Universidade Federal de Pernambuco
Centro de Informática

Graduação em Ciências da Computação

MÉTODOS DE KERNEL PARA AGRUPAMENTOS DE DADOS DE TIPO INTERVALO

Bruno Almeida Pimentel

Trabalho de Graduação

Recife
16 de Dezembro de 2010

Universidade Federal de Pernambuco
Centro de Informática

Bruno Almeida Pimentel

MÉTODOS DE KERNEL PARA AGRUPAMENTOS DE DADOS DE TIPO INTERVALO

Trabalho apresentado ao Programa de Graduação em Ciências da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Ciências da Computação.

Orientadora: *Profa. Dra. Renata Maria C R de Souza*

Recife
16 de Dezembro de 2010

Dedico esse trabalho aos meus pais, por terem ensinado o verdadeiro valor dos estudos. E à professora Renata, por me apresentar a ciência e pesquisa e me guiar nessa jornada de descobertas.

Agradecimentos

Agradeço à minha família por ter me ajudado a chegar nesta etapa da minha vida, com insentivo e compreensão, em especial aos meus pais que estão me acompanhando e orientando em todos os momentos. Por terem me ensinado o valor do estudo e que a busca pelo conhecimento e pela verdadeira vocação devem ser características essenciais presentes em qualquer cidadão.

Aos professores e monitores desta instituição, por fazerem parte da minha formação acadêmica. Especialmente, à professora e amiga Renata, que desde os primeiros semestres do curso me auxiliu nessa construção do conhecimento, permitindo, primeiramente, o ingresso na monitoria de Estatística, a qual deu ocasião a ajudar outros colegas e aprender mais sobre a disciplina, e depois a participação de pesquisas através da iniciação científica, atividade que me fez ficar ainda mais admirado pela ciência.

Agradeço, também aos meus colegas de turma, com os quais debatemos idéias e discutimos sobre os mais disversos temas ajudando na minha formação acadêmica e profissional. Aos amigos de pesquisa, que me auxiliaram no desenvolvimento de trabalhos de iniciação científica e estudo de assuntos científicos juntamente com a professora Renata. Aos meus colegas de monitoria, com os quais contribuimos para que vários outros colegas de curso consolidassem seus conhecimentos de Estatística.

Uma jornada de duzentos quilômetros começa com um simples passo.
—PROVÉBIO CHINÊS

Resumo

Este trabalho relata sobre os diferentes métodos de particionamento presentes na literatura atual e introduz um novo método de *clustering* baseado em funções de *Kernel* para dados intervalares. Os métodos de nuvens dinâmicas usam algoritmos que buscam encontrar uma partição em classes para os dados apresentados, ao mesmo tempo em que realizam operações para minimizar um critério obtido a partir da comparação entre os elementos e os seus respectivos protótipos de acordo com cada classe. Neste trabalho abordaremos os métodos de nuvens dinâmicas usando distâncias adaptativas, as quais consideram, além da partição e da minimização de um critério, um fator variante em cada interação do algoritmo fazendo essa distância ser diferente em cada comparação da classe com o seu protótipo. Uma das vantagens desses métodos usando distâncias adaptativas é que conseguem identificar *clusters* de tamanhos e formas diferentes, entretanto, são limitados quando os dados são dispostos de forma não-linear. A proposta dos métodos de *Kernel* para agrupamento de dados do tipo intervalo é realizar esse tipo de separação entre as classes. A comparação entre os métodos clássicos e os que usam funções de *Kernel* é feita através de experimentos realizados entre dados sintéticos, os quais demonstram explicitamente a não-linearidade das classes, e dados reais intervalares fazendo uso do experimento Monte Carlo e métricas estatísticas.

Palavras-chave: *Clustering*, *Kernel*, Dados do Tipo Intervalo, Não-Linear

Abstract

This paper reports on the different partitioning methods present in literature and introduces a new clustering method based on Kernel functions. The dynamic clustering method use algorithms that seek to find a partition into classes for the data presented, while carrying out operations to minimize a criterion obtained from the comparison between the elements and their respective prototypes according to each class. In this paper we discuss the dynamic clustering methods using adaptive distances, which consider, besides the partition and the minimization of a criterion, a variant factor in each iteration of the algorithm making this distance is different in each comparison of class with its prototype. One of these methods advantages using adaptive distances that is able to identify clusters of different sizes and shapes, however are limited when data is arranged in a nonlinear manner. The proposed methods for kernel grouping of data-type interval is performing this type of separation between classes. The comparison between the classical methods and those that use kernel functions is done through experiments made between synthetic data that demonstrate explicitly the non-linearity classes and real data interval of the experiment using Monte Carlo and statistics metrics.

Keywords: Clustering, Kernel, Data-type Interval, Nonlinear

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Introdução

Aprender é a única de que a mente nunca se cansa, nunca tem medo e nunca se arrepende.

—LEONARDO DA VINCI

Atualmente, com o grande aumento do uso de base de dados e o grande volume de dados armazenados, o agrupamento tornou-se um assunto muito importante e a análise de *cluster* tem sido utilizada em diversos domínios de aplicações como mineração de dados, reconhecimento de padrões, bioinformática e assim por diante. Esses métodos podem ser divididos em métodos hierárquicos e de particionamento [1] [2]. Os métodos de particionamento de clusters tentam obter uma única partição de dados sem qualquer outra sub-partição como ocorre nos algoritmos hierárquicos. Outra característica do particionamento é que eles são baseados na otimização de uma função objetivo adequado.

Na análise de *clusters* clássico, os dados são frequentemente representados como matrizes de valores quantitativos ou qualitativos, onde cada coluna representa uma variável. Entretanto, pesquisas mostraram que essa presentação não é tão rica em informação para a complexidade dos dados encontrados nos problemas reias, como os bancos de dados usados pela sociedade [3]. Com o objetivo de mostrar a variabilidade e a incerteza presente nos dados, estes são representados por meio de conjuntos de intervalos. O estudo desses tipos de dados é realizado, principalmente, pela Análise de Dados Simbólicos (ADS). A intenção do ADS é extender os métodos tradicionais com dados clássicos para métodos com dados do tipo intervalo, fazendo uso de técnicas estatísticas. O ADS fornece métodos em que vários tipos de dados simbólicos podem ser usados em algoritmos de particionamento, tais como métodos de nuvens dinâmicas usando distâncias adaptativas de Hausdorff [6], distâncias quadráticas L_2 [7] ou as baseadas em *city-block* [8].

Um dos mais populares algoritmos de agrupamento é o *K-Means*, no qual grupos homogêneos são identificados, minimizando o erro do agrupamento definida como a soma das distâncias euclidianas quadradas entre cada conjunto de dados pontuais e os correspondentes centros dos aglomerados. A simplicidade desses algoritmos de agrupamento é uma característica importante, assim como a grande variedade de problemas de particionamento não-supervisionado para os quais são usados. Entretanto, resultados insatisfatórios encontrados quando o K-Means é usado em dados dispostos não-linearmente motivam a evolução de outros métodos de particionamento que tenham maior eficiência na separação dos grupos.

É com essa necessidade de separação de dados não-lineares que o método de *clustering Kernel* é estudado [9]. Este método é capaz de produzir uma separação não-linear entre os

hiper-espaços dos *clusters*. Em Aprendizagem de Máquina, a utilização das funções de *Kernel* foram introduzidas há muito tempo por Aizerman [10]. Em 1995, Cortes e Vapnik introduziram o *Support Vector Machines* (SVMs) [4], que têm melhores desempenhos que outros algoritmos classificadores nos mais variados problemas, por exemplo, *Kernel PCA* [5]. Neste trabalho será apresentado um novo método de *clusters* para o particionamento de conjunto de dados simbólicos do tipo intervalo. Este método é uma extensão do algoritmo de cluster *Kernel K-Means* proposto em [25].

O capítulo 2 a seguir relata sobre os métodos de agrupamente presentes na literatura atual, mostrando os seus diferentes tipos tais como os sequenciais, os hierárquicos e os de particionamento. No capítulo 3, será apresentada uma introdução dos métodos de nuvens dinâmicas para dados simbólicos do tipo intervalo com distâncias adaptativas quadráticas L_2. No capítulo 4, será introduzido o novo método de *clusters* para o particionamento de conjunto de dados simbólicos do tipo intervalo. No capítulo 5, uma série de experimentos é relatado com o objetivo de comparar os métodos de nuvens dinâmicas com o método proposto, utilizando o índice de Rand corrigido e a taxa de erro de classificação global. Finalmente, o capítulo 6 tratará das conclusões obtidas após a experimentação e trabalhos futuros idealizados.

CAPÍTULO 2

Métodos de Agrupamento

O saber é saber que nada se sabe. Este é a definição do verdadeiro conhecimento.
—CONFÚCIO

2.1 Introdução

Uma das mais básicas abilidades dos seres vivos envolve o agrupamento de objetos similares para produzir uma classificação. Desde os primórdios do seu surgimento, o homem, por exemplo, obteve habilidades para indentificar que muitos objetos possuiam certas propriedades, tais como a comestibilidade de alimentos, a usabilidade de ferramentas, a ferocidade de animais, entre outros. Desta forma, surge a idéia de agrupamento (*cluster* ou classe), no qual os objetos são reunidos de modo que a semelhança entre eles é maior do que qualquer outra classe existente. O conceito de agrupamento está relacionado a diversos ramos do conhecimento, fazendo parte das pesquisas de muitas áreas, tais como Reconhecimento de Padrões, Estatística, Matemática, Engenharia e Física, sendo usado em várias aplicações, como medicina, psiquiatria, serviços sociais, pesquisa de mercado, educação e arqueologia.

A Análise de Agrupamentos (*Clustering Analysis*) objetiva separar um conjunto inicial de objetos em um determinado número de agrupamentos, de modo que os elementos pertecentes ao mesmo agrupamento possuam mais semelhanças (similaridades) entre si, e sejam mais diferentes (dissimilares) aos pertecentes a outros agrupamentos. O conjunto de c classes disjuntas, as quais quando unidas obtem-se n objetos é denominado partição. A figura a seguir ilustra um particionamento de objetos dispostos no $\Re^2$:

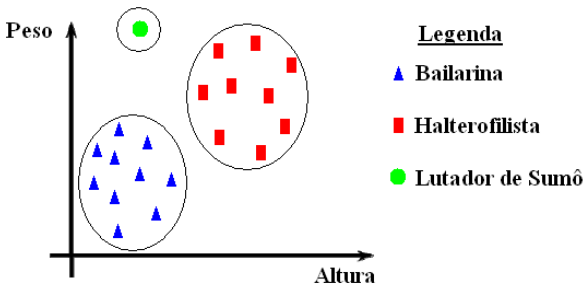

Figura 2.1 Exemplos de agrupamentos de dados

Diferentemente dos algoritmos de classificação, nos quais os objetos precisam ser rotulados por classe caracterizando um aprendizado supervisionado, os algoritmos de agrupamento não nessecitam dessa rotulação, isto é, ocorre a aprendizagem não-supervisionada. Na aprendizagem supervisionada, a separação das classes é realizada por um supervisor, que mede o grau de desempenho desse algoritmo e realiza ajustes sobre o mesmo até que seja atingida alguma medida que seja considerada satisfatória, utilizando como recurso alguma informação externa sobre o domínio avaliado. Entretanto, essa abordagem possui uma limitação: é necessário um conhecimento preliminar do domínio estudado, o que não é possível em diversas situações. Na aprendizagem não-supervisionada, como ocorre nos métodos de agrupamento, não há a necessidade de informações a priori sobre o domínio avaliado, levando-se em consideração, apenas, a disposição dos dados e suas propriedades internas.

Como esses métodos são executados de forma não-supervisionada, existem diversas técnicas para a estimação do número ideal de conjuntos finais que devem ser criados de forma a tornar a divisão dos dados mais representativa para o problema estudado, como apresentado em [12]. Além disso, podem ser classificados de acordo com a forma que interpretam os dados e a maneira com que esses objetos se organizam em agrupamentos.

2.2 Métodos Sequenciais

O algoritmo de métodos sequenciais possuem uma descrição simples e, consequentemente, fácil de entender. Os dados são apresentados poucas vezes e o número de classes não é conhecido a priori, apenas um número limite q de *clusters* definido pelo usuário, assim como um único limiar Θ, para o Esquema de Algoritmo Seqüencial Básico (BSAS), ou mais, para o Esquema de Algoritmo com Dois Limiares (TTSAS). Além disso, é necessária a medida de proximidade d tal que $d(x,C)$ mede o quão próximo um objeto x está de um conjunto C (esse conjunto pode estar sendo representado por um protótipo, de forma que d pode ser calculado entre o objeto e o representante de C).

2.2.1 Esquema de Algoritmo Seqüencial Básico

O algoritmo é simples e rápido de executar, já que lê apenas uma vez todo o conjunto de dados. Entretanto, esses algoritmos possuem a desvantagem de serem sensíveis a ordem de apresentação dos elementos estudados, uma vez que a representação dos conjuntos por meio de protótipos pode mudar a cada iteração e com isso a relação de proximidade entre o elemento e o grupo seja alterada. O esquema de algoritmo seqüencial básico pode ser definido como [13]:

1. $m = 1$
2. $C_m = x_1$
3. Para $i = 2$ até N
    - Encontre C_k: $d(x_i, C_k) = \min_j d(x_i, C_j)$
    - Se $d(x_i, C_k) > \Theta$ **E** $m < q$ então

  - $m = m + 1$
  - $C_m = x_i$
- Senão
  - $C_k = C_k \cup x_i$
  - Se d for calculada a partir de protótipos, atualize o protótipo do novo conjunto C_k
- Fim_Se

4. Fim_Para

Como se pode perceber, o algoritmo executa apanas uma única vez todo o conjunto de dados, possuindo, portanto, o custo computacional da ordem de $O(n)$. A distância d utilizada é importante para o resultado final da partição, uma vez que ela irá definir se o elemento atual pertencerá ao conjunto C_k ou se será necessário criar outro. O que é importante também é o limiar Θ, já que ele tem efeito direto sobre o número de conjunto formados: se for pequeno demais, conjuntos desnecessários serão criados; ser for grande demais, serão poucos conjuntos criados, podendo ser insuficiente para representar a partição ideal dos dados.

