguntas de pesquisas sobre quatro conjuntos de dados reais e três conjuntos de dados sintéticos. Os resultados obtidos mostraram que o framework CABESS apresenta melhor eficiência quando comparado com outras abordagens semissupervisionadas enquanto preserva uma boa eficácia em boa parte dos cenários e mostra resultados equivalentes nos demais cenários. Os experimentos corroboram a generalidade do framework, pois avaliou-se o framework em duas instanciações usando algoritmos de detecção de agrupamentos distintos que, inclusive, recebem diferentes tipos de semissupervisão. Além disso, notou-se que a semissupervisão na forma de restrições ML e CL proporciona eficácia mais estável ao longo do tempo em comparação com os experimentos usando semissupervisão especificada em rótulos.

6.1 Principais Contribuições

A partir dos resultados experimentais obtidos e das análises que corroboram o potencial do framework CABESS, são apresentados a seguir as principais contribuições alcançadas no desenvolvimento do trabalho descrito nesta dissertação:

1. O desenvolvimento de um método genérico que faz a detecção semissupervisionada de agrupamentos no contexto online, referido aqui como framework CABESS. O framework recebe como entrada, além das instâncias de dados, os feedbacks do usuário na forma positiva e deslocamento. Além disso, o CABESS permite ser instanciado com diferentes tipos de algoritmos de detecção de agrupamentos. No trabalho descrito nesta dissertação, instanciou-se duas versões do framework: uma com algoritmos de detecção de agrupamentos em densidade, Pointwise CABESS, e outra com algoritmos de agrupamentos baseados em particionamento, Pairwise CA-

BESS. Os resultados obtidos apresentaram, de maneira geral, eficácia e eficiência satisfatória em relação às demais abordagens utilizadas para comparação quando se tem uma taxa de semissupervisão de 10%.

2. Criação de um método que faz a extração de rótulos em nível instância a partir dos feedbacks do usuário e faz a dedução de rótulos em nível sumário a partir dos rótulos em nível instância. Os resultados obtidos mostraram que o método consegue fornecer satisfatoriamente os rótulos para o método de detecção semissupervisionada de agrupamentos de tal forma que a eficácia do método é equivalente ou superior

3. O uso de um detector de transições externas de agrupamentos para fazer o gerenciamento e manutenção da semissupervisão ao longo do tempo. Geralmente, na literatura científica, o detector de transições é usado para entender o comportamento global dos agrupamentos a medida que o tempo passa. No trabalho descrito aqui, as transições detectadas são usadas como indicadores para o processo de manutenção ou descarte da semissupervisão ao longo do tempo, juntamente com uma janela deslizante. A utilização do detector de transições é uma das razões da relevância do framework CABESS, pois é um fator que diferencia o conjunto de semissupervisão usado pelo framework ao longo do tempo em comparação com as demais abordagens.

6.2 Trabalhos Futuros

Com o framework CABESS, pode-se vislumbrar novas direções para estender as técnicas tradicionais de detecção de agrupamentos semissupervisionadas. O trabalho descrito nesta dissertação poderá ser estendido para explorar outros tipos de semissupervisão, por exemplo restrições em grupo ou *ranking*, e também empregar outros algoritmos de detecção de agrupamentos dentro do framework CABESS. Outras interessantes direções como trabalho futuro são listadas a seguir.

- ❏ Analisar a evolução da semissupervisão utilizando métricas como coerência e informatividade.

- ❏ Utilizar e avaliar outro tipo de técnica para sumarização dos dados no primeiro módulo do CABESS.

- ❏ Avaliar a eficácia e eficiência do framework CABESS em conjuntos de dados de outros domínios, por exemplo, conjuntos de documentos e conjuntos de imagens.

- ❏ Explorar outros tipos de transições de agrupamentos no gerenciamento da evolução da semissupervisão.

- Explorar e avaliar a eficácia do processo de detecção de agrupamentos do CABESS em cenários com mais de um tipo de transição ocorrendo em sequência e/ou paralelo.

6.3 Contribuição em Produção Bibliográfica

A pesquisa envolvida nesta dissertação gerou a seguinte publicação até o momento.

- Artigo intitulado "*A Framework for Online Clustering Based on Evolving Semi-Supervision*", submetido e aceito no Simpósio Brasileiro de Bancos de Dados (SBBD 2017). Nesse artigo foram apresentados os resultados referentes a instanciação do framework CABESS que lida com semissupervisão especificada em rótulos, Pointwise CABESS, considerando somente um tipo de transição externa dos agrupamentos.

