Rogério Rodrigues de Vargas

Uma Nova Forma de Calcular os Centros dos Clusters em Algoritmos de Agrupamento Tipo Fuzzy C-Means

Natal - RN

2012

Rogério Rodrigues de Vargas

Uma Nova Forma de Calcular os Centros dos Clusters em Algoritmos de Agrupamento Tipo Fuzzy C-Means

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Sistemas e Computação da Universidade Federal do Rio Grande do Norte.

Orientador:

Dr. Benjamín René Callejas Bedregal

UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

Natal - RN

2012

UFRN / Biblioteca Central Zila Mamede
Catalogação da Publicação na Fonte

Vargas, Rogério Rodrigues de.
Uma nova forma de calcular os centros dos clusters em algoritmos de agrupamento tipo *Fuzzy C-Means* / Rogério Rodrigues de Vargas. – Natal, RN, 2012.
97 f. : il.

Orientador: Prof. Dr. Benjamín René Callejas Bedregal.

Tese (Doutorado) – Universidade Federal do Rio Grande do Norte. Centro de Tecnologia. Programa de Pós-Graduação em Sistemas de Computação.

1. Algoritmos de agrupamento - Tese. 2. Centros de clusters - Tese. 3. ckMeans - Tese. 4. Fuzzy C-Means - Tese. 5. Dados intervalares - Tese. 6. Lógica Fuzzy – Tese. I. Bedregal, Benjamín René Callejas. II. Universidade Federal do Rio Grande do Norte. V. Título.

RN/UF/BCZM CDU
004.021

ROGERIO RODRIGUES DE VARGAS

Uma Nova Forma de Calcular os Centros dos Clusters em Algoritmos de Agrupamento Tipo Fuzzy C-Means

Esta Tese foi julgada adequada para a obtenção do título de doutor em Ciência da Computação e aprovado em sua forma final pelo Programa de Pós-Graduação em Sistemas e Computação do Departamento de Informática e Matemática Aplicada da Universidade Federal do Rio Grande do Norte.

Prof. Dr. Benjamin Rene Callejas Bedregal – UFRN
Orientador

Prof. Dr. Martin Alejandro Musicante – UFRN
Coordenador do Programa

Banca Examinadora

Prof. Dr. Benjamin Rene Callejas Bedregal – UFRN
Presidente

Prof. Drª. Anne Magály de Paula Canuto – UFRN

Prof. Dr. Regivan Hugo Nunes Santiago – UFRN

Prof. Drª. Renata Hax Sander Reiser – UFPel

Prof. Dr. Ronei Marcos de Morais – UFPB

Março, 2012

Este trabalho é dedicado aos meus pais,
Élida Vargas e Jesus Vargas, por tudo
o que eles representam em minha vida.

Agradecimentos

Chegando ao fim de mais uma jornada, gostaria de agradecer a todos aqueles que em contextos diferentes, contribuíram para que eu chegasse até aqui. Na impossibilidade de citar todos os nomes que fizeram a travessia comigo nesta construção, permito-me a destacar os seguintes nomes, na certeza de que somos gratos a todos que direta ou indiretamente contribuíram para o sucesso deste trabalho.

A Deus por tudo.

Minha gratidão ao meu orientador, prof Benjamín R. C. Bedregal, por ter me acolhido com tanta paciência e carinho. Por ter sido um guia em meio à escuridão que era o início da minha pesquisa. Por ter sido firme e prestativo nos momentos cruciais. Por seu conhecimento. Por sua amizade.

Aos meus pais, pela sólida formação e exemplo de vida dada até hoje em minha vida, que me proporcionou a continuidade nos estudos até o fim deste doutorado, meus eternos agradecimentos.

Aos amigos e colegas do Departamento de Informática e Matemática Aplicada (DIMAp) pelo agradável convívio. Em especial gostaria de agradecer aos colegas Macilon, Eduardo e Araken pelas diversas vezes que saímos para almoçar ou jantar. Vocês tornaram-se grandes amigos para mim.

À Daiane. Minha companheira de pesquisa, minha companheira de vida. Ainda não foram inventadas palavras que possam traduzir o sentimento que habita em mim. Não há palavras que possam explicar a gratidão que tenho por estar ao meu lado, carinho, compreensão, paciência, estímulo, esforço, troca e aconchego.

À Capes (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) pelo auxílio financeiro.

“Uns são homens;
Alguns são professores;
Poucos são mestres.
Aos primeiros, escuta-se;
Aos segundos, respeita-se;
Aos últimos, segue-se.
Se hoje enxergo longe, é porque
fui colocado em ombros de gigantes!”
Autor Desconhecido

Resumo

Agrupar dados é uma tarefa muito importante em mineração de dados, processamento de imagens e em problemas de reconhecimento de padrões. Um dos algoritmos de agrupamentos mais popular é o Fuzzy C-Means (FCM). Esta tese propõe aplicar uma nova forma de calcular os centros dos clusters no algoritmo FCM, que denominamos de ckMeans, e que pode ser também aplicada em algumas variantes do FCM, em particular aqui aplicamos naquelas variantes que usam outras distâncias. Com essa modificação, pretende-se reduzir o número de iterações e o tempo de processamento desses algoritmos sem afetar a qualidade da partição ou até melhorar o número de classificações corretas em alguns casos. Também, desenvolveu-se um algoritmo baseado no ckMeans para manipular dados intervalares considerando graus de pertinência intervalares. Este algoritmo possibilita a representação dos dados sem conversão dos dados intervalares para pontuais, como ocorre com outras extensões do FCM que lidam com dados intervalares. Para validar com as metodologias propostas, comparou-se o agrupamento ckMeans com os algoritmos K-Means (pois o algoritmo proposto neste trabalho para cálculo dos centros se assemelha à do K-Means) e FCM, considerando três distâncias diferentes. Foram utilizadas várias bases de dados conhecidas. No caso, os resultados do ckMeans intervalar, foram comparadas com outros algoritmos de agrupamento intervalar quando aplicadas a uma base de dados intervalar com a temperatura mínima e máxima do mês de um determinado ano, referente a 37 cidades distribuídas entre os continentes.

Palavras-chaves: Agrupamentos, Centros dos Clusters, ckMeans, Fuzzy C-Means, Dados Intervalares, Lógica Fuzzy.

Abstract

A New Way to Calculate the Clusters Centers of Clustering Fuzzy C-Means-Type Algorithms

Clustering data is a very important task in data mining, image processing and pattern recognition problems. One of the most popular clustering algorithms is the Fuzzy C-Means (FCM). This thesis proposes to implement a new way of calculating the cluster centers in the procedure of FCM algorithm which are called ckMeans, and in some variants of FCM, in particular, here we apply it for those variants that use other distances. The goal of this change is to reduce the number of iterations and processing time of these algorithms without affecting the quality of the partition, or even to improve the number of correct classifications in some cases. Also, we developed an algorithm based on ckMeans to manipulate interval data considering interval membership degrees. This algorithm allows the representation of data without converting interval data into punctual ones, as it happens to other extensions of FCM that deal with interval data. In order to validate the proposed methodologies it was made a comparison between a clustering for ckMeans, K-Means and FCM algorithms (since the algorithm proposed in this paper to calculate the centers is similar to the K-Means) considering three different distances. We used several known databases. In this case, the results of Interval ckMeans were compared with the results of other clustering algorithms when applied to an interval database with minimum and maximum temperature of the month for a given year, referring to 37 cities distributed across continents

Keywords: ckMeans, Cluster Center, Clustering, Fuzzy C-Means, Fuzzy Logic.

Sumário

Lista de Abreviaturas

CAD	Computer Aidded Design
Capes	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
C-XSC	C++ Class Library for eXtended Scientific Computing
FCM	Fuzzy C-Means
Ibope	Instituto Brasileiro de Opinião Pública e Estatística
IFCM	Interval Fuzzy C-Means
MSV	Mutual Similarity Value
PSDB	Partido da Social Democracia Brasileira
PT	Partido dos Trabalhadores
SDA	Symbolic Data Analysis
SGBD	Softwares de Gerenciamento de Banco de Dados
UFRN	Universidade Federal do Rio Grande do Norte

Lista de Figuras

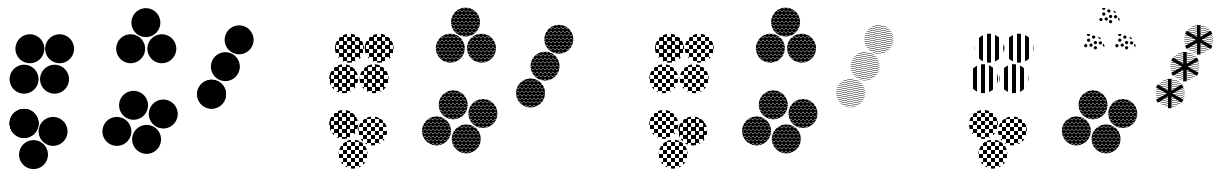
Lista de Tabelas

1 *Introdução*

Por permitir um enorme poder de processamento, os microprocessadores podem manipular grandes volumes de dados compostos por múltiplos atributos. Isto acontece em várias áreas da vida real, em bases de dados corporativos, financeiros, telecomunicações, medicina, imagens de satélite, etc. Muitas são as aplicações que buscam o gerenciamento de bases de dados a fim de aprender sobre a identificação dos dados em grupos, isso pode ser realizado com a utilização de algum algoritmo adequado de agrupamento de dados. Aplicações e desenvolvimento de novos algoritmos ainda é um grande desafio.

O agrupamento de dados em **clusters** (grupos) pode oferecer uma maneira de entender e extrair informações relevantes de conjuntos de dados. A ideia é que dados de um mesmo grupo tenham mais características em comum entre si do que com dados de qualquer outro grupo.

A Figura 1.1 apresenta um exemplo de agrupamento de dados. A Figura 1.1(a) mostra o conjunto de dados que será separado em grupos, onde cada padrão corresponde a um **cluster**; na Figura 1.1(b) os dados são separados em dois **clusters**; na Figura 1.1(c) em três **clusters** e na Figura 1.1(d) em cinco **clusters**.



(a) *Dados Iniciais* (b) *Dois Clusters* (c) *Três Clusters* (d) *Cinco Clusters*

Figura 1.1: Exemplo de agrupamento de dados.

O agrupamento de dados é relevante em diversas áreas de pesquisa, como mineração de dados (TAN; STEINBACH; KUMAR, 2006), aprendizado de máquina (MITCHELL, 1997)

e reconhecimento de padrões (BISHOP, 2006). Entre as aplicações realizadas nessas áreas, encontram-se a segmentação de imagens (CELENK, 1990), bioinformática e biologia da computação (JIN; WANG, 2009), (JIANG; TANG; ZHANG, 2004), (GOLUB et al., 1999) e classificação de documentos da Web (BOLEY et al., 1999).

Nesta tese de doutorado, discutiremos sobre agrupamentos de dados tanto pontuais quanto simbólicos (intervalares). Para o agrupamento de dados pontuais, propõe-se uma nova forma de calcular o centro dos clusters. Essa nova forma permite que algumas variantes de algoritmos do tipo Fuzzy C-Means (FCM) possam se beneficiar reduzindo a quantidade de iterações até o sistema convergir, diminuindo o tempo de processamento computacional e até mesmo obtendo uma classificação melhor do que se fosse utilizado o algoritmo FCM. O trabalho de (WU; YANG, 2002) propõe uma variante do algoritmo FCM chamado AFCM (Alternative Fuzzy C-Means). A principal mudança nesse algoritmo esta em usar outras distâncias em vez da distância Euclidiana (mais usual) de tal forma que melhore o índice de acerto.

A ideia do nosso trabalho é utilizar a matriz do grau de pertinência, a fim de obter uma matriz *crisp* que possibilite calcular os novos centros usando uma estratégia semelhante à do algoritmo K-Means (MACQUEEN, 1967). Por este motivo, denominou-se o nome de ckMeans a este algoritmo. Para a manipular dados intervalares, criou-se um novo algoritmo baseado no algoritmo ckMeans. Essa nova proposta de algoritmo não realiza qualquer conversão dos dados intervalares para pontuais, conforme acontece em diversos algoritmos encontrados na literatura, denominou-se o nome de Interval ckMeans a este algoritmo.

1.1 Motivação

Devido à quantidade cada vez maior de dados armazenados pelas instituições, a área de mineração de dados tem ganhado grande relevância e vários métodos têm sido propostos para aumentar sua aplicabilidade e desempenho.

Existe uma grande quantidade de algoritmos de agrupamento de dados. Entretanto, muitos apresentam dificuldades em alguns aspectos. Por isso, essa é uma área que permite ser explorada, propondo uma nova ou alterando uma técnica de agrupamento já existente para que seja capaz de suprir e/ou amenizar as dificuldades encontradas na técnica original.

O algoritmo K-Means está entre os algoritmos de agrupamento mais conhecidos e utilizados porque a sua implementação é relativamente simples e eficaz. Porém, este é um algoritmo *crisp*, e portanto os cluster determinam uma partição no sentido matemático, ou seja todo dado pertence a extamente um cluster. Diante disso, uma solução encontrada na literatura é trabalhar

com agrupamentos fuzzy, onde um dado pode pertencer a mais de um grupo, mas com diferentes graus de pertinência (NASCIMENTO; MIRKIN; F., 2000). O algoritmo de agrupamento fuzzy mais popular é o FCM.

É comum ouvir a expressão “tempo é dinheiro”, atualmente otimizar e economizar tempo para realizar uma tarefa tornou-se imprescindível. Os algoritmos computacionais desenvolvidos nos dias de hoje devem ser projetados com a sua complexidade computacional mais baixa possível, pois assim o custo para obtenção das soluções desses sistemas será menor. Pensando nisso, aceitamos o desafio de melhorar o algoritmo de agrupamento de dados FCM.

Existem diversas propostas de variantes para o algoritmo FCM na literatura como (MARTINO; LOIA; SESSA, 2007), (DEMING; QIANG, 2006) e (LI; BECERRA; DENG, 2004) . Em (ZANG et al., 2009), por exemplo, é proposto uma nova métrica, que utiliza uma função exponencial para substituir a distância euclidiana no algoritmo FCM. No artigo proposto por (ESCHRICH et al., 2003) o objetivo principal é reduzir o tempo processamento e o número de iterações no algoritmo FCM, a redução é feita através da agregação de exemplos similares. No entanto, nenhum desses autores consideram uma nova forma de calcular os centros dos clusters.

Por outro lado, representar o conjunto de dados como intervalos consiste em delimitar os erros ocasionados por estimativas de medições, de simplificações, modelagem, por falha humana ou pelo instrumento de medição. Os trabalhos (de CARVALHO, 2007), (ZHANG; HU; LIU, 2007), (BOCK, 2000) e (SATO; LAKHMI, 2006) lidam com agrupamento de dados simbólicos. Porém, com uma perspectiva pontual no sentido que os graus de pertinências e as métricas sejam pontuais. Este trabalho apresenta uma forma de agrupamento para os valores amostrais que consideram os erros contidos. Então a entrada dos dados são valores intervalares. Desta forma, julga-se que a classificação de cada elemento a um determinado cluster (o grau de pertinência) também seja um intervalo.

1.2 Contexto Histórico

Esta seção apresenta os principais fatos que conduziram à utilização dos principais algoritmos de agrupamento de dados, surgimento da Matemática Intervalar e da Lógica Fuzzy.

1.2.1 Agrupamento de Dados

A utilização comercial de banco de dados começou nos anos 60. Inicialmente a informação era guardada em ficheiros e a sua consulta e manipulação era muito pouco prática. A medida que

passam os anos, as organizações acumulam mais e mais informações em suas bases de dados. Como consequência, estas bases de dados passam a conter verdadeiros tesouros de informação.

Toda essa informação pode tornar essas organizações mais competitivas no mundo de hoje, permitindo que elas detectem tendências e características escondidas, e que forneçam uma reação mais rápida a eventos futuros.

No início dos anos 70 surgiram os Softwares de Gerenciamento de Banco de Dados (SGBD) relacionais cuja popularidade não tem parado de crescer até hoje. Entretanto, poucas organizações conseguem tirar proveito do que está armazenada em seus arquivos. Na verdade, esta informação valiosa está disfarçada, implícita nesses grandes conjuntos de dados e não pode ser descoberta utilizando-se sistemas de gerenciamento de banco de dados tradicionais (PUNTAR, 2003).

Nos dias de hoje existe uma vasta quantidade de algoritmos de agrupamento de dados. Entre os mais utilizados/conhecidos estão o K-Means e FCM[1].

O termo "K-Means" foi primeiramente usado por James MacQueen em 1967 (MACQUEEN, 1967), embora a ideia remonta ao trabalho de Hugo Steinhaus, publicado em 1956 (STEINHAUS, 1957). O algoritmo padrão foi primeiramente proposto por Stuart Lloyd em 1957 como uma técnica de pulso modulação de código, mas não foi publicado até 1982 (LLOYD, 1982).

Segundo (ZOU; WANG; HU, 2008), o algoritmo de agrupamento de dados fuzzy foi proposto por (DUNN, 1973), e estendido por (BEZDEK, 1981). Suas vantagens em relação ao K-Means já foram verificadas em diversas aplicações, que podem serem vistas em (BEZDEK; EHRLICH; FULL, 1984), (BEZDEK et al., 2005) e (CANNON; DAVE; BEZDEK, 1986).

O algoritmo ckMeans (versão pontual e intervalar) proposto nesta tese, surgiu da ideia de transformar o algoritmo FCM em intervalar. Na implementação desse algoritmo, como não foi possível simplesmente converter as variáveis para intervalos, fez-se necessário "pensar intervalarmente", surgindo assim, uma nova forma de calcular os centros dos clusters. Dessa forma, como foi possível trabalhar com dados intervalares, desenvolveu-se a versão pontual do algoritmo ckMeans, a qual se mostrou muito rápida e com a mesma ou melhor qualidade das partições.

[1]Melhores detalhados nesta tese.

1.2.2 Matemática Intervalar

Norbert Wiener, nos anos de 1914 (WIENER, 1914) e 1920 (WIENER, 1920), teria sido uma das primeiras pessoas a utilizar intervalos para descrever resultados de medições de distâncias e tempo. Também Burkill e Young (BURKILL, 1924), (YOUNG, 1931), descrevem uma forma de aritmética intervalar. No entanto, os primeiros registos de trabalhos independentes devem-se a P. S. Dwyer , M. Warmus, T. Sunaga e R. E. Moore (DWYER, 1951), (WARMUS, 1956), (SUNAGA, 1958) e (MOORE, 1959). Entretanto, foram os artigos de Moore e em particular a sua tese datada de 1962 (MOORE, 1966), onde introduz a aritmética intervalar como uma ferramenta para se ter um controle automático e rigoroso dos erros em computações científicas, que foram os principais responsáveis pela consolidação e desenvolvimento de diversas aplicações bem sucedidas desta teoria.

Por outro lado, o nome "matemática intervalar" foi proposto por Leslie Fox em 1974 (FOX, 1974) para referenciar a área que agrupa diferentes tópicos relativos ao conjunto de intervalos reais como a aritmética intervalar, análise intervalar, topologia intervalar, álgebra intervalar e outras.

Inicialmente, algumas críticas foram feitas à técnica da matemática intervalar, a qual para muitos autores apresentam dois problemas:

- Fornece resultados com intervalos pessimistas, ou seja, demasiadamente grandes, e portanto não retornavam uma informação relevante;
- Necessitava de um grande tempo de processamento computacional.

Essas duas críticas foram mostradas infundadas (HÖLBIG, 2005), uma vez que, com a utilização da aritmética de exatidão máxima (KULISCH; MIRANKER, 1981), é possível garantir que os resultados finais sejam produzidos com a máxima exatidão possível na máquina. Quanto ao tempo de processamento, este pode ser consideravelmente reduzido dependendo da forma como o algoritmo é implementado, existindo várias possibilidades tanto via software quanto via hardware (RUMP, 1999).

1.2.3 Lógica Fuzzy

Na antiga Grécia, Aristóteles introduziu as Leis do Conhecimento, que posteriormente seriam o sustento para a Lógica Clássica. Suas três leis fundamentais eram:

- Princípio de Identidade;

- Lei da Contradição;
- Lei do Terceiro Excluído.

A Lei do Terceiro Excluído afirma que para toda proposição p, p ou $\sim p$, deve ser verdadeira. Exemplo: Ou este homem é Sócrates ou não é Sócrates. Uma proposição só pode ser verdadeira se não for falsa e só pode ser falsa se não for verdadeira, porque só existem essas duas possibilidades. Entre ser ou não ser, não pode existir uma terceira alternativa, algo intermediário, que não seria o ser, nem seria o não ser.

De acordo com (ALTROCK, 1997) e (RUSS; SIMPSON; DOBBINS, 1996) a Lógica Fuzzy, Lógica Nebulosa ou ainda Lógica Difusa foi introduzida em 1965, com uma publicação de Lotfi A. Zadeh (ZADEH, 1965), que propicia uma nova teoria de Conjuntos. Professor da Universidade da Califórnia, Berkeley, considerado um grande colaborador do controle moderno, Zadeh criou uma teoria de conjuntos em que é permitido que a função de pertinência seja contínua, ou seja, não haja uma distinção abrupta entre elementos pertencentes e não pertencentes a um conjunto, esses conjuntos chamam-se conjuntos nebulosos.

A Lógica Fuzzy é baseada na teoria dos conjuntos fuzzy. Ela é uma generalização da Lógica Clássica que permite resolver os paradoxos gerados a partir da classificação “verdadeiro ou falso” da Lógica Clássica, como por exemplo algum paradoxo do mentiroso. Tradicionalmente, uma proposição lógica tem dois valores: ou “completamente verdadeiro” ou “completamente falso”. Entretanto, na Lógica Fuzzy, uma premissa varia em grau de verdade de 0 a 1, o que leva a ser parcialmente verdadeira e parcialmente falsa ao mesmo tempo.

As primeiras aplicações industriais da Lógica Fuzzy na Europa, ocorreram após 1970. Em Londres, Ebrahim Mamdani a usou para controlar um gerador a vapor. Na Alemanha, Hans Zimmermann usou Lógica Fuzzy para sistemas de apoio à decisão. Após 1980, na Europa, a Lógica Fuzzy ganhou mais impulso pela suas aplicações no apoio à tomada de decisões e na análise de dados (EKLUND, 1994). Inspirado pelas primeiras aplicações de Lógica Fuzzy europeias, companhias japonesas começaram a usar Lógica Fuzzy em engenharia de controle. As primeiras aplicações de Lógica Fuzzy foram em uma planta de tratamento de água da Fuji Electric em 1983 e um sistema de metrô da Hitachi em 1987. Hoje em dia, os carros japoneses, máquinas fotográficas e filmadoras utilizam o conceito de Lógica Fuzzy (VARGAS, 2008). Na comunidade científica, em particular na tarefa de agrupamento de dados, a Lógica Fuzzy têm uma grande aceitação.

1.3 Objetivos

O objetivo geral desta tese de doutorado é melhorar a qualidade dos agrupamentos, tanto a rapidez quanto o índice de acertos, de algoritmos de agrupamento de dados tipo FCM. Outro objetivo, é desenvolver um algoritmo de agrupamento de dados capaz de manipular dados simbólicos.

