"Agrupamento fuzzy c-medoids semi-supervisionado de dados relacionais representados por múltiplas matrizes de dissimilaridade"

Por

Filipe Martins de Melo

Dissertação de Mestrado

Universidade Federal de Pernambuco
Centro de Informática
posgraduacao@cin.ufpe.br
http://www.cin.ufpe.br/~posgraduacao

Recife, 16 de abril de 2014

Universidade Federal de Pernambuco
Centro de Informática
Mestrado em Ciência da Computação

Filipe Martins de Melo

"Agrupamento fuzzy c-medoids semi-supervisionado de dados relacionais representados por múltiplas matrizes de dissimilaridade"

Trabalho apresentado ao Programa de Mestrado em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientador: *Francisco de Assis Tenório de Carvalho*

Recife, 16 de abril de 2014

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Martins de Melo, Filipe.
Agrupamento fuzzy c-medoids semi-supervisionado de dados relacionais representados por múltiplas matrizes de dissimilaridade / Filipe Martins de Melo. - Recife, 2014.
69 p : il., tab.

Orientador(a): Francisco de Assis Tenório de Carvalho
Dissertação (Mestrado) - Universidade Federal de Pernambuco, Centro de Informática, Programa de Pós-Graduação em Ciência da Computação, 2014.
Inclui referências, apêndices, anexos.

1. Agrupamento de dados com restrições. 2. Agrupamento fuzzy. 3. Aprendizagem semi-supervisionada. 4. Dados Relacionais. 5. Múltiplas matrizes de dissimilaridade. I. de Assis Tenório de Carvalho, Francisco. (Orientação). II. Título.

000 CDD (22.ed.)

FILIPE MARTINS DE MELO

AGRUPAMENTO FUZZY C-MEDOIDS SEMI-SUPERVISIONADO DE DADOS RELACIONAIS REPRESENTADOS POR MÚLTIPLAS MATRIZES DE DISSIMILARIDADE

Dissertação apresentada ao Programa de Pós Graduação em Ciência da Computação da Universidade Federal de Pernambuco, Centro de Informática, como requisito para a obtenção do título de Mestre em Ciência da Computação. Área de concentração: Agrupamento de dados.

Aprovado em: 26/02/2014.

BANCA EXAMINADORA

_____

Prof. Dr. Francisco de Assis Tenório de Carvalho (Orientador)
Centro de Informática - UFPE

_____

Prof. Dr. Sérgio Ricardo de Melo Queiroz (Examinador Interno)
Centro de Informática - UFPE

_____

Prof. Dr. Renato Fernandes Corrêa (Examinador Externo)
Departamento de Ciência da Informação - UFPE

Dedico à minha família,
especialmente a minha mãe e minha avó
pelo imenso incentivo e carinho.

Agradecimentos

Nessa parte dessa dissertação gostaria de citar e agradecer as pessoas que me ajudaram e estiveram presente durante minha vida acadêmica.

À minha mãe, Silvana Martins, que sempre me motivou e foi de fundamental importância em todas minhas conquistas.

À minha família, pelo apoio e torcida em tudo que fiz.

Ao professor Francisco Carvalho, pela sua orientação e confiança em diversos projetos.

Aos meus amigos na maratona de programação, em especial aos meus colegas dos times Carcará, Razão Cruzada e Challenge Accepted!: Davi Pinheiro, Luiz Silva, Laís Andrade, Pablo Pinheiro e Renan Pires, e também sou muito grato aos grandessíssimos técnicos Liliane Salgado e Pedro Bello.

Aos meus amigos de graduação, não seria possível ter passado por todos aqueles projetos sem a presença animada de vocês.

An eye for an eye
only ends up making
the whole world blind.

—MAHATMA GANDHI (1958)

Resumo

Agrupamento semi-supervisionado de dados é uma forma especial de classificação que usa uma grande quantidade de dados não rotulados combinados com uma pequena parcela de informação supervisionada para melhorar a qualidade dos grupos.

Esse trabalho introduz um algoritmo semi-supervisionado do tipo *fuzzy c-medoids* para agrupamento de dados relacionais representados por múltiplas matrizes de dissimilaridade que tem como objetivo produzir uma partição *fuzzy* e um conjunto de *medoids* para cada um desses grupos ao mesmo tempo em que aprende um vetor de relevância para cada tabela, e otimiza o critério de adequação entre os grupos *fuzzy* de forma competitiva com restrições do tipo *must-link* e *cannot-link*. Experimentos com conjuntos de dados reais indicam a utilidade desse modelo.

Palavras-chave: Agrupamento de dados com restrições, Aprendizagem semi-supervisionada, Agrupamento fuzzy, Dados Relacionais, Múltiplas matrizes de dissimilaridade

Abstract

Semi-supervised clustering is a special form of classification that uses a large amount of unlabeled data combined with a small amount of supervised information to improve the quality of the groups.

This dissertation introduces a semi-supervised fuzzy c-medois clustering algorithm of relational data with descriptions given by multiple dissimilarity matrices that aims to furnish a fuzzy partition and a set of medoids for each fuzzy cluster as well as to learn a relevance weight for each dissimilarity matrix, and by optmizing an adequacy criterion between the fuzzy clusters in a competitive way with that takes account of pairwise constraints must-link and cannot-link. Experiments with real-valued data sets show the usefulness of this algorithm.

Keywords: Constrained clustering, Fuzzy clustering, Multiple dissimilarity matrices, Relational data, Semi-supervised learning

Sumário

Lista de Figuras **x**

Lista de Tabelas **xi**

Lista de Algoritmos **xii**

1 Introdução **1**

1.1 Objetivo 3

1.2 Estrutura da Dissertação 3

2 Aprendizagem Semi-Supervisionada **5**

2.1 Introdução 5

2.2 Agrupamento semi-supervisionado com restrições 7

3 Modelo Proposto **11**

3.1 Trabalhos Relacionados 11

3.2 Descrição 11

3.2.1 Passo 1: melhores medoids 13

3.2.2 Passo 2: melhor vetor de relevância 13

3.2.3 Passo 3: atualização da partição fuzzy 14

3.3 Escolha do α 16

3.4 Escolha do β 17

3.5 Algoritmo 18

3.6 Considerações Finais 19

4 Experimentos **20**

4.1 Metodologia 20

4.2 Medidas de performance 21

4.2.1 Matriz de contingência 21

4.2.2 Taxa de erro global de classificação 22

4.2.3 Taxa de associação 22

4.2.4 F-measure 23

4.2.5 Coeficiente de Rand 24

4.2.6 Coeficiente de Rand Fuzzy 25

4.2.6.1 Frigui 25

4.2.6.2 Hüllemeier 25
4.3 Bases de Dados 26
4.3.1 Iris 26
4.3.2 Wine 27
4.3.3 Glass 27
4.3.4 Multiple features 28
4.3.5 Yeast 28
4.3.6 Flower 29

5 Resultados **31**
5.1 Desempenho dos Algoritmos 31
5.2 Iris 38
5.3 Wine 39
5.4 Glass 40
5.5 Multiple features 42
5.6 Yeast 43
5.7 Flower 43

6 Conclusão **46**

Referencias Bibliográficas **51**

Apêndices **52**

A *Derivação das equações de atualização da partição fuzzy* **53**

B *Derivação das equações de atualização do vetor de relevância* **56**

Lista de Figuras

2.1 Agrupamento de dados com restrições. (a) Imagem original com as restrições sendo mostradas pelos traços azuis e vermelhos. (b) Classificação em 5 grupos sem utilizar nenhuma restrição. (c) Classificação em 5 grupos usando restrições baseadas em 10% dos *pixels* rotulados. (Jain *et al.* [1]). 9

4.1 Categorias dos tipos de flores (Nilsback e Zisserman [2]) 30

5.1 Iris: intervalo de confiança de nível 95% para o valor médio das medidas de perfomance 32
5.2 Wine: intervalo de confiança de nível 95% para o valor médio das medidas de perfomance 33
5.3 Glass: intervalo de confiança de nível 95% para o valor médio das medidas de perfomance 34
5.4 Mfeat: intervalo de confiança de nível 95% para o valor médio das medidas de perfomance 35
5.5 Yeast: intervalo de confiança de nível 95% para o valor médio das medidas de perfomance 36
5.6 Flower: intervalo de confiança de nível 95% para o valor médio das medidas de perfomance 37
5.7 Relação entre as variáveis do Iris 38

Lista de Tabelas

4.1 A Matriz de Contingência, $a_{ij} = |P_i \cap Q_j|$ 22
4.2 Estatísticas gerais das bases de dados 26
4.3 Distribuição das classes em Yeast 29

5.1 Matriz de confusão do Iris com 10% de dados rotulados 39
5.2 Vetor de relevância do Iris com 10% de dados rotulados 39
5.3 Wine: grupo 1 40
5.4 Wine: grupo 2 40
5.5 Wine: grupo 3 40
5.6 Wine: impacto do α nas iterações 41
5.7 Glass: vetor de relevância com 10% de rótulos e $\beta = 5 \times 10^{-5}$ . . . 42
5.8 Glass: matriz de confusão com 10% de rótulos 42
5.9 Matriz de confusão para o *MFCM-dd-RWL-P* no Flower 44
5.10 Matriz de confusão para o *SS-CLAMP* no Flower (10% dados rotulados) 44

Lista de Algoritmos

3.1 Atualizar os medoids de G_k 13
3.2 Atualizar partição fuzzy 15
3.3 Encontrar valor adequado de α 17

1 Introdução

Uma das práticas mais comuns na busca pelo conhecimento é através da análise e extração de dados. Um ponto fundamental no tratamento de dados é ter a capacidade de fornecer uma classificação concisa em grupos ou categorias, de forma que se torna possível a compreensão e aprendizagem deles [1]. Trata-se de uma das formas mais empregadas desde tempos mais primitivos, a organização da informação desempenhou um papel indispensável para o desenvolvimento da humanidade ao longo da história. Seja para aprender sobre um novo objeto ou entender um novo fenômeno, pessoas sempre tentam procurar características que podem descrever cada um deles, e isso pode ser intuitivamente feito através da comparação com outros objetos ou fenômenos existentes e conhecidos, baseando-se na semelhança e diferença entre eles [3].

Agrupamento de dados é o método que tem como finalidade organizar um conjunto de objetos em grupos de forma que os objetos de um mesmo grupo apresentam um alto grau de semelhança entre si, enquanto que os objetos de grupos diferentes apresentam um alto grau de dissimilaridade [1] [4]. Esses métodos possuem uma natureza multi-disciplinar e têm sido amplamente utilizados em diversos tipos de aplicações e problemas que necessitam da identificação de características e distribuição de padrões nos dados, tais como aprendizagem de máquina, mineração de dados, recuperação de informação, segmentação de imagens, e taxonomia [5].

As técnicas mais populares de agrupamento de dados são dividas entre métodos hierárquicos e métodos particionais [1] [3]. Métodos hierárquicos fornecem uma sequência de partições aninhadas dos dados de entrada formando uma hierarquia, que pode ser feita através do modo aglomerativo (abordagem *bottom up*) em que cada objeto é inicialmente seu próprio grupo e gradativamente são feitas fusões entre esses grupos segundo um critério de união até que todos os objetos pertencem

ao mesmo grupo, ou do modo divisivo (abordagem *top down*) em que todos objetos pertencem inicialmente a um mesmo grupo e em cada iteração os grupos são subdivididos em grupos menores. Os métodos particionais geram uma divisão dos dados em um número fixo de grupos, eles geralmente procuram obter uma partição que otimiza (normalmente localmente) uma função objetivo diminuindo seu valor até a corvergência. Para melhorar o resultado do algoritmo esses algoritmos são executado múltiplas vezes com diferentes pontos de partida e a melhor inicialização, a que apresenta o menor critério, é escolhida como resultado final.

Métodos particionais podem ser divididos em agrupamentos do tipo *hard* ou do tipo *fuzzy* [3]. A principal diferença entre eles sendo a restrição sobre a pertinência de cada objeto, nas partições *hard* cada objeto pertence a exatamente um único grupo e nas partições *fuzzy* essa condição é mais relaxada com a existência de um grau de pertinência parcial de cada objeto com todos os grupos. Métodos particionais são uma escolha mais comum em reconhecimento de padrões do que métodos hierárquicos [1].

Algoritmos de agrupamento de dados geralmente operam em objetos que estão descritos através dos seus atributos ou por dados relacionais. Quando cada objeto é representado por um vetor d-dimensional com os valores dos seus atributos eles são chamados de dados originais[1]. Alternativamente, quando a informação disponível é uma coleção de proximidades entre todos os pares de objetos, eles são chamados de dados relacionais. Essas proximidades sao comumente pré-computadas e armazenadas numa tabela de dados onde cada célula dela mede a dissimilaridade (distância) entre dois objetos [6]. Métodos de classificação podem tomar vantagem do uso de dados relacionais representados por múltiplas matrizes de dissimilaridade. Em uma base de dados de imagens, por exemplo, a relação entre os objetos pode ser descrita por múltiplas matrizes de dissimilaridade, uma para a cor, outra contendo informação sobre a textura, e outra para a estrutura entre elas [7]. Ao mesmo tempo que se torna importante associar um fator de relevância para cada tabela, pois a influência delas na definição dos grupos é diferente [7].

A tarefa de agrupamento de dados pode ser bastante difícil em alguns casos, e com o motivo de ajudar na formação do grupos tem emergido o agrupamento semi-supervisionada de dados como uma nova excitante direção de pesquisa na área de aprendizagem de máquina. Esta técnica é intermediária entre a métodos supervionados e não-supervisionados. Além dos dados não-rotulados os algoritmos

[1]Do inglês: *feature data*

semi-supervisionados também usam informação supervisionada sobre alguns objetos para guiar no processamento de todos eles [8].

Uma das formas mais simples de ter informação sobre os dados foi proposto por Wagstaff e Cardie [9] e consiste na forma de restrições do tipos *must-link* e *cannot-link* entre pares de objetos. Esse tipo de conhecimento é considerado ser um dos mais genérico por não precisar da informação explícita sobre as classes dos objetos [8]. A restrição *must-link* indica uma relação de conformidade entre pares de objetos e significa que eles deveriam pertencer ao mesmo grupo, enquanto que a restrição *cannot-link* indica divergência e contém pares de objetos não deveriam pertencer ao mesmo grupo [8]. O principal objetivo do uso dessa informação é servir como uma extensão nos métodos de agrupamento e possibilitar uma melhor categorização dos dados.

1.1 Objetivo

Nesse contexto, o objetivo desse trabalho é a proposta de um algoritmo do tipo *fuzzy c-medoids* semi-supervisionado de dados relacionais representados por múltiplas matrizes de dissimilaridades (SS-CLAMP). O SS-CLAMP tem por finalidade encontrar uma partição *fuzzy* do conjunto de dados de entrada ao mesmo tempo que encontra um vetor de pesos de relevância para cada matriz de dissimilaridade e um conjunto de *medoids* para cada grupo. O critério de adequação também leva em consideração restrições do tipo *must-link* e *cannot-link* sobre os dados.

Esse trabalho também apresenta uma análise dos resultados da aplicação do modelo proposto em experimentos com bases de dados reais que foram extraídas principalmente do repositório da UCI (*University of California, Irvine*) [10] para avaliar o desempenho desse modelo.

1.2 Estrutura da Dissertação

Esta dissertação está organizada em 6 capítulos mais o apêndice. Nesse primeiro capítulo foi apresentado a introdução dos principais tópicos desse trabalho, as motivações e objetivos dessa dissertação.

No Capítulo 2 será apresentado os principais conceitos e trabalhos relacionados aos métodos de aprendizagem semi-supervisionada.

O algoritmo proposto nessa dissertação será apresentado no Capítulo 3. Ele

mostrará as etapas e a descrição algorítmica do modelo e discutirá sobre a escolha de valores para alguns parâmetros em sua execução.

No Capítulo 4 será apresentado as medidas de perfomance utilizadas na comparação do desempenho entre os algoritmos implementados, a metodologia e a descrição das características das bases de dados que serão utilizadas nos experimentos. Depois, o Capítulo 5 mostrará os resultados experimentais e discutirá o desempenho do modelo proposto em relação a um outro algoritmo.

Finalmente, o Capítulo 6 trará os comentários finais e as propostas de trabalhos futuros.

2

Aprendizagem Semi-Supervisionada

Nesse capítulo discutiremos alguns conceitos básicos e os principais aspectos da aprendizagem semi-supervisionada. Apresentamos também alguns trabalhos relacionados que usam técnicas baseadas nessa abordagem. Em seguida, é apresentado o agrupamento de dados semi-supervisionado que usa restrições, onde discutiremos suas características e aplicações.

2.1 Introdução

Mais recentemente, aprendizagem semi-supervisionada tornou-se um tópico de atenção crescente na comunidade, principalmente devido à sua importância e aplicação em vários cenários [8]. Diversas abordagens exploram o uso parcial de informação supervisionada, dentre as mais conhecidas temos: modelos baseados em sementes, modelos probabilísticos, modelos que otimizam uma função objetivo, algoritmos genéticos, re-treinamento, modelos que usam máquina de vetores de suporte, e modelos baseados em grafos [11].