Referências

AGGARWAL, C. C. et al. A framework for clustering evolving data streams. In: VLDB ENDOWMENT. **Proceedings of the 29th International Conference on Very Large Data Bases**. [S.l.], 2003. p. 81–92.

AGGARWAL, C. C.; REDDY, C. K. **Data clustering: algorithms and applications**. [S.l.]: CRC Press, 2013.

AGRAWAL, R. et al. Automatic subspace clustering of high dimensional data for data mining applications. **ACM SIGMOD Record**, ACM, v. 27, n. 2, p. 94–105, jun. 1998. ISSN 01635808. Disponível em: <http://dl.acm.org/citation.cfm?id=276305.276314>.

BARIONI, M. C. N. et al. Open issues for partitioning clustering methods: an overview. **Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery**, v. 4, n. 3, p. 161–177, maio 2014. ISSN 19424787. Disponível em: <http://doi.wiley.com/10.1002/widm.1127>.

BASU, S.; DAVIDSON, I.; WAGSTAFF, K. **Constrained Clustering: Advances in Algorithms, Theory, and Applications**. [S.l.]: Chapman and Hall/CRC, 2008. v. 20084530. 961–970 p. (Chapman & Hall/CRC Data Mining and Knowledge Discovery Series, v. 20084530). ISSN 10016538. ISBN 978-1-58488-996-0.

Ben Ahmed, E.; NABLI, A.; GARGOURI, F. A new semi-supervised hierarchical active clustering based on ranking constraints for analysts groupization. **Applied Intelligence**, v. 39, n. 2, p. 236–250, jan. 2013. ISSN 0924-669X. Disponível em: <http://link.springer.com/10.1007/s10489-012-0407-3>.

BILENKO, M.; BASU, S.; MOONEY, R. J. Integrating constraints and metric learning in semi-supervised clustering. In: **21th International Conference on Machine learning - ICML '04**. New York, New York, USA: ACM Press, 2004. p. 11. ISBN 1581138285. Disponível em: <http://dl.acm.org/citation.cfm?id=1015330.1015360http://portal.acm.org/citation.cfm?doid=1015330.1015360>.

BRZEZIŃSKI, D. **Mining Data Streams with Concept Drifts**. Dissertação (Mestrado) — Poznan University of Technology, Poland, 2010.

CAO, F. et al. Density-based clustering over an evolving data stream with noise. In: SIAM. **SDM**. [S.l.], 2006. v. 6, p. 328–339.

CASTELLANO, G.; FANELLI, A. M.; TORSELLO, M. A. Shape Annotation by Incremental Semi-supervised Fuzzy Clustering. In: MASULLI, F.; PASI, G.; YAGER, R. (Ed.). **Fuzzy Logic and Applications**. Cham: Springer International Publishing, 2013, (Lecture Notes in Computer Science, v. 8256). p. 193–200. ISBN 978-3-319-03199-6. Disponível em: <http://link.springer.com/10.1007/978-3-319-03200-9>.

CASTELLS, M. **A sociedade em rede**. São Paulo, Brasil: Paz e terra, 2007. v. 1.

CHAPELLE, O. et al. **Semi-supervised learning**. [S.l.]: MIT press Cambridge, 2006.

COLONNA, J. G.; GAMA, J.; NAKAMURA, E. F. Recognizing Family, Genus, and Species of Anuran Using a Hierarchical Classification Approach. In: . Springer, Cham, 2016. p. 198–212. Disponível em: <http://link.springer.com/10.1007/978-3-319-46307-0_13>.

DAVIDSON, I.; BASU, S. A survey of clustering with instance level constraints. **ACM Transactions on Knowledge Discovery from Data**, v. 1, p. 1–41, 2007.

DAVIDSON, I.; WAGSTAFF, K.; BASU, S. Measuring constraint-set utility for partitional clustering algorithms. In: FüRNKRANZ, J.; SCHEFFER, T.; SPILIOPOULOU, M. (Ed.). **Knowledge Discovery in Databases: PKDD 2006**. Springer Berlin Heidelberg, 2006, (Lecture Notes in Computer Science, v. 4213). p. 115–126. ISBN 978-3-540-45374-1. Disponível em: <http://dx.doi.org/10.1007/11871637_15>.

