Combinando semi-supervisão e *hubness* para aprimorar o agrupamento de dados em alta dimensão

Mateus Curcino de Lima

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

Uberlândia
2017

Mateus Curcino de Lima

Combinando semi-supervisão e *hubness* para aprimorar o agrupamento de dados em alta dimensão

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

Área de concentração: Ciência da Computação

Orientadora: Maria Camila Nardini Barioni

Uberlândia
2017

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

L732c 2017

Lima, Mateus Curcino de, 1992-

Combinando semi-supervisão e hubness para aprimorar o agrupamento de dados em alta dimensão / Mateus Curcino de Lima. - 2017.

90 f. : il.

Orientador: Maria Camila Nardini Barioni.

Dissertação (mestrado) - Universidade Federal de Uberlândia, Programa de Pós-Graduação em Ciência da Computação.

Inclui bibliografia.

1. Computação - Teses. 2. Mineração de dados (Computação) - Teses. 3. Banco de dados - Teses. I. Barioni, Maria Camila Nardini. II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU: 681.3

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Os abaixo assinados, por meio deste, certificam que leram e recomendam para a Faculdade de Computação a aceitação da dissertação intitulada "**Combinando semi-supervisão e *hubness* para aprimorar o agrupamento de dados em alta dimensão**" por **Mateus Curcino de Lima** como parte dos requisitos exigidos para a obtenção do título de **Mestre em Ciência da Computação**.

Uberlândia, 23 de janeiro de 2017

Orientadora: _____
Prof.ª Dr.ª Maria Camila Nardini Barioni
Universidade Federal de Uberlândia

Banca Examinadora:

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

_____
Prof.ª Dr.ª Marcela Xavier Ribeiro
Universidade Federal de São Carlos

Dedico este trabalho aos meus pais, José Bastos e Alair,
por todo apoio, amor e incentivo durante cada desafio.

Agradecimentos

Agradeço primeiramente a Deus pela vida e por possibilitar a realização deste sonho.

Obrigado à minha orientadora Prof[a]. Maria Camila pelos ensinamentos, paciência, incentivo e disponibilidade que tornaram possível a conclusão deste trabalho. Também agradeço o Prof. Humberto por todas as orientações que muito contribuíram para o desenvolvimento da dissertação.

Agradeço a todos os professores e funcionários do PPGCO, em especial ao Erisvaldo, pelo grande auxílio nas questões administrativas.

Palavras não são o bastante para agradecer aos meus pais, José Bastos e Alair, por todo o apoio durante o mestrado e por não medirem esforços para que eu chegasse até esta etapa da vida.

Agradeço imensamente à minha namorada Nathália por todo o companheirismo em questões pessoais, de trabalho ou acadêmicas.

Também sou grato a todos os meus familiares e amigos de laboratório e do IFTM, que auxiliaram diretamente ou indiretamente, mas de uma maneira inesquecível.

Agradeço à CAPES e ao CNPq pelo apoio financeiro durante a realização do trabalho.

A todos, o meu muito obrigado!

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

Resumo

A chamada *maldição da dimensionalidade* faz com que a análise de dados em alta dimensão seja uma tarefa desafiadora para técnicas de agrupamento de dados. Para tratar desta questão, trabalhos recentes têm empregado de forma eficiente um aspecto inerente de dados de alta dimensão na realização de processos de agrupamentos de dados. Esse aspecto, denominado *hubness*, consiste na tendência de algumas instâncias de dados, chamadas *hubs*, ocorrerem com maior frequência nas listas dos K-vizinhos mais próximos de outras instâncias. Contudo, os *hubs* podem não refletir a semântica implícita dos dados, levando a uma partição de dados inadequada. Esta dissertação apresenta uma abordagem de agrupamento que explora a combinação de duas estratégias: semi-supervisão e estimativa de densidade baseada em pontuações *hubness*. Os resultados dos experimentos realizados com 23 conjuntos de dados reais mostram que a abordagem proposta tem um desempenho superior quando aplicada em conjuntos de dados com características diferentes.

Palavras-chave: Agrupamento semi-supervisionado. Análise de dados em alta dimensão. *Hubness*. Mineração de dados.

Abstract

The curse of dimensionality turns the high-dimensional data analysis a challenging task for data clustering techniques. Recent works have efficiently employed an aspect inherent to high-dimensional data in the proposal of clustering approaches guided by *hubs* which provide information about the distribution of the data instances among the K-nearest neighbors. Though, *hubs* can not well reflect the implicit semantics of the data, leading to an unsuitable data partition. In order to cope with both issues (i.e., high-dimensional data and meaningful clusters), this dissertation presents a clustering approach that explores the combination of two strategies: semi-supervision and density estimation based on *hubness* scores. The experimental results conducted with 23 real datasets show that the proposed approach has a good performance when applied on datasets with different characteristics.

Keywords: Data mining. High-dimensional data analysis. Hubness. Semi-supervised clustering.

Lista de ilustrações

Lista de tabelas

Lista de siglas

DBSCAN	*Density-Based Spatial Clustering of Applications with Noise*
HPC	*Hubness-Proportional Clustering*
HPKM	*Hubness-Proportional k-means*
SSDBSCAN	*Semi-Supervised DBSCAN*
SSHub Clustering	*Semi-Supervised Hubness-based Clustering*

Lista de símbolos

k	Números de grupos
$\pi = \{C_1; ...; C_k\}$	Partições geradas
$X = \{x_1; ...; x_n\}$	Conjunto de dados
K	Vizinhança para cálculo dos K vizinhos mais próximos
$h_K(x)$	Pontuação *hubness* da instância x considerando a vizinhança K
$N = \{h_K(x_1); ...; h_K(x_n)\}$	Conjunto das pontuações *hubness* das intâncias
p_i	Representante principal do grupo C_i
a_i	Representante auxiliar do grupo C_i
A_i	Conjunto de representantes auxiliares do grupo C_i
Q_i	Conjunto de representantes (principal e auxiliares) do grupo C_i
$r_{ml}(x_i, x_j)$	Restrição *must-link* entre x_i e x_j
$r_{cl}(x_i, x_j)$	Restrição *cannot-link* entre x_i e x_j
R_m	Conjunto de r_{ml}
R_c	Conjunto de r_{cl}
R	$R_{ml} \cup R_{cl}$
$\bar{R}_i$	*Rank* médio do i-ésimo algoritmo
R_i	Soma dos *ranks* do i-ésimo algoritmo
χ_F^2	Estatística de *Friedman*
F_F	Derivação da estatística de *Friedman*
CD	Diferença crítica
z	Diferença de desempenho entre dois algoritmos

Sumário

Capítulo 1

Introdução

Existem vários domínios de aplicação para os quais o emprego de técnicas de mineração de dados é útil em que a dimensionalidade dos dados é notavelmente elevada. Dentre eles estão bancos de dados, onde cada instância de dados é descrita por uma coleção de características, como no caso de imagens e de expressões gênicas. A chamada *maldição da dimensionalidade* (SAMET, 2005) faz com que a análise de dados em alta dimensão seja uma tarefa desafiadora para qualquer técnica de mineração de dados baseada em cálculos de distância, uma vez que a medida em que a dimensionalidade aumenta, também aumentam a esparsidade dos dados e a dificuldade de diferenciar instâncias de dados. Nesse contexto é interessante considerar o emprego de técnicas que permitam atenuar esses efeitos prejudiciais a eficiência e a eficácia de algoritmos de mineração de dados.

Tradicionalmente, a questão da alta dimensionalidade tem sido tratada na literatura científica da área com a utilização de estratégias que procuram atenuar os efeitos da *maldição da dimensionalidade*, realizando a redução de dimensionalidade por meio de extração e de seleção de atributos ou propondo métodos que trabalham em subespaços. Entretanto, informações podem ser perdidas quando a dimensionalidade é reduzida (SILVESTRE, 2007). De uma maneira oposta, estratégias mais recentes têm empregado de maneira eficaz um aspecto, denominado *hubness*, inerente aos dados de alta dimensão na proposta de técnicas que permitem a classificação (TOMASEV; MLADENIC, 2013)(BUZA, 2016), a busca de vizinhos mais próximos (FLEXER; SCHNITZER, 2013)(TOMASEV; MLADENIC, 2014) e o agrupamento (TOMASEV et al., 2014)(TOMASEV et al., 2015) em bancos de dados de alta dimensão. O aspecto *hubness* consiste na tendência de algumas instâncias de dados, chamadas *hubs*, ocorrerem com maior frequência nas listas dos K-vizinhos mais próximos de outras instâncias.

Apesar dessas técnicas terem demonstrado que agrupamentos de dados orientados pelas informações da pontuação *hubness* das instâncias de dados apresentam bons resultados, quando aplicados a dados de alta dimensão, é importante destacar que os *hubs* são centros de influência, sendo que possíveis imprecisões associadas a eles podem ser facilmente propagadas (TOMASEV; MLADENIC, 2013). Uma estratégia que pode ser utilizada

para minimizar o risco de induzir um particionamento não adequado nos dados, uma vez que *hubs* podem não refletir bem a semântica implícita dos dados, é a incorporação de semi-supervisão (BASU; DAVIDSON; WAGSTAFF, 2008).

Esta dissertação apresenta a proposta de uma abordagem de agrupamento que explora a combinação de estratégias de semi-supervisão e de utilização das pontuações *hubness* a respeito das instâncias de dados com foco em dados de alta dimensão. Essa abordagem revisitou o algoritmo *HPKM* (TOMASEV et al., 2011) dando origem ao método denominado *SSHub Clustering* (*Semi-supervised hubness-based clustering*). O presente trabalho também apresenta os resultados obtidos com um conjunto exaustivo de experimentos que utilizou 23 conjuntos de dados reais. Analisando os resultados experimentais obtidos é possível constatar que o *SSHub Clustering* obteve bom desempenho quando aplicado em conjuntos de dados reais de diferentes tamanhos e dimensões, superando a eficácia de quatro algoritmos selecionados como linha de base para comparação.

1.1 Objetivos

O objetivo direto do trabalho realizado descrito nesta dissertação consistiu em abordar o estudo e o desenvolvimento de um ferramental teórico e prático que contribuísse para a obtenção de resultados mais acurados em métodos de detecção semi-supervisionada de agrupamentos no contexto de dados de alta dimensão. Para tanto, foram exploradas estratégias para:

- Combinar o aspecto *hubness* e técnicas de semi-supervisão para a tarefa de detecção de agrupamentos;
- Utilizar *aprendizado ativo* para selecionar instâncias mais representantivas como representantes principais de grupos, com base nas pontuações *hubness*;
- Analisar elementos de fronteira dos grupos para a obtenção de informação adicional na forma de restrições em nível instância.

O foco do trabalho descrito nesta dissertação é apresentar o método desenvolvido e analisar sua eficácia e eficiência, destacando principalmente a qualidade do particionamento de dados gerado em conjuntos de dados de alta dimensão. A partir disso, pode-se destacar as contribuições apresentadas na Seção 1.2.

1.2 Principais contribuições

As principais contribuições do trabalho descrito aqui são:

- A criação de um novo método de detecção de agrupamentos com foco em conjuntos de dados de alta dimensão, que considera a incorporação de técnicas de semi-supervisão e do aspecto *hubness* para a obtenção de particionamentos que condizem melhor com a real estrutura do conjunto de dados;
- O oferecimento de uma alternativa aos métodos existentes, com um novo método que tem a capacidade de trabalhar com conjuntos de dados de estruturas complexas, pois considera vários representantes em cada grupo, o que possibilita realizar processos de detecção de agrupamentos mais acurados em conjuntos de dados que possuem diferentes características, isto é, variadas quantidades de instâncias, grupos e dimensões.

1.3 Organização da dissertação

Esta dissertação está organizada em cinco capítulos e um Apêndice, como mostrado a seguir:

- Capítulo 2: apresenta os conceitos fundamentais para o entendimento do trabalho descrito aqui, dentre eles: detecção de agrupamentos, semi-supervisão, alta dimensão e aspecto *hubness*. Além disso, esse capítulo também discute os principais trabalhos relacionados;
- Capítulo 3: apresenta os detalhes do algoritmo de detecção de agrupamentos, que incorpora o aspecto *hubness* e semi-supervisão, criado durante a realização do trabalho descrito aqui;
- Capítulo 4: apresenta os resultados dos experimentos realizados para comparar o algoritmo *SSHub Clustering* frente a quatro outros algoritmos da literatura. Nesse capítulo são discutidos a análise da eficácia e da eficiência do algoritmo proposto;
- Capítulo 5: apresenta a conclusão desta dissertação com as considerações finais e propostas para possíveis trabalhos futuros. Além disso, destaca as publicações geradas com a pesquisa desenvolvida;
- Apêndice A: apresenta as tabelas de valores críticos para avaliação estatística realizada nos experimentos.

Capítulo 2

Fundamentação Teórica

De uma maneira geral, o objetivo da tarefa de agrupamento de dados consiste em encontrar grupos de acordo com uma medida de similaridade de maneira que as instâncias de dados de um mesmo grupo possuam alta similaridade, enquanto instâncias de dados de grupos diferentes possuam baixa similaridade (AGGARWAL; REDDY, 2013). Para computar essa medida são necessárias: uma descrição das instâncias de dados e uma função de distância. Dependendo do domínio de dados, as instâncias podem ser descritas por um conjunto de atributos de domínios tradicionais (por exemplo, textuais ou numéricos) ou por uma coleção pré-definida de descritores de características inerentes aos dados (considerando o domínio de imagens, por exemplo: cor, textura e forma, entre outros). A escolha da função de distância a ser empregada também depende do domínio de dados manipulado, sendo que dentre as mais comumente usadas estão as funções de distância da família *Minkowski* (TANIAR; IWAN, 2011).

É importante destacar que a expressividade dessa medida de similaridade é um fator importante para a obtenção de bons resultados em processos de agrupamento de dados. Quando se lida com dados em alta dimensão essa característica desejável das medidas de similaridade se degrada devido aos efeitos da assim chamada *maldição da dimensionalidade* (SAMET, 2005). Para tentar atenuar os efeitos indesejáveis dessa maldição em processos de agrupamento de dados em alta dimensão é possível empregar duas abordagens: aplicar técnicas de redução de dimensionalidade como um primeiro passo antes de realizar o agrupamento dos dados (veja Seção 2.2.1); ou utilizar soluções especializadas para analisar dados de alta dimensão (veja Seção 2.2.2).

Outro aspecto que também pode interferir na expressividade das medidas de similaridade surge quando é necessário lidar com domínios de dados mais complexos, como imagens. Isso é devido à existência do assim chamado *gap semântico* entre as características de baixo nível que representam as instâncias de dados e o alto nível da percepção humana, o que faz com que agrupamentos obtidos com base apenas nas características de baixo nível possam não corresponder a noção do que é mais similar para um usuário. Dentre as estratégias propostas para lidar com essa questão estão as técnicas de agrupamento

semi-supervisionado de dados (veja Seção 2.1.2).

O capítulo está organizado da seguinte maneira: a Seção 2.1 descreve a tarefa de detecção de agrupamentos, incluindo medidas de similidade e semi-supervisão; a Seção 2.2 descreve formas de tratar dados de alta dimensão, com foco principal no aspecto *hubness*; a Seção 2.3 descreve os algoritmos selecionados como linha de base para comparação; a Seção 2.4 descreve os métodos de avaliação utilizados para avaliar a qualidade dos particionamentos gerados. Por fim, a Seção 2.5 apresenta as considerações finais do capítulo.

2.1 Detecção de agrupamentos

A detecção de agrupamentos é uma das principais tarefas da área de pesquisa em mineração de dados, sendo bastante utilizada em diversas aplicações, tais como: psicologia e outras ciências sociais, biologia, estatísticas, reconhecimento de padrões, recuperação de informação e aprendizado de máquina (TAN; STEINBACH; KUMAR, 2006). De uma maneira geral, o processo de detecção de agrupamentos pode ser descrito como a divisão de um conjunto de instâncias (elementos de dados) em grupos, de acordo com uma medida de similaridade. As instâncias de dados de um grupo devem possuir propriedades comuns, o que possibilita a distinção com outros subconjuntos e maximiza a similaridade das instâncias de um mesmo grupo (HAN; KAMBER, 2006; ZAKI; JR, 2014). Tradicionalmente, a detecção de agrupamentos é realizada de maneira não-supervisionada (ou semi-supervisionado em alguns modelos, detalhado na Seção 2.1.2), isto é, sua avaliação é intrínseca, objetivando descobrir um conjunto de grupos (MAIMON; ROKACH, 2005).

A definição de um grupo é imprecisa. Para entender melhor a dificuldade de decidir o que constitui um grupo, a Figura 1 mostra três maneiras diferentes para dividir vinte instâncias. As Figuras 1(b) e 1(d) dividem as instâncias em dois e seis grupos, respectivamente. No entanto, a divisão aparente de cada um dos dois conjuntos maiores em três subgrupos pode ser simplesmente um artefato do sistema visual humano. Além disso, pode-se dizer que as instâncias formam quatro grupos, como mostrado na Figura 1(c) (TAN; STEINBACH; KUMAR, 2006).

O conceito de grupo é de difícil definição, pois depende da natureza dos dados e dos resultados desejados. A seguir, diferentes definições comumente utilizadas para o termo (FACELI; LORENA; GAMA, 2011):

- **Grupo bem separado:** qualquer instância em um grupo está mais próxima à todas as outras do mesmo grupo do que qualquer outra instância fora dele. Um *threshold* pode ser usado para especificar que todas instâncias em um grupo devem estar suficientemente próximas umas das outras;

- **Grupo baseado em centro:** qualquer instância em grupo está mais próxima do "centro" de seu grupo do que do centro de outro grupo. Esse centro geralmente é

um centróide, que é a média de todas as instâncias em um grupo, ou um medóide, que é a instância mais representativa do grupo;

- **Grupo contínuo:** uma instância em um grupo está mais próxima a uma ou mais instâncias nesse grupo do que para qualquer instância fora dele;
- **Grupo baseado em densidade:** um grupo é uma região densa de instâncias, o qual é separado por regiões de baixa densidade, a partir de outras regiões de alta densidade. Esta definição é frequentemente usada quando os grupos são irregulares, e quando *outliers* estão presentes;
- **Grupo baseado em similaridade:** é um grupo de instâncias que são similares, por meio de alguma medida de similaridade, em relação a instâncias em outros grupos que não são similares. Esse tipo de grupo define um conjunto de instâncias que, quando juntas, criam uma região com propriedade local uniforme, por exemplo, densidade ou forma.

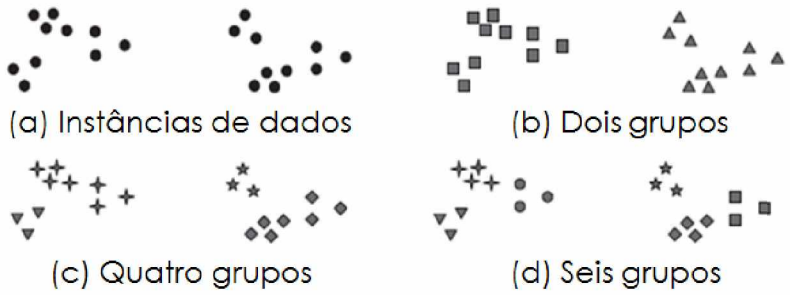

Figura 1 – Diferentes maneiras de agrupar instâncias. Adaptado de (TAN; STEINBACH; KUMAR, 2006).

Devido ao fato da noção de grupo não ser precisamente definida, existem diversos algoritmos para a detecção de agrupamentos e cada um deles pode assumir um critério que resulta em algum tipo específico de estrutura nos agrupamentos formados (TAN; STEINBACH; KUMAR, 2006). Consequentemente, muitos métodos de agrupamento são desenvolvidos, sendo que cada um utiliza um princípio de indução diferente. As principais abordagens de agrupamento são: hierárquica (MANNING; RAGHAVAN; SCHüTZE, 2008), baseada em densidade (TAN; STEINBACH; KUMAR, 2006), baseada em grade (GAN; MA; WU, 2007) e particionamento (HAN; KAMBER, 2006). O foco do trabalho descrito aqui está nessa última abordagem que é detalhada na Seção 2.1.1.

2.1.1 Detecção de agrupamentos por particionamento

O conceito de partições é baseado na teoria de conjuntos, sendo que uma partição dos dados consiste em uma divisão de um conjunto de instâncias em subconjuntos menores.

