Pós-Graduação em Ciência da Computação

ALGORITMOS PARTICIONAIS SEMISSUPERVISIONADOS COM PONDERAÇÃO AUTOMÁTICA DE VARIÁVEIS

Por

Valmir Macario Filho

Tese de Doutorado

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
<www.cin.ufpe.br/~posgraduacao>

RECIFE

2015

Universidade Federal de Pernambuco
Centro de Informática
Pós-graduação em Ciência da Computação

Valmir Macario Filho

ALGORITMOS PARTICIONAIS SEMISSUPERVISIONADOS COM PONDERAÇÃO AUTOMÁTICA DE VARIÁVEIS

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Doutor em Ciência da Computação.

Orientador: Francisco de Assis Tenório de Carvalho

RECIFE
2015

Catalogação na fonte
Bibliotecária Jane Souto Maior, CRB4-571

M115a Macario Filho, Valmir
Algoritmos particionais semissupervisionados com ponderação automática de variáveis / Valmir Macario Filho. – Recife: O Autor, 2015.
165 f.: il., fig., tab.

Orientador: Francisco de Assis Tenório de Carvalho.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da computação, 2015.
Inclui referências e apêndices.

1. Ciência da computação. 2. Inteligência artificial. 3. Sistemas difusos. I. Carvalho, Francisco de Assis Tenório de (orientador). II. Título.

004 CDD (23. ed.) UFPE- MEI 2015-55

Tese de Doutorado apresentada por **Valmir Macario Filho** à Pós Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "**Algoritmos Particionais Semissupervisionados com Ponderação Automática de Variáveis"** orientada pelo **Prof. Francisco de Assis Tenório de Carvalho** e aprovada pela Banca Examinadora formada pelos professores:

_____

Prof. Adriano Lorena Inácio de Oliveira
Centro de Informática / UFPE

_____

Prof. George Darmiton da Cunha Cavalcanti
Centro de Informática / UFPE

_____

Prof. Carlos Alexandre Barros Mello
Centro de Informática / UFPE

_____

Profa. Anne Magaly de Paula Canuto
Departamento de Informática e Matemática Aplicada/UFRN

_____

Profa. Heloisa de Arruda Camargo
Departamento de Computação/UFSCAR

Visto e permitida a impressão.
Recife, 30 de janeiro de 2015.

_____

Profa. Edna Natividade da Silva Barros
Coordenadora da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

RESUMO

Nas aplicações tradicionais de aprendizagem de máquina, os classificadores utilizam apenas dados rotulados em seu treinamento. Os dados rotulados, por sua vez, são difíceis, caros, consomem tempo e requerem especialistas humanos para serem obtidos em algumas aplicações reais. Entretanto, dados não rotulados são abundantes e fáceis de serem obtidos mas há poucas abordagens que os utilizam no treinamento. Para contornar esse problema existe a aprendizagem semissupervisionada. A aprendizagem semissupervisionada utiliza dados não rotulados, juntamente com dados rotulados, com a finalidade de melhorar o desempenho dos algoritmos. A abordagem semissupervisionada, geralmente, obtém resultados melhores do que se utilizassem apenas poucos padrões rotulados em uma abordagem supervisionada ou se utilizassem apenas padrões não rotulados numa abordagem não supervisionada. Um algoritmo semissupervisionado pode se basear em algoritmos de agrupamento não supervisionado, geralmente, adicionando-se um termo ou estratégia que faz uso de informações rotuladas para guiar o processo de aprendizagem deste algoritmo. Os algoritmos de agrupamento são bastante influenciados pelo cálculo da similaridade entre dois items, ou seja, a distância entre dois itens. Quando o algoritmo semissupervisionado é um extensão de um algoritmo de agrupamento, este também é bastante influenciado por esta distância. Desse modo, distâncias adaptativas são utilizadas para que o algoritmo tenha capacidade de se adequar a diferentes distribuições dos dados, geralmente, melhorando o desempenho em relação aos algoritmos que não utilizam uma distância adaptativa. Este trabalho apresenta novos algoritmos de agrupamento semissupervisionado baseados no algoritmo *Fuzzy C-Means* que utilizam distâncias adaptativas com ponderação automática de variáveis. Estudos experimentais no contexto da aprendizagem a partir de dados parcialmente rotulados são apresentados. Além disso, o comportamento dos algoritmos é discutido e os resultados examinados através de testes estatísticos de Friedman. Desse modo, foi possível certificar que os novos algoritmos de agrupamento semissupervisionado com distâncias adaptativas apresentam desempenho melhor que algoritmos já consolidados na literatura.

Palavras-chave: Aprendizagem semissupervisionada. Agrupamento semissupervisionado. Agrupamento Fuzzy. Função Objetivo. Distância Adaptativa. Distância Ponderada

ABSTRACT

In traditional machine learning applications, one uses only labeled data to train the algorithm. Labeled data are difficult, expensive, time consuming and require human experts to be obtained in some real applications. However, unlabeled data are abundant and easy to be obtained but there has been few approaches to use them in training. Semi-supervised learning addresses this problem. The semi-supervised learning uses large amount of unlabeled data, together with the labeled data, to build better algorithms. The semi-supervised approach, usually, obtains better results than if using a few labeled patterns in a supervised approach or using only standard not supervised approach. The semi-supervised algorithm can be an extension of an unsupervised algorithm. Such algorithm can be based on unsupervised clustering algorithms, adding a term or strategy, that makes use of labeled information to guide the learning process of the algorithm. Unsupervised algorithms are greatly influenced by calculating the similarity between two items, ie, the distance between two items. When the semi-supervised algorithm is an extension of an unsupervised algorithm, it is also quite influenced by this distance. Thus, adaptive distances are utilized, so the algorithm is able to adapt to different data distributions, generally improving performance compared to algorithms that use the standard Euclidean distance. This work presents new algorithms for semi-supervised clustering based on Fuzzy C-Means algorithm using adaptive distances with automatic variable weighting. Experimental studies in the context of learning from partially labeled data are presented. In addition, the behavior of the algorithm is discussed and the results are investigated using the Friedman Test. Thus, it was possible to certify that the performance of the new semi-supervised clustering algorithms are better than other consolidated algorithms.

Keywords: Semi-Supervised Learning. Semi-Supervised Clustering. Fuzzy Clustering. Objective Function. Adaptive Distance. Weighted Distance

LISTA DE FIGURAS

Figura 1 – Agrupamentos em duas dimensões 20
Figura 2 – Grupos *versus* classes 22
Figura 3 – *Fuzzy Versus Hard* 23
Figura 4 – Aprendizagem semissupervisionada 38
Figura 5 – Aprendizagem semissupervisionada 38
Figura 6 – Conjunto de dados simulados 1. 88
Figura 7 – Conjunto de dados simulados 2. 89
Figura 8 – *Ranking* de Friedman dos algoritmos BSSC 113
Figura 9 – *Ranking* de Friedman dos algoritmos sSSC 114
Figura 10 – Ranking de Friedman dos algoritmos ESSC 115
Figura 11 – *Ranking* de Friedman dos algoritmos NebFuzz 117
Figura 12 – *Ranking* de Friedman dos algoritmos Isodata-PS 118
Figura 13 – *Ranking* de Friedman para 1% de dados rotulados 121
Figura 14 – *Ranking* de Friedman para 10% de dados rotulados 122
Figura 15 – *Ranking* de Friedman para 30% de dados rotulados 123
Figura 16 – *Ranking* de Friedman para 50% de dados rotulados 124
Figura 17 – Base de dados simulados 1 com 1% de dados rotulados 127
Figura 18 – Base de dados simulados 1 com 10% de dados rotulados 128
Figura 19 – Base de dados simulados 1 com 30% de dados rotulados 129
Figura 20 – Base de dados simulados 1 com 50% de dados rotulados 130
Figura 21 – Base de dados simulados 2 com 1% de dados rotulados 132
Figura 22 – Base de dados simulados 2 com 10% de dados rotulados 133
Figura 23 – Base de dados simulados 2 com 30% de dados rotulados 134
Figura 24 – Base de dados simulados 2 com 50% de dados rotulados 135
Figura 25 – Desempenho na base de dados simulados 1 com 1% de dados rotulados 158
Figura 26 – Desempenho na base de dados simulados 1 com 1% de dados rotulados 159
Figura 27 – Desempenho na base de dados simulados 1 com 30% de dados rotulados 160
Figura 28 – Desempenho na base de dados simulados 1 com 50% de dados rotulados 161
Figura 29 – Desempenho na base de dados simulados 2 com 1% de dados rotulados 162
Figura 30 – Desempenho na base de dados simulados 2 com 1% de dados rotulados 163
Figura 31 – Desempenho na base de dados simulados 2 com 30% de dados rotulados 164
Figura 32 – Desempenho na base de dados simulados 2 com 50% de dados rotulados 165

LISTA DE TABELAS

Tabela 1 – Resumo dos algoritmos de agrupamento semissupervisionado. 17
Tabela 2 – Exemplos no formato atributo-valor 21
Tabela 3 – Tabela de contingência para duas partições 32
Tabela 4 – Grau de pertinência antes da otimização 43
Tabela 5 – Grau de pertinência após otimização 43
Tabela 6 – Grau de pertinência antes da otimização 45
Tabela 7 – Grau de pertinência após otimização 45
Tabela 8 – Grau de pertinência antes da otimização 47
Tabela 9 – Grau de pertinência após otimização 47
Tabela 10 – Grau de pertinência antes da otimização 49
Tabela 11 – Grau de pertinência após otimização 49
Tabela 12 – Símbolo que representa cada algoritmo semissupervisionado. 49
Tabela 13 – Equações dos algoritmos semissupervisionados. 50
Tabela 14 – Complexidade de tempo. 54
Tabela 15 – Símbolo que representa cada algoritmo semissupervisionado. 75
Tabela 16 – Equações dos algoritmos semissupervisionados com distância adaptativa 77
Tabela 17 – Complexidade de Tempo. 81
Tabela 18 – Resumo das bases de dados reais retiradas do repositório UCI. 85
Tabela 19 – Configuração do conjunto de dados simulados 1. 88
Tabela 20 – Configuração do conjunto de dados simulados 2. 90
Tabela 21 – Exemplo de cálculo do ponto de fronteira 93
Tabela 22 – Construção do *ranking* de Friedman 94
Tabela 23 – Comparação entre algoritmos com diferentes alfa 96
Tabela 24 – *Ranking* de Friedman para valores de alfa do algoritmo BSSC 98
Tabela 25 – *Ranking* de Friedman para valores de alfa do algoritmo BSSC-GP . . . 99
Tabela 26 – *Ranking* de Friedman para valores de alfa do algoritmo BSSC-GS . . . 99
Tabela 27 – *Ranking* de Friedman para valores de alfa do algoritmo BSSC-LP . . . 100
Tabela 28 – *Ranking* de Friedman para valores de alfa do algoritmo BSSC-LS . . . 100
Tabela 29 – *Ranking* de Friedman para valores de alfa do algoritmo ESSC 101
Tabela 30 – *Ranking* de Friedman para valores de alfa do algoritmo ESSC-GP . . . 101
Tabela 31 – *Ranking* de Friedman para valores de alfa do algoritmo ESSC-LP . . . 102
Tabela 32 – *Ranking* de Friedman para valores de alfa do algoritmo ESSC-GS . . . 103
Tabela 33 – *Ranking* de Friedman para valores de alfa do algoritmo NebFuzz 103
Tabela 34 – *Ranking* de Friedman para valores de alfa do algoritmo NebFuzz-GP . 104
Tabela 35 – *Ranking* de Friedman para valores de alfa do algoritmo NebFuzz-LP . . 104
Tabela 36 – *Ranking* de Friedman para valores de alfa do algoritmo NebFuzz-GS . . 105

Tabela 37 – *Ranking* de Friedman para valores de alfa do algoritmo NebFuzz-LS . . 105
Tabela 38 – *Ranking* de Friedman para valores de alfa do algoritmo Isodata-PS . . 106
Tabela 39 – *Ranking* de Friedman para valores de alfa do algoritmo Isodata-PS-GP 107
Tabela 40 – *Ranking* de Friedman para valores de alfa do algoritmo Isodata-PS-LP 107
Tabela 41 – *Ranking* de Friedman para valores de alfa do algoritmo Isodata-PS-GS 108
Tabela 42 – *Ranking* de Friedman para valores de alfa do algoritmo Isodata-PS-LS 108
Tabela 43 – Valores de alfa selecionados para bases de dados reais 111
Tabela 44 – Teste de Friedman dos algoritmos da família BSSC 112
Tabela 45 – Teste de Friedman dos algoritmos da família sSSC 115
Tabela 46 – Teste de Friedman dos algoritmos da família ESSC 116
Tabela 47 – Teste de Friedman dos algoritmos da família NebFuzz 117
Tabela 48 – Teste de Friedman dos algoritmos da família Isodata-PS 119
Tabela 49 – Melhor e pior algoritmo por porcentagem 123
Tabela 50 – Teste de Wilcoxon para seleção de rótulos 125
Tabela 51 – Valores de alfa selecionados para bases de dados simulados 1 126
Tabela 52 – Valores de alfa selecionados para bases de dados simulados 2 131
Tabela 53 – Pesos das bases de dados simulados 1 136
Tabela 54 – Pesos das bases de dados simulados 2 136
Tabela 55 – Resumo dos algoritmos de agrupamento semissupervisionado. 138
Tabela 56 – Desempenho médio dos algoritmos BSSC com 10% de rotulados 148
Tabela 57 – Desempenho médio dos algoritmos BSSC com 50% de rotulados 149
Tabela 58 – Desempenho médio dos algoritmos sSSC com 10% de rotulados 150
Tabela 59 – Desempenho médio dos algoritmos sSSC com 50% de rotulados 151
Tabela 60 – Desempenho médio dos algoritmos ESSC com 10% de rotulados 152
Tabela 61 – Desempenho médio dos algoritmos ESSC com 50% de rotulados 153
Tabela 62 – Desempenho médio dos algoritmos NebFuzz com 10% de rotulados . . 154
Tabela 63 – Desempenho médio dos algoritmos NebFuzz com 50% de rotulados . . 155
Tabela 64 – Desempenho médio dos algoritmos Isodata-PS com 10% de rotulados . 156
Tabela 65 – Desempenho médio dos algoritmos Isodata-PS com 50% de rotulados . 157

SUMÁRIO

1 INTRODUÇÃO 12
1.1 Motivação 13
1.2 Objetivos 16
1.3 Estrutura da tese 17

2 AGRUPAMENTO E DISTÂNCIAS ADAPTATIVAS 19
2.1 Agrupamento 19
2.1.1 Representação de objetos do mundo real 20
2.1.2 Distâncias e similaridades 21
2.1.3 Centróide, partição, classe e grupo 22
2.1.4 Agrupamento *hard* e agrupamento *fuzzy* 22
2.1.5 Processo de agrupamento 23
2.2 Algoritmo de agrupamento *Fuzzy C-Means* 24
2.3 Agrupamento com distâncias adaptativas 25
2.4 Algoritmos de agrupamento com distância adaptativa 28
2.4.1 Algoritmo GK 28
2.4.2 FCM-DFCV 30
2.5 Medidas de avaliação de algoritmos de agrupamento 31
2.5.1 Índice de Rand corrigido 31
2.5.2 Índice de informação mútua normalizado 32
2.5.3 Taxa de acerto de classificação 33
2.6 Propriedades da inicialização de algoritmos de agrupamento . . . 33
2.6.1 Normalização da base de dados 33
2.6.2 Inicialização da matriz de protótipos 34
2.6.3 Inicialização da matriz de grau de pertinência 34
2.6.4 Critério de parada 35
2.7 Síntese do capítulo 36

3 AGRUPAMENTO SEMISSUPERVISIONADO 37
3.1 Aprendizagem semissupervisionada 37
3.1.1 Agrupamento semissupervisionado baseados em função objetivo 37
3.2 Algoritmos de agrupamento semissupervisionado 41
3.2.1 Algoritmo Isodata-PS 41
3.2.2 Algoritmo BSSC 43
3.2.3 Algoritmo semissupervisionado padrão (sSSC) 45
3.2.4 Algoritmo semissupervisionado por entropia (ESSC) 46

3.2.5 Algoritmo NebFuzz . . . 48
3.2.6 Etapas dos algoritmos . . . 49
3.2.7 Propriedades de convergência . . . 51
3.2.8 Custo computacional . . . 53
3.2.9 Resumo dos algoritmos de agrupamento semissupervisionados 54

4 SEMISSUPERVISIONADOS COM DISTÂNCIAS ADAPTATIVAS 55
4.1 Algoritmos semissupervisionados com distâncias adaptativas . . . 55
4.1.1 NebFuzz com distâncias adaptativas . . . 56
4.1.2 BSSC com distâncias adaptativas . . . 64
4.1.3 ISODATA-PS com Distâncias Adaptativas . . . 68
4.1.4 sSSC com distâncias adaptativas . . . 70
4.1.5 ESSC com distâncias adaptativas . . . 73
4.1.6 Etapas dos algoritmos . . . 75
4.1.7 Propriedades de convergência . . . 76
4.1.8 Custo computacional . . . 80
4.2 Resumo . . . 81
4.3 Considerações finais . . . 82

5 VALIDAÇÃO EXPERIMENTAL . . . 84
5.1 Base de dados . . . 84
5.1.1 Bases de dados reais . . . 84
5.1.2 Conjuntos de dados simulados . . . 87
5.1.2.1 Conjunto de dados simulados 1 . . . 87
5.1.2.2 Conjunto de dados simulados 2 . . . 87
5.2 Experimentos . . . 89
5.2.1 Agrupamento semissupervisionado . . . 90
5.2.2 Experimentos com bases de dados reais . . . 91
5.2.2.1 Análise do parâmetro alfa (α) . . . 91
5.2.2.2 Análise da ponderação automática de variáveis . . . 92
5.2.2.3 Comparação entre algoritmos semissupervisionados . . . 92
5.2.2.4 Seleção de rótulos . . . 92
5.2.3 Experimentos com bases de dados simuladas . . . 93
5.2.4 Teste de Friedman . . . 94
5.2.5 Teste de Wilcoxon . . . 95
5.3 Síntese do capítulo . . . 96

6 RESULTADOS E DISCUSSÃO . . . 97
6.1 Bases de dados reais . . . 97
6.1.1 Análise do parâmetro alfa (α) . . . 97

6.1.1.1 Discussão . . . 109
6.1.2 Comparação dos algoritmos com distância adaptativa . . . 111
6.1.2.1 BSSC com distâncias adaptativas . . . 112
6.1.2.2 sSSC com distância adaptativa . . . 113
6.1.2.3 ESSC com distância adaptativa . . . 114
6.1.2.4 NebFuzz com distância adaptativa . . . 116
6.1.2.5 Isodata-PS com distância adaptativa . . . 116
6.1.2.6 Discussão . . . 119
6.1.3 Algoritmos de agrupamento semissupervisionado . . . 120
6.1.3.1 Discussão . . . 122
6.1.4 Análise dos rótulos . . . 124
6.2 Bases de dados simulados . . . 125
6.2.1 Base de dados simulados 1 . . . 126
6.2.2 Base de dados simulados 2 . . . 130
6.2.3 Discussão . . . 132
6.3 Síntese do capítulo . . . 137

7 CONCLUSÕES E TRABALHOS FUTUROS . . . 138
7.1 Conclusão . . . 138
7.2 Contribuições . . . 141
7.3 Trabalhos futuros . . . 142

REFERÊNCIAS . . . 143

APÊNDICES 147

APÊNDICE A – RESULTADOS . . . 148

APÊNDICE B – DADOS SIMULADOS . . . 158

1 INTRODUÇÃO

Na medida em que a tecnologia evolui e que mais pessoas têm acesso à mesma, altera-se continuamente a forma de interação, busca e disponibilização da informação. Assim, torna-se evidente a necessidade de se construir ferramentas e técnicas capazes de ajudar as pessoas a extrairem conhecimento de forma automática e inteligente dessa enorme quantidade de informação armazenada.

Uma das técnicas amplamente empregadas na tarefa de extração do conhecimento é a aprendizagem de máquina (MITCHELL, 1997). A aprendizagem de máquina constrói modelos computacionais que aprendem a extrair conhecimento a partir da análise de dados. Dois paradigmas clássicos de aprendizagem são a aprendizagem supervisionada e a aprendizagem não supervisionada.

Na aprendizagem supervisionada, um conjunto de exemplos cujas classes são conhecidas é apresentado ao algoritmo de classificação. O objetivo é aprender uma função que mapeia o exemplo e_i ao rótulo h_i a partir do conjunto de treinamento que contém pares (e_i, h_i), onde o $h_i \in H$ é o rótulo associado ao exemplo e_i. O objetivo é predizer o rótulo h_i dos exemplos que não possuem seus rótulos conhecidos. Existem duas famílias de algoritmos supervisionados (CHAPELLE; ZIEN; SCHOLKOPF, 2006): (i) algoritmos generativos, que tentam modelar uma densidade condicional por algum procedimento de aprendizagem não supervisionado e (ii) algoritmos discriminativos que determinam o valor da probabilidade de se encontrar e_i numa determinada classe e não em saber como a distribuição da classe foi gerada, como no método anterior.

Na aprendizagem não supervisionada, os dados apresentados ao algoritmo são não rotulados. O objetivo é aprender a estrutura dos dados apresentados. Esses algoritmos procuram aprender a distribuição dos dados de acordo com alguma medida de similaridade. Portanto, os padrões mais similares entre si são dispostos num mesmo grupo. Pela distribuição dos dados em grupos, esse tipo de aprendizagem é usualmente associada aos algoritmos de agrupamento. Um algoritmo de agrupamento distribui os dados de modo que os exemplos mais similares são colocados num mesmo grupo e os exemplos não similares são colocados em grupos diferentes com uma boa separabilidade dos dados para refletir a distribuição inerente dos dados mais fielmente possível (JAIN; MURTY; FLYNN, 1999; DUDA; HART; STORK, 2001; XU e WUNSCH II, 2005; JAIN, 2010).

Além dos paradigmas de aprendizagem já citados, existe outro paradigma chamado aprendizagem semissupervisionada. Nele, os exemplos utilizados podem ter rótulos conhecidos, como na aprendizagem supervisionada e também exemplos com rótulos desconhecidos, como na aprendizagem não supervisionada. Esses algoritmos podem ser uma

adaptação de um algoritmo supervisionado, que vai levar em conta exemplos não rotulados na aprendizagem. E também podem ser uma adaptação de um algoritmo de agrupamento não supervisionado, que vai levar em conta exemplos rotulados em sua aprendizagem. Neste trabalho é dada ênfase aos algoritmos de agrupamento semissupervisionados (CHAPELLE; ZIEN; SCHOLKOPF, 2006).

Nos últimos anos, um número significativo de métodos para agrupamentos, assim como várias aplicações de sucesso têm sido desenvolvidas (JAIN, 2010). Métodos de agrupamento podem ser divididos em duas categorias principais: hierárquicos e particionais. Métodos hierárquicos produzem uma hierarquia completa, ou seja, uma sequência aninhada de partições dos dados de entrada, enquanto que os métodos de particionamento procuram obter uma única partição dos dados de entrada em um número fixo de grupos, geralmente por meio da otimização de uma função objetivo.

Os métodos particionais podem ser divididos em agrupamento *hard* e *fuzzy*. No agrupamento *hard*, cada objeto é atribuído a apenas um grupo. No agrupamento *fuzzy*, a função objetivo fornece um aprimoramento conceitual, permitindo que um objeto seja associado a todos os grupos de acordo com um grau de pertinência. Isto pode ser útil quando existe uma sobreposição entre os grupos.

Em geral, os algoritmos de agrupamento *fuzzy* baseados em função objetivo, particionam um conjunto de dados em grupos, minimizando a função objetivo derivada da distância entre os protótipos do grupo e os exemplos (ou objetos) representadas pela matriz de características dos dados. Dessa forma, a maneira como a distância é calculada tem grande influência nos resultados dos agrupamentos, uma vez que o tipo de distância implementada determina o formato dos grupos. Como consequência, há uma quantidade significativa de literatura em agrupamento *fuzzy*, que tem por objetivo investigar o uso de diferentes funções de distância em agrupamentos capazes de lidar com dados cuja estrutura é complexa. Dentre estes, encontram-se os algoritmos de agrupamento baseados em distâncias adaptativas (GUSTAFSON e KESSEL, 1979). Esta tese está inserida no contexto de algoritmos de agrupamento semissupervisionados com distâncias adaptativas.

1.1 MOTIVAÇÃO

A escolha das abordagens clássicas entre as aprendizagens supervisionadas e não supervisionadas se baseia principalmente no aspecto da quantidade de informação existente sobre o problema em questão. Escolhe-se a primeira se houver informações descritivas acerca dos dados, ou seja, rótulos que indiquem a classe de um padrão pertencente a base de dados. Assim, os dados rotulados, que serão parte do conjunto de treinamento, são utilizados para treinar o algoritmo. A aprendizagem não supervisionada é escolhida quando não há informações de rótulo dos dados. Os dados são agrupados de acordo com algum critério que aumente a homogeneidade dos exemplos num mesmo grupo e aumente

a heterogeneidade entre exemplos de grupos diferentes. Existem diversas técnicas de agrupamento (JAIN, 2010). Algumas dessas técnicas realizam a otimização de uma função objetivo. O processo de agrupamento é definido pela otimização dessa função objetivo.

A aprendizagem não supervisionada sofre com algumas limitações significativas. Diferente do que acontece no processo supervisionado, os resultados de um aprendizado não supervisionado são apenas os grupos ou partições, sem informações descritivas acerca dessas partições geradas. Porém, algumas vezes, os usuários não estão interessados apenas nessas partições, mas também em alguma explicação do que essas partições representam. Para superar essa limitação é necessária uma interpretação das partições encontradas.

Pode-se então pensar na alternativa de realizar uma marcação manual dos dados para utilização de algoritmos supervisionados. Entretanto, essa tarefa pode ser extremamente complexa, cara, demorada e requerer especialistas humanos em algumas aplicações reais. Pode-se citar, como exemplo de aplicação, a extração de processos em diários oficiais. Essa tarefa requer que cada linha de um diário oficial seja rotulada, sendo que a publicação de apenas um dia desse documento pode conter mais de 100000 linhas (MACARIO et al., 2008). Também, há os casos de *Spam*, onde o classificador é obrigado a aprender os e-mails não desejados a partir de uma porcentagem pequena de e-mails marcados (CHENG e LI, 2006). Ainda podemos citar aplicações na mineração de textos (NIGAM et al., 2000) ou ainda aplicações em processamento de imagens (BENSAID et al., 1996). Desse modo, a utilização de algoritmos supervisionados é inviável, pois existem poucos dados rotulados disponíveis para o treinamento desse tipo de algoritmo. Tais fatores sugerem uma técnica que utilize exemplos não rotulados para aumentar a precisão de um algoritmo.

A aprendizagem semissupervisionada (CHAPELLE; ZIEN; SCHOLKOPF, 2006; ZHU, 2008) é uma abordagem intermediária entre a aprendizagem supervisionada e a aprendizagem não supervisionada. Na aprendizagem semissupervisionada são utilizados exemplos rotulados e não rotulados para guiar a aprendizagem de um algoritmo com a finalidade de construir algoritmos com melhor desempenho. Os algoritmos de agrupamento não supervisionados podem ser modificados para que seu aprendizado também considere dados rotulados, se transformando em um algoritmo semissupervisionado. A melhoria no processo de agrupamento pode ser conseguida acrescentando qualquer informação externa suplementar disponível para orientar o processo de agrupamento (JAIN, 2010). Assim, a abordagem de agrupamento semissupervisionado visa utilizar alguns dados rotulados (PEDRYCZ e WALETZKY, 1997; MACARIO e DE CARVALHO, 2010) ou algumas restrições (GRIRA; CRUCIANU; BOUJEMAA, 2008) (*must-link* e *cannot-link*) para melhorar o desempenho de algoritmos de agrupamento não supervisionado. No caso de dados rotulados, um especialista faz a rotulação de alguns exemplos da base de dados. Nas restrições, o especialista seleciona exemplos que devem ser colocados no mesmo grupo (*must-link*) e exemplos que não devem ser colocados no mesmo grupo (*cannot-link*).

Outra modificação que pode ser realizada para melhorar o desempenho de algoritmos de agrupamento leva em conta a natureza desses algoritmos, pois os resultados deste tipo de algoritmo são largamente influenciados pela forma como a similaridade entre dois exemplos é calculada através da medida de dissimilaridade (ou de similaridade). Medidas de distância são exemplos básicos de medidas de dissimilaridade e a distância Euclidiana é a mais comumente utilizada em métodos de agrupamento particionais. Métodos de agrupamento baseados na distância Euclidiana apresentam bom desempenho quando aplicados a conjuntos de dados nos quais os grupos são esféricos. Contudo, quando a estrutura dos dados é complexa, *i.e.*, grupos com formas não hiperesféricas e/ou problemas não-linearmente separáveis, essa distância pode não apresentar desempenho satisfatório. Por causa dessa limitação, diversos métodos baseados em distâncias adaptativas têm sido propostos (GUSTAFSON e KESSEL, 1979; FRIGUI e HWANG, 2008). Estes algoritmos são capazes de construir diferentes formas de agrupamentos com o objetivo de refletir de forma mais fiel a real distribuição do conjunto de dados.

Em análise de agrupamentos, os padrões a serem agrupados são geralmente representados como vetores nos quais cada componente é uma medição referente a uma variável. Os algoritmos de agrupamento convencionais (*e.g. Fuzzy C-Means*, etc.) consideram que todas as variáveis são igualmente importantes, no sentido de que todas possuem o mesmo peso para a definição dos grupos. Não obstante, na maioria das áreas do conhecimento, e, especialmente se estiver lidando com conjuntos de dados de alta dimensionalidade, algumas variáveis podem ser irrelevantes. Além disso, dentre aquelas que são relevantes, algumas podem ter uma importância maior do que outras na construção dos grupos. Ainda, a contribuição de cada variável para cada grupo pode ser diferente, *i.e.*, cada grupo pode ter um conjunto diferente de variáveis importantes. Diversas modificações do algoritmo *Fuzzy C-Means* têm sido propostas na literatura para aprender automaticamente os pesos das variáveis e melhorar seu desempenho. Na abordagem em que todas as variáveis possuem o mesmo peso para todos os grupos, esta realiza ponderação global de variáveis. Para a abordagem onde cada grupo pode ter um conjunto diferente de variáveis importantes, esta realiza ponderação local de variáveis (GUSTAFSON e KESSEL, 1979).

Em ambas as abordagens, a derivação das expressões matemáticas para a obtenção dos pesos das variáveis pode ser feita considerando dois tipos de restrições: na primeira, assume-se que a soma dos pesos das variáveis deve ser igual a um, enquanto que na segunda, assume-se que o produto dos pesos das variáveis deve ser igual a um (GUSTAFSON e KESSEL, 1979; DE CARVALHO; TENORIO; CAVALCANTI JUNIOR, 2006).

1.2 OBJETIVOS

Esta tese de doutorado propõe novos métodos de agrupamento semissupervisionado com ponderação automática das variáveis através de distâncias adaptativas. As principais vantagens das abordagens propostas sobre os métodos de agrupamento não supervisionados convencionais é a utilização de dados parcialmente rotulados para guiar o processo de agrupamento e também a possibilidade do uso de distâncias adaptativas, as quais mudam a cada iteração do algoritmo e podem ser a mesma para todos os grupos (distâncias adaptativas globais) ou diferentes de um grupo para outro (distâncias adaptativas locais). Este tipo de medida de dissimilaridade é adequado ao aprendizado dos pesos das variáveis dinamicamente durante o processo de agrupamento, muitas vezes, levando a uma melhora no desempenho dos algoritmos. Ademais, todos os métodos propostos nesta tese, a derivação das expressões matemáticas utilizando o multiplicador de lagrange para a obtenção dos pesos das variáveis foi realizada considerando dois tipos de restrições. A primeira restrição determina que o produto do peso das variáveis deve ser igual a um e a segunda restrição determina que a soma dos pesos das variáveis deve ser igual a um. Os algoritmos propostos nesta tese adicionam um termo supervisionado na função objetivo original do algoritmo de agrupamento proposto por Gustafson e Kessel (GK) (GUSTAFSON e KESSEL, 1979). Foram modificados cinco algoritmos de agrupamento semissupervisionado para trabalhar com distâncias adaptativas. O primeiro algoritmo foi proposto por Pedrycz e Walesky (Isodata-PS) (PEDRYCZ e WALETZKY, 1997), o segundo (sSSC) e terceiro (ESSC) algoritmos foram propostos no trabalho de Endo et al. (ENDO et al., 2009), o quarto algoritmo foi proposto pelo autor desta tese e De Carvalho (Nebfuzz) (MACARIO e DE CARVALHO, 2010) e o último algoritmo foi proposto por Bouchachia e Pedrycz (BSSC) (BOUCHACHIA e PEDRYCZ, 2006a). A Tabela 1 faz um resumo dos algoritmos propostos neste trabalho; todos os algoritmos marcados com (✓) foram propostos neste trabalho. Contudo, apenas dois algoritmos não foram propostos neste trabalho, o primeiro algoritmo marcado com (x_1), foi proposto nos trabalhos de Marcelloni (MARCELLONI, 2003) e de Kong et al. (KONG e WANG, 2009); o segundo marcado com (x_2) foi proposto no trabalho de Bouchachia e Pedrycz (BOUCHACHIA e PEDRYCZ, 2006b). Além disso, foi utilizada uma abordagem para escolher exemplos de fronteira para serem rotulados, seguindo a abordagem realizada por Frigui em (FRIGUI e HWANG, 2008). Os exemplos de fronteira são mais difíceis de serem rotulados por algoritmos, então um especialista realiza este trabalho para que os algoritmos de agrupamento semissupervisionado se beneficiem desse conhecimento, melhorando seus desempenhos.

Tabela 1 – Resumo dos algoritmos de agrupamento semissupervisionado.

Algoritmo	Tipo de Distância Adaptativa			
	Local		Global	
	Soma (LS)	Produto (LP)	Soma (GP)	Produto (GS)
Isodata-PS	x_1	✓	✓	✓
BSSC	✓	x_2	✓	✓
sSSC	✓	✓	✓	✓
ESSC	✓	✓	✓	✓
NebFuzz	✓	✓	✓	✓

1.3 ESTRUTURA DA TESE

Esta tese está dividida em 7 capítulos no total. Além deste capítulo de Introdução, essa tese possui 6 capítulos organizados como segue:

Capítulo 2 - Agrupamento e distâncias adaptativas: este capítulo apresenta o conceito de algoritmos de agrupamento não supervisionado e descreve o algoritmo clássico *Fuzzy C-Means*. Depois, é descrito o conceito de distâncias adaptativas como medida de similaridade entre dois itens e logo após são apresentados dois algoritmos de agrupamento não supervisionado que utilizam distância adaptativa.

Capítulo 3 - Agrupamento semissupervisionado: o conceito de agrupamento semissupervisionado é apresentado neste capítulo. Depois alguns algoritmos de agrupamento semissupervisionado são descritos detalhadamente. Um estudo das propriedades de convergência e complexidade desses algoritmos é realizada no final do capítulo.

Capítulo 4 - Novos algoritmos de agrupamento semissupervisionado com distâncias adaptativas: os algoritmos propostos nesse trabalho são apresentados nesse capítulo. No final do capítulo, é realizado um estudo das propriedades de convergência e complexidade dos algoritmos propostos.

Capítulo 5 - Validação experimental: este capítulo descreve a metodologia dos experimentos. O ambiente, as técnicas utilizadas, bem como as configurações dos experimentos são explicadas. Além disso, são apresentadas todas as bases de dados utilizadas na validação deste trabalho.

Capítulo 6 - Resultados e discussão: Os resultados dos experimentos realizados neste trabalho que realizam a comparação dos algoritmos propostos com outros já descritos na literatura são apresentados neste capítulo. As análises dos experimentos são realizadas no final de cada Seção.

Capítulo 7 - Conclusões e trabalhos futuros: As conclusões deste trabalho, assim como perspectivas de pesquisas futuras a partir das ideias apresentadas nesta tese estão presentes neste capítulo.

Resultados adicionais do Capítulo 6 podem ser encontrados nos Apêndices A e B

2 AGRUPAMENTO E DISTÂNCIAS ADAPTATIVAS

Este capítulo abrange questões relevantes na abordagem de agrupamento. Uma pequena introdução à tarefa de agrupamento é apresentada na Seção 2.1. Na Seção 2.2 é descrito o algoritmo de agrupamento clássico *Fuzzy C-Means* (FCM). O cálculo de agrupamentos com distâncias adaptativas é detalhado na Seção 2.3. Por fim, na Seção 2.4, algoritmos de agrupamento com distância adaptativa clássicos são apresentados.

2.1 AGRUPAMENTO

Agrupamento é uma tarefa não supervisionada que reúne um conjunto de dados num grupo onde esses dados são mais similares entre si do que entre dados reunidos em diferentes grupos (JAIN e DUBES, 1988). Os dados apresentados aos algoritmos não supervisionados não possuem a informação dos rótulos da classe (categoria). Levam em conta apenas as descrições dos exemplos para realizar a tarefa de agrupamento. As descrições dos exemplos são vetores contendo características descritivas de cada exemplo.

Agrupamento também pode ser categorizado como generativo e discriminativo. O modelo generativo assume uma distribuição paramétrica de dados. Assim, é realizada a maximização da verossimilhança que possui como objetivo encontrar os parâmetros que maximizem a probabilidade dos dados terem sidos gerados pelo modelo avaliado. Por exemplo, numa distribuição gaussiana, temos que escolher a média e a covariância que gerou a distribuição dos dados. Na formulação mais geral, o número de grupos k também é considerado um parâmetro desconhecido. O valor de k é escolhido tal que o modelo de agrupamento melhor se ajuste aos dados. No modelo de agrupamento discriminativo, o algoritmo tenta agrupar os dados de forma a maximizar a similaridade dentro de cada grupo e minimizar a similaridade entre os grupos, com base numa matriz de similaridade definida sobre os dados do conjunto de entrada. Neste paradigma, não é necessário considerar um modelo de geração de base de dados paramétricos. Em ambos, modelos generativos e discriminativo, os algoritmos de agrupamento são geralmente colocados como problemas de otimização e resolvidos por métodos como *Expectation Maximization (EM)* (DEMPSTER; LAIRD; RUBIN, 1977) e o *Fuzzy C-Means* (BEZDEK, 1981).

Agrupamento tem sido utilizado para resolver diversos problemas. Para citar alguns, temos a mineração de textos, expressões gênicas, processamento de imagens, entre outras. O agrupamento é a aglomeração de objetos (ou exemplos) em grupos, de modo que os objetos pertencentes ao mesmo grupo sejam mais similares entre si de acordo com alguma medida de similaridade, enquanto os objetos pertencentes a grupos diferentes tenham uma similaridade menor. O objetivo do processo de agrupamento é maximizar a

homogeneidade dos objetos de um mesmo grupo enquanto maximiza a heterogeneidade entre objetos de grupos diferentes (STEPP e MICHALSKI, 1986). Segundo Jain e Dubes (JAIN e DUBES, 1988), agrupamento é o estudo formal de algoritmos e métodos para agrupar exemplos que não estão rotulados com uma classe correspondente.

De modo intuitivo, agrupamento é a criação de grupos que aglomeram objetos similares num mesmo grupo e objetos não similares em grupos diferentes. Everitt (EVERITT, 1980) diz que a definição formal de um grupo é inoportuna e resume a definições de grupos como segue:

Definição 1: um grupo é um conjunto de entidades que são similares, e entidades de diferentes grupos não são similares.

Definição 2: Um grupo é uma aglomeração de pontos no espaço de tal forma que a distância entre quaisquer dois pontos num mesmo grupo é menor que a distância entre qualquer ponto desse grupo a outro ponto não pertencente a ele.

Definição 3: Grupos podem ser descritos como regiões conectadas de um espaço multidimensional contendo uma alta densidade relativa de pontos, separadas de outras regiões por uma região contendo baixa densidade relativa de pontos.

Na Figura 1, retirada de Jain e Dubes (JAIN e DUBES, 1988), são ilustrados alguns exemplos de agrupamentos, onde cada exemplo é definido por dois atributos e está representado por cada ponto no espaço bidimensional. Nessa figura, percebe-se a existência de sete grupos.

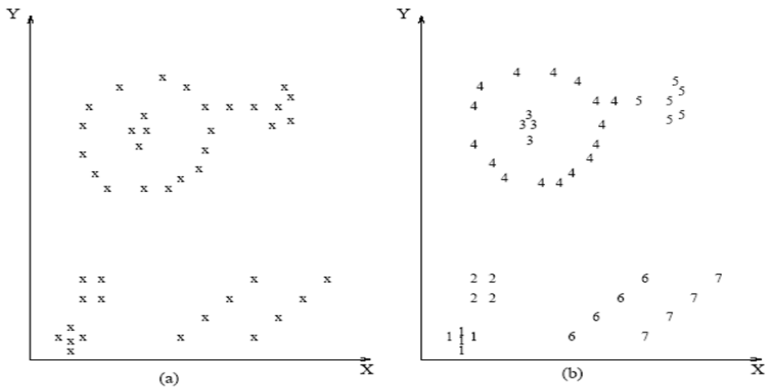


Figura 1 – Agrupamentos em duas dimensões

2.1.1 Representação de objetos do mundo real

Na maioria dos algoritmos de aprendizagem de máquina, a representação de um objeto do mundo real é feita por um vetor de atributos. Os objetos podem estar rotulados ou não com o atributo da classe. Uma das maneiras mais utilizadas para representar os exemplos é uma tabela no formato atributo-valor. A Tabela 2 representa o formato geral

de uma tabela atributo-valor de um conjunto com n exemplos $\mathbf{E} = e_i, \ldots, e_n$ e p atributos $\mathbf{X} = \mathbf{x}_j, \ldots, \mathbf{x}_p$. Nessa tabela, a linha i refere-se ao i-ésimo exemplo e_i, a coluna j refere-se ao j-ésimo $j = (1, 2, \ldots, p)$ atributo, e o valor x_{ij} refere-se ao valor do j-ésimo atributo do exemplo e_i. Já o valor h_i representa a classe do exemplo e_i. Essa classe pertence ao conjunto dos H possíveis valores de classes prévias $l = 1, \ldots, H$. Para a aprendizagem não supervisionada, a última coluna que contém os valores de classe não é considerada pelo algoritmo.

Tabela 2 – Exemplos no formato atributo-valor

	$\mathbf{X}_1$	$\mathbf{X}_2$	$\ldots$	$\mathbf{X}_p$	H
e_1	x_{11}	x_{12}	$\vdots$	x_{1p}	h_1
e_2	x_{21}	x_{22}	$\vdots$	x_{2p}	h_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
e_n	x_{n1}	x_{n2}	$\vdots$	x_{np}	h_n

2.1.2 Distâncias e similaridades

O objetivo do agrupamento é justamente agrupar exemplos de acordo com a similaridade entre eles. Desse modo, a similaridade é um conceito importante no processo de agrupamento. Na literatura, medidas de similaridade, coeficientes de similaridade ou distâncias são utilizadas para descrever quantitativamente a similaridade e a dissimilaridade entre dois exemplos, dois grupos ou entre um exemplo e um grupo.

Em geral, similaridade e distância são conceitos recíprocos. A similaridade e os coeficientes de similaridade são utilizados para descrever quantitativamente o quanto dois exemplos, dois grupos ou um exemplo e um grupo são similares entre si. Ou seja, quanto maior a similaridade entre eles, maior o valor da medida de similaridade. A medida de dissimilaridade e a distância são utilizadas para medir quantitativamente o quanto dois exemplos, dois grupos ou um exemplo e um grupo não são similares. Ou seja, quanto menor a similaridade entre eles, maior o valor da medida de dissimilaridade. Para exemplificar, considere dois pontos $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ e $x_m = (x_{m1}, x_{m2}, \ldots, x_{mp})$, sendo p o número de atributos. A distância Euclidiana entre os pontos x_i e x_m é calculada como segue:

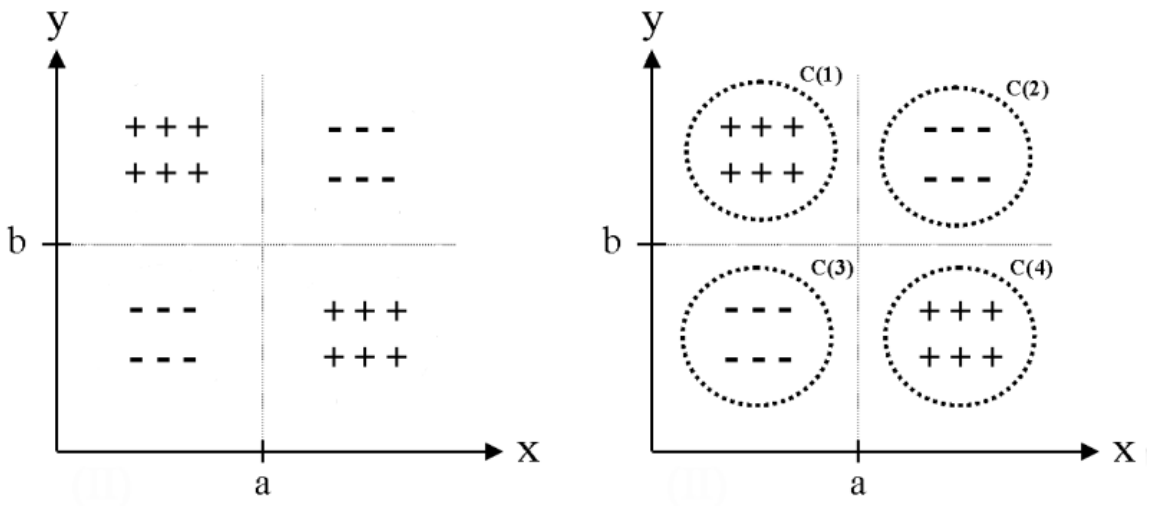
$$d_{i,m} = d(x_i, x_m) = \sqrt{\sum_{j=1}^{p} (x_{ij} - x_{mj})^2} \tag{2.1}$$

O trabalho de Hathaway *et al.* (HATHAWAY; BEZDEK; HU, 2000) descreve mais detalhes sobre tipos de distâncias. Na Seção 2.3, são apresentadas alguns tipos de distâncias adaptativas.

2.1.3 Centróide, partição, classe e grupo

Os grupos são aglomerados de exemplos similares entre si. A representação desse grupo de exemplos similares é usualmente realizada por um vetor da mesma dimensão dos exemplos contidos nesse grupo, o centróide. O cálculo do centróide depende do algoritmo utilizado para o agrupamento.

Um outro fator importante a ser observado é que um ou mais grupos podem conter exemplos que representem a mesma classe de um determinado conceito. A Figura 2(a) apresenta a formação original dos dados, onde cada classe é representada pelos símbolos (+) e (-). A Figura 2(b) apresenta quatro grupos distintos, sendo que cada classe é representada por dois desses grupos. Nas duas figuras, a e b são características que descrevem cada um dos exemplos. A formação de grupos é o resultado do algoritmo de agrupamento. Essa formação é chamada de partição. A partição é o conjunto de grupos resultado do algoritmo de agrupamento.



(a) Distribuição original dos dados

(b) Grupos diferentes representando a mesma classe

Figura 2 – Grupos *versus* classes

2.1.4 Agrupamento *hard* e agrupamento *fuzzy*

No agrupamento clássico (DUDA; HART; STORK, 2001), um exemplo da base de dados pode ser associado ou pertencer a apenas um grupo (agrupamento *hard*). O número de grupos formados pelo algoritmo de agrupamento é dado pelo número K e o número de exemplos da base de dados é dado por n. No agrupamento *fuzzy*, um exemplo e_i será associado a todos os grupos $C_k(k = 1, \ldots, K)$ de acordo com um grau de pertinência *fuzzy* (u_{ik}). O valor do grau de pertinência está no intervalo $[0, 1]$, onde 1 representa pertinência total do padrão e_i no grupo C_k, enquanto 0 indica que este padrão não pertence ao grupo

C_k. Caso o valor esteja no intervalo $(0 < u_{ik} < 1)$, significa que o padrão e_i pertence parcialmente ao grupo C_k. O grau de pertinência obedece as seguintes condições:

$$0 \leq u_{ik} \leq 1, \sum_{k=1}^{C} u_{ik} = 1 \,\forall\, i,\ 0 < \sum_{i=1}^{n} u_{ik} < n \,\forall\, k \tag{2.2}$$

Uma ilustração desses dois paradigmas é apresentada na Figura 3. Para o conjunto *fuzzy*, o padrão x_1 pertence parcialmente às três classes. Podemos quantificar isso a partir do grau de pertinência, que para o grupo 1 é $u_{11} = 0,22$, enquanto para o grupo 2 é $u_{12} = 0,63$ e para o grupo 3 é $u_{13} = 0,15$, sendo a soma deles igual a um. No conjunto *hard*, o padrão x_1 está apenas associado a um grupo.

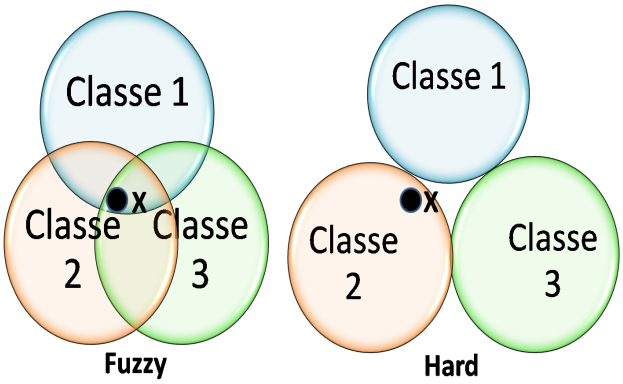


Figura 3 – *Fuzzy Versus Hard*

2.1.5 Processo de agrupamento

A tarefa de agrupamento não supervisionado é composta por 3 etapas como descrito a seguir (JAIN; MURTY; FLYNN, 1999):

1. **Representação dos dados:** nessa etapa, objetos reais são transformados em alguma representação (ver Seção 2.1.1). Na representação, os parâmetros escolhidos nesta etapa são o número de classes, o número de padrões disponíveis, bem como o número, tipo, e a escala das características disponíveis para representação da informação. A representação dos dados determina o tipo de estrutura de grupos que serão encontrados nos dados e será a entrada do algoritmo de agrupamento;

2. **Agrupamento:** Nessa etapa, o algoritmo de agrupamento é executado com as informações disponíveis. Esta etapa envolve o treinamento do algoritmo e a formação de grupos com base na similaridade entre os padrões da base de dados. O processamento consiste de diversas iterações até que seja satisfeito um critério de parada. O resultado do processo de agrupamento pode ser um agrupamento *hard*, exemplos

contidos em apenas um dos grupos formados, ou *fuzzy*, onde um exemplo pertence a cada um dos grupos formados baseado num grau de pertinência;

3. **Entendendo os grupos:** quando o algoritmo termina sua execução, os grupos estão formados. Os grupos formados são então associados a uma classe pré-definida utilizando alguma estratégia que possa indicar a real distribuição desses dados.

2.2 ALGORITMO DE AGRUPAMENTO *FUZZY C-MEANS*

O algoritmo *Fuzzy C-Means* (FCM) (BEZDEK, 1981) é um dos algoritmos de agrupamento mais conhecidos da literatura. A função objetivo desse algoritmo é geralmente modificada com o objetivo de criar outros algoritmos de agrupamento. O algoritmo FCM é a base dos algoritmos de agrupamento semissupervisionados que descrevemos no próximo capítulo.

O algoritmo FCM otimiza uma função objetivo através das iterações para calcular seus parâmetros desconhecidos, tornando o valor dessa função menor a cada iteração.

Com respeito a notação que é utilizada neste trabalho, temos que $\mathbf{E} = \{e_1, \ldots, e_n\}$ é o conjunto de n objetos. Cada objeto e_i é descrito por um vetor de característica $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})$ onde $x_{ij} \in \mathbb{R}\,(j = 1, \ldots, p)$. $\mathbf{X} = [x_{ij}]$ é uma matriz de características de dados $n \times p$. Cada grupo $C_k\,(k = 1, \ldots, K)$ possui um representante (ou protótipo) $\mathbf{v}_k = (v_{k1}, \ldots, v_{kp})$, onde $v_{kj} \in \mathbb{R}\,(j = 1, \ldots, p)$. $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_K)$ é a matriz de protótipos. A partição *fuzzy* é representada por $\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_n)$, onde $\mathbf{u}_i = (u_{i1}, \ldots, u_{iK})$, e $u_{ik} \in [0, 1] : \sum_{k=1}^{K} u_{ik} = 1$ é o grau de pertinência do objeto e_i no grupo C_k. Assume-se que os objetos pertencem à classe h das classes possíveis $P_h\,(h = 1, \ldots, H)$. A dissimilaridade $d^2(\mathbf{x}_i, \mathbf{v}_k)$ entre os vetores do padrão $\mathbf{x_i}$ e do protótipo $\mathbf{v_k}$ é representada pela distância Euclidiana entre eles. O fuzificador $m(> 1)$ atua como parâmetro para a otimização da função objetivo. A função objetivo do algoritmo FCM é:

$$J(U, V) = \sum_{k=1}^{K} \sum_{i=1}^{n} u_{ik}^m d^2(\mathbf{x}_i, \mathbf{v}_k) \tag{2.3}$$

onde U, respeita as condições da Equação 2.2.

A distância é calculada pela Equação 2.4:

$$d^2(\mathbf{x}_i, \mathbf{v}_k) = d_{ik}^2 = \|\mathbf{x_i} - \mathbf{v_k}\|_2^2 = (\mathbf{x_i} - \mathbf{v_k})^T(\mathbf{x_i} - \mathbf{v_k}) = \sum_{j=1}^{p}(x_{ij} - v_{kj})^2 \tag{2.4}$$

Diferentes valores de m na Equação 2.3 podem ter como consequência mudanças no resultado do algoritmo.

Para encontrar as equações dos parâmetros desconhecidos é realizada a minimização da Equação 2.3 utilizando o multiplicador de lagrange.

$$J(U, V, \lambda_1, \ldots, \lambda_n) = \sum_{i=1}^{n}\sum_{k=1}^{K} u_{ik}^m d^2(\mathbf{x}_i, \mathbf{v}_k) - \sum_{i=1}^{n} \lambda_i (\sum_{k=1}^{K} u_{ik} - 1) \tag{2.5}$$

Por exemplo, para encontrar a equação que calcula os graus de pertinência u_{ik} de cada exemplo aos grupos, é realizada a minimização com relação à matriz U, tendo como resultado a seguinte Equação:

$$u_{ik} = \frac{1}{\sum_{l=1}^{K} (\frac{d^2(\mathbf{x}_i, \mathbf{v}_k)}{d^2(\mathbf{x}_i, \mathbf{v}_l)})^{\frac{1}{m-1}}} \tag{2.6}$$

A dissimilaridade é uma distância entre dois pontos. No caso do algoritmo FCM, essa distância é dada pela Equação 2.4. Minimizando a função objetivo em relação a V, temos a fórmula que calcula o protótipo:

$$\mathbf{v_k} = \frac{\sum_{i=1}^{n} (u_{ik})^2 \mathbf{x_i}}{\sum_{i=1}^{n} (u_{ik})^2} \tag{2.7}$$

Em geral, o critério da função objetivo é otimizado quando o protótipo dos grupos estão próximos daquele ponto que possui a maior probabilidade de ser o representante do grupo $\mathbf{v_k}$. A característica desse algoritmo, é que após o passo de inicialização do algoritmo, o cálculo da matriz de graus de pertinência U e da matriz de protótipos se alternam até que seja alcançado algum critério de parada.

2.3 AGRUPAMENTO COM DISTÂNCIAS ADAPTATIVAS

Em geral, os algoritmos de agrupamento *fuzzy* com função objetivo particionam um conjunto de dados em grupos, em que um exemplo é associado a todos os grupos, minimizando uma função objetivo derivada da distância entre os protótipos do grupo e os objetos representadas pela matriz de características dos dados. Como consequência, os resultados do agrupamento são bastante influenciados pela forma como essa distância é calculada. A distância determina o formato dos grupos produzidos pelo algoritmo, e por isso, é influenciada pela natureza dos dados. Como resultado, existe uma quantidade significativa de algoritmos de agrupamentos *fuzzy* na literatura. O objetivo desses algoritmos é investigar o uso de diferentes funções de distância nos agrupamentos *fuzzy*, resultando em grupos que capturam diferentes distribuições e formatos dos dados com a finalidade de melhorar o desempenho dos algoritmos de agrupamentos.

No trabalho de Gustafson e Kessel (GUSTAFSON e KESSEL, 1979), os autores propuseram o algoritmo de agrupamento não supervisionado GK, que utiliza a distância de Mahalanobis que é definida por uma matriz *fuzzy* de covariância. Eles relataram

que o algoritmo GK é melhor do que algoritmos baseados distância Euclidiana, quando o formato dos dados é considerado. Frigui e Nasraoui (FRIGUI e NASRAOUI, 2004) introduziram uma abordagem onde a partição final do algoritmo minimiza a soma das distâncias ponderadas interna dos grupos e os pesos são otimizados para cada grupo. De Carvalho *et al.* (DE CARVALHO; TENORIO; CAVALCANTI JUNIOR, 2006) propuseram métodos adaptativos onde a distância adaptativa quadrática de cada grupo é definida por uma matriz *fuzzy* de covariância diagonal que são facilmente invertidas. Tang *et al.* (TANG; WANG; XU, 2010) propuseram um fator exponencial que pondera a distância Euclidiana no algoritmo FCM. Hung *et al.* (HUNG; KULKARNI; KUO, 2011) propuseram uma abordagem adaptativa híbrida em que adiciona à uma abordagem adaptativa, o centróide de cada grupo para aumentar a estabilidade no processo de agrupamento do algoritmo. Wu *et al.* (WU et al., 2012) generalizam vários tipos de distâncias para o algoritmo FCM. Eles apresentam experimentos extensivos para demonstrar que as mudanças de distâncias levam à convergência global do FCM em que o desempenho do agrupamento são significativamente afetados pelas opções de funções de distância. Ferreira e De Carvalho (FERREIRA e DE CARVALHO, 2014) propuseram métodos de agrupamento baseados em funções kernel com ponderação automática das variáveis através de distâncias adaptativas onde medidas de dissimilaridade são obtidas como somas de distâncias Euclidianas entre padrões e protótipos calculadas individualmente para cada variável através de funções kernel. Nos próximos parágrafos, são analisados brevemente algumas categorias de distância, que ajuda a contrastar as diferenças e enfatizar o nível de flexibilidade que essas trazem para o processo de agrupamento.

Os algoritmos discutidos neste trabalho baseiam-se na distância Euclidiana descrita na forma padrão:

$$d^2(\mathbf{x}_i, \mathbf{v}_k) = d_{ik}^2 = (\mathbf{x_i} - \mathbf{v_k})^T(\mathbf{x_i} - \mathbf{v_k}) = \sum_{j=1}^{p}(x_{ij} - v_{kj})^2 \tag{2.8}$$

A distância Euclidiana produz bons resultados apenas quando todos os grupos são esferóides, ou quando todos os grupos são bem separados. Isso é verdade mesmo quando todos os grupos têm o mesmo número de objetos (KRISHNAPURAM e KIM, 1999). As distâncias adaptativas permitem a construção de partições em vários formatos, além da forma esférica gerada pela distância Euclidiana, deste modo, pode aprender estruturas de distribuição de dados mais complexas. Basicamente, nesta seção são investigadas a distância adaptativa local e a distância adaptativa global. Essas distâncias permitem o algoritmo construir agrupamentos hiperesféricos (por exemplo, circulares, elípticos, etc) no formato:

$$d_{\mathbf{A}_k}^2(\mathbf{x}_i, \mathbf{v}_k) = (x_i - v_k)^T\mathbf{A}_k(x_i - v_k) = \sum_{j=1}^{p}\lambda_{kj}(\mathbf{x}_{ij} - \mathbf{v}_{kj})^2 \tag{2.9}$$

Essa distância adaptativa pode resultar em diferentes formas geométricas e orientação diferente nos grupos dependendo dos valores e formato da matriz de covariância $\mathbf{A}_k$. Estes grupos são representados por hiper-elipsóides planas, que podem ser considerados como hiperplanos. Por exemplo, se $\mathbf{A}_k$ é uma matriz identidade I, então essa distância possui o formado da Euclidiana padrão. Além disso, se $\mathbf{A}_k$ é uma matriz diagonal definida positiva, pode ser vista como um mecanismo de ponderação das características individuais. A matriz $\mathbf{A}_k$ é definida positiva se for simétrica e obedecer a restrição $x^T\mathbf{A}_k x > 0 \; \forall x \neq 0$. Neste caso, $\mathbf{A}_k = [\lambda_{kj}](j = 1, \ldots, p)$ é uma matriz diagonal, onde cada valor de λ_{kj} está associado com o valor correspondente do atributo $j(j = 1 \ldots p)$. A matriz $\mathbf{A_k} = \lambda_{k1}, \ldots, \lambda_{Kp}$ corresponde ao conjunto de matrizes de ponderação de características. A matriz $\mathbf{A_k}$ onde $(k = 1, \ldots, K)$ caracteriza a forma geral da medida de distância adaptativa local. Desse modo, a contribuição de cada variável para cada grupo pode ser diferente, *i.e.*, cada grupo pode ter um conjunto diferente de variáveis relevantes.

$$\mathbf{A}_k = \begin{pmatrix} \lambda_{k1} & 0 & \cdots & 0 \\ 0 & \lambda_{kj} & \cdots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \lambda_{kp} \end{pmatrix}$$

Ao considerar a medida de distância adaptativa global, todos os grupos são ponderados apenas por uma matriz $\mathbf{A}$. Igualmente, temos as mesmas propriedades descritas anteriormente, com a diferença que a matriz de pesos é única para todos os grupos. Assim, as variáveis possuem o mesmo peso para todos os grupos.

$$\mathbf{A} = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_j & \cdots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \lambda_p \end{pmatrix}$$

A medida de distância adaptativa é utilizada para cálculo de similaridade entre objetos num algoritmo de agrupamento que otimiza uma função objetivo. A otimização da função objetivo é obtida com derivações utilizando o multiplicador de lagrange. Desse modo, devido às restrições do multiplicador de lagrange, é preciso restringir o determinante da matriz $\mathbf{A}_k$. Assim, o determinante da matriz $\mathbf{A}_k$ assume a forma

$$det(\mathbf{A}_k) = \rho_k \;\; ; \;\; \rho_k > 0 \;\; \forall k \tag{2.10}$$

onde ρ_k é um parâmetro de restrição para o determinante e geralmente $\rho_k = 1$. A restrição pode ser descrita por dois métodos. O primeiro método restringe a soma dos valores da

diagonal da matriz $\mathbf{A}_k$, chamado de traço da matriz, na seguinte maneira

$$\sum_{j=1}^{p} \lambda_{kj} = \rho_k; \lambda_{kj} > 0 \ \ \forall k \tag{2.11}$$

O outro método restringe os valores da diagonal da matriz $\mathbf{A}_k$ utilizando produto de seus componentes da seguinte maneira

$$\prod_{j=1}^{p} \lambda_{kj} = \rho_k; \lambda_{kj} > 0 \ \ \forall k \tag{2.12}$$

No primeiro método assume-se que a soma dos pesos das variáveis deve ser igual a ρ_k, enquanto que no segundo, assume-se que o produto dos pesos das variáveis deve ser igual a ρ_k. Sendo que geralmente $\rho_k = 1 \ \ \forall k$.

