JEAN PAUL BARDDAL

AGRUPAMENTO *ONLINE*: UMA ABORDAGEM BASEADA NA TEORIA DE REDES SOCIAIS

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática.

Curitiba
2015

JEAN PAUL BARDDAL

AGRUPAMENTO *ONLINE*: UMA ABORDAGEM BASEADA NA TEORIA DE REDES SOCIAIS

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática.

Área de Concentração: Ciência da Computação

Orientador: Fabrício Enembreck

Curitiba
2015

Barddal, Jean Paul
AGRUPAMENTO *ONLINE*: UMA ABORDAGEM BASEADA NA TEORIA DE REDES SOCIAIS. Curitiba, 2015.

Dissertação - Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática.

1. Mineração de Fluxos de Dados 2. Agrupamento *Online* 3. Mudança de Conceito I.Pontifícia Universidade Católica do Paraná. Escola Politécnica. Programa de Pós-Graduação em Informática II - t

Pontifícia Universidade Católica do Paraná
Escola Politécnica
Programa de Pós-Graduação em Informática

ATA DE DEFESA DE DISSERTAÇÃO DE MESTRADO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

DEFESA DE DISSERTAÇÃO DE MESTRADO Nº 01/2015

Aos 03 dias do mês de Março de 2015 realizou-se a sessão pública de Defesa da Dissertação **" Agrupamento Online: Uma Abordagem Baseada na Teoria de Redes Sociais"** apresentado pelo aluno **Jean Paul Barddal,** como requisito parcial para a obtenção do título de Mestre em Informática, perante uma Banca Examinadora composta pelos seguintes membros:

Prof. Dr. Fabrício Enembreck
PUCPR (Orientador) _____ APROVADO
(assinatura) (Aprov/Reprov)

Prof. Dr. Alceu de Souza Britto Jr.
PUCPR _____ APROV.
(assinatura) (Aprov/Reprov)

Prof. Dr. Júlio César Nievola
PUCPR _____ APROVADO
(assinatura) (Aprov/Reprov)

Prof. Dr. Luiz Eduardo Soares de Oliveira
UFPR _____ APROV
(assinatura) (Aprov/Reprov)

Conforme as normas regimentais do PPGIa e da PUCPR, o trabalho apresentado foi considerado APROVADO (aprovado/reprovado), segundo avaliação da maioria dos membros desta Banca Examinadora. Este resultado está condicionado ao cumprimento integral das solicitações da Banca Examinadora registradas no Livro de Defesas do programa.

Programa de Pós-Graduação em Informática PPGIa PUCPR

Prof.ª Dr.ª Andreia Malucelli
Coordenadora do Programa de Pós-Graduação em Informática.

Dissertação preparada com o formatador de textos LaTeX. A bibliografia é gerada automaticamente utilizando BibTeX e estilo abnTeX2. Todas as figuras e gráficos desta Dissertação foram desenvolvidas com os pacotes *TikZ*, *PGFPlots* e *GNUPlot*, sendo gerados durante compilação. Favor respeitar a licença requisitada pelo autor e citar seus trabalhos de maneira adequada.

Success is the result of perfection, hard work, learning from failure, loyalty and persistence. – Colin Powell

Agradecimentos

Primeiramente, ao meu orientador Prof. Fabrício Enembreck, aquele que me apresentou a pesquisa. O Prof. Fabrício proveu imensuráveis idéias, comentários e críticas sempre que necessário. Além de tantos atributos técnicos-científicos relevantes, devo ressaltar atributos como seriedade, objetividade, presteza e disponibilidade; todos essenciais para que este e outros trabalhos evoluíssem de tal maneira. Espero herdar tantas destas qualidades para minha futura carreira acadêmica. A sua participação neste e em outros projetos foi fundamental e impactou diretamente meu crescimento pessoal, intelectual e acadêmico. Muito obrigado pela confiança depositada em mim nestes anos de trabalho.

Agradeço ainda os colegas de laboratório André Pinz Borges e Osmar Betazzi Dordal pela companhia diária, amizade e troca de experiências. Desejo enorme sucesso aos dois em suas respectivas carreiras.

Em especial, agradeço ao Heitor Murilo Gomes, pela amizade, colaboração e inúmeras discussões sobre incontáveis temas. Espero ter a possibilidade de ajudá-lo tanto quanto me ajudou neste meu início de trabalhos de pesquisa.

Aos professores Alceu Britto Jr., Julio Cesar Nievola e Luiz Eduardo Soares de Oliveira, um agradecimento especial pelas considerações feitas sobre este trabalho. Suas participações foram de imensa valia para a melhoria deste projeto como um todo e as levarei para futuros trabalhos.

A Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) e Fundação Araucária, pelo apoio financeiro cedido a parte deste projeto.

Finalmente, aos meus pais. Nenhuma de minhas conquistas na vida teriam sido possíveis sem eles e seus incontáveis esforços e paciência. Por vocês, busco a excelência todos os dias.

Sumário

Lista de Algoritmos

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

$\mathcal{S}$	Fluxo contínuo de dados
$\vec{x}_i$	Objeto de dados formado por um vetor de características de tamanho d cujo instante de chegada é t_i
d	Dimensionalidade de um objeto
Cl_i	Grupo verdadeiro arbitrário i
$\mathcal{CL}$	Conjunto de grupos verdadeiros
l	Cardinalidade do conjunto $\mathcal{CL}$
$\mathscr{N}$	Conjunto de dados estático ou subpartição do fluxo de dados $\mathcal{S}$
$Cl(\vec{x}_i)$	Grupo verdadeiro de uma instância arbitrária $\vec{x}_i$
W	Tamanho da janela de mudança de conceito
t_{drift}	Momento da mudança de conceito
$\mathscr{K}$	Conjunto de grupos
K	Cardinalidade do conjunto $\mathscr{K}$
$\mathcal{H}_a$	Horizonte de avaliação
$\mathcal{H}$	Horizonte
CF	*Feature vector* composto por LS, SS e N
LS	Soma das componentes das instâncias sumarizadas
SS	Soma dos quadrados dos componentes das instâncias sumarizadas
N	Quantidade de instâncias sumarizadas
$\mu(\cdot)$	Centro de um *feature vector*
$r(\cdot)$	Raio de um *feature vector*
$diam(\cdot)$	Diâmetro de um *feature vector*
CF_i	Um *feature vector* arbitrário i
$\mathscr{W}$	Tamanho de janela interna de algoritmos de agrupamento
Ξ	Parâmetro que determina a taxa de decaimento de funções exponenciais
t_{atual}	Instante atual
α	Nível de significância para testes estatísticos

$\mathcal{M}$	Conjunto de *feature vectors* mantido pelo algoritmo CluStream
q	Cardinalidade do conjunto $\mathcal{M}$
$\mathcal{N}$	Tamanho de janela inicial
m	Número mínimo de entradas em uma árvore R
M	Número máximo de entradas em uma árvore R
$\mathbb{L}$	Número mínimo de entradas em um nó folha de árvore R
L	Número máximo de entradas em um nó folha de árvore R
CF_{objs}	*Feature vector* que sumariza objetos no algoritmo ClusTree
CF_{buffer}	*Feature vector* que sumariza objetos em *buffer* no algoritmo ClusTree
ϵ	Raio máximo de um *micro-cluster*
ψ	Número mínimo de vizinhos
$CMC(w, c, r, t_c, t_u)$	*Core micro-cluster* com peso w, centro c, raio r, instante de criação t_c e instante de última atualização t_u
w	Peso de um *micro-cluster*
c	Centro de um *micro-cluster*
r	Raio de um *micro-cluster*
t_c	Instante de criação de um *micro-cluster*
t_u	Instante da última atualização de um *micro-cluster*
$d(\cdot, \cdot)$	Distância Euclidiana
$f(\cdot)$	Função de decaimento exponencial
N_c	Número de *micro-clusters* mantido pelo algoritmo DenStream
Ψ	Somatório dos pesos dos *core micro-clusters*
β	Variável de controle de densidade
OMC_i	*Outlier micro-cluster* arbitrário i
PMC_i	Potencial *micro-cluster* arbitrário i
T_p	Tamanho da janela de avaliação de pesos
ξ	Limite mínimo de peso de *micro-clusters*
CS	Estabilidade de grupo – *Cluster Statibility*
$SSQ(\mathscr{K})$	Soma do quadrado das distâncias de um agrupamento $\mathscr{K}$
$\vec{\mu}_j$	Média de um grupo k_j
H	Entropia
$h(\mathscr{K}, \mathcal{CL})$	Homogeneidade
$c(\mathscr{K}, \mathcal{CL})$	Completude
Λ	Parâmetro de ponderação da medida V-Measure
$V(\mathscr{K}, \mathcal{CL}, \Lambda)$	Medida V-Measure utilizando o parâmetro Λ de ponderação
$py(\mathscr{K}\|\mathcal{CL})$	Pureza
$pr(k_i, Cl_j)$	*Precision*

$rc(k_i, Cl_j)$	*Recall*
$s(\vec{x}_i)$	Coeficiente da Silhueta para um objeto $\vec{x}_i$
$s(\mathscr{K})$	Coeficiente da silhueta de um agrupamento $\mathscr{K}$
λ	Parâmetro de uma função exponencial
Υ	Limiar de definição do horizonte de avaliação $\mathcal{H}_a$
$knhD(\cdot, \cdot)$	Distância média da k-vizinhança
$knh(\vec{x}_i, Cl_j)$	Conjunto dos k vizinhos mais próximos de $\vec{x}_i$ em Cl_j
$\mathcal{F}$	Conjunto de erros da métrica CMM
$Cl_{\text{ruído}}$	Grupo que representa dados ruidosos
$con(\vec{x}_i, Cl_j)$	Conectividade de uma instância $\vec{x}_i$ a um grupo verdadeiro Cl_j
$pen(\cdot, \cdot)$	Função de penalização da métrica CMM
$\mathcal{G}$	Notação utilizada para denotar um grafo
$\mathcal{V}$	Conjunto de vértices de um grafo
V	Cardinalidade do conjunto $\mathcal{V}$
$\mathcal{E}$	Conjunto de arestas de um grafo
E	Cardinalidade do conjunto $\mathcal{E}$
$\mathcal{W}$	Conjunto de pesos das arestas de um grafo
e_i	Aresta arbitrária i
w_i	Peso associado a uma aresta arbitrária e_i
$P(v_i, v_z)$	Caminhamento entre dois vértices arbitrários v_i e v_z
$\overline{P}(\mathcal{G})$	Tamanho médio de caminho de um grafo
$deg(\cdot)$	Grau de um vértice
$c_{deg}(v_i)$	Centralidade de grau de um vértice arbitrário v_i
$g(\cdot)$	Conjunto de geodésicas que passam por um vértice
$g'(\cdot)$	Centralidade de intermediação
$\mathcal{O}$	Notação de complexidade assintótica de pior caso
Q	Modularidade
L	Matriz laplaciana
D	Matriz diagonal
S	Matriz simétrica de similaridades/dissimilaridades de um grafo $\mathcal{G}$
S	Subconjunto de vértices em uma rede
v_i	Vértice arbitrário i pertencente a $\mathcal{V}$
ζ	Desvio entre o número correto de grupos l e o número K de grupos encontrado pelo algoritmo de agrupamento
$\overline{\zeta}$	Erro médio entre o número correto de grupos l e o número K de grupos encontrado pelo algoritmo de agrupamento computado através das subavaliações

Lista de Abreviações

CMM	*Cluster Mapping Measure*
SSQ	*Sum of Squared Distances – Soma do Quadrado das Distâncias*
NMI	*Normalized Mutual Information – Informação Mútua Normalizada*
MI	*Mutual Information – Informação Mútua*
CSV	*Comma Separated Values*
ARFF	*Attribute-Relation File Format*
RBF	*Radial Basis Function*
RAM	*Random Access Memory – Memória de Acesso Aleatório*
CD	*Critical Difference – Diferença Crítica*

Resumo

A Mineração de Fluxos Contínuos de Dados é uma área ativa de pesquisa que apresenta diversos desafios. Aplicações práticas da Mineração de Fluxos de Dados, como monitoramento de redes de computadores, pesquisas na Internet, serviços de telefonia e compras e detecção de fraudes de transações de cartão de crédito são caracterizadas pela necessidade de minerar massivas quantidades de dados que são obtidas de maneira serializada. Neste contexto, uma variedade de algoritmos de agrupamento foram propostos para realizar Aprendizagem de Máquina Não Supervisionada utilizando um modelo de duas etapas. Ainda, lidar com estes fluxos de dados não estacionários e possivelmente infinitos requer algoritmos capazes de realizar agrupamento de maneira rápida e incremental considerando limitações de tempo e memória sem comprometer a qualidade do agrupamento. Atualmente, os principais tópicos de pesquisa focam na possibilidade de encontrar grupos não hiper-esféricos e sensibilidade de parametrização. Este trabalho apresenta os algoritmos CNDenStream e SNCStream. Estes algoritmos foram projetados para a tarefa de agrupamento para fluxos de dados e são capazes de encontrar grupos não hiper-esféricos. Ao contrário dos demais algoritmos da literatura, estes utilizam apenas uma etapa de processamento para encontrar grupos ao utilizar uma rede inspirada no modelo de construção e evolução de redes sociais e em homofilia. Estudos empíricos mostram que os algoritmos CNDenStream e SNCStream são capazes de superar algoritmos não informados em termos de qualidade de agrupamento, não apresentam diferença estatística significativa quando comparados a abordagens informadas e requerem quantidade prática de recursos (tempo de processamento e memória) quando comparados a todos os demais algoritmos da literatura.

Palavras-chave: Mineração de Fluxos de Dados; Agrupamento *Online*, Mudança de Conceito; Análise de Redes Complexas.

Abstract

Data Stream Mining is an active area of research which poses challenging research problems. Typical applications, such as network monitoring, web searching, telephone services and credit card purchases and fraud detection are characterized by the need to mine massive amounts of data, which arrives continuously. In this context, a variety of data stream clustering algorithms have been proposed to perform unsupervised learning using a two-step framework. In addition, dealing with these non-stationary, unbounded data streams requires the development of algorithms capable of performing both fast and incremental clustering addressing time and memory limitations without jeopardizing clustering quality. Nowadays, current research topics regard the possibility of finding non-hyper-spherical clusters and parametrization sensitivity. This work presents the CNDenStream and SNCStream algorithms. CNDenStream and SNCStream are one-step data stream clustering algorithms capable of finding non-hyper-spherical clusters. In opposition to other data stream clustering algorithms, CNDenStream and SNCStream are capable of finding clusters by using a social network inspired formation and evolution model based on homophily. Empirical evaluations show that CNDenStream and SNCStream are able to surpass non-informed algorithms in clustering quality, does not present significant statistical difference when compared to informed approaches and require a feasible amount of resources (processing time and memory space) when compared to other algorithms presented in the literature.

Keywords: Data Stream Mining; Data Stream Clustering; Concept Drift; Social Network Analysis.

Capítulo 1

Introdução

Devido aos avanços tecnológicos em dispositivos para aquisição e armazenamento de dados, tornou-se possível que organizações adquirissem e armazenassem um extraordinário volume de dados. Por exemplo, a cada dia, são produzidos 2,5 quintilhões de *bytes*, e do total de volume de dados gerados desde o início da informática, estima-se que 90% foram criados apenas no período entre 2012 e 2014 (AMINI; WAH, 2014).

Como os dados brutos não permitem a identificação rápida de padrões de comportamento, técnicas de Mineração de Dados, especialmente o Aprendizado Indutivo, são bastante difundidas para extrair conhecimento a partir destes dados. O Aprendizado Indutivo tem como objetivo realizar generalizações a partir de exemplos particulares de dados (instâncias) e é um dos principais métodos utilizados para derivar novos conhecimentos e predizer eventos futuros (REZENDE, 2003). O Aprendizado Indutivo é dividido em **supervisionado** e **não supervisionado**.

No aprendizado supervisionado, define-se um algoritmo de aprendizado (indutor) e a ele é fornecido um conjunto de exemplos rotulados por um atributo meta. Neste caso, o objetivo é construir um indutor que possa determinar corretamente o atributo meta de exemplos ainda não rotulados. Quando os valores possíveis de rótulos são discretos, têm-se um problema de **classificação**. Por outro lado, quando os valores possíveis do atributo meta são contínuos, a tarefa é denominada **regressão**.

No aprendizado não supervisionado, destaca-se a tarefa de **agrupamento**, onde o indutor analisa os exemplos fornecidos e tenta determinar se alguns deles podem ser agrupados de alguma maneira (CHEESEMAN et al., 1988). Essencialmente, o problema de agrupamento pode ser definido como determinar um conjunto finito de grupos (*clusters*) que representem de maneira fiel um conjunto de dados (HAN; KAMBER; PEI, 2011).

Em primeira instância, o Aprendizado Indutivo parece ser uma abordagem interessante para extrair conhecimento a partir de massivas quantidades de dados. Contudo,

esta abordagem é muitas vezes impossível pelo alto custo computacional de suas técnicas.

Atualmente, o interesse de minerar um tipo específico de massivas quantidades de dados, que por sua vez são geradas de maneira contínua, serializada e potencialmente infinitas, denominadas fluxos contínuos de dados *data streams* tem crescido nos últimos anos (GAMA et al., 2014; GAMA, 2010; SILVA et al., 2013). Por mais que obter estas imensas quantidades de dados serializados não seja um desafio, extrair conhecimento útil e não óbvio é, principalmente pelo fato de que a rotulação das instâncias é comumente feita por especialistas humanos ou fontes externas de dados que não são facilmente acessáveis.

Como não é plausível carregar os dados vindos de um fluxo de dados em memória, seja pela escala de espaço necessário ou assumir que o poder computacional é suficiente, técnicas convencionais de aprendizagem de máquina em lote (*batch*) não são viáveis. Deste modo, a área de Mineração de Fluxos Contínuos de Dados (*Data Stream Mining*), ou Aprendizagem de Máquina *Online*, foi criada com o objetivo de descobrir conhecimento incrementalmente a partir destas gigantescas sequências de dados.

Dentro da mineração de fluxos contínuos de dados, deve-se ressaltar algumas restrições (SILVA et al., 2013): instâncias são obtidas continuamente (de maneira serializada) e não existe controle sobre a ordem de processamento destas. Ainda, o tamanho de um fluxo contínuo de dados é potencialmente infinito, logo, instâncias devem ser descartadas logo após seu processamento para que restrições de tempo de processamento e de armazenamento em memória sejam respeitadas. Finalmente, a geração de dados desconhecida é possivelmente **não estacionária** (evolucionária), ou seja, sua distribuição de probabilidade pode mudar com o tempo.

O foco deste trabalho é a tarefa de Agrupamento para fluxos contínuos de dados. A importância da tarefa de Agrupamento para fluxos de dados se dá pela facilidade da obtenção de dados, contudo, é inviável assumir que estes estejam rotulados. Deste modo, processá-los em alta velocidade e fornecer conhecimento vantajoso a partir de dados não rotulados para especialistas é um desafio atual de pesquisa alcançável apenas com abordagens não supervisionadas. Como aplicações de algoritmos de Agrupamento para fluxos de dados, elenca-se: fluxos de cliques de consumidores, fluxos de utilização de telefone, dados multimídia e em especial vídeo (GUHA, 2009; SILVA et al., 2013), detecção de intrusão em redes de computadores (AGGARWAL et al., 2003), mineração de estruturas XML e HTML (AGGARWAL; YU, 2006) e agrupamento de dados providos por redes de sensores (RODRIGUES; GAMA; PEDROSO, 2008).

Este trabalho apresenta os algoritmos CNDenStream e SNCStream, desenvolvidos para realizar agrupamento de fluxos contínuos de dados respeitando as limitações de tempo e espaço decorrentes do ambiente *online*. Ainda, estes algoritmos não necessitam de um

parâmetro que defina o número de grupos a serem encontrados e são capazes de encontrar grupos não hiper-esféricos, sobrepujando limitações de outros algoritmos da literatura. O interesse em encontrar grupos não hiper-esféricos se dá pelo fato da maioria dos fluxos de dados reais não ser regido por Gaussianas, deste modo, abordagens baseadas em *k-means* (LLOYD, 1982) são incapazes de encontrar tais grupos perfeitamente.

1.1 Motivação e Hipótese

A maioria dos algoritmos capazes de realizar a tarefa de agrupamento para fluxos de dados são parcialmente incrementais (por computarem apenas seus sumários estatísticos de maneira incremental, mas não seus grupos) ou possuem quantidade elevada de parâmetros. Ainda, muitas vezes, são incapazes de encontrar grupos não hiper-esféricos e outros, recebem como parâmetro o número de grupos a ser encontrado. A motivação deste trabalho é tentar sobrepujar tais aspectos, inicialmente, apresentando uma maneira de encontrar grupos de forma completamente incremental; diminuindo o número de parâmetros necessários; encontrando grupos de formato arbitrário e eliminando a intervenção do usuário acerca do número de grupos a serem encontrados.

Um dos principais métodos para encontrar grupos de formato não hiper-esférico apresentado na literatura é o agrupamento baseado em Grafos (HAN; KAMBER; PEI, 2011). Não obstante, algoritmos de agrupamentos baseados em Grafos e algoritmos para detecção de grupos (comunidades) em redes são baseados na Teoria de Grafos e são custosos computacionalmente, logo, não são aplicáveis ao problema de agrupamento *Online*. Por outro lado, a Teoria de Redes Sociais provê mecanismos que formalizam a formação e evolução de redes de acordo com o tempo, se tornando uma base essencial para os algoritmos propostos.

A hipótese deste trabalho é que uma rede, inspirada no modelo de formação e evolução de redes livre de escala, é capaz de representar a dinamicidade dos grupos durante o fluxo de dados. Ainda, devido ao modelo de evolução baseado em homofilia, que visa manter objetos de dados similares conectados, e objetos de dados dissimilares, disconexos; nenhum tipo de processamento custoso em etapa *offline* para encontrar grupos é necessário. Como uma rede social é representada computacionalmente como um grafo, não existem restrições sobre a forma dos grupos, permitindo então que os algoritmos propostos sejam capazes de encontrar grupos de formato arbitrário.

1.2 Objetivos

Este trabalho tem como objetivo principal o desenvolvimento de algoritmos baseados na teoria de redes sociais para o problema de Agrupamento *Online*. Os objetivos específicos incluem a especificação e implementação de um modelo evolutivo de rede que permita encontrar grupos de maneira incremental, o estudo da rede social construída, a análise do algoritmo em fluxos contínuos de dados estacionários e evolucionários, assim como uma avaliação empírica dos resultados.

1.3 Organização

Este trabalho está dividido da seguinte maneira: O Capítulo 2 apresenta o problema de Mineração de Fluxos Contínuos de Dados, focando na tarefa de Agrupamento *Online*. O Capítulo 3 discute sobre os conceitos de Análise de Redes Sociais relevantes para este trabalho. O Capítulo 4 apresenta os algoritmos propostos, enquanto o Capítulo 5 discute os resultados obtidos quando estes algoritmos são avaliados perante seus principais parâmetros e confrontados a problemas sintéticos e reais. Finalmente, o Capítulo 6 apresenta a conclusão deste trabalho.

Capítulo 2

Mineração de Fluxos Contínuos de Dados

Avanços recentes em *hardware* e *software* permitiram a aquisição de dados em larga escala. Contudo, os avanços em termos de aquisição são incompatíveis com os de processamento. Logo, lidar com esta massiva quantidade de dados se tornou um desafio para pesquisadores devido às limitações físicas dos computadores atuais.

Na última década, o interesse em gerenciar e extrair conhecimento de sequências de dados potencialmente infinitas geradas rapidamente, denominadas **fluxos contínuos de dados** (*data streams*) tem aumentado (AGGARWAL, 2007; GAMA, 2010). Formalmente, tem-se um fluxo de dados $\mathcal{S}$ que gera objetos de dados (instâncias) $\vec{x}_i$ de dimensionalidade d, de maneira serializada e potencialmente infinita: $\mathcal{S} = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_\infty\}$. Para acessar o valor de uma dimensão (atributo) v qualquer de uma instância $\vec{x}_i$, adota-se a notação $\vec{x}_{i;v}$.

Um grupos verdadeiro Cl_i é um conjunto de objetos cuja distribuição no espaço de características é dada por uma densidade específica. Tendo em vista um conjunto de grupos verdadeiros $\mathcal{CL}$ e em que $\mathcal{CL} = \{Cl_1, Cl_2, Cl_3, \ldots, Cl_l\}$, assume-se que $l \geq 1$, i.e. deve existir ao menos um grupo verdadeiro em $\mathcal{S}$ ou em cada subpartição $\mathscr{N}$ de $\mathcal{S}$, onde $\mathscr{N} \subseteq \mathcal{S}$. Para determinar o grupo verdadeiro de um objeto $\vec{x}_i$, adota-se a notação $Cl(\vec{x}_i)$.

Aplicações de fluxos de dados incluem, por exemplo, a mineração de dados gerados por redes de sensores (SABIT; AL-ANBUKY; GHOLAMHOSSEINI, 2011), análise da bolsa de valores (BARDDAL; ENEMBRECK, 2013; BARDDAL; GOMES; ENEMBRECK, 2015a) e monitoramento de tráfego de rede (LALL et al., 2006). Todas estas aplicações envolvem conjuntos de dados grandes demais para serem armazenados em memória principal, logo, são armazenadas em memória secundária. Como realizar buscas utilizando acesso aleatório em memória secundária é um processo custoso, o único método de acesso plausível é desempenhar acessos lineares aos dados, de acordo com a sua chegada, procedimento denominado *single-pass processing* (GUHA, 2009).

A extração de conhecimento de fluxos de dados é um desafio por si só. A maioria das técnicas de mineração de dados e descoberta do conhecimento assume que existe uma quantidade da dados conhecida, modelada por uma distribuição de probabilidade estacionária e passível de ser analisada em vários passos por um algoritmo em formato *batch*, contudo, nenhuma destas condições é verificada no contexto de fluxos de dados.

O desenvolvimento de algoritmos de agrupamento para fluxos contínuos de dados é um desafio para pesquisadores. Em contraste com o número de algoritmos desenvolvidos para o problema de Classificação, a quantidade de algoritmos para agrupamento é pequena, contudo, tem ganho ênfase na comunidade científica (AMINI; WAH, 2014; SILVA et al., 2013).

Este capítulo está dividido da seguinte maneira: A Seção 2.1 apresenta o problema de Mudança de Conceito, focando o tarefa de agrupamento, enquanto a Seção 2.2 apresenta o problema de Evolução de Conceito. A Seção 2.3 discute os conceitos básicos de Agrupamento *Online*. A Seção 2.4 discorre acerca das principais características de algoritmos de Agrupamento *Online*: estruturas de dados, modelos de janelas, mecanismos de detecção de dados ruidosos e *outliers* e a divisão nas etapas *online* e *offline*. A Seção 2.5 apresenta os principais algoritmos para esta tarefa, explicitando suas principais características. A Seção 2.7 apresenta um resumo deste capítulo, concluindo sobre os principais aspectos de Agrupamento *Online* relevantes para o remanescente deste trabalho.

2.1 Mudança de Conceito

Um conceito pode ser representado por uma função lógica que mapeia valores de atributos para um determinado atributo meta (WIDMER; KUBAT, 1996). Uma maneira de formalizar este tipo de função é através de uma disjunção de conjunções. A Tabela 2.1 fornece um exemplo de conceito no domínio comercial, onde os atributos sumarizam uma compra de um determinado produto, especificado pelo seu "preço" e "qualidade". Finalmente, o atributo meta "compra" determina se um determinado cliente efetuou a compra ou não daquele produto.

O conceito apresentado na Tabela 2.1 representa a hipótese de que se uma instância qualquer possuir os valores de preço inferior ou igual a 500, qualidade "baixa" ou "normal", logo, a classe (compra) assumirá o valor "sim".

Todavia, um dos maiores problemas com o aprendizado indutivo em fluxos de dados é que um conceito pode depender de valores não presentes na base de dados, não sendo representado por nenhum dos atributos, mas sim, por um contexto desconhecido (*hidden context*) (SCHLIMMER; GRANGER, 1986; TSYMBAL, 2004). Deste modo, o

Atributo	Domínio
Preço	$\mathbb{R}^+$
Qualidade	[baixa, normal, alta]
Compra (classe)	[sim, não]
Conceito	Preço $\leq 500 \wedge$ (Qualidade = baixa $\vee$ Qualidade = normal) $\Rightarrow$ compra = sim

Tabela 2.1: Exemplo de conceito.

Atributo	Domínio
Preço	$\mathbb{R}^+$
Qualidade	[baixa, normal, alta]
Compra (classe)	[sim, não]
Conceito	Preço $\geq 750 \wedge$ (Qualidade = normal $\vee$ Qualidade = alta) $\Rightarrow$ compra = sim

Tabela 2.2: Exemplo de possível modificação no conceito apresentado na Tabela 2.1.

conceito apresentado na Tabela 2.1 pode variar de acordo com a mudança no padrão de compra dos consumidores. A variação do padrão de compra pode se dar, por exemplo, devido à mudanças na inflação, época do ano e/ou campanhas publicitárias.

Um exemplo prático seria a mudança do conceito da Tabela 2.1 devido a proximidade do Natal. É possível que durante o ano inteiro os consumidores realizem suas compras de acordo com o conceito apresentado na Tabela 2.1, mas, durante o período entre o final de Outubro e Dezembro, o padrão de compra mude, afetando assim o conceito que deverá ser derivado pelo indutor de acordo com a chegada dos objetos de dados. A Tabela 2.2 apresenta um possível conceito derivado desta mudança, onde o cliente compraria um produto caso o preço fosse menor ou igual a 750 e a qualidade do produto fosse normal ou alta.

Em outras palavras, uma mudança de conceito acontece quando instâncias que eram mapeadas por um conceito A para um atributo meta Cl_y passam a ser mapeadas por um conceito B. Mesmo a mudança de conceito sendo possível em problemas de Regressão, neste documento formaliza-se apenas a noção de conceito em problemas de Classificação, tendo em vista que esse segundo modelo de aprendizagem possui características que auxiliam a compreensão do problema de Agrupamento *Online*, sendo este o foco do presente trabalho. O problema de mudança de conceito em problemas de regressão é discutido em (BARDDAL; ENEMBRECK, 2013; BARDDAL; GOMES; ENEMBRECK, 2015a; YEON et al., 2010).

A causa de mudanças de conceito pode não ser determinada, muito menos prevista por algoritmos convencionais de indução, uma vez que não se pode assumir que estes dispõem de fontes de dados secundárias, ou que seu custo de acesso seja muito alto. Espera-se então que um algoritmo de indução detecte esta mudança e se adapte rapidamente a ela de maneira automática e autônoma (sem interferência de usuários).

Caso o processo gerador de dados seja não estacionário (como na maioria das aplicações reais), mudanças no contexto realizarão mudanças no conceito a ser aprendido; logo, detectar e se adaptar a mudanças de conceito é uma obrigatoriedade (GAMA et al., 2014). Todavia, espera-se ainda que algoritmos desenvolvidos com esta finalidade sejam capazes de discernir entre uma verdadeira mudança de conceito e o aparecimento de dados ruidosos e *outliers* (WIDMER; KUBAT, 1996).

Uma das principais maneiras de modelar mudanças de conceito de maneira matemática é através de probabilidades. De acordo com a teoria Bayesiana, um problema de classificação pode ser descrito pelas probabilidades *a priori* das classes $P[Cl(\vec{x})]$ e as funções de densidade das probabilidades condicionais $P[\vec{x}|Cl(\vec{x})]$ (DUDA; HART; STORK, 2001). A decisão da classificação é realizada de acordo com a probabilidade *a posteriori* das classes. A Equação 2.1 apresenta o cálculo *a posteriori* bayesiano onde $P[\vec{x}] = \sum_{Cl_i}^{\mathcal{CL}} P[Cl_i] \times P[\vec{x}|Cl_i]$.

$$P[Cl(\vec{x})|\vec{x}] = \frac{P[Cl(\vec{x})] \times P[\vec{x}|Cl(\vec{x})]}{P[\vec{x}]} \tag{2.1}$$

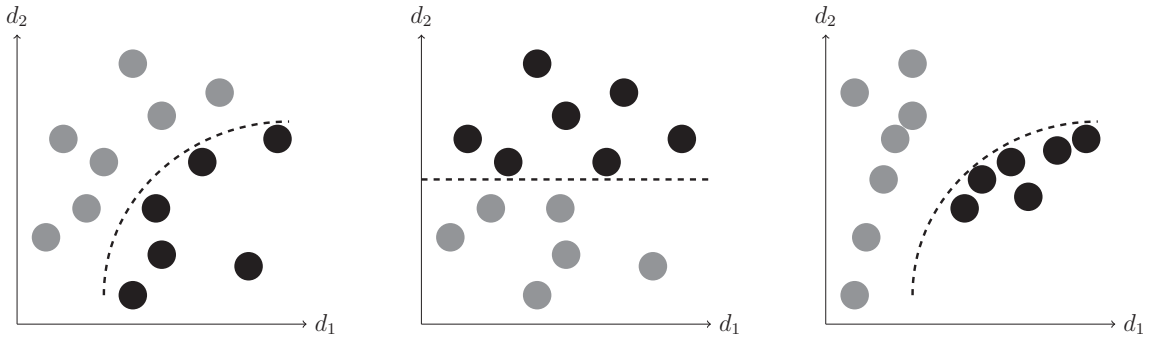
Formalmente, uma mudança de conceito entre um instante t_0 e um instante t_1 é definida pela Expressão 2.2 onde p_{t_0} denota a distribuição conjunta no instante t_0 entre o conjunto de atributos $\vec{x}$ e a classe $Cl(\vec{x})$.

$$\exists \vec{x}_i : P_{t_0}[\vec{x}_i, Cl(\vec{x}_i)] \neq P_{t_1}[\vec{x}_i, Cl(\vec{x}_i)] \tag{2.2}$$

Mudanças nos dados podem ser caracterizadas como mudanças nos componentes desta relação. Em outros termos, as probabilidades *a priori* $P[Cl(\vec{x})]$ ou as probabilidades condicionais $P[\vec{x}|Cl(\vec{x})]$ podem mudar, acarretando em mudanças nas probabilidades *a posteriori* das classes $P[Cl(\vec{x})|\vec{x}]$.

Deve-se ainda fazer distinção entre dois tipos de mudanças de conceito apresentadas na literatura: mudanças de conceito **reais** e **virtuais**.

Mudanças reais de conceito se referem a mudanças nas probabilidades $P[Cl(\vec{x})|\vec{x}]$. Estas mudanças podem acontecer com ou sem mudanças em $P[\vec{x}]$. Por outro lado, mudanças virtuais de conceito acontecem quando a probabilidade $P[\vec{x}]$ é alterada, contudo, sem afetar $P[Cl(\vec{x})|\vec{x}]$. O termo "Mudança Virtual de Conceito" é apresentado de dife-



(a) Dados originais.

(b) Mudança real de conceito, onde $P[Cl(\vec{x})|\vec{x}]$ muda.

(c) Mudança virtual de conceito onde $P[\vec{x}]$ muda, mas $P[Cl(\vec{x})|\vec{x}]$ não.

Figura 2.1: Tipos de mudança de conceito: círculos representam instâncias, níveis de cinza diferentes representam classes diferentes e a linha tracejada define o limiar de separação inter-classes.

rentes maneiras na literatura. Originalmente, uma mudança virtual acontecia devida à incompleta representação dos dados, ao contrário de mudanças na distribuição dos dados (WIDMER; KUBAT, 1996). Mudanças virtuais poderiam corresponder também a mudanças na distribuição dos dados que acarretavam em mudanças na tomada de decisão (TSYMBAL, 2004). Finalmente, mudanças virtuais de conceito eram aquelas que não alteravam o conceito a ser aprendido (DELANY et al., 2005).

A Figura 2.1 apresenta mudanças de conceito reais e virtuais em um ambiente bidimensional, descrito pelas dimensões d_1 e d_2. Os gráficos apresentam que apenas a mudança real afeta o limiar de separação entre as classes, fazendo com que o modelo antigo se torne obsoleto. Na prática, é possível que mudanças reais sejam acompanhadas de mudanças virtuais. Nestes casos, o limiar de separação entre as classes também é afetado.

Mudanças de conceito podem acontecer de duas maneiras: abrupta ou gradualmente. Para determinar se uma mudança está ocorrendo de maneira abrupta ou gradual, deve-se analisar o tamanho da janela de mudança W. Hipoteticamente, considerando que uma mudança ocorra a partir de uma instância $\vec{x}_i$ e que ela se torna estável a partir da instância $\vec{x}_{i+W}$, se $W = 1$, a mudança é denominada abrupta, caso contrário ($W \geq 2$), gradual. A Figura 2.2 exemplifica uma mudança abrupta de conceito em $\vec{x}_6$, enquanto a Figura 2.3 demonstra uma mudança gradual com $W = 4$ onde $\vec{x}_4$ é o início da janela e $\vec{x}_7$ o seu término.

Dentro de uma janela de mudança, a probabilidade de uma instância $\vec{x}_i$ qualquer pertencer ao conceito antigo (A, no exemplo) ou ao novo (B, no exemplo) é específica de cada problema. Contudo, muitas destas probabilidades podem ser modeladas através de funções de probabilidades conhecidas. Um exemplo prático e simples seria uma função

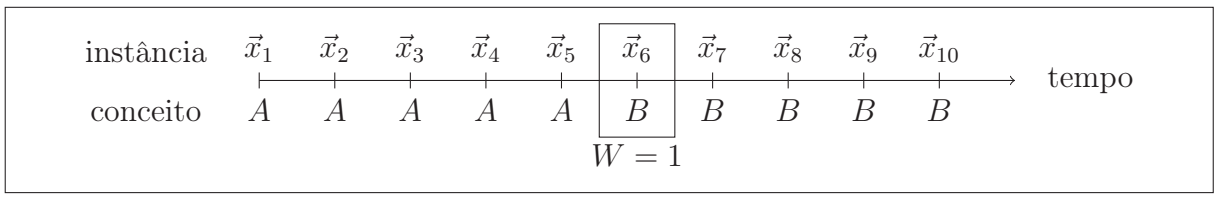


Figura 2.2: Exemplo de mudança abrupta de conceito.

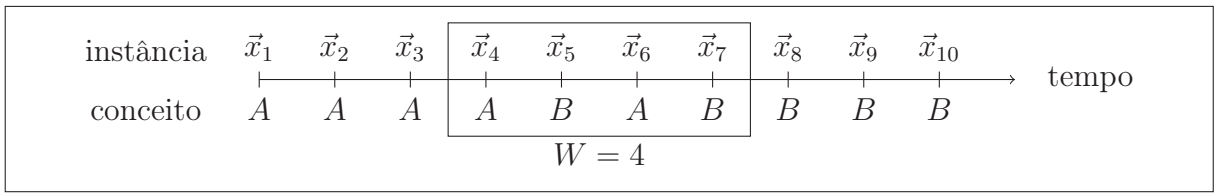


Figura 2.3: Exemplo de mudança gradual de conceito.

i	Situação	Conceito que rege $\vec{x}_i$	$P[\vec{x}_i \in A]$	$P[\vec{x}_i \in B]$
1	A estável	A	1	0
2	A estável	A	1	0
3	A estável	A	1	0
4	Zona de mudança	A	$1 - \frac{4-4}{4-1} = 0$	$\frac{4-4}{4-1} = 0$
5	Zona de mudança	B	$1 - \frac{5-4}{4-1} \approx 0,667$	$\frac{5-4}{4-1} \approx 0,333$
6	Zona de mudança	A	$1 - \frac{6-4}{4-1} \approx 0,333$	$\frac{6-4}{4-1} \approx 0,667$
7	Zona de mudança	B	$1 - \frac{7-4}{4-1} = 0$	$\frac{7-4}{4-1} = 1$
8	B estável	B	0	1
9	B estável	B	0	1
10	B estável	B	0	1

Tabela 2.3: Detalhamento de mudança de conceito gradual.

linear $P[\vec{x}_i \in B] = \frac{(i-c)}{W-1}$, onde i é o índice da instância atual, c é o índice do início da janela e W o comprimento da janela. A probabilidade de uma instância $\vec{x}_i$ qualquer pertencer ao conceito B é $P[\vec{x}_i \in B]$ e de pertencer ao conceito A é $P[\vec{x}_i \in A] = 1 - P[\vec{x}_i \in B]$. Um ponto importante em diversas funções de probabilidades é o definido pela interseção das probabilidades, ou seja, quando $P[\vec{x}_i \in A] = P[\vec{x}_i \in B] = 0,5$. Este ponto é conhecido como "momento de mudança", doravante denominado t_{drift}.

A Tabela 2.3 apresenta um detalhamento sobre estas probabilidades juntamente da Figura 2.4, onde uma abordagem linear de mudança de conceito é utilizada.

Problemas onde um conceito é substituído, e após uma janela de instâncias W, volta a acontecer, não é abordado neste trabalho. Este problema é denominado "recorrência de conceito" e técnicas para este tipo de problema são discutidas em (MASUD et al., 2011; SILVA et al., 2013). O problema de recorrência de conceito não é um tema bastante

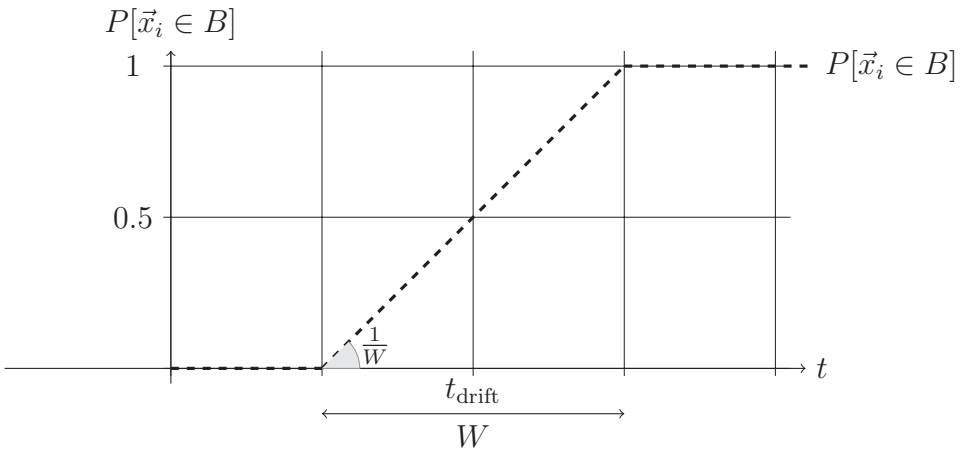
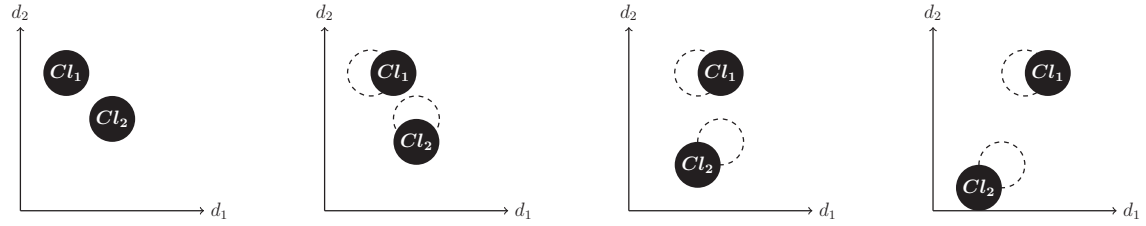


Figura 2.4: Modelagem linear de mudança de conceito gradual.

aprofundado na literatura. A principal razão para este fato se dá pela possibilidade de dividir este problema em duas mudanças de conceito. Deste modo, caso o algoritmo de aprendizagem utilizado possua alta capacidade de detectar e se adaptar a mudanças de conceito, este será capaz de se adaptar também a recorrências de conceito. Uma abordagem ingênua para tratar o problema de recorrência de conceito é armazenar os sumários estatísticos ou grupos que representam o estado atual do fluxo de dados. Todavia, tal abordagem pode não ser aplicável devido a restrição de memória imposta pelo ambiente *online*.

Dentro do contexto de agrupamento, a mudança de conceito pode ser facilmente visualizada graficamente. Para este exemplo, assume-se um espaço de atributos bidimensional ($d = 2$) representado pelas coordenadas d_1 e d_2 na Figura 2.5 onde são representadas dois grupos verdadeiros hiper-esféricos (Cl_1 e Cl_2). Os grupos verdadeiros Cl_1 e Cl_2 desempenham uma mudança de conceito gradual. A Figura 2.5a apresenta a disposição inicial dos grupos verdadeiros Cl_1 e Cl_2 no espaço de atributos. A Figura 2.5b apresenta a movimentação destes grupos verdadeiros, onde o grupo Cl_1 agora é representado por um conceito levemente diferente deslocado a direita e Cl_2, por um conceito deslocado para baixo e esquerda. A Figura 2.5c apresenta a continuidade desta movimentação, sendo que tanto Cl_1 quanto Cl_2 continuam se deslocando nas mesmas direções e sentidos. Finalmente, a Figura 2.5d apresenta a estagnação do conceito, ou seja, o final da janela de mudança, onde Cl_1 e Cl_2 são representados em novas posições dentro do espaço de dados.

Outra possibilidade de mudança de conceito no contexto de agrupamento é a mudança dos grupos verdadeiros no espaço de características sem existência de submovimentações. Um exemplo é apresentado na Figura 2.6, onde o grupos Cl_2 muda sua posição no espaço de atributos sem existir posições intermediárias.



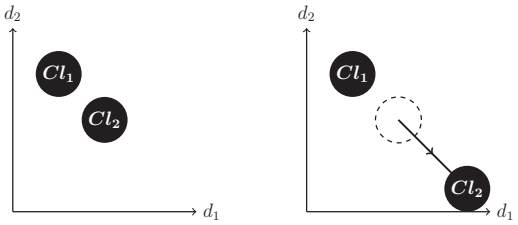
(a) Disposição inicial dos grupos verdadeiros.

(b) Primeira movimentação dos grupos verdadeiros no espaço.

(c) Segunda movimentação dos grupos verdadeiros no espaço.

(d) Disposição final dos grupos verdadeiros.

Figura 2.5: Exemplo de grupos verdadeiros em processo de mudança de conceito em espaço bidimensional.



(a) Disposição inicial dos grupos verdadeiros Cl_1 e Cl_2.

(b) Disposição final dos grupos verdadeiros.

Figura 2.6: Exemplo de grupos verdadeiros em processo de mudança de conceito sem posições intermediárias.

2.2 Evolução de Conceito

Maior parte dos algoritmos criados para minerar fluxos de dados focam no problema de mudança de conceito. Contudo, outra importante característica de fluxos de dados é a Evolução de Conceito (MASUD et al., 2010). Muitos algoritmos da literatura assumem que o número de grupos verdadeiros é fixo durante o tempo, o que pode não se verificar no ambiente de aprendizado *online* (MASUD et al., 2011). Muitos destes algoritmos só são capazes de determinar a existência de novos grupos quando, manualmente, o usuário identifica e rotula as instâncias com este novo valor de grupo.

Ao contrário da mudança de conceito, a evolução de conceito é a identificação de novas classes que o indutor não conhecia (MARKOU; SINGH, 2003). Isso ocorre quando classes (ou grupos verdadeiros, no contexto de agrupamento) **aparecem** ou **desaparecem** com o tempo, também alterando o conceito a ser aprendido. Por exemplo, em um sistema de detecção de intrusão em redes computacionais, se cada tipo de ataque é um grupo, uma evolução de conceito acontece quando um novo tipo de ataque ocorre na rede sendo desconhecido até então.

Existem diversos problemas relativos a detecção de novas classes. Esses problemas

podem ser definidos com os princípios apresentados a seguir (MARKOU; SINGH, 2003):

- O *princípio da robustez* afirma que um método de detecção de evolução de conceito deve ser capaz de possuir resultados mesmo quando o conjunto de dados possui poucas instâncias de classes novas (ou grupos verdadeiros novos) e muitas de classes (ou grupos verdadeiros) já conhecidas (desbalanceamento). Em outras palavras, caso um sistema classificador possua taxas de acerto baixas, deve-se distinguir entre a possibilidade do modelo não refletir o conceito a ser aprendido e das instâncias mal classificadas pertencerem a uma nova classe ou grupo verdadeiro ainda não cobertos pelo modelo.