Existe uma modificação do algoritmo BSAS chamado *Modified* BSAS (MBSAS), que é executado fazendo a leitura dos dados duas vezes. Ela supera a desvantagem de um conjunto final de uma única amostra ser decidida antes de todos os grupos serem criados. A primeira fase do algoritmo cria *clusters* (tal como em BSAS) e atribui uma única amostra de cada cluster. Então a segunda fase percorre as amostras restantes e as classifica para os grupos criados.

2.2.2 Esquema de Algoritmo com Dois Limiares

A principal desvantagem dos algoritmos BSAS e MBSAS é a order com que os exemplos são apresentados, bem como o verdadeiro valor de Θ. Estes incovenientes podem ser reduzidos pela utilização de dois limiares Θ_1 e Θ_2. Distâncias menores que o primeiro valor Θ_1 denotam que a amostra em questão provavelmente pertence ao *cluster* do qual a distância foi calculada. Por outro lado, distâncias maiores que Θ_2 denotam que o objeto não pertence ao *cluster*. Valores entre esses dois limiares estão uma faixa chamada "zona cizenta"e devem ser avalidos em uma fase posterior ao algoritmo.

Considerando $clas(x)$ um booleano indicando se uma amostra foi classificada ou não e assumindo que não existe limites para o número de *clusters*, o esquema de algoritmo seqüencial básico de dois limiares pode ser descrito da seguinte forma:

1. $m = 0$
2. Para todo x, $clas(x) = False$
3. $alteracao\_anterior = 0$; $alteracao\_atual = 0$; $existe\_alteracao = 0$;
4. Enquanto exixtir algum exemplo não classificado
   - Para $i = 1$ até N

- Se $clas(x) = False$ **E** é o primeiro neste *loop while* **E** $existe\_alteracao = 0$
  - $m = m + 1$
  - $C_m = x; clas(x) = True$
  - $alteracao\_atual = alteracao\_atual + 1$
- Senão se $clas(x) = False$
  - Encontre $\min d(x, C_k)$
  - Se $d(x, C_k) < \Theta_1$
    - $C_k = C_k + x; clas(x) = True$
    - $alteracao\_atual = alteracao\_atual + 1$
  - Senão se $d(x, C_k) < \Theta_2$
    - $m = m + 1$
    - $C_m = x; clas(x) = True$
    - $alteracao\_atual = alteracao\_atual + 1$
- Senão
  - $alteracao\_atual = alteracao\_atual + 1$

- Fim_Para
- $existe\_alteracao = |alteracao\_atual - alteracao\_anterior|$
- $alteracao\_anterior = alteracao\_atual$; $alteracao\_atual = 0$;

5. Fim_Enquanto

A variável *existe_alteracao* verifica se o último elemento do conjunto de elementos foi classificado na passagem atual do *loop while*. Se nenhum foi classificado, o primeiro exemplo não classificado é usado para gerar uma nova classe e isso tenta garantir que, no máximo, N passagens pelo *loop while* foram executadas. De acordo com a complexidade teórica computacional deste algoritmo, o custo é da ordem de $O(n^2)$.

2.3 Métodos Hierárquicos

Nos algoritmos de agrupamento hierárquicos, os dados não são particionados em um determinado número classes ou *clusters* em um único passo. Nestes métodos, o objetivo é obter uma informação mais completa sobre o conjunto de objetos por meio de um conjunto hirarquicalmente aninhado de partições. Em taxonomia, por exemplo, um objeto pode pertencer sucessivamente a uma espécie a um gênero, uma família, uma ordem, etc. O agrupamento consiste de uma série de partições, as quais podem possuir apenas uma classe contendo todos os indivíduos, ou *n clusters*, cada um contendo apenas um elemento.

As técnicas de agrupamento hierárquicos podem ser divididos em duas categorias: algoritmos aglomerativos e algoritmos divisivos. Os aglometativos reduzem os dados em um único *clusters* contendo todos os elementos, enquanto que a técnica divisiva irá dividir o conjunto de

entrada em n grupos, onde cada um contém apenas um indivíduo. Com isso, surge a necessidade de decidir em qual estágio do algoritmo irá parar, uma vez que é preciso encontrar uma solução com um número "ótimo"de classes.

O agrupamento hierárquico pode ser representado por um diagrama bidimensional conhecido como dendograma, o qual ilustra a união ou divisão realizada nos sucessivos estágios de análise. Um dendograma pode ser considerado um conjunto D de subconjuntos Ω satisfazendo as seguintes condições:

1. $\Omega \in D$;
2. D é um conjunto não-vazio;
3. $i \in D \; \forall i \in \Omega$;
4. Se $A, B \in D$ então $A \cap B \in \{\emptyset, A, B\}$.

A imagem a seguir exemplifica a relação entre as duas categorias de algoritmos hierárquicos por meio de um dendograma:

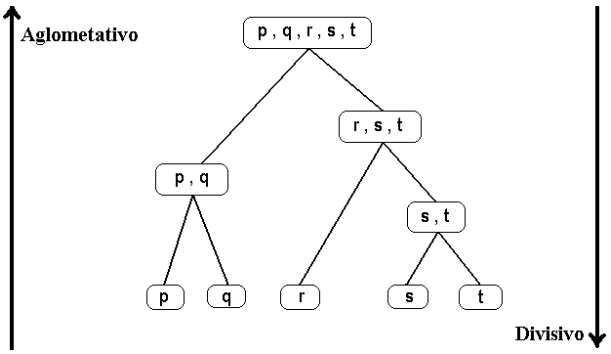

Figura 2.2 Exemplo de um dendograma

2.3.1 Algoritmo Aglomerativo

Um método de agrupamento hierárquico aglometativo produz um série de partições do conjunto de dados, $P_n, P_{n-1}, \ldots, P_1$. O primeiro, P_n, consiste de n classes com um único membro. O último, P_1, consite de um único grupo contendo todos os n elementos. Uma operação básica do método pode ser represntada a seguir:

1. INICIALIZAR: Faça $C_1, C_2, \ldots, C_n$ classes contendo apenas um elemento;
2. Encontre o par mais próximo de classes distintas C_i e C_j;
3. Una C_i e C_j, apague C_j e decremente o número de classes por um;
4. Se o número de classes for igual a um, então pare, senão siga no passo 2.

Em cada estágio do método, indivíduos ou grupos de indivíduos se fundem de acordo com sua proximidade ou por sua semelhança. Existem diferentes maneiras de se medir a distância entre os grupos, desta forma, existem várias técnicas usadas nos algoritmos aglomerativos. As mais comuns são o método aglomerativos de ligação simples (*Single-link method*) e de ligação completa (*Complete-link method*).

No de ligação simples, a distância entre os grupos é definida a partir do par de indivíduos mais próximos, onde somente pares de um indivíduo de cada classe são considerados. Como exemplo, considere o seguinte conjunto de distâncias entre 5 elementos:

	1	2	3	4	5
1	0.0				
2	2.0	0.0			
3	6.0	5.0	0.0		
4	10.0	9.0	4.0	0.0	
5	9.0	8.0	5.0	3.0	0.0

Tabela 2.1 Distâncias entre elementos

Usando o método algomerativo de ligação simples obtem-se o dendograma:

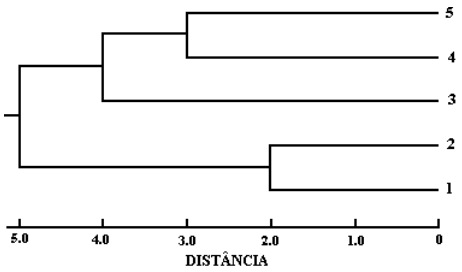

Figura 2.3 Dendograma de ligação simples

No de ligação completa, ocorre o oposto ao de ligação simples, devido ao fato que a dissimilaridade entre os dois grupos será a dissimilaridade máxima calculada entre todos os pares de indivíduos pertecentes a estes grupos. Usando esse método para a tabela anterior, o dendograma a seguir pode ser esquematizado:

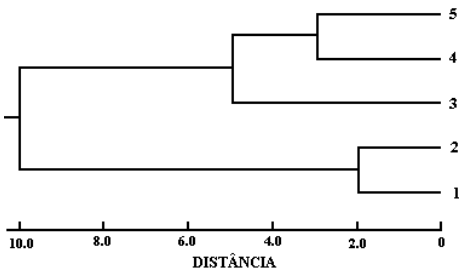

Figura 2.4 Dendograma de ligação completa

Além dos de ligação simples e completa, existem outras técnicas para medir a dissimilaridade entre dois grupos. No agrupamento de médias de grupos, a dissimilaridade é obtida a partir da média das distâncias entre todos os pares de objetos, sendo cada um pertecente a um grupo diferente. Existe o agrupamento por centróides, onde cada grupo é representado por um centróide (vetor de médias) e a distância é medida entre os centróides de cada grupo. No agrupamento de medianas, o processo é semelhante ao por centróides, entretanto a distância é obtida a partir de vetores de medianas.

2.3.2 Algoritmo Divisivo

Os métodos hierárquicos divisivos são caracterizados por reunir todos os n elementos da base em uma única classe e progressivamente separa-los de modo que no final da execução tenham-se n classes com apenas um elemento em cada uma. Os algoritmos divisivos podem ser divididos em dois principais grupos: monotéticos (*monothetics*), que dividem o conjunto de dados de levando-se em conta apenas um único atributo, e politéticos (*polythetics*), cujas divisões são realizadas a partir dos diversos atributos que um padrão pode ter.

No contexto de métodos de agrupamento, uma potencial desvantagem dos métodos divisivos sobre o métodos aglomerativos pode ocorrer quando o objetivo é encontrar uma partição dos dados em um relativamente pequeno número de *clusters* [14].

Estes métodos podem funcionar em conjunto com métodos de particionamento tais como *K-Means* ou *K-Medoids* aplicados recursivamente na função de dividir o grupo anterior em dois grupos distintos fazendo-se $K = 2$. Entretanto, essa modificação irá depender da configuração inicial em cada etapa do método hierárquico divisivo. Porém, isto não produz necessariamente uma separação cuja propriedade de monoticidade do dendograma aconteça. Isto é, a altura de cada nó não é proporcional para o valor da dissimilaridade intergrupais entre as suas duas filhas [14].

2.4 Métodos de Particionamento

Esses algoritmos incluem os do tipo rígidos (*hard*) e os do tipo difusos (*fuzzy*). Os rígidos se caracterizam por considerarem as classes disjuntas, ou seja, não exitem elementos em comum em duas diferentes classes, desta forma, cada indivíduo pertence a somente um grupo. Por outro lado, os algoritmos do tipo difuso extendem esse conceito de associação de cada elemento em uma classe: um indivíduo pode pertencer a diversas classes de acordo com uma função de pertinência capaz de associar cada padrão a cada um dos *clusters* assumindo valores no intervalo $[0, 1]$.

2.4.1 Algoritmos Rígidos

Consistem em obter uma partição a partir de um determinado conjunto de n elementos agrupados em um número pré-definido de k classes, onde $k \leqslant n$, de forma que cada classe possua pelo menos um elemento e cada elemento deve pertencer unicamente a uma classe, isto é, não admitem a existância de grupos vazios e que estes não tenham elementos em comum.

Existem versões desse tipo de algoritmo onde, além de encontrarem a partição composta por n elementos e k classes, fazem uso da otimização de função de custo: algoritmos de nuvem dinâmica. Estas funções se caracterizam por analisar as classes em cada iteração e usar métricas para fornecer informações sobre o particionamento. O problema da otimização de funções de custo consiste em fazer uso da função e encontrar a melhor solução dentre todas as possíveis soluções objetivando minimizar um critério antes estabelecido.

Um dos mais populares algoritmos de particionamento é o *K-Means* [17], no qual grupos homogêneos são identificados, minimizando o erro do agrupamento definida como a soma das distâncias euclidianas quadradas entre cada conjunto de dados e os correspondentes centros dos aglomerados (centróides). O *K-Means* é um caso particular dos algoritmos de nuvens dinâmicas, o qual reconhece apenas regiões esféricas devido a sua distância fixa. A versão adaptativa desses algoritmos, por outro lado, é capaz de associar uma distância diferente para cada classe a cada nova iteração, resultando em um reconhecimento de formas e tamanhos variados entre as classes, sendo isto uma vantagem sobre os métodos que usam distâncias fixas.

2.4.2 Algoritmos Difusos

Os conjuntos *fuzzy* foram introduzidos em 1965 por Kadeh como uma nova maneira de representar imprecisões do cotidiano [16]. Esta teoria fornece um conceito eficaz para aproximar e descrever as características de um sistema que é muito complexo ou mal definido para admitir análise matemática precisa. Admite-se que a forma com que o pensamento humano trabalha com conceitos-chave não são apenas números, mas também uma aproximação de conjuntos difusos.

Os algoritmos do tipo difuso extendem o conceito de associação de cada elemento em uma classe, isto é, um indivíduo pode pertencer a diversas classes de acordo com uma função de pertinência capaz de associar cada padrão a cada um dos *clusters* assumindo valores no intervalo $[0,1]$. Neste caso, cada classe é um conjunto nebuloso de todos os objetos. Cada elemento x possui um grau de pertinância para uma classe k, de forma que a soma de todos os graus relativos a essa elemento x tem que valer 1. Isto é: $\sum_k^K u_k(x) = 1$.