Ao considerar as equações da distância adaptativa com ponderação global de variáveis, é preciso apenas uma única matriz de pesos $\mathbf{A}$ para todos os grupos. Do mesmo modo, essa distância pode ter seus valores da diagonal restritos pela soma ou pelo produto de seus componentes igual a ρ. Como consequência, teremos dois tipos de distâncias adaptativas com ponderação local e com ponderação global de variáveis com restrições em que a soma ou o produto dos pesos das variáveis deve ser igual a um. São então criadas pelo menos quatro novas versões de algoritmos de agrupamento. Nesta tese, esses algoritmos são apresentados como:

1. Algoritmos com distância adaptativa global com restrição de produto (GP).
2. Algoritmos com distância adaptativa global com restrição de soma (GS);
3. Algoritmos com distancia adaptativa local com restrição de produto (LP);
4. Algoritmos com distância adaptativa local com restrição de soma (LS);

Nas próximas seções, apresentamos alguns algoritmos com distâncias adaptativas.

2.4 ALGORITMOS DE AGRUPAMENTO COM DISTÂNCIA ADAPTATIVA

Esta seção apresenta alguns algoritmos de agrupamento relacionados com distâncias adaptativas.

2.4.1 Algoritmo GK

A primeira e mais geral versão adaptativa do algoritmo FCM, é baseada numa distância quadrática para cada grupo definida por uma matriz *fuzzy* de covariância introduzida por Gustagson e Kessel (GUSTAFSON e KESSEL, 1979). Para este método, o

critério que avalia a otimização apropriada entre os grupos e seus representantes é definido como

$$J(\mathbf{U}, \mathbf{A_1}, \ldots, \mathbf{A}_k, \mathbf{V}) = \sum_{k=1}^{K} \sum_{i=1}^{n} (u_{ik})^2 d^2_{\mathbf{A}_k}(\mathbf{x}_i, \mathbf{v}_k) \tag{2.13}$$

onde

- $\mathbf{U}$ é a matriz de graus de pertinência;
- $\mathbf{A_1} \ldots \mathbf{A_K}$ é a matriz de vetores de pesos
- $\mathbf{V}$ é a matriz de protótipos;
- $d^2_{\mathbf{A}_k}(\mathbf{x}_i, \mathbf{v}_k)$ calcula a distância quadrada Euclidiana entre um protótipo $\mathbf{v}_k$ e um objeto $\mathbf{x}_i$:

$$d^2_{\mathbf{A}_k}(\mathbf{x}_i, \mathbf{v}_k) = (x_i - v_k)^T \mathbf{A}_k (x_i - v_k) \tag{2.14}$$

O algoritmo começa a partir de um grau de pertinência inicial para cada objeto $\mathbf{e}_i$ em cada grupo C_k e alterna um passo de representação e um passo a alocação até a convergência do algoritmo, quando o critério J atinge um valor estacionário que representa um mínimo local. A etapa de representação tem duas fases: na primeira fase, o grau de pertinência de cada padrão de $\mathbf{e}_i$ em cada grupo C_k e as matrizes $\mathbf{A_k}$ são fixados. Os protótipos $\mathbf{v}_k$ dos grupos $C_k(k = 1, \ldots, K)$, que minimizam o critério de agrupamento J, são atualizados de acordo com

$$\mathbf{v}_k = \frac{\sum_{i=1}^{n} (u_{ik})^2 \mathbf{x}_i}{\sum_{i=1}^{n} (u_{ik})^2} \tag{2.15}$$

Na segunda etapa, o grau de pertinência de cada objeto $\mathbf{e}_i$ no grupo C_k, e os protótipos $\mathbf{v}_k$ dos grupos correspondentes $C_k(k = 1, \ldots, K)$ são fixados. A matriz simétrica definida positiva $\mathbf{A_k}(k = 1, \ldots, K)$, que minimiza o critério de agrupamento J em $det(\mathbf{A_k}) = 1$, é atualizada de acordo com a seguinte expressão:

$$\mathbf{A}_k = det(\mathbf{A}_k)^{1/p} \mathbf{A}_k^{-1} \; ; \; \mathbf{A}_k = \sum_{i=1}^{n} u_{ij}^m (\mathbf{x}_i - \mathbf{v}_k)(\mathbf{x}_i - \mathbf{v}_k)^T. \tag{2.16}$$

Na etapa de alocação, os protótipos $\mathbf{v}_k$ correspondente aos grupos $C_k(k = 1, \ldots, K)$ e as matrizes $\mathbf{A}_k(k = 1, \ldots, K)$ são fixados. O grau de pertinência $u_{ik}(i = 1, \ldots, n)$ de cada objeto e_i em cada grupo $C_k(k = 1, \ldots, K)$, que minimizam o critério de agrupamento J sob as restrições 2.2, são atualizados de acordo com a seguinte Equação:

$$
\begin{aligned}
u_{ik} &= \left[\sum_{l=1}^{K}\left(\frac{d^2_{\mathbf{A}_k}(\mathbf{x}_i,\mathbf{v}_k)}{d^2_{\mathbf{A}_l}(\mathbf{x}_i,\mathbf{v}_l)}\right)^{\frac{1}{(m-1)}}\right]^{-1} \\
&= \left[\sum_{l=1}^{K}\left(\frac{(\mathbf{x}_i-\mathbf{v}_k)^T\mathbf{A}_k(\mathbf{x}_i-\mathbf{v}_k)}{(\mathbf{x}_i-\mathbf{v}_l)^T\mathbf{A}_l(\mathbf{x}_i-\mathbf{v}_l)}\right)^{\frac{1}{(m-1)}}\right]^{-1}
\end{aligned}
\tag{2.17}
$$

2.4.2 FCM-DFCV

De Carvalho *et al.* (DE CARVALHO; TENORIO; CAVALCANTI JUNIOR, 2006) introduziram um caso especial do algoritmo GK. A matriz simétrica definida positiva $\mathbf{A}_k$ do algoritmo GK deve ser invertida, a fim de atualizar as distâncias na segunda fase da etapa de representação do algoritmo. No entanto, nos casos em que essas matrizes são singulares, problemas numéricos podem ocorrer e as matrizes podem ser inversíveis. No algoritmo FCM-DFCV, a distância Euclidiana adaptativa é definida por matrizes diagonais definidas positivas que são facilmente invertidas, como mostrado na Seção 2.3. Assim, a função objetivo é definida como

$$
J(\mathbf{U},\mathbf{A_1},\ldots,\mathbf{A_K},\mathbf{V}) = \sum_{k=1}^{K}\sum_{i=1}^{n}(u_{ik})^2 d^2_{\mathbf{A}_k}(\mathbf{x}_i,\mathbf{v}_k) \tag{2.18}
$$

onde

- $\mathbf{U}$,$\mathbf{A_1}\ldots,\mathbf{A_K}$, $\mathbf{V}$ são como descritas anteriormente;
- A matriz $\mathbf{A}_k$ é uma matriz diagonal como a descrita na Seção 2.3 $d^2_{\mathbf{A}_k}(\mathbf{x}_i,\mathbf{v}_k) = \sum_{j=1}^{p}\lambda_{kj}(\mathbf{x}_{ij}-\mathbf{v}_{kj})^2$

Esse algoritmo inicia como o algoritmo GK, com três etapas. A primeira etapa é igual à etapa do algoritmo GK, com os protótipos calculados pela Equação 2.15.

Na segunda etapa, os elementos λ_{kj} da matriz diagonal definida positiva $\mathbf{A_k}(k = 1,\ldots,K)$, conforme Seção 2.3, que minimiza o critério de agrupamento J com a restrição $det(\mathbf{A}_k) = 1$, são atualizados de acordo com

$$
\lambda_{kj} = \tag{2.19}
$$

$$
\frac{\left\{\prod_{h=1}^{p}\left(\sum_{i=1}^{n}(u_{ij})^2(x_{hj}-v_{hj})^2\right)\right\}^{\frac{1}{p}}}{\sum_{i=1}^{n}(u_{ik})^2(x_{ij}-v_{kj})^2}
$$

Na etapa de alocação, o grau de pertinência $u_{ik}(i = 1, \ldots, n)$ de cada objeto e_i em cada grupo $C_k(k = 1, \ldots, K)$, que minimiza o critério de agrupamento J sob as restrições (2.2), são atualizados como o algoritmo GK de acordo com a Equação 2.17.

2.5 MEDIDAS DE AVALIAÇÃO DE ALGORITMOS DE AGRUPAMENTO

Nesta seção, a metodologia de validação para comparar o desempenho dos algoritmos semissupervisionados é descrita. A precisão dos resultados obtidos pode ser medida comparando-se a partição de teste com uma partição classificada previamente utilizando o índice de Rand Corrigido (CR) (HUBBERT e ARABIE, 1985), o índice Informação Mútua Normalizada (NMI) (COVER e THOMAS, 2006) e uma taxa de acerto de classificação (ACC) (BREIMAN et al., 1984).

É importante ressaltar que os algoritmos de agrupamento *fuzzy* são utilizados com o objetivo de obter uma partição *fuzzy*, $Z = (Z_1, \ldots, Z_K)$ de E em K grupos *fuzzy* representados por $\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_n)$, com $\mathbf{u}_{ik} = (u_{i1}, \ldots, u_{iK})\,(i = 1, \ldots, n)$. Depois, uma partição *hard* $Q = (Q_1, \ldots, Q_K)$ é obtida a partir dessa partição *fuzzy* pela definição de uma partição *hard* $Q_k(k = 1, \ldots, K)$ com $Q_k = \{e_i : u_{ik} \geq u_{il}\, \forall l \in \{1, \ldots, K\}\}$. A seguir, apresentamos três índices para avaliação de algoritmos de agrupamento.

2.5.1 Índice de Rand corrigido

Um índice externo de adequação de partições avalia o grau em que duas partições de n objetos se correspondem. Uma partição é proveniente do resultado de um algoritmo de agrupamento. Há uma série de índices externos definidos na literatura, tais como Hubbert, Jaccard, Rand e Rand Corrigido (JAIN e DUBES, 1988). Uma característica da maioria destes índices é que eles podem ser sensíveis ao número de classes nas partições ou a distribuições de elementos nos agrupamentos. Por exemplo, alguns índices possuem uma tendência a apresentar valores mais elevados para as partições com mais classes (Hubbert e Rand), outros para as partições com um menor número de classes (Jaccard) (DUBES, 1987). O índice de Rand Corrigido (CR), possui seus valores corrigidos de acordo com acertos nas comparações das partições, por isso não possui nenhuma destas características indesejáveis (MILLIGAN e COOPER, 1986). Esse índice pode ser obtido a partir da tabela de contingência e construído a partir de duas partições mostradas na Tabela 3. Seja $Q = \{Q_1, \ldots, Q_k, \ldots, Q_K\}$ a partição dada pelo algoritmo de agrupamento, e $R = \{R_1, ..., R_h, ..., R_H\}$ a partição definida pela classificação prévia. O CR pode ser definido pela Equação 2.20 como segue:

Tabela 3 – Tabela de contingência para duas partições

	Q_1	Q_2	...	Q_K	
R_1	n_{11}	n_{12}	...	n_{1K}	$n_{1.}$
R_2	n_{21}	n_{22}	...	n_{2K}	$n_{2.}$
...	...	...		...	
R_H	n_{H1}	n_{H2}	...	n_{HK}	$n_{R.}$
	$n_{.1}$	$n_{.2}$		$n_{.K}$	$n_{..} = n$

$$CR = \frac{\sum_h^H \sum_k^K \binom{n_{hk}}{2} - \binom{n}{2}^{-1} \sum_h^H \binom{n_{h\cdot}}{2} \sum_k^K \binom{n_{\cdot k}}{2}}{\frac{1}{2}\left[\sum_h^H \binom{n_{h\cdot}}{2} + \sum_k^K \binom{n_{\cdot k}}{2}\right] - \binom{n}{2}^{-1} \sum_h^H \binom{n_{h\cdot}}{2} \sum_k^K \binom{n_{\cdot k}}{2}} \tag{2.20}$$

onde $\binom{n}{2} = \frac{n(n-1)}{2}$ e n_{hk} é o número de objetos que estão em ambas partições Q_k e R_h. O termo $n_{\cdot k}$ representa o número de objetos no grupo Q_k e $n_{h\cdot}$ é o número de objetos na classe R_h, e n é o número total de objetos.

Esse índice possui um valor no intervalo [-1, 1], onde o valor 1 indica uma coesão perfeita entre as partições e -1 indica que não existe coesão entre as partições. Os valores próximos a 1 indicam boa coesão. Milligan e Cooper (MILLIGAN e COOPER, 1986) indicam que valores abaixo de 0.05 são partições geradas por acaso.

2.5.2 Índice de informação mútua normalizado

O índice externo de informação mútua (I) tem origem nos princípios da teoria da informação e na noção de entropia (COVER e THOMAS, 2006). Quando a entropia é aplicada ao processo de agrupamento, é uma medida que mede a incerteza de que um elemento e_i estará associado a um determinado grupo C_k. Para o caso de um elemento estar associado a um grupo por acaso, a entropia é 0. A noção de entropia pode ser estendida para informação mútua, que mede o quanto podemos, em média, reduzir a incerteza sobre a escolha aleatória de que um elemento e_i seja associado a um grupo C_k, levando em conta que seu grupo é conhecido de outro agrupamento prévio. A informação mútua é calculada através da Equação 2.21.

$$I = \sum_{k=1}^{K} \sum_{h=1}^{H} n_{hk} \log\left(\frac{n\, n_{hk}}{n_{\cdot k} n_{h\cdot}}\right) \tag{2.21}$$

A informação mútua I, é uma medida sobre o espaço de todos os agrupamentos. No entanto, não é limitada por um valor constante, que faz com que seja difícil de interpretar. No trabalho de Strehl e Ghosh (STREHL e GHOSH, 2002), foi proposta uma normalização pela média geométrica. Para este fim, eles determinam o agrupamento que tem a média máxima da informação mútua normalizada de todos os agrupamentos em

consideração, onde a informação mútua normalizada (NMI) entre dois agrupamentos é definida como como segue

$$NMI = \frac{I(R,Q)}{\sqrt{EN(R)En(Q)}} \tag{2.22}$$

onde $I(R,Q)$ é a informação mútua entre R e Q. $EN(R)$ e $EN(Q)$ são entropias de R e Q respectivamente. Assim, a informação mútua normalizada pode ser calculada pela Equação 2.23.

$$NMI = \frac{\sum_{k=1}^{K}\sum_{h=1}^{H} n_{hk}\log\left(\frac{n\,n_{hk}}{n_{\cdot k}n_{h\cdot}}\right)}{\sqrt{\left(\sum_{k=1}^{K} n_{\cdot k}\log\frac{n_{\cdot k}}{n}\right)\left(\sum_{h=1}^{H} n_{h\cdot}\log\frac{n_{h\cdot}}{n}\right)}} \tag{2.23}$$

onde n_{hk}, $n_{\cdot k}$ e $n_{h\cdot}$ são como já descritos. NMI possui valores entre [0,1]. A informação mútua normalizada foi utilizada para validação de algoritmos de agrupamentos semissupervisionados nos trabalhos (MARAZIOTIS, 2012; ZHANG et al., 2012; YIN; SHU; HUANG, 2012).

2.5.3 Taxa de acerto de classificação

A taxa de acerto ou acurácia (ACC) procura uma regra de decisão que minimiza a probabilidade de erro (BREIMAN et al., 1984). O valor máximo de ACC indica melhor desempenho do algoritmo de agrupamento.

$$ACC = \left(\frac{\sum_{k=1}^{K}\max_{1\leq h\leq H}\ n_{hk}}{n}\right) \tag{2.24}$$

onde n_{hk} é como já descrito.

2.6 PROPRIEDADES INERENTE À INICIALIZAÇÃO E PARADA DE ALGORITMOS DE AGRUPAMENTO

Os algoritmos de agrupamento particionais são sensíveis à normalização da base de dados, às inicializações das matrizes de protótipo, de grau de pertinência e ao momento de parada do algoritmo. Nessa seção são apresentadas algumas abordagens para execução dos algoritmos de agrupamento particionais.

2.6.1 Normalização da base de dados

É indicado que a normalização dos dados seja utilizada. A normalização possui como objetivo evitar discrepância entre a relevância dos atributos de uma base de dados. Por exemplo, no mundo real, pode-se facilmente encontrar atributos como idade e peso

de uma pessoa. Diferenças de escala entre esses atributos podem levar a uma distorção no cálculo da distância Euclidiana. Existem algumas opções para se fazer a normalização.

A primeira pode transformar os valores em escala decimal, dividindo os valores dos atributos pela mesma potência de 10 de acordo com a Equação 2.25.

$$x_{ij}^{novo} = \frac{x_{ij}}{10^d} \tag{2.25}$$

onde x_{ij}^{novo} é o valor normalizado e d é o menor inteiro de forma que $Max(|x_{ij}^{novo}|) < 1$.

Também existe a normalização Min-Max, onde os valores são substituídos por valores entre um intervalo pré-definido, geralmente na escala entre 0 e 1. Esses valores são encontrados a partir do cálculo de onde se utiliza o valor mínimo e o valor máximo da base de dados conforme a Equação 2.26:

$$x_{ij}^{novo} = \frac{x_{ij} - Min(X)}{Max(X) - Min(X)} \tag{2.26}$$

onde $Max(X)$ é o valor mais alto do atributo e $Min(X)$ é o menor valor do atributo.

Ainda há o *Z-score* onde os valores são padronizados através da diferença da média do atributo. Os novos valores estarão na escala entre 0 e 1.

$$x_{ij}^{novo} = \frac{x_{ij} - \mu_X}{\sigma_X} \tag{2.27}$$

onde μ_X é a média dos atributos e σ_X é o desvio padrão dos atributos.

2.6.2 Inicialização da matriz de protótipos

A matriz de protótipos é inicializada geralmente escolhendo aleatoriamente um exemplo da base de dados. Para que essa abordagem consiga algum efeito, é importante repetir a execução do algoritmo diversas vezes com diferentes exemplos escolhidos aleatoriamente da base de dados para compor a matriz inicial de protótipos. Algumas outras abordagens também podem ser realizadas. No trabalho de Bouchachia e Pedrycz (BOUCHACHIA e PEDRYCZ, 2006a), por exemplo, a matriz de protótipos é inicializada a partir dos resultados do algoritmo *Fuzzy C-Means* original.

2.6.3 Inicialização da matriz de grau de pertinência

A inicialização da matriz de grau de pertinência pode ser feita de diversas maneiras. Essa matriz é característica de algoritmos que obtém partição *fuzzy* como resultado. Nesta seção abordamos três abordagens para iniciar essa matriz.

A primeira, é chamada neste trabalho de inicialização *fuzzy*. Nessa abordagem, o grau de pertinência u_{ik} de um exemplo e_i a cada um dos grupos C_k é atribuído aleatoriamente e possui valor no intervalo $[0, 1]$. A soma dos graus de pertinência de um padrão aos grupos deve ser 1.

A segunda abordagem, é chamada de inicialização *hard*. O grau de pertinência u_{ik} de um exemplo e_i a um grupo C_k é atribuído aleatoriamente à apenas um dos grupos. O valor desse grau de pertinência é um. Para manter a condição da soma dos graus de pertinência de um padrão igual a um, os graus de pertinência desse padrão e_i aos outros grupos é zero.

A terceira abordagem é um pouco mais elaborada. Nessa abordagem, chamada de inicialização por agrupamento, é calculada a similaridade do exemplo e_i ao grupo C_k pela Equação 2.28.

$$u_{ik} = \left[\frac{d^2(\mathbf{x}_i, \mathbf{v}_k)}{\sum_{l=1}^{K} d^2(\mathbf{x}_i, \mathbf{v}_l)} \right]^{-1} \tag{2.28}$$

obedecendo as condições da Equação 2.2. No entanto, há uma exceção para o caso de um protótipo de um grupo C_k ser igual a um determinado exemplo e_i. Nesse caso, a similaridade é igual a 1 dividido pelo número de grupos iguais ao exemplo e 0 para todos os grupos que não possui valor igual ao exemplo e_i.

2.6.4 Critério de parada

Ainda relativo aos parâmetros inerentes aos algoritmos de agrupamento, existe o critério de parada. Esse parâmetro diz respeito à maneira de como os algoritmos param seu processamento. São apresentadas três opções para critério de parada. Para todos eles, utiliza-se um limiar ϵ positivo com um valor baixo. O primeiro critério, é a estabilização da norma da diferença das matrizes de graus de pertinência formadas da iteração anterior $U^{(t-1)}$ e da iteração atual $U^{(t)}$, sendo t o contador de iterações. O valor dessa diferença deve ser menor que o limiar ϵ para que o algoritmo pare. Este limiar deve ser positivo maior que 0 e muito menor que 1. Este critério é expresso na Equação (2.29).

$$\|U^{(t-1)} - U^{(t)}\| = \sum_{k=1}^{K} \sum_{i=1}^{n} (u_{ik}^{(t-1)} - u_{ik}^{(t)})^2 \tag{2.29}$$

O segundo critério segue o mesmo princípio do primeiro, porém a matriz considerada é a do protótipo. A norma da diferença entre as matrizes de protótipos da iteração anterior $V^{(t-1)}$ e atual $V^{(t)}$ deve ser menor que o limiar ϵ. O critério é apresentado na Equação (2.30).

$$\|V^{(t-1)} - V^{(t)}\| = \sum_{k=1}^{K} \sum_{j=1}^{p} (v_{kj}^{(t-1)} - v_{kj}^{(t)})^2 \tag{2.30}$$

onde t é o contador de iterações.

O terceiro critério é mais direto. Esse critério interrompe o processamento do algoritmo quando os valores da diferença do valor da função de otimização da iteração anterior $J^{(t-1)}$ e da atual $J^{(t)}$ for menor que o limiar ϵ. Após aplicar o critério de parada,

o algoritmo de agrupamento cessa o processamento formando as partições (grupos) e seus respectivos protótipos e a matriz final de graus de pertinência dos padrões a cada um dos grupos formados.

2.7 SÍNTESE DO CAPÍTULO

Este capítulo apresentou os conceitos básicos de aprendizagem de máquina não supervisionada. Descrevemos o algoritmo *Fuzzy C-Means*, que é um dos algoritmos mais importantes deste tipo. Em seguida, foi descrito o conceito de distâncias adaptativas, que procuram adequar o agrupamento a distribuições mais complexas dos dados, utilizando uma matriz diagonal definida positiva para ponderar o conjunto de variáveis. Depois, foram apresentados alguns algoritmos de agrupamento que utilizam distância adaptativa. Pra finalizar, foram descritos métodos de avaliação desses algoritmos e alguns parâmetros importantes para os algoritmos de agrupamento.

3 AGRUPAMENTO SEMISSUPERVISIONADO

Este capítulo apresenta os principais conceitos de agrupamento semissupervisionado na Seção 3.1. Depois, descreve e analisa alguns algoritmos de agrupamento semissupervisionado na Seção 3.2.

3.1 APRENDIZAGEM SEMISSUPERVISIONADA

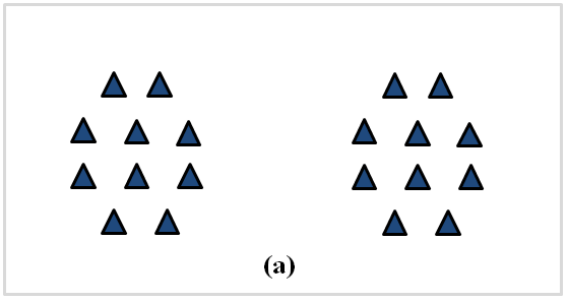
Os algoritmos semissupervisionados são uma abordagem intermediária entre os algoritmos de aprendizagem supervisionada e não supervisionada. O detalhe crucial de um algoritmo semissupervisionado é que o mesmo seja capaz de aprender a partir de dados rotulados junto com dados não rotulados para realizar uma tarefa de classificação ou de agrupamento.

Para ilustrar a aprendizagem semissupervisionada, a Figura 4 apresenta as diferenças entre os conjuntos de treinamento dos diferentes paradigmas de aprendizagem. A Figura 4(a) apresenta os dados não rotulados, ou seja, dados que não possuem informação acerca do rótulo. A Figura 4(b) representa o conjunto de treinamento de dados rotulados, ou seja, sabe-se previamente a classe, ou rótulo, de cada item desse conjunto. O conjunto de treinamento é formado por duas classes, uma delas representada por estrelas e a outra por sinais positivos. Dessa forma, a aprendizagem supervisionada é guiada pelo conjunto de treinamento totalmente rotulado. Por último, temos a Figura 4(c) que representa a aprendizagem semissupervisionada. Note que no conjunto de treinamento existe dados não rotulados, representados pelos triângulos e dados rotulados, representados pela estrela e pelo sinal positivo. Dessa forma, o treinamento semissupervisionado é guiado pelas informações disponíveis da descrição dos dados não rotulados e também pelo rótulo dos dados rotulados.

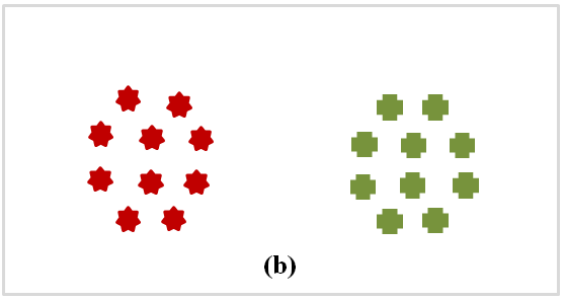
Um exemplo de como a informação rotulada pode ser útil é apresentado na Figura 5. Nela há dois grupos representando duas classes. Se não houvesse informação nenhuma acerca de alguns exemplos do conjunto de dados os grupos seriam colocados em classes erradas, pois o algoritmo poderia agrupar os grupos verticalmente como mostrado na Figura 5(a). Mas na Figura 5(b) existe informação de rótulos de alguns exemplos da base de dados, desse modo o algoritmo agrupa corretamente os exemplos, colocando-os dentro da representação correta.

3.1.1 Agrupamento semissupervisionado baseados em função objetivo

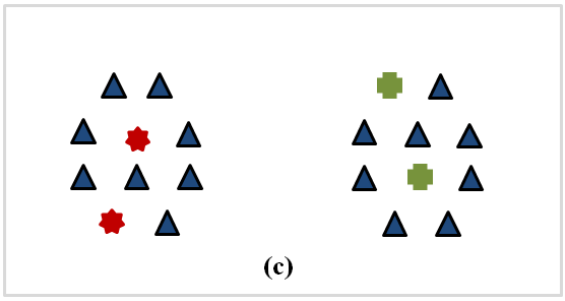
Os algoritmos de agrupamento semissupervisionados realizam alguma estratégia para levar em conta dados rotulados em seu treinamento. Os algoritmos de agrupamento



(a) Dados não rotulados

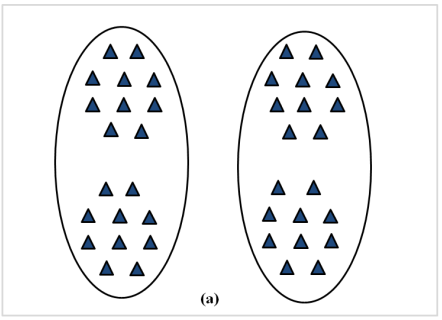


(b) Dados totalmente rotulados

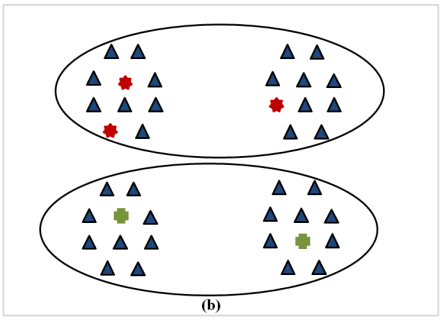


(c) Dados parcialmente rotulados

Figura 4 – Aprendizagem semissupervisionada



(a) Dados Não Rotulados



(b) Dados Parcialmente Rotulados

Figura 5 – Aprendizagem semissupervisionada

semissupervisionados baseados em função objetivo, geralmente, são compostos por, pelo menos, uma expressão herdada de algum algoritmo de agrupamento não supervisionado e, pelo menos, mais uma expressão supervisionada desenvolvida para utilizar informações conhecidas previamente para guiar o aprendizado do algoritmo.

Agrupamento é uma tarefa muito difícil por se tratar de uma tarefa guiada apenas pela natureza intrínseca dos dados baseada apenas no vetor de características desses dados (ou na matriz relacional). Por conseguinte, qualquer informação externa suplementar disponível pode ser muito útil na tarefa de encontrar grupos nesses dados (JAIN, 2010).

Como informação prévia, os algoritmos de agrupamento semissupervisionado podem utilizar alguns dados rotulados, ou restrições em alguns exemplos, para melhorar o desempenho de algoritmos de agrupamento não supervisionados clássicos. Métodos de agrupamento semissupervisionado foram desenvolvidos ao longo da última década (CHAPELLE; ZIEN; SCHOLKOPF, 2006; ZENG et al., 2012; ZHU, 2008). Eles podem ser basicamente em dois tipos: baseados em similaridade e baseados em busca (BASU; BILENKO; MOONEY, 2003).

Nas abordagens baseadas na similaridade, são utilizados algoritmos de agrupamento existentes com alguma medida de similaridade, porém a medida de similaridade é adaptada para satisfazer as restrições ou rótulos disponíveis para guiar o agrupamento. Várias abordagens semissupervisionadas baseadas em similaridade foram propostas. Wu *et al.* (WU et al., 2012) propuseram uma abordagem não-paramétrica para aprender funções de distância Bregman para o agrupamento semissupervisionado, que é capaz de lidar com dados de alta dimensão de forma eficiente. Wang *et al.* (WANG; YUEN; FENG, 2013) propõem um novo método de aprendizagem semissupervisionada com uma medida de distância regularizada de acordo com a topologia local dos dados.

Em abordagens baseadas em busca, o algoritmo de agrupamento é modificado para que algum especialista forneça rótulos ou restrições que são usadas para guiar a busca por um melhor agrupamento. As estratégias para realizar as modificações podem ser categorizadas de acordo com a maneira que a informação externa ou prévia é caracterizada para orientar o processo de agrupamento: sementes, restrições e rótulos.

A abordagem baseada em "sementes" utiliza exemplos rotulados para inicializar a configuração dos grupos. Estas sementes são utilizadas como protótipos iniciais para orientar o processo de agrupamento. Basu *et al.* (BASU; BANERJEE; MOONEY, 2002) apresentou duas versões modificadas do algoritmo *k-means* clássico que utilizam dados rotulados iniciais como sementes.

As outras duas abordagens, restrições e rótulos, geralmente modificam a função objetivo existente de um algoritmo de agrupamento, adicionando pelo menos uma expressão que leva em conta um tipo específico de informação para orientar o processo de agrupamento. O principal desafio do projeto destes algoritmos reside na criação de uma função objetivo que seja capaz de refletir a natureza do problema, de modo que sua otimização revele uma estrutura significativa no conjunto de dados.

A abordagem que utiliza "restrições" faz uso de informações na forma de restrições de pares. Restrição pareada é um termo usado para especificar as relações entre pares de exemplos nos dados. Os métodos de agrupamento com restrições de pares incorporam e impõem tais restrições. Wagstaff *et al.* (WAGSTAFF et al., 2001) introduziu as restrições *must-link* e *cannot-link*. Restrições *must-link* especificam os pares de itens que devem ser atribuídos ao mesmo grupo. Por outro lado, as restrições *cannot-link* especificam os pares

de items que devem ser atribuídos a grupos diferentes. Alguns exemplos de algoritmos de agrupamento semissupervisionado que utilizam este tipo de restrição podem ser encontrados em (GRIRA; CRUCIANU; BOUJEMAA, 2008; BASU et al., 2006; MEI e CHENVC, 2013). O trabalho de Maraziotis (MARAZIOTIS, 2012) quantifica as restrições *must-link* e/ou restrições *cannot-link* para aplicar em dados de expressão de genes. Outros tipos de restrições têm sido desenvolvidas, como a restrição tripla, utilizada por Wang *et al.* (WANG; YUEN; FENG, 2013). A restrição em Tripla (e_i, e_j, e_m), é tal que (e_i, e_j) devem estar mais próximos que (e_i, e_m). Assim, se (e_i, e_j) é uma restrição positiva e (e_i, e_m) é uma restrição negativa, (e_i, e_j, e_m) é uma restrição tripla.

Finalmente, a abordagem que utiliza "rótulos" é baseada no uso de exemplos rotulados para orientar o processo de agrupamento a partir de exemplos parcialmente rotulados. Exemplos rotulados, que muitas vezes são um pequeno subconjunto dos dados, são utilizados para orientar o processo de agrupamento e, ao mesmo tempo, para aumentar a precisão do algoritmo de agrupamento não supervisionado.

O trabalho de Pedrycz e Waletzky (PEDRYCZ, 1985; PEDRYCZ e WALETZKY, 1997) foi um dos primeiros a adaptar a função objetivo do algoritmo *Fuzzy C-Means*, mostrando que os dados não rotulados utilizados em conjunto com dados rotulados melhoraram a qualidade dos grupos formados pelo algoritmo. Este algoritmo restringe o grau de pertinência de um objeto a um grupo a ter um valor próximo de um, se este grupo está em uma correspondência de um-para-um com uma determinada classe prévia. Seguindo essa linha, Bouchachia e Pedrycz (BOUCHACHIA e PEDRYCZ, 2006a) adicionaram um termo com duas otimizações na função objetivo do algoritmo *Fuzzy C-Means*. Os autores testaram ainda esse algoritmo como sendo um classificador supervisionado, utilizando apenas dados rotulados no treinamento e obtiveram resultados melhores do que os algoritmos clássicos como a rede neural SVM (VAPNIK, 1998). Endo *et al.* (ENDO et al., 2009) propõem dois algoritmos de agrupamento semissupervisionados. O primeiro ele chama de algoritmo semissupervisionado padrão, pois ele modifica o algoritmo clássico de agrupamento FCM para utilizar diretamente rótulos em sua função objetivo. O segundo algoritmo adiciona um termo supervisionado baseado em entropia ao termo clássico do algoritmo FCM. Kanzawa *et al.* (KANZAWA; ENDO; MIYAMOTOE, 2011) impulsiona o processo de agrupamento com um termo que utiliza regularização da entropia entre o grau de pertinência e seu valor da classe, otimizando a função objetivo com o objetivo de minimizar valores de entropia. Zeng *et al.* (ZENG et al., 2012) atribuem pesos arbitrariamente maiores para as exemplos rotulados no caso de falta de algum conhecimento prévio sobre os exemplos.

Pode-se perceber que existem diferentes abordagens para utilizar a informação supervisionada nos algoritmos de agrupamento. Neste trabalho é dada atenção particular à informação dos rótulos de exemplos previamente rotulados na otimização de uma função

objetivo baseada na função original do algoritmo clássico *Fuzzy C-Means*. Na Seção 3.2, esta abordagem é apresentada detalhadamente através de alguns algoritmos de agrupamento semissupervisionado. Além disso, é apresentado um estudo das propriedades de convergência e complexidade desses algoritmos.

3.2 ALGORITMOS DE AGRUPAMENTO SEMISSUPERVISIONADO

Na classificação de padrões, as duas abordagens clássicas mais utilizadas são a aprendizagem não supervisionada e a supervisionada. Porém, existem algumas limitações inerentes à cada uma dessas abordagens. A classificação não supervisionada não possui informações descritivas acerca dos padrões, desse modo, muitas vezes a tarefa de associar grupos a uma categoria pré-definida é uma tarefa complexa. Quando se possui informações sobre os padrões, a abordagem supervisionada é geralmente utilizada. Mas existem bases de dados onde essas informações são custosas e às vezes até impossíveis de se obter. Por exemplo, imagine rotular uma base contendo 10000 textos ou imagens, ou ainda, rotular cadeias genéticas contendo milhares de genes. A classificação semissupervisionada surgiu como uma opção para atenuar algumas das limitações das duas abordagens acima.

O algoritmo semissupervisionado é importante justamente por precisar de pouca informação prévia (rótulos ou restrições) sobre a base de dados que queremos realizar o agrupamento. A abordagem semissupervisionada, geralmente, obtém um resultado melhor do que se utilizassem apenas padrões não rotulados numa abordagem não supervisionada ou se utilizasse os poucos padrões rotulados em uma abordagem supervisionada.

Nesta seção são descritos os algoritmos de agrupamento semi supervisionados explorados nesta tese para efeito de análise comparativa. A diferença entre esses algoritmos estão em algumas características inerentes à função objetivo de cada um dos algoritmos. Os algoritmos são:

1. Algoritmo Isodata-PS
2. Algoritmo BSSC
3. Algoritmo semissupervisionado Padrão (sSSC)
4. Algoritmo semissupervisionado com Entropia (ESSC)
5. Algoritmo NebFuzz

3.2.1 Algoritmo Isodata-PS

O método proposto por Pedrycz e Waletzky (PEDRYCZ e WALETZKY, 1997) é um dos primeiros algoritmos a utilizar a abordagem de agrupamento para algoritmos semissupervisionados. Este algoritmo, assim como outros desse paradigma, foi concebido

para atenuar os defeitos dos paradigmas mais utilizados, supervisionado e não supervisionado. De posse de bases de sinais biológicos (eletrocardiogramas) onde havia poucos elementos rotulados, eles formularam um novo algoritmo modificando a função objetivo (2.3) do algoritmo FCM. O termo adicionado a função objetivo do algoritmo FCM utiliza os rótulos disponíveis dos exemplos da base de dados para aumentar os graus de pertinência atribuídos pelo algoritmo para os grupos que representam a classe que o exemplo pertence. A função objetivo desse algoritmo depois de modificada do original *Fuzzy C-Means* (FCM) (BEZDEK, 1981) se torna:

$$J(\mathbf{U},\mathbf{V}) = \sum_{k=1}^{K}\sum_{i=1}^{n}(u_{ik})^2 d^2(\mathbf{x}_i,\mathbf{v}_k) + \alpha \sum_{k=1}^{K}\sum_{i=1}^{n}(u_{ik} - b_i f_{ik})^2 d^2(\mathbf{x}_i,\mathbf{v}_k) \tag{3.1}$$

onde

- $\mathbf{U}$ é a matriz de grau de pertinência;
- $\mathbf{V}$ é a matriz de protótipos;
- dados rotulados e dados não rotulados são identificados pelo vetor booleano $\mathbf{b_i} = (b_1,\ldots,b_n)$, onde $b_i = 1$, se o exemplo $e_i\,(i = 1,\ldots,n)$ é rotulado, e $b_i = 0$ caso contrário;
- a matriz booleana $\mathbf{F} = [f_{ik}]$ $(n \times H)$ dá a classificação prévia dos objetos, onde $f_{ik} = 1\,(i = 1,\ldots,n; k = 1,\ldots,H)$, se o objeto e_i pertence à classe P_k, e $f_{ik} = 0$ caso contrário. Note que as classes e os grupos são indexados por k. Além disso, se o número de classes H é inferior ao número de grupos K, $f_{ik} = 0$ para $k = H{+}1,\ldots,K$. Se o número de grupos K é inferior ao número de classes H, as classes com índices maiores que K não terão grupos associados;
- O parâmetro α é um fator de escala para equilibrar os termos supervisionados e não supervisionados da função objetivo. Se $\alpha = 0$, a função objetivo fica reduzida ao algoritmo FCM;
- $d^2(\mathbf{x}_i,\mathbf{v}_k)$ calcula a distância Euclidiana entre o protótipo do grupo $\mathbf{v}_k$ e o objeto $\mathbf{x}_i$ de acordo com a Equação 2.4

Note que o termo supervisionado deste algoritmo é otimizado de tal forma que, se uma classe *a priori* de um objeto e_i é P_k $(f_{ik} = 1)$, seu grau de pertinência ao grupo correspondente C_K é induzido a ser próximo de 1.

Observe o exemplo da Tabela 4 em que antes da otimização, os graus de pertinência de um exemplo aos grupos são muito próximos. Após a otimização do termo supervisionado, o grau de pertinência desse exemplo ao grupo um é forçado a ter o valor um pelo termo supervisionado da função objetivo (3.1) conforme apresentado na Tabela 5.

Tabela 4 – Grau de pertinência antes da otimização

	$k=1$	$k=2$
$\mathbf{u}_1$	$u_{11}=0,6$	$u_{12}=0,4$

Tabela 5 – Grau de pertinência após otimização

	$k=1$	$k=2$
$\mathbf{u}_1$	$u_{11}=0,87$	$u_{12}=0,13$

Isodata-PS é um algoritmo iterativo que começa a partir de uma solução inicial e alterna dois passos (de representação e de alocação). Na etapa de representação, a matriz $\mathbf{U}$ de graus de pertinência é mantida fixa. A função objetivo é minimizada em relação à matriz de protótipos $\mathbf{V}$. Os protótipos dos grupos $\mathbf{v}_k\,(k=1,\ldots,K)$ são calculados de acordo com a Equação 3.2.

$$\mathbf{v}_k = \frac{\sum_{i=1}^{n}\left[(u_{ik})^2 + \alpha \times (u_{ik} - b_i f_{ik})^2\right]\mathbf{x}_i}{\sum_{i=1}^{n}\left[(u_{ik})^2 + \alpha \times (u_{ik} - b_i f_{ik})^2\right]} \tag{3.2}$$

Na etapa de alocação, a matriz de protótipos $\mathbf{V}$ é mantida fixa. A função objetivo é minimizada no que diz respeito à matriz $\mathbf{U}$ de graus de pertinência. O grau de pertinência do objeto $e_i\,(i=1,\ldots,n)$ num grupo *fuzzy* $C_k\,(k=1,\ldots,K)$ é calculado de acordo com (PEDRYCZ e WALETZKY, 1997; PEDRYCZ, 1985).

$$u_{ik} = \frac{1}{1+\alpha}\left\{\frac{1+\alpha(1-b_i\sum_{l=1}^{K} f_{il})}{\sum_{l=1}^{K}\left(\frac{d^2(\mathbf{x}_i,\mathbf{v}_k)}{d^2(\mathbf{x}_i,\mathbf{v}_l)}\right)} + \alpha f_{ik} b_i\right\} \tag{3.3}$$

Essas duas etapas são calculadas iterativamente até que um critério de parada seja satisfeito.

3.2.2 Algoritmo BSSC

O algoritmo proposto por Bouchachia e Pedrycz (BOUCHACHIA e PEDRYCZ, 2006a) basicamente estende a função objetivo (2.3) do algoritmo *Fuzzy C-Means*. O objetivo dessa mudança é realizar duas otimizações em relação aos graus de pertinências entre diferentes grupos que podem representar uma mesma classe utilizando informação dos rótulos para guiar essas otimizações. O algoritmo BSSC busca uma partição *fuzzy* com K grupos e seus protótipos correspondentes de tal forma que é otimizado pela função objetivo:

$$J(\mathbf{U},\mathbf{V}) \;=\; \sum_{k=1}^{K}\sum_{i=1}^{n}(u_{ik})^2 d^2(\mathbf{x}_i,\mathbf{v}_k) + \alpha\sum_{k=1}^{K}\sum_{i=1}^{n}(u_{ik}-\tilde{u}_{ik})^2 d^2(\mathbf{x}_i,\mathbf{v}_k) \tag{3.4}$$

onde $\mathbf{U}$, $\mathbf{V}$, α e $d^2(\mathbf{x}_i,\mathbf{v}_k)$ são como descritos anteriormente.

BSSC é um algoritmo iterativo que começa a partir de uma solução inicial e alterna, até que um critério de parada é satisfeito. O cálculo da matriz $\tilde{\mathbf{U}}$ é mais um passo que

alterna o cálculo da matriz $\mathbf{V}$ de protótipos (passo representação) e o cálculo da matriz $\mathbf{U}$ de graus de pertinência (etapa de alocação).

O termo $\tilde{u}_{ik}$ da matriz $\tilde{\mathbf{U}}$ é obtido resolvendo (BOUCHACHIA e PEDRYCZ, 2006a)

$$\tilde{u}_{ik}^{(t)} = \tilde{u}_{ik}^{(t-1)} - \beta \frac{\partial Q(\mathbf{F}, \tilde{\mathbf{U}})}{\partial \tilde{u}_{ik}}; \tag{3.5}$$

onde

- $\tilde{u}_{ik} \in [0, 1]$;
- $Q(\mathbf{F}, \tilde{\mathbf{U}}) = \sum_{h=1}^{H} \sum_{i=1}^{n} b_i \left(f_{ih} - \sum_{k \in \pi_h} \tilde{u}_{ik} \right)^2$
- b_i é como já descrito;
- $\mathbf{F}$ é como já descrito. Supõe-se que uma classe pode ser dividida entre vários grupos e se H é o número de classes e K o número de grupos, então $K \geq H$. Senão, cada classe P_h contém um número de grupos C_l e $\sum_{l=1}^{H} C_l = K$;
- $\beta > 0$ é a taxa de aprendizagem;
- t é o contador de iterações.

O objetivo é minimizar a diferença entre o grau de pertinência de um objeto e_i e uma classe h, e a soma dos graus de pertinência do objeto e_i para todos os grupos pertencentes a classe h. Eles são calculados de forma iterativa da seguinte forma (BOUCHACHIA e PEDRYCZ, 2006a)

$$\begin{aligned} \tilde{u}_{ik}^{(t)} &= \tilde{u}_{ik}^{(t-1)} + \\ &+ 2\beta b_i \sum_{h=1}^{H} (f_{ih} - \sum_{C_l \in \pi_h} \tilde{u}_i^{(t-1)}) * \begin{cases} 1 & se\ C_k \in \pi_h \\ 0 & senão \end{cases} \end{aligned} \tag{3.6}$$

O termo π_h é importante na Equação (3.6), pois contém o conjunto de grupos que representam a classe h. O processo para se calcular a soma ψ_h é descrito logo abaixo:

$$\psi_h = \sum_{k \in \pi_h} \tilde{u}_{ik} \tag{3.7}$$

Duas matrizes tem que ser calculadas para se computar ψ, uma matriz de mapeamento $M_{H \times K}$ e uma matriz correspondente $P_{H \times K}$. A primeira especifica o relacionamento entre as classes e os grupos enquanto a última especifica o número de padrões de cada classe em cada grupo. A célula $P(h, k)$ representa o número de padrões de uma classe h

no grupo C_k. Um exemplo e_i pertence ao grupo C_k, se o grau de pertinência do exemplo e_i a C_k é o maior. Assim, para cada classe, é construída uma lista de grupos e o número de padrões associados a essa classe. Ordenando a matriz P na ordem ascendente, cada linha de M contém a lista de grupos ordenados por dominância. O número de classes por grupo é especificado no começo do algoritmo e levado em conta. Assim, o conjunto π_h de ψ_h contém os grupos dominantes na classe h e assim, pode-se realizar a soma.

Note que a otimização desse algoritmo é divida em duas etapas. Primeiro, o grau de pertinência u_{ik} é minimizado para ter o valor de $tildeu_{ik}$. Depois, o valor de $\tilde{u}_{ik}$ é otimizado de forma que os graus de pertinência dos grupos que representam a classe prévia desse exemplo (π_h) são forçados a ter valor um. Desse modo, observe o exemplo mostrado na Tabela 6 em que $\pi_1 = 1, 3$, ou seja, a classe um é representada pelos grupos um e três. Após a otimização do termo supervisionado, a soma dos graus de pertinência aos grupos um é três são forçados a ter o valor um, conforme mostrado na Tabela 7.

Tabela 6 – Grau de pertinência antes da otimização

	$k=1$	$k=2$	$k=3$
$\mathbf{u}_1$	$u_{11}=0,6$	$u_{12}=0,2$	$u_{13}=0,2$

Tabela 7 – Grau de pertinência após otimização

	$k=1$	$k=2$	$k=3$
$\mathbf{u}_1$	$u_{11}=0,5$	$u_{12}=0,1$	$u_{12}=0,4$

Na etapa de representação, a matriz $\tilde{\mathbf{U}}$ e a matriz $\mathbf{U}$ de graus de pertinência são mantidas fixas. A função objetivo é minimizada em relação à matriz de protótipos $\mathbf{V}$. Os protótipos dos grupos $\mathbf{v}_k\,(k = 1, \ldots, K)$ são calculados de acordo com

$$\mathbf{v}_k = \frac{\sum_{i=1}^{n}[(u_{ik})^2 + \alpha(u_{ik} - \tilde{u}_{ik})^2]\mathbf{x}_i}{\sum_{i=1}^{n}[(u_{ik})^2 + \alpha(u_{ik} - \tilde{u}_{ik})^2)]} \tag{3.8}$$

Na etapa de alocação, a matriz $\tilde{\mathbf{U}}$ e a matriz $\mathbf{V}$ de protótipos são mantidas fixas. A função de objetivo é minimizada em relação à matriz $\mathbf{U}$ de graus de pertinência. O grau de pertinência do objeto $e_i\,(i = 1, \ldots, n)$ num grupo fuzzy $C_k\,(k = 1, \ldots, K)$ é calculado de acordo com

$$u_{ik} = \frac{\alpha\tilde{u}_{ik}}{(1+\alpha)} + \frac{1 - \frac{\alpha}{(1+\alpha)}\sum_{l=1}^{K}\tilde{u}_{il}}{\sum_{l=1}^{K}\left(\frac{d^2(\mathbf{x}_i,\mathbf{v}_k)}{d_{il}^2}\right)} \tag{3.9}$$

As etapas de representação e de alocação são calculadas iterativamente até que um critério de parada seja satisfeito.

3.2.3 Algoritmo semissupervisionado padrão (sSSC)

Endo *et al.* (ENDO et al., 2009) desenvolveram o algoritmo de agrupamento semissupervisionado padrão (sSSC), introduzindo uma matriz de pertinência supervisionada

f_{ik} do objeto $e_i\,(i = 1, \ldots, n)$ no grupo $C_k\,(k = 1, \ldots, K)$ no algoritmo de agrupamento padrão FCM. Note que esse algoritmo é o único que possui apenas um termo. Caso o item seja rotulado e pertença à mesma classe representada pelo grupo C_k, $f_{ik} = 1$. O grau de pertinência é influenciado a ter o valor 1 neste grupo. Caso isso não aconteça, o algoritmo é igual ao FCM (BEZDEK, 1981) clássico. A função objetivo deste algoritmo é a seguinte (ENDO et al., 2009):

$$J(\mathbf{U}, \mathbf{V}) \;=\; \sum_{k=1}^{K}\sum_{i=1}^{n}(u_{ik} - f_{ik})^2 d^2(\mathbf{x}_i, \mathbf{v}_k) \tag{3.10}$$

onde

- $\mathbf{U}$, $\mathbf{V}$, f_{ik}, e $d^2(\mathbf{x}_i, \mathbf{v}_k)$ são como já descritos;

A otimização desse algoritmo é igual ao algoritmo Isodata-PS. A diferença é que nesse algoritmo não existe o termo não supervisionado.

O algoritmo sSSC também é um algoritmo iterativo com duas etapas (de representação e de alocação). Na etapa de representação, a matriz $\mathbf{U}$ de graus de pertinência é mantida fixa. A função objetivo é minimizada em relação à matriz de protótipos $\mathbf{V}$. Os protótipos dos grupos $\mathbf{v}_k\,(k = 1, \ldots, K)$ são calculados de acordo com

$$\mathbf{v}_k = \frac{\sum_{i=1}^{n}\left(u_{ik} - f_{ik}\right)^2 \mathbf{x}_i}{\sum_{i=1}^{n}\left(u_{ik} - f_{ik}\right)^2} \tag{3.11}$$

Na etapa de alocação, a matriz de protótipos $\mathbf{V}$ é mantida fixa. A função objetivo é minimizada no que diz respeito à matriz $\mathbf{U}$ de graus de pertinência. O grau de pertinência do objeto $e_i\,(i = 1, \ldots, n)$ num grupo *fuzzy* $C_k\,(k = 1, \ldots, K)$ é calculado de acordo com (ENDO et al., 2009)

$$u_{ik} = f_{ik} + \left(1 - \sum_{l=1}^{K} f_{il}\right) \frac{\left(\frac{1}{d^2(\mathbf{x}_i, \mathbf{v}_k)}\right)^{\frac{1}{m-1}}}{\sum_{l=1}^{K}\left(\frac{1}{d_{il}}\right)^{\frac{1}{m-1}}} \tag{3.12}$$

As etapas de representação e de alocação são calculadas iterativamente até que um critério de parada seja satisfeito.

3.2.4 Algoritmo semissupervisionado regularizado por entropia (ESSC)

No trabalho de Endo *et al.* (ENDO et al., 2009), também é proposto um novo algoritmo semissupervisionado (chamado aqui de ESSC) em que adiciona um termo supervisionado baseado na entropia ao termo do algoritmo de agrupamento FCM. A entropia é calculada utilizando a diferença entre o grau de pertinência de um objeto e_i ao grupo

C_k e o valor de f_{ik} que indica se um objeto e_i pertence à classe representada pelo grupo C_k. Caso o objeto pertença ao grupo C_k, o valor da entropia é baixo, caso contrário, esse valor é alto. A função objetivo deste algoritmo é a seguinte (ENDO et al., 2009):

$$J(\mathbf{U},\mathbf{V}) = \sum_{k=1}^{K}\sum_{i=1}^{n}(u_{ik})^2 d^2(\mathbf{x}_i,\mathbf{v}_k) + \alpha\sum_{k=1}^{K}\sum_{i=1}^{n}(|(u_{ik}-f_{ik})|\log|(u_{ik}-f_{ik})|) \quad (3.13)$$

onde

- $\mathbf{U}$, $\mathbf{V}$, f_{ik}, e $d^2(\mathbf{x}_i,\mathbf{v}_k)$ são como já descritos.

O termo supervisionado desse algoritmo é otimizado de tal forma que, caso o exemplo e_i pertence ao grupo C_k, o valor do grau de pertinência desse exemplo ao grupo C_k é forçado a ser igual a um, caso contrário, o valor do grau de pertinência é forçado a ser igual a zero. Assim, observe o exemplo da Tabela 8 em que antes da otimização, os graus de pertinência de um exemplo aos grupos são muito próximos. Após a otimização do termo supervisionado, o grau de pertinência desse exemplo ao grupo um é forçado a ter o valor um conforme apresentado na Tabela 9.

Tabela 8 – Grau de pertinência antes da otimização

	$k=1$	$k=2$
$\mathbf{u}_1$	$u_{11}0,6$	$u_{12}=0,4$

Tabela 9 – Grau de pertinência após otimização

	$k=1$	$k=2$
$\mathbf{u}_1$	$u_{11}=1,0$	$u_{12}=0,0$

O algoritmo ESSC também é um algoritmo de iterativo com duas etapas (de representação e de alocação). Na etapa de representação, a matriz $\mathbf{U}$ de graus de pertinência é mantida fixa. A função objetivo é minimizada em relação à matriz de protótipos $\mathbf{V}$. Os protótipos dos grupos $\mathbf{v}_k\,(k=1,\ldots,K)$ são calculados de acordo com

$$\mathbf{v}_k = \frac{\sum_{i=1}^{n}(u_{ik}-f_{ik})^2\,\mathbf{x}_i}{\sum_{i=1}^{n}(u_{ik}-f_{ik})^2} \quad (3.14)$$

Na etapa de alocação, a matriz de protótipos $\mathbf{V}$ é mantida fixa. A função objetivo é minimizada no que diz respeito à matriz $\mathbf{U}$ de graus de pertinência. O grau de pertinência do objeto $e_i\,(i=1,\ldots,n)$ num grupo *fuzzy* $C_k\,(k=1,\ldots,K)$ é calculado de acordo com

$$u_{ik} = f_{ik} + \frac{e^{-\alpha d^2(\mathbf{x}_i,\mathbf{v}_k)}}{\sum_{l-1}^{K} e^{-\alpha d^2(\mathbf{x}_i,\mathbf{v}_l)}}\left(1-\sum_{l=1}^{K} f_{il}\right) \quad (3.15)$$

As etapas de representação e de alocação são calculadas iterativamente até que um critério de parada seja satisfeito.

3.2.5 Algoritmo NebFuzz

O algoritmo NebFuzz foi proposto na minha dissertação de mestrado e publicado no trabalho (MACARIO e DE CARVALHO, 2010). Este algoritmo também aproveita informação de rótulos existentes e adiciona um segundo termo supervisionado ao algoritmo de agrupamento clássico FCM (BEZDEK, 1981). Supõe-se que os objetos que pertencem à mesma classe *a priori* devem ter seus graus de pertinência em um determinado grupo mais semelhantes possível. Com base nessa premissa, o termo supervisionado da função objetivo é otimizado de tal forma que, se um par de objetos está na mesma classe *a priori*, seus graus de pertinência correspondentes aos grupos *fuzzy* são influenciados de forma a terem valores próximos um do outro.

Assim como o algoritmo FCM, este algoritmo procura uma partição *fuzzy* em K grupos e seus protótipos correspondentes de tal forma que é otimizado um critério de adequação (função objetivo) medindo o ajuste entre os grupos *fuzzy* e seus protótipos. O critério de adequação é

$$\begin{aligned} J(\mathbf{U},\mathbf{V}) &= \sum_{k=1}^{K}\sum_{i=1}^{n}(u_{ik})^2 d^2(\mathbf{x}_i,\mathbf{v}_k) \\ &+ \alpha \times \sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{m=1}^{n} b_i b_m t_{im}(u_{ik}-u_{mk})^2 d^2(\mathbf{x}_i,\mathbf{x}_m) \end{aligned} \tag{3.16}$$

onde

- $\mathbf{U}$, $\mathbf{V}$ e $d_k^2(\mathbf{x}_i,\mathbf{v}_k)$ são como já descritos;
- objetos rotulados e não rotulados são identificados por um vetor booleano $b_i = (b_1,\ldots,b_n)$, onde $b_i = 1$ se o objeto $e_i\,(i = 1,\ldots,n)$ é rotulado, e $b_i = 0$ se não é rotulado; $b_j = 1$ so o objeto e_j é rotulado, senão $b_j = 0$;
- $t_{im} = 1$, se os objetos e_i e e_m são da mesma classe; senão, $t_{im} = 0$;
- o parâmetro α é um fator de balanceamento entre os termos supervisionado e não supervisionado que compõem a função objetivo. Se $\alpha = 0$, a função objetivo reduz-se ao algoritmo GK;
- $d^2(\mathbf{x}_i,\mathbf{x}_m)$ é a distância Euclidiana entre os exemplos e_i e e_m:

$$d^2(\mathbf{x}_i,\mathbf{x}_m) = \|\mathbf{x_i}-\mathbf{x_m}\|_2^2 = (\mathbf{x_i}-\mathbf{x_m})^T(\mathbf{x_i}-\mathbf{x_m}) = \sum_{j=1}^{p}(x_{ij}-x_{mj})^2 \tag{3.17}$$

Observe o exemplo da Tabela 10 em que antes da otimização o grau de pertinência de dois exemplos vizinhos, ou seja, da mesma classe são bem diferentes. Após a otimização

Tabela 10 – Grau de pertinência antes da otimização

	$k=1$	$k=2$
$\mathbf{u}_1$	$u_{11}=0,9$	$u_{12}=0,6$
$\mathbf{u}_2$	$u_{21}0,6$	$u_{22}=0,4$

Tabela 11 – Grau de pertinência após otimização

	$k=1$	$k=2$
$\mathbf{u}_1$	$u_{11}=0,81$	$u_{12}=0,19$
$\mathbf{u}_2$	$u_{21}=0,77$	$u_{22}=0,23$

do termo supervisionado os graus de pertinência desses exemplos ficam mais parecidos em relação aos grupos, como mostrado na Tabela 11.

O algoritmo NebFuzz também é um algoritmo iterativo com duas etapas (de representação e de alocação). Na etapa de representação, a matriz $\mathbf{U}$ de graus de pertinência é mantida fixa. A função objetivo é minimizada em relação à matriz de protótipos $\mathbf{V}$. Os protótipos dos grupos $\mathbf{v}_k\,(k=1,\ldots,K)$ são calculados de acordo com

$$\mathbf{v}_k = \frac{\sum_{i=1}^{n}\left(u_{ik}\right)^2\mathbf{x}_i}{\sum_{i=1}^{n}\left(u_{ik}\right)^2} \tag{3.18}$$

Na etapa de alocação, a matriz de protótipos $\mathbf{V}$ é mantida fixa. A função objetivo é minimizada no que diz respeito à matriz $\mathbf{U}$ de graus de pertinência. O grau de pertinência do objeto $e_i\,(i=1,\ldots,n)$ num grupo *fuzzy* $C_k\,(k=1,\ldots,K)$ é calculado de acordo com

$$u_{ik} = \frac{1+\alpha b_i\left[\left(\sum_{l=1}^{K}\frac{\sum_{m=1}^{n}b_m t_{im}(u_{mk}-u_{ml})d^2(\mathbf{x}_i,\mathbf{x}_m)}{\left(d^2(\mathbf{x}_i,\mathbf{v}_l)\right)^2+\alpha b_i\sum_{m=1}^{n}b_m t_{im}d^2(\mathbf{x}_i,\mathbf{x}_m)}\right)\right]}{\sum_{l=1}^{K}\left[\frac{\left(d^2(\mathbf{x}_i,\mathbf{v}_k)\right)^2+\alpha b_i\sum_{m=1}^{n}b_m t_{im}d^2(\mathbf{x}_i,\mathbf{x}_m)}{\left(d^2(\mathbf{x}_i,\mathbf{v}_l)\right)^2+\alpha b_i\sum_{m=1}^{n}b_m t_{im}d^2(\mathbf{x}_i,\mathbf{x}_m)}\right]} \tag{3.19}$$

As etapas de representação e de alocação são calculadas iterativamente até que um critério de parada seja satisfeito.

3.2.6 Etapas dos algoritmos

Segue a descrição das etapas de todos os algoritmos de agrupamento descritos. Cada algoritmo de agrupamento semissupervisionado é representada por uma letra conforme Tabela 12. As equações desses algoritmos são apresentadas na Tabela 13.

Tabela 12 – Símbolo que representa cada algoritmo de agrupamento semissupervisionado no esquema.