- O *princípio da escala* dos dados define que um algoritmo de agrupamento, quando executado com dados com e sem normalização, devem apresentar os mesmos resultados.

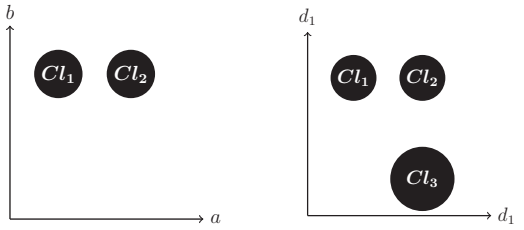
- O *princípio da minimização de parâmetros* afirma que o método deve possuir a menor quantidade possível de parâmetros definidos pelo usuário, e os que ainda assim existirem, devem possuir a menor influência nos resultados obtidos. Ainda, ressalta-se a possibilidade de fazer com que parâmetros sejam adaptativos, ou seja, variem seu valor de acordo com o tempo, uma vez que valores diferentes para os diversos parâmetros podem gerar diferentes resultados de acuidade em diferentes momentos do fluxo de dados.

- O *princípio da generalização* promove que o algoritmo de agrupamento deve ser capaz de atualizar as suas generalizações com novos objetos de dados sem confundí-los com sementes de novos grupos.

- O *princípio da independência* se refere à independência do algoritmo em relação ao número de atributos d e de grupos verdadeiros l do problema, reafirmando que deve obter performance razoável em contextos de conjuntos de dados não balanceados, de poucas instâncias e/ou ruidosos.

- O *princípio da adaptabilidade* afirma que um algoritmo deve ser capaz de detectar evoluções e mudanças de conceito sem intervenção de usuários.

- O *princípio da complexidade computacional* discorre sobre o fato destes algoritmos serem *online*, logo, devem possuir complexidades de tempo e memória baixas para apresentarem resultados em tempo real.



(a) Disposição inicial dos grupos verdadeiros.

(b) Surgimento do grupo verdadeiro Cl_3 em um instante t_i.

Figura 2.7: Exemplo de evolução de conceito com o surgimento do grupo Cl_3.

A Figura 2.7 apresenta um exemplo de surgimento de um novo grupo verdadeiro em um problema bidimensional onde Cl_1 e Cl_2 são grupos cujas instâncias já foram apresentadas ao algoritmo indutor e Cl_3 é um novo grupo emergente a partir de um tempo arbitrário t_i.

2.3 Introdução ao Agrupamento *Online*

Essencialmente, o problema de agrupamento pode ser descrito como a descoberta de um número finito de grupos (*clusters*) que descrevam um conjunto de dados (instâncias). Logo, um algoritmo de agrupamento deve abstrair um conjunto de grupos $\mathscr{K}$ de tamanho K, sendo $\mathscr{K} = \{k_1, k_2, \ldots, k_K\}$ independente do conjunto de grupos verdadeiros $\mathcal{CL}$. Contudo, espera-se que o conjunto $\mathscr{K}$ represente com maior fidelidade possível os grupos verdadeiros definidos em $\mathcal{CL}$ e vice-versa, ou seja, idealmente, $\forall k_i \ \exists! Cl_j, \ k_i = Cl_j \wedge \forall Cl_i \ \exists! k_j, \ Cl_i = k_j$.

O desenvolvimento de algoritmos eficientes e eficazes para fluxos de dados é um problema atual de pesquisa. Particularmente, dentro do problema de agrupamento, algoritmos devem ser capazes de atender todos os requisitos previamente citados, assim como (i) necessitam de um processo capaz de agrupar objetos levando em conta restrições de tempo e memória, (ii) apresentar resultados de maneira rápida ao tratar novos dados de maneira incremental, (iii) utilizar uma estrutura de dados que seja compacta e que não cresça de acordo com a chegada de novas instâncias (nem mesmo crescimentos lineares são tolerados) e (iv) detectar e eliminar dados ruidosos e *outliers* sem prejudicar a detecção de novos grupos. Muitos destes requisitos são atendidos parcialmente pelos algoritmos da literatura, contudo, ressalta-se a preocupação com as limitações de **espaço e memória**, **apresentação de resultados de maneira rápida** e a **utilização de estruturas compactas e facilmente incrementáveis** (SILVA et al., 2013). Contudo, outra vertente da pesquisa foca em requisitos como (CAO et al., 2006): não realizar conjecturas acerca do

número de grupos, uma vez que ele é desconhecido *a priori* e pode mudar com o tempo; e ser capaz de encontrar grupos não hiper-esféricos, uma vez que a maioria das distribuições reais não é regida por uma Gaussiana.

Para desenvolver um algoritmo de agrupamento para fluxos contínuos de dados, deve-se ter em mente que os dados são apresentados de maneira contínua (incremental) ao algoritmo e não existe controle sobre a ordem de chegada destes dados. O fluxo de dados é potencialmente infinito, logo, armazenar todos os dados em memória primária (ou até mesmo secundária) seria um processo custoso tanto em termos de memória quanto de tempo de processamento. Dessa forma, espera-se que o algoritmo descarte os dados logo após seu processamento (*single pass processing*). Contudo, na prática, é possível armazenar uma quantidade limitada de dados, que deve ser descartada de acordo com algum mecanismo de esquecimento (WIDMER; KUBAT, 1996). Para resolver este problema, pesquisadores desenvolvem estruturas de dados não apenas compactas, mas que sejam capazes de crescer de acordo com a chegada de novos dados. Ainda, estas representações de dados não podem ter sua complexidade associada ao número de objetos processados pois nem mesmo crescimentos lineares de uso de memória são aceitáveis (SILVA et al., 2013).

2.4 Técnicas para Agrupamento *Online*

Diversos algoritmos para Agrupamento *Online* foram desenvolvidos nos últimos anos. Neste capítulo alguns dos principais algoritmos são apresentados: ClusTree (KRANEN et al., 2011), CluStream (AGGARWAL et al., 2003) e DenStream (CAO et al., 2006). Estes algoritmos foram escolhidos a partir de suas características de não necessitar de antemão do número de grupos a serem encontrados, tratarem dados ruidosos e apresentarem bons resultados na literatura (AMINI; WAH, 2014; AGGARWAL et al., 2003; CAO et al., 2006; KRANEN et al., 2011; PEREIRA; MELLO, 2011).

Contudo, antes de apresentar os algoritmos, deve-se introduzir os conceitos que os abrangem: as estruturas de dados para sumarização estatística, os modelos de janelas, os mecanismos de detecção de dados ruidosos e a divisão do algoritmo nas etapas *online* e *offline*.

Algoritmos de Agrupamento *Online* são divididos em duas etapas: etapa de abstração de dados (também conhecida como etapa *online*) e etapa de agrupamento (etapa *offline*). A Figura 2.8 apresenta o *framework* genérico de funcionamento para algoritmos de agrupamento para fluxos contínuos de dados onde $\mathcal{H}_a$ é o Horizonte de Avaliação. Basicamente, o valor de $\mathcal{H}_a$ define o tamanho de uma janela estática de avaliação. Dentro

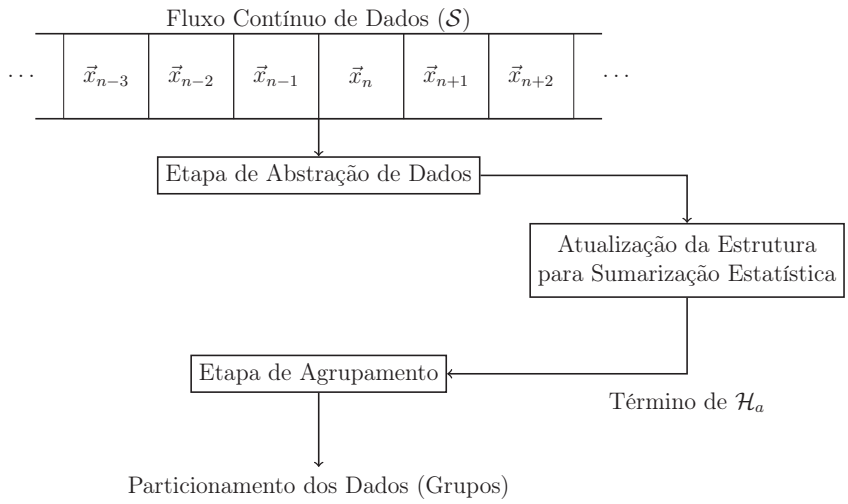


Figura 2.8: *Framework* genérico para agrupamento de fluxos contínuos de dados. Adaptado de (SILVA et al., 2013).

desta janela, instâncias são obtidas de $\mathcal{S}$ e armazenadas em um subconjunto $\mathscr{N}$, onde $|\mathscr{N}| = \mathcal{H}_a$, permitindo então que métricas de avaliação de agrupamentos sejam aplicadas em $\mathscr{N}$ (KREMER et al., 2011). A avaliação de algoritmos de Agrupamento é discutida na Seção 2.6.

A etapa *online* realiza uma sumarização dos dados com o auxílio de estruturas de dados específicas desenvolvidas para lidar com restrições de espaço e tempo. Estas estruturas tem como objetivo armazenar as principais características dos dados vindos do fluxo de dados, contudo, sem ter que armazenar todos os dados em si, o que violaria a restrição de memória e consequentemente, a de tempo. As principais estruturas são: *feature vectors*, *coreset trees*, *data grids* e *prototype arrays* (SILVA et al., 2013).

Ainda para sumarizar os dados vindos do fluxo e para dar maior importância aos dados mais recentes, uma abordagem comum é definir uma janela de tempo que cubra apenas estes dados. Dentro das diversas modelagens para definição de janelas, destaca-se os modelos de janela deslizante, *damped* e *landmark* (SILVA et al., 2013).

Ainda na etapa de abstração de dados, os algoritmos devem incorporar mecanismos de detecção de dados ruidosos, distinguindo mudanças e evoluções de conceito. Há também algoritmos que realizam a detecção de dados ruidosos apenas na etapa *offline*, se tornando uma questão dependente de cada algoritmo.

Já na etapa *offline*, algoritmos de agrupamento obtém partições dos dados baseando-se nas estruturas de sumário estatístico e outros parâmetros fornecidos pelo usuário, e.g. número de grupos a serem encontrados e horizonte $\mathcal{H}$. Como estes algoritmos não lidam com a quantidade massiva de dados vindas do fluxo de dados, mas somente com resumos estatísticos, tais algoritmos acabam sendo eficientes. Logo, algoritmos para formato *batch*

Algoritmo	Estrutura de Dados	Modelo de Janela	Método de Detecção de Dados Ruidosos	Algoritmo Base	Formato dos Grupos	Referência
Cell Trees	*Coreset Tree*	*Damped*	–	Em Aberto	Arbitrário	(PARK; LEE, 2007)
CluStream	*Feature Vector*	*Landmark*	Estatístico	*k-means* ou DBSCAN[1]	Arbitrário	(AGGARWAL et al., 2003)
ClusTree	*Feature Vector*	*Damped*	Densidade	*k-means* ou DBSCAN	Arbitrário	(KRANEN et al., 2011)
DenStream	*Feature Vector*	*Damped*	Densidade	DBSCAN	Arbitrário	(CAO et al., 2006)
HAStream	*Feature Vector*	*Damped*	Densidade	Hierárquico Divisivo	Arbitrário	(HASSANI; SPAUS; SEIDL, 2014)
LiarTree	*Feature Vector*	*Damped*	Densidade	DBSCAN	Arbitrário	(HASSANI; KRANEN; SEIDL, 2011)
PreDeConStream	*Feature Vector*	*Damped*	Densidade	DBSCAN	Arbitrário	(HASSANI et al., 2012)
StreamKM++	*Coreset Tree*	*Landmark*	–	*k-means*	Hiper-esférico	(ACKERMANN et al., 2012)
PL-Stream	*Feature Vector*	*Damped*	–	Em Aberto	Arbitrário	(JIANG; YU; WANG, 2011)

Tabela 2.4: Sumário dos principais algoritmos de agrupamento para fluxos contínuos de dados.

como *k-means* (LLOYD, 1982) e DBSCAN (ESTER et al., 1996) podem ser utilizados e afetam diretamente o formato dos grupos a serem encontrados. A Tabela 2.4 detalha os principais algoritmos para agrupamento em fluxos contínuos de dados, focando nas estruturas utilizadas, tipos de janelas, métodos de detecção de dados ruidosos, algoritmo base e o formato dos grupos encontrados. Variações destes algoritmos foram omitidas por apresentarem relaxamentos ou restrições específicos para alguns tipos de fluxos de dados ou de domínio dos dados. Estas variações e suas respectivas aplicações são discutidas em (AMINI; WAH, 2014; SILVA et al., 2013).

2.4.1 Estruturas de Dados

O desenvolvimento de estruturas de dados para armazenamento de sumários estatísticos é um passo crucial na definição de qualquer algoritmo de Agrupamento *Online*, especialmente pela restrição de memória. Considerando que o fluxo de dados ($\mathcal{S}$) inteiro não pode ser armazenado em memória, estruturas especiais foram desenvolvidas para su-

[1]Originalmente, o algoritmo CluStream utiliza apenas *k-means* em sua etapa *offline*, contudo, o algoritmo DBSCAN foi utilizado e discutido em (PEREIRA; MELLO, 2011).

marizar o fluxo de maneira incremental. Dentro destas estruturas, são encontrados na literatura os *feature vectors*, *prototype arrays*, *coreset trees* e *grids*. Nesta seção aborda-se apenas o *feature vector*, devido a sua presença nos algoritmos utilizados para avaliação e comparativo empírico dos algoritmos propostos.

O primeiro uso de vetores de características (*Feature Vectors*) para sumarizar grandes volumes de dados ocorreu com a introdução do algoritmo BIRCH (ZHANG; RAMAKRISHNAN; LIVNY, 1996). Um *feature vector* é uma tripla $CF = \langle LS, SS, N \rangle$, onde LS representa a soma linear dos objetos sumarizados, SS a soma do quadrado destes objetos e N a quantidade de objetos sumarizados. As estruturas LS e SS são vetores d-dimensionais, onde d é a dimensionalidade das instâncias $\vec{x}_i$ obtidas de $\mathcal{S}$. Baseando-se nestas três componentes, é possível calcular três medidas essenciais de um CF: a sua média (centro), raio e diâmetro. Deste modo, CFs são estruturas capazes de sumarizar instâncias formadas por atributos contínuos intervalados e representar apenas grupos hiper-esféricos.

As Equações 2.3, 2.4 e 2.5 apresentam o cálculo da média $\mu(\cdot)$, raio $r(\cdot)$ e diâmetro $diam(\cdot)$ de um *feature vector* arbitrário CF_i, respectivamente.

$$\mu(CF_i) = \frac{LS_i}{N_i} \tag{2.3}$$

$$r(CF_i) = \sqrt{\frac{SS_i}{N_i} - \left(\frac{LS_i}{N_i}\right)^2} \tag{2.4}$$

$$diam(CF_i) = \sqrt{\frac{2N_i \times SS_i - 2 \times (LS_i)^2}{N_i \times (N_i - 1)}} \tag{2.5}$$

Contudo, as principais características relativas a um *feature vector* são suas importantes propriedades de incrementalidade e aditividade. A propriedade de incrementalidade afirma que um objeto de dados $\vec{x}_j$ pode ser facilmente inserido em um CF_i ao atualizar os sumários estatísticos de CF_i de acordo com as Equações 2.6, 2.7 e 2.8.

$$LS_i \leftarrow LS_i + \vec{x}_j \tag{2.6}$$

$$SS_i \leftarrow SS_i + (\vec{x}_j)^2 \tag{2.7}$$

$$N_i \leftarrow N_i + 1 \tag{2.8}$$

A propriedade de aditividade determina que dois vetores CF_i e CF_j podem ser fundidos em um vetor CF_l ao simplesmente somar seus componentes, como apresentado nas Equações 2.9, 2.10 e 2.11.

$$LS_l \leftarrow LS_i + LS_j \tag{2.9}$$

$$SS_l \leftarrow SS_i + SS_j \tag{2.10}$$

$$N_l \leftarrow N_i + N_j \tag{2.11}$$

2.4.2 Modelos de Janelas

Na maioria dos cenários de fluxos de dados, informações mais recentes tendem a refletir a emergência de novos conceitos ou mudanças na distribuição dos dados. Sistemas que dão importância igual para dados mais antigos quanto para os novos não são capazes de capturar as características evolutivas do fluxo de dados (CHEN; TU, 2007). Para solucionar este tipo de problema, modelos de janelas móveis foram propostos. Existem três tipos de modelos de janelas na literatura: janela deslizante, janela *damped* e janela *landmark* (SILVA et al., 2013).

Todavia, de maneira genérica, o problema em utilizar janelas de tamanho fixo é determinar o seu tamanho ideal. Ao utilizar janelas pequenas, garante-se que o algoritmo seja capaz de detectar mudanças e evoluções de conceito de maneira eficaz, contudo, em fases estacionárias do fluxo de dados, é possível que sua acurácia seja afetada por realizar adaptações a dados apenas ruidosos. Janelas grandes, por sua vez, são desejáveis em fases estacionárias mas podem não responder rapidamente às mudanças e evoluções de conceito.

No modelo de janela deslizante, apenas a informação mais recente obtida do fluxo de dados é armazenada em uma estrutura de tamanho fixo ou dinâmico $\mathscr{W}$. Esta estrutura normalmente possui política de acesso FIFO (*first in, first out*, "primeiro a entrar é o primeiro a sair") considerando apenas as instâncias de um momento atual até certo período no passado. A Figura 2.9 apresenta um exemplo de janela deslizante com tamanho $\mathscr{W} = 4$.

Diferentemente das janelas deslizantes, o modelo *damped* associa pesos aos objetos de dados obtidos do fluxo de dados, onde objetos mais recentes possuem pesos maiores e estes pesos decaem com o tempo (JIANG; GRUENWALD, 2006). A Figura 2.10 apresenta um exemplo de janela *damped* onde o peso dos objetos decai exponencialmente, onde preto representa um maior peso, por ser um objeto recém obtido de $\mathcal{S}$ e branco representa

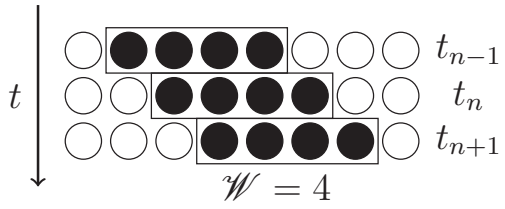


Figura 2.9: Modelo de janela deslizante. Adaptado de (SILVA et al., 2013).

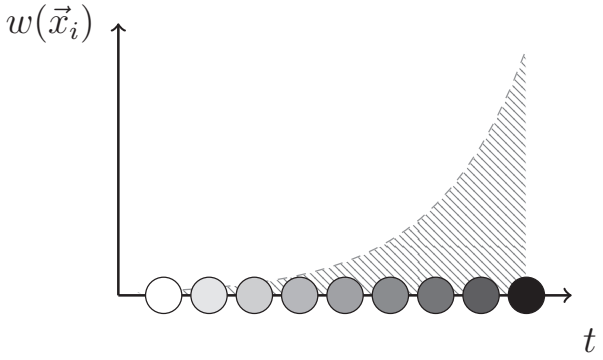


Figura 2.10: Modelo de janela *damped*. Adaptado de (SILVA et al., 2013).

um objeto muito antigo. Este modelo é bastante utilizado em algoritmos de agrupamento baseados em densidade. Estes algoritmos usualmente assumem uma função de decaimento exponencial para associar pesos aos objetos obtidos de $\mathcal{S}$. O peso de cada objeto pode então ser calculado a partir da Equação 2.12, onde $\Xi > 0$ é o parâmetro que determina a taxa de decaimento, t_{atual} é o instante atual e t_i é o instante de chegada de uma instância $\vec{x}_i$. Quão maior for o valor de Ξ, menor a importância dos objetos mais antigos em comparação aos mais recentes.

$$w(\vec{x}_i) = \Xi^{t_{\text{atual}} - t_i} \tag{2.12}$$

Finalmente, processar um fluxo de dados utilizando janelas do modelo *landmark* requer tratar partições disjuntas de instâncias (*chunks*) que são separadas por objetos de dados relevantes, denominados *landmarks*. *Landmarks* podem ser definidos em termos de tempo (e.g. diariamente ou semanalmente) ou em termos de número de instâncias observadas após o último *landmark*. Normalmente, os algoritmos possuem um parâmetro de Horizonte ($\mathcal{H}$) que define o tamanho destes *chunks*. Logo, todos os objetos apresentados após um *landmark* são sumarizados em uma janela de dados recentes. Quando um novo *landmark* é alcançado, todos os objetos da última janela (*chunk*) são descartados, dando espaço para os novos dados. A Figura 2.11 exemplifica o modelo de janela *landmark*, onde três *chunks* de tamanho $\mathcal{H} = 3$ são apresentados.

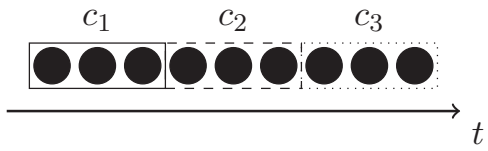


Figura 2.11: Modelo de janela *landmark* com $\mathcal{H} = 3$. Adaptado de (SILVA et al., 2013).

2.4.3 Mecanismos de Detecção de Dados Ruidosos e *Outliers*

Além de satisfazer os requisitos de serem incrementais e rápidos, os algoritmos para Agrupamento *Online* devem ser capazes de detectar dados ruidosos e *outliers*.

Dados ruidosos são objetos que apresentam valores que se desviam do comportamento geral do modelo, e são derivados de problemas na coleta, transmissão ou armazenamento. *Outliers*, por sua vez, são objetos de dados que também desviam do comportamento geral do sistema, mas são dados verdadeiros (HAN; KAMBER; PEI, 2011), as ditas exceções.

Vários dos algoritmos de aprendizagem não supervisionada descritos na literatura possuem seus próprios métodos para detectar dados ruidosos e *outliers*, que são divididos em modelos estatísticos, baseados em proximidade espacial e de densidade (ACHTERT et al., 2010).

Métodos de detecção de dados ruidosos e *outliers* baseados em estatística comumente assumem que a distribuição dos dados é regida por uma Gaussiana, logo, valores que se encontram nas caudas da distribuição são descartados de acordo com um nível de significância α. Por outro lado, nos métodos baseados em densidade, assume-se que instâncias sejam dados ruidosos ou *outliers* quando estas são formam sub-regiões densas, ou seja, a distância de uma instância $\vec{x}_i$ com a outras k instâncias mais próximas excede um limiar pré-definido. O método mais conhecido para detecção de dados ruidosos e *outliers* é o *Local Outlier Factor* (BREUNIG et al., 2000).

2.4.4 Etapas dos Algoritmos de Agrupamento

Todas as seções anteriores apresentaram elementos da etapa *online* dos algoritmos de agrupamento para fluxos de dados.

Como discutido anteriormente, a etapa *online* realiza sumários estatísticos dos dados obtidos do fluxo de dados utilizando-se de estruturas de dados específicas para lidar com restrições de espaço e tempo. Ainda, para favorecer instâncias mais recentes, modelos de janelas foram propostos e abordam este problema, mas ainda de maneira parcial. Finalmente, os algoritmos devem possuir métodos para detectar dados ruidosos,

distinguindo mudanças e evoluções de conceito. Existem discussões na literatura sobre a etapa de detecção de dados ruidosos ser realizada apenas na etapa *offline*, logo, é uma questão em aberto e que depende da abordagem de cada algoritmo proposto (SILVA et al., 2013).

A etapa *offline*, por sua vez, se baseia nas partições de dados determinadas na etapa *online* e em parâmetros fornecidos pelo usuário (e.g. número de *clusters* e horizonte). Normalmente, na etapa *offline*, algoritmos de agrupamento de formato *batch* são utilizados, e.g. *k-means* (LLOYD, 1982) e DBSCAN (ESTER et al., 1996). Como estes algoritmos não lidam com a quantidade massiva de dados, mas apenas com resumos estatísticos, eles acabam sendo eficientes.

A utilização do algoritmo DBSCAN na etapa *offline* é bastante comum, uma vez que este algoritmo permite encontrar grupos não hiper-esféricos, mesmo a partir de sumários estatísticos hiper-esféricos como *feature vectors* (AMINI; WAH, 2014). Por outro lado, a utilização do algoritmo DBSCAN implica na definição de um número maior de parâmetros.

2.5 Algoritmos de Agrupamento *Online*

Nesta seção são apresentados os principais algoritmos para a tarefa de agrupamento em fluxos de dados. Esta seção não clama pela completude da cobertura dos algoritmos, pois são discutidos apenas aqueles que **não** necessitam de parâmetros relativos a quantidade de grupos a serem encontrados. Esta propriedade é fortemente desejada, pois, devido as evoluções de conceito, o número de grupos em fluxo de dados pode variar.

2.5.1 CluStream

O algoritmo CluStream contrariou os algoritmos desenvolvidos até o momento de sua criação, onde grupos eram computados durante o fluxo de dados inteiro (AGGARWAL et al., 2003). O algoritmo CluStream é dividido em duas etapas: *online* e *offline*, conforme apresentado anteriormente.

Na etapa *online*, o algoritmo CluStream é dependente de um único parâmetro fornecido pelo usuário $\mathcal{H}$ e tem como objetivo manter estatísticas em diferentes níveis de granularidade para aspectos temporais e espaciais dos dados obtidos do fluxo de dados. O algoritmo assume que um total de q *feature vectors* são mantidos em todos os momentos em um conjunto $\mathcal{M} = \{CF_1, CF_2, \ldots, CF_q\}$. O valor de q é definido a partir da quantidade de memória disponível para o processo. Logo, valores típicos de q são maiores que o

número de grupos a serem encontrados mas ainda assim significantemente menores que o número de instâncias a serem processadas. Estes *feature vectors* representam o estado atual do sumário estatístico que evolui de acordo com a chegada de novas instâncias.

O algoritmo inicia armazenando uma quantidade de instâncias $\mathcal{N}$ para definir os primeiros q *feature vectors* utilizando-se do algoritmo *k-means*. Logo, o valor de $\mathcal{N}$ é definido a partir da complexidade computacional do algoritmo *k-means*. Tendo este conjunto inicial de *feature vectors*, novas instâncias $\vec{x}_i$ são obtidas de $\mathcal{S}$ e devem ser inseridas em *feature vectors* já definidos ou iniciar um novo. Primeiramente, calcula-se a distância Euclidiana entre $\vec{x}_i$ e os centros dos *feature vectors* existentes em $\mathcal{M}$, encontrando o mais próximo: CF_j. Caso $\vec{x}_i$ esteja a uma distância menor do centro que o raio definido em CF_j, $\vec{x}_i$ é incorporado em CF_j. Caso contrário, deve-se distinguir entre as possibilidades de que $\vec{x}_i$ seja um dado ruidoso/*outlier* ou ainda, seja o início de um novo grupo. Em ambos os casos, $\vec{x}_i$ inicia um novo *feature vector* que é tratado apenas na etapa *offline*. Caso o número de *feature vectors* tenha excedido q, dois destes, caso suficientemente próximos tomando por base um limiar δ, são combinados de acordo com a propriedade aditiva. Esta etapa se repete até que o número de instâncias obtidas de $\mathcal{S}$ satisfaça o tamanho de $\mathcal{H}$. Ao final de um *chunk*, o sumário estatístico é armazenado em disco e novos *feature vectors* são formados com as $\mathcal{N}$ próximas instâncias obtidas de $\mathcal{S}$.

A etapa *offline* realiza uma modificação do algoritmo *k-means* para definir os grupos a partir dos *feature vectors* em $\mathcal{M}$. Ao contrário do algoritmo *k-means* convencional, na etapa de inicialização, as sementes não são geradas aleatoriamente, mas são amostradas com probabilidade proporcional ao número de pontos em cada *feature vector*. A semente é então o centróide do *feature vector* escolhido garantindo assim convergência mais rápida do algoritmo *k-means*.

Existem duas implementações do algoritmo CluStream, uma onde o usuário define o número de grupos a serem encontrados K e outra onde este número é encontrado automaticamente, se baseando em uma sub-janela onde as instâncias são apresentadas ao algoritmo juntamente de seus rótulos reais o que pode não ser possível em ambientes reais.

2.5.2 ClusTree

O algoritmo ClusTree utiliza uma árvore R (*R-Tree*) (GUTTMAN, 1984) para realizar indexação de *feature vectors*, diminuindo assim o número de comparações realizadas na chegada de cada instância (KRANEN et al., 2011). Árvores R são estruturas de dados similares as árvores B (BAYER; MCCREIGHT, 1972), contudo, são utilizadas para

métodos de acesso no espaço, indexando informações multi-dimensionais, neste caso, os *feature vectors*. A hipótese do algoritmo ClusTree é a criação de uma hierarquia de *feature vectors* (CFs) em diversos níveis de granularidade. Dependendo do tempo disponível, o algoritmo realiza uma busca na árvore até encontrar o *feature vector* mais similar a $\vec{x}_i$, o último objeto obtido de $\mathcal{S}$. Caso o *feature vector* escolhido seja suficientemente similar baseando-se em parâmetros definidos pelo usuário, ele será atualizado incrementalmente de acordo com a propriedade descrita anteriormente. Caso contrário, um novo *feature vector* será criado e adicionado na árvore.

Uma ClusTree pode ser formalizada como uma árvore *multiway* de indexação com capacidade entre m e M de entradas em cada nó e entre $\mathbb{L}$ e L para os nós folha. A única exceção é o nó raiz, por poder possuir apenas uma entrada. Cada entrada dos nós internos de uma ClusTree armazena: um CF_{objs} dos objetos sumarizados por este nó, um CF_{buffer} dos objetos em *buffer* e um ponteiro para o nó filho. Uma entrada em um nó folha, por sua vez, armazena um CF dos objetos sumarizados por tal nó. Ainda, um percurso da raiz até qualquer nó folha possuirá sempre o mesmo comprimento, ou seja, a árvore é balanceada.

A árvore é criada e atualizada como uma árvore R convencional. Para a inserção, um percurso na árvore é realizado baseando-se nas distâncias Euclidianas entre $\vec{x}_i$ e os centros dos *feature vectors*. O tipo de percurso é variável e deve ser escolhido de acordo com os recursos disponíveis. Diversas abordagens de percurso e suas respectivas implicações são discutidas em (KRANEN et al., 2011). Caso o tempo disponível para a inserção de uma nova instância não seja o suficiente para encontrar um nó folha, $\vec{x}_i$ instancia um novo *feature vector* que é incorporado em CF_{buffer} do nó em que o percurso parou.

Caso em algum outro percurso, haja tempo de sobra, as instâncias presentes nos *buffers* da árvore são realocados em nós de sub-níveis mais baixos. Idealmente, quando o tempo disponível é grande, o procedimento de realocação é realizado até que todos os CFs presentes em *buffers* sejam associados apenas aos nós folhas. Quando um CF_i é adicionado ao CF_j participante de um nó folha, todas as entradas dos nós de níveis superiores da árvore, até a raiz, são atualizados utilizando a propriedade aditiva com CF_i.

Para prover maior importância às instâncias mais recentes, o algoritmo ClusTree utiliza o modelo de janela *damped*. Como os nós da árvore R armazenam apenas *feature vectors*, as suas componentes N, LS e SS são atualizadas de acordo com as Equações 2.13, 2.14 e 2.15, respectivamente, onde $w(\Delta t) = \Xi^{-\lambda \Delta t}$.

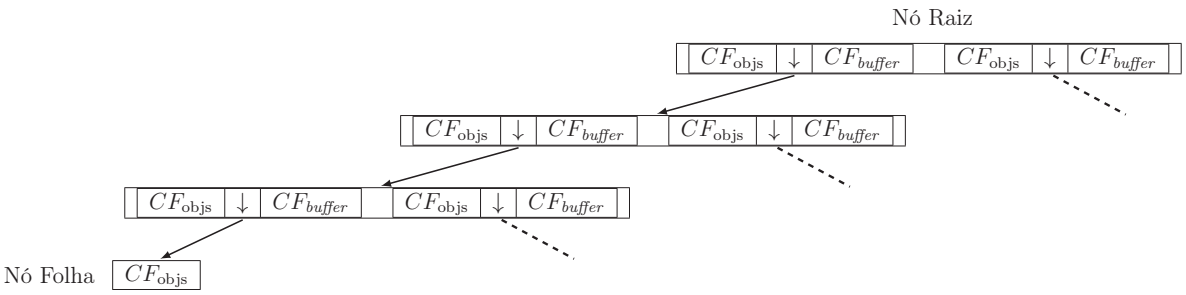


Figura 2.12: Exemplo de uma árvore R do algoritmo ClusTree. Adaptado de (KRANEN et al., 2011).

$$N_{t+N} = \sum_{i=1} w(t - t_i) \tag{2.13}$$

$$LS_{t+N} = \sum_{i=1} w(t - t_i) \times \vec{x}_i \tag{2.14}$$

$$SS_{t+N} = \sum_{i=1} w(t - t_i) \times (\vec{x}_i)^2 \tag{2.15}$$

Em (KRANEN et al., 2011), autores discutem sobre os possíveis valores do parâmetro M e sobre os métodos de percurso na árvore. Contudo, mesmo sendo valores e heurísticas dependentes do domínio da aplicação ou do tempo disponível, $M = 3$ e o método de busca em profundidade com arrefecimento simulado (CERNY, 1985) apresentaram bons resultados na maioria dos problemas testados. A Figura 2.12 apresenta um exemplo de árvore R de uma ClusTree com altura 3, $M = 2$ e $L = 1$. A Figura 2.12 omite vários nós internos e folhas com o intuito de prezar pela visibilidade.

Finalmente, na etapa *offline*, algoritmos como *k-means* e DBSCAN podem ser utilizados para encontrar grupos utilizando-se das médias dos *feature vectors* como pontos mais representativos.

2.5.3 DenStream

O algoritmo DenStream foi desenvolvido com o intuito de atender requisitos – muitas vezes – não tratados pelos demais algoritmos: não deve existir suposições acerca do número de grupos, principalmente pelo problema de evolução de conceito; deve existir um método de descoberta de grupos de formato arbitrário, pois a geração de dados pode ser irregular (não Gaussiana); e deve-se detectar e eliminar dados ruidosos e *outliers* (CAO et al., 2006).

Em ambientes estáticos, grupos com formatos arbitrários são comumente repre-

sentados por todos os pontos que os constituem. Logo, uma abordagem ingênua seria armazenar todos os pontos na memória. Evidentemente, pelas restrições do ambiente *online*, esse armazenamento é impossível e a execução dos algoritmos seria muito custosa computacionalmente.

No algoritmo algoritmo DBSCAN, quando uma requisição de visualização de grupos é realizada, o resultado é um grupo de objetos *core* ponderados divididos em grupos, com a garantia que a união da ϵ-vizinhança de $\mathscr{K}$ cubra as áreas densas do espaço de atributos. Um objeto *core* é um objeto, cuja ϵ-vizinhança possua ao menos ψ vizinhos e uma área densa é a união das ϵ-vizinhanças de todos os objetos *core*.

Contudo, não é realístico apresentar tal resultado em um ambiente *online* onde a memória é limitada. Logo, o algoritmo DenStream utiliza a noção de *core micro-cluster*. Um *core micro-cluster* é denotado $CMC(w, c, r, t_c, t_u)$ para um grupo de instâncias próximas $\vec{x}_i, \vec{x}_{i+1}, \ldots, \vec{x}_n$ onde w é seu peso, c o seu centro, r seu raio, t_c seu instante de criação e t_u o momento de sua última atualização (inserção ou adição). As Equações 2.16, 2.17 e 2.18 apresentam, respectivamente, o cálculo do peso w, centro c e raio r, sendo que $w \geq \psi$, $r \leq \epsilon$, $d(\cdot, \cdot)$ é uma distância Euclidiana e $f(\cdot)$ é uma função de decaimento exponencial na forma $2^{-\lambda \Delta t}$.

$$w(CMC) = \sum_{k=1}^{n} f(t_u - k) \tag{2.16}$$

$$c(CMC) = \frac{\sum_{k=1}^{n} f(t_u - k) \times \vec{x}_k}{w} \tag{2.17}$$

$$r(CMC) = \frac{\sum_{k=1}^{n} f(t_u - k) \times d(\vec{x}_k, c)}{w} \tag{2.18}$$

Ressalta-se que o peso de um CMC deve ser maior ou igual a ψ e que o raio deve ser menor ou igual a ϵ. Portanto, um CMC, por definição, é um *micro-cluster* hiper-esférico "denso".

Devido a restrição no tamanho do raio de cada CMC, o número de *core micro-clusters* N_c, é muito maior que o número de grupos reais existentes no fluxo de dados (ou até mesmo no em uma sub-partição avaliada). Por outro lado, N_c ainda é inferior ao número de instâncias em $\mathcal{S}$. Como cada instância obtida de $\mathcal{S}$ é associada a apenas um CMC, tem-se ainda que $N_c \leq \frac{W}{\Psi}$, onde $\Psi = \sum_{CMC_i} \sum_{t=0}^{t_c} 2^{\lambda t}$.

Em fluxos de dados evolucionários, o papel de grupos e dados ruidosos pode ser permutado, assim como os *micro-clusters* são formados de maneira incremental e durante evoluções e mudanças de conceito. Logo, dois tipos especiais de *micro-clusters* são

apresentados: potenciais *micro-clusters* e *outlier micro-clusters*.

Um potencial *micro-cluster* PMC possui definição bastante similar a um CMC comum, com a diferença da restrição do peso w, onde $w \geq \beta\psi$ e $0 \leq \beta \leq 1$ é um parâmetro que define o limiar para dados ruidosos relativo aos CMCs.

Por outro lado, um *outlier micro-cluster* OMC_i possui definição análoga a de um potencial *micro-cluster*, contudo, onde $w < \beta\psi$.

Deste modo, a etapa *online* do algoritmo DenStream tem como objetivo manter um grupo de potenciais *micro-clusters* e *outlier micro-clusters*. Os *outlier micro-clusters* são armazenados em um espaço separado de memória, assumindo que a maioria das instâncias $\vec{x}_i$ obtidas de $\mathcal{S}$ pertencerão a alguns dos potenciais *micro-clusters*.

Na chegada de uma instância $\vec{x}_i$, o algoritmo DenStream tenta agregar $\vec{x}_i$ ao potencial *micro-cluster* de centro mais próximo PMC_p. Caso $r(PMC_p + \vec{x}_i) < \epsilon$, o raio de um potencial *micro-cluster* PMC_p fundido com $\vec{x}_i$ seja menor que ϵ, então PMC_p é atualizado com a propriedade incremental apresentada anteriormente.

Caso contrário, DenStream tenta armazenar $\vec{x}_i$ no *outlier micro-cluster* mais próximo OMC_o. Novamente, verifica-se se o raio de OMC_o fundido com $\vec{x}_i$ é menor que ϵ e $\vec{x}_i$ é então adicionado em OMC_o. Caso esta condição seja satisfeita, verifica-se o novo peso w de OMC_o. Caso $w \geq \beta\psi$, significa então que este *outlier micro-cluster* se tornou um potencial *micro-cluster*. Logo, este *outlier micro-cluster* é retirado deste *buffer* $\mathcal{B}$ localizado em memória secundária e é alocado na memória principal juntamente dos demais potenciais *micro-clusters*.

Finalmente, caso a condição da comparação do raio com a distância não seja satisfeita, um novo *outlier micro-cluster* é criado com $\vec{x}_i$, pois esta instância pode ser simplesmente um *outlier* ou vir a evoluir a um novo grupo posteriormente.

Para cada potencial *micro-cluster* PMC_i, caso nenhuma instância seja adicionada a este, seu peso decairá exponencialmente. Como afirmado anteriormente, caso $w(PMC_i) < \beta\psi$, este PMC_i acabou se tornando um *outlier*, logo, não deverá afetar a etapa *offline* e indiretamente, os grupos obtidos. Verificar esta condição após a chegada de cada instância $\vec{x}_i$ pode se tornar um processo custoso, caso o número de potenciais *micro-clusters* seja elevado. Logo, o algoritmo DenStream verifica esta condição periodicamente. A Equação 2.19 apresenta o cálculo da janela de avaliação de pesos T_p do algoritmo DenStream.

$$T_p = \left\lceil \frac{1}{\lambda} \log \left(\frac{\beta\psi}{\beta\psi - 1} \right) \right\rceil \tag{2.19}$$

Ainda, o número de *outlier micro-clusters* tende a crescer indefinidamente de

acordo com a chegada de novas instâncias, principalmente quando muitos dados ruidosos e *outliers* existem. Deste modo, o algoritmo DenStream, ao verificar quais potenciais *micro-clusters* devem ser removidos, verifica também quais *outlier micro-clusters* devem ser removidos. Para evitar a remoção de *outliers micro-clusters*, definiu-se um limite mínimo ξ de peso para *outlier micro-clusters*, prevenindo então a remoção daqueles que possuem chances de se tornarem potenciais *micro-clusters*. A Equação 2.20 apresenta a Equação do limite inferior de peso para *outlier micro-clusters*, onde t_i é o instante atual e t_c o instante de criação do *outlier micro-cluster*.

$$\xi(t_i, t_c) = \frac{2^{t_i - t_c + T_p} - 1}{2^{-\lambda T_p} - 1} \tag{2.20}$$

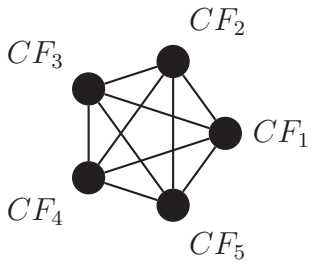
A etapa *offline* do algoritmo DenStream utiliza uma variação do algoritmo DBSCAN para, a partir das regiões densas encontradas na etapa *online*, determinar os grupos existentes baseando-se apenas nos potenciais *micro-clusters* e ignorando os *outlier micro-clusters*.

2.5.4 HAStream

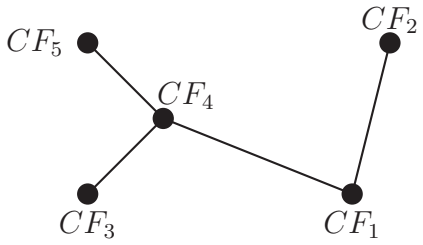
Algoritmos baseados em densidade como o DenStream (CAO et al., 2006) são capazes de encontrar grupos de formato arbitrário, contudo, falham ao encontrar grupos com diferentes densidades por possuírem valores de limiares de densidade fixos. Com o intuito de sobrepujar esta limitação, o algoritmo HAStream (HASSANI; SPAUS; SEIDL, 2014) realiza um agrupamento hierárquico baseado em densidade que, de maneira automática e independente, adapta seus limiares de densidade de acordo com o fluxo de dados fornecido.

Assim como os demais algoritmos discutidos anteriormente, o HAStream também está dividido em duas etapas. Durante a etapa *online*, todas as instâncias obtidas do fluxo de dados são tratadas de acordo com um modelo baseado em *feature vector* como no CluStream, ClusTree ou DenStream.

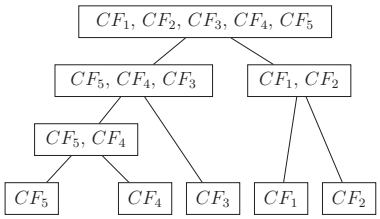
Durante a etapa *offline*, o algoritmo HAStream gera os grupos finais utilizando um agrupamento hierárquico baseado em densidade. Primeiramente, todos os *feature vectors* computados durante a etapa *online* geram um componente fortemente conectado (grafo completo) conforme apresentado na Figura 2.13a. Após a computação deste componente, uma árvore geradora mínima é derivada, utilizando o algoritmo de Prim (PRIM, 1957), conforme apresentado na Figura 2.13b. A partir desta árvore geradora mínima, um dendograma é construído, onde os nós folha contém *feature vectors* individuais e o nó raiz representa um grupo que contém todos os *feature vectors* da árvore geradora mínima,



(a) Computação do grafo completo.



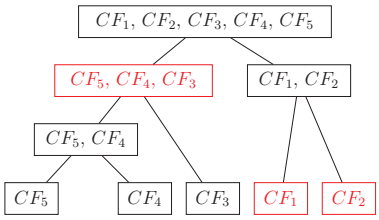
(b) Árvore geradora mínima.



(c) Dendograma de todas as possibilidades de agrupamento hierárquico.

Grupo	Estabilidade de Grupo $CS(\cdot)$
CF_1	0,02
CF_2	0,04
CF_3	0,001
CF_4	0,005
CF_5	0,06
CF_5, CF_4	0,005
CF_1, CF_2	0,001
CF_5, CF_4, CF_3	0,07

(d) Computação das estabilidades de grupos.



(e) Extração dos grupos finais.

Figura 2.13: Passos do algoritmo HAStream.

como apresentado na Figura 2.13c.

Como apresentar todas as possibilidades de agrupamento hierárquico força o avaliador a definir o número correto de grupos a serem encontrados para então determinar o nível do corta no dendograma, o algoritmo HAStream utiliza o conceito de CS – "Estabilidade de Grupo" (*Cluster Stability*).

A estabilidade de grupo CS para um grupo k_i é dada pela Equação 2.21, onde $w(\cdot)$ é o peso de dado *feature vector* e ϵ_{max} e ϵ_{min} são os raios máximos extraídos de cada grupo participante do dendograma. A Figura 2.13d apresenta um exemplo de computação de estabilidades de grupos.

$$CS(k_i) = \sum_{PMC_j}^{k_i} w(PMC_j) \times \left[\frac{1}{\epsilon_{min}(PMC_j, k_i)} - \frac{1}{\epsilon_{max}(k_i)} \right] \tag{2.21}$$

Para obter os grupos finais, o algoritmo HAStream computa o agrupamento final $\mathscr{K}$ na forma de um problema de otimização, onde o objetivo é maximizar as estabilidades dos grupos escolhidos. Assumindo um agrupamento hierárquico $HC = \{k_2, \ldots, k_n\}$, o agrupamento final $\mathscr{K}$ é o conjunto de grupos que maximiza a Equação 2.22, onde $\mathcal{N}_i$ é o conjunto de índices dos grupos que formam um grupo k_i, $s_i \in \{0, 1\}$, $2 \leq i \leq n$ e $\forall j \in \mathcal{N}_i, (s_i = 1) \Leftrightarrow (s_j = 0)$.

$$\max_{s_2,\ldots,s_n} \sum_{i=2}^{n} s_i \times CS(k_i) \tag{2.22}$$

Após o processo de otimização das estabilidades dos grupos, todos os grupos k_i onde $s_i = 1$ formarão o conjunto de grupos finais $\mathscr{K}$. A Figura 2.13e apresenta um exemplo de execução do algoritmo HAStream, onde pode-se ver dois grupos com diferentes densidades (dispostas em níveis diferentes no dendograma).

2.6 A Avaliação de Agrupamentos *Online*

Avaliar se um agrupamento é bom ou não é uma situação controvérsa e problemática. Não existe uma definição universal do que é um agrupamento bom, logo, a avaliação depende exclusivamente do avaliador (BONNER, 1964). Ainda assim, diversos critérios foram desenvolvidos e discutidos na literatura com o intuito de quantificar algumas características de agrupamentos. Estes critérios foram divididos em **internos** e **externos**.

Métricas internas de avaliação medem quão compactos os grupos são ao utilizar medidas de similaridade ou de dissimilaridade. Normalmente estas métricas medem a homogeneidade intra-grupo, a separabilidade inter-grupos ou a combinação destes fatores. Esse tipo de avaliação **não** se vale de informações externas, ou seja, se baseia apenas nos dados fornecidos para a tarefa de agrupamento.

Contrariamente, métricas externas de avaliação conhecem a distribuição real de grupos $\mathcal{CL}$, permitindo então comparar os grupos obtidos $\mathscr{K}$ com os grupos reais. Ressalta-se que em vários problemas, a distribuição real dos grupos $\mathcal{CL}$ é desconhecida, logo, apenas métricas internas são passíveis de utilização.