Por outro lado, os objetivos específicos são:

- Modificar a forma de calcular o centro dos clusters no algoritmo FCM (ckMeans);
- Comparar o desempenho desta nova variante do FCM com K-Means;
- Aplicar outras distâncias nos algoritmos, K-Means, FCM e ckMeans, comparando seu desempenho;
- Estender o algoritmo ckMeans para dados intervalares.

1.4 Organização da Tese

Este texto se divide em 5 Capítulos. O Capítulo 2 apresenta o estudo do referencial teórico onde são abordados os seguintes aspectos: as operações com a matemática intervalar utilizadas na construção do algoritmo e um breve funcionamento da biblioteca C-XSC usada como recurso computacional, os algoritmos para o agrupamento de dados pontuais (K-Means e FCM); No Capítulo 3, são apresentadas as metodologias propostas para a resolução de agrupamento de dados como a nova forma de calcular os centros dos clusters assim como uma versão intervalar desse algoritmo que permita manipular dados simbólicos. No Capítulo 4, apresenta-se a comparação dos algoritmos já implementados com as novas propostas. No Capítulo 5, a conclusão, os trabalhos futuros e as principais publicações são mostrados.

2 *Preliminares*

Este capítulo apresenta o estudo do problema a ser abordado, isto é, o agrupamento de dados, além de duas teorias para lidar com incertezas e imprecisão: a matemática intervalar e a Lógica Fuzzy e uma seção dedicada as distâncias ou métricas, tanto pontual quanto intervalar. Essas duas técnicas foram empregadas para o agrupamento de dados. Para a aplicação de dados simbólicos, utilizou-se uma biblioteca que possibilita o uso de intervalos, a biblioteca C-XSC (HOFSCHUSTER; KRÄMER, 2003).

2.1 Matemática Intervalar

A matemática intervalar considera um conjunto de métodos para manipulação de intervalos numéricos que aproximam dados incertos. Na computação científica, os intervalos podem ser aplicados para representar valores desconhecidos e, também para representar valores contínuos. Servem para controlar o erro de arredondamento e para representar dados inexatos, aproximações e erros de truncamento de procedimentos (OLIVEIRA; DIVERIO; CLAUDIO, 2001). Estes métodos, baseiam-se na definição da aritmética intervalar e do produto escalar ótimo (KULISCH, 2002).

Destacam-se a seguir, as principais operações e funções da matemática intervalar utilizadas para este trabalho:

2.1.1 Operações Básicas

Sejam, $X, Y \in \mathbb{IR}$ dois intervalos reais, com $X = [\underline{x}; \overline{x}]$ e $Y = [\underline{y}; \overline{y}]$. As operações aritméticas intervalares de adição, subtração, multiplicação e divisão são vistas na Tabela 2.1.

Tabela 2.1: Principais operações sobre intervalos.

Descrição	**Operações**
Adição	$X+Y=\left[\left(\underline{x}+\underline{y}\right);\left(\overline{x}+\overline{y}\right)\right]$
Subtração	$X-Y=\left[\left(\underline{x}-\overline{y}\right);\left(\overline{x}-\underline{y}\right)\right]$
Multiplicação	$X\times Y=[min\{\underline{x}\times\underline{y},\underline{x}\times\overline{y},\overline{x}\times\underline{y},\overline{x}\times\overline{y}\};\ max\{\underline{x}\times\underline{y},\underline{x}\times\overline{y},\overline{x}\times\underline{y},\overline{x}\times\overline{y}\}]$
Divisão	$\frac{X}{Y}=\left[\min\left\{\frac{\underline{x}}{\underline{y}},\frac{\underline{x}}{\overline{y}},\frac{\overline{x}}{\underline{y}},\frac{\overline{x}}{\overline{y}}\right\};\max\left\{\frac{\underline{x}}{\underline{y}},\frac{\underline{x}}{\overline{y}},\frac{\overline{x}}{\underline{y}},\frac{\overline{x}}{\overline{y}}\right\}\right]$ com $0\notin\left[\underline{y};\overline{y}\right]$

Os intervalos reais têm várias semânticas como, por exemplo, a representação de números reais; neste caso, quanto mais próximos os extremos do intervalo estiverem do valor “correto", melhor será a representação desse valor. A definição de ordem entre intervalos depende da abordagem ou semântica utilizada. Para este trabalho utilizou-se da ordem de Kulisch-Miranker (KULISCH; MIRANKER, 1981), conforme definição a seguir.

Definição 2.1 *Sejam dois intervalos* $X=[\underline{x};\overline{x}]$ *e* $Y=[\underline{y};\overline{y}]$*, a ordem de Kulisch-Miranker estabelece que* $[\underline{x};\overline{x}]\leq_K[\underline{y};\overline{y}]\Leftrightarrow\underline{x}\leq\underline{y}$ *e* $\overline{x}\leq\overline{y}$*.*

2.1.2 Funções Elementares

Algumas funções da matemática intervalar foram utilizadas nas implementações dos algoritmos, estas são descritas a seguir.

Ponto Médio de um Intervalo

Seja $X=[\underline{x};\overline{x}]\in\mathbb{IR}$ um intervalo. Define-se o ponto médio do intervalo X, conforme ilustrado na Figura 2.1, como sendo o número real $m=\frac{\underline{x}+\overline{x}}{2}$.

Notação:
$med(X)=med\left([\underline{x};\overline{x}]\right)=\frac{\underline{x}+\overline{x}}{2}$

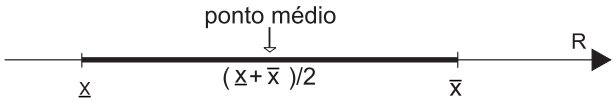


Figura 2.1: Representação geométrica do ponto médio de um intervalo X.

Diâmetro Intervalar

Seja $X = [\underline{x};\overline{x}] \in \mathbb{IR}$ um intervalo. Define-se diâmetro ou amplitude do intervalo X como sendo o número real não-negativo $d = \overline{x} - \underline{x}$.

Notação: $diam(X) = diam([\underline{x};\overline{x}]) = \overline{x} - \underline{x}$.

A Figura 2.2 exibe o diâmetro do intervalo $[-3;4]$ que corresponde a 7 unidades.

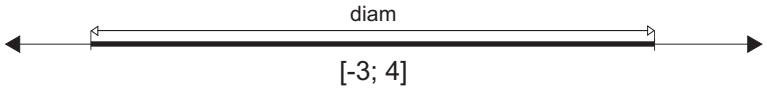


Figura 2.2: Representação geométrica do diâmetro de um intervalo.

Módulo Intervalar

Seja $X = [\underline{x};\overline{x}] \in \mathbb{IR}$ um intervalo conforme a Figura 2.3. Define-se o módulo do intervalo X como o número real não-negativo μ, que corresponde à maior distância de um $x \in X$ ao valor zero.

Notação: $|X| = |\,[\underline{x};\overline{x}]\,| = max\{|\underline{x}|, |\overline{x}|\}$.

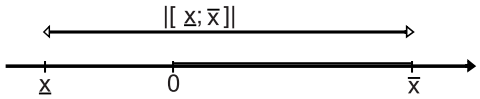


Figura 2.3: Módulo de um intervalo.

Função Exponencial

Define-se a função exponencial intervalar $Exp : \mathbb{IR} \to \mathbb{IR}$ pela Equação (2.1).

$$Exp([\underline{x};\overline{x}]) = [exp(\underline{x}), exp(\overline{x})] \tag{2.1}$$

Potência fracionária de um intervalo real

Dado $m, n \in \mathbb{N}$ com $n > 0$, define-se a potência de um intervalo $X = [\underline{x};\overline{x}]$ por $X^{\frac{m}{n}} = (\sqrt[n]{X})^m$, onde a raiz m de X e a potência n de um intervalo real X são calculadas conforme as equações (2.2) e (2.3), respetivamente.

$$\sqrt[n]{X} = \begin{cases} \left[\sqrt[n]{\underline{x}}; \sqrt[n]{\overline{x}}\right] & \text{se } \underline{x} \geq 0 \text{ ; e} \\ Indefinido & \text{, caso contrário.} \end{cases} \tag{2.2}$$

$$X^n = \begin{cases} [\overline{x}^n; \underline{x}^n] & \text{, se } \overline{x} < 0 \text{ e } n \text{ for par;} \\ [0; \max([\underline{x}^n; \overline{x}^n])] & \text{, se } \underline{x} < 0 < \overline{x} \text{ e } n \text{ for par; e} \\ [\underline{x}^n; \overline{x}^n] & \text{, caso contrário.} \end{cases} \tag{2.3}$$

2.1.3 C-XSC

De acordo com (HOFSCHUSTER; KRÄMER, 2003), o C-XSC é uma ferramenta para o desenvolvimento de algoritmos numéricos que permite uma alta precisão. A biblioteca C-XSC é muito aplicada na computação científica como uma biblioteca numérica, para o desenvolvimento de algoritmos numéricos, utilizando a linguagem de programação C++. Assim, o C-XSC permite a programação de alto nível de aplicações numéricas em C++ (HÖLBIG, 2005).

O C-XSC permite trabalhar com intervalos reais e também com intervalos complexos (interval e cinterval). Existem funções predefinidas para a manipulação de intervalos, nos quais destacam-se *Inf()*, *Sup()*, *diam()* e *mid()* (extremo inferior de um intervalo, extremo superior de um intervalo, diâmetro de um intervalo e o ponto médio de um intervalo, respectivamente).

No Código 1 é apresentado algumas operações aritméticas utilizando o tipo de dado intervalar (interval). Na Linha 13 deste código é realizada a soma dos intervalos *oper1* e *oper2*, na Linha 14 faz-se a multiplicação do intervalo *oper1* por 2, o ponto médio do intervalo *oper1* é realizado na Linha 15 e o extremo inferior de um intervalo é obtido na Linha 16.

Código 1 Codificação em C-XSC: Uso de Intervalos.

```
1  #include <iostream>
2  #include <interval.hpp>
3  using namespace cxsc;
4  using namespace std;
5
6  int main()
7  {
8    interval oper1, oper2;
9    real a = 1.0;
10   real b = 3.0;
11   oper1 = interval(a, b);
12   oper2 = interval(b, b);
13   cout << oper1 + oper2 << endl; // [  4.000000,  6.000000]
14   cout << oper1 * 2 << endl;     // [  2.000000,  6.000000]
15   cout << mid(oper1) << endl;    // 2.000000
16   cout << Inf(oper1) << endl;    // 1.000000
17   return 0;
18 }
```

Além das classes real e interval, no C-XSC existem as classes dinâmicas de maior precisão, reais longos (l_real) e intervalo longo (l_interval), exemplos podem ser vistos no Código 2.

Código 2 Codificação em C-XSC: Múltipla Precisão.

```
1  #include <iostream>
2  #include <interval.hpp>
3  using namespace cxsc;
4  using namespace std;
5
6  int main()
7  {
8    l_interval a, b;
9    a = 1.0;              //  a = [ 1.0, 1.0 ]
10   b = 3.0;              //  b = [ 3.0, 3.0 ]
11   cout << a/b << endl;
12 }
```

Como resultado da divisão de *a* por *b* ($a \div b$), obtido na Linha 11, tem-se o intervalo: $[0,33333333333333333333333; 0,33333333333333333333334]$.

No C-XSC também são implementados vetores e matrizes dinâmicos com toda a sua aritmética, além de operadores relacionais e funções *standard* de precisão múltipla. A Tabela 2.2 (HOFSCHUSTER; KRÄMER, 2003) mostra as operações aritméticas possíveis entre os diferentes tipos de dados.

Tabela 2.2: Operadores em C-XSC.

TIPOS DADOS	integer real complex	interval cinterval	rvector cvector	ivector civector	rmatrix cmatrix	imatrix cimatrix
integer real complex	$+,-,*,/,\|$	$+,-,*,/,\|$	$*$	$*$	$*$	$*$
interval cinterval	$+,-,*,/,\|$	$+,-,*,/,\|,$ &	$*$	$*$	$*$	$*$
rvector cvector	$*,/$	$*,/$	$*,-,*^1,/,\|$	$*,-,*^1,/$		
ivector civector	$*,/$	$*,/$	$*,-,*^1,/,\|$	$*,-,*^1,/,\|,$ &		
rmatrix cmatrix	$*,/$	$*,/$	$*^1$	$*^1$	$*,-,*^1,/,\|$	$*,-,*^1,/,\|$
rmatrix cmatrix	$*,/$	$*,/$	$*^1$	$*^1$	$*,-,*^1,/,\|$	$*,-,*^1,/,\|,$ &

Na Tabela 2.2 o operador $|$ é o polígono convexo, o operador $*^1$ indica o produto com alta precisão. É melhor detalhado o operador **&** o qual realiza a intersecção entre os intervalos. Por exemplo, no Código 3, a Linha 14 realiza a intersecção entre os intervalos $[3;5]$ e $[4;6]$, obtendo como resultado o intervalo $[4;5]$.

Código 3 Codificação em C-XSC: Operador Intersecção.

```
#include <iostream>
#include <interval.hpp>
using namespace cxsc;
using namespace std;

int main()
{
  interval A, B, interseccao;
  A = interval(3, 5);
  B = interval(4, 6);
  interseccao = A&B;
  cout << A << endl;           // [  3.000000,  5.000000]
  cout << B << endl;           // [  4.000000,  6.000000]
  cout << interseccao << endl; // [  4.000000,  5.000000]
  return 0;
}
```

O *C++ Toolbox for Verified Computing* (HAMMER; HOCKS; RATZ, 1995) é um conjunto de ferramentas para resolução de diversos problemas numéricos com a biblioteca C-XSC. Esse *toolbox* dispõe de funções para calcular raízes de equações, resolução de sistemas lineares, otimização de sistemas, entre outras aplicações da área científica.

Além das funções que são disponibilizadas ao instalar o C-XSC, que pode ser obtido no seguinte endereço: `www.xsc.de`, existem outras funções que podem ser agregadas ao C-XSC, disponível em: `http://www.math.uni-wuppertal.de/~xsc/xsc/cxsc_software.html`. O processo de instalação detalhado pode ser visto em (VARGAS, 2008).

2.2 Lógica Fuzzy

Mais de 40 anos já se passaram desde que Lotfi A. Zadeh introduziu a teoria dos conjuntos fuzzy. Durante este período, temos assistido a uma impressionante evolução teórica e conceitual, o surgimento de novas metodologias, algoritmos e uma variedade de aplicações. Tecnologias em áreas como armazenamento e recuperação da informação, pesquisas na web, processamento e compreensão de imagem, reconhecimento de padrões, bioinformática, comércio eletrônico, navegação autônoma e etc., beneficiaram-se consideravelmente a partir da evolução e consolidação desta teoria (PEDRYCZ; GOMIDE, 2007).

Esta seção traz uma breve introdução à Lógica Fuzzy, conforme mostrado em (VARGAS, 2008).

2.2.1 Definição de Conjuntos Fuzzy

Na teoria de Conjuntos Clássicos, um elemento pertence ou não pertence a um determinado conjunto. Atribuindo um grau pertinência, um valor compreendido entre o intervalo zero e um, a restrição de pertencer ou não pertencer a um conjunto é enfraquecida. Em teoria dos conjuntos (clássica) um conjunto admite uma forma alternativa de ser representado via uma função que mapeia um elemento do universo ao valor 1, se esse elemento fizer parte do conjunto e zero caso contrário. Formalmente, dado um conjunto A no universo U, a função $\chi_A : U \rightarrow \{0,1\}$ definida na Equação (2.4), é chamada de função característica de A.

$$\chi_A(x) = \begin{cases} 1, \text{ se } x \text{ é um elemento do conjunto } A, \text{ e} \\ 0, \text{ se } x \text{ não é um elemento do conjunto } A \end{cases} \tag{2.4}$$

Por exemplo, a seguinte proposição *"a água está quente"*, pode ser modelada através de conjuntos clássicos definindo parâmetros, representando a temperatura da água pela variável T, onde esta representa o conjunto de todos os valores para a temperatura da água. Este conjunto pode ser definido na Equação (2.5).

$$A \subseteq T, \text{ onde } A = \{x \in T : 50 \leq x\} \tag{2.5}$$

ou também pode ser representado por uma função característica $\chi_A : T \rightarrow \{0,1\}$, definida na Equação (2.6).

$$\chi_A(x) = \begin{cases} 1, \text{ se } 50 \leq x \\ 0, \text{ se } x < 50 \end{cases} \tag{2.6}$$

A Figura 2.4 representa o conceito *"quente"* de modo clássico, onde a temperatura acima ou igual a 50 graus indica que a água está quente; caso contrário não está.

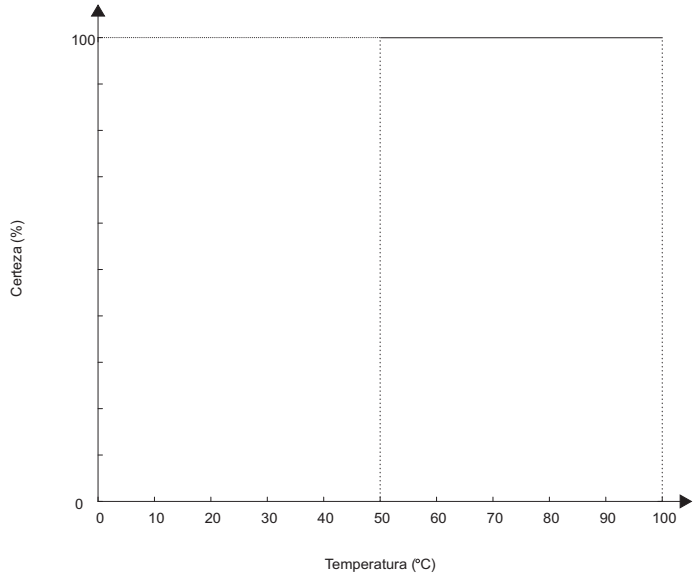


Figura 2.4: Conceito clássico de “quente”.

Esta modelagem tem muitas aplicações e utilidades práticas, contudo, em muitos casos, sofre perda e distorção da informação. Este exemplo mostra, que a temperatura de 49°C é considerada não quente enquanto que uma temperatura de 50°C seria considerada como quente e em função disso tomar decisões abruptas, como por exemplo desligar a fonte de energia que aquece a água. Enquanto que o ideal poderia ser que diminuíse a fonte de energia para evitar que a temperatura da água chegue a ficar quente.

Analisados esses valores, nota-se que é necessária haver existência de uma teoria que permita uma maior suavidade ao momento da passagem de um conjunto para outro da mesma natureza (variável linguística, no caso *Temperatura*), como por exemplo *frio*, *morno* e *quente* (termos linguísticos).

Na teoria dos conjuntos fuzzy um elemento pode pertencer parcialmente a um dado conjunto. O grau de pertinência é definido através de uma generalização da função característica chamada de *função de pertinência* e é definida pela Equação (2.7).

$$\mu_A(x) : U \rightarrow [0,1] \tag{2.7}$$

onde U chama-se conjunto universo e A é conjunto fuzzy.

Os valores da função de pertinência aplicada a elementos do universo de discurso são números reais no intervalo $[0;1]$, onde 0 significa que certamente o elemento não é membro do conjunto e 1 significa que o mesmo pertence absolutamente ao conjunto. Cada valor da função de pertinência é chamado de *grau de pertinência*.

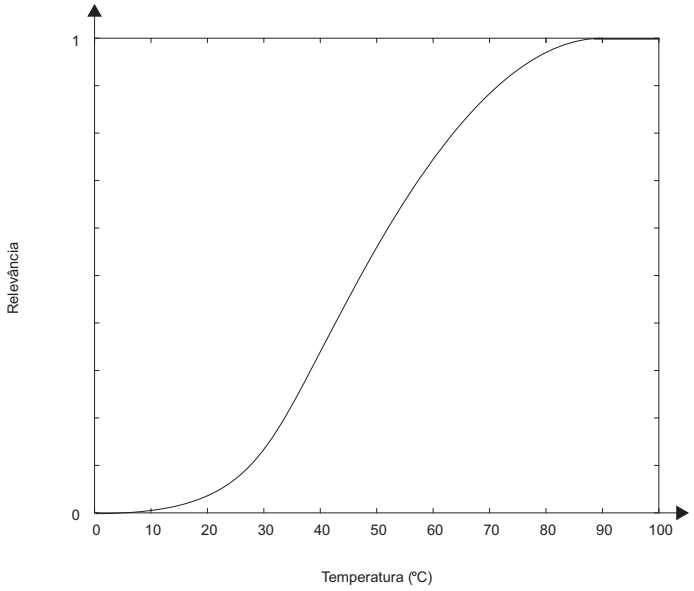


Figura 2.5: Conceito fuzzy de “quente”.

Na Figura 2.5 tem-se uma possível representação do conceito fuzzy *quente*. A função de pertinência μ_{QUENTE} associa o quanto a temperatura é considerada *quente*. Pode-se ter, por exemplo, as temperaturas 40, 50 e 60, com graus de pertinências $0,2$, $0,3$ e $0,6$ respectivamente.

2.2.2 Sobre as Funções de Pertinência

Um conceito fundamental na teoria dos conjuntos fuzzy é a noção de função de pertinência, como vista, é uma generalização da função característica de um conjunto clássico (GIARRATANO; RILEY, 1994). A função de pertinência de um conjunto fuzzy estabelece uma correspondência entre um elemento no domínio (universo de discurso) e um valor no intervalo $[0;1]$, que indica o grau de pertinência desse elemento no conjunto. Um conjunto fuzzy codifica a ambiguidade associada a um fenômeno através de sua superfície; na verdade, o desenho da curva representa a semântica do conceito em questão (GOTTGTROY, 1996).

Componentes de um Conjunto Fuzzy

Um conjunto fuzzy consiste basicamente de três componentes:

- Um eixo horizontal, de valores crescentes monotonicamente, que constituem o conjunto que representa o domínio;
- Um eixo vertical, entre 0 e 1, que indica o grau de pertinência ao conjunto;
- Uma função, que estabelece a relação entre os dois eixos.

A Figura 2.6, ilustra um exemplo de conjunto fuzzy que representa a ideia de *número pequeno*. Para esse exemplo, os componentes do conjunto fuzzy *número pequeno* são:

- No eixo horizontal: os números reais positivos, num intervalo considerado representativo do contexto do modelo em questão;
- No eixo vertical: o quanto cada número do eixo horizontal reflete *ser pequeno*;
- Uma função, que estabelece a relação entre os dois eixos.

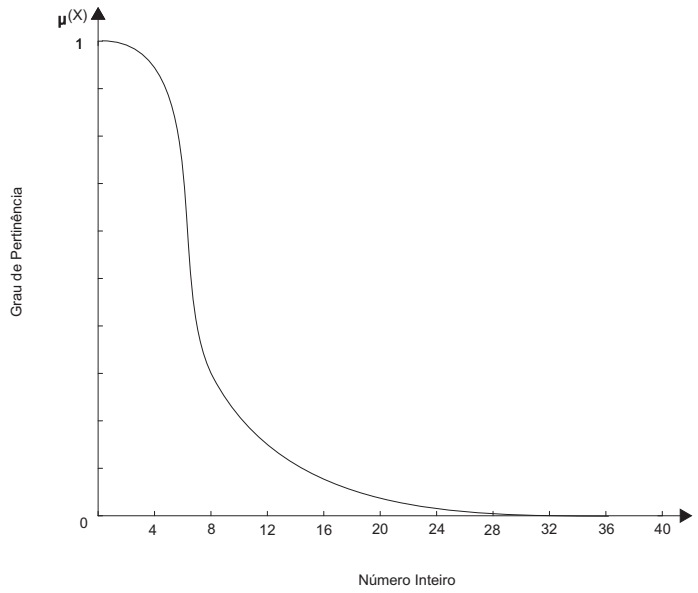


Figura 2.6: Conjunto fuzzy de “números pequenos”.