	Algoritmos
A	NebFuzz
B	BSSC
C	Isodata-PS
D	sSSC
E	ESSC

ESQUEMA DOS ALGORITMOS DE AGRUPAMENTO SEMI-SUPERVISIONADOS

1. **Inicialização (Igual para todos os algoritmos)**

a. Fixar o número de grupos K; Fixar $MaxIter$ (número de máximos de iterações); Fixar $\epsilon << 0$; Fixar $\alpha > 0$; Fixar β no caso do algoritmo BSSC.

b. Execute o algoritmo FCM até que $|J^{t+1} - J^t| \leq \epsilon$ **ou** $s > MaxIter$ para inicializar a matriz de graus de pertinências **U** e de protótipos **V**

c. Calcular $J^{(0)}$, e atribuir $t = 1$

Repetir :

2. **Etapa de Representação:**
   Calcular os protótipos $\mathbf{v}_k$ com as Equações: A1, B1, C1, D1 ou E1
3. **Etapa de Alocação:** A2, B2, C2, D2 ou E2
   Calcular o grau de pertinência u_{ik} dos objetos $e_i (i = 1, \ldots, n)$ no grupo C_k $(k = 1, \ldots, K)$ com as Equações
   Apenas para o algoritmo BSSC, calcular a matriz de pertinência $\tilde{u}_{ik}$ do objeto $e_i (i = 1, \ldots, n)$ no grupo C_k $(k = 1, \ldots, K)$ com a Equação (3.6)
4. **Cálculo da Função Objetivo:**
   Calcular $J^{(s)}$ com as Equações: A3, B3, C3, D3 ou E3
   t=t+1

Até que $|J^{s+1} - J^s| \leq \epsilon$ **ou** $s > MaxIter$

Tabela 13 – Equações relativas aos algoritmos semissupervisionados.

Algoritmo	Equações		
A	A1 (3.19)	A2 (3.19)	A3 (3.16)
B	B1 (3.3)	B2 (3.3)	B3 (3.1)
C	C1 (3.9)	C2 (3.9)	C3 (3.4)
D	D1 (3.12)	D2 (3.12)	D3 (3.10)
E	E1 (3.15)	E2 (3.15)	E3 (3.15)

3.2.7 Propriedades de convergência

Nesta seção, é apresentada a prova da convergência do algoritmo NebFuzz. A mesma prova pode ser facilmente estendida para os outros algoritmos de agrupamento semissupervisionado apresentados neste capítulo.

O algoritmo NebFuzz busca por uma partição *fuzzy* $P^* = \{C_1^*, \ldots, C_K^*\}$ de E em K grupos *fuzzy* representados por $\mathbf{U}^* = (\mathbf{u}_1^*, \ldots, \mathbf{u}_n^*)$, um vetor de protótipos K-dimensional correspondente $\mathbf{V}^* = (\mathbf{v}_1^*, \ldots, \mathbf{v}_K^*)$ representando os grupos *fuzzy* nas partições P, tal que

$$J(\mathbf{V}^*, \mathbf{U}^*) = min\left\{J(\mathbf{V}, \mathbf{U}) : \mathbf{V} \in \mathbb{L}^K, \mathbf{U} \in \mathbb{U}^n\right\}, \tag{3.20}$$

onde

- $\mathbb{L}$ é o espaço de representação de protótipos tal que $\mathbf{v}_k \in \mathbb{L}\,(k = 1, \ldots, K)$. Aqui, $\mathbb{L} = \mathbb{R}^p$ e $\mathbf{V} \in \mathbb{L}^K = \mathbb{L} \times \ldots \times \mathbb{L} = \mathbb{R}^{K \times p}$;

- $\mathbb{U}$ é o espaço de graus de pertinência de partição *fuzzy* tal que $\mathbf{u}_k \in \mathbb{U}\,(k = 1, \ldots, K)$. Aqui $\mathbb{U} = \{\mathbf{u} = (u_1, \ldots, u_K) \in [0,1] \times \ldots \times [0,1] = [0,1]^K : \sum_{k=1}^K u_k = 1\}$ e $\mathbf{U} \in \mathbb{U}^n = \mathbb{U} \times \ldots \times \mathbb{U}$;

De acordo com Diday e Simon (DIDAY e SIMON, 1976), as propriedades de convergência deste tipo de algoritmo pode ser estudada a partir de duas séries: $v_t = (\mathbf{V}^t, \mathbf{U}^t) \in \mathbb{L}^K \times \mathbb{U}^n$ e $u_t = J(v_t) = J(\mathbf{V}^t, \mathbf{U}^t)$, $t = 0, 1, \ldots$. Partindo de uma solução inicial $v_0 = (\mathbf{V}^0, \mathbf{U}^0)$, o algoritmo calcula diferentes termos da série v_t até a convergência (que é demonstrada) quando o critério J atinge um valor estacionário.

Proposição 1. *As séries $u_t = J(v_t)$ diminuem a cada iteração e convergem.*

Prova. Primeiro se mostra que as desigualdades (I) e (II)

$$\underbrace{J(\mathbf{V}^t, \mathbf{U}^t)}_{u_t} \overset{(I)}{\geq} J(\mathbf{V}^{t+1}, \mathbf{U}^t) \overset{(II)}{\geq} \underbrace{J(\mathbf{V}^{t+1}, \mathbf{U}^{t+1})}_{u_{t+1}}$$

mantém-se (isto é, a série diminui em cada iteração).

A desigualdade (I) se mantém porque

a)

$$\begin{aligned} J(\mathbf{V}^t, \mathbf{U}^t) &= \sum_{k=1}^{K}\sum_{i=1}^{n}(u_{ik}^t)^2 d^2(\mathbf{x}_i, \mathbf{v}_k^t) \\ &+ \alpha \times \sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{m=1}^{n} b_i b_m t_{im}(u_{ik}^t - u_{mk}^t)^2 d^2(\mathbf{x}_i, \mathbf{x}_m) \end{aligned}$$

b)

$$\begin{aligned} J(\mathbf{V}^{t+1}, \mathbf{U}^{t}) &= \sum_{k=1}^{K}\sum_{i=1}^{n}(u_{ik}^{t})^2 d^2(\mathbf{x}_i, \mathbf{v}_k^{t+1}) \\ &+ \alpha \times \sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{m=1}^{n} b_i b_m t_{im}(u_{ik}^{t} - u_{mk}^{t})^2 d^2(\mathbf{x}_i, \mathbf{x}_m) \end{aligned}$$

e

$$\begin{aligned} \mathbf{V}^{(t+1)} &= (\mathbf{v}_1^{(t+1)}, \ldots, \mathbf{v}_K^{(t+1)}) = \\ &= \underbrace{argmin}_{\mathbf{V}=(\mathbf{v}_1,\ldots,\mathbf{v}_K)\in \mathbb{I}\mathbf{L}^K} \sum_{k=1}^{K}\sum_{i=1}^{n}(u_{ik}^{t})^2 d^2(\mathbf{x}_i, \mathbf{v}_k) + \\ &+ \alpha \times \sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{m=1}^{n} b_i b_m t_{im}(u_{ik}^{t} - u_{mk}^{t})^2 d^2(\mathbf{x}_i, \mathbf{x}_m) \end{aligned}$$

Além disso, a desigualdade (II) também se mantém porque

$$\begin{aligned} J(\mathbf{V}^{t+1}, \mathbf{U}^{t+1}) &= \sum_{k=1}^{K}\sum_{i=1}^{n}(u_{ik}^{t+1})^2 d^2(\mathbf{x}_i, \mathbf{v}_k^{t+1}) \\ &+ \alpha \times \sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{m=1}^{n} b_i b_m t_{im}(u_{ik}^{t+1} - u_{mk}^{t+1})^2 d^2(\mathbf{x}_i, \mathbf{x}_m) \end{aligned}$$

e

$$\begin{aligned} \mathbf{U}^{t+1} &= (\mathbf{u}_1^{t+1}, \ldots, \mathbf{u}_K^{t+1}) \\ &= \underbrace{argmin}_{\mathbf{U}=(\mathbf{u}_1,\ldots,\mathbf{u}_K)\in \mathbb{I}\mathbf{U}^n} \sum_{k=1}^{K}\sum_{i=1}^{n}(u_{ik})^2 d^2(\mathbf{x}_i, \mathbf{v}_k^{t+1}) + \\ &+ \alpha \times \sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{m=1}^{n} b_i b_m t_{im}(u_{ik} - u_{mk}^{t})^2 d^2(\mathbf{x}_i, \mathbf{x}_m) \end{aligned}$$

Finalmente, porque as séries u_t diminuem e são limitadas por $(J(v_t) \geq 0)$, converge.

Proposição 2. *As séries* $v_t = (\boldsymbol{V}^t, \boldsymbol{U}^t)$ *convergem.*

Prova. Assuma que a estacionariedade das séries u_t é alcançada na iteração $t = T$. Então, temos que $u_T = u_{T+1}$ e, em seguida,

Partindo de $J(v_T) = J(v_{T+1})$, temos $J(\mathbf{V}^T, \mathbf{U}^T) = J(\mathbf{V}^{T+1}, \mathbf{U}^{T+1})$ e essa igualdade, de acordo com proposição 1, pode ser reescrita como igualdades (I) e (II):

$$J(\mathbf{V}^T, \mathbf{U}^T) \overset{I}{=} J(\mathbf{V}^{T+1}, \mathbf{U}^T) \overset{II}{=} J(\mathbf{V}^{T+1}, \mathbf{U}^{T+1})$$

Da primeira igualdade (I), temos que $\mathbf{V}^T = \mathbf{V}^{T+1}$ porque $\mathbf{V}$ é única minimizando J quando a partição *fuzzy* representada por $\mathbf{U}^T$ é mantida fixa. A partir da segunda igualdade (II), temos que $\mathbf{U}^T = \mathbf{U}^{T+1}$, porque $\mathbf{U}$ é única minimizando J quando o vetor de protótipos $\mathbf{V}^{T+1}$ é mantido fixo.

Finalmente, concluímos que $v_T = v_{T+1}$. Esta conclusão é válida para todos os $t \geq T$ e $v_t = v_T, \forall t \geq T$ e segue até que as séries v_t convergem.

3.2.8 Custo computacional

Nesta seção é realizada a análise de complexidade dos algoritmos de agrupamento semissupervisionados apresentados neste capítulo. Conforme apresentado na Subseção 3.2.6, os algoritmos de agrupamento semissupervisionados realizam realocação de duas etapas com dois laços: (i) um laço externo de tamanho $T = Max$ e (ii) um laço aninhado: este otimizando J (ou seja, dois passos para computar $\mathbf{V}$ e $\mathbf{U}$). A complexidade desses algoritmos é calculada pelo valor da complexidade máxima das matrizes $\mathbf{U}$ e $\mathbf{V}$. E depois, o laço externo de tamanho T é considerado no cálculo da complexidade.

Os algoritmos Isodata-PS (PEDRYCZ e WALETZKY, 1997), sSSC (ENDO et al., 2009) e ESSC (ENDO et al., 2009), podem ser analisados da mesma maneira, pois eles possuem complexidade $O(Kn)$ para calcular as matrizes $\mathbf{U}$ e $\mathbf{V}$, onde K é número de grupos e n é o número de exemplos. A complexidade é desses algoritmos é $Max(O(Kn), O(Kn))$ resultando em uma complexidade $O(Kn)$ para esta etapa. Por último, levando em conta o laço externo de tamanho T, a complexidade final é $O(TKn)$, sendo linear para cada um desses parâmetros.

Em relação à complexidade de tempo do algoritmo NebFuzz, este algoritmo de agrupamento semissupervisionado também é um algoritmo de realocação de duas etapas com dois laços. A complexidade deste algoritmo é $Max(O(Kn), O(Kn^2))$ para o cálculo das matrizes $\mathbf{V}$ e $\mathbf{U}$ respectivamente, o que produz $O(Kn^2)$. Note que as fórmulas são divididos em sub-fórmulas. Levando em conta o laço externo, a complexidade de tempo total é $O(TKn^2))$. Isto significa que o tempo de complexidade de NebFuzz é linear em ambos, o número de iterações, e no número de grupos e quadrática no número de objetos.

O algoritmo BSSC possui um laço a extra em relação aos outros algoritmos para o cálculo da matriz $\tilde{\mathbf{U}}$. Uma análise desse algoritmo foi realizada no artigo de Bouchachia e Pedrycz (BOUCHACHIA e PEDRYCZ, 2006a). Assim, no algoritmo BSSC, temos dois laços internos de tamanho L_1 e L_2 respectivamente e um laço externo de tamanho T. O primeiro laço interno possui complexidade $O(L_1Kn)$, o segundo laço externo possui complexidade $Max(O(L_2Kn), O(L_2Hn))$, onde H é o número de Classes. Sabendo que $H < K$, a complexidade é $O(L_2Kn)$. Agora, levando em conta o tamanho do laço

externo, temos que a complexidade é $Max(O(TL_1Kn), O(TL_2Kn))$. Note que a complexidade escolhida é a que for maior entre L_1 e L_2, então deixe que L seja $Max(L_1, L2)$, a complexidade do algoritmo BSSC é $O(TLKn)$. Note que a complexidade é linear para cada um desses parâmetros. A Tabela 14 apresenta um resumo da complexidade para cada um dos algoritmos.

Tabela 14 – Complexidade de tempo.

Algoritmo	Complexidade
BSSC	$O(TLKn)$
sSSC	$O(TKn)$
ESSC	$O(TKn)$
Isodata-PS	$O(TKn)$
NebFuzz	$O(TKn^2)$

3.2.9 Resumo dos algoritmos de agrupamento semissupervisionados

Todos os algoritmos explicados nesta seção se baseiam no algoritmo clássico de agrupamento *fuzzy*, o *Fuzzy C-Means* (FCM) (BEZDEK, 1981). A maioria deles adiciona um segundo termo à função objetivo do algoritmo FCM que possui como objetivo otimizar a função levando em conta os rótulos conhecidos dos exemplos das bases de dados. Apenas o algoritmo sSSC funciona um pouco diferente dos demais, pois nele, o termo que utiliza a informação do rótulo é adicionada ao próprio termo não supervisionado do FCM.

No algoritmo Isodata-PS, a otimização do termo supervisionado é realizada pela maior penalização caso um grau de pertinência seja pequeno em relação a classe que este padrão pertença. Assim, é realizada a diferença deste padrão a 1 e a multiplicação pela distância entre este padrão e o protótipo da classe ao qual ele pertence. No algoritmo BSSC, a diferença para a otimização é entre o grau de pertinência e uma segunda otimização que leva em conta que uma classe pode representar mais de um grupo. Esta diferença é multiplicada pela distância entre o padrão e o protótipo do grupo. Para o algoritmo sSSC, a otimização é realizada forçando um item ter o valor próximo de 1, quando este pertencer ao grupo que representa à classe que ele pertence. No algoritmo ESSC, caso o padrão pertença à classe representada pelo grupo, o valor da entropia é baixo, caso contrário, esse valor é alto. Se o grupo for formado apenas por exemplos da mesma classe, a entropia é baixa, o algoritmo minimiza essa função. No algoritmo NebFuzz, a otimização é realizada pela diferença entre os graus de pertinência entre padrões que pertencem a uma mesma classe multiplicado pela distância entre esses dois padrões. Esta otimização parte da premissa que dois padrões que estão na mesma classe possuem graus de pertinência parecidos à todos os grupos.

4 NOVOS ALGORITMOS DE AGRUPAMENTO SEMISSUPERVISIONADO COM DISTÂNCIAS ADAPTATIVAS

Neste capítulo, os algoritmos de agrupamentos semissupervisionados com ponderação automática de variáveis propostos nesta tese são descritos.

4.1 ALGORITMOS SEMISSUPERVISIONADOS COM DISTÂNCIAS ADAPTATIVAS

Em capítulos anteriores, foram apresentadas duas técnicas importantes para melhorar o processo de agrupamento padrão: distâncias adaptativas e aprendizagem semissupervisionada. Geralmente, essas abordagens superam em desempenho as abordagens não supervisionadas quando existe alguma informação prévia. Dessa maneira, existem algoritmos que utilizam ambas as abordagens para melhorar o desempenho de algoritmos de agrupamento. Por exemplo, o primeiro algoritmo que utilizou uma distância adaptativa num algoritmo de agrupamento semissupervisionado encontra-se descrito no trabalho de Bouchachia e Pedrycz (BOUCHACHIA e PEDRYCZ, 2006b). Eles exploraram medidas de distância adaptativas com ponderação local de variáveis e uma distância baseada em *kernel*. Kong e Wang (KONG e WANG, 2009) utilizaram uma distância adaptativa com ponderação local de variáveis com restrição da soma dos pesos das variáveis no algoritmo de agrupamento semissupervisionado proposto por (PEDRYCZ e WALETZKY, 1997). Yin *et al.* (YIN; SHU; HUANG, 2012) introduziram um novo algoritmo, chamado nesse trabalho de ESSC-SMUC, que utiliza uma distância adaptativa global à técnica de entropia na função objetivo do agrupamento *fuzzy* semissupervisionado, utilizando como inspiração o algoritmo GK (GUSTAFSON e KESSEL, 1979).

Este trabalho realiza um estudo e propostas de utilização das duas abordagens, semissupervisionada e distância adaptativa, para melhorar o desempenho de alguns algoritmos na tarefa de agrupamento. São propostos alguns novos algoritmos de agrupamento semissupervisionado baseados no *Fuzzy C-Means*, relacionados com o algoritmo GK (GUSTAFSON e KESSEL, 1979), que utilizam (uma grande quantidade de) exemplos sem rótulo e (uma pequena quantidade de) exemplos rotulados, guiados por uma distância adaptativa, resultando em diferentes formatos de grupos, além do formato de grupo esférico resultado do uso da distância Euclidiana clássica. Este tipo de algoritmo de agrupamento semissupervisionado basicamente adiciona um termo supervisionado para a função objetivo original do algoritmo de agrupamento GK. A distância adaptativa utilizada neste trabalho pode ser definida por uma matriz diagonal onde as distâncias adaptativas quadráticas dos grupos são definidas por uma matriz *fuzzy* de covariância. Estes algoritmos consideram a medida adaptativa local e global. A primeira é ponderada por vetores de pesos diferentes para cada grupo, enquanto a segunda é ponderada por vetores

de pesos iguais para todos os grupos. Além disso, a derivação das expressões que calculam os vetores de pesos precisam ter restrições para que seja possível utilizar o multiplicador de lagrange para otimizar as funções objetivos dos algoritmos de agrupamento semissupervisionados. Nesse trabalho foram levadas em conta dois tipos de restrições. A primeira, assume que a soma dos pesos das variáveis tem que ser igual a um, a segunda, assume que o produto dos pesos das variáveis tem que ser igual a um. Essas restrições foram motivadas pelo trabalho de Gustafson e Kessel (GUSTAFSON e KESSEL, 1979), que é baseado na distância adaptativa quadrada definida por uma matriz simétrica definida positiva $\mathbf{A}_k$, associada pelo grupo C_k $(k = 1, \ldots, K)$, sob $det(\mathbf{A}_k) = 1$.

Nesta seção, são detalhados os algoritmos de agrupamento *fuzzy* semissupervisionados com distâncias adaptativas propostos nesta tese, com exceção apenas dos algoritmos BSSC-LP (BOUCHACHIA e PEDRYCZ, 2006b) e Isodata-PS-LS (KONG e WANG, 2009) que foram propostos anteriormente por outros autores. Após apresentar os algoritmos propostos, é realizado um estudo das propriedades de convergência e complexidade dos algoritmos propostos.

4.1.1 NebFuzz com distâncias adaptativas

Nesta seção, apresentamos em detalhes novos algoritmos com ponderação automática de variáveis baseados no algoritmo NebFuzz (MACARIO e DE CARVALHO, 2010) apresentado na Seção 3.2.5.

O algoritmo semissupervisionado NebFuzz assume que os objetos na mesma classe possuem um alto grau de pertinência e seus graus de pertinência para um dado grupo tem que ser similar. Baseado nessas suposições, o termo supervisionado do algoritmo NebFuzz é otimizado nessa direção.

Da mesma forma que o algoritmo GK, este algoritmo procura por uma partição *fuzzy* em K grupos e seus protótipos correspondentes que otimizam o critério de adequação (função objetivo) medindo o acoplamento entre os grupos *fuzzy* e seus protótipos. A função objetivo é

$$\begin{aligned} J(\mathbf{U}, \mathbf{A_1}, \ldots, \mathbf{A_k}, \mathbf{V}) &= \sum_{k=1}^{K}\sum_{i=1}^{n}(u_{ik})^2 d^2_{\mathbf{A}_k}(\mathbf{x}_i, \mathbf{v}_k) \\ &+ \alpha \sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{m=1}^{n} b_i b_m t_{im}(u_{ik} - u_{mk})^2 d^2_{\mathbf{A}_k}(\mathbf{x}_i, \mathbf{x}_m) \end{aligned} \tag{4.1}$$

onde

- $\mathbf{U}$, $\mathbf{V}$ são como já descritos na Seção 3.2.1;

- $\mathbf{A_1}, \ldots, \mathbf{A_k}$ são como descritos na Seção 2.3;

- $t_{im} = 1$ se os objetos e_i e e_m são da mesma classe a priori; $t_{im} = 0$, caso contrário;

- $d^2_{\mathbf{A}_k}(\mathbf{x}_i, \mathbf{v}_k)$ é a distância quadrática euclidiana adaptativa entre as descrições do exemplo $\mathbf{e}_i$ e do protótipo $\mathbf{v}_k$:

$$d^2_{\mathbf{A}_k}(\mathbf{x}_i, \mathbf{v}_k) = (\mathbf{x}_i - \mathbf{v}_k)^T \mathbf{A}_k (\mathbf{x}_i - \mathbf{v}_k) = \sum_{j=1}^{p} \lambda_{kj} (\mathbf{x}_{ij} - \mathbf{v}_{kj})^2; \tag{4.2}$$

- $d^2_{\mathbf{A}_k}(\mathbf{x}_i, \mathbf{x}_m)$ é a distância quadrática euclidiana adaptativa entre as descrições dos exemplos $\mathbf{e}_i$ e $\mathbf{e}_m$:

$$d^2_{\mathbf{A}_k}(\mathbf{x}_i, \mathbf{x}_m) = (\mathbf{x}_i - \mathbf{x}_m)^T \mathbf{A}_k (\mathbf{x}_i - \mathbf{x}_m) = \sum_{j=1}^{p} \lambda_{kj} (\mathbf{x}_{ij} - \mathbf{x}_{mj})^2. \tag{4.3}$$

Dado o problema de otimização da função objetivo (4.1) levando-se em conta $(\mathbf{U}, \mathbf{A_1}, \ldots, \mathbf{A_K}, \mathbf{V})$, obtemos as soluções ótimas de u_{ik} e λ_{jk}, com as restrições (2.2). Definimos a função de Lagrange L de (4.1) como segue:

$$\begin{aligned} L(\mathbf{U}, \mathbf{A_1}, \ldots, \mathbf{A_K}, \mathbf{V}, \gamma) &= \sum_{k=1}^{K} \sum_{i=1}^{n} (u_{ik})^2 d^2_{\mathbf{A}_k}(\mathbf{x}_i, \mathbf{v}_k) \\ &+ \alpha \sum_{k=1}^{K} \sum_{i=1}^{n} \sum_{m=1}^{n} b_i b_m t_{im} (u_{ik} - u_{mk})^2 d^2_{\mathbf{A}_k}(\mathbf{x}_i, \mathbf{x}_m) \\ &- \sum_{k=1}^{K} \gamma_k \left(\prod_{j=1}^{p} \lambda_{kj} - 1 \right) \end{aligned} \tag{4.4}$$

onde $\gamma = [\gamma_k] = [\gamma_1, \ldots, \gamma_K]$ são multiplicadores de lagrange.

O algoritmo NebFuzz com distância adaptativa é um algoritmo iterativo com três etapas (de representação, de ponderação e de alocação), executado até que um critério de parada seja satisfeito.

Na etapa de representação, o grau de pertinência de cada padrão e_i em cada grupo C_k e as matrizes de peso $\mathbf{A_k}$ são mantidas fixas. Os protótipos $\mathbf{v}_k$ dos grupos $C_k (k = 1, \ldots, K)$, que minimizam o critério de agrupamento J, são atualizados de acordo com

$$\mathbf{v}_k = \frac{\sum_{i=1}^{n} (u_{ik})^2 \mathbf{x}_i}{\sum_{i=1}^{n} (u_{ik})^2} \tag{4.5}$$

Na etapa de alocação, os protótipos $\mathbf{v}_k$ correspondentes aos grupos $C_k (k = 1, \ldots, K)$ e as matrizes $\mathbf{A}_k (k = 1, \ldots, K)$ são fixados. O grau de pertinência $u_{ik} (i = 1, \ldots, n)$ de cada objeto e_i em cada grupo $C_k (k = 1, \ldots, K)$, que minimizam o critério de agrupamento

J sob as restrições (2.2), são atualizados de acordo com a seguinte expressão

$$u_{ik} = \frac{1 + \alpha b_i \left[\left(\sum_{l=1}^{K} \frac{\sum_{m=1}^{n} b_m t_{im} \left(u_{mk} d^2_{\mathbf{A_k}}(\mathbf{x}_i, \mathbf{x}_m) - u_{ml} d^2_{\mathbf{A_l}}(\mathbf{x}_i, \mathbf{x}_m) \right)}{\left(d^2_{\mathbf{A_l}}(\mathbf{x}_i, \mathbf{v}_l) \right)^2 + \alpha b_i \sum_{m=1}^{n} b_m t_{im} d^2_{\mathbf{A_l}}(\mathbf{x}_i, \mathbf{x}_m)} \right) \right]}{\sum_{l=1}^{K} \left[\frac{\left(d^2_{\mathbf{A_k}}(\mathbf{x}_i, \mathbf{v}_k) \right)^2 + \alpha b_i \sum_{m=1}^{n} b_m t_{im} d^2_{\mathbf{A_k}}(\mathbf{x}_i, \mathbf{x}_m)}{\left(d^2_{\mathbf{A_l}}(\mathbf{x}_i, \mathbf{v}_l) \right)^2 + \alpha b_i \sum_{m=1}^{n} b_m t_{im} d^2_{\mathbf{A_l}}(\mathbf{x}_i, \mathbf{x}_m)} \right]} \tag{4.6}$$

A etapa de ponderação calcula os componentes da matriz de vetores de pesos $\mathbf{A}_k (k = 1, ..., K)$. Os pesos da matriz A_k são intimamente relacionados à distância adaptativa do algoritmo semissupervisionado de agrupamento e o formato dos grupos que são criados, como mostrado na Seção 2.3. Nesta tese, foi proposta a utilização de quatro tipos de distâncias adaptativas apresentadas a seguir:

(a) NebFuzz-LP: algoritmo NebFuzz com ponderação local de variáveis em que a restrição do produto dos pesos das variáveis tem que ser igual a um.

$$d^2_{\mathbf{A}_k}(\mathbf{x}_i, \mathbf{x}_m) = \sum_{j=1}^{p} \lambda_{kj} (x_{ij} - x_{mj})^2 \tag{4.7}$$

$$\prod_{j=1}^{p} \lambda_{kj} = 1; \lambda_{kj} > 0 \tag{4.8}$$

Queremos minimizar J com respeito a $\lambda_{kj} (k = 1, \ldots, K, j = 1, \ldots, p)$ sob as restrições (4.8). Com o grau de pertinência $u_{ik} (i = 1, \ldots, n, k = 1, \ldots, K)$, os protótipos $v_k (k = 1, \ldots, K)$, fixos, podemos reescrever J da forma

$$\begin{aligned} J(\mathbf{A}_1, \ldots, \mathbf{A}_k) = \sum_{k=1}^{K} J_k(\mathbf{A}_k) &= \sum_{k=1}^{K} \sum_{j=1}^{p} \lambda_{kj} \sum_{i=1}^{n} u_{ik}^2 (x_{ij} - v_{kj})^2 \\ &+ \alpha \sum_{i=1}^{n} \sum_{m=1}^{n} b_i b_m t_{im} (u_{ik} - u_{mk})^2 (x_{ij} - x_{mj})^2 \end{aligned}$$

onde

$$J_k(\mathbf{A}_k) = J_k(\lambda_{k1}, \ldots, \lambda_{kp}) = \sum_{j=1}^{p} \lambda_{kj} J_{kj}$$

com

$$\begin{aligned} J_{kj} &= \sum_{i=1}^{n} u_{ik}^2 (\mathbf{x}_{ij} - \mathbf{v}_{kj})^2 \\ &+ \alpha \sum_{i=1}^{n} \sum_{m=1}^{n} b_i b_m t_{im} (u_{ik} - u_{mk})^2 (x_{ij} - x_{mj})^2 \end{aligned}$$

Como o critério J é aditivo, o problema se torna minimizar $J_k(k = 1, \ldots, K)$. Seja $\gamma_k(\lambda_{k1} \ldots, \lambda_{kp}) = \prod_{j=1}^{p} \lambda_{kj} \ \forall k$. Agora, queremos determinar os extremos de $J_k(\lambda_{kj}, \ldots, \lambda_{kp})$ com a restrição $\gamma_k(\lambda_{k1} \ldots, \lambda_{kp}) = 0$. Para tanto, devemos aplicar o método dos multiplicadores de Lagrange para resolver o seguinte sistema

$$\nabla J(\lambda_{k1}, \ldots, \lambda_{kp}) = \mu \nabla \gamma_k(\lambda_{k1}, \ldots, \lambda_{kp})$$

Depois, para $k = 1, \ldots, K$ e $j = 1, \ldots, p$, temos

$$\frac{\nabla J_k(\lambda_{k1}, \ldots, \lambda_{kp})}{\nabla \lambda_{kj}} = \mu \frac{\nabla \gamma_k(\lambda_{k1}, \ldots, \lambda_{kp})}{\nabla \lambda_{kj}} \ \Rightarrow \ J = \frac{\mu}{\lambda_{kj}}$$

como sabemos que $\prod_{l=1}^{p} \lambda_{kl} = 1$, temos

$$\prod_{l=1}^{p} \frac{\mu}{\lambda_{kl}} = 1 \ \Rightarrow \ \frac{\mu^p}{\prod_{l=1}^{p} J} = 1 \ \Rightarrow \ \mu = \left(\prod_{l=1}^{p} J \right)^{\frac{1}{p}}$$

e segue até que um valor extremo de J é alcançado quando $\lambda_{kj} = \frac{\prod_{l=1}^{p} J_{kl}}{J_{kj}}$ com

$$\lambda_{kj} = \frac{\left\{ \prod_{h=1}^{p} \left(\sum_{i=1}^{n} \left[(u_{ik})^2 (x_{ih} - v_{kh})^2 + \sum_{m=1}^{n} w_{im} (x_{ih} - x_{mh})^2 \right] \right) \right\}^{\frac{1}{p}}}{\sum_{i=1}^{n} \left[(u_{ik})^2 (x_{iz} - v_{kj})^2 + \sum_{m=1}^{n} w_{im} (x_{ij} - x_{mj})^2 \right]} \tag{4.9}$$

onde $w_{im} = \alpha b_i b_m t_{im} (u_{ik} - u_{mk})^2$.

Nesse tipo de distância é importante notar que o conjunto de variáveis que são relevantes ou irrelevantes, pode ser diferente para cada um dos grupos. Neste trabalho, o algoritmo adaptativo local semissupervisionado de agrupamento considerando a distância dada pela Equação (4.7) é chamado de NebFuzz-LP. Este algoritmo foi apresentado em (MACARIO e DE CARVALHO, 2012a).

(b) NebFuzz-LS: algoritmo NebFuzz com ponderação local de variáveis em que a soma dos pesos de cada variável de cada grupo tem que ser igual a um.

$$d^2_{\mathbf{A}_k}(\mathbf{x}_i, \mathbf{x}_m) = \sum_{j=1}^{p} (\lambda_{kj})^s (x_{ij} - x_{mj})^2 \tag{4.10}$$

sujeito à

$$\sum_{j=1}^{p} \lambda_{kj} = 1; \lambda_{kj} > 0 \tag{4.11}$$

em que o vetor de pesos considerando o grupo C_k, e $s \in (1, \infty)$ é um parâmetro que controla o grau de influência do peso em cada variável, em cada grupo. Desse modo, se s é grande o suficiente, então todas as variáveis têm a mesma importância para todos os grupos; entretanto, se $s \to 1$, então a influência dos pesos das variáveis é máxima.

Agora, minimizando J com respeito a $\lambda_{kj}(k = 1, \ldots, K, j = 1, \ldots, p)$ sob as restrições (4.11). Com o grau de pertinência $u_{ik}(i = 1, \ldots, n, k = 1, \ldots, K)$, os protótipos $v_k(k = 1, \ldots, K)$, fixos, podemos reescrever J da forma

$$\begin{aligned} J(\mathbf{A}_1, \ldots, \mathbf{A}_k) = \sum_{k=1}^{K} J_k(\mathbf{A}_k) &= \sum_{k=1}^{K}\sum_{j=1}^{p} \lambda_{kj} \sum_{i=1}^{n} u_{ik}^2 (\mathbf{x}_{ij} - \mathbf{v}_{kj})^2 \\ &+ \alpha \sum_{i=1}^{n}\sum_{m=1}^{n} b_i b_m t_{im} (u_{ik} - u_{mk})^2 (x_{ij} - x_{mj})^2 \end{aligned}$$

onde

$$J_k(\mathbf{A}_k) = J_k(\lambda_{k1}, \ldots, \lambda_{kp}) = \sum_{j=1}^{p} \lambda_{kj}^s J_{kj}$$

com

$$\begin{aligned} J_{kj} &= \sum_{i=1}^{n} u_{ik}^2 (\mathbf{x}_{ij} - \mathbf{v}_{kj})^2 \\ &+ \alpha \sum_{i=1}^{n}\sum_{m=1}^{n} b_i b_m t_{im} (u_{ik} - u_{mk})^2 (x_{ij} - x_{mj})^2 \end{aligned}$$

Como o critério J é aditivo, o problema se torna minimizar $J_k(k = 1, \ldots, K)$. Seja $\gamma_k(\lambda_{k1} \ldots, \lambda_{kp}) = \sum_{j=1}^{p} \lambda_{kj}$ $\forall k$. Agora, queremos determinar os extremos de $J_k(\lambda_{kj}, \ldots, \lambda_{kp})$ com à restrição $\gamma_k(\lambda_{k1} \ldots, \lambda_{kp}) = 0$. Para tanto, devemos aplicar o método dos multiplicadores de Lagrange para resolver o seguinte sistema

$$\nabla J(\lambda_{k1}, \ldots, \lambda_{kp}) = \mu \nabla \gamma_k(\lambda_{k1}, \ldots, \lambda_{kp}).$$

Depois, para $k = 1, \ldots, K$ e $j = 1, \ldots, p$, temos

$$\begin{aligned} \frac{\nabla J(\lambda_{k1}, \ldots, \lambda_{kp})}{\nabla \lambda_{kj}} &= \mu \frac{\nabla \gamma_k(\lambda_{k1}, \ldots, \lambda_{kp})}{\nabla \lambda_{kj}}, \\ s\lambda_{kj}^{s-1} J_{kj} &= \mu, \\ \lambda_{kj} &= \left(\frac{\mu}{s}\right)^{\frac{1}{s-1}} . \frac{1}{(J_{kj})^{\frac{1}{s-1}}}. \end{aligned}$$

Como sabemos que $\sum_{l=1}^{p} \lambda_{kl} = 1$, temos

$$\sum_{l=1}^{p} \left[\left(\frac{\mu}{s} \right)^{\frac{1}{s-1}} . \frac{1}{(J_{kl})^{\frac{1}{s-1}}} \right] = 1.$$

e segue um valor extremo de J é alcançado quando $\lambda_{kj} = \left[\sum_{l=1}^{p} \left(\frac{J_{kj}}{J_{kl}} \right)^{\frac{1}{s-1}} \right]^{-1}$ pela Equação 4.12:

$$\lambda_{kj} = \left\{ \sum_{h=1}^{p} \left(\frac{\sum_{i=1}^{n} \left[(u_{ik})^2 (x_{ij} - v_{kj})^2 + \sum_{m=1}^{n} w_{im} (x_{ij} - x_{mj})^2 \right]}{\sum_{i=1}^{n} \left[(u_{ik})^2 (x_{ih} - v_{kh})^2 + \sum_{m=1}^{n} w_{im} (x_{ih} - x_{mh})^2 \right]} \right)^{\frac{1}{s-1}} \right\}^{-1} \tag{4.12}$$

onde $w_{im} = \alpha b_i b_m t_{im} (u_{ik} - u_{mk})^2$ e $s = 2$ neste trabalho.

Também é importante notar que nesse caso o conjunto de variáveis que são relevantes e irrelevantes pode ser diferente para cada grupo. Neste trabalho, o algoritmo adaptativo local semissupervisionado de agrupamento considerando a distância dada pela Equação (4.10) é chamado de NebFuzz-LS.

(c) NebFuzz-GP: algoritmo NebFuzz com ponderação global de variáveis em que o produto dos pesos de cada variável tem que ser igual a um.

$$d^2_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_m) = \sum_{j=1}^{p} \lambda_j (x_{ij} - x_{mj})^2 \tag{4.13}$$

em que

$$\prod_{j=1}^{p} \lambda_j = 1; \lambda_j > 0 \tag{4.14}$$

Agora, queremos minimizar J com respeito a $\lambda_j (j = 1, \ldots, p)$ sob as restrições (4.14). Com o grau de pertinência $u_{ik} (i = 1, \ldots, n, k = 1, \ldots, K)$, os protótipos $v_k (k = 1, \ldots, K)$, fixos, podemos reescrever J da forma

$$\begin{aligned} J(\mathbf{A}) = J(\mathbf{A}) &= \sum_{j=1}^{p} \lambda_j \sum_{k=1}^{K} \sum_{i=1}^{n} u_{ik}^2 (\mathbf{x}_{ij} - \mathbf{v}_{kj})^2 \\ &+ \alpha \sum_{i=1}^{n} \sum_{m=1}^{n} b_i b_m t_{im} (u_{ik} - u_{mk})^2 (x_{ij} - x_{mj})^2 \end{aligned}$$

onde

$$J(\mathbf{A}) = J(\lambda_1, \ldots, \lambda_p) = \sum_{j=1}^{p} \lambda_j J_j$$

onde

$$\begin{aligned} J_j &= \sum_{k=1}^{K}\sum_{i=1}^{n} u_{ik}^2 (\mathbf{x}_{ij} - \mathbf{v}_{kj})^2 \\ &+ \alpha \sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{m=1}^{n} b_i b_m t_{im} (u_{ik} - u_{mk})^2 (x_{ij} - x_{mj})^2 \end{aligned}$$

Agora o problema é minimizar J. Seja $\gamma(\lambda_1 \ldots, \lambda_p) = \prod_{j=1}^{p} \lambda_j$. Queremos determinar os extremos de $J(\lambda_j, \ldots, \lambda_p)$ com à restrição $\gamma_k(\lambda_1 \ldots, \lambda_p) = 0$. Para tanto, devemos aplicar o método dos multiplicadores de Lagrange para resolver o seguinte sistema

$$\nabla J(\lambda_1, \ldots, \lambda_p) = \mu \nabla \gamma(\lambda_1, \ldots, \lambda_p).$$

Depois, para $j = 1, \ldots, p$, temos

$$\frac{\nabla J(\lambda_1, \ldots, \lambda_p)}{\nabla \lambda_j} = \mu \frac{\nabla \gamma(\lambda_1, \ldots, \lambda_p)}{\nabla \lambda_j} \;\Rightarrow\; J = \frac{\mu}{\lambda_j}$$

como sabemos que $\prod_{l=1}^{p} \lambda_l = 1$, temos

$$\prod_{l=1}^{p} \frac{\mu}{\lambda_l} = 1 \;\Rightarrow\; \frac{\mu^p}{\prod_{l=1}^{p} J} = 1 \;\Rightarrow\; \mu = \left(\prod_{l=1}^{p} J \right)^{\frac{1}{p}}$$

e segue até que um valor extremo de J é alcançado quando $\lambda_j = \frac{\prod_{l=1}^{p} J_l}{J_j}$ com

$$\lambda_j = \frac{\left\{ \prod_{h=1}^{p} \left(\sum_{k=1}^{K}\sum_{i=1}^{n} \left[(u_{ik})^2 (x_{ih} - v_{kh})^2 + \sum_{m=1}^{n} w_{im} (x_{ih} - x_{mh})^2 \right] \right) \right\}^{\frac{1}{p}}}{\sum_{k=1}^{K}\sum_{i=1}^{n} \left[(u_{ik})^2 (x_{ij} - v_{kj})^2 + \sum_{m=1}^{n} w_{im} (x_{ij} - x_{mj})^2 \right]} \tag{4.15}$$

onde $w_{im} = \alpha b_i b_m t_{im} (u_{ik} - u_{mk})^2$.

Nesse tipo de distância é importante notar que o conjunto de variáveis que são mais importantes ou não são importantes são sempre as mesmas para todos os grupos. Neste trabalho, o algoritmo adaptativo global semissupervisionado de agrupamento considerando a distância dada pela Equação (4.13) é chamado de NebFuzz-GP.

(d) NebFuzz-GS: algoritmo NebFuzz com ponderação global de variáveis em que a soma dos pesos de cada variável tem que ser igual a um.

$$d^2_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_m) = \sum_{j=1}^{p} (\lambda_j)^s (x_{ij} - x_{mj})^2 \tag{4.16}$$

sujeito à

$$\sum_{j=1}^{p} \lambda_j = 1; \lambda_j > 0 \tag{4.17}$$

em que o vetor de pesos considerando o grupo C_k, e $s \, \epsilon \, (1, \infty)$ é um parâmetro que controla o grau de influência do peso em cada variável, em cada grupo. Desse modo, se s é grande o suficiente, então todas as variáveis têm a mesma importância para todos os grupos; entretanto, se $s \to 1$, então a influência dos pesos das variáveis é máxima.

Agora, queremos minimizar J com respeito a $\lambda_j (j = 1, \ldots, p)$ sob as restrições (4.17). Com o grau de pertinência $u_{ik} (i = 1, \ldots, n, k = 1, \ldots, K)$, os protótipos $v_k (k = 1, \ldots, K)$, fixos, podemos reescrever J da forma

$$\begin{aligned} J(\mathbf{A}) = J(\mathbf{A}) &= \sum_{j=1}^{p} \lambda_j \sum_{k=1}^{K} \sum_{i=1}^{n} u_{ik}^2 (\mathbf{x}_{ij} - \mathbf{v}_{kj})^2 \\ &+ \alpha \sum_{k=1}^{K} \sum_{i=1}^{n} \sum_{m=1}^{n} b_i b_m t_{im} (u_{ik} - u_{mk})^2 (x_{ij} - x_{mj})^2 \end{aligned}$$

onde

$$J(\mathbf{A}) = J(\lambda_1, \ldots, \lambda_p) = \sum_{j=1}^{p} \lambda_j^s J_j$$

com

$$\begin{aligned} J_j &= \sum_{k=1}^{K} \sum_{i=1}^{n} u_{ik}^2 (\mathbf{x}_{ij} - \mathbf{v}_{kj})^2 \\ &+ \alpha \sum_{k=1}^{K} \sum_{i=1}^{n} \sum_{m=1}^{n} b_i b_m t_{im} (u_{ik} - u_{mk})^2 (x_{ij} - x_{mj})^2 \end{aligned}$$

Agora, o problema se torna minimizar J. Seja $\gamma(\lambda_1 \ldots, \lambda_p) = \sum_{j=1}^{p} \lambda_j$. Queremos determinar os extremos de $J(\lambda_j, \ldots, \lambda_p)$ com à restrição $\gamma(\lambda_1 \ldots, \lambda_p) = 0$. Para tanto, devemos aplicar o método dos multiplicadores de Lagrange para resolver o seguinte sistema

$$\nabla J(\lambda_1, \ldots, \lambda_p) = \mu \nabla \gamma(\lambda_1, \ldots, \lambda_p).$$

Depois, para $k = 1, \ldots, K$ e $j = 1, \ldots, p$, temos

$$\begin{aligned} \frac{\nabla J(\lambda_1, \ldots, \lambda_p)}{\nabla \lambda_j} &= \mu \frac{\nabla \gamma(\lambda_1, \ldots, \lambda_p)}{\nabla \lambda_j}, \\ s\lambda_j^{s-1} J_{kj} &= \mu, \\ \lambda_j &= \left(\frac{\mu}{s}\right)^{\frac{1}{s-1}} . \frac{1}{(J_j)^{\frac{1}{s-1}}}. \end{aligned}$$

Como sabemos que $\sum_{l=1}^{p} \lambda_l = 1$, temos

$$\sum_{l=1}^{p} \left(\frac{\mu}{s}\right)^{\frac{1}{s-1}} . \frac{1}{(J_l)^{\frac{1}{s-1}}} = 1.$$

e segue até um valor extremo de J é alcançado quando $\lambda_j = \left[\sum_{l=1}^{p} \left(\frac{J_j}{J_l}\right)^{\frac{1}{s-1}}\right]^{-1}$ pela Equação 4.18:

$$\lambda_j = \left\{ \sum_{h=1}^{p} \left(\frac{\sum_{k=1}^{K} \sum_{i=1}^{n} \left[(u_{ik})^2 (x_{ij} - v_{kj})^2 + \sum_{m=1}^{n} w_{im} (x_{ij} - x_{mj})^2 \right]}{\sum_{k=1}^{K} \sum_{i=1}^{n} \left[(u_{ik})^2 (x_{ih} - v_{kh})^2 + \sum_{m=1}^{n} w_{im} (x_{ih} - x_{mh})^2 \right]} \right)^{\frac{1}{s-1}} \right\}^{-1} \tag{4.18}$$

onde $w_{im} = \alpha b_i b_m t_{im} (u_{ik} - u_{mk})^2$ e geralmente $s = 2$.

Nesse tipo de distância é importante notar que o conjunto de variáveis que são mais importantes ou não são importantes são sempre as mesmas para todos os grupos. Neste trabalho, o algoritmo adaptativo global semissupervisionado de agrupamento considerando a distância dada pela Equação 4.16 é chamado de NebFuzz-GS.

O algoritmo de agrupamento semissupervisionado NebFuzz que origina esses algoritmos com distâncias adaptativas pode ser formulado ignorando a etapa de ponderação e fixando $\mathbf{A}_k = (\lambda_{k1}, \ldots, \lambda_{kp}) = (1, \ldots, 1)(k = 1, \ldots, K)$ em todas as equações apresentadas onde a matriz $\mathbf{A}_k$ é utilizada.

4.1.2 BSSC com distâncias adaptativas

Nesta seção, apresentamos os algoritmos de agrupamento semissupervisionados com distâncias adaptativas baseados no algoritmo não adaptativo (BSSC) proposto por Bouchachia e Pedrycz (BOUCHACHIA e PEDRYCZ, 2006b).

O propósito desse algoritmo é capturar estruturas de dados rotulados e não rotulados através de dois termos da função objetivo. O primeiro termo é igual à função objetivo do algoritmo não supervisionado GK (GUSTAFSON e KESSEL, 1979). O segundo termo é otimizado levando em conta os dados rotulados.

Bouchachia e Pedrycz (BOUCHACHIA e PEDRYCZ, 2006b) criaram versões adaptativas desse algoritmo explorando a distância adaptativa local com restrição de produto, e uma distância baseada em *kernel*. A função objetivo do algoritmo BSSC com distância adaptativa é:

$$\begin{aligned} J(\mathbf{U}, \tilde{\mathbf{U}}, \mathbf{A_1}, \ldots, \mathbf{A_K}, \mathbf{V}) &= \sum_{k=1}^{K}\sum_{i=1}^{n}(u_{ik})^2 d^2_{\mathbf{A}_k}(\mathbf{x}_i, \mathbf{v}_k) \\ &+ \alpha \sum_{k=1}^{K}\sum_{i=1}^{n}(u_{ik} - \tilde{u}_{ik})^2 d^2_{\mathbf{A}_k}(\mathbf{x}_i, \mathbf{v}_k) \end{aligned} \tag{4.19}$$

com

$$\begin{aligned} \tilde{u}_{ik}^{(t)} &= \tilde{u}_{ik}^{(t-1)} + \\ &+ 2\beta b_i \sum_{h=1}^{H}(f_{ih} - \sum_{C_l \in \pi_h} \tilde{u}_i^{(t-1)}) \times \begin{cases} 1 & se\ C_k\ \in\ \pi_h \\ 0 & senão \end{cases} \end{aligned} \tag{4.20}$$

onde

- $\mathbf{U}$, $\tilde{\mathbf{U}}$, $\mathbf{A_1}, \ldots, \mathbf{A_k}$,$\mathbf{V}$, b_i, α e $d^2_{\mathbf{A}_k}(\mathbf{x}_i, \mathbf{v}_k)$ são como descritos anteriormente;
- $\tilde{u}_{ik} \in [0, 1]$;
- $\mathbf{F}$ é como já descrito. Supõe-se que uma classe pode ser dividida entre vários grupos e se H é o número de classes e K o número de grupos, então $K \geq H$. Senão, cada classe P_h contém um número de grupos C_l e $\sum_{l=1}^{H} C_l = K$;
- π_h é como descrito na Seção 3.2.2
- $\beta > 0$ é a taxa de aprendizagem;

Esse algoritmo possui quatro etapas (alocação composta por duas etapas, representação e ponderação). O objetivo é minimizar a diferença entre o grau de pertinência de um objeto e_i e uma classe h, e a soma dos graus de pertinência do objeto e_i para todos os grupos pertencentes a classe h.

A etapa de alocação alterna entre o cálculo das matrizes $\mathbf{U}$ e $\tilde{\mathbf{U}}$. Na primeira etapa de alocação, a matriz $\tilde{\mathbf{U}}$, a matriz $\mathbf{V}$ de protótipos e a matriz de pesos $\mathbf{A}$ são mantidas

fixas. A função de objetivo é minimizada em relação à matriz **U** de graus de pertinência. O grau de pertinência do objeto $e_i\,(i = 1, \ldots, n)$ num grupo fuzzy $C_k\,(k = 1, \ldots, K)$ é calculado de acordo com (BOUCHACHIA e PEDRYCZ, 2006b)

$$u_{ik} = \frac{\alpha \tilde{u}_{ik}}{(1+\alpha)} + \frac{1 - \frac{\alpha}{(1+\alpha)} \sum_{l=1}^{K} \tilde{u}_{il}}{\sum_{l=1}^{K} \left(\frac{d^2_{\mathbf{A}_k}(\mathbf{x}_i, \mathbf{v}_k)}{d^2_{\mathbf{A}_l}(\mathbf{x}_i, \mathbf{v}_l)} \right)} \tag{4.21}$$

Na segunda etapa de alocação, a matriz **U**, a matriz **V** de protótipos e a matriz de pesos **A** são mantidas fixas. A função de objetivo é minimizada em relação à matriz $\tilde{\mathbf{U}}$. A função de pertinência do objeto $e_i\,(i = 1, \ldots, n)$ num grupo fuzzy $C_k\,(k = 1, \ldots, K)$ é calculado de acordo com a Equação (4.20).

Na etapa de representação, a matriz $\tilde{\mathbf{U}}$, a matriz **U** de graus de pertinência e a matriz de pesos **A** são mantidas fixas. A função objetivo é minimizada em relação ao protótipo matriz **V**. Os protótipos dos grupos $\mathbf{v}_k\,(k = 1, \ldots, K)$ são calculados de acordo com (BOUCHACHIA e PEDRYCZ, 2006b)

$$\mathbf{v}_k = \frac{\sum_{i=1}^{n} [(u_{ik})^2 + \alpha(u_{ik} - \tilde{u}_{ik})^2] \mathbf{x}_i}{\sum_{i=1}^{n} [(u_{ik})^2 + \alpha(u_{ik} - \tilde{u}_{ik})^2)]} \tag{4.22}$$

Os componentes dos vetores de peso de $\mathbf{A}_k = [\lambda_{kj}](k = 1, \ldots, K, j = 1, \ldots, p)$ são calculados utilizando quatro tipos de distâncias adaptativas de acordo com as restrições em que a soma ou produto dos pesos das variáveis tem que ser igual a um. As equações podem ser obtidas de maneira similar à explicada na Seção 4.1.1.

(a) BSSC-LP: algoritmo BSSC com ponderação local de variáveis em que o produto dos pesos de cada variável de cada grupo tem que ser igual a um.

$$\lambda_{kj} = \frac{\{\prod_{h=1}^{p} (\sum_{i=1}^{n} (u_{ik}^2 + \alpha(u_{ik} - \tilde{u}_{ik})^2)(x_{ih} - v_{kh}))\}^{\frac{1}{p}}}{\sum_{i=1}^{n} (u_{ik}^2 + \alpha(u_{ik} - \tilde{u}_{ik})^2)(x_{ij} - v_{kj})} \tag{4.23}$$

É importante notar que nesse caso o conjunto de variáveis que são importantes pode ser diferente para todos cada grupo. Neste trabalho, o algoritmo adaptativo local semissupervisionado de agrupamento considerando a distância dada pela Equação (4.23) é chamado de BSSC-LP. Este algoritmo é foi proposto por Bouchachia e Pedrycz (BOUCHACHIA e PEDRYCZ, 2006b).

(b) BSSC-LS: algoritmo BSSC com ponderação local de variáveis em que a soma dos pesos de cada variável de cada grupo tem que ser igual a um.

Neste trabalho exploramos uma nova restrição. Utilizamos a restrição em que a soma dos pesos dos componentes dos vetores de pesos de $\mathbf{A}_k = [\lambda_{kj}](k = 1, \ldots, K, j =$

$1, \ldots, p$) tem que ser igual a um. A nova Equação para calcular esses componentes é:

$$\lambda_{kj} = \left\{ \sum_{h=1}^{p} \left(\frac{\sum_{i=1}^{n} [u_{ik}^2 + \alpha(u_{ik} - \tilde{u}_{ik})^2](x_{ij} - v_{kj})^2}{\sum_{i=1}^{n} [u_{ik}^2 + \alpha(u_{ik} - \tilde{u}_{ik})^2](x_{ih} - v_{kh})^2} \right)^{\frac{1}{s-1}} \right\}^{-1} \tag{4.24}$$

Também é importante notar que nesse caso o conjunto de variáveis relevantes e irrelevantes pode ser diferente para cada grupo. Neste trabalho, o algoritmo adaptativo local semissupervisionado de agrupamento considerando a distância dada pela Equação (4.24) é chamado de BSSC-LS.

(c) BSSC-GP: algoritmo BSSC com ponderação global de variáveis em que o produto dos pesos de cada variável tem que ser igual a um.

$$\lambda_j = \frac{\left\{ \prod_{h=1}^{p} \left(\sum_{k=1}^{K} \sum_{i=1}^{n} (u_{ik}^2 + \alpha(u_{ik} - \tilde{u}_{ik})^2)(x_{ih} - v_{kj}) \right) \right\}^{\frac{1}{p}}}{\sum_{k=1}^{K} \sum_{i=1}^{n} (u_{ik}^2 + \alpha(u_{ik} - \tilde{u}_{ik})^2)(x_{ij} - v_{kh})} \tag{4.25}$$

É importante notar que nesse caso o conjunto de variáveis que são relevantes e irrelevantes é igual para todos os grupos. Neste trabalho, o algoritmo adaptativo global semissupervisionado de agrupamento considerando a distância dada pela Equação (4.25) é chamado de BSSC-GP.

(d) BSSC-GS: algoritmo BSSC com ponderação global de variáveis em que a soma dos pesos de cada variável tem que ser igual a um.

À restrição em que a soma dos componentes dos vetores de pesos da matriz $\mathbf{A} = [\lambda_j](j = 1, \ldots, p)$ tem que ser igual a um. A nova Equação para calcular esses componentes é:

$$\lambda_j = \left\{ \sum_{h=1}^{p} \left(\frac{\sum_{k=1}^{K} \sum_{i=1}^{n} [u_{ik}^2 + \alpha(u_{ik} - \tilde{u}_{ik})^2](x_{ij} - v_{kj})^2}{\sum_{k=1}^{K} \sum_{i=1}^{n} [u_{ik}^2 + \alpha(u_{ik} - \tilde{u}_{ik})^2](x_{ih} - v_{kh})^2} \right)^{\frac{1}{s-1}} \right\}^{-1} \tag{4.26}$$

Também é importante notar que nesse caso o conjunto de variáveis que são relevantes e irrelevantes é igual para todos os grupos. Neste trabalho, o algoritmo adaptativo global semissupervisionado de agrupamento considerando a distância dada pela Equação (4.26) é chamado de BSSC-GS.

O algoritmo de agrupamento semissupervisionado (BSSC) que origina esses algoritmos com distâncias adaptativas pode ser formulado ignorando a etapa de ponderação e fixando a matriz $\mathbf{A}_k = (\lambda_{k1}, \ldots, \lambda_{kp}) = (1, \ldots, 1)(k = 1, \ldots, K)$ em todas as equações apresentadas onde a matriz $\mathbf{A}_k$ é utilizada.

4.1.3 ISODATA-PS com Distâncias Adaptativas

Pedrycz e Waletzky (PEDRYCZ, 1985; PEDRYCZ e WALETZKY, 1997) introduziram o primeiro algoritmo semissupervisionado que modifica a versão clássica do algoritmo (BEZDEK, 1981) pra lidar com o problema da supervisão parcial. Nesta subseção introduzimos a versão desse algoritmo com distância adaptativa que a função objetivo é

$$\begin{aligned} J(\mathbf{U}, \mathbf{A_1}, \ldots, \mathbf{A_K}, \mathbf{V}) &= \sum_{k=1}^{K}\sum_{i=1}^{n}(u_{ik})^2 d^2_{\mathbf{A}_k}(\mathbf{x}_i, \mathbf{v}_k) \\ &+ \alpha \sum_{k=1}^{K}\sum_{i=1}^{n}(u_{ik} - b_i f_{ik})^2 d^2_{\mathbf{A}_k}(\mathbf{x}_i, \mathbf{v}_k) \end{aligned} \tag{4.27}$$

onde $\mathbf{U}$, $\mathbf{V}$, b_i, f_{ik}, α e $d^2_{\mathbf{A}_k}(\mathbf{x}_i, \mathbf{v}_k)$ são como já descritos anteriormente.

O algoritmo Isodata-PS com distância adaptativa também é um algoritmo iterativo com três etapas (de representação, de ponderação e de alocação).

A etapa de representação tem duas fases: na primeira fase, o grau de pertinência de cada padrão de e_i em cada grupo C_k e as matrizes $\mathbf{A_k}$ são mantidas fixas. Os protótipos $\mathbf{v}_k$ dos grupos $C_k(k = 1, \ldots, K)$, que minimizam o critério de agrupamento J, são atualizados de acordo com (PEDRYCZ e WALETZKY, 1997)

$$\mathbf{v}_k = \frac{\sum_{i=1}^{n}[u_{ik}^2 + \alpha(u_{ik} - b_i f_{ik})^2]\mathbf{x}_i}{\sum_{i=1}^{n}[u_{ik}^2 + \alpha(u_{ik} - b_i f_{ik})^2]} \tag{4.28}$$

Na etapa de alocação, os protótipos $\mathbf{v}_k$ correspondentes aos grupos $C_k(k = 1, \ldots, K)$ e as matrizes $\mathbf{A}_k(k = 1, \ldots, K)$ são fixadas. O grau de pertinência $u_{ik}(i = 1, \ldots, n)$ de cada objeto e_i em cada grupo $C_k(k = 1, \ldots, K)$, que minimizam o critério de agrupamento J sob as restrições (2.2), são atualizados de acordo com a seguinte expressão (PEDRYCZ e WALETZKY, 1997)

$$u_{ik} = \frac{1}{1+\alpha}\left\{\frac{1 + \alpha(1 - b_i \sum_{l=1}^{K} f_{il})}{\sum_{l=1}^{K}\left(\frac{d^2_{\mathbf{A}_k}(\mathbf{x}_i, \mathbf{v}_k)}{d^2_{\mathbf{A}_l}(\mathbf{x}_i, \mathbf{v}_l)}\right)} + \alpha f_{ik} b_i\right\} \tag{4.29}$$

Os componentes da matriz de vetores de pesos $\mathbf{A}_k(k = 1, ..., K)$ são intimamente relacionados à distância adaptativa do algoritmo semissupervisionado de agrupamento e o formato dos grupos que são criados, como mostrado na Seção 2.3. Foram propostos quatro tipos de distâncias adaptativas baseado em como a matriz de distâncias de pesos é calculada e seus restritos pela soma ou produto de seus componentes igual a um. A seguir, apresentamos esses quatro tipos de distâncias adaptativas para o algoritmo Isodata-PS com distância adaptativa. As equações podem ser obtidas de maneira similar à explicada na Seção 4.1.1.

(a) ISODATA-PS-LP: algoritmo Isodata-PS com ponderação local de variáveis em que o produto dos pesos de cada variável de cada grupo tem que ser igual a um.

Neste trabalho, exploramos essa restrição de maneira inédita nesse algoritmo. Dessa forma, para calcular os componentes dos vetores de pesos de $\mathbf{A}_k = [\lambda_{kj}](k = 1, \ldots, K, j = 1, \ldots, p)$ temos uma nova Equação

$$\lambda_{kj} = \frac{\left\{ \prod_{h=1}^{p} \left(\sum_{i=1}^{n} \left[(u_{ik})^2 + \alpha(u_{ik} - b_i f_{ik})^2 \right] (x_{ih} - v_{kh})^2 \right) \right\}^{\frac{1}{p}}}{\sum_{i=1}^{n} \left[(u_{ik})^2 + \alpha(u_{ik} - b_i f_{ik})^2 \right] (x_{ij} - v_{kj})^2} \tag{4.30}$$

O conjunto de variáveis que são relevantes irrelevantes pode ser diferente para cada grupo. Neste trabalho, o algoritmo adaptativo local semissupervisionado de agrupamento considerando a distância dada pela Equação (4.30) é chamado de Isodata-PS-LP.

(b) ISODATA-PS-LS: algoritmo Isodata-PS com ponderação local de variáveis em que a soma dos pesos de cada variável de cada grupo tem que ser igual a um.

Os componentes dos vetores de pesos $\mathbf{A}_{k,s}(k = 1, ..., C)$ são calculados de acordo com:

$$\lambda_{kj} = \left\{ \sum_{h=1}^{p} \left(\frac{\sum_{i=1}^{n} [u_{ik}^2 + \alpha(u_{ik} - b_i f_{ik})^2](x_{ij} - v_{kj})^2}{\sum_{i=1}^{n} [u_{ik}^2 + \alpha(u_{ik} - b_i f_{ik})^2](x_{ih} - v_{kh})^2} \right)^{\frac{1}{s-1}} \right\}^{-1} \tag{4.31}$$

Também é importante notar que nesse caso o conjunto de variáveis que são relevantes e irrelevantes pode ser diferente para cada grupo. Neste trabalho, o algoritmo adaptativo local semissupervisionado de agrupamento considerando a distância dada pela Equação (4.31) é chamado de Isodata-PS-LS. Este algoritmo foi apresentado nos trabalhos de Marcelloni (MARCELLONI, 2003) e de Kong e Wang (KONG e WANG, 2009).

(c) ISODATA-PS-GP: algoritmo Isodata-PS com ponderação global de variáveis em que o produto dos pesos de cada variável de tem que ser igual a um.