Dentro do problema do Agrupamento *Online*, métricas de avaliação para algoritmos de formato em lote (*batch*) são comumente utilizadas. Contudo, nenhuma destas métricas possui modelagem matemática que considere problemas inerentes do ambiente incremental, pois não foram elaboradas para lidar com a sobreposição de grupos, existência de dados ruidosos e obtém resultados sub-ótimos mesmo quando avaliam a distribuição

de grupos real $\mathcal{CL}$.

Como discutido anteriormente, não é coerente armazenar o fluxo de dados inteiro em memória por questões de tempo e memória. Ainda, avaliar o agrupamento resultante de um algoritmo usando o fluxo de dados inteiro pode derivar métricas imprecisas, uma vez que certos grupos podem aparecer apenas em determinados intervalos do fluxo.

Logo, com o intuito de avaliar os algoritmos utilizando métricas não propostas especificamente para o problema de Agrupamento *Online*, definiu-se o conceito de Horizonte de Avaliação $\mathcal{H}_a$. O valor de $\mathcal{H}_a$ define o tamanho de uma janela estática de avaliação. Para utilizar as métricas apresentadas a seguir, instâncias são obtidas de $\mathcal{S}$ e armazenadas em um subconjunto estático $\mathscr{N}$. Logo, $\mathscr{N} = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{\mathcal{H}_a}\}$, $\mathscr{N} \subseteq \mathcal{S}$. Deste modo, métricas convencionais para avaliação de algoritmos de agrupamento podem ser realizadas sobre $\mathscr{N}$ (KREMER et al., 2011).

Evidentemente, definir o valor de $\mathcal{H}_a$ é uma tarefa subjetiva que pode beneficiar ou prejudicar a avaliação de um ou mais algoritmos. Tamanhos de $\mathcal{H}_a$ muito pequenos podem apresentar valores de acuidade baixos para algoritmos durante mudanças de conceito. Por outro lado, caso algoritmos possuam boa adaptabilidade a mudanças de conceito, tamanhos de horizonte muito grandes podem não permitir a visualização da perda de acuidade destes.

Esta seção tem como objetivo apresentar as principais métricas de avaliação para a tarefa de agrupamento utilizadas na literatura, apresentar suas deficiências perante o problema de Agrupamento *Online* e comentar uma métrica específica para este problema: o CMM (*Cluster Mapping Measure*) (KREMER et al., 2011).

2.6.1 *Sum of Squared Distances* – SSQ

Um dos mais simples e utilizados critérios de avaliação interna de agrupamentos é o SSQ (*Sum of Squared Distances – Soma do Quadrado das Distâncias*) (HAN; KAMBER; PEI, 2011). O $SSQ(\mathscr{K})$ de um agrupamento $\mathscr{K}$ é calculado de acordo com a Equação 2.23, onde $\vec{\mu}_j$ é um objeto que representa a média de um grupo k_j e $d(\cdot, \cdot)$ é uma função de dissimilaridade entre dois objetos.

$$SSQ(\mathscr{K}) = \sum_{k_j}^{\mathscr{K}} \sum_{\vec{x_i}}^{k_j} d(\vec{x}_i, \vec{\mu}_j)^2 \tag{2.23}$$

2.6.2 Homogeneidade

Com o intuito de satisfazer o critério de homogeneidade, um agrupamento deve associar apenas os objetos pertencentes a um grupo verdadeiro, a um grupo (ROSENBERG; HIRSCHBERG, 2007). Em outras palavras, a distribuição de grupos verdadeiros dentro de um grupo deve ser enviesada em um único grupo verdadeiro, ou seja, a entropia H deve ser nula. A métrica externa de Homogeneidade é então calculada baseando-se em um cálculo de entropia, onde, em seu caso perfeito, $H(\mathcal{CL}|\mathscr{K}) = 0$. Contudo, em uma situação imperfeita, o número deste valor, em *bits*, é dependente do tamanho do conjunto de dados N e da distribuição dos grupos verdadeiros. Logo, ao invés de calcular a entropia convencional, normaliza-se este valor pela máxima redução na entropia gerada pelo agrupamento, ou seja, $H(\mathcal{CL})$. Como $H(\mathcal{CL})$ e $H(\mathcal{CL}|\mathscr{K})$ são iguais e máximos quando o agrupamento não trouxe nenhum ganho de informação, logo tem-se $H(\mathcal{CL}|\mathscr{K}) = 0$ quando cada grupo possui apenas objetos de um mesmo grupo verdadeiro, ou seja, um agrupamento **homogêneo**. Quando $H(\mathcal{CL}) = 0$, onde existe apenas um grupo verdadeiro, define-se a homogeneidade como 1. Para uma solução homogênea, esta normalização, $\frac{H(\mathcal{CL}|\mathscr{K})}{H(\mathcal{CL})}$, será igual a 0. Finalmente, para adotar a convenção de que o valor 1 é desejável e 0, indesejável, a Equação 2.24 apresenta o cálculo da métrica de Homogeneidade $h(\mathscr{K}, \mathcal{CL})$.

$$h(\mathscr{K}, \mathcal{CL}) = \begin{cases} 1, & \text{se } H(\mathcal{CL}|\mathscr{K}) = 0 \\ 1 - \frac{H(\mathcal{CL}|\mathscr{K})}{H(\mathcal{CL})}, & \text{caso contrário} \end{cases} \tag{2.24}$$

Os cálculos das componentes $H(\mathcal{CL}|\mathscr{K})$ e $H(\mathcal{CL})$ são apresentadas nas Equações 2.25 e 2.26, respectivamente, onde a_{ij} é o número de objetos pertencentes a um grupo verdadeiro Cl_i associados ao grupo k_j.

$$H(\mathcal{CL}|\mathscr{K}) = \sum_{k_j}^{\mathscr{K}} \sum_{Cl_i}^{\mathcal{CL}} \frac{a_{Cl_i k_j}}{N} \log_2 \frac{a_{Cl_i k_j}}{\sum_{Cl_m}^{\mathcal{CL}} a_{Cl_m k_j}} \tag{2.25}$$

$$H(\mathcal{CL}) = -\sum_{Cl_i}^{\mathcal{CL}} \frac{\sum_{k_j}^{\mathscr{K}} a_{ij}}{l} \log_2 \frac{\sum_{k_m}^{\mathscr{K}} a_{im}}{l} \tag{2.26}$$

2.6.3 Completude

A Completude é uma métrica simétrica a de Homogeneidade. Para satisfazer o critério de completude, um agrupamento deve associar todos os objetos de um grupo verdadeiro a um único grupo (ROSENBERG; HIRSCHBERG, 2007). Para avaliar a

completude, examina-se a distribuição das associações de objetos de cada grupo verdadeiro. Em uma solução que atenda o critério de completude perfeitamente, cada uma das distribuições será associada a um único grupo. Avalia-se este viés ao calcular a entropia condicional do grupo proposto dado o grupo verdadeiro, $H(\mathscr{K}|\mathcal{CL})$. Ainda no caso perfeito, tem-se $H(\mathscr{K}|\mathcal{CL}) = 0$, e no pior caso, cada grupo verdadeiro seria representado por todos os grupos com uma distribuição igual à distribuição de tamanhos de grupos, onde $H(\mathscr{K}|\mathcal{CL})$ é máximo e igual a $H(\mathscr{K})$. Assim como no cálculo da Homogeneidade, a notação é invertida para atender a notação de completude máxima ser igual a 1 e mínima igual a 0. A Equação 2.27 apresenta o cálculo da Completude $c(\mathscr{K}, \mathcal{CL})$.

$$c(\mathscr{K}, \mathcal{CL}) = \begin{cases} 1, & \text{se } H(\mathscr{K}|\mathcal{CL}) = 0 \\ 1 - \frac{H(\mathscr{K}|\mathcal{CL})}{H(\mathscr{K})}, & \text{caso contrário} \end{cases} \tag{2.27}$$

As componentes $H(\mathscr{K}|\mathcal{CL})$ e $H(\mathscr{K})$ são apresentadas nas Equações 2.28 e 2.29, respectivamente, onde a_{ij} é o número de objetos pertencentes a um grupo verdadeiro Cl_i associados ao grupo k_j.

$$H(\mathscr{K}|\mathcal{CL}) = -\sum_{Cl_i}^{\mathcal{CL}} \sum_{k_j}^{\mathscr{K}} \frac{a_{ij}}{N} \log_2 \frac{a_{ij}}{\sum_{k_m}^{\mathscr{K}} a_{im}} \tag{2.28}$$

$$H(\mathscr{K}) = -\sum_{k_j}^{\mathscr{K}} \frac{\sum_{Cl_i}^{\mathcal{CL}} a_{ij}}{l} \log_2 \frac{\sum_{Cl_i}^{\mathcal{CL}} a_{ij}}{l} \tag{2.29}$$

2.6.4 V-Measure (*Validity Measure*)

Baseando-se no cálculo de Homogeneidade e Completude, a V-Measure realiza uma média harmônica destas outras métricas (ROSENBERG; HIRSCHBERG, 2007). A V-Measure possui um parâmetro Λ que define a ponderação dos fatores de Homogeneidade $h(\mathscr{K}, \mathcal{CL})$ e Completude $c(\mathscr{K}, \mathcal{CL})$ no cálculo final. Caso $\Lambda \geq 1$, o fator Completude possuirá valor maior na relação, caso contrário ($\Lambda < 1$), a Homogeneidade possuirá peso maior. A medida V-Measure $V(\mathscr{K}, \mathcal{CL}, \Lambda)$, utilizando o fator Λ é calculada de acordo com a Equação 2.30.

$$V(\mathscr{K}, \mathcal{CL}, \Lambda) = \frac{(1 + \Lambda) \times h(\mathscr{K}, \mathcal{CL}) \times c(\mathscr{K}, \mathcal{CL})}{\Lambda \times h(\mathscr{K}, \mathcal{CL}) + c(\mathscr{K}, \mathcal{CL})} \tag{2.30}$$

2.6.5 Pureza

A Pureza é uma métrica externa, simples e transparente para avaliação de agrupamentos (ZHAO; KARYPIS, 2004). A Pureza $py(\mathscr{K}|\mathcal{CL})$ é uma medida que determina se um grupo contém objetos de um único grupo verdadeiro. A pureza de um agrupamento é calculada através da Equação 2.31, onde para cada grupo k_i encontra-se o grupo verdadeiro Cl_j que maximize o número de objetos associados entre k_i e Cl_j, representado por n_{ij}. Finalmente, divide-se a soma de todos os n_{ij} pelo número total de objetos N que compõe o agrupamento.

$$py(\mathscr{K}|\mathcal{CL}) = \frac{1}{N}\sum_{k_i}^{\mathscr{K}} \max_j |k_i \cap Cl_j| \tag{2.31}$$

2.6.6 *Precision* e *Recall*

Precision e *Recall* são duas métricas externas de avaliação de agrupamentos. *Precision* retrata a fração de um grupo que consiste em objetos de um determinado grupo verdadeiro (RIJSBERGEN, 1979). Em outras palavras, representa a proporção de objetos dispostos em um grupo, sendo que eles de fato pertencem a este, em relação a todos os objetos daquele grupo.

A Equação 2.32 apresenta o cálculo da métrica *Precision* $pr(k_i, Cl_j)$ onde m_i representa o número de objetos em um grupo k_i e m_{ij} representa o número de objetos de um grupo verdadeiro Cl_j dispostos em um grupo k_i.

$$pr(k_i, Cl_j) = \frac{|k_i \cap Cl_j|}{|k_i|} \tag{2.32}$$

Por outro lado, *Recall* retrata a fração de objetos postos em um grupo em relação a todos aqueles objetos que de fato pertencem à aquele grupo (MOISE; SANDER; ESTER, 2006; RIJSBERGEN, 1979).

A Equação 2.33 apresenta o cálculo da métrica *Recall* $rc(k_i, Cl_j)$ onde m_j é o número de objetos no grupo verdadeiro Cl_j e m_{ij} denota o número de objetos de um grupo k_i em um grupo verdadeiro Cl_j.

$$rc(k_i, Cl_j) = \frac{|k_i \cap Cl_j|}{|Cl_j|} \tag{2.33}$$

2.6.7 Coeficiente da Silhueta

O Coeficiente de Silhueta é uma medida interna de avaliação de agrupamentos baseada nas métricas de coesão e separação (KAUFMANN; ROUSSEEUW, 1990; ROUSSEEUW, 1987). Este coeficiente é definido através de duas medidas importantes: a distância média entre $\vec{x}_i$ e os demais objetos dentro de um grupo, a_i; e a distância média entre $\vec{x}_i$ e os demais objetos do grupo mais próximo, b_i.

O Coeficiente da Silhueta $s(\vec{x}_i)$, para um objeto $\vec{x}_i$ é calculado a partir da Equação 2.34.

$$s(\vec{x}_i) = \frac{b_i - a_i}{\max(a_i, b_i)} \tag{2.34}$$

O valor deste coeficiente varia no intervalo $[-1; 1]$, sendo -1 o valor que representa um agrupamento incorreto e 1 um agrupamento de alta densidade, assim como espera-se que $a_i = 0$, fazendo com que o coeficiente assuma seu valor máximo.

Pode-se calcular o coeficiente da silhueta médio de um grupo ao realizar a média de todos os objetos pertences a este. Assim como pode-se calcular uma média de coeficiente de silhueta global $s(\mathscr{K})$ de um agrupamento $\mathscr{K}$ ao realizar a média de todos os objetos em $\mathscr{N}$ de acordo com a Equação 2.35.

$$s(\mathscr{K}) = \frac{1}{N} \times \sum_{\vec{x}_i}^{\mathscr{N}} s(\vec{x}_i) \tag{2.35}$$

2.6.8 Normalized Mutual Information – NMI

A métrica externa de NMI (*Normalized Mutual Information – Informação Mútua Normalizada*) tem sua origem na Teoria da Informação e se baseia na noção de Entropia. O cálculo da Entropia para um agrupamento $\mathscr{K}$ foi apresentado na Equação 2.29.

A noção de Entropia pode ser estendida para o conceito de MI (*Mutual Information – Informação Mútua*) que descreve o quanto se pode reduzir a incerteza sobre o grupo de um elemento arbitrário $\vec{x}_i$ em um agrupamento $\mathscr{K}$ quando sabe-se a alocação deste mesmo elemento na distribuição de grupos verdadeiros $\mathcal{CL}$ (STREHL; GHOSH, 2003). Formalmente, calcula-se a Informação Mútua MI de acordo com a Equação 2.36.

$$MI(\mathscr{K}, \mathcal{CL}) = \sum_{k_i}^{\mathscr{K}} \sum_{Cl_j}^{\mathcal{CL}} \frac{|k_i \cap Cl_j|}{|\mathscr{N}|} \log_2 \frac{N^2 \times |k_i \cap Cl_j|}{N \times |k_i| \times |Cl_j|} \tag{2.36}$$

A métrica de Informação Mútua não está limitada em um intervalo bem-definido,

tornando-a difícil de interpretar. Contudo, provas matemáticas provam que a Inequação 2.37 é sempre verdadeira, onde a Informação Mútua é sempre menor ou igual o menor valor entre a entropia do grupo $\mathscr{K}$ e dos grupos verdadeiros $\mathcal{CL}$.

$$MI(\mathscr{K}, \mathcal{CL}) \leq \min\{H(\mathscr{K}); H(\mathcal{CL})\} \tag{2.37}$$

Deste modo, uma possível normalização da métrica de informação mútua NMI é feita através da média geométrica, permitindo então que o valor da Informação Mútua esteja normalizada no intervalo $[0; 1]$ (STREHL; GHOSH, 2003). A Equação 2.38 apresenta o cálculo da métrica NMI.

$$NMI(\mathscr{K}, \mathcal{CL}) = \begin{cases} 1, & \text{se } H(\mathscr{K}) \times H(\mathcal{CL}) \neq 0 \\ \frac{\mathcal{MI}}{\sqrt{H(\mathscr{K}) \times H(\mathcal{CL})}}, & \text{caso contrário} \end{cases} \tag{2.38}$$

2.6.9 *Cluster Mapping Measure* (CMM)

Tendo em vista o fluxo contínuo de dados $\mathcal{S}$ que gera instâncias no formato $\vec{x}$ e de dimensão d, espera-se que instâncias mais recentes possuam maior relevância que as demais. Até o momento, nenhuma das métricas apresentadas avalia instâncias desta maneira. Deste modo, espera-se associar a cada instância um peso, dependente de sua "idade". Um dos exemplos mais comumente utilizados é uma função de decaimento exponencial apresentado na Equação 2.39, onde os parâmetros Φ e λ definem a forma da função; t_{atual} é o instante atual e t_i é o instante de chegada da instância $\vec{x_i}$.

$$w(\vec{x_i}) = \Phi^{-\lambda \times (t_{\text{atual}} - t_i)} \tag{2.39}$$

Computacionalmente, considerar todas as instâncias já apresentadas ao indutor ($t_i \leq t_{\text{agora}}$) acarretaria em armazenar todas estas instâncias, logo, tem-se o conceito de Horizonte de Avaliação ($\mathcal{H}_a$). Neste ponto, define-se formalmente o Horizonte de Avaliação $\mathcal{H}_a = \{\vec{x_i} \in \mathcal{S} \mid w(\vec{x_i}) \geq \Upsilon\}$ de um fluxo de dados $\mathcal{S}$, baseando-se em um limiar Υ. Basicamente, $\mathcal{H}_a$ tem como objetivo realizar avaliações do agrupamento dentro de janelas disjuntas de dados (*chunks*), sendo Υ a variável que controla o tamanho de $\mathcal{H}_a$.

No processamento de fluxos contínuos de dados, os algoritmos possuem suas próprias restrições e a avaliação destes não pode ser diferente. Deste modo, elencam-se algumas circunstâncias.

O processo de "envelhecimento" das instâncias é a tarefa mais simples a ser resolvida. Como visto na Equação 2.39, funções podem definir o peso das instâncias, permi-

tindo também ponderar os erros cometidos pelo indutor.

Grupos que realizam movimentações dentro do espaço de atributos acarretam em erros para instâncias tidas como ruído. Estes erros refletem a seriedade, i.e. quão próximo a instância está do grupo.

Evolução, fusão e divisão de grupos geram grupos sobrepostos. Deste modo, instâncias podem ser associadas ao grupo errado. Medidas que penalizem o algoritmo pela associação errônea destas instâncias da mesma maneira que penaliza instâncias associadas a grupos não sobrepostos não levam em conta as circunstâncias do processamento de fluxos de dados.

Finalmente, a inclusão de ruído em um grupo encontrado é muitas vezes inevitável. Assim, medidas de avaliação para fluxos contínuos de dados devem levar este fator em conta.

Três propriedades devem ser consideradas com maior significância: **instâncias não associadas**, **instâncias associadas a grupos errados** e **inclusão de ruído** (KREMER et al., 2011). O *Cluster Mapping Measure* (CMM) é uma soma normalizada destas penalidades.

Um dos conceitos centrais do CMM é a conectividade entre instâncias e grupos. Um dos pré-requisitos para definir quão conectado uma instância é, é o cálculo da distância média da k-vizinhança. A distância média da k-vizinhança $knhD(\cdot,\cdot)$ para uma instância $\vec{x}_i$ em um grupo Cl_j e seus k vizinhos mais próximos é dado pela Equação 2.40, onde $knh(\vec{x}_i, Cl_j)$ é o conjunto dos k-vizinhos mais próximos de $\vec{x}_i$ em Cl_j.

$$knhD(\vec{x}_i, Cl_j) = \frac{1}{k} \times \sum_{\vec{x_k}}^{knh(\vec{x}_i, Cl_j)} d(\vec{x}_i, \vec{x}_k) \tag{2.40}$$

A Equação 2.41 apresenta a distância média da k-vizinhança para um cluster Cl_j.

$$knhD(Cl_j) = \frac{1}{|Cl_j|} \times \sum_{\vec{x}_i}^{Cl_j} knhD(\vec{x}_i, Cl_j) \tag{2.41}$$

Para computar a métrica CMM, deve-se definir o conjunto de erros $\mathcal{F}$. Para um subconjunto de instâncias $\mathfrak{S} \subseteq \mathcal{S}$, sendo $\mathfrak{S}^+ = \mathfrak{S} \cup Cl_{\text{ruído}}$, o conjunto de erros $\mathcal{F}$ é definido pela Equação 2.42, onde $Cl'(\vec{x}_i)$ determina o grupo mapeado para o grupo verdadeiro pelo algoritmo como solução.

$$\mathcal{F} = \{\vec{x}_i \in \mathfrak{S}^+ \mid \exists Cl_j : \vec{x}_i \in Cl_j \wedge Cl'(\vec{x}_i) \neq Cl_j\} \tag{2.42}$$

Outro cálculo necessário é a definição da connectividade de uma instância $\vec{x}_i$ a um

grupo verdadeiro Cl_j: $con(\vec{x}_i, Cl_j)$. Esta distância é calculada a partir da Equação 2.43.

$$con(\vec{x}_i, Cl_j) = \begin{cases} 1, & \text{se } knhD(\vec{x}_i, Cl_j) < knhD(Cl_j) \\ 0, & \text{se } Cl_j = \emptyset \\ \frac{knhD(Cl_j)}{knhD(\vec{x}_i, Cl_j)}, & \text{caso contrário} \end{cases} \tag{2.43}$$

Tendo em vista estes valores, pode-se então computar o valor de CMM para um agrupamento obtido $\mathscr{K}$ e conjunto de grupos verdadeiros $\mathcal{CL}$. A Equação 2.44 apresenta o cálculo da métrica CMM, onde $pen(\cdot,\cdot)$ é uma função que determina se a instância $\vec{x}_i$ é uma instância não associada, associada ao grupo incorreto ou ruidosa indevidamente incluída. O valor da métrica CMM reside no intervalo $[0;1]$, onde 1 representa erro mínimo e 0, máximo.

$$CMM(\mathscr{K}, \mathcal{CL}) = \begin{cases} 1, & \text{se } \mathcal{F} = \emptyset \\ 1 - \frac{\sum_{\vec{x}_i}^{\mathcal{F}} w(\vec{x}_i) \times pen(\vec{x}_i, \mathscr{K})}{\sum_{\vec{x}_j}^{\mathcal{F}} w(\vec{x}_j) \times con(\vec{x}_j, Cl(\vec{x}_j))}, & \text{caso contrário} \end{cases} \tag{2.44}$$

Existem diversas versões do CMM. Neste documento aborda-se apenas a principal e completa versão. Para maiores detalhes sobre variações desta métrica, sugere-se ao leitor a leitura de (KREMER et al., 2011).

2.7 Considerações Finais

Este capítulo apresentou o problema de mineração de fluxos de dados, focando na tarefa de agrupamento e sua avaliação. Como discutido, existem diversos desafios a serem enfrentados no desenvolvimento de novos algoritmos de Agrupamento *Online*, onde ressalta-se a preocupação com tempo e memória, maximização de métricas de qualidade de agrupamento, realizar atualizações do agrupamento de maneira incremental, utilizar estruturas de dados compactas e detectar e eliminar dados ruidosos e *outliers*.

Ressalta-se neste ponto que os algoritmos apresentados na literatura possuem uma limitação ao apresentarem soluções para tais problemas: parametrização. Os algoritmos CluStream, ClusTree e DenStream utilizam variações dos algoritmos *k-means* e DBSCAN em suas etapas *offline*.

Ao utilizar o algoritmo *k-means*, uma forte limitação é imposta: deve-se oferecer o número de grupos K a ser encontrado a cada *chunk* de avaliação; ou deve-se informar este valor no início do fluxo de dados, assumindo que este valor não variará durante o fluxo de dados. Em ambos os casos, não é possível assumir estas limitações para fluxos de

dados, uma vez que não se pode garantir que um usuário será capaz de definir o número de grupos a cada *chunk* de dados de maneira correta ou que o número de grupos reais não mude.

Por outro lado, a utilização do algoritmo DBSCAN permite encontrar grupos não hiper-esféricos mas implica na definição de um maior número de parâmetros, que por sua vez, afetam diretamente os grupos encontrados.

O próximo capítulo apresenta conceitos básicos sobre análise de redes sociais. Estes conceitos são fundamentais para a apresentação dos algoritmos propostos, que utilizam uma rede social para encontrar grupos de maneira incremental sem necessidade de processamento *batch* na etapa *offline*.

Capítulo 3

Análise de Redes Sociais

Desde muito antes da Internet se tornar parte da vida das pessoas, sociólogos e outros pesquisadores da área de Ciências Humanas tem analisado a estrutura derivada da interação das pessoas: as redes. Na maioria destes estudos, grupos pequenos foram considerados, principalmente devido a dificuldade de realizar a análise em grupos maiores.

Uma importante contribuição para a Análise de Redes Sociais foi o sociograma (MORENO, 1934). Um sociograma pode ser visto como uma representação gráfica de uma rede: atores são representados por pontos (vértices) e relacionamentos entre atores são linhas ligando estes pontos (arestas). Contudo, somente décadas depois da definição de sociograma, matemáticos formalizaram sociogramas na forma de grafos. Grafos são representações que permitem pesquisadores focar na estrutura destas redes, com o intuito de realizar generalizações sobre o comportamento de um grupo.

Deste modo, a Análise de Redes Sociais tem contribuído para o desenvolvimento da Teoria dos Grafos, particularmente ao introduzir métricas de importância (centralidade e prestígio) de atores ou grupos. Enquanto a Teoria dos Grafos nos provê ferramentas para descrever formalmente redes e a importância de seus componentes, a Análise de Redes Sociais se preocupa ainda com a formação e evolução destas redes (STEEN, 2010).

Este capítulo está dividido da seguinte maneira: A Seção 3.1 apresenta conceitos básicos acerca de redes sociais. A Seção 3.2 formaliza uma notação para redes baseada em grafos, assim como métricas e conceitos importantes para o remanescente do documento. A Seção 3.3 relata os principais modelos de formação e evolução de redes sociais: Aleatório, de Mundo Pequeno e Livre de Escala. A Seção 3.4 apresenta os principais métodos de detecção de comunidades em rede, um problema análogo ao de agrupamento. Finalmente, a Seção 3.5 conclui este capítulo, revisando os tópicos relevantes para a apresentação dos algoritmos propostos.

3.1 Redes Sociais

Redes Sociais provêem uma maneira precisa de definir conceitos sociais importantes. A modelagem de Redes Sociais foca na conceitualização de estruturas sociais, e.g. comportamentais, econômicas e socio-políticas, como sendo padrões de relações entre indivíduos. A Análise de Redes Sociais engloba teorias, modelos e aplicações que são expressas através de conceitos relacionais como formalismos matemáticos da Teoria dos Grafos e Estatística (WASSERMAN; FAUST, 1994) que focam a construção e evolução destas redes. Basicamente, uma rede social é um conjunto finito de atores e a relação entre eles, sendo que a presença de informação das relações é uma característica fundamental (WASSERMAN; FAUST, 1994).

Atores são entidades individuais, que juntos, compõe uma sociedade. Atores podem ser pessoas, organizações e sistemas que podem ser analisadas individualmente, ou em escopo mais abrangente, ou seja, analisando a relação entre eles, como: amizade, conhecimento profissional ou proximidade geográfica.

Deste modo, relacionamentos entre atores, denominados "laços de relacionamento" são altamente dependentes do tipo de rede. Por exemplo, em uma rede de colegas de um departamento, podem existir relacionamentos de amizade, apreciação ou de respeito.

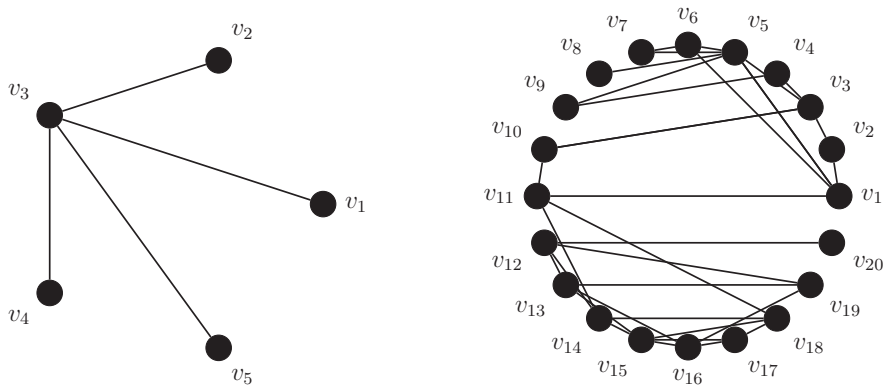
Redes sociais tem como objetivo medir e representar a estrutura de relações entre atores, explicar o motivo da existência destas relações e quais as suas consequências. Originalmente, elas eram utilizadas na sociologia e antropologia, contudo, com o desenvolvimento dos formalismos matemáticos e computação, sua aplicação se expandiu para áreas como: Computação, Matemática, Física, Saúde, Economia; e são aplicadas em domínios da Internet, Modelos de Propagação de Vírus, Movimentos Sociais, Redes de Terrorismo e Redes de Distribuição (NEWMAN, 2010).

Para fornecer notações a uma Rede Social, existem três abordagens: baseada em Grafos, Sociométrica e Algébrica (WASSERMAN; FAUST, 1994). Neste trabalho utiliza-se apenas a baseada em grafos, devido a seu rigor matemático e sua facilidade de implementação no ambiente computacional.

3.2 Teoria dos Grafos

A Teoria dos Grafos fornece uma possível representação de redes sociais e um conjunto de conceitos que podem ser utilizados para o estudo formal de suas propriedades. Logo, uma rede social pode ser representada por um grafo de forma precisa e inequívoca.

Além disso, devido a formalização matemática da Teoria dos Grafos, é possível que



(a) Exemplo de Rede Social com 5 atores.

(b) Exemplo de Rede Social com 20 atores.

Figura 3.1: Exemplos de redes sociais representadas por sociogramas.

idéias subjetivas vindas da Análise de Redes Sociais sejam expressas de maneira precisa, logo, passíveis de mensurar, reproduzir e testar.

A representação visual de uma rede social através de um grafo é comumente denominada "Sociograma" (MORENO, 1934). Representações visuais permitem a pesquisadores observar padrões que poderiam passar desapercebidos de outra maneira. A Figura 3.1 apresenta exemplos de redes sociais representadas como sociogramas. A rede da Figura 3.1a possui 5 atores (v_1 a v_5), onde percebe-se claramente as relações entre os vértices, assim como a rede da Figura 3.1b possui 20 atores (v_1 a v_{20}) e suas relações.

Não obstante, a utilização desta representação é inviável ou pode dificultar a descoberta de padrões por parte dos pesquisadores, primariamente devido a grande quantidade de elementos que realizam a sua composição. Por exemplo, a rede da Figura 3.1b possui duas comunidades fortemente coesas, contudo, a representação visual pode desfavorecer a percepção deste fato.

Outra possível representação é a sociomatriz: uma matriz de adjacências. Em matrizes de adjacências, atores são posicionados nas linhas e colunas; e a interseção entre dois atores determina se o laço de relacionamento existe (✓) ou não. As Figuras 3.2a e 3.2b apresentam as matrizes de adjacências para as redes das Figuras 3.1a e 3.1b, respectivamente. Este tipo de representação é inviável para o escopo deste trabalho, devido a alteração constante dos atores, deste modo, sugere-se ao leitor a leitura de (STEEN, 2010; WASSERMAN; FAUST, 1994) para mais detalhes.

Grafos podem ser divididos em dois tipos: direcionados (dígrafos) e não direcionados. Grafos não direcionados representam relações onde a ordem dos seus componentes (atores) é irrelevante, por exemplo: "v_1 é amigo de v_2" é o mesmo que "v_2 é amigo de v_1". Por outro lado, grafos direcionados possuem relações onde a ordem destes componentes importa, por exemplo: "v_1 é pai de v_2" é diferente de "v_2 é pai de v_1". Todas as redes até

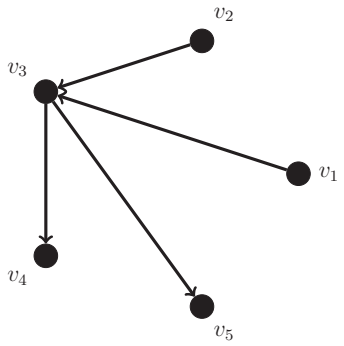
	v_1	v_2	v_3	v_4	v_5
v_1			✓		
v_2			✓		
v_3	✓	✓		✓	✓
v_4			✓		
v_5			✓		

(a) Matriz de Adjacências para a Rede da Figura 3.1a.

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}	v_{15}	v_{16}	v_{17}	v_{18}	v_{19}	v_{20}
v_1		✓			✓	✓														
v_2	✓		✓																	
v_3		✓		✓	✓					✓										
v_4			✓						✓											
v_5	✓		✓			✓	✓	✓	✓											
v_6	✓				✓		✓													
v_7					✓	✓														
v_8					✓															
v_9				✓	✓															
v_{10}			✓								✓									
v_{11}										✓				✓				✓		
v_{12}													✓	✓					✓	✓
v_{13}												✓			✓	✓			✓	
v_{14}											✓	✓			✓			✓		
v_{15}													✓	✓		✓	✓	✓		
v_{16}													✓		✓		✓		✓	
v_{17}															✓		✓	✓		
v_{18}											✓			✓	✓		✓			
v_{19}												✓	✓			✓				
v_{20}												✓								

(b) Matriz de Adjacências para a Rede da Figura 3.1b.

Figura 3.2: Exemplos de redes sociais representadas por sociomatrizes.



(a) Exemplo de dígrafo com 5 atores.

	v_1	v_2	v_3	v_4	v_5
v_1			✓		
v_2			✓		
v_3				✓	✓
v_4					
v_5					

(b) Matriz de Adjacências para a Rede da Figura 3.3a.

Figura 3.3: Exemplo de rede social com 5 atores em forma de dígrafo.

o momento descritas são grafos não direcionados. A Figura 3.3 apresenta um exemplo de rede social em forma de dígrafo, onde a Figura 3.3a a representa em forma de sociograma e a Figura 3.3b através de matriz de adjacências.

Para este trabalho, formaliza-se um grafo na forma $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, onde $\mathcal{V} = \{v_1, v_2, \ldots, v_V\}$ representa o conjunto de atores (vértices) onde $|\mathcal{V}| = V$, $\mathcal{E} = \{e_1, e_2, \ldots, e_E\}$ ($|\mathcal{E}| = E$) o conjunto de relações (arestas) entre atores na forma $e_i = \langle v_a, v_b \rangle$, onde v_a e v_b são vértices arbitrários e $\mathcal{W} = \{w_1, w_2, \ldots, w_E\}$ representa ponderações para cada aresta, i.e. para cada aresta e_i existirá um peso w_i associado. Para este trabalho, considera-se apenas grafos não direcionados.

3.2.1 Caminhamentos em Grafos

Além de analisar as conexões diretas que ocorrem entre vértices adjacentes é interessante analisar se um vértice é "alcançável" a partir de outro. Em um grafo, é de interesse determinar a possibilidade de gerar percursos de movimentação no grafo (cami-

nhamentos) partindo de um vértice v_i até outro v_j respeitando as arestas descritas por $\mathcal{E}$. No caso positivo, é interessante saber de quantas formas isso é possível, quais são ótimas e quais foram os percursos realizados (quais vértices foram visitados e arestas). Um caminhamento é denotado por $P(v_i, v_z) = v_i \Rightarrow v_z = \{v_i \rightarrow v_j, \ldots, v_x \rightarrow v_z\}$: uma sequência de movimentações dentro do grafo onde cada item é uma adjacência explorada de $\mathcal{E}$.

Ainda, uma importante definição é a de tamanho médio de caminho em grafos. O tamanho médio de caminho de uma rede $\overline{P}(\mathcal{G})$ computa a média dos tamanhos de $|P(\cdot,\cdot)|$ possíveis para todas as combinações de vértices $\forall v_i, v_j, v_i \neq v_j, \in \mathcal{V}$. O tamanho médio de caminho é calculado através da Equação 3.1.

$$\overline{P}(\mathcal{G}) = \frac{2}{V(V-1)} \sum_{v_i, v_j, v_i \neq v_j}^{\mathcal{V}} |P(v_i, v_j)| \tag{3.1}$$

Finalmente, deve-se definir o conceito de grafo conexo. Um grafo é dito conexo se todos os pares de vértices do grafo estiverem conectados, i.e. $\forall v_i, v_j \in \mathcal{V}, \exists(v_i \Rightarrow v_j)$.

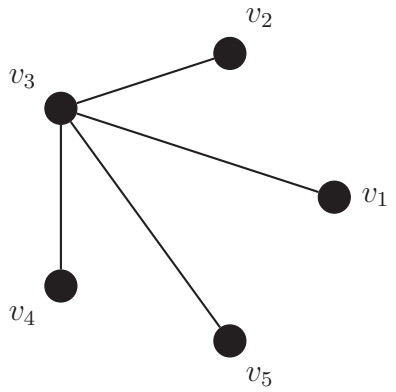
3.2.2 Métricas de Centralidade

A utilização da Teoria dos Grafos permite definir a importância de um vértice dentro da rede, utilizando conceitos de centralidade e prestígio. Métricas de prestígio tem como objetivo mensurar quão "receptor" um vértice é (NEWMAN, 2010), deste modo, são computáveis apenas em redes direcionadas (dígrafos) e não serão abordadas neste trabalho.

Métricas de centralidade, por outro lado, são medidas de acordo com as relações às quais um vértice em questão está envolvido. Calcular a centralidade de um vértice consiste em identificar a posição em que ele se encontra perante as trocas e a comunicação na rede, ou seja, sua posição dentro da topologia. Logo, quão mais central for um indivíduo, melhor posicionado ele está na rede, aumentando também o seu "poder" e importância dentro do conjunto (ZAFARANI; ABBASI; LIU, 2014). Métricas de centralidade podem ser computadas para subgrupos de uma rede através do cálculo da média das centralidades de seus vértices componentes.

Para este trabalho, duas métricas de centralidade são essenciais: Centralidade de Grau (*Degree*) e Centralidade de Intermediação (*Betweenness*).

A centralidade de grau para um vértice v_i é o grau do vértice, ou seja, a quantidade de vizinhos $deg(\cdot)$ que v_i possui dentro de G. Para eliminar o efeito da variação do tamanho da rede, permitindo comparações entre redes, uma versão normalizada da



- Geodésica #1: $v_1 \Rightarrow v_2$
- Geodésica #2: $v_1 \Rightarrow v_4$
- Geodésica #3: $v_1 \Rightarrow v_5$
- Geodésica #4: $v_2 \Rightarrow v_4$
- Geodésica #5: $v_2 \Rightarrow v_5$
- Geodésica #6: $v_4 \Rightarrow v_5$

Figura 3.4: Exemplo de rede onde $deg(v_3) = 4$, $c_{deg}(v_3) = \frac{4}{8} = \frac{1}{2}$, $g(v_3) = 6$ e $g'(v_3) = \frac{6}{6} = 1$.

métrica de centralidade de grau $c_{deg}(v_i)$ no intervalo $[0; 1]$ é apresentada na Equação 3.2 (WASSERMAN; FAUST, 1994).

$$c_{deg}(v_i) = \frac{deg(v_i)}{\sum_{v_j}^{\mathcal{V}} deg(v_j)} \tag{3.2}$$

Interações entre dois vértices não adjacentes podem depender de outros atores em $\mathcal{V}$, especialmente de atores que estão em caminhamentos entre estes dois vértices. Estes outros atores, potencialmente, podem ter controle sobre a interação destes outros vértices (STEEN, 2010). Deste modo, considera-se o exemplo da Figura 3.4 onde são apresentados 4 vértices: $V = \{v_1, v_2, v_3, v_4\}$, as arestas $E = \{e_1 = \langle v_1, v_3\rangle, e_2 = \langle v_2, v_3\rangle, e_3 = \langle v_3, v_4\rangle, e_4 = \langle v_4, v_5\rangle\}$ e $\forall w_i \in \mathcal{W}, w_i = 1$.

Logo, computa-se os caminhos mais curtos (geodésicas) $g(\cdot)$ entre cada par de vértices de $\mathcal{V}$ que passe por v_3: $g(v_3) = \{v_1 \Rightarrow v_2, v_1 \Rightarrow v_4, v_1 \Rightarrow v_5, v_2 \Rightarrow v_4, v_2 \Rightarrow v_5, v_4 \Rightarrow v_5\}$.

Para computar o valor da centralidade de intermediação $g'(\cdot)$ de um vértice v_i, deve-se computar a relação entre a sua quantidade de geodésicas $g(v_i)$ e a quantidade de geodésicas que passam por todos os vértices de $\mathcal{V}$, conforme a Equação 3.3 (NEWMAN, 2010). Ou seja, a métrica de centralidade de intermediação mede quão próximo ao centro um ator está na comunicação entre os demais.

$$g'(v_i) = \frac{g(v_i)}{\sum_{v_j}^{\mathcal{V}} g(v_j)} \tag{3.3}$$

Outras métricas de centralidade como: Auto-vetor, Autoridade HITS, Baricentro, Pagerank e de Proximidade são discutidas em (NEWMAN, 2010).

3.2.3 Coeficiente de Agrupamento

Na Teoria dos Grafos, o coeficiente de agrupamento é uma métrica do grau de tendência que os vértices possuem de se agruparem. Trabalhos mostram que, em grande parte das redes do mundo real, vértices possuem a tendência de criarem subgrupos bastante específicos, caracterizados por uma alta densidade de conexões (ALBERT; BARABÁSI, 2002; STEEN, 2010).

Coeficientes de agrupamento podem ser medidos de duas maneiras: local e globalmente, onde esta quantifica o agrupamento da rede inteira, enquanto aquela indica quão embutido na rede um vértice está.

O coeficiente de agrupamento local de um vértice v_i, denotado por $\mathcal{CC}(v_i)$, mensura quão próximo os seus vértices adjacentes $adj(v_i)$ estão de formar um subgrafo completo (clique). Esta medida foi introduzida com a finalidade de determinar se uma rede possuía topologia de mundo pequeno, modelo discutido na Seção 3.3.3 (WATTS; STROGATZ, 1998). A Equação 3.4 apresenta o cálculo do coeficiente de agrupamento para um vértice arbitrário v_i, onde e_i representa o número de conexões dos vizinhos de v_i, i.e. $e_i = \sum_{v_k}^{adj(v_i)} deg(v_k)$. Caso os vértices vizinhos a v_i componham um clique (subgrafo completo), o número de conexões entre eles será igual a $\frac{deg(v_i)\times(deg(v_i)-1)}{2}$ e, consequentemente, $\mathcal{CC}(v_i)$ assumirá valor máximo 1.

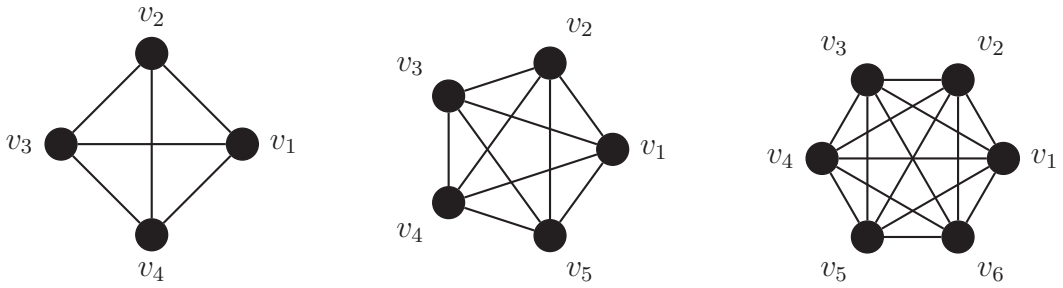
$$\mathcal{CC}(v_i) = \frac{2 \times e_i}{deg(v_i) \times (deg(v_i) - 1)} \tag{3.4}$$

Contudo, uma medida útil para analisar agrupamentos em redes é o coeficiente de agrupamento médio. O coeficiente de agrupamento médio assume o valor médio dos coeficientes de agrupamentos locais de todos os vértices em $\mathcal{V}$. A Equação 3.5 apresenta o cálculo do coeficiente de agrupamento médio.

$$\overline{\mathcal{CC}}(\mathcal{G}) = \frac{1}{V} \times \sum_{v_i}^{\mathcal{V}} \mathcal{CC}(v_i) \tag{3.5}$$

3.3 Modelos de Rede

Com o intuito de representar a formação e evolução de redes sociais reais diversos modelos foram propostos na literatura. Exemplos reais de redes sociais envolvem, por exemplo: a *World Wide Web*, Redes de Colaboração Científica, Redes Sociais Virtuais, Redes Ecológicas, Redes de Chamadas Telefônicas e Redes intra e inter-atômicas (ALBERT; BARABÁSI, 2002).



(a) K_4 (b) K_5 (c) K_6

Figura 3.5: Exemplo de grafos completos.

3.3.1 Regular

Um primeiro modelo é o modelo Regular. Este modelo é derivado da Teoria dos Grafos, onde todos os vértices em $\mathcal{V}$ possuem o mesmo número de adjacências (grau) $deg(\cdot)$.

Um caso específico de grafo regular são os grafos completos. Grafos completos são aqueles onde cada vértice está conectado a todos os demais, logo $\forall v_i \in \mathcal{V}, deg(v_i) = k = V - 1$. As Figuras 3.5 apresenta os grafos completos K_4, K_5 e K_6, respectivamente.

Uma importante característica de grafos completos é que estes possuem coeficiente de agrupamento global máximo (1).

3.3.2 Aleatório

Inicialmente, acreditava-se que seria possível modelar os efeitos de redes sociais reais usando componentes de aleatoriedade. O primeiro modelo aleatório é o de Erdos-Renyi, que afirma que a inserção de um novo vértice v_i em uma rede deve estabelecer conexões de acordo com uma probabilidade p (ERDOS; RÉNYI, 1960). Deste modo, este modelo assume que a rede é igualitária, pois todos os vértices possuem a mesma probabilidade de estabelecerem conexões. Em uma rede com V vértices, arestas são escolhidas aleatoriamente entre $\frac{V(V-1)}{2}$ arestas. Logo, existem $\binom{V(V-1)/2}{E}$ possíveis redes com V vértices e E arestas.

A Figura 3.6 apresenta um exemplo de rede aleatória com 10 vértices.

Redes aleatórias possuem coeficiente de agrupamento dado de acordo com a Equação 3.6, onde $\langle k \rangle$ é o grau médio da rede. Logo, redes aleatórias são conhecidas por possuírem coeficientes de agrupamento baixos, uma vez que são aproximadamente iguais a probabilidade p.

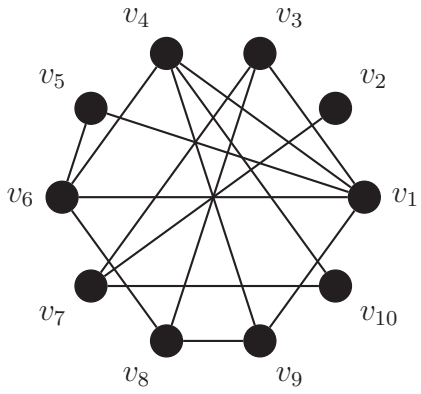


Figura 3.6: Exemplo de rede aleatória.

$$\mathcal{CC}(\mathcal{G}) = \frac{\langle k \rangle}{V} = \frac{\frac{2E}{V}}{V} = \frac{pV}{V} = p \tag{3.6}$$

3.3.3 Mundo Pequeno

Os modelos de Mundo Pequeno são inspirados no experimento de (MILGRAM, 1967), onde observou-se a probabilidade de que duas pessoas escolhidas ao acaso se conheçam. Neste experimento, cartas foram enviadas aleatoriamente a vários indivíduos, pedindo que estes as enviassem a um destinatário específico desconhecido. Logo, os remetentes deveriam enviar cartas para outros indivíduos, crendo que tais indivíduos conhecessem o destinatário final. O resultado foi que todas as cartas que chegaram aos destinatários haviam passado por um número pequeno de pessoas (na média, 6), denominando então o fenômeno de "mundo pequeno" e os 6 graus de separação.

O principal modelo de Redes de Mundo Pequeno é o Watts-Strogatz (WATTS; STROGATZ, 1998). O modelo de Watts-Strogatz também se encaixa como um modelo aleatório, contudo, tenta combinar características de redes regulares.

Enquanto redes aleatórias possuem coeficiente de agrupamento global baixo e distância média entre vértices também inferior a redes regulares, redes de mundo pequeno apresentam coeficientes de agrupamento similares a redes regulares e tamanho de caminho médio similar a redes aleatórias.