DEMPSTER, A. P.; LAIRD, N. M.; RUBIN, D. B. Maximum likelihood from incomplete data via the em algorithm. **Journal of the Royal Statistical Society. Series B (Methodological)**, JSTOR, p. 1–38, 1977.

DUBEY, A.; BHATTACHARYA, I.; GODBOLE, S. A cluster-level semi-supervision model for interactive clustering. In: **Machine Learning and Knowledge Discovery in Databases**. [S.l.]: Springer, 2010. p. 409–424.

ESTER, M. et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: **2nd International Conference on Knowledge Discovery and Data Mining**. [s.n.], 1996. p. 226–231. Disponível em: <http://www.aaai.org/Papers/KDD/1996/KDD96-037https://www-m9.ma.tum.de/foswiki/pub/WS2010/CombOptSem/DBScan.pdf>.

_____. A density-based algorithm for discovering clusters in large spatial databases with noise. In: **Knowledge Discovery and Data Mining - KDD**. [S.l.: s.n.], 1996. v. 96, n. 34, p. 226–231.

FACELI, K. et al. **Inteligência artificial: uma abordagem de aprendizado de máquina**. [S.l.]: Grupo Gen-LTC, 2011.

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

GAMA, J. **Knowledge discovery from data streams**. [S.l.]: CRC Press, 2010.

_____. **Knowledge discovery from data streams**. Chapman & Hall/CRC, 2010. 237 p. ISBN 9781439826126. Disponível em: <https://books.google.com.br/books?hl=pt-BR&lr=&id=r1XRBQAAQBAJ&oi=fnd&pg=PP1&ots=vk_UlkEZS-&sig=tRG6K15PaWqCJ1FSwRYPIT_buQE#v=onepage&q&f=false>.

GAMA, J. et al. A survey on concept drift adaptation. **ACM Computing Surveys (CSUR)**, ACM, v. 46, n. 4, p. 44, 2014.

GUHA, S.; RASTOGI, R.; SHIM, K. CURE: an efficient clustering algorithm for large databases. **ACM SIGMOD Record**, 1998. Disponível em: <http://dl.acm.org/citation.cfm?id=276312>.

HALKIDI, M.; SPILIOPOULOU, M.; PAVLOU, A. A Semi-supervised Incremental Clustering Algorithm for Streaming Data. In: TAN, P.-N. et al. (Ed.). **16th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD) Proceedings, Part I**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. (Lecture Notes in Computer Science, v. 7301), p. 578–590. ISBN 978-3-642-30216-9. Disponível em: <http://www.springerlink.com/index/10.1007/978-3-642-30217-6>.

HARTIGAN, J. A. **Clustering algorithms**. [S.l.]: John Wiley & Sons, Inc., 1975.

HARTUV, E.; SHAMIR, R. A clustering algorithm based on graph connectivity. **Information processing letters**, Elsevier, v. 76, n. 4, p. 175–181, 2000.

HINNEBURG, A.; KEIM, D. A. An efficient approach to clustering in large multimedia databases with noise. In: AAAI. **Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD)**. [S.l.], 1998. p. 58–65.

HORTON, P.; NAKAI, K. A probabilistic classification system for predicting the cellular localization sites of proteins. **Ismb**, 1996. Disponível em: <http://www.aaai.org/Papers/ISMB/1996/ISMB96-012.pdf>.

HUBERT, L.; ARABIE, P. Comparing partitions. **Journal of classification**, Springer, v. 2, n. 1, p. 193–218, 1985.

JAIN, A. K. Data clustering: 50 years beyond K-means. **Pattern Recognition Letters**, v. 31, n. 8, p. 651–666, jun. 2010. ISSN 01678655. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167865509002323>.

JIANG, H. et al. Extracting elite pairwise constraints for clustering. **Neurocomputing**, Elsevier, v. 99, p. 124–133, 2013.

KLEIN, D.; KAMVAR, S.; MANNING, C. From Instance-level Constraints to Space-Level Constraints: Making the Most of Prior Knowledge in Data Clustering. **Proceedings of the Nineteenth International Conference on Machine Learning**, p. 307–314, 2002. Disponível em: <http://dl.acm.org/citation.cfm?id=655989>.

KOHONEN, T. Self-organized formation of topologically correct feature maps. **Biological cybernetics**, Springer, v. 43, n. 1, p. 59–69, 1982.