Considere um conjunto $X = \{x_1; ...; x_n\}$, com n instâncias de dados, e um dado valor k que corresponde ao número de grupos desejado, um algoritmo de particionamento organiza as instâncias em k partições $\pi = \{C_1; ...; C_k\}$ com $k < n$, de maneira que cada partição representa um agrupamento e (XU; WUNSCH II, 2005):

- $C_j \neq \emptyset$, $j = 1, ..., k$: todos os grupos contêm pelo menos uma instâncias de dados;
- $\bigcup_{j=1}^{k} C_j = X$: todas as instâncias de dados pertencem a algum grupo;
- $C_j \bigcap C_l = \emptyset$, $j, l = 1, ..., k$ e $j \neq l$: cada instância de dados pertence exclusivamente a um único grupo.

De maneira geral, os algoritmos de particionamento utilizam centróides para representar cada partição, isto é, cada grupo possui um centróide representante, dado pela média aritmética de todas as instâncias pertencentes a esse grupo, como demonstrado na Equação 1:

$$\mu_j = \frac{1}{n_j} \sum_{x_i \in C_j} x_i \tag{1}$$

onde n_j é o número de instâncias do um grupo C_j. Outras abordagens utilizam medóides, ou seja, a instância mais representativa de cada grupo (FACELI; LORENA; GAMA, 2011).

O processo de particionamento de instâncias em grupos é guiado por uma medida de similaridade baseada em distância, de modo que as instâncias dentro de um agrupamento são "semelhantes", e que as instâncias de diferentes grupos são "diferentes" em termos de atributos (HAN; KAMBER, 2006). Para identificar o centróide μ_j mais próximo de uma instância x_i deve-se considerar uma medida de similaridade (veja Seção 2.1.1.1).

Um dos algoritmos que empregam a abordagem de agrupamento por particionamento mais conhecidos é o *k-means* (MACQUEEN, 1967). Conforme detalhado no Algoritmo 1, o *k-means* inicia selecionando k centróides, um centróide por grupo (linha 2 do Algoritmo 1). Após essa etapa, ele utiliza uma abordagem iterativa para a atribuição de instâncias aos grupos, ou seja, a cada iteração o algoritmo atribui cada instância x_i do conjunto de dados X a um grupo C_j, tal que o centróide μ_j seja o mais próximo (linha 5 do Algoritmo 1) (MACQUEEN, 1967). O critério de convergência do *k-means* (linha 8 do Algoritmo 1) é baseado no erro quadrático, ilustrado na Equação 2:

$$E = \sum_{j=1}^{k} \sum_{x_i \in C_j} \delta(x_i, \mu_j)^{(2)} \tag{2}$$

onde μ_j é o centróide do grupo C_j, como definido na Equação 1, e $\delta()$ é uma função de distância.

O algoritmo *k-means* possui como uma de suas características ser sensível a escolha dos centróides iniciais. Além disso, ele consegue detectar os agrupamentos somente se

separados de maneira linear. Para minimizar essa deficiência, agravada principalmente na manipulação de dados de alta dimensão, foi proposto o algoritmo *Kernel k-means* (DHILLON; GUAN; KULIS, 2004) para lidar com problemas não-linearmente separáveis. Mais detalhes do algoritmo *Kernel k-means* são apresentados na Seção 2.3.1.

Algoritmo 1: *k-means*

Entrada: Conjunto de dados X, Número de grupos k
Saída: $\pi = \{C_1, ..., C_k\}$

```
início
    Selecione aleatoriamente μ_1, ..., μ_k como centróides iniciais;
    repita
        para cada instância x_i ∈ X faça
            Atribua x_i ao grupo C_j com μ_j mais próximo;
        para cada grupo C_i faça
            Atualize seus centróides μ_i pela média de todas as instâncias x_j que
            foram atribuídos a ele;
    até convergir ;
    retorna π
fim
```

2.1.1.1 Medidas de similaridade

Conforme descrito nas considerações iniciais deste capítulo, para calcular as medidas de similaridade são necessárias: uma descrição das instâncias de dados e uma função de distância. Formalmente, a descrição das instâncias de dados pode ser representada por um vetor de características (ou atributos) d-dimensional, onde d corresponde ao número de características de cada instância de dados e define a dimensionalidade das instâncias de dados. A escolha da função de distância a ser empregada irá depender do domínio de dados manipulado. A seguir, são apresentadas as funções de distância da família *Minkowski*:

- **Distância Euclidiana:** é considerada a distância mais utilizada para dados numéricos. Para duas instâncias de dados x e y, no espaço d-dimensional, a distância Euclidiana entre elas é definida como:

$$d(x,y) = \sqrt{\sum_{j=1}^{d} (x_j - y_j)^2} \tag{3}$$

onde x_j e y_j são os valores do j-ésimo atributo de x e y, respectivamente.

- **Distância Manhattan:** também conhecida como distância *City-Block*. É definida como sendo a soma das distâncias de todos os atributos. Isto é, para duas instâncias de dados x e y, no espaço d-dimensional, a distância entre elas é:

$$d(x,y) = \sum_{j=1}^{d} |x_j - y_j| \tag{4}$$

  A distância de *Manhattan* é muito usada para lidar com atributos discretos, por exemplo, valores binários (FACELI; LORENA; GAMA, 2011).

- **Distância Máxima:** também é chamada de distância *Sup*. Ela é definida como sendo o valor máximo das distâncias dos atributos, isto é, para duas instâncias de dados x e y, no espaço d-dimensional, a distância máxima entre elas é:

$$d(x,y) = \max_{1 \leq j \leq d} |x_j - y_j| \tag{5}$$

Existem outras diversas medidas de similaridade propostas para outros diferentes domínios de dados. Uma visão geral sobre essas funções de distância pode ser encontrada em (COLLINS; OKADA, 2012) e (SAAD; KAMARUDIN, 2013).

2.1.2 Agrupamento semi-supervisionado de dados

Diferente da abordagem tradicional de agrupamento de dados, a abordagem de agrupamento semi-supervisionado conta com alguma informação adicional que guia o processo de divisão das instâncias de dados em grupos. Essa informação adicional pode ser obtida por meio da rotulação de uma pequena porção do conjunto de dados (CHAPELLE; SCHLKOPF; ZIEN, 2010) ou informada na forma de restrições nos dados (DAVIDSON; BASU, 2007)(XIONG; AZIMI; FERN, 2014).

As restrições podem ser definidas de diferentes formas, por exemplo: em nível instância (WAGSTAFF; CARDIE, 2000), em nível atributo (SCHMIDT; BRANDLE; KRAMER, 2011), em nível grupo (DUBEY; BHATTACHARYA; GODBOLE, 2010), relativa (KUMAR; KUMMAMURU, 2008) e ranqueada (AHMED; NABLI; GARGOURI, 2012). Dentre esses tipos de restrição, as restrições de nível instância são o tipo mais empregado por algoritmos de agrupamento baseados em particionamento melhorando de forma significativa o desempenho desses algoritmos (DUBEY; BHATTACHARYA; GODBOLE, 2010).

As restrições de nível instância são divididas em dois tipos: restrições *must-link* e restrições *cannot-link*. Restrições do tipo *must-link* definem que um par de instâncias deve pertencer ao mesmo grupo, já as restrições do tipo *cannot-link* definem que um par de instâncias não deve pertencer ao mesmo grupo (WAGSTAFF; CARDIE, 2000). Formalmente, considerando um conjunto de instâncias de dados $X = \{x_1; ...; x_n\}$, uma

restrição *must-link* $r_{ml}(x_1, x_2)$ indica que as instâncias x_1 e x_2 devem estar no mesmo grupo, já a restrição *cannot-link* $r_{cl}(x_1, x_3)$ indica que as instâncias x_1 e x_3 não devem estar no mesmo grupo. A Figura 2(a) ilustra exemplos de definição de restrições em nível instância.

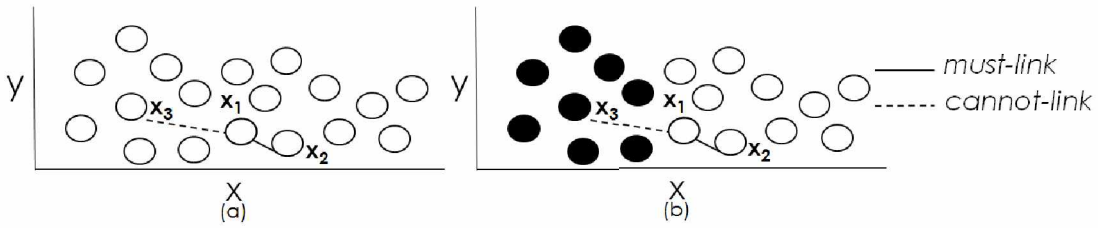

Figura 2 – (a) Exemplo de definição de restrições *must-link* e *cannot-link* em um conjunto de dados. (b) Exemplo de particionamento gerado que respeita as restrições definidas em (a).

As restrições *must-link* e *cannot-link*, embora aparentemente simples, possuem propriedades interessantes como as descritas em (DAVIDSON; BASU, 2007). As restrições *must-link* são um exemplo de relação de equivalência e, portanto, são simétricas, reflexivas e transitivas; isto significa que considerando as instâncias de dados x_1, x_2 e x_4 e a definição das restrições $r_{ml}(x_1, x_2)$ e $r_{ml}(x_2, x_4)$ isto implica que $r_{ml}(x_1, x_4)$, ou seja, que x_1, x_2 e x_4 formam uma componente conexa, isto é, cada instância de dados está ligada a outra por meio de uma explícita ou implícita restrição *must-link*.

Além disso, as restrições *must-link* e *cannot-link* são poderosas e podem especificar propriedades espaciais. Por exemplo, considerando a situação em que se deseja que dois grupos tenham suas instâncias separadas por uma distância δ ou maior, pode-se utilizar restrições *must-link* entre todas as instâncias com distâncias inferiores a δ. De maneira similar, em um cenário em que se deseja que os diâmetros dos grupos sejam no máximo σ, pode-se utilizar um conjunto de restrições *cannot-link* entre todas as instâncias com distâncias superiores a σ.

As abordagens existentes para a incorporação das restrições informadas nos algoritmos de agrupamento de dados podem ser divididas em duas categorias: baseada em similaridade (COHN; CARUANA; MCCALLUM, 2003)(HU et al., 2010) e baseada em busca (BASU; BANERJEE; MOONEY, 2002)(LI et al., 2014)(SILVA et al., 2015). As estratégias que empregam a abordagem baseada em similaridade alteram a medida de similaridade usada no processo de agrupamento, buscando satisfazer os rótulos ou restrições nos dados de maneira que instâncias, que devem estar juntas em um mesmo agrupamento, se aproximem e instâncias, que não devem estar juntas, se afastem. Já as estratégias que empregam a abordagem baseada em busca usam os rótulos ou restrições fornecidos pelos usuários, com o objetivo de orientar o processo de detecção de agrupamentos para uma partição de dados mais semelhante a real estrutura do conjunto de dados (BARIONI et al., 2014). O trabalho descrito nesta dissertação se enquadra nessa última categoria.

Outra questão importante está relacionada a seleção de instâncias de dados para a obtenção das restrições *must-link* e *cannot-link*. Quando essa seleção é realizada aleatoriamente os agrupamentos formados podem não atingir a estrutura desejada. Para resolver esta limitação, estratégias de *aprendizado ativo* podem ser consideradas. Com *aprendizado ativo* o algoritmo pode solicitar o rótulo de instâncias específicas. Essas estratégias tem como objetivo selecionar as instâncias (ou pares de instâncias) mais significativas para rotular e com isso podem permitir a obtenção de bons resultados de agrupamento com um número menor de instâncias analisadas (LI et al., 2014). Por exemplo, considerando que a instância x_1, presente na Figura 2(a), é significativa para o particionamento do conjunto de dados, pode-se criar as restrições tomando-a como base e guiar para um particionamento mais adequado que respeite as restrições definidas, como apresentado na Figura 2(b).

Exemplos de trabalhos correlatos que consideram a utilização de estratégias de *aprendizado ativo* no processo de agrupamento semi-supervisionado de dados são: (MALLAPRAGADA; JIN; JAIN, 2008), (VU; LABROCHE; BOUCHON-MEUNIER, 2010) e (XIONG; AZIMI; FERN, 2014). Em (MALLAPRAGADA; JIN; JAIN, 2008) foi proposta uma estratégia de *aprendizado ativo* baseada no critério *Min-Max*, onde a ideia básica é obter um conjunto de instâncias de dados rotuladas de tal forma que tais instâncias estejam distantes entre si e garantam uma boa cobertura do conjunto de dados como um todo. Contudo, somente o critério de distância não significa que as instâncias rotuladas correspondam as mais representativas do conjunto de dados. Além disso, o processo de obtenção dos rótulos pode ser caro e demorado, pois questionar os rótulos de um grande número de instâncias muitas vezes não é viável.

Em (VU; LABROCHE; BOUCHON-MEUNIER, 2010) foi proposta uma estratégia de *aprendizado ativo* para uma abordagem semi-supervisionada do algoritmo *k-means*, denominada *Seed k-means*. Tal estratégia também foi baseada no critério *Min-Max*, mas com o objetivo de selecionar os representantes iniciais de cada grupo. Os experimentos mostraram que usando o recurso de *aprendizado ativo*, cada grupo possuía pelo menos um representante após um pequeno número de instâncias de dados rotuladas e, além disso, o número de iterações até a convergência do algoritmo foi reduzido.

Na abordagem de (XIONG; AZIMI; FERN, 2014) o conjunto de dados é dividido em vizinhanças e restrições *must-link* e *cannot-link* são criadas entre as instâncias de dados consideradas mais representativas das regiões estabelecidas. Para definir as instâncias mais representativas uma estratégia baseada em densidade é considerada, na qual as instâncias de dados em regiões mais densas têm prioridade na seleção. Após essa etapa, adota-se uma abordagem incremental de expensão das vizinhanças, por meio da criação de restrições e consequentemente a formação dos grupos. Essas estratégias serviram de inspiração para a definição da abordagem empregada pelo algoritmo *SSHub Clustering* na fase de inicialização dos representantes iniciais (para maiores detalhes, veja Seção 3.1).

2.2 Tratamento de dados de alta dimensão

Existem muitos domínios nos quais a dimensionalidade dos dados é consideravelmente mais elevada. Por exemplo, em bancos de dados de reconhecimento de padrões e de imagem, nos quais os dados são constituídos por um conjunto de objetos e a alta-dimensionalidade é um resultado direto de tentar descrever os objetos por meio de um conjunto de características (também conhecido como um vetor de características). Exemplos de características incluem cor, textura, descrições de forma, e assim por diante (SAMET, 2005).

Dados de alta dimensão geralmente levam a chamada *maldição da dimensionalidade*, no qual o desempenho de muitos algoritmos de aprendizado de máquina são prejudicados. Um fator prejudicial é o fato de todas as distâncias entre as instâncias de dados serem mais difíceis de distinguir com o aumento da dimensionalidade, o que pode causar problemas com algoritmos baseados em distância (TOMASEV et al., 2011). A questão da alta dimensionalidade, tradicionalmente, tem sido tratada na literatura científica da área com a utilização de estratégias que permitem realizar a redução de dimensionalidade por meio de extração e de seleção de atributos (FACELI; LORENA; GAMA, 2011), que são detalhadas na Seção 2.2.1.

As dificuldades em lidar com dados de alta dimensão são onipresentes e abundantes, pois por mais sofisticadas que sejam as técnicas de redução de dimensionalidade, informações podem ser perdidas quando a dimensionalidade é reduzida (SILVESTRE, 2007). No entanto, nem todos os fenômenos que surgem em alta dimensão são necessariamente prejudiciais para técnicas de agrupamento, por exemplo, o aspecto *hubness* (descrito na Seção 2.2.2).

2.2.1 Redução de dimensionalidade

As estratégias propostas na literatura científica da área para a redução de dimensionalidade são provenientes de áreas de pesquisa como Reconhecimento de Padrões, Estatística e Teoria da Informação e objetivam reduzir o número de dimensões agrupando ou excluindo atributos irrelevantes ou redundantes da descrição das instâncias de dados. Essas estratégias podem ser divididas em duas categorias de métodos: extração de atributos (ou agregação) e seleção de atributos. As técnicas de extração de atributos substituem os atributos originais por novos atributos formados pela combinação de grupos de atributos. Já as técnicas de seleção mantêm parte dos atributos originais e descartam os demais atributos (FACELI; LORENA; GAMA, 2011).

As principais técnicas de extração de atributos combinam os atributos originais por meio de funções lineares ou não lineares, com o objetivo de mapear o espaço de dados original em um espaço de menor dimensão. Uma das principais técnicas de extração de atributos é a *Principal Component Analysis* - PCA (HAIR JR. et al., 1995), que

elimina redundâncias de atributos, por meio da correlação estatística dos atributos, e consequentemente, reduz a dimensionalidade do conjunto de dados (FACELI; LORENA; GAMA, 2011).

Outros exemplos interessantes de técnicas de extração de atributos são: *Multidimensional Scaling* - MDS (BORG; GROENEN, 2005), *MetricMap* (WANG et al., 2005) e *FastMap* (FALOUTSOS; LIN, 1995). Esta última permite mapear dados para um espaço de menor dimensionalidade em tempo computacional linear (BARIONI et al., 2014). Além disso, tenta preservar as distâncias entre as instâncias minimizando as distorções causadas pela redução da dimensionalidade.

As técnicas de extração de atributos podem, ao combinar os atributos, levar à perda dos valores originais. Contudo, em várias aplicações é importante preservá-los, pois é necessário interpretar os resultados produzidos. Por isso, em algumas áreas, por exemplo: biologia; finanças e medicina, é mais comum o uso de técnicas de seleção de atributos. As técnicas baseadas na abordagem de seleção de atributos tem como objetivo escolher um pequeno subconjunto de atributos de acordo com um dado critério. A seleção de atributos permite: identificar atributos importantes, eliminar atributos irrelevantes, reduzir ruídos, simplificar o modelo gerado e facilitar a visualização dos dados (FACELI; LORENA; GAMA, 2011).

Diversas técnicas automáticas têm sido propostas na literatura para a seleção de atributos. No geral, essas técnicas procuram por um subconjunto ótimo de atributos de acordo com um critério estabelecido. São exemplos de técnicas de seleção de atributos: abordagens baseadas no cálculo de dimensão fractal (JR.; TRAINA; FALOUTSOS, 2010), Ganho da Informação (KULLBACK; LEIBLER, 1951) e *Relief* (KIRA; RENDELL, 1992).

De maneira geral, as técnicas de redução de dimensionalidade fornecem apenas um subespaço do espaço de dados original no qual o processo de agrupamento de dados pode então ser realizado. Entretanto, para as situações em que atributos diferentes podem ser relevantes para grupos diferentes em um mesmo agrupamento de dados, tais métodos tendem a falhar. Nesses casos, algoritmos de agrupamento em subespaço surgiram como uma nova estratégia para analisar dados de alta dimensão (AGGARWAL; YU, 2000)(KAILING; KRIEGEL; WANKA, 2003)(KAILING; KRIEGEL, 2004)(MULLER et al., 2009).

Outro exemplo de solução especializada para analisar dados de alta dimensão, mas que não aplica técnicas de redução da dimensionalidade, são os algoritmos de agrupamento baseados em *hubness* (TOMASEV et al., 2011). Ao contrário das abordagens anteriores, ao invés de tentar evitar os efeitos associados a análise de dados em alta dimensão, essas estratégias empregam informações sobre *hubness*, que consiste na tendência de algumas instâncias de dados ocorrerem com maior frequência nas listas dos K-vizinhos mais próximos de outras instâncias, na obtenção de soluções de agrupamento de dados. Este é o foco do trabalho descrito aqui. Assim, os conceitos fundamentais a respeito do aspecto

hubness são apresentados na Seção 2.2.2.

2.2.2 Aspecto *Hubness*

Hubness é um aspecto da *maldição da dimensionalidade*. *Hubs* são instâncias de dados que aparecem na lista de vizinhos mais próximos de um grande número de outras instâncias. Tem sido demonstrado que o conceito *hubness* consiste em uma propriedade inerente de grande parte dos conjuntos de dados de alta dimensão, não sendo peculiar de conjuntos de dados específicos (TOMASEV et al., 2011).