Os passos para geração de redes de mundo pequeno Watts-Strogatz são (ALBERT; BARABÁSI, 2002; WATTS; STROGATZ, 1998):

1. *Iniciar com ordem*: Iniciar com um reticulado de V vértices, onde cada vértice v_i está conectado com k outros vértices ($\frac{k}{2}$) de cada lado. Para possuir uma rede esparsa, contudo, conexa, deve-se considerar $V \gg k \gg \ln(V) \gg 1$.

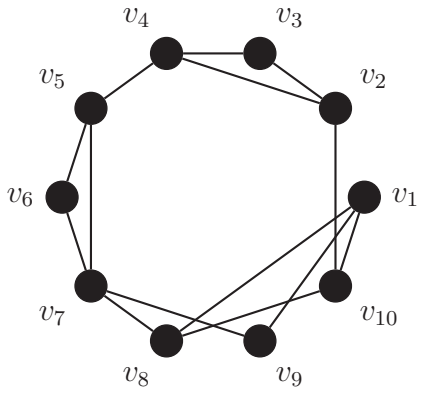


Figura 3.7: Exemplo de rede de mundo pequeno.

2. *Imprimir aleatoriedade*: Efetuar religações das arestas do reticulado com probabilidade p garantindo que auto-ciclos não existam. Este processo introduz $\frac{pVk}{2}$ arestas de "longa distância" originalmente de outra vizinhança.

Ainda, redes de mundo pequeno possuem coeficiente de agrupamento dado pela Equação 3.7, sendo, na média, superiores quando comparados com redes aleatórias (ALBERT; BARABÁSI, 2002).

$$\mathcal{CC}(\mathcal{G}) = \frac{3k(k-1)}{2k(2k-1) + 8pk^2 + 4p^2k^2} \tag{3.7}$$

A Figura 3.7 apresenta um exemplo de rede de mundo pequeno onde percebe-se que vértices possuem probabilidade de serem vizinhos de seus vizinhos.

3.3.4 Livre de Escala

Todos os modelos comentados anteriormente são fundamentalmente aleatórios. Nossa intuição tende a nos desacreditar que modelos tão complexos como, por exemplo, da Internet, possuam processos de construção e evolução apenas aleatórios (ALBERT; BARABÁSI, 2002).

A emergência de uma rede livre de escala depende de dois elementos fundamentais, que juntos, são capazes de representar a dinâmica de redes sociais reais. O primeiro é a evolução da rede. Nos modelos discutidos anteriormente, assume-se que o processo inicia com um número fixo de vértices (em redes aleatórias) ou com um número fixo de vértices e algumas conexões (redes de mundo pequeno). Contudo, processos de formação reais divergem destes, pois se iniciam com um número pequeno de vértices (ou até mesmo nenhum) e que cresce ao longo do tempo. O segundo é o processo que determina como conexões (arestas) são criadas. Nos modelos anteriores, assume-se uma probabilidade para

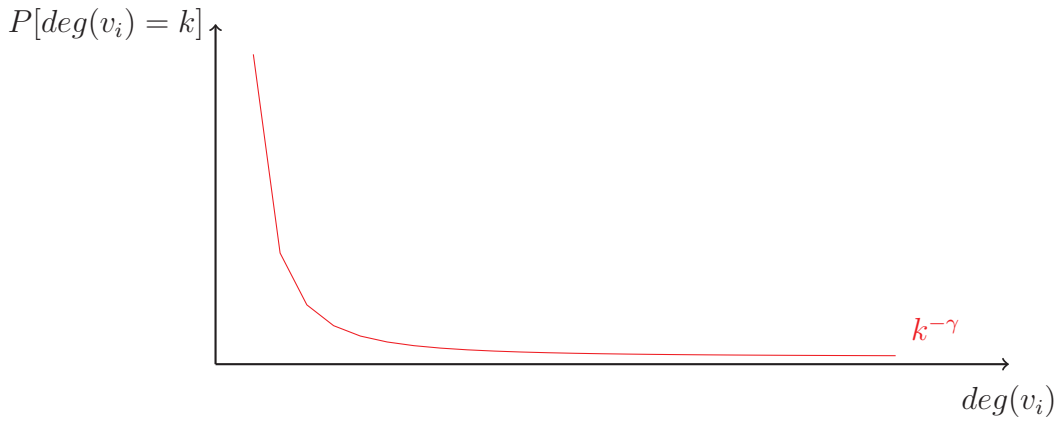


Figura 3.8: Distribuição de grau em redes livres de escala na forma $k^{-\gamma}$.

que conexões sejam criadas, ou que estas efetuem religações. Contudo, em redes reais, observa-se que vértices com mais conexões tendem a estabelecer ainda mais conexões, e vértices com poucas conexões tendem a receber menos. Esta propriedade é denominada "Anexação Preferencial" (*Preferential Attachment*) e simula o "efeito Mateus", onde ricos ficam mais ricos e os pobres, mais pobres (ALBERT; BARABÁSI, 2002; BARDDAL; GOMES; ENEMBRECK, 2014). A Equação 3.8 apresenta a probabilidade de um vértice v_i estabelecer uma nova conexão de acordo com a Anexação Preferencial.

$$\prod(v_i) = \frac{deg(v_i)}{\sum_{v_j}^{\mathcal{V}} deg(v_j)} \tag{3.8}$$

Em contraste com os outros modelos, onde a probabilidade de conexões independe do grau do vértice, nas redes livres de escala, a probabilidade de um vértice receber uma nova conexão é proporcional a quantidade de conexões que ele já possui em relação aos demais. Logo, a distribuição de grau dos atores é dado segundo uma lei de potência, apresentada na Equação 3.9, onde γ varia no intervalo $[2;3]$ para redes reais (ALBERT; BARABÁSI, 2002).

$$P[deg(v_i) = k] \sim k^{-\gamma} \tag{3.9}$$

Através de levantamentos e estudos empíricos, constatou-se que redes sociais reais, como a *World Wide Web*, a Internet, as Redes de Colaboração Científica, Redes Ecológicas e Redes de Smart Grid; possuem uma mesma característica: distribuição de grau seguindo uma lei de potência (ALBERT; BARABÁSI, 2002), determinando então o nome do modelo, uma vez que leis de potência são livres de uma escala característica. A Figura 3.8 apresenta um exemplo de distribuição de grau genérica para redes livres de escala.

Deste modo, em redes cuja distribuição de grau segue uma lei de potência, alguns

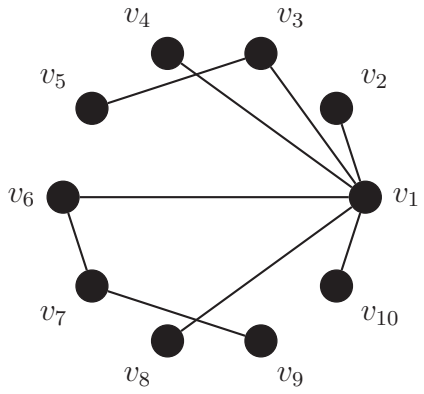


Figura 3.9: Exemplo de rede livre de escala.

vértices possuem alto grau (elevado número de conexões) e possuem maior probabilidade de receberem novas conexões. Estes vértices são denominados *hubs*, por possuírem métricas de centralidade de grau altas.

O algoritmo base para construção de redes livres de escala é dado por (ALBERT; BARABÁSI, 2002):

1. *Crescimento*: Iniciando com um número pequeno de vértices m_0, a cada unidade de tempo t, adiciona-se um vértice com m conexões.

2. *Anexação Preferencial*: Ao escolher os vértices aos quais m vértices este novo vértice deverá se conectar, utiliza-se a probabilidade de Anexação Preferencial (Equação 3.8) para determinar estes m atores.

A Figura 3.9 apresenta um exemplo de rede livre de escala com 10 vértices criada utilizando este algoritmo. Ainda na Figura 3.9, percebe-se que o vértice v_1 possui grau bastante superior aos demais, logo, é o principal *hub* desta rede.

Por um lado, o modelo genérico apresentado é capaz de representar as características de distribuição de grau de redes reais, contudo, redes reais além desta distribuição exponencial, também apresentam alto coeficiente de agrupamento global (STEEN, 2010). Logo, pesquisadores possuem alto interesse em gerar modelos capazes de estruturar redes livres de escala com esta característica.

Outro modelo para geração de redes livres de escala utiliza a noção de "eventos locais". Neste modelo, além da inserção de vértices baseando-se em anexação preferencial, existe um processo de religação estocástico dos vértices (ALBERT; BARABÁSI, 2000). Este processo é análogo ao apresentado em redes de mundo pequeno. Iniciando com m_0 vértices isolados ($\mathcal{E} = \emptyset$), utiliza-se uma das três operações:

- Com probabilidade p insere-se m arestas, sendo $m \leq m_0$;

- Com probabilidade q são religadas m arestas. Para esta aleatoriedade seleciona-se um vértice v_i e uma aresta e_j que o conecte a outro vértice v_j. Posteriormente, substitui-se e_j por uma nova aresta e_k conectando v_i a v_k de acordo com a probabilidade da Equação 3.8.

- Com probabilidade $1-p-q$ adiciona-se um novo vértice. O novo vértice estabelecerá m conexões com probabilidade $\prod(v_i)$ (apresentada na Equação 3.10) para os vértices v_i dispostos na rede.

$$\prod(v_i) = \frac{deg(v_i)+1}{\sum_{v_j}^{\mathcal{V}}(deg(v_j)+1)} \tag{3.10}$$

3.4 Algoritmos para Detecção de Comunidades

Uma vez que estudos de redes sociais têm ganho proeminência, um dos principais desafios de pesquisa é como extrair conhecimento a partir destas redes. Pesquisadores, buscando respostas para este problema, apontam que a estrutura da rede possui grande valia (STEEN, 2010). Por exemplo, em uma rede de consumidores, as subestruturas da rede podem determinar grupos de consumidores com mesmo padrão de compra; uma informação valiosa para planos de marketing, sistemas de recomendação e adaptações de interface humano-computador (PAPADOPOULOS et al., 2012). Estas subestruturas são também conhecidas como comunidades, grupos, *clusters* ou subgrupos coesos (ZAFARANI; ABBASI; LIU, 2014).

Uma das principais propriedades que causam a coesão de comunidades de indivíduos em redes sociais é a **homofilia**. O processo de homofilia ocorre quando indivíduos em uma rede estabelecem novas conexões devido a sua similaridade. Contudo, modelar o processo de homofilia ainda é uma questão em aberto para pesquisadores (ZAFARANI; ABBASI; LIU, 2014).

A Detecção de Comunidades é uma questão comumente discutida em diversas áreas e em diferentes formas. Quantização na Engenharia Elétrica, Discretização em Estatística, agrupamento em Aprendizagem de Máquina são exemplos de tarefas que objetivam desafios similares. Ressalta-se neste ponto que as comunidades apresentadas nesta seção são diferentes quando comparados aos grupos dentro do contexto de agrupamento, contudo, muitos dos algoritmos para detecção de comunidades são os mesmos utilizados para a análise de agrupamentos, principalmente quando algoritmos de agrupamento representam instâncias em forma de vértices em grafos. As próximas seções tem como objetivo apresentar sucintamente os principais métodos para detecção de comunidades em redes

sem clamar pela completude.

3.4.1 Hierárquicos

Métodos hierárquicos criam composições ou decomposições a partir de uma rede cujas arestas são baseadas em métricas de similaridade/dissimilaridade e são divididos em aglomerativos e divisivos. Ainda, estes algoritmos derivam uma árvore geradora mínima, utilizando os algoritmos de Prim (PRIM, 1957) ou Kruskal (KRUSKAL, 1956). A abordagem aglomerativa se inicia com cada vértice formando um grupo unitário. A partir destes grupos iniciais, os mais próximos são agrupados iterativamente, até que apenas um grupo exista, ou até que um critério de parada definido seja satisfeito.

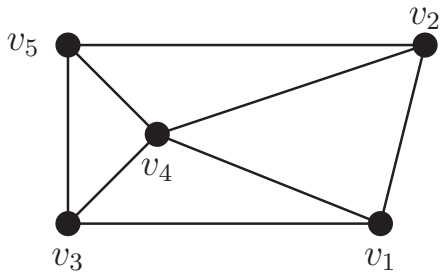
Por outro lado, a abordagem divisiva se inicia com todos os vértices em um único grupo. A cada iteração, um grupo é dividido em subgrupos, até que todos os vértices estejam em apenas um grupo ou algum critério de parada seja satisfeito. A Figura 3.10a apresenta um conjunto de uma rede composta por 5 vértices $\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\}$. A Figura 3.10b apresenta a árvore geradora mínima e seus agrupamentos/divisões realizados(as) em uma representação bidimensional. Finalmente, a Figura 3.10c apresenta os mesmos vértices, com os mesmos agrupamentos/divisões em formato de dendograma. Ainda, esta Figura apresenta setas auxiliares exemplificando o processo das abordagens aglomerativas e divisivas.

3.4.2 Baseados em Geodésicas

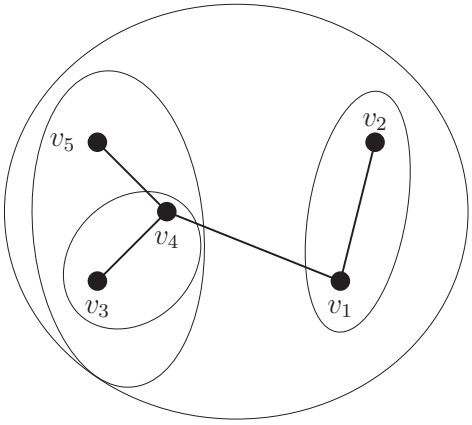
Como discutido na seção anterior, algoritmos hierárquicos comumente utilizam estratégias de geração de árvores mínimas para então remover arestas cujos valores de dissimilaridade são muito altos. Por outro lado, algoritmos baseados em geodésicas não necessitam deste pré-processamento, e utilizam o conceito de grau de centralidade para arestas, determinando quais devem ser removidas. O conceito de grau de centralidade de intermediação para arestas empregado neste tipo de algoritmo é derivado do apresentado na Seção 3.2.2, contudo, o grau de centralidade de intermediação para arestas é calculado a partir do número de geodésicas que passa por cada aresta ao invés de vértices.

A hipótese deste tipo de algoritmo é de que arestas com alto grau de intermediação estão entre comunidades e logo, para determinar estas comunidades, elas devem ser removidas de maneira iterativa, até que um critério da parada seja satisfeito.

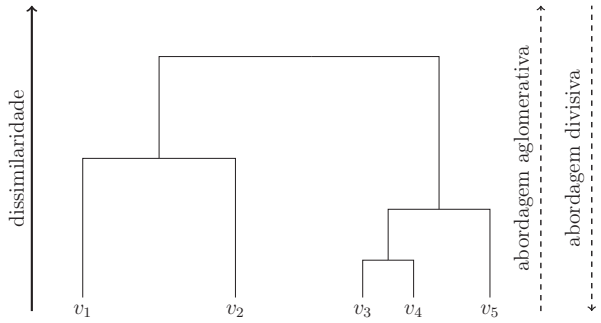
O mais conhecido algoritmo é o de Girvan-Newman (NEWMAN; GIRVAN, 2004). A forma geral deste algoritmo é:



(a) Rede inicial.



(b) Árvore geradora mínima e vértices hierarquicamente agrupados.



(c) Representação do agrupamento em forma de dendograma.

Figura 3.10: Exemplo de agrupamento hierárquico de grupos.

1. Computar o grau de intermediação para cada aresta na rede;
2. Remover a aresta com maior grau de intermediação;
3. Recomputar o grau de intermediação para as arestas remanescentes;
4. Caso o critério de parada não tenha sido satisfeito, repetir o processo a partir do passo 2.

Utilizando algoritmos de complexidade reduzida (BRANDES, 2001) para computar o grau de intermediação de cada aresta, cada repetição deste algoritmo possui complexidade $\mathcal{O}(VE)$, logo, realizar isso para cada aresta, o tornaria muito custoso, com complexidade $\mathcal{O}(VE^2)$, sendo impraticável grandes redes ($V > 10.000$) (SHIOKAWA; FUJIWARA; ONIZUKA, 2013) ou em processos de tempo real, como o de Agrupamento *Online*.

Assim como no problema de agrupamento, a avaliação das comunidades obtidas é um dos maiores problemas para a Detecção de Comunidades. Logo, introduziu-se o conceito de **modularidade**. Considerando uma rede $\mathcal{G}$ dividida em K comunidades,

define-se uma matriz de ordem $K \times K$ onde cada elemento e_{ij} é a fração de arestas de $\mathcal{E}$ que liguem vértices das comunidades k_i e k_j. Ainda, assume-se $a_i = \sum_{k_j}^{\mathscr{K}} e_{ij}$ como o número de arestas que ligam vértices pertencentes a uma comunidade k_i. Em redes onde arestas ligam vértices independentemente de suas comunidades têm-se $e_{ij} = a_i \times a_j$. A modularidade Q pode ser computada através da Equação 3.11.

$$Q = \sum_{k_i}^{\mathscr{K}} (e_{ij} - a_i^2) \tag{3.11}$$

A modularidade mede a fração de arestas na rede que conectam vértices inter-comunidades menos o valor esperado da mesma quantidade em uma rede com as mesmas divisões de comunidades com conexões aleatórias. Caso o número de arestas intra-comunidade não seja melhor que em redes aleatórias, têm-se $Q = 0$. Valores que aproximam $Q = 1$, o máximo, indicam forte estrutura de comunidade. Na prática, redes reais apresentam valores de modularidade no intervalo $[0,3; 0,7]$ (NEWMAN; GIRVAN, 2004).

Assim como métricas de avaliação internas de agrupamentos, a métrica de modularidade é comumente utilizada para determinar um critério de parada para os algoritmos de detecção de comunidades. Ainda, diversos algoritmos tentam otimizar o valor de modularidade para detectar comunidades (BANSAL; BHOWMICK; PAYMAL, 2011; RAVASZ et al., 2002; SHIOKAWA; FUJIWARA; ONIZUKA, 2013; YANG; LIU, 2006), contudo, tal otimização é um problema de natureza NP-Difícil (FORTUNATO, 2010).

3.4.3 Espectrais

Algoritmos espectrais transformam o conjunto inicial de vértices $\mathcal{V}$ em um conjunto de pontos no espaço, cujas coordenadas são elementos de auto-vetores, que posteriormente são agrupados utilizando técnicas convencionais de agrupamento como o algoritmo *k-means*. O diferencial deste tipo de método, quando comparado com métodos que utilizam métricas de similaridade convencionais, é que a representação induzida pelos auto-vetores fazem com que as características dos grupos se tornem mais evidentes (FORTUNATO, 2010) devido à menor dimensionalidade d.

A matriz Laplaciana é a matriz mais comumente utilizada para detecção de comunidades em redes. A Equação 3.12 apresenta o cálculo da matriz Laplaciana L, onde D é a matriz diagonal onde cada posição l_{ii} apresenta o grau de um vértice v_i e S é a matriz simétrica de similaridades/dissimilaridades de $\mathcal{G}$.

$$L = D - S \tag{3.12}$$

Com a matriz L computada, variações de algoritmos como *k-means* são utilizadas, encontrando então as comunidades presentes na rede. Aplicações de algoritmos espectrais são apresentadas em (LIU et al., 2013; LUXBURG, 2007).

3.4.4 Baseados em Passeio Aleatório

Passeios aleatórios também podem ser utilizados para encontrar comunidades. Caso uma rede possua forte estrutura de comunidade, um passeio aleatório passará boa parte do tempo dentro de uma comunidade devido a alta densidade derivada das arestas desta comunidade (FORTUNATO, 2010).

Contudo, passeios aleatórios são comumente utilizados para determinar a distância entre dois vértices em uma rede. Como vértices que participam de uma mesma comunidade tendem a possuir distâncias menores entre si, espera-se que tais distâncias (em número de saltos) sejam minimizadas. Posteriormente, com tais distâncias computadas para todo par de vértices pertencente a rede, métodos hierárquicos ou baseados em modularidade são comumente utilizados (DONGEN, 2000; WARD, 1963; ZHOU, 2003).

3.5 Considerações Finais

Este capítulo apresentou a Análise de Redes Sociais, incluindo seus conceitos fundamentais e teóricos (sob a perspectiva da Teoria dos Grafos) focando nos principais modelos de rede discutidos na literatura.

Foram apresentados ainda medidas de centralidade, coeficiente de agrupamento e modularidade. Neste ponto, deve-se ressaltar a proximidade entre algoritmos de agrupamento dos algoritmos de Detecção de Comunidades. Como apresentado anteriormente, algoritmos de agrupamento são bastante utilizados para o problema de Detecção de Comunidades (BANSAL; BHOWMICK; PAYMAL, 2011; RAVASZ et al., 2002; SHIOKAWA; FUJIWARA; ONIZUKA, 2013; YANG; LIU, 2006). Por outro lado, existem ainda trabalhos que realizam a tarefa de agrupamento utilizando conceitos de Detecção de Comunidades e Teoria dos Grafos (DONGEN, 2000; NEWMAN; GIRVAN, 2004; ZHOU, 2003; ZHONG; MIAO; WANG, 2010). Não obstante, todos estes trabalhos assumem que o grafo é estático, ou no melhor dos casos, que apenas as arestas apareçam e desapareçam com o tempo. Ainda, estes algoritmos possuem complexidade quadrática ou exponencial, se tornando inaplicáveis ao problema de Agrupamento *Online* devido a restrição de tempo de processamento.

O capítulo seguinte apresenta o algoritmos propostos para a tarefa de Agrupa-

mento *Online*, baseados na Teoria de Redes Sociais, mais especificadamente no conceito de homofilia. Estes algoritmos independem da existência de uma rede inicial e permite que vértices sejam adicionados e removidos de acordo com a evolução do fluxo de dados.

Capítulo 4

O Método

Uma das principais limitações dos algoritmos de Agrupamento *Online* é a existência de uma etapa *offline*, responsável por definir os grupos a partir de sumários estatísticos. Idealmente, um algoritmo de Agrupamento *Online* deve ser capaz de atualizar incrementalmente não apenas os seus sumários estatísticos, mas também os grupos, eliminando a necessidade de um processamento em lote (*batch*) na etapa *offline*.

Na etapa *offline*, algoritmos como *k-means* e DBSCAN são comumente utilizados e geram forte dependências perante seus parâmetros. No caso do algoritmo *k-means*, além de ser capaz apenas de encontrar grupos hiper-esféricos, é inviável assumir que o usuário conhece o domínio do problema e que este será capaz de definir corretamente o número de grupos a serem encontrados. Outra principal limitação se deve ao fator temporal intrínseco de fluxos de dados, onde grupos podem aparecer e desaparecer, fenômeno denominado *evolução de conceito*. Uma abordagem comum em implementações é alimentar o algoritmo *k-means* de maneira "supervisionada", ou seja, informar o número de grupos verdadeiros no momento de cada avaliação.

Por outro lado, a principal limitação da utilização do algoritmo DBSCAN é sua quantidade de parâmetros e sua influência nos resultados obtidos. Variações leves nos parâmetros de densidade podem fazer com que todos os sumários estatísticos gerem um número demasiadamente grande de grupos ou que nenhum grupo seja encontrado. Logo, duas possibilidades são apresentadas: assumir que o usuário é especialista no domínio do problema e que este será capaz de definir os parâmetros de maneira correta; ou utilizar algoritmos de otimização para definir valores de parâmetros sub-ótimos para cada problema.

Todas as considerações sobre parametrização acima citadas não são desejáveis e muitas vezes não são aplicáveis no ambiente de fluxos de dados: assumir que o usuário conhece o domínio é utópico, informar o número de grupos verdadeiros distorce o propósito

da tarefa de agrupamento e utilizar algoritmos de otimização pode encontrar bons valores para certos *chunks* do fluxo de dados, mas não se pode assumir que tais valores implicarão em bons resultados nos demais.

Deste modo, espera-se que um algoritmo de Agrupamento *Online* seja capaz de encontrar grupos de maneira completamente não supervisionada e com número limitado de parâmetros, lidando ainda com restrições de tempo e memória, encontrando grupos não hiper-esféricos, detectando e se adaptando a mudanças de conceito e discernindo entre novos grupos e dados ruidosos (AGGARWAL et al., 2003; AMINI; WAH, 2014; CAO et al., 2006; GAMA, 2010; KRANEN et al., 2011; SILVA et al., 2013).

A hipótese dos algoritmos propostos é que, instâncias e *micro-clusters*, além de serem atualizados incrementalmente de acordo com a chegada de instâncias, podem também ser divididos em grupos de maneira incremental, utilizando um modelo de evolução de rede inspirado em redes sociais. Deste modo, utilizando o conceito de **homofilia**, instâncias e *micro-clusters*, ao realizarem religações, são capazes de definir grupos sem a necessidade de algoritmos em formato de lote como *k-means* e DBSCAN. Ressalta-se neste ponto que os algoritmos propostos são **inspirados** em conceitos sobre redes sociais apresentados no Capítulo 3, contudo, dois são essenciais: **religação** e **homofilia**. Ainda, estes dois conceitos são adaptados para o problema de aprendizagem de máquina, logo, diferem dos convencionais provindos da teoria de redes sociais.

Neste capítulo são apresentados dois algoritmos desenvolvidos com o objetivo de realizar a tarefa de Agrupamento *Online*. A Seção 4.1 discute o algoritmo CNDenStream enquanto a Seção 4.2 apresenta sua evolução, o algoritmo SNCStream. Finalmente, a Seção 4.4 apresenta as considerações finais acerca dos algoritmos propostos.

4.1 O Algoritmo CNDenStream

O algoritmo *Complex Network DenStream* (CNDenStream) é baseado na hipótese que dados intra-grupos são relacionados devido a sua alta similaridade (ou baixa dissimilaridade) e dados inter-grupos não são relacionados devido a sua baixa similaridade (ou alta dissimilaridade). CNDenStream modela este problema como uma rede social de potenciais *micro-clusters*, onde o conjunto de vértices $\mathcal{V}$ representa potenciais *micro-clusters*, arestas $\mathcal{E}$ representam conexões entre estes vértices e o conjunto $\mathcal{W}$ representa as distâncias associadas às arestas em $\mathcal{E}$. Ainda, subgrupos de vértices $S \subseteq \mathcal{V}$ dentro da rede representam os grupos que formam o agrupamento $\mathscr{K}$. Para manter grupos atualizados de maneira *online* satisfazendo a propriedade *anytime*, o algoritmo CNDenStream utiliza um processo de formação e evolução de redes sociais onde religações são realizadas

baseando-se no conceito de **homofilia**.

O algoritmo CNDenStream está divido em dois passos: a **Derivação dos Primeiros *Micro-clusters* e Construção Inicial da Rede** e a **Evolução da Rede**. Ambos estes passos estão localizados na etapa *online* do algoritmo.

4.1.1 Derivação dos Primeiros *Micro-clusters* e Construção Inicial da Rede

O passo de derivação e construção inicial da rede tem como objetivo estender o passo de geração de *micro-clusters* iniciais do algoritmo DenStream. O algoritmo CNDenStream inicia obtendo as primeiras $\mathcal{N}$ instâncias do fluxo de dados $\mathcal{S}$ e armazenando-as em um *buffer*.

Após a obtenção destas primeiras $\mathcal{N}$ instâncias, o algoritmo CNDenStream executa o algoritmo DBSCAN, para a geração de potenciais e *outlier micro-clusters*. Os *outlier micro-clusters* são armazenados em um *buffer* armazenado em memória secundária $\mathcal{B}$ enquanto os potenciais *micro-clusters* geram uma rede social utilizando um processo de inserção e evolução baseado em **homofilia**, permitindo que o algoritmo CNDenStream seja capaz de manter grupos (*clusters*) atualizados após a chegada de cada instância e não apenas após a execução de um algoritmo *batch* localizado na etapa *offline*.

Após a execução do algoritmo DBSCAN, todos os potenciais *micro-clusters* PMC_i encontrados são adicionados na rede, estabelecendo conexões (arestas) com os ω *micro-clusters* mais próximos em $\mathcal{V}$. Após a inserção de um PMC_i a $\mathcal{V}$, as novas arestas estabelecidas e seus pesos são adicionados aos seus respectivos conjuntos: $\mathcal{E}$ e $\mathcal{W}$.

O processo de inserção é capaz de conectar o último vértice adicionado aos ω vizinhos mais próximos na rede $\mathcal{G}$, contudo, o mesmo não pode ser dito para os outros vértices previamente existentes em $\mathcal{G}$.

Deste modo, o algoritmo CNDenStream utiliza um processo de religação baseado em homofilia. Após a inserção de um vértice PMC_i na rede, todos os demais vértices $PMC_j \in \mathcal{V}$ substituirão as suas arestas de alta dissimilaridade por arestas com novos vizinhos, minimizando tais distâncias. Para este procedimento assume-se uma distância de salto 2, com a hipótese de que tais vizinhos sejam mais próximos que os vizinhos atuais (salto 1).

Este processo será formalizado e detalhado na Seção 4.2.1 por se repetir no algoritmo SNCStream. Contudo, ressalta-se que, devido a este processo de religação, comunidades de potenciais *micro-clusters* tendem a aparecer naturalmente, uma vez que o número de arestas intra-grupos é maximizada ao contrário do número de arestas inter-grupos.

4.1.2 Evolução da Rede

Após a execução do algoritmo DBSCAN e a construção inicial da rede, todas as instâncias $\vec{x}_i$ obtidas de $\mathcal{S}$ são tratadas com uma adaptação do algoritmo DenStream. Primeiramente, o algoritmo CNDenStream encontra o potencial *micro-cluster* em $\mathcal{V}$ que minimiza a dissimilaridade com $\vec{x}_i$: PMC_i. Posteriormente, verifica-se se o incremento de PMC_i com $\vec{x}_i$ resulta em um *micro-cluster* cujo raio seja menor que ϵ. Em caso positivo, $\vec{x}_i$ é utilizado para incrementar PMC_i. Caso contrário, o mesmo processo se repete para *outlier micro-clusters*: o $OMC_o \in \mathcal{B}$ mais próximo é encontrado ao computar distâncias pivotando $\vec{x}_i$. Logo, verifica-se se seu incremento em OMC_o resulta em um *micro-cluster* de raio menor que ϵ, permitindo e realizando então o incremento de OMC_o com $\vec{x}_i$. No último caso, isto é, $\vec{x}_i$ não incrementou nenhum *micro-cluster* existente, ele é utilizado para instanciar um novo *outlier micro-cluster* que será persistido em $\mathcal{B}$.

Quando um OMC_o arbitrário é promovido a um potencial *micro-cluster*, isto é, $w(OMC_o) \geq \beta\psi$, ele é removido do *buffer* $\mathcal{B}$ e inserido na rede utilizando o processo de inserção e religação apresentados na seção anterior.

Como no algoritmo DenStream, o peso dos *micro-clusters* $w(\cdot)$ são atualizados de acordo com uma função de decaimento exponencial. Quando o peso $w(\cdot)$ de um *micro-cluster* estiver abaixo do limiar $\beta\psi$, ele é removido da rede ou do *buffer* $\mathcal{B}$, dependendo do seu tipo.

Este passo será formalizado em forma de pseudocódigo e discutido em maiores detalhes na Seção 4.2.3, pois é idêntico ao utilizado no algoritmo SNCStream.

4.1.3 Considerações acerca do algoritmo CNDenStream

O algoritmo CNDenStream apresentado nesta primeira seção é um algoritmo baseado na teoria de redes sociais e que utiliza conceitos de densidade para encontrar e atualizar grupos de forma completamente incremental, i.e. sem necessidade de agrupamento *batch* na etapa *offline*. Ainda, o algoritmo CNDenStream é independente de parâmetros relativos ao número de grupos a ser encontrado.

Contudo, este algoritmo ainda depende de uma execução inicial do algoritmo DBSCAN. Assim como o algoritmo DenStream, caso o resultado de um agrupamento seja requisitado antes da derivação dos primeiros *micro-clusters* pelo algoritmo DBSCAN, o algoritmo CNDenStream será incapaz de apresentar agrupamento algum.

Na próxima seção será apresentado o algoritmo SNCStream. O algoritmo SNCStream tem como objetivo eliminar a necessidade desta execução do algoritmo DBSCAN, deste modo, apresentando resultados de agrupamento mesmo antes da derivação dos

micro-clusters iniciais.

4.2 O Algoritmo SNCStream

O algoritmo *Social Network Clusterer Stream* (SNCStream) é baseado na mesma hipótese do algoritmo CNDenStream: de que dados intra-grupos são relacionados devido a sua alta similaridade (ou baixa dissimilaridade) e dados inter-grupos não são relacionados devido a sua baixa similaridade (ou alta dissimilaridade). Estendendo o algoritmo CNDenStream, o algoritmo SNCStream modela este problema como uma rede social, onde o conjunto de vértices $\mathcal{V}$ representa instâncias ou *micro-clusters* (dependendo do passo em que o algoritmo se encontra), arestas $\mathcal{E}$ representam conexões entre estes vértices e o conjunto $\mathcal{W}$ representa as distâncias associadas às arestas em $\mathcal{E}$. Novamente, subgrupos de vértices $S \subseteq \mathcal{V}$ dentro da rede representam os grupos que formam o agrupamento $\mathscr{K}$. Assim como o algoritmo CNDenStream, o algoritmo SNCStream é capaz de satisfazer a propriedade *anytime* ao utilizar um modelo de rede social baseado em homofilia.

A Figura 4.1 apresenta o fluxograma do algoritmo SNCStream. O algoritmo SNCStream está dividido em três passos: a **construção inicial da rede**, a **transformação da rede** e **evolução da rede**. Estes passos são responsáveis por, respectivamente, criar uma estrutura de rede básica e derivar *micro-clusters* iniciais; transformar a rede de instâncias em uma rede de *micro-clusters*; e evoluir a rede baseando-se nestes *micro-clusters*. Ainda, deve-se ressaltar que todos estes passos se encontram na etapa *online* do algoritmo.

Na etapa de construção inicial da rede, o algoritmo SNCStream inicia com uma rede $\mathcal{G}$ vazia ($\mathcal{V} = \mathcal{E} = \mathcal{W} = \emptyset$) que será construída com instâncias $\vec{x}_i$ obtidas de $\mathcal{S}$. Ainda, o parâmetro T_p, herdado do algoritmo DenStream, é inicializado de acordo com a Equação 2.19.

Para cada instância $\vec{x}_i$, SNCStream encontra os ω vértices mais próximos computando distâncias Euclidianas, sendo ω um valor definido pelo usuário.

Posteriormente, SNCStream insere $\vec{x}_i$ na rede $\mathcal{G}$ estabelecendo conexões com os ω vizinhos mais próximos escolhidos. Após a inserção de $\vec{x}_i$ na rede $\mathcal{G}$, todos os vértices $v_i \in \mathcal{V}$ realizam o processo de religação. O processo de religação é o responsável por encontrar e manter grupos dentro do algoritmo SNCStream. Devido ao processo de religação, comunidades de instâncias aparecem naturalmente, uma vez que o número de arestas entre instâncias intra-grupos cresce e inter-grupos é eliminada.

O conjunto de passos acima é capaz de encontrar e manter grupos de maneira incremental, contudo, não é prático para o ambiente *online* devido ao grande número de instâncias. Consequentemente, o algoritmo SNCStream utiliza uma janela de tamanho

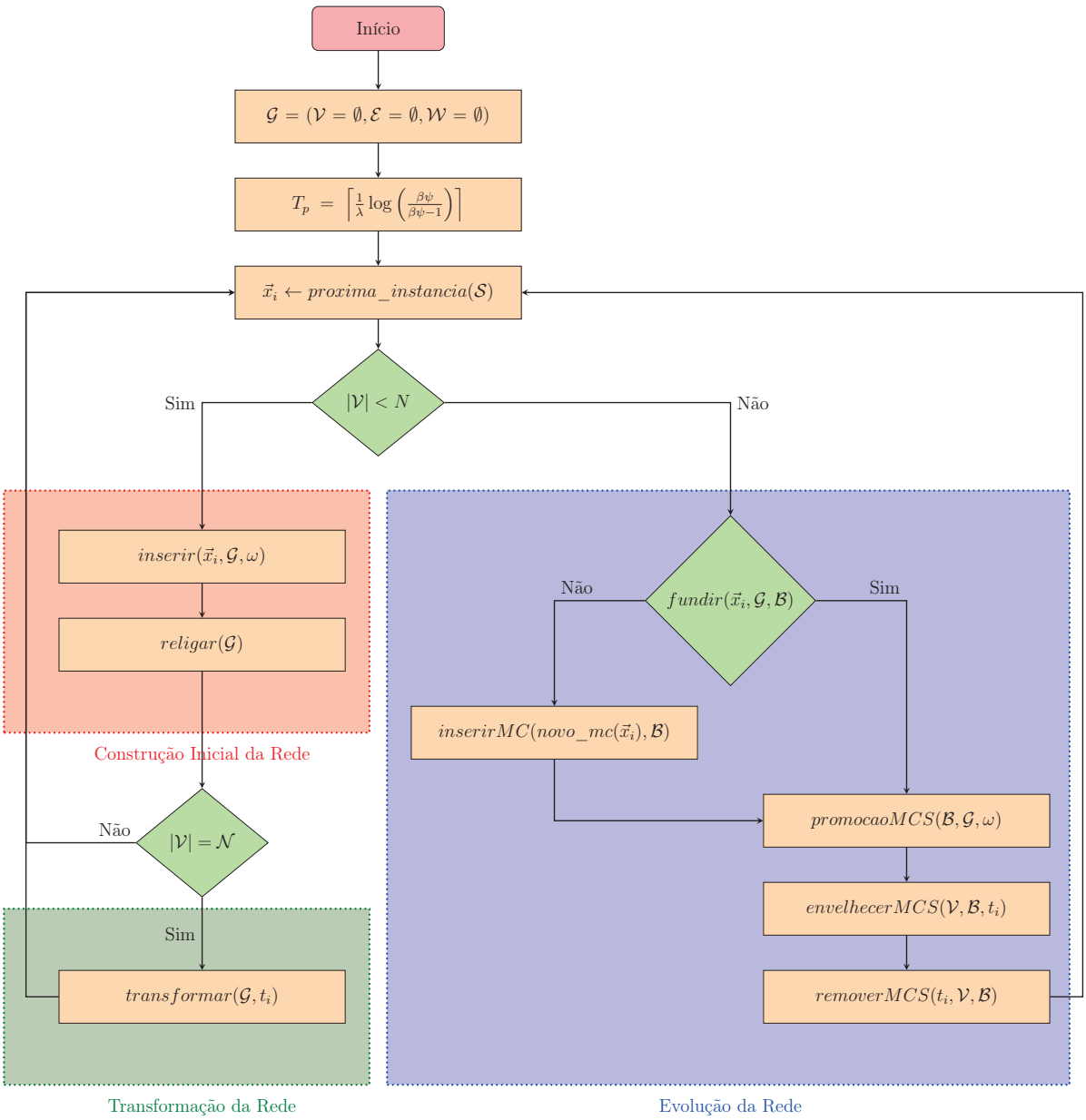
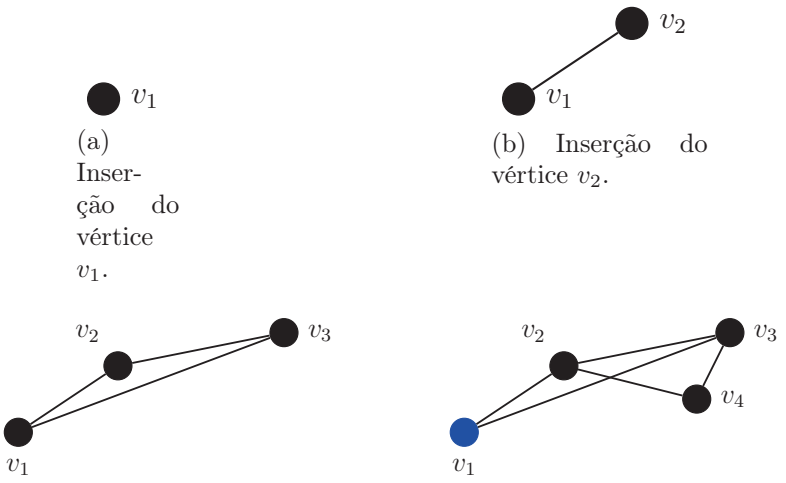


Figura 4.1: Fluxograma do algoritmo SNCStream.

$\mathcal{N}$ que determina o número de instâncias inicial que será utilizado para a transformação da rede, que deixa de tratar instâncias e passa a tratar potenciais *micro-clusters*. Deste modo, a etapa de construção inicial da rede é finalizada, e um processo de transformação ocorre, onde cada instância em $\mathcal{V}$ é substituída por um potencial *micro-cluster*. As etapas de construção inicial e transformação da rede são detalhadas nas Seções 4.2.1 e 4.2.2, respectivamente.

Após a transformação da rede, o processo existente no passo de construção inicial da rede é adaptado para tratar a massiva quantidade de dados vindas do fluxo de dados $\mathcal{S}$ ao utilizar *micro-clusters*. Este passo é denominado evolução da rede e é repetido para o restante do fluxo de dados. Este passo final será discutido na Seção 4.2.3.



(a) Inserção do vértice v_1.

(b) Inserção do vértice v_2.

(c) Inserção do vértice v_3.

(d) Inserção do vértice v_4.

Figura 4.2: Exemplo de processo de inserção usando $\omega = 2$.

As próximas seções apresentam detalhadamente cada um dos passos do algoritmo SNCStream.

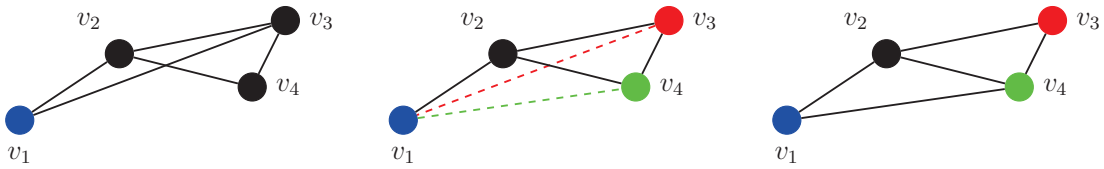
4.2.1 A Construção Inicial da Rede

O passo de construção inicial tem como objetivo derivar uma rede baseando-se no início do fluxo de dados. O algoritmo SNCStream inicia com uma rede $\mathcal{G}$ vazia $\mathcal{V} = \mathcal{E} = \mathcal{W} = \emptyset$. Baseando-se em um parâmetro de janela inicial $\mathcal{N}$, instâncias $\vec{x}_i$ são obtidas do fluxo de dados $\mathcal{S}$ e inseridas na rede $\mathcal{G}$.

O procedimento de inserção de uma instância $\vec{x}_i$ é baseada em um parâmetro que define o número de conexões a serem realizadas: ω. Deste modo, o algoritmo SNCStream computa distâncias euclidianas entre $\vec{x}_i$ e todas as instâncias $\vec{v}_j \in \mathcal{V}$, determinando assim as ω instâncias mais próximas. Posteriormente, o algoritmo SNCStream insere $\vec{x}_i$ em $\mathcal{V}$, assim como as ω arestas novas e seus pesos (distâncias Euclidianas) em seus respectivos conjuntos $\mathcal{E}$ e $\mathcal{W}$. O Algoritmo 1 apresenta o pseudocódigo para o procedimento de inserção de uma instância.

O procedimento de inserção garante com que a última instância esteja conectada com as ω instâncias mais próximas existentes na rede, contudo, não garante que todas as demais instâncias estejam. Para exemplificar este problema, remete-se a Figura 4.2, onde é apresentada a inserção passo a passo de 4 vértices. Atenta-se em especial a Figura 4.2d, onde pode-se ver que o vértice v_1 não está conectado com os ω melhores vértices mais próximos adotando uma distância Euclidiana.

Ainda, o procedimento de inserção por si só forma redes conexas. Assim como discutido na Seção 3.4, grafos comumente dependem de algoritmos em formato de lote para



(a) Religação do vértice v_1. (b) Comparação de distâncias, onde $d(v_1, v_4) < d(v_1, v_3)$. (c) vértice v_1 devidamente religado.

Figura 4.3: Exemplo de religação de v_1, onde $deg(v_1) = 2$.

detectar comunidades, logo, não são facilmente aplicáveis ao problema de Agrupamento *Online*.

Deste modo, o algoritmo SNCStream aplica um procedimento de religação baseado em homofilia apresentado no Algoritmo 2. Este procedimento de religação possui dois objetivos: promover que vértices estejam conectados com os vértices mais próximos e segmentar a rede em subredes (grupos). Para isso, o processo de religação é baseado na hipótese de que os vizinhos de um vértice v_j tendem a ser vizinhos de v_i caso v_i seja vizinho de v_j. Em outras palavras, os vértices destino v_j de caminhamentos de tamanho 2 partindo de v_i ($|P(v_i, v_j)| = 2$) possuem grande probabilidade de se tornarem um dos ω vértices mais próximos. Para exemplificar o procedimento de religação, a Figura 4.3 apresenta o exemplo iniciado na Figura 4.2. A Figura 4.3 apresenta o processo de religação do vértice v_1. Neste caso, v_1 encontrará seus vizinhos baseando em um caminhamento de distância 2, realizará cálculos de distância com estes, determinando quais apresentam menor dissimilaridade em relação aos seus vizinhos atuais. Na Figura 4.3b pode-se ver que $d(v_1, v_4) < d(v_1, v_3)$, logo, v_1 deixa de se conectar com v_3 para estabelecer uma aresta com v_4 (vide Figura 4.3c).

Diferentes números de saltos também podem ser aplicados, contudo, impactam em maior custo computacional por aumentar o número de distâncias a serem calculadas. Como será apresentado no Capítulo 5, os resultados obtidos utilizando um salto de tamanho dois é viável tanto em termos de acuidade quanto de tempo de processamento nos domínios avaliados.

De maneira genérica, quando um vértice v_i realizar seu procedimento de religação, ele primeiramente encontrará seus vizinhos baseando-se em caminhamentos de tamanho 2, determinará quais destes possuem dissimilaridade inferior aos seus vizinhos atuais, realizando assim a substituição das arestas que estabelecem conexões com estes vértices, logo, v_i deve sempre manter seu grau, $deg(v_i)$. Isso implica que alguns vértices, por serem mais representativos dentro de seus grupos se tornam (*hubs*) e por outro lado, vértices que representam as extremidades (bordas) dos grupos possuem grau baixo, exatamente por

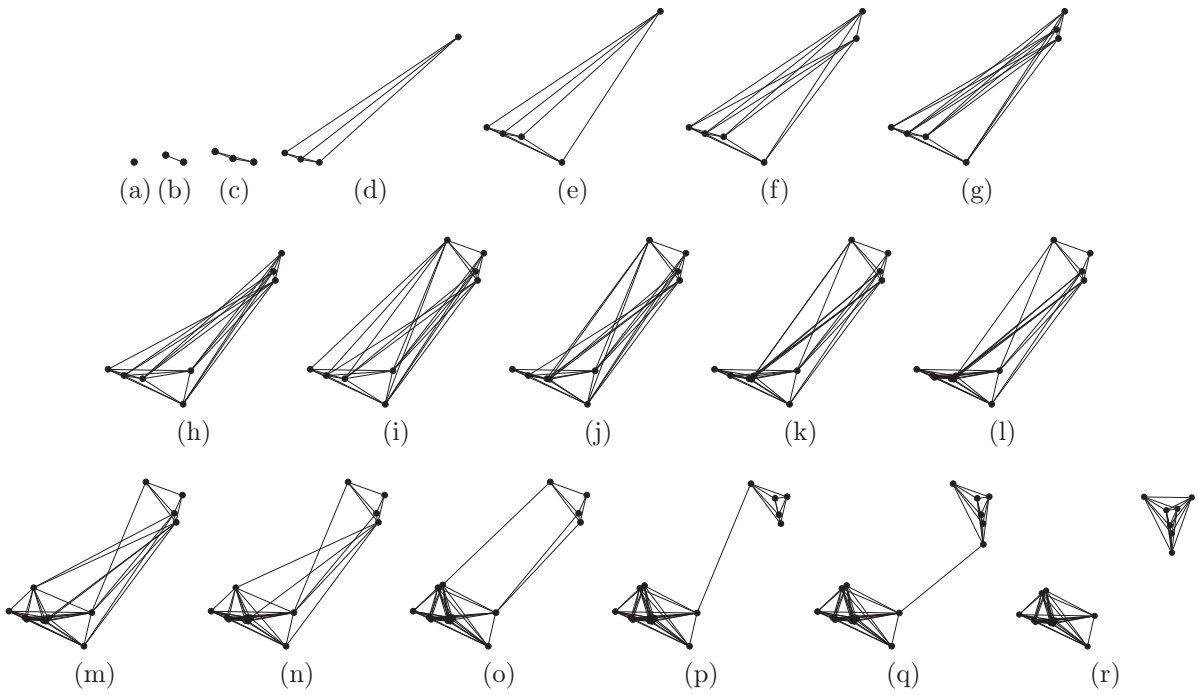


Figura 4.4: Exemplo de construção inicial da rede onde dois grupos são formados utilizando o processo de religação.

desfazer conexões com vértices mais afastados presentes em outros grupos. Um exemplo disto ocorre ainda na Figura 4.3, onde o vértice v_3 perde uma conexão, logo, seu grau diminui; e o vértice v_4 recebe uma nova conexão, logo, seu grau aumenta.