Neste trabalho, exploramos essa restrição de maneira inédita nesse algoritmo. Dessa forma, para calcular os componentes dos vetores de pesos de $\mathbf{A} = [\lambda_j](j = 1, \ldots, p)$ temos uma nova Equação

$$\lambda_j = \frac{\left\{ \prod_{h=1}^{p} \left(\sum_{k=1}^{K} \sum_{i=1}^{n} \left[(u_{ik})^2 + \alpha(u_{ik} - b_i f_{ik})^2 \right] (x_{ih} - v_{kh})^2 \right) \right\}^{\frac{1}{p}}}{\sum_{k=1}^{K} \sum_{i=1}^{n} \left[(u_{ik})^2 + \alpha(u_{ik} - b_i f_{ik})^2 \right] (x_{ij} - v_{kj})^2} \tag{4.32}$$

É importante notar que nesse caso o conjunto de variáveis que são relevantes e irrelevantes é igual para todos os grupos. Neste trabalho, o algoritmo adaptativo global

semissupervisionado de agrupamento considerando a distância dada pela Equação (4.25) é chamado de Isodata-PS-GP.

(d) ISODATA-PS-GS: algoritmo Isodata-PS com ponderação global de variáveis em que a soma dos pesos de cada variável tem que ser igual a um.

Os componentes dos vetores de pesos $\mathbf{A} = [\lambda_j](j = 1, \ldots, p)$ são calculados de acordo com:

$$\lambda_j = \left\{ \sum_{h=1}^{p} \left(\frac{\sum_{k=1}^{K} \sum_{i=1}^{n} [u_{ik}^2 + \alpha(u_{ik} - b_i f_{ik})^2](x_{ij} - v_{kj})^2}{\sum_{k=1}^{K} \sum_{i=1}^{n} [u_{ik}^2 + \alpha(u_{ik} - b_i f_{ik})^2](x_{ih} - v_{kh})^2} \right)^{\frac{1}{s-1}} \right\}^{-1} \tag{4.33}$$

Também é importante notar que nesse caso o conjunto de variáveis que são relevantes e irrelevantes é igual para todos os grupos. Neste trabalho, o algoritmo adaptativo global semissupervisionado de agrupamento considerando a distância dada pela Equação (4.33) é chamado de Isodata-PS-GS.

O algoritmo de agrupamento semissupervisionado (Isodata-PS) que origina esses algoritmos com distâncias adaptativas pode ser formulado ignorando a etapa de ponderação e fixando $\mathbf{A}_k = (\lambda_{k1}, \ldots, \lambda_{kp}) = (1, \ldots, 1)(k = 1, \ldots, K)$ em todas as equações apresentadas onde a matriz $\mathbf{A}_k$ é utilizada.

4.1.4 sSSC com distâncias adaptativas

Endo *et al.* (ENDO et al., 2009) desenvolveram o chamado algoritmo de agrupamento semissupervisionado padrão (chamado sSSC), introduzindo uma matriz de pertinência supervisionada f_{ik} de objeto $e_i\,(i = 1, \ldots, n)$ no grupo $C_k\,(k = 1, \ldots, K)$ no algoritmo de agrupamento padrão FCM. Neste trabalho, introduzimos a versão desse algoritmo com distância adaptativa cuja função objetivo é

$$J_s(\mathbf{U}, \mathbf{A_1}, \ldots, \mathbf{A_K}, \mathbf{V}) \;=\; \sum_{k=1}^{K} \sum_{i=1}^{n} (u_{ik} - f_{ik})^2 d_{\mathbf{A}_k}^2(\mathbf{x}_i, \mathbf{v}_k) \tag{4.34}$$

onde

- $\mathbf{U}$, $\mathbf{A_1}, \ldots, \mathbf{A_k}$,$\mathbf{V}$, f_{ik}, e $d_{\mathbf{A}_k}^2(\mathbf{x}_i, \mathbf{v}_k)$ são como já descritos;

O algoritmo sSSC Adaptativo também é um algoritmo iterativo com três etapas (de representação, de ponderação e de alocação).

A etapa de representação tem duas fases: na primeira fase, o grau de pertinência de cada padrão de e_i em cada grupo C_k e as matrizes $\mathbf{A_k}$ são mantidas fixas. Os protótipos $\mathbf{v}_k$ dos grupos $C_k(k = 1, \ldots, K)$, que minimizam o critério de agrupamento J_s, são

atualizados de acordo com

$$\mathbf{v}_k = \frac{\sum_{i=1}^{n} (u_{ik} - f_{ik})^2 \mathbf{x}_i}{\sum_{i=1}^{n} (u_{ik} - f_{ik})^2} \tag{4.35}$$

Na etapa de alocação, os protótipos $\mathbf{v}_k$ correspondentes aos grupos $C_k (k = 1, \ldots, K)$ e as matrizes $\mathbf{A}_k (k = 1, \ldots, K)$ são fixados. O grau de pertinência $u_{ik} (i = 1, \ldots, n)$ de cada objeto e_i em cada grupo $C_k (k = 1, \ldots, K)$, que minimizam o critério de agrupamento J_s sob as restrições (2.2), são atualizados de acordo com a seguinte expressão

$$u_{ik} = f_{ik} + \left(1 - \sum_{l=1}^{K} f_{il}\right) \frac{\left(\frac{1}{d^2_{\mathbf{A}_k}(\mathbf{x}_i, \mathbf{v}_k)}\right)^{\frac{1}{m-1}}}{\sum_{l=1}^{K} \left(\frac{1}{d^2_{\mathbf{A}_k}(\mathbf{x}_i, \mathbf{v}_l)}\right)^{\frac{1}{m-1}}} \tag{4.36}$$

Os componentes da matriz de vetores de pesos $\mathbf{A}_k (k = 1, ..., K)$ são intimamente relacionados à distância adaptativa do algoritmo semissupervisionado de agrupamento e o formato dos grupos que são criados, como mostrado na Seção 2.3. Nesta tese, foram propostas a utilização de quatro tipos de distâncias adaptativas baseadas em como a matriz de distâncias de pesos é calculada e seus componentes restritos pela soma ou produto igual a um. A seguir, apresentamos esses quatro tipos de distâncias adaptativas para o algoritmo sSSC Adaptativo. As equações podem ser obtidas de maneira similar à explicada na Seção 4.1.1.

(a) sSSC-LP: algoritmo sSSC com ponderação local de variáveis em que o produto dos pesos de cada variável de cada grupo tem que ser igual a um.

Neste trabalho exploramos essa restrição de maneira inédita nesse algoritmo. Dessa forma, para calcular os componentes dos vetores de pesos de $\mathbf{A}_k = [\lambda_{kj}] (k = 1, \ldots, K, j = 1, \ldots, p)$ temos uma nova Equação

$$\lambda_{kj} = \frac{\left\{\prod_{h=1}^{p} \left(\sum_{i=1}^{n} |u_{ik} - f_{ik}|^m (x_{ih} - v_{kh})^2\right)\right\}^{\frac{1}{p}}}{\sum_{i=1}^{n} |u_{ik} - f_{ik}|^m (x_{ij} - v_{kj})^2} \tag{4.37}$$

É importante notar que nesse caso, o conjunto de variáveis que são relevantes e irrelevantes, pode ser diferente para cada grupo. Neste trabalho, o algoritmo adaptativo local semissupervisionado de agrupamento considerando a distância dada pela Equação (4.37) é chamado de sSSC-LP.

(b) sSSC-LS: algoritmo sSSC com ponderação local de variáveis em que a soma dos pesos de cada variável de cada grupo tem que ser igual a um.

Os componentes dos vetores de pesos $\mathbf{A} = [\lambda_{kj}] (k = 1, \ldots, K, j = 1, \ldots, p)$ são calculados de acordo com:

$$\lambda_{kj} = \left\{\sum_{h=1}^{p} \left(\frac{\sum_{i=1}^{n} |u_{ik} - f_{ik}|^m (x_{ij} - v_{kj})^2}{\sum_{i=1}^{n} |u_{ik} - f_{ik}|^m (x_{ih} - v_{kh})^2}\right)^{\frac{1}{s-1}}\right\}^{-1} \tag{4.38}$$

onde geralmente $s = 2$.

Também é importante notar que nesse caso o conjunto de variáveis que são relevantes e irrelevantes, pode ser diferente para cada grupo. Neste trabalho, o algoritmo adaptativo local semissupervisionado de agrupamento considerando a distância dada pela Equação (4.38) é chamado de sSSC-LS.

(c) sSSC-GP: algoritmo sSSC com ponderação global de variáveis em que o produto dos pesos de cada variável de tem que ser igual a um.

Nesse trabalho, também exploramos essa restrição de maneira inédita nesse algoritmo. Dessa forma, para calcular os componentes dos vetores de pesos de $\mathbf{A} = [\lambda_j](j = 1, \ldots, p)$ temos uma nova Equação

$$\lambda_j = \frac{\left\{ \prod_{h=1}^{p} \left(\sum_{k=1}^{K} \sum_{i=1}^{n} |u_{ik} - f_{ik}|^m (x_{ih} - v_{kh})^2 \right) \right\}^{\frac{1}{p}}}{\sum_{k=1}^{K} \sum_{i=1}^{n} |u_{ik} - f_{ik}|^m (x_{ij} - v_{kj})^2} \tag{4.39}$$

É importante notar que nesse caso o conjunto de variáveis que são relevantes e irrelevantes é o mesmo para todos os grupos. Neste trabalho, o algoritmo adaptativo global semissupervisionado de agrupamento considerando a distância dada pela Equação (4.39) é chamado de sSSC-GP.

(d) sSSC-GS: algoritmo sSSC com ponderação global de variáveis em que a soma dos pesos de cada variável tem que ser igual a um.

Os componentes dos vetores de pesos $\mathbf{A} = [\lambda_j](j = 1, \ldots, p)$ são calculados de acordo com:

$$\lambda_j = \left\{ \sum_{h=1}^{p} \left(\frac{\sum_{k=1}^{K} \sum_{i=1}^{n} |u_{ik} - f_{ik}|^m (x_{ij} - v_{kj})^2}{\sum_{k=1}^{K} \sum_{i=1}^{n} |u_{ik} - f_{ik}|^m (x_{ih} - v_{kh})^2} \right)^{\frac{1}{s-1}} \right\}^{-1} \tag{4.40}$$

Também é importante notar que nesse caso o conjunto de variáveis que relevantes e irrelevantes é o mesmo para todos os grupos. Neste trabalho, o algoritmo adaptativo global semissupervisionado de agrupamento considerando a distância dada pela Equação (4.40) é chamado de sSSC-GS.

O algoritmo de agrupamento semissupervisionado (sSSC) (ENDO et al., 2009) que origina esses algoritmos com distâncias adaptativas pode ser formulado ignorando a etapa de ponderação e fixando $\mathbf{A}_k = (\lambda_{k1}, \ldots, \lambda_{kp}) = (1, \ldots, 1)(k = 1, \ldots, K)$ em todas as equações apresentadas onde a matriz $\mathbf{A}_k$ é utilizada.

4.1.5 ESSC com distâncias adaptativas

No trabalho de Endo *et al.* (ENDO et al., 2009), eles propõem um novo algoritmo semissupervisionado (chamado aqui de ESSC) em que adiciona um termo supervisionado baseado na entropia ao termo do algoritmo de agrupamento FCM. A entropia é calculada utilizando a diferença entre o grau de pertinência de um objeto e_i ao grupo C_k e o valor de f_ik que indica se um objeto e_i pertence à classe representada pelo grupo C_k. Caso o objeto pertença ao grupo C_k, o valor da entropia é baixo, caso contrário, esse valor é alto. A função objetivo deste algoritmo com distância adaptativa é a seguinte

$$\begin{aligned} J(\mathbf{U}, \mathbf{A_1}, \ldots, \mathbf{A_K}, \mathbf{V}) &= \sum_{k=1}^{K}\sum_{i=1}^{n}(u_{ik})d^2_{\mathbf{A}_k}(\mathbf{x}_i, \mathbf{v}_k) \\ &+ \alpha\sum_{k=1}^{K}\sum_{i=1}^{n}(|(u_{ik} - f_{ik})|\log|(u_{ik} - f_{ik})|) \end{aligned} \tag{4.41}$$

onde

- $\mathbf{U}$, $\mathbf{A_1}, \ldots, \mathbf{A_k}$,$\mathbf{V}$, f_{ik}, e $d^2_{\mathbf{A}_k}(\mathbf{x}_i, \mathbf{v}_k)$ são como já descritos;

Assim como os algoritmos descritos até agora, o algoritmo ESSC Adaptativo também é um algoritmo iterativo com três etapas (de representação, de ponderação e de alocação).

Na etapa de representação, o grau de pertinência de cada padrão de e_i em cada grupo C_k e as matrizes $\mathbf{A_k}$ são mantidas fixas. Os protótipos $\mathbf{v}_k$ dos grupos $C_k(k = 1, \ldots, K)$, que minimizam o critério de agrupamento J, são atualizados de acordo com

$$\mathbf{v}_k = \frac{\sum_{i=1}^{n} u_{ik}\mathbf{x}_i}{\sum_{i=1}^{n} u_{ik}} \tag{4.42}$$

Na etapa de alocação, os protótipos $\mathbf{v}_k$ correspondentes aos grupos $C_k(k = 1, \ldots, K)$ e as matrizes $A_k(k = 1, \ldots, K)$ são fixadas. O grau de pertinência $u_{ik}(i = 1, \ldots, n)$ de cada objeto e_i em cada grupo $C_k(k = 1, \ldots, K)$ que minimiza o critério de agrupamento J sob as restrições (2.2), são atualizados de acordo com a seguinte expressão

$$u_{ik} = f_{ik} + \frac{e^{-\alpha d^2_{\mathbf{A}_k}(\mathbf{x}_i, \mathbf{v}_k)}}{\sum_{l-1}^{K} e^{-\alpha d^2_{\mathbf{A}_l}(\mathbf{x}_i, \mathbf{v}_l)}}\left(1 - \sum_{l=1}^{K} f_{il}\right) \tag{4.43}$$

Neste trabalho propomos quatro tipos de distâncias adaptativas baseado em como a matriz de distâncias de pesos é calculada e restritos pela soma ou produto de seus componentes igual a um. A seguir, apresentamos esses quatro tipos de distâncias adaptativas para o algoritmo ESSC. As equações podem ser obtidas de maneira similar à explicada na Seção 4.1.1.

(a) ESSC-LP: algoritmo ESSC com ponderação local de variáveis em que o produto dos pesos de cada variável de cada grupo tem que ser igual a um.

Neste trabalho exploramos essa restrição de maneira inédita nesse algoritmo. Dessa forma, para calcular os componentes dos vetores de pesos de $\mathbf{A}_k = [\lambda_{kj}](k = 1, \ldots, K, j = 1, \ldots, p)$ temos uma nova Equação

$$\lambda_{kj} = \frac{\left\{\prod_{h=1}^{p}\left(\sum_{i=1}^{n}\left[u_{ik}(x_{ih} - v_{kh})^2\right]\right)\right\}^{\frac{1}{p}}}{\sum_{i=1}^{n}\left[u_{ik}(x_{ij} - v_{kj})^2\right]} \tag{4.44}$$

É importante notar que nesse caso o conjunto de variáveis que são relevantes e irrelevantes pode ser diferente para cada grupo. Neste trabalho, o algoritmo adaptativo local semissupervisionado de agrupamento considerando a distância dada pela Equação (4.44) é chamado de ESSC-LP.

(b) ESSC-LS: algoritmo ESSC com ponderação local de variáveis em que a soma dos pesos de cada variável de cada grupo tem que ser igual a um.

Os componentes dos vetores de pesos $\mathbf{A} = [\lambda_{kj}](k = 1, \ldots, K, j = 1, \ldots, p)$ são calculados de acordo com:

$$\lambda_{kj} = \left\{\sum_{h=1}^{p}\left(\frac{\sum_{i=1}^{n} u_{ik}(x_{ij} - v_{kj})^2}{\sum_{i=1}^{n} u_{ik}(x_{ih} - v_{kh})^2}\right)^{\frac{1}{1-s}}\right\}^{-1} \tag{4.45}$$

Também é importante notar que nesse caso o conjunto de variáveis que são relevantes e irrelevantes pode ser diferente para cada grupo. Neste trabalho, o algoritmo adaptativo local semissupervisionado de agrupamento considerando a distância dada pela Equação (4.45) é chamado de ESSC-LS.

(c) ESSC-GP: algoritmo ESSC com ponderação global de variáveis em que o produto dos pesos de cada variável tem que ser igual a um.

Neste trabalho também exploramos essa restrição de maneira inédita nesse algoritmo. Dessa forma, para calcular os componentes dos vetores de pesos de $\mathbf{A} = [\lambda_j](j = 1, \ldots, p)$ temos uma nova Equação

$$\lambda_j = \frac{\left\{\prod_{h=1}^{p}\left(\sum_{k=1}^{K}\sum_{i=1}^{n}\left[u_{ik}(x_{ih} - v_{kh})^2\right]\right)\right\}^{\frac{1}{p}}}{\sum_{k=1}^{K}\sum_{i=1}^{n}\left[u_{ik}(x_{ij} - v_{kj})^2\right]} \tag{4.46}$$

É importante notar que nesse caso o conjunto de variáveis que são relevantes e irrelevantes é igual para todos os grupos. Neste trabalho, o algoritmo adaptativo global semissupervisionado de agrupamento considerando a distância dada pela Equação (4.46) é chamado de ESSC-GP.

(d) ESSC-GS: algoritmo ESSC com ponderação global de variáveis em que a soma dos pesos de cada variável tem que ser igual a um.

Os componentes dos vetores de pesos $\mathbf{A} = [\lambda_j](j = 1, \ldots, p)$ são calculados de acordo com:

$$\lambda_j = \left\{ \sum_{h=1}^{p} \left(\frac{\sum_{k=1}^{K} \sum_{i=1}^{n} u_{ik}(x_{ij} - v_{kj})^2}{\sum_{k=1}^{K} \sum_{i=1}^{n} u_{ik}(x_{ih} - v_{kh})^2} \right)^{\frac{1}{s-1}} \right\}^{-1} \tag{4.47}$$

Também é importante notar que nesse caso o conjunto de variáveis que são relevantes e irrelevantes é igual para todos os grupos. Neste trabalho, o algoritmo adaptativo global semissupervisionado de agrupamento considerando a distância dada pela Equação (4.47) é chamado de ESSC-GS.

O algoritmo de agrupamento semissupervisionado (ESSC) (ENDO et al., 2009) que origina esses algoritmos com distâncias adaptativas pode ser formulado ignorando a etapa de ponderação e fixando $\mathbf{A}_k = (\lambda_{k1}, \ldots, \lambda_{kp}) = (1, \ldots, 1)(k = 1, \ldots, K)$ em todas as equações apresentadas onde a matriz $\mathbf{A}_k$ é utilizada.

4.1.6 Etapas dos algoritmos

Nesta subseção, apresentamos os passos comuns a todos os algoritmos descritos neste capítulo. Cada família de algoritmos de agrupamento semissupervisionado com distância adaptativa é representada por uma letra conforme Tabela 15. As equações desses algoritmos são apresentadas na Tabela 16.

Tabela 15 – Símbolo que representa cada família de algoritmos de agrupamento semissupervisionado com distância adaptativa no esquema.

	Algoritmos
A	NebFuzz
B	BSSC
C	Isodata-PS
D	sSSC
E	ESSC

ESQUEMA DOS ALGORITMOS DE AGRUPAMENTO SEMISSUPERVISIONADO COM DISTÂNCIAS ADAPTATIVAS

1. **Inicialização (Igual para todos os algoritmos)**

a. Fixar o número de grupos K; Fixar $MaxIter$ (número de máximos de iterações); Fixar $\epsilon << 0$; Fixar $\alpha > 0$; Fixar β no caso do algoritmo BSSC.

b. Conjunto $\mathbf{A}_k^{(0)} = (\lambda_{k1}^{(0)}, \ldots, \lambda_{kp}^{(0)}) = (1, \ldots, 1)(k = 1, \ldots, K)$;

c. Execute o algoritmo FCM até que $|J_{FCM}^{t+1} - J_{FCM}^{t}| \leq \epsilon$ **ou** $s > MaxIter$ para inicializar a matriz de graus de pertinências $\mathbf{U}$ e de protótipos $\mathbf{V}$

d. Calcular $J^{(0)}$, e atribuir $t = 1$

Repetir :

2. **Etapa de Representação:**
   Calcular os protótipos $\mathbf{v}_k$ com as Equações: A1, B1, C1, D1 ou E1

3. **Etapa de Ponderação:**
   Se distância adaptativa é local:
   Calcular os vetores de pesos para cada grupo $\mathbf{A}_k\,(k = 1, \ldots, K)$ com as Equações:
   Se restrição é o produto (LP): A2, B2, C2, D2 ou E2
   Se restrição é a soma (LS): A3, B3, C3, D3 ou E3

   Se distância adaptativa é global:
   Calcular os vetores de pesos $\mathbf{A}$ com as Equações:
   Se restrição é o produto (GP): A4, B4, C4, D4 ou E4
   Se restrição é a soma (GS): A5, B5, C5, D5 ou E5

4. **Etapa de Alocação:**
   Calcular o grau de pertinência u_{ik} dos objetos $\mathbf{x}_i (i = 1, \ldots, n)$ no grupo $C_k\,(k = 1, \ldots, K)$ com as Equações A6, B6, C6, D6 ou E6
   Apenas para o algoritmo BSSC, calcular a matriz de pertinência $\tilde{u}_{ik}$ do objeto $\mathbf{x}_i (i = 1, \ldots, n)$ no grupo $C_k\,(k = 1, \ldots, K)$ com a Equação (4.20)

5. **Etapa da Função Objetivo:**
   Calcular $J^{(t)}$ com as Equações A7,
   t=t+1

Até que $|J^{t+1} - J^{t}| \leq \epsilon$ **ou** $s > MaxIter$

4.1.7 Propriedades de convergência

Nesta seção é apresentada a prova de convergência dos algoritmos que são baseados em distâncias adaptativas com ponderação local ou global de variáveis em que a soma ou o produto dos pesos das variáveis são restritos ao valor um. Vamos considerar nesta seção a

Tabela 16 – Equações relativas aos algoritmos semissupervisionados com distância adaptativa.

Algoritmo	Equações						
A	A1 (4.5)	A2 (4.9)	A3 (4.12)	A4 (4.15)	A5 (4.18)	A6 (4.6)	A7 (4.1)
B	B1 (4.22)	B2 (4.23)	B3 (4.24)	B4 (4.25)	B5 (4.26)	B6 (4.21))	B7 (4.19)
C	C1 (4.28)	C2 (4.30)	C3 (4.31)	C4 (4.32)	C5 (4.33)	C6 (4.29)	C7 (4.27)
D	D1 (4.35)	D2 (4.37)	D3 (4.38)	D4 (4.39)	D5 (4.40)	D6 (4.36)	D7 (4.34)
E	E1 (4.42)	E2 (4.44)	E3 (4.45)	E4 (4.46)	E5 (4.47)	E6 (4.43)	E7 (4.41)

versão do algoritmo de agrupamento semissupervisionado com distância adaptativa local com à restrição de produto NebFuzz-LP. Esta prova pode ser diretamente estendida para todos os algoritmos de agrupamento semissupervisionados com distâncias adaptativas propostos neste trabalho.

O algoritmo NebFuzz busca por uma partição *fuzzy* $P^* = \{C_1^*, \ldots, C_K^*\}$ de E em K grupos *fuzzy* representados por $\mathbf{U}^* = (\mathbf{u}_1^*, \ldots, \mathbf{u}_n^*)$ um vetor de protótipos K-dimensional correspondente $\mathbf{V}^* = (\mathbf{v}_1^*, \ldots, \mathbf{v}_K^*)$ representando os grupos *fuzzy* nas partições P, e K distâncias adaptativas parametrizados por uma K-tupla de vetores de pesos $\mathbf{A}^* = (\mathbf{A}_\mathbf{1}^*, \ldots, \mathbf{A}_K^*)$, tal que

$$J(\mathbf{V}^*, \mathbf{A}^*, \mathbf{U}^*) = min\left\{J(\mathbf{V}, \mathbf{A}, \mathbf{U}) : \mathbf{V} \in \mathbb{V}^K, \mathbf{A} \in \mathbb{L}^K, \mathbf{U} \in \mathbb{U}^n\right\}, \tag{4.48}$$

onde

- $\mathbb{L}$ é o espaço de representação de protótipos tal que $\mathbf{v}_k \in \mathbb{L}\,(k = 1, \ldots, K)$. Aqui, $\mathbb{L} = \mathbb{R}^p$ e $\mathbf{V} \in \mathbb{L}^K = \mathbb{L} \times \ldots \times \mathbb{L} = \mathbb{R}^{K \times p}$;

- $\mathbb{L}$ é o espaço de vetores de pesos que parametrizam as distâncias adaptativas, tal que $\mathbf{A}_k \in \mathbb{L}\,(k = 1, \ldots, K)$. Se estamos considerando o produto dos pesos das variáveis tem que ser igual a um (restrição de produto), temos $\mathbb{L} = \{\mathbf{A} = (\lambda_{j=1}, \ldots, \lambda_p) \in \mathbb{R}_+^* \times, \ldots, \times \mathbb{R}_+^* = (\mathbb{R}_+^*)^p : \prod_{j=1}^p \lambda_j = 1\}$. Entretanto, se estamos considerando a soma dos pesos das variáveis igual um (restrição de soma), temos $\mathbb{L} = \{\mathbf{A} = (\lambda_1, \ldots, \lambda_p) \in \mathbb{R}_+^* \times, \ldots, \times \mathbb{R}_+^* = (\mathbb{R}_+^*)^p : \sum_{j=1}^p \lambda_j = 1\}$. Finalmente, temos $\mathbf{A} \in \mathbb{L}^K = \mathbb{L} \times \ldots \times \mathbb{L} = (\mathbb{R}_+^*)^{K \times p}$; obs: para versões adaptativas globais, considere $K = 1$;

- $\mathbb{U}$ é o espaço de graus de pertinência de partição *fuzzy* tal que $\mathbf{u}_k \in \mathbb{U}\,(k = 1,\ldots,K)$. Aqui $\mathbb{U} = \{\mathbf{u} = (u_1,\ldots,u_K) \in [0,1]\times\ldots\times[0,1] = [0,1]^K : \sum_{k=1}^{K} u_k = 1\}$ e $\mathbf{U} \in \mathbb{U}^n = \mathbb{U}\times\ldots\times\mathbb{U}$;

De acordo com Diday e Simon (DIDAY e SIMON, 1976), as propriedades de convergência deste tipo de algoritmo podem ser estudadas a partir de duas séries: $y_t = (\mathbf{V}^t, \mathbf{A}^t, \mathbf{U}^t) \in \mathbb{V}^K \times \mathbb{L}^K \times \mathbb{U}^n$ and $z_t = J(y_t) = J(\mathbf{V}^t, \mathbf{A}^t, \mathbf{U}^t),\ t = 0, 1, \ldots$. Partindo de uma solução inicial $y_0 = (\mathbf{V}^0, \mathbf{A}^0, \mathbf{U}^0)$, o algoritmo calcula diferentes termos da série y_t até a convergência (que é demonstrada) quando o critério J atinge um valor estacionário.

Proposição 3. *As séries $z_t = J(y_t)$ diminuem a cada iteração e convergem.*

Prova. Primeiro vamos mostrar que as desigualdades (I), (II) e (III)

$$\begin{aligned} \underbrace{J(\mathbf{V}^t, \mathbf{A}^t, \mathbf{U}^t)}_{z_t} &\overbrace{\geq}^{(I)} J(\mathbf{V}^{t+1}, \mathbf{A}^t, \mathbf{U}^t) \\ &\overbrace{\geq}^{(II)} J(\mathbf{V}^{t+1}, \mathbf{A}^{t+1}, \mathbf{U}^t) \overbrace{\geq}^{(III)} \underbrace{J(\mathbf{V}^{t+1}, \mathbf{A}^{t+1}, \mathbf{U}^{t+1})}_{z_{t+1}} \end{aligned} \tag{4.49}$$

mantêm-se (isto é, a série diminui em cada iteração).

A desigualdade (I) se mantém porque

a)

$$\begin{aligned} J(\mathbf{V}^t, \mathbf{A}^t, \mathbf{U}^t) &= \sum_{k=1}^{K}\sum_{i=1}^{n}(u_{ik}^t)^2 d_{\mathbf{A}}^2(\mathbf{x}_i, \mathbf{v}_k^t) \\ &+ \alpha \times \sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{m=1}^{n} b_i b_m t_{im}(u_{ik}^t - u_{mk}^t)^2 d_{\mathbf{A}}^2(\mathbf{x}_i, \mathbf{x}_m) \end{aligned}$$

b)

$$\begin{aligned} J(\mathbf{V}^{t+1}, \mathbf{A}^t, \mathbf{U}^t) &= \sum_{k=1}^{K}\sum_{i=1}^{n}(u_{ik}^t)^2 d_{\mathbf{A}}^2(\mathbf{x}_i, \mathbf{v}_k^{t+1}) \\ &+ \alpha \times \sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{m=1}^{n} b_i b_m t_{im}(u_{ik}^t - u_{mk}^t)^2 d_{\mathbf{A}}^2(\mathbf{x}_i, \mathbf{x}_m) \end{aligned}$$

e

$$\begin{aligned}\mathbf{V}^{(t+1)} &= (\mathbf{v}_1^{(t+1)}, \ldots, \mathbf{v}_K^{(t+1)}) = \\ &= \underbrace{argmin}_{\mathbf{V}=(\mathbf{v}_1,\ldots,\mathbf{v}_K)\in \mathbb{I}\mathbf{V}^K} \sum_{k=1}^{K}\sum_{i=1}^{n}(u_{ik}^t)^2 d_{\mathbf{A}}^2(\mathbf{x}_i, \mathbf{v}_k) + \\ &+ \alpha \times \sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{m=1}^{n} b_i b_m t_{im}(u_{ik}^t - u_{mk}^t)^2 d_{\mathbf{A}}^2(\mathbf{x}_i, \mathbf{x}_m)\end{aligned}$$

Além disso, a desigualdade (II) também se mantém porque

$$\begin{aligned}J(\mathbf{V}^{t+1}, \mathbf{A}^{t+1}, \mathbf{U}^t) &= \sum_{k=1}^{K}\sum_{i=1}^{n}(u_{ik}^{t+1})^2 d_{\mathbf{A}}^2(\mathbf{x}_i, \mathbf{v}_k^{t+1}) \\ &+ \alpha \times \sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{m=1}^{n} b_i b_m t_{im}(u_{ik}^{t+1} - u_{mk}^{t+1})^2 d_{\mathbf{A}}^2(\mathbf{x}_i, \mathbf{x}_m)\end{aligned}$$

e

$$\begin{aligned}\mathbf{A}^{t+1} &= (\mathbf{A}_1^{t+1}, \ldots, \mathbf{A}_K^{t+1}) \\ &= \underbrace{argmin}_{\mathbf{A}=(\mathbf{A}_1,\ldots,\mathbf{A}_K)\in \mathbb{I}\mathbf{L}^K} \sum_{k=1}^{K}\sum_{i=1}^{n}(u_{ik})^2 d_{\mathbf{A}}^2(\mathbf{x}_i, \mathbf{v}_k^{t+1}) + \\ &+ \alpha \times \sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{m=1}^{n} b_i b_m t_{im}(u_{ik} - u_{mk}^t)^2 d_{\mathbf{A}}^2(\mathbf{x}_i, \mathbf{x}_m)\end{aligned}$$

A desigualdade (III) também se mantém porque

$$\begin{aligned}J(\mathbf{V}^{t+1}, \mathbf{A}^{t+1}, \mathbf{U}^{t+1}) &= \sum_{k=1}^{K}\sum_{i=1}^{n}(u_{ik}^{t+1})^2 d_{\mathbf{A}}^2(\mathbf{x}_i, \mathbf{v}_k^{t+1}) \\ &+ \alpha \times \sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{m=1}^{n} b_i b_m t_{im}(u_{ik}^{t+1} - u_{mk}^{t+1})^2 d_{\mathbf{A}}^2(\mathbf{x}_i, \mathbf{x}_m)\end{aligned}$$

e

$$\begin{aligned}\mathbf{U}^{t+1} &= (\mathbf{u}_1^{t+1}, \ldots, \mathbf{u}_K^{t+1}) \\ &= \underbrace{argmin}_{\mathbf{U}=(\mathbf{u}_1,\ldots,\mathbf{u}_K)\in \mathbb{I}\mathbf{U}^n} \sum_{k=1}^{K}\sum_{i=1}^{n}(u_{ik})^2 d_{\mathbf{A}}^2(\mathbf{x}_i, \mathbf{v}_k^{t+1}) + \\ &+ \alpha \times \sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{m=1}^{n} b_i b_m t_{im}(u_{ik} - u_{mk}^t)^2 d_{\mathbf{A}}^2(\mathbf{x}_i, \mathbf{x}_m)\end{aligned}$$

Finalmente, porque as séries z_t diminuem e são limitadas por $(J(y_t) \geq 0)$, converge.

Proposição 4. *As séries* $y_t = (\boldsymbol{V}^t, \mathbf{A}^t, \boldsymbol{U}^t)$ *convergem.*

Prova. Assuma que a estacionariedade das séries z_t é alcançada na iteração $t = T$. Então, temos que $z_T = z_{T+1}$ e, em seguida, $J(y_T) = J(y_{T+1})$.

Partindo de $J(y_T) = J(y_{T+1})$, temos $J(\mathbf{V}^T, \mathbf{A}^T, \mathbf{U}^T) = J(\mathbf{V}^{T+1}, \mathbf{A}^{T+1}, \mathbf{U}^{T+1})$ e essa igualdade, de acordo com proposição 1, pode ser reescrita como igualdades (I), (II) e (III):

$$J(\mathbf{V}^T, \mathbf{A}^T, \mathbf{U}^T) \overset{I}{=} J(\mathbf{V}^{T+1}, \mathbf{A}^T, \mathbf{U}^T)$$
$$\overset{II}{=} J(\mathbf{V}^{T+1}, \mathbf{A}^{T+1}, \mathbf{U}^T) \overset{III}{=} J(\mathbf{V}^{T+1}, \mathbf{A}^{T+1}, \mathbf{U}^{T+1})$$

Da primeira igualdade (I), temos que $\mathbf{V}^T = \mathbf{V}^{T+1}$ porque $\mathbf{V}$ é única minimizando J quando a partição *fuzzy* representada por $\mathbf{U}^T$ e os vetores de pesos representados por $\mathbf{A}^T$ são mantidos fixos. Da segunda igualdade (II), temos que $\mathbf{A}^T = \mathbf{A}^{T+1}$ porque $\mathbf{A}$ é único minimizando J quando os vetores de protótipos $\mathbf{V}^{T+1}$ e a partição *fuzzy* representada por $\mathbf{U}^T$ são mantidos fixos. Além disso, na terceira igualdade (III), temos que $\mathbf{U}^T = \mathbf{U}^{T+1}$, porque $\mathbf{U}$ é única minimizando J quando o vetor de protótipos $\mathbf{V}^{T+1}$ e os vetores de pesos representados por $\mathbf{A}^T$ são mantidos fixos.

Finalmente, concluímos que $y_T = y_{T+1}$. Esta conclusão é válida para todos os $t \geq T$ and $y_t = y_T, \forall t \geq T$ e segue até que as séries y_t convergem.

4.1.8 Custo computacional

Nesta seção realizamos a análise de complexidade dos algoritmos de agrupamento semissupervisionados com ponderação automática de variáveis apresentados neste capítulo. Conforme apresentado na Subseção 4.1.6, os algoritmos de agrupamento semissupervisionados realizam realocação de três etapas com dois laços: (i) um laço externo de tamanho $T = Max$ e (ii) um laço aninhado: este otimizando J (ou seja, três passos para computar $\mathbf{V}$, $\mathbf{A}$ e $\mathbf{U}$. Ao analisar a complexidade, verificamos a complexidade máxima para o cálculo das matrizes $\mathbf{U}$, $\mathbf{A}$ e $\mathbf{V}$ desses algoritmos. E depois, consideramos o laço externo de tamanho T no cálculo da complexidade. O cálculo da matriz $\mathbf{A}_k$ é realizado com complexidades semelhantes em todas as versões apresentadas neste trabalho: (LP), (LS), (GP) e (GL). Portanto, as versões com distâncias adaptativas de um mesmo algoritmo são analisados da mesma maneira.

Os algoritmos com ponderação automática de variáveis baseados nos algoritmos Isodata-PS (PEDRYCZ e WALETZKY, 1997), sSSC e ESSC (ENDO et al., 2009), podem ser igualmente analisados, pois eles possuem as mesmas complexidades $O(Kn)$ para calcular as matrizes $\mathbf{U}$ e $\mathbf{V}$, onde K é número de grupos e n é o número de exemplos e a

mesma complexidade $O(pkn)$ para calcular a matriz $\mathbf{A}_k$, onde p é o número de atributos de cada exemplo e_i. A complexidade desses algoritmos é $Max(O(Kn), O(Kn), O(Kpn))$ resultando em uma complexidade para $O(Kpn)$ para este laço. Por último, levamos em conta o laço externo de tamanho T, a complexidade final é $O(TKpn)$, sendo linear para cada um desses parâmetros.

Em relação à complexidade de tempo dos algoritmos com distância adaptativa baseados no NebFuzz, estes algoritmos de agrupamento semissupervisionado também são um algoritmo de realocação de três etapas com dois laços. A complexidade deste algoritmo é $Max(O(Kn), O(Kpn^2), O(Kn^2))$ para o cálculo das matrizes $\mathbf{V}$, $\mathbf{A}$ e $\mathbf{U}$ respectivamente, o que produz $O(Kpn^2)$. Note que as fórmulas são divididas em sub-fórmulas. Levando em conta o laço externo, a complexidade de tempo total é $O(TKpn^2))$. Isto significa que o tempo de complexidade de NebFuzz é linear em relação ao número de iterações, número de atributos de um exemplo e no número de grupos e quadrática no número de objetos.

O algoritmo BSSC possui um laço extra em relação aos outros algoritmos para o cálculo da matriz $\tilde{\mathbf{U}}$. Uma análise desse algoritmo foi realizada no artigo de Bouchachia e Pedrycz (BOUCHACHIA e PEDRYCZ, 2006a). Assim, no algoritmo BSSC, temos dois laços internos de tamanho L_1 e L_2 respectivamente e um laço externo de tamanho T. O primeiro laço interno possui complexidade $O(L_1Kn)$, o segundo laço possui complexidade $Max(O(L_2Kn), O(L_2Kpn), O(L_2Hn))$, onde H é o número de Classes. Sabendo que $H < K$, a complexidade será $O(L_2Kpn)$. Agora, considerando o laço externo, temos a complexidade $Max(O(TL_1Kn), O(TL_2Kpn))$. Note que a complexidade escolhida será a que for maior entre L_1 e L_2p, então deixe que L seja $Max(L_1, L2p)$, a complexidade do algoritmo BSSC é $O(TLKn)$. Note que a complexidade é linear para cada um desses parâmetros. A Tabela 17 apresenta um resumo da complexidade para cada um dos algoritmos.

Tabela 17 – Complexidade de Tempo.

Algoritmo	Complexidade
BSSC-LP,BSSC-LS,BSSC-GP,BSSC-GS	$O(TLKn)$
sSSC-LP,sSSC-LS,sSSC-GP,sSSC-GS	$O(TKpn)$
ESSC-LP,ESSC-LS,ESSC-GP,ESSC-GS	$O(TKpn)$
Isodata-PS-LP,Isodata-PS-LS,Isodata-PS-GP,Isodata-PS-GS	$O(TKpn)$
NebFuzz-LP,NebFuzz-LS,NebFuzz-GP,NebFuzz-GS	$O(TKpn^2)$

4.2 RESUMO DOS ALGORITMOS SEMISSUPERVISIONADOS COM DISTÂNCIAS ADAPTATIVAS

Todos os algoritmos explicados neste capítulo se baseiam no algoritmo clássico de agrupamento FCM, relacionados com o algoritmo GK (GUSTAFSON e KESSEL, 1979).

Os algoritmos adicionam um segundo termo à função objetivo do algoritmo GK, que considera rótulos conhecidos dos exemplos das bases de dados para otimizar a função objetivo. Esses algoritmos possuem como ponto importante o cálculo de similaridades entre dois objetos, pois os objetos mais similares são colocados no mesmo grupo, objetos diferentes são colocados em grupos diferentes. Os exemplos a serem agrupados são usualmente representados como vetores nos quais cada componente é uma medição referente a uma variável. Uma das distâncias adaptativas utilizadas pelos algoritmos de agrupamento semissupervisionados propostos nesta tese, a distância adaptativa local, considera que a contribuição de cada variável para cada grupo pode ser diferente, *i.e.*, cada grupo pode ter um conjunto diferente de variáveis importantes. Na distância adaptativa global, todas as variáveis possuem o mesmo peso para todos os grupos. Em ambas as abordagens, a derivação das expressões matemáticas para a obtenção dos pesos das variáveis foi feita considerando dois tipos de restrições: na restrição de soma, assume-se que a soma dos pesos das variáveis deve ser igual a um, enquanto que na restrição de produto, assume-se que o produto dos pesos das variáveis deve ser igual a um (GUSTAFSON e KESSEL, 1979; DE CARVALHO; TENORIO; CAVALCANTI JUNIOR, 2006).

Desse modo, nesta tese foram desenvolvidas versões com ponderação de variáveis local com restrição de soma e produto dos componentes da matriz de pesos e distância com ponderação global de variáveis com restrição de soma e produto dos componentes da matriz de pesos. Foram propostos quatro tipos de distâncias adaptativas para algoritmos de agrupamento semissupervisionados. No total, são cinco algoritmos de agrupamento semissupervisionado sem ponderação automática de variáveis mais quatro algoritmos com ponderação automática de variáveis baseados em cada um desses cinco, totalizando 20 algoritmos com distância adaptativa. Desses, apenas 2 foram propostos na literatura e 18 propostos nesta tese.

4.3 CONSIDERAÇÕES FINAIS

Note que para as distâncias adaptativas locais, quanto mais próximos os objetos estão do protótipo de um determinado grupo *fuzzy* relativo à uma determinada variável, maior é o peso da relevância dessa variável sobre este grupo *fuzzy*. Por outro lado, quanto mais longe os objetos estão do protótipo, menor é o peso relevância dessa variável sobre este grupo *fuzzy*. Esta propriedade é avaliada por todos os algoritmos de agrupamento semissupervisionado com distâncias adaptativas locais descritos neste trabalho.

O mesmo pode-se falar das distâncias adaptativas globais, porém, nas distâncias adaptativas globais, temos uma média das distâncias desses objetos. Sendo assim, quanto mais próximo os objetos estão do protótipo de todos os grupos *fuzzy* relativo à uma determinada variável, maior é o peso da relevância dessa variável. Por outro lado, quanto mais longe os objetos estão dos protótipos de todos os grupos, menor é o peso da relevância

dessa variável. Esta propriedade é avaliada para todos os algoritmos de agrupamento semissupervisionado com distâncias adaptativas globais descritos neste trabalho.

5 VALIDAÇÃO EXPERIMENTAL

Neste capítulo são apresentados detalhes dos experimentos realizados com os algoritmos de agrupamento semissupervisionado. A Seção 5.1 descreve as bases de dados utilizadas nos experimentos. A Seção 5.2 apresenta a metodologia dos experimentos utilizada neste trabalho para avaliação dos algoritmos de agrupamento semissupervisionados com ponderação automática de variáveis.

5.1 BASE DE DADOS

Neste trabalho utilizamos dois tipos de bases de dados. As bases de dados reais para comparar o desempenho dos algoritmos de agrupamento semissupervisionados em problemas reais e as bases de dados simulados que são utilizadas para responder perguntas sobre a diferença entre o comportamento dos algoritmos com ponderação automática de variáveis local e global.

5.1.1 Bases de dados reais

Nesta seção, as bases de dados de problemas reais utilizadas neste trabalho são apresentadas. Ao todo, foram escolhidas 18 bases de dados de *benchmark* retiradas do repositório digital UCI (*http://archive.ics.uci.edu/ml/*) mais uma base de dados para o problema de agrupamento de expressão gênica retirada do trabalho (MACARIO et al., 2012), totalizando 19 bases de dados de problemas reais. A Tabela 18 resume essas bases de dados. Todas as bases de dados são descritas por matriz de objetos $\times$ atributos de valores reais.

A base de dados *Australia Credit* contém atributos de aplicações de cartões de crédito utilizados na Austrália com a finalidade de aprovar ou não um novo cartão de crédito ao cliente. São 690 exemplos e 14 características para aprovação ou não do cliente.

A base de dados *Balance Scale*, foi gerada para modelar resultados de testes psicológicos experimentais. Cada exemplo é classificado como tendo tendência para a direita, para a esquerda, ou equilibrada. São 625 exemplos, sendo 49 equilibrados, 288 para esquerda e 288 para a direita.

A base de dados *CMC* tem como objetivo prever a escolha do método contraceptivo de mulheres da Indonésia através de características de cunho social. São 1473 exemplos, 10 atributos e 3 tipos de contraceptivos.

A base de dados Diabetes consiste numa coleção de padrões que indicam a existência da diabetes ou não. Essa base contém um conjunto de 768 padrões, sendo que 500

Tabela 18 – Resumo das bases de dados reais retiradas do repositório UCI.

Base de dados	n (Número de objetos)	p (Número atributos)	H (Número *a priori* de classes)
Australia Credit	690	14	2
Balance Scale	625	4	3
CMC	1473	10	3
Diabetes	768	8	2
E. Coli	336	7	8
Glass	214	9	7
Haberman	306	4	2
Hayes-Hoth	160	4	3
Heart Statlog	270	14	2
ICU	300	19	2
Iris	150	4	3
Image Segmentation	210	19	7
Libras	45	90	15
Mammographic	961	5	2
Spam	4601	57	2
Vehicle	946	18	4
Teaching	151	5	3
Yeast	603	300	2
Wine	178	13	3

representam a ausência de diabetes e os outros 268 representam a presença da diabetes. Existem 8 atributos com características sanguíneas e corporais, mais a idade. O objetivo é informar se um padrão pertence ao grupo de diabéticos ou não.

A base de dados *E. Coli* trata-se de características para predição da localização de uma proteína na célula da bactéria E. Coli. São 336 exemplos em que a proteína pode estar localizada em 8 possíveis lugares. Essa base de dados utiliza 7 características da célula.

A base de dados *Glass* contém 214 padrões para classificação de 7 tipos de vidros motivada pela investigação criminológica. Na cena do crime, o vidro pode ser usado como prova se for identificado corretamente.

A base de dados *Haberman* é parte de um estudo para saber a sobrevivência de pacientes que foram submetidos a cirurgia de câncer de mama. São 4 atributos de 306 pacientes.

A base de dados *Hayes-Roth* possui 132 exemplos com 5 atributos com informações de fictícias para que o algoritmo seja capaz de aprender 3 conceitos.

A base de dados *Heart Statlog* é parte de um estudo para diagnosticar se alguém possui ou não uma doença do coração. São 14 atributos de resultados de exames realizados

por 270 pacientes.

A base de dados ICU consiste em 200 exemplos de um estudo maior sobre a sobrevida de pacientes após a admissão a uma unidade de terapia intensiva (UTI). O estudo utilizou regressão logística para prever a probabilidade de sobrevida para esses pacientes até a alta do hospital. Dezenove características possíveis foram observadas.

A base de dados *Iris* é uma das base dados mais utilizadas em reconhecimento de padrões. Essa base consiste num conjunto de 150 padrões que descrevem 3 tipos de plantas da família Iris. São 4 atributos que descrevem as medidas de largura e comprimento de pétalas e sépalas da planta. Existem 50 padrões para representar cada uma das espécies. A tarefa é formar 3 grupos que representam cada um dos tipos de Iris.

A base de dados *Image Segmentation* possui 19 características extraídas de uma região de pixels 3×3 de imagens de cenas naturais. O objetivo é segmentar corretamente as imagens em 7 tipos possíveis de textura.

A base de dados Libra contém 15 tipos de movimentos da linguagem brasileira de sinais com 24 exemplos de cada. São extraídas 90 características do movimento realizado pela mão numa período de tempo.

A base de dados *Mammography* consiste de 961 exames de mamografia (para verificar se um tumor de câncer de mama é maligno ou benigno) com 5 características do paciente e do exame.

A base de dados *Spam* consiste de uma coleção de mensagens eletrônicas chamadas de *e-mails*. O conteúdo dessas mensagens é classificado como *spam* (lixo eletrônico) ou não *spam*. A base contém 4601 mensagens descrita através de 57 atributos. Os atributos, em sua maioria representam o TF-IDF (*Term-frequency Inverse-Document-Frequency*) (SALTON e BUCKLEY, 1988) de uma determinada palavra.

O objetivo da base de dados *vehicle* é classificar uma determinada silhueta como um dos quatro tipos de veículos, com 18 características extraídas da silhueta. O veículo pode ser visto a partir de um dos muitos ângulos diferentes.

A base de dados *Teaching* (Tae) consiste em avaliações de desempenho de ensino de três semestres regulares e dois semestres de verão por de 151 professores assistentes do Departamento de Estatística da Universidade de Wisconsin-Madison. Os escores foram divididos em 3 categorias ("baixa", "média"e "alta").

A base de dados *Wine* é uma das base dados mais utilizadas para testar novos algoritmos. Consiste num conjunto de 178 padrões que descrevem 3 tipos de vinhos. Existem 13 atributos que descrevem fatores químicos desses vinhos. As classes são desbalanceadas, sendo 59 padrões para representar o vinho 1, 71 para o vinho 2 e 48 padrões para o vinho 3. A tarefa é formar 3 grupos que representem cada um dos tipos de vinho.

A base de dados *Yeast* foi desenvolvida para descoberta de novas funções biológicas de genes da cevada (*yeast cerevisae*). Esta base de dados possui alta dimensionalidade, contendo 300 experimentos de microarranjos de deleção do gene e tratamentos com drogas. No trabalho de Macario et al. (MACARIO et al., 2012), esta base de dados foi utilizada para avaliar algoritmos de agrupamento semissupervisionado na tarefa de encontrar novas funções biológicas para genes utilizando informações sobre funções já conhecidas de alguns genes. São suas funções biológicas possíveis: controle do ciclo celular e célula mitocôndria.

5.1.2 Conjuntos de dados simulados

Para demonstrar a capacidade dos métodos de agrupamento semissupervisionado com ponderação automática das variáveis aprenderem dinamicamente os pesos das variáveis apropriadamente, foram consideradas duas configurações de conjuntos de dados em três dimensões com quatro e três grupos, respectivamente, cada grupo constituído por cem padrões. Essa metodologia foi adotada a partir do trabalho de Ferreira e De Carvalho (FERREIRA e DE CARVALHO, 2014).

5.1.2.1 Conjunto de dados simulados 1

O conjunto de dados simulados 1 Figura 6(a) é constituído por 400 padrões e está dividido em quatro grupos, cada um com 100 padrões descritos por vetores em três dimensões gerados aleatoriamente a partir de uma distribuição normal com um vetor de médias e uma matriz de variância para cada grupo. Cada grupo está bem definido em somente duas das três variáveis. Os vetores de médias e as estruturas de variância de cada grupo estão apresentados na Tabela 19. A correlação entre as variáveis é definida igual a zero em todos os grupos.

Retirada do trabalho de Ferreira e De Carvalho (FERREIRA e DE CARVALHO, 2014), a partir da Figura 6, pode ser observado que os grupos 1 e 2 são descritos pelas variáveis x_1 e x_2, ou seja, x_1 e x_2 são relevantes para a definição dos grupos 1 e 2 Figura 6(b), enquanto que os grupos 3 e 4 são claramente descritos pela variáveis x_2 e x_3, ou seja, x_2 e x_3 são relevantes para a definição dos grupos 3 e 4 Figura 6(c). Isso significa que, para os grupos 1 e 2, a variável x_3 representa ruído, enquanto que para os grupos 3 e 4, a variável x_1 representa ruído, como pode ser notado a partir da Figura 6(d).

5.1.2.2 Conjunto de dados simulados 2

O conjunto de dados simulados 2 Figura 7(a) é constituído por 300 padrões e está dividido em três grupos, cada um constituído por 100 padrões. As variáveis x_1 e x_2 foram geradas a partir de distribuições normais com um vetor de médias específico e uma mesma matriz de variância todas as classes. A variável x_3 é gerada como uma combinação linear das variáveis x_1 e x_2 mais um ruído aleatório gerado a partir de uma distribuição

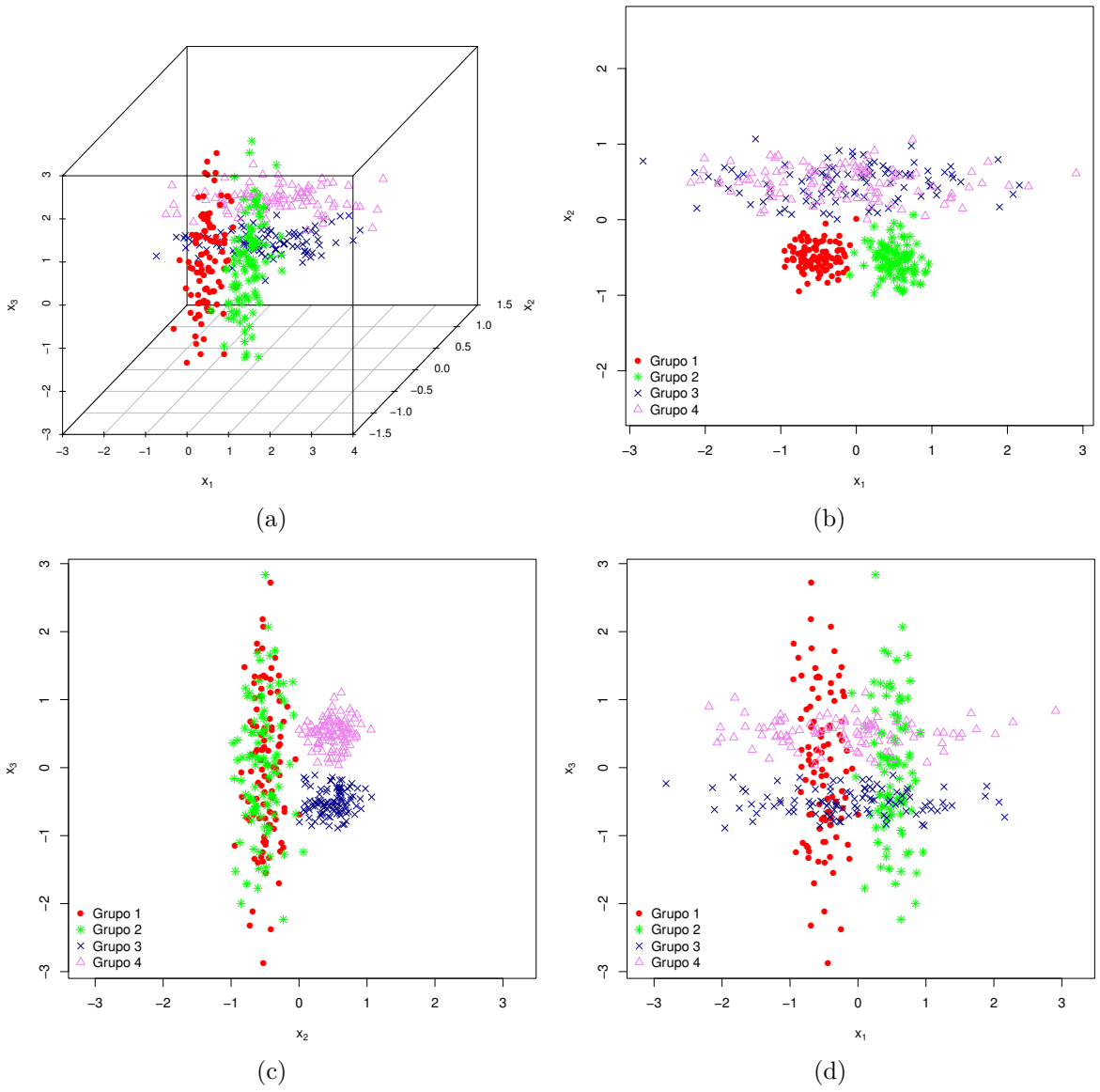


Figura 6 – Conjunto de dados simulados 1.

Tabela 19 – Configuração do conjunto de dados simulados 1.

$\boldsymbol{\mu}$	Grupo 1	Grupo 2	Grupo 3	Grupo 4
μ_1	$-0{,}5$	$0{,}5$	$0{,}0$	$0{,}0$
μ_2	$-0{,}5$	$-0{,}5$	$0{,}5$	$0{,}5$
μ_3	$0{,}0$	$0{,}0$	$-0{,}5$	$0{,}5$
$\boldsymbol{\Sigma}$	Grupo 1	Grupo 2	Grupo 3	Grupo 4
σ_1^2	$0{,}04$	$0{,}04$	$1{,}00$	$1{,}00$
σ_2^2	$0{,}04$	$0{,}04$	$0{,}04$	$0{,}04$
σ_3^2	$1{,}00$	$1{,}00$	$0{,}04$	$0{,}04$

normal padrão, *i.e.*, $x_{3i} = 2x_{1i} - 1.5x_{2i} + u_i$, onde $u_i \sim N(0,1)$, $i = 1, \ldots, n$. Os vetores de médias e as estruturas de variância de cada grupo com relação às variáveis x_1 e x_2 estão apresentados na Tabela 20. A correlação entre as variáveis x_1 e x_2 é definida igual a zero em todos os grupos. Em virtude da combinação linear adotada, as variáveis x_1 e x_3

são positivamente correlacionadas em cada grupo, enquanto que as variáveis x_2 e x_3 são negativamente correlacionadas em cada grupo.

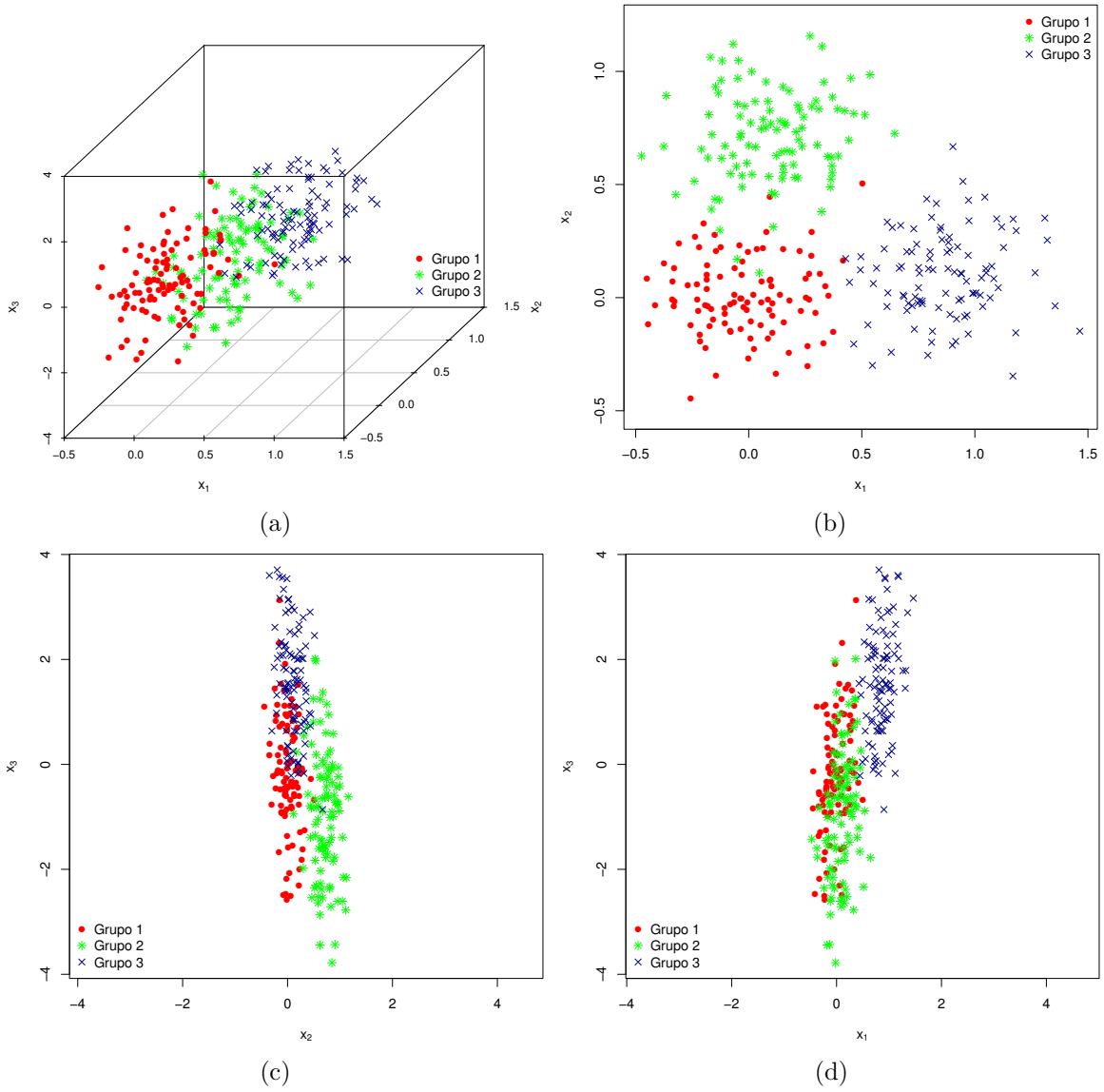


Figura 7 – Conjunto de dados simulados 2.

Pode-se observar, a partir da Figura 7, retirada do trabalho de Ferreira e De Carvalho (FERREIRA e DE CARVALHO, 2014), que as variáveis x_1 e x_2 são relevantes para os grupos 1, 2 e 3 (Figura 7(b)), *i.e.*, o conjunto de variáveis relevantes é o mesmo para todas os grupos. Pode-se observar também que, para todos os grupos, a variável x_3 representa ruído, como pode ser notado a partir das Figuras 7(c) e 7(d).

5.2 EXPERIMENTOS

Os experimentos são divididos em dois tipos. No primeiro tipo de experimento, os novos algoritmos são comparados com outros algoritmos de agrupamento semissupervisionado nas tarefas de agrupamento em problemas reais. O segundo tipo de experimento é

voltado para bases de dados simulados com o objetivo de prever o comportamento dos algoritmos de agrupamentos semissupervisionados com ponderação automática de variáveis descritos neste trabalho.

5.2.1 Agrupamento semissupervisionado

Nesta seção, vamos focar no agrupamento de dados parcialmente rotulados. A metodologia utilizada neste trabalho possui como objetivo principal validar os rótulos encontrados pelos algoritmos propostos com diferentes porcentagens de dados rotulados disponível para o treinamento desses algoritmos na tarefa de agrupamento. A metodologia utilizada neste trabalho é baseada nos trabalhos de Amini e Gallinary (AMINI e GALLINARI, 2005) e Pedrycz e Waletzky (PEDRYCZ e WALETZKY, 1997).