Um aspecto da *maldição da dimensionalidade* é a concentração de distâncias, que indica a tendência das distâncias serem quase iguais entre todas as instâncias em alta dimensão. Essa característica tem um impacto negativo em qualquer algoritmo baseado em distâncias. Contudo, em alta dimensão, observa-se o conceito *hubness*, pois a medida que a dimensionalidade intrínseca dos dados aumenta, a distribuição das K-ocorrências na lista de vizinhos mais próximos de cada instância de dados torna-se distorcida e com maior variância. Assim, algumas instâncias de dados (chamadas de *hubs*) aparecem frequentemente na lista dos K-vizinhos mais próximos e, ao mesmo tempo, algumas outras instância de dados (chamadas de *anti-hubs*) tornam-se vizinhos infrequentes.

O conceito *hubness* foi tratado pela primeira vez em (AUCOUTURIER; F., 2004), no contexto de recuperação da informação de música (MARQUES, 2015), e tem sido observado em muitas áreas, tais como: processamento de imagem (HICKLIN, 2005), mineração de dados (RADOVANOVIC; NANOPOULOS; IVANOVIC, 2010) (TOMASEV et al., 2015) e biometria (YAGER; DUNSTONE, 2010). Para analisar um conjunto de dados com o conceito *hubness* é necessário calcular a pontuação *hubness* ($h_K(x)$) de cada instância. Segundo (TOMASEV et al., 2014) e (MARQUES, 2015), pontuação *hubness*, *hubs* e *anti-hubs* podem ser definidos da seguinte maneira:

- **pontuação *hubness***: seja $X = \{x_1; ...; x_n\}$ um conjunto de instâncias de dados, $h_K(x)$ representa o número de K-ocorrências de instâncias $x \in X$, isto é, o número de vezes que x ocorre na listagem dos K-vizinhos mais próximos de outras instâncias de dados pertencentes a X. Valores comumente utilizados para K estão no intervalo [5, 20] (TOMASEV et al., 2011);

- ***hubs***: correspondem à instâncias de dados $x \in X$ que aparecem, notavelmente, em muitas listagens de K-vizinhos mais próximos das demais instâncias de dados, isto é, possuem $h_K(x)$ significativamente acima da média;

- ***anti-hubs***: correspondem à instâncias de dados $x \in X$ que não aparecem em praticamente nenhuma listagem de K-vizinhos mais próximos das demais instâncias de dados, ou seja, possuem $h_K(x)$ extremamente baixo, ou até mesmo, $h_K(x) = 0$.

De acordo com (TOMASEV et al., 2011), (TOMASEV; MLADENIC, 2013) e (TOMASEV et al., 2014) o aspecto *hubness* pode desempenhar um importante papel na tarefa de detecção de agrupamentos de dados de alta dimensão, pois é uma boa referência do ponto de centralidade dos agrupamentos. Os principais *hubs* podem ser usados efetivamente como representantes para guiar o processo de atribuição de instâncias de dados a grupos, de maneira similar a como ocorre nas técnicas baseadas em centróides. Contudo, a instância com maior pontuação *hubness* ($h_k(x)$) não necessariamente corresponde ao medóide de um agrupamento. A Figura 3 ilustra essa questão, onde o losango é o centróide do grupo representado em preto, o círculo vermelho corresponde ao medóide e os *hubs* são destacados em verde (considerando $K = 3$).

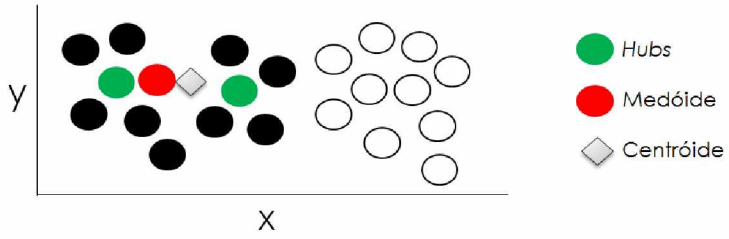

Figura 3 – Comparação *Hub vs.* Medóide *vs.* Centróide. Adaptado de (TOMASEV et al., 2011).

Dentre as propostas de utilização do aspecto *hubness* na tarefa de agrupamento de dados está o algoritmo *k-hubs* (Algoritmo 2), proposto em (TOMASEV et al., 2011), que consiste em uma variação do algoritmo *k-means*. Na versão tradicional do algoritmo *k-means* os agrupamentos são representados pelo centróide, já na abordagem *k-hubs* o centróide é substituído pela instância do agrupamento com maior valor $h_K(x)$ (linha 7 do Algoritmo 2). Conforme os resultados de (TOMASEV et al., 2011), no geral, a acurácia dos resultados do algoritmo *k-hubs* foi superior a da abordagem original, em conjuntos de dados reais e sintéticos, com diferentes dimensões, mostrando o potencial da proposta.

Algoritmo 2: *k-hubs*. Adaptado de (TOMASEV et al., 2011).

Entrada: Conjunto de dados X, Número de grupos k
Saída: $\pi = \{C_1, ..., C_k\}$

```
1  início
2      Selecione aleatoriamente μ_1, ..., μ_k como centróides iniciais;
3      repita
4          para cada instância x_i ∈ X faça
5              Atribua x_i ao grupo C_j com μ_j mais próximo;
6          para cada grupo C_j ∈ π faça
7              Procure a instância x_i com o maior valor de h_K e defina μ_j = x_i;
8      até convergir ;
9      retorna π
10 fim
```

As instâncias com os maiores valores de pontuações *hubness* são de fato as melhores candidatas para serem utilizadas como representantes dos grupos. Contudo, o algoritmo *k-hubs* ignora as informações do aspecto *hubness*, isto é, os valores $h_K(x)$, das demais instâncias de dados. Então, foi definido o algoritmo *HPC* (*Hubness-Proportional Clustering*) (TOMASEV et al., 2011) que considera a pontuação de outras instâncias de dados para permitir que os representantes dos grupos possam ser alterados por outros *hubs* durante as iterações do algoritmo. O *HPC* está definido no Algoritmo 3.

Algoritmo 3: *HPC*. Adaptado de (TOMASEV et al., 2011).

Entrada: Conjunto de dados X, Número de grupos k, Número de iterações aleatórias iA
Saída: $\pi = \{C_1, ..., C_k\}$

```
início
    Selecione aleatoriamente μ_1, ..., μ_k como centróides iniciais;
    para cada instância x_i ∈ X faça
        Atribua x_i ao grupo C_j com μ_j mais próximo;
        hA_K(x_i) = h_K(x_i);
    it = 1;
    repita
        θ = it/iA;
        para cada grupo C_j ∈ π faça
            se numeroAleatorio(0,1) < θ então
                Procure a instância x_i ∈ C_j com o maior valor de h_K e defina
                μ_j = x_i;
            senão
                para cada instância x_i ∈ C_j faça
                    se x_i não mudou de grupo então
                        hA_K(x_i) = hA_K(x_i)^2;
                    senão
                        hA_K(x_i) = h_K(x_i);
                Procure a instância x_i ∈ C_j com o maior valor de hA_K e defina
                μ_j = x_i;
        para cada instância x_i ∈ X faça
            Atribua x_i ao grupo C_j com μ_j mais próximo;
        it++;
    até convergir ;
    retorna π
fim
```

O algoritmo *HPC* consiste em um modelo estocástico, isto é, depende de eventos aleatórios para definir os *hubs* que serão considerados como representantes dos grupos. Para tanto, deve-se informar um parâmetro para definir um número máximo de iterações aleatórias (parâmetro iA do Algoritmo 3). Então, considera-se um número aleatório de

ponto flutuante θ no intervalo [0,1]. Se o valor θ for inferior ao número da iteração atual dividida por iA (linha 10 do Algoritmo 3) define-se as instâncias com as maiores pontuações *hubness* (h_K) como representantes dos respectivos grupos, da mesma forma que o algoritmo *k-hubs*. Senão, deve-se considerar as pontuações *hubness* acumuladas das instâncias de dados.

O algoritmo *HPC* considera uma abordagem de pontuações *hubness* acumuladas, que consiste em elevar o valor da pontuação *hubness* ao quadrado (linha 15 do Algoritmo 3) no caso da instância de dados não mudar de grupo de uma iteração para outra. Caso a instância de dados mude de grupo, sua pontuação *hubness* retorna ao valor original (linha 17 do Algoritmo 3). Com essa estratégia são privilegiadas as instâncias de dados mais estáveis, isto é, que não alteram de grupos. É importante ressaltar que quando a iteração corrente atingir o valor iA definido a abordagem de pontuações acumuladas não é considerada, pois a condição definida (linha 10 do Algoritmo 3) sempre é satisfeita.

Contudo, a abordagem do algoritmo *HPC* considera somente os *hubs* e não utiliza os centróides durante o processo de agrupamento. Então, em (TOMASEV et al., 2011) o algoritmo *HPKM (Hubness-proportional k-means)* foi definido. O *HPKM* também é uma variação do tradicional algoritmo *k-means*, mas emprega uma abordagem estocástica de agrupamento híbrida, isto é, utiliza ambos (*hubs* e centróides) como representantes de grupos. O algoritmo *HPKM* está definido no Algoritmo 4.

A ideia geral do algoritmo *HPKM* consiste em usar as informações sobre a pontuação *hubness* das instâncias de dados e utilizar os *hubs* para guiar os particionamentos nas primeiras iterações e no final escolher uma configuração de agrupamento baseada em centróides, como o tradicional algoritmo *k-means*. Observando o Algoritmo 4, nota-se que o *HPKM* é bastante semelhante ao algoritmo *HPC*, pois também considera as iterações aleatórias e o recurso de pontuações *hubness* acumuladas. Contudo, observa-se que somente os centróides são considerados quando a iteração atual atinge o valor iA definido (linha 10 do Algoritmo 4).

Conforme os resultados dos experimentos realizados em (TOMASEV et al., 2011), com conjuntos de dados reais e sintéticos, com diferentes dimensões, a acurácia dos resultados do algoritmo *HPC* foi superior ao algoritmo *k-hubs*, o que mostra que é interessante considerar as pontuações *hubness* de mais instâncias do conjunto de dados, além de somente os *hubs* definidos como representantes iniciais. Entretanto, no mesmo cenário de experimentos, no geral, o algoritmo *HPKM* foi o que apresentou as melhores acurácias, mostrando o potencial da proposta híbrida, que considera *hubs* e centróides como representantes dos agrupamentos. Além dos algoritmos *k-hubs* e *HPC*, o algoritmo *HPKM* foi comparado aos algoritmos descritos na Seção 2.3 e também apresentou resultados, no geral, superiores.

Apesar da boa eficácia demonstrada pelos algoritmos *k-hubs*, *HPC* e *HPKM* para lidar com dados em alta dimensão, a eficiência computacional dos algoritmos de agrupamento

baseados em *hubs* pode ser um motivo de preocupação quando é desejado analisar grandes conjuntos de dados. Considerando que a complexidade computacional desses algoritmos está diretamente relacionada ao cálculo das pontuações *hubness* das instâncias de dados é importante considerar estratégias que permitam otimizar esses cálculos. Dentre essas estratégias é possível mencionar a utilização de um grafo K-NN aproximado para a obtenção das pontuações *hubness*, que apresenta complexidade de tempo de $\Theta(ndt)$, sendo n o número de instâncias, d a quantidade de dimensões e t representa a construção do grafo (CHEN; SAAD; DASGUPTA, 2009). Outra alternativa apontada em (TOMASEV et al., 2014) consiste em utilizar abordagens baseadas em *hash* sensíveis a localidade, que são métodos de redução de espaços d-dimensionais sem perder a localidade, isto é, mantêm instâncias de dados próximas no espaço original e próximas no espaço de menor dimensão (SATULURI; PARTHASARATHY, 2012).

Algoritmo 4: *HPKM*. Adaptado de (TOMASEV et al., 2011).

Entrada: Conjunto de dados X, Número de grupos k, Número de iterações aleatórias iA
Saída: $\pi = \{C_1, ..., C_k\}$

```
1  início
2      Selecione aleatoriamente μ_1, ..., μ_k como centróides iniciais;
3      para cada instância x_i ∈ X faça
4          Atribua x_i ao grupo C_j com μ_j mais próximo;
5          hA_K(x_i) = h_K(x_i);
6      it = 1;
7      repita
8          θ = it/iA;
9          para cada grupo C_j ∈ π faça
10             se numeroAleatorio(0,1) < θ então
11                 Calcule o centróide do grupo C_j e defina em μ_j;
12             senão
13                 para cada instância x_i ∈ C_j faça
14                     se x_i não mudou de grupo então
15                         hA_K(x_i) = hA_K(x_i)^2;
16                     senão
17                         hA_K(x_i) = h_K(x_i);
18                 Procure a instância x_i ∈ C_j com o maior valor de hA_K e defina
                   μ_j = x_i;
19         para cada instância x_i ∈ X faça
20             Atribua x_i ao grupo C_j com μ_j mais próximo;
21         it + +;
22     até convergir ;
23     retorna π
24 fim
```

Também é importante destacar que, como os *hubs* são centros de influência, possíveis imprecisões associadas a eles podem ser facilmente propagadas podendo impactar negativamente na formação dos agrupamentos. Uma forma de lidar com essa questão, minimizando o risco de induzir um particionamento não adequado nos dados, uma vez que *hubs* podem não refletir bem a semântica implícita dos dados, é permitir a incorporação de técnicas de semi-supervisão nos processos de agrupamento de dados (veja Seção 2.1.2).

2.3 Algoritmos selecionados para comparação

Além do algoritmo *HPKM*, descrito na Seção 2.2.2, os algoritmos *Kernel k-means*, *DBSCAN* e *SSDBSCAN* também foram selecionados como linha de base para comparação e estão detalhados a seguir.

2.3.1 *Kernel k-means*

Uma deficiência da abordagem tradicional do algoritmo *k-means* é não ter a capacidade de lidar com problemas não-linearmente separáveis. Sendo que esse problema é agravado principalmente na manipulação de dados de alta dimensão. Com o objetivo de contribuir para sanar essa deficiência, foi proposto o algoritmo *Kernel k-means* (DHILLON; GUAN; KULIS, 2004). Esse algoritmo é uma extensão do método de agrupamento *k-means* tradicional, que realiza transformações no espaço de características do conjunto de dados por meio de funções *kernel*.

As funções *kernel* mapeiam o espaço de características do conjunto de dados em um espaço com um número diferente de dimensões utilizando, por exemplo, uma função que realiza o produto interno entre as dimensões de duas instâncias de dados x_1 e x_2. Geralmente, esse procedimento é chamado de "truque do *Kernel*". Uma função *kernel* (Ker) que realiza o produto interno de x_1 e x_2 pode ser representada conforme demonstrado na Equação 6. O algoritmo *Kernel k-means* é apresentado no Algoritmo 5.

$$Ker = (\phi(x_1), \phi(x_2)) \tag{6}$$

Observando o Algoritmo 5, nota-se que o algoritmo *Kernel k-means* é semelhante a abordagem tradicional do algoritmo *k-means*, pois considera centróides para representar os grupos (linha 7 do Algoritmo 5) e utiliza funções de distância para verificar a similaridade entre duas instâncias de dados (linha 5 do Algoritmo 5). Contudo, percebe-se que para representar os centróides também deve-se considerar a função *kernel* ϕ utilizada, conforme demonstrado na Equação 7.

$$\mu_j^{\phi} = \frac{1}{n_j} \sum_{x_i \in C_j} \phi(x_i) \tag{7}$$

Métodos de agrupamento que consideram funções *kernel* além de trabalhar com problemas não-linearmente separáveis também são conhecidos por lidarem bem com grupos de formatos arbitrários, isto é, formatos que não necessariamente correspondem a formas esféricas. Essas características motivaram a escolha do algoritmo *Kernel k-means* para a comparação com os demais algoritmos selecionados para o tratamento de dados de alta dimensão. Mais detalhes a respeito do algoritmo *Kernel k-means* podem ser encontrados em (DHILLON; GUAN; KULIS, 2004) e (DHILLON; GUAN; KULIS, 2007).

Algoritmo 5: *Kernel k-means*. Adaptado de (DHILLON; GUAN; KULIS, 2004).

Entrada: Conjunto de dados X, Número de grupos k
Saída: $\pi = \{C_1, ..., C_k\}$

```
início
    Selecione aleatoriamente μ_1^φ, ..., μ_k^φ como centróides iniciais;
    repita
        para cada instância x_i ∈ X faça
            Atribua φ(x_i) ao grupo C_j com μ_j^φ mais próximo;
        para cada grupo C_i faça
            Atualize seus centróides μ_i^φ considerando a Equação 7;
    até convergir ;
    retorna π
fim
```

2.3.2 *DBSCAN*

O foco do trabalho descrito nesta dissertação consiste em algoritmos de agrupamento que empregam a abordagem particional. Contudo, o aspecto *hubness* é uma forma de estimativa de densidade, pois é capaz de detectar instâncias de dados que estão em regiões mais densas (*hubs*), isto é, com um número significativo de instâncias, e também pode detectar possíveis *outliers* (*anti-hubs*). Com isso, considerou-se importante avaliar o desempenho do algoritmo proposto frente a um método de agrupamento baseado em densidade, como o *DBSCAN*.

O algoritmo *DBSCAN* (*Density-Based Spatial Clustering of Applications with Noise*) (SANDER et al., 1998) é um dos algoritmos de agrupamento mais conhecidos baseados em densidade. Para os algoritmos baseados na abordagem particional, geralmente, é necessário informar previamente a quantidade de grupos a serem formados. Já para os algoritmos da abordagem baseada em densidade o número de grupos é definido automaticamente por meio da separação entre regiões de alta e baixa densidade.

No algoritmo *DBSCAN* é necessário informar apenas dois parâmetros: *eps* (ϵ) e *minPoints*. O *eps* é uma espécie de raio que define a distância máxima entre duas instâncias de dados para que possam pertencer ao mesmo grupo. Já o parâmetro *minPoints* define

o número mínimo de instâncias de dados necessário para se criar um grupo. Isso permite remover dos grupos instâncias de dados não desejadas ou agrupar todas as instâncias de dados, consideradas *outliers*, em um grupo específico.

O Algoritmo *DBSCAN* é composto de uma rotina principal, apresentada no Algoritmo 6, e por uma subrotina responsável por expandir os grupos, apresentada no Algoritmo 7. Um dos primeiros passos do algoritmo *DBSCAN* é obter as instâncias de dados que estão na vizinhança v de uma instância de dados x_i (linha 4 do Algoritmo 6). Então, é necessário verificar se o tamanho dessa vizinhança v atende o parâmetro *minPoints* (linha 5 do Algoritmo 6). Caso atenda, é solicitada a subrotina para expandir o grupo (linha 9 do Algoritmo 6). Caso contrário, x_i é considerada como *outlier* (linha 6 do Algoritmo 6).

Algoritmo 6: *DBSCAN*. Adaptado de (SANDER et al., 1998).

```
   Entrada: Conjunto de dados X, Eps ε, MinPoints m
   Saída: π = {C_1, ..., C_k}
 1 início
 2     para cada instância x_i não verificada ∈ X faça
 3         Marque x_i como verificada;
 4         v = instâncias vizinhas de x_i considerando ε;
 5         se |v| < m então
 6             Marque x_i como outlier;
 7         senão
 8             Cria novo grupo C;
 9             ExpandeGrupo(C, x_i, v, ε, m);
10     retorna π
11 fim
```

Algoritmo 7: ExpandeGrupo

```
   Entrada: Grupo C, Instância de dados x, Vizinhanca v, Eps ε, MinPoints m
 1 início
 2     Adicione x ao grupo C;
 3     para cada instância x_i' ∈ v faça
 4         se x_i' não foi verificada então
 5             Marque x_i' como verificada;
 6             v' = instâncias vizinhas de x_i' considerando ε;
 7             se |v'| ≥ m então
 8                 v = v ∪ v';
 9         se x_i' não pertence a nenhum grupo então
10             Atribua x_i' a C;
11 fim
```

A subrotina detalhada no Algoritmo 7 tem o objetivo de agrupar as vizinhanças de diferentes instâncias de dados. Primeiramente, considera-se a vizinhança v da instância de dados x_i que estava sendo tratada no Algoritmo 6. Então, as vizinhanças $v^{'}$ de cada

instância de dados pertencente a v são verificadas (linha 6 do Algoritmo 7). Logo após, é certificado se $v^{'}$ atende o critério do parâmetro *minPoints*. Caso atenda, realiza-se a junção das vizinhanças v e $v^{'}$ (linha 8 do Algoritmo 7). Por fim, a instância de dados $x_i^{'}$ é atribuída a C caso não pertença a nenhum grupo (linha 10 do Algoritmo 7).