Uma desvantagem desse tipo de algoritmo é a definição da função de pertinência. Diferentes funções são usadas, entre elas estão as baseadas em centróides de *clusters*. Outra desvangem é a dependência da escolha inicial dos protótipos das classes, como ocorre no algoritmo rígido *K-Means*. O algoritmo difuso mais usado é o *Fuzzy-C-Means*, onde os elementos mais próximos das bordas possuem um menor grau de pertinência, enquanto aqueles mais próximos ao centróide têm uma pertinência maior e o centríode é obtido fazendo-se uma média ponderada do grau de todos os indivíduos daquele grupo.

2.5 Outros Métodos

Algoritmos Evolutivos

Estes tipos de algoritmos se baseiam em mecanismos da evolução biológica, tais como reprodução, mutação, recombinação e seleção. O conjunto de dados representa a população sob evolução e esta é simulada através de repetidas operações associadas às multações genéticas

comuns na evolução. Usam a idéia de que os indivíduos mais áptos têm mais chances de deixar informações sobre o seu código genético para as próximas gerações do que outros indivíduos menos preparados. Os algoritmos evolutivos são aplicados para aproximar funções com bom desempenho e, desta forma, são usados em diversos domínios como engenharia, economia, genética e robótica. Sub-grupos dos algoritmos evolutivos foram criados de acordo com a aplicação para as quais são destinados. O mais popular é o algoritmo genético, no qual busca a solução através de operações, como mutação, sobre cadeias de números, geralemente binários. Outro tipo é a evolução de estratégia, na qual trabalha com vetores de números reais como representação da solução.

QT (*Quality Threshold*)
É uma alternativa de método de agrupamento feito especialmente para ser usado em banco de dados com informações sobre genes [18]. Neste algoritmo, cada elemento é comparado em pares com todos os outros e é calculado um coeficiente de correlação. Elementos são agrupados de modo que todos os elementos dentro de um *cluster* devem ser mais fortemente correlacionadas com um único elemento central do que o limite de qualidade de entrada (*threshold*). Elementos são acrescentados a esse *cluster* enquanto esse limite é satisfeito. Se o limite for ultrapassado, cria-se outra classe agrupando-se elementos próximos de forma que respeitem o limite. Desta forma, o QT é mais custoso computacionamente do que o tradicional *K-Means*, já que todo elemento é comparado com todos os outros da base. Uma vantagem desse método é que não são sensíveis à ordem dos dados apresentados, todas as informações na base de dados é considerada. Outra vantagem é que o algoritmo pode facilmente identificar e ranquear os conjuntos de genes que são mais correlacionados de acordo com uma determinada característica.

Redes Neurais
São algoritmos que tantam representar as redes neurais biológicas por meio de grupos de unidades as quais se comunicam para chegarem a uma resposta final. Cada unidade faz o processamento dos sinais recebidos advindos de outras possíveis ligações, as quais possuem pesos com a finalidade de simular a sinapses presentes nas atividades cerebrais. De acorco com o processamento, é possível dizer a saída fazendo uso de um limiar e função de ativação. As saidas combinadas entre as unidades da rede neural artificial dão respostas a repeito de um determinado problema, geralmente classificando elementos de um conjunto de dados. São modelos muito usados onde é difícil criar um comportamento matemético bem definido para um problema, ou que a estrutura da informação a qual irão analisar são bastante complexos se fossem usados por outros modelos computacionais.

Existem diversas versões de redes neurais dependendo da finalidade para as quais são destinadas. De acordo com a topologia, podem ser do tipo *feedforward*, onde os sinais de entrada sempre chegam através de conexões de camadas anteriores da rede ou da própria entrada. Outra topologia é a recorrente, na qual os sinais de saída podem ser usados como entrada da rede, realimentando com conexões de *feedback*. E as construtivas que se caracterizam por poderem criar neurônios entre a camada de entrada e a saída de acordo com os dados analisados com o objetivo de reduzir ao máximo o erro final entre a resposta desejada e a obtida. Desta forma, redes neurais são usadas em diversas categorias, uma delas está relacionada com as funções de

aproximação, como análise de regressão e séries de previsão de tempo; outra está relacionada com a classificação, reconhecimento de padrões e tomadas de decisão; e uma terceira com processamento de dados, tais como filtragem, compactação e *clustering*.

Classificação com Sobreposição

Em muitos métodos de agrupamento, existe o processamento de dados de forma que não haja sobreposição de dados ou de classes, isto é, os grupos são disjuntos entre si. Entretanto, na abordagem de classificação com sobreposição, há a possibilidade de um elemento pertencer a mais de uma classe, sendo bastente útil para muitas aplicações onde o conceito de sobreposição é importante [19]. Na biologia, por exemplo, genes participam simultaneamente de vários processos. Desta forma, quando for necessário agrupar os genes de acordo com suas expressões gênicas fazendo uso de micro-arrays, é conveniente atribuir a esses genes a sobreposição de classes, de forma que a consulta de genes pode está associada a diversas características importantes.

Alguns dos métodos que usam esse conceito de sobreposição são o B_k e o de pirâmides. O método B_k é caracterizado pelas classes poderem ter no máximo $k-1$ elementos em comum. Quando se faz B_1, ou $k=1$, o método fica equivalente ao método hierárquico aglomerativo de ligação simples. A construção das classes é feita utilizando-se teoria dos grafos, onde cada elemento é representado por um vértece e as arestas estão associadas à dissimilaridade em cada par de elementos. Enquanto no do tipo pirâmide, o agrupamento é feito hierarquicamente levando em conta a ordenação das classes, de forma que a pirâmide assume uma forma generalizada do dendograma, uma vez que pode ser obtida a partir de um método hierárquico aglomerativo.

CAPÍTULO 3

Métodos de Nuvens Dinâmicas: *K-Means* para Dados Intervalares

Os que se encantam com a prática sem a ciência são como os timoneiros que entram no navio sem timão nem bússola, nunca tendo certeza do seu destino.

—LEONARDO DA VINCI

3.1 Introdução

Os algoritmos de nuvens dinâmicas compreende diversos métodos de *clusters* não hierárquicos. O objetivo principal é obter um particionamento em um número de classes pré-definido a partir de um conjunto de dados de modo que o conjunto de protótipos escolhidos minimizem um critério que mede a adequação entre esses representates e os respectivos elementos de cada classe. O ajustamento dos elementos de acordo com o critério permite encontrar uma solução ótima local para o problema caracterizando uma vantagem desse método. Outra propriedade importante desses algoritmos é a capacidade de reconhecer classes de formas e tamanhos diferentes quando usam distâncias adaptativas, as quais consideram a dispersão dos elementos em cada classe ou em todo o conjunto de dados. Por outro lado, existe o problema de convergência nesses algoritmos, uma vez que a partição final tem relação direta tanto dos protótipos escolhidos para definir o conjunto de classes inicial quanto da distância usada para definir o representante mais próximo e o ajuste dos grupos.

O algoritmo começa com um conjunto de protótipos escolhidos aleatoramente, os quais definirão a configuração inicial das classes afetando cada elemento de acordo a distância L_2 fixa encontrada entre o protótipo e o padrão. A classe pertecente ao elemento será a classe do protótipo que possui a menor distância caracterizando a etapa de alocação. Em seguida uma etapa de representação é realizada, onde os protótipos são atualizados de acordo com os elementos pertecentes em cada classe. Essa duas etapas prosseguem iterativamente até a convergência do algoritmo definida a priori, mas com uma diferença na etapa de alcação: a distância usada não precisa ser a L_2 fixa. Uma forma de avaliar a convergência é verificar se o valor do critério do algoritmo estabilizou, caracterizando pouca variação entre os elementos das classes. Outra forma é verificar se todas as classes permaneceram invariantes, sem que nenhum elemento seja realocado. Caso a convergência não seja alcançada, o ciclo de alocação e representação se repete. Tradicionalmente, o método é executado diversas vezes com diferentes combinações de protótipos com o objetivo de obter a melhor partição para a resposta final.

Nas próximas seções, serão apresentados os diferentes tipos de métodos de nuvens dinâmicas variando de acordo com o tipo de distância usada. Na seção 2.1, serão apresentados os métodos com distância fixa, enquanto na 2.2 com distâncias adaptativas. Estas, por sua vez, podem ser do tipo única ou por classe.

3.2 Métodos de Nuvens Dinâmicas com Distância Fixa

Seja $\Omega = \{1,\ldots,n\}$ um conjunto de n objetos indexados por i descrito por p variáveis simbólicas intervalares. Uma variável simbólica intervalar X é uma correspondência definida de Ω em $\Re$, tal que, para cada $i \in \Omega, X(i) = [a,b] \in \Im$, onde $\Im$ é um conjunto de intervalos fechados definidos de $\Re$.

Cada objeto i $(i = 1,\ldots,n)$ é representado como um vetor de intervalos $\mathbf{x}_i = ([a_i^1,b_i^1],\ldots,[a_i^p, b_i^p])^T$. Semelhantente ao objeto, o protótipo de um *cluster* C_k $(k = 1,\ldots,K)$ também é representado como um vetor de intervalos $\mathbf{y}_k = ([\alpha_k^1,\beta_k^1],\ldots,[\alpha_k^p,\beta_k^p])^T$. Desta forma, a partição $P = (C_1,\ldots,C_K)$ de Ω em K classes onde cada classe $C_k(k = 1,\ldots,K)$ tem um representante (protótipo) L_k que pode ser descrito pelo vetor $\mathbf{y}_k$.

Sejam $\mathbf{P}_K$ um conjunto de partições, onde $P = (C_1,\ldots,C_K)$ de Ω em K classes e um conjunto $\mathbf{L}^K = \mathbf{L}\times\mathbf{L}\ldots\times\mathbf{L}$ de *K-uplas* $L = (L_1,\ldots,L_K)$ com $L_k \in \mathbf{L}$. O problema do agrupamento está relacionado a encontrar uma partição $\mathbf{P}^* \in \mathbf{P}_K$ em K classes e um conjunto de protótipos $L^* \in \mathbf{L}^K$ relativo às classes tal que

$$J(P^*,L^*) = Min\{J(P,L)|P \in P^K, L \in L^K\}, \tag{3.1}$$

onde o critério $J(P,L)$ mede a adequação entre a partição P e o conjunto de protótipos L.

Quando a distância usada nesse método é a L_2, o método de nuvens dinâmicas assume o caso particular conhecida como *K-Means*. Este método representa cada classe por um vetor de centros, isto é, cada protótipo da classe é um centróide. Com o uso da distância Euclidiana L_2, o algoritmo é capaz de identificar classes dispostas em forma esférica, onde o critério é dado da seguinte forma:

$$J(P,L) = \sum_{k=1}^{K}\sum_{i\in C_k} d^2(x_i,y_k). \tag{3.2}$$

O critério medirá o quão distante cada elemento x_i da classe k está do seu centróide y_k do conjunto L usando a distância euclidiana quadrática. Isso é feito para todas a K classes do particionamento P. Onde:

$$d(x_i,y_k) = \sqrt{\sum_{j=1}^{p}\left[\left(a_i^j - \alpha_k^j\right)^2 + \left(b_i^j - \beta_k^j\right)^2\right]}. \tag{3.3}$$

E cada limite inferior e superior dos intervalos do protótipo $y_k = ([\alpha_k^1, \beta_k^1], \ldots, [\alpha_k^p, \beta_k^p])^T$:

$$\alpha_k^j = \frac{1}{n_k} \sum_{i \in C_k} a_i^j, \beta_k^j = \frac{1}{n_k} \sum_{i \in C_k} b_i^j. \tag{3.4}$$

Ou seja, cada limite dos intervalos do protótipo y_k é formada pela média aritmética simples entre os respectivos limites dos elementos x_i da classe C_k.

Algoritmo

1. **Inicialização**
   Escolha aleatoriamente uma partição $P = (C_1, \ldots, C_K)$ de Ω ou escolha, também aleatoriamente, K elementos diferentes $(y_1, \ldots, y_K)$ do conjunto de objetos Ω e associe cada elemento x_i ao objeto mais próximo y_{k*}, onde $k* = \arg\min_{k=1,\ldots,K} \sum_{j=1}^{p} [(a_i^j - \alpha_k^j)^2 + (b_i^j - \beta_k^j)^2]$ para construir a partição inicial $(C_1, \ldots, C_K)$;

2. **Etapa de representação: definição do melhor protótipo**
   Para $k = 1, \ldots, K$ calcule os protótipos $y_k = ([\alpha_k^1, \beta_k^1], \ldots, [\alpha_k^p, \beta_k^p])^T$ onde α_k^j é a média dos valores do conjunto $\{a_i^j : i \in C_k\}$ e β_k^j é a média dos valores do conjunto $\{b_i^j : i \in C_k\}$ e $(j = 1, \ldots, p)$;

3. **Etapa de alocação: definição da melhor partição**
   $teste \leftarrow 0$
   Para $i = 1, \ldots, n$ faça

   definia a classe vencedora C_{k*} tal que:

   $$k* = \arg\min_{k=1,\ldots,K} \sum_{j=1}^{p} d^2(x_i, y_k).$$

   se $i \in C_k$ e $k* \neq k$

   $teste \leftarrow 1$

   $C_{k*} \leftarrow C_{k*} \cup \{i\}$

   $C_k \leftarrow C_k \setminus \{i\}$

4. **Critério de parada**
   Se $teste = 0$ então PARE, caso contrário siga na etapa 2.

O algoritmo inicia com uma partição P definida aleatoriamente a partir da uma já pronta ou construida usando elementos aleatórios como protótipos para fazer uma alocação segundo as distâncias mais próximas. Em seguida, duas etapas são realizadas iterativamente: a representação para a obtenção dos protótipos $L = (L_1, \ldots, L_K)$ das K classes fazendo uso da partição definida até o momento pelo algoritmo; e a definição da nova partição $P = (C_1, \ldots, C_K)$ através

da distância entre os elementos e os protótipos obtidos na fase anterior. Isso se repete até que nenhum elemento seja alocado, ou seja, a partição no ciclo anterior é igual a atual.