Os padrões da base de dados são divididos em 2 conjuntos, o conjunto de dados rotulados e o conjunto de dados não rotulados. O conjunto de dados rotulados contém uma certa porcentagem de exemplos rotulados do total da base de dados. O interessante dessa divisão é demonstrar que com poucos dados rotulados, algoritmos conseguem obter bons desempenhos. Assim, as seguintes porcentagens de dados rotulados foram consideradas: 1%, 5%, 10%, 20%, 30%, 40% e 50% do total da base. O restante do conjunto de dados faz parte do segundo conjunto, o de dados não rotulados. Assim, para os valores respectivos de base rotulada, esses conjuntos utilizam as porcentagens 99%, 95%, 90%, 80%, 70%, 60% e 50% do total da base. Para uma porcentagem fixa de exemplos rotulados, cada algoritmo de agrupamento semissupervisionado é inicializado com uma execução do algoritmo clássico *Fuzzy C-Means*. São realizadas 100 repetições em cada configuração do experimento, cada repetição possui diferentes exemplos nos subconjuntos de dados rotulados e não rotulados. As etapas do experimento são apresentadas abaixo:

1. **Particionamento dos Dados Originais:** Os dados originais são particionados em dois subconjuntos (rotulados e não rotulados). Nesta etapa, os dados originais são particionados de forma estratificada. Dessa maneira, a distribuição de dados das classes original é mantida, ou seja, se existe originalmente 30% de uma classe A e 70% de uma classe B, esses valores são mantidos nos conjuntos de dados rotulados e não

Tabela 20 – Configuração do conjunto de dados simulados 2.

$\boldsymbol{\mu}$	Classe 1	Classe 2	Classe 3
μ_1	0,0	0,1	0,9
μ_2	0,0	0,7	0,1
$\boldsymbol{\Sigma}$	Classe 1	Classe 2	Classe 3
σ_1^2	0,04	0,04	0,04
σ_2^2	0,04	0,04	0,04

rotulados. Os exemplos que compõem cada um dos dois conjuntos são selecionados aleatoriamente na base de dados;

2. **Execução do Algoritmo:** Execução do algoritmo tendo como entrada os dois subconjuntos obtidos na etapa 1 e tendo como saída uma partição *fuzzy*;

3. **Análise dos Rótulos:** Nesta etapa, os rótulos obtidos são comparados com os rótulos originais através das medidas de acurácia e do índice de informação mútua normalizado para avaliação de algoritmos de agrupamento (ver Seção 2.5).

Nos experimentos realizados, os algoritmos semissupervisionados que contêm o parâmetro alfa, utilizaram alfa igual a 0,1; 0,5; 1; 5; 10; 50 e 100, $\varepsilon = 10^{-7}$ e $T = 300$. No algoritmo BSSC, a taxa de aprendizado β é fixada em 0,6 para todas as bases de dados (ver Ref. (BOUCHACHIA e PEDRYCZ, 2006a)). Finalmente, o número de grupos é fixado no mesmo número de classes conhecidas, conforme os trabalhos (ZENG et al., 2012; YIN; SHU; HUANG, 2012; MEI e CHENVC, 2013).

A partir deste experimento foram realizadas várias análises dos resultados. A primeira análise é realizada para verificar o efeito do parâmetro alfa no desempenho destes algoritmos. A segunda análise é para verificar se as versões com ponderação automática de variáveis foram melhores que as versões dos algoritmos com distância Euclidiana padrão. A terceira, é feita uma análise comparativa do desempenho de todos os algoritmos de agrupamento semissupervisionados considerados nesta tese. Por fim, uma estratégia para selecionar exemplos rotulados é avaliada.

5.2.2 Experimentos com bases de dados reais

O objetivo desse experimento é analisar o desempenho dos algoritmos de agrupamento semissupervisionados com ponderação automática de variáveis considerados nessa tese em problemas reais. As análises realizadas nesse experimento são explicadas a seguir.

5.2.2.1 Análise do parâmetro alfa (α)

O parâmetro alfa é um valor positivo que multiplica o termo supervisionado da maioria dos algoritmos semissupervisionados deste trabalho. A única exceção é o algoritmo semissupervisionado padrão (sSSC) e suas versões com distância adaptativa. O objetivo deste parâmetro é ponderar o termo supervisionado desses algoritmos. O objetivo da análise desse parâmetro é verificar a influência desse parâmetro no desempenho dos algoritmos. Desse modo, o parâmetro alfa é avaliado para cada um dos algoritmos semissupervisionados por 7 valores possíveis: 0,1; 0,5; 1; 5; 10; 50 e 100. O teste estatístico de Friedman (ver Seção 5.2.4) é utilizado para analisar qual as melhores configurações para o parâmetro alfa de cada um desses algoritmos.

5.2.2.2 Análise da ponderação automática de variáveis

Para atestar a hipótese que os algoritmos com ponderação automática de variáveis são melhores que suas respectivas versões sem ponderação automática de variáveis, é realizada uma análise comparativa do desempenho das versões desses algoritmos com distâncias adaptativas *versus* o algoritmo com distância Euclideana que o originou. Nessa análise, foram utilizados os valores de alfa selecionados na primeira análise. Os algoritmos considerados nessa comparação foram o algoritmo BSSC *versus* suas respectivas versões com ponderação automática de variáveis, o algoritmo sSSC *versus* suas respectivas versões com ponderação automática de variáveis, o algoritmo ESSC *versus* suas respectivas versões com ponderação automática de variáveis, o algoritmo NebFuzz *versus* suas respectivas versões com ponderação automática de variáveis e o algoritmo Isodata-PS *versus* suas respectivas versões com ponderação automática de variáveis.

5.2.2.3 Comparação entre algoritmos semissupervisionados

Depois, é realizada uma análise comparativa entre todos os algoritmos de agrupamento semissupervisionados considerados neste trabalho. Desse modo, pode-se verificar quais algoritmos se destacam em termos de desempenho.

5.2.2.4 Seleção de rótulos

Os algoritmos fornecem uma estrutura matemática para os dados de agrupamento representados por matrizes de características e utilizam o conhecimento prévio de domínio (na forma de rótulos) para orientar o processo de agrupamento. Neste trabalho, vamos supor que os dados são parcialmente rotulados. Obviamente, um número maior de exemplos rotulados iria melhorar ainda mais os resultados. No entanto, a taxa de melhoria no desempenho pode não ser uniforme, e pode não justificar o esforço para adquirir os rótulos. Em geral, a quantidade de dados rotulados (ou o número de respostas de um especialista) é dependente da aplicação. Por exemplo, em aplicações críticas, o usuário pode estar disposto a identificar um grande número de exemplos para uma pequena melhoria. Em outras aplicações, onde a precisão não é crítica e onde o usuário pode ser impaciente (por exemplo, em texto ou recuperação de imagens), não podemos esperar que o usuário ou especialista rotulem mais do que alguns exemplos. Neste trabalho é considerando o número de exemplos a serem rotulados dos exemplos totais.

Neste ponto, temos usado a estratégia de Frigui e Hwang (FRIGUI e HWANG, 2008), assumindo que os exemplos de fronteira são os pontos mais informativos, e que esses exemplos devem ser rotulados. Em primeiro lugar, nós executamos o algoritmo FCM para algumas iterações. Então, identificamos alguns exemplos na fronteira (exemplos com valores de graus de pertinência *fuzzy* parecidos para todos os grupos). Para identificar esses exemplos, multiplicamos os graus de pertinência de um exemplo candidato e_i para

todos os grupos $C_k(k = 1, \ldots, K)$, tal como apresentado na Equação (5.1). Os exemplos com valores mais altos de $prod_i$, são os pontos de fronteira. Os exemplos rotulados por esse método são ordenados de acordo com o valor de $prod_i$. Desse modo, escolhemos a porcentagem requerida pelo experimento de acordo com a ordem dos valores de $prod_i$, por exemplo, se a configuração do experimento for 10%, seleciona-se essa porcentagem de exemplos com os maiores valores de $prod_i$.

$$prod_i = \prod_{k=1}^{K} u_{ik}(i = 1, \ldots, n) \tag{5.1}$$

Note o exemplo com dois grupos apresentado na Tabela 21 que o exemplo e_2 é o ponto de fronteira, pois os graus de pertinência para os dois grupos possuem valores muito próximos, enquanto os graus de pertinência do exemplo e_1 possuem uma diferença maior entre eles.

Tabela 21 – Exemplo de cálculo do ponto de fronteira

Exemplo	$k = 1$	$k = 2$	$prod_i$
e_1	$u_{11} = 0,9$	$u_{12} = 0,1$	$prod_1 = 0,09$
e_2	$u_{21} = 0,49$	$u_{22} = 0,51$	$prod_2 = 0,2499$

Desse modo, foi realizado um experimento utilizando essa técnica de seleção de exemplos para serem rotulados para avaliar a efetividade dessa técnica.

5.2.3 Experimentos com bases de dados simuladas

No experimento com bases de dados simulados, o objetivo é responder a perguntas sobre o comportamento dos algoritmos com ponderação automática de variáveis local e global. Na base de dados simulados 1, observe na Figura 6 que os grupos são definidos claramente pela variável x_1, ou pela variável x_2. Desse modo, espera-se que os algoritmos com ponderação local sejam capazes de atribuir peso maior para as variáveis que definem o grupo e como consequência, obtenham um desempenho melhor nessa base de dados. Na base de dados simuladas 2, note na Figura 7 que as variáveis possuem relevância igual na definição dos grupos. Nesse caso, o melhor desempenho pode ser obtido pelos algoritmos com ponderação global, porém, os algoritmos com ponderação local também podem conseguir bons desempenhos.

Nesse experimento, foram geradas 25 réplicas de cada uma das bases de dados simuladas. A metodologia empregada é exatamente a mesma descrita na seção anterior. Primeiro, é realizada a análise do melhor alfa (α) pra essas bases de dados. Depois, é realizada uma análise comparativa em termos de desempenho entre todos os algoritmos com ponderação automática de variáveis considerados nesta tese.

5.2.4 Teste de Friedman

O teste estatístico de Friedman (DEMSAR, 2006) é um teste não paramétrico para comparar o desempenho de vários algoritmos executados em várias bases de dados. O número mínimo recomendado de algoritmos é $(A > 5)$ e o número de base de dados é $(B > 10)$ (DEMSAR, 2006).

O primeiro passo para realização do teste de Friedman é a classificação da posição r_a^b em termos de desempenho de cada algoritmo a, em cada base de dados b. O objetivo dessa etapa é construir um *ranking* em termos de desempenho dos algoritmos analisados. O *ranking* de Friedman (SHESKIN, 2006) é calculado para ordenar estatisticamente os algoritmos (esse *ranking* aumenta com o erro do algoritmo), levando em conta todo o conjunto de base de dados. No trabalho de Delgado *et al.* (FERNANDEZ-DELGADO et al., 2014), esse *ranking* é utilizado para comparação de 179 algoritmos em 121 bases de dados. Na construção do *ranking* de Friedman, o algoritmo com melhor desempenho é colocado na posição 1, o segundo na posição 2, e assim por diante. Em caso de empate, a média da posição desses algoritmos é calculada. Observe o exemplo da Tabela 22 que apresenta a construção do *ranking* de Friedman de um algoritmo com variações no parâmetro alfa. Nesse exemplo temos $A = 7$ e $B = 19$. Na última linha, apresentamos o *ranking* de Friedman $R_a = \frac{1}{A}\sum_b r_a^b$.

Tabela 22 – Exemplo da construção do *ranking* de Friedman de um algoritmo com variações no parâmetro alfa.

Base de dados	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
Australian	66,69(2)	66,68(3)	66,71(1)	66,65(4,5)	66,55(7)	66,63(6)	66,65(4,5)
Balance	78,57(1)	78,31(3)	78,49(2)	57,53(5)	51,32(7)	61,44(4)	53,43(6)
CMC	43,44(5)	43,49(4)	43,28(6)	45,84(2)	46,07(1)	42,82(7)	44,33(3)
Diabetes	63,53(2)	62,35(4)	63,59(1)	62,16(5)	51,80(6)	62,73(3)	50,53(7)
Ecoli	77,60(1)	76,96(2)	76,15(3)	62,20(5)	38,79(7)	74,04(4)	44,70(6)
Glass	47,66(6)	47,77(4)	47,93(3)	48,23(1)	47,03(7)	48,05(2)	47,69(5)
Haberman	60,14(1)	59,08(3)	59,28(2)	57,64(4)	52,65(7)	56,82(5)	52,95(6)
Hayes	48,34(4)	49,08(3)	49,11(2)	47,69(5)	42,66(7)	49,42(1)	42,75(6)
Heart	62,02(3)	62,35(1)	62,11(2)	61,58(4)	51,79(7)	61,38(5)	60,71(6)
ICU	53,81(5)	54,02(3)	54,41(1)	54,27(2)	53,23(6)	53,83(4)	52,67(7)
Iris	71,14(4)	71,15(3)	71,37(2)	65,75(5)	32,15(7)	71,98(1)	36,86(6)
Libras	22,96(6)	13,38(7)	33,11(5)	47,84(1)	47,60(3)	46,87(4)	47,80(2)
Mammographic	68,81(2)	68,90(1)	68,69(4)	68,50(5)	66,02(7)	68,75(3)	66,77(6)
Segmentation	49,54(3)	49,22(5)	50,10(1)	46,85(7)	49,35(4)	49,87(2)	48,55(6)
Spam	69,52(6,5)	69,59(1)	69,56(2,5)	69,52(6,5)	69,56(2,5)	69,55(4)	69,54(5)
Teaching	40,25(1)	40,18(2)	39,82(5)	39,64(6)	40,04(4)	40,15(3)	38,43(7)
Vehicle	40,48(3)	40,61(1)	40,47(4)	40,34(5)	34,95(6)	40,57(2)	34,88(7)
Wine	70,44(1)	70,25(2)	69,92(3)	68,88(5)	39,67(7)	69,65(4)	45,11(6)
Yeast	72,55(3)	72,61(2)	72,73(1)	72,54(4)	72,31(6,5)	72,47(5)	72,31(6,5)
Ranking de Friedman	3,13	2,84	**2,66**	4,32	5,74	3,63	5,68

O teste de Friedman compara a média da classificação de cada algoritmo (R_a). A hipótese nula, quando os algoritmos são equivalentes, é aceita quando a média da classificação R_a de todos os algoritmos é equivalente à estatística de Friedman com $(A-1)$

graus de liberdade:

$$\chi_F^2 = \frac{12N}{k(k+1)} \left[\sum_a R_a^2 - \frac{k(k+1)^2}{4} \right] \tag{5.2}$$

Para o exemplo da Tabela 22, o resultado da Equação (5.2) é ($\chi_F^2 = 40,652$). Porém, Iman and Davenport (IMAN e DAVENPORT, 1980) consideram χ_F^2 muito conservativo e propuseram uma estatística melhor distribuída de acordo com a distribuição F com (A-1) e (A-1)(B-1) graus de liberdade:

$$F_f = \frac{(B-1)\chi_F^2}{B(A-1) - \chi_F^2} \tag{5.3}$$

Para o caso do exemplo anterior, rejeitamos a hipótese nula caso ($F_f > F$), o que acontece, pois $F_f = 9,976$ e $F(6,108) = 2,184$, com um nível confiança de 90%.

Caso a hipótese nula seja rejeitada, procedemos com um pós-teste. O teste de Nemenyi (NEMENYI, 1963) é realizado quando todos os algoritmos são comparados entre si. O desempenho de dois algoritmos são significativamente diferentes se a média da classificação desses algoritmos possui uma diferença maior que o valor crítico:

$$CD = q_{conf} \sqrt{\frac{A(A+1)}{6B}} \tag{5.4}$$

onde $conf$ é o grau de confiança para o teste. Este valor costuma ser 90%, 95% ou 99%. Para o exemplo, o valor é $CD = 1,262$. Se olharmos o exemplo anterior, percebe-se que o algoritmo com $\alpha = 1$ difere significativamente do algoritmo com $\alpha = 10$.

Quando todos os algoritmos são comparados com um algoritmo controle, podemos utilizar um procedimento como o teste de Holm (HOLM, 1979) ou similares. Esse teste avalia as hipóteses ordenadas pelo seu nível de significância $p_1 \leq p_2 \leq \ldots \leq p_{a-1}$, onde o nível de significância é dado pelo valor-p. O valor-p representa a probabilidade de erro envolvida em aceitar o resultado observado como válido, desse modo, quanto menor o valor-p, mais confiança temos que a hipótese nula seja rejeitada. O teste de Holm compara o valor-p mais significativo (menor valor) com o resultado de ($conf/(B-a)$). Caso o primeiro valor-p esteja abaixo de $conf/(B-1)$, essa hipótese é rejeitada e segue comparando o próximo valor-p com $conf/(B-2)$. Se a segunda hipótese é rejeitada, segue comparando com o terceiro valor-p, e assim por diante. Caso uma determinada hipótese nula não seja rejeitada, todas as hipóteses restantes são rejeitadas. Dessa forma, o teste de Holm retorna os algoritmos que são significativamente melhores que o algoritmo controle na ordem de significância. Para o nosso exemplo, observe a Tabela 23, que mostra que o algoritmo com $\alpha = 10$ e $\alpha = 100$ possuem desempenho significativamente menor que o algoritmo com $\alpha = 1$, $\alpha = 0,5$, $\alpha = 0,1$ e por fim $\alpha = 50$, nessa ordem.

5.2.5 Teste de Wilcoxon

O teste estatístico de Wilcoxon (WILCOXON, 1945) é uma opção não paramétrica bastante conhecida para comparação pareada entre o desempenho de dois algoritmos de

Tabela 23 – Comparação entre o algoritmo analisado com diferentes parâmetros de alfa utilizando o teste de Friedman e o pós-teste de Holm a um nível significância de 5%.

Algoritmo	Pior que (ordem de significância)			
$\alpha = 10$	$\alpha = 1$	$\geq \alpha = 0,5$	$\geq \alpha = 0,1$	$\geq \alpha = 50$
$\alpha = 100$	$\alpha = 1$	$\geq \alpha = 0,5$	$\geq \alpha = 0,1$	$\geq \alpha = 50$

classificação ou agrupamento. O teste de Wilcoxon realiza a classificação de acordo com a diferença absoluta entre os resultados de cada algoritmo para cada base de dados. Ele classifica a menor diferença (pode ser 0) entre o desempenho entre os dois algoritmos em primeiro, seguindo da segunda menor diferença em segundo e assim por diante. Em caso de empate, é calculada a média da classificação dos casos de empate.

Após realizar a classificação do desempenho dos algoritmos em termos da diferença entre seus resultados, soma-se a classificação onde o algoritmo comparado é maior que o outro, e depois soma a classificação em que o algoritmo é pior que o outro. Assim, temos dois tipos de classificação, R^+ e R^- respectivamente. O resultado do teste de Wilcoxon é $T = (\min(R^+, R^-))$. A maioria dos livros de estatística apresenta uma tabela que varia em relação ao número de bases de dados utilizadas no experimento. Esse número deve ser maior que 6 para um nível de confiança de 0.95 (DEMSAR, 2006). O valor T é comparado com o número dessa tabela, caso T seja menor, a hipótese nula é rejeitada.

5.3 SÍNTESE DO CAPÍTULO

Neste capítulo explicamos em detalhes a metodologia empregada em todos os experimentos realizados neste trabalho. A execução dos experimentos produziu resultados que são apresentados através de gráficos, explicações e discussões no próximo capítulo.

6 RESULTADOS E DISCUSSÃO

Os resultados da análise comparativa entre os algoritmos explorados neste trabalho são apresentados e analisados neste capítulo. A Seção 6.1 descreve os resultados obtidos na análise comparativa dos métodos de agrupamento semissupervisionados na tarefa de agrupamento em bases de dados de problemas reais. A Seção 6.2 descreve os resultados obtidos na análise comparativa do desempenho nas bases de dados simulados.

6.1 BASES DE DADOS REAIS

Os experimentos apresentados nesta seção foram realizados com o objetivo de avaliar os algoritmos de agrupamento semissupervisionado com ponderação automática de variáveis propostos na tarefa de agrupamento em problemas reais descritos na Seção 5.1.1. Os algoritmos avaliados foram o BSSC, sSSC, ESSC, Isodata-PS, NebFuzz e suas respectivas versões com distâncias adaptativas com ponderação de variáveis local e global, com restrições da soma e produto dos pesos das variáveis igual a um. A análise dos resultados desse experimento é dividida em quatro partes. A primeira avalia o efeito do parâmetro alfa para cada algoritmo individualmente. A partir desse estudo, é escolhido o melhor valor do parâmetro alfa para cada algoritmo de agrupamento semissupervisionado levando-se em conta a quantidade de dados rotulados disponíveis para o treinamento. A segunda análise avalia o efeito da ponderação automática de variáveis em contraponto aos algoritmos de agrupamento semissupervisionado com distância Euclidiana. A terceira parte realiza um estudo de comparação em termos de desempenho entre todos os algoritmos apresentados neste trabalho. A última parte analisa a seleção de exemplos rotulados localizados na fronteira dos agrupamentos formados pelo algoritmo *Fuzzy C-Means* explicado na Seção 5.2.2.4.

6.1.1 Análise do parâmetro alfa (α)

O *ranking* de Friedman é construído para cada algoritmo avaliado, de acordo com o desempenho da acurácia e o índice NMI (ver Seção 2.5) para as diferentes configurações de alfa (α: 0,1; 0,5; 1; 5; 10; 50 e 100) levando-se em conta cada configuração de dados rotulados (1%, 5%, 10%, 20%, 30%, 40% e 50%). A partir do *ranking* de Friedman, foi utilizado o teste de Friedman com 90% de confiança para avaliar se diferentes configurações de alfa possuem ou não desempenhos similares. Caso o desempenho de diferentes configurações de alfa possuam diferenças estatísticas significativas no Teste de Friedman, o pós-teste de Nemenyi foi realizado para verificar quais valores de alfa tiveram melhor desempenho. Os valores de alfa que forem melhores que, pelo menos, um outro valor de alfa, são apresentados em negrito nas tabelas abaixo. Os resultados apresentados sem nenhuma

configuração em negrito não tiveram diferença estatística significativa. Os melhores valores selecionados de acordo com o desempenho medido pelos índices ACC (acurácia) ou NMI (informação mútua normalizada) são apresentados no final dessa seção.

O primeiro algoritmo analisado através do *ranking* de Friedman apresentados na Tabela 24 é o algoritmo BSSC. De acordo com o Teste de Friedman, os algoritmos com configurações menores que 20% de dados rotulados não possuem uma diferença estatística significativa em termos de desempenho para os diferentes valores de alfa. De 30% até 50% de dados rotulados, os parâmetros ($\alpha = 1$) e ($\alpha = 50$) tiveram melhor desempenho para os índices ACC e NMI, respectivamente.

Tabela 24 – *Ranking* de Friedman para diferentes valores do parâmetro alfa do algoritmo BSSC

Acurácia							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	3,82	3,76	3,74	4,21	4,05	4,37	4,05
5%	3,84	3,37	4,13	4,13	4,84	3,58	4,11
10%	3,42	3,58	4,05	4,13	4,39	4,32	4,11
20%	3,89	4,21	3,68	4,11	4,13	3,76	4,21
30%	4,92	4,34	3,16	3,89	4,03	3,21	4,45
40%	4,92	4,97	**2,68**	3,79	3,95	3,63	4,05
50%	5,68	5,68	**2,47**	3,26	3,82	3,68	3,39
NMI							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	4,24	3,71	3,84	4,32	3,95	4,00	3,95
5%	4,03	3,61	3,47	4,34	4,24	4,00	4,32
10%	4,66	4,24	4,05	3,63	4,13	3,68	3,61
20%	4,79	5,13	3,50	3,68	3,68	3,58	3,63
30%	5,89	5,66	3,61	3,26	3,21	**2,87**	3,50
40%	6,42	5,53	3,29	**3,00**	3,39	**2,82**	3,55
50%	6,45	5,66	**2,74**	3,29	3,47	2,97	3,42

O *ranking* de Friedman para os valores de alfa do algoritmo BSSC-GP são apresentados na Tabela 25. Nesse algoritmo, com 1% e 5% de dados rotulados os melhores resultados foram obtidos por $\alpha = 0,1$. Com 1%, 10% e 20% de dado rotulados não há diferença estatística significativa entre os valores de alfa. Para 30%, 40% e 50% de dados rotulados, o melhor valor de alfa foi $\alpha = 1$.

Para o algoritmo BSSC-GS, o *ranking* de Friedman para diferentes valores de alfa é apresentado na Tabela 26 em que percebe-se que para 5% de dados rotulados os melhores desempenhos foram alcançados por $\alpha = 0,1$. Para configurações de 30% a 50% de dados rotulados, os melhores resultados foram obtidos por $\alpha = 1$. Para 1%, 10% e 20% de dados rotulados, não houve diferença estatística significativa para diferentes valores de alfa.

O *ranking* de Friedman para diferentes valores de alfa para o algoritmo BSSC-LP é apresentado na Tabela 27, em que percebe-se que para 1% de dados rotulados o melhor desempenho foi alcançado por $\alpha = 0,1$. Para 40% e 50% de dados rotulados, o melhor

Tabela 25 – *Ranking* de Friedman para diferentes valores do parâmetro alfa do algoritmo BSSC-GP

Acurácia							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	3,32	4,13	3,55	4,37	4,05	4,58	4,00
5%	**3,05**	3,53	3,58	4,53	5,16	3,82	4,34
10%	3,21	3,53	3,32	4,26	4,82	4,21	4,66
20%	4,16	3,45	3,32	4,34	4,50	3,87	4,37
30%	4,55	3,13	**2,79**	4,18	4,68	3,53	5,13
40%	4,74	3,39	**2,26**	4,21	4,58	4,05	4,76
50%	4,74	3,26	**2,47**	4,05	4,79	4,34	4,34
NMI							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	3,18	4,08	3,55	4,58	4,24	4,13	4,24
5%	**2,97**	3,55	3,53	4,53	4,61	4,13	4,68
10%	3,55	3,76	3,74	4,11	4,55	4,21	4,08
20%	4,21	3,95	3,26	4,13	4,24	4,05	4,16
30%	4,97	3,68	**3,08**	4,21	4,16	3,45	4,45
40%	5,16	3,76	**2,89**	3,66	4,39	3,63	4,50
50%	5,45	3,47	**2,61**	3,95	4,34	3,84	4,34

Tabela 26 – *Ranking* de Friedman para diferentes valores do parâmetro alfa do algoritmo BSSC-GS

Acurácia							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	3,18	3,97	3,50	4,42	4,11	4,68	4,13
5%	**3,00**	3,89	3,50	4,47	5,11	3,74	4,29
10%	3,11	3,82	3,21	4,21	4,79	4,18	4,68
20%	3,89	3,55	3,47	4,37	4,53	3,84	4,34
30%	3,97	3,50	**2,89**	4,24	4,74	3,50	5,16
40%	4,63	3,74	**2,53**	4,16	4,45	3,84	4,66
50%	5,84	3,32	**2,32**	3,84	4,53	4,08	4,08
NMI							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	3,03	4,21	3,61	4,53	4,21	4,21	4,21
5%	**2,95**	3,37	3,55	4,58	4,66	4,16	4,74
10%	3,47	3,92	3,74	4,08	4,53	4,21	4,05
20%	3,76	4,13	3,45	4,18	4,24	4,08	4,16
30%	5,05	3,84	3,26	4,13	4,05	3,32	4,34
40%	5,84	3,68	**2,84**	3,53	4,24	3,47	4,39
50%	6,32	3,53	**2,47**	3,82	4,11	3,61	4,16

resultado foi obtido por $\alpha = 1$. De 5% até 30% de dados rotulados, não houve diferença estatística significativa para diferentes valores de alfa.

Tabela 27 – *Ranking* de Friedman para diferentes valores do parâmetro alfa do algoritmo BSSC-LP

Acurácia							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	**2,95**	3,42	3,74	4,79	4,13	4,87	4,11
5%	3,13	3,39	3,97	4,45	4,61	4,11	4,34
10%	3,00	3,92	4,39	3,89	4,34	4,39	4,05
20%	3,95	4,53	3,79	3,74	3,92	4,03	4,05
30%	4,24	4,24	3,53	3,97	3,84	3,55	4,63
40%	5,50	4,74	**2,89**	3,61	4,03	**3,11**	4,13
50%	5,29	4,55	**3,26**	3,82	3,87	3,66	3,55
NMI							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	**2,95**	3,84	3,66	4,55	4,34	4,32	4,34
5%	3,26	3,61	3,79	4,42	4,39	3,97	4,55
10%	3,34	4,11	4,03	4,05	4,37	4,13	3,97
20%	4,08	4,71	3,97	3,66	3,82	3,87	3,89
30%	4,97	4,34	3,55	3,79	3,84	3,37	4,13
40%	5,45	4,39	3,53	**3,24**	3,92	**3,45**	4,03
50%	5,61	4,11	**3,37**	3,61	3,76	**3,58**	3,97

O *ranking* de Friedman para diferentes valores de alfa para o algoritmo BSSC-LS é apresentado na Tabela 28 em que de 1% até 10% de dados rotulados, os melhores desempenhos foram alcançados por $\alpha = 0,1$. De 30% a 50% de dados rotulados, os melhores resultados foram obtidos por $\alpha = 1$. Apenas para 20% de dados rotulados, não houve diferença estatística significativa para diferentes valores de alfa.

Tabela 28 – *Ranking* de Friedman para diferentes valores do parâmetro alfa do algoritmo BSSC-LS

Acurácia							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	**2,89**	3,37	3,84	4,74	4,26	4,68	4,21
5%	**2,82**	3,61	4,05	4,63	4,87	3,95	4,08
10%	3,21	4,13	4,37	3,92	4,32	4,00	4,05
20%	3,74	5,08	3,47	4,03	4,05	3,68	3,95
30%	3,84	4,66	**2,74**	4,24	4,34	3,24	4,95
40%	4,84	4,63	**2,47**	3,76	4,45	3,42	4,42
50%	5,58	4,24	**2,05**	3,71	4,34	3,87	4,21
NMI							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	**2,37**	3,76	3,76	4,61	4,39	4,71	4,39
5%	**2,26**	3,58	3,50	4,89	4,74	4,21	4,82
10%	**2,39**	3,95	4,21	4,24	4,53	4,50	4,18
20%	3,63	4,89	3,71	4,00	3,95	3,95	3,87
30%	5,37	4,66	**3,29**	3,82	3,71	**3,16**	4,00
40%	6,16	4,29	**2,76**	3,29	4,05	3,26	4,18
50%	6,34	4,24	**2,42**	3,76	3,84	3,45	3,95

O próximo algoritmo analisado é o ESSC que apresenta os valores do *ranking* de Friedman para diferentes valores de alfa na Tabela 29. Para 5% e 30% de dados rotulados,

o melhor desempenho foi obtido por ($\alpha = 1$). Para 10%, 20% e 40% de dados rotulados, o melhor desempenho foi obtido por ($\alpha = 0,5$). Com 50% de dados rotulados o melhor resultado foi obtido por ($\alpha = 0,1$).

Tabela 29 – *Ranking* de Friedman para diferentes valores do parâmetro alfa do algoritmo ESSC

Acurácia							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	3,55	4,29	3,42	4,11	4,53	3,39	4,71
5%	3,13	**2,84**	**2,66**	4,32	5,74	3,63	5,68
10%	**2,82**	**3,05**	3,29	3,45	5,53	3,82	6,05
20%	3,47	3,13	**2,71**	4,00	6,21	3,63	4,84
30%	3,29	3,61	**3,00**	4,37	5,16	3,29	5,29
40%	4,00	**2,58**	3,76	4,24	5,34	3,11	4,97
50%	**3,34**	3,82	3,61	4,11	5,34	**3,37**	4,42
NMI							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	3,71	4,45	3,76	3,87	3,92	3,82	4,47
5%	3,37	3,34	**2,89**	3,84	5,42	3,50	5,63
10%	3,11	**3,05**	**3,08**	3,82	5,63	3,45	5,87
20%	**3,05**	**3,26**	**3,29**	4,16	5,68	3,47	5,08
30%	**2,97**	3,45	**3,13**	4,34	5,58	3,45	5,08
40%	3,76	**2,89**	3,29	4,42	5,50	3,26	4,87
50%	3,24	3,37	3,45	4,03	5,55	**3,13**	5,24

Na Tabela 30 são apresentados os valores do *ranking* de Friedman para diferentes valores de alfa para o algoritmo ESSC-GP. O índice ACC apresentou diferença estatística significativa para 1%, 30% e 40% de dados rotulados com ($\alpha = 0,1$), ($\alpha = 0,1$) e ($\alpha = 0,5$) respectivamente. Para o índice NMI, apenas 30% de dados rotulados obteve um desempenho melhor com ($\alpha = 1$).

Tabela 30 – *Ranking* de Friedman para diferentes valores do parâmetro alfa do algoritmo ESSC-GP

Acurácia							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	4,29	4,03	3,76	4,26	4,58	3,61	3,47
5%	**3,24**	3,58	3,61	3,68	5,37	4,26	4,26
10%	3,24	4,00	3,47	3,95	4,76	3,92	4,66
20%	3,42	3,53	4,16	3,76	5,05	3,97	4,11
30%	**3,39**	**3,55**	**3,55**	4,32	5,45	3,76	3,97
40%	4,47	**2,95**	3,47	3,47	4,97	4,00	4,66
50%	3,50	3,61	3,97	3,92	4,61	3,97	4,42
NMI							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	4,47	4,37	3,58	3,55	4,55	3,66	3,82
5%	3,76	3,82	3,58	3,74	4,63	4,16	4,32
10%	3,47	3,92	3,24	4,11	4,79	3,84	4,63
20%	4,03	3,45	3,97	4,03	4,87	3,89	3,76
30%	3,74	3,61	**3,29**	4,16	5,37	3,71	4,13
40%	3,97	3,26	3,74	3,68	4,84	3,87	4,63
50%	3,89	3,26	3,89	4,11	4,61	3,95	4,29

Na Tabela 31 são apresentados os valores do *ranking* de Friedman para diferentes valores de alfa para o algoritmo ESSC-LP. Apenas com 1%, 30% e 40% de dados rotulados não houve diferença estatística para o índice ACC. Para o índice NMI, todas as configurações de alfa foram equivalentes estatisticamente no Teste de Friedman. Para o índice ACC, com 5% e 10% de dados rotulados, o melhor valor de alfa foi ($\alpha = 1$). Com 20% e 50% de dados rotulados, ($\alpha = 0,1$) obteve os melhores resultados em termos de desempenho.

Tabela 31 – *Ranking* de Friedman para diferentes valores do parâmetro alfa do algoritmo ESSC-LP

Acurácia							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	3,71	3,89	4,37	3,89	4,32	3,71	4,11
5%	3,42	3,89	**3,21**	3,76	5,24	4,21	4,26
10%	3,47	4,47	**2,76**	3,87	5,16	3,74	4,53
20%	**3,24**	3,71	3,68	3,79	5,26	3,87	4,45
30%	3,39	3,34	4,18	4,29	4,97	3,61	4,21
40%	4,11	3,37	3,39	3,66	5,00	3,84	4,63
50%	**3,42**	3,89	**3,45**	**3,37**	5,58	3,87	4,42

NMI							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	4,34	4,11	4,11	3,61	3,74	4,05	4,05
5%	4,00	3,87	3,47	4,24	3,92	4,50	4,00
10%	3,92	4,08	3,58	4,45	4,13	3,39	4,45
20%	3,95	3,87	4,11	3,87	4,34	3,71	4,16
30%	3,32	4,16	4,39	4,32	4,21	3,61	4,00
40%	3,84	3,50	3,55	4,29	4,47	4,13	4,21
50%	3,39	3,63	4,24	3,68	4,79	4,11	4,16

Na Tabela 32 é apresentado o *ranking* de Friedman para diferentes valores de alfa para o algoritmo ESSC-GS. Para esse algoritmo, apenas com 40% de dados rotulados foi obtido um desempenho estatisticamente significativo por ($\alpha = 0,5$) para o índice ACC. O algoritmo ESSC-LS não obteve nenhuma configuração com uma diferença estatística significativa em relação aos diferentes valores de alfa, portanto não mostramos aqui a tabela contendo o *ranking* de Friedman para esse algoritmo.

De agora em diante, são analisados os algoritmo NebFuzz e suas versões com ponderação automática de variáveis. O *ranking* de Friedman para diferentes valores de alfa do algoritmo NebFuzz são apresentados na Tabela 33. De acordo com o Teste de Friedman, apenas para 30% e 40% de dados rotulados houve uma diferença estatística significativa em termos de desempenho para os diferentes valores de alfa. Com 30% de dados rotulados, ($\alpha = 5$) obteve o melhor desempenho para as bases de dados reais. Para 40% de dados rotulados, o melhor desempenho foi obtido por ($\alpha = 1$).

O *ranking* de Friedman para diferentes valores de alfa do algoritmo NebFuzz-GP são apresentados na Tabela 34 em que apenas para 30% e 50% de dados rotulados, para os valores de alfa ($\alpha = 0,5$) e ($\alpha = 1$) respectivamente, obteve diferença estatística

Tabela 32 – *Ranking* de Friedman para diferentes valores do parâmetro alfa do algoritmo ESSC-GS

Acurácia							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	4,84	3,87	3,68	3,37	4,53	3,63	4,08
5%	4,13	3,87	3,74	4,03	4,71	3,61	3,92
10%	3,92	4,11	4,08	4,13	4,24	3,53	4,00
20%	3,45	3,68	4,00	4,00	4,89	3,87	4,11
30%	3,95	3,63	4,29	3,45	4,55	3,87	4,26
40%	4,63	**3,26**	3,95	3,50	4,03	4,29	4,34
50%	4,16	4,08	3,71	3,89	4,32	4,16	3,68

NMI							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	4,58	3,74	3,95	4,05	4,21	3,58	3,89
5%	3,68	3,61	3,53	4,37	4,37	4,24	4,21
10%	3,55	4,37	3,87	3,66	4,63	4,18	3,74
20%	3,42	4,11	3,29	4,32	4,58	4,03	4,26
30%	3,71	3,84	4,50	3,66	4,39	4,21	3,68
40%	4,61	2,61	3,55	3,55	4,03	4,66	5,00
50%	3,82	3,84	4,55	4,00	4,39	3,97	3,42

Tabela 33 – *Ranking* de Friedman para diferentes valores do parâmetro alfa do algoritmo NebFuzz

Acurácia							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	3,68	3,61	4,21	4,58	4,00	4,24	3,68
5%	4,24	3,89	4,26	3,47	3,97	3,89	4,26
10%	3,92	4,05	4,39	3,50	3,79	4,37	3,97
20%	4,63	4,39	4,11	3,29	3,89	3,71	3,97
30%	5,21	4,26	3,84	**3,05**	4,16	3,32	4,16
40%	4,21	3,95	3,16	3,95	4,71	3,50	4,53
50%	4,55	4,13	3,97	3,79	4,32	3,76	3,47

NMI							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	4,05	3,95	4,13	4,18	3,68	4,05	3,95
5%	4,21	4,16	4,08	3,89	3,92	4,00	3,74
10%	4,66	4,29	4,05	3,79	3,53	3,66	4,03
20%	5,34	4,21	3,58	**2,92**	4,26	3,32	4,37
30%	5,42	3,66	3,50	3,68	4,16	**3,16**	4,42
40%	4,45	3,50	**2,97**	4,11	5,00	3,11	4,87
50%	4,18	3,42	3,34	4,24	4,63	3,92	4,26

significativa em relação a outros valores de alfa para o índice NMI.

Tabela 34 – *Ranking* de Friedman para diferentes valores do parâmetro alfa do algoritmo NebFuzz-GP

Acurácia							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	3,50	4,16	3,97	4,11	4,11	4,32	3,84
5%	3,58	3,37	4,26	3,97	4,82	3,79	4,21
10%	3,71	3,24	4,00	4,11	4,58	4,18	4,18
20%	3,37	4,00	3,84	4,18	4,34	4,24	4,03
30%	4,45	3,55	3,53	4,03	4,34	3,89	4,21
40%	4,29	3,82	3,53	3,39	4,32	3,97	4,68
50%	4,05	3,53	3,32	3,89	4,63	4,11	4,47
NMI							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	4,00	4,11	3,92	4,13	4,00	3,95	3,89
5%	3,92	3,55	4,11	3,74	4,61	3,68	4,39
10%	4,18	3,47	3,61	3,63	4,39	4,05	4,66
20%	3,58	3,53	3,66	4,39	4,32	4,13	4,39
30%	4,39	3,16	3,47	4,08	4,42	3,82	4,66
40%	4,39	3,45	3,58	3,39	4,34	4,21	4,63
50%	4,29	3,53	3,32	3,61	4,92	4,08	4,26

O *ranking* de Friedman para diferentes valores de alfa do algoritmo NebFuzz-LP são apresentados na Tabela 35 em que apenas para configurações de 20% a 50% de dados rotulados houve diferença estatística de acordo com o Teste de Friedman. Para 20% de dados rotulados o melhor valor de alfa foi ($\alpha = 100$). Para 30%, 40% e 50% de dados rotulados, ($\alpha = 5$) obteve o melhor desempenho.

Tabela 35 – *Ranking* de Friedman para diferentes valores do parâmetro alfa do algoritmo NebFuzz-LP

Acurácia							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	3,32	3,84	3,37	4,37	4,16	4,63	4,32
5%	3,50	3,87	4,39	3,50	4,29	4,42	4,03
10%	4,66	4,74	4,53	3,71	3,37	3,66	3,34
20%	5,24	4,45	4,08	3,47	3,63	4,00	**3,13**
30%	4,74	4,24	3,89	3,55	3,79	4,05	3,74
40%	5,18	4,89	3,89	**3,08**	3,76	3,32	3,87
50%	4,55	4,32	4,34	3,53	3,79	3,92	3,55
NMI							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	3,55	4,08	3,50	4,13	4,29	4,03	4,42
5%	4,29	4,42	4,79	3,47	3,34	4,37	3,32
10%	4,84	4,47	4,08	3,42	4,03	3,61	3,55
20%	5,08	4,58	4,05	3,34	3,76	3,71	3,47
30%	5,18	4,47	3,92	**3,18**	3,82	3,55	3,87
40%	5,18	4,16	4,13	**3,39**	3,87	3,74	3,53
50%	4,97	4,39	4,11	**3,05**	3,95	3,68	3,84

O *ranking* de Friedman para diferentes valores de alfa do algoritmo NebFuzz-GS são apresentados na Tabela 36. Para este algoritmo, apenas para 5% de dados rotulados, houve diferença estatística significativa de acordo com o Teste de Friedman, obtida pelo

valor ($\alpha = 0,1$). Para o algoritmo NebFuzz-GP, os diferentes valores de alfa não tiveram diferença estatística significativa em nenhum experimento, portanto não apresentamos a tabela para esse algoritmo.

Tabela 36 – *Ranking* de Friedman para diferentes valores do parâmetro alfa do algoritmo NebFuzz-GS

Acurácia							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	3,68	3,76	3,50	4,34	4,47	4,08	4,16
5%	**2,63**	3,50	4,37	4,63	3,95	4,39	4,53
10%	3,61	4,32	4,05	4,11	4,00	3,82	4,11
20%	3,68	4,05	4,13	3,97	4,00	4,39	3,76
30%	4,13	3,92	3,97	3,95	4,39	4,37	3,26
40%	3,82	4,11	4,42	3,95	3,39	4,16	4,16
50%	4,29	4,32	3,74	3,16	3,84	3,97	4,68

NMI							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	3,79	3,87	3,68	3,87	4,45	4,00	4,34
5%	3,63	4,42	4,08	3,89	3,87	3,97	4,13
10%	4,47	4,18	3,53	4,29	3,87	3,84	3,82
20%	4,11	3,76	3,92	3,74	3,97	4,11	4,39
30%	3,95	3,79	3,66	4,34	3,89	4,26	4,11
40%	3,82	4,13	4,00	3,84	3,74	4,08	4,39
50%	3,92	3,92	3,53	3,76	4,34	4,11	4,42

O *ranking* de Friedman para o algoritmo NebFuzz-LS são apresentados na Tabela 37 em que apenas para 1%, 30% e 40% de dados rotulados houve diferença estatística significativa analisando a acurácia. Para o índice NMI, apenas para 30% de dados rotulados houve diferença estatística no desempenho obtido por ($\alpha = 10$). Na análise da acurácia, com 5%, 30% e 40% de dados rotulados, os melhores desempenhos foram obtidos pelos valores ($\alpha = 0,1$), ($\alpha = 100$) e ($\alpha = 50$) respectivamente.

Tabela 37 – *Ranking* de Friedman para diferentes valores do parâmetro alfa do algoritmo NebFuzz-LS

Acurácia							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	4,03	3,39	4,16	3,58	4,39	4,45	4,00
5%	**2,66**	3,74	4,05	4,32	4,03	4,50	4,71
10%	3,82	3,92	3,74	4,08	3,84	4,00	4,61
20%	4,34	4,45	4,29	3,84	2,79	4,63	3,66
30%	5,21	4,00	4,00	4,03	3,53	4,24	**3,00**
40%	5,16	4,13	4,18	3,79	**3,05**	3,71	3,97
50%	5,08	3,92	3,66	3,24	4,50	3,55	4,05

NMI							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	4,11	4,00	3,82	4,00	4,05	3,89	4,13
5%	4,55	4,13	4,03	3,66	4,11	3,61	3,92
10%	4,82	4,53	3,82	3,71	3,50	3,71	3,92
20%	4,97	4,08	4,11	3,50	3,66	3,71	3,97
30%	5,32	4,18	3,76	3,71	**3,61**	3,68	3,74
40%	4,84	3,97	3,92	3,63	3,79	3,82	4,03
50%	4,82	4,00	3,61	3,45	4,47	3,55	4,11

Os resultados do *ranking* de Friedman para os experimentos realizados com o algoritmo Isodata-PS e suas respectivas versões com ponderação automática de variáveis são analisados a seguir. O *ranking* de Friedman para diferentes valores de alfa do algoritmo Isodata-PS são apresentados na Tabela 38 em que com apenas 1% de dados rotulados, não existe uma diferença estatística significativa em termos de desempenho para os diferentes valores de alfa. Para 5% e 10% de dados rotulado, o melhor valor foi ($\alpha = 0,1$). Para configurações de 20% a 50% de dados rotulados, o valor de alfa que obteve os melhores resultados foram ($\alpha = 10$) e ($\alpha = 100$) para as duas medidas. Para esse trabalho, o alfa selecionado foi ($\alpha = 10$).

Tabela 38 – *Ranking* de Friedman para diferentes valores do parâmetro alfa do algoritmo Isodata-PS

Acurácia							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	3,76	3,24	4,45	4,05	3,92	4,58	4,00
5%	**1,79**	2,55	4,89	5,00	3,87	5,82	4,08
10%	**1,61**	3,29	6,34	4,82	3,26	4,84	3,84
20%	3,32	3,89	5,74	4,05	**3,03**	4,92	**3,05**
30%	4,63	4,82	5,53	3,50	**2,24**	4,58	**2,71**
40%	5,58	5,26	5,16	3,47	**2,24**	3,89	**2,39**
50%	5,97	5,39	4,92	3,13	**2,16**	4,26	**2,16**
NMI							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	4,03	3,16	4,34	4,24	3,97	4,45	3,82
5%	3,34	3,00	4,34	4,37	4,03	4,82	4,11
10%	**3,00**	3,71	5,13	4,87	3,55	3,97	3,76
20%	4,34	4,87	5,11	3,82	**2,89**	4,24	**2,74**
30%	5,76	4,76	5,53	3,29	**1,89**	4,47	**2,29**
40%	6,50	5,32	5,16	2,97	**2,11**	3,66	**2,29**
50%	6,47	5,37	5,00	2,92	**2,16**	3,74	**2,34**

O *ranking* de Friedman para diferentes valores de alfa para o algoritmo Isodata-PS-GP são apresentados na Tabela 39. Com apenas 1% de dados rotulados, não existe uma diferença estatística significativa em termos de desempenho para os diferentes valores de alfa. Com 5% e 10% de dados rotulados, o valor ($\alpha = 0,5$) teve o melhor desempenho. De 20% a 50% de dados rotulados o melhor desempenho foi alcançado por ($\alpha = 1$).

O *ranking* de Friedman para diferentes valores de alfa para o algoritmo Isodata-PS-LP são apresentados na Tabela 40. Com configurações menores que 20% de dados rotulados, não existe uma diferença estatística significativa em termos de desempenho para os diferentes valores de alfa. Com 30% e 40% de dados rotulado, o melhor desempenho foi obtido pelo valor ($\alpha = 1$). Com 50% de dados rotulados, o melhor desempenho foi obtido por ($\alpha = 50$) para o índice NMI.

O *ranking* de Friedman para diferentes valores de alfa para o algoritmo Isodata-PS-GS são apresentados na Tabela 41. Com 1%, 10% e 20% de dados rotulados, não existe uma diferença estatística significativa em termos de desempenho para os diferentes valores de

Tabela 39 – *Ranking* de Friedman para diferentes valores do parâmetro alfa do algoritmo Isodata-PS-GP

Acurácia							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	3,55	3,76	3,76	4,29	4,08	4,37	4,18
5%	3,21	**2,71**	3,26	4,84	5,32	4,03	4,63
10%	3,26	**2,95**	3,47	4,55	5,53	3,34	4,89
20%	3,97	2,84	**2,18**	4,53	5,79	3,47	5,21
30%	4,68	3,05	**1,68**	3,95	5,92	3,24	5,47
40%	5,21	3,08	**1,71**	3,82	5,92	2,87	5,39
50%	5,68	3,21	**2,00**	3,50	6,11	2,50	5,00
NMI							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	4,68	4,00	4,26	3,92	3,61	3,97	3,55
5%	3,21	3,37	3,84	4,39	4,82	4,13	4,24
10%	3,92	3,37	3,34	4,26	5,37	**3,16**	4,58
20%	4,68	3,08	**2,05**	3,89	5,71	3,61	4,97
30%	5,29	2,97	**1,71**	4,13	5,82	3,00	5,08
40%	5,74	3,29	**1,97**	3,74	5,50	2,68	5,08
50%	6,13	3,37	**1,87**	3,32	5,82	2,26	5,24

Tabela 40 – *Ranking* de Friedman para diferentes valores do parâmetro alfa do algoritmo Isodata-PS-LP

Acurácia							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	4,00	3,95	3,63	4,21	3,63	4,74	3,84
5%	3,87	3,16	3,29	3,95	4,89	4,42	4,42
10%	3,50	3,84	4,39	4,08	4,34	3,61	4,24
20%	3,74	4,34	3,34	3,76	4,68	3,74	4,39
30%	4,45	3,97	**2,47**	3,71	4,55	3,79	5,05
40%	5,61	4,68	**2,11**	3,34	4,71	**2,68**	4,87
50%	5,39	4,24	**2,89**	3,53	4,89	**2,92**	4,13
NMI							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	3,92	4,05	4,05	3,87	3,87	4,45	3,79
5%	3,97	3,63	3,74	4,08	4,37	4,24	3,97
10%	4,13	4,16	4,34	3,61	4,39	3,39	3,97
20%	4,29	5,05	3,21	3,58	4,08	3,76	4,03
30%	5,13	4,39	**2,66**	3,61	4,68	3,21	4,32
40%	5,58	4,42	**2,58**	3,29	4,58	**2,74**	4,82
50%	5,71	4,29	**2,92**	3,18	4,82	**2,47**	4,61

alfa. Com 5% de dados rotulados, o valor de alfa ($\alpha = 0,1$), obteve o melhor desempenho apenas para a acurácia. Para 30% e 40% de dados rotulados, o melhor desempenho obtido foi por ($\alpha = 1$). Para 50% de dados rotulados, o melhor desempenho foi alcançado por ($\alpha = 50$).

Tabela 41 – *Ranking* de Friedman para diferentes valores do parâmetro alfa do algoritmo Isodata-PS-GS

Acurácia							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	4,05	3,68	4,24	4,03	3,79	4,08	4,13
5%	**2,89**	3,24	3,84	4,26	5,29	4,00	4,47
10%	3,00	3,47	4,53	4,34	4,42	3,82	4,42
20%	3,50	3,79	3,29	4,39	4,87	4,24	3,92
30%	4,61	3,92	**2,63**	3,55	4,87	3,87	4,55
40%	4,92	4,29	**2,68**	3,92	4,66	**2,74**	4,79
50%	5,92	4,50	**2,63**	3,68	4,74	**2,16**	4,37
NMI							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	3,61	3,66	3,61	4,37	4,45	3,95	4,37
5%	3,34	3,63	3,97	4,18	4,45	4,03	4,39
10%	4,21	4,13	3,68	4,26	4,32	3,37	4,03
20%	5,11	4,26	3,42	3,61	4,53	3,47	3,61
30%	5,53	4,37	**2,66**	3,63	4,29	3,26	4,26
40%	5,76	4,47	**2,84**	3,58	4,42	**2,66**	4,26
50%	6,08	4,76	**2,84**	3,55	4,34	**2,13**	4,29

Na Tabela 42, o *ranking* de Friedman para diferentes valores de alfa para o algoritmo Isodata-PS-LS são apresentados. Nesse experimento, com 1% 5% e 10% de dados rotulados, não existe diferença estatística significativa em termos de desempenho para os diferentes valores de alfa. De 20% até 40% de dados rotulados, o melhor desempenho foi obtido por ($\alpha = 1$). Com 50% de dados rotulados, ($\alpha = 50$) teve o melhor desempenho.

Tabela 42 – *Ranking* de Friedman para diferentes valores do parâmetro alfa do algoritmo Isodata-PS-LS

Acurácia							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	3,82	3,58	4,34	3,89	4,03	4,03	4,32
5%	2,76	3,16	3,55	4,39	4,92	4,47	4,74
10%	3,16	3,32	4,08	4,76	4,47	3,95	4,26
20%	3,42	4,00	3,37	4,34	4,53	4,26	4,08
30%	4,42	4,37	3,26	3,82	4,45	4,03	3,66
40%	5,05	4,95	**2,50**	4,08	3,95	3,11	4,37
50%	5,79	5,11	**2,89**	3,18	4,29	**2,74**	4,00
NMI							
Rótulos	$\alpha = 0,1$	$\alpha = 0,5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
1%	3,84	3,66	4,26	3,71	4,34	3,92	4,26
5%	3,74	4,24	3,55	4,03	4,24	3,92	4,29
10%	4,76	5,00	3,63	3,58	3,66	3,53	3,84
20%	5,16	5,26	**3,21**	3,63	3,92	3,58	3,24
30%	5,87	5,11	**3,05**	3,05	3,95	3,47	3,50
40%	6,24	5,00	**2,61**	3,76	3,68	2,71	4,00
50%	6,32	5,03	**2,76**	3,00	4,18	**2,76**	3,95

6.1.1.1 Discussão

O objetivo desse experimento foi avaliar o efeito do parâmetro alfa em cada algoritmo considerado nesta tese, com exceção do algoritmo sSSC e suas variações com ponderação automática de variáveis, que não possuem esse parâmetro. Nesse experimento, foram avaliados alguns valores de alfa (α: 0,1; 0,5; 1; 5; 10; 50 e 100), executados em 19 bases de dados de problemas reais, levando em conta a quantidade de dados rotulados disponíveis para o treinamento que variou de (1%,5%,10%,20%,30%,40% e 50%) do total da base de dados. Os valores de alfa selecionados nesse experimento são apresentados na Tabela 43. Nessa Tabela, os valores de alfa apresentados com a sigla NS, entre parênteses, não foram escolhidos por uma diferença estatística significativa no Teste de Friedman. Eles foram escolhidos de acordo com o menor valor do *ranking* de Friedman. Os valores apresentados diretamente, sem a sigla NS na Tabela 43, foram escolhidos de acordo o pós teste de Nemenyi, pois eram melhores que pelo menos um ou mais valores de alfa, em termos de desempenho para a acurácia ou o índice NMI.

Ao fazer uma análise do algoritmo BSSC, verificamos que existe grande variação do valor de alfa, que muda de acordo com a quantidade de dados rotulados, variando de ($\alpha = 0,1$) até ($\alpha = 50$). Ao fazer a análise das versões do algoritmo BSSC com ponderação automática de variáveis, podemos perceber que para poucos dados rotulados, 10% ou menos, o melhor valor de alfa foi ($\alpha = 0,1$). Para mais que isso, o melhor valor de alfa encontrado foi ($\alpha = 1$), com exceção do algoritmo BSSC-LP com 20% de dados rotulados em que o melhor valor de alfa foi ($\alpha = 5$). Esse experimento mostra que para poucos dados rotulados, o termo supervisionado não possui um peso relevante nesses algoritmos, porém de 20% a 50% de dados rotulados, os rótulos aumentam o desempenho dos algoritmos baseados no algoritmo BSSC.

A análise do algotimo ESSC e suas versões com ponderação automática de variáveis mostram que os valores de alfa variaram de ($\alpha = 0,1$) a no máximo ($\alpha = 5$). Apenas o algoritmo ESSC-GS, que para 1% e 50% de dados rotulados, obteve o melhor alfa, ($\alpha = 50$) e ($\alpha = 100$) respectivamente. Para 40% de dados rotulados, todos esses algoritmos tiveram ($\alpha = 0,5$) como melhor valor de alfa. Para 1% de dados rotulados, nenhum algoritmo dessa família obteve diferença estatística significativa. Os algoritmos ESSC, ESSC-GP e ESSC-LP apresentaram mais configurações com valores de alfa obtidos por diferença estatística significativa. Enquanto, o algoritmos ESSC-GS obteve diferença estatística significativa apenas para 40% de dados rotulados. O algoritmo ESSC-LS não teve nenhum valor de alfa obtido por diferença estatística significativa. Esse experimento mostrou que esses algoritmos conseguem melhores desempenho com valores baixos de alfa.

A análise do algoritmo NebFuzz e suas respectivas versões com ponderação automática de variáveis mostram que a maioria das configurações do experimento não obteve um valor de alfa com diferença estatística significativa. Especificamente, os al-

goritmos NebFuzz-GP, NebFuzz-GS e NebFuzz-LS tiveram apenas um valor de alfa com diferença estatística significativa. O valor de alfa mais comum para os algoritmos NebFuzz e NebFuzz-LP foi ($\alpha = 5$). Nesse experimento, percebemos que os valores de alfa variaram bastante, de ($\alpha = 0,1$) ao valor máximo ($\alpha = 100$), para os algoritmos da baseados no NebFuzz.

Os algoritmos Isodata-PS e suas versões com ponderação automática de variáveis tiveram valores de alfa variando de ($\alpha = 0,1$) até ($\alpha = 50$). Com 1% de dados rotulados, nenhum desses algoritmos obteve diferença estatística significativa. Os algoritmos Isodata-PS e Isodata-PS-GP tiveram valores de alfa estatisticamente relevantes para a maioria das configurações de dados rotulados. Para configurações de 20% a 50% de dados rotulados, o valor de alfa para o algoritmo Isodata-PS foi ($\alpha = 10$) e para o algoritmo Isodata-PS-GP foi ($\alpha = 1$). Mais uma característica dos algoritmos Isodata-PS e Isodata-PS-GP é que os valores de alfa foram aumentando de acordo com a quantidade de dados rotulados. Com relação aos algoritmos Isodata-GS, Isodata-LP e Isodata-LS, os valores de alfa estatisticamente relevantes foram obtidos para mais 20% de dados rotulados. Para esses três últimos algoritmos, o valor máximo de alfa foi ($\alpha = 50$), obtido para 40% e 50% de dados rotulados pelo algoritmo Isodata-PS-GS, e apenas para 50% de dados rotulados, pelos algoritmos Isodata-PS-LP e Isodata-PS-LS. Para o algoritmo Isodata-PS, o valor mais comum de alfa foi ($\alpha = 10$), que se manteve constante após 20% de dados rotulados. Para as versões com ponderação automática de variáveis baseados no Isodata-PS, o valor mais comum de alfa foi ($\alpha = 1$), de 20% até 40% de dados rotulados. Com exceção do algoritmo Isodata-PS-GP que se manteve com ($\alpha = 1$) para 50% de dados rotulados, os outros algoritmos da família Isodata-PS com ponderação automática de variáveis tiveram ($\alpha = 50$) para 50% de dados rotulados.

Como pode-se constatar, para esse experimento com bases de dados reais, não existe um valor de alfa padrão para um algoritmo de agrupamento semissupervisionado. O valor de alfa pode ser diretamente afetado caso a quantidade de dados rotulados seja diferente, bem como também depende do problema ou base de dados considerada. Além do mais, cada um dos algoritmos analisados nesse experimento possui uma característica própria para selecionar o melhor valor de alfa, que pode ser percebida pela análise desse experimento. O alfa escolhido nessa análise é utilizado nos próximos experimentos com bases de dados reais.

Tabela 43 – Valores de alfa selecionados para bases de dados reais

	Alfa						
Algoritmos	1	5	10	20	30	40	50
BSSC	NS(0,5)	NS(0,5)	NS(0,5)	NS(1)	50	1	1
BSSC-GP	NS(0,1)	0,1	NS(0,1)	NS(1)	1	1	1
BSSC-LP	0,1	NS(0,1)	NS(0,1)	NS(5)	NS(1)	1	1
BSSC-GS	NS(0,1)	0,1	NS(0,1)	NS(1)	1	1	1
BSSC-LS	0,1	0,1	0,1	NS(1)	1	1	1
ESSC	NS(1)	1	0,5	1	1	0,5	0,1
ESSC-GP	NS(1)	0,1	NS(0,1)	NS(0,5)	1	0,5	NS(0,5)
ESSC-LP	NS(5)	1	1	0,1	NS(0,1)	NS(0,5)	0,1
ESSC-GS	NS(50)	NS(1)	NS(0,1)	NS(0,1)	NS(5)	0,5	NS(100)
ESSC-LS	NS(5)	NS(0,1)	NS(0,1)	NS(50)	NS(0,5)	NS(0,5)	NS(0,1)
NebFuzz	NS(0,5)	NS(5)	NS(5)	5	50	1	1
NebFuzz-GP	NS(0,1)	NS(0,5)	NS(0,5)	NS(0,1)	NS(0,5)	NS(5)	1
NebFuzz-LP	NS(0,1)	NS(5)	NS(100)	100	5	5	5
NebFuzz-GS	NS(1)	0,1	NS(50)	NS(5)	NS(0,5)	NS(10)	NS(5)
NebFuzz-LS	NS(0,5)	0,1	NS(10)	NS(10)	NS(100)	NS(10)	NS(5)
Isodata-PS	NS(0,5)	0,1	0,1	10	10	10	10
Isodata-PS-GP	NS(0,5)	0,5	0,5	1	1	1	1
Isodata-PS-LP	NS(10)	NS(0,5)	NS(50)	NS(1)	1	1	50
Isodata-PS-GS	NS(0,5)	0,1	NS(50)	1	1	50	50
Isodata-PS-LS	NS(0,5)	NS(0,1)	NS(0,1)	1	1	1	50

6.1.2 Comparação dos algoritmos com ponderação automática de variáveis

Este experimento compara os algoritmos de agrupamento semissupervisionado com ponderação automática de variáveis baseados no mesmo algoritmo de agrupamento semissupervisionado sem distância adaptativa com o objetivo de avaliar se a ponderação automática de variáveis melhora o desempenho desses algoritmos. São apresentadas duas figuras contendo o *ranking* de Friedman com 95% de confiança calculado a partir da acurácia e do índice NMI e mais duas figuras contendo a média do desempenho da acurácia e do índice NMI para avaliar o desempenho dos algoritmos de cada família na execução em 19 bases de dados reais. Pra complementar a avaliação, é realizado o teste estatístico de Friedman. Caso os algoritmos não tenham desempenho equivalentes, foi realizado o pós-teste de Holm, que compara um algoritmo controle, contra todos os outros. Nas tabelas que apresentam o resultado do pós-teste de Holm, o símbolo (=) representa que os dois algoritmos são equivalentes, o símbolo (<) representa que o algoritmo da primeira coluna é pior que o algoritmo da segunda coluna e o símbolo (>) representa que o algoritmo da primeira coluna é melhor que o algoritmo da segunda coluna. Para resultados mais detalhados, verifique as tabelas no Apêndice A que apresentam a média do desempenho de 100 execuções dos algoritmos com diferentes exemplos no conjunto de dados rotulados que variou de (1%,5%,10%,20%,30%,40% e 50%), do total da base de dados. Nas tabelas do Apêndice A, a posição do desempenho de cada algoritmo, em cada base de dados é colocado entre parênteses.