Em [12], Basu *et al.* aprensentam dois algoritmos semi-supervisionados variantes do *K-Means* [13]. Eles exploram o uso dos dados rotulados como sementes no processo de inicialização dos algoritmos, essa mesma informação ainda pode continuar a ser usada para guiar o processo de agrupamento dos dados durante os próximos passos. São eles:

- *Seeded-KMeans*: em vez de escolher os centróides iniciais randomicamente ele usa as informações no conjunto de sementes, assumindo que existe pelo menos um objeto rotulado associado a cada grupo, e posteriormente as sementes não são mais usadas durante a parte de atribuição dos grupos.

- *Constrained-KMeans*: ele é equivalente ao *Seeded-KMeans*, a diferença é que ele continua a usar as informações no conjunto de sementes na fase de afetação dos grupos, isto é, fazendo com que o grupo de cada objetos rotulados seja o mesmo grupo definido na inicialização e permitindo que apenas os objetos sem rótulo tenham seus grupos atualizados nas próximas iterações.

Pedrycz e Waletzky [14] modificam o FCM [15] para usar informação supervisionada na forma de dados rotulados, no entanto os rótulos definem apenas uma classe para esses objetos e isso faz com seja produzida uma classificção do tipo *hard* para eles. Em [16], Bouchachia e Pedrycz ampliam esse modelo para que ele permita que cada classe pode ser representada por mais de um grupo. Eles também apresentaram uma transformação de agrupamento para um classificador do seu algoritmo. Experimentos realizados com uma base médica e com dados sintéticos indicaram um bom desempenho na discriminação dos grupos. Eles também discutiram sobre o valor do fator de controle da informação supervisionada na função objetivo e sua influência nos resultados finais. Em [17], Bouchachia e Pedrycz ainda fazem mudanças no FCM para estudar o efeito de diferentes funções de distância aplicadas ao seu modelo semi-supervisionado.

Nigam *et al.* [18] introduzem uma modelagem probalística para classificação de documentos, através de uma abordagem que combina um classificador Bayesiano ingênuo (NB) com o algoritmo Esperança-Maximização (EM). O algoritmo começa treinando um classificador usando apenas os dados rotulados, depois disso ele faz uma categorização probabilística dos dados sem rótulo e então o classificador é novamente treinado usando todo o conjunto de dados e continua repetindo nesse processo até a convergência. Resultados experimentais sobre três diferentes bases de dados indicaram uma redução do erro de classificação em até 30%. Ainda no seu trabalho eles propõem extensões ao esquema utilizado no EM para que ele permita que os dados ainda não rotulados acrescentem melhoras na precisão do classificador, apesar de que isso viola algumas suposições desse modelo. Outros trabalhos que usam uma estratégia similar foram apresentados por Baluja [19] que também adota um modelo probabilístico para classificação de um conjunto de imagens dentre cinco expressões faciais, e por Fujino *et al.* [20] que usa um modelo híbrido com técnicas generativas e discriminativas para classificação de textos.

Blum e Mitchell [21] apresentam o *co-training*, que é uma técnica que assume que é possível ter uma partição da representação dos dados segundo duas visões independentes onde cada um delas contém informações suficiente para produzir

uma classificação correta deles isoladamante. O algoritmo introduzido por eles inicialmente treina dois classificadores distintos para cada visão utilizando apenas os dados rotulados e iterativamente são selecionadas as predições mais confiáveis dos dados sem rótulo segundo ambos classificadores e depois disso elas são adicionadas nos modelos que são treinados novamente, esse processo é repetido até a construção dos classificadores finais.

Em [22], Klinkenberg propõe o uso de máquina de vetores de suporte (SVM) [23] para classificação semi-supervisionada, ele explora o uso de dados sem rótulos para diminuir a necessidade dados rotulados na construção do classificador. O algoritmo segue um processo transdutivo usando séries de amostras de tamanhos fixo dos dados, no começo eles usa apenas as amostras rotuladas e depois de já ter consolidado algum conhecimento o modelo faz predições sobre os dados não rotulados e acrescenta esse novo conceito no classificador.

Demiriz *et al.* [24] desenvolveram um algoritmo semi-supervisionado que usa algoritmo genético (GA) como alternativa de otimização da função critério. Sua função objetivo é uma combinação linear da medida da dispersão e da impureza da partição encontrada, a parte que mede a dispersão está associada a um termo puramente não-supervisionado e a impureza calcula a informação supervisionada em relação à partição.

Técnicas baseadas em grafos para classificação semi-supervisionada também têm sido alvo de pesquisa, elas aprensentam um grande número de métodos que exploram essa representação. A idéia principal dessas abordagens é representar os objetos como vértices e as arestas como a relação entre dois objetos, geralmente com um peso que indica a similaridade deles [11]. Os dados rotulados propagam sua informação através da estrutura do grafo e vão gerando rótulos em todos os outros vértices [8].

2.2 Agrupamento semi-supervisionado com restrições

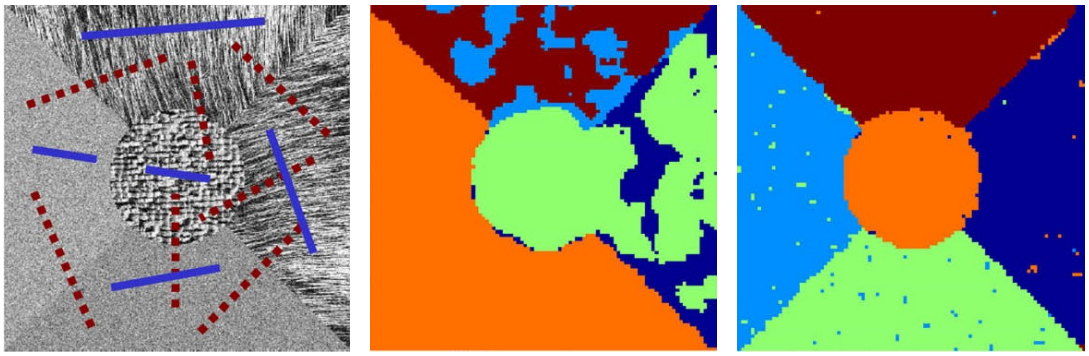
Além da descrição explícita dos rótulos de classe dos objetos, uma outra forma de conhecimento que pode ser utilizado no agrupamento semi-supervisionado de dados foi introduzida por Wagstaff e Cardie [9], eles propõem dois tipos de restrições entre as instâncias dos próprios dados. A informação nesse caso consiste de restrições dos tipos *must-link* e *cannot-link*, que também são conhecidas alternativamente por

restrições de equivalência e de não equivalência, respectivamente [8]. As restrições *must-link* especificam pares de instâncias que devem ser atribuídas no mesmo grupo, enquanto que as restrições *cannot-link* especificam pares de instâncias que não devem ser atribuídas no mesmo grupo.

A construção desses conjuntos de restrições é, geralmente, feita através da ajuda de um especialista que tem conhecimento do domínio do problema [25]. Embora pareçam simples, esses tipos de restrições são na verdade bastante eficazes e com um número suficiente de restrições é possível obter melhoras significativas no resultado [18] [26]. As restrições são capazes de servir como um fator para especificar propriedades desejadas na solução do agrupamento final, elas também estreitam e diminuem o espaço de busca promovendo uma direção reduzida no processo de categorização dos dados. Elas são usadas principalmente como um operador para guiar o agrupamento em cada iteração nos algoritmos que tentam evitar possíveis conflitos [8]. Dentre os principais benefícios temos a melhora na precisão dos resultados e por servir como um fator responsável para a formação de grupos com uma geometria desejada [25]. A Figura 2.1 ilustra o comportamento de um algoritmo semi-supervisionado comparado a um algoritmo não-supervisionado que ignora a informação supervisionada no processo de segmentação de uma imagem [1]. A textura da imagem juntamente com as restrições entre pares de pixels são exibidas na Figura 2.1(a) onde os traços azuis representam relações *must-link* e os vermelhos as relações *cannot-link*. O algoritmo da Figura 2.1(b) não usa as restrições, enquanto o da Figura 2.1(c) apresenta melhor perfomance utilizando-a na formação dos 5 grupos.

A informação sobre a classe dos objeto pode não estar disponível ou até mesmo não ser possível, enquanto que a extração de objetos similares pode ser feita sem muito esforço ou até mesmo naturalmente, por exemplo, é esperada a presença de um mesmo agente entre trechos com pouca variação de tempo em gravações de áudio e vídeo [19]. Em trechos de filmes a localização dos atores não sofre grandes variações entre cenas sucessivas, e o mesmo caso se aplica em gravações de áudio onde a voz de um mesmo falante *desconhecido* tem grandes chances de estar presente entre intervalos próximos durante uma conversa.

Restrições entre pares de objetos também são consideradas uma forma mais suave de representar informação sobre um conjunto de dados já que não há a necessidade de explicitar classes para os objetos [8]. Além disso, essa abordagem é mais genérica do que os métodos que apenas usam os rótulos das classes, pois é



(a) Imagem com restrições (b) Sem restrições (c) 10% de rótulos

Figura 2.1: Agrupamento de dados com restrições. (a) Imagem original com as restrições sendo mostradas pelos traços azuis e vermelhos. (b) Classificação em 5 grupos sem utilizar nenhuma restrição. (c) Classificação em 5 grupos usando restrições baseadas em 10% dos *pixels* rotulados. (Jain *et al.* [1]).

possível contruir as restrições entre os pares de objetos usando a informação sobre os rótulos de suas classes de origem, mas a recíproca não é verdadeira.

Em aplicações com algoritmos de agrupamento de dados o uso de restrições em modelos particionais é mais comum do que em modelos hierárquicos. As restrições são tipicamente usadas para mudar a função objetivo de forma a se ajustar com um termo de penalidade sobre as restrições que influenciam na distorção e formação dos grupos.

Wagstaff *et al.* [26] desenvolveram um sistema que faz uso de restrições entre pares de faixas rodoviárias, os dados foram gerados através das coordenadas de GPS (*Global Positioning System*) de vários motoristas. Se dois pontos estivessem a uma distância de pelo menos quatro metros então eles deveriam pertencer a grupos diferentes. Em seus experimentos o algoritmo utilizado foi uma extensão do *K-Means*, o *COP-K-Means*, para se adaptar com as restrições. Ele conseguiu consistentemente encontrar uma classificação perfeita das faixas rodoviárias.

Basu *et al.* em seus trabalhos anteriores usavam a informação dos rótulos no processo de inicialização dos grupos usando *seeding* [12], já em [27] eles aderem as restrições *must-link* e *cannot-link* usando um algoritmo que é um refinamento do *K-Means* para se adaptar a essa informação, o *PCK-Means* (*Pairwise Constrained K-Means*).

Grira *et al.* [28] apresentam o algoritmo PCCA (*Pairwise Constrained Competitive Agglomeration*) que usa o FCM como critério principal para formação dos grupos *fuzzy* combinado com um termo de penalidade para as restrições e um termo

de *bias* para encontrar o número de grupos automaticamente.

Aprendizagem ativa para seleção automática de restrições também é uma área de estudo em agrupamento de dados com restrição [27] [29]. Essa técnica traz o benefício de possibilitar o uso de dados sem rótulos como fonte adicional de informação [8], e isso pode ser indispensável em certos problemas com alto custo na descoberta de rótulos em relação a um processo mecânico. Basu *et al.* [27] apresentam o *Explore* que foi usado juntamente com o *PCK-Means* para melhorar o desempenho dele ao mesmo tempo que tenta usar uma menor quantidade de restrições. O *Explore* gera uma estrutura de vizinhança a partir dos dados de entrada e tenta escolher os pares de objetos que mais auxiliam para a formação dos K grupos. Eles assumem que a informação sobre a relação entre os pares escolhidos é acessada através de um *oráculo* é sempre correta. O algoritmo procura por novos pares fazendo perguntas sobre a relação entre pares de objetos que estão mais distantes dos que ele já conhece, com isso ele tenta garantir a divisão explícita dos dos grupos sempre selecionando diferentes (distantes) novos objetos.

Mais relacionado ao método aprensetando nesse trabalho é o algoritmo SS-CARD que foi proposto por Frigui *et al.* [30]. O SS-CARD [30] *Semi-Supervised Clustering and Aggregation of Relational Data*) é um algoritmo semi-supervisionado do tipo *fuzzy* para objetos que são representados por múltiplas matrizes de dissimilaridades. Ele foi desenvolvido como uma extensão do algoritmo CARD [7] que por sua vez é baseado no algoritmo NERF [31]. Ele foi projetado para agregar as distâncias entre pares de múltiplas matrizes de dados relacionais e particionar eles em grupo, e simultaneamente aprender um vetor de pesos de relevância para cada matriz.

3

Modelo Proposto

Nesse capítulo apresentamos o algoritmo semi-supervisionado do tipo *fuzzy c-medoids* que é proposto nessa dissertação. Descrevemos suas etapas algorítmicas em mais detalhe e discutimos também sobre a escolha de alguns parâmetros de sua execução. No final é apresentado uma versão dele que tenta explicitá-lo de forma mais simplificada.

3.1 Trabalhos Relacionados

O SS-CLAMP é uma extensão do modelo MFCMdd-RWL-P [32] para incorporar o uso de informação supervisionada no processo de agrupamento dos dados. O MFCMdd-RWL-P é um algoritmo de agrupamento de dados do tipo *fuzzy* que procura por uma partição do conjunto de dados de entrada levando em consideração a representação deles através de múltiplas matrizes de dissimilaridade, ele produz como resultado final um conjunto de *medoids* para cada grupo *fuzzy* e um vetor de relevância para cada tabela. Outros algoritmos que seguem a mesma estratégia mas com o valor da relevância estimadas de formas diferentes também foram apresentandos em [32]. Resultados experimentais usando bases do repositório da UCI [10] em comparação com os modelos NERF [31] e CARD-R [7] indicaram um resultado favorável para o MFCMdd-RWL-P.

3.2 Descrição

Seja $E = \{e_1, \ldots, e_n\}$ um conjunto de n objetos e sejam T matrizes $(n \times n)$ de dissimilaridades $D_j = [d_j(e_i, e_l)]\,(j = 1, \ldots, T)$, em que $d_j(e_i, e_l)$ representa a dissimilaridade entre os objetos e_i e e_l na matriz de dissimilaridade D_j. Os K grupos

fuzzy $(C_1, \ldots, C_K)$ têm como representantes um conjunto de *medoids* para cada um deles, esse vetor é um conjunto de cardinalidade fixa $1 \leq p << n$ representado por $(G_1, \ldots, G_K)$, isto é, $G_k \in E^{(p)} = \{A \subset E : |A| = p\}$.

Os conjuntos de restrições *must-link* e *cannot-link* são representandos respectivamente por $\mathcal{M}$ e $\mathcal{C}$, tal que $\mathcal{M}$ contém pares (l, m) que indica que os objetos e_l and e_m devem ser atribuídos ao mesmo grupos e que $\mathcal{C}$ contém pares (l, m) que devem ser atribuídos em diferentes grupos. As restrições são do tipo *soft* já que sua representação na função objetivo está associada com o grau de pertinência dos objetos na partição *fuzzy*.

O resultado final produz também um vetor de relevância para cada tabela de dissimilaridade relacionado localmente para cada grupo, $\lambda_k = (\lambda_{k1}, \ldots, \lambda_{kT})\,(k = 1, \ldots, K)$, seus valorem criam uma relação de importância entre as tabelas para a formação dos grupos *fuzzy*. A partição *fuzzy* nesse modelo é representada por uma matriz U que associa um grau de pertinência para cada objeto em cada grupo, $U = (u_1, \ldots, u_n)$ em que $u_i = (u_{i1}, \ldots, u_{iK})$.

Esse algoritmo será denotado aqui por SS-CLAMP, sua função objetivo é apresentada abaixo:

$$
\begin{aligned}
J \;=\; & \sum_{i=1}^{n}\sum_{k=1}^{K}(u_{ik})^2 \sum_{j=1}^{T}\lambda_{kj}\sum_{e\in G_k} d_j(e_i, e) \\
& + \alpha\left(\sum_{(l,m)\in\mathcal{M}}\sum_{r=1}^{K}\sum_{\substack{s=1\\ s\neq r}}^{K} u_{lr}u_{ms} + \sum_{(l,m)\in\mathcal{C}}\sum_{r=1}^{K} u_{lr}u_{mr}\right) + \beta\left(\sum_{k=1}^{K}\sum_{j=1}^{T}\lambda_{kj}^2\right)
\end{aligned}
\tag{3.1}
$$

sujeito a

$$
u_{ik} \geq 0 \qquad \sum_{k=1}^{C} u_{ik} = 1 \qquad \forall i \quad \text{e} \quad \lambda_{kj} > 0 \qquad \prod_{j=1}^{T}\lambda_{kj} = 1 \qquad \forall k.
$$

O primeiro termo na equação (3.1) é baseado no MFCMdd-RWL-P [32] para produzir uma partição concisa. O segundo termo vem do PCCA [28] com o α controlando a importância das restrições *must-link* e *cannot-link*. O último termo foi inspirando no SCAD1 [33] com a soma dos quadrados das relevâncias que funciona como um termo regularizador, sua importância é controlada pelo fator de influência β.

O algoritmo executa N_{init} inicializações e em cada uma delas um número máximo

de N_{iter} iterações. Dentre todas as execuções é selecionada a que apresenta menor valor da função objetivo como resultado final.

Esse algoritmo parte de uma partição *fuzzy* inicial e alterna entre os passos de atualização dos *medoids* e do vetor de relevância. Ele mantém a sequência de atualização até a convergência da partição ou quando o máximo número de iterações é atingido.