Em um outro exemplo, encontrado em (GOTTGTROY, 1996), pode-se especificar um conjunto Fuzzy através da explicação da função de pertinência de alguns elementos do domínio; assim, no universo de *altura*, define-se o universo $U = (1.60, 1.70, 1.75, 1.80)$ e pode-se representar o conjunto fuzzy *alto* na Equação (2.8) por:

$$alto = 0,01/1,60 + 0,3/1,70 + 0,65/1,75 + 1,0/1,80 \tag{2.8}$$

A função de pertinência para o conjunto fuzzy alto, no caso uma curva linear simples, representa que *ser alto* é diretamente proporcional a um valor, que representa a altura em metros.

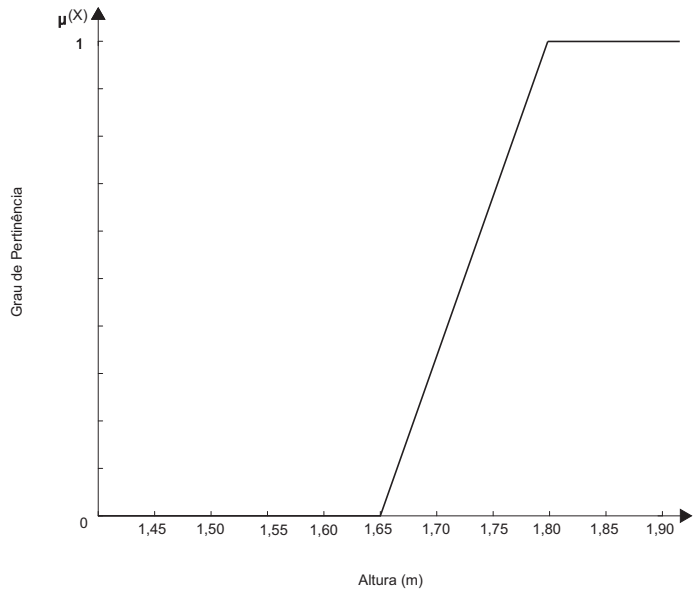


Figura 2.7: Conjunto fuzzy de “alto”.

O conjunto fuzzy “alto”, Figura 2.7, diz que, abaixo de $1,60$m, inclusive, definitivamente não significa ser alto; $1,75$m significa moderadamente alto, com grau de pertinência de $0,65$ e acima de $1,80$m, definitivamente significa ser alto. Observe que esse conjunto está representando o conceito *alto* para a realidade brasileira.

Cabe ao engenheiro do conhecimento, pela sua experiência, definir qual o “desenho” que melhor caracteriza o conceito em questão. Entretanto, os sistemas fuzzy têm demonstrado ser tolerantes a aproximações na representação dos conceitos (COX, 1994), isto é, eles demonstram bom desempenho mesmo quando o desenho do conjunto fuzzy não mapeia exatamente o conceito modelado.

2.3 Distância Euclidiana

Intuitamente, a distância é a medida da separação de dois pontos. A distância é sempre uma medida positiva e tem a propriedade de que a distância de um ponto X até um ponto Y é idêntica à distância do ponto Y até o ponto X.

Definição 2.2 *(RUDIN, 1976) Dado um conjunto $\mathbb{S}$, uma métrica em $\mathbb{S}$ é uma função*

$$d : \mathbb{S} \times \mathbb{S} \to \mathbb{R}^+$$

que possui as seguintes propriedades:

1. *É positivamente definida, ou seja, é tal que*

$$d(X,Y) \geq 0,$$

para todo $X, Y \in \mathbb{S}$*;*

2. *É simétrica, ou seja, é tal que*

$$d(X,Y) = d(Y,X),$$

para todos os elementos X, Y *de* $\mathbb{S}$*;*

3. *Obedece a desigualdade triangular; para todos os* X, Y, Z *elementos de* $\mathbb{S}$ *, d satisfaz*

$$d(X,Z) \leq d(X,Y) + d(Y,Z);$$

4. *É nula apenas para pontos coincidentes. Ou seja,*

$$d(X,Y) = 0 \Longleftrightarrow X = Y.$$

A forma mais usual para calcular a distância entre dois pontos, em particular para o caso do cálculo do centro dos clusters, é a distância Euclidiana. Formalmente, a distância Euclidiana é a função $d_E : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^+$ definida na Equação (2.9):

$$d_E(X,Y) = \sqrt{\sum_{i=1}^{n} |X_i - Y_i|^2}, \tag{2.9}$$

para todo $X = (X_1, \ldots, X_n)$ e $Y = (Y_1, \ldots, Y_n)$ em $\mathbb{R}^n$.

O código fonte em C++ usando C-XSC é mostrado no Código 4.

Código 4 Distância Euclidiana.

```
/* (Global Variable) */
int p = 2;
int u = 3;
int n = 3;
rmatrix X (u, n);
rmatrix Y (u, p);
rmatrix Distance (u, p);

/* Euclidean Distance */
void euclideanDistance()
{
  real ed;
  real acum;
  real temp;
  int i, j, f;
  for (i = 1 ; i <= n; i++)
  {
    for (j = 1; j <= p; j++)
    {
      acum = 0; temp = 0;
      for (f = 1; f <= u; f++)
      {
        Distance[j][i] += acum + pow(abs(X[f][i] - Y[f][j]), 2);
      }
    }
  }
  for (i = 1; i <= n; i++)
  {
    for (j = 1; j <= p; j++)
    {
      Distance[j][i] = sqrt(Distance[j][i]);
    }
  }
}
```

No caso dos intervalos reais, Ramon E. Moore em (MOORE, 1966) considerou a seguinte métrica: $d_M : \mathbb{IR} \times \mathbb{IR} \to \mathbb{R}^+$ definida na Equação (2.10):

$$d_M(X,Y) = d_M\left([\underline{x};\overline{x}], [\underline{y};\overline{y}]\right) = max\left\{|\underline{x}-\underline{y}|, |\overline{x}-\overline{y}|\right\} \tag{2.10}$$

Pode-se estender esta distância de Moore para $\mathbb{IR}^n$ de diversas maneiras. Por exemplo, $d_{EM} : \mathbb{IR}^n \times \mathbb{IR}^n \to \mathbb{R}^+$ definida para cada $\mathbf{X} = (X_1, \ldots, X_n)$ e $\mathbf{Y} = (Y_1, \ldots, Y_n)$ em $\mathbb{IR}^n$ é expressa por

$$d_{EM}(\mathbf{X}, \mathbf{Y}) = \sqrt{\sum_{i=1}^{n} d_M(X_i, Y_i)^2}$$

ou ainda, peloa expressão:

$$d'_{EM}(\mathbf{X}, \mathbf{Y}) = \max\{d_E(\underline{\mathbf{X}}, \underline{\mathbf{Y}}), d_E(\overline{\mathbf{X}}, \overline{\mathbf{Y}})\}$$

onde $\underline{\mathbf{X}} = (\underline{X_1}, \ldots, \underline{X_n})$, e analogamente para $\underline{\mathbf{Y}}$, $\overline{\mathbf{X}}$ e $\overline{\mathbf{Y}}$.

2.4 Distância Intervalar

A distância entre dois intervalos é um número real, o que não é natural quando lidamos com intervalos. Sejam $X = [\underline{x};\overline{x}]$ e $Y = [\underline{y};\overline{y}]$ dois intervalos conforme Figura 2.8.

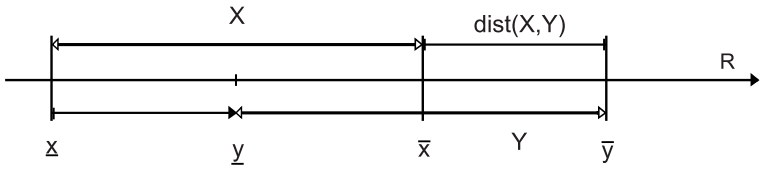


Figura 2.8: Representação geométrica da distância entre dois intervalos.

A diferença na pontuação entre o time *A* para o time *B*, são 3 pontos, supondo que eles têm 27 e 30 pontos, respectivamente. Pode-se prever essa distância na tabela de pontuação nas próximas duas rodadas. Há como imaginar essa diferença dentre as possibilidades possíveis, porém não há como ter certeza dos resultados de cada jogo. Ao analisar o time *A*, considerando sua pontuação mínima, não havendo vitórias e a sua pontuação máxima, com duas vitórias, poderíamos representar a previsão da pontuação a ser alcançada na forma de intervalos, sendo $A = [27;33]$ e o outro time, $B = [30;36]$. Com esta representação podemos prever a diferença de pontuação mínima e máxima entre os times após duas rodadas, neste exemplo, o intervalo $[0;9]$. Um outro exemplo seria a pesquisa de intenção de votos, desconsiderando os votos nulo, branco e indecisos para presidente do Brasil, realizada pelo Ibope entre os dias 2 a 5 de agosto de 2010, em um eventual segundo turno entre Dilma Rousseff (PT) e José Serra (PSDB). O Ibope apurou que a petista teria 44% das intenções de votos (considerando a margem de erro de dois pontos percentuais para mais ou para menos, ou seja, a sua votação estaria no intervalo ($[42;46]$%) e Serra, 39% ($[37;41]$%). Portanto, a menor diferença possível entre os candidatos ocorreria se Dilma obtivesse 42% e Serra 41% dos votos, já a maior diferença possível aconteceria caso Dilma obtenha 46% e Serra 37%, ou seja a diferença entre os candidatos estaria no intervalo $[1;9]$.

Estes exemplos, mostram que o mais natural quando lidamos com dados intervalares é que a distância entre dois intervalos também seja um intervalo. Para isto Trindade (TRINDADE, 2009) proporcionou uma definição baseada na Definição 2.2 mas considerando valores intervalares como medidas de distancias entre elementos de conjuntos.

Definição 2.3 *(TRINDADE, 2009) Dado um conjunto $\mathbb{S}$, uma métrica intervalar em $\mathbb{S}$ é uma função*

$$d : \mathbb{S} \times \mathbb{S} \to \mathbb{IR}^{+}$$

que possui as seguintes propriedades para todo X, Y e Z em $\mathbb{S}$:

1. $0 \in d(X,X)$;
2. $|d(X,Y)| \leq |d(X,Z)| + |d(Z,Y)|$
3. $d(X,Y) = d(Y,X)$ *e*
4. *se* $0 \in d(X,Y) = d(X,X) = d(Y,Y)$ *então* $X = Y$

Caso o resultado sempre seja um intervalo degenerado, esta noção de métrica intervalar coincidiria com a noção usual de métrica.

No caso de $\mathbb{S}$ ser $\mathbb{IR}$, Trindade propõe a seguinte métrica intervalar:

$$m_{ei}(X,Y) = [\inf\{|x-y| : x \in X \text{ e } y \in Y\}, \sup\{|x-y| : x \in X \text{ e } y \in Y\}]$$

Uma caracterização desta metrica intervalar em termos dos extremos não é simples. De fato, em (TRINDADE, 2009) foram considerados só para alguns casos específicos de X e Y, o qual dificulta sua manipulação.

Porém, na proposta deste trabalho, esta métrica intervalar não se mostrou apropriada. Isso levou a introduzir uma nova métrica intervalar. A nova métrica intervalar proposta satisfaz propriedades mais similares às métricas (Definição 2.2).

Definição 2.4 (Distância Intervalar) *Sejam X e $Y \in \mathbb{IR}$. A distância intervalar entre X e Y, denotada por $d_I(X,Y)$, é o intervalo obtido pela Equação (2.11):*

$$d_I(X;Y) = \left[\min\left\{d(\underline{x};\underline{y}); d(\overline{x};\overline{y})\right\}; \max\left\{d(\underline{x};\underline{y}); d(\overline{x};\overline{y})\right\}\right] \tag{2.11}$$

onde $d(x;y)$ é a distância absoluta $(|x-y|)$ entre dois números reais.

Teorema 2.1 *d_I satisfaz as seguintes propriedades para todo X,Y e Z em $\mathbb{IR}$:*

1. $d_I(X,Y) = [0;0]$ *se e somente se* $X = Y$*;*
2. $d_I(X,Y) = d_I(Y,X)$*;*
3. $|d_I(X,Y)| \leq |d_I(X,Z)| + |d_I(Z,Y)|$

Demonstração As duas primeiras propriedades são triviais.

Sejam $X,Y,Z \in \mathbb{IR}$. Uma vez que $d_I(X,Y) = \left[\min\left\{|\underline{X}-\underline{Y}|, |\overline{X}-\overline{Y}|\right\}; \max\left\{|\underline{X}-\underline{Y}|, |\overline{X}-\overline{Y}|\right\}\right]$ então $|d_I(X,Y)| = \max\left\{|\underline{X}-\underline{Y}|, |\overline{X}-\overline{Y}|\right\}$ e analogamente para $|d_I(X,Z)|$ e $|d_I(Z,Y)|$. Portanto,

$$\begin{aligned}|d_I(X,Y)| &= \max\left\{|\underline{X}-\underline{Y}|,|\overline{X}-\overline{Y}|\right\}\\ &\leq \max\left\{|\underline{X}-\underline{Z}|+|\underline{Z}-\underline{Y}|,|\overline{X}-\overline{Z}|+|\overline{Z}-\overline{Y}|\right\}\\ &\leq \max\left\{|\underline{X}-\underline{Z}|,|\overline{X}-\overline{Z}|\right\}+\max\left\{|\underline{Z}-\underline{Y}|,|\overline{Z}-\overline{Y}|\right\}\\ &= |d_I(X,Z)|+|d_I(Z,Y)|\end{aligned}$$

Corolário 2.1 *A d_I é uma métrica intervalar.*

E analogamente ao caso de m_{ei}, também podemos estender esta distancia intervalar para $\mathbb{IR}^n$ de diversas maneiras. Por exemplo, $d_{IE} : \mathbb{IR}^n \times \mathbb{IR}^n \to \mathbb{R}^+$ definida para cada $\mathbf{X} = (X_1,\ldots,X_n)$ e $\mathbf{Y} = (Y_1,\ldots,Y_n)$ em $\mathbb{IR}^n$ por

$$d_{IE}(\mathbf{X},\mathbf{Y}) = \sqrt{\sum_{i=1}^{n} d_I(X_i,Y_i)^2} \tag{2.12}$$

2.5 Agrupamento de Dados

De acordo com (FAYYAD, 1996), a técnica de agrupamento (ou *clustering*) procura identificar um conjunto de categorias ou classes para descrever os dados. Segundo (HAN et al., 1996) e (AGRAWAL, 1996), no agrupamento, parte-se de uma situação em que não existem classes, somente elementos de um universo. A partir destes elementos, as técnicas de agrupamento são responsáveis por definir as classes e enquadrar os elementos.

Recentemente, a Análise de Dados Simbólicos (SDA, do inglês, (Simbolyc Data Analysis)) (BOCK, 2000) e (H. KIERS et al, 2000) foi proposta como um método de análise. Isso tem gerado um grande interesse por parte dos pesquisadores. Análise de dados convencionais, geralmente faz uso de dados quantitativos ou categóricos. Na Análise de Dados Simbólicos, quantitativo ou categóricos, dados intervalares, ou onde um conjunto de valores com pesos associados (dados modais) também podem ser usados. Focou-se nos dados com valores intervalares.

Por exemplo, se observarmos a situação de quanto tempo as pessoas assistem TV por dias, e propomos fazer o seguinte questionamento: “Quanto tempo você assiste TV por dia?”. Se o entrevistado responder 4,5 horas, essa observação tem um risco de não ser uma situação realista. Uma resposta entre 3 e 5 horas pode ser mais realista. Assim, uma observação deve ser tratada como intervalo de valor de dados. A motivação de trabalhar com dados simbólicos vêm da publicação de (SATO; LAKHMI, 2006).

A Figura 2.9 mostra os procedimentos de análise de clusters com quatro passos básicos. A típica análise de cluster consiste em quatro passos com retro alimentação. Esses passos estão intimamente relacionados e afetam os clusters resultantes.

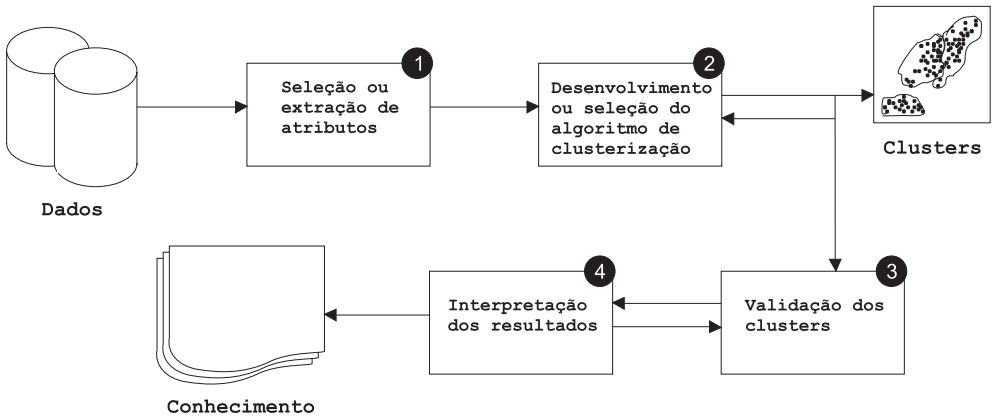


Figura 2.9: Procedimento de agrupamento.

1. *Seleção ou extração de atributos:* Na seleção de atributos são escolhidos aqueles mais representativos de um conjunto de atributos candidatos, enquanto na extração de atributos, utiliza-se transformações para gerar atributos úteis e novos a partir dos originais;

2. *Desenvolvimento ou seleção do algoritmo de agrupamento:* Este passo é usualmente combinado com a seleção de uma medida de proximidade e a definição de uma função de critério. Padrões são agrupados de acordo com a semelhança mútua. Uma vez que a medida de proximidade for escolhida, a construção de uma função de critério de agrupamento torna a partição dos padrões em clusters um problema de otimização, o qual é bem definido matematicamente;

3. *Validação dos clusters:* Dado um conjunto de dados, todo algoritmo de agrupamento pode gerar uma partição, não importando se existe uma estrutura ou não que justifique o conjunto. É preciso haver padrões e critérios de avaliações eficientes para fornecer ao usuário confiabilidade em relação ao resultado obtido a partir do algoritmo utilizado;

4. *Interpretação dos resultados:* O objetivo final da análise de cluster é prover aos usuários impressões significativas a partir dos dados originais de forma que estes possam resolver efetivamente os problemas encontrados.

2.5.1 Validação de Agrupamento

Aplicações que envolvem análise de agrupamentos são muito utilizadas em distintos campos da ciência e dos negócios como: Bioinformática, análise de dados espaciais, Mineração de

Dados e na Web, redução de dados, geração e prova de hipóteses, predição baseada em agrupamento, para posterior classificação. O principal desafio em tarefas de agrupamento é a análise e definição do número ótimo de grupos. Uma forma alternativa de análise de agrupamentos é dar uma nota à qualidade do resultado da agrupação levando em conta distâncias intra e inter grupos em busca de grupos compactos e bem separados (VILLANUEVA, 2008).

Através de um índice ou critério de validade é possível validar de maneira quantitativa um agrupamento. Tais índices podem ser de três tipos (HALKIDI; BATISTAKIS; VAZIRGIANNIS, 2001) e (JAIN; DUBES, 1988):

- **Externos**: Avalia o grau de correspondência entre a estrutura de grupos (partição ou hierarquia) sob avaliação e informação a priori na forma de uma solução de agrupamento esperada ou conhecida;
- **Internos**: Avalia o grau de compatibilidade entre a estrutura de grupos sob avaliação e os dados, usando apenas os próprios dados;
- **Relativos**: Avaliam qual dentre duas ou mais estruturas de grupos é melhor sob algum aspecto. Tipicamente são critérios internos capazes de quantificar a qualidade de agrupamentos.

Embora o problema de agrupamento seja não supervisionado, em alguns cenários o resultado do agrupamento desejado pode ser conhecido. Por exemplo:

- Reconhecimento visual dos agrupamentos naturais (bases 2D, 3D);
- Especialista do domínio;
- Bases geradas sinteticamente com distribuições conhecidas, é o caso dos resultados mostrados na Seção 4.2.
- Bases de classificação sob a hipótese que classes são clusters.

Índices que medem o nível de compatibilidade entre uma partição obtida e uma partição de referência dos mesmos dados são denominados critérios de validade externos.

Existem vários critérios externos na literatura, entre eles estão:

- Rand Index (RAND, 1971);

- Jaccard Coefficient (JACCARD, 1901);
- Rand Index Ajustado (HUBERT; ARABIE, 1985);
- Fowlkes-Mallows (FOWLKES; MALLOWS, 1983).

Para esta tese escolhemos utilizar e detalhar o índice *Adjusted Rand Index* por ser simples e intuitivo.

O *Adjusted Rand Index* pode ser visto como um critério absoluto (externo) ou como um padrão referencial que permite o uso de um conjunto de dados de classificação para realizar avaliação não somente de classificadores (que podem produzir diferentes partições de dados com o número correto de classes), mas também de resultados de agrupamentos de dados (em que diferentes partições de dados podem ser compostas de diferentes números de grupos). Este índice avalia duas partições rígidas (*hard* ou *crisp*) R e G do mesmo conjunto de dados. A partição de referência, R, codifica o rótulo das classes, ou seja, ela particiona o conjunto de dados em k^* classes conhecidas. A partição G, por sua vez, seleciona o conjunto de dados em k categorias (classes ou grupos) e é aquela a ser avaliada. As categorias codificadas por G serão, daqui em diante, chamadas de grupos, pelo contexto de algoritmos de agrupamento de dados. Desta forma, é possível distinguir entre estas e as classes corretas codificadas por R (ALVES, 2007). Então, após estas observações, o *Adjusted Rand Index* é definido como:

$$\Omega(R,G) = \frac{a+d}{a+b+c+d} \tag{2.13}$$

Onde:

- a: Número de pares de objetos do conjunto que pertencem à mesma classe em R e ao mesmo grupo em G;
- b: Número de pares de objetos que pertencem à mesma classe em R e a diferentes grupos em G;
- c: Número de pares de objetos que pertencem a classes diferentes em R e ao mesmo grupo em G;
- d: Número de pares de objetos que pertencem a classes distintas em R e a grupos distintos em G.

Os termos a e d são medidas do nível de consistência da classificação, enquanto os termos b e c são medidas de classificações inconsistentes. Note que: (i) $\Omega \in [0,1]$; (ii) $\Omega = 0$ se e somente

se G é completamente inconsistente, ou seja, $a = d = 0$; e (iii) $\Omega = 1$ se e somente se a partição sob avaliação corresponde exatamente à partição de referência, ou seja, $b = c = 0(G = R)$.