Considerando que o algoritmo proposto no trabalho descrito nessa dissertação é um algoritmo de agrupamento semi-supervisionado, também é importante comparar seu desempenho frente a algoritmos da literatura que também empreguem essa abordagem. Para tanto, foi selecionado um algoritmo que reúne as duas características exploradas na definição no *SSHUB CLustering*, estimativa de densidade e semi-supervisão: o algoritmo *SSDBSCAN* descrito na Seção 2.3.3.

2.3.3 *SSDBSCAN*

O *SSDBSCAN* (*Semi-Supervised DBSCAN*) (LELIS; SANDER, 2009)(LI et al., 2014) é um algoritmo de agrupamento semi-supervisionado baseado no algoritmo *DBSCAN*. O algoritmo *SSDBSCAN* usa uma pequena quantidade de instâncias de dados rotuladas para encontrar parâmetros de densidade e definir os grupos.

O algoritmo *SSDBSCAN* não considera a definição de restrições *must-link* e *cannot-link* entre pares de instâncias de dados. Ele necessita de algumas instâncias de dados rotuladas, que são utilizadas da seguinte maneira: a partir de cada instância de dados rotulada é construída uma árvore de extensão mínima (PETTIE; RAMACHANDRAN, 2002), cuja raiz é a instância de dados rotulada. Em seguida, os grupos são formados por meio das árvores criadas.

Contudo, para construir uma árvore de extensão mínima para cada grupo é necessário que as instâncias de dados rotuladas atinjam uma boa cobertura da estrutura do conjunto de dados, isto é, pelo menos uma instância de dados de cada grupo deve ser incluída no subconjunto de instâncias de dados rotuladas. Por outro lado, a utilização dessa estrutura de dados elimina a necessidade de definir os parâmetros *eps* (ϵ) e *minPoints*, que são tipicamente de difícil parametrização.

As árvores de extensão mínima são espécies de subgrafos que conectam os vértices de um grafo principal, isto é, conectam instâncias de dados no contexto empregado. Para tanto, determina-se um peso para cada um dos vértices, com o objetivo de especificar o grau de conectividade entre duas instâncias de dados. Então, considerando que trata-se de um algoritmo que trabalha com a abordagem em densidade, é necessário informar um parâmetro γ que define o menor peso para que duas instâncias de dados estejam conectadas. Uma visão geral do algoritmo *SSDBSCAN* é apresentada no Algoritmo 8.

Observando o Algoritmo 8 nota-se que além do *threshold* γ utilizado para os pesos, é necessário informar dois parâmetros: o conjunto de dados X e outro conjunto de dados $X^{'}$ com instâncias de dados rotuladas. Então, realiza-se a construção das árvores de extensão mínima (linha 2 do Algoritmo 8), considerando as instâncias de dados rotuladas de $X^{'}$.

Em seguida, seleciona-se uma instância x_i das árvores criadas e atribui-se a um grupo C_j (linha 5 do Algoritmo 8). Então, o algoritmo analisa todas as instâncias de dados que estão na mesma árvore de x_i. Essas instâncias também são inseridas em C_j até que se encontre uma instância de dados $x_j^{'}$ que possua o rótulo diferente x_i ou que possua um peso inferior ao *threshold* γ definido (linha 10 do Algoritmo 8). Mais detalhes a respeito do algoritmo *SSDBSCAN* podem ser encontrados em (LELIS; SANDER, 2009) e (LI et al., 2014).

Algoritmo 8: *SSDBSCAN*. Adaptado de (LI et al., 2014).

Entrada: Conjunto de dados X, Subconjunto com instâncias de dados rotuladas $X^{'}$, *Threshold* γ
Saída: $\pi = \{C_1, ..., C_k\}$

```
1  início
2      A = Construa as árvores de extensão mínima a partir de X';
3      D = X;
4      repita
5          x_i = Selecione uma instância de dados de A;
6          Crie um grupo C_j e atribua x_i a C_j;
7          Remova x_i de D;
8          I = instâncias de dados da árvore A de x_i;
9          para cada instância x'_j ∈ I faça
10             se x'_j ∈ X' e rótulo de x'_j ≠ rótulo de x_i
11             ou peso de x'_j < γ então
12                 break;
13             senão
14                 Atribua x'_j a C_j;
15                 Remova x'_j de D;
16     até D ≠ ∅ ;
17     retorna π
18 fim
```

2.4 Métodos de avaliação

Os métodos de avaliação são empregados com objetivo de comparar os particionamentos gerados por diferentes algoritmos de agrupamento. Geralmente, para se realizar tal avaliação considera-se índices estatísticos, que são selecionados com base em algum critério relacionado ao conjunto de dados. Os principais critérios utilizados são detalhados a seguir:

- **Critério relativo**: compara agrupamentos gerados por um mesmo algoritmo, mas obtidos por diferentes configurações de parâmetros. Esse critério pode auxiliar na decisão da melhor configuração de parâmetros a ser empregada. Os índices de

Conectividade e Silhueta consideram esse critério (VENDRAMIN; CAMPELLO; HRUSCHKA, 2010);

- **Critério interno**: considera informações do próprio conjunto de dados, por exemplo, uma matriz de similaridade. Índices que consideram esse critério podem inferir o quanto os grupos gerados por um processo de agrupamento representam a real estrutura do conjunto de dados. Um exemplo de índice para esse critério é o índice *Gap* (TIBSHIRANI; GUENTHER; HASTIE., 2001);

- **Critério externo**: considera informações prévias do conjunto de dados, por exemplo, o conhecimento prévio de um especialista sobre os rótulos das instâncias de dados, e consequentemente, a quantidade de grupos. O critério externo é capaz de medir a correspondência da estrutura original do conjunto de dados com os grupos obtidos por um processo de agrupamento. Os índices *Jaccard*, *Rand* e *Rand* Corrigido são exemplos de índices que consideram esse critério (FACELI; LORENA; GAMA, 2011).

Considerando que o objetivo do trabalho descrito aqui (veja Seção 1.1) é contribuir para a obtenção de resultados mais acurados em métodos de agrupamento semi-supervisionado de dados em alta dimensão o critério de avaliação mais adequado é o externo. Com isso, para a avaliação dos experimentos considerou-se o índice *Rand* Corrigido (HUBERT; ARABIE, 1985), que é bastante utilizado na literatura de agrupamento semi-supervisionado de dados.

2.4.1 Índice *Rand* Corrigido

O índice *Rand*, em sua abordagem original (RAND, 1971), tem o objetivo de comparar dois particionamentos, por exemplo, o particionamento gerado por um algoritmo de agrupamento e outro particionamento com as informações originais de classe das instâncias de dados. Para calcular o índice *Rand* (Equação 8), deve-se analisar o número de pares de instâncias de dados nas seguintes situações:

- a: pertencem a mesma classe e foram atribuídas ao mesmo grupo;

- b: pertencem a mesma classe e foram atribuídas a grupos distintos;

- c: pertencem a classes distintas e foram atribuídas ao mesmo grupo;

- d: pertencem a classes distintas e foram atribuídas a grupos distintos.

$$R = \frac{a + d}{a + b + c + d} \tag{8}$$

O valor do índice *Rand* é apresentado no intervalo [0,1], sendo que quando o valor retornado for igual a zero (0), significa que os dois particionamentos considerados não possuem nenhuma similaridade. Já o valor um (1) significa que os dois particionamentos são idênticos. Contudo, por meio da Equação 8 percebe-se que o índice *Rand*, em sua abordagem original, possui limitações. Um dos principais problemas é atribuir o mesmo peso para as instâncias de dados agregadas (termo a) ou separadas (termo d). Em conjuntos de dados com um número elevado de grupos, o termo d tende a dominar o cálculo, já que quanto maior o número de grupos, mais pares pertencem a grupos distintos.

Outra limitação é o índice *Rand* não considerar a possibilidade de acertos aleatórios. De forma a corrigir essas limitações em (HUBERT; ARABIE, 1985) definiu-se o índice *Rand* Corrigido (ou Ajustado), apresentado na Equação 9:

$$RC = \frac{a - \frac{(a+c)\cdot(a+b)}{a+b+c+d}}{\frac{(a+c)+(a+b)}{2} - \frac{(a+c)\cdot(a+b)}{a+b+c+d}} \tag{9}$$

onde os valores de a, b, c e d são os mesmos utilizados para o cálculo do índice *Rand* original (Equação 8).

O valor do índice *Rand* Corrigido é apresentado no intervalo [-1,1]. Nessa abordagem, valores próximos a zero (0) ou negativos indicam que a semelhança entre as partições se deve ao acaso. Da mesma forma que o índice *Rand* original, o valor um (1) significa que os dois particionamentos comparados são idênticos. Após calcular o índice *Rand* Corrigido para cada algoritmo em cada conjunto de dados, pode-se considerar outros índices com o objetivo de realizar a validação estatística.

2.4.2 Validação estatística

Em cenários de experimentos com diversos conjuntos de dados, o resultado apresentado por um índice de validação, por exemplo, o índice *Rand* Corrigido, pode não ser suficiente para concluir se existe diferença de desempenho entre os algoritmos. Com esse objetivo, pode-se aplicar um teste de hipóteses para avaliar os resultados. Primeiramente, formula-se uma hipótese nula (H_0) e outra hipótese alternativa (H_A). Geralmente, a hipótese nula (H_0) afirma que os desempenhos dos algoritmos comparados são equivalentes. Caso a hipótese nula seja rejeitada, H_A é considerada. A hipótese alternativa afirma que um dos algoritmos apresenta desempenho superior aos demais (DEMSAR, 2006).

Após elaborar H_0 e H_A, deve-se determinar o nível de confiança para o teste, ou seja, definir a probabilidade de se rejeitar a hipótese nula mesmo quando é verdadeira. Para tanto, é necessário determinar os valores críticos considerados, pois eles irão separar as regiões de aceitação e rejeição. Então, é possível rejeitar H_0 caso o resultado esteja na região crítica ou aceitar H_0 na situação contrária.

Tradicionalmente, para realizar essa análise são selecionados testes paramétricos ou não-paramétricos. Um exemplo de teste paramétrico é o ANOVA (VIEIRA, 2006). Con-

tudo, testes paramétricos exigem algumas características aos conjuntos de dados, tais como: obedecer uma distribuição normal, possuir variância homogênea e estar em um intervalo contínuo. Então, optou-se pela abordagem não paramétrica, que requer menos exigências sobre o conjunto de dados. O teste de *Friedman* é um exemplo de teste paramétrico e está detalhado na Seção 2.4.2.1.

2.4.2.1 Teste de *Friedman*

O teste de *Friedman* (FRIEDMAN, 1937) ranqueia os algoritmos para cada conjunto de dados. Considerando que um algoritmo A apresentou o melhor desempenho para um conjunto de dados C em relação a um algoritmo B. Então, o algoritmo A é ranqueado em primeiro lugar e o algoritmo B em segundo. No caso de empates, isto é, os dois algoritmos apresentarem o mesmo desempenho, realiza-se a *soma* dos *ranks* dos algoritmos dividido pelo total de algoritmos empatados. Considerando um empate entre os algoritmo A e B o resultado para ambos os algoritmos é $1,5$ $((1+2)/2)$.

Após a definição dos *ranks* (R) de cada algoritmo, realiza-se o cálculo do teste de *Friedman* χ_F^2 (DEMSAR, 2006), apresentado na Equação 10:

$$\chi_F^2 = \left[\frac{12}{NA(A+1)} \cdot \sum_{i=1}^{A} R_i^2\right] - (3N(A+1)) \tag{10}$$

onde N corresponde ao número de conjuntos de dados, A ao número de algoritmos testados e R aos valores da soma dos *ranks* de cada algoritmo (veja Tabela 2).

Contudo, o trabalho descrito (IMAN; DAVENPORT, 1980) mostrou que o teste de *Friedman* é indesejavelmente conservador e é necessário realizar outros cálculos para complementar o teste. Com isso, o valor de χ_F^2 é usado para compor o cálculo F_F, apresentado na Equação 11.

$$F_F = \frac{(N-1)\chi_F^2}{N(A-1) - \chi_F^2} \tag{11}$$

Para a aplicação do teste são considerados $(A-1)$ e $(A-1)(N-1)$ graus de liberdade. Pode-se utilizar a tabela apresentada em (ZAR, 2007) e disponível no Apêndice A.1 para a consulta e assim obter um valor w para a análise. Caso o valor de F_F seja menor que w, é possível afirmar que existe diferença significativa entre o desempenho dos algoritmos e então rejeitar a hipótese nula. Contudo, o teste de *Friedman* verifica apenas se os resultados possuem diferença estatística, mas não aponta qual dos algoritmos teve o desempenho superior. Então, pode-se aplicar o teste *post-hoc Bonferroni-Dunn*, apresentado na Seção 2.4.2.2, para complementar o resultado da avaliação com o teste de *Friedman*.

2.4.2.2 Teste de *post-hoc Bonferroni-Dunn*

No teste *post-hoc Bonferroni-Dunn* (ZAR, 2007), primeiramente, define-se a diferença crítica (CD) de desempenho dos algoritmos para delimitar um valor que possibilite comparar se um algoritmo é superior a outro ou não. O cálculo de CD é apresentado na

Equação 12:

$$CD = q_\alpha \sqrt{\frac{A(A+1)}{6N}} \tag{12}$$

onde o valor crítico q_α é baseado na *Studentized Range Statistic* dividida por $\sqrt{2}$, apresentada em (DEMSAR, 2006) e no Apêndice A.2.

Após o cálculo de CD, a diferença (z) do *rank* médio ($\bar{R}$) entre cada par de algoritmos é calculada. Para tanto, deve-se fixar um dos algoritmos para ser considerado o de controle e o comparar aos demais. O cálculo de z é apresentado na Equação 13:

$$z = \frac{(\bar{R}_{alg_j} - \bar{R}_{alg_c})}{\sqrt{\frac{A(A+1)}{6N}}} \tag{13}$$

onde $\bar{R}_{alg_c}$ corresponde ao *rank* médio do algoritmo de controle e $\bar{R}_{alg_j}$ se refere ao *rank* médio de outro algoritmo comparado.

Então, é preciso verificar se o valor calculado em z é superior a diferença crítica (CD). Caso seja superior, conclui-se que o algoritmo de controle é superior ao outro algoritmo comparado, com o nível de confiança utilizado para o teste. Essa análise é realizada, em relação ao algoritmo de controle, para todos os demais algoritmos selecionados.

2.5 Considerações finais

Este capítulo apresentou os principais conceitos relacionados com a tarefa de detecção de agrupamentos, tendo como foco as abordagens de agrupamento particional e semi-supervisionada. Além disso, também foram discutidos trabalhos relacionados e foram detalhados os algoritmos selecionados com linha de base para comparação.

Também, foram apresentados os principais conceitos relativos ao tratamento de dados de alta dimensão, incluindo: seleção de atributos; extração de atributos e o aspecto *hubness*, que consiste na motivação para a proposta do algoritmo descrito no próximo capítulo.

Capítulo 3

SSHub Clustering

Este capítulo descreve o algoritmo *SSHub Clustering* (*Semi-supervised hubness-based clustering*) desenvolvido no trabalho descrito aqui. O *SSHub Clustering* é um algoritmo de agrupamento semi-supervisionado de dados desenvolvido, com foco em conjuntos de dados de alta dimensão. O objetivo deste algoritmo é incorporar semi-supervisão, por meio da definição de restrições de nível instância e de outros recursos de *aprendizado ativo*, juntamente com uma solução especializada para o tratamento de dados de alta dimensão.

O capítulo está organizado da seguinte maneira: a Seção 3.1 descreve as principais etapas do algoritmo proposto, contemplando a definição dos representantes principais, a fase de atribuição de instâncias aos grupos e atualização dos representantes; a Seção 3.1.1 descreve a subrotina responsável pela análise das instâncias de fronteira e por definir as restrições *must-link* e *cannot-link*. Por fim, a Seção 3.2 apresenta as considerações finais do capítulo.

3.1 Algoritmo proposto

A abordagem de agrupamento empregada pelo algoritmo descrito aqui combina estratégias de semi-supervisão e de estimativa de densidade baseada em pontuações *hubness*, com o objetivo de contribuir para a análise de dados de alta dimensão. Tal algoritmo é denominado *SSHub Clustering* e suas principais etapas estão ilustradas no fluxograma apresentado na Figura 4.

Como pode ser observado na Figura 4, a primeira etapa consiste no pré-processamento dos conjuntos de dados, na qual realiza-se o cálculo da pontuação *hubness* de cada instância de dados. Esta etapa é considerada pré-processamento, pois pode ser realizada por um algoritmo externo ao algoritmo de agrupamento. Além disso, caso os resultados sejam armazenados, podem ser utilizados como parâmetro de entrada, em várias execuções dos algoritmos de agrupamento. As etapas 2, 3 e 4, apresentadas na Figura 4, estão

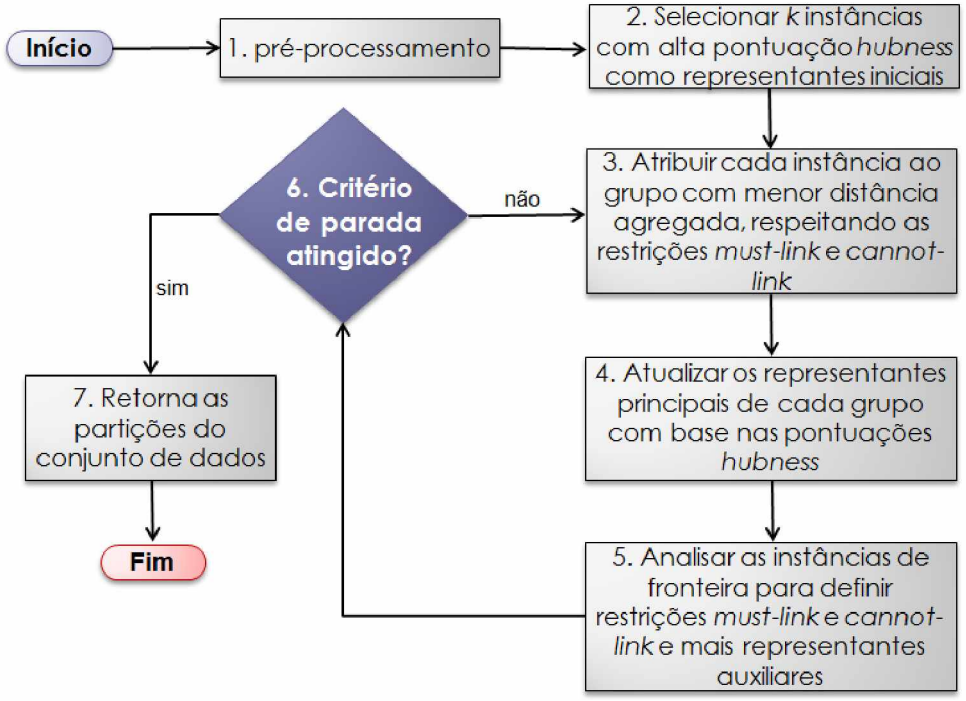

Figura 4 – Fluxograma do processo de agrupamento realizado pelo *SSHub Clustering*.

detalhadas no Algoritmo 9. Já a etapa 5 é realizada por uma subrotina, apresentada no Algoritmo 10 e detalhada na Seção 3.1.1.

Por meio do Algoritmo 9, nota-se que a partir do conjunto de dados $X = \{x_1; ...; x_n\}$, do conjunto H com a pontuação *hubness* $h_K(x)$ de cada instância x_i, e do percentual f de elementos de fronteira a serem analisados para a definição das restrições *must-link* e *cannot-link*, o *SSHub Clustering* emprega uma abordagem de agrupamento semi-supervisionada baseada no particionamento de X em uma quantidade k de grupos. Cada grupo é representado por múltiplos protótipos, um principal $p_i \in P$ e vários auxiliares $a_i \in A$. Essa estratégia de representação foi adotada com o intuito de permitir que o algoritmo possa lidar bem tanto com grupos com formatos hiperesféricos, como com grupos que apresentem formas mais complexas.