3.2.1 Passo 1: melhores medoids

Nesse passo, a partição *fuzzy* representada por $\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_n)$, e o vetor de relevância $\lambda = (\lambda_1, \ldots, \lambda_K)$ são mantidos fixos. O novo conjunto de *medoids* $G_k = G^* \in E^p \, (k = 1, \ldots, K)$ pode ser escolhido otimamente através do procedimento guloso apresentado no Algoritmo (3.1).

A prova dele é simples e se baseia no fato de que os objetos no conjunto de *medoids* de cada grupo se comportam de forma independente, logo a escolha ótima pode ser feita gulosamente para cada um deles através dos p objetos de menor distância em relação a matriz da partição *fuzzy* de um dado grupo.

Algoritmo 3.1 Atualizar os medoids de G_k

$G^* \leftarrow \emptyset$;
while $|G^*| \neq p$ **do**
Encontre $e_l \in E, e_l \notin G^*$ tal que:

$$l = argmin_{1 \leq h \leq n} \sum_{i=1}^{n} (u_{ik})^2 \sum_{j=1}^{T} \lambda_{kj} d_j(e_i, e_h)$$

$G^* \leftarrow G^* \cup \{e_l\}$;
end while

3.2.2 Passo 2: melhor vetor de relevância

Nesse passo, a partição *fuzzy*, representada por $\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_n)$, e o conjunto de *medoids* $G = (G_1, \ldots, G_K)$ são mantidos fixos. As condições $\prod_{j=1}^{T} \lambda_{kj} = 1$ e $\lambda_{kj} > 0$ devem ser satisfeitas, e usando os multiplicadores de Lagrange o novo vetor de relevância pode ser calculado segundo a equação (3.2) quando $\beta = 0$.

$$\lambda_{kj} = \frac{\left\{ \prod_{h=1}^{p} \left[\sum_{i=1}^{n} (u_{ik})^2 \sum_{e \in G_k} d_h(e_i, e) \right] \right\}^{\frac{1}{p}}}{\left[\sum_{i=1}^{n} (u_{ik})^2 \sum_{e \in G_k} d_j(e_i, e) \right]} \tag{3.2}$$

Quando $\beta \neq 0$ o vetor de relevância é calculado segundo a equação (3.3), onde θ_k é a solução dessa mesma equação quando fazemos $\prod_{h=1}^{p} \lambda_{kh} = 1$. O valor de θ_k nessa equação pode ser calculada eficientemente usando Newton's method [34] ou simplesmente usando uma busca binária [35].

$$\lambda_{kj} = \frac{\sqrt{\left[\sum_{i=1}^{n}(u_{ik})^2 \sum_{e \in G_k} d_j(e_i, e)\right]^2 + 8\beta\theta_k} - \left[\sum_{i=1}^{n}(u_{ik})^2 \sum_{e \in G_k} d_j(e_i, e)\right]}{4\beta} \tag{3.3}$$

A mudança do fator de relevância nesse modelo pode ser facilmente incorporada pois a afetação dos cálculos são independentes entre si, o único detalhe que precisa ser cuidadosamente avaliado é a escolha e cálculo do valor de β em uma nova representação.

3.2.3 Passo 3: atualização da partição fuzzy

Nesse passo, o conjunto de *medoids* $G = (G_1, \ldots, G_K)$ e o vetor de relevância $\mathbf{\Lambda}$= $(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_K)$ são mantidos fixos. A partição *fuzzy*, representada por $\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_n)$, onde $\mathbf{u}_i = (u_{i1}, \ldots, u_{iK})\,(i = 1, \ldots, N)$ que minimiza o a função critério, é tal que o grau de pertinência $u_{ik}\,(i = 1, \ldots, n; k = 1, \ldots, K)$, com a condição que $\sum_{k=1}^{K} u_{ik} = 1$ e $u_{ik} \geq 0$, foi calculado usando o método dos multiplicadores de Lagrange.

Além disso, para garantir valores não-negativos para os novos valores do grau de pertinência u_{ik} foi adicionado também as condições do *Kuhn e Tucker*, que foram facilmente incorporadas nessa atualização similarmente como no algoritmo FANNY [6].

$$u_{ik} = \frac{\gamma_i - b_{ik}}{a_{ik}} \tag{3.4}$$

onde

$$\gamma_i = \frac{1 + \sum_{w \in V_i}(\frac{b_{iw}}{a_{iw}})}{\sum_{w \in V_i}(\frac{1}{a_{iw}})} \tag{3.5}$$

e

$$a_{ik} = 2\sum_{j=1}^{T} \lambda_{kj} \sum_{e \in g_k} d_j(e_i, e)$$

$$b_{ik} = \alpha \left(\sum_{(i,m) \in \mathcal{M}} \sum_{\substack{s=1 \\ s \neq k}}^{K} u_{ms} + \sum_{(i,m) \in \mathcal{C}} u_{mk} + \sum_{(l,i) \in \mathcal{M}} \sum_{\substack{s=1 \\ s \neq k}}^{K} u_{sm} + \sum_{(l,i) \in \mathcal{C}} u_{kl} \right)$$

Algoritmo 3.2 Atualizar partição fuzzy

```
for i = 1 to n do
  V_i ← {1, ..., K};
end for
update ← true;
while update do
  update ← false;
  for i = 1 to n do
    Calcule γ_i usando Equação (3.5);
    for k = 1 to K do
      if k ∈ V_i then
        Calcule u_ik usando Equação (3.4);
        if u_ik ≤ 0 then
          u_ik ← 0;
          V_i ← V_i \ {k};
          update ← true;
        end if
      end if
    end for
  end for
end while
```

De acordo com todas essas condições pudemos encontrar um procedimento simples e iterativo para computar o novo valor do grau de pertinência que é apresentando no Algoritmo (3.2).

O único caso especial acontece quando o conjunto $Z_i = \{k \mid a_{ik} = 0\}$ não é vazio pois isso causa uma degenaração no cálculo das equações (3.4) e (3.5). Analisando essas equações podemos notar que o termo a_{ik} mede a distância do indivíduo e_i em relação aos *medoids* do grupo k, mas mesmo que esse termo seja zero também é preciso levar em consideração o termo b_{ik} que causa um efeito negativo para o u_{ik} como é mostrado na equacao (3.4). Com o objetivo de maximizar o grau de pertinência para os grupos mais próximos pode ser provado que isso pode ser possível se $\gamma_i = min(\{b_{ik} \mid k \in Z_i\})$. Os valores de u_{ik} então podem ser calculados de acordo com as equações (3.6). Como pode ser visto no caso em que $a_{ik} = 0$, foi escolhida uma divisão igualitária para esses grupos com distância zero, mas outra distribuição de valores também é possível, apenas é necessário que a condição $\sum_{k=1}^{K} u_{ik} = 1$ seja verdadeira. Se depois depois dessa atualização ainda exista algum valor negativo para u_{ik} basta recalcular eles segundo o Algoritmo (3.2) com uma pequena alteração no começo em que removemos de V_i os elementos $\{k \mid b_{ik} \geq \gamma_i\}$.

$$u_{ik} = \begin{cases} 0 & \text{se } \gamma_i \leq b_{ik} \\ \frac{\gamma_i - b_{ik}}{a_{ik}} & \text{se } a_{ik} > 0 \\ \frac{1 - \sum_{w \notin Z_i} u_{iw}}{|Z_i|} & \text{se } a_{ik} = 0 \end{cases} \tag{3.6}$$

3.3 Escolha do α

O parâmetro alfa controla importância das restrições *must-link* e *cannot-link* ao mesmo tempo que mantém um equilíbrio com a componente não-supervisionada da função objetivo.

Em [7] Frigui *et al.* discutem que quando o valor do α escolhido for muito pequeno é esperado que a maioria das informações nas restrições acabem sendo ignoradas, e pelo outro lado quando o valor for grande demais ele pode causar uma forte afetação na estrutura dos grupos. Eles propõem um abordagem para calcular o valor automático que é similar a proposto por Grira *et al.* [29]. O α é atualizado em cada iteração do algoritmo com valor igual a razão do termo não-supervisionado pelo supervisionado, mantendo assim uma escala proporcial entre eles. Possíveis problemas dessa abordagem são que o comportamento da função objetivo não apresenta mais minimização garantida visto que o α muda arbitrariamente em cada iteração, e também que os valores usados podem não ser suficientes para garantir que as restrições sejam consideradas.

Para manter esse problema de minimização com convergência garantida durante a execução do algoritmo a escolha do α para o modelo apresentando nesse trabalho será fixo, e para fazer a escolha do seu valor consideramos que as informações das restrições são corretos e que é importante mantê-las na partição *fuzzy* final, ou seja, o α escolhido faz com que o termo da equação (3.7) apresente um valor próximo de zero quando aplicada a partição U encontrada no final do algoritmo.

$$restrictions = \left(\sum_{(l,m) \in \mathcal{M}} \sum_{r=1}^{K} \sum_{\substack{s=1 \\ s \neq r}}^{K} u_{lr} u_{ms} + \sum_{(l,m) \in \mathcal{C}} \sum_{r=1}^{K} u_{lr} u_{mr} \right) \tag{3.7}$$

Uma maneira simples e eficiente de encontrar um valor que siga essas condiçoes desejadas é apresentando no Algoritmo (3.3), ele parte de um valor de α inicial que não seja muito grande, optamos por escolher 1 como padrão e $maxAlpha = 100$, e enquanto que o valor atual de α não satisfaz a condição de que o termo da função objetivo das restrições seja bem próximo de zero ele dobra o valor de α durante a

iteração. O crescimento segue uma escala exponencial até um valor máximo para α, o número de atualizações é de $O(log(maxAlpha))$. Esse algoritmo encontra um valor discreto para o α em até 7 passos, mas nada impede em fazer uma extensão nele para lidar com uma busca por um valor real pois a idéia seria bastante similar. Ele apenas foi simplificado para minimizar o tempo de execução em virtude do grande número de experimentos que foram realizados.

Algoritmo 3.3 Encontrar valor adequado de α

```
α ← 1
found ← False
while found ≠ True do
  partition ← run_clustering()
  restriction ← compute_restriction(partition)
  if α >= maxAlpha or restriction ≤ ε then
    found ← True
  else
    α ← α * 2
  end if
end while
```

3.4 Escolha do β

O parâmetro β é utilizado como termo regularizador [33] para os valores no vetor de relevância. O valor de β é importante para manter o modelo mais estável e usamos ele para evitar que na atualização do vetor de relevância utilizando a equação (3.2) aconteca uma divisão por zero. Esse problema acontece quando a distância do grau de pertinência dos indivíduos em relação aos protótipos de um grupo usando uma matriz de dissimilaridade é zero.

As dissimilaridades em uma matriz pode apresentar valores que não seja heterogêneos e isso faz com o que o fator de relevância pra essa tabela fique muito grande ao mesmo tempo que diminui em grande proporção para as outras, que pode causar uma instabilidade nas operacoes númericas em ponto flutuante, pois no processo de otimização da função objetivo os valores podem apresentar problema de precisão já que esse é o fator de escolha da melhor partição. Quando isso acontece podemos simplicar o valor para a relevância da equação (3.3), usando que o termo da distância intra-cluster igual a zero temos $\lambda_{kj} = \sqrt{8\beta\theta_k}/4\beta$. Podemos então encontrar a ordem do valor esperado para os fatores de relevância através de β, por exemplo, fazendo $\beta = 5 \times 10^{-5}$ temos $\lambda_{kj} = 100\sqrt{\theta_k}$ quando $a_{ik} = 0$.

3.5 Algoritmo

O algoritmo pode ser sumarizado assim:

1. Inicialização

   (a) Faça $t = 0$

   (b) Fixe o número máximo de iterações N_{iter}

   (c) Fixe o número de grupos K

   (d) Fixe o tamanho do conjunto em p para os *medoids*

   (e) Fixe os parâmetros α e β

   (f) Faça $\lambda_k^{(0)} = (\lambda_{k1}^{(0)}, \ldots, \lambda_{kT}^{(0)}) = (1, \ldots, 1)\,(k = 1, \ldots, K)$

   (g) Selecione randomicamente p objetos para formar os *medoids* de cada um dos K clusters formando $G_k^{(0)}\,(k = 1, \ldots, K)$

   (h) Calcule o grau de pertinência segundo a equação abaixo:
   $u_{ik}^{(0)} = \left[\sum_{h=1}^{K}\left(\frac{\sum_{j=1}^{T}\lambda_{kj}^{(0)}\sum_{e\in G_k^{(0)}}d_j(e_i,e)}{\sum_{j=1}^{T}\lambda_{hj}^{(0)}\sum_{e\in G_h^{(0)}}d_j(e_i,e)}\right)\right]^{-1}$

   (i) Calcule $J^{(0)} = \sum_{i=1}^{n}\sum_{k=1}^{K}(u_{ik}^{(0)})^2\sum_{j=1}^{T}\lambda_{kj}^{(0)}\sum_{e\in G_k^{(0)}}d_j(e_i,e) +$
   $\alpha\left(\sum_{(l,m)\in\mathcal{M}}\sum_{r=1}^{K}\sum_{\substack{s=1\\s\neq r}}^{K}u_{lr}^{(0)}u_{ms}^{(0)} + \sum_{(l,m)\in\mathcal{C}}\sum_{r=1}^{K}u_{lr}^{(0)}u_{mr}^{(0)}\right) + \beta\left(\sum_{k=1}^{K}\sum_{j=1}^{T}(\lambda_{kj}^{(0)})^2\right)$

2. Atualização dos *medoids*

   (a) Faça $t = t + 1$

   (b) Fixe a matriz da partiçao *fuzzy* U

   (c) Fixe o vetor de relevância $\Lambda = (\lambda_1, \ldots, \lambda_K)$

   (d) Selecione os melhores p elementos e atualize os *medoids* de cada grupo da forma como foi apresentando no Algoritmo (3.1)

3. Atualização do vetor de relevância

   (a) Fixe a matriz da partiçao *fuzzy* U

   (b) Fixe a vetor de *medoids* $G_k\,(k = 1, \ldots, K)$

   (c) Calcule o novo vetor de relevância segundo a equação (3.2) se $\beta = 0$ ou usando a equação (3.3) caso contrário

4. Atualização da partição *fuzzy*

   (a) Fixe o vetor de relevância $\Lambda = (\lambda_1, \ldots, \lambda_K)$

   (b) Fixe a vetor de *medoids* $G_k\,(k = 1, \ldots, K)$

   (c) Faça a atualização sobrescrevendo os novos valores do grau de pertinência usando o Algoritmo (3.2)

5. Critério de Parada

   (a) Calcule o novo valor do critério $J^{(t)}$ usando a equação (3.1)

   (b) Se $t = N_{iter}$ ou $|J^{(t)} - J^{(t-1)}| \leq \epsilon$ então pare, caso contrário volte ao passo de atualização dos *medoids*.

3.6 Considerações Finais

O algoritmo aqui proposto utiliza um vetor de relevância que é estimado localmente para cada tabela de dissimilaridade, cada uma delas pode ser processada de forma independente entre si, por exemplo, cada uma delas pode ser modelada utilizando diferentes funções de distância. Se contruída uma tabela de dissimilaridade para cada variável é possível identificar o grau de importância de cada uma delas na formação dos grupos.

O conjunto de *medoids* apresenta um resumo de cada um dos grupos e corresponde aos melhores e mais representantivos indivíduos de cada uma delas.

Dada a matriz da partição *fuzzy* também é possível contruir uma partição *hard* dela selecionando para cada indivíduo o grupo que ele apresenta maior valor do grau de pertinência.

A complexidade desse algoritmo é $O(Kn^2T)$ em cada iteração, pois é necessário testar cada indivíduo como um possível membro no conjunto de *medoids* e ao mesmo tempo consultar todas as matrizes de dissimilaridade. Após esse pré-processamento a escolha dos novos *medoids* é feita de forma totalmente independente, basta fazer a seleção dos p objetos que apresentam menor distância segundo o Algoritmo (3.1). Essa parte pode ser ser resolvida eficientemente usando o algoritmo de partição randômica da rotina do *quicksort* em tempo amortizado de $O(K(n + p))$.

4 Experimentos

Esse capítulo apresenta a metodologia que será adotada nos experimentos e descreve os índices externos de performance que serão usados para analisar a eficiência entre eles. Em seguida apresentamos a descrição dos atributos e caracterítiscas das bases de dados que serão utilizadas.

4.1 Metodologia

Para analisar a perfomance do algoritmo proposto nós o testamos em 6 conjuntos de dados reais. A maioria das bases de dados foram extraídas do *University of California Irvine (UCI) Machine Learning Repository* [10], são elas: *Iris*, *Wine*, *Glass*, *Multiple Features (Digits)* e *Yeast*; e a última escolha foi uma base de dados de imagens de flores criada pelo *Robotic Research Group of University of Oxford* (disponível em `http://www.robots.ox.ac.uk/~vgg/data/flowers`) [2]. Para todas as aplicações nós comparamos os resultados do modelo proposto com o *SS-CARD* [30].

Todos os algoritmos foram executado com 100 inicializações para as bases *Iris*, *Wine*, *Glass* que têm um número pequeno de objetos e 10 para as outras. Em cada inicialização o número máximo de iterações foi 50, e o melhor resultado foi escolhido de acordo o valor do critério de adequação.