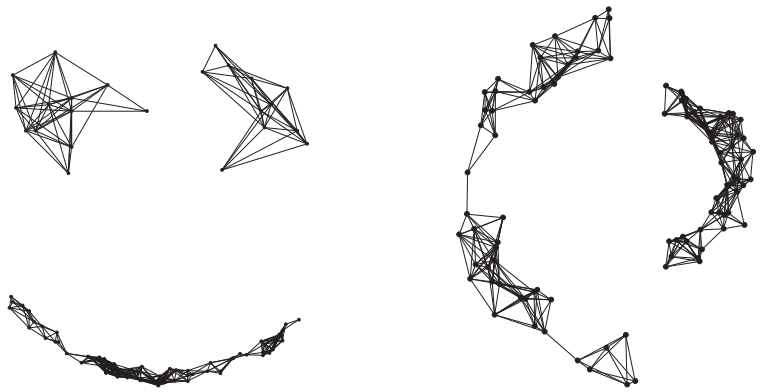
Ainda, o processo de religação possui uma propriedade: tende a dividir a rede. A Figura 4.5 apresenta um exemplo de evolução de rede onde dois grupos hiper-esféricos emergem naturalmente, devido apenas ao procedimento de religação.

Ao contrário do que a Figura 4.5 possa indicar, o processo de religação permite que o algoritmo SNCStream seja capaz de encontrar grupos não hiper-esféricos. A Figura 4.5 apresenta exemplos de redes obtidas para um problema onde grupos são não hiper-esféricos.

4.2.2 A Transformação da Rede

Apenas os procedimentos de inserção e religação são capazes de encontrar grupos não hiper-esféricos de maneira incremental. Contudo, eles possuem uma forte limitação: a rede apenas cresce. Este é um impedimento que torna o algoritmo inviável para o ambiente *online*, seja pela impossibilidade de armazenar o fluxo de dados inteiro em memória ou pela incoerência de realizar cálculos de distâncias Euclidianas para cada instância $\vec{x}_i$ recém-obtida de $\mathcal{S}$.

Logo, o algoritmo SNCStream utiliza estruturas de sumarização estatística para



(a) (b)

Figura 4.5: Exemplos de construção inicial da rede onde grupos não hiper-esféricos são formados utilizando o processo de religação.

Algoritmo 1: Pseudocódigo do procedimento $inserir(\vec{x}_i, \mathcal{G}, \omega)$

Entrada: uma instância $\vec{x}_i$, a rede $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ e o número de conexões a serem criadas ω.

[1] $vizinhos \leftarrow \omega$ vértices mais próximos a $\vec{x}_i$ em $\mathcal{V}$ ordenados crescentemente de acordo com uma distância Euclidiana $d(\cdot, \vec{x}_i)$;
[2] $\mathcal{V} \leftarrow \mathcal{V} \cup \{\vec{x}_i\}$;
[3] **para cada** $\vec{x}_j \in vizinhos$ **faça**
[4] $\quad novaAresta \leftarrow nova\_aresta(\vec{x}_i, \vec{x}_j)$;
[5] $\quad peso \leftarrow d(\vec{x}_i, \vec{x}_j)$;
[6] $\quad \mathcal{E} \leftarrow \mathcal{E} \cup \{novaAresta\}$;
[7] $\quad \mathcal{W} \leftarrow \mathcal{W} \cup \{peso\}$;

promover o "esquecimento" de dados mais antigos, assim como diminuir a quantidade de dados em memória. Para tal, o algoritmo SNCStream utiliza uma janela inicial de tamanho $\mathcal{N}$. Quando o número de vértices $|\mathcal{V}|$ na rede atinge $\mathcal{N}$, todos estes vértices $v_i \in \mathcal{V}$, que até este momento representavam instâncias, passarão a representar *micro-clusters*.

Como discutido na Seção 2.4.1, *micro-clusters* são sumários de dados na forma de *micro-clusters* CMCs. *Micro-clusters* são capazes de representar grupos densos de instâncias recém-obtidas do fluxo de dados $\mathcal{S}$ e possuem duas propriedades importantes: **incrementalidade** e **adicionalidade**.

Adota-se para este trabalho *micro-clusters* na forma $CMC = (LS, SS, N, W, t_c, t_u)$, onde LS é a soma de todas as instâncias sumarizadas, SS é a soma do quadrado destas instâncias, N a quantidade de instâncias, W é o seu peso, t_c o seu instante de criação e t_u o instante de sua última atualização.

Para transformar a rede de instâncias para uma rede de *micro-clusters*, o algoritmo SNCStream substitui a estrutura interna de seus vértices, que armazenam instâncias, por

Algoritmo 2: Pseudocódigo do procedimento $religar(\mathcal{G})$

Entrada: a rede $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$.

[1] **para cada** $\vec{x}_i \in \mathcal{V}$ **faça**
[2] $\quad \omega_i \leftarrow deg(\vec{x}_i)$;
[3] $\quad novosVizinhos \leftarrow$ os ω_i vizinhos mais próximos a $\vec{x}_i$ em uma vizinhança de salto 2;
[4] $\quad$ Remove todas as arestas conectando $\vec{x}_i$ em $\mathcal{E}$ e seus respectivos pesos em $\mathcal{W}$;
[5] $\quad$ **para cada** $\vec{x}_j \in novosVizinhos$ **faça**
[6] $\quad\quad novaAresta \leftarrow nova\_aresta(\vec{x}_i, \vec{x}_j)$;
[7] $\quad\quad peso \leftarrow d(\vec{x}_i, \vec{x}_j)$;
[8] $\quad\quad \mathcal{E} \leftarrow \mathcal{E} \cup \{novaAresta\}$;
[9] $\quad\quad \mathcal{W} \leftarrow \mathcal{W} \cup \{peso\}$;

Algoritmo 3: Pseudocódigo do procedimento $transformar(\mathcal{G}, t_i)$

Entrada: a rede $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ e o instante atual t_i.

[1] **para cada** $\vec{x}_i \in \mathcal{V}$ **faça**
[2] $\quad$ Transforma $\vec{x}_i$ em um PMC com $LS = \vec{x}_i$, $SS = (\vec{x}_i)^2$, $N = 1$, $W = 1$ e $t_c = t_u = t_i$;
[3] $\mathcal{B} \leftarrow \emptyset$;

potenciais *micro-clusters* com $LS = \vec{x}_i$, $SS = (\vec{x}_i)^2$, $N = 1$ e $W = 1$, $t_c = t_u = t_i$.

Ainda, neste passo de transformação, um *buffer* para *outlier micro-clusters* $\mathcal{B}$ é inicializado. Assim como no algoritmo DenStream, este *buffer* tem como objetivo armazenar *micro-clusters* cuja densidade não é suficiente para afetarem os grupos.

O Algoritmo 3 apresenta o pseudocódigo do passo de transformação da rede.

4.2.3 A Evolução da Rede

Após a transformação da rede em uma rede de *micro-clusters*, o algoritmo SNCStream atualiza seus sumários estatísticos CMCs baseando-se no algoritmo DenStream. O Algoritmo 4 apresenta o processo de incremento de uma instância em um *micro-cluster* existente.

Ao receber uma instância $\vec{x}_i$ de $\mathcal{S}$, o algoritmo SNCStream computa distâncias euclidianas entre $\vec{x}_i$ e o centro dos PMCs que compõe o conjunto de vértices da rede $\mathcal{V}$, encontrando o mais próximo: PMC_i. Verifica-se então se a adição de $\vec{x}_i$ em PMC_i resulta em um *micro-cluster* cujo raio é inferior a ϵ. Caso isso ocorra, $\vec{x}_i$ é adicionado a PMC_i. No caso negativo, o processo se repete para os *outlier micro-clusters*: encontra-se o *outlier micro-cluster* OMC_i mais próximo a $\vec{x}_i$ e se o raio da adição desta instância gera um CMC com raio inferior a ϵ. No caso positivo, $\vec{x}_i$ é adicionado a OMC_i.

Algoritmo 4: Pseudocódigo do procedimento $fundir(\vec{x}_i, \mathcal{G}, \mathcal{B})$

Entrada: uma instância $\vec{x}_i$, a rede $\mathcal{G} = (\mathcal{G}, \mathcal{E}, \mathcal{W})$ e o conjunto de *outlier micro-clusters* $\mathcal{B}$.

```
[1] fundiu ← FALSO;
[2] PMC_i ← argmin_{PMC_i∈V} d(x_i, c(PMC_i));
[3] se r(PMC_i + x_i)+ ≤ ε então
[4]     PMC_i ← PMC_i + x_i;
[5]     fundiu ← VERDADEIRO;
[6] se não fundiu então
[7]     OMC_i ← argmin_{OMC_i∈B} d(x_i, c(OMC_i));
[8]     se r(OMC_i + x_i)+ ≤ ε então
[9]         OMC_i ← OMC_i + x_i;
[10]        fundiu ← VERDADEIRO;
[11] retorne fundiu;
```

Algoritmo 5: Pseudocódigo do procedimento $inserirMC(OMC_i, \mathcal{B})$

Entrada: um novo *outlier micro-cluster* OMC_i e o conjunto de *outlier micro-clusters* $\mathcal{B}$.

```
[1] B ← B ∪ {OMC_i};
```

Caso o processo $fundir$ não tenha adicionado $\vec{x}_i$, $\vec{x}_i$ será utilizado para instanciar um novo *micro-cluster* que será adicionado ao *buffer* de *outlier micro-clusters* $\mathcal{B}$ de acordo com o Algoritmo 5.

Quando o processo de adição for bem sucedido, ou um novo *micro-cluster* tenha sido adicionado a $\mathcal{B}$, um processo de promoção verifica se algum *outlier micro-cluster* se tornou um potencial *micro-cluster*, e em caso positivo, adiciona-os à rede. O Algoritmo 6 apresenta o pseudocódigo do procedimento $promocaoMCS$.

Após o processo de promoção, um processo de envelhecimento atua sobre todos os *micro-clusters* do algoritmo SNCStream. O Algoritmo 7 apresenta o processo de envelhecimento. Primeiramente, SNCStream calcula novos pesos para cada *micro-cluster* CMC_i adotando a Equação 4.1, onde Δt é a diferença entre o instante atual do fluxo de dados (número de instâncias já observadas) t_i e o instante da última atualização em CMC_i: $t_c(CMC_i)$; N é o número de instâncias sumarizadas por CMC_i e λ um parâmetro da forma da exponencial.

$$w(CMC_i) = N \times 2^{-\lambda \times \Delta t} \tag{4.1}$$

Finalmente, o último passo do algoritmo SNCStream é a remoção de *micro-clusters*.

Algoritmo 6: Pseudocódigo do procedimento $promocaoMCS(\mathcal{B}, \mathcal{G}, \omega)$.

Entrada: o *buffer* de *outlier micro-clusters* $\mathcal{B}$, a rede $\mathcal{G} = (\mathcal{G}, \mathcal{E}, \mathcal{W})$ e o número de conexões a serem criadas ω.

```
[1] para cada OMC_i ∈ B faça
[2]     se w(OMC_i) > βψ então
[3]         B ← B − {OMC_i};
[4]         vizinhos ← ω vértices mais próximos a OMC_i em V ordenados
            crescentemente de acordo com uma distância Euclidiana d(·, c(OMC_i));
[5]         V ← V ∪ {OMC_i};
[6]         para cada PMC_j ∈ vizinhos faça
[7]             novaAresta ← nova_aresta(OMC_i, PMC_j);
[8]             peso ← d(c(OMC_i), c(PMC_j));
[9]             E ← E ∪ {novaAresta};
[10]            W ← W ∪ {peso};
[11]        religar(G);
```

Algoritmo 7: Pseudocódigo do procedimento $envelhecerMCS(\mathcal{V}, \mathcal{B}, t_i)$

Entrada: o conjunto de potenciais *micro-clusters* $\mathcal{V}$, o *buffer* de *outlier micro-clusters* $\mathcal{B}$.

```
[1] para cada CMC_i ∈ V ∪ B faça
[2]     Δt ← t_i − t_u(CMC_i);
[3]     w(CMC_i) ← N × 2^{−λ∗Δt};
```

O Algoritmo 8 apresenta o pseudocódigo deste procedimento. Primeiramente, é verificado se o peso $w(\cdot)$ de cada potencial *micro-cluster* PMC_i em $\mathcal{V}$ está abaixo da densidade mínima dada por $\beta\psi$. Caso isso ocorra, um processo de religação ocorre para todos os vizinhos PMC_j de PMC_i. Este processo de religação atua como o processo de religação de instâncias, contudo, ignorando PMC_i da lista de melhores vértices, uma vez que este vértice será removido da rede.

Após a verificação da densidade dos *micro-clusters* em $\mathcal{V}$, o mesmo ocorre para os *outlier micro-clusters* no *buffer* $\mathcal{B}$. Neste caso, o processo é mais simples, onde OMC_i é removido de $\mathcal{B}$.

Uma questão acerca do método é a respeito do efeito que o parâmetro ω possui na construção e evolução da rede. Logo, a Seção 5.6.1 apresenta uma análise empírica acerca dos resultados obtidos variando este parâmetro, determinando então que $k = 4$ é um valor apropriado para os conjuntos de dados avaliados.

Algoritmo 8: Pseudocódigo do procedimento $removerMCS(t_i, \mathcal{G}, \mathcal{B})$

Entrada: o instante atual t_i, a rede $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ e o *buffer* de *outlier micro-clusters* $\mathcal{B}$.

```
[1] se i mod T_p = 0 então
[2]     para cada PMC_i ∈ V faça
[3]         se w(PMC_i) ≤ βψ então
[4]             para cada PMC_j ∈ adj(PMC_i) faça
[5]                 ω_j ← deg(PMC_j);
[6]                 novosVizinhos ← os ω_j micro-clusters mais próximos em V
                    considerando os vizinhos atuais e vizinhos em distância de 2 saltos
                    com exceção de PMC_i;
[7]                 Remove todas as arestas conectando PMC_j em E e seus respectivos
                    pesos em W;
[8]                 para cada PMC_k ∈ novosVizinhos faça
[9]                     novaAresta ← nova_aresta(PMC_k, PMC_j);
[10]                    peso ← d(c(PMC_k), c(PMC_j));
[11]                    E ← E ∪ {novaAresta};
[12]                    W ← W ∪ {peso};
[13]    para cada OMC_i ∈ B faça
[14]        se w(PMC_i) ≤ ξ(t_i, t_c(OMC_i)) então
[15]            B ← B − {OMC_i};
```

4.3 Melhorias nos Algoritmos CNDenStream e SNCStream

A essência dos algoritmos CNDenStream e SNCStream reside no procedimento de religação. Contudo, tal procedimento é relativamente custoso por necessitar realizar computações de distância após a chegada de cada instância ou inserção de um novo potencial *micro-cluster* na rede. Deste modo, dois detalhes sobre o procedimento de religação merecem atenção: (i) o procedimento de religação muito provavelmente computará distâncias entre vértices que já foram comparados previamente; e (ii) ao efetuar a religação após a inserção de um vértice v_i é necessário realizar acesso linear aos demais vértices presentes em $\mathcal{V}$ tendo custo computacional $\mathcal{O}(V-1)$. Ainda, deve-se ter em mente que vértices em outros subgrupos da rede ou cuja distância em número de saltos seja elevada provavelmente não sejam afetados pelo processo de religação, logo, realizar estes acessos lineares é uma abordagem ingênua e que deve ser aprimorada.

Para dirimir o custo computacional dos algoritmos CNDenStream e SNCStream em termos de processamento, duas melhorias são propostas:

1. A utilização de uma estrutura de *hash* para armazenar e reutilizar distâncias entre

vértices da rede (instâncias ou potenciais *micro-cluster*); e

2. A melhoria no procedimento de religação onde o mesmo será realizado via propagação, deste modo, diminuindo o número de vértices verificados desnecessariamente.

Estas melhorias são detalhadas nas seções a seguir.

4.3.1 *Memoization* de Distâncias via *Hashing* e Função de Pareamento de Cantor

Computar distâncias entre pontos no espaço pode ser uma situação problemática, especialmente em espaços de alta dimensionalidade. Os algoritmos propostos neste trabalho, como a maioria dos algoritmos de agrupamento, residem na noção de distância entre pontos no espaço. Devido ao procedimento de religação, distâncias entre tais pontos (vértices) são muitas vezes re-computadas sem necessidade. Para sobrepujar este aspecto, os algoritmos CNDenStream e SNCStream realizam o processo de *memoization*[1] (MICHIE, 1968) para armazenar resultados de distâncias computadas.

O processo de *memoization* proposto tem como objetivo armazenar as distâncias entre pares de vértices em uma tabela *hash* no formato $\langle chave, valor \rangle$, que permite, na teoria, acesso em $\mathcal{O}(1)$. Para a composição de uma chave para indexar a distância entre dois vértices v_i e v_j de índices i e j, utiliza-se a função de pareamento de Cantor $\pi(\cdot, \cdot)$ (ROSENBERG, 2002) apresentada na Equação 4.2, onde $\langle i, j \rangle$ é um número natural composto a partir de i e j. Para utilizar a função de Cantor, assume-se que $i < j$, uma vez que $\pi(i, j) \neq \pi(j, i)$.

$$\pi(i, j) = \langle i, j \rangle = \frac{1}{2} \times (i + j) \times (i + j + 1) + j \tag{4.2}$$

A função de Cantor $\pi(\cdot, \cdot)$, apresentada na Equação 4.2 é invertível, isto é, a partir de um número natural $\langle i, j \rangle$ é possível determinar os valores originais de i e j. Para inverter a função de pareamento de Cantor, isto é, determinar os valores de i e j a partir de $\langle i, j \rangle$, deve-se seguir os seguintes passos:

[1] Processo análogo ao de *tabling* utilizado em sistemas para linguagens de paradigma lógico como Prolog.

$$
\begin{aligned}
z &= \langle i, j \rangle \\
w &= \left\lfloor \frac{\sqrt{8z} - 1}{2} \right\rfloor \\
t &= \frac{w^2 + w}{2} \\
j &= z - t \\
i &= w - j
\end{aligned}
$$

Com a função de pareamento de Cantor, é possível que os algoritmos CNDenStream e SNCStream, após computar distâncias entre dois vértices v_i e v_j da rede, armazenem tais dissimilaridades em uma hash na forma $\langle \pi(\min\{i,j\}, \max\{i,j\}), d(v_i, v_j) \rangle$, permitindo sua recuperação de maneira rápida, sem necessidade de recomputações.

4.3.2 Melhoria no Procedimento de Religação

O procedimento de religação apresentado nas Seções 4.1.1 e 4.2.1 realiza acessos lineares a todos os vértices existentes v_i na rede $\mathcal{G}$. Como discutido anteriormente, este acesso pode ser desnecessário e realizar acessos a vértices que participem de subgrupos diferentes quando comparados ao último vértice inserido ou cujo número de saltos seja alto. Para diminuir o número de acessos aos vértices da rede, esta seção apresenta um novo algoritmo para realizar o procedimento de religação através de propagação.

O Algoritmo 9 apresenta o procedimento de religação otimizado, que recebe como entrada o último vértice inserido v_i e a rede $\mathcal{G}$. Este Algoritmo primeiramente inicia duas estruturas auxiliares (linhas 1 e 2): *religados* (uma lista de vértices já religados) e *religar* (uma fila de vértices que devem realizar o procedimento de religação).

Enquanto existirem nós a serem religados na fila *religar*, repete-se o procedimento de religação convencional apresentado anteriormente com o primeiro vértice removido desta lista v_j. Primeiramente, o algoritmo verifica se os novos vizinhos escolhidos para v_j são diferentes de seus vizinhos atuais. Caso esta condição seja confirmada, ou seja, houve alteração na vizinhança deste vértice v_j, todos os seus vizinhos, com exceção de v_i e outros nós já religados são inseridos na fila para religação.

A Figura 4.6 apresenta um exemplo de rede onde, após a inserção do vértice v_{13}, dois subgrupos não seriam verificados pelo algoritmo de maneira desnecessária, deste

Algoritmo 9: Pseudocódigo do procedimento otimizado $religar(v_i, \mathcal{G})$

Entrada: o último vértice inserido na rede v_i e a rede $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$.

```
[1] religados ← ∅;
[2] religar ← adj(v_i);
[3] enquanto religar ≠ ∅ faça
[4]     v_j ← removePrimeiroElemento(religar);
[5]     vizinhosAtuais ← adj(v_j);
[6]     ω_j ← deg(v_j);
[7]     novosVizinhos ← encontra os ω_j vizinhos mais próximos de v_j assumindo uma
        distância de salto 2;
[8]     se vizinhosAtuais ≠ novosVizinhos então
[9]         Remove todas as arestas conectando v_j em E e seus respectivos pesos em W;
[10]        para cada v_k ∈ novosVizinhos faça
[11]            novaAresta ← nova_aresta(v_k, v_j);
[12]            peso ← d(v_k, v_j);
[13]            E ← E ∪ {novaAresta};
[14]            W ← W ∪ {peso};
[15]        religar ← religar ∪ (adj(v_j) − {v_i} − religados);
[16]    religados ← religados ∪ {v_j};
```

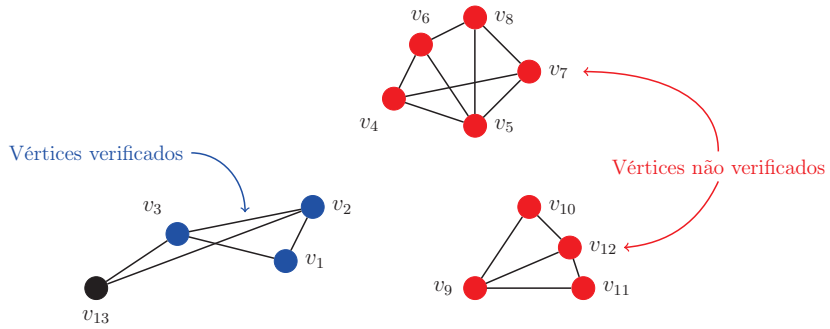


Figura 4.6: Exemplo de procedimento de religação, demonstrando vértices verificados e não verificados após a inserção de um vértice v_{13}.

modo, diminuindo o custo computacional do algoritmo de religação.

4.4 Considerações Finais

Neste capítulo foram apresentados os algoritmos CNDenStream e SNCStream. Estes algoritmos são inspirados em conceitos de redes sociais e objetivam o problema de Agrupamento *Online*. Ao contrário dos demais algoritmos da literatura apresentados, CNDenStream e SNCStream possuem apenas uma etapa, são capazes de encontrar grupos não hiper-esféricos e não necessitam de um parâmetro explícito do número de grupos a serem encontrados.

Com o intuito de avaliar os algoritmos propostos, um protocolo experimental e um ambiente de avaliação empírica foram desenvolvidos de acordo com os principais trabalhos da literatura (AMINI; WAH, 2014; AGGARWAL et al., 2003; AGGARWAL, 2003; KRANEN et al., 2011), contudo, tentando sobrepujar limites em questões de validação estatística. Ainda, uma variedade de experimentos é apresentada com o intuito de discutir o impacto dos principais parâmetros do algoritmo SNCStream, por ser a versão completa do algoritmo proposto.

Capítulo 5

Análise e Avaliação Empírica

Com o intuito de realizar uma avaliação adequada dos algoritmos propostos, objetivando compará-los com os demais algoritmos da literatura, um ambiente de análise e validação foi gerado. Neste ambiente, são utilizados tanto dados sintéticos quanto reais. Esta avaliação é baseada em diversos trabalhos da literatura (AMINI; WAH, 2014; AGGARWAL et al., 2003; AGGARWAL, 2003; CAO et al., 2006; KRANEN et al., 2011), contudo, tenta sobrepujar limitações dos mesmos, tais como a avaliação de métricas não específicas para o ambiente *online* e comparação apenas visual dos resultados, sem rigor estatístico.

Este capítulo está dividido da seguinte maneira: A Seção 5.1 apresenta o *framework Massive Online Analysis*, focando nos aspectos da Agrupamento de Fluxos de Dados. A Seção 5.2 apresenta os geradores de dados utilizados enquanto a Seção 5.3 discute os conjuntos de dados reais. A Seção 5.4 formaliza o protocolo experimental, enquanto a Seção 5.5 discute os resultados preliminares obtidos. Posteriormente, a Seção 5.6 discute acerca dos parâmetros do algoritmo SNCStream e seus respectivos impactos em termos de qualidade de agrupamento, tempo de processamento e uso de memória enquanto a Seção 5.7 apresenta um comparativo do algoritmo SNCStream com os demais da literatura, levando em conta a mesma tríade de métricas. Seguidamente, a Seção 5.8 discute sobre os coeficientes de agrupamentos e os valores de modularidade obtidos durante os experimentos. Finalmente, a Seção 5.9 apresenta as conclusões obtidas deste capítulo.

5.1 O *Framework Massive Online Analysis*

O *Massive Online Analysis* (MOA) é um ambiente de implementação de algoritmos e de execução de experimentos que envolvem fluxos contínuos de dados (BIFET et al., 2010). O MOA foi concebido para lidar com vários dos problemas relativos a fluxos

de dados, desde a geração de dados sintéticos, utilização de conjuntos de dados reais, implementação de algoritmos e avaliação destes. Deste modo, o MOA acaba contendo uma coleção de algoritmos de Aprendizagem de Máquina bastante vasta, principalmente quando usuários fazem uso da interação com o ambiente WEKA[1] (HALL et al., 2009). Ainda, o MOA foi desenvolvido para facilitar o teste de hipóteses para algoritmos de pesquisadores da área de mineração de fluxos de dados.

O MOA possui forte foco para a tarefa de Classificação, contudo, compreende ainda as áreas de Regressão, Detecção de Dados Ruidosos e *Outliers*, Agrupamento e Sistemas de Recomendação em ambiente *online*. Dentro de cada uma destas tarefas, existem diversos geradores de dados, leitores de arquivos de conjuntos de dados i.e. CSV (*Comma Separated Values*) e ARFF (*Attribute-Relation File Format*), métodos de avaliação e demais estatísticas como tempo de processamento e uso de memória.

O MOA foi desenvolvido em Java e foi concebido como um ambiente de fácil extensão, logo, se tornou uma ferramenta bastante difundida na comunidade de mineração de fluxos de dados para implementar e validar algoritmos. Permite-se então citar trabalhos recentes que utilizaram o *framework* MOA para implementação de novos algoritmos: IBLStreams (SHAKER; HÜLLERMEIER, 2012), SAE e SAE2 (GOMES; ENEMBRECK, 2013, 2014), SFNClassifier (BARDDAL; GOMES; ENEMBRECK, 2014), SFNRegressor e SFNRegressor+ADWIN (BARDDAL; ENEMBRECK, 2013; BARDDAL; GOMES; ENEMBRECK, 2015a) e Online Updated Accuracy Ensemble (BRZEZINSKI; STEFANOWSKI, 2013). O *framework* MOA está disponível para *download* em <http://http://moa.cms.waikato.ac.nz/>.

Para entender a dinâmica do ambiente MOA na tarefa de agrupamento, remete-se a Figura 5.1. Na Figura 5.1, pode-se ver quatro componentes: o fluxo de dados $\mathcal{S}$ provendo instâncias, a etapa *trainOnInstance* representando a etapa *online*, a etapa *getClusteringResult* (etapa *offline*) e a Avaliação dos Grupos (externa ao algoritmo de agrupamento).

Considerações sobre como implementar algoritmos de agrupamento no *framework* MOA são apresentadas no Apêndice A.

5.2 Geradores de Dados

Dentro da literatura, é bastante comum avaliar algoritmos de mineração de fluxos de dados utilizando dados sintéticos. Dados sintéticos, ao contrário de conjuntos de dados reais, permitem ao avaliador um melhor entendimento sobre a distribuição dos dados e dos modelos derivados pelos algoritmos de indução.

[1] <http://www.cs.waikato.ac.nz/ml/weka/>

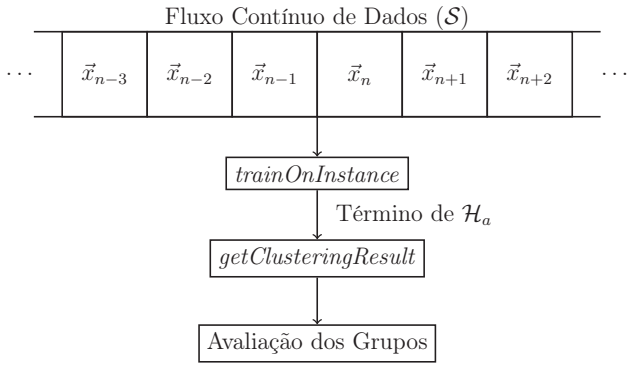


Figura 5.1: *Framework* utilizado pelo MOA para agrupamento em fluxos contínuos de dados.

Dentro do problema de Agrupamento *Online*, algoritmos são comumente avaliados com dados sintéticos de duas formas: utilizando conjuntos de dados de brinquedo (*toy problems*) ou através da utilização de dados gerados a partir de distribuições Gaussianas. Deste modo, são apresentados dois geradores que visam replicar este tipo de avaliação realizada na literatura: *Radial Basis Function* e *Two Moon*. O gerador *Radial Basis Function* se encontra originalmente no *framework* MOA, enquanto o gerador *Two Moon* foi desenvolvido especialmente para este estudo.

5.2.1 *Radial Basis Function* (RBF)

O gerador RBF (*Radial Basis Function*) gera um número definido pelo usuário de centróides que se movimentam dentro do espaço de atributos. Cada centróide gera um grupo definido por um rótulo (grupo verdadeiro/classe), posição no espaço (centro), peso e desvio padrão (raio) calculado a partir de uma distribuição Gaussiana. A Figura 5.2 apresenta um exemplo de conjunto de dados gerado utilizando o gerador RBF com dois atributos d_1 e d_2; 2 grupos reais Cl_1 e Cl_2 e dados ruidosos.

Ainda, o gerador RBF permite que estes centróides se movimentem dentro do espaço de características, gerando assim mudanças virtuais de conceito. A movimentação destes centróides é baseada em um parâmetro *speed* (velocidade), que determina a frequência em que cada centróide deve alterar sua posição.

Outras possibilidades providas pelo gerador RBF são a criação de problemas de dimensionalidade d definida pelo usuário e a variação do número de centróides durante o fluxo de dados, se tornando um gerador de problemas de Agrupamento *Online* bastante escalável. Adicionalmente, o gerador RBF foi estendido para permitir a variação do tamanho de raio em $\pm 15\%$ durante a execução, que ocorre de maneira aleatória, assim como proposto em (HASSANI; SPAUS; SEIDL, 2014).

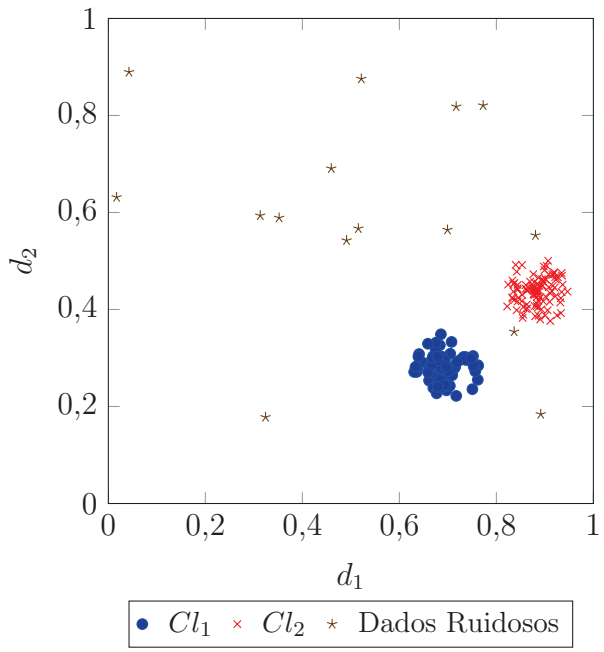


Figura 5.2: Exemplo de conjunto de dados gerado utilizando o gerador RBF com 2 atributos (d_1 e d_2), dois grupos verdadeiros Cl_1 e Cl_2 e dados ruidosos.

Experimento	Mudança de Conceito	Evolução de Conceito	d	Tamanho de Raio Variável
RBF_2	A cada 500 instâncias	A cada 3.000 instâncias	2	$0,07 \pm 15\%$
RBF_5	A cada 500 instâncias	A cada 3.000 instâncias	5	$0,07 \pm 15\%$
RBF_{20}	A cada 500 instâncias	A cada 3.000 instâncias	20	$0,07 \pm 15\%$
RBF_{50}	A cada 500 instâncias	A cada 3.000 instâncias	50	$0,07 \pm 15\%$

Tabela 5.1: Fluxos de dados sintetizados utilizando o gerador RBF.

Deste modo, o gerador RBF permite gerar problemas d-dimensionais de formato hiper-esférico com mudanças de conceito abruptas de acordo com uma frequência *speed* e com evoluções de conceito. A Tabela 5.1 sumariza os experimentos sintéticos realizados utilizando o gerador RBF. Todos os experimentos apresentados na Tabela 5.1 possuem 50.000 instâncias, 5 grupos verdadeiros (na média, pois um grupo aparece/desaparece a cada 3.000 instâncias) e 10% de ruído.

5.2.2 Two Moon

O gerador sintético *Two Moon* gera instâncias em dois conjuntos não hiper-esféricos em um espaço bidimensional (ZHOU; HUANG; SCHÖLKOPF, 2005). Este gerador é baseado no conjunto de dados de brinquedo conhecido como "*Banana Dataset*" e forma dois semicírculos Cl_1 e Cl_2, como apresentado na Figura 5.3.

Por gerar conjuntos grupos inerentemente não hiper-esféricos em espaço bidimensional, é um gerador de dados utilizado com o intuito de verificar a capacidade dos algo-

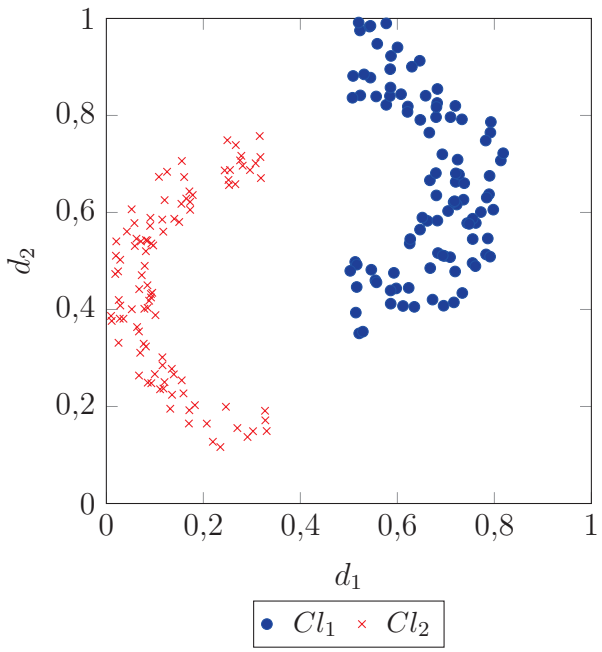


Figura 5.3: Exemplo de conjunto de dados gerado utilizando o gerador *Two Moon* com dois grupos verdadeiros Cl_1 e Cl_2.

ritmos de agrupamento ao encontrar originalmente este tipo de grupo.

5.3 Conjuntos de Dados Reais

Ao contrário de dados sintéticos, problemas reais exibem comportamento diferenciado: a distribuição das instâncias entre as os grupos verdadeiros normalmente não é uniforme, i.e. grupos não são hiper-esféricos, e duas instâncias $\vec{x}_i$ e $\vec{x}_{i+1}$, por serem "vizinhas" na sua chegada, apresentam grande probabilidade de pertencerem ao mesmo grupo verdadeiro (KREMER et al., 2011); e no caso da tarefa de agrupamento, de pertencerem ao mesmo grupo.

Nesta seção são apresentados conjuntos de dados reais utilizados nesta avaliação, juntamente de sua descrição e referências de utilização na literatura.

5.3.1 *Airlines*

O conjunto *Airlines*[2] foi desenvolvido para avaliar algoritmos de mineração de fluxos de dados (IKONOMOVSKA et al., 2011). Este conjunto de dados contém 539.383 instâncias e 8 atributos, representando detalhes sobre todas decolagens e aterrisagens de vôos nos Estados Unidos da América, de Outubro de 1987 até Abril de 2008.

[2]<http://kt.ijs.si/elena_ikonomovska/data.html>

5.3.2 *Electricity*

O conjunto *Electricity*[3] é outro conjunto comumente utilizado na literatura para avaliação de algoritmos para fluxos de dados (RODRIGUES; GAMA; PEDROSO, 2008). Este conjunto foi criado pelo *Australian New South Wales Electricity Market*. Neste problema, preços da energia elétrica não são fixos e são afetados pela relação entre oferta e demanda. Para este conjunto, os preços foram captados a cada 5 minutos, logo, formam uma sequência temporal facilmente aplicável no escopo de mineração de fluxos de dados (HARRIES; WALES, 1999).

Este conjunto de dados contém 45.312 instâncias e 8 atributos, dos quais 7 são contínuos intervalados normalizados no intervalo $[0; 1]$ e um atributo categórico.

5.3.3 *Forest Covertype*

O conjunto de dados *Forest Covertype*[4] é normalmente utilizado para a tarefa de Classificação (BIFET et al., 2013b, 2013a; KOSINA; GAMA, 2012) onde classificadores devem predizer o tipo de vegetação de florestas baseando-se em variáveis cartográficas (KOSINA; GAMA, 2012). Contudo, este conjunto também é utilizado para avaliação de algoritmos de Agrupamento *Online* em (AMINI; WAH, 2014; KRANEN et al., 2011). Este conjunto de dados é composto por células de 900m^2 obtidas da US Forest Service Region 2 Resource Information System (RIS).

Este conjunto de dados possui 581.012 instâncias com 54 atributos contínuos intervalados não normalizados. Para avaliar os algoritmos de maneira igual, este conjunto de dados foi normalizado no intervalo $[0; 1]$.

5.3.4 KDD'98

O conjunto de dados KDD'98[5] possui distribuição não estacionária durante o tempo (KRANEN et al., 2011) e é comumente utilizada para avaliar algoritmos de Agrupamento *Online* (AGGARWAL, 2007; AGGARWAL et al., 2003; CAO et al., 2006). O conjunto KDD'98 contém 95.412 instâncias compostas por 56 atributos sobre pessoas que realizaram doações após receber cartas requisitando-as. O objetivo da utilização deste conjunto de dados para a tarefa de agrupamento é encontrar grupos de doadores com comportamentos similares.

[3] <https://archive.ics.uci.edu/ml/datasets/Individual+household+electric+power+consumption>
[4] <https://archive.ics.uci.edu/ml/datasets/Covertype>
[5] <http://kdd.ics.uci.edu/databases/kddcup98/kddcup98.html>

5.3.5 KDD'99

O conjunto KDD'99[6] é frequentemente utilizado para avaliar algoritmos de mineração de fluxos de dados, devido a sua natureza evolucionária (AMINI; WAH, 2014, 2013; AGGARWAL et al., 2003; CAO et al., 2006). Este conjunto de dados corresponde ao problema de detecção automática de ciberataques, um problema por definição *online*, uma vez que as instâncias são apresentadas de maneira serializada como uma série temporal, onde, devido às evoluções de conceito existentes (aparecimento e desaparecimento de certos tipos de ataques, representados por grupos verdadeiros, com o tempo) (AGGARWAL et al., 2003).

Este conjunto de dados é formado por 4.898.431 instâncias e 38 atributos dos quais 33 são contínuos intervalados não normalizados, 4 binários e 2 categóricos.

5.3.6 *Body Posture and Movements* (BPaM)

O conjunto *Body Posture and Movements* (BPaM) é constituído por dados obtidos de acelerômetros colocados em quatro indivíduos, nas seguintes posições: cintura, coxa esquerda, bíceps direito e tornozelo direito. Estes dados foram obtidos a cada 150ms, formando então uma série temporal aplicável ao problema de mineração de fluxos de dados e totalizam um conjunto de 165,632 instâncias. Originalmente, o intuito deste conjunto de dados é realizar o problema de classificação, determinando se o sujeito está sentado, levantando, de pé, andando ou sentando baseando-se em 18 atributos (UGULINO et al., 2012).

5.4 Protocolo Experimental

Com o intuito de sobrepujar limitações das avaliações apresentadas na literatura, um estudo bastante aprofundado em termos de avaliação foi realizado. Nesta avaliação, os algoritmos são avaliados com uma gama de métricas de qualidade apresentadas na Seção 2.6. Neste capítulo são discutidos apenas os resultados obtidos para: CMM, Tempo de CPU e *RAM-Hours*. O Tempo de CPU é uma métrica que avalia o tempo do processo já escalonado para processamento e um *RAM-Hour* equivale a um *GigaByte* de RAM (*Random Access Memory – Memória de Acesso Aleatório*) utilizado por uma hora (BIFET, 2010). Resultados obtidos para demais métricas de qualidade de agrupamento são apresentadas no Apêndice B.

[6] <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>

Algoritmo	Parâmetro	Valor
CluStream (AGGARWAL et al., 2003)	Horizonte $\mathcal{H}$	1.000
	q	100
ClusTree (KRANEN et al., 2011)	Horizonte $\mathcal{H}$	1.000
	M	3
	Altura máxima da árvore	8
DenStream (CAO et al., 2006)	ψ	1
	$\mathcal{N}$	1.000
	λ	0,25
	ϵ	0,02
	β	0,2
	η	2
HAStream (HASSANI; SPAUS; SEIDL, 2014)	ψ	1
	$\mathcal{N}$	1.000
	λ	0,25
	ϵ	0,02
	β	0,2
CNDenStream	ψ	1
	$\mathcal{N}$	1.000
	λ	0,25
	ϵ	0,02
	β	0,2
	ω	4
SNCStream (BARDDAL; GOMES; ENEMBRECK, 2015b)	ψ	1
	$\mathcal{N}$	1.000
	λ	0,25
	ϵ	0,02
	β	0,2
	ω	4

Tabela 5.2: Algoritmos e seus respectivos parâmetros.

Todos os algoritmos foram parametrizados de acordo com seus artigos originais. A Tabela 5.2 apresenta os algoritmos utilizados e seus respectivos parâmetros. Todos os parâmetros de todas as publicações foram escolhidos de acordo com propriedades teóricas, restrições de tempo e memória e impacto na média da qualidade do agrupamento. Ressalta-se que a escolha do valor ω é um assunto discutido na Seção 5.6.1.

É importante ressaltar que as implementações dos algoritmos ClusTree e CluStream no *framework* MOA possuem um importante relaxamento: **a etapa *offline* é alimentada por um parâmetro externo, o número de grupos verdadeiros**. Ou seja, estes algoritmos contrariam uma fundamental propriedade desejada: ser independente de parâmetros que informem o número de grupos a serem encontrados.

Deste modo, uma adaptação a estes algoritmos foi desenvolvida no *framework* MOA, substituindo a etapa *offline* para executar o algoritmo DBSCAN (ESTER et al., 1996). Contudo, esta avaliação contemplará as duas abordagens: informada (*k-means*) e não informada (DBSCAN).

Para esta avaliação, todos os experimentos assumem um Horizonte de Avaliação $\mathcal{H}_a = 1.000$. Para comparação estatística de CMM, um valor médio foi calculado baseando-se nestas subavaliações. As métricas de Tempo de CPU e *RAM-Hours* são cumulativas e refletem o valor obtido para processar o fluxo de dados inteiro.

Com o intuito de comparar o conjunto de algoritmos no conjunto de fluxos de dados sintéticos e reais apresentados, foram utilizados testes de hipóteses para verificar a existência ou não de diferença estatística entre os algoritmos.os algoritmos. Logo, uma combinação de testes não paramétricos foi utilizada, usando o teste de Friedman (FRIEDMAN, 1937) para determinar se existe diferença estatística entre os algoritmos; e *post-hoc* de Nemenyi (CORDER; FOREMAN, 2011) para determinar quais algoritmos e/ou grupos de algoritmos são diferentes, ambos utilizando um nível de significância $\alpha = 0.05$. A escolha pelos testes de Friedman e Nemenyi se deve às correções realizadas para diminuir erros do tipo II (CORDER; FOREMAN, 2011) e sua aceitação e utilização recorrente pela comunidade de mineração de fluxos de dados para avaliação de algoritmos (JAPKOWICZ; SHAH, 2011).

Todos os experimentos foram executados em um computador de arquitetura Intel Xeon E5649 2.53GHz $\times 8$ com sistema operacional centOS e 16GB de RAM.

5.5 Resultados Preliminares

Algoritmos de agrupamento devem idealmente: encontrar o número correto de grupos, maximizar métricas de qualidade e minimizar tempo de processamento e uso de memória.

Para computar o erro ζ entre o número correto de grupos l e o número K de grupos encontrados pelo algoritmo de agrupamento, utiliza-se a Equação 5.1.

$$\zeta = |K - l| \tag{5.1}$$

A Figura 5.4 apresenta o valor médio $\overline{\zeta}$ dos desvios obtidos durante as subavaliações realizadas em todos os experimentos realizados. Ainda na Figura 5.4, percebe-se que o algoritmo SNCStream obtém o menor erro médio absoluto $\overline{\zeta}$ para todos os experimentos, demonstrando a capacidade deste algoritmo de encontrar grupos de maneira não supervisionada.

Por outro lado, a Figura 5.5 apresenta outro comparativo de erro médio absoluto para os experimentos reais. Nesta Figura, percebe-se que o algoritmo SNCStream também apresenta o menor erro médio absoluto para estes experimentos.

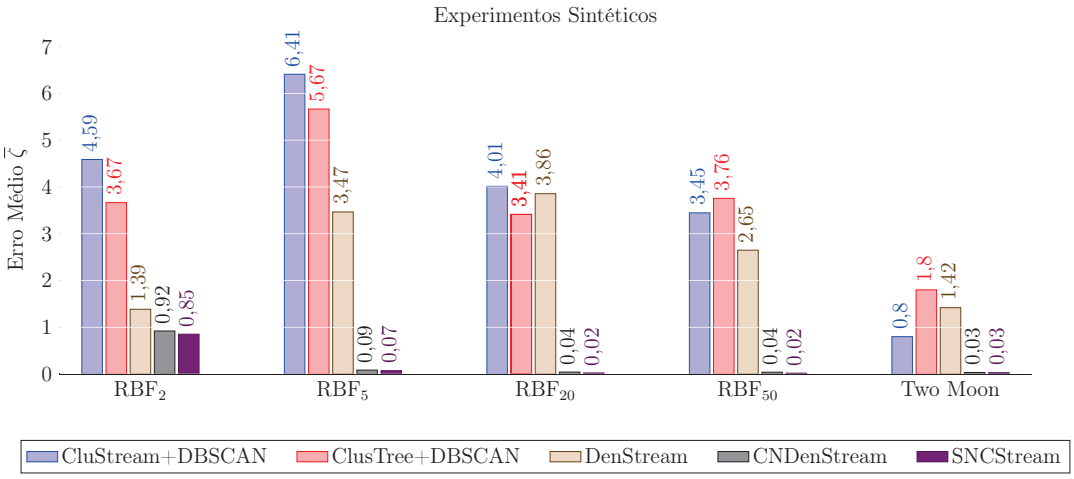


Figura 5.4: Erro médio entre número de grupos reais e obtido pelos algoritmos em experimentos sintéticos.