Nas próximas subseções são apresentadas as análises separadamente para cada família de algoritmo na seguinte ordem: BSSC, sSSC, ESSC, NebFuzz e Isodata-PS e

suas respectivas versões com ponderação automática de variáveis.

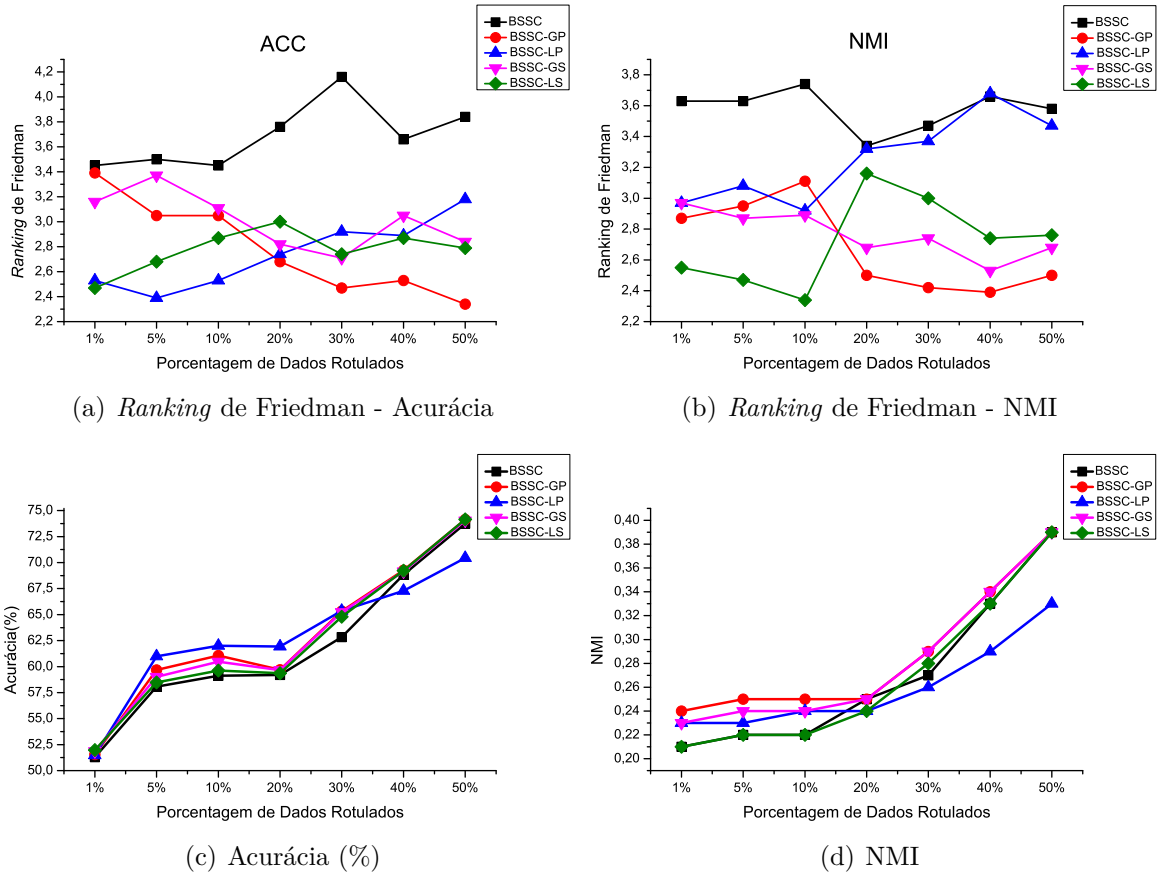
6.1.2.1 BSSC com distâncias adaptativas

Os resultados do desempenho dos algoritmos BSSC, BSSC-PG, BSSC-PL, BSSC-GS e BSSC-LS são apresentados na Figura 8. Através de gráficos apresentando o *ranking* de Friedman calculado a partir da acurácia e do índice NMI, percebe-se que todos os algoritmos com ponderação automática de variáveis tiveram desempenho melhor que o algoritmo BSSC com distância Euclidiana. Até 10% de dados rotulados, a maior acurácia foi obtida pelo algoritmo BSSC-LP, enquanto pelo índice NMI, o melhor desempenho foi obtido pelo algoritmo BSSC-LS. De 20% até 50% de dados rotulados, o melhor desempenho foi obtido pelo algoritmo BSSC-GP para os dois índices de avaliação. Na análise do *ranking* de Friedman, nota-se que para 1% de dados rotulados o desempenho dos algoritmos é semelhante. De 5% até 20%, o melhor desempenho foi alcançado pelo algoritmo BSSC-LP. Para 30%, 40% e 50% de dados rotulados, o desempenho foi semelhante para todos os algoritmos analisados. Analisando o índice NMI, nota-se que o melhor desempenho foi obtido pelo algoritmo BSSC-GP. Com configurações de 20% a 50% de dados rotulados, os algoritmos possuem desempenho semelhantes, com um pequeno destaque para o algoritmo BSSC-GS.

A análise estatística do teste de Friedman e o pós-teste de Holm dos algoritmos BSSC e versões com ponderação automática de variáveis é apresentada na Tabela 44. Analisando a acurácia, só existe diferença estatística entre os algoritmos a partir de 30% de dados rotulados. O algoritmo que obteve melhor desempenho foi o BSSC-GP. O algoritmo BSSC foi pior que outros algoritmos com ponderação automática para 30% e 50% de dados rotulados. O índice NMI apresenta que o algoritmo BSSC-LS foi estatisticamente melhor que algoritmo BSSC para 5% e 10% de dados rotulados. Para 40% de dados rotulados, os melhores desempenhos são obtidos pelos algoritmos BSSC-GP e BSSC-GS, nessa ordem. Os piores desempenhos são obtidos pelos algoritmos BSSC e BSSC-LP.

Tabela 44 – Teste de Friedman e pós-teste de Holm a um nível significância de 5% dos algoritmos da família BSSC.

		Porcentagem de Rótulos													
		1%		5%		10%		20%		30%		40%		50%	
		ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
BSSC	BSSC-GP	=	=	=	=	=	=	=	=	<	=	=	<	<	=
	BSSC-GS	=	=	=	=	=	=	=	=	<	=	=	=	=	=
	BSSC-LP	=	=	=	=	=	=	=	=	<	=	=	<	=	=
	BSSC-LS	=	=	=	<	=	<	=	=	<	=	=	=	=	=
BSSC-GP	BSSC-GS	=	=	=	=	=	=	=	=	=	=	=	=	=	=
	BSSC-LP	=	=	=	=	=	=	=	=	=	=	=	=	=	=
	BSSC-LS	=	=	=	=	=	=	=	=	=	=	=	=	=	=
BSSC-GS	BSSC-LP	=	=	=	=	=	=	=	=	=	=	=	=	=	=
	BSSC-LS	=	=	=	=	=	=	=	=	=	=	=	=	=	=
BSSC-LP	BSSC-LS	=	=	=	=	=	=	=	=	=	=	=	=	=	=



(a) *Ranking* de Friedman - Acurácia

(b) *Ranking* de Friedman - NMI

(c) Acurácia (%)

(d) NMI

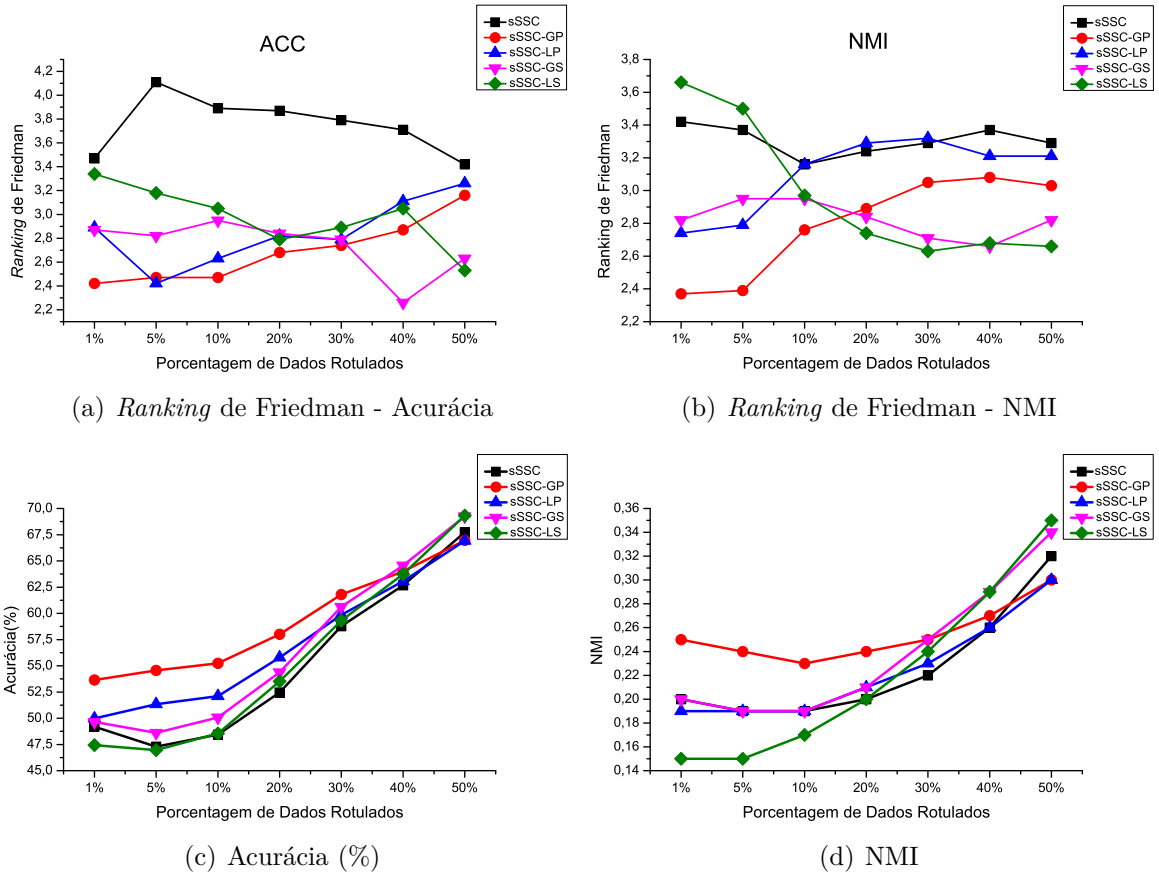
Figura 8 – *Ranking* de Friedman e desempenho dos algoritmos BSSC e suas respectivas versões com distância adaptativa.

6.1.2.2 sSSC com distância adaptativa

Os resultados do desempenho dos algoritmos sSSC, sSSC-PG, sSSC-PL, sSSC-GS e sSSC-LS são apresentados na Figura 9. São apresentados gráficos mostrando o *ranking* de Friedman do desempenho da acurácia e do índice NMI em bases de dados reais. Analisando o *ranking* de Friedman calculado a partir da acurácia nota-se que os algoritmos com distâncias adaptativas tiveram melhor desempenho que o algoritmo sSSC com distância Euclidiana. O melhor desempenho foi obtido pelo algoritmo sSSC-GP, seguido do algoritmo sSSC-LP. Com 30% de dados rotulados, os algoritmos com distância adaptativa possuem desempenho semelhantes. Com 40% de dados, o algoritmo sSSC-GS obteve o melhor desempenho. Com 50% de dados rotulados, o algoritmo sSSC-LS obteve o melhor desempenho. Esse mesmo comportamento se repete na análise do desempenho do desempenho da acurácia.

Analisando o *ranking* de Friedman calculado a partir do índice NMI, percebe-se que o melhor desempenho foi obtido pelo algoritmo sSSC-GP de 1% a 10% de dados rotulados. O algoritmo sSSC com distância euclidiana foi melhor que o algoritmo sSSC-LS com 1% e 5% de dados rotulados. De 20% a 50% de dados rotulados, o melhor desempenho foi obtido

pelo algoritmo sSSC-LS, seguido do algoritmo sSSC-GS. Esse mesmo comportamento se repete na análise da média do desempenho do índice NMI. Note que, tanto em relação à acurácia e ao índice NMI, o desempenho de todos os algoritmos aumentam com a quantidade de dados rotulados.



(a) *Ranking* de Friedman - Acurácia

(b) *Ranking* de Friedman - NMI

(c) Acurácia (%)

(d) NMI

Figura 9 – *Ranking* de Friedman e desempenho dos algoritmos sSSC e suas respectivas versões com distância adaptativa.

A análise estatística do teste de Friedman e o pós-teste de Holm dos algoritmos sSSC e versões com ponderação automática de variáveis é apresentada na Tabela 45. Analisando a acurácia, o algoritmo sSSC foi pior que todos os outros algoritmos com 5% de dados rotulados. Com 10%, 20% e 40% de dados rotulados o algoritmo sSSC-GP obteve o melhor desempenho. Analisando o índice NMI, o algoritmo sSSC-GP foi melhor que o algoritmo sSSC-LS. No restante das configurações, não há diferença estatística entre os algoritmos.

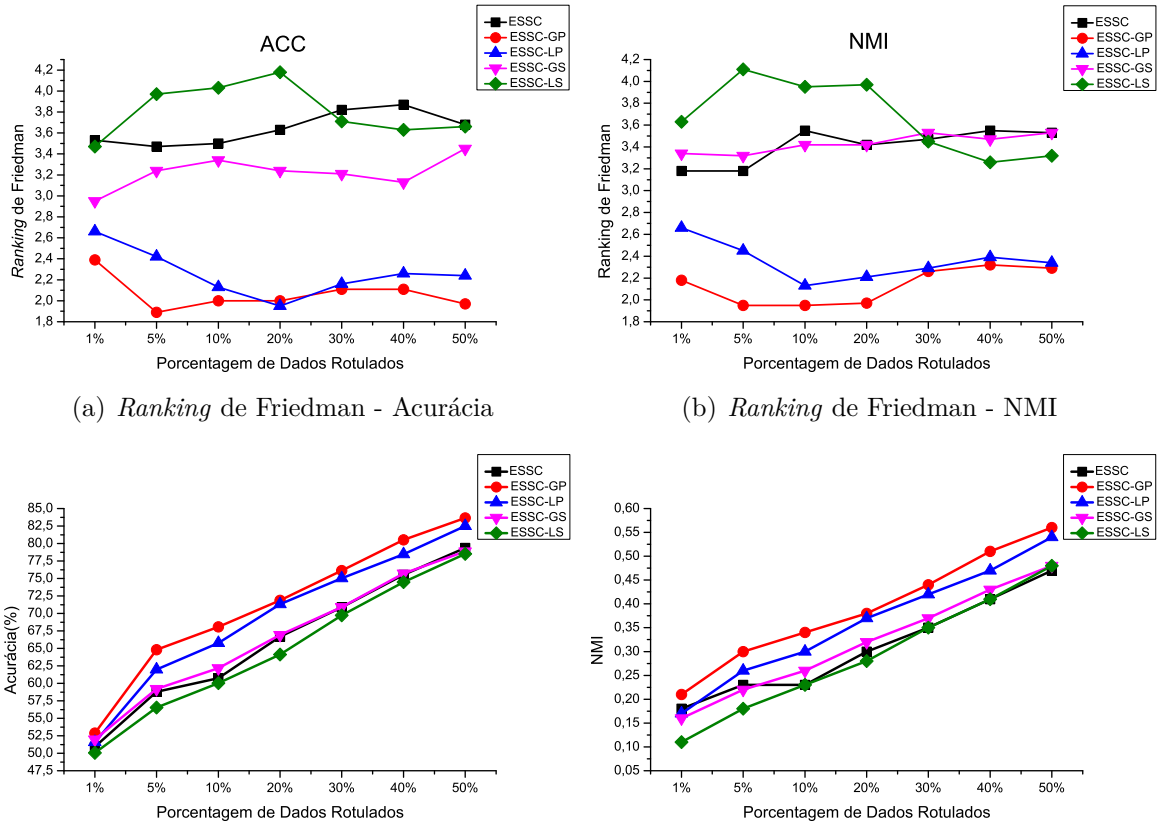
6.1.2.3 ESSC com distância adaptativa

Os resultados do desempenho dos algoritmos ESSC, ESSC-PG, ESSC-PL, ESSC-GS e ESSC-LS são apresentados na Figura 10. São apresentados gráficos mostrando o *ranking* de Friedman do desempenho da acurácia e do índice NMI em bases de dados reais. Analisando o *ranking* de Friedman calculado a partir da acurácia e do índice NMI,

Tabela 45 – Comparação entre os algoritmos sSSC e suas respectivas versões com ponderação automática de variáveis utilizando o teste de Friedman e o pós-teste de Holm a um nível significância de 5%.

		Porcentagem de Rótulos													
		1%		5%		10%		20%		30%		40%		50%	
		ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
sSSC	sSSC-GP	=	=	<	=	<	=	<	=	=	=	=	=	=	=
	sSSC-GS	=	=	<	=	<	=	=	=	=	=	=	=	=	=
	sSSC-LP	=	=	<	=	=	=	=	=	=	=	<	=	=	=
	sSSC-LS	=	=	<	=	=	=	=	=	=	=	=	=	=	=
sSSC-GP	sSSC-GS	=	=	=	=	=	=	=	=	=	=	=	=	=	=
	sSSC-LP	=	=	=	=	=	=	=	=	=	=	=	=	=	=
	sSSC-LS	=	>	=	=	=	=	=	=	=	=	=	=	=	=
sSSC-GS	sSSC-LP	=	=	=	=	=	=	=	=	=	=	=	=	=	=
	sSSC-LS	=	=	=	=	=	=	=	=	=	=	=	=	=	=
sSSC-LP	sSSC-LS	=	=	=	=	=	=	=	=	=	=	=	=	=	=

percebe-se claramente que dois algoritmos se destacam em termos de desempenho, o algoritmo ESSC-GP e o algoritmo ESSC-LP. Os algoritmos ESSC, ESSC-GS e ESSC-LS se alternam como terceiro melhor desempenho, com leve destaque ao analisar a acurácia para o algoritmo ESSC-GS. Esse comportamente também é visto ao analisar o desempenho da média da acurácia e do índice NMI. Note que todos os algoritmos dessa família possuem um aumento linear de desempenho para os dois índices na medida que se aumenta a quantidade de dados rotulados.



(a) *Ranking* de Friedman - Acurácia

(b) *Ranking* de Friedman - NMI

(c) Acurácia (%)

(d) NMI

Figura 10 – Ranking de Friedman e desempenho dos algoritmos ESSC e suas respectivas versões com distância adaptativa.

A análise estatística do teste de Friedman e o pós-teste de Holm dos algoritmos

ESSC e versões com ponderação automática de variáveis é apresentada na Tabela 46. A análise desse teste apresenta claramente que os algoritmos ESSC-GP e ESSC-LP, tiveram os melhores desempenhos em relação aos outros algoritmos nesse experimento. Esse resultado é observado principalmente de 5% até 50% de dados rotulados para a acurácia e o índice NMI.

Tabela 46 – Teste de Friedman e pós-teste de Holm a um nível significância de 5% dos algoritmos da família ESSC.

		Porcentagem de Rótulos													
		1%		5%		10%		20%		30%		40%		50%	
		ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
ESSC	ESSC-GP	=	=	<	<	<	=	<	<	<	<	<	<	<	<
	ESSC-GS	=	=	=	=	<	<	<	<	<	<	<	<	<	<
	ESSC-LP	=	=	=	=	=	=	=	=	=	=	=	=	=	=
	ESSC-LS	=	=	=	=	=	=	=	=	=	=	=	=	=	=
ESSC-GP	ESSC-GS	=	=	=	=	=	=	=	=	=	=	=	=	=	=
	ESSC-LP	=	>	>	>	>	>	>	>	=	>	=	>	>	>
	ESSC-LS	=	>	>	>	>	>	>	>	>	>	>	=	>	=
ESSC-GS	ESSC-LP	=	=	=	=	>	>	>	>	=	>	=	=	>	>
	ESSC-LS	=	=	>	>	>	>	>	>	>	>	>	=	>	=
ESSC-LP	ESSC-LS	=	=	=	=	=	=	=	=	=	=	=	=	=	=

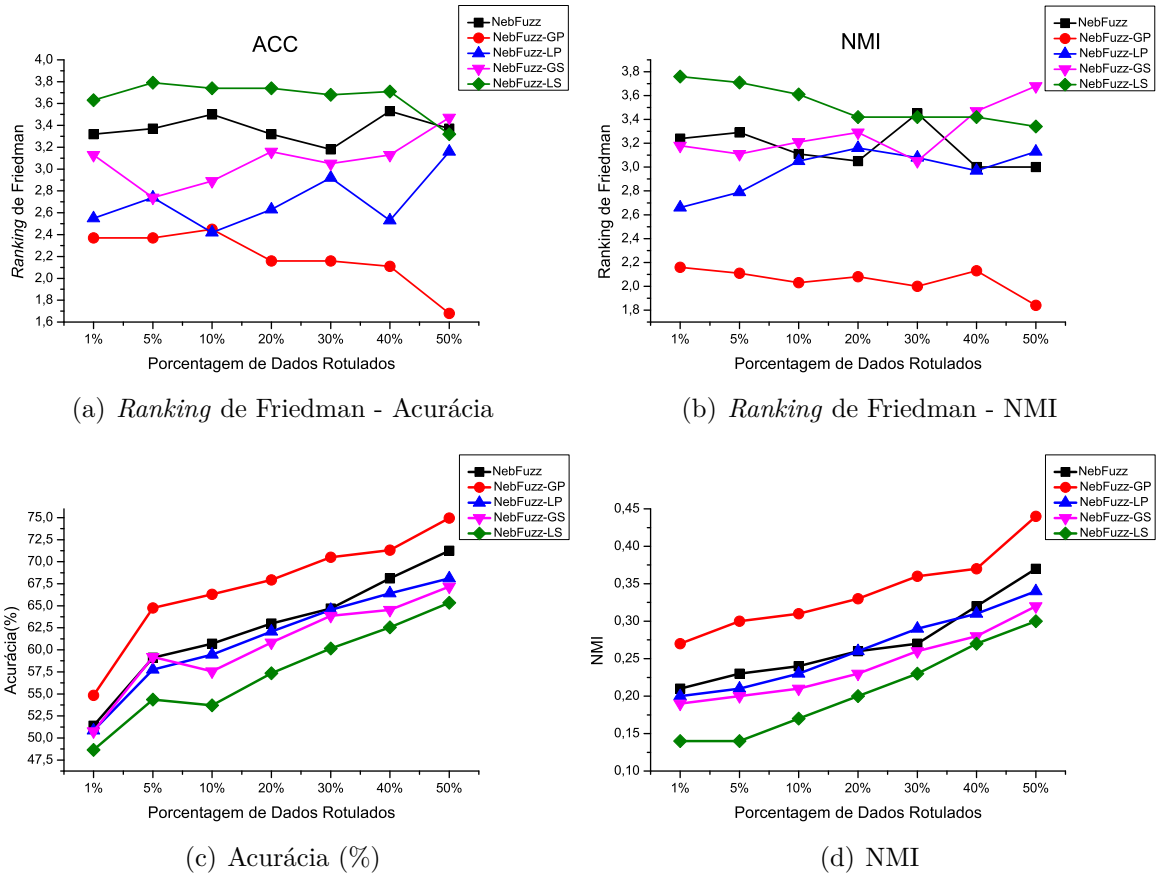
6.1.2.4 NebFuzz com distância adaptativa

Os resultados do desempenho dos algoritmos NebFuzz, NebFuzz-PG, NebFuzz-PL, NebFuzz-GS e NebFuzz-LS são apresentados na Figura 11. São apresentados gráficos mostrando o *ranking* de Friedman do desempenho calculados a partir da acurácia e do índice NMI em bases de dados reais. Analisando o *ranking* de Friedman calculado das duas medidas, acurácia e NMI, percebe-se que o algoritmo NebFuzz-GP obtém os melhores resultados. Ainda, os algoritmos NebFuzz-LP e NebFuzz-GS são melhores que o algoritmo NebFuzz. O pior desempenho foi obtido pelo algoritmo NebFuzz-LS. Analisando o *ranking* de Friedman calculado a partir do índice NMI, o algoritmo NebFuzz-LP teve o segundo melhor desempenho até 10% de dados rotulados. Para 20% até 50% de dados rotulados, os desempenhos dos algoritmos NebFuzz, NebFuzz-LP e NebFuzz-GS se alternam como segundo melhor desempenho.

A análise estatística do teste de Friedman e o pós-teste de Holm dos algoritmos NebFuzz e versões com ponderação automática de variáveis é apresentada na Tabela 47. Para fins de simplificação, nessa tabela o nome do algoritmo foi abreviado de "NebFuzz"para "Neb". A análise da Tabela 47 mostra que o algoritmo NebFuzz-GP obteve os melhores resultados, principalmente em relação ao algoritmo NebFuzz-LS. Em relação ao algoritmo NebFuzz, para a maioria das configurações de dados rotulados, o algoritmo NebFuzz-GP foi melhor ao analisar a acurácia ou o índice NMI. O algoritmo NebFuzz-GP é melhor que os outros algoritmos dessa família para 50% de dados rotulados.

6.1.2.5 Isodata-PS com distância adaptativa

Os resultados do desempenho dos algoritmos Isodata-PS, Isodata-PS-PG, Isodata-PS-PL, Isodata-PS-GS e Isodata-PS-LS são apresentados na Figura 12. São apresentados



(a) *Ranking* de Friedman - Acurácia

(b) *Ranking* de Friedman - NMI

(c) Acurácia (%)

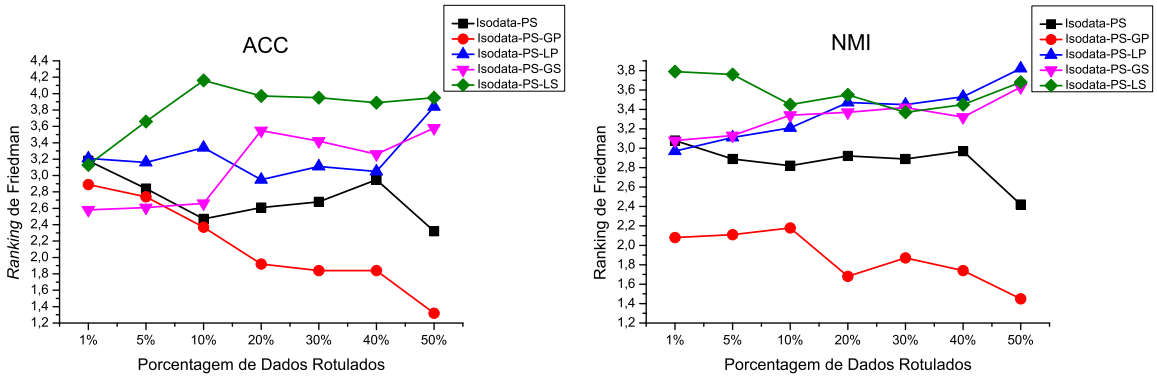
(d) NMI

Figura 11 – *Ranking* de Friedman e desempenho dos algoritmos NebFuzz e suas respectivas versões com distância adaptativa.

Tabela 47 – Teste de Friedman e pós-teste de Holm a um nível significância de 5% dos algoritmos da família NebFuzz.

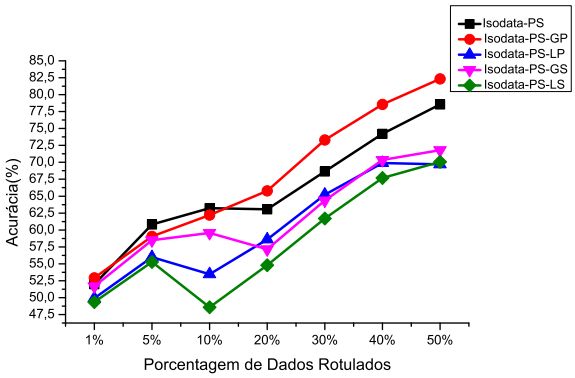
		Porcentagem de Rótulos													
		1%		5%		10%		20%		30%		40%		50%	
		ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
NebFuzz	Neb-GP	=	=	=	<	=	=	=	=	=	<	<	=	<	<
	Neb-GS	=	=	=	=	=	=	=	=	=	=	=	=	=	=
	Neb-LP	=	=	=	=	=	=	=	=	=	=	=	=	=	=
	Neb-LS	=	=	=	=	=	=	=	=	=	=	=	=	=	=
Neb-GP	Neb-GS	=	=	=	=	=	=	=	=	=	=	=	=	>	>
	Neb-LP	=	=	=	=	=	>	=	>	=	=	=	>	>	>
	Neb-LS	>	>	>	>	>	>	>	>	>	>	>	>	>	>
Neb-GS	Neb-LP	=	=	=	=	=	=	=	=	=	=	=	=	=	=
	Neb-LS	=	=	=	=	>	=	>	=	=	=	>	=	=	=
Neb-LP	Neb-LS	=	=	=	=	=	=	=	=	=	=	=	=	=	=

gráficos mostrando o *ranking* de Friedman do desempenho da acurácia e do índice NMI em bases de dados reais. Analisando o *ranking* de Friedman calculado a partir da acurácia e do índice NMI, nota-se que o algoritmo que obteve melhor desempenho foi o Isodata-PS-GP. O algoritmo Isodata-PS com distância Euclidiana foi melhor que o restante dos algoritmos com ponderação automática de variáveis. O algoritmo que teve desempenho inferior nesse experimento foi o Isodata-PS-LS.

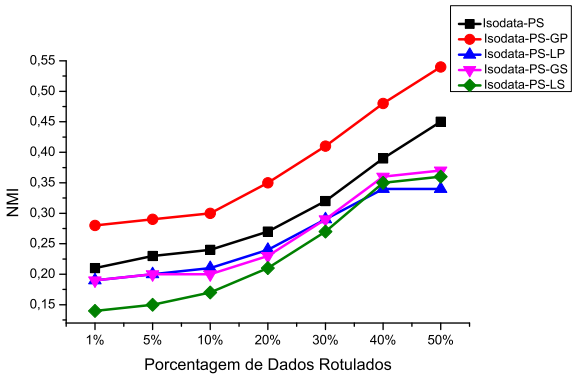


(a) *Ranking* de Friedman - Acurácia

(b) *Ranking* de Friedman - NMI



(c) Acurácia (%)



(d) NMI

Figura 12 – *Ranking* de Friedman e desempenho dos algoritmos Isodata-PS e suas respectivas versões com distância adaptativa.

A análise estatística do teste de Friedman e o pós-teste de Holm dos algoritmos Isodata-PS e versões com ponderação automática de variáveis é apresentada na Tabela 48. Nessa tabela o nome do algoritmo foi abreviado de "Isodata-PS"para "Iso". A análise da Tabela 48 mostra que o algoritmo Isodata-PS com distância Euclidiana foi pior apenas que a variação com distância adaptativa Isodata-PS-GP com 20% e 40% de dados rotulados. De 20% até 50% de dados rotulados, o algoritmo Isodata-GP é melhor que todos os outros algoritmos dessa família com ponderação automática de variáveis. O algoritmo Isodata-PS é melhor que os algoritmos Isodata-PS-LP, Isodata-PS-GS e Isodata-PS-LS com 50% de dados rotulados. O algoritmo Isodata-PS-LS desempenho inferior aos outros algoritmos em praticamente todas configurações de dados rotulados.

Tabela 48 – Teste de Friedman e pós-teste de Holm a um nível significância de 5% dos algoritmos da família Isodata-PS.

		Porcentagem de Rótulos													
		1%		5%		10%		20%		30%		40%		50%	
		ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
Iso	Iso-GP	=	=	=	=	=	=	=	<	=	=	=	<	=	=
	Iso-GS	=	=	=	=	=	=	=	=	=	=	=	=	>	>
	Iso-LP	=	=	=	=	=	=	=	=	=	=	=	=	>	>
	Iso-LS	=	=	=	=	>	=	>	=	>	=	=	=	>	>
Iso-GP	Iso-GS	=	=	=	=	=	=	=	>	>	>	>	>	>	>
	Iso-LP	=	=	=	=	=	>	>	>	>	>	>	>	>	>
	Iso-LS	=	>	=	>	>	>	>	>	>	>	>	>	>	>
Iso-GS	Iso-LP	=	=	=	=	=	=	=	=	=	=	=	=	=	=
	Iso-LS	=	=	=	=	>	=	>	=	=	=	=	=	=	=
Iso-LP	Iso-LS	=	=	=	=	=	=	=	=	=	=	=	=	=	=

6.1.2.6 Discussão

Podemos destacar nesse experimento que os algoritmos propostos com ponderação automática de variáveis tiveram os melhores resultados para a maioria das configurações dos experimentos apresentados. Para cada uma dos cinco algoritmos de agrupamento semissupervisionado modificados com distância adaptativa, sempre havia pelo menos uma versão que obteve melhores resultados quando comparado com o algoritmo sem distância adaptativa que o originou.

Em geral, os melhores algoritmos com distância adaptativa baseado no algoritmo BSSC foram os algoritmos BSSC-LP e BSSC-LS para poucos dados rotulados (1% até 10%) e o algoritmo BSSC-GP para configurações de 20% a 50% de dados rotulados. Note que as versões com ponderação local de variáveis foram melhores para poucos dados rotulados, e com ponderação global foi melhor para mais dados rotulados.

Ao fazer uma análise para os algoritmos com distância adaptativa baseados no algoritmo sSSC mostra que para configurações menores que 10% de dados rotulados o melhor desempenho foi obtido pelos algoritmos sSSC-GP e sSSC-LP. De 20% a 50% de dados rotulados, o melhor desempenho foi obtido pelos algoritmos sSSC-GS e sSSC-LS. Esse resultado demonstra que as versões com ponderação global também podem funcionar bem com poucos dados rotulados. O destaque nessa versão com mais dados rotulados foi para os algoritmos com restrição de soma, tanto com ponderação local, quanto com ponderação global de variáveis.

Analisando os algoritmos com distância adaptativa baseados no algoritmo ESSC, nota-se o destaque dos algoritmos ESSC-GP e ESSC-LP. A restrição em que o produto dos pesos das variáveis é igual a um funcionou melhor no cálculo da ponderação automática de variáveis para os algoritmos baseados no ESSC.

Em geral, o melhor algoritmo com distância adaptativa baseado no algoritmo NebFuzz foi o algoritmo NebFuzz-GP. Assim como, o melhor algoritmo baseado no algoritmo Isodata-PS foi o algoritmo Isodata-PS-GP. Note que a versão com ponderação global de variáveis em que a restrição do produto dos pesos das variáveis é igual a um foi destaque em todos os algoritmos de agrupamento semissupervisionado considerados nessa

tese. Também, pode-se observar que os rótulos disponíveis sempre ajudaram a melhorar o desempenho desses algoritmos.

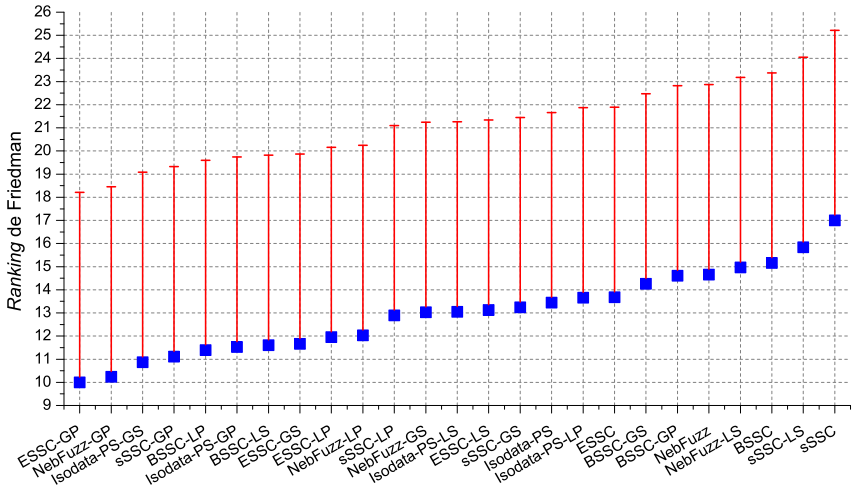
6.1.3 Algoritmos de agrupamento semissupervisionado

Nesta seção apresentamos uma comparação envolvendo todos os algoritmos de agrupamento semissupervisionado considerados neste trabalho. O objetivo aqui é avaliar esses algoritmos em termo de desempenho no experimento com bases de dados reais. Seguindo o trabalho de Delgado et al. (FERNANDEZ-DELGADO et al., 2014), utilizamos o *ranking* de Friedman para ordenar os algoritmos. Os algoritmos com valores mais baixos do *ranking* de Friedman possuem melhor desempenho. Pra fins de simplificação, são apresentados os resultados para 1%, 10%, 30% e 50% de dados rotulados para o *ranking* de Friedman calculados a partir da acurácia e do índice NMI. O pós-teste de Nemenyi compara todos os algoritmos envolvidos no experimento. O valor do *ranking* de Friedman de dois algoritmos são comparados, caso a diferença entre esses valores seja maior que o valor calculado pelo pós-teste de Nemenyi, esses algoritmos possuem uma diferença estatística significativa. Nesse experimento, o valor calculado pelo pós-teste de Nemenyi foi $CD = 8,21$. Essa diferença é indicada pelo intervalo vertical desenhado acima dos símbolos de cada algoritmo. Para saber se um algoritmo é ou não equivalente a outro, basta traçar uma linha reta horizontal no final do intervalo vertical desse algoritmo. Caso o símbolo do algoritmo (quadrado) comparado esteja além dessa linha reta horizontal, os algoritmos são estatisticamente diferentes, senão, são estatisticamente equivalentes.

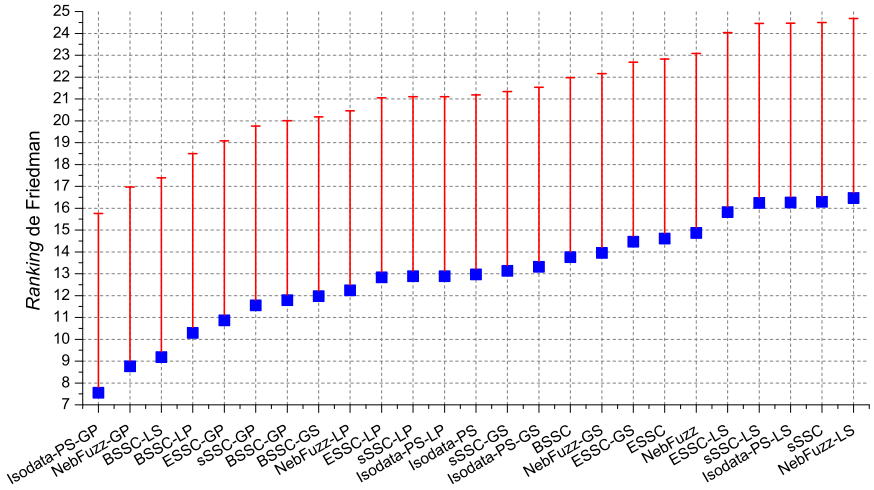
A análise do resultado com 1% de dados rotulados apresentado na Figura 13 mostra que cinco algoritmos tiveram versões com distância adaptativas entre os melhores desempenhos. A Figura 13(a) mostra que os quatro melhores desempenhos foram obtidos por versões com ponderação global. O algoritmo BSSC-LP e BSSC-LP foram os melhores colocados com ponderação local. A restrição de produto é maioria também entre os primeiros colocados. O pós-teste de Nemeny não mostra nenhuma diferença estatística entre os algoritmos. A Figura 13(b) mostra algoritmos diferentes nas primeiras posições. O algoritmo melhor colocado foi Isodata-PS-GP, que é melhor estatisticamente do que os algoritmos posicionados após o algoritmo sSSC-LS.

O *ranking* de Friedman com 10% de dados rotulados é apresentado na Figura 14. Quatro algoritmos tiveram versões com distância adaptativas entre os melhores desempenhos. A exceção foram os algoritmos baseados no algoritmo sSSC. A restrição de produto é maioria também entre os primeiros colocados. Os três melhores algoritmos foram o ESSC-GP e ESSC-LP e o NebFuzz-GP, todos com restrição de produto. O primeiro algoritmo é estatisticamente melhor do que os algoritmos posicionados a partir do algoritmo sSSC-GP na Figura 14(a) e a partir do algoritmo BSSC-GP na Figura 14(a).

O *ranking* de Friedman calculado a partir da acurácia e do índice NMI com 30%



(a) *Ranking* de Friedman - Acurácia

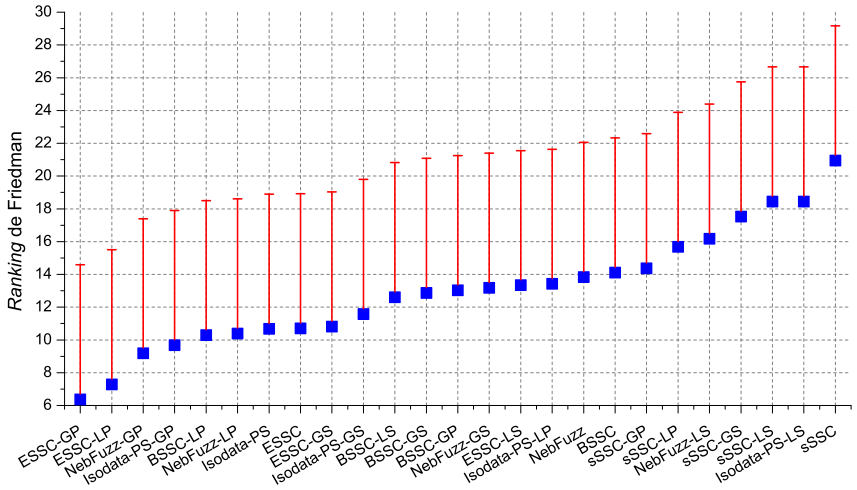


(b) *Ranking* de Friedman - NMI

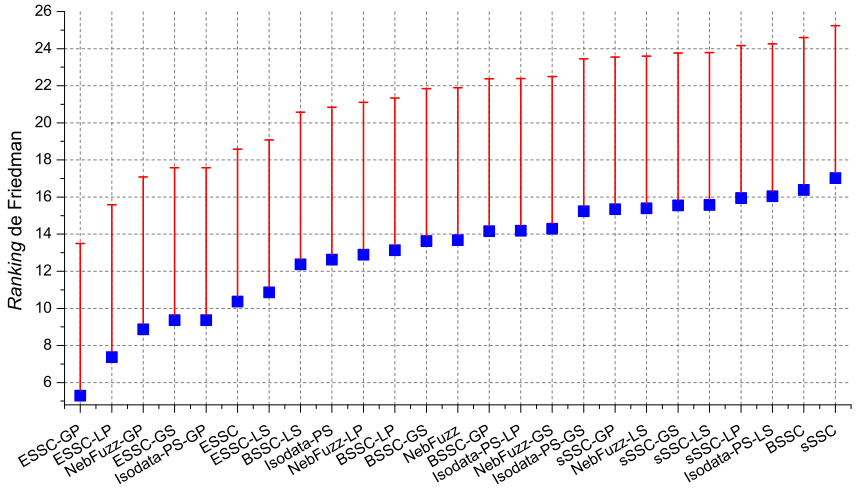
Figura 13 – *Ranking* de Friedman para 1% de dados rotulados

de dados rotulados são apresentados na Figura 15. Os algoritmos com distância adaptativa baseados no algoritmo ESSC, incluindo a versão sem distância adaptativa e o algoritmo Isodata-PS-GP, tiveram os melhores desempenhos nessa configuração. O algoritmo Isodata-PS-GP obteve o terceiro melhor desempenho. Os piores resultados foram obtidos pelos algoritmos baseados no sSSC. O algoritmo com melhor desempenho, ESSC-GP, é melhor estatisticamente que os algoritmos posicionados a partir do algoritmo Isodata-PS-GS.

O *ranking* de Friedman calculado a partir da acurácia e do índice NMI com 50% de dados rotulados são apresentados na Figura 16. Novamente, os algoritmos com distância adaptativa baseados no algoritmo ESSC tiveram os melhores desempenhos. O algoritmo Isodata-PS-GP obteve novamente o terceiro melhor desempenho. O algoritmo ESSC-GP possui desempenho estatisticamente superior aos algoritmos posicionados a partir do algoritmo BSSC-GP.



(a) *Ranking* de Friedman - Acurácia



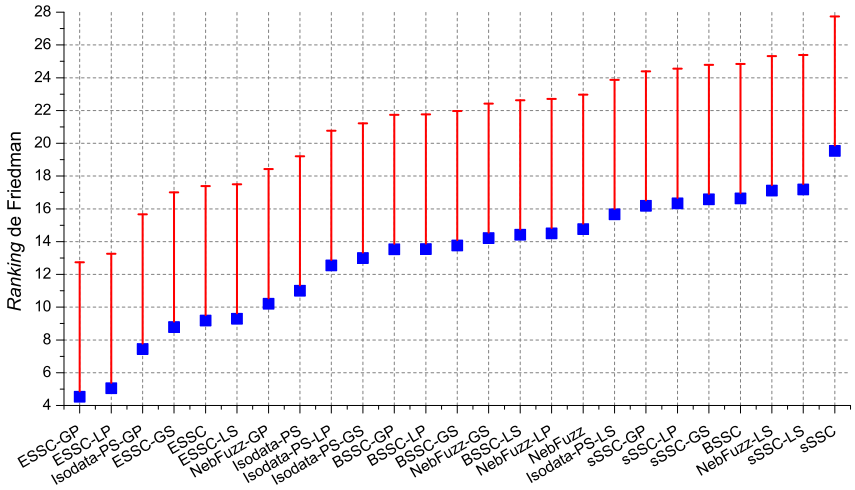
(b) *Ranking* de Friedman - NMI

Figura 14 – *Ranking* de Friedman para 10% de dados rotulados

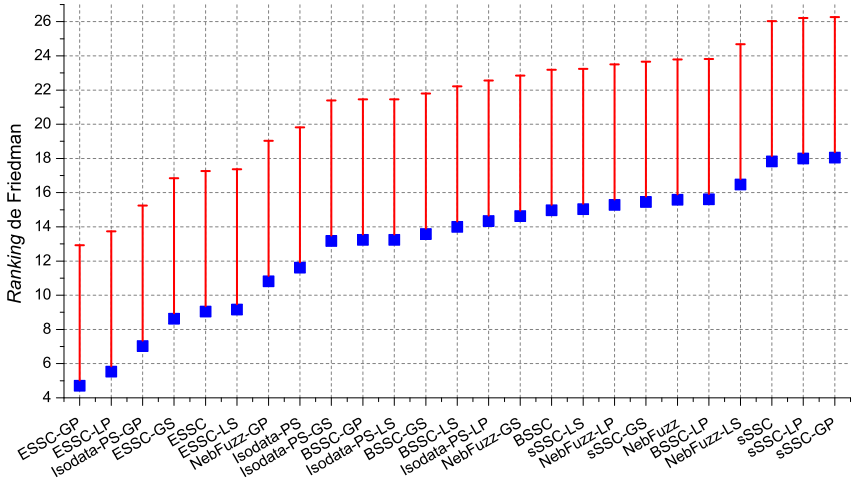
6.1.3.1 Discussão

Em geral, em termos de desempenho nesse experimento, destacamos as versões baseadas no algoritmo ESSC, principalmente as versões ESSC-GP e ESSC-LP. O algoritmo ESSC-GP obteve o melhor resultado para a maioria das configurações como mostra a Tabela 49. Além desses dois algoritmos, o algoritmo Isodata-PS-PG também obteve bons resultados. Podemos destacar que a maioria dos novos algoritmos de agrupamento semissupervisionado com distância adaptativa, geralmente, tiveram desempenho superior aos os algoritmos sem distância adaptativa. Os piores resultados foram obtidos pelas versões do algoritmo sSSC na maioria das configurações de porcentagem de dados rotulados como mostra a Tabela 49.

Esses resultados justificam a etapa extra para o cálculo dessa distância, fazendo com que os algoritmos tenham uma adequação melhor aos dados originais. Assim, en-



(a) *Ranking* de Friedman - Acurácia



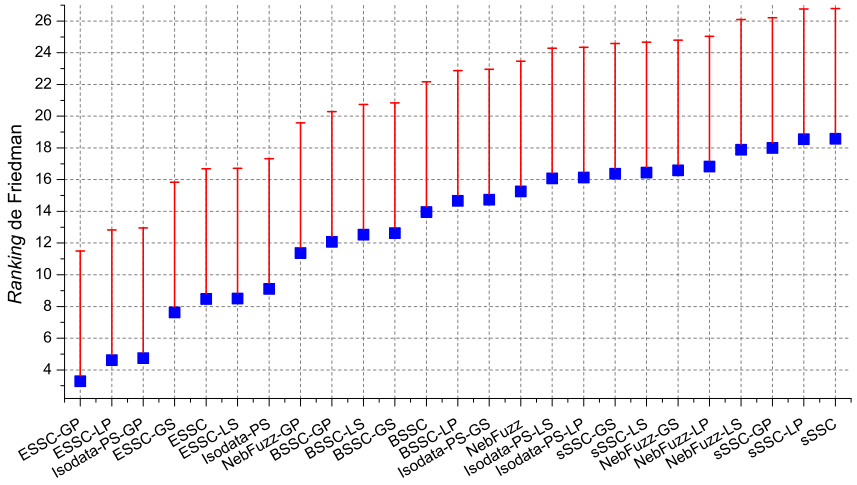
(b) *Ranking* de Friedman - NMI

Figura 15 – *Ranking* de Friedman para 30% de dados rotulados

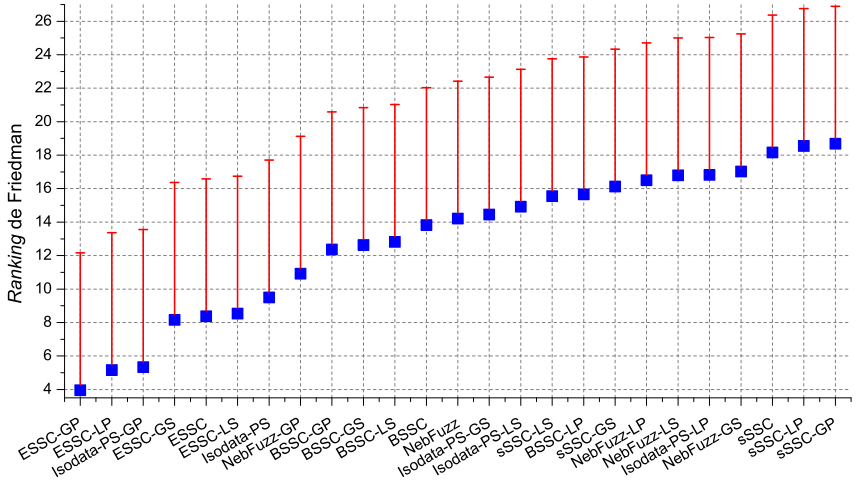
Tabela 49 – Melhor e pior algoritmo por porcentagem

		1%	10%	30%	50%
ACC	Melhor	ESSC-GP	ESSC-GP	ESSC-GP	ESSC-GP
	Pior	sSSC	sSSC	sSSC	sSSC
NMI	Melhor	Isodata-PS-GP	ESSC-GP	ESSC-GP	ESSC-GP
	Pior	NebFuzz-LS	sSSC	sSSC-GP	sSSC-GP

fatizamos que os novos algoritmos de agrupamento semissupervisionado com distâncias adaptativas são uma alternativa viável em aplicações que existam dados rotulados e não rotulados disponíveis para o treinamento.



(a) *Ranking* de Friedman - Acurácia



(b) *Ranking* de Friedman - NMI

Figura 16 – *Ranking* de Friedman para 50% de dados rotulados

6.1.4 Análise dos rótulos

Os algoritmos semissupervisionados levam em conta rótulos em seu treinamento. Desse modo, procuramos uma estratégia inteligente para selecionar os exemplos localizados na fronteira dos grupos formados pelo algoritmo *Fuzzy C-Means* (ver Seção 5.2.2.4). Os dados rotulados seguem a mesma configuração do experimento anterior (1%, 5%, 10%, 20%, 30%, 40% e 50%). Esta seção apresenta resultados da análise dessa seleção em todos os algoritmos de agrupamento semissupervisionado considerados nesta tese. O teste de Wilcoxon compara a versão do algoritmo com os rótulos selecionados aleatoriamente contra a mesma versão do algoritmo com rótulos selecionados na fronteira. A Tabela 50 apresenta os resultados dessa análise.

Os algoritmos que mais se beneficiaram da seleção de rótulos foram os algoritmos BSSC e suas versões com ponderação automática de variáveis e o algoritmo sSSC e suas

versões com ponderação automática de variáveis. Para todos os algoritmos, pelo menos em alguma configuração de dados rotulados houve melhora significativa nos resultados e em nenhum caso essa seleção de exemplos piorou em relação à seleção aleatória. Esse fato indica que a utilização dessa seleção de rótulos é efetiva e, se não melhorar, possui desempenho estatisticamente igual a seleção aleatória.

Tabela 50 – Comparação entre o algoritmo com seleção aleatória de rótulos e o algoritmo com seleção de rótulos na fronteira utilizando o Teste de Wilcoxon a um nível significância de 5%.

	Porcentagem de Rótulos													
	1%		5%		10%		20%		30%		40%		50%	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
BSSC	<	=	<	=	<	<	<	<	<	<	<	<	<	<
BSSC-GP	<	=	<	<	<	<	<	<	<	<	<	<	<	<
BSSC-LP	<	<	<	=	<	<	<	<	=	<	<	<	<	<
BSSC-GS	=	=	<	=	<	=	<	<	<	<	<	<	<	<
BSSC-LS	<	=	<	=	<	<	<	<	<	<	<	<	<	<
sSSC	<	=	<	<	<	<	<	<	<	<	<	<	<	<
sSSC-GP	=	=	<	=	<	=	<	<	<	<	<	<	<	<
sSSC-LP	<	=	<	=	<	<	=	<	<	<	<	<	<	<
sSSC-GS	<	=	<	=	<	=	<	=	<	<	=	<	=	<
sSSC-LS	<	=	<	=	<	=	=	=	<	<	<	<	<	<
ESSC	<	=	=	=	=	=	=	=	=	<	=	=	=	=
ESSC-GP	<	=	=	<	=	=	=	=	=	<	=	<	=	<
ESSC-LP	=	=	=	<	=	<	<	<	<	<	=	<	=	<
ESSC-GS	<	=	=	=	=	=	=	=	=	=	=	=	=	=
ESSC-LS	=	=	=	=	=	=	=	<	=	=	=	=	=	=
NebFuzz	<	=	<	=	=	=	<	=	=	=	=	=	=	=
NebFuzz-GP	<	=	=	=	=	<	=	=	=	=	=	=	=	=
NebFuzz-LP	=	=	=	=	=	=	=	=	=	<	=	=	=	=
NebFuzz-GS	<	=	<	=	=	=	=	=	=	=	=	=	=	=
NebFuzz-LS	=	=	<	=	=	=	=	<	=	=	=	=	=	=
Isodata-PS	<	=	=	=	=	=	=	=	=	=	=	=	=	<
Isodata-PS-GP	<	=	<	=	<	=	=	=	=	=	=	=	=	<
Isodata-PS-LP	<	=	=	=	=	=	<	=	<	=	=	=	=	<
Isodata-PS-GS	<	=	<	=	<	=	<	=	<	=	=	=	=	<
Isodata-PS-LS	<	=	<	=	<	=	<	=	<	=	=	=	=	=

6.2 BASES DE DADOS SIMULADOS

Para demonstrar a capacidade dos métodos de agrupamento semissupervisionados aprenderem dinamicamente os pesos das variáveis apropriadamente, foram consideradas duas configurações de conjuntos de dados em três dimensões com quatro e três grupos, respectivamente, cada grupo constituído por cem padrões.

Para as configurações de dados simulados 1 e 2, a acurácia (ACC) e o índice NMI para os métodos de agrupamento com ponderação automática considerados nessa tese foram estimados através de um experimento onde foram criadas 25 réplicas de cada uma dessas bases de dados simuladas. Em cada réplica, cada algoritmo foi executado 100 vezes com diferentes configurações no conjunto de dados rotulados nos mesmos moldes do experimento anterior. Para cada conjunto de dados simulados, foram realizadas a análise do parâmetro alfa (α) e, por conseguinte, os algoritmos com ponderação automática de variáveis foram comparados entre si em termos de desempenho. O teste de Friedman também foi realizado nesse experimento e o valor do pós-teste de Nemenyi para esses experimentos foi $CD = 7,15885$, apresentado pelo intervalo no eixo-Y mostrado nas figuras desta seção. No Apêndice B são apresentados o desempenho médio da acurácia e do índice NMI.

6.2.1 Base de dados simulados 1

A análise do parâmetro alfa foi realizada da mesma maneira apresentada na Seção 6.1.1, com a diferença que os algoritmos foram executados nas 25 bases de dados do conjunto de dados simulados 1. Para fins de simplificação, é apresentada apenas a tabela final contendo os melhores valores de alfa para cada um dos algoritmos considerados nessa tese nas bases de dados simulados 1. A Tabela 51 apresenta os resultados da análise do parâmetro alfa nas bases de dados simulados 1.

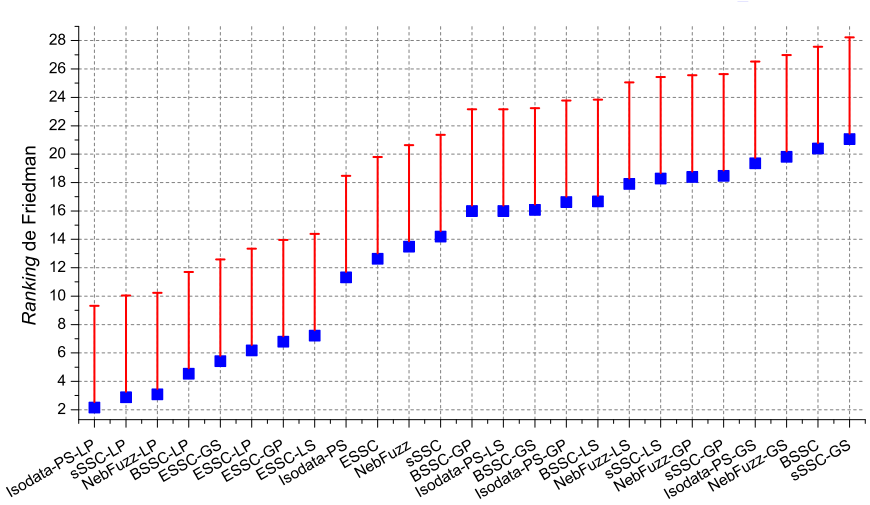
Tabela 51 – Valores de alfa selecionados para bases de dados simulados 1

	Alfa						
Algoritmos/Porcentagem	1%	5%	10%	20%	30%	40%	50%
BSSC	0,5	0,1	1	1	1	1	1
BSSC-GP	0,5	1	1	1	1	1	1
BSSC-LP	0,1	0,1	0,1	0,1	0,1	0,1	0,1
BSSC-GS	0,5	1	1	1	1	1	1
BSSC-LS	0,1	0,1	0,1	0,1	0,1	0,1	0,1
ESSC	1	1	1	0,5	0,5	0,5	0,5
ESSC-GP	0,1	0,1	50	0,5	0,5	50	0,1
ESSC-LP	5	10	100	5	5	1	1
ESSC-GS	5	NS(0,5)	NS(0,1)	NS(0,5)	NS(0,5)	50	50
ESSC-LS	0,1	100	10	10	10	10	100
NebFuzz	NS(50)	10	10	5	5	5	5
NebFuzz-GP	NS(0,1)	10	10	100	5	1	1
NebFuzz-LP	NS(10)	NS(1)	NS(1)	1	0,1	0,1	0,5
NebFuzz-GS	NS(10)	NS(5)	100	10	10	100	10
NebFuzz-LS	NS(0,5)	NS(0,1)	NS(10)	10	100	10	10
Isodata-PS	0,1	0,1	0,1	0,1	0,1	0,1	0,1
Isodata-PS-GP	0,5	0,5	0,5	0,5	1	1	1
Isodata-PS-LP	0,1	0,5	0,1	0,1	1	1	1
Isodata-PS-GS	0,1	0,5	50	1	1	1	50
Isodata-PS-LS	0,5	0,5	1	1	1	50	1

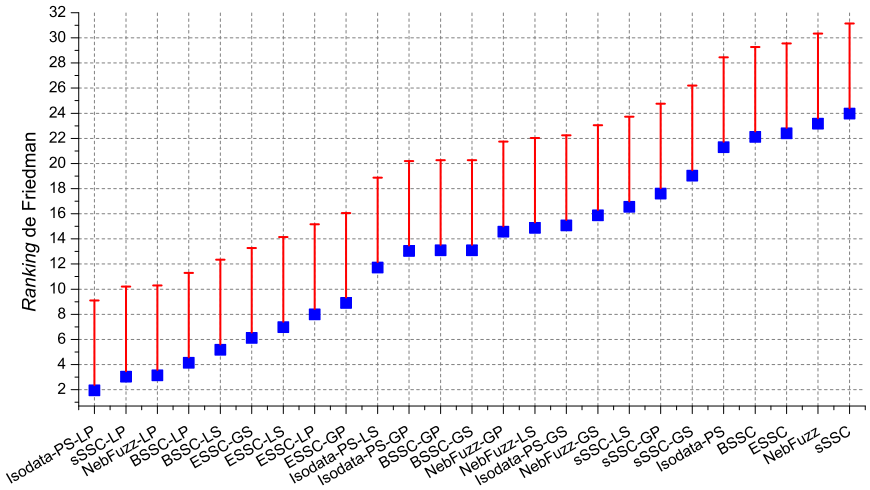
Os resultados do parâmetro alfa para as bases de dados simulados 1 tiveram a maioria de seus valores escolhidos por diferença estatística significativa. Apenas algumas configurações de dados rotulados para os algoritmos baseados no NebFuzz e o algoritmo ESSC-GS não tiveram diferença estatística. Existe algumas coincidências desses valores com os encontrados nas bases de dados reais. Por exemplo, os algoritmos NebFuzz-LS e Isodata-PS-GP tiveram apenas uma configuração diferente. Entretanto, também houve casos onde ocorreram muitas mudanças, os algoritmos baseados no algoritmo ESSC para essas bases de dados tiveram valores de alfa mais altos em relação aos valores de alfa encontrados para as bases de dados reais. Contudo, devido a diversas mudanças encontradas, isso confirma que esse parâmetro é bastante influenciado pelo problema analisado. Os valores de alfa selecionados nesse experimento são utilizados na avaliação dos algoritmos nessa base de dados.