Visto que os atributos nessas bases são descritos por valores reais foi construída uma matriz de dissimilaridade para cada um deles de acordo com a distância Euclidiana entre todos os pares de objetos. Para que eles tivessem os valores na mesma escala, as tabelas foram normalizadas para que seus valores ficassem na faixa $[0, 1]$, exceto para a base de dados de imagens que já tinham seus atributos processados e normalizados atraves de distâncias mais complexas com valores entre

$[0, 2]$.

Para estudar a influência das restrições no algoritmo selecionamos diferentes quantidade de dados rotulados e a partir dessa informação foram geradas as restriçõs, se dois objetos tivessem o mesmo rótulo eles foram adicionados ao conjunto *must-link*, caso contrário eles foram adicionados ao conjunto *cannot-link*. Foram escolhidas diferentes porcentagens $0\%, 10\%, \ldots, 100\%$ de rótulos conhecidos na geração das restrições, a escolha dos objetos em cada uma delas foi aleatória. Esse processo foi repetido 30 vezes e um intervalo de confiança de nível 95% foi gerado para as medidas de perfomance. Em cada execução os algoritmos utilizaram mesma a partição *fuzzy* inicial.

Para cada conjunto de dados O número de grupos foi o mesmo da partição *a priori*, e o tamanho do conjunto de *medoids* foi de 2 para garantir maior estabilidade na atualização na atualização do vetor de relevância. As comparações e operações entre ponto flutuante usaram $\epsilon = 1 \times 10^{-7}$. O parâmetro de fuzzificação na função objetivo é fixo e com valor $m = 2$, esse mesmo valor será usado no *SS-CARD*.

4.2 Medidas de performance

Afim de comparar o desempenho dos algoritmos selecionamos algumas medidas de perfomance que serão utilizadas para analisar os resultados. A informação disponível sobre os dados reais dizem o rótulo de cada objeto e serão feitas conversões entre essas representações dependendo do índice utilizado. Os rótulos caracterizam uma partição do tipo *hard* e sua representação *fuzzy* é por uma matriz U tal que $u_{ik} = 1$ se o objeto e_i tem como rótulo k e $u_{ik} = 0$ caso contrário. Similarmente dada a partição *fuzzy* representada por U, o rótulo do indivíduo e_i será dado por $k = argmax_{1 \leq h \leq K}\, u_{ih}$.

Os índices do tipo *hard* escolhidos foram o *OERC*, *AR*, *ARI* e *F-Measure*; e os do tipo *fuzzy* foram os índices de *Rand* de *Campello-Frigui* e *Hüllemeier*. As próximas subseções descrevem como cada um deles é calculado.

4.2.1 Matriz de contingência

A matriz de contingência é muito utilizada para organizar e exibir informações que mostram o desempenho de algum algoritmo de classifição de dados, e também é comumente usada em diversas métricas de precisão que agregam dados dessa matriz no cálculo de índices de similaridade entre partições [36].

Tabela 4.1: A Matriz de Contingência, $a_{ij} = |P_i \cap Q_j|$

Classe/Grupo	Q_1	$\cdots$	Q_j	$\cdots$	Q_l	$\sum$
P_1	a_{11}	$\cdots$	a_{1j}	$\cdots$	a_{1l}	$a_{1.} = \sum_j a_{1j}$
$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$
P_i	a_{i1}	$\cdots$	a_{ij}	$\cdots$	a_{il}	$a_{i.} = \sum_j a_{ij}$
$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$
P_k	a_{k1}	$\cdots$	a_{kj}	$\cdots$	a_{kl}	$a_{k.} = \sum_j a_{kj}$
$\sum$	$a_{.1} = \sum_i a_{i1}$	$\cdots$	$a_{.j} = \sum_i a_{ij}$	$\cdots$	$a_{.l} = \sum_i a_{il}$	$n = \sum_{ij} a_{ij}$

Seja $E = \{e_1, \ldots, e_n\}$ um conjunto de n objetos e sejam $P = \{P_1, \ldots, P_k\}$ e $Q = \{Q_1, \ldots, Q_l\}$ duas partições *hard* desse conjunto, ou seja, $P_i \cap P_{i'} = \emptyset = Q_j \cap Q'_j$ para todo $1 \leq i \neq i' \leq k$ e $1 \leq j \neq j' \leq l$, e $\bigcup_{i=1}^{k} P_i = E = \bigcup_{j=1}^{l} Q_j$. A matriz de contingência é construída através da intersecção entre os elementos de P e Q resumidos na forma de uma tabela $A = [a_{ij}]_{j=1,\ldots,l}^{i=1,\ldots,k}$ conforme ilustrado na Tabela (4.1). As k linhas estão associadas as classes de P e as l colunas estão associadas aos grupos de Q, onde a_{ij} representa o número de objetos comum entre a classe P_i e o grupo Q_j.

4.2.2 Taxa de erro global de classificação

O OERC (*overall error rate of classification*) mede a habilidade de um algoritmo de agrupamento de dados de encontrar uma classe a priori presente em um conjunto de dados [37]. Dada as partições *hard* P e Q seu valor é calculado de acordo com a equação (4.1).

$$OERC(P, Q) = \sum_{j=1}^{l} \frac{a_{.j}}{n} \left(1 - max_{1 \leq i \leq k} \frac{a_{ij}}{a_{.j}}\right) \tag{4.1}$$

4.2.3 Taxa de associação

A taxa de associação é baseada no OERC considerando o caso especial em que o numero de grupos é igual ao numero de classes do algoritmo de classificação, ela produz uma atribuição bijetiva entre uma classe *a priori* e um grupo. Esse pareamento é escolhido de forma a maximimizar a soma das intersecções entre as partições *hard*, essa correspondência é equivalente ao valor de soma máxima dos elementos na diagonal dentre todas as possíveis $k!$ permutações das linhas na matriz de contigência.

O AR (*accuracy rate*) corresponde ao problema de maximização na equação (4.2), as restrições são usadas para garantir que cada linha da matriz (classe *a priori*) está associada a uma única coluna (grupo).

$$AR(P,Q) = \frac{\sum_{i=1}^{k}\sum_{j=1}^{l} a_{ij}x_{ij}}{n} \rightarrow \text{Máximo} \tag{4.2}$$

sujeito a

$$\sum_{i=1}^{k} x_{il} = 1, \quad \sum_{j=1}^{l} x_{kj} = 1 \quad x_{ij} \in \{0,1\}$$

Esse problema pode ser representando como uma rede de fluxos e custos num grafo bipartido, onde as linhas e colunas da matriz são associados com $l+k$ vértices que têm $k \times l$ arestas entre eles com capacidade 1 e com custo igual aos valores na matriz de contingência. O resultado do custo no fluxo máximo de custo máximo [35] nesse grafo é exatamente o valor do numerador do AR.

4.2.4 F-measure

A *F-measure* é uma métrica comumente utilizada para analisar a perfomance de algum algoritmo de classificação de dados, ela foi inspirada de uma generalização da *F1-score* para múltiplas classes.

O *F1-score* [38] é calculada através da média harmônica entre a *precision* e do *recall*. *Precision* é a razão do número de verdadeiros positivos pelo número total de predições que foram positivas, e similarmente o *recall* é a razão do número de verdadeiros positivos pelo número de predições que realmente são positivas. O *F1-score* é calculado segundo a equação (4.3).

$$F1(P_i,Q_j) = 2\left(\frac{precision(P_i,Q_j) \times recall(P_i,Q_j)}{precision(P_i,Q_j) + recall(P_i,Q_j)}\right) \tag{4.3}$$

onde

$$precision(P_i,Q_j) = \frac{a_{ij}}{a_{.j}} \tag{4.4}$$

e

$$recall(P_i,Q_j) = \frac{a_{ij}}{a_{i.}} \tag{4.5}$$

A *F-measure* faz uma generalização do *F1-score* para comparação de desempenho entre partições com um número maior de classes. Seu valor é calculado como apresentado na equação (4.6), onde é feita a associação entre os pares das partições que tem maior similaridade.

$$F\text{-}measure(P,Q) = \left(\frac{1}{n}\right) \times \sum_{i=1}^{k} (a_{i.} \times max_{1 \leq j \leq l} F1(P_i, Q_j)) \tag{4.6}$$

4.2.5 Coeficiente de Rand

O índice de *Rand* [39] propõe um modo de medir a similaridade entre duas partições *hard* usando como critério a forma com que estão organizados os pares de objetos de E. É feita uma distinção entre quatro possíveis categorias, e seu valor é calculado de acordo com a equação 4.7, onde o conjunto N_{00} contém as tuplas de objetos que não estão pareadas em P nem em Q, os conjuntos N_{01} e N_{10} contém as tuplas que estão em P ou em Q exclusivamente, e N_{11} contém as tuplas que estão presentes em ambos P e Q. Apenas os tipos N_{00} e N_{11} são interpretados como valor de conformidade entre as partições, enquanto que as tuplas N_{01} e N_{10} representam discordância. O índice de *Rand* é então definido pela razão do número de pares concordantes pelo número total de pares.

$$RI(P,Q) = \frac{|N_{00} \cup N_{11}|}{|N_{00} \cup N_{01} \cup N_{10} \cup N_{11}|} = \frac{|N_{00}| + |N_{11}|}{\binom{n}{2}} \tag{4.7}$$

onde

$$\begin{aligned}
N_{00} &= \{(e_i, e_j) \mid 1 \leq i < j \leq n \wedge \nexists i'(e_i, e_j \in P_{i'}) \wedge \nexists j'(e_i, e_j \in Q_{j'})\} \\
N_{01} &= \{(e_i, e_j) \mid 1 \leq i < j \leq n \wedge \nexists i'(e_i, e_j \in P_{i'}) \wedge \exists j'(e_i, e_j \in Q_{j'})\} \\
N_{10} &= \{(e_i, e_j) \mid 1 \leq i < j \leq n \wedge \exists i'(e_i, e_j \in P_{i'}) \wedge \nexists j'(e_i, e_j \in Q_{j'})\} \\
N_{11} &= \{(e_i, e_j) \mid 1 \leq i < j \leq n \wedge \exists i'(e_i, e_j \in P_{i'}) \wedge \exists j'(e_i, e_j \in Q_{j'})\}
\end{aligned}$$

Em [40], Hubert e Arabie apresentam uma correção para o índice de *Rand* a fim de que seu valor fique constante para a comparação entre duas partições randômicas. Eles assumiram como modelo de randomização distribuição hipergeométrica construído na tabela de contingência, que significa o caso que as partições P e Q são escolhidas ao acaso. O valor do índice ajustado de Rand é então calculado segundo a equação (4.8), seu valor é limitado por 1 quando as partições são equivalentes e por valores próximos de zero quando a classificação foi gerada ao acaso.

$$ARI(P,Q) = \frac{\sum_{i,j} \binom{a_{ij}}{2} - \left[\sum_i \binom{a_{i.}}{2} \sum_j \binom{a_{.j}}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_{i.}}{2} + \sum_j \binom{a_{.j}}{2}\right] - \left[\sum_i \binom{a_{i.}}{2} \sum_j \binom{a_{.j}}{2}\right] / \binom{n}{2}} \tag{4.8}$$

4.2.6 Coeficiente de Rand Fuzzy

Os índices até então apenas lidam com a comparação de partições do tipo *hard* e para tratar o caso de avaliar o caso *fuzzy* foram propostas generalizações do índice de *Rand* por Campello e Frigui [7] e por Hüllermeier *et al.* [41]. Diferentemente das outras métricas citadas esse índice não usa a matriz de contigência.

4.2.6.1 Frigui

Em [7], Frigui *et al.* propõem uma abordagem para o cálculo do índice de *rand fuzzy* onde dada duas matrizes de grau de pertinência $U^{(1)}$ e $U^{(2)}$. Esse índice se baseia no trabalho de Campello [42] e pode ser visto como um caso especial dele. Inicialmente é calculado as matrizes coincidências $\Psi^{(1)}$ e $\Psi^{(2)}$ para as duas matrizes $U^{(1)}$ e $U^{(2)}$, onde $\Psi = [\psi_{jk}]\,(1 \leq j,k \leq n)$ e $\psi_{jk} = \sum_{i=1}^{K} u_{ij}u_{ik}$. O índice de *rand fuzzy* Frigui é então calculado segundo a equação (4.9).

$$RCF(U^{(1)}, U^{(2)}) = \frac{N_{SS} + N_{DD}}{N_{SS} + N_{SD} + N_{DS} + N_{DD}} \tag{4.9}$$

onde

$$\begin{aligned}
N_{SS}(\Psi^{(1)}, \Psi^{(2)}) &= \sum_{j=2}^{n}\sum_{k=1}^{j-1} \psi_{jk}^{(1)}\psi_{jk}^{(2)} \\
N_{SD}(\Psi^{(1)}, \Psi^{(2)}) &= \sum_{j=2}^{n}\sum_{k=1}^{j-1} \psi_{jk}^{(1)}(1 - \psi_{jk}^{(2)}) \\
N_{DS}(\Psi^{(1)}, \Psi^{(2)}) &= \sum_{j=2}^{n}\sum_{k=1}^{j-1} (1 - \psi_{jk}^{(1)})\psi_{jk}^{(2)} \\
N_{DD}(\Psi^{(1)}, \Psi^{(2)}) &= \sum_{j=2}^{n}\sum_{k=1}^{j-1} (1 - \psi_{jk}^{(1)})(1 - \psi_{jk}^{(2)})
\end{aligned}$$

4.2.6.2 Hüllemeier

Em [41], Hüllemeier *et al.* discutem os princípios por trás da métrica de Campello e do Frigui e apresentam os principais problemas dela. Eles argumentam que o índice de Campello não é uma métrica apropriada, com uma falha na propriedade de reflexividade, e mostram um exemplo em que o valor do índice na comparação de duas partições *fuzzy* idênticas não é 1.

Eles propõem uma nova abordagem para o cálculo do índice de *rand fuzzy*, onde eles mantém as propriedades métricas (relação de identidade, simetria, reflexividade e desigualdade triangular). O valor é calculado segundo a equação (4.10) onde $M_x(y, z)$ é uma função métrica e que $U^{(1)}$ e $U^{(2)}$ são as matrizes das partições *fuzzy*.

$$RH(U^{(1)}, U^{(2)}) = \frac{\sum_{(e_i,e_j)\in E} |M_{U^{(1)}}(i,j) - M_{U^{(2)}}(i,j)|}{\binom{n}{2}} \tag{4.10}$$

A função métrica que usamos foi a distância métrica da distância de *Manhattan* como mostrado na equação (4.11).

$$M_U(i,j) = \frac{\sum_{k=1}^{K} |u_{ik} - u_{jk}|}{2} \tag{4.11}$$

4.3 Bases de Dados

A escolha desses conjunto de dados foi balanceada com base no número de objetos, com uma parte com um número relativamente pequeno de objetos com respectivamente de $150, 178, 214$ para as bases *Iris*, *Wine* e *Glass*; e bases com um número maior de objetos com respectivamente $1484, 2000, 1360$ para as bases *Multiple features (Mfeat)*, *Yeast* e *Flower*. A tabela (4.2) sumariza as principais informações sobre todos os conjuntos de dados.

Tabela 4.2: Estatísticas gerais das bases de dados

Nome	K	n	T	Origem
Iris	3	150	4	UCI
Wine	3	178	13	UCI
Glass	6	214	9	UCI
Yeast	10	1484	8	UCI
Multiple features	10	2000	6	UCI
Flower	17	1360	4	UOX

4.3.1 Iris

Essa base de dados consiste de 50 instâncias para cada uma das três espécies de plantas iris: iris setosa, iris versicolour e iris virginica; num total de 150 indivíduos. A primeira classe é linearmente separável das outras duas; as duas últimas não são linearmente separáveis entre si. Cada objeto é descrito por quatro atributos:

(1) comprimento da sépala, (2) largura da sépala, (3) comprimento da pétala e (4) largura da pétala, em centímetros. As dimensões das pétalas e das sépalas estão entre os melhores atributos para a categorização das três espécies [43].

4.3.2 Wine

Essa base de dados contém 178 amostras de três tipos de vinhos que vem da Itália, mas derivados de cultivares diferentes [10]. As classes tem tamanhos diferentes com 59,71 e 48 instancias. Cada vinho é descrito por 13 atributos com valores reais que representam a quantidade de seus componentes químicos. Os atributos são: (1) *alcohol*, (2) *malic acid*, (3) *ash*, (4) *alkalinity of ash*, (5) *magnesium*, (6) *total phenols*, (7) *flavonoids*, (8) *non-flavonoid phenols*, (9) *proanthocyanins*, (10) *colour intensity*, (11) *hue*, (12) *OD280/OD315 of diluted wines*, e (13) *proline*.

4.3.3 Glass

Essa base de dados e formada por 214 instâncias distribuídas em seis classes. O estudo de classificação de tipos de vidros foi motivado por investigação criminalística que pode usar algum vidro na cena do crime como evidência se ele puder ser corretamente identificado. Nove atributos descrevem cada um deles através da descrição do material: (1) índice refrativo, (2) Na, (3) Mg, (4) Al, (5) Si, (6) K, (7) Ca, (8) Ba e (9) Fe.