Um problema com o Rand Index é que o valor esperado do Rand Index entre duas partições aleatoria não tem um valor constante (digamos zero). O *Adjusted Rand Index* proposto por (HUBERT; ARABIE, 1985) assume a distribuição generalizada hipergeométrica como o modelo de aleatoriedade, isto é,

As partições A e B são escolhidas aleatoriamente, o número de objetos em classes e grupos são fixos. Seja n_{ij} o número de objetos que estão em ambas as classes A_i e cluster B_j. Seja n_i e $n_{.j}$ o número de objetos na classe A_i e a classe B_j, respectivamente. As notações são ilustradas na Tabela 2.3.

Tabela 2.3: Tabela contingência. n_{ij} denota o número de elementos que são comuns aos conjuntos A_i e B_j.

		Partição B				
Partição A	Class	B_1	B_2	$\ldots$	B_c	Somatório
	A_1	n_{11}	n_{12}	$\ldots$	n_{1C}	$n_{1\bullet}$
	A_2	n_{21}	n_{22}	$\ldots$	n_{2C}	$n_{2\bullet}$
	$\vdots$	$\vdots$	$\ldots$	$\ddots$	$\vdots$	$\vdots$
	A_C	n_{R1}	n_{R2}	$\ldots$	n_{RC}	$n_{R\bullet}$
	Somatório	$n_{\bullet 1}$	$n_{\bullet}$	$\ldots$	$n_{\bullet C}$	N

O valor esperado não é nulo para duas partições completamente aleatórias de um conjunto de dados, que é delimitado por um, e toma o valor 0 quando o índice é igual a seu valor esperado. Sob o modelo generalizado hipergeométrica, pode ser mostrado em (HUBERT; ARABIE, 1985) que:

$$E\left[\sum_{i,j}\binom{n_{i,j}}{2}\right] = \left[\sum_{i}\binom{n_{i.}}{2}\sum_{j}\binom{n_{.j}}{2}\right]/\binom{n}{2}$$

A expressão $a+d$ pode ser simplificada por uma transformação linear de $\sum_{ij}\binom{n_{ij}}{2}$. Com uma álgebra simples, o *Adjusted Rand Index* pode ser simplificado por:

$$\frac{\sum_{i,j}\binom{n_{i,j}}{2} - \left[\sum_{i}\binom{n_{i.}}{2}\sum_{j}\binom{n_{.j}}{2}\right]/\binom{n}{2}}{\frac{1}{2}\left[\sum_{i}\binom{n_{i.}}{2} + \sum_{j}\binom{n_{.j}}{2}\right] - \left[\sum_{i}\binom{n_{i.}}{2}\sum_{j}\binom{n_{.j}}{2}\right]/\binom{n}{2}}$$

Um exemplo (YEUNG; RUZZO, 2001) da tabela de contingência para ilustrar o *Adjusted Rand Index* é mostrado na Tabela 2.4.

Tabela 2.4: Exemplo de uma Tabela contingência.

Partição A \ Partição B	B_1	B_2	B_c	Somatório
A_1	1	1	0	2
A_2	1	2	1	4
A_3	0	0	4	4
Somatório	2	3	5	$n = 10$

Ainda observando a Observando a Tabela 2.4, temos:

- a é o número de pares que pertencem à mesma classe e ao mesmo cluster, por isso a pode ser escrito como $\sum_{ij}\binom{n_{ij}}{2}$, onde i e j variam de 1 até 3. Como $\binom{1}{2}$ + $\binom{0}{2}$ são zero, logo temos no exemplo $a = \binom{2}{2} + \binom{4}{2} = 7$.
- b é o numero de pares que pertencem à mesma classe e a clusters distintos, b pode ser escrito como $\sum_i \binom{n_i}{2} - \sum_{ij}\binom{n_{ij}}{2}$. No exemplo, $b = \binom{2}{2} + \binom{4}{2} + \binom{4}{2} - 7 = 6$.
- c é o numero de pares que pertencem a classes distintas e ao mesmo cluster, então c pode ser definido como $\sum_j \binom{n_j}{2} - \sum_{ij}\binom{n_{ij}}{2}$. No exemplo, $c = \binom{2}{2} + \binom{3}{2} + \binom{5}{2} - 7 = 7$.
- d é o numero de pares que pertencem a classes e clusters distintos, exemplificado como $d = \binom{10}{2} - a - b - c = \binom{10}{2} - 7 - 6 - 7 = 25$.

O *Adjusted Rand Index* do exemplo mostrado na Tabela 2.4 é $\frac{7-14*13/45}{(14+13/2-14*13/45} = 0,313$.

2.5.2 Algoritmo K-Means

O algoritmo K-Means (MACQUEEN, 1967) é um dos mais simples algoritmos de aprendizado não-supervisionado que resolve o problema de agrupamento de dados. O procedimento segue uma maneira relativamente simples e fácil de classificar um determinado conjunto de dados através de um certo número de clusters (assumindo k-clusters), fixado a priori.

O fluxograma do algoritmo é mostrado na Figura 2.10.

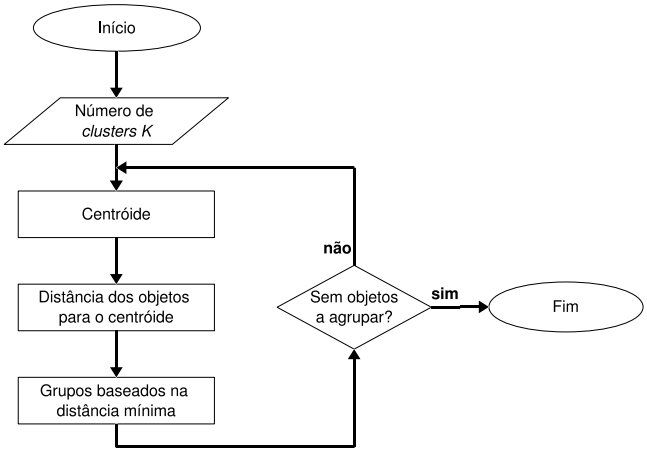


Figura 2.10: Fluxograma do algoritmo K-Means.

O algoritmo funciona basicamente enquanto houver elementos na base sem pertencer a nenhum cluster, faz-se a comparação entre os elementos (nova informação comparada aos centroides dos clusters existentes) para poder incluir os novos.

O algoritmo segue basicamente os seguintes passos:
Seja um k dado a priori que indica a quantidade de cluster nos quais se quer classificar os elementos da base.

Passo 1. Particionar os objetos, de forma aleatória, em k grupos não vazios, sendo que cada partição será o centroide desses grupos;

Passo 2. Assinalar cada objeto restante como pertencente ao cluster ao qual apresenta-se mais próximo do seu centroide de acordo com a função distância;

Passo 3. Atualizar os centroides de cada cluster. As novas posições dos centroides será a media dos elementos que pertencem ao cluster conforme a Equação (2.14):

$$c_j = \frac{\sum_{i=1}^{n^{(j)}} x_i^{(j)}}{n^{(j)}} \tag{2.14}$$

onde $x_i^{(j)}$ é um elemento associado ao cluster j e $n^{(j)}$ é o número de elementos associados ao cluster j;

Passo 4. Se elementos mudarem de cluster no Passo 2, repetir 2-3. Se não, o algoritmo para.

Se o número de dados é menor ou igual ao número de clusters então atribuir cada dado como o centroide de um cluster. Cada centroide terá associado um dos k cluster. Se o número

de dados é maior do que o número de cluster, para cada um dos dados, calcular a distância a todos os centroides até obter a distância mínima. Cada dado pertencerá ao cluster para o qual possui a menor distância ao seu centroide.

Uma vez que não tenha certeza sobre a localização do centroide, precisa-se ajustar a localização do centroide com base nos dados atuais. Então, atribui-se todos os dados para este novo centroide. Este processo é repetido até que não haja mais dados que se deslocam para outros clusters. A convergência irá sempre ocorrer quando a soma das distâncias de cada amostra de treinamento para o centroide é reduzida.

Finalmente, este algoritmo visa minimizar uma função objetivo, neste caso uma função de erro quadrado. A função objetivo é mostrada na Equação (2.15).

$$J = \sum_{i=1}^{k} \sum_{j=1}^{n} d(X_i, C_j)^2, \tag{2.15}$$

onde:

- n é o número de instâncias;
- k é o número de clusters considerados no algoritmo, o qual deve ser decidido antes da execução;
- X_i é o i-ésimo dado;
- C_j é o centroíde do j-ésimo cluster; e
- $d(X_i, C_j)$ é a distância entre X_i e C_j, usualmente a distância é a Euclidiana (d_E).

O algoritmo K-Means é um algoritmo simples que foi adaptado para vários domínios de problemas. Como será visto na próxima subseção, ele é um bom candidato para a extensão proposta, trabalhando com vetores de características fuzzy.

2.5.3 Algoritmo Fuzzy C-Means

Segundo (ZOU; WANG; HU, 2008), o algoritmo para agrupamento de dados fuzzy foi proposto por (DUNN, 1973) e estendido por (BEZDEK, 1981). A ideia basicamente é de que o conjunto $X = \{x_1, x_2, \ldots, x_n\}$ seja dividido em p clusters, o resultado do agrupamento é expresso pelos graus de pertinência na matriz μ.

O algoritmo FCM procura dividir os dados em conjuntos, minimizando a função objetiva mostrada na Equação (2.16):

$$J = \sum_{i=1}^{n} \sum_{j=1}^{p} \mu_{ij}^{m} d\left(x_i; c_j\right)^2 \tag{2.16}$$

onde:

- μ_{ij} é o grau de pertinência da amostra x_i ao j-ésimo cluster;
- n é o número de instâncias;
- p é o número de clusters considerados no algoritmo o qual deve ser decidido antes da execução;
- $m > 1$ é o parâmetro da fuzzificação[1]. Usualmente, m esta no intervalo de $[1,25;2]$ (COX, 2005);
- x_i um vetor de dados de treinamento, onde $i = 1,2,\ldots,n$ e representa um atributo do dado;
- c_j é o centro de um agrupamento fuzzy ($j = 1,2,\ldots,p$);
- $d\left(x_i; c_j\right)$ é a distância[2] entre x_i e c_j;

A entrada do algoritmo são os n dados, o número de clusters p e o valor de m. Os passos são:

1. Inicialize μ com valores aleatórios entre 0 (nenhuma pertinência) e 1 (pertinência total), onde a soma das pertinências para uma instância deve ser 1;
2. A inicialização do centro do cluster j é mostrado na Equação (2.17):

$$c_{j_{(l)}} = \frac{\sum_{i=1}^{n} \mu_{ij} x_{i_{(l)}}}{\sum_{i=1}^{n} \mu_{ij}} \tag{2.17}$$

   onde (l) é o $l-$ésimo atributo.

[1] Considerando somente valores racionais para simplificar o cálculo das equações (2.16), (2.18) e (3.9). Uma vez que na prática, são usados m racionais.

[2] Quando são valores numéricos, normalmente é usado a distância Euclidiana

3. Calcule o centro do cluster j da seguinte maneira:

$$c_{j_{(l)}} = \frac{\sum_{i=1}^{n} \mu_{ij}^{m} x_{i_{(l)}}}{\sum_{i=1}^{n} \mu_{ij}^{m}} \tag{2.18}$$

   onde (l) é o $l-$ésimo atributo.

4. Calcule um valor inicial para J usando a Equação (2.16);

5. Calcule a tabela da função de pertinência fuzzy μ conforme mostrado na Equação (2.19)

$$\mu_{ij} = \frac{\left(\frac{1}{d(x_i;c_j)}\right)^{\frac{2}{m-1}}}{\sum_{k=1}^{p}\left(\frac{1}{d(x_i;c_k)}\right)^{\frac{2}{m-1}}} \tag{2.19}$$

6. Retornar à etapa 2 até que uma condição de parada seja alcançada.

Algumas condições de parada possíveis são:

- Um número de iterações pré-fixado for executado e/ou;
- O usuário informa um valor de parada $\varepsilon > 0$, e se

$$d(J_U; J_A) \leq \varepsilon$$

  então para, onde J_A é a função objetivo (Equação (2.16)) calculada na iteração atual e J_U é a função objetiva da iteração anterior.

Uma forma alternativa porém equivalente a Equação (2.19) é mostrada na Equação (2.20):

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{p}\left(\frac{d(x_i,C_j)}{d(x_i;c_k)}\right)^{\frac{2}{m-1}}} \tag{2.20}$$

2.5.4 Extensões Intervalares do Algoritmo Fuzzy C-Means

A proposta em (ZHANG; HU; LIU, 2007) é uma extensão do algoritmo FCM para o processamento de dados intervalares. Simulações são realizadas de um conjunto de dados reais

obtidos de um sistema de transporte real. O algoritmo proposto provém do algoritmo FCM e permite processar conjunto de dados intervalares e mostra que a proposta desse algoritmo pode ser usada para extrair regras de intervalos fuzzy tipo 2 (NIERADKA; BUTKIEWICZ, 2007).

Foi proposto em (BOCK, 2000) outra maneira de trabalhar com dados intervalares, denominado como método do centro. Este método consiste em calcular a média aritmética dos valores mínimos e máximos para cada dado intervalar de entrada e depois agrupar eles usando o FCM.

O método proposto em (SATO; LAKHMI, 2006) é uma extensão do método do centro (BOCK, 2000). Nessa extensão, os dados são decompostos em dois conjunto de dados. Um consiste nos valores mínimos e o outro consiste nos valores máximos. Atribui-se pesos para essa séries de dados nos valores mínimos e máximos, respectivamente.

Em diversas pesquisas utilizando dados intervalares, como por exemplo descrito em (de CARVALHO, 2007) e (ZHANG; HU; LIU, 2007), são propostas adaptações no algoritmo FCM para lidar com dados intervalares. Porém, estes algoritmos usam graus de pertinência pontuais. Agrupando dados de entrada como intervalos, (BOCK, 2000) e (SATO; LAKHMI, 2006) também não consideram graus de pertinências intervalares. Os trabalhos (de CARVALHO, 2007), (ZHANG; HU; LIU, 2007), (BOCK, 2000) e (SATO; LAKHMI, 2006) lidam com dados intervalares mas com uma perspectiva pontual no sentido que os graus de pertinências e as métricas são pontuais.

Nesta tese iremos comparar os resultados simbólicos pelo algoritmo proposto Interval ckMeans com os resultados de dois algoritmos que utilizam dados simbólicos. Primeiramente com o algoritmo MSV (Mutual Similarity Value) proposto por (GURU; KIRANAGI; NAGABHUSHAN, 2004) e a seguir o algoritmo IFCM (Interval Fuzzy C-Means) proposto por (de CARVALHO, 2007).

Algoritmo MSV

No trabalho proposto por (GURU; KIRANAGI; NAGABHUSHAN, 2004) foram realizados 3 experimentos. O primeiro experimento consiste numa base de dados que trata gorduras e óleos (dados intervalares), o segundo experimento consiste em agrupar 12 padrões de microcomputadores e o terceiro experimento consiste em classificar baseado nas temperaturas (mínima e máxima) de 37 cidades.

O algoritmo MSV introduz um novo método para o cálculo do grau de similaridade entre os padrões cujas características são do tipo intervalo. Esse método estima o grau de semelhança entre dois padrões em termos de dados múltiplos. Além disso, uma técnica de agrupamento

aglomerativo convencional, introduzindo o conceito de valor de similaridade mútua, é apresentado para agrupamento padrões simbólicos.

Algoritmo IFCM

Em (de CARVALHO, 2007) foi proposto também uma extensão intervalar do algoritmo FCM, onde cada dado de entrada é um intervalo. Este artigo apresenta algoritmos de agrupamento de dados Fuzzy C-Means adaptativos e não-adaptativos para dados intervalares simbólicos, bem como ferramentas para partição fuzzy e de interpretação de clusters. A ideia principal deste método é que não haja uma distância diferente de cada clusters a suas instâncias. O método começa a partir de uma partição inicial e a cada passo modifica a distância do centro dos clusters a suas instâncias, isso é feito até que o critério de convergência atinja um valor estacionário que representa um mínimo local, ou seja, não haja mais mudança na sua distância. Na finalidade de validar o método proposto, foram realizados vários testes em conjuntos de dados intervalares, um consistindo na classificação de carros por determinada característica e outro pela variação da temperatura em diversas cidades.

2.6 Síntese

Neste capítulo apresentou-se uma revisão bibliográfica das técnicas extraídas da computação científica e da mineração de dados, ou seja, a matemática intervalar e agrupamento de dados, respectivamente. Assim, pode-se resolver alguns problemas no agrupamento de dados, tanto pontuais como simbólicos.

Com o uso da matemática intervalar e/ou Lógica Fuzzy, pode-se tirar um grande proveito para a exatidão de resultados. Pode-se empregar essas técnicas onde se tenha incertezas, como de erros de modelagem, de equipamentos de medição e até mesmo erros computacionais (VARGAS, 2008).

Uma correta utilização desses conhecimentos possibilita identificar ou delimitar os graus de erros, ou seja, conhecer os seus limites:

1. O uso da matemática intervalar, percebe-se que obter uma resposta intervalar não garante que esta contenha algo de interesse. Os algoritmos a serem desenvolvidos devem ser algoritmos intervalares e não versões intervalares dos algoritmos pontuais. Os sistemas computacionais atuais são incapazes de representar todos os números reais.

2. A escolha da biblioteca C-XSC é justificada por dois motivos: (i) disponibilidade; e (ii) usabilidade. Disponibilidade, por ser uma biblioteca *freeware*. E usabilidade, pois o C-XSC é uma biblioteca para trabalhar com a linguagem de programação C/C++ e permite escrever algoritmos numéricos produzindo resultados confiáveis num ambiente de programação confortável. Esta linguagem está praticamente disponível em todos os sistemas operacionais, inclusive o sistema operacional *livre* Linux, distribuição Ubuntu (VARGAS, 2008).

3 *Métodos Propostos*

Este capítulo apresenta o ckMeans, uma variante do algoritmo de agrupamento FCM que se diferencia dele na forma de calcular os centroides dos clusters. Em seguida, são apresentadas algumas formas alternativas de calcular as distâncias ao centro dos clusters, dando origem a variantes de FCM e ckMeans e depois é apresentada uma adaptação do algoritmo ckMeans para permitir lidar com dados intervalares.

3.1 Algoritmo ckMeans

O algoritmo K-Means, proposto por (MACQUEEN, 1967), é um método de particionamento (método não-hierárquico) que divide as observações dos dados em K clusters mutuamente exclusivos. Esse algoritmo considera como centro de um grupo o seu centroide. O centroide de um grupo é definido como o vetor soma de todos os vetores correspondentes aos objetos associados a este grupo. Então, a tarefa do algoritmo K-Means é minimizar a função objetivo correspondente à distância total entre os objetos e os centroides dos grupos aos quais esses objetos foram associados.

É importante ressaltar que o algoritmo K-Means é um algoritmo iterativo que minimiza a soma das distâncias de cada objeto ao seu centroide, sobre todos os clusters, movendo objetos entre os grupos até que a soma não possa ser mais diminuída (LLETÍ et al., 2004). O resultado é um conjunto de aglomerados que são tão compactos e bem separados quanto possível.

Devido a se basear no FCM, mudando só a forma de calcular o centro de cada cluster para uma similar à empregada pelo algoritmo K-Means, nomeou-se este novo algoritmo de ckMeans. Com este intuito é criada uma nova matriz auxiliar baseada na matriz μ, chamada de μCrisp, contendo valores 1 ou 0. Cada linha dessa nova matriz tem 1 na posição do maior valor dessa mesma linha na matriz μ e zero nas demais posições da linha. Quando uma coluna da matriz μCrisp, for toda com zeros, é atribuído o valor 1 na posição que corresponde ao maior valor dessa mesma coluna na matriz μ.

O algoritmo ckMeans proposto segue a mesma estrutura do algoritmo FCM, porém, a única alteração deu-se em como calcular o centro dos clusters, ou seja, o c_j.

O algoritmo ckMeans retorna uma matriz μCrisp com os valores dos elementos pertencendo ao conjunto $\{0,1\}$ conforme mostrado na Equação (3.1)[1]. Em outras palavras, μCrisp é uma matriz enquanto μCrisp_{ij} é conteúdo da posição (ij) da matriz μCrisp.

$$\mu Crisp_{ij} = \max\left(\left\lfloor \frac{\mu_{ij}}{\max\limits_{l=1}^{p} \mu_{il}} \right\rfloor, \left\lfloor \frac{\mu_{ij}}{\max\limits_{l=1}^{n} \mu_{lj}} \right\rfloor\right) \tag{3.1}$$

O primeiro argumento no lado direito da Equação (3.1) garante que cada dado tenha o valor 1 no cluster ao qual pertence com maior grau de pertinência e 0 grau de pertinência nos demais. O segundo argumento é para que o maior grau de cada coluna (cluster) seja 1, de modo a assegurar que todos os clusters tenham pelo menos um elemento. Em raras ocasiões, pode acontecer que uma linha tenha mais de um valor 1 (o que não ocorre o algoritmo K-Means original), mas como esta matriz é apenas auxiliar, não ocasionará qualquer transtorno. Por exemplo, se isso fosse permitido no algoritmo K-Means teríamos elementos podendo pertencer, em qualquer iteracao, a mais de um cluster.

Os passos do algoritmo para calcular o μCrisp_{ij} são os seguintes:

1. Ler μ;
2. Em cada linha encontrar o maior valor da matriz μ e atribuir 1 a essa mesma posição em μCrisp e zero nas restantes;
3. Em cada coluna encontrar o maior valor da matriz μ e atribuir 1 a essa mesma posição em μCrisp mantendo os restantes valores inalterados;

Após calcular a matriz μCrisp calculam-se os novos centros dos clusters conforme a Equação (3.2).

$$c_j = \frac{\Sigma_{i=1}^{n} x_i \mu Crisp_{ij}}{\Sigma_{i=1}^{n} \mu Crisp_{ij}} \tag{3.2}$$

O c_j é calculado pela somatória das instâncias que pertencem ao cluster (de forma crisp) e dividido pela quantidade de objetos classificados como 1 na matriz μCrisp deste cluster.

[1]Como usual $\lfloor x \rfloor$ é o maior inteiro menor ou igual a x

Na linguagem de programação C++ utilizando a biblioteca C-XSC, a declaração das váriveis são globais, como é mostrado no Código 5.

Código 5 Declaração das variáveis.

```
/* (Global Variable) */
int p; //Cluster Number
int u; //Attribute Number (Dimension)
int n; //Instance Number
real epsilon; //Epsilon (Stop Criterion)
real m; //Value Fuzziness
rmatrix Xi (u, n); //Matrix Data
rmatrix Cj (u, p); //Matrix Centroid
rmatrix Mij (p, n);//Matrix Membership
```

O pseudocódigo para atualizar a função objetiva de J é mostrado no Código 6.