Para criar a partição inicial dos dados o algoritmo *SSHub Clustering* inicia selecionando, de modo semi-supervisionado, um representante principal para cada grupo (linha 2 do Algoritmo 9). A estratégia proposta utiliza o recurso de *aprendizado ativo*, com o objetivo de selecionar instâncias de dados mais representativas. Para tanto, essa seleção permite guiar o usuário, a partir de um *ranking* das maiores pontuações *hubness* de cada instância de dados, na indicação de k instâncias que devam ser usadas como representantes. Os demais representantes auxiliares são derivados de restrições *must-link* $r_m(x_i, x_j) \in R_m$ a partir da segunda iteração do algoritmo. É importante destacar que os representantes iniciais selecionados são considerados representantes permanentes dos grupos, isto é, mesmo que os representantes principais dos grupos sejam atualizados eles permanecem como representantes auxiliares (linha 8 do Algoritmo 9). A Figura 5 ilustra a

Algoritmo 9: SSHub Clustering

Entrada: Conjunto de dados X, Número de grupos k, Hubness H, Elementos de fronteira f, Máximo iterações $maxIt$
Saída: $\pi = \{C_1, ..., C_k\}$

início
  $P \leftarrow$ Seleção dos k representantes iniciais;
  $A \leftarrow \{\}$;
  $R \leftarrow \{\}$;
  $hA_K \leftarrow H$;
  $it = 0$;
  para *cada representante* $p_i \in P$ **faça**
    $A_i = A_i \cup p_i$;
  repita
    it++;
    para *cada instância* $x_i \in X$ **faça**
      Atribua x_i ao grupo com menor distância agregada;
      se x_i *não mudou de grupo* **então**
        $hA_K(x_i) = hA_K(x_i)^2$;
      senão
        $hA_K(x_i) = h_K(x_i)$;
    para *cada grupo* C_i **faça**
      Atualiza o representante p_i com base em hA_K;
    CriaRestricoes(π, H, f, A, R);
  até *convergir ou* $it = maxIt$;
  retorna π
fim

ideia do processo de seleção de k representantes iniciais ($k = 4$) em um conjunto de dados bidimensional, considerando $K = 5$ vizinhos mais próximos para o cálculo da pontuação *hubness* ($h_K(x)$) de cada instância de dados.

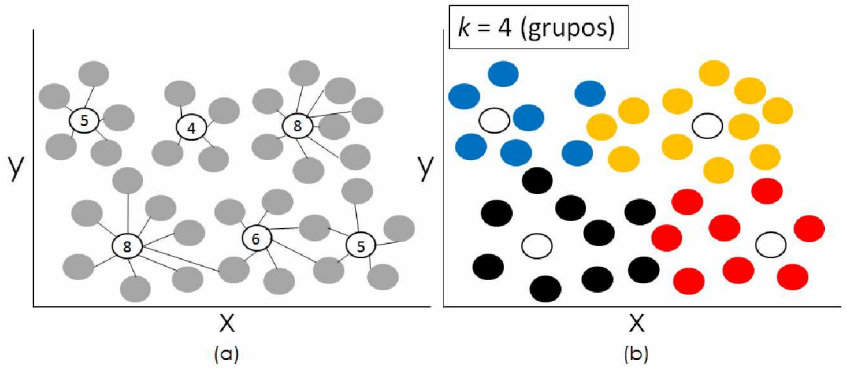

Figura 5 – Seleção de $k = 4$ representantes iniciais. (a) *Hubs* (b) k *hubs* selecionados e o particionamento inicial gerado.

Considerando a possibilidade da existência de vários *hubs* no conjunto de dados (pontos

destacados em branco com o valor h_K na Figura 5(a)), a ideia é permitir que o usuário possa interferir já no início do processo de agrupamento, selecionando k representantes iniciais dentre esses *hubs* (Figura 5(b)). Essa característica do algoritmo é especialmente útil na análise de domínios de dados complexos, que pode ser afetada pelo *gap semântico* existente entre as características de baixo nível, que representam as instâncias de dados, e o alto nível da percepção humana. Um exemplo de uma aplicação prática da seleção de representantes iniciais no domínio de dados de imagens pode ser visualizado na Figura 6. Considerando a existência de uma aplicação que exiba miniaturas correspondentes as instâncias de maior pontuação *hubness* (calculadas com base em características de baixo nível, como: cor, textura ou forma) de um conjunto de dados de imagens, um usuário pode, com base em seu conhecimento do domínio de dados, selecionar representantes inicias para cada grupo desejado.

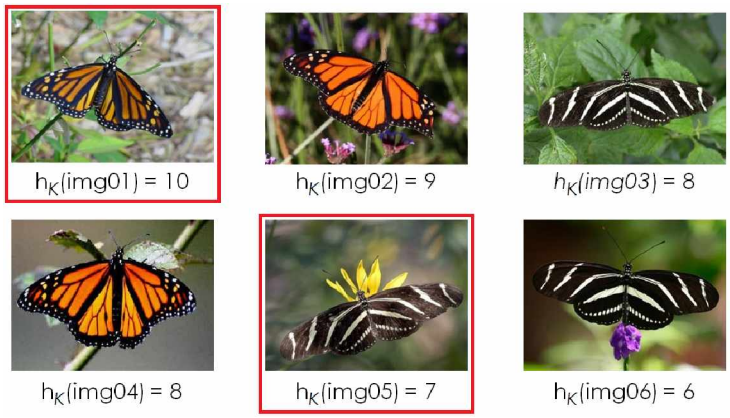

Figura 6 – Seleção de k ($k = 2$) representantes iniciais em um conjunto de dados de imagens, com base no conhecimento de um usuário especialista. Adaptado de (WANG; MARKERT; EVERINGHAM, 2009).

Após a seleção dos representantes iniciais, realiza-se a atribuição de instâncias aos grupos (linha 12 do Algoritmo 9). Nessa etapa, o algoritmo *SSHub Clustering* considera a menor distância agregada da instância x_i para o representante principal e os auxiliares respeitando as restrições *must-link* e *cannot-link* informadas. A Equação 14 exibe como é feito o cálculo dessa distância:

$$\delta_g(Q_k, x_i) = \min_{q_j \in Q_k} (\delta(q_j, x_i)) \tag{14}$$

onde Q_k é o conjunto de representantes (principal e auxiliares) de um grupo C_k e $\delta()$ é uma função de distância. É importante ressaltar que na primeira iteração do algoritmo não existem restrições e nem representantes auxiliares. Essas informações são consideradas somente a partir da segunda iteração do algoritmo.

Também cabe destacar que no algoritmo *SSHub Clustering* é possível violar restrições no caso de contradições na definição. Por exemplo, caso a instância x_i possua restrições

must-link com representantes de um grupo C_k e também com representantes de um grupo C_l. Neste cenário, considera-se somente a primeira restrição analisada e as demais são descartadas. Outro caso, é a possibilidade de existir uma restrição *must-link* e outra *cannot-link* com representantes de um mesmo grupo C_k. Neste caso, privilegia-se restrições *must-link* em relação as *cannot-link*, mas é necessário analisar todos os grupos, pois outros grupos não necessariamente contêm contradições de restrições com a instância x_i.

A atualização dos representantes principais de cada grupo é baseada na pontuação *hubness* das instâncias pertencentes a cada grupo (linhas 13 a 18 do Algoritmo 9). Contudo, essa pontuação pode ser alterada ao longo das iterações do algoritmo. Inspirado nas abordagens dos algoritmos *HPC* e *HPKM* (TOMASEV et al., 2014), instâncias que não trocam de grupo ao longo das iterações são privilegiadas elevando suas pontuações *hubness* ao quadrado (linha 14 do Algoritmo 9). No caso de haver mudança de grupo, a pontuação *hubness* retorna ao valor original (linha 16 do Algoritmo 9). Dessa forma, o representante de cada grupo corresponde a instância com a maior pontuação *hubness* acumulada (linha 18 do Algoritmo 9).

Na linha 19 do Algoritmo 9, considerando as instâncias de dados atribuídas a cada grupo $C_i \in \pi$, deseja-se selecionar um subconjunto de instâncias, de tal forma que obter conhecimento adicional sobre elas permita guiar o agrupamento dos dados para um particionamento mais adequado. Para tanto, as f instâncias mais distantes de cada $p_i \in P$ com pontuação *hubness* ≥ 1 são avaliadas. Essa restrição para a pontuação *hubness* é para evitar que possíveis *anti-hubs* sejam selecionados. A subrotina responsável por essa tarefa está detalhada na Seção 3.1.1.

Resumidamente, os passos realizados pelo *SSHub Clustering* são descritos a seguir:

(1) Selecionar k instâncias com alta pontuação *hubness* como representantes iniciais $P = \{p_i, ..., p_k\}$ e fazer $A = P$, por meio da interação com o usuário, explorando o recurso de *aprendizado ativo*;

(2) Atribuir cada instância $x_i \in X$ ao grupo com menor distância agregada δ_g em relação a Q;

(3) Atualizar os representantes principais P de cada grupo com base nas pontuações *hubness* das instâncias atribuídas para cada grupo;

(4) Analisar as f instâncias de fronteira para definir restrições *must-link* e *cannot-link* e mais representantes auxiliares;

(5) Retornar ao passo (2) até convergir ou até atingir o número máximo de iterações permitidas.

O critério de parada do algoritmo *SSHub Clustering* (passo 5) verifica duas possibilidades: se o número máximo de iterações foi atingido ou se a convergência do algoritmo foi obtida. A função de convergência considerada verifica a soma das distâncias para os representantes (principal e auxiliares) da iteração atual em relação a iteração anterior. A

Equação 15 apresenta como esse cálculo é realizado:

$$\sum_{k=1}^{k} \sum_{x_i \in C_k} \min_{q_j \in Q_k} (\delta(q_j, x_i)) \tag{15}$$

onde x_i é uma instância de um grupo C_k, Q_k é o conjunto de representantes (principal e auxiliares) do respectivo grupo e $\delta()$ é uma função de distância.

3.1.1 Detalhamento sobre a criação das restrições

Algoritmo 10: CriaRestricoes

Entrada: Grupos π, *Hubness* H, Elementos de fronteira f, Representantes A, Restrições R

```
1  início
2      t = |N| · f;
3      para cada grupo C_i ∈ π faça
4          q = (|C_i|/|N|) · t;
5          para j = 1 até j = q faça
6              x_y = Instância em C_i mais distante de p_i com h_K ≥ 1;
7              x_w = Instância mais perto de x_y em C_w | w ≠ i;
8              se p_i e x_y devem estar no mesmo grupo então
9                  R = R ∪ r_ml(p_i, x_y);
10                 A_i = A_i ∪ x_y;
11             senão
12                 R = R ∪ r_cl(p_i, x_y);
13             se x_w e x_y devem estar no mesmo grupo então
14                 R = R ∪ r_ml(x_w, x_y);
15             senão
16                 R = R ∪ r_cl(x_w, x_y);
17             Cria restrições via transitividade;
18             j++;
19 fim
```

A definição da quantidade de instâncias q que devem ser analisadas em cada grupo é feita de maneira proporcional a quantidade de instâncias atribuídas para cada grupo (linha 4 do Algoritmo 10). A partir da identificação de cada instância de fronteira x_y de um dado grupo C_i, seleciona-se a instância x_w de C_j mais próxima de x_y, tal que $C_i \neq C_j$. Com isso, é possível analisar pares de instâncias de fronteira de grupos distintos e questionar se elas devem ou não estar em um mesmo grupo e gerar restrições *must-link*

$r_{ml}(x_y, x_w)$ (linha 14 do Algoritmo 10) ou restrições *cannot-link* $r_{cl}(x_y, x_w)$ (linha 16 do Algoritmo 10), respectivamente.

Além disso, para obter representantes auxiliares para os grupos, cada x_y de um dado grupo C_i é analisado para questionar se x_y e p_i devem ou não estar em um mesmo grupo. Para tanto, utiliza-se novamente o recurso de *aprendizado ativo*, com a intervenção do usuário. Caso a resposta seja positiva, cria-se uma restrição *must-link* $r_{ml}(x_y, p_i)$ (linha 9 do Algoritmo 10) e atribui-se x_y a A (linha 10 do Algoritmo 10). Caso contrário, cria-se uma restrição *cannot-link* $r_{cl}(x_y, p_i)$ (linha 12 do Algoritmo 10). Por fim, são criadas restrições via transitividade (linha 17 do Algoritmo 10). Por exemplo, se existem as restrições $r_{ml}(p_i, x_y)$ e $r_{ml}(x_y, x_w)$, então a restrição $r_{ml}(p_i, x_w)$ também deve existir.

A Figura 7 ilustra os quatro cenários possíveis do processo de análise de elementos de fronteira de cada grupo do Algoritmo 10. A Figura 7(a) apresenta o cenário no qual são geradas somente restrições *must-link*, isto é, p_i, x_y e x_w devem estar no mesmo grupo. A Figura 7(b) apresenta o cenário no qual é gerada uma restrição *must-link* entre a instância de fronteira x_y com o seu respectivo representante principal p_i e uma restrição *cannot-link* entre a instância de fronteira x_y e a instância do outro grupo x_w, ou seja, p_i e x_y devem estar no mesmo grupo e x_y e x_w não devem estar no mesmo grupo.

A Figura 7(c) apresenta o cenário no qual é gerada uma restrição *cannot-link* entre a instância de fronteira x_y e o seu respectivo representante principal p_i e uma restrição *must-link* entre a instância de fronteira x_y e a instância do outro grupo x_w. Neste caso, as duas instâncias de fronteira, x_y e x_w, devem estar no mesmo grupo. E o último cenário no qual somente restrições *cannot-link* são geradas é apresentado na Figura 7(d). Neste cenário, é gerada uma restrição *cannot-link* entre a instância de fronteira x_y e o seu respectivo representante principal p_i e uma restrição *cannot-link* entre a instância de fronteira x_y e a instância do outro grupo x_w, ou seja, p_i, x_y e x_w não devem estar no mesmo grupo.

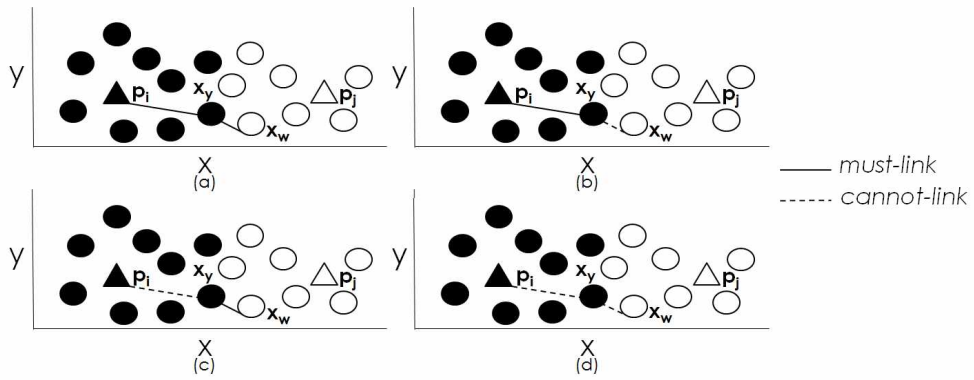

Figura 7 – Cenários possíveis para a criação das restrições *must-link* e *cannot-link* no Algoritmo 10.

3.2 Considerações finais

Este capítulo apresentou a descrição do algoritmo de agrupamento semi-supervisionado de dados para dados de alta dimensão, chamado *SSHub Clustering*, desenvolvido no trabalho descrito aqui. Este algoritmo combina estratégias de semi-supervisão e de estimativa de densidade baseada em pontuações *hubness*, a fim de obter particionamentos mais próximos da real estrutura do conjunto de dados. Para tanto, ele analisa as instâncias de fronteira, para a definição de restrições *must-link* e *cannot-link*, e posteriormente deriva representantes auxiliares para os grupos. Essa característica contribui para permitir que o algoritmo possa lidar bem tanto com grupos com formatos hiperesféricos como com grupos que apresentem formas mais complexas. No próximo capítulo são apresentados os experimentos realizados e a discussão a respeito da eficácia e eficiência dos resultados obtidos.

Capítulo 4

Experimentos e Análise dos Resultados

Este capítulo apresenta os experimentos realizados com o algoritmo *SSHub Clustering* e mais quatro outros algoritmos da literatura com o objetivo de avaliar a eficácia e a eficiência dos resultados obtidos dos algoritmos comparados. O capítulo está organizado da seguinte maneira: a Seção 4.1 descreve o método de avaliação empregado, detalhando os conjuntos de dados considerados para os experimentos e os parâmetros utilizados; a Seção 4.2 descreve os resultados obtidos, incluindo o teste estatístico e a análise da eficiência dos algoritmos; a Seção 4.3 detalha os experimentos que foram realizados com a variação de parâmetros, com o objetivo de analisar o quanto os valores definidos para alguns parâmetros podem influenciar na qualidade dos resultados. Por fim, a Seção 4.4 apresenta as considerações finais do capítulo.

4.1 Descrição do método de avaliação

Para a realização dos experimentos foram considerados 23 conjuntos de dados reais. Os detalhes a respeito de cada um deles são apresentados na Tabela 1. Desses conjuntos de dados, 22 foram obtidos da *UCI Machine Learning* (LICHMAN, 2013) e o conjunto de dados *NBA* está disponível no repositório *BasketballStats* (DATABASESPORTS.COM, 2011). É importante destacar que foram selecionados conjuntos de dados com diferentes números de classes, densidades e dimensionalidades, com o intuito de verificar o desempenho do algoritmo proposto frente a outros quatro algoritmos descritos na literatura científica da área em diferentes cenários.

Todos esses conjuntos de dados foram pré-processados de acordo com as recomendações indicadas nesses repositórios. Além disso, para o conjunto de dados *Covertype* (5) foi considerada uma amostra com 19.229 instâncias selecionadas aleatoriamente do conjunto original mantendo a proporção de instâncias por classe. Para o conjunto de dados *Hepmass* (8) também foi considerada uma amostra com 45.000 instâncias do conjunto original, igualmente divididas entre as duas classes. E para o conjunto de dados *Digits389* (6) foram consideradas apenas as instâncias das classes 3, 8 e 9 do conjunto de dados *Pendigits*,

assim como foi feito em (XIONG; AZIMI; FERN, 2014).

Tabela 1 – Conjuntos de dados considerados nos experimentos.

ID	Conjunto	Instâncias	Dimensões	Classes
1	Abalone	4.176	8	3
2	Aloi	1.098	64	10
3	Breast	683	9	2
4	Churn	5.000	14	2
5	Covertype	19.229	54	7
6	Digits389	3.165	16	3
7	Ecoli	336	7	8
8	Hepmass	45.000	27	2
9	Isolet	6.237	617	26
10	Musk2	6.598	166	2
11	NBA	22.064	17	2
12	Page blocks	5.473	10	5
13	Pendigits	10.992	16	10
14	Segment	2.310	19	7
15	Sonar	208	60	2
16	Spambase	4.599	57	2
17	Urbanlandcover	675	147	9
18	Vehicle	846	18	4
19	Vowel	990	13	11
20	Wine	178	13	3
21	WDBC	569	30	2
22	Yeast	1.484	8	10
23	Zoo	101	16	7

Os experimentos foram executados em um computador equipado com um processador Intel Core i5-3230M CPU @ 2.60GHz com 6 GB de RAM, disco rígido SATA de 500 GB (7.200 rpm) utilizando o sistema operacional Microsoft Windows 7 Ultimate SP1 64 bits. Os algoritmos *HPKM*, *Kernel k-means* (DHILLON; GUAN; KULIS, 2004), *DBSCAN* (SANDER et al., 1998) e *SSDBSCAN* (LI et al., 2014) foram selecionados como linha de base para comparação (veja Seção 2.3).

O *HPKM* foi o algoritmo de agrupamento baseado em *hubs* que inspirou a proposta do *SSHub Clustering*. O *Kernel k-means* é um representante clássico de algoritmos de agrupamento baseados em *kernel* que são conhecidos por lidarem bem com grupos não hiperesféricos e o *DBSCAN* é um representante padrão de algoritmos de agrupamento baseados em densidade. Como representante da abordagem de agrupamento semi-supervisionado foi escolhido a extensão do *DBSCAN*, denominada *SSDBSCAN*, que incorpora informação adicional na forma de rótulos no processo de agrupamento baseado em densidade. A implementação desse algoritmo foi disponibilizada pelos autores do trabalho e foi realizada na linguagem Java. As implementações utilizadas dos algoritmos *HPKM*, *Kernel k-means* e *DBSCAN* também foram disponibilizadas pelos autores do trabalho, descrito em (TO-

MASEV et al., 2014), no repositório (TOMASEV, 2014), sendo todas na linguagem Java. Com o intuito de permitir a comparação dos algoritmos, o algoritmo *SSHub Clustering* também foi implementado utilizando a linguagem Java.