A distribuição dos tipos de vidros é mostrado abaixo:

- 163 vidros de janela
  - 87 flotado
    - 70 janelas de edifícios
    - 17 janelas de veículos
  - 76 não flotado
    - 76 janelas de edifícios
    - 0 janelas de veículos
- 51 não vidros de janela
  - 13 containers
  - 9 talheres
  - 29 faróis

4.3.4 Multiple features

Essa base de dados consiste de características dos numerais ('0' – '9') escritos a mão que foram extraídos de uma coleção de mapas de serviços públicos holandenses [44]. Ele contém duzentos padrões por classe (totalizando 2000 padrões) que foram digitalizados na forma de images binárias.

Esses dígitos são representados em termo de seis diferentes visões independentes, em cada uma delas cada dígito é então representado por um vetor. O resumo de cadas uma delas é mostrada abaixo:

- 76 coeficientes de *Fourier* da formas dos caracteres
- 216 correlações do perfil
- 64 coeficientes de *Karhunen-Loève*
- 240 média dos pixels em janelas 2×3
- 47 momentos de *Zernike*
- 6 características morfológicas

4.3.5 Yeast

Essa base de dados apresenta 1484 instâncias descritas por oito atributos. Os atributos sao usados para a localização de padrões dentro das células de levedura [10], são esses: (1) *mcg: McGeoch's method for signal sequence recognition*, (2) *gvh: von Heijne's method for signal sequence recognition*, (3) *alm: Score of the ALOM membrane spanning region prediction program*, (4) *mit: Score of discriminant analysis of the amino acid content of the N-terminal region (20 residues long) of mitochondrial and non-mitochondrial proteins*, (5) *erl: Presence of "HDEL"substring (thought to act as a signal for retention in the endoplasmic reticulum lumen)*, (6) *pox: Peroxisomal targeting signal in the C-terminus*, (7) *vac: Score of discriminant analysis of the amino acid content of vacuolar and extracellular proteins*, e (8) *nuc: Score of discriminant analysis of nuclear localization signals of nuclear and non-nuclear proteins.*

A distribuição das classes apresenta um grande fator de desbalanceamento como pode ser analisado na tabela 4.3, que mostra elas ordenadas pela frequência na ordem descrescente.

Tabela 4.3: Distribuição das classes em Yeast

Classe (localização)	Frequência
CYT (cytosolic or cytoskeletal)	463
NUC (nuclear)	429
MIT (mitochondrial)	244
ME3 (membrane protein, no N-terminal signal)	163
ME2 (membrane protein, uncleaved signal)	51
ME1 (membrane protein, cleaved signal)	44
EXC (extracellular)	35
VAC (vacuolar)	30
POX (peroxisomal)	20
ERL (endoplasmic reticulum lumen)	5

4.3.6 Flower

Essa base de dados contém 1360 imagens de 17 espécies de flores com 80 imagem para cada classe. As imagens foram coletadas através de buscas na internet e fotografias. As flores representam algumas das mais comuns flores no Reino Unido e suas categorias são: *Buttercup*, *Colts'Foot*, *Daffodil*, *Daisy*, *Dandelion*, *Fritillary*, *Iris*, *Pansy*, *Sunflower*, *Windflower*, *Snowdrop*, *Lily Valley*, *Bluebell*, *Crocus*, *Tigerlily*, *Tulip* e *Cowslip*.

Essas imagens formam conjunto de dados bastante desafiador porque elas apresentam uma grande escala, variações de perspectivas e iluminação, e esse fator favorece numa deformação entre imagens da mesma classe para que eles apresentem uma grande variação entre si, ao mesmo tempo que promove uma similaridade com imagens de outras classes.

Em [2], Nilsback e Zisserman criaram um vocabulário para esse conjunto de dados baseado e apresentam mais informacoes sobre as imagens. A figura (4.1) mostra exemplos de cada uma das classes [2].

Diferentes características foram escolhidas para descrever as propriedades de cada flor. Os atributos usados foram: (1) cor (HSV), (2) o histograma de orientações gradiente (HOG), e o amostras do SIFT no (3) foreground region e na (4) fronteira. As distâncias foram calculadas como descrito em [45].

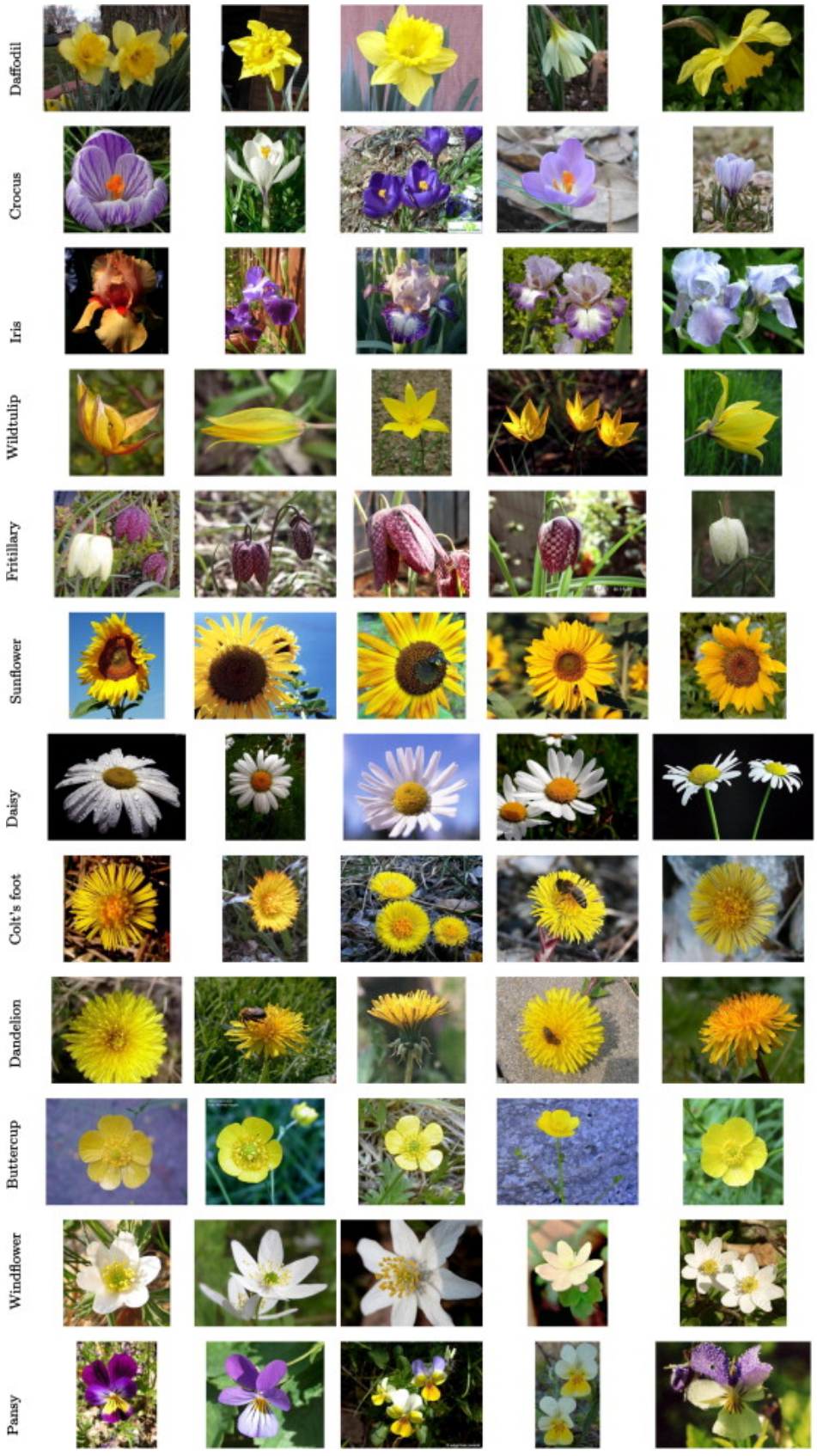


Figura 4.1: Categorias dos tipos de flores (Nilsback e Zisserman [2])

5
Resultados

Nesse capítulo descrevemos os resultados sobre as bases de dados apresentadas no capítulo anterior segundo a metodologia apresentada nele. Os modelos usados nos experimentos foram o SS-CARD e o algoritmo proposto nesse trabalho segundo duas escolhas do α. Depois é apresentado as estatísticas gerais dos resultados em relação a todas as bases de dados, em seguida mostramos mais resultados que estão organizados por base de dados e com figuras e tabelas para melhor visualização e análise dos modelos.

5.1 Desempenho dos Algoritmos

As Figuras (5.1 - 5.6) apresentam um sumário geral dos resultados para todas as bases de dados nos experimentos. Os modelos utilizado foram o algoritmo proposto nesse trabalho (SS-CLAMP) que faz a escolha do α automática, para fins de investigação da influência do α também será utilizado o SS-CLAMP com a atualização do α segundo a metodologia do Frigui *et al.* [30] que será denotado nesse texto por SS-CLAMP', e o algoritmo SS-CARD [30].

Os resultados exibem o intervalo de confiança de nível 95% da média dos indíces de perfomance: *oerc*, *accuracy*, *rand*, *f-measure*, *campello* e *hüllemeier*. As execuções utilizaram diferentes porcentagens de rótulos disponíveis para a construção das restrições, de 0% a 100% (com incremento de 10%). Mais informações sobre os resultados de cada base são discutidos com maiores detalhes em sua correspondente seção ainda nesse capítulo.

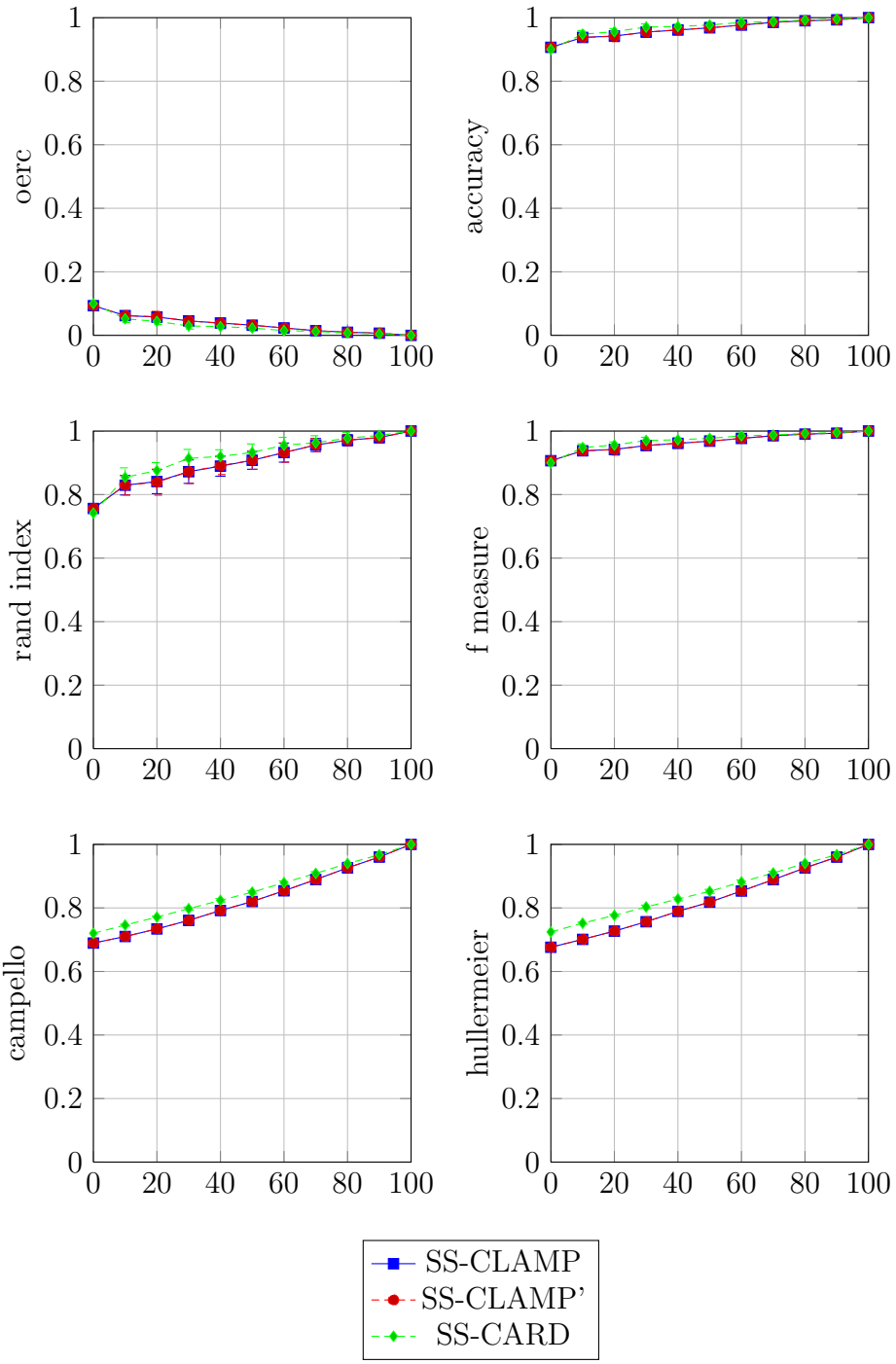


Figura 5.1: Iris: intervalo de confiança de nível 95% para o valor médio das medidas de perfomance

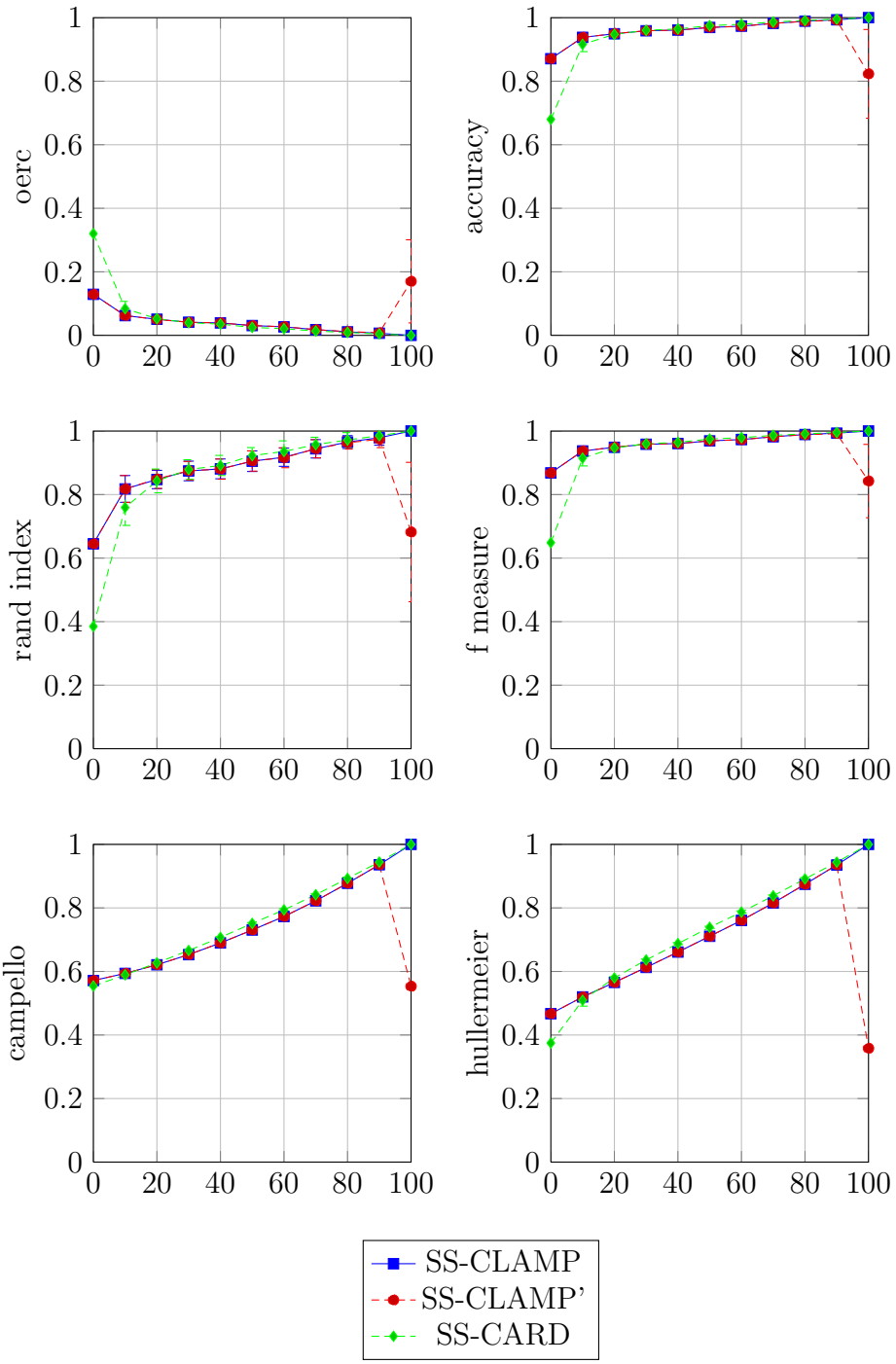


Figura 5.2: Wine: intervalo de confiança de nível 95% para o valor médio das medidas de perfomance