LAI, H. P. et al. A new interactive semi-supervised clustering model for large image database indexing. **Pattern Recognition Letters**, Elsevier B.V., v. 37, n. 1, p. 94–106, 2 2014. ISSN 01678655. Disponível em: <http://dx.doi.org/10.1016/j.patrec.2013.06.014http://linkinghub.elsevier.com/retrieve/pii/S016786551300247X>.

_____. A new interactive semi-supervised clustering model for large image database indexing. **Pattern Recognition Letters**, Elsevier, v. 37, p. 94–106, 2014.

LELIS, L.; SANDER, J. Semi-supervised Density-Based Clustering. In: **2009 Ninth IEEE International Conference on Data Mining**. IEEE, 2009. p. 842–847. ISBN 978-1-4244-5242-2. ISSN 15504786. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5360321>.

LIU, E. Y.; ZHANG, Z.; WANG, W. Clustering with relative constraints. In: **Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '11**. New York, New York, USA: ACM Press, 2011. p. 947. ISBN 9781450308137. Disponível em: <http://dl.acm.org/citation.cfm?id=2020408.2020564>.

MACQUEEN, J. et al. Some methods for classification and analysis of multivariate observations. In: OAKLAND, CA, USA. **Proceedings of the fifth Berkeley symposium on mathematical statistics and probability**. [S.l.], 1967. v. 1, n. 14, p. 281–297.

OLIVEIRA, M.; GAMA, J. Bipartite graphs for monitoring clusters transitions. **Advances in Intelligent Data Analysis IX**, 2010. Disponível em: <http://link.springer.com/chapter/10.1007/978-3-642-13062-5_12>.

_____. A framework to monitor clusters evolution applied to economy and finance problems. **Intelligent Data Analysis**, IOS Press, v. 16, n. 1, p. 93–111, 2012.

PELLEG, D. et al. X-means: Extending K-means with efficient estimation of the number of clusters. **Proceedings of the Seventeenth International Conference on Machine Learning table of contents**, p. 727–734, 2000. ISSN 1467-9280. Disponível em: <http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.19.3377http://portal.acm.org/citation.cfm?id=657808>.

PENSA, R. G. et al. Co-clustering numerical data under user-defined constraints. **Statistical Analysis and Data Mining: The ASA Data Science Journal**, Wiley Online Library, v. 3, n. 1, p. 38–55, 2010.

RAND, W. M. Objective criteria for the evaluation of clustering methods. **Journal of the American Statistical association**, Taylor & Francis Group, v. 66, n. 336, p. 846–850, 1971.

RUIZ, C.; MENASALVAS, E.; SPILIOPOULOU, M. C-denstream: Using domain knowledge on a data stream. In: SPRINGER. **Discovery Science**. [S.l.], 2009. p. 287–301.

_____. C-DenStream: Using Domain Knowledge on a Data Stream. In: GAMA, J. et al. (Ed.). **12th International Conference on Discovery Science (DS)**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. (Lecture Notes in

Computer Science, v. 5808), p. 287–301. ISBN 978-3-642-04746-6. Disponível em: <http://www.springerlink.com/index/10.1007/978-3-642-04747-3>.

RUIZ, C.; SPILIOPOULOU, M.; MENASALVAS, E. **C-DBSCAN: Density-Based Clustering with Constraints**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. v. 4482. 216–223 p. (Lecture Notes in Computer Science, v. 4482). ISBN 978-3-540-72529-9. Disponível em: <http://www.springerlink.com/index/10.1007/978-3-540-72530-5>.

SCHMIDT, J.; BRÄNDLE, E. M.; KRAMER, S. Clustering with attribute-level constraints. In: IEEE. **Data Mining (ICDM), 2011 IEEE 11th International Conference on**. [S.l.], 2011. p. 1206–1211.

SCUDDER, H. J. Probability of error of some adaptive pattern-recognition machines. **IEEE Transactions on Information Theory**, IEEE, v. 11, n. 3, p. 363–371, 1965.

SHARAN, R.; SHAMIR, R. Click: a clustering algorithm with applications to gene expression analysis. In: **Proc Int Conf Intell Syst Mol Biol**. [S.l.: s.n.], 2000. v. 8, n. 307, p. 16.

SILVA, J. A. et al. Data stream clustering. **ACM Computing Surveys**, ACM, v. 46, n. 1, p. 1–31, out. 2013. ISSN 03600300. Disponível em: <http://dl.acm.org/citation.cfm?id=2522968.2522981>.