De agora em diante, são apresentados os resultados para as bases de dados simulados 1. Os resultados apresentados são para 1%, 10%, 30% e 50% de dados rotulados.

Primeiro, os resultados para o experimento para as bases de dados simulados 1 com 1% de dados rotulados são apresentados na Figura 17, onde nota-se que oito algoritmos se destacaram nesse experimento. Desses, sete são algoritmos com ponderação local de variáveis. Com exceção dos algoritmos ESSC-GS e ESSC-GP, os melhores desempenhos foram obtidos por versões dos algoritmos semissupervisionados de ponderação local de variáveis com restrição do produto das variáveis igual a um. Ao analisar a acurácia, alguns algoritmos com distância padrão Euclidiana se destacaram: NebFuzz, Isodata-PS e ESSC. Porém ao analisar o índice NMI, os algoritmos com distância Euclidiana tiveram os piores desempenhos.



(a) Acurácia (1% de Dados Rotulados)

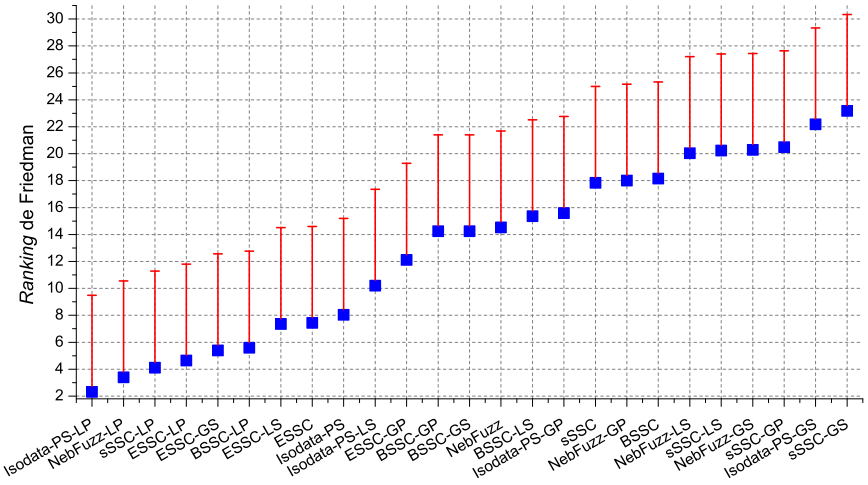


(b) NMI (1% de Dados Rotulados)

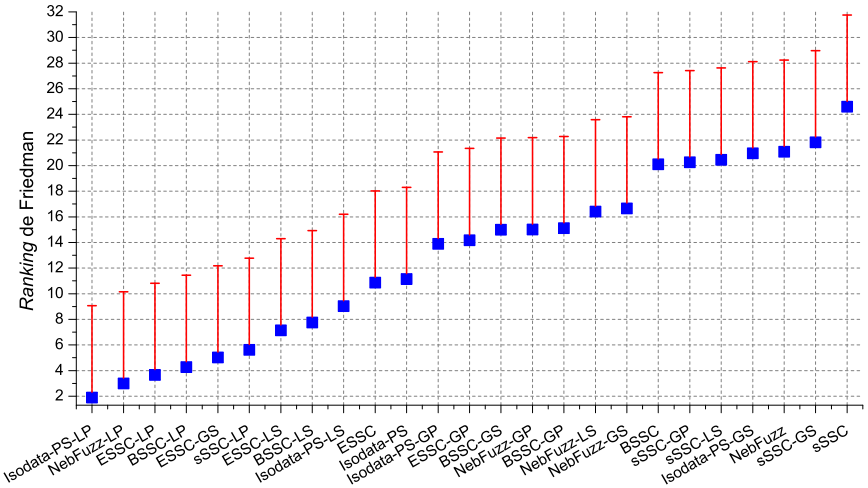
Figura 17 – Comparação dos algoritmos na base de dados simulados 1 com 1% de dados rotulados.

Os resultados para as bases de dados simulados 1 com 10% de dados rotulados são apresentados na Figura 18. Novamente, os melhores resultados são obtidos por algoritmos semissupervisionados com ponderação de variáveis local e restrição em que o

produto dos pesos das variáveis deve ser igual a um. Os algoritmos ESSC e Isodata-PS com distância Euclidiana tiveram os melhores desempenhos quando comparados com os algoritmos NebFuzz, BSSC e sSSC. Porém, os melhores algoritmos dessas versões foram o Isodata-PS-LP, NebFuzz-LP, sSSC-LP, ESSC-LP e BSSC-LP nessa ordem. Algoritmos com ponderação local de variáveis em que a soma dos pesos das variáveis deve ser igual a um tiveram desempenho entre os 5 melhores analisando a acurácia e o índice NMI. Da mesma forma com 1% de dados rotulados, os únicos algoritmos com ponderação global que se destacaram foram os algoritmos ESSC-GS e ESSC-GP.



(a) Acurácia (10% de Dados Rotulados)

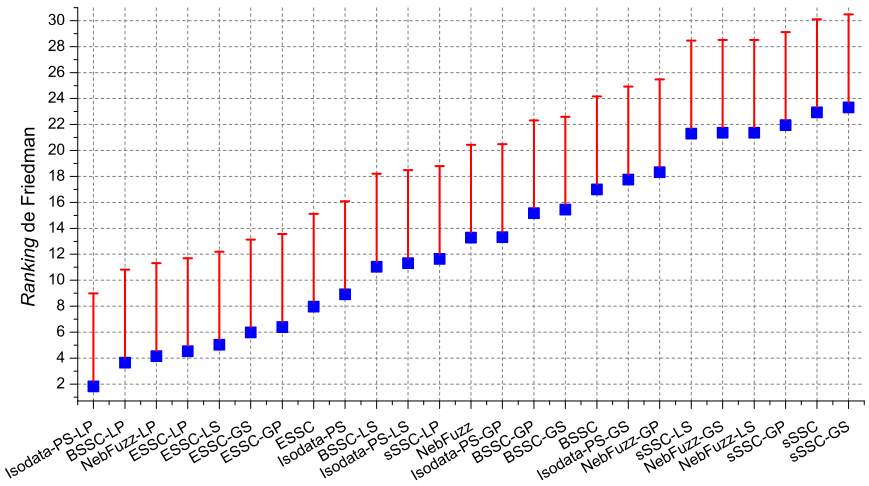


(b) NMI (10% de Dados Rotulados)

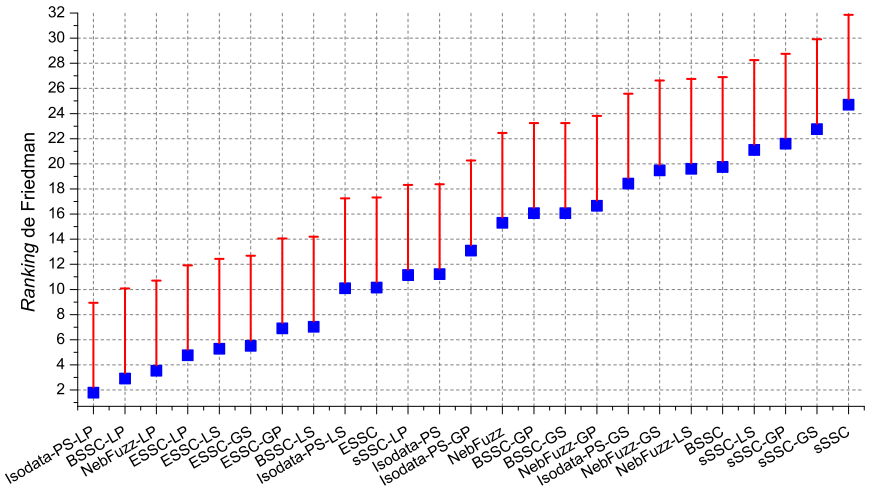
Figura 18 – Comparação dos algoritmos na base de dados simulados 1 com 10% de dados rotulados.

Analisando os resultados para as bases de dados simulados 1 com 30% de dados rotulados que são apresentados na Figura 19, nota-se que os cinco melhores desempenhos foram obtidos algoritmos com ponderação local de variáveis, sendo quatro deles com restrição de produto de variáveis igual a um, Isodata-PS-LP, BSSC-LP, NebFuzz-

LP e ESSC-LP. O quinto melhor desempenho foi obtido pelo algoritmo ESSC-LS. Entre os 10 melhores desempenhos, também encontram-se algoritmos com ponderação global, ESSC-GS e ESSC-GP, e algoritmos com distância Euclidiana, Isodata-PS e ESSC. Porém, novamente os melhores desempenhos foram obtidos com os algoritmos com ponderação local de variáveis e restrição em que o produto dos pesos das variáveis deve ser igual a um.



(a) Acurácia (30% de Dados Rotulados)

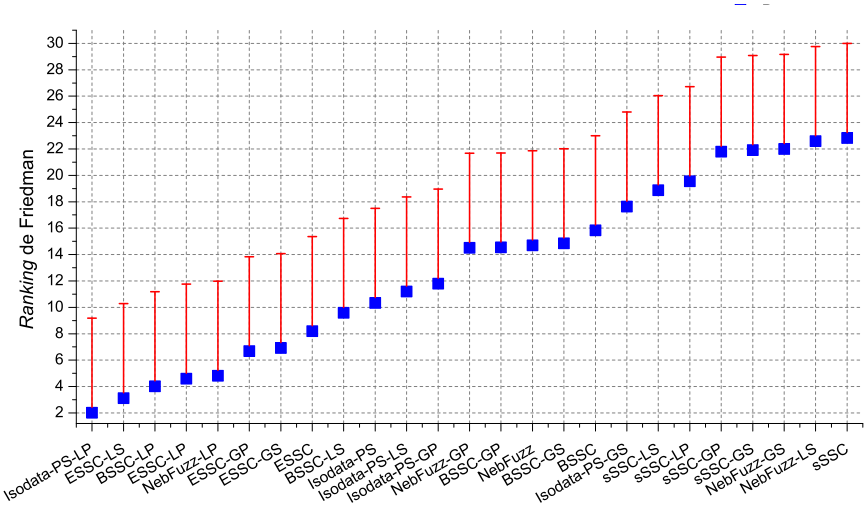


(b) NMI (30% de Dados Rotulados)

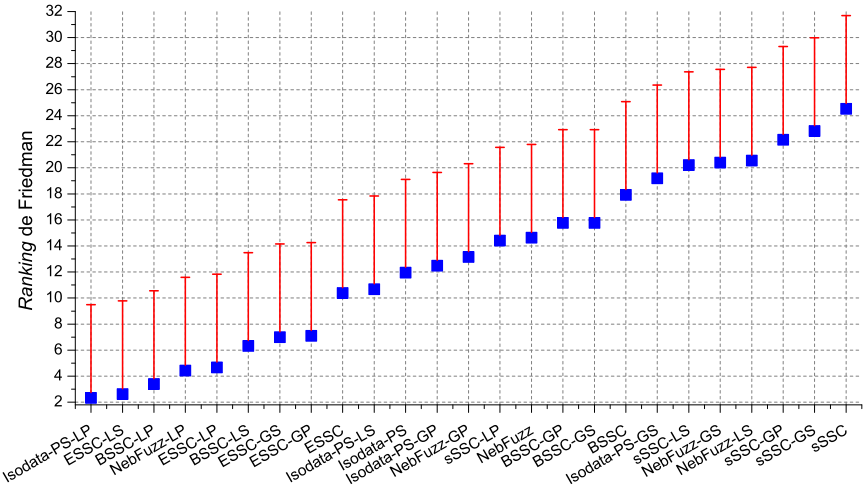
Figura 19 – Comparação dos algoritmos na base de dados simulados 1 com 30% de dados rotulados.

Enfim, os resultados para as bases de dados simulados 1 com 50% de dados rotulados que são apresentados na Figura 20. Percebe-se nesses resultados que novamente os algoritmos com melhor desempenho são algoritmos com ponderação local de variáveis e restrição em que o produto dos pesos das variáveis deve ser igual a um. Os melhores algoritmos nesse experimento foram: Isodata-PS-LP, ESSC-LS, BSSC-LP, ESSC-LP e NebFuzz-LP. As versões baseadas no algoritmo sSSC tiveram os piores desempenhos. A

maioria dos algoritmos nas dez primeiras posições são versões com ponderação local de variáveis.



(a) Acurácia (50% de Dados Rotulados)



(b) NMI (50% de Dados Rotulados)

Figura 20 – Comparação dos algoritmos na base de dados simulados 1 com 50% de dados rotulados

6.2.2 Base de dados simulados 2

A Tabela 52 apresenta os melhores valores do parâmetro alfa para cada um dos algoritmos considerados nessa tese nas bases de dados simulados 2. Note que os valores de alfa para essa base de dados são praticamente todos escolhidos por uma diferença estatística significativa. Apenas algumas configurações de dados rotulados para os algoritmos baseados no NebFuzz não tiveram diferença estatística. Existe apenas uma coincidência de valores com os encontrados nas bases de dados reais para o algoritmo Isodata-PS-GP na maioria de configurações de dados rotulados. Nessa base de dados, houve diversas mudanças encontradas em relação aos valores de alfa da base de dados simulados 1 e das bases

de dados reais, isso confirma mais uma vez que esse parâmetro é bastante influenciado pelo problema analisado. Os valores de alfa selecionados nesse experimento são utilizados na avaliação dos algoritmos nessa base de dados.

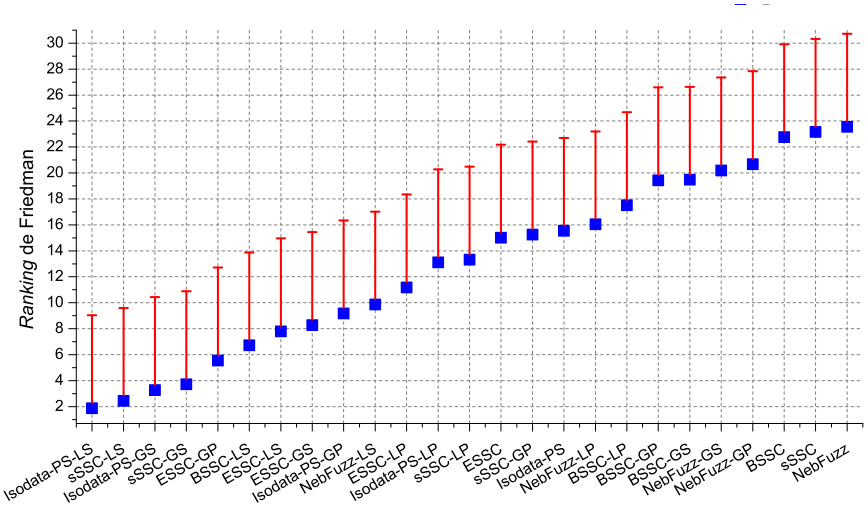
Tabela 52 – Valores de alfa selecionados para bases de dados simulados 2

	Alfa						
Algoritmos	1	5	10	20	30	40	50
BSSC	0,5	0,1	0,1	10	10	10	10
BSSC-GP	0,1	0,1	0,1	0,1	0,1	0,1	0,1
BSSC-LP	0,1	0,1	0,1	0,1	0,1	0,1	0,5
BSSC-GS	0,1	0,1	0,1	0,1	0,1	0,1	0,1
BSSC-LS	0,1	0,1	0,1	0,1	0,1	0,1	0,1
ESSC	1	50	1	1	1	1	1
ESSC-GP	0,1	50	50	5	5	5	5
ESSC-LP	0,5	0,5	0,5	0,5	10	50	50
ESSC-GS	0,1	0,5	0,5	50	0,5	10	10
ESSC-LS	5	5	0,5	10	0,1	0,1	0,1
NebFuzz	NS(5)	0,1	0,1	0,1	0,1	0,1	0,1
NebFuzz-GP	NS(100)	100	50	5	5	5	5
NebFuzz-LP	NS(0,5)	100	50	5	5	5	50
NebFuzz-GS	NS(5)	0,1	NS(0,5)	5	0,5	0,5	0,5
NebFuzz-LS	NS(0,5)	NS(0,1)	0,1	1	1	0,5	1
Isodata-PS	0,5	1	0,1	0,1	10	10	10
Isodata-PS-GP	0,5	0,5	0,5	0,5	0,5	1	1
Isodata-PS-GS	0,5	0,5	0,5	0,5	0,5	50	1
Isodata-PS-LP	0,5	0,1	1	1	1	50	1
Isodata-PS-LS	0,5	0,5	0,5	1	1	1	1

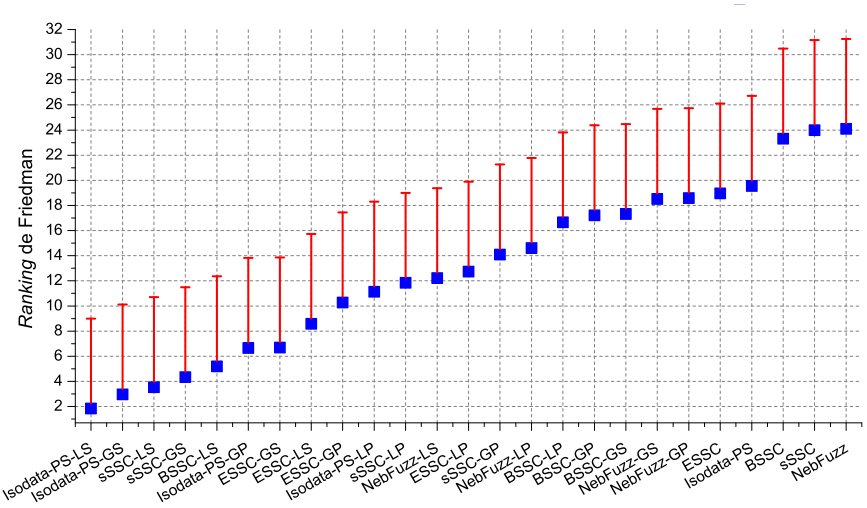
Os resultados do experimento para as bases de dados simulados 2 com 1% de dados rotulados são apresentados na Figura 21. As versões de algoritmos com distância adaptativa baseadas nos algoritmos sSSC e Isodata-PS tiveram os melhores desempenhos. Os melhores algoritmos foram o Isodata-PS-LS, Isodata-PS-GS, sSSC-LS e sSSC-GS. Note que a restrição de soma foram as melhores para essa configuração de dados rotulados. Os algoritmos com ponderação local e global de variáveis conseguiram igualmente bons resultados. As versões com distância adaptativa do algoritmo NebFuzz tiveram os piores desempenho nessas bases de dados com 1% de dados rotulados.

Os resultados do experimento para as bases de dados simulados 2 com 10% de dados rotulados são apresentados na Figura 22. Os algoritmos com ponderação automática de variáveis dos algoritmos Isodata-PS e sSSC alcançaram o melhor desempenho nesse experimento. Dos cinco melhores algoritmos, quatro possuem restrição de soma. Quanto à ponderação local ou global de variáveis, ambos alcançaram bons resultados nessa base de dados.

Os resultados do experimento para as bases de dados simulados 2 com 30% de dados rotulados são apresentados na Figura 23. Com 30% de dados rotulados, o melhor desempenho foi obtido pelo algoritmo ESSC-GP. Ainda, dos cinco melhores algoritmos, quatro são versões com ponderação global de variáveis.



(a) Acurácia (1% de Dados Rotulados)



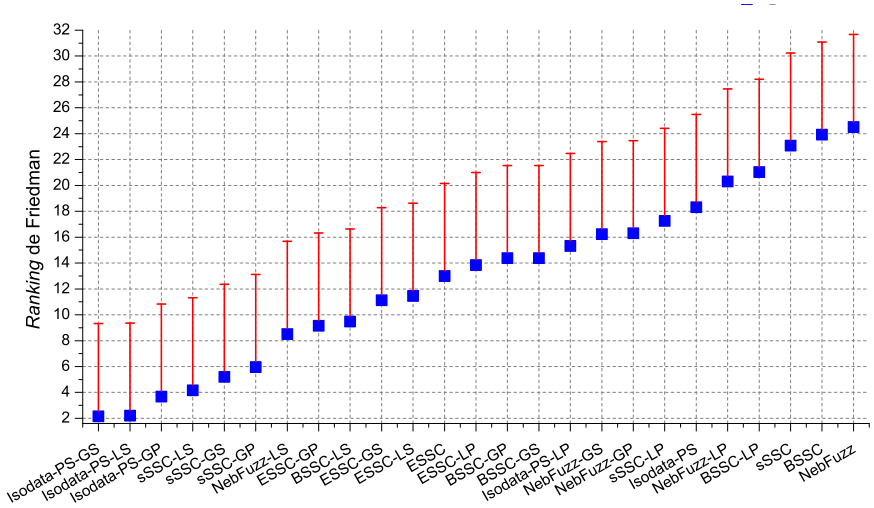
(b) NMI (1% de Dados Rotulados)

Figura 21 – Comparação dos algoritmos nas bases de dados simulados 2 com 1% de dados rotulados.

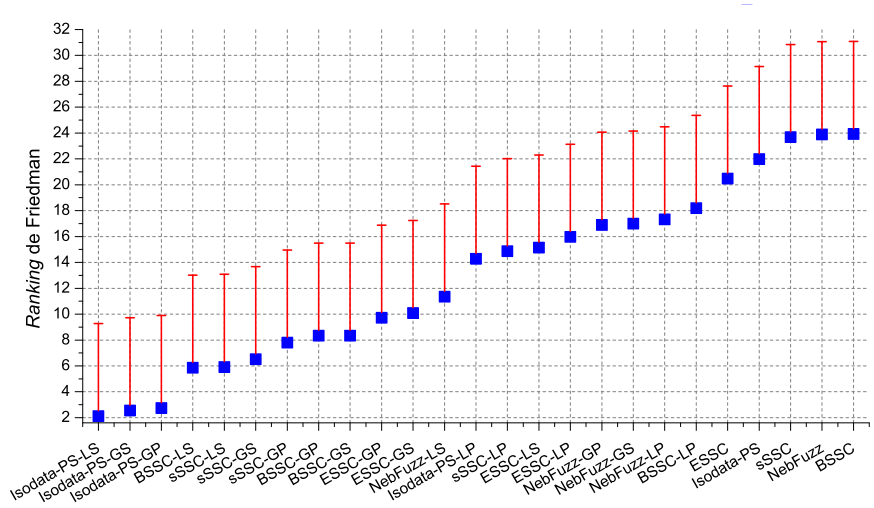
Os resultados do experimento para as bases de dados simulados 2 com 50% de dados rotulados são apresentados na Figura 24. O algoritmo com melhor desempenho foi o ESSC-GP, seguido dos algoritmos Isodata-PS-GP, Isodata-PS-LS, NebFuzz-GP e Isodata-PS-GS. Note que a maioria desses algoritmos são versões de distância adaptativa com ponderação global de variáveis.

6.2.3 Discussão

De acordo com os resultados encontrados nas bases de dados simulados 1, os algoritmos com distâncias adaptativas com ponderação local de variáveis sob a restrição de que o produto dos pesos das variáveis em cada grupo deve ser igual a um apresentaram os melhores desempenhos nessa base de dados. Para a configuração de dados simulados 2, os algoritmos com distâncias adaptativas com ponderação global de variáveis sob a restrição



(a) Acurácia (10% de Dados Rotulados)

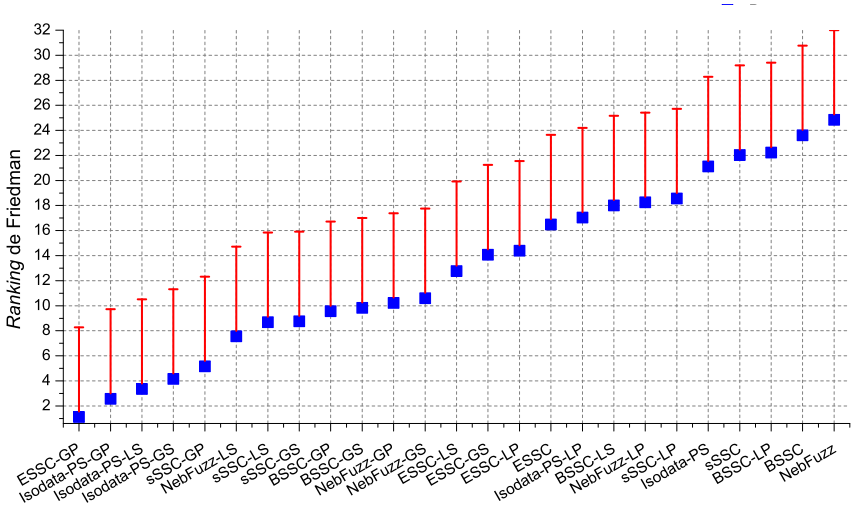


(b) NMI (10% de Dados Rotulados)

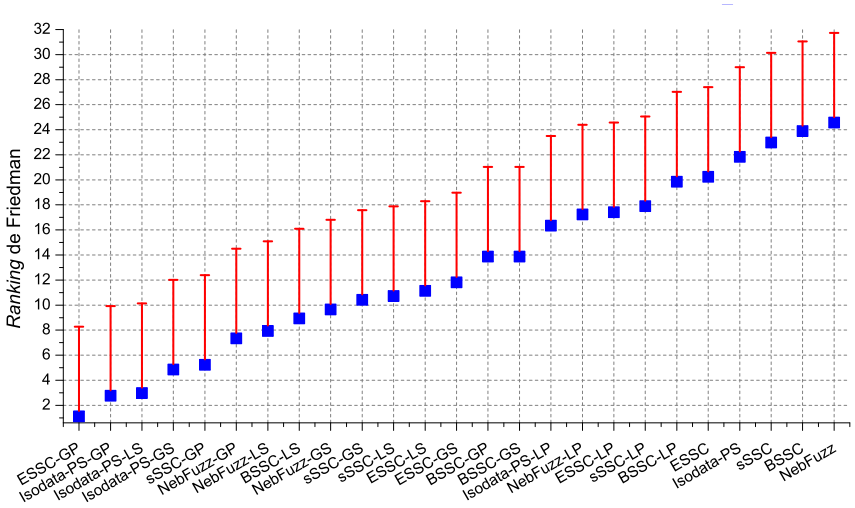
Figura 22 – Comparação dos algoritmos na base de dados simulados 2 com 10% dados rotulados.

de que o produto dos pesos das variáveis deve ser igual a um, geralmente, tiveram os melhores desempenhos. No entanto, na base de dados simulados 2, alguns algoritmos com ponderação local de variáveis também conseguiram bons desempenhos.

Com relação aos pesos das variáveis, são analisadas as melhores versões com ponderação automática de variáveis de cada algoritmo. Para o caso das bases de dados simulados 1, é apresentado o resultado para uma das réplicas com 10% de dados rotulados. Os algoritmos analisados são as versões com ponderação local de variáveis com restrição do produto dos pesos das variáveis igual a um. Nas três primeiras colunas da Tabela 53, são apresentados os pesos desses algoritmos. A matriz de confusão definida por esses pesos são apresentados nas últimas quatro colunas. Na base de dados simulados 1, as variáveis x_1 e x_2 definem os grupos 1 e 2, enquanto as variáveis x_2 e x_3 definem os grupos 3 e 4. Dessa forma, espera-se pesos relevantes para a variável x_1 e x_2 na definição dos grupos 1



(a) Acurácia (30% de Dados Rotulados)

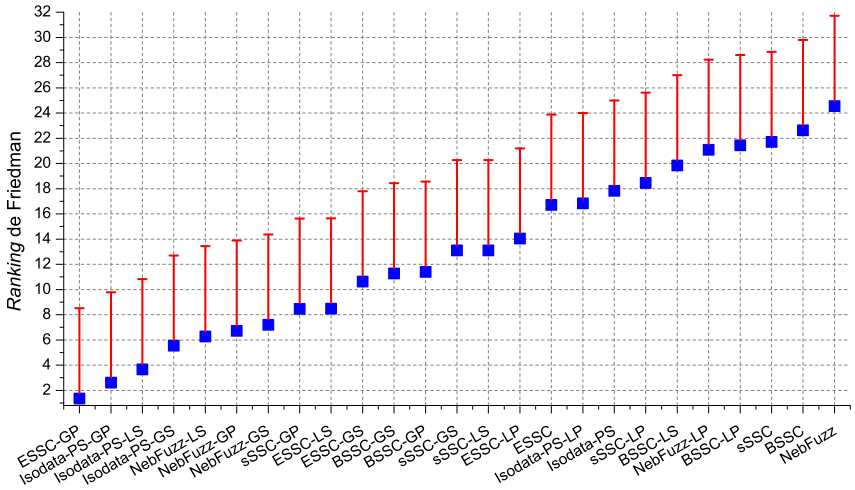


(b) NMI (30% de Dados Rotulados)

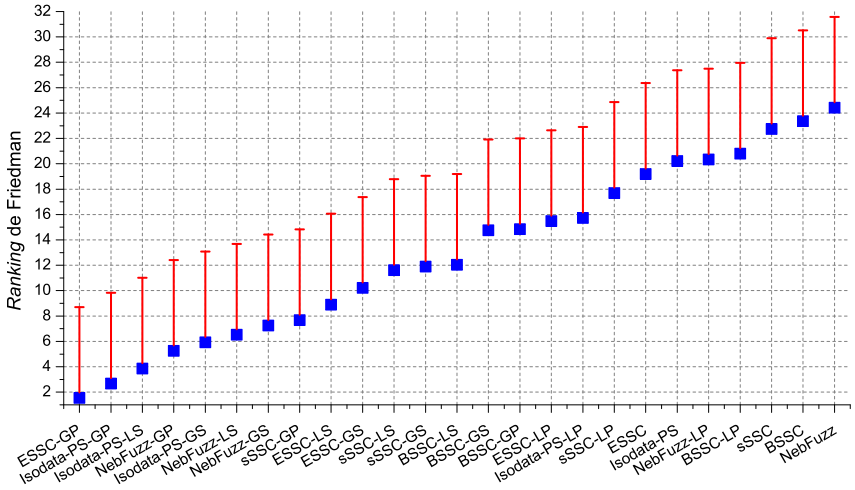
Figura 23 – Comparação dos algoritmos na base de dados simulados 2 com 30% de dados rotulados.

e 2 e pesos relevantes para as variáveis x_2 e x_3 na definição dos grupos 3 e 4. Note que a diferença no peso dessas variáveis definiram os grupos em todos os exemplos mostrados na Tabela 53. Os valores da variável x_2 são mais altos, pois está variável é relevante para todos os grupos. Desse modo, valores mais altos de x_1 associam os exemplos aos grupos de 1 e 2, enquanto valores mais altos de x_3 associam exemplos aos grupos 3 e 4. O único caso diferente foi o algoritmo BSSC-LP, em que valores da variável x_2 mais altos colocaram os exemplos nos grupos 1 e 2. A variável x_3 possui valores, em geral, mais baixos, porém quando os items são associados aos grupos 3 e 4, seu valor é pelo menos 4 vezes maior que os valores dessa mesma variável na definição dos grupos 1 e 2, como podemos observar no algoritmo ESSC-LP. Essa diferença é bem mais alta nos algoritmos sSSC-LP, NebFuzz-LP e Isodata-PS-LP.

Para o caso das bases de dados simulados 2, selecionamos uma das réplicas com 50%



(a) Acurácia (50% de Dados Rotulados)



(b) NMI (50% de Dados Rotulados)

Figura 24 – Comparação dos algoritmos na base de dados simulados 2 com 50% de dados rotulados.

de dados rotulados por se tratar de uma base de dados com distribuição mais complexa. São analisados os algoritmos com ponderação global de variáveis com restrição em que o produto dos pesos das variáveis é igual a um. Os pesos desses algoritmos são apresentados na Tabela 54 nas três primeiras colunas e é apresentada a matriz de confusão definida por esses pesos nas últimas três colunas. Na base de dados simulados 2, as variáveis x_1 e x_2 são relevantes para os grupos 1 e 2, enquanto a variável x_3 representa ruído. Nessa base de dados, os pesos de todos os algoritmos tiveram características semelhantes, pesos com valor baixo na variável x_1, peso com valores alto na variável x_2, e um valor médio na variável x_3. Nesse tipo de algoritmo, esses pesos são iguais para todos os grupos e os algoritmos encontraram que a ordem de relevância das variáveis é $(x_2 \geq x_3 \geq x_1)$.

Tabela 53 – Pesos das bases de dados simulados 1

Algoritmos	Pesos das variáveis			Matriz de Confusão			
	x_1	x_2	x_3	$k=1$	$k=2$	$k=3$	$k=4$
BSSC-LP	0,04291	19,13531	1,21781	0	100	0	0
	1,64701	9,56212	0,06349	0	0	100	0
	1,61972	9,51445	0,06489	0	0	0	100
	0,04371	16,24330	1,408462	100	0	0	0
sSSC-LP	0,01441	8,31481	8,34618	10	0	90	0
	0,01213	10,57005	7,80028	0	10	0	90
	9,43300	7,42002	0,01429	90	0	10	0
	8,45649	8,21508	0,01439	0	90	0	10
ESSC-LP	0,29343	43,48837	0,07836	100	0	0	0
	0,28473	43,46322	0,08081	0	100	0	0
	0,07094	46,66547	0,30206	0	0	100	0
	0,06828	47,19905	0,31030	0	0	0	100
NebFuzz-LP	9,32458	7,51689	0,014267	100	0	0	0
	7,92946	8,58308	0,014693	0	100	0	0
	0,01315	10,32662	7,363344	0	0	0	100
	0,01452	8,48296	8,118049	0	0	100	0
Isodata-PS-LP	9,31517	7,79935	0,01376	100	0	0	0
	7,92914	8,90496	0,01416	0	100	0	0
	0,01396	8,35138	8,57383	0	0	100	0
	0,01334	10,49385	7,13988	0	0	0	100

Tabela 54 – Pesos das bases de dados simulados 2

Algoritmos	Pesos das variáveis			Matriz de Confusão		
	x_1	x_2	x_3	$k=1$	$k=2$	$k=3$
BSSC-GP	0,00139	208,03001	3,44021	22	0	78
				0	100	0
				75	0	25
sSSC-GP	0,00138	552,80399	1,31155	96	0	4
				0	100	0
				9	0	91
ESSC-GP	6,14754E-4	893,85553	1,81983	95	0	5
				0	100	0
				4	0	96
NebFuzz-GP	7,81568E-4	804,27640	1,59084	0	6	94
				100	0	0
				0	94	6
Isodata-PS-GP	6,97090E-4	842,99838	1,70170	96	0	4
				0	100	0
				5	0	95

6.3 SÍNTESE DO CAPÍTULO

Este capítulo apresentou e discutiu resultados dos experimentos realizados com os algoritmos de agrupamento semissupervisionado: BSSC e suas respectivas versões com ponderação local de variáveis, sSSC e suas respectivas versões com ponderação local de variáveis, ESSC e suas respectivas versões com ponderação local de variáveis, NebFuzz e suas respectivas versões com ponderação local de variáveis e Isodata-PS e suas respectivas versões com ponderação local de variáveis. Os experimentos foram executados para avaliar o desempenho desses algoritmos em bases de dados de problemas reais e em dados simulados com diferentes porcentagens de dados rotulados disponíveis no treinamento dos algoritmos.

7 CONCLUSÕES E TRABALHOS FUTUROS

7.1 CONCLUSÃO

O aprendizado semissupervisionado foi criado para atenuar limitações das duas abordagens mais clássicas de aprendizagens, supervisionada e não supervisionada. A aprendizagem supervisionada precisa de dados rotulados suficientes para que o algoritmo possa alcançar um bom desempenho, enquanto a abordagem não supervisionado, geralmente, consegue desempenho pior que os algoritmos supervisionados. A aprendizagem semissupervisionada utiliza uma certa quantidade de dados rotulados e consegue melhorar o desempenho utilizando dados rotulados e não rotulados em seu treinamento.

O cálculo da distância entre dois itens influencia bastante no desempenho dos algoritmos de agrupamento. O uso de distâncias adaptativas permite que os grupos criados se adequem melhor a dados mais complexos. Neste trabalho, desenvolvemos novos algoritmos de agrupamento semissupervisionado que utilizam distâncias adaptativas com ponderação automática de variáveis local (cada grupo possui um conjunto de pesos diferentes que multiplicam suas variáveis) e global (o conjunto de pesos que multiplica as variáveis de todos os grupos é o mesmo), que utilizam a restrição em que o produto dos pesos das variáveis tem que ser igual a um e a restrição em que a soma dos pesos das variáveis tem que ser igual a um. Desenvolvemos quatro versões que combinam essas duas características para cada um dos algoritmos semissupervisionados: Isodata-PS (PEDRYCZ e WALETZKY, 1997), BSSC (BOUCHACHIA e PEDRYCZ, 2006a), sSSC (ENDO et al., 2009), ESSC (ENDO et al., 2009) e NebFuzz (MACARIO e DE CARVALHO, 2010). No total foram 18 algoritmos propostos neste trabalho e mais dois trabalhos que já tinham sido propostos anteriormente na literatura, Isodata-PS-LS (MARCELLONI, 2003; KONG e WANG, 2009) e BSSC-LP (BOUCHACHIA e PEDRYCZ, 2006b), apresentados na Tabela 55.

Tabela 55 – Resumo dos algoritmos de agrupamento semissupervisionado.

Algoritmo	Tipo de Distância Adaptativa			
	Local		Global	
	Soma (LS)	Produto (LP)	Soma (GP)	Produto (GS)
Isodata-PS	x_1	✓	✓	✓
BSSC	✓	x_2	✓	✓
sSSC	✓	✓	✓	✓
ESSC	✓	✓	✓	✓
NebFuzz	✓	✓	✓	✓

Todos os algoritmos neste trabalho são iterativos, ou seja, começam de uma partição *fuzzy* inicial e alternam três etapas até a convergência, quando um critério de parada é

satisfeito. A solução desses algoritmos são os melhores protótipos, partições *fuzzy* e matriz de pesos que minimizam o critério de adequação, ou função objetivo.

Nesta tese foram descritos em detalhes a derivação desses algoritmos. Além disso, provamos a convergência desses algoritmos e foi realizada uma análise de complexidade de tempo desses algoritmos.

Para avaliar o desempenho dos algoritmos, foram realizados experimentos com bases de dados de problemas reais e com dados simulados. A partir do experimento com bases de dados reais pudemos realizar quatro análises. A primeira foi estudar alguns valores para o parâmetro alfa. Selecionado o valor alfa de cada algoritmo, foi realizada uma análise para confirmar que algoritmos com distância adaptativa realmente melhoram o desempenho em relação aos algoritmos com distância Euclidiana. Depois, foi realizado um estudo para comparar o desempenho de todos os algoritmos de agrupamento semissupervisionados considerados nesse trabalho. Por último, utilizamos uma técnica para escolher de forma inteligente os exemplos rotulados dos algoritmos. Com base nos resultados desses experimentos, podemos concluir que:

- Com relação ao parâmetro alfa dos algoritmos com ponderação automática de variáveis baseados no algoritmo BSSC, na maioria dos casos, com poucos dados rotulados (1% até 10%) o valor é abaixo de um, enquanto com mais de 20% de dados rotulados o valor de alfa (α) é igual a um;

- Com relação ao parâmetro alfa dos algoritmos com ponderação automática de variáveis baseados no algoritmo ESSC, na maioria dos casos, possui valores baixos, variando de 0,1 até 1;

- Com relação ao parâmetro alfa dos algoritmos com ponderação automática de variáveis baseados no algoritmo NebFuzz, eles variaram bastante, na maioria dos casos, os valores de alfa estudados são estatisticamente equivalentes;

- Com relação ao parâmetro alfa dos algoritmos com ponderação automática de variáveis baseados no algoritmo Isodata-PS, com poucos dados rotulados (1% até 10%), o alfa possui valores baixos (0,1 e 0,5), com apenas 3 casos com valores maiores que esse valor. E com mais de 20% de dados rotulados, os valores de alfa variaram de 1 até 50;

- Em geral, não existe um valor padrão de alfa para um algoritmo de agrupamento semissupervisionado. O valor de alfa pode ser diretamente afetado caso a quantidade de dados rotulados seja diferente, bem como, também depende do problema ou base de dados considerada;

- A ponderação automática de variáveis se mostrou uma abordagem interessante, pois para os cinco algoritmos que foram modificados para utilizar essa abordagem, pelo

menos uma versão com distância adaptativa obteve desempenho melhor quando comparado com o algoritmo com a distância Euclidiana que o originou;

- Os melhores algoritmos com distância adaptativa baseados no algoritmo BSSC, foram os algoritmos BSSC-LP e BSSC-LS para poucos dados rotulados (1% até 10%) e o algoritmo BSSC-GP para mais de 20% de dados rotulados;

- Para os algoritmos com distância adaptativa baseados no algoritmo sSSC, para menos de 10% de dados rotulados o melhor desempenho foi obtido pelos algoritmos sSSC-GP e sSSC-LP. Para mais que 20% de dados rotulados, o melhor desempenho foi obtido pelos algoritmos sSSC-GS e sSSC-LS;

- Os melhores algoritmos com distância adaptativa baseados no algoritmo ESSC foram os algoritmos ESSC-GP e ESSC-LP;

- O melhor algoritmo com distância adaptativa baseado no algoritmo NebFuzz foi o algoritmo NebFuzz-GP;

- O melhor algoritmo baseado no algoritmo Isodata-PS foi o algoritmo Isodata-PS-GP;

- A versão com ponderação automática de variáveis que sempre esteve entre os melhores algoritmos foi a versão com ponderação global de variáveis em que o produto dos pesos das variáveis tem que ser igual a um;

- Se houver uma comparação entre as versões com restrição do produto dos pesos das variáveis tem que ser igual a um versus a restrição da soma dos pesos das variáveis igual a um, nota-se que a versão com restrição de produto teve mais algoritmos com melhor desempenho;

- O pior desempenho foi obtido pelas versões com ponderação local de variáveis com restrição da soma dos pesos das variáveis igual a um;

- De forma geral, os algoritmos com ponderação automática de variáveis que se destacaram em termos de desempenho foram os algoritmos ESSC-GP e ESSC-LP. O algoritmo Isodata-PS-PG também obteve bons resultados, estatisticamente equivalentes aos dois primeiros algoritmos;

- Outro ponto a se destacar é que todos os algoritmos melhoraram o desempenho de acordo com o aumento de dados rotulados;

- Rotular itens localizados na fronteira da análise dos graus de pertinência do algoritmo *Fuzzy C-Means* aumentou o desempenho de todos os algoritmos na maioria das diferentes quantidade de rótulos avaliadas nesse trabalho.

O experimento na base de dados simulados 1 mostrou que os algoritmos com ponderação local de variáveis se comportam conforme esperado e diferenciam a relevância de cada variável na definição de cada grupo. Os algoritmos com ponderação global de variáveis se mostraram bastante eficientes quando a relevância das variáveis na definição dos grupos era semelhante, como mostrado no experimento com base de dados simulados 2.

Em virtude dos resultados apresentados, podemos afirmar que os algoritmos com ponderação automática de variáveis alcançaram excelentes desempenhos. Principalmente os algoritmos com ponderação global de variáveis em que a restrição do produto dos pesos da variável é igual a um, pois esses foram melhores ou pelo menos estatisticamente equivalentes a outros algoritmos considerados nesse trabalho.

7.2 CONTRIBUIÇÕES

Durante o doutorado foram publicados alguns artigos científicos com os algoritmos desenvolvidos.

O artigo abaixo apresentou o algoritmo Isodata-PS-LP (MACARIO e DE CARVALHO, 2012b):
*MACARIO, VALMIR ; CARVALHO, DE FRANCISCO DE A.T. An adaptive isodata fuzzy clustering algorithm with partial supervision. In: 2012 IEEE International Conference on Systems, Man and Cybernetics SMC, 2012, Seoul. 2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC). p. 1978-1983.

O algoritmo NebFuzz-LP (MACARIO e DE CARVALHO, 2012a) foi apresentado pelo artigo:
*MACARIO, VALMIR ; DE CARVALHO, FRANCISCO DE A.T. An adaptive semi-supervised fuzzy clustering algorithm based on objective function optimization. In: 2012 IEEE International Conference on Fuzzy Systems (FUZZIEEE), 2012, Brisbane. 2012 IEEE International Conference on Fuzzy Systems. p. 2015-2022.

O próximo artigo realizou uma aplicação numa base de dados de expressão gênica com os algoritmos semissupervisionados (MACARIO et al., 2012):
*MACARIO, VALMIR ; COSTA, IVAN G. ; OLIVEIRA, JOAO F.L. ; DE CARVALHO, FRANCISCO DE A.T. . Predicting Gene Functions Using Semi-supervised Clustering Algorithms with Objective Function Optimization. In: 2012 Brazilian Symposium on Neural Networks (SBRN), 2012, Curitiba. 2012 Brazilian Symposium on Neural Networks. p. 61-66.

Também, desenvolvemos um novo algoritmo de agrupamento não supervisionado chamado XPFCM (MACARIO e DE CARVALHO, 2014), no futuro, será desenvolvido uma versão semissupervisionada desse algoritmo:
*MACARIO, VALMIR ; DE CARVALHO, FRANCISCO DE A.T. . An adjustable p-exponential clustering algorithm. In: European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, 2014, Bruges. Proceedings of European Symposium on Artificial Neural Networks, 2014.

7.3 TRABALHOS FUTUROS

Neste trabalho foram utilizadas bases de dados disponíveis em *sites* conhecidos da academia. O objetivo é estender os testes para bases de dados de aplicações reais. Duas aplicações serão testadas prioritariamente: uma de segmentação de imagens e outra de classificação de expressões gênicas, conhecidas por serem bases de dados difíceis de rotular por seres humanos.

Existem alguns algoritmos de agrupamento semissupervisionado que utilizam restrições para otimizar o critério de agrupamento Basu *et al.* (BASU et al., 2006) em seu treinamento. Outra proposta é comparar o desempenho dos novos algoritmos de agrupamento semissupervisionado com algoritmos desse tipo.

Neste trabalho, apresentamos quatro tipos de distâncias adaptativas. Além desta, existem outros tipos de distâncias adaptativas. Como trabalhos futuros, pretendemos utilizar outros tipos de distâncias adaptativas nesses algoritmos.

Planeja-se realizar um estudo do parâmetro alfa utilizando algoritmos evolucionários, que possibilitam utilizar um intervalo maior de possibilidades de valores para o parâmetro alfa.

REFERÊNCIAS

AMINI, M. R.; GALLINARI, P. Semi-supervised learning with an imperfect supervisor. *Knowledge and Information Systems*, v. 8, p. 385–413, 2005.

BASU, S.; BANERJEE, S.; MOONEY, R. Semi-supervised clustering by seeding. *Proceedings of the International Conference on Machine Learning*, p. 19–26, 2002.

BASU, S. et al. Probabilistic semi-supervised clustering with constraints. In: CHAPELLE, O.; SCHOLKOPF, B.; ZIEN, A. (Ed.). *Semi-Supervised Learning*. [S.l.]: Mit Press, 2006. p. 7–39.

BASU, S.; BILENKO, M.; MOONEY, R. Comparing and unifying search-based and similarity-based approaches to semi-supervised clustering. *Proceedings of the International Conference on Machine Learning*, 2003.

BENSAID, A. M. et al. Partially supervised clustering for image segmentation. *Pattern Recognition*, v. 29, n. 5, p. 859–871, 1996.

BEZDEK, J. *Pattern Recognition With Fuzzy Objective Function Algorithms*. [S.l.]: Plenum, 1981.

BOUCHACHIA, A.; PEDRYCZ, W. Data clustering with partial supervision. *Data Mining and Knowledge Discovery*, n. 12, p. 47–78, 2006.

BOUCHACHIA, A.; PEDRYCZ, W. Enhancement of fuzzy clustering by mechanisms of partial supervision. *Fuzzy Sets and Systems*, v. 157, p. 1733–1759, 2006.

BREIMAN, L. et al. *Classification and Regression Trees*. Boca Raton: Chapman and Hall/CRC, 1984.

CHAPELLE, O.; ZIEN, A.; SCHOLKOPF, B. *Semi-supervised learning*. [S.l.]: MIT Press, 2006.

CHENG, V.; LI, C. Personalized spam filtering with semi-supervised classifier ensemble. *International Conference on Web Intelligence*, 2006.

COVER, T. M.; THOMAS, J. A. *Elements of Information Theory*. 2. ed. [S.l.]: Wiley-Interscience, 2006.

DE CARVALHO, F. A. T.; TENORIO, C. P.; CAVALCANTI JUNIOR, N. L. Partitional fuzzy clustering methods based on adaptive quadratic distances. *Fuzzy Sets and Systems*, v. 157, p. 2833–2857, 2006.

DEMPSTER, A.; LAIRD, N.; RUBIN, D. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, v. 39, p. 1–38, 1977.

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

DIDAY, E.; SIMON, J. C. Clustering analysis. In: *Digital Pattern Classificationx*. Berlin: Springer, 1976. p. 47–94.

DUBES, R. How many clusters are best? an experiment. *Pattern Recognition*, v. 20, n. 6, p. 645–663, 1987.

DUDA, R.; HART, P.; STORK, D. *Pattern Classification*. [S.l.]: John Wiley & Sons, 2001.

ENDO, Y. et al. On semi-supervised fuzzy c-means clustering. *Proceedings of the IEEE International Conference on Fuzzy Systems*, p. 1119–1124, 2009.

EVERITT, B. *Cluster Analysis*. [S.l.]: New York: Academic Press, 1980.

FERNANDEZ-DELGADO, M. et al. Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research*, n. 15, p. 3133–3181, 2014.

FERREIRA, M. R.; DE CARVALHO, F. de A. Kernel fuzzy c-means with automatic variable weighting. *Fuzzy Sets and Systems*, v. 237, p. 1–46, 2014.

FRIGUI, H.; HWANG, C. Fuzzy clustering and aggregation of relational data with instance-level constraints. *IEEE Transaction on Fuzzy Systems*, v. 16, n. 6, p. 1565–1581, 2008.

FRIGUI, H.; NASRAOUI, O. Unsupervised learning of prototypes and attribute weights. *Pattern Recognition*, v. 37, p. 567–581, 2004.

GRIRA, N.; CRUCIANU, M.; BOUJEMAA, N. Active semi-supervised fuzzy clustering. *Pattern Recognition*, v. 41, p. 1834 –1844, 2008.

GUSTAFSON, D.; KESSEL, W. Fuzzy clustering with a fuzzy covariance matrix. *Proceedings of the IEEE Conference on Decision and Control*, p. 61–76, 1979.

HATHAWAY, R. J.; BEZDEK, J.; HU, Y. Generalized fuzzy c-means clustering strategies using lp-norm distances. *IEEE Transaction on Fuzzy Systems*, v. 8, n. 5, p. 576–582, 2000.

HOLM, S. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, v. 6, p. 65–70, 1979.

HUBBERT, L. J.; ARABIE, P. Comparing partitions. *Journal of Classification*, v. 2, p. 63–76, 1985.

HUNG, C. C.; KULKARNI, S.; KUO, B. C. A new weighted fuzzy c-means clustering algorithm for remotely sensed image classification. *IEEE Journal of Selected Topics in Signal Processing*, v. 5, n. 3, p. 543–553, 2011.

IMAN, R. L.; DAVENPORT, J. M. Approximations of the critical region of the friedman statistic. *Communications in Statistics*, p. 571–595, 1980.

JAIN, A. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, v. 31, p. 651–666, 2010.

JAIN, A.; MURTY, M.; FLYNN, P. Data clustering: A review. *ACM Computing Surveys*, v. 31, n. 3, p. 264–323, 1999.

JAIN, A. K.; DUBES, R. C. *Algorithms for clustering data*. New Jersey: Prentice Hall, 1988.

KANZAWA, Y.; ENDO, Y.; MIYAMOTOE, S. Kl-divergence-based and manhattan distance-based semisupervised entropy-regularized fuzzy c-means. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, v. 15, n. 8, p. 1057–1064, 2011.

KONG, Y.; WANG, S. Feature selection and semisupervised fuzzy clustering. *Fuzzy Information and Engineering*, v. 1, n. 2, p. 179–190, 2009.

KRISHNAPURAM, R.; KIM, J. A note on the gustafson-kessel and adaptive fuzzy clustering algorithms. *IEEE Transaction on Fuzzy Systems*, v. 7, n. 4, p. 453–461, 1999.

MACARIO, V. et al. Predicting gene functions using semi-supervised clustering algorithms with objective function optimization. *Brazilian Symposium on Neural Networks (SBRN)*, p. 61–66, 2012.

MACARIO, V.; DE CARVALHO, F. A new approach for semi-supervised clustering based on fuzzy c-means. *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ)*, p. 298–305, 2010.

MACARIO, V.; DE CARVALHO, F. An adaptive semi-supervised fuzzy clustering algorithm based on objective function optimization. *Proceedings of the IEEE International Conference on Fuzzy Systems*, p. 2015–2022, 2012.

MACARIO, V.; DE CARVALHO, F. A. T. An adaptive isodata fuzzy clustering algorithm with partial supervision. *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, p. 1978–1983, 2012.

MACARIO, V.; DE CARVALHO, F. A. T. An adjustable p-exponential clustering algorithm. *Proceedings of European Symposium on Artificial Neural Networks*, 2014.

MACARIO, V. et al. Automatic information extraction in semi-structured official journals. *Brazilian Symposium on Neural Networks*, p. 51–56, 2008.

MARAZIOTIS, I. A. A semi-supervised fuzzy clustering algorith mapplied to gene expression data. *Pattern Recognition*, v. 45, p. 637–648, 2012.

MARCELLONI, F. Feature selection based on a modified fuzzy c-means algorithm with supervision. *Information Sciences*, v. 151, 2003.

MEI, J.-P.; CHENVC, L. Linkfcm: Relation integrated fuzzy c-means. *Pattern Recognition*, v. 46, p. 272–283, 2013.

MILLIGAN, G. W.; COOPER, M. C. A study of the comparability of external criteria for hierarchical cluster analysis. *Multivariate Behavorial Research*, v. 21, p. 441–458, 1986.

MITCHELL, T. *Machine Learning*. [S.l.]: McGraw Hill, 1997.

NEMENYI, P. B. *Distribution-free multiple comparisons*. Tese (Doutorado) — Princeton University, 1963.

NIGAM, K. et al. Text classification from labeled and unlabeled documents using em. *Machine Learning*, v. 39, p. 103–134, 2000.

PEDRYCZ, W. Algorithms of fuzzy clustering with partial supervision. *Pattern Recognition*, v. 3, p. 13–20, 1985.

PEDRYCZ, W.; WALETZKY, J. Fuzzy clustering with partial supervision. *IEEE Transactions on System, Man and Cybernetics*, v. 27, n. 5, 1997.

SALTON, G.; BUCKLEY, C. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, v. 5, p. 513–523, 1988.

SHESKIN, D. J. *Handbook of Parametric and Nonparametric Statistical Procedures.* [S.l.]: CRC Press, 2006.

STEPP, R. E.; MICHALSKI, R. S. *Machine Learning: An Artificial Intelligence Approach.* [S.l.]: Morgan Kaufmann, 1986. v. 2. 471-478 p.

STREHL, A.; GHOSH, J. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, n. 3, p. 583–617, 2002.

TANG, C.; WANG, S.; XU, W. New fuzzy c-means clustering model based on the data weighted approach. *Data & Knowledge Engineering*, v. 69, p. 881–900, 2010.

VAPNIK, V. *Statistical learning theory.* [S.l.]: Wiley-Interscience, 1998.

WAGSTAFF, K. et al. Constrained k-means clustering with background knowledge. *Proceedings of the International Conference on Machine Learning*, p. 577–584, 2001.

WANG, Q.; YUEN, P. C.; FENG, G. Semi-supervised metric learning via topology preserving multiple semi-supervised assumptions. *Pattern Recognition*, v. 46, n. 9, p. 2576–2587, 2013.

WILCOXON, F. Individual comparisons by ranking methods. *Biometrics*, n. 1, p. 80–83, 1945.

WU, J. et al. A generalization of distance functions for fuzzy c-means clustering with centroids of arithmetic means. *IEEE Transaction on Fuzzy Systems*, v. 20, p. 557–571, 2012.

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

YIN, X.; SHU, T.; HUANG, Q. Semi-supervised fuzzy clustering with metric learning and entropy regularization. *Knowledge-Based Systems*, v. 35, p. 304–3011, 2012.

ZENG, S. et al. A study on semi-supervised fcm algorithm. *Knowledge and Information Systems*, p. 585–612, 2012.

ZHANG, H. et al. Semi-supervised distance metric learning based on local linear regression for data clustering. *Neurocomputing*, v. 93, p. 100–105, 2012.

ZHU, X. *Semi-Supervised Learning Literature Survey.* [S.l.], 2008.

Apêndices

APÊNDICE A – RESULTADOS

Tabela 56 – Desempenho médio dos algoritmos BSSC e suas respectivas versões com ponderação automática de variáveis com 10% de dados rotulados.

Acurácia (%)					
Base de Dados	BSSC	BSSC-PG	BSSC-PL	BSSC-GS	BSSC-LS
Australian	58,18(3)	56,26(4,5)	58,66(2)	56,26(4,5)	66,11(1)
Balance	53,66(5)	68,02(1,5)	64,13(4)	68,01(3)	68,02(1,5)
CMC	40,79(3)	36,14(5)	44,31(1)	36,16(4)	42,42(2)
Diabetes	65,10(3,5)	63,67(5)	65,63(1)	65,10(3,5)	65,20(2)
Ecoli	62,23(4)	79,54(2)	72,39(3)	79,56(1)	42,59(5)
Glass	53,71(1)	52,80(2)	50,35(5)	52,22(3)	51,01(4)
Haberman	70,73(5)	73,53(2,0)	73,53(2,0)	73,53(2,0)	73,15(4)
Hayes	44,19(1)	41,49(5)	43,21(3)	41,50(4)	43,38(2)
Heart	47,87(2)	42,09(3)	57,77(1)	39,83(4)	39,14(5)
ICU	72,91(3)	50,72(5)	80,00(1,5)	52,23(4)	80,00(1,5)
Iris	89,66(5)	91,98(2)	90,61(4)	92,13(1)	90,77(3)
Libras	46,67(3)	46,62(4)	47,22(1)	46,60(5)	46,89(2)
Mammographic	67,78(5)	78,63(1)	76,56(4)	78,60(2)	78,17(3)
Segmentation	62,12(3)	67,80(2)	23,29(5)	67,88(1)	46,00(4)
Spam	64,45(3)	60,60(5)	76,99(1)	64,40(4)	65,53(2)
Teaching	39,62(5)	43,01(2)	46,32(1)	42,16(3)	41,09(4)
Vehicle	41,22(3)	38,72(4)	41,76(2)	38,67(5)	42,31(1)
Wine	68,69(5)	93,17(1)	92,42(2)	78,03(4)	78,07(3)
Yeast	73,84(3)	75,43(2)	73,10(4,5)	76,14(1)	73,10(4,5)

NMI					
Base de Dados	BSSC	BSSC-PG	BSSC-PG	BSSC-GS	BSSC-LS
Australian	0,08(2,5)	0,04(4,5)	0,08(2,5)	0,04(4,5)	0,16(1)
Balance	0,16(4)	0,17(2,0)	0,14(5)	0,17(2,0)	0,17(2,0)
CMC	0,03(3,5)	0,03(3,5)	0,03(3,5)	0,03(3,5)	0,04(1)
Diabetes	0,02(5)	0,03(3,0)	0,03(3,0)	0,04(1)	0,03(3,0)
Ecoli	0,56(3,0)	0,56(3,0)	0,58(1)	0,56(3,0)	0,15(5)
Glass	0,39(2)	0,36(4,5)	0,36(4,5)	0,38(3)	0,41(1)
Haberman	0,00(3,5)	0,00(3,5)	0,00(3,5)	0,00(3,5)	0,01(1)
Hayes	0,01(2,0)	0,00(4,5)	0,01(2,0)	0,00(4,5)	0,01(2,0)
Heart	0,01(5)	0,02(3,5)	0,02(3,5)	0,03(1,5)	0,03(1,5)
ICU	0,00(4,0)	0,00(4,0)	0,01(2)	0,00(4,0)	0,04(1)
Iris	0,76(5)	0,80(2)	0,78(3,5)	0,81(1)	0,78(3,5)
Libras	0,65(5)	0,66(2,5)	0,66(2,5)	0,66(2,5)	0,66(2,5)
Mammographic	0,11(5)	0,27(2,0)	0,25(4)	0,27(2,0)	0,27(2,0)
Segmentation	0,55(3)	0,63(1)	0,18(5)	0,61(2)	0,47(4)
Spam	0,05(3,5)	0,00(5)	0,25(1)	0,05(3,5)	0,10(2)
Teaching	0,03(4,5)	0,04(3)	0,08(1)	0,03(4,5)	0,07(2)
Vehicle	0,18(2,5)	0,14(4,5)	0,19(1)	0,14(4,5)	0,18(2,5)
Wine	0,43(5)	0,78(1)	0,76(2)	0,50(3,5)	0,50(3,5)
Yeast	0,17(3)	0,19(2)	0,11(5)	0,21(1)	0,15(4)

Tabela 57 – Desempenho médio dos algoritmos BSSC e suas respectivas versões com ponderação automática de variáveis com 50% de dados rotulados.