Código 6 Função objetiva de J.

```
void updateMij()
{
  rmatrix Xi_minus_Cj(p, n);
  int i, j, f, k;
  real dESP =0;
  Xi_minus_Cj = 0;
  for (i = 1 ; i <= n; i++)
  {
     for (j = 1; j <= p; j++)
     {
       for (f = 1; f <= u; f++)
       {
         Xi_minus_Cj[j][i] += euclideanDistance(Xi, Cj);
       }
     }
   }
   real coeff;
   for (i = 1 ; i <= n; i++)
   {
     for (j = 1; j <= p ; j++)
     {
       coeff = 0;
       for (k = 1; k <= p; k++)
       {
         if (Xi_minus_Cj[k][i] == 0)
         {
           Xi_minus_Cj[k][i] = dESP + 0.00001;
         }
         coeff += pow( (Xi_minus_Cj[j][i] / Xi_minus_Cj[k][i]), Q );
       }
       Mij[j][i] = 1 / coeff;
     }
   }
}
```

O Código 7 mostra a função para atualizar o centro dos clusters do algoritmo ckMeans.

Código 7 Atualiza o centro dos clusters c_j.

```
1  void computeCentroid(){
2    int j, i, f, Max = 0;
3    rmatrix MijCrisp(p, n);
4    rvector A(p);
5    Cj = 0; MijCrisp = 0; A = 0;
6    for (i = 1; i <= n; i++){
7      Max = 1;
8      MijCrisp[1][i] = 1;
9      for (j = 2; j <= p; j++){
10       if (Mij[j][i] > Mij[Max][i]){
11         MijCrisp[Max][i] = 0;
12         MijCrisp[j][i] = 1;
13         Max = j;
14       }
15     }
16     A[Max] = A[Max] + 1;
17   }
18   for (j = 1; j <= p; j++){
19     if (A[j] == 0){
20       Max = 1;
21       for (i = 2; i <= n; i++){
22         if (Mij[j][i] > Mij[j][Max]){
23           Max = i;
24         }
25       }
26       MijCrisp[j][Max] = 1;
27       A[j] = 1;
28     }
29   }
30   for (j = 1; j <= p; j++){
31     for (i = 1; i <= n; i++){
32       for (f = 1; f <= u; f++){
33         if (MijCrisp[j][i] == 1){
34           Cj[f][j] = Xi[f][i] + Cj[f][j];
35         }
36       }
37     }
38   }
39   for (j=1; j <= p; j++){
40     for (f=1; f<=u; f++){
41       Cj[f][j] = Cj[f][j] / A[j];
42     }
43   }
44 }
```

As Linhas 1-10 do Código 7 mostram as variáveis e seus respectivos valores iniciais. Das Linhas 12-26 é calculado o μCrisp$_{ij}$ e a soma de objetos classificados como 1 é armazenado na variável A, conforme pode ser visto na Linha 25. Se houver uma classificação onde um cluster não possua os maiores graus de pertinência na matriz μCrisp$_{ij}$, são executadas as Linhas 28-43, onde o algoritmo vai atribuir 1 na posição da matriz μCrisp onde se encontra o maior elemento da coluna da matriz μ. Nas Linhas 45-57 atribui-se a c_j o somatório de todos x que pertencem ao j-ésimo cluster (segundo a matriz μCrisp). Por fim, as Linhas 59-65 dividem esse valor de c_j pela quantidade, de instâncias que pertencem ao j-ésimo cluster, ou seja, ao final c_j terá a média aritmética de todos as instâncias que pertencem ao j-ésimo cluster (segundo a matriz μCrisp).

O pseudocódigo do critério de parada é mostrado no Código 8.

O resultado (saída) de cada iteração do algoritmo K-Means retorna qual cluster pertence

Código 8 Critério de parada.

```
1  real stopCriterion(rmatrix Mij, rmatrix Mij_last)
2  {
3    real temp = 0;
4    int j, i;
5    rmatrix subMij(p, n);
6    subMij = abs(Mij_last - Mij);
7    for (i = 1 ; i <= p; i++)
8    {
9      for (j = 1; j <= n; j++)
10     {
11       temp = abs(temp + subMij[i][j]);
12     }
13   }
14   return temp;
15 }
```

determinada instância. Para os algoritmos FCM e ckMeans é retornado o grau de pertinência de determinada instância a cada cluster. No final, os três algoritmos distribuem os elementos em p clusters. Cabe ao usuário ou especialista interpretar cada cluster. No entanto, quando estamos testando a eficácia de um algoritmo para comparar com outros, usualmente usamos bases de dados já validadas, ou seja onde cada dado tenha uma classificação previamente conhecida ($CC_1, \ldots, CC_p$) para podermos assim determinar a porcentagem de acertos dos algoritmos. Porém um problema é que o algoritmo de agrupamento determina p classes ($CA_1, \ldots, CA_p$) e não necessariamente CA_i corresponde à classe CC_i, com $i = 1, \ldots, m$, e nem sempre é possível determinar a olho nu a melhor forma de associar as classes ($CA_1, \ldots, CA_p$) às classes ($CC_1, \ldots, CC_p$) de forma a maximizar a porcentagem de acertos.

Para este fim foi desenvolvido um algoritmo de "força bruta" que combina todas as possibilidades de associação determinando a porcentagem de acertos de cada associação, dando como saída aquela que devolve o maior número de acertos.

Para a instalação da biblioteca C-XSC, o código fonte dos algoritmos de agrupamento de dados discutidos podem serem obtidos em: `http://rogerio.in`.

3.1.1 Distância Não-Métrica e Métrica-Normalizada

Diversas variantes do algoritmo FCM como os trabalhos de (WU; YANG, 2002) e (ZHANG; CHEN, 2004) tem sido propostos, muitos consideram uma distância diferente da Euclidiana para calcular as Equações (2.16) e (2.19).

Substituindo a distância Euclidiana nos algoritmos K-Means, FCM e ckMeans pelas distâncias que serão mostradas a seguir, estamos obtendo novas variantes desses algoritmos. As distâncias que esta tese utiliza nessas variantes dos algoritmos K-Means, FCM e ckMeans estão descritas a seguir:

- Distância Não-Métrica proposta por (WU; YANG, 2002);
- Distância Métrica-Normalizada proposta por (ZHANG; CHEN, 2004).

Distância Não-Métrica

O trabalho de (WU; YANG, 2002), Wu e Yang propôs duas alternativas de algoritmos de agrupamento de dados, nos algoritmos K-Means e FCM, substituindo a distância mais usual, a distância Euclidiana nesses algoritmos com uma nova função Gaussiana. Embora a função da nova distância seja mais robusta do que a distância Euclidiana, Zhang e Chen (ZHANG; CHEN, 2004) mostram que a distância utilizada por Wu e Yang não é de fato uma métrica.

A distância Não-Métrica $d_{NM} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^+$ é definida na Equação (3.3).

$$d_{NM}(X,Y) = 1 - \exp(-\beta d_E(X,Y)^2) \tag{3.3}$$

para todo $X = (X_1,\ldots,X_n)$ e $Y = (Y_1,\ldots,Y_n)$ em $\mathbb{R}^n$. Onde β é uma função que é calculada a cada iteração conforme mostrado na Equação (3.4).

$$\beta = \left(\frac{\sum_{j=1}^{n} d_E(X_j, \overline{X})}{n} \right)^{-1} \tag{3.4}$$

sendo

$$\overline{X}[k] = \frac{\sum_{j=1}^{n} X_j[k]}{n}$$

com $k = 1,\ldots,p$.

Wu e Yang afirmam que a função de distância na Equação (3.3) é uma métrica, ou seja, todas condições de uma métrica foram satisfeitas. No entanto, o artigo de (ZHANG; CHEN, 2004) mostra que essa distância não é uma métrica por não satisfazer uma das condições da Definição 2.2. A razão pela qual a função da distância na Equação (3.3) não é uma métrica, é que a aplicação da raiz quadrada a uma métrica, não necessariamente é uma métrica.

O pseudocódigo dessa função é mostrado no Código 9.

Código 9 Distância Não-Métrica

```
/* (Global Variable) */
int p = 5; // Cluster Number
int u = 16; // Attribute Number (Dimension)
int n = 3878; // Instance Number
rmatrix Xi (u, n);  // Matrix Data
rmatrix Cj (n, p);  // Matrix Centroid
rmatrix Xi_minus_Cj(p, n); //Distance

/* NMDistance */
void NMDistance()
{
  real acum, vBeta=0;
  real exp = 2.718281826;
  int i, j;
  void euclideanDistance(rmatrix Xi, rmatrix Cj);
  Xi_minus_Cj = 0;
  vBeta = Beta();
  euclideanDistance(Xi, Cj);
  for (i = 1; i <= p; i++)
  {
    acum = 0;
    for (j = 1; j <= n; j++)
    {
      Xi_minus_Cj[i][j] = 1 - pow(exp, -vBeta *
                              pow(Xi_minus_Cj[i][j], 2));
    }
  }
}

```

O pseudocódigo da função β é mostrado no Código 10.

Código 10 Função β.

```
1  /* Beta */
2  real Beta()
3  {
4    rmatrix overline_X(u,1);
5    real sumVectorSQRT(rvector X);
6    real B, BetaTemp = 0;
7    int i, f;
8    real SumVector = 0;
9    rvector acum(u);
10   acum = 0;
11   overline_X = 0;
12   SumVector = 0;
13   B = 0;
14   BetaTemp = 0;
15
16   /* Overline_X */
17   for (f = 1 ; f <= u; f++)
18   {
19     for (i = 1; i <= n; i++)
20     {
21       overline_X[f][1] = (overline_X[f][1] + Xi[f][i]);
22     }
23   }
24   for (f = 1; f <= u; f++)
25   {
26     overline_X[f][1] = overline_X[f][1] / n;
27   }
28
29   /* Beta with Euclidean Distance */
30   for (i = 1 ; i <= n; i++)
31   {
32     for (f = 1; f <= u; f++)
33     {
34       acum[f] = acum[f] + pow(abs(overline_X[f][1] - Xi[f][i]), 2);
35     }
36     SumVector = sumVectorSQRT(acum);
37     BetaTemp = BetaTemp + SumVector;
38     acum = 0;
39   }
40
41   /* Beta */
42   B = pow((BetaTemp / n),-1);
43   return B;
44 }
```

A Linha 36 do Código 10 chama a função sumVectorSQRT(), que é mostrada no Código 11.

Código 11 Função sumVectorSQRT().

```
1  /* Sum Vector */
2  real sumVectorSQRT(rvector acum)
3  {
4    real sumVector;
5    sumVector = 0;
6    for (int i = 1; i <= u; i++)
7    {
8      sumVector = sumVector + acum[i];
9    }
10   return sqrt(sumVector);
11 }
```

Distância Métrica-Normalizada

A distância Métrica-Normalizada é uma alteração da distância Não-Métrica. Essa alteração, permite que a nova função satisfaça as condições da Definição 2.2, isto é, a condição 3 é satisfeita, devido às propriedades da normalização. Assim, a função de distância $d_{MN} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^+$, definida na Equação (3.5) é uma métrica:

$$d_{MN}(X,Y) = \sqrt{1 - \exp(-\beta(d_E(X,Y)^2))} \tag{3.5}$$

para todo $X = (X_1, \ldots, X_n)$ e $Y = (Y_1, \ldots, Y_n)$ em $\mathbb{R}^n$.

Observe que a função de calcular o β (Código 10) é a mesma da distância Não-Normalizada e que a única mudança na equação é o uso da raiz quadrada. O Código 12 mostra o pseudocódigo da implementação dessa função.

Código 12 Distância Métrica Normalizada

```
1  /* (Global Variable) */
2  int p = 5; // Cluster Number
3  int u = 16; // Attribute Number (Dimension)
4  int n = 3878; // Instance Number
5  rmatrix Xi (u, n);  // Matrix Data
6  rmatrix Cj (n, p);  // Matrix Centroid
7  rmatrix Xi_minus_Cj(p, n); //Distance
8
9  /* NMDistance */
10 void MNDistance()
11 {
12   real acum, vBeta=0;
13   real exp = 2.718281826;
14   int i, j;
15   void euclideanDistance(rmatrix Xi, rmatrix Cj);
16   Xi_minus_Cj = 0;
17   vBeta = Beta();
18   euclideanDistance(Xi, Cj);
19   for (i = 1; i <= p; i++)
20   {
21     acum = 0;
22     for (j = 1; j <= n; j++)
23     {
24       Xi_minus_Cj[i][j] = sqrt(1 - pow(exp, -vBeta *
25                                        pow(Xi_minus_Cj[i][j], 2)));
26     }
27   }
28 }
29
```

3.2 Algoritmo Interval ckMeans

De acordo com (FAYYAD, 1996), a técnica de agrupamento procura identificar um conjunto de categorias ou classes para descrever os dados. Segundo (HAN et al., 1996) e (AGRAWAL,

1996), no agrupamento parte-se de uma situação em que não existem classes, somente elementos de um universo. A partir destes elementos, as técnicas de agrupamento são responsáveis por definir as classes e enquadrar os elementos.

O algoritmo ckMeans segue a mesma estrutura do algoritmo Fuzzy C-Means (FCM) (BEZDEK, 1981), porém, a única alteração deu-se em como calcular o centro dos clusters. Por outro lado, o algoritmo Interval ckMeans segue a mesma estrutura do algoritmo ckMeans. Entretanto a forma de inicializar o algoritmo é diferente. Similarmente ao algoritmo FCM, o algoritmo Interval ckMeans também tenta encontrar conjuntos nos dados, minimizando a função objetiva mostrada na equação (3.6).

$$J = \sum_{i=1}^{n} \sum_{j=1}^{p} \mu_{ij}^{m} d_{IE}\left(x_i; c_j\right)^2 \tag{3.6}$$

onde:

- n é o número de instâncias de dados intervalares;
- c é o número de *clusters* considerados no algoritmo, o qual deve ser decidido antes da execução;
- m é um fator de *fuzziness* (um valor maior do que 1) [2];
- X_i é o i-ésimo dado intervalar;
- C_j é o centro (intervalo) do j-ésimo *cluster*;
- $d_{IE}\left(X_i; C_j\right)$ é a distância intervalar definida na Equação 2.11 entre X_i e C_j;

O pseudocódigo das variáveis e da atualização da função objetiva de J são mostrados no Códigos 13 e 14.

[2] Só consideramos valores racionais para não complicar o cálculo das equações (3.6), (2.18) e (3.9). Uma vez que na prática são usados m racionais.

Código 13 Variáveis globais.

```
/* (Global Variable) */
int p = 5; // Cluster Number
int u = 16; // Attribute Number (Dimension)
int n = 3878; // Instance Number
real epsilon = 0.001; // Epsilon (Stop Criterion)
rvector Result(n); // Vector Result
interval m = interval(2, 2); // Value Fuzziness
interval Q = interval(2, 2) / ( m - interval(1, 1) ); // Variable Q
imatrix Xi (u, n);  // Matrix Data
imatrix Cj (u, p);  // Matrix Centroid
imatrix Mij (p, n); // Matrix Membership
```

Código 14 Função objetiva de *J*.

```
/* Atualiza Mij */
void updateMij()
{
  imatrix Xi_minus_Cj(p, n);
  int i, j, f, k;
  real dESP;
  Xi_minus_Cj = interval (0, 0);
  for (i = 1 ; i <= n; i++)
  {
    for (j = 1; j <= p; j++)
    {
      for (f = 1; f <= u; f++)
      {
         Xi_minus_Cj[j][i] += interval(min(abs(Inf(Xi[f][i])-Inf(Cj[f][j])),
                                           abs(Sup(Xi[f][i])-Sup(Cj[f][j]))),
                                       max(abs(Inf(Xi[f][i])-Inf(Cj[f][j])),
                                           abs(Sup(Xi[f][i])-Sup(Cj[f][j]))));
      }
     }
  }
  interval coeff;
  for (i = 1 ; i <= n; i++)
  {
    for (j = 1; j <= p; j++)
    {
      coeff = 0;
      for (k = 1; k <= j-1; k++)
      {
        coeff += interval(1, 1) / pow(Xi_minus_Cj[k][i], Q);
      }
      for (k = j+1; k <= p; k++)
      {
         coeff += interval(1, 1) / pow(Xi_minus_Cj[k][i], Q);
      }
      coeff = coeff * pow(Xi_minus_Cj[j][i], Q) + interval(1, 1);
      Mij[j][i] = interval(1, 1) / coeff;
    }
  }
}
```

A entrada do algoritmo são *n* instâncias de dados intervalares, o número de cluster *c*, ε para o critério de parada e o valor *m* para a "fuzzificação". Suas etapas são:

1. Inicialize μ com subintervalos de $[0;1]$ aleatórios associados a cada par (dados/*clusters*) tais que para cada par dados/*cluster* $(X_i; j)$ e $a_j \in \mu_{i,j}$ temos que existem $a_k \in \mu_{i,k}$ para todo $k \in \{1,\ldots,j-1,j+1,\ldots,c\}$ satisfazendo

$$\sum_{k=1}^{c} a_k = 1$$

2. Calcule o centro do *cluster* j:

   Cria-se uma nova matriz μ, chamada de μCrisp contendo valores 1 ou 0. Cada linha dessa nova matriz tem 1 na posição do maior valor dessa linha na matriz μ e zero nas demais posições da linha.

   O algoritmo Interval ckMeans retorna uma matriz μCrisp com valores em {0, 1} conforme é mostrado na equação (3.1). Ou seja, μCrisp é a matriz enquanto μCrisp_{ij} é o conteúdo dessa matriz na posição (ij).

$$\mu Crisp_{ij} = \left\lfloor \frac{\mu_{ij}}{\max\limits_{l=1}^{p} \mu_{il}} \right\rfloor \tag{3.7}$$

   Após calculada a matriz μCrisp calculam-se os novos centros dos clusters conforme a equação (3.8).

$$C_j = \frac{\Sigma_{i=1}^{n} X_i \mu Crisp_{ij}}{\Sigma_{i=1}^{n} \mu Crisp_{ij}} \tag{3.8}$$

   O C_j é calculado pela somatória dos dados que pertencem ao cluster (de forma crisp) e dividido pela quantidade de objetos classificados como 1 na matriz μCrisp deste cluster.

3. Calcule um valor inicial (um intervalo de dado) para J usando a Equação (3.6);

4. Calcule a tabela de função de pertinência fuzzy intervalar conforme mostrado na equação (3.9);

$$\mu_{ij} = \frac{\left(\frac{1}{d_{IE}\left(X_i;C_j\right)}\right)^{\frac{1}{m-1}}}{\sum\limits_{k=1}^{c}\left(\frac{1}{d_{IE}\left(X_i;C_k\right)}\right)^{\frac{1}{m-1}}} \tag{3.9}$$

5. Retornar a etapa 2 até que uma condição de parada seja alcançada.

Observe que Equação (3.9) é a extensão natural da Equação (2.19). Assim, analogamente ao caso pontual poderíamos reescrever a Equação (3.9) da seguinte maneira:

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{d_{IE}(X_i, C_j)}{d_{IE}(X_i; C_k)} \right)^{\frac{1}{m-1}}} \tag{3.10}$$

Ou seja,

$$\mu_{ij} = \frac{1}{\left(\sum_{k=1}^{j-1} \left(\frac{d_{IE}(X_i, C_j)}{d_{IE}(X_i, C_k)} \right)^{\frac{1}{m-1}} \right) + \left(\frac{d_{IE}(X_i, C_j)}{d_{IE}(X_i, C_j)} \right)^{\frac{1}{m-1}} + \sum_{k=j+1}^{c} \left(\frac{d_{IE}(X_i, C_j)}{d_{IE}(X_i, C_k)} \right)^{\frac{1}{m-1}}} \tag{3.11}$$

A divisão de um intervalo por ele mesmo não és $[1,1]$, o qual aumenta desnecessariamente o diâmetro do resultado envolvendo esse tipo de divisões como no caso da Equação (3.11). Assim, se substituirmos nessa equação

$$\left(\frac{d_{IE}(X_i, C_j)}{d_{IE}(X_i; C_j)} \right)^{\frac{1}{m-1}}$$

por $[1,1]$ obteremos um intervalo mais estreito para μ_{ij}.

Portanto, nós usaremos a seguinte forma de calcular μ_{ij}.

$$\mu_{ij} = \frac{1}{\left(\sum_{k=1}^{j-1} \left(\frac{d_{IE}(X_i, C_j)}{d_{IE}(X_i, C_k)} \right)^{\frac{1}{m-1}} \right) + [1,1] + \sum_{k=j+1}^{c} \left(\frac{d_{IE}(X_i, C_j)}{d_{IE}(X_i, C_k)} \right)^{\frac{1}{m-1}}} \tag{3.12}$$

Após calculada a matriz μCrisp calculam-se os novos centros dos clusters conforme a Equação (3.2). O pseudocódigo da atualização do centro dos clusters pode ser visto no Código 15.

Código 15 Atualiza o centro dos clusters (c_j).

```
/* Atualiza cj */
void computeCentroid()
{
  int l, j, i, f, k, acumj, y=0;
  rmatrix MijCrisp(p,   n);
  Cj = 0;
  int Max[p];
  MijCrisp = 0;
  for (i = 1; i <= n; i++)
  {
    f = 0;
    Max[0] = 1;
    for (int z=1; z<=p; z++)
    {
      Max[z] = 0;
    }
    MijCrisp[1][i] = 1;
    for (j = 2; j <= p; j++)
    {
      y = Max[0];
      if (Mij[j][i] == Mij[y][i])
      {
        MijCrisp[j][i] = 1;
        f = f + 1;
        Max[f] = j;
      }
      if ( (Inf(Mij[j][i]) > Inf(Mij[y][i])) || ( Inf(Mij[j][i]) == Inf(Mij[y][i]) &&
                                                  Sup(Mij[y][i]) > Sup(Mij[j][i])) )
      {
        MijCrisp[j][i] = 1;
        for (k = 0; k <= f; k++)
        {
          y = Max[k];
          MijCrisp[y][i] = 0;
        }
        Max[0] = j;
        for (int z=1; z<=p; z++)
        {
          Max[z] = 0;
        }
        f = 0;
      }
    }
  }
  for (j = 1; j <= p; j++)
  {
    Max[0] = 1;
    f = 0;
    for (i = 2; i <= n; i++)
    {
      y = Max[0];
      if (Mij[j][i] == Mij[j][y])
      {
        f = f + 1;
        Max[f] = i;
      }
      if ( (Inf(Mij[j][i]) > Inf(Mij[j][y])) || ( Inf(Mij[j][i]) == Inf(Mij[j][y]) &&
                                                  Sup(Mij[j][y]) > Sup(Mij[j][i])) )
      {
        Max[0] = i;
        for (int z = 1; z < p; z++)
        {
          Max[z] = 0;
        }
        f = 0;
      }
    }
    l = 0;
    while (Max[l] > 0)
    {
      y = Max[l];
      MijCrisp[j][y] = 1;
      l = l + 1;
    }
  }
  for (j = 1; j <= p; j++)
  {
    acumj = 0;
    for (i = 1; i <= n; i++)
    {
      if (MijCrisp[j][i] == 1)
      {
        acumj = acumj + 1;
        for (f = 1; f <= u; f++)
        {
          Cj[f][j] = Xi[f][i] + Cj[f][j];
        }
      }
    }
    for (f = 1; f <= u; f++)
    {
      Cj[f][j] = Cj[f][j] / acumj;
    }
  }
}
```

Algumas condições de parada possíveis são:

- Um número de iterações pré-fixado for executado, e pode-se considerar que o algoritmo conseguiu agrupar os dados;
- O usuário informe um valor de parada $\varepsilon > 0$, e se

$$d_{IE}\left(J_U; J_A\right) \leq [\varepsilon; \varepsilon]$$

então o algoritmo para. Neste caso, J_A é a função objetiva (Equação (3.6)) calculada da iteração atual e J_U é a função objetiva da iteração anterior.