SILVA, W. **Incorporação de múltiplos representantes auxiliares em processos de detecção de agrupamentos semi-supervisionados**. Dissertação (Mestrado), 2015. Disponível em: <http://repositorio.ufu.br/handle/123456789/12596>.

SILVA, W. J. et al. Semi-supervised clustering using multi-assistant-prototypes to represent each cluster. In: **Proceedings of the 30th Annual ACM Symposium on Applied Computing - SAC '15**. New York, New York, USA: ACM Press, 2015. p. 831–836. ISBN 9781450331968. Disponível em: <http://dl.acm.org/citation.cfm?id=2695664.2695738>.

SIRAMPUJ, T.; KANGKACHIT, T.; WAIYAMAI, K. CE-Stream : Evaluation-based technique for stream clustering with constraints. In: **The 2013 10th International Joint Conference on Computer Science and Software Engineering (JCSSE)**. IEEE, 2013. p. 217–222. ISBN 978-1-4799-0806-6. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6567348>.

_____. Ce-stream: Evaluation-based technique for stream clustering with constraints. In: IEEE. **Computer Science and Software Engineering (JCSSE), 2013 10th International Joint Conference on**. [S.l.], 2013. p. 217–222.

SPILIOPOULOU, M. et al. MONIC: modeling and monitoring cluster transitions. In: **Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '06**. New York, New York, USA: ACM Press, 2006. p. 706. ISBN 1595933395. Disponível em: <http://dl.acm.org/citation.cfm?id=1150402.1150491>.

TAN PANG-NING; STEINBACH, M.; KUMAR, V. **Introduction to Data Mining**. [S.l.]: Addison Wesley, ISBN 0-321-32136-7, 2006.

TREECHALONG, K.; RAKTHANMANON, T.; WAIYAMAI, K. Semi-supervised stream clustering using labeled data points. In: PERNER, P. (Ed.). **Machine Learning and Data Mining in Pattern Recognition**. Springer International Publishing, 2015, (Lecture Notes in Computer Science, v. 9166). p. 281–295. ISBN 978-3-319-21023-0. Disponível em: <http://dx.doi.org/10.1007/978-3-319-21024-7_19>.

_____. Semi-supervised stream clustering using labeled data points. In: **11th International Conference on Machine Learning and Data Mining in Pattern Recognition (MLDM)**. [S.l.: s.n.], 2015. v. 9166, p. 281–295. ISBN 9783319210230. ISSN 16113349.

UDOMMANETANAKIT, K.; RAKTHANMANON, T.; WAIYAMAI, K. E-stream: Evolution-based technique for stream clustering. In: **Advanced Data Mining and Applications**. [S.l.]: Springer, 2007. p. 605–615.

WAGNER, S.; WAGNER, D. **Comparing clusterings: an overview**. [S.l.]: Universität Karlsruhe, Fakultät für Informatik Karlsruhe, 2007. <http://staff.ustc.edu.cn/~zwp/teach/MVA/cluster_validation.pdf>. (online; acessado em 14/12/2015).

WAGSTAFF, K. et al. Constrained k-means clustering with background knowledge. In: **ICML**. [S.l.: s.n.], 2001. v. 1, p. 577–584.

WAGSTAFF, K. L.; BASU, S.; DAVIDSON, I. When is constrained clustering beneficial, and why? **Ionosphere**, v. 58, n. 60.1, p. 62–3, 2006.

WAIYAMAI, K. et al. Sed–stream: discriminative dimension selection for evolution–based clustering of high dimensional data streams. **International Journal of Intelligent Systems Technologies and Applications**, Inderscience Publishers, v. 13, n. 3, p. 187–201, 2014.

ZAKI, M. J.; JR, W. M. **Data mining and analysis: fundamental concepts and algorithms**. [S.l.]: Cambridge University Press, 2014.