Considerando que o tamanho ideal de vizinhança a ser analisado no cálculo da pontuação *hubness* pode variar de acordo com o domínio de dados, foram considerados quatro valores distintos de vizinhança ($K = 5$, 10, 15 e 20) para o *SSHub Clustering* e o *HPKM*. A quantidade de grupos solicitada (k) seguiu a quantidade de classes definida para cada conjunto de dados e detalhada na Tabela 1. Todos os algoritmos consideraram a função de distância Euclidiana.

Para automatizar a realização de experimentos com o *SSHub Clustering*, a etapa de interação com o usuário na escolha dos representantes iniciais foi simulada da seguinte maneira: levando em consideração as informações de rótulo dos conjuntos de dados, os representantes iniciais foram sorteados entre as cinco instâncias de cada grupo com as maiores pontuações *hubness*. Dessa forma, é possível simular o fato de que a cada execução o usuário poderia escolher diferentes instâncias de alta pontuação *hubness* como representantes. Além disso, a quantidade total de elementos de fronteira (f) analisados pelo *SSHub Clustering* foi definida em 1% do total de instâncias. O mesmo valor de 1% foi considerado para as instâncias rotuladas do algoritmo *SSDBSCAN*. Esse limite para a informação de semi-supervisão foi definido em 1% pelo fato de que em aplicações reais nas quais o usuário interage no processo de agrupamento semi-supervisionado, não se espera demandar que o usuário informe uma quantidade muito elevada de restrições.

Conforme mostrado em (TOMASEV et al., 2011), a utilização de *hubs* como representantes iniciais resulta em uma rápida convergência do algoritmo, isto é, são necessárias poucas iterações para o algoritmo convergir. Esse fato também foi observado na experimentação conduzida na realização do trabalho descrito aqui. Assim, o número máximo de iterações para o algoritmo *SSHub Clustering* foi definido como 15. Para os algoritmos *Kernel k-means*, *DBSCAN* e *SSDBSCAN* foram selecionados os melhores parâmetros conhecidos na literatura para cada conjunto de dados.

O principal objetivo dos experimentos realizados foi avaliar e comparar a eficácia dos algoritmos em diferentes cenários, principalmente naqueles em que existem um maior número de dimensões. Nesse contexto, foi necessário utilizar uma medida de avaliação que permitisse mensurar o quanto um algoritmo é capaz de encontrar a verdadeira estrutura de um conjunto de dados. Considerando a existência de conhecimento a respeito do particionamento desejado para os conjuntos de dados, foi utilizado o índice *Rand* Corrigido (HUBERT; ARABIE, 1985) para avaliar a eficácia dos algoritmos. Nesse índice, a qualidade dos resultados é representada no intervalo $[-1, 1]$, sendo que quanto mais próximo de 1, melhor é a correspondência entre o agrupamento obtido e o particionamento desejado para o conjunto de dados avaliado.

Como os algoritmos empregados nos experimentos apresentam a característica de inici-

alização não determinística, eles foram executados 50 vezes para cada avaliação realizada. Assim, cada medida apresentada considera a média de 50 execuções dos algoritmos desconsiderando a melhor e a pior execução de cada um deles. As Seções 4.2 e 4.3 apresentam as análises dos resultados obtidos na experimentação.

4.2 Comparação entre os algoritmos

Esta seção apresenta os resultados dos experimentos que permitem comparar a eficiência do algoritmo *SSHub Clustering* em relação aos algoritmos *HPKM*, *Kernel k-means*, *DBSCAN* e *SSDBSCAN*. Além disso, também são apresentadas medidas do tempo requerido por cada um dos algoritmos na execução desses experimentos. Os resultados obtidos são exibidos nas Tabelas 2 e 3. Nessas tabelas, os rótulos *KerKM* e *SSDBS* representam os algoritmos *Kernel k-means* e *SSDBSCAN*, respectivamente.

Tabela 2 – Resultados dos experimentos, considerando o índice *Rand* Corrigido. Os valores em negrito destacam os melhores desempenhos. Os números entre () indicam a posição no *rank*.

	SSHub Clustering				***HPKM***						
ID	$K = 5$	$K = 10$	$K = 15$	$K = 20$	$K = 5$	$K = 10$	$K = 15$	$K = 20$	***KerKM***	***DBSCAN***	***SSDBS***
1	0,08	**0,17 (1)**	0,10	0,11	0,14 (2,5)	0,14	0,14	0,13	0,00 (4,5)	0,00 (4,5)	0,14 (2,5)
2	0,65	0,61	0,62	**0,72 (1)**	0,53	0,52	0,53	0,54 (3)	0,00 (5)	0,28 (4)	0,68 (2)
3	**0,90 (1)**	0,88	0,86	0,85	0,85 (2)	0,85	0,85	0,85	0,78 (4)	0,01 (5)	0,84 (3)
4	-0,02	**0,15 (1)**	0,03	0,02	-0,09	-0,09	-0,09	-0,05 (5)	0,00 (3,5)	0,00 (3,5)	0,10 (2)
5	0,20	0,23	0,24 (2)	0,19	0,18	0,19	0,19	0,20 (3)	0,00 (5)	0,11 (4)	**0,37 (1)**
6	0,10	0,74	0,79 (2)	0,47	0,33	0,54 (4)	0,33	0,35	0,00 (5)	0,66 (3)	**0,95 (1)**
7	0,70	0,73	0,63	**0,74 (1)**	0,44	0,42	0,43	0,49 (2)	0,00 (5)	0,06 (4)	0,31 (3)
8	0,32 (2)	0,22	0,30	0,19	0,33	0,32	**0,34 (1)**	0,32	0,00 (4,5)	0,00 (4,5)	0,05 (3)
9	**0,52 (1)**	0,49	0,49	0,50	0,43	0,40	0,44 (2)	0,43	0,00 (5)	0,02 (4)	0,39 (3)
10	0,07 (2)	-0,01	-0,01	-0,02	-0,06	-0,03 (5)	-0,04	-0,03	0,02 (3)	0,00 (4)	**0,27 (1)**
11	-0,01	**0,74 (1)**	-0,03	0,63	0,00 (5)	-0,01	0,00	0,00	0,07 (4)	0,13 (3)	0,14 (2)
12	-0,04	0,11	0,10	0,17 (3)	0,10	0,11 (4)	0,08	0,10	0,00 (5)	0,36 (2)	**0,55 (1)**
13	0,67 (2)	0,49	0,56	0,54	0,54	0,55	0,56 (3)	0,53	0,00 (5)	0,37 (4)	**0,87 (1)**
14	0,37	0,50 (3)	0,46	0,50	0,48	0,48	0,51 (2)	0,48	0,00 (5)	0,28 (4)	**0,65 (1)**
15	0,02	0,00	0,02	**0,03 (1)**	0,00	0,00	0,00	0,01 (2)	0,00 (4)	0,00 (4)	0,00 (4)
16	-0,03	-0,01	-0,01	0,00 (4)	-0,01 (5)	-0,01	-0,03	-0,02	0,10 (2)	0,03 (3)	**0,22 (1)**
17	0,08	0,42 (2)	0,42	0,42	0,44	0,44	**0,51 (1)**	0,42	0,00 (5)	0,06 (4)	0,23 (3)
18	0,13 (2)	0,07	0,08	0,09	0,08 (3)	0,08	0,08	0,07	0,00 (4,5)	0,00 (4,5)	**0,17 (1)**
19	0,05	0,07	**0,09 (1,5)**	0,06	0,07	0,07	0,08 (3)	0,08	0,00 (5)	**0,09 (1,5)**	0,04 (4)
20	0,37	0,83	0,86 (2)	0,77	**0,91 (1)**	0,85	0,86	0,85	0,00 (4)	-0,01 (5)	0,45 (3)
21	0,60	0,73 (2)	0,19	0,26	**0,74 (1)**	0,74	0,74	0,71	0,00 (5)	0,01 (4)	0,30 (3)
22	**0,16 (1,5)**	0,15	0,15	0,15	0,14	0,15	**0,16 (1,5)**	0,15	0,00 (5)	0,02 (4)	0,08 (3)
23	0,80	**0,96 (1)**	0,91	0,91	0,68	0,61	0,67	0,77 (2)	0,73 (3)	0,40 (4)	0,10 (5)
Soma *ranks*	40				63				101	87,5	53,5
Média *ranks*	1,74				2,74				4,39	3,8	2,32

Observando a Tabela 2 é possível verificar que os algoritmos semi-supervisionados, *SSHub Clustering* e *SSDBSCAN*, apresentaram eficácia superior em 17 dos 23 conjuntos de dados testados, além de 2 empates na primeira colocação, sendo que desses 17 conjuntos

de dados, o *SSHub Clustering* apresentou eficácia superior em 9 deles. Esses resultados ajudam a corroborar a afirmação de que abordagens semi-supervisionadas auxiliam na obtenção de particionamentos mais próximos da real estrutura dos conjuntos de dados.

Analisando os resultados apresentados na Tabela 2 também é possível constatar que a vizinhança definida para o cálculo da pontuação *hubness* pode impactar nos particionamentos gerados pelo *SSHub Clustering*. Este fato pode ser observado, principalmente, nos conjuntos (2), (6), (7), (8), (11), (13), (14), (17), (20), (21) e (23). O mesmo comportamento não é evidente nos resultados apresentados pelo *HPKM*. Assim, torna-se necessário investigar o quanto cada uma das estratégias empregadas pelo *SSHub Clustering* colaboram para a obtenção dos bons resultados. Os resultados dessa avaliação são apresentados na Seção 4.3.

4.2.1 Teste estatístico

Com o objetivo de verificar se existe diferença significativa entre a eficácia dos algoritmos avaliados foram considerados os testes estatísticos de *Friedman* (DEMSAR, 2006) e *post-hoc de Bonferroni-Dunn* (ZAR, 2007). A Tabela 2 apresenta o valor da soma dos *ranks* e a média dos *ranks* do desempenho de cada algoritmo, que são utilizadas durante o teste estatístico (mais detalhes na Seção 2.4).

Primeiramente, duas hipóteses são criadas:

- H_0 (hipótese nula): "os cinco algoritmos apresentaram a mesma eficiência nos experimentos realizados";
- H_A (hipótese alternativa): "o algoritmo *SSHub Clustering* apresentou eficiência superior aos outros algoritmos nos experimentos realizados".

Então, busca-se responder com $\alpha = 0{,}05$, ou seja, com 95% de confiança, se H_0 é verdadeira. Caso H_0 seja rejeitada, H_A é considerada. Para realizar essa verificação, o teste de *Friedman* χ_F^2 foi aplicado e é apresentado a seguir:

$$\chi_F^2 = \left[\frac{12}{NA(A+1)} \cdot \sum_{i=1}^{A} R_i^2\right] - (3N(A+1))$$

$$\chi_F^2 = \left[\frac{12}{23 \cdot 5(5+1)} \cdot (40^2 + 63^2 + 101^2 + 87,5^2 + 53,5^2)\right] - (3 \cdot 23(5+1))$$

$$\chi_F^2 = [0,01739 \cdot 26288,5] - (414)$$

$$\chi_F^2 = 457,15 - 414 = 43,15$$

onde N corresponde ao número de conjuntos de dados, A ao número de algoritmos testados e R aos valores da soma dos *ranks* de cada algoritmo (veja Tabela 2).

Contudo, o trabalho descrito em (IMAN; DAVENPORT, 1980) mostrou que o teste de *Friedman* é indesejavelmente conservador e é necessário realizar outros cálculos para

complementar o teste. Com isso, o valor de χ_F^2 é usado para compor o cálculo F_F, apresentado a seguir:

$$F_F = \frac{(N-1)\chi_F^2}{N(A-1)-\chi_F^2} = \frac{(23-1)43,15}{23(5-1)-43,15} = \frac{22 \cdot 43,15}{92-43,15} = \frac{949,3}{48,85} = 19,43$$

Para a aplicação do teste $(A-1)$ e $(A-1)(N-1)$ graus de liberdade são considerados. Sendo assim, com os valores de A e N utilizados nos experimentos, utilizou-se 4 e 88 graus de liberdade para a consulta na tabela de valores críticos da *F-Distribution*. Observando a Tabela apresentada em (ZAR, 2007) e no Apêndice A.1, o valor 2,47 foi obtido. Com esse resultado, é possível afirmar que existe diferença significativa entre o desempenho dos algoritmos, pois 2,47 < 19,43, e rejeita-se a hipótese nula. Então, para complementar o resultado da avaliação anterior e verificar qual dos algoritmos teve o desempenho superior, o teste *post-hoc Bonferroni-Dunn* foi aplicado.

No teste *post-hoc Bonferroni-Dunn*, primeiramente, define-se a diferença crítica (CD) de desempenho dos algoritmos, para delimitar um valor que possibilite comparar se um algoritmo é superior a outro ou não. O cálculo de CD é apresentado a seguir:

$$CD = q_\alpha \sqrt{\frac{A(A+1)}{6N}} = 2,498\sqrt{\frac{5 \cdot 6}{6 \cdot 23}} = 2,498\sqrt{0,21739} = 2,498 \cdot 0,4662 = 1,16$$

onde o valor crítico q_α é baseado na *Studentized Range Statistic* dividida por $\sqrt{2}$, apresentada em (DEMSAR, 2006) e no Apêndice A.2.

Após o cálculo de CD é necessário calcular a diferença (z) do *rank* médio ($\bar{R}$) entre cada par de algoritmos. O algoritmo *SSHub Clustering* foi fixado como o algoritmo de controle para compará-lo aos demais. O cálculo de z_1 refere-se a comparação entre o *HPKM* e o *SSHub Clustering*:

$$z_1 = \frac{(\bar{R}_{HPKM} - \bar{R}_{SSHub})}{\sqrt{\frac{A(A+1)}{6N}}} = \frac{(2,74-1,74)}{\sqrt{\frac{5(5+1)}{6 \cdot 23}}} = \frac{1}{\sqrt{0,21739}} = \frac{1}{0,4662} = 2,14$$

Considerando o valor calculado em z_1, é possível afirmar que o algoritmo *SSHub Clustering* é estatisticamente superior ao algoritmo *HPKM*, pois 2,14 é maior que CD (1,16). A seguir, o cálculo de z_2, que corresponde a comparação entre os algoritmos *Kernel k-means* e *SSHub Clustering*, é apresentado:

$$z_2 = \frac{(\bar{R}_{kerKM} - \bar{R}_{SSHub})}{\sqrt{\frac{A(A+1)}{6N}}} = \frac{(4,39-1,74)}{\sqrt{\frac{5(5+1)}{6 \cdot 23}}} = \frac{2.65}{0,4662} = 5,68$$

Observando o valor obtido em z_2, é possível concluir que o algoritmo *SSHub Clustering* também é estatisticamente superior ao algoritmo *Kernel k-means*, pois o valor calculado (5,68) é maior que CD (1,16). Em seguida, o cálculo de z_3 é apresentado e se refere à comparação entre os algoritmos *DBSCAN* e *SSHub Clustering*:

$$z_3 = \frac{(\bar{R}_{DBSCAN} - \bar{R}_{SSHub})}{\sqrt{\frac{A(A+1)}{6N}}} = \frac{(3,8-1,74)}{\sqrt{\frac{5(5+1)}{6 \cdot 23}}} = \frac{2.06}{0,4662} = 4,41$$

Com o valor calculado em z_3, é possível afirmar que o algoritmo *SSHub Clustering* é estatisticamente superior ao algoritmo *DBSCAN*, pois 4,41 é maior que *CD* (1,16). A última comparação é realizada em z_4 e corresponde aos algoritmos *SSDBSCAN* e *SSHub Clustering*:

$$z_4 = \frac{(\bar{R}_{SSDBS} - \bar{R}_{SSHub})}{\sqrt{\frac{A(A+1)}{6N}}} = \frac{(2,32 - 1,74)}{\sqrt{\frac{5(5+1)}{6 \cdot 23}}} = \frac{0,58}{0,4662} = 1,24$$

Considerando o valor obtido com o cálculo de z_4, nota-se que o algoritmo *SSHub Clustering* também é estatisticamente superior ao algoritmo *SSDBSCAN*, pois 1,24 é maior que *CD* (1,16). Com isso, baseado nos valores de z_1, z_2, z_3 e z_4, é possível afirmar, com 95% de confiança, que a eficácia do algoritmo *SSHub Clustering* é superior a dos demais algoritmos selecionados como linha de base para os conjuntos de dados testados.

4.2.2 Análise da eficiência

Além da eficácia dos algoritmos também foi realizada a análise da eficiência dos mesmos. A Tabela 3 apresenta os tempos (em milissegundos) gastos para a obtenção dos resultados apresentados na Tabela 2 para cada algoritmo, além do tempo requerido pelos algoritmos *HPKM* e *SSHub Clustering* para o cálculo da pontuação *hubness* dos conjuntos de dados. O tempo para o cálculo da pontuação *hubness* foi computado para as diferentes vizinhanças consideradas, isto é, (5), (10), (15) e (20). Esse tempo foi considerado separadamente, pois se trata de uma fase de pré-processamento dos algoritmos *HPKM* e *SSHub Clustering*.

As Figuras 8 e 9 também apresentam os tempos (em milissegundos) gastos para a obtenção dos resultados descritos na Tabela 2, mas em formato de gráfico de colunas. A Figura 8 mostra os gráficos dos conjuntos de dados (1) a (14) e a Figura 9 dos conjuntos de dados (15) a (23). Nos gráficos os rótulos das colunas foram apenas numerados por questões de espaço. A Tabela 4 apresenta a legenda de cada rótulo empregado nas Figuras 8 e 9. Para os algoritmos *SSHub Clustering* e *HPKM*, considerou-se a soma do tempo gasto com o cálculo *hubness* e do tempo de execução dos algoritmos. Para tanto, as respectivas colunas foram divididas em duas cores: a parte em azul se refere ao cálculo *hubness* e a parte em vermelho a execução do algoritmo.

Nota-se, por meio da análise da Tabela 3 e das Figuras 8 e 9, que o algoritmo *HPKM* apresenta os menores tempos de execução em 19 dos 23 conjuntos de dados testados. Comparando o *HPKM* e o *SSHub Clustering*, já era esperado que o algoritmo *HPKM* apresentasse um menor tempo de execução, uma vez que ele não possui a sobrecarga da abordagem de semi-supervisão. Entretanto, é importante ressaltar que, apesar disso, o *SSHub Clustering* demandou menos tempo do que o *HPKM* para 4 dos conjuntos de dados avaliados (os conjuntos 5, 15, 20 e 23).

Analisando o desempenho do algoritmo *SSHub Clustering* em relação aos demais algoritmos (isto é, desconsiderando o algoritmo HPKM) é possível verificar que ele apresenta os menores tempos de execução em 18 dos 23 conjuntos de dados avaliados. Além disso, quando são consideradas apenas as abordagens de agrupamento semi-supervisionadas, *SSHub Clustering* e *SSDBSCAN*, o algoritmo *SSHub Clustering* apresenta os menores tempos para todos os conjuntos de dados. Vale destacar que, mesmo quando considera-se a soma do tempo gasto com o cálculo da pontuação *hubness* ao tempo do processamento do algoritmo *SSHub Clustering*, ele demandou menos tempo do que o *SSDBSCAN* para 19 dos 23 conjuntos de dados. Por exemplo, para o conjunto de dados 8 o *SSHub Clustering* demandou 74% menos tempo do que o *SSDBSCAN* e obteve uma melhor eficácia (veja a Tabela 2).

Tabela 3 – Tempo (em milissegundos) das execuções de cada algoritmo. A coluna *hubness* representa o tempo para cálculo da pontuação *hubness* dos conjuntos de dados considerando os valores (5), (10), (15) e (20) para a vizinhança (K). Os valores em negrito destacam os menores tempos. A coluna C representa o ID do conjunto de dados.