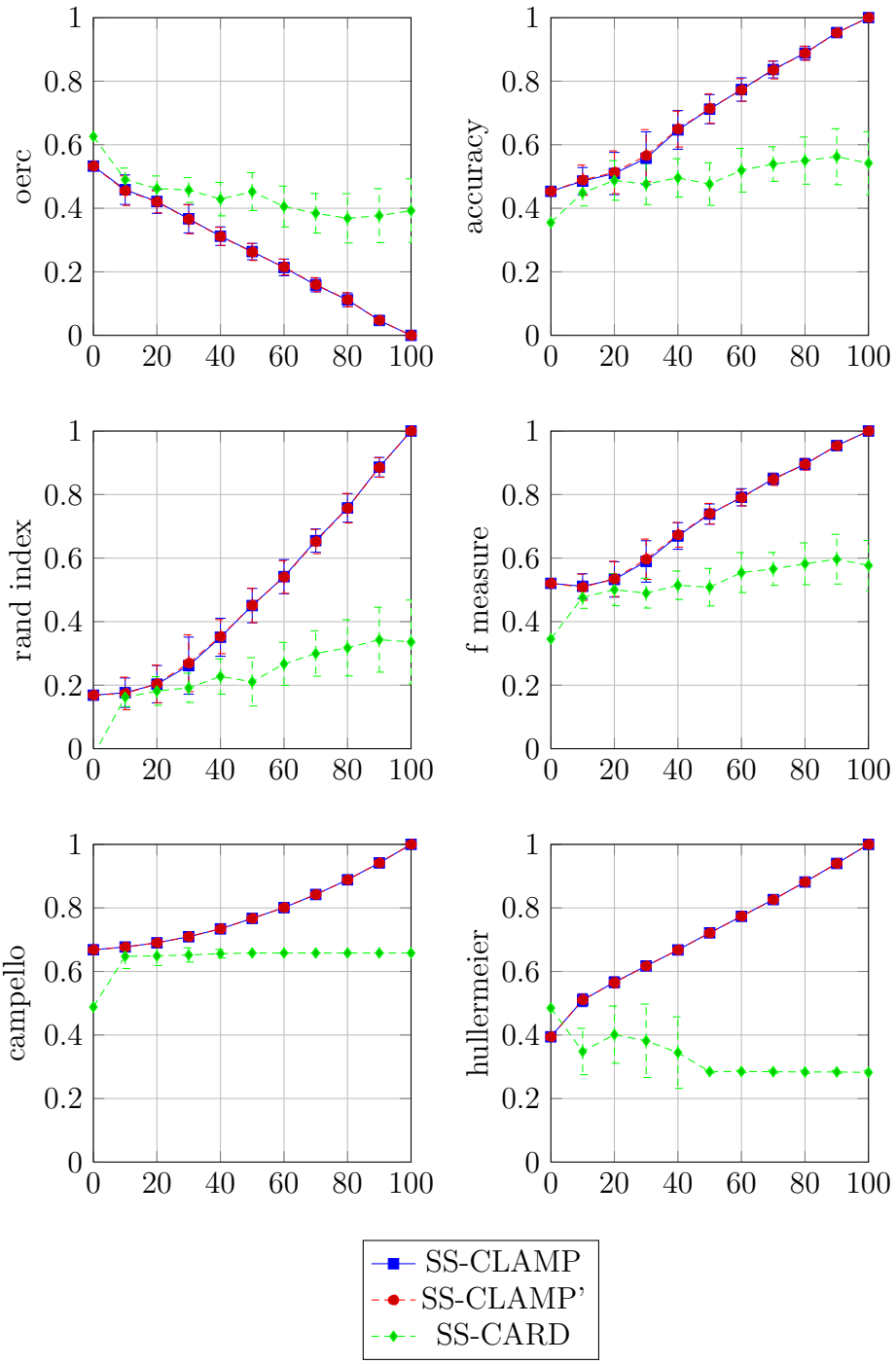


Figura 5.3: Glass: intervalo de confiança de nível 95% para o valor médio das medidas de perfomance

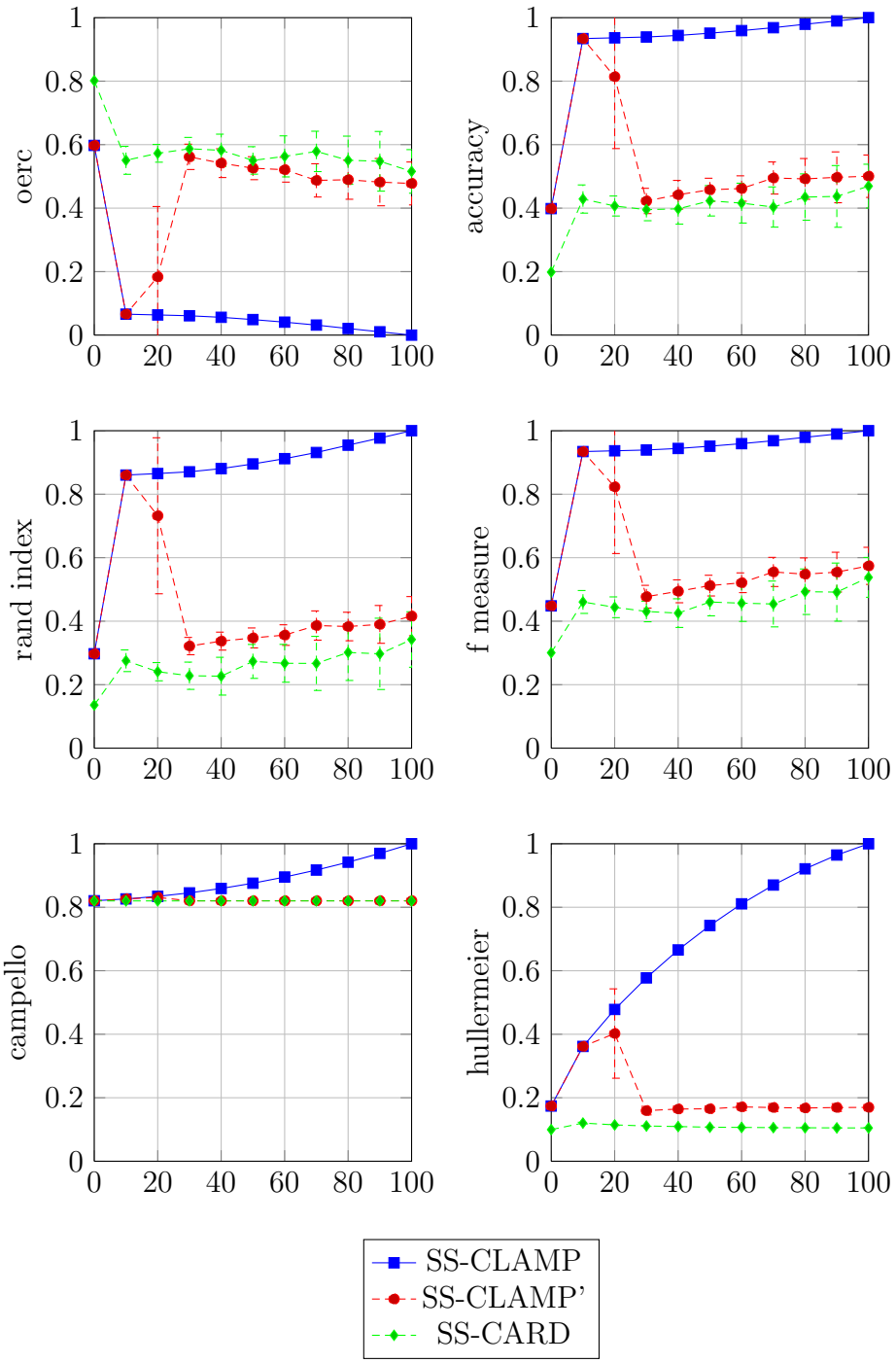


Figura 5.4: Mfeat: intervalo de confiança de nível 95% para o valor médio das medidas de perfomance

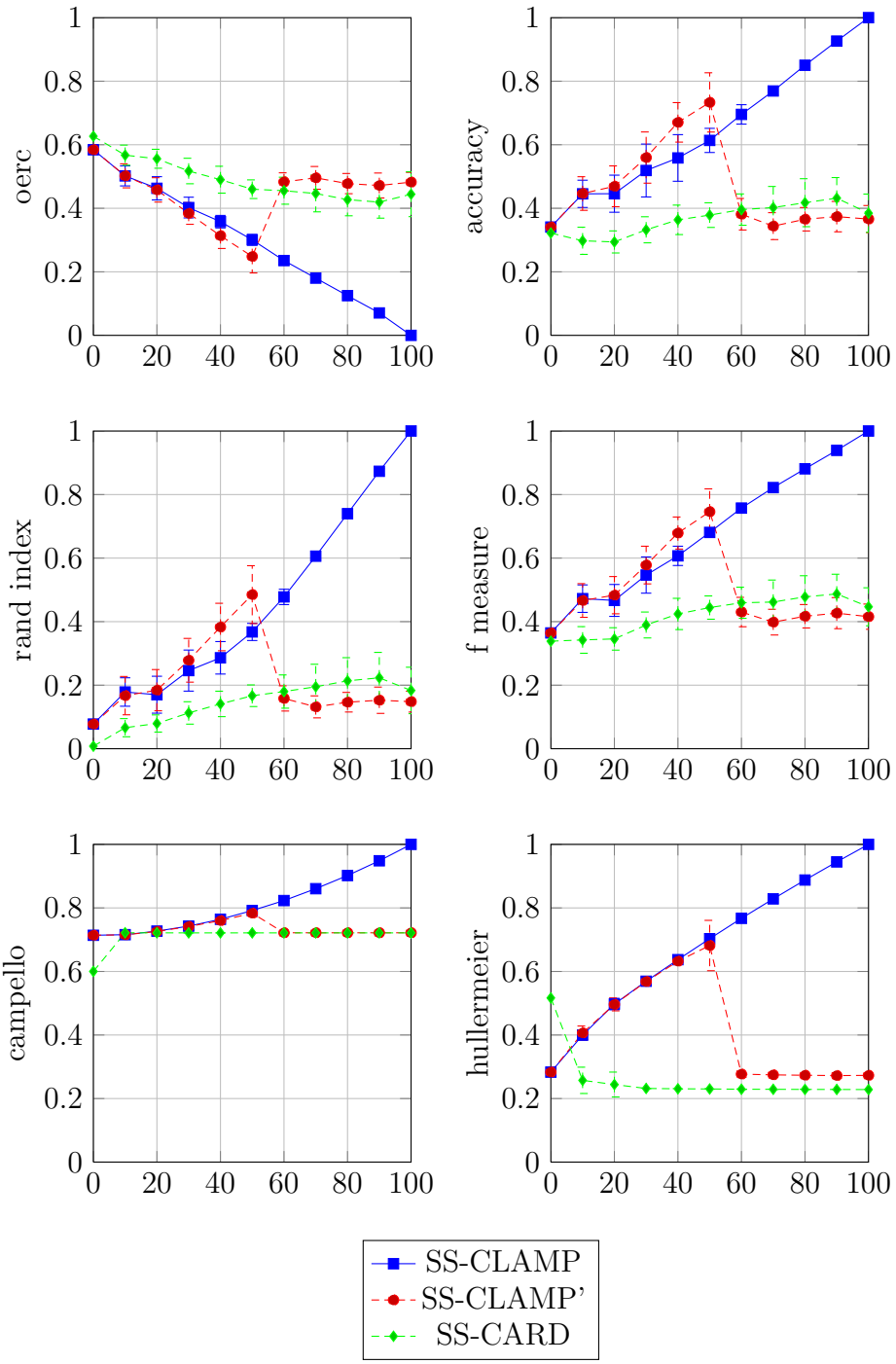


Figura 5.5: Yeast: intervalo de confiança de nível 95% para o valor médio das medidas de perfomance

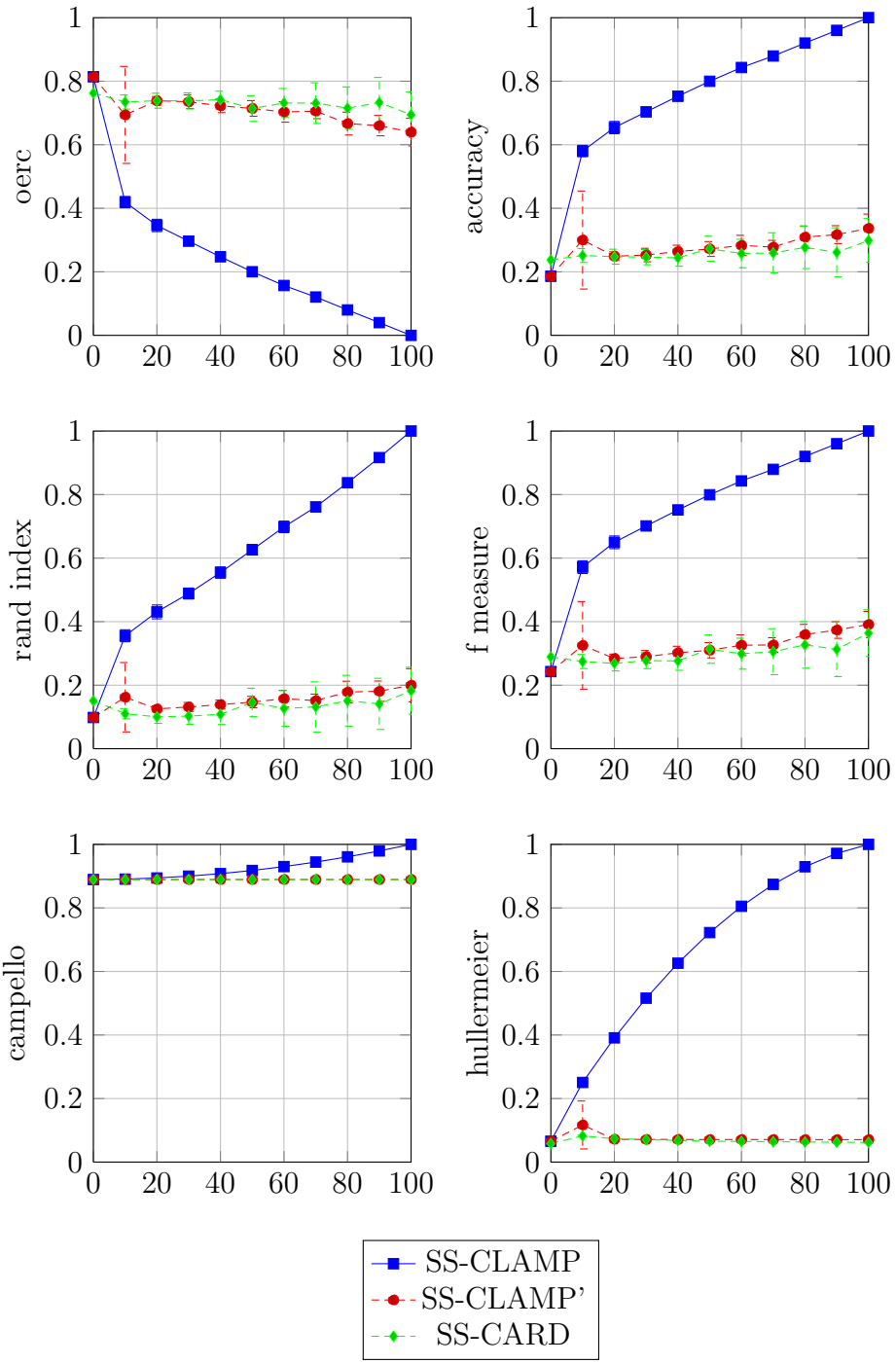


Figura 5.6: Flower: intervalo de confiança de nível 95% para o valor médio das medidas de perfomance

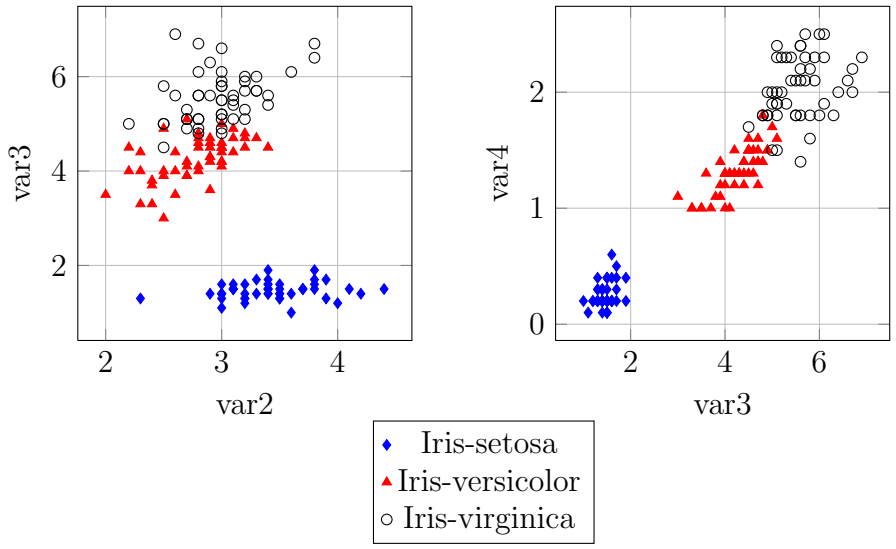


Figura 5.7: Relação entre as variáveis do Iris

5.2 Iris

A Figura 5.1 mostra o desempenho dos algoritmos *SS-CLAMP*, *SS-CLAMP'* e *SS-CARD* nessa base de dados utilizando diferentes porcentagem de dados rotulados. O comportamento segundo os índices foram basicamente o mesmo em relação a performance entre os modelos, com valores similares entre todos eles exceto para o índice *rand fuzzy* de hüllermeier que tem um valor levemente melhor para o *SS-CARD* nas primeiras porcentagem de dados rotulados, com 0% dos dados com $(0.676, 0.676)$ e $(0.724, 0.724)$ e para 10% com $(0.698, 0.704)$ e $(0.749, 0.755)$ para o *SS-CLAMP* e o *SS-CARD* respectivamente.