O algoritmo converge na direção de uma solução ótima local para J. Como diferentes J podem ser obtidos a partir de diferentes inicilizações aleatórias, é comum realizar um determinado número de repetições no algoritmo com diferentes inicializações de forma que a solução para o problema seja o par (P^*, L^*) cujo valor de J seja o mínimo dentre aqueles calculados em cada repetição.

3.3 Métodos de Nuvens Dinâmicas com Distância Adaptativa

O objetivo principal desse método é associar uma distância diferente para cada classe ou elemento, a qual muda a cada iteração do algoritmo. A partir dessas variações o algoritmo é capaz de reconhecer diferentes dispersões entre os padrões e as classes, de modo que *clusters* de tamanhos diferentes e/ou não isomorfos sejam identificados. A idéia é associar diferentes distâncias para cada *clusters* de forma que, no final de cada etapa de alocação, a distância entre os elementos e os seus respectivos representantes seja a menor possível.

Seja $\Omega = \{1,\ldots,n\}$ um conjunto de n objetos indexados por i descrito por p variáveis simbólicas intervalares. Uma variável simbólica intervalar X é uma correspondência definida de Ω em $\Re$, tal que, para cada $i \in \Omega, X(i) = [a,b] \in \Im$, onde $\Im$ é um conjunto de intervalos fechados definidos de $\Re$.

Cada objeto i $(i = 1,\ldots,n)$ é representado como um vetor de intervalos $\mathbf{x}_i = ([a_i^1, b_i^1], \ldots, [a_i^p, b_i^p])^T$. Semelhantemente ao objeto, o protótipo de um *cluster* C_k $(k = 1,\ldots,K)$ também é representado como um vetor de intervalos $\mathbf{y}_k = ([\alpha_k^1, \beta_k^1], \ldots, [\alpha_k^p, \beta_k^p])^T$.

Seja $\mathbf{P}_K$ um conjunto de partições, onde $P = (C_1,\ldots,C_K)$ de Ω em k classes e um conjunto $\mathbf{L}^K = \mathbf{L} \times \mathbf{L} \ldots \times \mathbf{L}$ de *K-uplas* $L = (L_1,\ldots,L_K)$ com $L_k \in \mathbf{L}$ e $\mathbf{d}^K = d \times d \ldots \times d$ um conjunto de K distâncias $d = (d_1,\ldots,d_K)$ com $d_k \in \mathbf{d}$.

O problema de classificação nos métodos de nuvens dinâmicas com distâncias adaptativas está associado a encontrar uma partição $\mathbf{P}^* \in \mathbf{P}_K$ em K classes, um conjunto de protótipos $L^* \in \mathbf{L}^K$ relativo às classes e um conjunto de distâncias $d^* \in \mathbf{d}^K$ tal que

$$J(P^*, L^*, d^*) = Min\{J(P, L, d) | P \in P^K, L \in L^K, d \in d^K\} \tag{3.5}$$

onde o critério $J(P,L,d)$ mede a adequação entre a partição P e o conjunto de protótipos L fazendo uso das distâncias d.

Os métodos de nuvens dinâmicas adaptativas consideraram um fator em cada cálculo da distância o qual pode variar considerando a organização do particinonamento. Esse fator considera as informações locais, sejam elas associadas à classe C_k ou a cada elemento i da partição P. Dependendo da forma como esse fator é calculado, a distância adaptativa pode ser por classe, quando o fator varia por grupos, ou única, quando considera somente o elemento em questão.

3.3.1 Métodos de Nuvens Dinâmicas com Distância Adaptativa por Classe

A idéia principal deste método [20] é associar a distância d_k a cada *cluster* C_k tal que a soma das distâncias entre os elementos e os seus respectivos protótipos seja a menor possível. Uma característica importante deste método é que a distância não é única para todas a classes, mas sim diferente obtida em cada iteração do método. Em cada iteração é medida a adequação do critério do cluster C_k que mede a sua qualidade através da soma das distâncias $d_k(x_i, y_k)$ entre os objetos $x_i \in C_k$. Assim, o critério J é localmente minimizado e pode ser obtido da seguinte forma:

$$J = \sum_{k=1}^{K} \sum_{i \in C_k} d_k^2(x_i, y_k), \tag{3.6}$$

onde d_k é a distância L_2 adaptativa da classe C_k. Nos métodos adaptativos por classe definidos por um único peso (chamado CP), as distâncias obtidas são calculadas a partir de um único vetor de pesos $\lambda_k = (\lambda_k^1, \ldots, \lambda_k^p)$, onde cada λ_k^p varia de acordo com cada *cluster*:

$$d_k^2(x_i, y_k) = \sum_{j=1}^{p} \lambda_k^j \left[\left(a_i^j - \alpha_k^j \right)^2 + \left(b_i^j - \beta_k^j \right)^2 \right]. \tag{3.7}$$

E cada limite inferior e superior dos intervalos do protótipo $y_k = ([\alpha_k^1, \beta_k^1], \ldots, [\alpha_k^p, \beta_k^p])^T$ são obtidos da mesma forma que na distância fixa:

$$\alpha_k^j = \frac{1}{n_k} \sum_{i \in C_k} a_i^j, \beta_k^j = \frac{1}{n_k} \sum_{i \in C_k} b_i^j. \tag{3.8}$$

Algoritmo

1. **Inicialização**
   Escolha aleatoriamente uma partição $P = (C_1, \ldots, C_K)$ de Ω ou escolha, também aleatoriamente, K elementos diferentes $(y_1, \ldots, y_K)$ do conjunto de objetos Ω e associe cada elemento x_i ao objeto mais próximo y_{k*}, onde $k* = \arg\min_{k=1,\ldots,K} \sum_{j=1}^{p} [(a_i^j - \alpha_k^j)^2 + (b_i^j - \beta_k^j)^2]$ para construir a partição inicial $(C_1, \ldots, C_K)$;

2. **Etapa de representação**

   - *Definição dos melhores protótipos*: Para $k = 1, \ldots, K$ calcule os protótipos $y_k = ([\alpha_k^1, \beta_k^1], \ldots, [\alpha_k^p, \beta_k^p])^T$ onde α_k^j é a média dos valores do conjunto $\{a_i^j : i \in C_k\}$ e β_k^j é a média dos valores do conjunto $\{b_i^j : i \in C_k\}$ e $(j = 1, \ldots, p)$;
   - *Definição das melhores distâncias*: Para $k = 1, \ldots, K$ calcule o vetor de pesos $\lambda_k = (\lambda_k^1, \ldots, \lambda_k^p)$ com $\lambda_k^j > 0$ e $\prod_{j=1}^{p} \lambda_k^j = 1$ onde λ_k^j é dado de acordo com a equação:

$$\lambda_k^j = \frac{\prod_{h=1}^{p}\left[\sum_{i\in C_k}\left(a_i^h-\alpha_k^h\right)^2+\left(b_i^h-\beta_k^h\right)^2\right]^{\frac{1}{p}}}{\sum_{i\in C_k}\left(a_i^j-\alpha_k^j\right)^2+\left(b_i^j-\beta_k^j\right)^2}.$$

3. **Etapa de alocação: definição da melhor partição**
   $teste \leftarrow 0$
   Para $i = 1,\ldots,n$ faça

   definia a classe vencedora C_{k*} tal que:

$$k* = \arg\min_{k=1,\ldots,K}\sum_{j=1}^{p} d^2(x_i,y_k).$$

   se $i \in C_k$ e $k* \neq k$

   $teste \leftarrow 1$

   $C_{k*} \leftarrow C_{k*} \cup \{i\}$

   $C_k \leftarrow C_k \setminus \{i\}$

4. **Critério de parada**
   Se $teste = 0$ então PARE, caso contrário siga na etapa 2.

O algoritmo começa definindo a partição inicial e segue semelhantemente ao algoritmo de nuvens dinâmicas usando distância fixa, isto é, fica-se alternando entre as etapas 2, 3 e 4 até a convergência e o critério J acha uma solução de mínimo local. Mas existem algumas diferenças características dos métodos adaptativos. Na primeira etapa, há a definição da partição P em K classes $C_1,\ldots,C_K$. Na etapa de representação, os protótipos são representados a partir das médias entre os elementos das classes e as distâncias adaptativas para cada classe são calculadas usando-se os pesos $\lambda_1^j,\ldots,\lambda_k^p$ objetivando minimizar o critétio J. Enquanto que na etapa de alocação, os elementos são atribuidos à classe correspondente a menor distância entre o respectivo protótipo.

Outra categoria de distâncias adaptativas por classe são aquelas que usam dois vetores de pesos (chamado CPP), um para cada limite do intervalo: um para o limite inferior $\lambda_{kL} = (\lambda_{kL}^1,\ldots,\lambda_{kL}^p)$ e um para o limite superior $\lambda_{kU} = (\lambda_{kU}^1,\ldots,\lambda_{kU}^p)$. O critério J é calculado de forma:

$$J = \sum_{k=1}^{K}\sum_{i\in C_k} d_k^2(x_i,y_k), \tag{3.9}$$

Mas com uma mudança no cálculo da distância d_k:

$$d_k^2(x_i,y_k) = \sum_{j=1}^{p}\left[\lambda_{kL}^j\left(a_i^j-\alpha_k^j\right)^2+\lambda_{kU}^j\left(b_i^j-\beta_k^j\right)^2\right], \tag{3.10}$$

ou seja, o cálculo é feito considerando, não semente as informações inerentes a cada classe k, mas também aos limites dos intervalos característicos em cada conjunto de dados.

O algoritmo é semelhante ao que usa vetores de peso com um componente, mas há uma alteração no cálculo da distância na etapa de representação, já que considera os componentes da forma:

$$\lambda_{kL}^{j} = \frac{\prod_{h=1}^{p}\left[\sum_{i\in C_k}\left(a_i^h - \alpha_k^h\right)^2\right]^{\frac{1}{p}}}{\sum_{i\in C_k}\left(a_i^j - \alpha_k^j\right)^2}, \tag{3.11}$$

para o limite inferior. E para o limite superior tem-se:

$$\lambda_{kU}^{j} = \frac{\prod_{h=1}^{p}\left[\sum_{i\in C_k}\left(b_i^h - \beta_k^h\right)^2\right]^{\frac{1}{p}}}{\sum_{i\in C_k}\left(b_i^j - \beta_k^j\right)^2}. \tag{3.12}$$

3.3.2 Métodos de Nuvens Dinâmicas com Distância Adaptativa Única

A proposta da ditância adapatativa única é, assim como a por classe, mudar a cada iteração do método, de modo que a forma e o tamanho das classes sejam identificados [21]. Entretanto, a principal diferença encontrada na distância adaptativa única é que esta é a mesma para todos os *clusters*. Para isso, o método associa a distância d_k a cada *cluster* C_k tal que a soma das distâncias entre os elementos e os seus respectivos protótipos seja localmente minimizado. Em cada iteração é medida a adequação do critério do cluster C_k que mede a sua qualidade através da soma das distâncias $d_k(x_i, y_k)$ entre os objetos $x_i \in C_k$. Assim, o critério J é obtido da seguinte forma:

$$J = \sum_{k=1}^{K}\sum_{i\in C_k} d_k^2(x_i, y_k), \tag{3.13}$$

em que

$$d_k^2(x_i, y_k) = \sum_{j=1}^{p} \lambda^j \left[\left(a_i^j - \alpha_k^j\right)^2 + \left(b_i^j - \beta_k^j\right)^2\right]. \tag{3.14}$$

que é a distância quadrática L_2 adaptativa única medindo a dissimilaridade entre um objeto $x_i (i = 1, \ldots, n)$ e o protótipo do *cluster* $y_k (k = 1, \ldots, K)$. No método de distância adaptativa única de um peso (chamado UP), a distância é parametrizada pelo vetor de pesos de um componente $\lambda = (\lambda^1, \ldots, \lambda^p)$, o qual muda a cada iteração, mas é o mesmo para qualquer classe

[21] [22]. Cada limite dos intervalos do protótipo $y_k = ([\alpha_k^1, \beta_k^1], \ldots, [\alpha_k^p, \beta_k^p])^T$ é obtido através da média aritmética simples entre os limites dos elementos da classe C_k:

$$\alpha_k^j = \frac{1}{n_k} \sum_{i \in C_k} a_i^j, \beta_k^j = \frac{1}{n_k} \sum_{i \in C_k} b_i^j. \tag{3.15}$$

Algoritmo

1. **Inicialização**
   Escolha aleatoriamente uma partição $P = (C_1, \ldots, C_K)$ de Ω ou escolha, também aleatoriamente, K elementos diferentes $(y_1, \ldots, y_K)$ do conjunto de objetos Ω e associe cada elemento x_i ao objeto mais próximo y_{k*}, onde $k* = \arg\min_{k=1,\ldots,K} \sum_{j=1}^{p} [(a_i^j - \alpha_k^j)^2 + (b_i^j - \beta_k^j)^2]$ para construir a partição inicial $(C_1, \ldots, C_K)$;

2. **Etapa de representação**

   - *Definição dos melhores protótipos*: Para $k = 1, \ldots, K$ calcule os protótipos $y_k = ([\alpha_k^1, \beta_k^1], \ldots, [\alpha_k^p, \beta_k^p])^T$ onde α_k^j é a média dos valores do conjunto $\{a_i^j : i \in C_k\}$ e β_k^j é a média dos valores do conjunto $\{b_i^j : i \in C_k\}$ e $(j = 1, \ldots, p)$;
   - *Definição das melhores distâncias*: Para $k = 1, \ldots, K$ calcule o vetor de pesos $\lambda = (\lambda^1, \ldots, \lambda^p)$ com $\lambda^j > 0$ e $\prod_{j=1}^{p} \lambda^j = 1$ onde λ_k^j é dado de acordo com a equação:

$$\lambda^j = \frac{\left\{ \prod_{h=1}^{p} \left[\sum_{k=1}^{K} \left(\sum_{i \in C_k} \left(a_i^h - \alpha_k^h \right)^2 + \left(b_i^h - \beta_k^h \right)^2 \right) \right] \right\}^{\frac{1}{p}}}{\sum_{k=1}^{K} \left(\sum_{i \in C_k} \left(a_i^h - \alpha_k^h \right)^2 + \left(b_i^h - \beta_k^h \right)^2 \right)}.$$

3. **Etapa de alocação: definição da melhor partição**
   $teste \leftarrow 0$
   Para $i = 1, \ldots, n$ faça

   definia a classe vencedora C_{k*} tal que:

$$k* = \arg\min_{k=1,\ldots,K} \sum_{j=1}^{p} d^2(x_i, y_k).$$

   se $i \in C_k$ e $k* \neq k$

   $teste \leftarrow 1$

   $C_{k*} \leftarrow C_{k*} \cup \{i\}$

   $C_k \leftarrow C_k \setminus \{i\}$

4. **Critério de parada**
   Se $teste = 0$ então PARE, caso contrário siga na etapa 2.