3.3 Síntese

Foram propostos dois algoritmos: o ckMeans que através de uma nova equação matemática, permite calcular o novos valores dos centros dos clusters e portanto outros algoritmos de agrupamento de dados tipo FCM poderiam também usar desta forma de calcular os centros dos clusters; e o algoritmo Interval ckMeans, que permite manipular dados simbólicos (intervalares) sem nenhuma conversão para dados pontuais. Além disso, foram consideradas duas distâncias alternativas para os algoritmos K-Means, FCM e ckMeans.

A matemática intervalar foi escolhida para modelar as operações intervalares devido a sua característica de tratar os números não como valores pontuais, mas sim através de um conjunto de valores (intervalo).

Adotou-se a teoria da Lógica Fuzzy por apresentar as seguintes características em relação a outras técnicas de controle (VARGAS, 2008):

- Robusta por não requerer entradas precisas;
- Modificada facilmente, pois é baseada em regras;
- Solução mais rápida e barata em alguns casos;
- Implementável em microprocessadores.

4 Resultados Comparativos

4.1 Experimentos

Inicialmente implementou-se o algoritmo K-Means seguindo o tutorial disponível em (TEKNOMO, 2010) e o algoritmo FCM (tradicional) baseada na implementação[1] de (DEGRUIJTER; MCBRATNEY, 1988), disponível em
`http://www.usyd.edu.au/agric/acpa/fkme/program.html`.

Todos os algoritmos aqui discutidos (K-Means, FCM, ckMeans e Interval ckMeans), foram executados em um Notebook Intel® Core^{TM} i3 CPU M 350 2.27GHz, 3 GB de memória principal, usando o sistema operacional Linux (Kernel 2.6.35-28-generic, GNOME 2.32, distribuição Ubuntu 10.10) e desenvolvidos em C++ (Versão 4.4) enriquecido com a biblioteca C-XSC (versão 2.5) (HOFSCHUSTER; KRÄMER, 2003), disponível em `http://www.xsc.de`.

4.2 Resultados Pontuais

No intuito de comparar os algoritmos K-Means, FCM e ckMeans, tanto do ponto de vista de eficácia (porcentagem de acertos) como eficiência (tempo de execução e quantidade de iterações), serão executados usando 3 bases de dados, as bases Iris (FISHER, 1936), Sonar (GORMAN; SEJNOWSKI, 1988) e Vogal. Essas bases são mostradas a seguir, com as melhores configurações dos parâmetros *m* e *epsilon*. No final desta seção, mostra-se estas mesmas bases e mais outras sete com o desempenho e variações dos parâmetros *m* e *epsilon*.

Os dados usados nas simulações são dados validados, ou seja, que trouxeram sua classificação original, facilitando assim a avaliação das soluções através do Corrected Rand Index (RAND, 1971) (índice que mede o quanto dois clusters são parecidos).

[1] De fato, essa implementação reporta exatamente os mesmos valores de (DEGRUIJTER; MCBRATNEY, 1988).

4.2.1 Base Iris

A base de dados Iris (FISHER, 1936) é talvez o banco de dados mais utilizado na literatura no reconhecimento de padrões. Foram simulados os algoritmos K-Means, FCM e ckMeans com esta base de dados (da UCI Repositório (FRANK; ASUNCION, 2010)). Esta base de dados contém 3 séries de 50 instâncias (150 instâncias ao todo), cada conjunto correspondente a uma das três classes da planta íris (Iris setosa, Iris Versicolour e Iris virginica) conforme Figura 4.2.1.

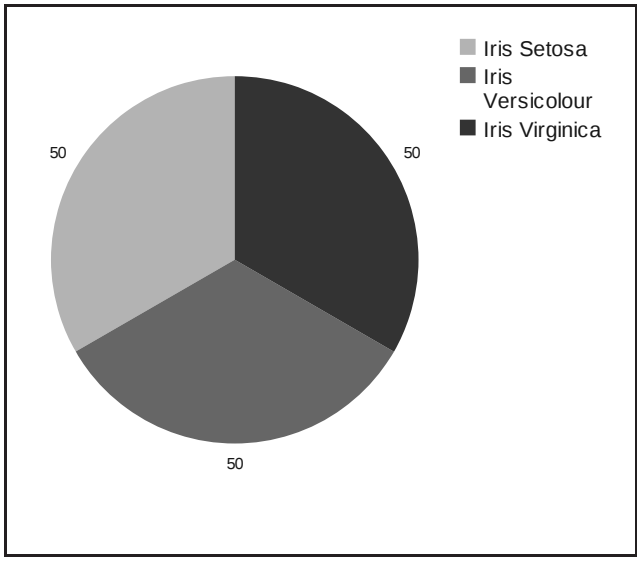


Figura 4.1: Divisão das instâncias.

Cada registro é descrito em termos de 4 variáveis numéricas (comprimento da sépala, largura da sépala, comprimento da pétala e largura da pétala) todos os dados em centímetros.

Parâmetros de Inicialização

Os parâmetros de entrada são 150 instâncias, com quatro atributos cada (comprimento da sépala, largura da sépala, comprimento da pétala e largura da pétala). Estes dados estão previamente classificados em uma das seguintes classes: Iris Setosa (instâncias de 1 a 50), Iris Versicolour (instâncias de 51 a 100) e Iris Virginica (instâncias de 101 a 150). O número de clusters são 3, o valor de fuzziness é $m = 2$ e $\varepsilon = 0,0001$. Estes parâmetros foram usados nas três configurações dos algoritmos simulados (K-Means, FCM e ckMeans). Nesta configuração os algoritmos FCM e ckMeans tiveram o melhor desempenho. Os valores iniciais de μ_{ij} são números aleatórios. Usou-se os mesmos valores para inicializar os algoritmos FCM e ckMeans.

A Tabela 4.1 mostra o valor inicial do centro dos clusters, isto é, c_j nos algoritmos FCM e

ckMeans.

Tabela 4.1: Inicialização de c_j.

	comprimento da sépala	**largura da sépala**	**comprimento da pétala**	**largura da pétala**
Cluster 1	5,7981	3,0431	3,6758	1,1721
Cluster 2	5,8653	3,0681	3,7862	1,2215
Cluster 3	5,8683	3,0511	3,8173	1,2034

O comprimento e a largura estão expressos em centímetros (cm).

Comparações entre os algoritmos

O resultado final do c_j com $j = 1, 2,$ e 3 nos algoritmos FCM e ckMeans é mostrado na Tabela 4.2. Observe que os centros dos clusters em todos os clusters são similares.

Tabela 4.2: c_j Resultado com FCM e ckMeans.

	comprimento da sépala		largura da sépala		comprimento da pétala		largura da pétala	
	FCM	**ckMeans**	**FCM**	**ckMeans**	**FCM**	**ckMeans**	**FCM**	**ckMeans**
Cluster 1	5,0035	5,0060	3,4030	3,4180	1,4849	1,4640	0,2515	0,2440
Cluster 2	5,8885	5,8836	2,7609	2,7409	4,3632	4,3885	1,3969	1,4344
Cluster 3	6,7741	6,8538	3,0521	3,0769	5,6457	5,7153	2,0531	2,0538

A alteração da distância Euclidiana para as distâncias Não-Normalizada ou Métrica-Normalizada, não influenciaram neste resultado e portanto as instâncias agrupadas em uma determinada classe pelos algoritmos FCM e ckMeans utilizando qualquer das distâncias apresentadas (distâncias: Euclidiana, Não-Métrica e Métrica-Normalizada) foram exatamente as mesmas. A Tabela 4.3 apresenta para cada cluster as quantidades de instâncias de cada classe que foram classificadas nesse cluster.

Tabela 4.3: Base Iris agrupada pelos algoritmos FCM e ckMeans utilizando as distâncias Euclidiana, Não-Métrica e Métrica-Normalizada.

Assinalado ao Cluster	1	2	3
Iris-setosa	50	0	0
Iris-virginica	0	15	35
Iris-versicolor	0	48	2

Utilizando o algoritmo K-Means com as mesmas distâncias simuladas dos algoritmos FCM e ckMeans obtém-se a Tabela 4.4. De forma análoga ao caso de FCM e ckMeans, as distâncias Euclidiana, Não-Métrica e Métrica-Normalizada não afetam a classificação deste algoritmo.

Tabela 4.4: Base Iris agrupada pelo algoritmo K-Means utilizando as distâncias Euclidiana, Não-Métrica e Métrica-Normalizada.

Assinalado ao Cluster	1	2	3
Iris-setosa	50	0	0
Iris-virginica	0	14	36
Iris-versicolor	0	47	3

Onde o rótulo dos clusters são os seguintes:

- Cluster 1 –> Iris-setosa;
- Cluster 2 –> Iris-versicolor;
- Cluster 3 –> Iris-virginica.

Embora a classificação das instâncias seja diferente, conforme mostrado nas Tabelas 4.3 e 4.4, onde os algoritmos FCM e ckMeans classificaram 2 instâncias incorretamente no Cluster 2 e 15 instâncias incorretamente no Cluster 3, respectivamente. Enquanto o algoritmo K-Means classificou 3 e 14 instâncias incorretamente nos Clusters 2 e 3, respectivamente. Dessa forma, a taxa de objetos classificados nos três algoritmos são os mesmos, isto é, 17 instâncias classificadas incorretamente, correspondendo a 11,33% das instâncias. Entretanto, o número de iterações e o tempo de processamento até o sistema convergir são diferentes, conforme mostrado na Tabela 4.5.

Tabela 4.5: Performance entre os algoritmos na base Iris.

	Quantidade de Iterações			Tempo do Processamento (s)		
Algoritmo	d_E	d_{NM}	d_{MN}	d_E	d_{NM}	d_{MN}
K-Means	13	13	13	0,56	0,98	0,98
FCM	13	13	13	1,67	2,42	2,46
ckMeans	11	11	11	1,33	2,03	2,06

Ao compararmos a performance entre os algoritmos FCM e ckMeans, nota-se que o algoritmo ckMeans tem o menor tempo de convergência e a mesma qualidade nos resultados do que o algoritmo FCM. Entretanto, nesta base de dados o algoritmo K-Means é o mais recomendado visto que a qualidade dos resultados (índice de acertos) é a mesma que a dos outros dois algoritmos, porém, mesmo que no caso do ckMeans o número de iterações tenha sido menor, o tempo até o sistema convergir no algoritmo K-Means é menor do que nos demais algoritmos simulados. Isto, deve-se ao fato que a base de dados Iris é relativamente pequena, assim o algoritmo K-Means tem um desempenho melhor que os demais algoritmos. A seguir, será mostrado que isso não ocorre em bases de dados maiores como a Vogal.

4.2.2 Base Sonar

Este conjunto de dados utilizado por (GORMAN; SEJNOWSKI, 1988), serviu como estudo para a classificação de sinais sonares no treinamento de uma rede neural. A tarefa é treinar uma rede neural que seja capaz de distinguir entre os sinais de sonar que ricocheteiam em um cilindro de metal e os que ricocheteiam em uma pedra no formato cilíndrico.

Esta base de dados está assim dividida: as primeiras 97 instâncias, correspondem à classe "Mine" e as demais instâncias (entre 98 a 208), pertencem à classe "Rock", conforme mostrado na Figura 4.2. Cada registro (instância) é descrito em termos de 60 atributos numéricos[2].

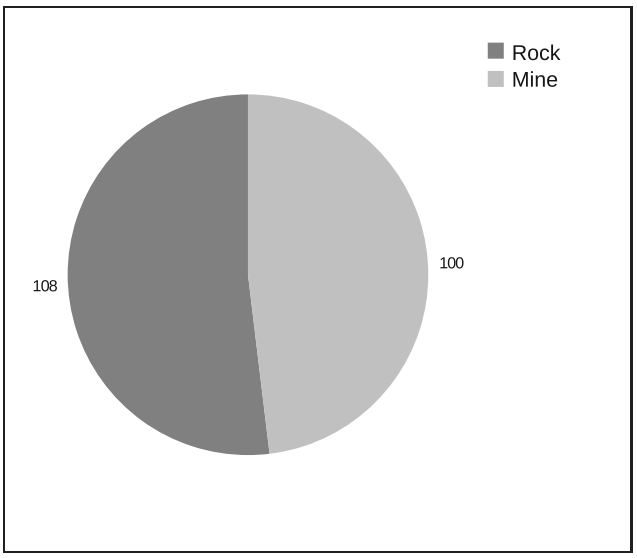


Figura 4.2: Divisão das instâncias.

A base de dados Sonar (da UCI Repositório (FRANK; ASUNCION, 2010)) esta disponível em: `http://www.cs.umb.edu/~rickb/files/UCI/sonar.arff`.

Parâmetros de Inicialização

O número de clusters são 2, o valor de fuzziness é $m = 1,5$ e $\varepsilon = 0,0001$. Estes parâmetros foram usados nas três configurações dos algoritmos simulados (K-Means, FCM e ckMeans). Nesta configuração os algoritmos FCM e ckMeans tiveram o melhor desempenho. Os valores iniciais de μ_{ij} são números aleatórios. Usou-se os mesmos valores para inicializar os algoritmos FCM e ckMeans.

[2] A especificação de cada atributo, pode ser visto em `www.cs.umb.edu/~rickb/files/UCI/sonar.arff`

Comparações entre os algoritmos

As instâncias agrupadas em classes nos algoritmos K-Means, FCM e ckMeans utilizando a distância Euclidiana foram as mesmas, mostrado na Tabela 4.6.

Tabela 4.6: Base Sonar agrupada pelos algoritmos K-Means, FCM e ckMeans utilizando a distância Euclidiana.

Assinalado ao Cluster	1	2
Rock	62	46
Mine	50	50

Onde o rótulo dos clusters são os seguintes:

- Cluster 1 –> Rock;
- Cluster 2 –> Mine.

A quantidade de clusters classificados corretamente utilizando a distância Euclidiana nos algoritmos simulados foram 102 instâncias.

A qualidade das classificações proporcionadas pelos algoritmos K-Means e ckMeans utilizando as distâncias Não-Métrica e Métrica-Normalizada é a mesma que utilizando a distância Euclidiana. As Tabelas 4.7 e 4.8 apresentam as classificações obtidas utilizando o algoritmo FCM e ckMeans com as distâncias Não-Métrica e Métrica-Normalizada, respectivamente.

Tabela 4.7: Base Sonar agrupada pelo algoritmo FCM e ckMeans utilizando a distância Não-Métrica.

Assinalado ao Cluster	1	2
Rock	62	46
Mine	55	45

Tabela 4.8: Base Sonar agrupada pelo algoritmo FCM e ckMeans utilizando a distância Métrica-Normalizada.

Assinalado ao Cluster	1	2
Rock	62	48
Mine	54	46

A distância Não-Métrica quando utilizada no algoritmo FCM e ckMeans obteve 107 instâncias classificadas corretamente, enquanto a distância Métrica-Normalizada teve 108 instâncias classificadas corretamente.

O número de iterações e o tempo de processamento até o sistema convergir são diferentes, conforme mostrado na Tabela 4.9.

Tabela 4.9: Performance entre os algoritmos na base Sonar.

	Quantidade de Iterações			Tempo do Processamento (s)		
Algoritmo	d_E	d_{NM}	d_{MN}	d_E	d_{NM}	d_{MN}
K-Means	16	16	16	8,94	13,61	8,94
FCM	12	18	27	13,73	31,63	46,65
ckMeans	5	8	5	5,74	8,73	8,74

Ao compararmos a performance entre os algoritmos mostrados, o algoritmo ckMeans, tem o menor tempo de convergência. Entretanto, o algoritmo FCM tem o pior desempenho se comparado com os demais. Mesmo o algoritmo K-Means ser mais simples (menos operações matemáticas), o algoritmo ckMeans tem as mesmas instâncias classificadas, porém com um tempo de convergência muito menor.

4.2.3 Base Mamografia

A mamografia é o método mais eficaz para o rastreamento do câncer de mama disponíveis atualmente. No entanto, a biópsia de mama resultantes da interpretação mamografia leva a aproximadamente 70% biópsias desnecessárias com resultados benignos. Para reduzir o elevado número de biópsias mamárias desnecessárias, vários CAD (Computer Aidded Design) têm sido propostas nos últimos anos. Este conjunto de dados[3] podem ser usados para predizer a gravidade (benigno ou maligno) de uma lesão de massa de mamografia. Esta base esta dividida em 516 dados classificados como benignos e 445 dados classificados como malignos que foram identificadas na mamografia coletadas no Instituto de Radiologia do da. Universidade Erlangen-Nuremberg, entre 2003 e 2006.

O número de instâncias distribuídas nos clusters usando os algoritmos K-Means, FCM e ckMeans são os mesmos. A Figura 4.3 mostra a distribuição das instâncias a cada classe.

[3]Esta base de dados está disponível em `http://archive.ics.uci.edu/ml/datasets/Mammographic+Mass`.

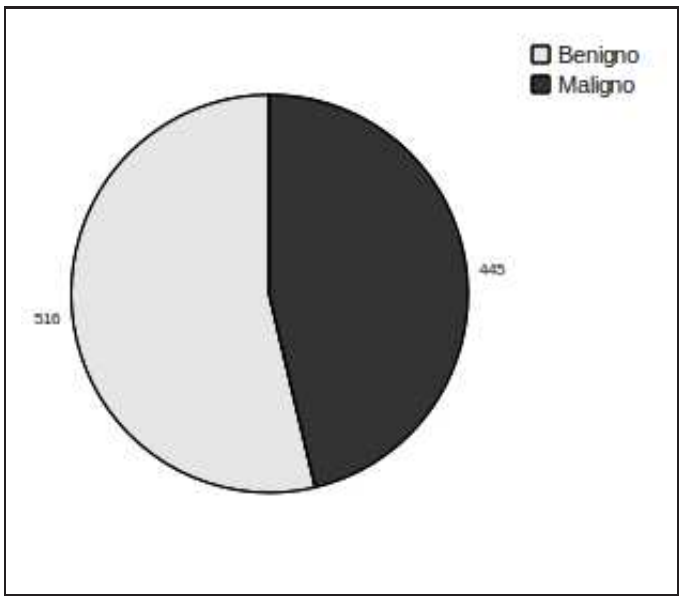


Figura 4.3: Instâncias

Observe que a classe Benigno teve 516 instâncias e a classe Maligno teve 445 instâncias, totalizando 961 instâncias.

Parâmetros de Inicialização

O número de clusters são 2, o valor de fuzziness é $m = 1,25$ e $\varepsilon = 0,0001$. Estes parâmetros foram usados nas três configurações dos algoritmos simulados (K-Means, FCM e ckMeans). Nesta configuração os algoritmos FCM e ckMeans tiveram o melhor desempenho. Os valores iniciais de μ_{ij} são números aleatórios. Usou-se os mesmos valores para inicializar os algoritmos FCM e ckMeans.

Comparações entre os algoritmos

Foram executadas diversas simulações nessa base de dados, alterou-se os parâmetros m e ε nos algoritmos ckMeans e FCM. Essa base de dados, pelas simulações não é possível classifcar corretamente as instâncias, visto que os graus de pertinências associado de cada instância ao seu respectivo cluster esta muito próximo de $0,5$.

4.2.4 Base Vogal

O objetivo é identificar cada letra de um grande número de pixels preto-e-branco, contendo as cinco vogais do alfabeto inglês. Os caracteres de imagens são de uma base de 20 fontes

diferentes e cada letra dentro dessas 20 fontes foram geradas aleatoriamente para produzir um arquivo de 3.878 instâncias únicas. Cada instância foi convertida em 16 atributos primitivos numéricos, que foram, então, dimensionadas para ficar em um intervalo de valores inteiros entre 0 a 5. O banco de dados Vogal consiste na extração de instâncias vogais do banco de dados Letter [4].

O número de instâncias distribuídas nos clusters usando os algoritmos K-Means, FCM e ckMeans são os mesmos. A Figura 4.4 mostra a distribuição das instâncias a cada classe.

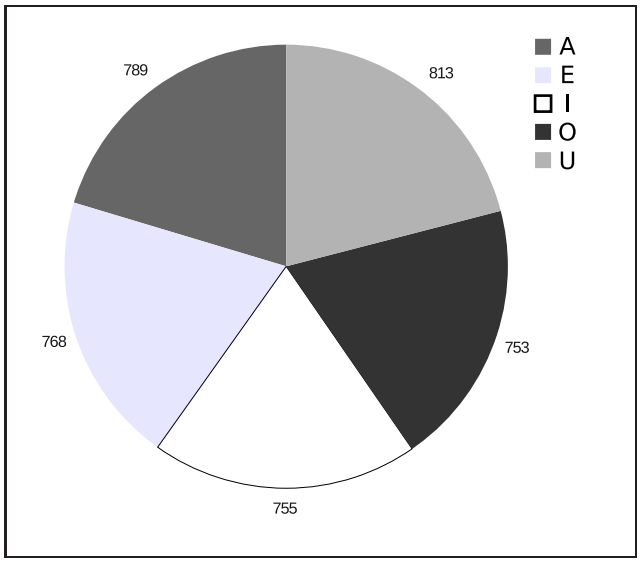


Figura 4.4: Instâncias.

Os valores iniciais de μ_{ij} são números aleatórios. Usou-se os mesmos valores para inicializar os algoritmos K-Means, FCM e ckMeans. O valor inicial de c_j foram os mesmos nos algoritmos FCM e ckMeans.

Parâmetros de Inicialização

O número de clusters são 5, o valor de fuzziness é $m = 1,5$ e $\varepsilon = 0,0001$. Estes parâmetros foram usados nas três configurações dos algoritmos simulados (K-Means, FCM e ckMeans). Nesta configuração os algoritmos FCM e ckMeans tiveram o melhor desempenho. Os valores iniciais de μ_{ij} são números aleatórios. Usou-se os mesmos valores para inicializar os algoritmos FCM e ckMeans.

[4]Esta base de dados está disponível em `http://archive.ics.uci.edu/ml/datasets/Letter+Recognition`.

Comparações entre os algoritmos

Ao simular os resultados usando as distâncias Não-Métrica e Métrica-Normalizada com os algoritmos FCM e ckMeans tiveram uma classificação muito imprecisa, ou seja, cada instância tinha os mesmos graus de pertinências aos clusters ($\mu_{ij} = 0,2$). Assim, optou-se em não representar através de tabelas e nem gráficos esses resultados. Podemos concluir que as distâncias Não-Métrica e Métrica-Normalizada não se aplicam (péssima classificação) a essa base de dados.

A Tabela 4.10 mostra as instâncias classificadas usando o algoritmo K-Means.