O valor automático do α encontrado nas execuções do *SS-CLAMP* foi de 1, que já foi suficiente para que as restrições fossem todas satisfeitas no resultado final.

A incorporação do uso de restrições para os modelos não apresentou muitas mudanças, apresentando crescimento suavemente linear, pois eles já foram capazes de produzir uma boa classificação sem o uso da informação sobre os rótulos, que correspondem ao modelo *MFCM-dd-RWL* para o *SS-CLAMP* e *CARD-R* para o *SS-CARD*.

A figura (5.7) mostra o conjunto de dados *Iris* segundo a relação entre pares de suas variáveis, na subfigura a direita é exibido entre as variáveis (2) e (3), e na subfigura a esquerda a relação entre as variáveis (3) e (4). Pode ser observado que a associação entre as variaveis (3) e (4) colabora mais para a formação do grupo da

Tabela 5.1: Matriz de confusão do Iris com 10% de dados rotulados

Grupo/Classe	1	2	3
1	50	0	0
2	0	7	47
3	0	43	3

Tabela 5.2: Vetor de relevância do Iris com 10% de dados rotulados

Grupo/Tabela	1	2	3	4
1	0.767	0.513	**1.174**	**1.484**
2	0.854	**1.198**	**1.167**	0.838
3	0.864	1.029	**1.086**	**1.035**

Iris-Setosa do que entre as variáveis (2) e (3). Para uma execução do *SS-CLAMP* com 10% de dados rotulados a tabela (5.1) mostra a matriz de confusão obtida e na tabela (5.2) podemos ver os valores das relevâncias para uma execução, e que a ordem de importância encontrado mostra um resultado favorável para a formação dos grupos.

5.3 Wine

A Figura 5.2 mostra o desempenho dos algoritmos *SS-CLAMP*, *SS-CLAMP'* e *SS-CARD* nessa base de dados utilizando diferentes porcentagem de dados rotulados. O comportamento segundo os índices foram próximos entre os modelos, e o algoritmo que melhorou mais utilizando as restrições foi o *SS-CARD* na transição de 0% e 10%, mas ainda assim apresentou uma perfomance pouco inferior ao *SS-CLAMP* com 10% de dados rotulados. O desempenho para o *SS-CLAMP'* foi degenerado para o caso onde a informação completa sobre os rótulos foi fornecida, muito embora esse caso seja bastante irreal ele mostra que a escolha do α proposta por Frigui *et al.* [30] pode ocasionar em situações que as restrições acabem não sendo respeitadas. O seu resultado final foi inferior a todos apresentados pelo *SS-CLAMP*, até mesmo sem usar toda informação mas tem um valor de α apropriado, de acordo com todos os índices de perfomance e até mesmo para o caso que não contém nenhuma restrição que corresponde ao *MFCM-dd-RWL-P*.

O valor automático do α encontrado para a maioria das inicializações das execuções do SS-CLAMP foi de 1 e algumas escolheram 2. A tabela (5.6) mostra o desempenho em relação as duas escolhas de α para o SS-CLAMP numa inicialização

Tabela 5.3: Wine: grupo 1

Tabela	Relevância
2	1.579
4	1.279
12	1.277
8	1.233
6	1.193
3	1.132
5	1.066
0	0.921
10	0.867
11	0.796
7	0.763
1	0.671
9	0.671

Tabela 5.4: Wine: grupo 2

Tabela	Relevância
4	1.289
9	1.251
1	1.184
3	1.136
8	1.059
10	1.047
6	0.994
5	0.980
2	0.942
12	0.916
0	0.895
7	0.893
11	0.619

Tabela 5.5: Wine: grupo 3

Tabela	Relevância
10	1.277
1	1.265
4	1.150
9	1.147
6	1.120
2	1.100
8	1.043
3	0.931
5	0.924
0	0.900
7	0.883
11	0.808
12	0.661

em que houve mudança do α segundo o Algoritmo (3.3). Com um fator de maior importância para as restrições o algoritmo convergiu mais rápido e também não apresentou um descrescimento muito suave do seu valor que foi para zero em apenas duas iterações desde a inicialização. Ainda mais sobre esse resultado, as Tabela 5.3, 5.4 e 5.5 mostram a escolha dos pesos de relevância em que pode ser observado que a ordem de importância delas varia entre os grupos, por exemplo, a matriz de dissimilaridade 12 e a menos importante para a formação do grupo 3 ao mesmo tempo que está entre as mais importantes para a formação do grupo 1. Isso reflete que a composição química dos tipos de vinhos contem uma maior proporção de algum elemento químico que os outros.

5.4 Glass

A Figura 5.3 mostra o desempenho dos algoritmos *SS-CLAMP*, *SS-CLAMP'* e *SS-CARD* nessa base de dados utilizando diferentes porcentagem de dados rotulados. Os resultados foram equivalentes para o *SS-CLAMP* e *SS-CLAMP'* de acordo com os índices. O *SS-CARD* apresentou um crescimento favorável com saltos no seu desempenho de 0% até 20% e depois disso não apresentou melhoras mesmo com mais informação supervisionada.

Na execução do *SS-CLAMP* esses dados apresentam uma grande homogeinidade de um grupo em relação a uma variável e isso gerou instabilidade no processo de

Tabela 5.6: Wine: impacto do α nas iterações

	iter	J	Restrição	OERC	AR	ARI	F	RCF	RH
$\alpha = 1$	0	463.707	126.530	0.354	0.590	0.369	0.679	0.553	0.362
	1	390.889	111.901	0.410	0.590	0.313	0.654	0.553	0.354
	2	321.419	20.994	0.320	0.680	0.316	0.675	0.560	0.397
	3	319.698	20.984	0.225	0.775	0.460	0.775	0.562	0.409
	4	319.674	20.958	0.225	0.775	0.465	0.775	0.562	0.409
	5	319.661	20.889	0.219	0.781	0.474	0.781	0.562	0.409
	6	319.624	20.706	0.219	0.781	0.474	0.781	0.562	0.409
	7	319.529	20.223	0.219	0.781	0.474	0.781	0.562	0.410
	8	319.276	18.946	0.213	0.787	0.483	0.786	0.562	0.410
	9	318.608	15.571	0.208	0.792	0.493	0.792	0.562	0.411
	10	314.058	5.808	0.331	0.635	0.376	0.677	0.566	0.431
	11	307.548	0.814	0.281	0.719	0.447	0.702	0.571	0.451
	12	303.748	0.240	0.225	0.775	0.488	0.767	0.575	0.468
	13	300.392	0.203	0.169	0.831	0.573	0.829	0.580	0.486
	14	296.984	0.201	0.140	0.860	0.622	0.857	0.584	0.502
	15	295.147	0.191	0.107	0.893	0.699	0.892	0.586	0.505
	16	295.129	0.189	0.112	0.888	0.685	0.886	0.586	0.505
	17	295.128	0.189	0.112	0.888	0.685	0.886	0.586	0.505
	18	295.128	0.189	0.112	0.888	0.685	0.886	0.586	0.505
	19	295.128	0.189	0.112	0.888	0.685	0.886	0.586	0.505
	20	295.128	0.189	0.112	0.888	0.685	0.886	0.586	0.505
	21	295.128	0.189	0.112	0.888	0.685	0.886	0.586	0.505
$\alpha = 2$	0	590.237	126.530	0.354	0.590	0.369	0.679	0.553	0.362
	1	416.975	59.755	0.421	0.579	0.307	0.649	0.555	0.371
	2	319.180	1.168	0.247	0.753	0.416	0.752	0.561	0.404
	3	310.639	0.000	0.129	0.871	0.666	0.870	0.569	0.440
	4	304.140	0.000	0.067	0.933	0.804	0.932	0.579	0.475
	5	293.604	0.000	0.045	0.955	0.864	0.955	0.590	0.512
	6	293.151	0.000	0.039	0.961	0.882	0.960	0.592	0.516
	7	293.143	0.000	0.039	0.961	0.882	0.960	0.592	0.517
	8	293.143	0.000	0.039	0.961	0.882	0.960	0.592	0.517
	9	293.143	0.000	0.039	0.961	0.882	0.960	0.592	0.517
	10	293.143	0.000	0.039	0.961	0.882	0.960	0.592	0.517
	11	293.143	0.000	0.039	0.961	0.882	0.960	0.592	0.517
	12	293.143	0.000	0.039	0.961	0.882	0.960	0.592	0.517

Tabela 5.7: Glass: vetor de relevância com 10% de rótulos e $\beta = 5 \times 10^{-5}$

Tabela/Grupo	1	2	3	4	5	6
1	0.424527	0.745479	0.969374	1.13186	0.474803	1.1586
2	0.690676	0.799201	0.866771	1.0718	0.680463	1.00822
3	0.40213	0.991969	0.674251	0.499646	0.367528	0.36264
4	0.473061	0.929978	0.683744	0.875644	0.524719	0.820815
5	0.46713	0.656377	1.1706	0.924123	0.503855	0.91571
6	1.36529	1.76172	1.54178	2.54225	1.20355	2.5259
7	0.586572	1.49354	0.947516	1.24119	0.609783	1.14628
8	**95.2688**	1.89953	0.509124	1.50525	**90.61**	0.759902
9	0.50304	0.554606	2.96516	0.429245	0.479011	1.42747

Tabela 5.8: Glass: matriz de confusão com 10% de rótulos

Grupo/Classe	1	2	3	4	5	6
1	21	45	4	3	1	1
2	17	0	2	0	0	0
3	0	0	0	1	1	23
4	4	5	1	1	0	0
5	28	25	10	8	1	2
6	0	1	0	0	6	3

atualização do vetor de relevância, por essa razão usamos $\beta = 5 \times 10^{-5}$. A Tabela 5.7 mostra o valor do vetor de relevância produzido no final do *SS-CLAMP* usando 10% de rótulos que indica a ordem de grandeza esperada como discutido na seção 3.4, a matriz de confusão e apresentada na tabela (5.8).

5.5 Multiple features

A Figura 5.4 mostra o desempenho dos algoritmos *SS-CLAMP*, *SS-CLAMP'* e *SS-CARD* nessa base de dados utilizando diferentes porcentagem de dados rotulados. Os resultados mostraram um dos melhores desempenho alcançado pelo *SS-CLAMP* em todas as bases de dados. O *SS-CLAMP'* manteve a mesma perfomance apenas para os casos com 10% e depois disso não conseguiu respeitar as restrições, o que foi o mesmo caso para o *SS-CARD*. No entanto o índice *fuzzy* que mostraram uma escala de pequenas mudanças, especialmente a de *Campello-Frigui*.

5.6 Yeast

A Figura 5.5 mostra o desempenho dos algoritmos *SS-CLAMP*, *SS-CLAMP'* e *SS-CARD* nessa base de dados utilizando diferentes porcentagem de dados rotulados. Ambos algoritmos partiram de uma performance similar com 0% de dados rotulados. O ganho de foi lento e basicamente linear para o *SS-CLAMP* sem grandes melhoras. Uma das razões que implicaram nesse desempenho pode está associada ao fator de desbalanceamento no número de objetos em cada grupo.

O *SS-CLAMP'* teve um resultado que mostrou a influência na escolha do α positivamente, conseguindo um dos melhores índices com 50% dos dados rotulados, com valor médio do α de 4.78×10^{-4} para uma inicialização que não respeita todas as restrições, usando a equação 3.7 seu valor é médio de 2.06×10^{4}.

5.7 Flower

A Figura 5.6 mostra o desempenho dos algoritmos *SS-CLAMP*, *SS-CLAMP'* e *SS-CARD* nessa base de dados utilizando diferentes porcentagem de dados rotulados. Os modelos que não utilizam informação supervisionada (0% de dados rotulados) correspondem as execuções do *MFCM-dd-RWL-P* e *CARD-R* que falharam na formação dos grupos para essa base de dados visto que eles apresentaram uma grande quantidade de grupos vazios. O *SS-CLAMP'* e o *SS-CARD* não mostraram muita evolução nos resultados ate mesmo usando uma grande quantidade de dados rotulados, apenas o *SS-CLAMP* conseguiu uma melhora expressiva especialmente na transição entre as porcentagens 0% e 10% onde o valor dos seus indices praticamente dobraram. O fator de escolha do α foi muito importante nessa classificação, dado que o *SS-CLAMP'* não pode conduzir o processo de otimização a produzir uma boa partição usando as restrições adequadamente, a forma como ele é calculado não equilibra bem esse fator durante as iterações e so faz que as restrições passassem a ser continuamente ignoradas.

A Tabela 5.9 mostra a matriz de confusão para o *MFCM-dd-RWL-P* que conseguiu produzir apenas 2 grupos dentre as 17 classes desse conjunto de dados. Durante sua execução ele teve uma queda de perfomance segundo todos os índices externos, uma vez que a partição fuzzy randômica inicial era uma melhor escolha pois apresentava um maior número de grupos não vazios, por exemplo, o índice de hullermeier foi de 0.0827 para 0.0656. Pode ser observado na Tabela 5.10 que com

Tabela 5.9: Matriz de confusão para o *MFCM-dd-RWL-P* no Flower

		Classes																
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Grupos	1	63	68	74	72	71	72	61	46	49	28	30	10	9	32	60	78	58
	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	5	17	12	6	8	9	8	19	34	31	52	50	70	71	48	20	2	22
	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Tabela 5.10: Matriz de confusão para o *SS-CLAMP* no Flower (10% dados rotulados)

		Classes																
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Grupos	1	4	0	0	0	0	0	1	9	0	8	1	56	22	4	0	0	0
	2	1	12	4	58	12	10	3	2	1	0	2	1	0	7	0	1	4
	3	0	0	2	3	0	0	0	3	1	0	0	0	0	0	0	0	41
	4	3	21	8	2	8	9	1	1	6	0	13	1	0	0	1	71	6
	5	1	14	3	1	4	6	0	2	0	1	58	0	0	0	0	7	2
	6	0	6	4	7	42	2	1	0	3	1	1	0	0	0	0	0	5
	7	5	0	0	1	1	0	0	1	0	47	1	1	1	2	1	0	0
	8	0	15	1	2	0	0	0	0	0	0	1	0	0	0	0	0	0
	9	2	11	45	1	1	1	0	1	1	0	0	0	0	2	0	1	0
	10	6	0	3	0	1	1	2	11	0	0	0	5	2	44	0	0	2
	11	0	0	0	1	5	1	2	0	64	0	0	0	0	0	0	0	0
	12	3	0	4	0	1	0	0	20	0	1	1	0	0	7	0	0	0
	13	5	0	0	0	0	0	0	7	0	1	0	0	0	0	49	0	7
	14	1	0	6	4	4	39	1	2	4	1	0	0	0	1	1	0	10
	15	47	0	0	0	0	8	0	8	0	2	0	1	0	11	24	0	2
	16	2	0	0	0	0	0	4	13	0	18	2	14	55	2	4	0	1
	17	0	1	0	0	1	3	65	0	0	0	0	1	0	0	0	0	0

10% de informação sobre os dados o SS-CLAMP produz uma matriz de confusão que mostra uma melhora significativa na identificação dos grupos.

É importante citar tambem que nos experimentos com esse conjunto de dados foi notado um comportamento inesperado para o índice de Campello, a figura (5.6) mostra uma escala de pequena mudanças entre as simulações enquanto em relação a todos os outros índices foi bem maior. Além disso, o valor desse índice não mudou desde a etapa de inicialização, ou seja, a diferença entre uma partição *fuzzy* aleatória em relação a final foi quase nenhuma, em [41] Hüllermeier e Rifqi apresentam um caso que mostra não conformidade entre essa métrica para comparação entre partições *fuzzy* para a medida de Campello. O constrante é bem notável, desde a inicialização até a convergência o índice de Campello vai de 0.889 para 0.890, enquanto o índice de Hüllermeier vai de 0.085 para 0.253 e a *F measure* vai de 0.339 para 0.592.

6
Conclusão

A principal contribuição deste trabalho foi a proposta de um algoritmo semi-supervisionado do tipo *fuzzy c-medoids* para agrupamento de dados relacionais representados por múltiplas matrizes de dissimilaridade que produz uma partição *fuzzy* dos dados de entrada e também encontra um conjunto de *medoids* para cada grupo enquanto otimiza uma função objetivo de forma competitiva com restrições do tipo *must-link* e *cannot-link*. Foi também apresentada uma abordagem simples e intuitiva para cálculo do α usando um algoritmo eficiente, os experimentos indicam a importância da escolha do seu valor é muito importante no processo de agrupamento dos dados. O valor da escolha do α para os modelos SS-CARD e SS-CLAMP' apresentaram anomalias em algumas bases de dados na presença de dados completamente rotulados e em casos com bases de dados maiores, enquanto que a escolha fixa e automática do α aprensentou maior estabilidade e melhores resultados.