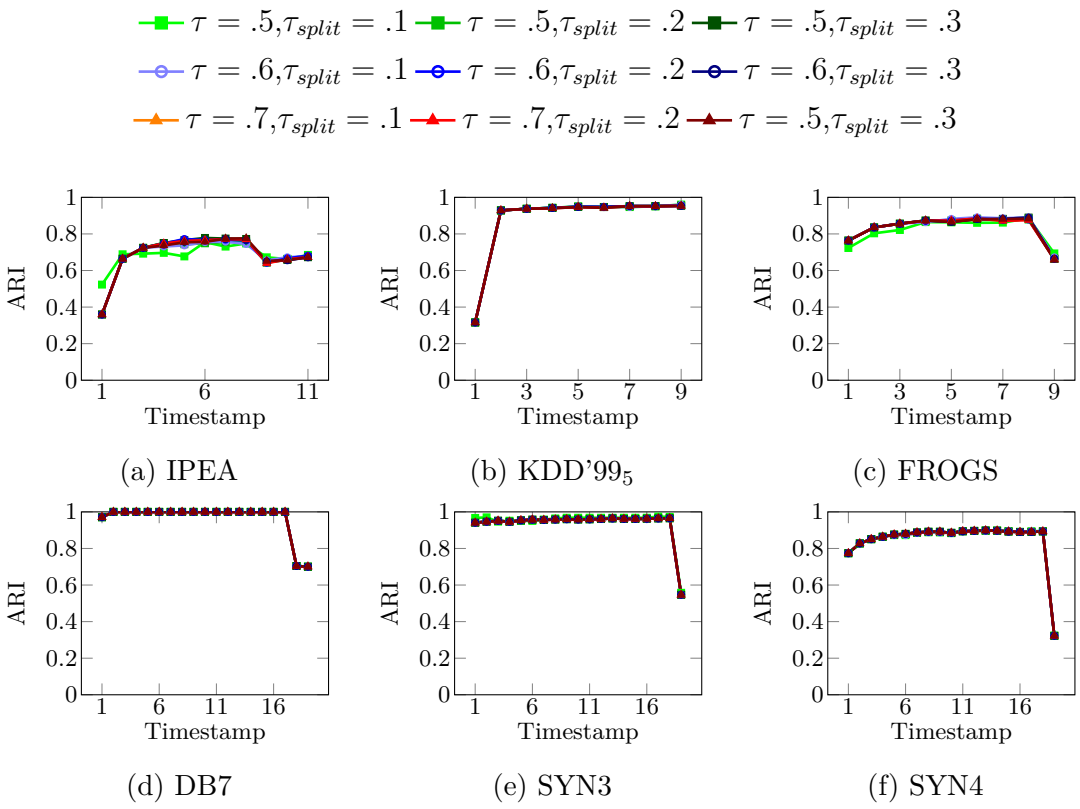
ZENG, Y. et al. An adaptive meta-clustering approach: combining the information from different clustering results. In: IEEE. **Bioinformatics Conference, 2002. Proceedings. IEEE Computer Society**. [S.l.], 2002. p. 276–287.

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

Apêndices

APÊNDICE A

Resultados experimentais variando os parâmetros do detector de transições



(a) IPEA (b) KDD'99$_5$ (c) FROGS

(d) DB7 (e) SYN3 (f) SYN4

Figura 37 – Efeito da variação dos parâmetros do detector de transições na eficácia do **Pointwise CABESS** considerando a **especialização dos grupos**.