O algoritmo é semelhante ao usado no método de nuvens dinâmicas com distância adaptativa por classe. Isto é, defini-se aleatoriamente a partição inicial e uma sequência de representação e alocação é feita até que o método alcance o critério de parada. A principal diferença ocorre na etapa de representação, com a definição das melhores distância, já que estas serão calculadas usando-se o vetor de pesos de um componente $\lambda = (\lambda^1, \ldots, \lambda^p)$, o qual não diferencia os *clusters* da partição durante o cálculo.

O critério de adequação do agrupamento também pode ser baseado em distâncias L_2 adaptativa definidas por dois diferentes vetores de pesos. A idéia principal deste método (chamado UPP) é que existam duas distâncias adaptativas usadas para comparar um par de vetores de intervalo assumindo valores para cada variável simbólica. Um de acordo com o limite inferior e outro correspondente ao limite superior do intervalo.

O método procura uma partição Ω de K classes $C_1, \ldots, C_K$ e seus respectivos protótipos $y_1, \ldots, y_K$ tal que o critério de adequação J obtido a partir da soma das distâncias entre os elementos e os seus correspondentes protótipos seja localmente minimizado. Assim, o critério pode ser expresso da seginte forma:

$$J = \sum_{k=1}^{K} \sum_{i \in C_k} d_k^2(x_i, y_k), \tag{3.16}$$

em que

$$d_k^2(x_i, y_k) = \sum_{j=1}^{p} \left[\lambda_L^j \left(a_i^j - \alpha_k^j \right)^2 + \lambda_U^j \left(b_i^j - \beta_k^j \right)^2 \right], \tag{3.17}$$

que é a distância quadrática L_2 adaptativa única medindo a dissimilaridade entre um objeto $x_i (i = 1, \ldots, n)$ e o protótipo do *cluster* $y_k (k = 1, \ldots, K)$. Ela é parametrizada por dois vetores de pesos $\lambda_L = (\lambda_L^1, \ldots, \lambda_L^p)$ para o limite inferior e $\lambda_U = (\lambda_U^1, \ldots, \lambda_U^p)$ para o limite superior do intervalo, os quais não diferenciam as classes e mudam a cada iteração de acordo com organização dos elementos na partição.

Os protótipos $y_k = ([\alpha_k^1, \beta_k^1], \ldots, [\alpha_k^p, \beta_k^p])^T$ também são definidos através dos elementos de cada *clusters*, onde α_k^j é a média dos valores do conjunto $\{a_i^j : i \in C_k\}$ e β_k^j é a média dos valores do conjunto $\{b_i^j : i \in C_k\}$ e $(j = 1, \ldots, p)$. Desta forma, o algoritmo do método é semelhante ao que usa apenas um vetor de pesos, mas com a diferença que a definição das melhores distâncias é obtida usando dois vetores calculados da forma:

$$\lambda_L^j = \frac{\left\{ \prod_{h=1}^{p} \left[\sum_{k=1}^{K} \left(\sum_{i \in C_k} \left(a_i^h - \alpha_k^h \right)^2 \right) \right] \right\}^{\frac{1}{p}}}{\sum_{k=1}^{K} \left(\sum_{i \in C_k} \left(a_i^h - \alpha_k^h \right)^2 \right)} \tag{3.18}$$

para o limite inferior, enquanto para o limite superior tem-se:

$$\lambda_U^j = \frac{\left\{\prod_{h=1}^{p}\left[\sum_{k=1}^{K}\left(\sum_{i\in C_k}\left(b_i^h-\beta_k^h\right)^2\right)\right]\right\}^{\frac{1}{p}}}{\sum_{k=1}^{K}\left(\sum_{i\in C_k}\left(b_i^h-\beta_k^h\right)^2\right)}. \tag{3.19}$$

O métodos adaptativos são mais eficientes do que os métodos não adaptativos, isto é, que usam distâncias fixas. Isso deve-se ao fato de que os métodos adaptativos são capazes de reconhecer classes de diferentes formas e tamanho.

O desempenho dos métodos de nuvens dinâmicas com distâncias quadráticas L_2 adaptativas depende da estrutura intra-classe definida a priori [21]. Para os conjuntos de dados intervalares em que as classes a priori tinham dispersões diferentes, o melhor desempenho foi obtido pelos métodos com distâncias adaptativas por classe. Para os conjuntos de dados em que as classes a priori tinham dispersões similares, o melhor desempenho foi obtido pelos métodos com distâncias adaptativas única. Além disso, em geral, métodos que usam dois vetores de pesos no cálculo da distância não diferem dos que usam apenas um único vetor.

CAPÍTULO 4

Método *K-Means* baseado em *Kernel* para Dados de Tipo Intervalo

A mente que se abre a uma nova idéia jamais voltará ao seu tamanho original.

—ALBERT EINSTEIN

4.1 Introdução

Os algortitmos de agrupamento *K-Means* são bastante populares. Isso deve-se ao fato deles serem simples de enterder e fáceis de implementar. Outra razão da sua popularidade é que são robustos, isto é, podem ser aplicados em diversos problemas, sejam eles com pequena ou grande quantidade de dados no agrupamento, ou de diferentes domínios como reconhecimento de padrões, aprendizagem de máquina, mineração de dados, visão computacional, biologia computacional, classificação de resultados estatísticos em estudos sociais, dentre outros. Entretanto, existem limitações nesses algoritmos, tais como a sensibilidade às posições iniciais dos protótipos das classes e a dificuldade de indentificar a não-linearidade dos dados no espaço de entrada.

O uso de *Kernel* tem sido usado como uma poderosa ferramenta para resolver os problemas relacinados à linearidade. Nos últimos anos, o interesse na utilização de métodos de *Kernel* em aplicações não-superviosionadas tem crescido. Em Aprendizagem de Máquina, a utilização das funções de *Kernel* foram introduzidas há muito tempo por Aizerman [10]. Em 1995, Cortes e Vapnik introduziram o *Support Vector Machines* (SVMs) [4], que têm melhores desempenhos que outros algoritmos classificadores nos mais variados problemas, por exemplo, *Kernel PCA* [5]. O método de *Kernel* é uma técnica desenvolvida especialmente para problemas não-linearmente separáveis de uma forma mais elegante, onde o produto interno entre as variáveis não-lineares é substituido por determinado *kernel* apropriado denominado *Mercer Kernel*, de tal forma que o agrupamento é realizado implicitamente em espaço de características, caracterizado por ser um espaço de mais alta dimensão. O uso do espaço de características durante a execução do algoritmo permite uma separação não-linear no espaço de entrada.

O método *Kernel K-Means* foi proposto como uma extensão do tradicional algoritmo K-Means [25], entretanto para dados clássicos, onde a representação não é tão rica em informação para a complexidade dos dados encontrados nos problemas reias, como os bancos de dados usados pela sociedade. Com o objetivo de mostrar a variabilidade e a incerteza presente nos dados, estes são representados por meio de conjuntos de intervalos. O estudo desses tipos de

dados é realizado, principalmente, pela Análise de Dados Simbólicos (ADS). A intenção do ADS é extender os métodos tradicionais com dados clássicos para métodos com dados de tipo intervalo, fazendo uso de técnicas estatísticas. A extensão do método *Kernel K-Means* para dados de tipo intervalo baseado em funções de *kernel* agregadas é a proposta deste trabalho.

4.2 *Kernel K-Means* definido por uma componente para dados de tipo intervalo

A idéia principal deste método (chamado KC) consiste em manipular funções de *kernel* para dados do tipo intervalo com o objetivo de calcular a distância entre os dois vetores que represetam os limites dos intervalos [23]. Cada objeto indexado por $i(i = 1,\ldots,n)$ da partição Ω é representado por um vetor de intervalos $y_i = ([a_i^1, b_i^1],\ldots,[a_i^p, b_i^p])^T$. Considere a função não-linear $\phi : y_i \rightarrow \phi(y_i)$ tal que mapeia valores de um espaço de entrada para um espaço de característica de mais alta dimensão H. Admita, também, os k centros $v_1^\phi,\ldots,v_k^\phi$ como os protótipos das classes, onde $v_k^\phi \subset H(k = 1,\ldots,K)$.

O objetivo é criar uma separação linear no espaço de característica, onde exista uma correspondência de uma separação não-linear no espaço de entrada. Inicialmente, os dados são mapeados para o espaço de característica de mais alta dimensão usando a função ϕ e então o método *K-Means* é aplicado neste espaço de característica. De acordo com a metodologia usada no método *Kernel K-Means*, o problema do agrupamento consiste em encontrar uma partição Ω composta por K classes disjuntas $C_1,\ldots,C_K$ tal que minimize o erro de agrupamento definido por:

$$E = \sum_{k=1}^{K} \sum_{i \in C_k} \|\phi(y_i) - v_k\|^2, \tag{4.1}$$

de forma que o protótipo da classe seja calculado da seguinte maneira:

$$v_k = \frac{\sum\limits_{i \in C_k} \phi(y_i)}{|C_k|}, \tag{4.2}$$

assim, a distância euclidiana quadrática pode ser expressa como:

$$\|\phi(y_i) - v_k\|^2 = \left\| \phi(y_i) - \frac{\sum\limits_{j \in C_k} \phi(y_j)}{|C_k|} \right\|^2, \tag{4.3}$$

expandindo:

$$\|\phi(y_i) - v_k\|^2 = \phi(y_i)\phi(y_i) - \frac{2\sum_{j \in C_k} \phi(y_i)\phi(y_j)}{|C_k|} + \frac{\sum_{j \in C_k}\sum_{l \in C_k} \phi(y_j)\phi(y_l)}{|C_k|^2}. \tag{4.4}$$

A função de *Kernel* $K(y_i, y_j)$ é usado para fornecer diretamente o produto interno $\phi(y_i) \cdot \phi(y_j)$ no espaço de características sem que a transformação ϕ seja explicitamente definida [24] [25]. A tabela a seguir mostra alguns exemplo mais populares de funções de *Kernel*:

Kernel Polinomial	$K(y_i, y_l) = (y_i^T y_l + \gamma)^\delta$
Kernel Gaussiano	$K(y_i, y_l) = exp(-\|y_i - y_l\|^2 / 2\sigma^2)$
Kernel Sigmoide	$K(y_i, y_l) = tanh(\gamma(y_i^T y_l) + \theta)$

Tabela 4.1 Exemplo de funções de *Kernel*

Onde a distância entre os elementos y_i e y_l usada na função de *Kernel* Gaussiano é equivalente à distância euclidiana quadrática:

$$\|y_i - y_l\|^2 = \sum_{j=1}^{p} \left[(a_i^j - a_l^j)^2 + (b_i^j - b_l^j)^2 \right] \tag{4.5}$$

Reformulando o produto interno em termos da função de *Kernel*, obtem-se:

$$\|\phi(y_i) - v_k\|^2 = K(y_i, y_i) - \frac{2\sum_{j \in C_k} K(y_i, y_j)}{|C_k|} + \frac{\sum_{j \in C_k} \sum_{l \in C_k} K(y_j, y_l)}{|C_k|^2}. \tag{4.6}$$

4.3 *Kernel K-Means* definido por dois diferentes componentes para dados de tipo intervalo

Nesta seção, será apresentado a extensão do método *Kernel K-Means* para dados do tipo intervalo, mas com uma principal diferença do que foi apresentado na seção anterior: o método é definido por dois diferentes componentes (chamado KCC) [26]. O objetivo é manipular funções de *Kernel* para dados intervalares onde haja um tratamento separado para os limites inferior e superior de cada intervalo.

Cada objeto indexado por $i (i = 1, \ldots, n)$ da partição Ω é representado por um vetor de intervalos $y_i = ([a_i^1, b_i^1], \ldots, [a_i^p, b_i^p])^T$. Considere a função não-linear $\xi : y_i \to \xi(y_i)$ tal que mapeia valores de um espaço de entrada para um espaço de característica de mais alta dimensão $\Im$.

Seja $y_{iL} = (a_i^1, \ldots, a_i^p)$ e $y_{iU} = (b_i^1, \ldots, b_i^p)$ os dois vetores dos limites inferior e superior dos intervalos descrevendo y_i. De forma semelhante, considere os k centros $g_k^\xi \subset \Im (k = 1, \ldots, K)$ como os protótipos das classes e descritos pelo par de vetores $g_{1L}^\xi, \ldots, g_{kL}^\xi$ e $g_{1U}^\xi, \ldots, g_{kU}^\xi$ relativo aos limites dos intervalos.