Para o algoritmo proposto, é apresentado a solução dos melhores *medoids* para os grupos *fuzzy*, do melhor vetor de relevância das matrizes de dissimilaridade, e a melhor partição *fuzzy* da função objetivo. Uma das grandes vantagens dele são sua precisão na etapa do cálculo e atualização da partição *fuzzy*, que foi resolvida com as condições de *Kuhn* e *Tucker* que não adicionaram nenhum grande problema. Os experimentos não indicaram pertubações no processo de convergência durante as iterações, o SS-CLAMP não teve problemas de minimização da função objetivo em todas as bases de dados.

Os testes com as bases de dados indicaram uma bom desempenho do *SS-CLAMP*, alguns conjuntos mostraram uma grande melhora na formação dos grupos na presença de informação supervisionada parcial.

Como trabalhos futuros podem ser feitas melhoras nesse modelo para simplificar

ele tirando parâmetros como o número de *medoids*. Pode também ser explorado o uso de *self-training* e seus impactos no modelo. A complexidade é um ponto importante que pode ser estudado em novas abordagens a fim de reduzir o custo na fase de atualização dos *medoids*, existe a possibilidade de usar estruturas de dados mais eficientes ou técnicas com algoritmos aleatórios de forma nessa atualização. Os experimentos mostraram a importância do α no resultado final e um estudo mais complexo sobre sua influência em outros modelos ou uma nova proposta de uma nova forma atualização são pontos importantes para análise.

Referências Bibliográficas

[1] Anil K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recogn. Lett.*, 31(8):651–666, June 2010.

[2] M-E. Nilsback and A. Zisserman. A visual vocabulary for flower classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1447–1454, 2006.

[3] Rui Xu and II Wunsch, D. Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3):645–678, 2005.

[4] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[5] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2-3):107–145, 2001.

[6] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. John Wiley and Sons, New York, 2005.

[7] Hichem Frigui, Cheul Hwang, and Frank Chung-Hoon Rhee. Clustering and aggregation of relational data with applications to image database categorization. *Pattern Recognition*, 40(11):3053 – 3068, 2007.

[8] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

[9] Kiri Wagstaff and Claire Cardie. Clustering with instance-level constraints. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 1103–1110, 2000.

[10] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[11] Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

[12] Sugato Basu, Arindam Banerjee, and Raymond J. Mooney. Semi-supervised clustering by seeding. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, pages 27–34, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[13] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[14] W. Pedrycz and J. Waletzky. Fuzzy clustering with partial supervision. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 27(5):787–795, Sep 1997.

[15] James C. Bezdek, Robert Ehrlich, and William Full. Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2–3):191 – 203, 1984.

[16] ABDELHAMID BOUCHACHIA and WITOLD PEDRYCZ. Data clustering with partial supervision. *Data Mining and Knowledge Discovery*, 12(1):47–78, 2006.

[17] Abdelhamid Bouchachia and Witold Pedrycz. Enhancement of fuzzy clustering by mechanisms of partial supervision. *Fuzzy Sets and Systems*, 157(13):1733 – 1759, 2006.

[18] Kamal Nigam, AndrewKachites Mccallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39(2-3):103–134, 2000.

[19] Shumeet Baluja. Probabilistic modeling for face orientation discrimination: Learning from labeled and unlabeled data. In *In NIPS*, pages 854–860, 1998.

[20] Akinori Fujino, Naonori Ueda, and Kazumi Saito. A hybrid generative/discriminative approach to semi-supervised classifier design. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 2*, AAAI'05, pages 764–769. AAAI Press, 2005.

[21] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, pages 92–100, New York, NY, USA, 1998. ACM.

[22] Ralf Klinkenberg. Using labeled and unlabeled data to learn drifting concepts. In *In Workshop notes of IJCAI-01 Workshop on Learning from Temporal and Spatial Data*, pages 16–24. AAAI Press, 2001.

[23] Vladimir N. Vapnik. *Statistical learning theory*. Wiley, 1 edition, September 1998.

[24] Ayhan Demiriz, Kristin Bennett, and Mark J. Embrechts. Semi-supervised clustering using genetic algorithms. In *In Artificial Neural Networks in Engineering (ANNIE-99*, pages 809–814. ASME Press, 1999.

[25] Ian Davidson and Sugato Basu. A survey of clustering with instance level constraints. *ACM Transactions on Knowledge Discovery from Data*, pages 1 – 41, 2007.

[26] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 577–584, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[27] Sugato Basu, A. Banjeree, ER. Mooney, Arindam Banerjee, and Raymond J. Mooney. Active semi-supervision for pairwise constrained clustering. In *In Proceedings of the 2004 SIAM International Conference on Data Mining (SDM-04*, pages 333–344, 2004.

[28] N. Grira, M. Crucianu, and Nozha Boujemaa. Semi-supervised fuzzy clustering with pairwise-constrained competitive agglomeration. In *Fuzzy Systems, 2005. FUZZ '05. The 14th IEEE International Conference on*, pages 867 – 872, 2005.

[29] Nizar Grira, Michel Crucianu, and Nozha Boujemaa. Active semi-supervised fuzzy clustering. *Pattern Recogn.*, 41(5):1851–1861, May 2008.

[30] H. Frigui and Cheul Hwang. Fuzzy clustering and aggregation of relational data with instance-level constraints. *Fuzzy Systems, IEEE Transactions on*, 16(6):1565–1581, 2008.

[31] Richard J. Hathaway and James C. Bezdek. Nerf c-means: Non-euclidean relational fuzzy clustering. *Pattern Recognition*, 27(3):429 – 437, 1994.

[32] Francisco de A.T. de Carvalho, Yves Lechevallier, and Filipe M. de Melo. Relational partitioning fuzzy clustering algorithms based on multiple dissimilarity matrices. *Fuzzy Sets and Systems*, 215(0):1 – 28, 2013.

[33] Hichem Frigui and Olfa Nasraoui. Unsupervised learning of prototypes and attribute weights. *Pattern Recognition*, 37(3):567 – 581, 2004.

[34] Kendall Atkinson. *An Introduction to Numerical Analysis*. Wiley, 2 edition, 1989.

[35] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition*. The MIT Press, 2 edition, September 2001.

[36] Stephen V. Stehman. Selecting and interpreting measures of thematic classification accuracy. *Remote Sensing of Environment*, 62(1):77 – 89, 1997.

[37] Francisco de A.T. de Carvalho, Yves Lechevallier, and Filipe M. de Melo. Partitioning hard clustering algorithms based on multiple dissimilarity matrices. *Pattern Recognition*, 45(1):447 – 464, 2012.

[38] C. J. Van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2nd edition, 1979.

[39] William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):pp. 846–850, 1971.

[40] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.

[41] Eyke Hüllermeier, Maria Rifqi, Sascha Henzgen, and Robin Senge. Comparing Fuzzy Partitions: A Generalization of the Rand Index and Related Measures. *IEEE Transactions on Fuzzy Systems*, (20):546–556, 2012.

[42] Ricardo J. G. B. Campello. A fuzzy extension of the rand index and other related indexes for clustering and classification assessment. *Pattern Recognition Letters*, 28(7):833–841, 2007.

[43] Edgar Anderson. The species problem in iris. *Annals of the Missouri Botanical Garden*, 23:457–509, 1936.

[44] M. Breukelen, Robert P. W. Duin, David M. J. Tax, and J. E. den Hartog. Handwritten digit recognition by combined classifiers. *Kybernetika*, 34(4):381 – 386, 1998.

[45] M-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

Apêndices

A

Derivação das equações de atualização da partição fuzzy

O *SS-CLAMP* minimiza:

$$J = \sum_{i=1}^{n}\sum_{k=1}^{K}(u_{ik})^2\sum_{j=1}^{T}\lambda_{kj}\sum_{e\in G_k}d_j(e_i,e) + \alpha\left(\sum_{(l,m)\in\mathcal{M}}\sum_{r=1}^{K}\sum_{\substack{s=1\\s\neq r}}^{K}u_{lr}u_{ms} + \sum_{(l,m)\in\mathcal{K}}\sum_{r=1}^{K}u_{lr}u_{mr}\right) + \beta\left(\sum_{k=1}^{K}\sum_{j=1}^{T}\lambda_{kj}^2\right) \tag{A.1}$$

sujeito a

$$u_{ik}\geq 0 \qquad \sum_{k=1}^{C}u_{ik}=1 \qquad \forall i \quad \text{e} \quad \lambda_{kj}>0 \qquad \prod_{j=1}^{T}\lambda_{kj}=1 \qquad \forall k.$$

Para minimizar a função objetivo J em respeito a matriz de pertinência U nós usamos o método dos multiplicadores de Lagrange, sendo assim temos:

$$L = \sum_{i=1}^{n}\sum_{k=1}^{K}(u_{ik})^2\sum_{j=1}^{T}\lambda_{kj}\sum_{e\in G_k}d_j(e_i,e) + \alpha\left(\sum_{(l,m)\in\mathcal{M}}\sum_{r=1}^{K}\sum_{\substack{s=1\\s\neq r}}^{K}u_{lr}u_{ms} + \sum_{(l,m)\in\mathcal{K}}\sum_{r=1}^{K}u_{lr}u_{mr}\right) + \beta\left(\sum_{k=1}^{K}\sum_{j=1}^{T}\lambda_{kj}^2\right) - \sum_{i=1}^{N}\gamma_i\left(\sum_{k=1}^{C}u_{ik}-1\right) - \sum_{i=1}^{N}\sum_{k=1}^{C}\psi_{ik}u_{ik} \tag{A.2}$$

em que γ_i e ψ_{ik} são os multiplicadores.

Calculando sua derivada em relação ao grau de pertinência encontramos:

$$\frac{\partial L}{\partial u_{ik}} = u_{ik}a_{ik} + b_{ik} - \gamma_i - \psi_{ik} \tag{A.3}$$

na qual

$$a_{ik} = 2\sum_{j=1}^{T} \lambda_{kj} \sum_{e \in g_k} d_j(e_i, e) \tag{A.4}$$

$$b_{ik} = \alpha \left(\sum_{(i,m)\in\mathcal{M}} \sum_{\substack{s=1 \\ s\neq k}}^{K} u_{ms} + \sum_{(i,m)\in\mathcal{C}} u_{mk} + \sum_{(l,i)\in\mathcal{M}} \sum_{\substack{s=1 \\ s\neq k}}^{K} u_{sm} + \sum_{(l,i)\in\mathcal{C}} u_{kl} \right) \tag{A.5}$$

Os requisitos para a a minimização dessa função de acordo com as condições de Kuhn e Tucker são:

$$\begin{cases} \psi_{ik} \geq 0 \\ \frac{\partial L}{\partial u_{ik}} = 0 \\ u_{ik}\psi_{ik} = 0 \end{cases} \tag{A.6}$$

Resolvendo a relação (A.3) com $\sum_{k=1}^{K} u_{ik} = 1$ temos:

$$\gamma_i = \frac{1 + \sum_{w=1}^{C} \left({}^{b_{ik}}/_{a_{iw}}\right) - \sum_{w=1}^{C} \left({}^{\psi_{ik}}/_{a_{iw}}\right)}{\sum_{w=1}^{C} \left({}^{1}/_{a_{iw}}\right)} \tag{A.7}$$

e

$$u_{ik} = \frac{\gamma_i + \psi_{ik} - b_{ik}}{a_{ik}} \tag{A.8}$$

Substituindo o termo (A.7) em (A.8), temos:

$$u_{ik} = \frac{\left({}^{1}/_{a_{ik}}\right)}{\sum_{w=1}^{C} \left({}^{1}/_{a_{iw}}\right)} + \frac{\sum_{w=1}^{C} \left({}^{b_{iw}}/_{a_{iw}}\right)}{a_{ik} \sum_{w=1}^{C} \left({}^{1}/_{a_{iw}}\right)} - \frac{b_{ik}}{a_{ik}} + \frac{\psi_{ik}}{a_{ik}} - \frac{\sum_{w=1}^{C} \left({}^{\psi_{iw}}/_{a_{iw}}\right)}{a_{ik} \sum_{w=1}^{C} \left({}^{1}/_{a_{iw}}\right)} \tag{A.9}$$

As condiçõoes (A.6) permitem a existência de apenas duas possibilidades:

$$\psi_{ik} = 0 \implies u_{ik} \geq 0 \tag{A.10}$$

$$\psi_{ik} > 0 \implies u_{ik} = 0 \tag{A.11}$$

Em (A.9) se considerarmos $\psi_{ik} = 0$ e obtivermos $u_{ik} < 0$ podemos seguramente afirmar que apenas a possibilidade (A.11) pode ser satisfeita para u_{ik} não ficar negativo.

No caso degenerado é quando existe um $a_{ik} = 0$ decidimos maximizar o u_{ik} pois isso indica que a dissimilaridade do objeto e_i em relação ao grupo k é zero. Nesse caso então temos que o caso da equação (A.12) e resolvendo a γ_i na equação (A.3) temos $\gamma_i = b_{ik}$. Quando existem diferentes valores de b_{ik} quando $a_{ik} = 0$ essa solução não é válida então temos que fazer a escolha de algum b_{ik} para γ_i. Como ainda temos que manter a equação (A.6) com solução, isto é, $\psi_{ik} >= 0$ logo a única escolha de γ_i que torna isso possível é quando $\gamma_i = min(\{b_{ik} \mid a_{ik} = 0\})$. Com o valor de γ_i e usando a equação (A.8) temos:

$$u_{ik} = \begin{cases} 0 & \text{se } \gamma_i \leq b_{ik} \\ \frac{\gamma_i - b_{ik}}{a_{ik}} & \text{se } a_{ik} > 0 \\ \frac{1 - \sum_{w \notin Z_i} u_{iw}}{|Z_i|} & \text{se } a_{ik} = 0 \end{cases} \tag{A.12}$$

onde

$$Z_i = \{k \mid a_{ik} = 0\}$$

Note que na equação (A.12) a divisão foi igualitária para os elementos de Z_i mas outras escolhas também são possíveis. Se ainda existir algum $u_{ik} < 0$ isso quer dizer que o valor de γ_i não foi válido ao mesmo tempo que qualquer outro valor maior igual ao que foi escolhido causará o mesmo problema logo temos que $\gamma_i < min(\{b_{ik} \mid a_{ik} = 0\})$. Mesmo que esse valor não seja exato essa condição faz com que se $a_{ik} = 0$ então $u_{ik} = 0$ via equação (A.12), anulando o caso degenerado.

B

Derivação das equações de atualização do vetor de relevância

Da equação (A.1) para minimizar J em respeito ao vetor de relevância usando o método dos multiplicadores de Lagrange temos:

$$
\begin{aligned}
L \;=\; & \sum_{i=1}^{n}\sum_{k=1}^{K}(u_{ik})^2\sum_{j=1}^{T}\lambda_{kj}\sum_{e\in G_k}d_j(e_i,e) \\
& + \alpha\left(\sum_{(l,m)\in\mathcal{M}}\sum_{r=1}^{K}\sum_{\substack{s=1\\ s\neq r}}^{K}u_{lr}u_{ms} + \sum_{(l,m)\in\mathcal{K}}\sum_{r=1}^{K}u_{lr}u_{mr}\right) + \beta\left(\sum_{k=1}^{K}\sum_{j=1}^{T}\lambda_{kj}^2\right) \\
& - \sum_{k=1}^{K}\theta k\left(\prod_{j=1}^{T}\lambda_{kj} - 1\right)
\end{aligned}
\tag{B.1}
$$

em que os θ_k é são os multiplicadores de Lagrange.

Calculado a derivada em relação ao vetor de relevância encontramos:

$$
\frac{\partial L}{\partial \lambda_{kj}} = c_{kj} - \theta_k\frac{1}{\lambda_{kj}} + 2\beta\lambda_{kj} \tag{B.2}
$$

na qual

$$
c_{kj} = \sum_{i=1}^{n}(u_{ik})^2\sum_{e\in G_k}d_j(e_i,e) \tag{B.3}
$$

Fazendo $\frac{\partial L}{\partial \lambda_{kj}} = 0$ e resolvendo o caso onde $\beta = 0$ temos que $\lambda_{kj} = \frac{c_{kj}}{\theta_k}$ e usando a condição que $\prod_{j=1}^{T}\lambda_{kj} = 1$ podemos calcular que $\theta_k = (\prod_{j=1}^{T}c_{kj})^{\frac{1}{T}}$. Sendo assim

temos que:

$$\lambda_{kj} = \frac{\{\prod_{h=1}^{p} [c_{kh}]\}^{\frac{1}{p}}}{[c_{kj}]} \tag{B.4}$$

Quando $\beta \neq 0$ temos que $(c_{kj} + 2\beta\lambda_{kj})\lambda_{kj} = \theta_k$ pois $\lambda_{kj} > 0$ o que fica uma equação do segundo grau para λ_{kj}, a solução fica:

$$\lambda_{kj} = \frac{\sqrt{[c_{kj}]^2 + 8\beta\theta_k} - [c_{kj}]}{4\beta} \tag{B.5}$$

A equação (B.5) ainda não diz o valor de θ_k, mas fazendo $\prod_{j=1}^{T} \lambda_{kj} = 1$ e substindo λ_{kj} por essa equação a única variável é o θ_k. Esse problema pode ser facilmente resolvido usando uma busca binária, visto que o valor de θ_k tem o mesmo impacto em λ_{kj}, ou seja, quãnto maior for θ_k maior também será todos os $\lambda_{kj}\,(j = 1, \ldots, T)$, logo essa equação tem solução única pois o λ_{kj} é uma função crescente em relação ao θ_k.