Tabela 4.10: Classificação das instâncias usando o algoritmo K-Means

	1	**2**	**3**	**4**	**5**
A	123	10	3	650	3
E	215	16	322	8	207
I	121	3	85	18	528
O	343	224	23	10	153
U	197	380	197	0	39

A Tabela 4.11 mostra as instâncias classificadas usando o algoritmo FCM.

Tabela 4.11: Classificação das instâncias usando o algoritmo FCM

	1	**2**	**3**	**4**	**5**
A	123	10	3	650	3
E	215	16	322	8	207
I	121	3	85	18	528
O	343	224	23	10	153
U	197	380	197	0	39

A Tabela 4.12 mostra as instâncias classificadas usando o algoritmo ckMeans. Observe que há uma diferença na classificação entre os algoritmos.

Tabela 4.12: Classificação das instâncias usando o algoritmo ckMeans

	1	**2**	**3**	**4**	**5**
A	133	1	9	645	1
E	266	267	13	3	219
I	150	18	3	9	575
O	379	5	217	9	143
U	201	192	397	0	23

A Tabela 4.13 mostra a classificação de cada dado nas classes. O número de instâncias classificadas incorretamente em cada cluster é 1900, o que corresponde a 48,99% com o algoritmo K-Means.

Tabela 4.13: Instâncias classificadas corretamente e incorretamente com o algoritmo K-Means.

	Quantidade de Instâncias
A	670
E	230
I	321
O	499
U	258
Incorretos	1900

A Tabela 4.14 mostra a classificação de cada dado nas classes. O número de instâncias classificadas incorretamente em cada cluster é 1655, o que corresponde a 42.67% com o algoritmo FCM.

Tabela 4.14: Instâncias classificadas corretamente e incorretamente com o algoritmo FCM.

	Quantidade de Instâncias
A	650
E	322
I	528
O	343
U	380
Incorretos	1655

A Tabela 4.15 mostra a classificação de cada dado nas classes. O número de instâncias classificadas incorretamente em cada cluster é 1615, o que corresponde a 41,64% com o algoritmo ckMeans.

Tabela 4.15: Instâncias classificadas corretamente e incorretamente com o algoritmo ckMeans.

	Quantidade de Instâncias
A	645
E	267
I	575
O	379
U	397
Incorretos	1615

A Tabela 4.16 mostra a quantidade de iterações, a média do tempo de processamento de cada iteração em segundos e o tempo total em segundos que os algoritmos levaram para convergir.

Tabela 4.16: Performance.

	K-Means	**FCM**	**ckMeans**
Iterações	15	127	22
Tempo médio de cada iteração	0,59	2,24	1,80
Tempo Convergência	8,9	285,2	39,67

O algoritmo K-Means tem o menor número de iterações e menor tempo de convergência. No entanto, se comparado com outros algoritmos simulados apresentou o maior número de instâncias classificadas incorretamente. Foram 15 iterações, 8,9 segundos de tempo médio para cada iteração e 0,59 segundos para o tempo total de convergência.

O algoritmo FCM convergiu com 127 iterações e o algoritmo ckMeans convergiu com 22 iterações. Note também que o tempo de processamento no algoritmo ckMeans foi menor do que o algoritmo FCM, com 39,67 segundos e 285,2 segundos, respectivamente.

Na simulação usando a base de dados Vogal, os algoritmos não obtiveram as mesmas classificações. O algoritmo ckMeans teve uma taxa de acerto superior se comparado com o algoritmo FCM, com 40 acertos a mais do que o algoritmo FCM, o ckMeans foi muito mais rápido. Isso representa que o algoritmo ckMeans teve 58.35% contra 57.32% de acertos com o algoritmo FCM.

O valor inicial de J foram iguais entre os algoritmos FCM e ckMeans (375.547) e a última iteração no algoritmo FCM foram de 0,0081445374 na iteração 127. No algoritmo ckMeans o valor de J foram zero na iteração 22.

4.2.5 Outras Configurações

O índice externo *Adjusted Rand Index* retorna um valor entre o intervalo $[0;1]$, similarmente a Lógica Fuzzy, valor mais próximo a um, identifica que o algoritmo obteve uma taxa de acerto elevada. O valor próximo a zero, significa que o algoritmo agrupou os dados erroneamente.

Para validar o desempenho dos algoritmos, testamos os algoritmos K-Means, FCM e ckMeans em três métricas, distância Euclidiana, Métrica-Normalizada e Não-Métrica. Nos algorimos FCM e ckMeans além de simular as três métricas, o parâmetro fuzziness m foi variado em (1,25), (1,5), (2) e (2,5).

O *Adjusted Rand Index* mostrado na Tabela 4.17 foram obtidos após 100 inicializações (aleatórias).

Na Tabela 4.17 (base Iris) observe que o *Adjusted Rand Index* ficou em torno de 0,87 no K-Means. A base Iris por possuir poucas instâncias e poucos atríbutos, o algoritmo K-Means obteve um melhor desempenho se compararmos a qualidade dos resultados. Entretanto, se compararmos a distância Euclidiana nos algoritmos FCM e ckMeans, observa-se a qualidade dos resultados são praticamente idênticos. Na distância Não-Métrica, o desvio padrão no FCM é praticamente zero, enquanto no algoritmo ckMeans é variado. Na Distância Métrica-Normalizada

o algoritmo ckMeans manteve um padrão, o índice *Adjusted Rand Index* ficou em torno de 0,87, o mesmo do algoritmo K-Means. Já em relação ao algoritmo FCM, a mediana da distância Métrica-Normalizada variou entre o intervalo $[0,74; 0,90]$.

A Tabela 4.18 (base Sonar) tem as mesmas configurações para obter os dados da Tabela 4.17, a diferença está no número de iterações, para obter o *Adjusted Rand Index* inicializou-se 50 vezes para cada parâmetro.

Analisando a Tabela 4.18 de forma qualitativa, o *Adjusted Rand Index* é similar nos três casos, possuindo o valor em torno de 0,5.

A Tabela 4.19 (base Vogais) mostra o *Adjusted Rand Index* obtido nas métricas: distância Euclidiana, Distância Não-Métrica e Distância Métrica-Normalizada. Nos algoritmos FCM e ckMeans variou-se também o parâmetro fuzziness *m* em (1,25), (1,5), (2) e (2,5). Nesta base, realizou-se 15 inicializações para obter o *Adjusted Rand Index*.

Ainda sobre a Tabela 4.19, de forma geral o algoritmo ckMeans teve um maior número de acerto se comparado aos algoritmos K-Means e FCM. Com o parâmetro fuzziness $m = 2,5$ na distância Métrica-Normalizada o algoritmo ckMeans teve a mediana em 15 iterações do *Adjusted Rand Index* em 0,78 enquanto nos algoritmos K-Means e FCM ficaram em 0,73 e 0,7112, respectivamente.

Observa-se que o algoritmo em bases com poucos atríbutos e instâncias como a Iris e a Sonar a utilização do algoritmo K-Means é recomendado. Enquanto em bases maiores, o algoritmo proposto ckMeans mostra-se uma excelente alternativa. Praticamente em todas as configurações das bases simuladas, o algoritmo ckMeans leva uma vantagem na classificação se comparado ao algoritmo FCM.

4.3 Resultados Intervalares

4.3.1 Base Temperatura

A base de dados a ser analisada pelos algoritmos é uma classificação de cidades baseada na temperatura usada por (GURU; KIRANAGI; NAGABHUSHAN, 2004). Foi obtido a temperatura mínima e máxima (em graus Celsius) do mês em um determinado ano entre 37 cidades. A Tabela 4.20 mostra a observação em meses das 37 cidades analisadas.

Parâmetros de Inicialização

Os parâmetros de entrada são 37 dados (obtidos da base de dados) e estes dados referem-se à 37 cidades espalhadas entre os continentes. O número de clusters são 4, o valor de fuzziness[5] é $m = 2$ e $\varepsilon = 0,001$. Estes parâmetros foram usados nos algoritmos IFCM e Interval ckMeans.

4.3.2 Resultados Intervalares

A Tabela 4.21 mostra a classificação final obtida com o algoritmo IFCM.

Tabela 4.21: Partição crisp com 4 clusters obtido com o algoritmo IFCM.

Partição	**IFCM**
Cluster 1	Bahrain, Cairo, Hong Kong, Mexico City, Nairobi, New Delhi, Sydney
Cluster 2	Amsterdan, Copenhagen, Frankfurt, Geneva, London, Moscow, Munich, New York, Paris, Stockholm, Toronto, Vienna
Cluster 3	Bombay, Calcutta, Colombo, Dubai, Kula Lampur, Madras, Manila, Mauritius
Cluster 4	Athens, Lisbon, Madrid, Rome, San Francisco, Seoul, Tehran, Tokyo, Zurich

O algoritmo IFCM realizou 60 iterações para obter o melhor resultado de acordo com o critério de convergência escolhido. Entretanto, com o algoritmo MSV (GURU; KIRANAGI; NAGABHUSHAN, 2004) obteve-se uma classificação diferente, conforme mostrado na Tabela 4.22.

Tabela 4.22: Partição crisp com 4 clusters obtido com o algoritmo MSV.

Partição	**MSV**
Cluster 1	Bahrain, Bombay, Cairo, Calcutta, Colombo, Dubai, Hong Kong, Kula Lumpur, Madras, Manila, Mexico City, Nairobi, New Delhi, Sydney
Cluster 2	Amsterdan, Athens, Copenhagem, Frankfurt, Geneva, Lisbon, London, Madrid, Moscow, Munich, New York, Paris, Rome, San Francisco, Seoul, Stockholm, Tokyo, Toronto, Vienna, Zurich
Cluster 3	Mauritius
Cluster 4	Tehran

Os cluster 3 e 4 só ficaram um dado, Mauritius e Tehran, respectivamente.

A Tabela 4.23 mostra o grau de pertinência após as 7 iterações.

[5]Nos testes realizados com $m < 2$ o sistema apresentou resultados insatisfatórios, com o diâmetro do intervalo próximo de zero.

Tabela 4.23: Grau de pertinência em cada cluster

Instância	**Cluster 1**	**Cluster 2**	**Cluster 3**	**Cluster 4**
1	$[0,003; 0,015]$	$[0,006; 0,037]$	$[0,786; 0,966]$	$[0,024; 0,169]$
2	$[0,036; 0,223]$	$[0,043; 0,267]$	$[0,032; 0,152]$	$[0,436; 0,884]$
3	$[0,588; 0,934]$	$[0,036; 0,263]$	$[0,009; 0,049]$	$[0,017; 0,156]$
4	$[0,810; 0,959]$	$[0,027; 0,134]$	$[0,004; 0,019]$	$[0,008; 0,045]$
5	$[0,255; 0,814]$	$[0,080; 0,519]$	$[0,018; 0,134]$	$[0,061; 0,343]$
6	$[0,766; 0,979]$	$[0,013; 0,164]$	$[0,002; 0,026]$	$[0,004; 0,060]$
7	$[0,645; 0,949]$	$[0,033; 0,269]$	$[0,005; 0,034]$	$[0,010; 0,083]$
8	$[0,000; 0,011]$	$[0,002; 0,028]$	$[0,779; 0,991]$	$[0,006; 0,191]$
9	$[0,667; 0,966]$	$[0,020; 0,236]$	$[0,003; 0,039]$	$[0,009; 0,079]$
10	$[0,005; 0,299]$	$[0,009; 0,349]$	$[0,036; 0,723]$	$[0,140; 0,944]$
11	$[0,002; 0,029]$	$[0,004; 0,066]$	$[0,392; 0,974]$	$[0,018; 0,569]$
12	$[0,196; 0,898]$	$[0,046; 0,574]$	$[0,012; 0,125]$	$[0,027; 0,525]$
13	$[0,768; 0,885]$	$[0,076; 0,160]$	$[0,012; 0,024]$	$[0,024; 0,055]$
14	$[0,021; 0,138]$	$[0,053; 0,386]$	$[0,038; 0,210]$	$[0,377; 0,880]$
15	$[0,008; 0,033]$	$[0,020; 0,094]$	$[0,425; 0,872]$	$[0,087; 0,514]$
16	$[0,756; 0,926]$	$[0,045; 0,163]$	$[0,009; 0,030]$	$[0,018; 0,060]$
17	$[0,005; 0,035]$	$[0,008; 0,067]$	$[0,018; 0,258]$	$[0,677; 0,965]$
18	$[0,537; 0,979]$	$[0,013; 0,381]$	$[0,002; 0,037]$	$[0,004; 0,102]$
19	$[0,066; 0,442]$	$[0,441; 0,905]$	$[0,009; 0,057]$	$[0,016; 0,131]$
20	$[0,108; 0,361]$	$[0,400; 0,771]$	$[0,023; 0,082]$	$[0,065; 0,296]$
21	$[0,024; 0,045]$	$[0,043; 0,084]$	$[0,675; 0,819]$	$[0,105; 0,215]$
22	$[0,000; 0,010]$	$[0,000; 0,022]$	$[0,894; 0,991]$	$[0,006; 0,077]$
23	$[0,016; 0,131]$	$[0,686; 0,954]$	$[0,009; 0,084]$	$[0,018; 0,129]$
24	$[0,425; 0,940]$	$[0,029; 0,347]$	$[0,008; 0,108]$	$[0,020; 0,254]$
25	$[0,011; 0,043]$	$[0,017; 0,072]$	$[0,115; 0,433]$	$[0,480; 0,853]$
26	$[0,009; 0,031]$	$[0,022; 0,086]$	$[0,386; 0,787]$	$[0,166; 0,556]$
27	$[0,008; 0,078]$	$[0,013; 0,131]$	$[0,014; 0,114]$	$[0,700; 0,964]$
28	$[0,032; 0,081]$	$[0,121; 0,320]$	$[0,129; 0,349]$	$[0,367; 0,676]$
29	$[0,003; 0,070]$	$[0,006; 0,122]$	$[0,021; 0,467]$	$[0,439; 0,968]$
30	$[0,624; 0,952]$	$[0,031; 0,280]$	$[0,005; 0,036]$	$[0,010; 0,096]$
31	$[0,004; 0,033]$	$[0,008; 0,073]$	$[0,658; 0,969]$	$[0,017; 0,265]$
32	$[0,076; 0,139]$	$[0,704; 0,829]$	$[0,036; 0,074]$	$[0,050; 0,102]$
33	$[0,046; 0,395]$	$[0,033; 0,319]$	$[0,047; 0,423]$	$[0,212; 0,855]$
34	$[0,005; 0,051]$	$[0,007; 0,077]$	$[0,010; 0,151]$	$[0,735; 0,978]$
35	$[0,007; 0,060]$	$[0,013; 0,114]$	$[0,451; 0,927]$	$[0,049; 0,453]$
36	$[0,000; 0,015]$	$[0,003; 0,039]$	$[0,634; 0,982]$	$[0,012; 0,332]$
37	$[0,009; 0,288]$	$[0,017; 0,423]$	$[0,065; 0,770]$	$[0,115; 0,891]$

Realizando um agrupamento crisp dos graus de pertinência de acordo com o ponto médio, a Tabela 4.24 mostra a classificação obtida. A sua convergência deu-se após 7 iterações, comportando-se mais rápido que o algoritmo IFCM.

Tabela 4.24: Partição crisp com 4 clusters obtido com o algoritmo Interval ckMeans.

Partição	**Interval ckMeans**
Cluster 1	Bahrain, Bombay, Cairo, Calcutta, Colombo, Dubai, Hong Kong, Kula Lumpur, Madras, Manila, New Delhi, Singapore
Cluster 2	Amsterdam, Copenhagen, Geneva, London, Moscow, Munich, Paris, Stockholm, Toronto, Viena, Zurich
Cluster 3	Mauritius, Mexico City, Nairobi, Sydney
Cluster 4	Athens, Frankfurt, Lisbon, Madrid, New York, Rome, San Francisco, Seoul, Tehran, Tokyo

A Tabela 4.25 mostra a quantidade de instâncias agrupadas em cada cluster. Como nos resultados dos algoritmos MSV e IFCM apresentados em (GURU; KIRANAGI; NAGABHUSHAN, 2004) e (de CARVALHO, 2007), respectivamente, a classificação da cidade Singapore foi ignorada por alguma razão, nós também a ignoramos para efeito de comparação embora tenha sido classificada no cluster 1.

Tabela 4.25: Quantidade de objetos agrupados entre os algoritmos

Cluster	**IFCM**	**MSV**	**Interval ckMeans**
1	7	14	11
2	12	20	11
3	8	1	4
4	9	1	10

O algoritmo Interval ckMeans classificou diversos dados em comuns se comparado com os algoritmos IFCM e MSV, a Tabela 4.26 mostra essa distribuição.

Tabela 4.26: Objetos classificados igualmente em relação aos algoritmos IFCM e MSV

Cluster	**IFCM**	**MSV**
1	4	11
2	10	10
3	1	1
4	9	1

Portanto, a Tabela 4.26 podemos concluir que há uma coincidência de classificações de 66,66% perante ao algoritmo IFCM e 63,88% perante ao algoritmo MSV. Por outro lado, apenas 21 cidades foram classificadas nos mesmos clusters pelos algoritmos IFCM e MSV, ou seja, o que resulta em uma coincidência de classificação de 58,33%.

Sobre o algoritmo Interval ckMeans, observe que o Cluster 1 teve 11 instâncias agrupadas de forma idêntica ao algoritmo MSV, a exceção foi a cidade de Sydney e com 4 cidades em comum na classificação com o algoritmo IFCM. O Cluster 2 com o algoritmo Interval ckMeans

teve todos os dados classificados idênticos ao algoritmo MSV e com a exceção de um dado ao algoritmo IFCM. O Cluster 3 foi o que apresentou maior diferença entre os algoritmos comparados, com apenas uma instância em comum. E o Cluster 4 mostrou uma instância em comum com o algoritmo MSV e 8 em comum com o algoritmo IFCM.

Tabela 4.17: Mostra o mínimo, a média, o máximo, o desvio padrão, a moda e a mediana do índice externo *Adjusted Rand Index* na base Iris.

		K-Means	FCM				ckMeans			
			1,25	1,50	2,00	2,50	1,25	1,50	2,00	2,50
DE	Mínimo	0,8737	0,7160	0,8737	0,8797	0.8859	0,7142	0,7142	0,7142	0,7142
	Média	0,8737	0,8721	0,8737	0,8797	0.8859	0,8567	0,8570	0,8523	0,8521
	Máximo	0,8737	0,8737	0,8737	0,8797	0.8859	0,8797	0,8797	0,8797	0,8797
	Desv.Pad	≈ 0	0,0157	≈ 0	≈ 0	≈ 0	0,0499	0,0497	0,0554	0,0553
	Moda	0,8737	0,8737	0,8737	0,8797	0.8859	0,8737	0,8737	0,8737	0,8737
	Mediana	0,8737	0,8737	0,8737	0,8797	0.8859	0,8737	0,8737	0,8737	0,8737
DNM	Mínimo	0,8737	0,8797	0,8797	0,8987	0,9055	0,7142	0,7153	0,7128	0,7139
	Média	0,8767	0,8797	0,8797	0,8987	0,9055	0,8570	0,8721	0,8533	0,8505
	Máximo	0,8797	0,8797	0,8797	0,8987	0,9055	0,8797	0,8737	0,8797	0,8797
	Desv.Pad	0,0042	≈ 0	≈ 0	≈ 0	≈ 0	0,0499	0,0158	0,0536	0,0564
	Moda	0,8772	0,8797	0,8797	0,8987	0,9055	0,8737	0,8737	0,8737	0,8737
	Mediana	0,8767	0,8797	0,8797	0,8987	0,9055	0,8737	0,8737	0,8737	0,8737
DMN	Mínimo	0,8737	0,8797	0,9055	0,7121	0,7047	0,7142	0,7142	0,7142	0,7142
	Média	0,8767	0,8797	0,9055	0,7552	0,7498	0,8598	0,8571	0,8645	0,8551
	Máximo	0,8797	0,8797	0,9055	0,7849	0,7964	0,8797	0,9123	0,8797	0,8797
	Desv.Pad	0,0042	≈ 0	≈ 0	0,011	0,011	0,0455	0,0501	0,0377	0,0516
	Moda	0,8772	0,8797	0,9055	0,7582	0,7493	0,8737	0,8737	0,8737	0,8737
	Mediana	0,8767	0,8797	0,9055	0,7566	0,7493	0,8737	0,8737	0,7982	0,8737

Tabela 4.18: Mostra o mínimo, a média, o máximo, o desvio padrão, a moda e a mediana do índice externo *Adjusted Rand Index* na base Sonar.

		K-Means	FCM				ckMeans			
			1,25	1,50	2,00	2,50	1,25	1,50	2,00	2,50
DE	Mínimo	0,4992	0,5054	0,5032	0,4977	0,4976	0,4992	0,5013	0,5013	0,5013
	Média	0,5031	0,5054	0,5037	0,5034	0,5040	0,5027	0,5031	0,5028	0,5030
	Máximo	0,5066	0,5054	0,5042	0,5161	0,5221	0,5054	0,5054	0,5054	0,5054
	Desv.Pad	0,0022	≈ 0	0,0005	0,0028	0,0051	0,0019	0,0019	0,0019	0,0019
	Moda	0,5054	0,5054	0,5042	0,5032	0,5022	0,5022	0,5013	0,5013	0,5013
	Mediana	0,5056	0,5054	0,5042	0,5032	0,5043	0,5022	0,5033	0,5033	0,5013
DNM	Mínimo	0,4998	0,4999	0,5013	0,4980	0,4975	0,4980	0,4998	0,5013	0,4998
	Média	0,5024	0,5005	0,5015	0,5046	0,5036	0,5028	0,5028	0,5025	0,5027
	Máximo	0,5066	0,5013	0,5066	0,5243	0,5243	0,5054	0,5054	0,5054	0,5054
	Desv.Pad	0,0017	0,0002	0,0007	0,0063	0,0057	0,0020	0,0020	0,0017	0,0019
	Moda	0,5013	0,5005	0,5013	0,5013	0,5022	0,5013	0,5013	0,5013	0,5013
	Mediana	0,5033	0,5005	0,5013	0,5087	0,5009	0,5033	0,5013	0,5033	0,5013
DMN	Mínimo	0,4983	0,4998	0,4977	0,4976	0,4976	0,5005	0,4998	0,4992	0,4998
	Média	0,5026	0,5021	0,5030	0,5038	0,5038	0,5030	0,5054	0,5029	0,5023
	Máximo	0,5066	0,5161	0,5200	0,5393	0,5200	0,5080	0,5054	0,5066	0,5054
	Desv.Pad	0,0021	0,0028	0,0042	0,0075	0,0053	0,0020	0,0020	0,0020	0,0016
	Moda	0,5013	0,5013	0,5004	0,5017	0,5043	0,5013	0,5054	0,5013	0,5013
	Mediana	0,5033	0,5017	0,5004	0,5017	0,5043	0,5013	0,5033	0,5033	0,5054

Tabela 4.19: Mostra o mínimo, a média, o máximo, o desvio padrão, a moda e a mediana do índice externo *Adjusted Rand Index* na base Vogais.