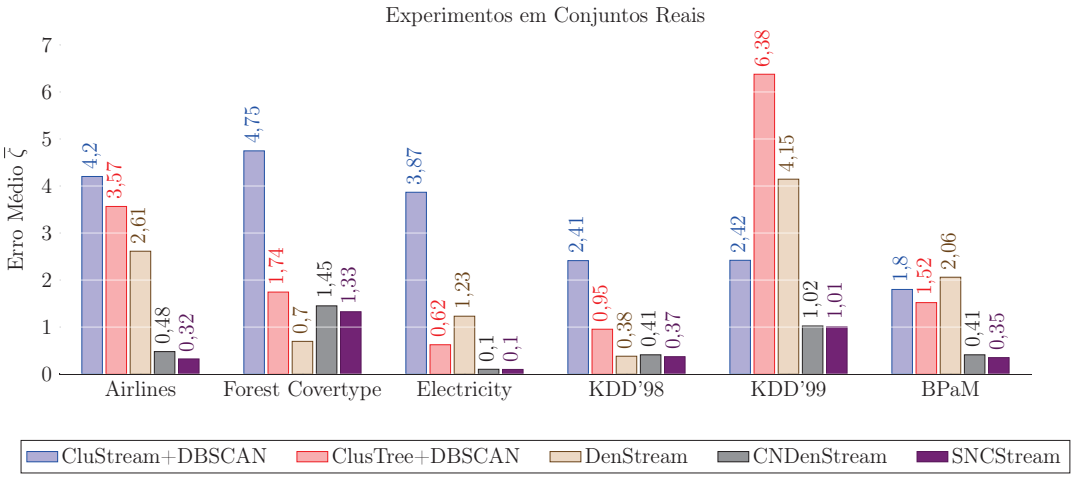


Figura 5.5: Erro médio entre número de grupos reais e obtido pelos algoritmos em experimentos reais.

Comparar apenas o número de grupos obtido K com o número de grupos real l pode derivar métricas falaciosas. É possível que em um problema onde 5 grupos reais existam, um algoritmo seja capaz de encontrar 5 grupos, contudo, diferentes dos ideais. A Figura 5.6 apresenta um exemplo onde existem 2 grupos reais e dados ruidosos. Ainda, vê-se que os grupos obtidos k_1 e k_2 não representam aos grupos verdadeiros Cl_1 e Cl_2.

Logo, deve-se avaliar também métricas de qualidade de agrupamentos. Para este trabalho, define-se como métrica de qualidade principal o CMM (discutida anteriormente na Seção 2.6.9), devido a sua formulação que leva em consideração características do ambiente *online*.

As próximas Seções apresentarão análises acerca dos parâmetros do algoritmo SNCStream, sempre a luz da qualidade do agrupamento obtido, tempo de processamento e utilização de memória.

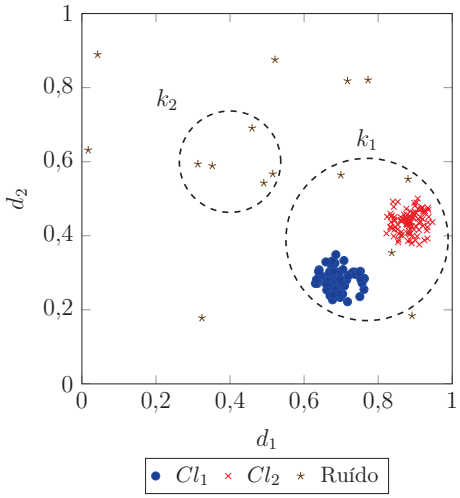


Figura 5.6: Exemplo de conjunto de dados gerado agrupado com o algoritmo ClusTree com etapa *offline k-means*, onde dois grupos reais foram agrupados como um só e outro grupo contempla apenas instâncias ruidosas.

5.6 Análise e Parametrização

Nesta seção é apresentada uma análise do algoritmo SNCStream perante seus principais parâmetros e seus respectivos impactos na qualidade do agrupamento final, tempo de processamento e uso de memória.

5.6.1 O Impacto do Parâmetro ω

Ao contrário de algoritmos baseados em densidade como DenStream (CAO et al., 2006), que utilizam uma variedade de parâmetros de densidade para executar o DBSCAN em sua etapa *offline*, o algoritmo SNCStream depende apenas de um parâmetro: o número de conexões (arestas) ω que cada instância ou *micro-cluster* estabelece ao ser inserido na rede.

Deste modo, uma das principais questões sobre o algoritmo SNCStream é o impacto de ω nos agrupamentos resultantes. Intuitivamente, caso ω possua valor baixo, a rede tenderá a ser bastante esparsa e de acordo com o processo de religação, um número de grupos muito grande será encontrado. Por outro lado, caso ω assuma um valor grande, a rede se tornará muito densa e o processo de religação não seria capaz de dividir a rede nos grupos corretos, gerando no pior caso, apenas um grupo.

Assim como qualquer parâmetro, encontrar o valor ótimo de ω depende do domínio dos dados, contudo, uma primeira análise discutida em (BARDDAL; GOMES; ENEMBRECK, 2015b) mostrou que $\omega = 4$ se mostrou um valor interessante para uma variedade de domínios. Nesta seção o impacto de ω é discutido em maiores detalhes, determinando

Experimento	*CMM*									
	SNCStream									
	$\omega = 1$	$\omega = 2$	$\omega = 3$	$\omega = 4$	$\omega = 5$	$\omega = 6$	$\omega = 7$	$\omega = 8$	$\omega = 9$	$\omega = 10$
RBF_2	0,82	0,97	**0,99**	**0,99**	0,90	0,89	0,88	0,88	0,87	0,87
RBF_5	0,80	0,97	**0,99**	**0,99**	0,91	0,91	0,90	0,87	0,85	0,88
RBF_{20}	0,71	**0,91**	0,89	0,88	0,80	0,78	0,81	0,76	0,75	0,75
RBF_{50}	0,65	**0,86**	0,82	0,84	0,75	0,77	0,74	0,69	0,73	0,78
Two Moon	0,92	0,97	0,99	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**
Airlines	0,84	0,89	0,94	0,96	0,97	0,97	0,97	**0,98**	**0,98**	**0,98**
Electricity	0,71	0,75	0,84	0,89	0,91	0,93	0,93	0,94	**0,95**	0,87
Forest Covertype	0,80	0,85	0,93	0,94	0,97	0,93	0,97	**0,99**	**0,99**	**0,99**
KDD'98	0,37	0,37	**0,38**	**0,38**	**0,38**	**0,38**	0,37	0,37	0,37	0,37
KDD'99	0,76	0,81	0,86	0,90	0,92	0,93	0,94	0,94	0,94	**0,95**
BPaM	0,88	0,93	0,96	0,98	0,98	**0,99**	**0,99**	**0,99**	**0,99**	**0,99**

Tabela 5.3: CMM obtido ao variar ω no algoritmo SNCStream.

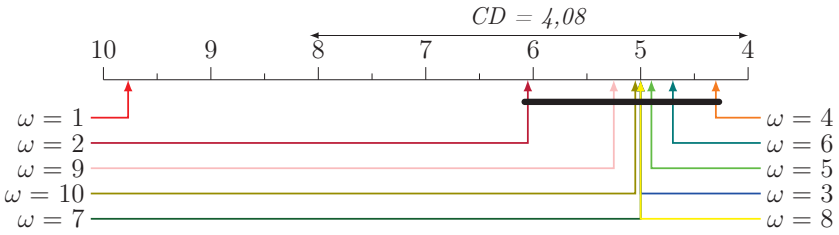


Figura 5.7: Resultado do teste de Nemenyi em termos de CMM ao variar ω.

seu impacto na topologia da rede e na qualidade do agrupamento final.

Na Tabela 5.3 é apresentado o valor de CMM obtido pelo algoritmo SNCStream ao variar o parâmetro ω no intervalo $[1; 10]$ em uma variedade de experimentos. Ainda, pode-se ver que nenhum valor de ω se sobressai em todos os experimentos, confirmando a afirmação anteriormente apresentada. Por outro lado, é importante ressaltar que $\omega = 1$ apresenta o pior valor de *ranking* médio, enquanto $\omega = 4$ apresenta o melhor.

Para determinar a existência ou não de diferença estatisticamente significativa entre os valores de ω testados, foram utilizados os testes de Friedman e Nemenyi. A Figura 5.7 apresenta o resultado do teste de Nemenyi, onde verifica-se que $\omega \in [2; 10] \succ (\omega = 1)$, por diferirem por mais de uma CD (*Critical Difference – Diferença Crítica*) corroborando que $\omega = 4$ é um valor padrão plausível (BARDDAL; GOMES; ENEMBRECK, 2015b).

A qualidade inferior de agrupamento quando $\omega = 1$ ocorre devido a uma característica das redes construídas: o baixo coeficiente de agrupamento. Como discutido na Seção 3.2.3, redes sociais reais que apresentam características de comunidade apresentam coeficientes de agrupamento médios superiores a $0,6$.

Na Figura 5.8 é apresentado o coeficiente de agrupamento médio obtido ao variar o parâmetro ω em todos os experimentos, onde pode-se ver que $\omega = 1$ apresenta baixo coeficiente de agrupamento médio em todos os casos, uma vez que tais redes não possuem

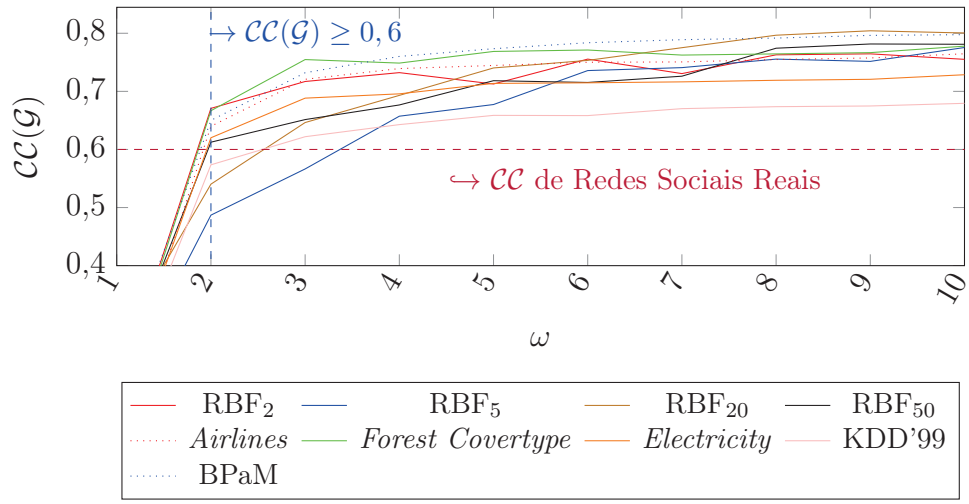


Figura 5.8: Coeficiente de agrupamento global *versus* ω nos experimentos realizados.

densidade (em termos de número de arestas) suficiente para apresentar características de comunidades e logo, de formação de grupos. A Seção 5.8 apresentará os coeficientes de agrupamento obtidos durante os experimentos assim como valores de modularidade.

5.6.2 O Impacto do Parâmetro λ

Outro parâmetro que possui influência direta na qualidade do agrupamento final é o parâmetro λ. O parâmetro λ é utilizado para definir a taxa de decaimento dos pesos dos *micro-clusters* e está definido no intervalo $[0; 1]$. Quão maior for o valor de λ, mais rapidamente estes pesos decairão (vide Equação 2.16). Caso estes pesos decaiam rápido demais, *micro-clusters* muito recentes serão removidos de maneira indevida. No outro oposto, caso o valor de λ seja muito pequeno, *micro-clusters* antigos não serão removidos e o algoritmo não se tornará capaz de "esquecer" os conceitos antigos e se adaptar aos novos de maneira eficiente.

Nesta seção avalia-se o impacto do parâmetro λ na qualidade do agrupamento obtido pelo algoritmo SNCStream. Para esta análise o parâmetro λ foi variado no intervalo $[0; 1]$ com incremento de $0,05$.

A Figura 5.9 apresenta os resultados obtidos ao variar o parâmetro λ. Nesta Figura pode-se perceber que onde $\lambda = 0,05$ a qualidade do agrupamento é a mais alta entre os valores testados para todos os experimentos, exceto nos experimentos RBF$_2$ e *Forest Covertype*. Contudo, ressalta-se ainda que demais valores de λ no intervalo $[0; 0,25]$ não apresentam forte impacto na qualidade do agrupamento, logo, assume-se o valor sugerido $\lambda = 0,25$ em (CAO et al., 2006) como o padrão.

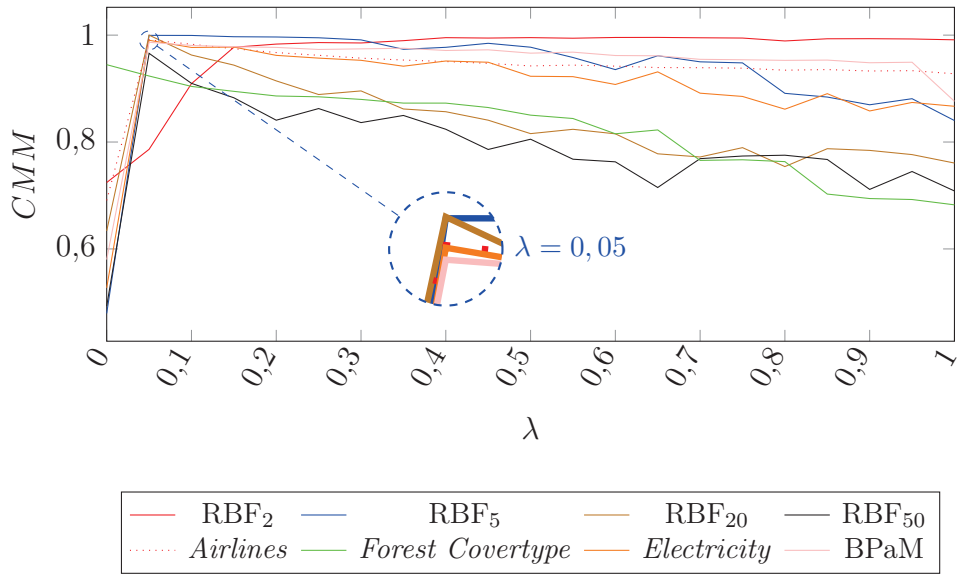


Figura 5.9: λ *versus* CMM.

5.6.3 O Impacto do Parâmetro $\mathcal{N}$

Outro parâmetro que merece uma análise mais refinada é o que determina quantas instâncias formarão a rede inicial antes da etapa de transformação. É importante ressaltar neste momento que o significado deste tamanho de janela inicial no algoritmo SNCStream difere do tamanho de janela inicial dos demais algoritmos. No algoritmo SNCStream, o valor de $\mathcal{N}$ define um número de instâncias que formarão uma rede inicial e que posteriormente serão transformados em *micro-clusters*.

Nos demais algoritmos, esta janela inicial é utilizada para reter instâncias, que posteriormente serão utilizadas para derivar *micro-clusters*. Ainda, caso uma requisição de agrupamento aconteça antes desta derivação dos *micro-clusters* iniciais, tais algoritmos serão incapazes de determinar agrupamentos, ao contrário do algoritmo SNCStream, que possuirá uma rede de instâncias e um agrupamento já formado.

Todavia, como qualquer outro parâmetro, definir um valor ótimo para $\mathcal{N}$ é uma tarefa difícil e subjetiva. Caso $\mathcal{N}$ assuma um valor pequeno, a rede formada pelas instâncias formará apenas um grupo, o que é bastante incomum em problemas de agrupamento. Por outro lado, caso $\mathcal{N}$ assuma um valor muito grande, o algoritmo SNCStream será incapaz de remover instâncias antigas, sendo incapaz de se adaptar a mudanças e evoluções de conceito, sem contar com o crescente número de distâncias a ser calculado na chegada de cada nova instância $\vec{x}_i$.

Nesta seção avalia-se a qualidade do agrupamento obtido ao variar o parâmetro $\mathcal{N}$ no intervalo $[20; 200]$. A Figura 5.10 apresenta os resultados obtidos, onde pode-se perceber que não existe um forte impacto do valor de $\mathcal{N}$ na qualidade do agrupamento

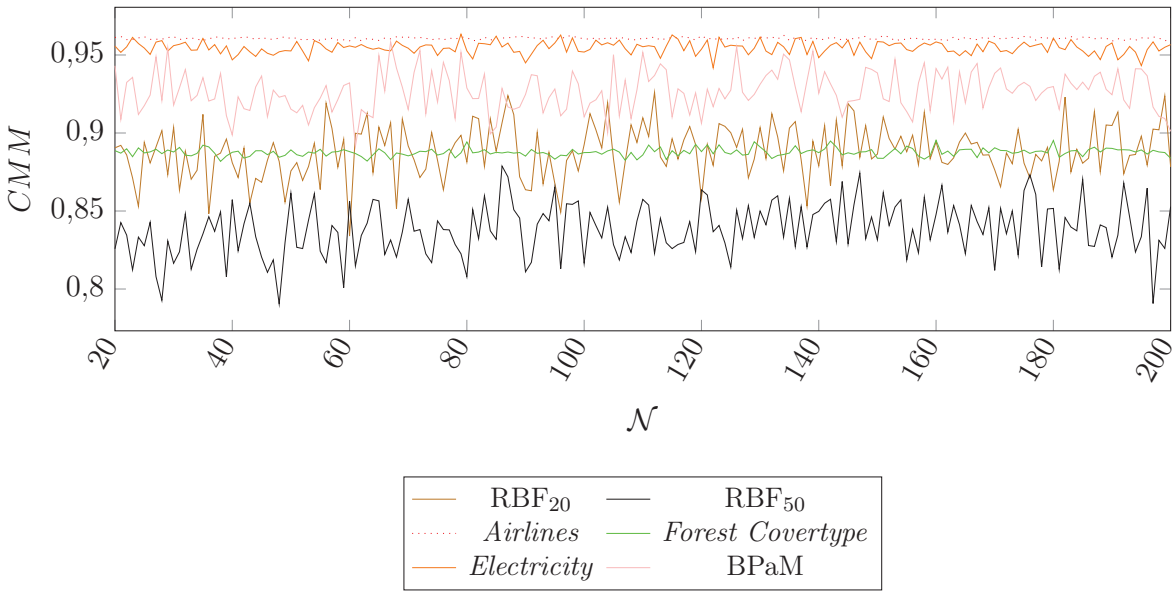


Figura 5.10: $\mathcal{N}$ *versus* CMM.

final. Ainda, não é possível perceber uma tendência de crescimento ou decrescimento da qualidade do agrupamento de acordo com a variação de $\mathcal{N}$. Deste modo, assume-se que qualquer valor dentro deste intervalo é plausível de execução dentro dos domínios avaliados.

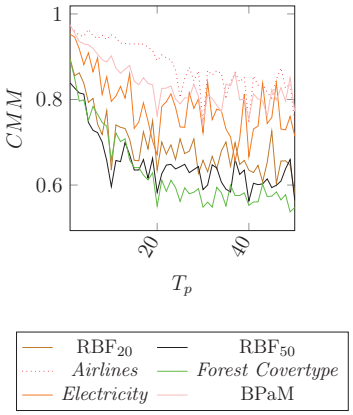
5.6.4 O Impacto do Parâmetro T_p

Por mais que o algoritmo SNCStream não utilize execuções do algoritmo DBSCAN, ele ainda é baseado na noção de densidade. Como no algoritmo DenStream (CAO et al., 2006), os pesos dos *micro-clusters* decaem exponencialmente de acordo com o tempo e estes são verificados periodicamente, de acordo com uma janela de tamanho T_p (discutida na Seção 2.5.3).

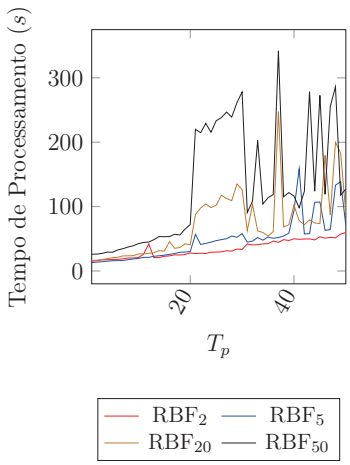
Caso os parâmetros λ, β e ψ sejam definidos de acordo com os originalmente propostos (CAO et al., 2006), a verificação dos pesos ocorrerá frequentemente pois $T_p \leq 4$, teoricamente induzindo alto custo computacional por realizar acessos lineares para verificação dos pesos dos *micro-clusters*.

Por outro lado, realizar tal verificação na chegada de cada instância ($T_p = 1$) soa apropriado, pois todos os *micro-clusters* com peso inadequado serão prontamente removidos da rede, não afetando o agrupamento final.

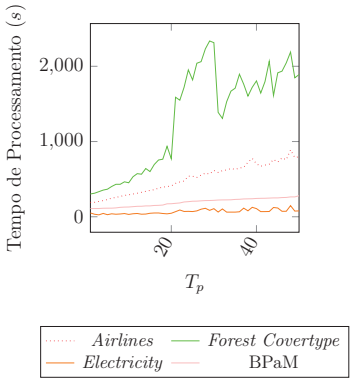
Nesta seção é apresentada uma avaliação de diferentes valores de T_p variados no intervalo $[1; 50]$ e seu impacto em qualidade de agrupamento e tempo de processamento focando no algoritmo SNCStream.



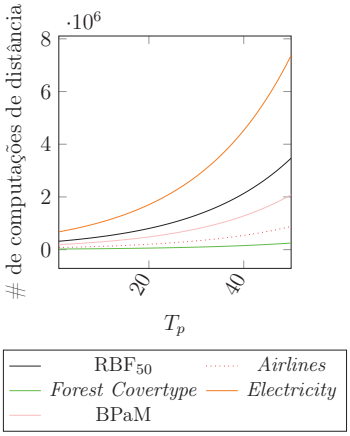
(a) T_p *versus* CMM.



(b) T_p *versus* Tempo de Processamento(s) nos experimentos sintéticos.



(c) T_p *versus* Tempo de processamento (s) nos experimentos de conjuntos de dados reais.



(d) T_p *versus* número de computações de distância.

Figura 5.11: Resultados obtidos ao variar T_p.

A Figura 5.11a apresenta o CMM médio obtido durante os experimentos RBF_{20}, RBF_{50}, *Airlines*, *Electricity*, *Forest Covertype* e BPaM. Nesta Figura pode-se ver que com o crescimento de T_p os valores médios de CMM decaem, corroborando a hipótese de que maiores valores de T_p permitem que *micro-clusters* antigos participem da rede enquanto eles não representam o estado atual do fluxo de dados, logo, acabam a diminuir a qualidade do agrupamento.

Nas Figuras 5.11b e 5.11c são apresentados os tempos de processamento para os experimentos sintéticos (RBF) e de conjuntos de dados reais, respectivamente. Nestas Figuras pode-se perceber que ao contrário do esperado, com o acréscimo de T_p, o tempo de processamento também cresce.

Com maiores valores de T_p, o número de *micro-clusters* armazenado também cresce, pois a remoção demora a ocorrer, logo, um maior número de computações de distâncias

devem ser realizados na chegada de cada instância $\vec{x}_i$ e em processos de religação. Na Figura 5.11d é apresentado um gráfico com a relação entre T_p e o número de computações de distância realizados durante os experimentos, onde pode-se perceber que, com o incremento de T_p, o número de distâncias calculado cresce exponencialmente, justificando o crescimento do tempo de processamento.

Deste modo, pode-se concluir que a configuração de $T_p = 1$ apresenta melhores resultados em termos de qualidade de agrupamento (CMM) para o algoritmo SNCStream, por remover *micro-clusters* indevidos de maneira eficaz; e em tempo de processamento, por implicar em um menor número de computações de distância.

5.6.5 Avaliação de Métricas de Distância

O algoritmo SNCStream original utiliza distâncias Euclidianas para determinar a dissimilaridade entre instâncias e *micro-clusters* (BARDDAL; GOMES; ENEMBRECK, 2015b). Como discutido em uma variedade de trabalhos (CAO et al., 2006; SILVA et al., 2013), distâncias Euclidianas falham ao representar de maneira significativa a dissimilaridade entre pontos em espaços de alta dimensionalidade, fazendo com que algoritmos caiam na sua vastidão, fenômeno conhecido como "Maldição da Dimensionalidade".

Um modo de mostrar tal "vastidão" é comparar a proporção de uma hiper-esfera de raio r e dimensionalidade d com um hiper-cubo de lado $2r$ e mesma dimensionalidade.

O volume desta hiper-esfera é $\frac{2r^d\pi^{\frac{d}{2}}}{d\Gamma(\frac{d}{2})}$ enquanto o do hiper-cubo é $(2r)^d$. Com o crescimento de d, o volume da hiper-esfera se torna insignificante quando comparado com o do hiper-cubo (BEYER et al., 1999; DOMINGOS, 2012). Isto pode ser visto através do limite da Equação 5.2, onde $\Gamma(\cdot)$ representa a função *Gamma*.

$$\lim_{d\to\infty} \frac{\pi^{\frac{d}{2}}}{2^{d-1} \times d\Gamma\left(\frac{d}{2}\right)} = 0 \tag{5.2}$$

Na prática, todo o hiper-espaço está "afastado" da origem e a relação entre a distância da origem para a borda mais próxima d_{min} com a distância para a borda mais afastada d_{max} converge para zero (BEYER et al., 1999) como apresentado na Equação 5.3.

$$\lim_{d\to\infty} \frac{d_{max} - d_{min}}{d_{min}} = 0 \tag{5.3}$$

Na maioria das aplicações de alta dimensionalidade a escolha da métrica de distância não é óbvia e a computação das dissimilaridades é muitas vezes heurística (AGGARWAL; HINNEBURG; KEIM, 2001). Existem poucos trabalhos na literatura que

visam prover um guia de como escolher a métrica de distância correta para calcular a dissimilaridade entre pontos nestes espaços. Deste modo, maior parte dos algoritmos são projetados utilizando a norma L_p, até mesmo para espaços de alta dimensionalidade, que também é conhecida pela sua susceptibilidade à maldição da dimensionalidade (AGGARWAL et al., 2003; CAO et al., 2006; KRANEN et al., 2011). Genericamente, a distância L_p entre duas instâncias $\vec{x}_i$ e $\vec{x}_j$ pode ser calculada de acordo com a Equação 5.4.

$$d_{L_p}(\vec{x}_i, \vec{x}_j) = \left(\sum_{v=1}^{d} \mid \vec{x}_{i;v} - \vec{x}_{j;v} \mid^p \right)^{\frac{1}{p}} \tag{5.4}$$

Em (AGGARWAL; HINNEBURG; KEIM, 2001), os autores discutem sobre diferentes valores de p, e elucidam que $p = 1$ (Distância *Manhattan*) e $p = 2$ (Distância Euclidiana) são teoricamente mais eficientes que $p \geq 3$. Adicionalmente, uma prova é apresentada confirmando que a significância da norma L_p decai rapidamente com o crescimento da dimensionalidade d (AGGARWAL; HINNEBURG; KEIM, 2001; BEYER et al., 1999). Encorajados por esta tendência, autores em (AGGARWAL; HINNEBURG; KEIM, 2001) examinaram o comportamento de distâncias fracionárias, onde $0 \leq p \leq 1$, e apontaram $p = 0,3$ como um valor interessante para vários domínios após a aplicação do algoritmo *k-means* em uma série de conjuntos de dados reais.

Outra popular métrica de distância comumente utilizada em técnicas de agrupamento é a distância Cosseno. A distância Cosseno assume que a dissimilaridade entre dois vetores $\vec{x}_i$ e $\vec{x}_j$ pode ser calculada como o ângulo entre tais vetores (AGGARWAL; HINNEBURG; KEIM, 2001). A Equação 5.5 apresenta a distância Cosseno adaptada para determinar a dissimilaridade entre duas instâncias.

$$d_{Cosseno}(\vec{x}_i, \vec{x}_j) = 1 - \frac{\sum_{v=1}^{d} \vec{x}_{i;v} \times \vec{x}_{j;v}}{\sqrt{\sum_{v=1}^{d} (\vec{x}_{i;v})^2} \times \sqrt{\sum_{v=1}^{d} (\vec{x}_{j;v})^2}} \tag{5.5}$$

Assumindo uma distância Euclidiana (L_2), implica-se que todas as instâncias $\vec{x}_i$ com a mesma distância da origem do espaço satisfazem a equação de uma hiper-esfera $\sum_{i=1}^{n} (\vec{x}_i)^2 = r^2$, onde r é seu raio. Isto significa que todos os componentes de um conjunto de dados contribuem de maneira igual para a distância Euclidiana do centro. Contudo, em Estatística, é preferível métricas de distância que considerem a variabilidade de cada dimensão.

Na distância de Mahalanobis (ACKERMANN; BLOMER; SOHLER, 2010; MAHALANOBIS, 1936) dimensões com alta variabilidade recebem menor peso que componentes com menor variabilidade. Isto é feito ao re-escalar as dimensões utilizando seus desvios

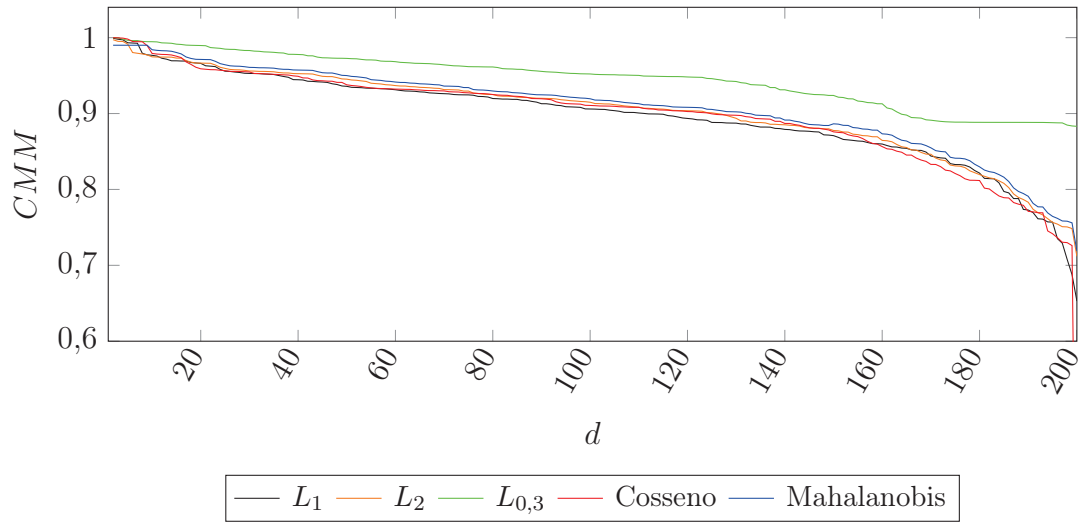


Figura 5.12: CMM obtido pelo algoritmo SNCStream adotando diferentes métricas de distância ao variar a dimensionalidade d de um problema RBF.

padrão σ como apresentado na Equação 5.6.

$$d_{Mahalanobis}(\vec{x}_i, \vec{x}_j) = \sqrt{\sum_{v=1}^{d} \frac{(\vec{x}_{i;v} - \vec{x}_{j;v})^2}{\sigma_v}} \tag{5.6}$$

Com as métricas acima descritas, é possível hipotetizar que o algoritmo SNCStream seja capaz de superar a maldição da dimensionalidade, mantendo a qualidade do agrupamento em fluxos de dados de alta dimensionalidade. Para verificar tal hipótese, o algoritmo SNCStream foi estendido para computar as métricas de distância acima descritas (*Manhattan* L_1, Fracionária $L_{0,3}$, Cosseno e Manhattan). Nesta seção o algoritmo SNCStream é avaliado perante a variação da métrica de distância focando na qualidade do agrupamento.

Na Figura 5.12 é apresentado de maneira gráfica o CMM obtido nos experimentos RBF variando a dimensionalidade d no intervalo $[2; 200]$, onde pode-se ver que com o crescimento de d, a qualidade do agrupamento decai, especialmente nas métricas L_1, L_2 e Cosseno. Ainda, destaca-se os expressivos resultados obtidos pela métrica fracionária $L_{0,3}$, especialmente com o crescimento de d, corroborando seu poder conforme apresentado em (AGGARWAL; HINNEBURG; KEIM, 2001).

Novamente, os experimentos sintéticos e reais apresentados foram utilizados para comparar as variações do algoritmo SNCStream. A Tabela 5.4 apresenta os resultados obtidos ao variar a métrica de distância do algoritmo SNCStream, onde pode-se ver que a métrica $L_{0,3}$ apresenta resultados superiores em todos os experimentos. Para determinar se existe diferença significativa entre os resultados obtidos pelas métricas, aplicou-se os

Experimento	*CMM*				
	SNCStream				
	L_1	L_2	$L_{0,3}$	Cosseno	Mahalanobis
RBF_2	0,98	0,98	**0,99**	0,92	0,98
RBF_5	0,98	0,98	**0,99**	0,95	0,97
RBF_{20}	0,89	0,88	**0,99**	0,85	0,88
RBF_{50}	0,84	0,84	**0,96**	0,76	0,86
Two Moon	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**
Airlines	0,97	0,96	**0,98**	0,96	0,96
Electricity	0,90	0,96	**0,98**	0,89	0,96
Forest Covertype	0,89	0,89	**0,97**	0,95	0,95
KDD'99	0,37	0,37	**0,38**	0,37	0,37
KDD'99	0,90	0,94	**0,95**	0,90	0,94
BPaM	0,98	0,98	**0,99**	0,98	0,98

Tabela 5.4: CMM obtido ao variar a métrica de distância do algoritmo SNCStream.

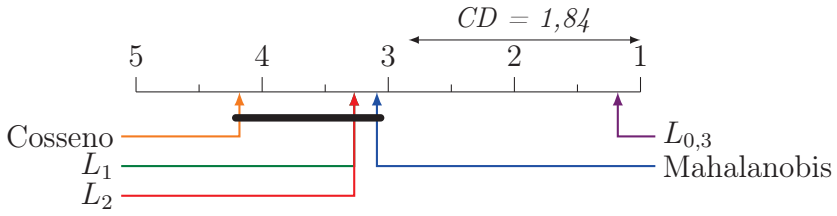


Figura 5.13: Resultado do teste de Nemenyi para diferentes métricas de distância.

testes de Friedman e Nemenyi.

A Figura 5.13 apresenta o resultado do teste de Nemenyi, onde pode-se ver que $\{L_{0,3}\} \succ \{\text{Mahalanobis}, L_2, L_1, \text{Cosseno}\}$ com 95% de confiança, por existir uma diferença maior que uma distância crítica.

5.6.6 Estratégias de Poda

Algoritmos desenvolvidos para minerar fluxos de dados são muitas vezes projetados para serem postos em execução em ambientes de *hardware* limitado, seja em termos de processamento ou de armazenamento. Focando no aspecto de memória, uma estratégia bastante comum para simular tais ambientes limitados é definir um tamanho máximo de memória de acesso aleatório (RAM) disponível. Um exemplo é a abordagem apresentada no algoritmo CluStream (AGGARWAL et al., 2003), onde é definido um número máximo q de *feature vectors* passível de ser mantido pelo algoritmo a todo instante da execução.

Para determinar a viabilidade deste tipo de limitação de memória no algoritmo SNCStream, técnicas de poda foram implementadas:

- Remoção do vértice com menor peso (*Weakest*): Abordagem que assume que potenciais *micro-clusters* com pesos mais baixos tendem a ser removidos por estarem mais próximos do limiar $\beta\psi$;

- Remoção do vértice com menor centralidade de grau *degree* (*Min Degree*): Abordagem que assume que *micro-clusters* com menor centralidade de grau tendem a

ser vértices pouco proeminentes dentro da rede e logo, podem ser removidos sem denegrir a qualidade do agrupamento;

- Remoção do vértice com maior centralidade de grau *degree* (*Max Degree*): Abordagem que assume que *micro-clusters* com maior centralidade de grau são vértices localizados nos "centros" dos grupos e que por não participarem das bordas, sua remoção não impactará negativamente a qualidade do agrupamento obtido;

- Fusão dos dois vértices (*micro-clusters*) mais próximos entre si dentro da rede (*Min Edge*): Abordagem que assume que os dois *micro-clusters* mais próximos entre si conectados dentro da rede podem ser fundidos e suas regiões dentro do espaço de atributos, somadas;

Em todos os casos descritos acima, o usuário deve definir o valor do parâmetro q, o número máximo de *micro-clusters* passível de ser armazenado em memória em todos os momentos do fluxo de dados.

Na Figura 5.14 são apresentados os valores obtidos pelo algoritmo SNCStream ao variar a estratégia de poda e o valor de q no intervalo $[1; 100]$ nos experimentos RBF_2, RBF_5, RBF_{20}, RBF_{50}, *Airlines*, *Forest Covertype*, *Electricity*, *KDD'99* e BPaM. Nestes gráficos, percebe-se que não existem grandes diferenças entre as variações de estratégias de poda, contudo, a variação de q demonstra impactar os resultados finais obtidos. Ainda ressalta-se ainda que em todos os casos o valor de CMM obtido cresce juntamente do valor de q, mesmo que não de maneira linear.

5.7 Comparativo com Demais Algoritmos

Nesta seção são avaliados os algoritmos CluStream (AGGARWAL et al., 2003), ClusTree (KRANEN et al., 2011), DenStream (CAO et al., 2006), HAStream (HASSANI; SPAUS; SEIDL, 2014), CNDenStream e SNCStream (BARDDAL; GOMES; ENEMBRECK, 2015b) utilizando tantos dados sintéticos quanto reais, conforme apresentados nas Seções 5.2 e 5.3. Ressalta-se ainda que os algoritmo CluStream e ClusTree são avaliados duas vezes, uma utilizando *k-means* na etapa *offline* e outra utilizando o algoritmo DBSCAN, conforme discutido na Seção 5.4.

Primeiramente, um comparativo empírico entre o algoritmo SNCStream original (BARDDAL; GOMES; ENEMBRECK, 2015b) será realizado comparando-o com abordagens não informadas. Posteriormente, a mesma análise se repetirá, desta vez, comparando o algoritmo SNCStream utilizando métrica de distância $L_{0,3}$ com todos os outros algoritmos, contemplando então as abordagens baseadas em *k-means* informado. Finalmente, o

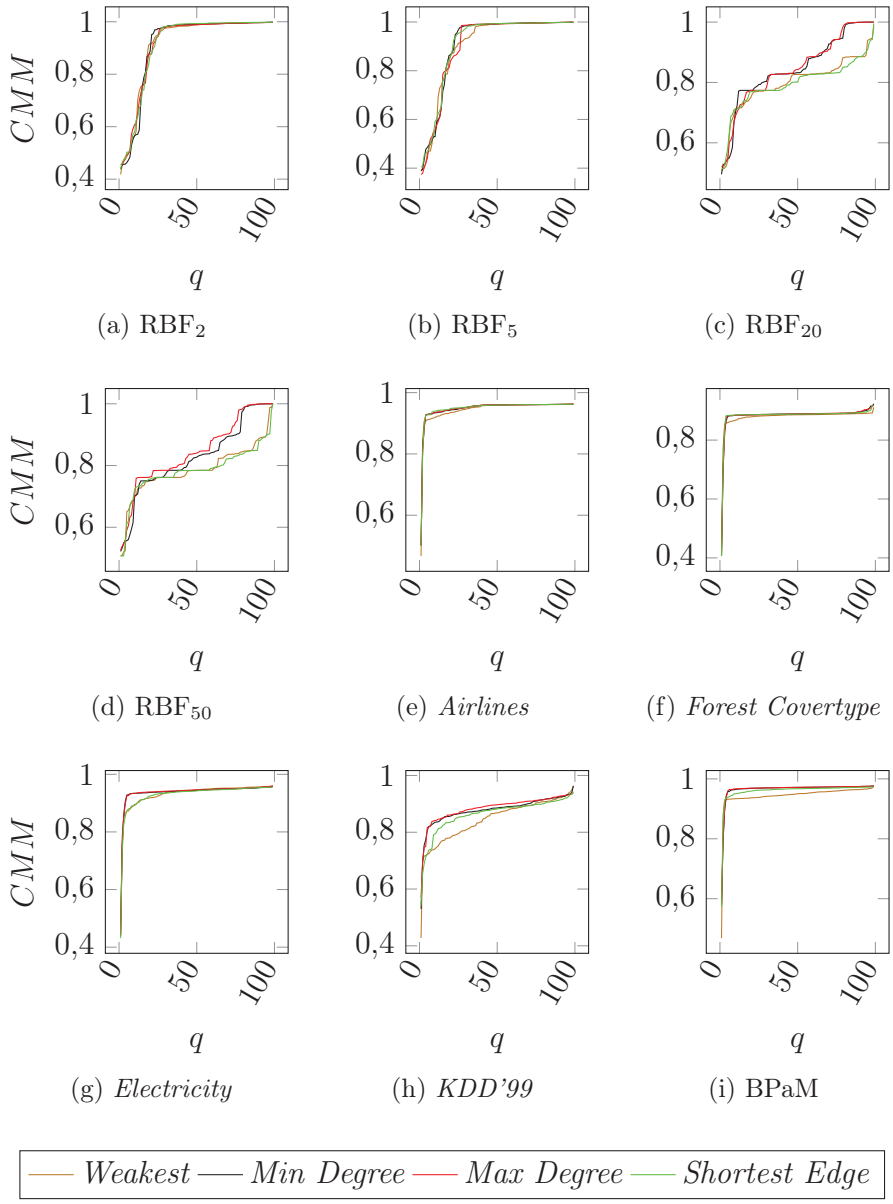


Figura 5.14: Resultados obtidos ao variar o parâmetro q e a estratégia de poda.

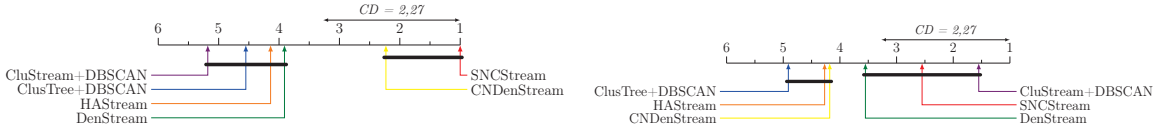
último comparativo apresentará uma análise dos resultados obtidos por todos os algoritmos utilizando a métrica de distância $L_{0,3}$.

5.7.1 SNCStream original *versus* Abordagens não informadas

Nesta seção são apresentados os resultados obtidos pelo algoritmo SNCStream original (utilizando distância L_2) e abordagens não informadas, i.e. CluStream e ClusTree utilizando DBSCAN na etapa *offline*, DenStream e HAStream. Todos os algoritmos citados acima foram avaliados de acordo com o protocolo experimental apresentado na Seção 5.4.

Experimento	*CMM*					
	CluStream+DBSCAN	ClusTree+DBSCAN	DenStream	HAStream	CNDenStream	SNCStream
RBF_2	0,83	0,88	0,94	0,87	0,96	**0,99**
RBF_5	0,67	0,80	0,84	0,86	0,93	**0,99**
RBF_{20}	0,47	0,88	0,85	0,91	0,92	**0,96**
RBF_{50}	0,48	0,44	0,85	0,81	0,93	**0,99**
Two Moon	0,83	0,49	0,99	0,82	0,99	**1,00**
Airlines	0,67	0,76	0,50	0,70	0,87	**0,92**
Electricity	0,41	0,44	0,61	0,43	0,88	**0,94**
Forest Covertype	0,41	0,43	0,38	0,37	0,81	**0,91**
KDD'98	0,37	0,37	0,37	0,37	0,37	**0,38**
KDD'99	0,50	0,49	0,69	0,77	0,84	**0,92**
BPaM	0,61	0,69	0,74	0,74	0,90	**0,91**

Tabela 5.5: *CMM* obtido comparando o algoritmo SNCStream original.



(a) *CMM* (b) *RAM-Hours*

Figura 5.15: Resultado do teste de Nemenyi para o comparativo do algoritmo SNCStream original e algoritmos não informados.

A Tabela 5.5 apresenta os resultados obtidos em termos de *CMM*, onde pode-se ver que o algoritmo SNCStream apresenta métricas superiores em todos os casos com exceção do experimento *Two Moon*, onde houve um empate com o algoritmo DenStream.

Para determinar a existência ou não de diferença estatística significativa entre os algoritmos neste conjunto de experimentos, utilizou-se o teste de Friedman, que apontou a existência de diferença. Finalmente, utilizou-se o teste *post-hoc* de Nemenyi, cujo resultado é apresentado na Figura 5.15a, onde pode-se ver que {SNCStream, CNDenStream} $\succ$ {HAStream, DenStream, ClusTree, CluStream} com $\alpha = 0,05$.

Os resultados obtidos para tempo de processamento e *RAM-Hours* são apresentados nas Tabelas 5.6 e 5.7, respectivamente. Novamente, a combinação de testes de Friedman e Nemenyi foi utilizada para determinar a existência de diferença estatística significativa nestes quesitos. No caso de tempo de processamento, o teste de Friedman apontou que **não** existe diferença entre os algoritmos. Em termos de *RAM-Hours*, o teste de Friedman apontou diferença significativa e o resultado do teste de Nemenyi é apresentado na Figura 5.15b, onde vê-se que {CluStream+DBSCAN, SNCStream, DenStream} $\succ$ {CNDenStream, HAStream, ClusTree+DSBCAN}.

5.7.2 SNCStream $L_{0,3}$ *versus* Demais algoritmos

Nesta seção o algoritmo SNCStream utilizando distância fracionária $L_{0,3}$ é comparado com os algoritmos discutidos na seção anterior com a adição de abordagens utilizando

Experimento	Tempo de CPU (s)					
	CluStream+DSBCAN	ClusTree+DBSCAN	DenStream	HAStream	CNDenStream	SNCStream
RBF_2	12,59	**9,31**	36,08	19,98	32,91	26,33
RBF_5	13,67	**9,02**	28,88	18,91	19,50	15,60
RBF_{20}	35,18	15,13	38,00	34,87	17,89	**14,31**
RBF_{50}	71,62	24,59	23,71	64,49	25,99	**20,79**
Two Moon	2,70	**1,05**	4,77	7,32	3,60	2,88
Airlines	262,18	179,65	240,76	**108,44**	156,16	124,93
Electricity	29,17	15,32	32,35	**9,22**	47,73	38,18
Forest Covertype	523,65	334,43	328,33	291,91	161,63	**129,30**
KDD'98	1774,32	1255,20	**1104,74**	1823,21	1485,15	1188,12
KDD'99	751,71	**280,30**	657,26	594,50	936,39	749,11
BPaM	114,29	**77,06**	179,46	176,39	125,26	100,21

Tabela 5.6: Tempo de processamento (s) obtido comparando o algoritmo SNCStream original e algoritmos não informados.