O objetivo desta versão do método, assim como a da sessão anterior, é criar uma separação linear no espaço de característica, onde exista uma correspondência de uma separação não-linear no espaço de entrada. Inicialmente, os dados são mapeados para o espaço de característica de mais alta dimensão usando a função ξ e então o método *K-Means* é aplicado neste espaço de característica. De acordo com a metodologia usada no método *Kernel K-Means*,

o problema do agrupamento consiste em encontrar uma partição Ω composta por K classes disjuntas $C_1, \ldots, C_K$ tal que minimize o erro de agrupamento definido por:

$$E = \sum_{k=1}^{K} \sum_{i \in C_k} \|\xi(y_{iL}) - g_{kL}\|^2 + \|\xi(y_{iU}) - g_{kU}\|^2, \tag{4.7}$$

de forma que o protótipo da classe seja calculado da seguinte maneira:

$$g_{kL} = \frac{\sum_{i \in C_k} \xi(y_{iL})}{|C_k|}, \tag{4.8}$$

relativo ao limite inferior. Enquanto para o limite superior, tem-se:

$$g_{kU} = \frac{\sum_{i \in C_k} \xi(y_{iU})}{|C_k|}, \tag{4.9}$$

assim, a distância euclidiana quadrática para o limite inferior pode ser expressa como:

$$\|\xi(y_{iL}) - g_{kL}\|^2 = K(y_{iL}, y_{iL}) - \frac{2\sum_{j \in C_k} K(y_{iL}, y_{jL})}{|C_k|} + \frac{\sum_{j \in C_k} \sum_{l \in C_k} K(y_{jL}, y_{lL})}{|C_k|^2}, \tag{4.10}$$

e para o limite superior:

$$\|\xi(y_{iU}) - g_{kU}\|^2 = K(y_{iU}, y_{iU}) - \frac{2\sum_{j \in C_k} K(y_{iU}, y_{jU})}{|C_k|} + \frac{\sum_{j \in C_k} \sum_{l \in C_k} K(y_{jU}, y_{lU})}{|C_k|^2}, \tag{4.11}$$

Na versão do método *Kernel K-Means* para dados intervalares com dois diferentes componentes, a função de *Kernel* Gaussiano é definida para os limites inferiores e superiores separadamente como:

$$K(y_i, y_l) = K(y_{iL}, y_{lL}) + K(y_{iU}, y_{lU}), \tag{4.12}$$

neste caso, tem-se:

$$K(y_i, y_l) = exp\left(\frac{-\|y_{iL} - y_{lL}\|^2}{2\sigma^2}\right) + exp\left(\frac{-\|y_{iU} - y_{lU}\|^2}{2\sigma^2}\right), \tag{4.13}$$

A distância entre os elementos y_{iL} e y_{lL} usada na função de *Kernel* Gaussiano mostrada na tabela anterior é equivalente à distância euclidiana quadrática. Para o limite inferior, faz-se:

$$\|y_{iL} - y_{lL}\|^2 = \sum_{j=1}^{p} \left[(a_i^j - a_l^j)^2 \right], \tag{4.14}$$

e para o limite superior, o cálculo é realizado de forma semelhante:

$$\|y_{iLU} - y_{lU}\|^2 = \sum_{j=1}^{p} \left[(b_i^j - b_l^j)^2 \right], \tag{4.15}$$

4.4 Algoritmo

O algoritmo faz uso de uma matriz de *Kernel* $K \in \Re^{n \times n}$ onde a posição (i, j) da matriz é da forma $K_{ij} = \phi(y_i)^T \phi(y_j)$, definindo o novo espaço de características de mais alta dimensão. Isto é, a matriz K armazena o valor das funções de *Kernel* calculado entre os possíveis pares de elemento $y_i (i = 1, \ldots, n)$ da partição Ω.

1. **Etapa de pré-processamento: definição do novo espaço de características**
   Compute a matriz de *Kernel* K usando a equação $K(y_i, y_l) = exp\left(\frac{-\|y_i - y_l\|^2}{2\sigma^2}\right)$ para o método *Kernel K-Means* definido por um componente, ou a equação $K(y_i, y_l) = exp\left(\frac{-\|y_{iL} - y_{lL}\|^2}{2\sigma^2}\right) + exp\left(\frac{-\|y_{iU} - y_{lU}\|^2}{2\sigma^2}\right)$ definido por dois diferentes componentes, para $i = 1, \ldots, n$ e $l = 1, \ldots, n$;

2. **Etapa de inicialização: obtenção da partição inicial**
   Escolha aleatoriamente uma partição $P = (C_1, \ldots, C_K)$ de Ω ou escolha, também aleatoriamente, K elementos diferentes $(v_1, \ldots, v_K)$ do conjunto de objetos Ω e associe cada elemento y_i ao objeto mais próximo v_{k*}, onde $k* = \arg\min_{k=1,\ldots,K} \sum_{j=1}^{p} [(a_i^j - \alpha_k^j)^2 + (b_i^j - \beta_k^j)^2]$ para construir a partição inicial $(C_1, \ldots, C_K)$;

3. **Etapa de alocação: definição da melhor partição**
   $teste \leftarrow 0$
   Para $i = 1, \ldots, n$ faça

   definia a classe vencedora C_{k*} tal que:

   Para KC:
   $k* = \arg\min_{k=1,\ldots,K} \|\phi(y_i) - v_k\|^2$;

   Para KCC:
   $k* = \arg\min_{k=1,\ldots,K} \|\xi(y_{iL}) - g_{kL}\|^2 + \|\xi(y_{iU}) - g_{kU}\|^2$;

se $i \in C_k$ e $k* \neq k$

$teste \leftarrow 1$

$C_{k*} \leftarrow C_{k*} \cup \{i\}$

$C_k \leftarrow C_k \setminus \{i\}$

4. **Critério de parada**
   Se $teste = 0$ então PARE, caso contrário siga na etapa 3.

Após o cálculo da matiz de *Kernel* K, o algoritmo busca uma partição P definida aleatoriamente a partir da uma já pronta ou construida usando elementos aleatórios como protótipos para fazer uma alocação segundo as distância mais próximas. A definição da melhor partição é semelhante aos métodos de nuvens dinâmicas apresentados na seção anterior, entretanto a principal ocorre na escolha da classe vencedora, uma vez que a distância usada para esse propósito é obtida a partir do espaço de características de mais alta dimensão representada pela matriz K. Segundo o algoritmo do método *Kernel K-Means*, o problema do agrupamento consiste em encontrar uma partição Ω composta por K classes disjuntas $C_1, \ldots, C_K$ tal que minimize o erro de agrupamento.

CAPÍTULO 5

Experimentos e Resultados

Uma teoria é algo que ninguém acredita, exceto a pessoa que a fez . Um experimento é algo que todos acreditam, exceto a pessoa que o fez.

—ALBERT EINSTEIN

5.1 Dados Sintéticos

Com o propósito de avaliar o desempenho do método proposto *Kernel K-Means* para dados do tipo intervalo em relação ao método de nuvens dinâmicas com distâncias adaptaticas, foram realizados experimentos com conjuntos de dados simbólicos intervalares sintéticos. O objetivo é que os conjuntos tenham os elementos dispostos de forma não-linear, para que os experimentos possam fornecer resultados referentes à eficiência dos métodos em encontrar partições de dados não-linearmente separáveis para o espaço de entrada.

Os conjuntos de dados sintéticos do tipo intervalo pertecentes ao $\Re^2$ foram criados a partir de pontos (y_1, y_2), os quais representam as sementes para um vetor de intervalos. Como se trata do espaço $\Re^2$, o vetor produzirá graficamente retângulos, de forma que podem ser expressos como:

$$([y_1 - \gamma_1/2, y_1 + \gamma_1/2], [y_2 - \gamma_2/2, y_2 - \gamma_2/2]) \tag{5.1}$$

onde os parâmetros γ_1 e γ_2 são selecionados aleatoriamente de um intervalo definido a priori. Além disso, os dados seguem a formação semelhante a espirais arquimedianas, onde cada braço da espiral é definida em cordenadas polares (r, θ) pela equação:

$$r = a + b\theta \tag{5.2}$$

sendo a e b números reais e θ em medidas angulares. O parâmetro a controla o giro da espiral, enquanto b está relaconado com a distância entre os sucessivos giros. Seguindo esse padrão, foram criados três conjuntos de dados simbólicos com duas, três e quatro classes.

O primeiro conjunto tem o total de 80 indivíduos, onde uma classe tem 30 e a outra tem 50 indivíduos e os parâmetros valendo $a = 10$, $b = 5$ e $\theta = 260$ graus ilustrado na figura 5.1:

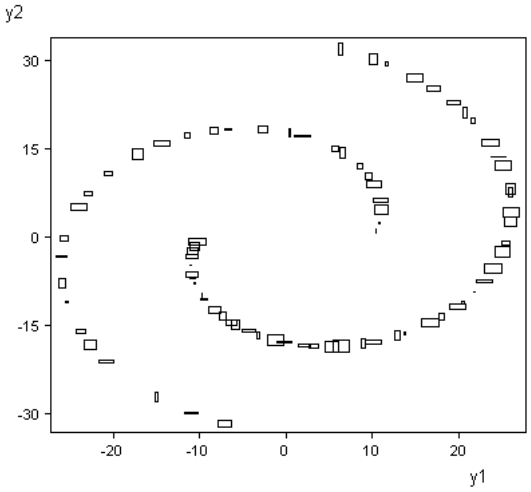

Figura 5.1 Conjunto com duas classes

O segundo conjunto possui 90 indivíduos e três classes, onde cada uma tem respectivamente 20, 30 e 40 indivíduos, os parâmetros a e b da equação têm os mesmos valores do conjunto anterior e $\theta = 200$ graus como mostra a figura 5.2:

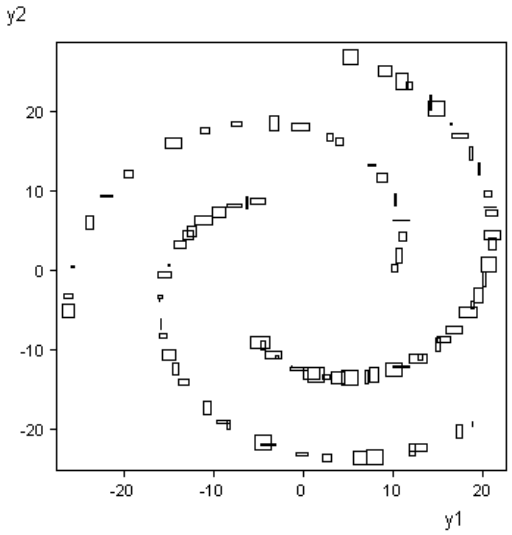

Figura 5.2 Conjunto com três classes

Enquanto o terceiro conjunto de dados simbólicos intervalares possui 90 indíviduos distribuidos entre as classes contendo 15, 20, 25 e 30. O ângulo total $\theta = 120$ graus tendo $a = 10$ e $b = 5$, representado pela figura 5.3:

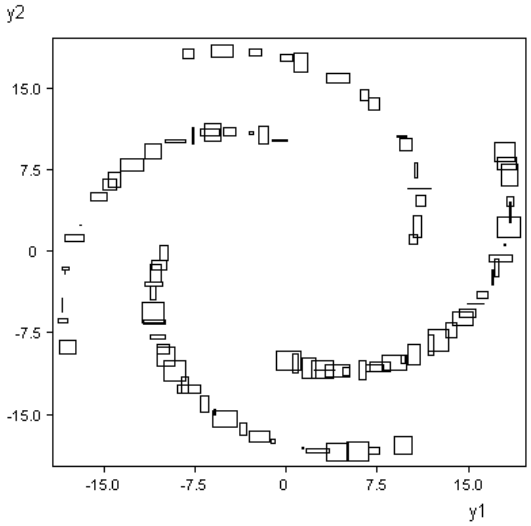

Figura 5.3 Conjunto com quatro classes

Como forma de avaliar o desempenho dos métodos, foi desenvolvido o experimento de Monte Carlo fornecendo a estimativa do índice corrigido de Rand (CR) e da taxa de erro de classificação global (OERC). Os valores de CR pertencem ao intervalo [-1,1], tal que quanto mais próximos de 1, mais semelhante a partição encontrada pelo método é à partição definida a priori. Enquanto os valores de OERC fazem parte do intervalo [0,1], de forma que aqueles perto de 0 estão associados às partições próximas da ideal. Foram realizados 100 replicações para cada conjunto de dados intervalares acima descritos variando aleatoriamente os parâmetros γ_1 e γ_2 de acordo com um dos intervalos predefinidos: [1,2], [1,10], [1,20], [1,30]. Desta forma, foram feitas o total de 400 replicações para cada conjunto.

A estimativa do índice corrigido de Rand é feita através da média das 100 replicações realizadas. Para cada replicação, um método de *cluster* é executado (até que nenhum elemento da partição mude de classe) 150 vezes variando aleatoriamente as sementes usadas na etapa de inicialização. O melhor critério dentre as repetições realizadas é escolhido e o CR e OERC são calculados para a partição correspondente.

Em todos os experimentos, os métodos propostos KC e KCC utilizam a função de *Kernel* Gaussino descrita na tabela 4.1, onde o valor do parâmetro σ definido a priori é igual a 1.

As tabelas 5.1, 5.2 e 5.3 a seguir mostram os resultados com dados sintéticos, com as quais pode-se analisar as médias dos índices corrigido de Rand e das taxas de erro, onde cada média possui um desvio-padrão associado entre parênteses. Os resultados comparam os métodos com distâncias adaptativas com um peso ou dois pesos e os métodos de *Kernel* propostos.