	Hubness				***SSHub Clustering***				***HPKM***						
C	$K=5$	$K=10$	$K=15$	$K=20$	$K=5$	$K=10$	$K=15$	$K=20$	$K=5$	$K=10$	$K=15$	$K=20$	***KerKM***	***DBSCAN***	***SSDBS***
1	1.327	1.327	1.343	1.358	540	630	586	586	39	**28**	40	34	3.866	1.386	1.670
2	297	296	296	296	162	165	150	150	143	**107**	116	113	334	316	588
3	47	47	31	63	23	22	24	21	10	10	28	**4**	107	57	351
4	2.325	2.341	2.356	2.326	1.229	1.307	1.351	1.351	78	**63**	66	96	2.236	2.360	5.460
5	81.011	81.198	81.432	81.588	**540**	755	555	555	1.073	1.177	1.199	1.131	98.989	80.988	2.760
6	1.202	1.157	1.204	1.214	1.200	1.064	1.120	1.049	184	100	**64**	86	3.259	1.195	8.935
7	15	13	15	16	19	25	19	19	26	28	7	9	16	19	140
8	321.890	361.937	305.449	413.557	891.045	752.290	691.510	910.188	**82.345**	102.945	86.362	89.370	1.529.628	404.494	5.069.043
9	75.801	75.472	75.972	75.863	17.170	26.458	16.546	16.546	15.156	15.106	**14.980**	15.969	67.495	75.953	229.852
10	24.399	24.586	24.710	24.632	3.975	3.834	3.847	3.847	383	310	**307**	466	24.181	24.627	35.010
11	49.842	49.873	50.388	50.622	35.016	34.167	32.526	32.526	477	**432**	469	650	98.211	50.084	70.193
12	2.451	2.435	2.465	2.481	1.451	1.707	1.810	1.810	58	**34**	66	50	5.761	2.472	3.543
13	14.103	16.866	16.611	14.461	21.463	23.159	24.320	23..312	**642**	870	918	1.311	37.468	13.964	41.862
14	578	577	594	608	253	269	259	259	81	79	68	**57**	1.071	606	8.329
15	15	16	15	15	**6**	**6**	9	9	26	12	23	16	22	25	117
16	4.727	4.774	4.805	4.789	1.501	1.497	1.863	1.863	144	103	**54**	101	6.331	4.943	7.024
17	218	266	234	234	244	237	294	294	132	**90**	106	122	219	247	885
18	79	79	79	79	31	47	38	38	27	21	**18**	21	65	90	695
19	94	93	94	109	65	56	78	78	46	44	46	**35**	147	103	574
20	14	16	15	15	**3**	**3**	**3**	**3**	16	14	12	13	13	16	374
21	47	47	46	47	22	34	34	34	25	16	13	**13**	56	63	347
22	156	156	172	156	69	65	66	66	60	59	**47**	63	156	178	207
23	1	1	1	1	3	6	**3**	**3**	12	13	13	12	16	9	40

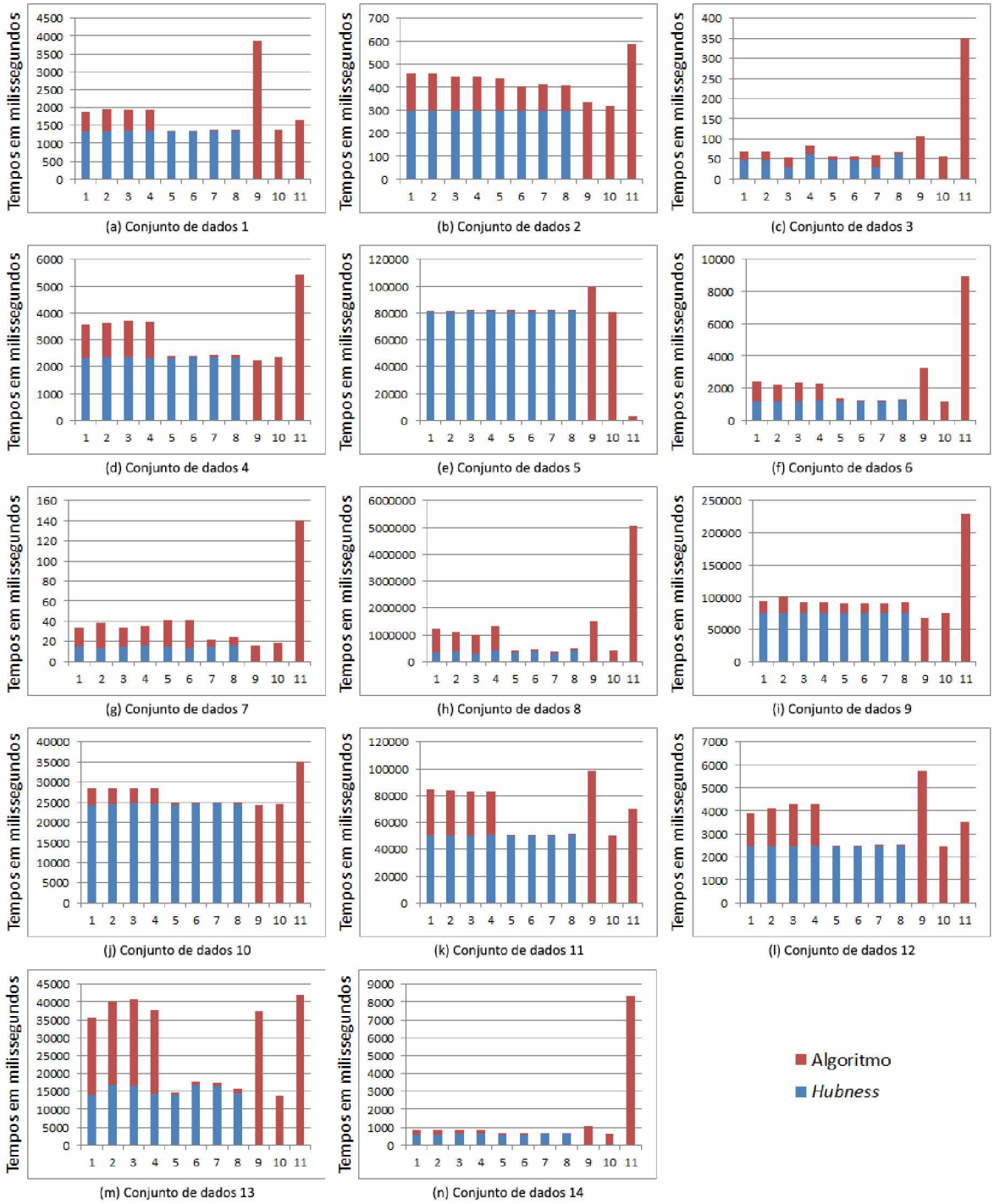

Figura 8 – Tempos (em milissegundos) das execuções de cada algoritmo nos conjuntos de dados (1) a (14). A parte em azul das colunas representa o tempo para cálculo da pontuação *hubness*. A parte em vermelho representa o tempo de execução dos algoritmos.

4.3 Avaliação da variação de parâmetros

Os resultados experimentais apresentados na Seção 4.2 mostram que o algoritmo *SSHub Clustering* possui eficácia superior a todos os algoritmos considerados na avaliação e eficiência superior a três desses algoritmos, perdendo apenas para a abordagem

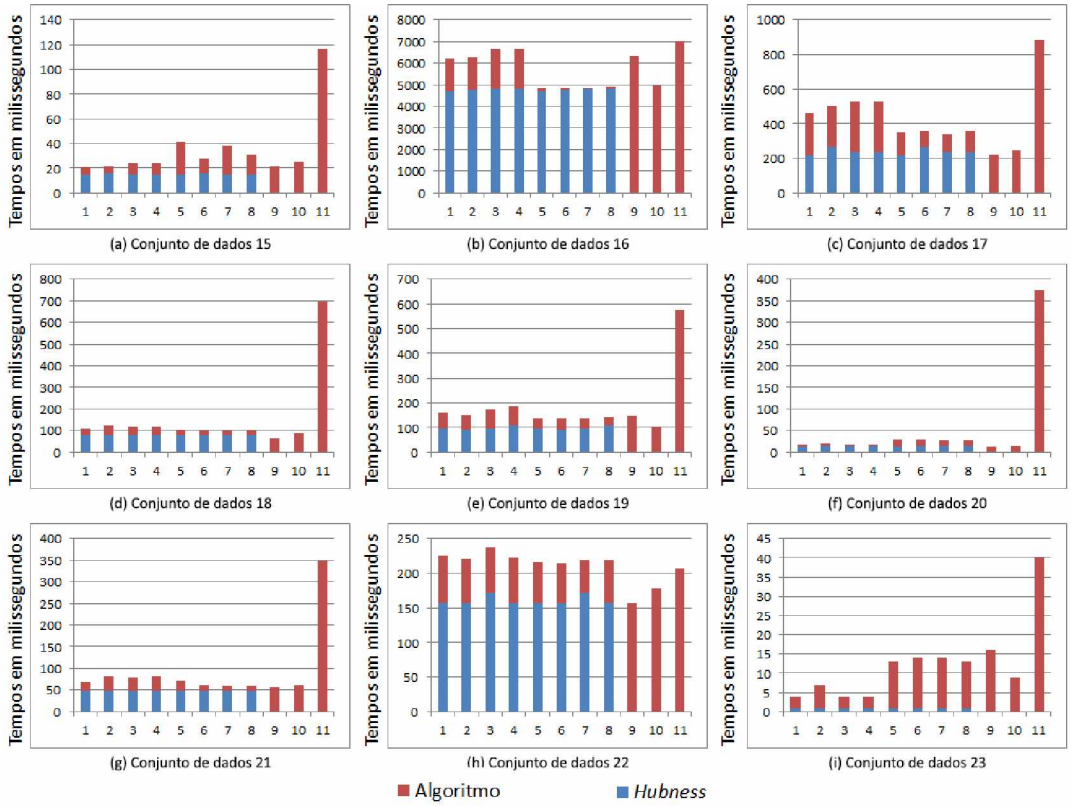

Figura 9 – Tempos (em milissegundos) das execuções de cada algoritmo nos conjuntos de dados (15) a (23). A parte em azul das colunas representa o tempo para cálculo da pontuação *hubness*. A parte em vermelho representa o tempo de execução dos algoritmos.

Tabela 4 – Rótulos utilizados nos gráficos das Figuras 8 e 9.

Coluna	Algoritmo
1	*SSHub Clustering* ($K = 5$)
2	*SSHub Clustering* ($K = 10$)
3	*SSHub Clustering* ($K = 15$)
4	*SSHub Clustering* ($K = 20$)
5	*HPKM* ($K = 5$)
6	*HPKM* ($K = 10$)
7	*HPKM* ($K = 15$)
8	*HPKM* ($K = 20$)
9	*Kernel k-means*
10	*DBSCAN*
11	*SSDBSCAN*

não-supervisionada do *HPKM*. Além disso, esses resultados também mostram uma variação na eficácia do algoritmo *SSHub Clustering* quando a vizinhança definida para o cálculo da pontuação *hubness* varia.

Com o objetivo de investigar a contribuição individual de cada uma das estratégias empregadas pelo *SSHub Clustering* no resultado final do agrupamento os experimentos

descritos aqui consideraram duas variações de parâmetros: não atualização dos representantes principais (Seção 4.3.1) e diferentes percentuais de elementos de fronteira (Seção 4.3.2). Para a realização desses experimentos foram selecionados 10 dos 23 conjuntos de dados utilizados nos experimentos descritos na Seção 4.2 (veja a Tabela 2): Churn (4), Covertype (5), Digits389 (6), Hepmass (8), Isolet (9), Musk2 (10), NBA (11), Pendigits (13), Segment (14) e Urbanlandcover (17). Esses conjuntos foram selecionados levando em conta que, no geral, apresentam as maiores quantidades de instâncias e maior variabilidade de classes e dimensões.

4.3.1 Atualização dos representantes principais

A versão original do *SSHub Clustering* apresentada no Algoritmo 9 (linhas 13 a 18) considera a atualização dos representantes principais a cada iteração. Contudo, como essa etapa é não-supervisionada, existe a possibilidade do representante principal poder ser alterado para uma instância que não possui o mesmo rótulo do representante da iteração anterior, com base no conhecimento do usuário.

Assim, considerou-se investigar se a atualização dos representantes principais pode impactar negativamente na eficácia do algoritmo, uma vez que pode divergir da informação obtida com a seleção dos representantes iniciais na etapa de interação com o usuário. Para essa investigação foi implementada uma nova versão do algoritmo que não atualiza os representantes principais a cada iteração. Essa versão manteve apenas a atualização dos representantes auxiliares que são derivados das restrições *must-link*. A Tabela 5 apresenta os resultados dessa avaliação. A coluna (1) representa a versão original do algoritmo *SSHub Clustering* e a coluna (2) a versão modificada que não considera a atualização dos representantes principais.

Observando a Tabela 5 é possível verificar que para valores menores de K ($K = 5$), percebe-se que a nova versão, sem a atualização dos representantes principais, supera a versão original em 7 dos 10 conjuntos de dados avaliados. Considerando $K = 20$ para as duas abordagens, a nova versão apresenta eficácia superior para todos os 10 conjuntos avaliados. A medida que o valor de K aumenta, a quantidade de conjuntos de dados para os quais a nova versão apresenta resultados superiores aumenta e, além disso, a diferença na eficácia também aumenta.

Esse comportamento pode ser explicado pelo fato de que quando o algoritmo considera uma vizinhança menor (por exemplo, $K = 5$), para a identificação dos *hubs*, estes podem não ser tão representativos inicialmente. Assim, nessa caso, o resultado final do agrupamento pode ser melhorado com a abordagem que permite a atualização dos representantes principais a cada iteração. Por outro lado, quando o algoritmo considera uma vizinhança maior (por exemplo, $K = 20$), os *hubs* tornam-se mais representativos e a seleção dos representantes iniciais na etapa de interação com o usuário já é suficiente para a obtenção de bons resultados.

Tabela 5 – Comparação do algoritmo *SSHub Clustering* com (1) e sem (2) a atualização dos representantes principais. Resultados considerando o índice *Rand* Corrigido.

	***SSHub Clustering* (1)**				***SSHub Clustering* (2)**			
Conjunto de dados	$K = 5$	$K = 10$	$K = 15$	$K = 20$	$K = 5$	$K = 10$	$K = 15$	$K = 20$
4	-0,02	**0,15**	0,03	0,02	0,03	**0,15**	0,11	0,10
5	0,20	0,23	0,24	0,19	0,22	0,24	**0,25**	0,24
6	0,10	0,74	0,79	0,47	0,50	0,95	0,89	**0,97**
8	**0,32**	0,22	0,30	0,19	0,23	0,22	0,17	0,25
9	0,52	0,49	0,49	0,50	0,28	0,55	**0,56**	0,54
10	0,07	-0,01	-0,01	-0,02	0,09	0,20	0,12	**0,25**
11	-0,01	0,74	-0,03	0,63	0,14	0,60	0,61	**0,75**
13	0,67	0,49	0,56	0,54	0,55	**0,79**	0,73	0,76
14	0,37	0,50	0,46	0,50	0,52	0,59	0,59	**0,61**
17	0,08	0,42	0,42	0,42	0,43	0,48	0,47	**0,49**

4.3.2 Variação do percentual das instâncias de fronteira

A abordagem de agrupamento empregada pelo algoritmo *SSHub Clustering* (veja Seção 3) permite semi-supervisão em dois momentos: durante a seleção dos representantes iniciais e durante a análise de elementos de fronteira. Os experimentos descritos nessa Seção têm o objetivo de permitir a avaliação da contribuição da etapa de análise de elementos de fronteira no resultado final do agrupamento. Assim, a Tabela 6 exibe os resultados dos experimentos considerando o parâmetro f variando de 0 a 10% do tamanho do conjunto de dados. É importante destacar que, quando $f = 0\%$, nenhuma instâncias de fronteira é analisada, isto é, a semi-supervisão só é considerada apenas na definição dos representantes iniciais na primeira iteração do algoritmo. Para os experimentos mostrados aqui foi adotado $K = 10$ para o cálculo da pontuação *hubness*.

Tabela 6 – Comparação do algoritmo *SSHub Clustering* com diferentes percentuais para as instâncias de fronteira: 0%, 1%, 5% e 10%. Resultados considerando o índice *Rand* Corrigido.

	SSHub Clustering			
Conjunto de dados	0%	1%	5%	10%
4	0,04	0,15	0,20	**0,33**
5	0,15	0,23	0,20	**0,27**
6	0,59	0,74	0,92	**0,95**
8	0,14	0,22	0,20	**0,27**
9	0,46	0,49	0,54	**0,55**
10	-0,01	-0,01	-0,06	**0,04**
11	0,04	0,74	0,75	**0,79**
13	0,50	0,49	0,53	**0,62**
14	0,45	0,50	0,50	**0,66**
17	0,46	0,42	0,45	**0,49**

Analisando os resultados apresentados na Tabela 6 é possível notar que, de maneira geral, a medida que o valor do parâmetro f aumenta a eficácia do algoritmo também aumenta. Por exemplo, para o conjunto de dados 4 o aumento de eficácia variou de $3,75$ a $8,25$ vezes quando comparado ao resultado obtido com a execução do *SSHub Clustering* com $f = 0$. Para o conjunto de dados 11 o aumento de eficácia variou de $18,50$ a $19,75$ vezes, quando comparado ao resultado obtido com $f = 0$.

4.4 Considerações finais

Os experimentos apresentados neste capítulo mostram a eficácia e a eficiência do algoritmo desenvolvido e descrito nesta dissertação. A análise dos resultados obtidos permitiu mostrar que o algoritmo proposto possui grande potencial para lidar com diferentes quantidades de instâncias, grupos e dimensões.

Na análise dos experimentos com a variação de parâmetros foi possível constatar que com um maior número de instâncias de fronteira analisadas é possível melhorar a qualidade dos particionamentos gerados. Além disso, quando o algoritmo considera uma vizinhança maior (por exemplo, $K = 20$), os *hubs* tornam-se mais representativos e não é necessário atualizar os representantes principais, pois a seleção inicial já é suficiente para a obtenção de bons resultados. O próximo capítulo apresenta as considerações finais desta dissertação revisando os objetivos do trabalho, além de considerações e propostas para trabalhos futuros.

Capítulo 5

Conclusão

O método de agrupamento proposto nesta dissertação combina o aspecto *hubness* com estratégias de semi-supervisão para permitir a obtenção de agrupamentos que condizem melhor com o particionamento esperado para dados de alta dimensão. É importante destacar que esta estratégia ainda não havia sido explorada na literatura científica da área. O algoritmo *SSHub Clustering* emprega uma abordagem de agrupamento semi-supervisionada, baseada em particionamento, na qual cada partição é representada por múltiplos protótipos, definidos com base em informações da pontuação *hubness* das instâncias de dados.

Primeiramente, estratégias foram desenvolvidas com o objetivo de permitir a interação do usuário na escolha dos representantes principais de cada grupo, considerando que instâncias com maiores pontuações *hubness* são mais representativas e podem impactar nas escolhas empregadas nas demais fases do algoritmo, principalmente na etapa de atribuição de instâncias aos grupos. Contudo, para que imprecisões associadas aos *hubs* não fossem propagadas, considerou-se a análise de elementos de fronteira dos grupos, para a criação de restrições *must-link* e *cannot-link* e, posteriormente, derivar representantes auxiliares. Para a fase de atribuição das instâncias aos grupos foi utilizada uma função de distância agregada que considera os múltiplos representantes (principal e auxiliares) de cada grupo, o que contribui para a detecção de agrupamentos de formas complexas.

Realizou-se um conjunto exaustivo de experimentos que utilizou 23 conjuntos de dados reais. O algoritmo *SSHub Clustering* foi comparado a quatro abordagens propostas na literatura que apresentam bons resultados no tratamento de dados de alta dimensão, são elas: *HPKM*, *Kernel k-means*, *DBSCAN* e *SSDBSCAN*. Os resultados obtidos demonstram que o método desenvolvido supera estatisticamente as abordagens comparadas. Além disso, foi mostrado que o algoritmo proposto possui grande potencial para lidar com diferentes quantidades de instâncias, grupos e dimensões. A seguir são apresentadas as principais contribuições alcançadas com essa pesquisa.