Experimento	*RAM-Hours* (GB/Hora)					
	CluStream+DSBCAN	ClusTree+DBSCAN	DenStream	HAStream	CNDenStream	SNCStream
RBF_2	$\mathbf{7,40 \times 10^{-8}}$	$1,74 \times 10^{-6}$	$1,04 \times 10^{-6}$	$1,01 \times 10^{-6}$	$2,91 \times 10^{-6}$	$2,33 \times 10^{-6}$
RBF_5	$\mathbf{9,73 \times 10^{-8}}$	$4,56 \times 10^{-6}$	$9,17 \times 10^{-7}$	$9,59 \times 10^{-7}$	$1,15 \times 10^{-6}$	$9,23 \times 10^{-7}$
RBF_{20}	$\mathbf{4,69 \times 10^{-7}}$	$8,76 \times 10^{-6}$	$2,43 \times 10^{-6}$	$1,77 \times 10^{-6}$	$9,71 \times 10^{-7}$	$7,77 \times 10^{-7}$
RBF_{50}	$1,84 \times 10^{-6}$	$3,70 \times 10^{-5}$	$7,93 \times 10^{-6}$	$3,27 \times 10^{-6}$	$2,02 \times 10^{-6}$	$\mathbf{1,61 \times 10^{-6}}$
Two Moon	$\mathbf{3,45 \times 10^{-9}}$	$4,18 \times 10^{-8}$	$3,62 \times 10^{-7}$	$5,54 \times 10^{-8}$	$8,67 \times 10^{-8}$	$6,34 \times 10^{-8}$
Airlines	$\mathbf{1,65 \times 10^{-6}}$	$8,32 \times 10^{-5}$	$7,06 \times 10^{-6}$	$3,41 \times 10^{-5}$	$8,67 \times 10^{-6}$	$6,93 \times 10^{-6}$
Electricity	$\mathbf{2,32 \times 10^{-7}}$	$7,88 \times 10^{-6}$	$1,09 \times 10^{-6}$	$3,25 \times 10^{-6}$	$2,40 \times 10^{-6}$	$1,92 \times 10^{-6}$
Forest Covertype	$4,81 \times 10^{-5}$	$2,27 \times 10^{-4}$	$1,37 \times 10^{-5}$	$1,27 \times 10^{-5}$	$1,39 \times 10^{-5}$	$\mathbf{1,11 \times 10^{-5}}$
KDD'98	$3,44 \times 10^{-5}$	$1,38 \times 10^{-4}$	$3,89 \times 10^{-5}$	$8,03 \times 10^{-4}$	$10,69 \times 10^{-5}$	$\mathbf{2,69 \times 10^{-5}}$
KDD'99	$\mathbf{1,44 \times 10^{-5}}$	$2,33 \times 10^{-5}$	$5,69 \times 10^{-5}$	$1,45 \times 10^{-3}$	$4,74 \times 10^{-5}$	$3,79 \times 10^{-5}$
BPaM	$\mathbf{7,66 \times 10^{-7}}$	$3,23 \times 10^{-5}$	$6,22 \times 10^{-6}$	$1,24 \times 10^{-4}$	$1,84 \times 10^{-5}$	$1,47 \times 10^{-5}$

Tabela 5.7: *RAM-Hours* obtido comparando o algoritmo SNCStream original e algoritmos não informados.

k-means informado na etapa *offline*: CluStream+*k-means* e ClusTree+*k-means*.

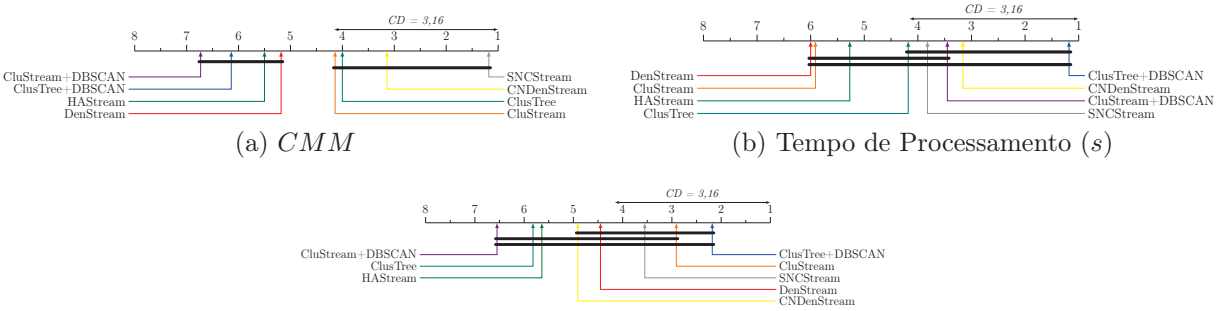
A Tabela 5.8 apresenta os resultados obtidos, onde pode-se ver que o algoritmo SNCStream apresenta métricas de qualidade de agrupamento bastante elevadas, contudo, empata com o algoritmo CluStream no experimento RBF_5 e com o algoritmo ClusTree+*k-means* no experimento RBF_{20}; e é inferior no experimento RBF_{50}.

Para determinar a existência (ou não) de diferença estatística significativa entre os algoritmos, utilizou-se o teste de Friedman e *post-hoc* de Nemenyi. O teste de Friedman apontou diferença significativa e o resultado do teste de Nemenyi é apresentado na Figura 5.16a, onde vê-se que {SNCStream, CNDenStream, ClusTree, CluStream} $\succ$ {DenStream, HAStream, ClusTree+DBSCAN, CluStream+DBSCAN}.

Adicionalmente, foram comparados os resultados obtidos em termos de tempo de processamento e *RAM-Hours*. As Figuras 5.16b e 5.16c apresentam os resultados obtidos após os testes de Friedman e Nemenyi, onde pode-se ver que, de maneira indireta, não existe diferença significativa entre os algoritmos em termos de tempo de processamento e de uso de memória. Os valores brutos obtidos para tempo de processamento e *RAM-Hours* nestes experimentos são apresentados nas Tabelas 5.9 e 5.10, respectivamente. É importante ressaltar ainda que o algoritmo SNCStream não pode ser dado como inferior

Experimento	*CMM*							
	CluStream		ClusTree		DenStream	HAStream	CNDenStream	SNCStream ($L_{0.3}$)
	k-means	DBSCAN	*k-means*	DBSCAN				
RBF_2	0,91	0,83	0,88	0,88	0,94	0,88	0,96	**0,99**
RBF_5	**0,99**	0,67	0,98	0,80	0,84	0,86	0,93	**0,99**
RBF_{20}	0,97	0,47	**0,99**	0,88	0,85	0,91	0,92	**0,99**
RBF_{50}	0,76	0,48	**0,99**	0,44	0,85	0,81	0,93	0,96
Two Moon	0,72	0,83	0,66	0,49	0,99	0,82	0,99	**1,00**
Airlines	0,93	0,67	0,76	0,94	0,50	0,70	0,87	**0,98**
Electricity	0,75	0,41	0,76	0,44	0,61	0,43	0,88	**0,98**
Forest Covertype	0,76	0,41	0,75	0,43	0,38	0,37	0,81	**0,97**
KDD'98	0,37	0,37	0,37	0,37	0,37	0,37	0,37	**0,38**
KDD'99	0,40	0,50	0,44	0,4	0,69	0,77	0,84	**0,95**
BPaM	0,92	0,61	0,92	0,69	0,74	0,74	0,90	**0,99**

Tabela 5.8: *CMM* médio obtido durante os experimentos no comparativo do algoritmo SNCStream ($L_{0,3}$) contra os demais.



(a) *CMM* (b) Tempo de Processamento (s)

(c) *RAM-Hours*

Figura 5.16: Resultados do teste de Nemenyi no comparativo do algoritmo SNCStream ($L_{0,3}$) contra os demais.

nestes aspectos para os demais, uma vez que o *ranking* médio deste **não** difere por mais de um CD.

5.7.3 SNCStream $L_{0,3}$ *versus* Demais algoritmos utilizando $L_{0,3}$

Finalmente, com o intuito de determinar se os outros algoritmos são capazes de alcançar a alta qualidade de agrupamento obtida pelo algoritmo SNCStream, implementações dos algoritmos CluStream, ClusTree, DenStream e HAStream foram realizadas adotando como métrica de distância a medida fracionária $L_{0,3}$. Nesta seção os resultados obtidos são apresentados focando apenas no aspecto de qualidade de agrupamento.

A Tabela 5.11 apresenta os resultados obtidos por todos os algoritmos ao utilizar a métrica de distância fracionária $L_{0,3}$. Novamente, é interessante ressaltar que o algoritmo SNCStream continua apresentando valores de qualidade de agrupamento bastante elevados quando comparados aos demais, exceto nos experimentos RBF_{20}, RBF_{50} e *Airlines*.

Para determinar a existência ou não de diferença estatística significativa repetiu-se o teste de Friedman e Nemenyi, cujo resultado é apresentado na Figura 5.17, onde vê-se que {SNCStream, CNDenStream, CluStream+*k-means*, ClusTree+*k-means*} $\succ$ {DenStream,

Experimento	Tempo de Processamento (s)							
	CluStream		ClusTree		DenStream	HAStream	CNDenStream	SNCStream ($L_{0.3}$)
	k-means	DBSCAN	*k-means*	DBSCAN				
RBF_2	12,59	9,31	7,82	**7,47**	36,08	19,98	32,91	26,33
RBF_5	13,67	9,02	10,39	**6,74**	28,88	18,91	19,50	15,60
RBF_{20}	35,18	15,13	27,72	18,99	38,00	34,87	17,89	**14,31**
RBF_{50}	71,62	24,59	56,27	**18,60**	23,71	64,49	25,99	20,79
Two Moon	2,31	2,70	2,28	**1,05**	4,77	7,32	3,60	2,88
Airlines	262,18	179,65	161,38	**108,12**	240,76	108,44	156,16	124,93
Electricity	29,17	15,32	17,24	9,98	32,35	**9,22**	47,73	38,18
Forest Covertype	523,65	334,43	335,90	218,14	328,34	291,91	161,63	**129,30**
KDD'98	1774,32	1255,20	1783,24	1261,51	**1104,74**	1823,21	1485,15	1188,12
KDD'99	751,71	280,30	319,10	**196,67**	657,26	594,50	936,39	749,11
BPaM	114,29	77,06	72,46	**53,90**	179,46	176,39	125,26	100,21

Tabela 5.9: Tempo de processamento (s) obtido durante os experimentos no comparativo do algoritmo SNCStream ($L_{0,3}$) contra os demais.

Experimento	*RAM-Hours* (GB/Hora)							
	CluStream		ClusTree		DenStream	HAStream	CNDenStream	SNCStream ($L_{0,3}$)
	k-means	DBSCAN	*k-means*	DBSCAN				
RBF_2	$\mathbf{7,40 \times 10^{-8}}$	$1,74 \times 10^{-6}$	$3,81 \times 10^{-6}$	$4,55 \times 10^{-8}$	$1,04 \times 10^{-6}$	$1,01 \times 10^{-6}$	$2,91 \times 10^{-6}$	$2,33 \times 10^{-6}$
RBF_5	$9,73 \times 10^{-8}$	$4,56 \times 10^{-6}$	$3,40 \times 10^{-6}$	$\mathbf{7,34 \times 10^{-8}}$	$9,17 \times 10^{-7}$	$9,59 \times 10^{-7}$	$1,15 \times 10^{-6}$	$9,23 \times 10^{-7}$
RBF_{20}	$4,69 \times 10^{-7}$	$8,76 \times 10^{-6}$	$6,24 \times 10^{-6}$	$\mathbf{3,68 \times 10^{-7}}$	$2,43 \times 10^{-6}$	$1,77 \times 10^{-6}$	$9,71 \times 10^{-7}$	$7,77 \times 10^{-7}$
RBF_{50}	$1,84 \times 10^{-6}$	$3,70 \times 10^{-5}$	$2,80 \times 10^{-5}$	$1,45 \times 10^{-6}$	$7,93 \times 10^{-6}$	$3,27 \times 10^{-6}$	$2,02 \times 10^{-6}$	$1,61 \times 10^{-6}$
Two Moon	$3,22 \times 10^{-9}$	$\mathbf{3,45 \times 10^{-9}}$	$3,99 \times 10^{-8}$	$4,18 \times 10^{-8}$	$3,62 \times 10^{-7}$	$5,54 \times 10^{-8}$	$8,67 \times 10^{-8}$	$6,34 \times 10^{-8}$
Airlines	$1,65 \times 10^{-6}$	$8,32 \times 10^{-5}$	$5,01 \times 10^{-5}$	$\mathbf{1,01 \times 10^{-6}}$	$7,06 \times 10^{-6}$	$3,41 \times 10^{-5}$	$8,67 \times 10^{-6}$	$6,93 \times 10^{-6}$
Electricity	$2,32 \times 10^{-7}$	$7,88 \times 10^{-6}$	$5,13 \times 10^{-6}$	$\mathbf{1,36 \times 10^{-7}}$	$1,09 \times 10^{-6}$	$3,25 \times 10^{-6}$	$2,40 \times 10^{-6}$	$1,92 \times 10^{-6}$
Forest Covertype	$4,81 \times 10^{-5}$	$2,27 \times 10^{-4}$	$1,48 \times 10^{-4}$	$\mathbf{3,07 \times 10^{-6}}$	$1,37 \times 10^{-5}$	$1,27 \times 10^{-5}$	$1,39 \times 10^{-5}$	$1,11 \times 10^{-5}$
KDD'98	$3,44 \times 10^{-5}$	$1,38 \times 10^{-4}$	$8,47 \times 10^{-5}$	$9,15 \times 10^{-5}$	$3,89 \times 10^{-5}$	$8,03 \times 10^{-4}$	$10,69 \times 10^{-5}$	$\mathbf{2,69 \times 10^{-5}}$
KDD'99	$1,44 \times 10^{-5}$	$2,33 \times 10^{-5}$	$\mathbf{6,08 \times 10^{-6}}$	$1,63 \times 10^{-4}$	$5,69 \times 10^{-5}$	$1,45 \times 10^{-3}$	$4,74 \times 10^{-6}$	$3,79 \times 10^{-5}$
BPaM	$7,66 \times 10^{-5}$	$3,23 \times 10^{-5}$	$2,26 \times 10^{-5}$	$\mathbf{4,82 \times 10^{-7}}$	$6,22 \times 10^{-6}$	$1,24 \times 10^{-4}$	$1,84 \times 10^{-5}$	$1,47 \times 10^{-5}$

Tabela 5.10: *RAM-Hours* obtido durante os experimentos no comparativo do algoritmo SNCStream ($L_{0,3}$) contra os demais.

HAStream, ClusTree+DBSCAN, CluStream+DBSCAN}.

5.8 Coeficientes de Agrupamento e Modularidade

Como discutido na Seção 3.4, duas das maneiras de avaliar quão bom um agrupamento é dentro do escopo de grafos é o Coeficiente de Agrupamento e Modularidade. Com este intuito, a Tabela 5.12 apresenta os valores médios dos coeficientes de agrupamento médios e modularidade obtidos para cada experimento. Na Tabela 5.12, percebe-se que tanto os coeficientes de agrupamento e valores de modularidade obtidos são altos, principalmente quando comparados com o valor médio de redes sociais reais que variam no intervalo $[0,3; 0,7]$ para ambas as métricas (NEWMAN; GIRVAN, 2004; CALLAWAY et al., 2000). Deste modo, pode-se inferir que o algoritmo SNCStream gera redes com alto coeficiente de agrupamento, ou seja, suas sub-redes são densas (com muitas arestas).

Ressalta-se ainda que pelo fato das redes obtidas serem desconexas, outras métricas de análise de redes podem ser calculadas, contudo, não possuem relações diretas com o contexto de agrupamento.

Experimento	$CMM - L_{0.3}$							
	CluStream		ClusTree		DenStream	HAStream	CNDenStream	SNCStream
	k-means	DBSCAN	*k-means*	DBSCAN				
RBF_2	0,91	0,87	0,94	0,92	0,98	0,91	0,96	**0,99**
RBF_5	0,99	0,70	0,98	0,84	0,88	0,90	0,93	**0,99**
RBF_{20}	0,97	0,49	**0,99**	0,92	0,89	0,95	0,92	0,96
RBF_{50}	**0,99**	0,50	**0,99**	**0,99**	0,89	0,85	0,93	**0,99**
Two Moon	0,72	0,83	0,66	0,49	0,99	0,82	0,99	**1,00**
Airlines	**0,98**	0,70	0,81	0,80	0,52	0,74	0,87	**0,98**
Electricity	0,77	0,43	0,79	0,46	0,64	0,45	0,88	**0,98**
Forest Covertype	0,80	0,44	0,81	0,45	0,40	0,39	0,81	**0,97**
KDD'98	0,37	0,37	0,37	0,37	0,37	0,37	0,37	**0,38**
KDD'99	0,62	0,53	0,59	0,51	0,72	0,81	0,84	**0,95**
BPaM	0,92	0,64	0,96	0,72	0,77	0,77	0,90	**0,99**

Tabela 5.11: CMM médio obtido durante experimentos utilizando $L_{0,3}$.

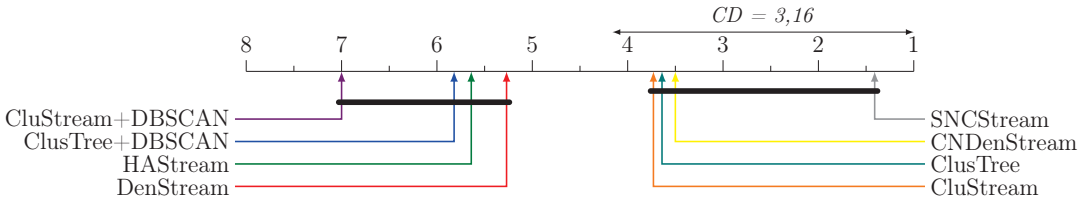


Figura 5.17: Resultados do teste de Nemenyi para o comparativo entre algoritmos adotando a métrica fracionária $L_{0,3}$.

5.9 Considerações Finais

Neste capítulo foram apresentados experimentos realizados com o intuito de comparar o algoritmo SNCStream. Nesta avaliação foram utilizados tanto algoritmos informados baseados em *k-means* quanto não informados, baseados em DBSCAN. Ainda, conjuntos de dados sintéticos e reais foram utilizados, provendo a este estudo experimental rigidez, fortalecida pelo uso de testes não-paramétricos.

Resultados permitem afirmar que os algoritmos proposto são superiores quando comparados a outras abordagens também não informadas em termos de qualidade de agrupamento (CMM) e práticos quando compara-se tempo de processamento e espaço em memória. De todos os experimentos realizados, os algoritmos propostos se mostram inferiores em apenas um caso, contudo, ainda demonstram altos valores de qualidade de agrupamento.

Adicionalmente, foram realizados comparativos do algoritmo SNCStream *versus* abordagens informadas e não informadas, onde o algoritmo proposto se mostrou bastante eficiente, sendo superior às abordagens não informadas e comparável com os algoritmos informados.

O próximo capítulo apresentará as conclusões obtidas deste trabalho, assim como futuros trabalhos.

Experimento	Coeficiente de Agrupamento Médio $\overline{CC}$	Modularidade Média $\overline{Q}$	# de Redes Analisadas
RBF_2	0,73 ± 0,01	0,68 ± 0,01	50
RBF_5	0,65 ± 0,00	0,65 ± 0,01	50
RBF_{20}	0,70 ± 0,02	0,72 ± 0,00	50
RBF_{50}	0,67 ± 0,01	0,68 ± 0,00	50
Two Moon	0,75 ± 0,00	0,66 ± 0,00	1.000
Airlines	0,74 ± 0,04	0,70 ± 0,00	1.000
Forest Covertype	0,73 ± 0,02	0,60 ± 0,00	540
Electricity	0,68 ± 0,01	0,70 ± 0,00	582
KDD'98	0,57 ± 0,03	0,62 ± 0,00	96
KDD'99	0,63 ± 0,02	0,60 ± 0,00	490
BPaM	0,75 ± 0,01	0,78 ± 0,00	165

Tabela 5.12: Coeficiente de agrupamento médio obtido pelo algoritmo SNCStream nos experimentos.

Capítulo 6

Conclusão

Algoritmos para Agrupamento *Online* devem ser capazes de lidar com restrições de tempo e memória sem degradar a qualidade dos grupos obtidos. Ainda, espera-se que estes algoritmos sejam capazes de encontrar grupos não hiper-esféricos, um dos principais desafios de pesquisa atuais.

As abordagens existentes para Agrupamento *Online* são baseadas em uma etapa *online* que tem como objetivo manter sumários estatísticos dos dados mais recentes obtidos do fluxo de dados; e uma etapa *offline*, responsável por encontrar grupos baseando-se nestes sumários. Na etapa *offline*, algoritmos comumente utilizam duas estratégias: utilização do algoritmo *k-means* ou DBSCAN. No primeiro caso, algoritmos recebem o número K de grupos verdadeiros no fluxo de dados (ou em sua sub-partição a ser analisada) para que K grupos sejam encontrados, algo inviável no ambiente de Aprendizagem de Máquina *Online*, seja por assumir que o usuário conhece a distribuição real dos grupos ou devido ao problema de Evolução de Conceito, pois o número de grupos é variável. Por outro lado, na abordagem baseada no DBSCAN, algoritmos recebem um grande número de parâmetros, os quais possuem forte influência nos grupos finais obtidos. Em ambientes reais, é impossível assumir que um processo de otimização de parâmetros seja passível, logo, espera-se que o número de parâmetros e sua influência sobre os resultados obtidos sejam diminuídos.

Neste trabalho foram apresentados os algoritmos CNDenStream e SNCStream. Estes algoritmos utilizam a noção de densidade, herdada do algoritmo DenStream, para manter sumários estatísticos de acordo com uma janela *damped*. O principal diferencial dos algoritmos aqui propostos é a atualização dos grupos em tempo real, ou seja, ainda na etapa *online*, sem necessidade de processamento *batch* na etapa *offline*.

Ambos algoritmos propostos são capazes de atualizar grupos na etapa *online* ao se basear no processo de homofilia, estendendo um modelo de formação e evolução de

redes sociais. Deste modo, para obter grupos respeitando a propriedade *anytime*, estes devem apenas retornar os subgrupos de vértices presentes em sua rede interna. Ainda, por possuir representação baseada em grafo e não possuir restrições de hiper-esfericidade para encontrar grupos, estes algoritmos são capazes de encontrar grupos não hiper-esféricos.

Uma das limitações dos algoritmos CNDenStream e SNCStream é a quantidade de computações de distâncias necessárias para a atualização da rede (inserção e remoção de vértices). Para dirimir o efeito negativo deste grande número de computações de distâncias, estes algoritmos realizam um processo de *memoization*, onde resultados de distâncias calculadas são armazenados em uma estrutura *hash*, permitindo acesso teórico em $\mathcal{O}(1)$, utilizando o função pareamento de Cantor (ROSENBERG, 2002).

Outra questão relevante discutida é sobre os parâmetros (ω, λ, $\mathcal{N}$ e T_p) e seus respectivos impactos no algoritmo SNCStream. Ainda, foram discutidos fatores de poda da rede e seus respectivos impactos na qualidade do agrupamento final.

Finalmente, uma análise permitiu demonstrar que de fato o algoritmo SNCStream original cai na vastidão de espaços de alta dimensionalidade e para sobrepujar tal aspecto, métricas de distância foram avaliadas, determinando então que a métrica fracionária $L_{0,3}$ permite manter altas métricas de qualidade de agrupamento nestes ambientes.

Referências Bibliográficas

ACHTERT, E.; KRIEGEL, H.-P.; REICHERT, L.; SCHUBERT, E.; WOJDANOWSKI, R.; ZIMEK, A. Visual evaluation of outlier detection models. In: *DASFAA (2)*. [S.l.: s.n.], 2010. p. 396–399.

ACKERMANN, M. R.; BLOMER, J.; SOHLER, C. Clustering for metric and nonmetric distance measures. *ACM Trans. Algorithms*, ACM, New York, NY, USA, v. 6, n. 4, p. 59:1–59:26, set. 2010. ISSN 1549-6325. Disponível em: <http://doi.acm.org/1.1145/1824777.1824779>.

ACKERMANN, M. R.; MARTENS, M.; RAUPACH, C.; SWIERKOT, K.; LAMMERSEN, C.; SOHLER, C. Streamkm++: A clustering algorithm for data streams. *J. Exp. Algorithmics*, ACM, New York, NY, USA, v. 17, p. 2.4:2.1–2.4:2.30, maio 2012. ISSN 1084-6654. Disponível em: <http://doi.acm.org/10.1145/2133803.2184450>.

AGGARWAL, C.; YU, P. On clustering techniques for change diagnosis in data streams. In: NASRAOUI, O.; ZAÃ¯ANE, O.; SPILIOPOULOU, M.; MOBASHER, B.; MASAND, B.; YU, P. (Ed.). *Advances in Web Mining and Web Usage Analysis*. Springer Berlin Heidelberg, 2006, (Lecture Notes in Computer Science, v. 4198). p. 139–157. ISBN 978-3-540-46346-7. Disponível em: <http://dx.doi.org/10.1007/11891321_8>.

AGGARWAL, C. C. A framework for diagnosing changes in evolving data streams. In: *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: ACM, 2003. (SIGMOD '03), p. 575–586. ISBN 1-58113-634-X. Disponível em: <http://doi.acm.org/10.1145/872757.872826>.

AGGARWAL, C. C. *Data Streams - Models and Algorithms*. [S.l.]: Springer, 2007. (Advances in Database Systems, v. 31). ISBN 978-0-387-28759-1.

AGGARWAL, C. C.; HAN, J.; WANG, J.; YU, P. S. A framework for clustering evolving data streams. In: *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*. VLDB Endowment, 2003. (VLDB '03), p. 81–92. ISBN 0-12-722442-4. Disponível em: <http://dl.acm.org/citation.cfm?id=1315451.1315460>.

AGGARWAL, C. C.; HINNEBURG, A.; KEIM, D. A. On the surprising behavior of distance metrics in high dimensional space. In: BUSSCHE, J. Van den; VIANU, V. (Ed.). *Database Theory ICDT 2001*. Springer Berlin Heidelberg, 2001, (Lecture Notes in Computer Science, v. 1973). p. 420–434. ISBN 978-3-540-41456-8. Disponível em: <http://dx.doi.org/1.1007/3-540-44503-X_27>.

ALBERT, R.; BARABÁSI, A.-L. Topology of evolving networks: Local events and universality. *Phys. Rev. Lett.*, American Physical Society, v. 85, p. 5234–5237, Dec 2000. Disponível em: <http://link.aps.org/doi/10.1103/PhysRevLett.85.5234>.

ALBERT, R.; BARABÁSI, A. L. Statistical mechanics of complex networks. In: THE AMERICAN PHYSICAL SOCIETY. *Reviews of Modern Physics*. [S.l.], 2002. p. 139–148.

AMINI, A.; WAH, T. Y. Leaden-stream: A leader density-based clustering algorithm over evolving data stream. *Journal of Computer Science and Communications*, Scientific Research, v. 1, n. 5, p. 26–31, 2013.

AMINI, A.; WAH, T. Y. On density-based data streams clustering algorithms: A survey. *Journal of Computer Science and Technology*, Springer US, v. 29, n. 1, p. 116–141, 2014. ISSN 1000-9000. Disponível em: <http://dx.doi.org/10.1007/s11390-014-1416-y>.

BANSAL, S.; BHOWMICK, S.; PAYMAL, P. Fast community detection for dynamic complex networks. In: COSTA, L. da F.; EVSUKOFF, A.; MANGIONI, G.; MENEZES, R. (Ed.). *Complex Networks*. Springer Berlin Heidelberg, 2011, (Communications in Computer and Information Science, v. 116). p. 196–207. ISBN 978-3-642-25500-7. Disponível em: <http://dx.doi.org/10.1007/978-3-642-25501-4_20>.

BARDDAL, J. P.; ENEMBRECK, F. Detecção de mudança de conceito em problemas de regressão utilizando a teoria de redes sociais. In: *ENIAC 2013*. Fortaleza, CE: [s.n.], 2013.

BARDDAL, J. P.; GOMES, H. M.; ENEMBRECK, F. SFNClassifier: A scale-free social network method to handle concept drift. In: *Proceedings of the 29th Annual ACM Symposium on Applied Computing (SAC)*. [S.l.]: ACM, 2014. (SAC 2014).

BARDDAL, J. P.; GOMES, H. M.; ENEMBRECK, F. (to appear): Advances on concept drift detection detection in regression tasks using social networks theory. *International Journal on Natural Computing Research*, p. 1–14, 2015.

BARDDAL, J. P.; GOMES, H. M.; ENEMBRECK, F. (to appear) SNCStream: A social network-based data stream clustering algorithm. In: *Proceedings of the 30th Annual ACM Symposium on Applied Computing (SAC)*. [S.l.]: ACM, 2015. (SAC 2015).

BAYER, R.; MCCREIGHT, E. Organization and maintenance of large ordered indexes. *Acta Informatica*, Springer-Verlag, v. 1, n. 3, p. 173–189, 1972. ISSN 0001-5903. Disponível em: <http://dx.doi.org/10.1007/BF00288683>.

BEYER, K. S.; GOLDSTEIN, J.; RAMAKRISHNAN, R.; SHAFT, U. When is "nearest neighbor" meaningful? In: *Proceedings of the 7th International Conference on Database Theory*. London, UK, UK: Springer-Verlag, 1999. (ICDT '99), p. 217–235. ISBN 3-540-65452-6. Disponível em: <http://dl.acm.org/citation.cfm?id=645503.656271>.

BIFET, A. *Adaptive Stream Mining: Pattern Learning and Mining from Evolving Data Streams*. [S.l.]: IOS Press, 2010. 1-212 p. (Frontiers in Artificial Intelligence and Applications, v. 207). ISBN 978-1-60750-090-2.

BIFET, A.; HOLMES, G.; KIRKBY, R.; PFAHRINGER, B. Moa: Massive online analysis. *The Journal of Machine Learning Research*, v. 11, p. 1601–1604, 2010.

BIFET, A.; PFAHRINGER, B.; READ, J.; HOLMES, G. Efficient data stream classification via probabilistic adaptive windows. In: *SAC*. [S.l.: s.n.], 2013. p. 801–806.

BIFET, A.; READ, J.; ZLIOBAITE, I.; PFAHRINGER, B.; HOLMES, G. Pitfalls in benchmarking data stream classification and how to avoid them. In: *ECML/PKDD (1)*. [S.l.: s.n.], 2013. p. 465–479.

BONNER, R. E. On some clustering techniques. *IBM J. Res. Dev.*, IBM Corp., Riverton, NJ, USA, v. 8, n. 1, p. 22–32, jan. 1964. ISSN 0018-8646.

BRANDES, U. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, v. 25, n. 163, p. 163–177, 2001.

BREUNIG, M. M.; KRIEGEL, H.-P.; NG, R. T.; SANDER, J. Lof: Identifying density-based local outliers. *SIGMOD Rec.*, ACM, New York, NY, USA, v. 29, n. 2, p. 93–104, maio 2000. ISSN 0163-5808. Disponível em: <http://doi.acm.org/10.1145/335191.335388>.

BRZEZINSKI, D.; STEFANOWSKI, J. Reacting to different types of concept drift: The accuracy updated ensemble algorithm. *IEEE Transactions on Neural Networks and Learning Systems*, v. 25, n. 1, p. 81–94, 2013.

CALLAWAY, D. S.; NEWMAN, M. E. J.; STROGATZ, S. H.; WATTS, D. J. Network robustness and fragility: Percolation on random graphs. *Physical Review Letters*, v. 85, p. 5468, dez. 2000.

CAO, F.; ESTER, M.; QIAN, W.; ZHOU, A. Density-based clustering over an evolving data stream with noise. In: *SDM*. [S.l.: s.n.], 2006. p. 328–339.

CERNY, V. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, Springer Netherlands, v. 45, n. 1, p. 41–51, jan. 1985. ISSN 0022-3239. Disponível em: <http://dx.doi.org/10.1007/bf00940812>.

CHEESEMAN, P.; KELLY, J.; SELF, M.; STUTZ, J.; TAYLOR, W.; FREEMAN, D. Autoclass: A bayesian classification system. In: *ML*. [S.l.: s.n.], 1988. p. 54–64.

CHEN, Y.; TU, L. Density-based clustering for real-time stream data. In: *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 2007. (KDD '07), p. 133–142. ISBN 978-1-59593-609-7. Disponível em: <http://doi.acm.org/10.1145/1281192.1281210>.

CORDER, G.; FOREMAN, D. *Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach*. Wiley, 2011. ISBN 9781118211250. Disponível em: <http://books.google.com.br/books?id=T3qOqdpSz6YC>.

DELANY, S. J.; CUNNINGHAM, P.; TSYMBAL, A.; COYLE, L. A case-based technique for tracking concept drift in spam filtering. *Know.-Based Syst.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 18, n. 4-5, p. 187–195, ago. 2005. ISSN 0950-7051.

DOMINGOS, P. A few useful things to know about machine learning. *Commun. ACM*, ACM, New York, NY, USA, v. 55, n. 10, p. 78–87, out. 2012. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/2347736.2347755>.

DONGEN, S. *A Cluster Algorithm for Graphs*. Amsterdam, The Netherlands, The Netherlands, 2000.

DUDA, R. O.; HART, P. E.; STORK, D. G. *Pattern Classification*. [S.l.]: Wiley-Interscience Publication, 2001.

ERDOS, P.; RÉNYI, A. On the evolution of random graphs. In: *Publication of the Mathematical Institute of the Hungarian Academy of Sciences*. [S.l.: s.n.], 1960. p. 17–61.

ESTER, M.; KRIEGEL, H.-P.; SANDER, J.; XU, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In: SIMOUDIS, E.; HAN, J.; FAYYAD, U. M. (Ed.). *KDD*. AAAI Press, 1996. p. 226–231. ISBN 1-57735-004-9. Disponível em: <http://dblp.uni-trier.de/db/conf/kdd/kdd96.html\#EsterKSX96>.

FORTUNATO, S. Community detection in graphs. *Physics Reports*, abs/0906.0612, n. 3-5, p. 75 – 174, 2010.

FRIEDMAN, M. The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *Journal of the American Statistical Association*, American Statistical Association, v. 32, n. 200, p. 675–701, dez. 1937. ISSN 01621459. Disponível em: <http://dx.doi.org/10.2307/2279372>.

GAMA, J. *Knowledge Discovery from Data Streams*. 1st. ed. [S.l.]: Chapman & Hall/CRC, 2010. ISBN 1439826110, 9781439826119.

GAMA, J.; ZLIOBAITE, I.; BIFET, A.; PECHENIZKIY, M.; BOUCHACHIA, A. A survey on concept drift adaptation. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 46, n. 4, p. 44:1–44:37, mar. 2014. ISSN 0360-0300. Disponível em: <http://doi.acm.org/10.1145/2523813>.

GOMES, H. M.; ENEMBRECK, F. Sae: Social adaptive ensemble classifier for data streams. In: *Computational Intelligence and Data Mining (CIDM), 2013 IEEE Symposium on*. [S.l.: s.n.], 2013. p. 199–206.

GOMES, H. M.; ENEMBRECK, F. Sae2: Advances on the social adaptive ensemble classifier for data streams. In: *Proceedings of the 29th Annual ACM Symposium on Applied Computing (SAC)*. [S.l.]: ACM, 2014. (SAC 2014).

GUHA, S. Tight results for clustering and summarizing data streams. In: *Proceedings of the 12th International Conference on Database Theory*. New York, NY, USA: ACM, 2009. (ICDT '09), p. 268–275. ISBN 978-1-60558-423-2.

GUTTMAN, A. R-trees: A dynamic index structure for spatial searching. In: *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: ACM, 1984. (SIGMOD '84), p. 47–57. ISBN 0-89791-128-8. Disponível em: <http://doi.acm.org/10.1145/602259.602266>.

HALL, M.; FRANK, E.; HOLMES, G.; PFAHRINGER, B.; REUTEMANN, P.; WITTEN, I. H. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, ACM,

New York, NY, USA, v. 11, n. 1, p. 10–18, nov. 2009. ISSN 1931-0145. Disponível em: <http://doi.acm.org/10.1145/1656274.1656278>.

HAN, J.; KAMBER, M.; PEI, J. *Data Mining: Concepts and Techniques*. 3rd. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011. ISBN 0123814790, 9780123814791.

HARRIES, M.; WALES, N. S. *SPLICE-2 Comparative Evaluation: Electricity Pricing*. 1999.

HASSANI, M.; KRANEN, P.; SEIDL, T. Precise anytime clustering of noisy sensor data with logarithmic complexity. In: *Proceedings of the Fifth International Workshop on Knowledge Discovery from Sensor Data*. New York, NY, USA: ACM, 2011. (SensorKDD '11), p. 52–60. ISBN 978-1-4503-0832-8. Disponível em: <http://doi.acm.org/10.1145/2003653.2003659>.

HASSANI, M.; SPAUS, P.; GABER, M.; SEIDL, T. Density-based projected clustering of data streams. In: HÃŒLLERMEIER, E.; LINK, S.; FOBER, T.; SEEGER, B. (Ed.). *Scalable Uncertainty Management*. Springer Berlin Heidelberg, 2012, (Lecture Notes in Computer Science, v. 7520). p. 311–324. ISBN 978-3-642-33361-3. Disponível em: <http://dx.doi.org/10.1007/978-3-642-33362-0_24>.

HASSANI, M.; SPAUS, P.; SEIDL, T. Adaptive multiple-resolution stream clustering. In: PERNER, P. (Ed.). *Machine Learning and Data Mining in Pattern Recognition*. Springer International Publishing, 2014, (Lecture Notes in Computer Science, v. 8556). p. 134–148. ISBN 978-3-319-08978-2. Disponível em: <http://dx.doi.org/10.1007/978-3-319-08979-9_11>.

IKONOMOVSKA, E.; GAMA, J.; ZENKO, B.; DZEROSKI, S. Speeding-up hoeffding-based regression trees with options. In: *ICML*. [S.l.: s.n.], 2011. p. 537–544.

JAPKOWICZ, N.; SHAH, M. *Evaluating Learning Algorithms: A Classification Perspective*. New York, NY, USA: Cambridge University Press, 2011. ISBN 0521196000, 9780521196000.

JIANG, H.; YU, Q.; WANG, D. A high-dimensional data stream clustering algorithm based on damped window and pruning list tree. In: *Biomedical Engineering and Informatics (BMEI), 2011 4th International Conference on*. [S.l.: s.n.], 2011. v. 4, p. 2036–2040.

JIANG, N.; GRUENWALD, L. Cfi-stream: Mining closed frequent itemsets in data streams. In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge*

Discovery and Data Mining. New York, NY, USA: ACM, 2006. (KDD '06), p. 592–597. ISBN 1-59593-339-5. Disponível em: <http://doi.acm.org/10.1145/1150402.1150473>.

KAUFMANN, L.; ROUSSEEUW, P. J. *Finding Groups in Data: An Introduction to Cluster Analysis*. 9th. ed. [S.l.]: Wiley-Interscience, 1990. Hardcover. ISBN 0471878766.

KOSINA, P.; GAMA, J. a. Very fast decision rules for multi-class problems. In: *Proceedings of the 27th Annual ACM Symposium on Applied Computing*. New York, NY, USA: ACM, 2012. (SAC '12), p. 795–800. ISBN 978-1-4503-0857-1. Disponível em: <http://doi.acm.org/10.1145/2245276.2245431>.

KRANEN, P.; ASSENT, I.; BALDAUF, C.; SEIDL, T. The clustree: Indexing micro-clusters for anytime stream mining. *Knowl. Inf. Syst.*, Springer-Verlag New York, Inc., New York, NY, USA, v. 29, n. 2, p. 249–272, nov. 2011. ISSN 0219-1377. Disponível em: <http://dx.doi.org/10.1007/s10115-010-0342-8>.

KREMER, H.; KRANEN, P.; JANSEN, T.; SEIDL, T.; BIFET, A.; HOLMES, G.; PFAHRINGER, B. An effective evaluation measure for clustering on evolving data streams. In: *Proc. of the 17th ACM Conference on Knowledge Discovery and Data Mining (SIGKDD 2011), San Diego, CA, USA*. New York, NY, USA: ACM, 2011. p. 868–876.

KRUSKAL, J. B. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. *Proceedings of the American Mathematical Society*, American Mathematical Society, v. 7, n. 1, p. 48–50, fev. 1956. Disponível em: <http://www.jstor.org/stable/2033241>.

LALL, A.; SEKAR, V.; OGIHARA, M.; XU, J.; ZHANG, H. Data streaming algorithms for estimating entropy of network traffic. *SIGMETRICS Perform. Eval. Rev.*, ACM, New York, NY, USA, v. 34, n. 1, p. 145–156, jun. 2006. ISSN 0163-5999. Disponível em: <http://doi.acm.org/10.1145/1140103.1140295>.

LIU, J.; WANG, C.; DANILEVSKY, M.; HAN, J. Large-scale spectral clustering on graphs. In: *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*. AAAI Press, 2013. (IJCAI'13), p. 1486–1492. ISBN 978-1-57735-633-2. Disponível em: <http://dl.acm.org/citation.cfm?id=2540128.2540342>.

LLOYD, S. Least squares quantization in pcm. *IEEE Trans. Inf. Theor.*, IEEE Press, Piscataway, NJ, USA, v. 28, n. 2, p. 129–137, set. 1982. ISSN 0018-9448. Disponível em: <http://dx.doi.org/10.1109/TIT.1982.1056489>.

LUXBURG, U. A tutorial on spectral clustering. *Statistics and Computing*, Kluwer Academic Publishers, Hingham, MA, USA, v. 17, n. 4, p. 395–416, dez. 2007. ISSN 0960-3174. Disponível em: <http://dx.doi.org/10.1007/s11222-007-9033-z>.

MAHALANOBIS, P. C. On the generalised distance in statistics. In: *Proceedings National Institute of Science, India.* [s.n.], 1936. v. 2, n. 1, p. 49–55. Disponível em: <http://ir.isical.ac.in/dspace/handle/1/1268>.

MARKOU, M.; SINGH, S. Novelty detection: A review—part 1: Statistical approaches. *Signal Process.*, Elsevier North-Holland, Inc., Amsterdam, The Netherlands, The Netherlands, v. 83, n. 12, p. 2481–2497, dez. 2003. ISSN 0165-1684. Disponível em: <http://dx.doi.org/10.1016/j.sigpro.2003.07.018>.

MASUD, M. M.; CHEN, Q.; KHAN, L.; AGGARWAL, C. C.; GAO, J.; HAN, J.; THURAISINGHAM, B. M. Addressing concept-evolution in concept-drifting data streams. In: *ICDM.* [S.l.: s.n.], 2010. p. 929–934.

MASUD, M. M.; GAO, J.; KHAN, L.; HAN, J.; THURAISINGHAM, B. M. Classification and novel class detection in concept-drifting data streams under time constraints. *IEEE Trans. Knowl. Data Eng.*, v. 23, n. 6, p. 859–874, 2011.

MICHIE, D. "Memo" Functions and Machine Learning. *Nature*, v. 218, p. 19–22, 1968.

MILGRAM, S. The small world problem. *Psychology Today*, v. 1, n. 1, p. 61–67, May 1967.

MOISE, G.; SANDER, J.; ESTER, M. P3c: A robust projected clustering algorithm. In: *Data Mining, 2006. ICDM '06. Sixth International Conference on.* [S.l.: s.n.], 2006. p. 414–425. ISSN 1550-4786.

MORENO, J. L. *Who shall survive? Who shall survive? : Foundations of Sociometry, Group Psychotherapy, and Sociodrama.* [S.l.]: Nervous and Mental Disease Publishing Corporation, 1934.

NEWMAN, M. E. J. *Networks: an introduction.* Oxford: Oxford University Press, 2010.

NEWMAN, M. E. J.; GIRVAN, M. Finding and evaluating community structure in networks. *Physical Review*, E 69, n. 026113, p. 026113, fev. 2004.

PAPADOPOULOS, S.; KOMPATSIARIS, Y.; VAKALI, A.; SPYRIDONOS, P. Community detection in social media. *Data Min. Knowl. Discov.*, Kluwer Academic Publishers,

Hingham, MA, USA, v. 24, n. 3, p. 515–554, maio 2012. ISSN 1384-5810. Disponível em: <http://dx.doi.org/10.1007/s10618-011-0224-z>.

PARK, N. H.; LEE, W. S. Cell trees: An adaptive synopsis structure for clustering multi-dimensional on-line data streams. *Data Knowl. Eng.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 63, n. 2, p. 528–549, nov. 2007. ISSN 0169-023X. Disponível em: <http://dx.doi.org/10.1016/j.datak.2007.04.003>.

PEREIRA, C. M.; MELLO, R. F. de. A comparison of clustering algorithms for data streams. In: HRUSCHKA ESTEVAM RAFAEL, J.; WATADA, J.; NICOLETTI, M. do C. (Ed.). *Integrated Computing Technology*. Springer Berlin Heidelberg, 2011, (Communications in Computer and Information Science, v. 165). p. 59–74. ISBN 978-3-642-22246-7. Disponível em: <http://dx.doi.org/10.1007/978-3-642-22247-4_6>.

PRIM, R. C. Shortest connection networks and some generalizations. *Bell System Technology Journal*, v. 36, p. 1389–1401, 1957.

RAVASZ, E.; SOMERA, A. L.; MONGRU, D. A.; OLTVAI, Z. N.; BARABáSI, A. L. Hierarchical organization of modularity in metabolic networks. *Science (New York, N.Y.)*, American Association for the Advancement of Science, v. 297, n. 5586, p. 1551–1555, ago. 2002. ISSN 1095-9203. Disponível em: <http://dx.doi.org/10.1126/science.1073374>.

REZENDE, S. *Sistemas inteligentes: fundamentos e aplicações*. [S.l.]: Manole, 2003. ISBN 9788520416839.

RIJSBERGEN, C. J. V. *Information Retrieval*. 2nd. ed. Newton, MA, USA: Butterworth-Heinemann, 1979. ISBN 0408709294.

RODRIGUES, P.; GAMA, J.; PEDROSO, J. Hierarchical clustering of time-series data streams. *Knowledge and Data Engineering, IEEE Transactions on*, v. 20, n. 5, p. 615–627, May 2008. ISSN 1041-4347.

ROSENBERG, A.; HIRSCHBERG, J. V-measure: A conditional entropy-based external cluster evaluation measure. In: *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning(EMNLP-CoNLL)*. [S.l.: s.n.], 2007. p. 410–420.

ROSENBERG, A. L. Efficient pairing functions - and why you should care. In: *Proceedings of the 16th International Parallel and Distributed Processing Symposium*. Washington, DC, USA: IEEE Computer Society, 2002. (IPDPS '02), p. 134–. ISBN 0-7695-1573-8. Disponível em: <http://dl.acm.org/citation.cfm?id=645610.662028>.

ROUSSEEUW, P. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 20, n. 1, p. 53–65, Nov 1987. ISSN 0377-0427.

SABIT, H.; AL-ANBUKY, A.; GHOLAMHOSSEINI, H. Data stream mining for wireless sensor networks environment: Energy efficient fuzzy clustering algorithm. *Int. J. Auton. Adapt. Commun. Syst.*, Inderscience Publishers, Inderscience Publishers, Geneva, SWITZERLAND, v. 4, n. 4, p. 383–397, nov. 2011. ISSN 1754-8632. Disponível em: <http://dx.doi.org/10.1504/IJAACS.2011.043478>.

SCHLIMMER, J. C.; GRANGER, R. H. Incremental learning from noisy data. *Machine Learning*, v. 1, n. 3, p. 317–354, 1986.

SHAKER, A.; HÜLLERMEIER, E. Iblstreams: a system for instance-based classification and regression on data streams. *Evolving Systems*, v. 3, n. 4, p. 235–249, 2012.

SHIOKAWA, H.; FUJIWARA, Y.; ONIZUKA, M. Fast algorithm for modularity-based graph clustering. In: *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA*. [s.n.], 2013. Disponível em: <http://www.aaai.org/ocs/index.php/AAAI/AAAI13/paper/view/6188>.

SILVA, J. A.; FARIA, E. R.; BARROS, R. C.; HRUSCHKA, E. R.; CARVALHO, A. C. P. L. F. d.; GAMA, J. a. Data stream clustering: A survey. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 46, n. 1, p. 13:1–13:31, jul. 2013. ISSN 0360-0300.

STEEN, M. V. *Graph Theory and Complex Networks: An Introduction*. [S.l.]: Maarten Van Steen, 2010. ISBN 9789081540612.

STREHL, A.; GHOSH, J. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res.*, JMLR.org, v. 3, p. 583–617, mar. 2003. ISSN 1532-4435. Disponível em: <http://dx.doi.org/10.1162/153244303321897735>.

TSYMBAL, A. *The problem of concept drift: definitions and related work*. Dublin, Ireland, 2004.

UGULINO, W.; CARDADOR, D.; VEGA, K.; VELLOSO, E.; MILIDIA, R.; FUKS, H. Wearable computing: Accelerometer's data classification of body postures and movements. In: *Advances in Artificial Intelligence - SBIA 2012*. Springer Berlin Heidelberg, 2012, (Lecture Notes in Computer Science). p. 52–61. ISBN 978-3-642-34458-9. Disponível em: <http://dx.doi.org/10.1007/978-3-642-34459-6_6>.