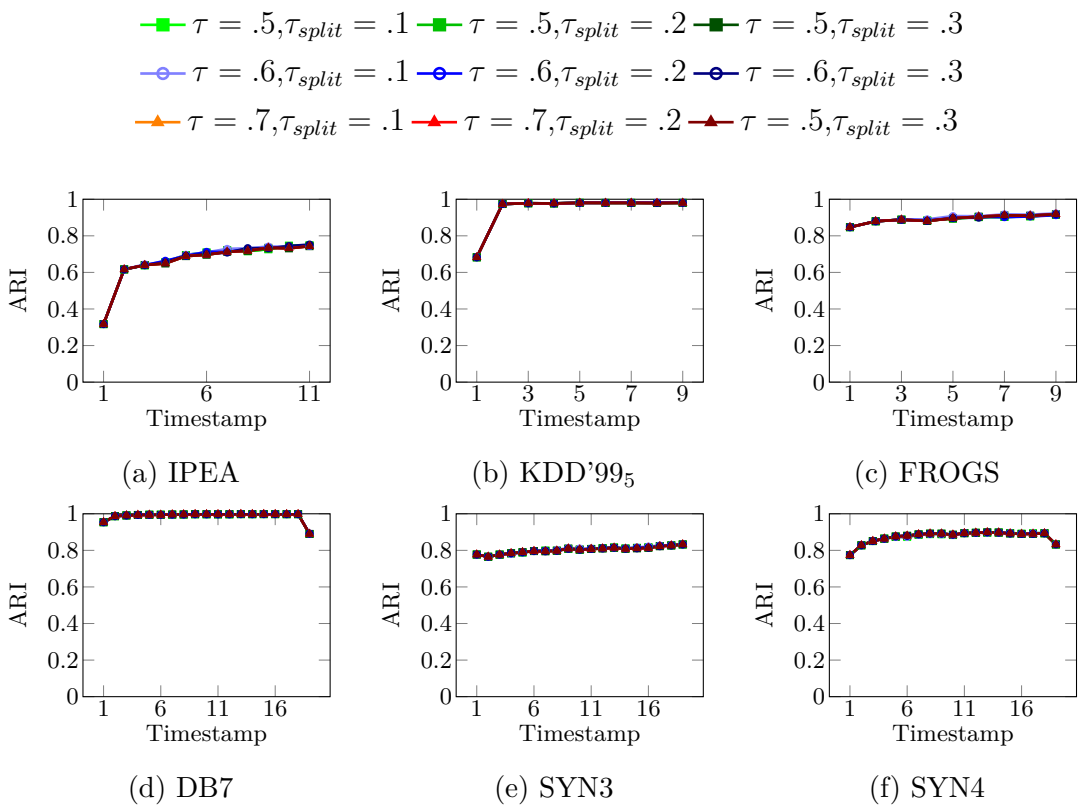


Figura 38 – Efeito da variação dos parâmetros do detector de transições na eficácia do **Pointwise CABESS** considerando a **generalização de grupos**.

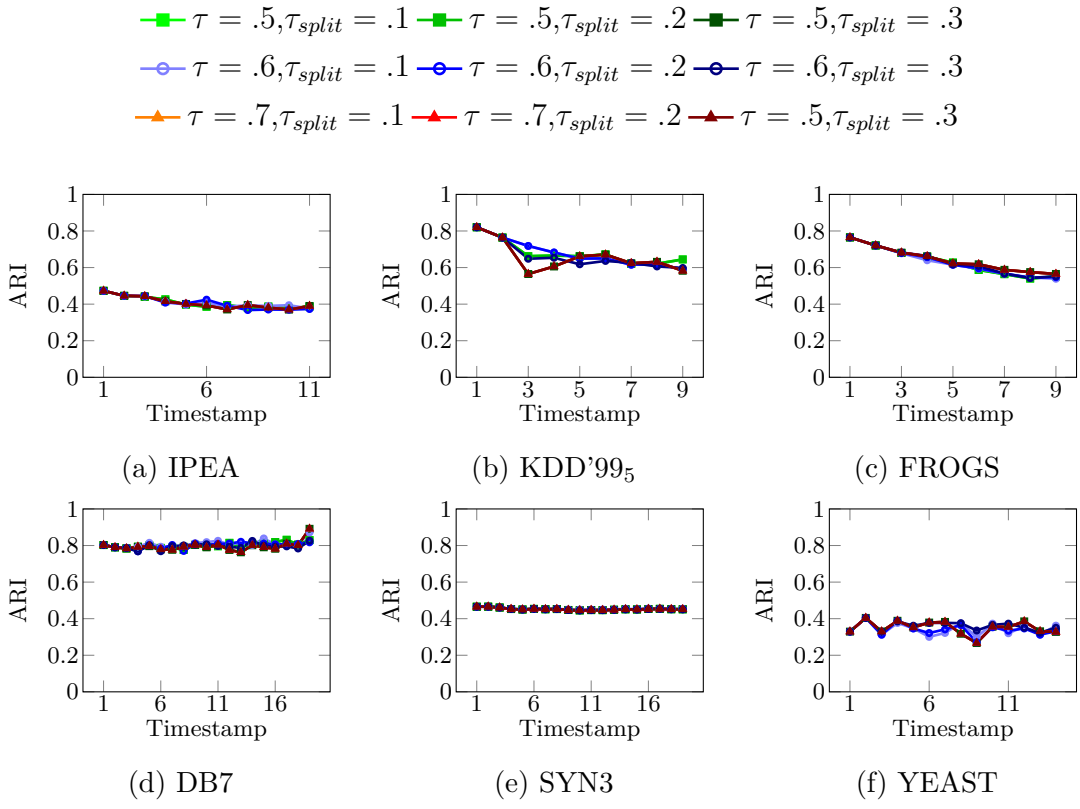


Figura 39 – Efeito da variação dos parâmetros do detector de transições na eficácia do **Pairwise CABESS** considerando a **generalização de grupos**.