A tabela 5.1 mostra os resultados da comparação dos métodos de agrupamento para o primeiro conjunto de dados ilustrado pela figura 5.1. Observando o índice CR, as melhores perfomances foram obtidas pelos métodos baseados em *Kernel* para dados intervalares (KC e KCC). O pior desempenho foi obtido pelos métodos de agrupamento adaptativos em todos os casos (os métodos de distância adaptativa por classe CP e CPP, e única UP e UPP). Como esperado, para esse tipo de configuração com duas classes não-lineares, os métodos de *Kernel* também obtiveram desempenho superior aos métodos adaptativos considerenado a métrica OERC.

Tabela 5.1 Comparação dos métodos de agrupamento para o conjunto com duas classes

Intervalo	Nuvens Dinâmicas				Kernel K-Means	
Predefinido	SP	SPP	CP	CPP	KC	KCC
[1,2]	-0.067	-0.068	-0.037	-0.037	0.200	0.200
CR	(0.024)	(0.024)	(0.059)	(0.059)	(0.033)	(0.033)
[1,10]	-0.070	-0.069	-0.043	-0.042	0.207	0.207
CR	(0.026)	(0.023)	(0.051)	(0.051)	(0.038)	(0.038)
[1,20]	-0.066	-0.067	-0.039	-0.039	0.202	0.202
CR	(0.023)	(0.024)	(0.067)	(0.063)	(0.039)	(0.039)
[1,30]	-0.068	-0.067	-0.051	-0.043	0.175	0.175
CR	(0.025)	(0.020)	(0.065)	(0.071)	(0.041)	(0.041)
[1,2]	0.309	0.309	0.297	0.297	0.196	0.196
OERC	(0.008)	(0.008)	(0.032)	(0.032)	(0.008)	(0.008)
[1,10]	0.310	0.309	0.299	0.298	0.194	0.194
OERC	(0.011)	(0.009)	(0.029)	(0.030)	(0.009)	(0.009)
[1,20]	0.307	0.307	0.291	0.294	0.196	0.196
OERC	(0.012)	(0.013)	(0.038)	(0.035)	(0.009)	(0.010)
[1,30]	0.309	0.308	0.294	0.296	0.211	0.211
OERC	(0.096)	(0.008)	(0.037)	(0.037)	(0.017)	(0.017)

Os resultados apresentados na tabela 5.2 mostram a comparação dos métodos de agrupamento para o segundo conjunto de dados com três classes não-lineares representado pela figura 5.2. Assim como pode ser observado na tabela 5.1, os resultados para a configuração do segundo conjunto encontrados pelos métodos de *Kernel* foram superiores aos métodos com distâncias adaptativas presentes na literatura, mesmo que o problema de encontrar a partição mostra-se mais fácil que a configuração do primeiro conjunto analisando o índice CR.

Tabela 5.2 Comparação dos métodos de agrupamento para o conjunto com três classes

Intervalo	Nuvens Dinâmicas				Kernel K-Means	
Predefinido	SP	SPP	CP	CPP	KC	KCC
[1,2]	0.030	0.030	0.056	0.056	0.234	0.234
CR	(0.057)	(0.057)	(0.013)	(0.014)	(0.020)	(0.020)
[1,10]	0.020	0.022	0.056	0.057	0.238	0.238
CR	(0.063)	(0.062)	(0.015)	(0.016)	(0.022)	(0.022)
[1,20]	0.026	0.016	0.052	0.052	0.231	0.232
CR	(0.056)	(0.061)	(0.018)	(0.020)	(0.027)	(0.026)
[1,30]	0.014	0.005	0.038	0.054	0.212	0.212
CR	(0.061)	(0.055)	(0.028)	(0.035)	(0.042)	(0.042)
[1,2]	0.400	0.400	0.405	0.406	0.315	0.315
OERC	(0.028)	(0.028)	(0.027)	(0.028)	(0.011)	(0.011)
[1,10]	0.408	0.407	0.397	0.401	0.313	0.313
OERC	(0.033)	(0.031)	(0.024)	(0.027)	(0.011)	(0.011)
[1,20]	0.402	0.411	0.402	0.404	0.315	0.315
OERC	(0.027)	(0.034)	(0.024)	(0.026)	(0.015)	(0.014)
[1,30]	0.411	0.414	0.403	0.394	0.325	0.325
OERC	(0.033)	(0.030)	(0.021)	(0.024)	(0.024)	(0.024)

A tabela 5.3 apresenta os resultados dos experimentos realizados entre os métodos de agrupamento para o terceiro conjunto de dados como mostra a figura 5.3. Analisando a métrica CR, é possível perceber que esta configuração de dados é a mais fácil de reconhecer os padrões quando comparada com os outros dois conjuntos. Mesmo assim, o método proposto *Kernel K-Means* para intervalos mostrou-se mais eficiente neste reconhecimento em relação aos métodos de nuvens dinâmicas.

Tabela 5.3 Comparação dos métodos de agrupamento para o conjunto com quatro classes

Intervalo	Nuvens Dinâmicas				Kernel K-Means	
Predefinido	SP	SPP	CP	CPP	KC	KCC
[1,2]	0.084	0.084	0.151	0.151	0.246	0.246
CR	(0.099)	(0.099)	(0.019)	(0.018)	(0.022)	(0.022)
[1,10]	0.101	0.092	0.140	0.143	0.247	0.247
CR	(0.120)	(0.111)	(0.016)	(0.018)	(0.021)	(0.021)
[1,20]	0.136	0.141	0.141	0.149	0.254	0.254
CR	(0.132)	(0.136)	(0.025)	(0.031)	(0.033)	(0.032)
[1,30]	0.179	0.152	0.161	0.176	0.244	0.247
CR	(0.130)	(0.103)	(0.042)	(0.051)	(0.044)	(0.044)
[1,2]	0.443	0.443	0.470	0.468	0.327	0.328
OERC	(0.068)	(0.068)	(0.088)	(0.089)	(0.026)	(0.030)
[1,10]	0.427	0.433	0.405	0.411	0.329	0.329
OERC	(0.074)	(0.069)	(0.031)	(0.048)	(0.032)	(0.032)
[1,20]	0.413	0.410	0.400	0.396	0.341	0.340
OERC	(0.088)	(0.096)	(0.018)	(0.024)	(0.056)	(0.055)
[1,30]	0.383	0.401	0.396	0.381	0.367	0.367
OERC	(0.083)	(0.071)	(0.034)	(0.034)	(0.074)	(0.073)

Comparando os resultados de todos os conjuntos apresentados, nota-se que os métodos de nuvens dinâmicas com distância adaptativa de dois pesos possuem uma performance similar aos métodos de um peso. Entretanto, existem diferenças entre os métodos com distância adaptativa única e por classe, de forma que o primeiro possui um desempenho inferior ao segundo em todos os casos. Estas diferenças ocorrem pelo fato de que os métodos de nuvens dinâmicas com distâncias quadráticas L_2 adaptativas depende da estrutura intra-classe definida a priori [21]. Como os conjuntos de espirais de dados intervalares possuem as classes definidas a priori com dispersões diferentes, o melhor desempenho foi obtido pelos métodos com distâncias adaptativas por classe.

5.2 Dados Reais

Foram realizados experimentos com conjuntos de dados simbólicos intervalares reais com o objetivo de avaliar o desempenho do método proposto *Kernel K-Means* para dados do tipo intervalo em relação ao método de nuvens dinâmicas com distâncias adaptaticas. Foram feitos experimentos com os conjuntos de dados reais intervalares *City-Temperature* e *Agaricus*. Para comparar os resultados obtidos a partir dos experimentos, foram utilizadas as mesmas métricas usadas em experimentos com dados sintéticos: o índice corrigido de Rand (CR) e da taxa de erro de classificação global (OERC).

City-Temperature

O conjunto de dados reais intervalares *City-Temperature* [29] consiste de informações relativas a 37 cidades, cada uma descrita por 12 variáveis intervalares com o máximo e o mínimo

de temperatura em graus centígrados durante 12 meses. A tabela a seguir mostra a estrutura desse conjunto.

Tabela 5.4 Mínimo e máximo das temperaturas das cidades em graus centrígrados

Cidades	Janeiro	Fevereiro	...	Novembro	Dezembro
Amsterdan	$[-4,4]$	$[-5,3]$	...	$[1,10]$	$[-1,4]$
Athens	$[6,12]$	$[6,12]$	...	$[11,18]$	$[8,14]$
Bahrain	$13,19]$	$[14,19]$	...	$[20,26]$	$[15,21]$
Bombay	$[19,28]$	$[19,28]$	...	$[14,26]$	$[10,20]$
Calcutta	$[13,27]$	$[16,29]$	...	$[18,29]$	$[13,26]$
Colombo	$[22,30]$	$[22,30]$	...	$[23,29]$	$[22,30]$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Toronto	$[-8,-1]$	$[-8,-1]$	...	$[-1,17]$	$[-5,1]$
Vienna	$[-2,1]$	$[-1,3]$	...	$[2,7]$	$[1,3]$
Zurich	$[-11,9]$	$[-8,15]$	...	$[0,19]$	$[-11,8]$

O conjunto possui 4 classes dispostas da seguinte forma:

a) Classe 1: Bahraim: 3, Bombay: 4, Cairo: 5, Calcutta: 6, Colombo: 7, Dubal: 9, Hong Kong: 12, Kula Lampur: 13, Madras: 16, Manila: 18, Mexico: 20, Nairobi: 23, New Delhi: 24, Sydney: 30 e Singapore: 32

b) Classe 2: Amsterdam: 1, Athens: 2, Copenhagen: 8, Frankfurt: 10, Geneva: 11, Lisbon: 14, London: 15, Madrid: 17, Moscow: 21, Munich: 22, New York: 25, Paris: 26, Rome: 27, San Francisco: 28, Seoul: 29, Stockholm: 31, Tokyo: 34, Toronto: 35, Vienna: 36 e Zurich: 37

c) Classe 3: Mauritius: 19

d) Classe 4: Tehran: 33

Cada método é executado (até que nenhum elemento da partição mude de classe) 300 vezes variando aleatoriamente as sementes usadas na etapa de inicialização. O melhor critério dentre as repetições realizadas é escolhido e o CR e OERC são calculados para a partição correspondente.

A tabela a seguir mostra a configuração da partição a priori e da obtida após aplicar os algoritmos dos métodos de nuvens dinâmicas e o método proposto *Kernel K-Means* para dados intervalares.

Tabela 5.5 Resultado dos agrupamentos para o conjunto *City-Temperature*

	Classe 1	Classe 2	Classe 3	Classe 4
Partição a priori	3 4 5 6 7 9 12 13 16 18 20 23 24 30 32	1 2 8 10 11 14 15 17 21 22 25 26 27 28 29 31 34 35 36 37	19	33
SP	3 4 6 7 9 13 16 18 24 30	1 8 10 11 15 17 21 22 25 26 28 29 31 36 35 37	2 5 12 14 27 33 34	19 20 23 32
CP	3 6 7 13 16 18 30	1 2 8 10 11 14 15 17 21 22 25 26 27 28 29 31 33 34 35 36 37	3 5 9 12 24	19 20 23 32
KC	3 4 6 7 9 13 16 18 19 30	1 2 8 10 11 14 15 17 21 22 25 26 27 28 29 31 34 35 36 37	5 12 20 23 24 32	33
SPP	4 6 7 13 16 28 24 30	1 8 19 11 14 15 17 21 22 25 26 27 28 29 31 34 35 36 37	2 3 5 9 12 33	19 20 23 32
CPP	4 6 7 13 16 18 24 30	1 8 10 11 14 15 17 21 22 25 26 27 28 29 31 34 35 36 37	2 3 5 9 12 33	19 20 23 32
KCC	4 6 7 9 13 16 18 19 30	1 2 8 10 11 14 15 17 21 22 25 26 27 28 29 31 34 35 36 37	3 4 12 20 23 24 32	33

Os índices CR obtidos da comparação entre a partição a priori e a obtida após a execução do algortimo de agrupamento para os métodos de nuvens dinâmicas SP e CP e o método *Kernel K-Means* KP foram, respectivamente, 0,505, 0,629 e 0,713 e para os métodos SPP, CPP e KPP foram 0,634, 0,628 e 0,672. O OERC para estas parições e entre os métodos SP, CP, KP valem, respectivamente, 0,297, 0,270 e 0,189 e para os métodos SPP, CPP e KPP valem 0,270, 0,243 e 0,216. Desta forma, a performance do método *Kernel K-Means* é superior aos dos métodos de nuvens dinâmicas adaptativos para esse conjunto de dados reais.

Agaricus

Outro conjunto de dados reais do tipo intervalo avaliado foi *Agaricus* [30], cujas informações foram extraidas de espécies de *fungi* presentes no estado da California, Estados Unidos. *Agaricus* é um gênero de cogumelos, tais que dois membros bastante conhecidos são o *A. bisporus*, cognominado como o cogumelo botão muito comum em supemercados e o *A. campestris*, também chamado de cogumelo de campo, é uma espécie popular comestível comum em pastagens e prados em áreas temperadas no mundo.