WARD, J. H. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, v. 58, n. 301, p. 236–244, 1963. Disponível em: <http://www.jstor.org/stable/2282967>.

WASSERMAN, S.; FAUST, K. *Social network analysis: Methods and applications*. [S.l.]: Cambridge university press, 1994.

WATTS, D. J.; STROGATZ, S. H. Collective dynamics of small-world networks. *Nature*, v. 393, n. 6684, p. 440–442, June 1998.

WIDMER, G.; KUBAT, M. Learning in the presence of concept drift and hidden contexts. *Mach. Learn.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 23, n. 1, p. 69–101, abr. 1996. ISSN 0885-6125.

YANG, B.; LIU, D. Force-based incremental algorithm for mining community structure in dynamic network. *J. Comput. Sci. Technol.*, v. 21, n. 3, p. 393–400, 2006.

YEON, K.; SONG, M. S.; KIM, Y.; CHOI, H.; PARK, C. Model averaging via penalized regression for tracking concept drift. *Journal of Computational and Graphical Statistics*, v. 19, n. 2, p. 457–473, jun. 2010. ISSN 1061-8600 (print), 1537-2715 (electronic).

ZAFARANI, R.; ABBASI, M.; LIU, H. *Social Media Mining: An Introduction*. Cambridge University Press, 2014. ISBN 9781139916127. Disponível em: <http://books.google.com.br/books?id=fVhzAwAAQBAJ>.

ZHANG, T.; RAMAKRISHNAN, R.; LIVNY, M. Birch: An efficient data clustering method for very large databases. In: *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: ACM, 1996. (SIGMOD '96), p. 103–114. ISBN 0-89791-794-4. Disponível em: <http://doi.acm.org/10.1145/233269.233324>.

ZHAO, Y.; KARYPIS, G. Empirical and theoretical comparisons of selected criterion functions for document clustering. *Mach. Learn.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 55, n. 3, p. 311–331, jun. 2004. ISSN 0885-6125.

ZHONG, C.; MIAO, D.; WANG, R. A graph-theoretical clustering method based on two rounds of minimum spanning trees. *Pattern Recogn.*, Elsevier Science Inc., New York, NY, USA, v. 43, n. 3, p. 752–766, mar. 2010. ISSN 0031-3203. Disponível em: <http://dx.doi.org/10.1016/j.patcog.2009.07.010>.

ZHOU, D.; HUANG, J.; SCHÖLKOPF, B. Learning from labeled and unlabeled data on a directed graph. In: *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005*. [s.n.], 2005. p. 1036–1043. Disponível em: <http://doi.acm.org/10.1145/1102351.1102482>.

ZHOU, H. Distance, dissimilarity index, and network community structure. American Physical Society, v. 67, n. 6, p. 83–91, jun. 2003.

Apêndice A

Relatório MOA: Agrupamento *Online*

PPGIA – PUCPR • 1-13

PPGIA – PUCPR

Relatório MOA: Agrupamento *Online*

Relatório

Jean Paul Barddal, Fabrício Enembreck[1]*

1 Programa de Pós-Graduação em Informática (PPGIA), Pontifícia Universidade Católica do Paraná, R. Imaculada Conceição, 80215-901, Curitiba, Brasil

Resumo: Este relatório tem como objetivo apresentar conceitos básicos sobre a tarefa de Agrupamento *Online* e considerações sobre como incorporar novos algoritmos para esta tarefa dentro do *framework* Massive Online Analysis (MOA). O *framework* MOA é um ambiente de implementação de algoritmos e de execução de experimentos que envolvem *data streams*. O MOA foi concebido para lidar com vários dos problemas relativos a *data streams*, desde a geração de dados sintéticos, implementação de algoritmos do estado da arte e avaliação destes.

Palavras-chave: Mineração de *Data Streams* • Agrupamento • *Framework* Massive Online Analysis

1. Introdução

Avanços recentes em *hardware* e *software* permitiram a aquisição de dados em larga escala. Contudo, os avanços em termos de aquisição não são compatíveis com os de processamento. Logo, lidar com esta massiva quantidade de dados se tornou um desafio para pesquisadores devido às limitações físicas dos computadores atuais.

Na última década, o interesse em gerenciar sequências de dados potencialmente infinitas geradas rapidamente, denominadas **fluxos contínuos de dados** (*data streams*) tem aumentado [Agg07, Gam10]. Aplicações de *data streams* incluem a mineração de dados gerados por redes de sensores [SAAG11], análise da bolsa de valores [BE13] e monitoramento de tráfego de rede [LSO+06]. Todas estas aplicações envolvem conjuntos de dados grandes demais para serem armazenados em memória principal, logo, são armazenadas em memória secundária. Como realizar buscas utilizando acesso aleatório em memória secundária é muito custoso computacionalmente, o único método de acesso plausível é desempenhar acessos lineares aos dados, de acordo com a sua chegada [Guh09].

A extração de conhecimento de *data streams* é um desafio por si só. A maioria das técnicas de mineração de dados e descoberta do conhecimento convencionais assumem que existe uma quantidade da dados conhecida, modelada por uma distribuição de probabilidade estacionária e passível de ser analisada em vários passos por um algoritmo em formato *batch*.

* *E-mail: {jean.barddal, fabricio}@ppgia.pucpr.br*

Para a mineração de *data streams*, deve-se atentar a algumas restrições [SFB+13]. Primeiramente, as instâncias são obtidas continuamente (de maneira serializada) e não existe controle sobre a ordem de processamento destas. Ainda, o tamanho de uma *data stream* é potencialmente infinito, deste modo, instâncias devem ser descartadas logo após seu processamento. Na prática, é possível armazenar uma quantidade destes dados por um período de tempo; contudo, deve-se utilizar um mecanismo de "esquecimento" [WK96] para realizar o descarte posteriormente. Finalmente, a geração de dados desconhecida é possivelmente **não estacionária** (evolucionária), ou seja, sua distribuição de probabilidade pode mudar com o tempo e os algoritmos devem possuir estratégias de detecção e adaptação para tais mudanças.

O desenvolvimento de algoritmos de Agrupamento para *data streams* é um desafio para pesquisadores. Em contraste com o número de algoritmos desenvolvidos para o problema de classificação, a quantidade de algoritmos para Agrupamento é escassa, contudo, tem ganho grande ênfase na comunidade científica [SFB+13].

O objetivo deste relatório é apresentar as características principais da tarefa de Agrupamento *Online* e o funcionamento do *framework* Massive Online Analysis (MOA) dentro deste contexto. Finalmente, um passo a passo de como introduzir um novo algoritmo dentro do MOA será apresentado. A versão do MOA utilizada para este relatório é a **2014.04**. Mudanças de arquitetura interna são constantemente realizadas no *framework* MOA, logo, muitos dos métodos apresentados podem sofrer alterações tanto de nomenclatura, quanto de semântica.

2. Agrupamento *Online*

O desenvolvimento de algoritmos de Agrupamento para *data streams* é um ramo de pesquisa bastante investigado nos últimos anos. Essencialmente, o problema de Agrupamento pode ser descrito como a descoberta de um número finito de *clusters* que descrevam o conjunto de dados. Realizar a tarefa de Agrupamento em *data streams* requer um processo que seja capaz de derivar estes *clusters* de maneira incremental, lidando principalmente com as restrições de memória e tempo descritas anteriormente.

Para desenvolver um algoritmo de Agrupamento para *data streams*, deve-se ter em mente que os dados são apresentados de maneira contínua (incremental) ao algoritmo e não existe controle sobre a ordem de chegada destes dados. Uma *data stream* é potencialmente infinita, logo, armazenar todos os dados em memória primária (ou até mesmo secundária) seria um processo muito custoso tanto em termos de espaço quanto em tempo de processamento. Dessa forma, espera-se que o algoritmo descarte os dados logo após seu processamento (*single pass processing*). Por outro lado, na prática, é possível armazenar uma quantidade limitada de dados, que deve ser descartada de acordo com algum mecanismo de esquecimento [WK96]. Para resolver este problema, pesquisadores buscam desenvolver estruturas de dados não apenas compactas, mas que sejam capazes de evoluir de acordo com a chegada de novos dados. Ainda, estas representações de dados não podem ter sua complexidade associada ao número de instâncias processadas, pois nem mesmo crescimentos lineares do uso de memória são aceitáveis [SFB+13].

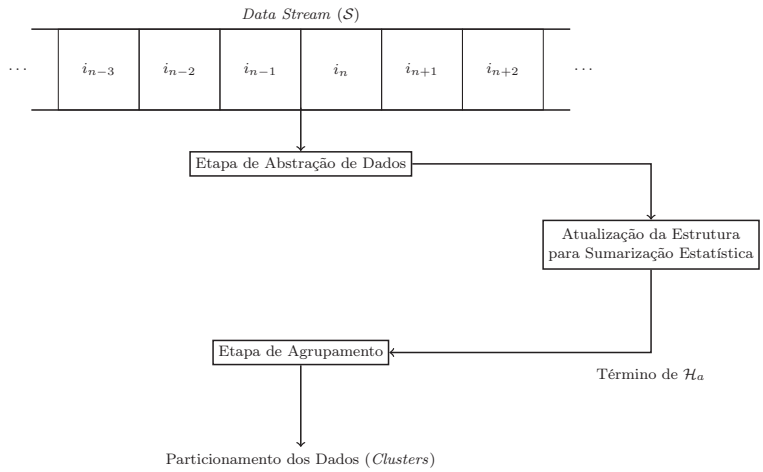


Figura 1. ***Framework* genérico para Agrupamento de *Data Streams*. Adaptado de [SFB+13].**

Por se tratar de uma *data stream* no tempo, não se pode assumir que o processo gerador de dados seja estacionário, onde a distribuição de probabilidade dos dados pode alterar de acordo com o tempo. Desta maneira, espera-se que o algoritmo possua um método de detecção de tais mudanças e de evoluções no conceito, diferenciando estas de dados ruidosos. Finalmente, espera-se que estes algoritmos sejam capazes de lidar com diferentes tipos de dados como árvores XML (eXpression Markup Language), sequências de DNA (DeoxyriboNucleic Acid) e sequências temporais de GPS (Global Positioning System).

Normalmente, os algoritmos de Agrupamento criados atendem apenas algumas destas restrições. Contudo, destaca-se a preocupação com **acuidade**, **tempo**, **espaço em memória** e **detecção e adaptação as mudanças e evoluções de conceito**. Estendendo os desafios apresentados, espera-se que novos algoritmos sejam capazes de: **encontrar grupos de formatos não hiper-esféricos**, tratar dados de alta dimensionalidade, possuir robustez a dados ruidosos, tratar o princípio de minimização de parâmetros e não existir suposições ou informações externas sobre o número de grupos.

Diversos algoritmos para Agrupamento de fluxos contínuos de dados foram desenvolvidos nos últimos anos [KABS11, AHWY03, CEQZ06]. Estes algoritmos são comumente divididos em duas etapas: etapa de abstração de dados (também conhecida como etapa *online*) e etapa de agrupamento (etapa *offline*). A Figura 1 apresenta o *framework* genérico de funcionamento para algoritmos de Agrupamento para *data streams* onde $\mathcal{H}_a$ é o Horizonte de Avaliação. Basicamente, o valor de $\mathcal{H}_a$ define o tamanho de uma janela estática de avaliação. Dentro desta janela, instâncias são obtidas da *data stream* $\mathcal{S}$ e armazenadas em um subconjunto de dados $\mathcal{N}'$, onde $|\mathcal{N}'| = \mathcal{H}_a$, permitindo então que métricas de avaliação sejam aplicadas em $\mathcal{N}'$ [KKJ+11].

A etapa *online* destes algoritmos tem como objetivo sumarizar os dados obtidos da *data stream*, utilizando estruturas de dados específicas desenvolvidas para lidar com as restrições de espaço e tempo. Estas estruturas visam armazenar as principais características dos dados da *data stream*, contudo, sem ter que armazenar todos

Algoritmo	Estrutura de Dados	Modelo de Janela	Método de Detecção de Dados Ruidosos	Algoritmo Base	Formato dos *Clusters*	Referência
BIRCH	*Feature Vector*	*Landmark*	Densidade	*k-means*	Hiper-esférico	[ZRL96]
CluStream	*Feature Vector*	*Landmark*	Estatístico	*k-means*	Hiper-esférico	[AHWY03]
ClusTree	*Feature Vector*	*Damped*	Densidade	*k-means* ou DBSCAN	Arbitrário	[KABS11]
DenStream	*Feature Vector*	*Damped*	Densidade	DBSCAN	Arbitrário	[CEQZ06]
StreamKM++	*Coreset Tree*	*Landmark*	–	*k-means*	Hiper-esférico	[AMR+12]

Tabela 1. Sumário dos principais algoritmos de Agrupamento para *Data Streams*.

os dados em si, o que violaria a restrição de memória e consequentemente, a de tempo. As principais estruturas são: *feature vectors*, *coreset trees*, *data grids* e *prototype arrays*. Dentro do *framework* MOA estão implementadas diversas variações de *feature vectors*, uma *coreset tree* e um *data grid* [SFB+13].

Outro ponto relevante para a sumarização dos dados obtidos da *data stream* é a importância dada a cada instância. Com o intuito de dar maior importância aos dados mais recentes, uma abordagem comum é definir uma janela de tempo que cubra apenas estes dados. Dentro das diversas modelagens para definição de janelas, destaca-se os modelos de janela deslizante, *damped* e *landmark* [SFB+13].

Ainda na etapa *online*, os algoritmos idealmente devem incorporar mecanismos de detecção de dados ruidosos, distinguindo mudanças e evoluções de conceito. Há também algoritmos que realizam a detecção de dados ruidosos apenas na etapa de agrupamento ou na etapa de atualização da estrutura para sumarização estatística, logo, é totalmente dependente da abordagem de cada algoritmo.

Já na etapa *offline*, algoritmos de Agrupamento obtém partições dos dados baseando-se nas estruturas de sumário estatístico computadas na etapa *online* e outros parâmetros fornecidos pelo usuário (e.g. número de *clusters*, horizonte). Como estes algoritmos não lidam com a quantidade massiva de dados vindas do fluxo de dados, mas somente com resumos estatísticos, tais algoritmos acabam sendo bastante eficientes. Logo, algoritmos para formato *batch* como *k-means* e DBSCAN podem ser utilizados e afetam diretamente o formato dos *clusters* a serem encontrados. A Tabela 1 detalha os principais algoritmos para Agrupamento em *data streams*, focando nas estruturas utilizadas, tipos de janelas, métodos de detecção de dados ruidosos, algoritmo base e o formato dos *clusters* encontrados.

3. O *Framework* MOA

O {M}assive {O}nline {A}nalysis (MOA) é um ambiente de implementação de algoritmos e de execução de experimentos que envolvem *data streams*. O MOA foi concebido para lidar com vários dos problemas relativos a *data streams*, desde a geração de dados sintéticos, implementação de algoritmos do estado da arte e avaliação

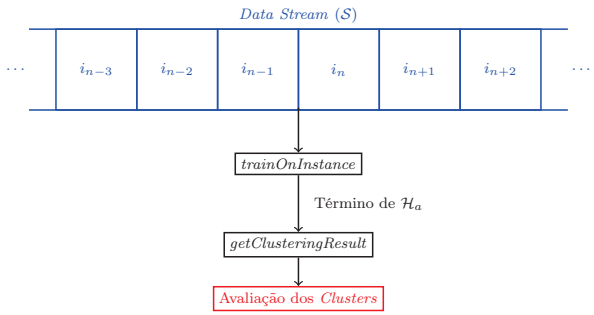


Figura 2. ***Framework* utilizado pelo MOA para Agrupamento de *Data Streams*.**

destes. Deste modo, o MOA acaba contendo uma coleção de algoritmos de Aprendizagem de Máquina bastante vasta, principalmente quando usuários fazem uso da interação com o ambiente WEKA[1].

Nota 3.1.
Assim como o WEKA, o ambiente MOA foi desenvolvido em Java, logo, para realizar implementações de extensões no ambiente MOA, espera-se do leitor entendimento sobre conceitos básicos de Orientação a Objetos e da linguagem de programação Java.

Um ambiente *online* de Aprendizagem de Máquina possui diferentes requisitos quando comparado à abordagem convencional. Os mais significativos, e que levaram a construção do ambiente MOA são:

1. Processar uma instância por vez, e utilizá-la apenas esta vez (na teoria);
2. Utilizar quantidade limitada de memória;
3. Trabalhar com quantidade limitada de tempo;
4. Estar pronto para apresentar respostas para o usuário a qualquer momento (Propriedade *anytime*).

Para entender a dinâmica do ambiente MOA no foco de Agrupamento, deve-se remeter a Figura 2. Na Figura 2, pode-se ver quatro componentes: a *data stream* $\mathcal{S}$ provendo instâncias, a etapa *trainOnInstance*, a etapa *getClusteringResult* e a Avaliação dos *Clusters*.
A *data stream*, em azul, é representada no MOA por objetos derivados das classes *ClusteringStream* e não serão abordados por este relatório. A etapa "Avaliação dos *Clusters*", em vermelho, é representada no MOA por objetos derivados da classe *MeasureCollection*.
Finalmente, as etapas *trainOnInstance* e *getClusteringResult* são implementadas em objetos derivados da classe *AbstractClusterer*. Estas etapas, correspondem, respectivamente, às etapas *online* e *offline* apresentadas anteriormente.

[1] http://www.cs.waikato.ac.nz/ml/weka/

4. Criando um novo Algoritmo de Agrupamento no MOA

Primeiramente, para criar um novo algoritmo de Agrupamento no MOA, é necessário possuir Java JDK 1.6 ou superior, e preferencialmente, um ambiente de desenvolvimento para Java. Não é necessário realizar download do código fonte do ambiente MOA, apenas os pacotes em byte-code são suficientes. Para executar o MOA utilizando as classes desenvolvidas, é necessário apenas informá-las no *classpath* da chamada Java, tanto para uso via interface gráfica quanto em linha de comando.

Inicialmente, deve-se criar uma nova classe que estenda a classe abstrata *AbstractClusterer*. Ainda, esta classe deve estar dentro do pacote *moa.clusterers* ou de algum subpacote subsequente. Ao realizar a extensão da classe *AbstractClusterer*, o usuário deverá realizar a implementação dos seguintes métodos: *resetLearningImpl()*, *trainOnInstanceImpl(Instance instnc)*, *getModelMeasurementsImpl()*, *getModelDescription(StringBuilder sb, int i)*, *isRandomizable()*, *getVotesForInstance(Instance instnc)*, *getClusteringResult()*.

A descrição de cada método é apresentada nas próximas seções. Finalmente, a Seção 4.8 apresentará como introduzir parâmetros nos algoritmos. A Figura 3 apresenta a estrutura básica necessária para um código de algoritmo de Agrupamento dentro do *framework* MOA.

4.1. *resetLearningImpl()*

O método *resetLearningImpl()* tem como objetivo realizar a limpeza de estruturas internas do algoritmo. Basicamente, este método deve repetir tudo que ocorre dentro do método construtor. Nenhum tipo de alteração deve ser feita nos parâmetros fornecidos pelo usuário dentro desta etapa.

4.2. *trainOnInstanceImpl(Instance instnc)*

O método *trainOnInstanceImpl(Instance instnc)* tem como objetivo realizar a etapa *online* do algoritmo, ou seja, atualizar os sumários estatísticos internos fazendo-se uso da instância fornecida como parâmetro *instnc*.

A implementação deste método é altamente dependente das estruturas internas utilizadas, logo, nenhum exemplo será apresentado. Para visualização de exemplos, sugere-se a leitura dos códigos-fonte de algoritmos disponíveis em `https://code.google.com/p/moa/source/checkout`.

4.3. *getModelMeasurementsImpl()*

O método *getModelMeasurementsImpl()* tem como objetivo fornecer ao MOA métricas internas do algoritmo para geração do arquivo de resultados. Para retornar resultados de métricas dentro do MOA, um padrão deve ser seguido.

É necessário primeiramente inicializar um vetor do tipo *Measurement* e alimentá-lo com instâncias desta mesma classe. Cada *Measurement* é composto por uma *String*, que apresenta textualmente seu significado, e um valor *double*, que representa o valor da métrica calculado. **Não é possível exportar métricas não numéricas**

```java
package moa.clusterers;

import moa.cluster.Clustering;
import moa.core.Measurement;
import weka.core.Instance;

/**
 * @author jeanpaul
 */
public class NovoAlgAgrupamento extends AbstractClusterer{

    //OPTIONS (PARAMETROS) DEVEM SER COLOCADOS AQUI

    public NovoAlgAgrupamento() {
    }

    @Override
    public void resetLearningImpl() {
    }

    @Override
    public void trainOnInstanceImpl(Instance instnc) {
    }

    @Override
    protected Measurement[] getModelMeasurementsImpl() {
    }

    @Override
    public void getModelDescription(StringBuilder sb, int i) {
    }

    @Override
    public boolean isRandomizable() {
    }

    @Override
    public double[] getVotesForInstance(Instance instnc) {
    }

    @Override
    public Clustering getClusteringResult() {
    }
}
```

Figura 3. **Exemplo de Classe de Agrupamento**

dentro do MOA fazendo-se uso dos métodos internos do *framework*.

A Figura 4 apresenta um exemplo de implementação deste método, onde existe uma estrutura interna que armazena *microclusters* e outra que armazena a quantidade de instâncias já utilizadas pelo algoritmo.

4.4. *getModelDescription(StringBuilder sb, int i)*

O método *getModelDescription(StringBuilder sb, int i)* tinha como objetivo apresentar graficamente a descrição do algoritmo. Atualmente, ela não está em funcionamento, e deve ser substituída pelo método *public String*

```
//DEFINICOES DE CLASSE
ArrayList<MicroCluster> microClusters;
long qtdInstancesSeen;

@Override
protected Measurement[] getModelMeasurementsImpl() {
    Measurement measures[] = new Measurement[2];
    measures[0] = new Measurement("# of microclusters", microClusters.size());
    measures[1] = new Measurement("# of instances seen", qtdInstancesSeen);
    return measures;
}
```

Figura 4. **Exemplo de Extração de Medidas Internas dentro do MOA**

```
@Override
public void getModelDescription(StringBuilder sb, int i) {

}

@Override
public String getPurposeString() {
    return "An implementation of CNDenStream using JUNG API.";
}
```

Figura 5. **Exemplo de Implementação do método *getPurposeString()*.**

getPurposeString(), onde uma *String* descrevendo o método pode ser retornada.

A Figura 5 apresenta um exemplo de implementação deste método e a Figura 6 apresenta o seu resultado graficamente.

4.5. *isRandomizable()*

Este método tem como objetivo informar o MOA se este algoritmo possui comportamento estocástico (*return true;*) ou não (*return false;*). No caso positivo, um parâmetro será automaticamente adicionado ao seu algoritmo: *randomSeed*, ou seja, a semente de geração de valores aleatórios. Deste modo, este algoritmo **deve** fazer uso do objeto derivado *clusterRandom*, que gera variáveis aleatórias de diversos tipos a partir das chamadas (*nextDouble()*,

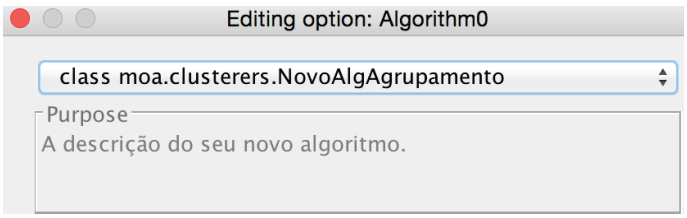


Figura 6. **Propósito do Algoritmo.**

```
@Override
public Clustering getClusteringResult() {
    Clustering myClusters = new Clustering();
    myClusters.add(new SphereCluster(new double[]{0.0, 0.0}, 2));
    myClusters.add(new SphereCluster(new double[]{0.3, 0.2}, 2));
    return myClusters;
}
```

Figura 7. **Exemplo de funcionamento do método *getClusteringResult()*.**

nextBoolean(), nextFloat(), nextGaussian(), nextInt(), nextLong()). Caso o usuário realize uso de componentes aleatórias utilizando outro tipo de chamada, como por exemplo: *Math.random()*, o algoritmo **não** possuirá o comportamento aleatório esperado e baseado no parâmetro *randomSeed*.

4.6. *getVotesForInstance(Instance instnc)*

O método *getVotesForInstance(Instance instnc)*, assim como o método *getModelDescription(StringBuilder sb, int i)*, não é mais utilizado. Ele foi substituído pelo método *getClusteringResult()*.

4.7. *getClusteringResult()*

O método *getClusteringResult()* tem como objetivo realizar a etapa *offline* do algoritmo de Agrupamento, ou seja, encontrar os *clusters*, processando os sumários estatísticos.

Assim como o método *trainOnInstanceImpl()*, este método é altamente dependente das estruturas de sumários estatísticos utilizados, e novamente, sugere-se ao leitor a averiguar implementações apresentadas em https://code.google.com/p/moa/source/checkout. Contudo, existem implementações do algoritmo *k-means* e *DBSCAN* já implementadas no MOA e facilmente utilizáveis.

A Figura 7 apresenta um exemplo de código onde dois *clusters* esféricos são montados, um com centro nas coordenadas [0.0, 0.0] e outro com centro em [0.3, 0.2] e ambos com duas dimensões.

4.8. Adicionando Parâmetros

Para alimentar a implementação dos métodos anteriormente descritos, muitas vezes, parâmetros devem ser fornecidos pelos usuários. No MOA, o usuário é capaz de inserir os seguintes tipos de parâmetros: de classe (*ClassOption*, de arquivo (*FileOption*), de valor lógico (*FlagOption*), de ponto flutuante (*FloatOption*), de inteiro (*IntOption*), de lista (*ListOption*), de múltipla escolha (*MultiChoiceOption*) e de texto (*StringOption*).

Todo e qualquer parâmetro definido dentro do MOA é conhecido como *Option*. Ainda, toda instância deste tipo deve ter nome com sufixo Option. Exemplos: minPointsOption, betaOption.

Nota 4.1.
Atenção! Todo e qualquer parâmetro que não termine com o sufixo Option não aparecerá na interface gráfica

```
//ClassOption(name, cliChar, purpose, ClassType, defaultValueAsString);
public ClassOption exampleClassOption = new ClassOption("exampleClass", 'c', "The purpose for this
    parameter.", AbstractClusterer.class, "moa.clusterers.social.CNDenStream");

//FileOption(name, cliChar, purpose, defaultFileName, defaultExtension, isOutput);
public FileOption exampleFileOption = new FileOption("exampleFile", 'f', "The purpose for this file.",
    "/defaultFile", ".csv", true);

//FlagOption(name, cliChar, purpose);
public FlagOption exampleFlagOption = new FlagOption("exampleFlag", 'b', "Purpose for this flag
    option.");

//FloatOption(name, cliChar, purpose, defaultValue, minValue, maxValue);
public FloatOption exampleFloatOption = new FloatOption("exampleFloat", 'd', "Purpose for this float
    option.", 0.1, 0.1, 1.0);

//IntOption(name, cliChar, purpose, defaultValue, minValue, maxValue);
public IntOption exampleIntOption = new IntOption("exampleInt", 'i', "Purpose for this int option.",
    1, 1, 10);

public IntOption anotherIntOption = new IntOption("exampleInt2", '2', "Purpose for this second int
    option.", 1, 1, 10);

//public ListOption a = new ListOption(null, cliChar, null, exampleIntOption, defaultList,
    separatorChar)
public ListOption exampleListOption = new ListOption("exampleList", 'l', "Purpose for this list of
    options.", exampleIntOption, new Option[]{exampleIntOption, anotherIntOption}, ';');

//MultiChoiceOption(name, cliChar, purpose, optionLabels[], optionDescriptions[], defaultIndex);
public MultiChoiceOption exampleMultiChoiceOption = new MultiChoiceOption("exampleMultiChoice", 'm',
    "Purpose for this Multi choice option.", new String[]{"Option 1", "Option 2"}, new
    String[]{"Option 1 executes X.", "Option 2 executes Y."}, 0);

//StringOption(name, cliChar, purpose, defaultValue);
public StringOption exampleStringOption = new StringOption("exampleString", 's', "Purpose for this
    string option.", "myString");
```

Figura 8. **Exemplo de Opções dentro do MOA**

do MOA, assim como não será acessível em execuções via linha de comando. Ainda, todas as instâncias *Option* devem possuir escopo *public*.

A Figura 8 apresenta um trecho de código que exemplifica a definição de cada tipo de parâmetro possível dentro do framework MOA. As Figuras 9 e 10 apresentam a interface gráfica gerada e a Figura 11 apresenta os detalhes de cada parâmetro. Para acessar o valor definido pelo usuário dentro do código, deve-se utilizar o método *getValue()*.

5. Conclusão

Neste relatório foi apresentado o *framework* para mineração de *data streams* Massiva Online Analysis. Ainda, uma breve fundamentação sobre o problema de Agrupamento *Online* foi apresentada, permitindo entender as etapas deste processo, e a sua respectiva lógica de implementação dentro do MOA. Além disso, considerações

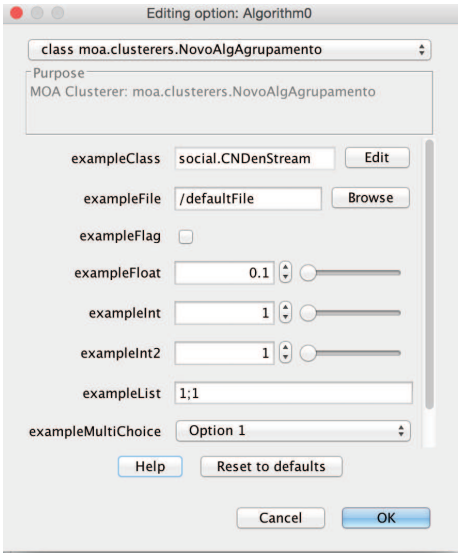


Figura 9. **Parâmetros apresentados na interface gráfica.**

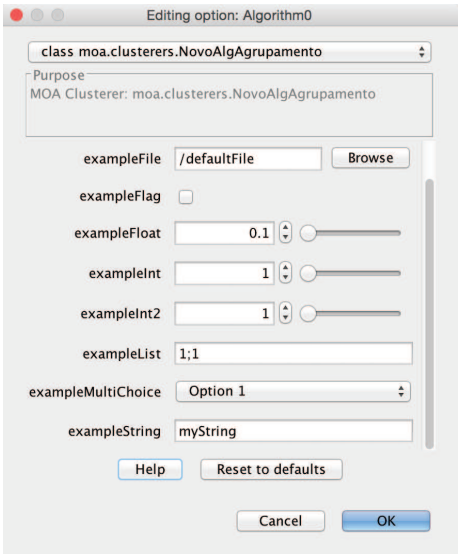


Figura 10. **Segunda Parte dos Parâmetros Apresentados na Interface Gráfica.**

sobre parametrizações e técnicas de implementação foram descritas, com o intuito de auxiliar o leitor a evitar erros bastante comuns. Com este relatório, pode-se perceber a facilidade provida pelo *framework* MOA para pesquisadores desta área, permitindo-os testar novas hipóteses para algoritmos rapidamente.

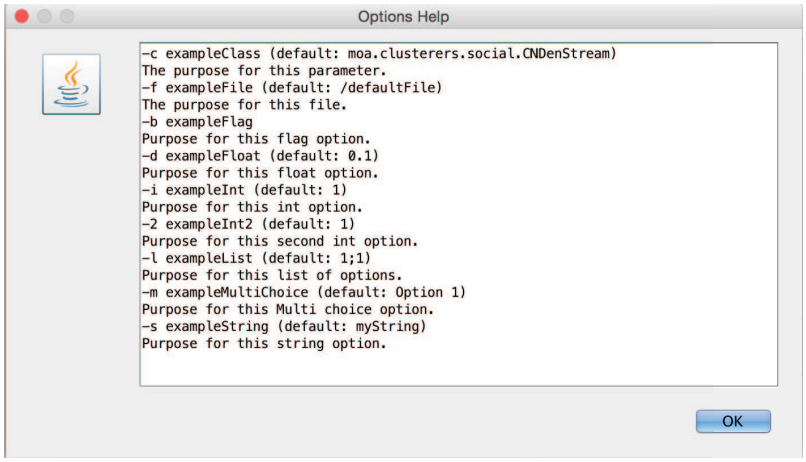


Figura 11. **Detalhamento dos parâmetros.**

Referências

[Agg07] Charu C. Aggarwal. *Data Streams - Models and Algorithms*, volume 31 of *Advances in Database Systems*. Springer, 2007.

[AHWY03] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. A framework for clustering evolving data streams. In *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*, VLDB '03, pages 81–92. VLDB Endowment, 2003.

[AMR+12] Marcel R. Ackermann, Marcus Märtens, Christoph Raupach, Kamil Swierkot, Christiane Lammersen, and Christian Sohler. Streamkm++: A clustering algorithm for data streams. *ACM Journal of Experimental Algorithmics*, 17(1), 2012.

[BE13] Jean Paul Barddal and Fabrício Enembreck. Detecção de mudança de conceito em problemas de regressão utilizando a teoria de redes sociais. In *ENIAC 2013*, Fortaleza, CE, Oct 2013.

[CEQZ06] Feng Cao, Martin Ester, Weining Qian, and Aoying Zhou. Density-based clustering over an evolving data stream with noise. In *SDM*, pages 328–339, 2006.

[Gam10] Joao Gama. *Knowledge Discovery from Data Streams*. Chapman & Hall/CRC, 1st edition, 2010.

[Guh09] Sudipto Guha. Tight results for clustering and summarizing data streams. In *Proceedings of the 12th International Conference on Database Theory*, ICDT '09, pages 268–275, New York, NY, USA, 2009. ACM.

[KABS11] Philipp Kranen, Ira Assent, Corinna Baldauf, and Thomas Seidl. The clustree: Indexing micro-clusters for anytime stream mining. *Knowl. Inf. Syst.*, 29(2):249–272, November 2011.

[KKJ+11] Hardy Kremer, Philipp Kranen, Timm Jansen, Thomas Seidl, Albert Bifet, Geoff Holmes, and Bernhard Pfahringer. An effective evaluation measure for clustering on evolving data streams. In *Proc. of the 17th ACM Conference on Knowledge Discovery and Data Mining (SIGKDD 2011), San Diego, CA, USA*, pages 868–876, New York, NY, USA, 2011. ACM.

[LSO+06] Ashwin Lall, Vyas Sekar, Mitsunori Ogihara, Jun Xu, and Hui Zhang. Data streaming algorithms for estimating entropy of network traffic. *SIGMETRICS Perform. Eval. Rev.*, 34(1):145–156, June 2006.

[SAAG11] Hakilo Sabit, Adnan Al-Anbuky, and Hamid Gholamhosseini. Data stream mining for wireless sensor networks environment: Energy efficient fuzzy clustering algorithm. *Int. J. Auton. Adapt. Commun. Syst.*, 4(4):383–397, November 2011.

[SFB+13] Jonathan A. Silva, Elaine R. Faria, Rodrigo C. Barros, Eduardo R. Hruschka, André C. P. L. F. de Carvalho, and João Gama. Data stream clustering: A survey. *ACM Comput. Surv.*, 46(1):13:1–13:31, July 2013.

[WK96] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Mach. Learn.*, 23(1):69–101, April 1996.

[ZRL96] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: An efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, SIGMOD '96, pages 103–114, New York, NY, USA, 1996. ACM.

Apêndice B

Resultados Para Outras Métricas de Qualidade de Agrupamento

Neste apêndice são apresentados os resultados obtidos pelos algoritmos para as seguintes métricas de qualidade de agrupamento:

- Soma do Quadrado das Distâncias – SSQ;
- Homogeneidade;
- Completude;
- *V-Measure*;
- Pureza;
- *Precision*;
- *Recall*;
- Coeficiente da Silhueta;
- Informação Mútua Normalizada – NMI,

Ressalta-se neste ponto que nenhuma das métricas apresentadas neste apêndice são tidas como apropriadas para a tarefa de Agrupamento *Online* (KREMER et al., 2011), contudo, são comumente utilizadas para avaliação de algoritmo em trabalhos propostos na literatura.

Experimento	*SSQ*							
	CluStream		ClusTree		DenStream	HAStream	CNDenStream	SNCStream
	k-means	DBSCAN	*k-means*	DBSCAN				
RBF_2	13,71	12,00	12,64	12,71	**10,48**	12,99	13,18	13,04
RBF_5	43,48	**10,63**	42,61	11,87	33,19	34,05	83,86	32,23
RBF_{20}	235,42	41,56	235,4	42,98	83,81	127,76	296,12	**12,17**
RBF_{50}	681,54	13,20	680,08	14,82	**11,45**	294,65	17,89	16,81
Two Moon	43,89	10,72	44,23	14,62	32,10	29,44	12,10	**9,25**
Airlines	**30,87**	41,94	40,12	40,31	82,25	52,48	53,48	54,72
Electricity	95,98	89,23	97,89	89,23	79,11	90,76	71,90	**69,99**
Forest Covertype	6,85	8,50	7,68	9,20	**6,45**	8,21	10,59	9,62
KDD'98	102,81	84,50	102,43	**82,14**	95,14	98,44	88,80	83,90
KDD'99	243,05	133,22	283,69	172,23	**123,53**	197,96	123,98	159,19
BPaM	14,58	**8,27**	15,19	9,67	12,79	12,44	10,43	8,43

Tabela B.1: *SSQ* médio obtido nos experimentos.

Experimento	Homogeneidade							
	CluStream		ClusTree		DenStream	HAStream	CNDenStream	SNCStream
	k-means	DBSCAN	*k-means*	DBSCAN				
RBF_2	0,64	0,77	0,92	0,82	0,41	0,77	**0,96**	0,88
RBF_5	**1,00**	0,12	**1,00**	0,75	0,36	0,68	0,58	0,94
RBF_{20}	**1,00**	0,99	**1,00**	0,79	0,30	0,81	0,70	0,92
RBF_{50}	0,68	0,76	**0,95**	0,82	0,41	0,74	0,70	0,85
Two Moon	0,55	0,77	0,64	0,77	0,20	0,66	0,82	**0,88**
Airlines	0,32	**0,99**	0,28	0,82	0,17	0,53	0,29	0,84
Electricity	0,99	0,78	**0,99**	0,62	0,88	0,87	0,90	0,92
Forest Covertype	0,96	0,77	**1,00**	0,68	0,22	0,60	0,02	0,56
KDD'98	0,00	0,88	0,00	0,73	0,47	0,44	0,00	**0,97**
KDD'99	**1,00**	0,88	**1,00**	0,57	0,34	0,79	0,76	1,00
BPaM	0,01	**1,00**	0,33	**1,00**	**1,00**	0,71	0,64	**1,00**

Tabela B.2: Homogeneidade média obtida nos experimentos.

Experimento	Completude							
	CluStream		ClusTree		DenStream	HAStream	CNDenStream	SNCStream
	k-means	DBSCAN	*k-means*	DBSCAN				
RBF_2	0,85	0,42	0,94	0,42	0,68	0,71	0,92	**0,96**
RBF_5	**0,99**	0,63	**0,99**	0,54	0,63	0,81	0,93	0,96
RBF_{20}	**0,99**	0,76	**0,99**	0,69	0,63	0,86	0,96	**0,99**
RBF_{50}	0,92	0,23	0,94	0,38	0,67	0,72	0,91	**0,99**
Two Moon	0,32	0,61	0,37	0,53	0,63	0,62	**0,94**	**0,94**
Airlines	0,47	0,74	0,42	0,65	0,60	0,55	0,84	**0,93**
Electricity	0,73	0,65	0,72	0,72	0,81	0,81	0,81	**0,91**
Forest Covertype	0,50	0,29	0,50	0,22	0,43	0,63	0,77	**0,82**
KDD'98	**0,99**	0,42	**0,99**	0,37	0,79	0,92	**0,99**	**0,99**
KDD'99	**0,99**	0,34	**0,99**	0,27	0,19	0,64	0,73	0,93
BPaM	**0,99**	0,82	**0,99**	0,89	**0,99**	0,89	0,91	**0,99**

Tabela B.3: Completude média obtida nos experimentos.

Experimento	*V-Measure*							
	CluStream		ClusTree		DenStream	HAStream	CNDenStream	SNCStream
	k-means	DBSCAN	*k-means*	DBSCAN				
RBF_2	0,72	0,30	0,93	0,54	0,48	0,67	**0,96**	0,77
RBF_5	0,99	0,50	**0,99**	0,62	0,45	0,72	0,56	0,94
RBF_{20}	0,99	0,92	**0,99**	0,73	0,39	0,84	0,82	**0,99**
RBF_{50}	0,75	0,33	**0,94**	0,46	0,47	0,64	0,82	0,73
Two Moon	0,46	0,51	0,52	0,61	0,26	0,47	0,48	**0,88**
Airlines	**0,99**	**0,99**	**0,99**	0,70	0,22	0,82	**0,99**	0,82
Electricity	**0,99**	0,71	**0,99**	0,75	0,87	0,90	**0,99**	**0,99**
Forest Covertype	0,96	**0,99**	**0,99**	**0,99**	**0,99**	**0,99**	**0,99**	0,94
KDD'98	0,94	0,97	0,98	0,88	0,98	0,96	**0,99**	**0,99**
KDD'99	**0,99**	0,88	0,99	0,65	0,99	0,93	0,99	0,99
BPaM	0,22	**0,99**	0,52	**0,99**	**0,99**	0,82	**0,99**	**0,99**

Tabela B.4: *V-Measure* médio obtido nos experimentos.

Experimento	Pureza							
	CluStream		ClusTree		DenStream	HAStream	CNDenStream	SNCStream
	k-means	DBSCAN	*k-means*	DBSCAN				
RBF_2	0,89	**0,98**	**0,98**	0,96	0,90	0,92	0,79	0,96
RBF_5	**0,99**	**0,99**	**0,99**	**0,99**	**0,99**	0,98	0,86	**0,99**
RBF_{20}	**0,99**	**0,99**	**0,99**	**0,99**	**0,99**	0,97	0,80	**0,99**
RBF_{50}	**0,99**	0,97	0,99	0,96	0,91	0,94	0,80	0,97
Two Moon	0,95	**0,99**	**0,99**	**0,99**	**0,99**	0,96	0,80	**0,99**
Airlines	0,63	**0,99**	0,63	**0,99**	**0,99**	0,84	0,63	**0,99**
Electricity	0,90	0,76	0,90	0,77	0,75	0,82	0,70	**0,99**
Forest Covertype	0,70	**0,75**	0,70	0,66	0,73	0,71	0,66	0,74
KDD'98	0,37	**0,99**	0,37	0,97	0,98	0,69	0,87	**0,97**
KDD'99	0,76	**0,95**	0,79	0,83	0,90	0,88	0,99	0,90
BPaM	0,78	0,65	0,76	0,73	0,73	0,77	0,78	**0,99**

Tabela B.5: Pureza média obtida nos experimentos.

Experimento	*Precision*							
	CluStream		ClusTree		DenStream	HAStream	CNDenStream	SNCStream
	k-means	DBSCAN	*k-means*	DBSCAN				
RBF_2	0,79	0,41	0,99	0,43	0,61	0,70	0,79	**0,89**
RBF_5	**1,00**	0,32	**1,00**	0,55	0,47	0,73	0,78	0,97
RBF_{20}	**1,00**	0,28	**1,00**	0,64	0,45	0,73	0,80	0,96
RBF_{50}	0,89	0,41	**0,95**	0,42	0,61	0,71	0,80	0,88
Two Moon	0,82	0,32	0,84	0,55	0,45	0,65	0,63	**0,92**
Airlines	0,35	0,28	0,36	0,66	0,42	0,49	0,49	**0,88**
Electricity	0,60	0,55	0,70	0,68	0,72	0,68	0,71	**0,81**
Forest Covertype	0,66	0,46	0,70	0,55	0,55	0,61	0,66	**0,71**
KDD'98	0,41	0,23	0,43	0,21	0,23	0,40	0,62	**0,66**
KDD'99	0,63	0,57	0,66	0,57	0,60	0,62	0,61	**0,68**
BPaM	0,29	0,20	0,28	0,18	0,22	0,29	0,32	**0,55**

Tabela B.6: *Precision* médio obtido nos experimentos.

Experimento	*Recall*							
	CluStream		ClusTree		DenStream	HAStream	CNDenStream	SNCStream
	k-means	DBSCAN	*k-means*	DBSCAN				
RBF_2	0,69	0,50	**0,75**	0,58	0,60	0,63	0,60	0,71
RBF_5	0,75	0,43	0,75	0,56	0,55	0,65	0,67	**0,81**
RBF_{20}	0,67	0,25	0,67	0,63	0,53	0,55	0,27	**0,80**
RBF_{50}	0,47	0,50	0,64	0,58	0,58	0,53	0,27	**0,69**
Two Moon	0,82	0,44	**0,84**	0,57	0,53	0,68	0,75	0,78
Airlines	0,44	0,27	0,44	0,68	0,51	0,53	0,57	**0,77**
Electricity	0,55	0,37	0,59	0,51	0,52	0,53	0,50	**0,69**
Forest Covertype	0,57	0,46	0,59	0,49	0,52	0,55	0,57	**0,62**
KDD'98	0,25	0,15	0,29	0,19	0,31	0,27	**0,36**	**0,36**
KDD'99	0,61	0,52	0,63	0,58	0,55	0,60	**0,65**	0,63
BPaM	0,26	0,21	0,27	0,19	0,47	0,35	0,33	**0,72**

Tabela B.7: *Recall* médio obtido nos experimentos.

Experimento	Coeficiente da Silhueta							
	CluStream		ClusTree		DenStream	HAStream	CNDenStream	SNCStream
	k-means	DBSCAN	*k-means*	DBSCAN				
RBF_2	0,89	0,54	**0,95**	0,61	0,75	0,78	0,87	0,84
RBF_5	**0,98**	0,85	0,98	0,69	0,87	0,90	0,93	0,97
RBF_{20}	**0,99**	0,97	**0,99**	0,78	0,87	0,93	0,90	0,97
RBF_{50}	**0,99**	0,56	**0,99**	0,60	0,75	0,80	0,90	0,80
Two Moon	0,87	0,86	0,92	0,69	0,87	0,85	0,79	**0,95**
Airlines	0,67	**0,97**	0,69	0,79	0,87	0,81	0,75	0,94
Electricity	0,78	0,66	0,79	0,64	0,72	0,75	0,81	**0,88**
Forest Covertype	0,72	0,20	0,73	0,44	0,30	0,58	0,71	**0,98**
KDD'98	0,50	0,60	0,50	0,73	0,74	0,65	0,50	**0,97**
KDD'99	0,60	0,44	0,60	0,53	0,49	0,61	0,70	**0,94**
BPaM	0,55	0,57	0,53	0,53	0,71	0,60	0,43	**0,85**

Tabela B.8: Coeficiente da silhueta médio obtido nos experimentos.

Experimento	*NMI*							
	CluStream		ClusTree		DenStream	HAStream	CNDenStream	SNCStream
	k-means	DBSCAN	*k-means*	DBSCAN				
RBF_2	0,84	0,64	**0,99**	0,73	0,68	0,80	0,83	0,86
RBF_5	**1,00**	0,49	**1,00**	0,73	0,61	0,82	0,95	0,98
RBF_{20}	**1,00**	0,75	**1,00**	0,79	0,59	0,86	0,94	0,97
RBF_{50}	0,84	0,77	**1,00**	0,77	0,70	0,86	0,94	0,97
Two Moon	0,82	0,61	**0,97**	0,58	0,58	0,77	0,89	0,92
Airlines	**0,75**	0,60	0,78	0,66	0,69	0,70	0,71	0,73
Electricity	0,64	0,61	**0,69**	0,68	0,66	0,66	0,67	**0,69**
Forest Covertype	0,65	0,59	0,69	0,61	0,10	0,58	0,70	**0,72**
KDD'98	**0,37**	**0,37**	**0,37**	**0,37**	**0,37**	**0,37**	0,36	**0,37**
KDD'99	0,64	0,55	0,51	0,61	0,70	0,65	0,77	**0,79**
BPaM	0,88	0,75	0,87	0,79	0,78	0,84	0,88	**0,91**

Tabela B.9: NMI médio obtido nos experimentos.