Acurácia (%)					
Base de Dados	BSSC	BSSC-PG	BSSC-PL	BSSC-GS	BSSC-LS
Australian	77,86(5)	77,88(3,5)	78,04(1,5)	77,88(3,5)	78,04(1,5)
Balance	76,13(4)	76,15(2,0)	62,83(5)	76,15(2,0)	76,15(2,0)
CMC	65,33(4)	65,55(1,5)	45,24(5)	65,55(1,5)	65,46(3)
Diabetes	67,57(1)	67,22(2)	65,79(5)	67,18(3)	67,16(4)
Ecoli	66,84(5)	66,92(3,5)	69,31(2)	66,92(3,5)	71,43(1)
Glass	72,75(5)	74,23(1)	73,98(3)	74,21(2)	73,17(4)
Haberman	75,98(4)	76,06(2,5)	73,53(5)	76,06(2,5)	76,34(1)
Hayes	67,11(3,5)	67,11(3,5)	67,12(1)	67,11(3,5)	67,11(3,5)
Heart	70,54(4)	71,46(1)	58,69(5)	71,35(2)	71,34(3)
ICU	74,39(3,5)	74,39(3,5)	80,00(1)	74,36(5)	75,04(2)
Iris	94,48(5)	94,94(1,5)	94,75(4)	94,94(1,5)	94,83(3)
Libras	58,16(3)	58,22(1,5)	57,73(5)	58,22(1,5)	57,87(4)
Mammographic	83,83(5)	84,54(2,5)	84,68(1)	84,54(2,5)	84,53(4)
Segmentation	75,40(4)	77,50(1)	54,79(5)	76,22(2)	75,70(3)
Spam	68,11(5)	68,27(2)	68,44(1)	68,19(4)	68,23(3)
Teaching	68,47(5)	68,58(3,5)	68,68(2)	68,58(3,5)	69,13(1)
Vehicle	65,50(1)	65,02(4,5)	65,14(3)	65,02(4,5)	65,23(2)
Wine	84,98(3)	85,31(2)	85,44(1)	84,96(5)	84,97(4)
Yeast	87,96(3)	89,34(2)	84,45(5)	89,39(1)	87,30(4)

NMI					
Base de Dados	BSSC	BSSC-PG	BSSC-PG	BSSC-GS	BSSC-LS
Australian	0,37(3,0)	0,37(3,0)	0,37(3,0)	0,37(3,0)	0,37(3,0)
Balance	0,39(2,5)	0,39(2,5)	0,12(5)	0,39(2,5)	0,39(2,5)
CMC	0,19(4)	0,20(2,0)	0,03(5)	0,20(2,0)	0,20(2,0)
Diabetes	0,12(4)	0,13(2,0)	0,03(5)	0,13(2,0)	0,13(2,0)
Ecoli	0,62(3,5)	0,62(3,5)	0,63(1)	0,62(3,5)	0,62(3,5)
Glass	0,54(5)	0,56(1,5)	0,55(3,5)	0,56(1,5)	0,55(3,5)
Haberman	0,17(3,0)	0,17(3,0)	0,00(5)	0,17(3,0)	0,18(1)
Hayes	0,22(3,0)	0,22(3,0)	0,22(3,0)	0,22(3,0)	0,22(3,0)
Heart	0,13(4)	0,15(2,0)	0,02(5)	0,15(2,0)	0,15(2,0)
ICU	0,12(3,0)	0,12(3,0)	0,01(5)	0,12(3,0)	0,13(1)
Iris	0,84(5)	0,85(2,5)	0,85(2,5)	0,85(2,5)	0,85(2,5)
Libras	0,72(3,0)	0,72(3,0)	0,72(3,0)	0,72(3,0)	0,72(3,0)
Mammographic	0,37(5)	0,39(3,0)	0,40(1)	0,39(3,0)	0,39(3,0)
Segmentation	0,66(4)	0,69(1)	0,44(5)	0,68(2)	0,67(3)
Spam	0,19(5)	0,20(3,0)	0,21(1)	0,20(3,0)	0,20(3,0)
Teaching	0,27(3,0)	0,27(3,0)	0,27(3,0)	0,27(3,0)	0,27(3,0)
Vehicle	0,36(1)	0,35(3,5)	0,35(3,5)	0,35(3,5)	0,35(3,5)
Wine	0,61(4,0)	0,62(1,5)	0,62(1,5)	0,61(4,0)	0,61(4,0)
Yeast	0,45(3)	0,48(1,5)	0,40(5)	0,48(1,5)	0,44(4)

Tabela 58 – Desempenho médio dos algoritmos sSSC e suas respectivas versões com ponderação automática de variáveis com 10% de dados rotulados

Acurácia (%)					
Base de Dados	BSSC	BSSC-PG	BSSC-PL	BSSC-GS	BSSC-LS
Australian	57,94(5)	74,63(3)	71,06(4)	82,22(1)	80,42(2)
Balance	39,34(5)	62,40(1,5)	62,40(1,5)	46,15(4)	46,83(3)
CMC	40,07(5)	44,21(1,5)	44,21(1,5)	40,29(4)	40,30(3)
Diabetes	57,08(3)	65,62(1,5)	65,62(1,5)	56,77(4)	56,66(5)
Ecoli	36,15(3)	36,50(2)	31,27(5)	36,81(1)	33,51(4)
Glass	28,64(5)	29,72(4)	32,23(1)	29,94(3)	30,23(2)
Haberman	55,05(4)	73,31(1,5)	73,31(1,5)	54,84(5)	55,50(3)
Hayes	44,83(5)	46,14(3)	45,53(4)	51,19(2)	52,24(1)
Heart	55,09(4)	56,42(1,5)	56,42(1,5)	55,10(3)	54,99(5)
ICU	55,10(5)	80,00(1,5)	80,00(1,5)	55,35(4)	55,99(3)
Iris	37,86(2)	37,21(5)	37,77(3)	37,35(4)	38,41(1)
Libras	46,04(5)	46,60(4)	47,73(1)	47,56(2)	47,40(3)
Mammographic	65,13(4)	74,77(1)	69,93(3)	73,91(2)	54,84(5)
Segmentation	57,16(2)	61,34(1)	27,29(3)	23,63(4)	22,86(5)
Spam	63,79(3)	72,97(1)	61,64(5)	71,08(2)	63,06(4)
Teaching	40,30(5)	47,87(1)	42,68(4)	45,67(2)	44,80(3)
Vehicle	32,50(3)	31,78(5)	32,94(1)	32,32(4)	32,81(2)
Wine	36,11(3)	35,63(5)	35,82(4)	38,92(2)	39,20(1)
Yeast	72,31(3,0)	72,31(3,0)	72,31(3,0)	72,31(3,0)	72,31(3,0)

NMI					
Base de Dados	BSSC	BSSC-PG	BSSC-PG	BSSC-GS	BSSC-LS
Australian	0,08(5)	0,26(3)	0,22(4)	0,36(1)	0,33(2)
Balance	0,10(3,5)	0,13(1,5)	0,13(1,5)	0,09(5)	0,10(3,5)
CMC	0,04(1,5)	0,03(4,0)	0,03(4,0)	0,03(4,0)	0,04(1,5)
Diabetes	0,04(1)	0,03(3,5)	0,03(3,5)	0,03(3,5)	0,03(3,5)
Ecoli	0,50(2,5)	0,51(1)	0,21(4)	0,50(2,5)	0,18(5)
Glass	0,34(1)	0,30(2)	0,26(3)	0,16(4,5)	0,16(4,5)
Haberman	0,01(2,0)	0,00(4,5)	0,00(4,5)	0,01(2,0)	0,01(2,0)
Hayes	0,02(4,5)	0,04(3)	0,02(4,5)	0,06(2)	0,07(1)
Heart	0,03(1)	0,02(3,5)	0,02(3,5)	0,02(3,5)	0,02(3,5)
ICU	0,01(4,0)	0,01(4,0)	0,01(4,0)	0,04(2)	0,05(1)
Iris	0,60(5)	0,67(3)	0,65(4)	0,69(1)	0,68(2)
Libras	0,65(5)	0,66(3,5)	0,67(1,5)	0,66(3,5)	0,67(1,5)
Mammographic	0,08(4)	0,21(1)	0,16(3)	0,19(2)	0,03(5)
Segmentation	0,47(2)	0,55(1)	0,24(3)	0,18(4,5)	0,18(4,5)
Spam	0,05(5)	0,17(1)	0,07(4)	0,14(2)	0,09(3)
Teaching	0,02(5)	0,07(4)	0,08(2,0)	0,08(2,0)	0,08(2,0)
Vehicle	0,16(2)	0,09(5)	0,17(1)	0,14(4)	0,15(3)
Wine	0,34(3)	0,64(1)	0,61(2)	0,29(4)	0,27(5)
Yeast	0,01(3,0)	0,01(3,0)	0,01(3,0)	0,01(3,0)	0,01(3,0)

Tabela 59 – Desempenho médio dos algoritmos sSSC e respectivas versões com ponderação automática de variáveis com 50% de dados rotulados

Acurácia (%)					
Base de Dados	BSSC	BSSC-PG	BSSC-PL	BSSC-GS	BSSC-LS
Australian	0,08(5)	0,26(3)	0,22(4)	0,36(1)	0,33(2)
Balance	0,10(3,5)	0,13(1,5)	0,13(1,5)	0,09(5)	0,10(3,5)
CMC	0,04(1,5)	0,03(4,0)	0,03(4,0)	0,03(4,0)	0,04(1,5)
Diabetes	0,04(1)	0,03(3,5)	0,03(3,5)	0,03(3,5)	0,03(3,5)
Ecoli	0,50(2,5)	0,51(1)	0,21(4)	0,50(2,5)	0,18(5)
Glass	0,34(1)	0,30(2)	0,26(3)	0,16(4,5)	0,16(4,5)
Haberman	0,01(2,0)	0,00(4,5)	0,00(4,5)	0,01(2,0)	0,01(2,0)
Hayes	0,02(4,5)	0,04(3)	0,02(4,5)	0,06(2)	0,07(1)
Heart	0,03(1)	0,02(3,5)	0,02(3,5)	0,02(3,5)	0,02(3,5)
ICU	0,01(4,0)	0,01(4,0)	0,01(4,0)	0,04(2)	0,05(1)
Iris	0,60(5)	0,67(3)	0,65(4)	0,69(1)	0,68(2)
Libras	0,65(5)	0,66(3,5)	0,67(1,5)	0,66(3,5)	0,67(1,5)
Mammographic	0,08(4)	0,21(1)	0,16(3)	0,19(2)	0,03(5)
Segmentation	0,47(2)	0,55(1)	0,24(3)	0,18(4,5)	0,18(4,5)
Spam	0,05(5)	0,17(1)	0,07(4)	0,14(2)	0,09(3)
Teaching	0,02(5)	0,07(4)	0,08(2,0)	0,08(2,0)	0,08(2,0)
Vehicle	0,16(2)	0,09(5)	0,17(1)	0,14(4)	0,15(3)
Wine	0,34(3)	0,64(1)	0,61(2)	0,29(4)	0,27(5)
Yeast	0,01(3,0)	0,01(3,0)	0,01(3,0)	0,01(3,0)	0,01(3,0)

NMI					
Base de Dados	BSSC	BSSC-PG	BSSC-PG	BSSC-GS	BSSC-LS
Australian	0,33(5)	0,57(2,5)	0,53(4)	0,58(1)	0,57(2,5)
Balance	0,29(2)	0,12(4,5)	0,12(4,5)	0,28(3)	0,30(1)
CMC	0,22(3)	0,03(4,5)	0,03(4,5)	0,27(2)	0,34(1)
Diabetes	0,19(1)	0,03(4,5)	0,03(4,5)	0,17(2,5)	0,17(2,5)
Ecoli	0,59(2)	0,60(1)	0,57(3)	0,53(4)	0,50(5)
Glass	0,42(3)	0,43(2)	0,47(1)	0,36(5)	0,37(4)
Haberman	0,17(2,0)	0,00(4,5)	0,00(4,5)	0,17(2,0)	0,17(2,0)
Hayes	0,24(2,5)	0,24(2,5)	0,22(5)	0,24(2,5)	0,24(2,5)
Heart	0,20(2,0)	0,02(4,5)	0,02(4,5)	0,20(2,0)	0,20(2,0)
ICU	0,16(3)	0,01(4,5)	0,01(4,5)	0,19(2)	0,21(1)
Iris	0,47(5)	0,51(1,5)	0,49(3,5)	0,51(1,5)	0,49(3,5)
Libras	0,62(4,5)	0,62(4,5)	0,64(1,5)	0,63(3)	0,64(1,5)
Mammographic	0,17(5)	0,19(3)	0,22(2)	0,18(4)	0,27(1)
Segmentation	0,53(2,5)	0,59(1)	0,53(2,5)	0,50(5)	0,51(4)
Spam	0,29(5)	0,51(1)	0,49(2,5)	0,49(2,5)	0,43(4)
Teaching	0,23(5)	0,25(4)	0,32(2)	0,31(3)	0,34(1)
Vehicle	0,33(2,0)	0,29(5)	0,33(2,0)	0,33(2,0)	0,32(4)
Wine	0,39(4,5)	0,47(1,5)	0,47(1,5)	0,40(3)	0,39(4,5)
Yeast	0,17(3,5)	0,18(1)	0,17(3,5)	0,17(3,5)	0,17(3,5)

Tabela 60 – Desempenho médio dos algoritmos ESSC e suas respectivas versões com ponderação automática de variáveis com 10% de dados rotulados.

ERC					
Base de Dados	ESSC	ESSC-PG	ESSC-PL	ESSC-GS	ESSC-LS
Australian	68,43(5)	86,16(1)	83,16(3)	83,90(2)	76,58(4)
Balance	82,76(1)	82,60(4)	82,63(2)	82,61(3)	81,41(5)
CMC	46,23(3)	50,39(1)	49,74(2)	45,98(4)	42,49(5)
Diabetes	66,57(3)	74,22(1)	73,53(2)	63,25(4)	63,16(5)
Ecoli	79,27(2)	81,21(1)	47,62(5)	78,73(3)	74,26(4)
Glass	51,79(1,5)	51,13(3)	51,79(1,5)	29,25(4)	29,24(5)
Haberman	61,76(3)	63,50(2)	67,13(1)	59,21(5)	61,49(4)
Hayes	52,49(3)	52,21(4)	53,44(1)	51,99(5)	53,41(2)
Heart	66,77(5)	83,23(1)	82,53(2)	77,48(3)	74,23(4)
ICU	56,99(5)	63,77(4)	68,92(1)	66,13(3)	67,23(2)
Iris	74,81(5)	81,49(2)	83,18(1)	81,34(3)	80,56(4)
Libras	13,40(5)	29,04(4)	37,80(1)	36,69(2)	34,93(3)
Mammographic	70,38(5)	81,59(1)	79,91(2)	78,45(3)	74,22(4)
Segmentation	58,29(1)	57,11(2)	39,50(3)	22,86(4,5)	22,86(4,5)
Spam	71,19(4)	89,80(1)	84,88(3)	86,10(2)	68,84(5)
Teaching	41,99(5)	49,62(1)	48,54(3)	48,81(2)	46,86(4)
Vehicle	44,25(5)	47,73(3)	51,34(1)	47,10(4)	49,61(2)
Wine	72,42(3)	92,31(1)	91,32(2)	67,69(4)	66,57(5)
Yeast	74,05(2)	76,27(1)	72,89(4)	73,64(3)	72,40(5)

NMI					
Base de Dados	ESSC	ESSC-PG	ESSC-PG	ESSC-GS	ESSC-LS
Australian	0,15(5)	0,42(1)	0,36(3)	0,38(2)	0,26(4)
Balance	0,49(3,0)	0,49(3,0)	0,50(1)	0,49(3,0)	0,48(5)
CMC	0,06(4,0)	0,08(1)	0,07(2)	0,06(4,0)	0,06(4,0)
Diabetes	0,07(3)	0,18(1)	0,16(2)	0,04(4,5)	0,04(4,5)
Ecoli	0,62(2)	0,65(1)	0,17(5)	0,61(3)	0,57(4)
Glass	0,39(1)	0,34(2,5)	0,34(2,5)	0,10(4,5)	0,10(4,5)
Haberman	0,04(3)	0,05(2)	0,06(1)	0,03(4,5)	0,03(4,5)
Hayes	0,13(3,0)	0,13(3,0)	0,14(1)	0,12(5)	0,13(3,0)
Heart	0,09(5)	0,35(1)	0,33(2)	0,23(3)	0,19(4)
ICU	0,03(5)	0,06(3,0)	0,07(1)	0,06(3,0)	0,06(3,0)
Iris	0,61(4)	0,68(2)	0,69(1)	0,63(3)	0,58(5)
Libras	0,13(5)	0,41(4)	0,59(1)	0,54(2,5)	0,54(2,5)
Mammographic	0,13(5)	0,33(1)	0,29(2)	0,26(3)	0,22(4)
Segmentation	0,50(2)	0,55(1)	0,35(3)	0,18(4,5)	0,18(4,5)
Spam	0,11(5)	0,52(1)	0,42(2)	0,40(3)	0,20(4)
Teaching	0,03(5)	0,08(3)	0,07(4)	0,12(1)	0,10(2)
Vehicle	0,21(2,5)	0,19(4,5)	0,22(1)	0,19(4,5)	0,21(2,5)
Wine	0,48(3)	0,78(1)	0,75(2)	0,34(4)	0,33(5)
Yeast	0,17(2)	0,20(1)	0,13(4)	0,16(3)	0,10(5)

Tabela 61 – Desempenho médio dos algoritmos ESSC e suas respectivas versões com ponderação automática de variáveis com 50% de dados rotulados.

Acurácia (%)					
Base de Dados	ESSC	ESSC-PG	ESSC-PL	ESSC-GS	ESSC-LS
Australian	82,44(5)	93,09(1)	91,73(3)	92,94(2)	89,30(4)
Balance	91,69(1)	90,96(4)	91,61(2)	80,42(5)	91,44(3)
CMC	69,66(4)	73,10(1)	72,75(2)	70,76(3)	68,50(5)
Diabetes	81,37(3)	86,12(2)	86,37(1)	80,16(5)	80,19(4)
Ecoli	90,08(1)	90,00(2)	79,68(5)	89,02(3)	87,95(4)
Glass	74,13(3)	74,57(2)	77,02(1)	59,01(4)	58,04(5)
Haberman	82,78(4)	84,58(2)	86,52(1)	80,22(5)	83,88(3)
Hayes	75,36(4)	75,43(3)	77,00(2)	74,71(5)	77,92(1)
Heart	82,60(5)	91,75(2)	92,30(1)	88,86(3)	88,61(4)
ICU	78,64(5)	84,09(4)	86,34(3)	86,59(2)	87,56(1)
Iris	91,29(4)	93,09(1,5)	93,09(1,5)	92,00(3)	91,10(5)
Libras	46,98(5)	47,84(4)	48,96(3)	50,09(1)	49,67(2)
Mammographic	83,53(5)	89,78(1)	89,37(2)	88,42(4)	88,52(3)
Segmentation	83,88(2)	86,30(1)	73,48(3)	57,14(4,5)	57,14(4,5)
Spam	84,02(4)	94,74(1)	92,10(3)	92,58(2)	81,49(5)
Teaching	66,92(5)	75,68(1)	74,52(2)	72,51(3)	70,10(4)
Vehicle	69,70(5)	70,84(3)	72,72(1)	70,43(4)	72,26(2)
Wine	84,27(3)	97,22(1)	96,47(2)	84,24(4)	83,22(5)
Yeast	88,67(2)	90,15(1)	85,64(4)	88,43(3)	84,98(5)

NMI					
Base de Dados	ESSC	ESSC-PG	ESSC-PG	ESSC-GS	ESSC-LS
Australian	0,39(5)	0,64(1,5)	0,60(3)	0,64(1,5)	0,53(4)
Balance	0,72(3)	0,71(4)	0,73(1,5)	0,49(5)	0,73(1,5)
CMC	0,31(3,0)	0,32(1)	0,31(3,0)	0,30(5)	0,31(3,0)
Diabetes	0,28(3)	0,41(2)	0,42(1)	0,25(4,5)	0,25(4,5)
Ecoli	0,78(1)	0,77(2)	0,62(5)	0,74(4)	0,75(3)
Glass	0,57(2)	0,56(3)	0,58(1)	0,36(4)	0,35(5)
Haberman	0,29(3,5)	0,32(2)	0,35(1)	0,25(5)	0,29(3,5)
Hayes	0,39(3,5)	0,39(3,5)	0,42(2)	0,35(5)	0,43(1)
Heart	0,33(5)	0,59(2)	0,61(1)	0,50(3)	0,49(4)
ICU	0,22(5)	0,31(4)	0,32(3)	0,33(1,5)	0,33(1,5)
Iris	0,78(3)	0,82(1,5)	0,82(1,5)	0,77(4)	0,75(5)
Libras	0,65(4,5)	0,65(4,5)	0,66(2,5)	0,67(1)	0,66(2,5)
Mammographic	0,36(5)	0,54(1)	0,52(2)	0,49(4)	0,50(3)
Segmentation	0,75(2)	0,79(1)	0,66(3)	0,51(4,5)	0,51(4,5)
Spam	0,36(5)	0,70(1)	0,62(2)	0,61(3)	0,42(4)
Teaching	0,24(5)	0,36(3)	0,35(4)	0,40(1)	0,37(2)
Vehicle	0,43(3)	0,41(4,5)	0,45(2)	0,41(4,5)	0,46(1)
Wine	0,58(3,5)	0,90(1)	0,87(2)	0,58(3,5)	0,57(5)
Yeast	0,46(2)	0,50(1)	0,42(4)	0,45(3)	0,38(5)

Tabela 62 – Desempenho médio dos algoritmos NebFuzz e suas respectivas versões com ponderação automática de variáveis com 10% de dados rotulados.

Acurácia (%)					
Base de Dados	NebFuzz	NebFuzz-PG	NebFuzz-PL	NebFuzz-GS	NebFuzz-LS
Australian	56,76(3)	56,04(5)	62,20(1)	57,74(2)	56,39(4)
Balance	85,84(2)	86,82(1)	62,26(3)	60,35(4)	48,11(5)
CMC	42,82(5)	45,22(2)	45,54(1)	43,56(4)	43,96(3)
Diabetes	66,03(2)	73,15(1)	65,64(3)	62,66(4,5)	62,66(4,5)
Ecoli	76,51(2)	80,35(1)	48,44(4)	49,13(3)	36,02(5)
Glass	46,21(1)	45,67(2)	38,75(4)	40,89(3)	36,32(5)
Haberman	64,17(4)	61,71(5)	73,53(1)	67,51(2)	67,45(3)
Hayes	45,16(4)	45,02(5)	45,47(3)	46,74(2)	49,39(1)
Heart	57,53(4)	78,80(1)	57,19(5)	65,48(2)	63,86(3)
ICU	67,47(5)	70,33(3)	80,00(1)	73,36(2)	69,89(4)
Iris	90,19(5)	95,73(3)	94,15(4)	95,96(2)	96,29(1)
Libras	46,60(5)	46,80(4)	47,71(2)	47,62(3)	47,93(1)
Mammographic	70,52(4)	80,50(1)	79,14(2)	78,69(3)	54,29(5)
Segmentation	62,64(2)	66,81(1)	25,78(3)	15,71(4)	14,88(5)
Spam	64,63(4)	80,15(1)	71,54(2)	70,10(3)	62,02(5)
Teaching	38,66(5)	53,04(1)	45,75(3)	46,59(2)	44,30(4)
Vehicle	34,24(2)	27,81(5)	34,74(1)	33,58(3)	31,54(4)
Wine	64,89(4)	93,65(1)	79,56(2)	65,62(3)	62,90(5)
Yeast	72,31(3,5)	72,31(3,5)	72,34(1)	72,31(3,5)	72,31(3,5)

NMI					
Base de Dados	NebFuzz	NebFuzz-PG	NebFuzz-PG	NebFuzz-GS	NebFuzz-LS
Australian	0,04(2)	0,03(3,5)	0,10(1)	0,03(3,5)	0,01(5)
Balance	0,55(2)	0,56(1)	0,40(3)	0,35(4)	0,24(5)
CMC	0,03(4)	0,06(1)	0,05(2,5)	0,01(5)	0,05(2,5)
Diabetes	0,04(2)	0,16(1)	0,03(3)	0,02(4,5)	0,02(4,5)
Ecoli	0,58(2)	0,59(1)	0,39(3)	0,35(4)	0,18(5)
Glass	0,38(1)	0,33(2)	0,32(3)	0,25(4)	0,18(5)
Haberman	0,00(3,0)	0,00(3,0)	0,00(3,0)	0,00(3,0)	0,00(3,0)
Hayes	0,03(4,5)	0,04(2,5)	0,03(4,5)	0,04(2,5)	0,05(1)
Heart	0,03(4)	0,29(1)	0,02(5)	0,11(2)	0,09(3)
ICU	0,01(4,5)	0,04(3)	0,01(4,5)	0,05(1,5)	0,05(1,5)
Iris	0,77(5)	0,87(2,5)	0,83(4)	0,88(1)	0,87(2,5)
Libras	0,66(3,0)	0,65(5)	0,66(3,0)	0,66(3,0)	0,67(1)
Mammographic	0,13(4)	0,31(1)	0,27(2)	0,26(3)	0,03(5)
Segmentation	0,55(2)	0,65(1)	0,20(3)	0,03(4,5)	0,03(4,5)
Spam	0,05(4)	0,26(1)	0,15(2,5)	0,15(2,5)	0,04(5)
Teaching	0,01(5)	0,10(1)	0,08(4)	0,09(2,5)	0,09(2,5)
Vehicle	0,19(1)	0,10(5)	0,16(3,0)	0,16(3,0)	0,16(3,0)
Wine	0,44(4)	0,81(1)	0,72(2)	0,50(3)	0,38(5)
Yeast	0,02(2,0)	0,02(2,0)	0,02(2,0)	0,01(4,5)	0,01(4,5)

Tabela 63 – Desempenho médio dos algoritmos NebFuzz e suas respectivas versões com ponderação automática de variáveis com 50% de dados rotulados.

Acurácia (%)					
Base de Dados	NebFuzz	NebFuzz-PG	NebFuzz-PL	NebFuzz-GS	NebFuzz-LS
Australian	62,33(3)	67,07(1)	65,94(2)	61,17(4)	59,79(5)
Balance	92,48(1)	92,31(2)	84,78(4)	83,61(5)	84,97(3)
CMC	49,01(4)	63,80(1)	45,35(5)	58,61(2)	56,42(3)
Diabetes	79,02(2)	85,15(1)	65,76(3)	63,67(5)	63,69(4)
Ecoli	86,92(2)	87,23(1)	71,02(3)	70,80(4)	48,91(5)
Glass	65,43(1)	64,78(2)	62,09(3)	57,20(5)	57,29(4)
Haberman	73,40(5)	78,26(1)	73,53(4)	73,54(3)	73,68(2)
Hayes	64,62(5)	64,89(4)	66,64(2)	66,72(1)	65,64(3)
Heart	79,26(4)	86,62(1)	58,46(5)	81,22(3)	81,43(2)
ICU	77,07(5)	80,67(1)	80,00(4)	80,64(3)	80,65(2)
Iris	95,30(5)	97,72(2)	97,55(4)	97,71(3)	97,91(1)
Libras	45,84(5)	46,49(4)	47,73(2)	47,58(3)	47,89(1)
Mammographic	83,61(4)	89,53(1)	88,42(2)	88,38(3)	78,18(5)
Segmentation	69,71(2)	74,92(1)	47,38(3)	18,41(5)	19,97(4)
Spam	69,09(4)	72,50(1)	66,90(5)	70,27(2)	70,18(3)
Teaching	46,62(5)	49,72(2)	52,54(1)	48,72(3)	47,61(4)
Vehicle	51,08(1)	47,24(4)	48,00(3)	47,11(5)	48,78(2)
Wine	85,20(4)	97,02(1)	95,15(2)	85,97(3)	83,84(5)
Yeast	77,51(2)	78,34(1)	77,08(3)	75,11(4)	74,88(5)

NMI					
Base de Dados	NebFuzz	NebFuzz-PG	NebFuzz-PG	NebFuzz-GS	NebFuzz-LS
Australian	0,11(2,5)	0,12(1)	0,11(2,5)	0,06(4)	0,05(5)
Balance	0,75(1,5)	0,75(1,5)	0,69(4)	0,67(5)	0,70(3)
CMC	0,07(4)	0,24(2)	0,03(5)	0,19(3)	0,25(1)
Diabetes	0,23(2)	0,39(1)	0,03(3)	0,01(4,5)	0,01(4,5)
Ecoli	0,71(1,5)	0,71(1,5)	0,59(3,5)	0,59(3,5)	0,27(5)
Glass	0,49(2,5)	0,49(2,5)	0,50(1)	0,32(4)	0,31(5)
Haberman	0,06(2)	0,22(1)	0,00(5)	0,01(4)	0,02(3)
Hayes	0,32(3)	0,31(4)	0,35(1)	0,30(5)	0,33(2)
Heart	0,27(4)	0,48(1)	0,02(5)	0,35(3)	0,36(2)
ICU	0,15(1)	0,09(2)	0,01(5)	0,08(3,5)	0,08(3,5)
Iris	0,85(5)	0,92(1,5)	0,91(3,5)	0,91(3,5)	0,92(1,5)
Libras	0,65(5)	0,66(3,5)	0,67(1,5)	0,66(3,5)	0,67(1,5)
Mammographic	0,36(4)	0,53(1)	0,50(2)	0,49(3)	0,33(5)
Segmentation	0,63(2)	0,72(1)	0,45(3)	0,07(5)	0,14(4)
Spam	0,12(3,5)	0,15(1)	0,06(5)	0,13(2)	0,12(3,5)
Teaching	0,09(5)	0,14(2,5)	0,18(1)	0,14(2,5)	0,13(4)
Vehicle	0,33(2)	0,29(5)	0,30(3,5)	0,30(3,5)	0,34(1)
Wine	0,61(4,5)	0,89(1)	0,86(2)	0,64(3)	0,61(4,5)
Yeast	0,18(2)	0,20(1)	0,17(3)	0,13(4,5)	0,13(4,5)

Tabela 64 – Desempenho médio dos algoritmos Isodata-PS e suas respectivas versões com ponderação automática de variáveis com 10% de dados rotulados.

Acurácia (%)					
Base de Dados	Isodata-PS	Isodata-PS-PG	Isodata-PS-PL	Isodata-PS-GS	Isodata-PS-LS
Australian	63,86(5)	78,53(2)	76,77(3)	70,49(4)	79,73(1)
Balance	72,33(1)	69,60(2)	62,59(4)	63,42(3)	48,08(5)
CMC	43,21(4)	48,83(1)	44,89(2)	41,78(5)	44,79(3)
Diabetes	64,15(4)	73,43(1)	65,70(2)	64,20(3)	55,56(5)
Ecoli	80,23(1)	67,53(3)	32,00(5)	74,98(2)	34,73(4)
Glass	50,27(1)	40,68(3)	30,73(4)	45,13(2)	30,15(5)
Haberman	71,31(3)	64,44(4)	73,53(1,5)	73,53(1,5)	55,49(5)
Hayes	54,06(1)	48,23(4)	45,32(5)	49,42(3)	52,83(2)
Heart	63,65(2)	81,56(1)	57,42(3)	53,10(5)	54,10(4)
ICU	76,80(2)	57,05(4)	80,00(1)	62,30(3)	47,20(5)
Iris	91,56(2)	90,90(3)	42,78(4)	95,75(1)	41,67(5)
Libras	46,18(4)	46,80(3)	48,40(1)	47,71(2)	46,07(5)
Mammographic	67,29(4)	78,05(1)	72,25(3)	76,95(2)	54,92(5)
Segmentation	64,54(1)	62,12(2)	30,06(3)	16,51(5)	22,86(4)
Spam	67,44(3)	69,57(2)	61,78(5)	71,99(1)	63,16(4)
Teaching	40,41(5)	50,68(1)	43,38(4)	47,02(2)	44,11(3)
Vehicle	39,07(2)	33,80(3)	33,70(4)	43,29(1)	32,04(5)
Wine	68,56(1)	44,56(3)	41,93(5)	60,65(2)	43,09(4)
Yeast	76,29(1)	75,65(2)	72,71(4)	73,55(3)	72,32(5)

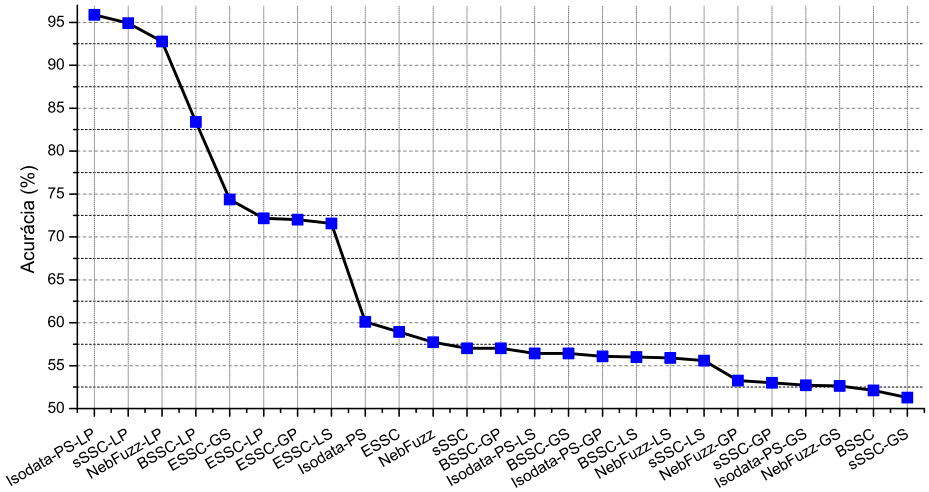
NMI					
Base de Dados	Isodata-PS	Isodata-PS-PG	Isodata-PS-PG	Isodata-PS-GS	Isodata-PS-LS
Australian	0,09(5)	0,28(2)	0,26(3)	0,18(4)	0,32(1)
Balance	0,38(1)	0,37(2)	0,27(3)	0,09(5)	0,10(4)
CMC	0,03(4)	0,07(1)	0,05(2)	0,01(5)	0,04(3)
Diabetes	0,02(4,0)	0,17(1)	0,03(2)	0,02(4,0)	0,02(4,0)
Ecoli	0,58(1,5)	0,58(1,5)	0,23(4)	0,55(3)	0,19(5)
Glass	0,35(1)	0,31(2)	0,26(3)	0,16(5)	0,17(4)
Haberman	0,02(1)	0,00(4,0)	0,00(4,0)	0,00(4,0)	0,01(2)
Hayes	0,14(1)	0,07(3)	0,02(5)	0,05(4)	0,08(2)
Heart	0,06(2)	0,33(1)	0,02(3)	0,01(4,5)	0,01(4,5)
ICU	0,03(3,5)	0,04(1,5)	0,01(5)	0,04(1,5)	0,03(3,5)
Iris	0,79(2)	0,78(3)	0,65(5)	0,86(1)	0,68(4)
Libras	0,66(3,0)	0,65(5)	0,67(1)	0,66(3,0)	0,66(3,0)
Mammographic	0,09(4)	0,26(1)	0,19(3)	0,23(2)	0,03(5)
Segmentation	0,56(2)	0,58(1)	0,24(3)	0,05(5)	0,18(4)
Spam	0,07(5)	0,15(2)	0,10(3,5)	0,16(1)	0,10(3,5)
Teaching	0,02(5)	0,08(3,0)	0,08(3,0)	0,08(3,0)	0,09(1)
Vehicle	0,14(4)	0,11(5)	0,17(1,5)	0,17(1,5)	0,15(3)
Wine	0,43(3)	0,69(1)	0,63(2)	0,36(4)	0,27(5)
Yeast	0,19(1,5)	0,19(1,5)	0,04(5)	0,16(3)	0,05(4)

Tabela 65 – Desempenho médio dos algoritmos Isodata-PS e suas respectivas versões com ponderação automática de variáveis com 50% de dados rotulados.

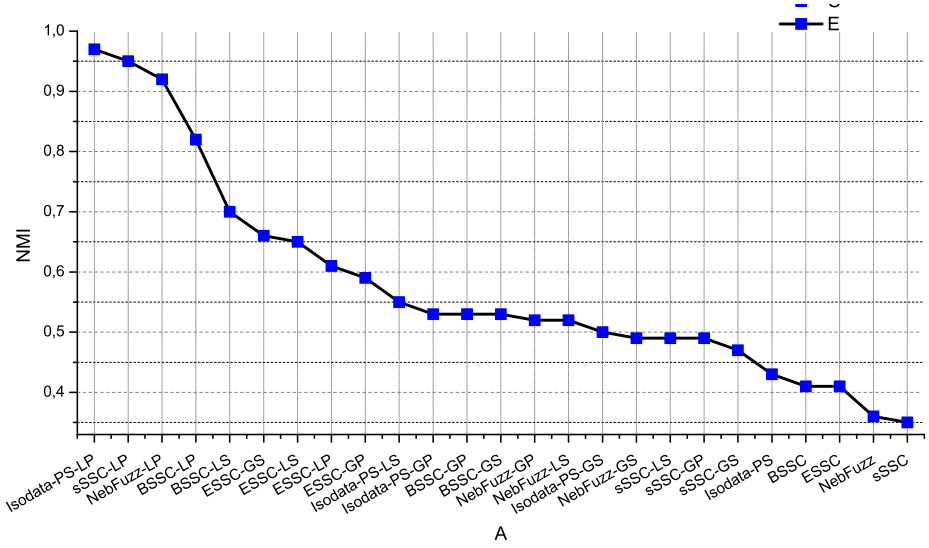
Acurácia (%)					
Base de Dados	Isodata-PS	Isodata-PS-PG	Isodata-PS-PL	Isodata-PS-GS	Isodata-PS-LS
Australian	81,61(5)	92,69(1)	88,55(4)	90,42(2)	90,07(3)
Balance	86,41(2)	86,85(1)	81,53(3)	73,24(5)	73,57(4)
CMC	67,93(3)	72,65(1)	69,18(2)	67,52(4)	66,22(5)
Diabetes	81,14(2)	85,74(1)	65,83(5)	75,59(4)	75,60(3)
Ecoli	83,35(2)	80,74(3)	68,77(4)	83,43(1)	67,40(5)
Glass	73,35(1)	69,55(2)	58,00(4)	57,89(5)	58,13(3)
Haberman	78,28(1)	76,88(2)	73,53(5)	75,11(4)	75,86(3)
Hayes	77,04(1)	75,17(2)	69,58(3)	68,86(4)	67,64(5)
Heart	81,13(2)	91,20(1)	58,66(5)	78,64(3)	76,22(4)
ICU	79,40(3)	82,77(1)	80,00(2)	74,52(4)	69,30(5)
Iris	95,45(2)	97,65(1)	65,69(4)	65,55(5)	71,22(3)
Libras	46,60(3)	48,87(1)	43,98(4)	47,16(2)	41,64(5)
Mammographic	83,28(2)	89,34(1)	72,35(4)	71,61(5)	74,76(3)
Segmentation	83,36(2)	86,99(1)	64,19(3)	57,18(4)	57,14(5)
Spam	83,59(3)	94,20(1)	79,49(5)	88,76(2)	82,87(4)
Teaching	66,44(5)	76,87(1)	66,68(4)	66,86(3)	67,73(2)
Vehicle	69,11(2)	69,88(1)	64,75(3)	63,73(4)	63,70(5)
Wine	85,94(2)	97,46(1)	69,30(5)	70,88(4)	71,60(3)
Yeast	89,06(1)	88,56(2)	84,22(4)	87,14(3)	80,56(5)

NMI					
Base de Dados	Isodata-PS	Isodata-PS-PG	Isodata-PS-PG	Isodata-PS-GS	Isodata-PS-LS
Australian	0,39(5)	0,62(1)	0,51(4)	0,57(2)	0,56(3)
Balance	0,65(1)	0,64(2)	0,52(3)	0,31(4,5)	0,31(4,5)
CMC	0,24(5)	0,31(2)	0,30(3)	0,25(4)	0,36(1)
Diabetes	0,27(2)	0,40(1)	0,03(5)	0,20(3,5)	0,20(3,5)
Ecoli	0,76(1)	0,74(2)	0,51(4)	0,63(3)	0,50(5)
Glass	0,52(1,5)	0,52(1,5)	0,47(3)	0,36(4,5)	0,36(4,5)
Haberman	0,21(1)	0,19(2)	0,00(5)	0,17(4)	0,18(3)
Hayes	0,42(1)	0,37(2)	0,27(3)	0,24(4,5)	0,24(4,5)
Heart	0,30(2)	0,57(1)	0,02(5)	0,27(3)	0,24(4)
ICU	0,22(2)	0,30(1)	0,01(5)	0,18(3)	0,17(4)
Iris	0,86(2)	0,91(1)	0,49(4)	0,48(5)	0,50(3)
Libras	0,63(4,5)	0,65(2)	0,64(3)	0,66(1)	0,63(4,5)
Mammographic	0,35(2)	0,52(1)	0,22(4)	0,21(5)	0,27(3)
Segmentation	0,74(2)	0,80(1)	0,55(3)	0,50(5)	0,51(4)
Spam	0,36(5)	0,67(1)	0,42(4)	0,51(2)	0,43(3)
Teaching	0,23(5)	0,38(1)	0,29(3,5)	0,29(3,5)	0,34(2)
Vehicle	0,38(1)	0,37(2)	0,32(4,0)	0,32(4,0)	0,32(4,0)
Wine	0,62(2)	0,90(1)	0,52(3)	0,40(4,5)	0,40(4,5)
Yeast	0,47(1)	0,46(2)	0,38(4)	0,43(3)	0,29(5)

APÊNDICE B – DADOS SIMULADOS

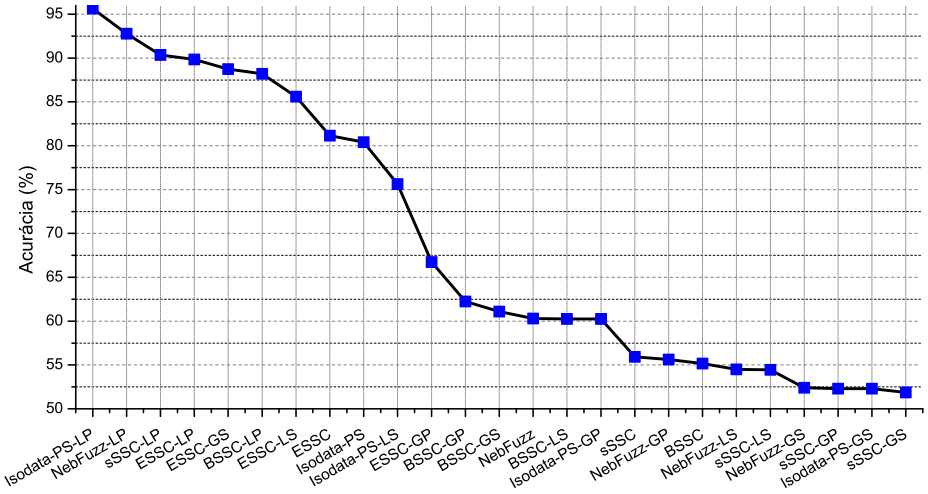


(a) Acurácia (1% de Dados Rotulados)

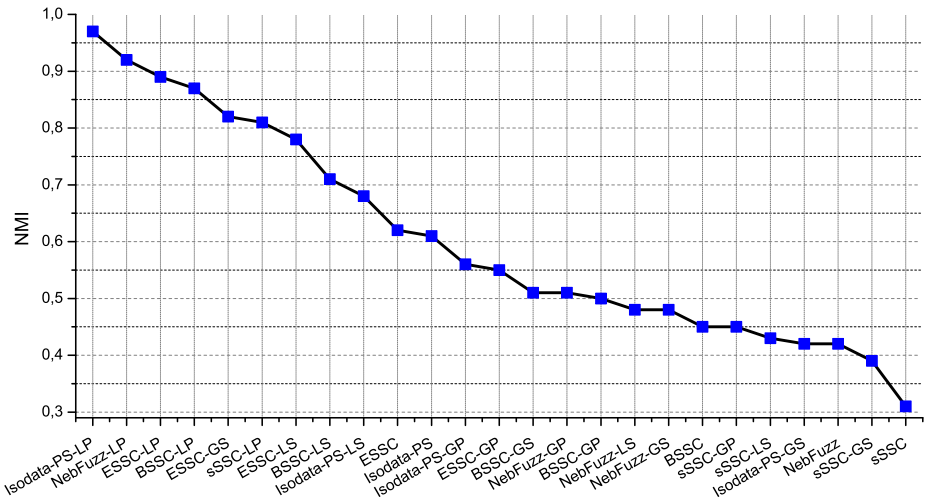


(b) NMI (1% de Dados Rotulados)

Figura 25 – Desempenho na base de dados simulados 1 com 1% de dados rotulados.

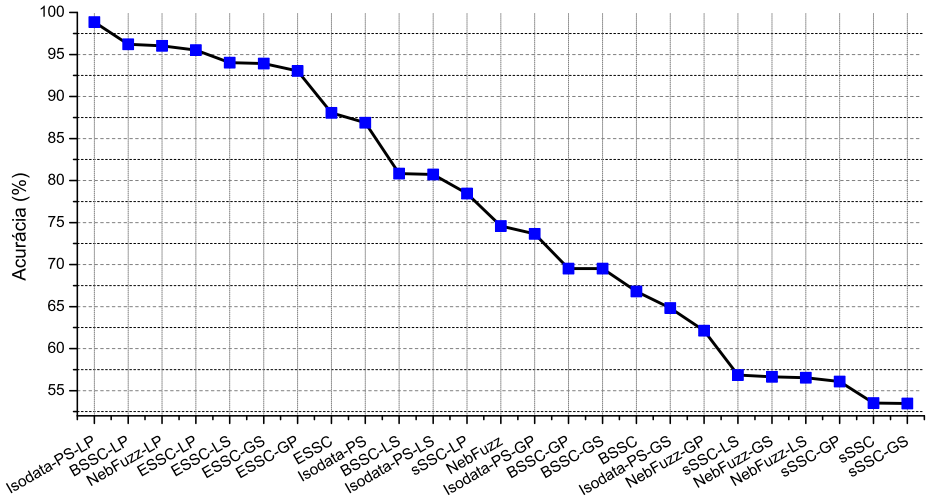


(a) Acurácia (10% de Dados Rotulados)

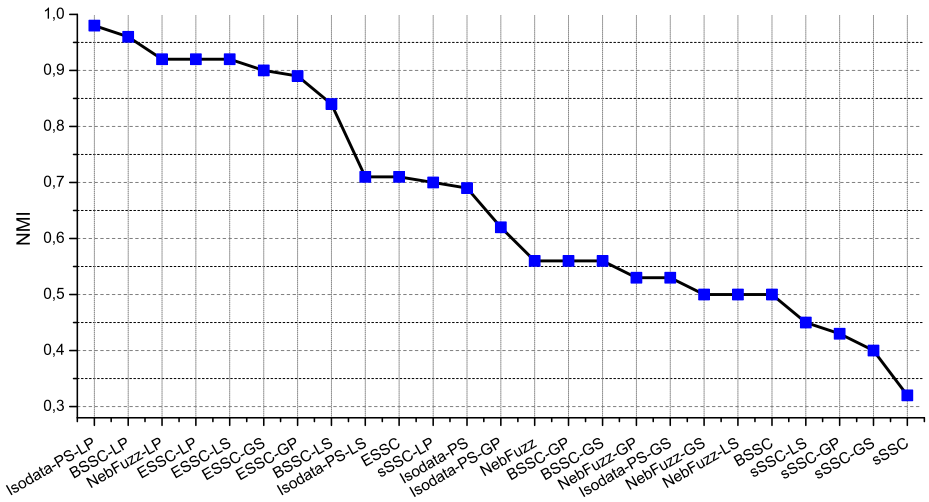


(b) NMI (10% de Dados Rotulados)

Figura 26 – Desempenho na base de dados simulados 1 com 10% de dados rotulados.

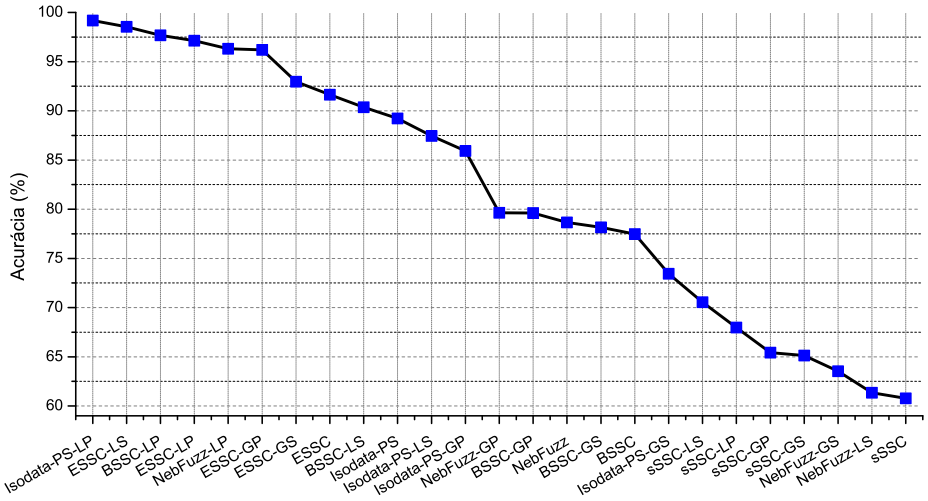


(a) Acurácia (30% de Dados Rotulados)

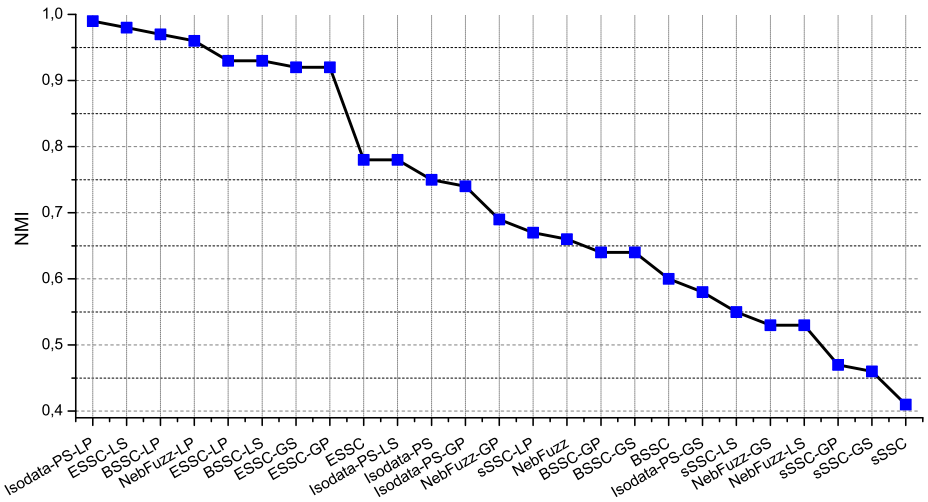


(b) NMI (30% de Dados Rotulados)

Figura 27 – Desempenho na base de dados simulados 1 com 30% de dados rotulados.

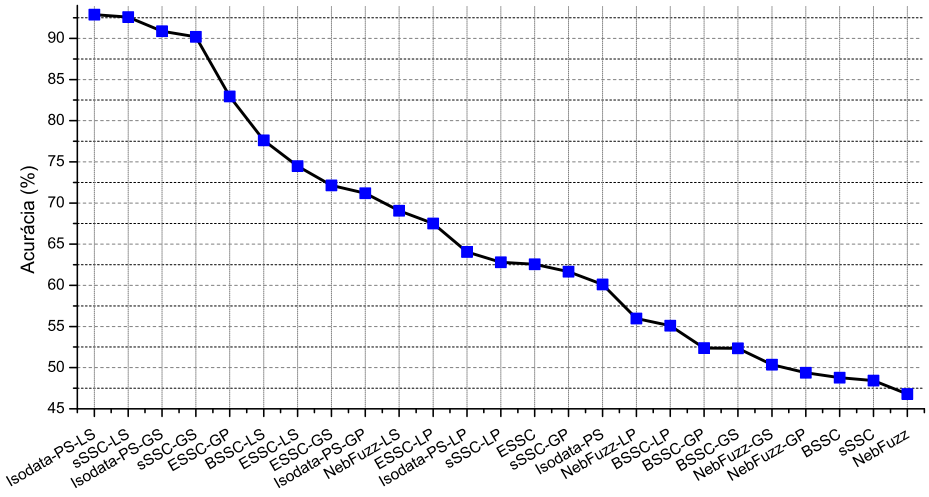


(a) Acurácia (50% de Dados Rotulados)

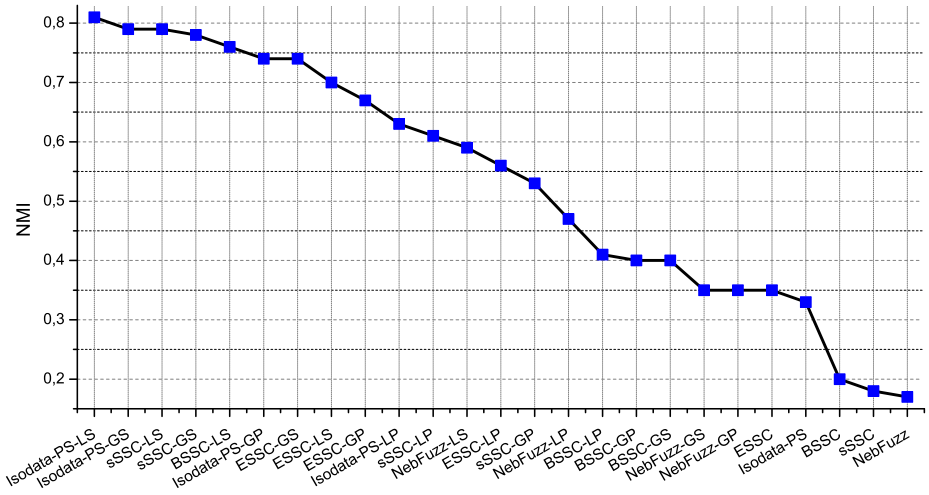


(b) NMI (50% de Dados Rotulados)

Figura 28 – Desempenho na base de dados simulados 1 com 50% de dados rotulados.

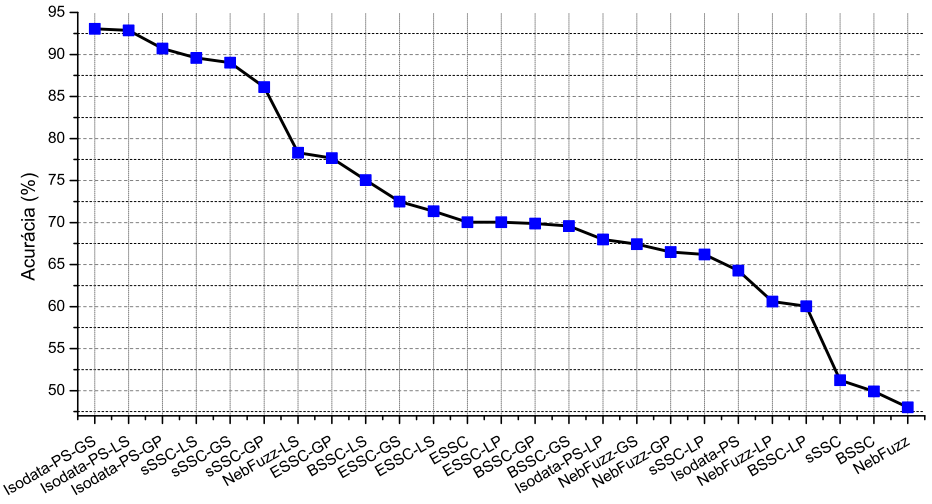


(a) Acurácia (1% de Dados Rotulados)

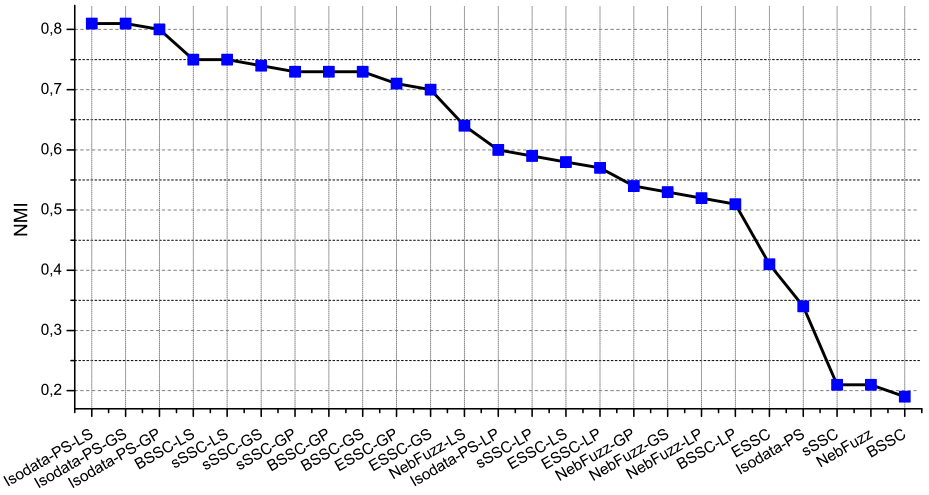


(b) NMI (1% de Dados Rotulados)

Figura 29 – Desempenho na base de dados simulados 2 com 1% de dados rotulados.

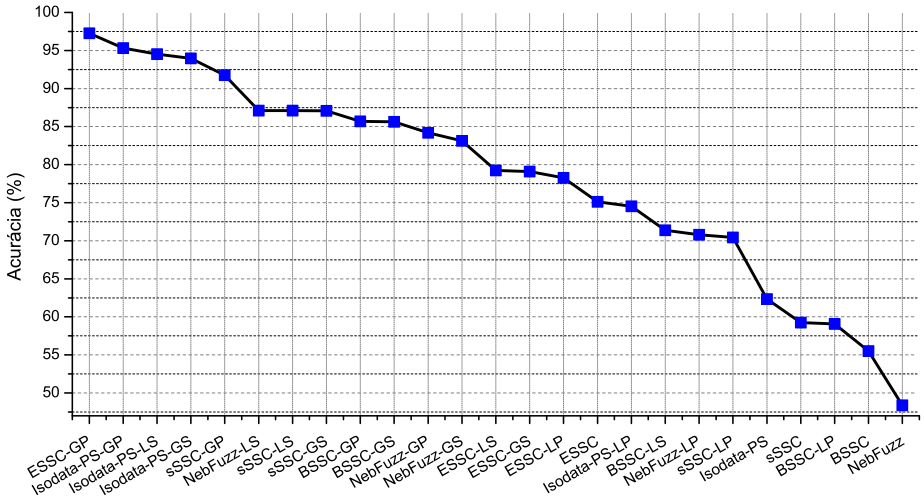


(a) Acurácia (10% de Dados Rotulados)

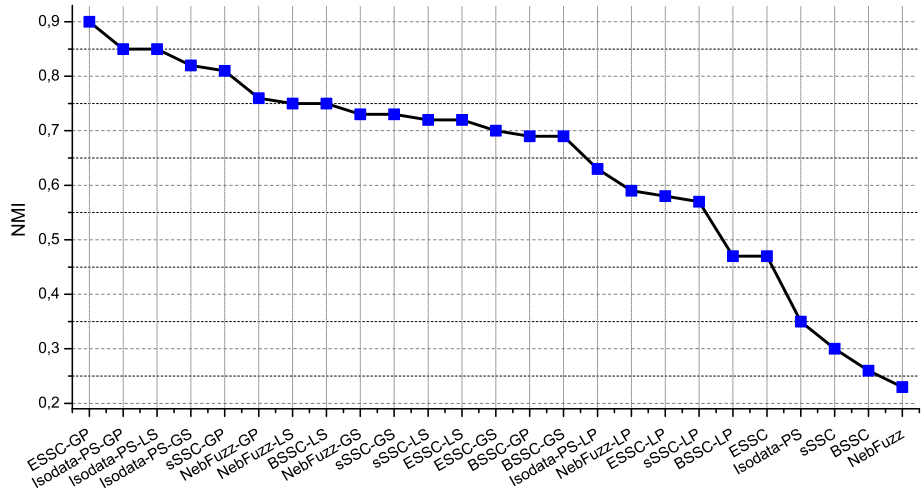


(b) NMI (10% de Dados Rotulados)

Figura 30 – Desempenho na base de dados simulados 2 com 10% de dados rotulados.

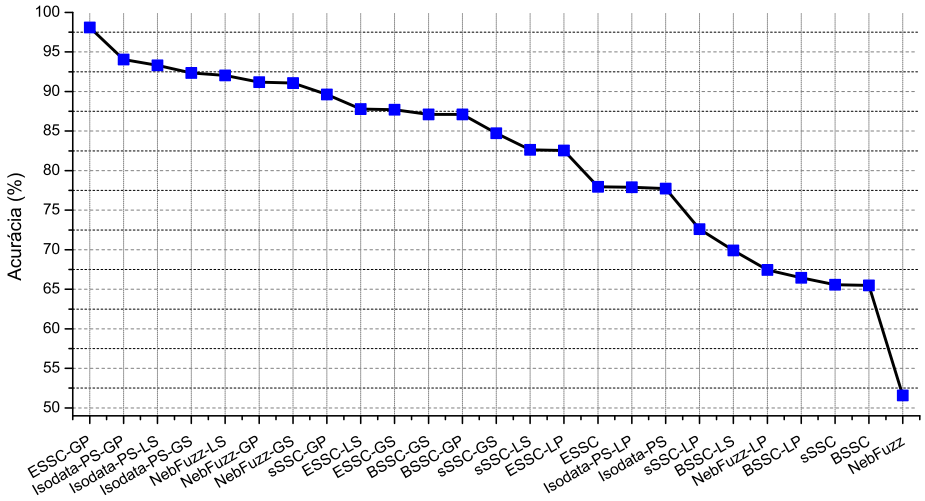


(a) Acurácia (30% de Dados Rotulados)

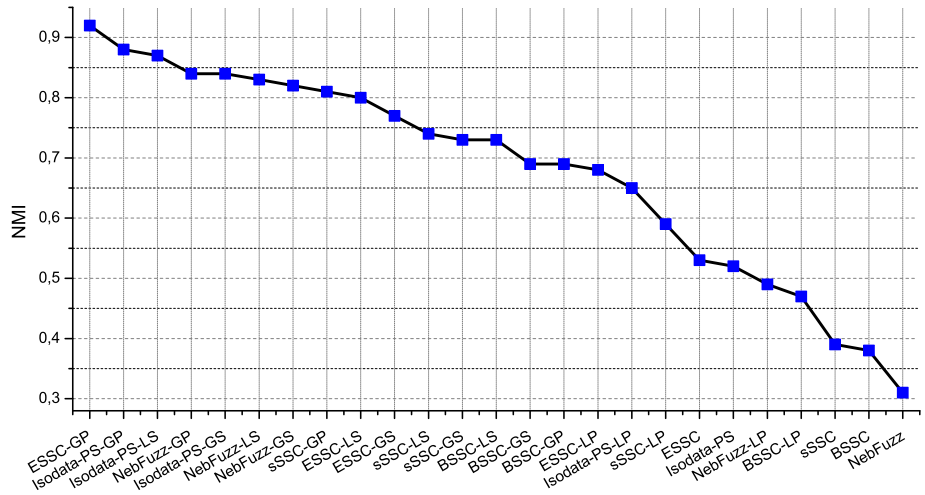


(b) NMI (30% de Dados Rotulados)

Figura 31 – Desempenho na base de dados simulados 2 com 30% de dados rotulados.



(a) Acurácia (50% de Dados Rotulados)



(b) NMI (50% de Dados Rotulados)

Figura 32 – Desempenho na base de dados simulados 2 com 50% de dados rotulados.