Esse conjunto possui 24 espécies do gênero *Agaricus*, onde cada uma é descrita por 5 variáveis do tipo intervalo: largura do píleo, largura do estipe, espessura do estipe, altura e largura dos esporos. Os dados são distribuidos entre duas classes: comestível e não-comestível. A tabela a seguir descreve as espécies de acordo com suas variáveis:

Tabela 5.6 Informações sobre as espécies de *Agaricus*

Espécies	largura do píleo	largura do estipe	espessura do estipe	altura dos esporos	largura dos esporos
Albolutescens	[6,12]	[2,7]	[1.5,3]	[6,7.5]	[4,5]
Arvensis	[6,21]	[4,14]	[1,3.5]	[6.5,9]	[4.5,6]
Benesii	[4,8]	[5,11]	[1,2]	[5,6]	[3,4]
Bernardii	[7,16]	[4,7]	[3,4.5]	[5.5,7]	[5.5,6.5]
Bisporus	[5,12]	[2,5]	[1.5,2.5]	[6,8.5]	[4.5,6]
Bitorquis	[5,15]	[4,10]	[2,4]	[5,6.5]	[4,5.5]
⋮	⋮	⋮	⋮	⋮	⋮
Subrutilescens	[6,14]	[6,12]	[1,2]	[4,6]	[3.5,4.5]
Subrufescens	[6,13]	[6,12]	[1.5,2.5]	[5.5,6.5]	[4,4.5]
Xanthodermus	[5,17]	[4,14]	[1,3.5]	[5,6]	[4,5.5]

De forma que a distribuição dos elementos entre as duas classes é descrita como:

a) Classe 1: Albolutescens: 1, Arvensis: 2, Benesii: 3, Bernardii: 4, Bisporus: 5, Bitorquis: 6, Campestris: 8, Comtulus: 9, Cupreo-brunneus: 10, Fusco-fibrillosus: 11, Fuscovelatus: 12, Lilaceps: 14, Micromegathus: 15, Pattersonae: 16, Perobscurus: 17, Semotus: 19, Silvicola: 20, Smithii: 21, Subrutilescens: 22 e Subrufescens: 23

b) Classe 2: Californicus: 7, Hondensis: 13, Praeclaresquamosus: 18, Xanthodermus: 24

A tabela a seguir mostra a configuração da partição a priori e da obtida após aplicar os algoritmos dos métodos de nuvens dinâmicas e o método proposto *Kernel K-Means* para dados intervalares.

Tabela 5.7 Resultado dos agrupamentos para o conjunto *Agaricus*

	Classe 1	Classe 2
Partição a priori	1 2 3 4 5 6 8 9 10 11 12 14 15 16 17 19 20 21 22 23	7 13 18 24
SP	1 2 3 4 5 6 7 8 9 10 11 12 14 15 16 18 19 20 21 22 23 24	13 17
CP	2 3 4 5 6 7 8 9 10 11 12 14 15 16 17 19 20 21 22 24	1 13 17 23
KC	2 3 4 5 6 7 8 9 10 11 14 15 16 18 19 20 21 22 24	1 12 13 17 23
SPP	1 2 3 4 5 6 7 8 9 10 11 12 14 15 16 18 19 20 21 22 23 24	13 17
CPP	1 2 3 4 5 6 7 8 9 10 11 14 15 16 18 19 20 21 22 23 24	12 13 17
KCC	2 3 4 5 6 7 8 9 10 11 14 15 16 18 19 20 21 22 24	1 12 13 17 23

Os resultados obtidos após a execução dos algoritmos de agrupamento avaliados através do índice CR comparando-se a partição a priori e a obtida pelos métodos SP, CP e KP foram, respectivamente, 0,127, 0,261 e 0,458 e para os métodos SPP, CPP e KPP foram 0,127, 0,390 e 0,458. Os OERC medidos para as partições correspondentes aos métodos SP, CP e KP valem, respectivamente, 0,167, 0,167 e 0,125 e para os métodos SPP, CPP, e KPP valem 0,167, 0,125 e 0,125. Desta forma, avaliando a métrica OERC juntamente com o índice CR, a performance do método *Kernel K-Means* é superior aos dos métodos de nuvens dinâmicas adaptativos para o conjunto *Agaricus*.

CAPÍTULO 6

Conclusão e Trabalhos Futuros

Temos o destino que merecemos. O nosso destino está de acordo com os nossos méritos.

—ALBERT EINSTEIN

6.1 Conclusão

Neste trabalho foi apresentado um novo método de agrupamento para dados intervalares baseado em funções de *Kernel*. Este método é uma extensão do *Kernel K-Means*, onde a principal diferença ocorre na capacidade de interpretar dados do tipo intervalo. A motivação desse estudo está relacionada à investigação de métodos que possam criar uma separação não-linear dos dados intervalares entre os hiper-espaços dos *clusters*, uma vez que métodos presentes na literatura de ADS, tais como os de nuvens dinâmicas descritos nesse trabalho, não são tão eficientes para esse propósito.

Inicialmente foi introduzido o *Kernel K-Means* definido por uma componente para dados de tipo intervalo com o objetivo de calcular a distância a partir de dois vetores de intervalos usando funções de *Kernel*. Em seguida, o *Kernel K-Means* definido por dois diferentes componentes, onde existe um tratamento separado para os limites inferior e superior de cada intervalo. Em ambos os casos, o objetivo é identificar a não-linearidade dos dados ao mesmo tempo em que estes são representados por meio de conjuntos de intervalos, caracterizando a variabilidade e a incerteza das informações.

Foram realizados experimentos com o uso da técnica de Monte Carlo com conjuntos de dados simbólicos intervalares. Nos conjuntos sintéticos, o objetivo é que os experimentos possam fornecer resultados referentes à eficiência dos métodos em encontrar partições de dados não-linearmente separáveis para o espaço de entrada. Os métodos usados para comparação com o métodos proposto *Kernel K-Means* para dados intervalares foram os de nuvens dinâmicas com distâncias adaptativas por classe e única e com um peso e dois pesos. Também foram feitos experimentos com dados simbólicos intervalares reais, com o conjunto *City-Temperature* e o *Agaricus*. Para isto, adoutou-se o índice de corrigido de Rand (CR) e a taxa de erro de classificação global (OERC) como forma de avaliar os métodos.

Os resultados com dados sintéticos mostram a capacidade do método proposto *Kernel K-Means* para dados intervalares em reconhecer com melhor performance configurações onde os dados são dispostos de forma não-linear, como pôde ser demonstrada com os conjuntos em forma de espiral. Como conclusão principal, o método mostrou-se mais eficiente do que os métodos de nuvens dinâmicas em todas as situações, tanto para dados sintéticos quanto

para os reais, quando avaliados através do CR e OERC conjuntamente. Desta forma, este trabalho contribuiu para o desenvolvimento de métodos para o reconhecimento de dados não-linearmente distribuidos, obtendo um avanço teórico no âmbito dos algoritmos e da abordagem simbólica.

6.2 Trabalhos Futuros

Um dos principais problemas encontrados no método *Kernel K-Means* para dados intervalares foi a sua sensibilidade à escolha dos protótipos para a definição da partição inicial. Desta forma, existe a necessidade de um número maior de execuções do algoritmo para obter a partição de menor critério quando comparado com os métodos de nuvens dinâmicas. Assim, é importante o estudo de novas técnicas para a definição da partição inicial ou novas maneiras de diminuir a sensibilidade do método proposto.

Existe, também, a possibilidade de analisar o desempenho do método verificando as diferentes funções de *Kernel*, tais como o Polinomial e o Sigmoide. Assim, pode ser possível concluir quais tipos de funções possuem uma melhor performance com determinados problemas onde os dados são organizados não-linearmente.

Outro ramo de pesquisa para o desenvolvimento do método é tornar possível que distância, usada para medir o erro entre o elemento e o protótipo, reconheça possíveis diferenças de dispersões encontradas no conjunto de dados. Uma alternativa é fazer com que o sigma usado na função de *Kernel* Gaussiano varie de acordo com as informações obtidas através dos intervalos, elementos, classe, ou de toda a partição.

Referências Bibliográficas

[1] JAIN, A. K.; MURTY, M. N.; FLYNN, P. J. **Data clustering: a review**. ACM Computing Surveys, p. 264323, 1999. 1

[2] XU, R.; WUNSCH, D. I. I. **Survey of clustering algorithms**. IEEE Transactions on Neural Networks, p. 645678, 2005. 1

[3] DIDAY, E.; NOIRHOMME-FRAITURE, M. **Symbolic Data Analysis And The Sodas Software**. Wiley- Interscience Publishers, 2008. 1

[4] CAMASTRA, F.; VERRI, A. **A novel kernel method for clustering**. IEEE Transactions on Pattern Analysis and Machine Intelligence, p. 801-804, 2005. 1, 4.1

[5] SCHOLKOPF, B.; SMOLA, A. J.; MULLER, K. R. **Nonlinear component analysis as a kernel eigenvalue problem**. Neural Computation, p. 1299-1319, 1998. 1, 4.1

[6] DE CARVALHO, F.A.T.; DE SOUZA, R.M.C.R.; CHAVENT, M.; LECHEVALLIER, Y. **Adaptive Hausdorff distances and dynamic clustering of symbolic data**. Pattern Recognition Letters, p. 167-179, 2006. 1

[7] DE CARVALHO, F.A.T.; BOCK, P. BRITO H.-H. **Dynamic Clustering for Interval Data Based on L2 Distance**. Computational Statistics, p. 231-250, 2006. 1

[8] DE SOUZA, R.M.C.R.; DE CARVALHO, F.A.T. **Clustering of interval data based on city-block distances**. Pattern Recognition Letter, p. 353-365, 2004. 1

[9] GIROLAMI, M. **Mercer kernel based clustering in feature space**. IEEE Transactions on Neural Networks, p. 780-784, 2002. 1

[10] FILIPPONE, M.; CAMASTRA, F.; MASULLI, F.; ROVETTA, S. **A survey of kernel and spectral methods for clustering**. Pattern Recognition, p. 176-190, 2008. 1, 4.1

[11] DHILLON, I. S.; GUAN, Y.; KULIS, B. **Kernel k-means, spectral clustering and normalized cuts**. In: Proc. ACMSIGKDD Intl Conf Knowledge Discovery and Data Mining, Seattle, WA, 2004. 1, 4.1, 4.2

[12] SERGIOS, THEODORIS; KOUTROUMBAS, KONSTANTINES. **Pattern Recognition**. Academic Press, 3a ed., 2006. 2.1

[13] BALL, G. H.; HALL, D. J. **A Clustering Technique for Summarizing Multivariate Data**. Behavioral Science, v. 12, p. 153-155, 1967. 2.2.1

[14] HASTIE, TREVOR; TIBSHIRANI, ROBERT; FRIEDMAN, JERONE. **The Elements of Statistical Learning**. Springer, 2a ed., 2009. 2.3.2

[15] YANG, XIN-SHE. **Introduction to Mathematical Optimization - From Linear Programming to Metaheuristics**. Cambridge International Science Publishing, 2008.

[16] MITRA, SUSHMITA; K. Pal, SANKAR. **Fuzzy sets in pattern recognition and machine intelligence**. Fuzzy Sets and Systems, p. 381386, 2005. 2.4.2

[17] DUDA, R. O.; HART, P. E.; STORK, D. G. Pattern Classification. John Wiely and Sons, 2001. 2.4.1

[18] HEYER, LJ; KRUGLYAK, S; YOOSEPH, S. **Exploring expression data: identification and analysis of coexpressed genes**. Genome Research, 1999. 2.5

[19] BANERJEE, A.; KRUMPELMAN, C.; GHOSH, J.; BASU, S.; MOONEY, R. J. **Model-based overlapping clustering**. Proc Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), p. 532537, 2005. 2.5

[20] DIDAY, E.; GOVAERT, G. **Classification Automatique avec Distances Adaptatives**. R.A.I.R.O. Informatique Computer Science, p. 329349, 1977. 3.3.1

[21] DE CARVALHO, F.A.T.; LECHEVALLIER, Y. **Partitional clustering algorithms for symbolic interval data based on single adaptive distances**. Pattern Recognition, p. 12231236, 2009. 3.3.2, 3.3.2, 3.3.2, 5.1

[22] DE CARVALHO, F.A.T.; LECHEVALLIER, Y. **Dynamic clustering of interval-valued data based on adaptive quadratic distances**. IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans, p. 1295-1306, 2009. 3.3.2

[23] COSTA, A. F. B. F. ; PIMENTEL, B. A. ; DE SOUZA, R. M. C. R. **A Kernel K-means Clustering Method for Symbolic Interval Data**. International Joint Conference on Neural Networks (IJCNN 2010), 2010, Barcelona. 4.2

[24] TZORTZIS ,G.; LIKAS, A. **The global kernel k-means clustering algorithm**. Proc. IEEE Int. Joint Conf. Neural Networks (IJCNN), p.19771984, Junho 2008. 4.2

[25] DHILLON, I.; GUAN, Y.; KULIS, B. **Kernel k-means, spectral clustering and normalized cuts**. in Proc. 10th ACM KDD Conference, 2004. 1, 4.1, 4.2

[26] COSTA, A. F. B. F. ; PIMENTEL, B. A. ; DE SOUZA, R. M. C. R. **K-means Clustering for Symbolic Interval Data based on Aggregated Kernel Functions**. International Conference on Tools with Artificial Intelligence (ICTAI 2010), 2010, France. 4.3

[27] CELEUX, G.; DIDAY, E.; GOVAERT, G.; LECHEVALLIER, Y.; RALAMBONDRAINY, H. Classification Automatique des Données. Paris, France: Bordas, 1989.

[28] DE CARVALHO, F. A. T.; BRITO, P.; BOCK, H.-H. **Dynamic clustering for interval data based on L2 distance**. Comput. Stat., vol. 21, no. 2, p. 231250, Junho 2006.

[29] GURU, D.S.; KIRANAGI, B.B.; NAGABHUSHAN, P. **Multivalued type dissimilarity measure and concept of mutual dissimilarity value for clustering symbolic patterns**. Pattern Recognition, p. 12031213, 2004. 5.2

[30] WOOD, M.; STEVENS, F. ***Agaricus***. http://www.mykoweb.com/CAF/genera/Agaricus.html. Acessado em Julho, 2010. 5.2

APÊNDICE A

Assinaturas

Renata Maria Cardoso Rodrigues de Souza
Orientadora

Bruno Almeida Pimentel
Aluno