5.1 Principais contribuições

Os resultados dos experimentos realizados ajudam a corroborar a afirmação do bom desempenho do algoritmo *SSHub Clustering* no tratamento de dados de alta dimensão. Com isso, a seguir são apresentadas as principais contribuições alcançadas no desenvolvimento do trabalho:

- Criação de um novo algoritmo para o tratamento dados de alta dimensão, que considera todas as dimensões dos conjuntos de dados, isto é, não emprega técnicas de redução da dimensionalidade. Para tanto, considerou-se a combinação do aspecto *hubness* e técnicas de semi-supervisão. Os resultados obtidos apresentaram eficácia e eficiência computacional satisfatórias em relação aos algoritmos selecionados para comparação;

- Proposta de alternativa de algoritmo de agrupamento que tem a capacidade de trabalhar com conjuntos de dados de diferentes características, isto é, variadas quantidades de instâncias, grupos e dimensões, o que possibilita realizar processos de detecção de agrupamentos mais acurados tanto em conjuntos de dados de formas esféricas quanto de formatos diversos (SILVA et al., 2015);

5.2 Trabalhos futuros

Os resultados obtidos no trabalho descrito aqui mostraram o bom desempenho do novo método desenvolvido e motivam novas investigações para trabalhos futuros. Algumas novas possíveis linhas de investigação são listadas a seguir:

- Explorar outras estratégias para a seleção de elementos de fronteira na etapa de semi-supervisão. Na abordagem empregada, considerou-se somente a distância das instâncias para o representante principal para a seleção das instâncias de fronteira. Contudo, planeja-se utilizar métodos mais sofisticados que não considerem somente a distância;

- Trabalhar para melhorar a eficiência computacional do método para o cálculo das pontuações *hubness*. Sabe-se que o cálculo das pontuações *hubness* possui custo computacional considerável. Então, pode-se tomar como base os trabalhos descritos na Seção 2.2.2 e empregar soluções para melhor a eficiência e utilizar o aspecto *hubness* em grandes conjuntos de dados;

- Considerar técnicas para não violar as restrições *must-link* e *cannot-link*. A abordagem empregada não considerou o tratamento de restrições que se contradizem. Por exemplo, caso a instância x_i possua uma restrição *must-link* e outra *cannot-link* com representantes de um mesmo grupo C_k. Então, durante o processo de

agrupamento pode-se não utilizar da melhor forma a informação adicional disponível, considerando de forma equivocada as restrições e resultar um particionamento inadequado.

5.3 Contribuições em produção bibliográfica

A pesquisa realizada durante esta dissertação gerou a publicação de um artigo em conferência e a submissão de um artigo para periódico. A seguir são apresentados detalhes dos dois artigos e dos respectivos meios de publicação:

- Artigo curto intitulado "Combinando semi-supervisão e *hubness* para aprimorar o agrupamento de dados em alta dimensão", apresentado no "Simpósio Brasileiro de Banco de Dados" (SBBD 2016). Neste artigo foram apresentados os resultados preliminares da primeira versão do algoritmo *SSHub Clustering*. Para a categoria de artigo curto foram 55 artigos submetidos e 25 aceitos. Na ocasião, o artigo foi agraciado com a premiação de melhor *short paper* do evento;

- Artigo intitulado "*Combining semi-supervision and hubness to enhance high-dimensional data clustering*", submetido para o periódico *"Journal of Information and Data Management"* (JIDM). Este artigo integra os conceitos introduzidos no artigo anterior e vai além apresentando: uma descrição mais detalhada de uma versão aprimorada do algoritmo proposto e os resultados obtidos com um conjunto exaustivo de experimentos que utilizou 23 conjuntos de dados reais.

Referências

AGGARWAL, C. C.; REDDY, C. K. **Data Clustering: Algorithms and Applications**. 1st. ed. [S.l.]: Chapman & Hall/CRC, 2013.

AGGARWAL, C. C.; YU, P. S. Finding generalized projected clusters in high dimensional spaces. In: **ACM International Conference on Management of Data**. Dallas, Texas: [s.n.], 2000. p. 70–81.

AHMED, E. B.; NABLI, A.; GARGOURI, F. Shacun: semi-supervised hierarchical active clustering based on ranking constraints. In: **Industrial Conference on Advances in Data Mining**. Berlin, Germany: [s.n.], 2012. p. 194–208.

AUCOUTURIER, J.-J.; F., P. Improving timbre similarity: How high is the sky? **Journal of Negative Results in Speech and Audio Sciences**, v. 1, n. 1, 2004.

BARIONI, M. et al. Open issues for partitioning clustering methods: An overview. **Wiley Int. Rev. Data Mining and Knowledge Discovery**, v. 4, n. 3, p. 161–177, 2014.

BASU, S.; BANERJEE, A.; MOONEY, R. J. Semi-supervised clustering by seeding. In: **International Conference on Machine Learning**. Sydney, Australia: [s.n.], 2002. p. 27–34.

BASU, S.; DAVIDSON, I.; WAGSTAFF, K. **Constrained Clustering: Advances in Algorithms, Theory, and Applications**. 1. ed. [S.l.]: Chapman & Hall/CRC, 2008.

BORG, I.; GROENEN, P. Modern multidimensional scaling: Theory and applications. In: . [S.l.]: Springer, 2005.

BUZA, K. Semi-supervised naive hubness bayesian k-nearest neighbor for gene expression data. In: **International Conference on Computer Recognition Systems**. Wroclaw, Poland: [s.n.], 2016. p. 101–110.

CHAPELLE, O.; SCHLKOPF, B.; ZIEN, A. **Semi-Supervised Learning**. 1. ed. [S.l.]: The MIT Press, 2010.

CHEN, J.; SAAD, Y.; DASGUPTA, S. Fast approximate knn graph construction for high dimensional data via recursive lanczos bisection. **Jornal of Machine Learning Research**, v. 10, n. 1, p. 1989–2012, 2009.

COHN, D.; CARUANA, R.; MCCALLUM, A. Semi-supervised clustering with user feedback. **Constrained Clustering: Advances in Algorithms, Theory, and Applications**, v. 4, n. 1, p. 17–32, 2003.

COLLINS, J.; OKADA, K. A comparative study of similarity measures for content-based medical image retrieval. In: **Conference and Labs of the Evaluation Forum**. Rome, Italy: [s.n.], 2012.

DATABASESPORTS.COM. **Database basketball**. 2011. <www.databasebasketball.com>. Accessed: March, 2016.

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

DEMSAR, J. Statistical comparisons of classifiers over multiple data sets. **Journal of Machine Learning Research**, v. 7, n. 1, p. 1–30, 2006.

DHILLON, I. S.; GUAN, Y.; KULIS, B. Kernel k-means: Spectral clustering and normalized cuts. In: **International Conference on Knowledge Discovery and Data Mining**. Seattle, WA: [s.n.], 2004. p. 551–556.

DHILLON, I. S.; GUAN, Y.; KULIS, B. Weighted graph cuts without eigenvectors a multilevel approach. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 29, n. 11, p. 1944–1957, 2007.

DUBEY, A.; BHATTACHARYA, I.; GODBOLE, S. A cluster-level semi-supervision model for interactive clustering. In: **European Conference on Machine Learning and Knowledge Discovery in Databases**. Barcelona, Spain: [s.n.], 2010. p. 409–424.

FACELI, K.; LORENA, A. C.; GAMA, J. **;CARVALHO, A. Inteligência Artificial: Uma Abordagem de Aprendizado de Máquina**. 1. ed. [S.l.]: LTC, 2011.

FALOUTSOS, C.; LIN, K.-I. Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. **ACM SIGMOD Record Journal**, v. 24, n. 2, p. 163–174, 1995.

FLEXER, A.; SCHNITZER, D. Can shared nearest neighbors reduce hubness in high-dimensional spaces? In: **International Conference on Data Mining Workshops**. Dallas, Texas: [s.n.], 2013. p. 460–467.

FRIEDMAN, M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. **Journal of the American Statistical Association**, v. 32, n. 200, p. 675–701, 1937.

GAN, G.; MA, C.; WU, J. **Data Clustering: Theory, Algorithms, and Applications**. [S.l.]: Society for Industrial and Applied Mathematics, 2007.

HAIR JR., J. F. et al. **Multivariate Data Analysis: With Readings**. 4. ed. [S.l.]: Prentice-Hall, Inc., 1995.

HAN, J.; KAMBER, M. **Data Mining: Concepts and Techniques**. 2. ed. [S.l.]: Morgan Kaufmann Publishers Inc., 2006.

HICKLIN, B. U. A. The myth of goats: how many people have fingerprints that are hard to match? **Technical Report NISTIR 7271, National Institute of Standards and Technology**, 2005.

HU, T. et al. Pairwise constrained clustering with group similarity-based patterns. In: **International Conference on Machine Learning and Applications**. Washington, DC: [s.n.], 2010. p. 260–265.

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

IMAN, R. L.; DAVENPORT, J. M. Approximations of the critical region of the friedman statistic. **Communications in Statistics - Theory and Methods**, Taylor e Francis, v. 9, n. 1, p. 571–595, 1980.

JR., C. T.; TRAINA, A. J. M.; FALOUTSOS, C. Fast feature selection using fractal dimension - ten years later. **Journal of Information and Data Management**, v. 1, n. 1, p. 17–20, 2010.

KAILING, K.; KRIEGEL, P. K. H. Density-connected subspace clustering for high-dimensional data. In: **SIAM International Conference on Data Mining**. Lake Buena Vista, Florida: [s.n.], 2004. p. 246–257.

KAILING, K.; KRIEGEL, P. K. H.; WANKA, S. Ranking interesting subspaces for clustering high dimensional data. In: **European Conference on Principles and Practice of Knowledge Discovery in Databases**. Cavtat/Dubrovnik,Croatia: [s.n.], 2003. p. 241–252.

KIRA, K.; RENDELL, L. A. The feature selection problem: Traditional methods and a new algorithm. In: **National Conference on Artificial Intelligence**. San Jose, California: [s.n.], 1992. p. 129–134.

KULLBACK, S.; LEIBLER, R. A. On information and sufficiency. **Annals of Mathematical Statistics**, v. 22, n. 1, p. 79–86, 1951.

KUMAR, N.; KUMMAMURU, K. Semisupervised clustering with metric learning using relative comparisons. **IEEE Transactions on Knowledge and Data Engineering**, v. 20, n. 4, p. 496–503, 2008.

LELIS, L.; SANDER, J. Semi-supervised density-based clustering. In: **International Conference on Data Mining**. Miami, Florida: [s.n.], 2009. p. 842–847.

LI, J. et al. Active learning strategies for semi-supervised DBSCAN. In: **Canadian Conference on Artificial Intelligence**. Montreal, Canada: [s.n.], 2014. p. 179–190.

LICHMAN, M. **UCI Machine Learning Repository**. 2013. <http://archive.ics.uci.edu/ml>. Accessed: September, 2016.

MACQUEEN, J. Some methods for classification and analysis of multivariate observations. In: **Berkeley Symposium on Mathematical Statistics and Probability**. Berkeley, CA: [s.n.], 1967. p. 281–297.

MACQUEEN, J. B. Some methods for classification and analysis of multivariate observations. In: **Symposium on Mathematical Statistics and Probability**. [S.l.: s.n.], 1967. p. 281–297.

MAIMON, O.; ROKACH, L. **Data Mining and Knowledge Discovery Handbook**. [S.l.]: Springer-Verlag New York, Inc., 2005.

MALLAPRAGADA, P. K.; JIN, R.; JAIN, A. K. Active query selection for semi-supervised clustering. In: **International Conferences on Pattern Recognition**. Florida, USA: [s.n.], 2008. p. 1–4.

MANNING, C. D.; RAGHAVAN, P.; SCHüTZE, H. **Introduction to Information Retrieval**. [S.l.]: Cambridge University Press, 2008.

MARQUES, G. C. **Machine Learning Techniques for Music Information Retrieval**. Tese (Doutorado) — Departamento de Informática, Universidade de Lisboa, Portugal, 2015.

MULLER, E. et al. Evaluating clustering in subspace projections of high dimensional data. **Very Large Data Base Endowment**, v. 2, n. 1, p. 1270–1281, 2009.

PETTIE, S.; RAMACHANDRAN, V. An optimal minimum spanning tree algorithm. **Journal of the ACM**, v. 49, n. 1, p. 16–34, 2002.

RADOVANOVIC, M.; NANOPOULOS, A.; IVANOVIC, M. Hubs in space: Popular nearest neighbors in high-dimensional data. **Journal of Machine Learning Research**, v. 11, p. 2487–2531, 2010.

RAND, W. Objective criteria for the evaluation of clustering methods. **Journal of the American Statistical Association**, v. 66, n. 336, p. 846–850, 1971.

SAAD, S. M.; KAMARUDIN, S. S. Comparative analysis of similarity measures for sentence level semantic measurement of text. In: **International Conference on Control System**. Penang, Malaysia: [s.n.], 2013. p. 90–94.

SAMET, H. **Foundations of Multidimensional and Metric Data Structures**. [S.l.]: Morgan Kaufmann, 2005.

SANDER, J. et al. Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications. **Data Mining and Knowledge Discovery**, v. 2, n. 2, p. 169–194, 1998.

SATULURI, V.; PARTHASARATHY, S. Bayesian locality sensitive hashing for fast similarity search. **Very Large Data Base Endowment**, v. 5, n. 5, p. 430–441, 2012.

SCHMIDT, J.; BRANDLE, E. M.; KRAMER, S. Clustering with attribute-level constraints. In: **International Conference on Data Mining**. Vancouver, Canada: [s.n.], 2011. p. 1206–1211.

SILVA, W. J. et al. Semi-supervised clustering using multi-assistant-prototypes to represent each cluster. In: **ACM Symposium on Applied Computing**. Salamanca, Spain: [s.n.], 2015. p. 831–836.

SILVESTRE, A. L. **Análise de Dados e Estatística Descritiva**. [S.l.]: Escolar Editora, 2007.

TAN, P.; STEINBACH, M.; KUMAR, V. **Introduction to Data Mining**. [S.l.]: Addison- Wesley, 2006.

TANIAR, D.; IWAN, L. H. **Exploring Advances in Interdisciplinary Data Mining and Analytics: New Trends**. 1. ed. [S.l.]: IGI Global, 2011.

TIBSHIRANI, R.; GUENTHER, W.; HASTIE., T. Estimating the number of clusters in a data set via the gap statistic. **Journal of the Royal Statistical Society Series B**, v. 63, n. 1, p. 411–423, 2001.

TOMASEV, N. **Hub Miner**. [S.l.]: GitHub, 2014. <https://github.com/datapoet/hubminer>. Accessed: November, 2014.

TOMASEV, N.; MLADENIC, D. Hub co-occurrence modeling for robust high-dimensional knn classification. In: **European Conference on Machine Learning and Knowledge Discovery in Databases**. Prague, Czech Republic: [s.n.], 2013. p. 643–659.

TOMASEV, N.; MLADENIC, D. Hubness-aware shared neighbor distances for high-dimensional k-nearest neighbor classification. **Knowledge and Information Systems**, v. 39, n. 1, p. 89–122, 2014.

TOMASEV, N. et al. The role of hubness in clustering high-dimensional data. In: **Pacific-Asia Conference on Knowledge Discovery and Data Mining**. Shenzhen, China: [s.n.], 2011. p. 183–195.

TOMASEV, N. et al. The role of hubness in clustering high-dimensional data. **IEEE Transactions on Knowledge and Data Engineering**, v. 26, n. 3, p. 739–751, 2014.

TOMASEV, N. et al. **Hubness-Based Clustering of High-Dimensional Data**. [S.l.: s.n.], 2015. 353–386 p.

VENDRAMIN, L.; CAMPELLO, R. J. G. B.; HRUSCHKA, E. R. Relative clustering validity criteria: A comparative overview. **Statistical Analysis and Data Mining**, v. 3, n. 4, p. 209–235, 2010.

VIEIRA, S. **Análise de variância: ANOVA**. [S.l.]: Atlas, 2006.

VU, V.-V.; LABROCHE, N.; BOUCHON-MEUNIER, B. Active learning for semi-supervised k-means clustering. In: **IEEE International Conference on Tools with Artificial Intelligence**. Arras, France: [s.n.], 2010. p. 12–15.

WAGSTAFF, K.; CARDIE, C. Clustering with instance-level constraints. In: **International Conference on Machine Learning**. Williamstown, MA: [s.n.], 2000. p. 1103–1110.

WANG, J.; MARKERT, K.; EVERINGHAM, M. Learning models for object recognition from natural language descriptions. In: **British Machine Vision Conference**. London, UK: [s.n.], 2009. p. 1–11.

WANG, J. T. et al. Metricmap: An embedding technique for processing distance-based queries in metric spaces. **IEEE Transactions on Systems, Man, and Cybernetics**, v. 35, n. 5, p. 973–987, 2005.

XIONG, S.; AZIMI, J.; FERN, X. Z. Active learning of constraints for semi-supervised clustering. **IEEE Transactions on Knowledge and Data Engineering**, v. 26, n. 1, p. 43–54, 2014.

XU, R.; WUNSCH II, D. Survey of clustering algorithms. **IEEE Transactions on Neural Networks**, v. 16, n. 3, p. 645–678, 2005.

YAGER, N.; DUNSTONE, T. The biometric menagerie. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 32, n. 2, p. 220–230, 2010.

ZAKI, M. J.; JR, W. M. **Data Mining and Analysis: Fundamental Concepts and Algorithms**. [S.l.]: Cambridge University Press, 2014.

ZAR, J. H. **Biostatistical Analysis**. 5. ed. [S.l.]: Prentice-Hall, Inc., 2007.

Apêndices

APÊNDICE A

Tabelas Estatísticas

As Tabelas 7 e 8 apresentam os valores críticos necessários para a validação estatística. Cada valor representa um nível de confiança para o teste, considerando também o grau de liberdade imposto pelo problema.

A.1 Valores críticos da *F-Distribution*

A Tabela 7 apresenta os valores críticos de acordo com a *F-Distribution* considerando $\alpha = 0,05$, isto é, com 95% de confiança. Mais informações sobre esses valores podem ser encontradas em (ZAR, 2007).

Tabela 7 – Valores críticos da *F-Distribution* para $\alpha = 0,05$.

$\alpha = 0,05$	$DF_1 = 1$	2	3	4	5	6	7	8	9	10
$DF_2 = 1$	161,4476	199,5	215,7073	224,5832	230,1619	233,986	236,7684	238,8827	240,5433	241,8817
2	18,51282	19	19,16429	19,24679	19,29641	19,32953	19,35322	19,37099	19,38483	19,3959
3	10,12796	9,552094	9,276628	9,117182	9,013455	8,940645	8,886743	8,845238	8,8123	8,785525
4	7,708647	6,944272	6,591382	6,388233	6,256057	6,163132	6,094211	6,041044	5,998779	5,964371
5	6,607891	5,786135	5,409451	5,192168	5,050329	4,950288	4,875872	4,81832	4,772466	4,735063
6	5,987378	5,143253	4,757063	4,533677	4,387374	4,283866	4,206658	4,146804	4,099016	4,059963
7	5,591448	4,737414	4,346831	4,120312	3,971523	3,865969	3,787044	3,725725	3,676675	3,636523
8	5,317655	4,45897	4,066181	3,837853	3,687499	3,58058	3,500464	3,438101	3,38813	3,347163
9	5,117355	4,256495	3,862548	3,633089	3,481659	3,373754	3,292746	3,229583	3,178893	3,13728
10	4,964603	4,102821	3,708265	3,47805	3,325835	3,217175	3,135465	3,071658	3,020383	2,978237
20	4,351244	3,492828	3,098391	2,866081	2,71089	2,598978	2,514011	2,447064	2,392814	2,347878
30	4,170877	3,31583	2,922277	2,689628	2,533555	2,420523	2,334344	2,266163	2,210697	2,16458
40	4,084746	3,231727	2,838745	2,605975	2,449466	2,335852	2,249024	2,18017	2,124029	2,077248
50	4,03431	3,18261	2,790008	2,557179	2,400409	2,286436	2,199202	2,129923	2,073351	2,026143
88	3,94932	3,100068	2,708186	2,475277	2,318052	2,203439	2,115471	2,045414	1,988047	1,940044

A.2 Valores críticos do teste *Bonferroni-Dunn*

A Tabela 8 descreve os valores críticos para o teste *post-hoc Bonferroni-Dunn*, que é aplicado após o teste de *Friedman*, considerando também o nível de confiança $\alpha = 0,05$. Os valores são baseados na *Studentized Range Statistic* divida por $\sqrt{2}$ (DEMSAR, 2006). O número de algoritmos considerado deve incluir o algoritmo de controle.

Tabela 8 – Valores críticos para o teste *Bonferroni-Dunn*.

#algoritmos	2	3	4	5	6	7	8	9	10
$\alpha = 0,05$	1,96	2,241	2,394	2,498	2,576	2,638	2,69	2,724	2,773