		K-Means	FCM				ckMeans			
			1,25	1,50	2,00	2,50	1,25	1,50	2,00	2,50
DE	Mínimo	0,7078	0,7402	0,7592	0,6086	0,5154	0,7080	0,7086	0,7340	0,7078
	Média	0,7442	0,7403	0,7592	0,6987	0,5294	0,7471	0,7453	0,7398	0,7473
	Máximo	0,7818	0,7404	0,7592	0,6091	0,5775	0,7799	0,7863	0,7631	0,7797
	Desv.Pad	0,0208	0	0	0,0001	0,0188	0,0169	0,0189	0,0072	0,0192
	Mediana	0,7303	0,7404	0,7592	0,6086	0,5154	0,7576	0,7086	0,7398	0,7340
DNM	Mínimo	0,7078	0,6496	0,6128	0,6364	0,6287	0,7086	0,7340	0,7078	0,7029
	Média	0,7512	0,6916	0,6789	0,6838	0,6915	0,7484	0,7534	0,7436	0,7423
	Máximo	0,7872	0,7294	0,7244	0,7157	0,7464	0,7797	0,7871	0,7640	0,7642
	Desv.Pad	0,0196	0,0265	0,0345	0,0256	0,0361	0,0207	0,0165	0,0146	0,0180
	Mediana	0,7358	0	0,6485	0,7119	0,6914	0,7424	0,7672	0,7336	0,7425
DMN	Mínimo	0,7154	0,6321	0,6311	0,6593	0,6079	0,7379	0,7078	0,7078	0,7025
	Média	0,7461	0,7031	0,6807	0,6925	0,6930	0,7472	0,7496	0,7453	0,7582
	Máximo	0,7830	0,7365	0,7294	0,7359	0,7367	0,7797	0,7674	0,7872	0,7870
	Desv.Pad	0,„158	0,0300	0,0339	0,0241	0,0434	0,0122	0,0162	0,0173	0,0228
	Mediana	0,7399	0,7334	0,7174	0,6695	0,7112	0,7401	0,7401	0,7574	0,7870

Tabela 4.20: Mínimo e máximo das temperaturas das cidades medidos em graus celsius

Cidade	Janeiro	Fevereiro	Março	Abril	Maio	Junho	Julho	Agosto	Setembro	Outubro	Novembro	Dezembro
Amsterdam	[−4;4]	[−5;3]	[2;12]	[5;15]	[7;17]	[10;20]	[10;20]	[12;23]	[10;20]	[5;15]	[1;10]	[−1;4]
Athens	[6;12]	[6;12]	[8;16]	[11;19]	[16;25]	[19;29]	[22;32]	[22;32]	[19;28]	[16;23]	[11;18]	[8;14]
Bahrain	[13;19]	[14;19]	[17;23]	[21;27]	[25;32]	[28;34]	[29;36]	[30;36]	[28;34]	[24;31]	[20;26]	[15;21]
Bombay	[19;28]	[19;28]	[22;30]	[24;32]	[27;33]	[26;32]	[25;30]	[25;30]	[24;30]	[24;32]	[23;32]	[20;30]
Cairo	[8;20]	[9;22]	[11;25]	[14;29]	[17;33]	[20;35]	[22;36]	[22;35]	[20;33]	[18;31]	[14;26]	[10;20]
Calcutta	[13;27]	[16;29]	[21;34]	[24;36]	[26;36]	[26;33]	[26;32]	[26;32]	[26;32]	[24;32]	[18;29]	[13;26]
Colombo	[22;30]	[22;30]	[23;31]	[24;31]	[25;31]	[25;30]	[25;29]	[25;29]	[25;30]	[24;29]	[23;29]	[22;30]
Copenhagen	[−2;2]	[−3;2]	[−1;5]	[3;10]	[8;16]	[11;20]	[14;22]	[14;21]	[11;18]	[7;12]	[3;7]	[1;4]
Dubai	[13;23]	[14;24]	[17;28]	[19;31]	[22;34]	[25;36]	[28;39]	[28;39]	[25;37]	[21;34]	[17;30]	[14;26]
Frankfurt	[−10;9]	[−8;10]	[−4;17]	[0;24]	[3;27]	[7;30]	[8;32]	[8;31]	[5;27]	[0;22]	[−3;14]	[−8;10]
Geneva	[−3;5]	[−6;6]	[3;9]	[7;13]	[10;17]	[15;17]	[16;24]	[16;23]	[11;19]	[6;13]	[3;8]	[−2;6]
Hong Kong	[13;17]	[12;16]	[15;19]	[19;23]	[22;27]	[25;29]	[25;30]	[25;30]	[25;29]	[22;27]	[18;23]	[14;19]
Kula Lumpur	[22;31]	[23;32]	[23;33]	[23;33]	[23;32]	[23;32]	[23;31]	[23;32]	[23;32]	[23;31]	[23;31]	[23;31]
Lisbon	[8;13]	[8;14]	[9;16]	[11;18]	[13;21]	[16;24]	[17;26]	[18;27]	[17;24]	[14;21]	[11;17]	[8;14]
London	[2;6]	[2;7]	[3;10]	[5;13]	[8;17]	[11;20]	[13;22]	[13;21]	[11;19]	[8;14]	[5;10]	[3;7]
Madras	[20;30]	[20;31]	[22;33]	[26;35]	[28;39]	[27;38]	[26;36]	[26;35]	[25;34]	[24;32]	[22;30]	[21;29]
Madrid	[1;9]	[1;12]	[3;16]	[6;19]	[9;24]	[13;29]	[16;34]	[16;33]	[13;28]	[8;20]	[4;14]	[1;9]
Manila	[21;27]	[22;27]	[24;29]	[24;31]	[25;31]	[25;31]	[23;29]	[24;28]	[25;28]	[24;29]	[22;28]	[22;27]
Mauritius	[22;28]	[22;29]	[22;29]	[21;28]	[19;25]	[18;24]	[17;23]	[17;23]	[17;24]	[18;25]	[19;27]	[21;28]
Mexico City	[6;22]	[15;23]	[17;25]	[18;27]	[18;27]	[18;27]	[18;27]	[18;26]	[18;26]	[16;25]	[14;25]	[8;23]
Moscow	[−13;−6]	[−12;−5]	[−8;0]	[0;8]	[7;18]	[11;23]	[13;24]	[11;22]	[6;16]	[1;8]	[−5;0]	[−11;−5]
Munich	[−6;1]	[−5;3]	[−2;9]	[3;14]	[7;18]	[10;21]	[12;23]	[11;23]	[8;20]	[4;13]	[0;7]	[−4;2]
Nairobi	[12;25]	[13;26]	[14;25]	[14;24]	[13;22]	[12;21]	[11;21]	[11;21]	[11;24]	[13;24]	[13;23]	[13;23]
New Delhi	[6;21]	[10;24]	[14;29]	[20;36]	[26;40]	[28;39]	[27;35]	[26;34]	[24;34]	[18;34]	[11;28]	[7;23]
New York	[−2;4]	[−3;4]	[1;9]	[6;15]	[12;22]	[17;27]	[21;29]	[20;28]	[16;24]	[11;19]	[5;12]	[−2;6]
Paris	[1;7]	[1;7]	[2;12]	[5;16]	[8;19]	[12;22]	[14;24]	[13;24]	[11;21]	[7;16]	[4;10]	[1;6]
Rome	[4;11]	[5;13]	[7;16]	[10;19]	[13;23]	[17;28]	[20;31]	[20;31]	[17;27]	[13;21]	[9;16]	[5;12]
San Francisco	[6;13]	[6;14]	[7;17]	[8;18]	[10;19]	[11;21]	[12;22]	[12;22]	[12;23]	[11;22]	[8;18]	[6;14]
Seoul	[0;7]	[1;6]	[1;8]	[6;16]	[12;22]	[16;25]	[18;31]	[16;30]	[9;28]	[3;24]	[7;19]	[1;8]
Singapore	[23;30]	[23;30]	[24;31]	[24;31]	[24;30]	[25;30]	[25;30]	[25;30]	[24;30]	[24;30]	[24;30]	[23;30]
Stockholm	[−9;−5]	[−9;−6]	[−4;2]	[1;8]	[6;15]	[11;19]	[14;22]	[13;20]	[9;15]	[5;9]	[1;4]	[−2;2]
Sydney	[20;30]	[20;30]	[18;26]	[16;23]	[12;20]	[5;17]	[8;16]	[9;17]	[11;20]	[13;22]	[16;26]	[20;30]
Tehran	[0;5]	[5;8]	[10;15]	[15;18]	[20;25]	[28;30]	[36;38]	[38;40]	[29;30]	[18;20]	[9;12]	[−5;0]
Tokyo	[0;9]	[0;10]	[3;13]	[9;18]	[14;23]	[18;25]	[22;29]	[23;31]	[20;27]	[13;21]	[8;16]	[2;12]
Toronto	[−8;−1]	[−8;−1]	[−4;4]	[−2;11]	[−8;18]	[13;24]	[16;27]	[16;26]	[12;22]	[6;14]	[−1;17]	[−5;1]
Vienna	[−2;1]	[−1;3]	[1;8]	[5;14]	[10;19]	[13;22]	[15;24]	[14;23]	[11;19]	[7;13]	[2;7]	[1;3]
Zurich	[−11;9]	[−8;15]	[−7;18]	[−1;21]	[2;27]	[6;30]	[10;31]	[8;25]	[5;23]	[3;22]	[0;19]	[−11;8]

5 *Considerações Finais*

A análise de clusters não é um processo realizado em apenas uma execução. Em muitas circunstâncias, é necessário uma série de tentativas e repetições. Além disso, não há um critério universal e efetivo para guiar a seleção de atributos e de algoritmos de agrupamentos. Critérios de validação provêm impressões sobre a qualidade dos clusters, mas como escolher este mesmo critério é ainda um problema que requer mais esforços(CAVALCANTI, 2006).

Esta tese apresentou dois estudos: (i) uma para o agrupamento de dados pontuais, mostrando um novo método para calcular os centros dos clusters, o algoritmo ckMeans. Esse algoritmo reduziu o tempo de processamento e o número de iterações na classificação de dados. O algoritmo ckMeans fornece uma aceleração perante a aplicação FCM tradicional e, (ii) o agrupamento de dados simbólicos, mostrando um estudo das principais operações e funções especiais da matemática intervalar e mostrou os procedimentos de análise de clusters.

Sobre os dados pontuais podemos concluir:

1. Compreende-se que a expressão para o cálculo da função objetivo e os centros dos cluster no algoritmo FCM é uma derivação matemática de uma função objetivo. Porém, não se tem essa preocupação no algoritmo ckMeans, os valores de J (função objetivo) são um pouco menores no algoritmo ckMeans do que no algoritmo FCM, e portanto, na prática o objetivo de minimizar J também pode ser alcançado pelo algoritmo ckMeans;

2. Observe que a condição de parada fornecido por *epsilon* quanto menor for, maior é o número de iterações no algoritmo FCM. Entretanto, no algoritmo ckMeans isso não ocorre, como o *epsilon* é utilizado para calcular a diferença do valor de J na iteração atual com a iteração anterior, no algoritmo ckMeans a tendência é que essa diferença seja zero;

3. Os experimentos nas bases simuladas mostram que a classificação do grau de pertinência com o algoritmo ckMeans em relação ao cluster é similar ou até melhor do que com o algoritmo FCM;

4. Em algumas base de dados simuladas citadas no capítulo resultados, utilizar outras distân-

cias como a distância Não-Métrica e Métrica-Normalizada em vez da distância Euclidiana (mais usual) a quantidade de dados classificados corretamente não é alterado ou é até pior que se comparado com a distância Euclidiana, em outros casos, por exemplo o uso da distância Métrica-Normalizada na base de dados Sonar é melhor empregada;

5. A utilização das distâncias Não-Métrica e Métrica-Normalizada aumentaram o tempo de processamento dos algoritmos K-Means, FCM e ckMeans igualando ou piorando os resultados se comparados com os algoritmos usando a distância Euclidiana.

E com respeito aos dados intervalares podemos dizer que:

1. O contexto deste trabalho está inserido na abordagem simbólica da análise de dados (SDA) relacionada com métodos para a extração de conhecimentos em grandes bases de dados. O principal objetivo da SDA é desenvolver métodos para o tratamento de dados mais complexos como intervalos. Neste contexto, vários pesquisadores (de CARVALHO, 2007), (GURU; KIRANAGI; NAGABHUSHAN, 2004), (ZHANG; HU; LIU, 2007), (BOCK, 2000) e (SATO; LAKHMI, 2006) vêm trabalhando no sentido de estabelecer e aplicar metodologias no agrupamento de dados intervalares. Este trabalho também pretende ser um aporte para essa área. Uma das propriedades da abordagem proposta é respeitar o princípio da corretude, no sentido de (HICKEY; JU; EMDEN, 2001), ou seja, se considerasse qualquer valor pontual entre os seus respectivos valores (intervalares) como dado de entrada e grau de pertinência, usando o algoritmo pontual FCM, o agrupamento resultante estaria contido no intervalo apresentado pelo algoritmo Interval ckMeans;

2. Outros algoritmos de agrupamento para a entrada de dados intervalares foram analisados. Então a vantagem é que o algoritmo Interval ckMeans considera graus de pertinências intervalares propiciando conhecer ainda mais a imprecisão nos dados de entrada. O grande trunfo deste algoritmo é sempre manter os dados de entrada e operações com intervalos e quando necessário calcular a distância de cada ponto ao centro de cada cluster, usar uma métrica intervalar em vez de usar uma métrica pontual como a distância Euclidiana;

3. O algoritmo Interval ckMeans proposto nesta tese, houve a aplicação de duas técnicas: a matemática intervalar e a teoria dos conjuntos fuzzy. Desta forma, é possível tratar os dados de entrada imprecisos em resultados com funções de pertinências intervalares.

Na implementação computacional do algoritmo proposto para as operações intervalares, usou-se a biblioteca C-XSC no qual se mostra adequada para realizar as operações intervalares.

5.1 Trabalhos Futuros

A seguir as principais etapas a serem alcançadas:

- Comparar os resultados de outras extensões do FCM com o algoritmo ckMeans;
- Modificar a forma de calcular os centros dos clusters em outras variantes do algoritmo FCM e comparar o seu desempenho;
- Trabalhar com outras bases de dados simbólicos que já estejam validadas;
- Verificar a propriedade de corretude do algoritmo Interval ckMeans com respeito ao algoritmo ckMeans;
- Utilizar imagens médicas no processo de agrupamento com os algoritmos K-Means, FCM e ckMeans;
- Manipular imagens intervalares com o algoritmo Interval ckMeans.

Como uma contribuição futura esta tese de doutorado deixará a possibilidade de usar a mesma ideia que se fez com algoritmo ckMeans (de transformar o algoritmo pontual para um algoritmo de agrupamento de dados simbólicos), podendo ser aplicada para agrupar dados granulares (computação granular) (GOMIDE, 2006), (BARGIELA; PEDRYCZ, 2002) e (PEDRYCZ; SKOWRON; KREINOVICH, 2008).

5.2 Publicações Obtidas

Ao longo dessa pesquisa, obteve-se algumas publicações, como descritas abaixo:

- de VARGAS, R. ; BEDREGAL, B. ; PALMEIRA, E. . A Comparison between K-Means, FCM and ckMeans Algorithms. In: Simone André da Costa Cavalheiro; Luciana Foss; Marilton Sanchotene de Aguiar;Graçaliz Pereira Dimuro; Antônio Carlos da Rocha Costa. (Org.). Post-Proceedings of the Workshop-School on Theoretical Computer Science. Los Alamitos: IEEE, 2011, v. 1, p. 32-38;
- de VARGAS, R. ; BEDREGAL, B.: Interval ckMeans: An Algorithm for Clustering Symbolic Data. In: Proc. Conf. North American Fuzzy Information Processing Society (NAFIPS 2011), El Paso, USA (2011);

- Vargas, R., Bedregal, B.: A Comparative Study Between fuzzy c-means and ckMeans Algorithms. In: Proc. Conf. North American Fuzzy Information Processing Society (NAFIPS 2010), Toronto, Canada (2010);

- Vargas, R., Bedregal, B.: Uma Nova Forma de Calcular o Centro dos Clusters no Algoritmo Fuzzy C-Means. In: Proceedings of CNMAC 2010 (33th Brazilian Conference on Applied and Computational Math), SBMAC (Brazilian Society of Applied and Computational Math), Águas de Lindóia, Brazil (2010);

- Vargas, R., Bedregal, B., Oliveira Filho, I.: Agrupamento de Dados Intervalares com o Algoritmo IFCM. In: Proceedings of CISAISI 2009 (13th Congresso Internacional Sudamericano de Ingeneariía de Sistemas e Informática), Arica, Chile (2009);

- Vargas, R., Bedregal, B.: Uma Extensão Intervalar do Algoritmo Fuzzy C-Means. In: Proceedings of CNMAC 2009 (32th Brazilian Conference on Applied and Computational Math), SBMAC (Brazilian Society of Applied and Computational Math), Cuiabá, Brazil (2009).

Essas e outras publicações estão disponíveis em `http://rogerio.in`, na opção *PUBLICATIONS* do menu superior.

Submeteu-se em setembro de 2011 um artigo para o periódico da Information Sciences, este encontra-se em processo de avaliação por parte da revista. A descrição do artigo submtido é mostrado abaixo:
R. de Vargas, B. Bedregal, and E. Palmeira, “ckmeans: A new variant of fuzzy c-means algorithm”, Information Sciences, 2011, submitted

As perpectivas a nível de publicações com frutos desta tese são as seguintes:

- Submeter a um períodico relevante a versão intervalar do algoritmo ckMeans, ou seja, o algoritmo Interval ckMeans;

- Assim que o algoritmo ckMeans manipular imagens médicas, submeter os resultados a congressos internacionais relevantes;

- Assim que implementado uma outra variante do algoritmo FCM, comparar os resultados e submeter a congressos os resultados;

- Publicar em algum periódico relevante a versão pontual e intervalar do algoritmo ckMeans, utilizar a mesma base de dados comparando os resultandos entre a versão pontual e intervalar.

Referências

AGRAWAL, R. Data Mining: The Quest Perspective. *Australian Computer Science Comm. — Proc. 7th Australian Database Conf., ADC*, v. 18, n. 2, p. 119–120, 1996.

ALTROCK, C. *Fuzzy Logic and NeuroFuzzy Applications in Business and Finance*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1997.

ALVES, V. *Um Algoritmo Evolutivo Rápido para Agrupamento de Dados*. Dissertação (Mestrado) — Universidade Católica de Santos, Santos, Brasil, 2007.

BARGIELA, A.; PEDRYCZ, W. *Granular Computing. An Introduction*. Boston, MA, USA: Kluwer Academic Publishers, 2002. (The International Series in Engineering and Computer Science, Vol. 717).

BEZDEK, J. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Norwell, MA, USA: Kluwer Academic Publishers, 1981.

BEZDEK, J.; EHRLICH, R.; FULL, W. FCM: The fuzzy c-means Clustering Algorithm. Computers & Geosciences, v. 10, 1984.

BEZDEK, J. et al. *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing (The Handbooks of Fuzzy Sets)*. Secaucus, USA: Springer-Verlag New York, Inc., 2005.

BISHOP, C. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006. ISBN 0387310738.

BOCK, H. *Analysis of Symbolic Data: Exploratory Methods for Extracting Statistical Information from Complex Data*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2000.

BOLEY, D. et al. *Partitioning-Based Clustering for Web Document Categorization. Decision Support Systems*. 1999.

BURKILL, J. Functions of Intervals. *Proc. London Math. Soc.*, v. 22, p. 275–336, 1924.

CANNON, R. L.; DAVE, J. V.; BEZDEK, J. C. Efficient implementation of the fuzzy c-means clustering algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.*, IEEE Computer Society, Washington, DC, USA, v. 8, n. 2, p. 248–255, 1986.

CAVALCANTI, J. *Clusterização Baseada em Algoritmos Fuzzy*. Dissertação (Mestrado) — Universidade Federal de Pernambuco, Recife, Brasil, 2006.

CELENK, M. A Color Clustering Technique for Image Segmentation. *Comput. Vision Graph. Image Process.*, Academic Press Professional, Inc., San Diego, USA, v. 52, n. 2, p. 145–170, 1990.

COX, E. The fuzzy systems handbook: A practitioner's guide to building, using, and maintaining fuzzy systems. *Academic Press Professional Inc.*, 1994.

COX, E. *Fuzzy modeling and genetic algorithms for data mining and exploration*. [S.l.]: Elsevier/Morgan Kaufmann, 2005. Hardcover. (Morgan Kaufmann series in data management systems).

de CARVALHO, F. Fuzzy c-means clustering methods for symbolic interval data. *Pattern Recogn. Lett.*, v. 28, n. 4, p. 423–437, 2007.

DEGRUIJTER, J.; MCBRATNEY, A. A modified fuzzy k-means for predictive classification. In: *Classification and Related Methods of Data Analysis*. Amsterdam: H.H. Bock, ed., Elsevier Science, 1988. p. 97–104.

DEMING, Z.; QIANG, L. An Extended Compensated Fuzzy C-Means Algorithm and Its Application in Optimizing the Acrylonitrile Reactor Parameters. In: *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on*. [S.l.: s.n.], 2006. v. 2, p. 5872–5876.

DUNN, J. C. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, Taylor & Francis, v. 3, n. 3, p. 32–57, 1973.

DWYER, P. Computation with Approximate Numbers. In: DWYER, P. (Ed.). *Linear Computations*. New York: Wiley & Sons Inc, 1951. p. 11–35.

EKLUND, P. Fuzzy Logic in Northern Europe: Industrial Applications and Software Developments. In: *Fuzzy Systems, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the Third IEEE Conference on*. [S.l.: s.n.], 1994. p. 712–715 vol.2.

ESCHRICH, S. et al. Fast accurate fuzzy clustering through data. *IEEE Transactions on Fuzzy Systems*, v. 11, p. 262–270, 2003.

FAYYAD, U. *Advances in Knowledge Discovery and Data Mining*. [S.l.]: AAAI/MIT Press, 1996.

FISHER, R. The use of multiple measurements in taxonomic problems. *Annals Eugen.*, v. 7, p. 179–188, 1936.

FOWLKES, E. B.; MALLOWS, C. L. A Method for Comparing Two Hierarchical Clusterings. *Journal of the American Statistical Association*, American Statistical Association, v. 78, n. 383, p. 553–569, 1983.

FOX, L. *An introduction to numerical linear algebra*. United Kingdom: Oxford University Press, 1974. 340 p. (Monographs on numerical analysis).

FRANK, A.; ASUNCION, A. *UCI Machine Learning Repository*. 2010. Disponível em: <http://archive.ics.uci.edu/ml>.

GIARRATANO, J.; RILEY, G. *Expert Systems: Principles and Programming*. 2nd. ed. Boston: PWS Publishing Company, 1994.

GOLUB, T. R. et al. Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. *Science*, Whitehead Institute/Massachusetts Institute of Technology Center for Genome Research, Cambridge, USA., v. 286, p. 531–537, 1999.

GOMIDE, F. Computação Granular - Computação Avança em Projetos de Sistemas Inteligentes. *Revista Fonte*, n. 4, p. 47–48, 2006.

GORMAN, R. P.; SEJNOWSKI, T. J. Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets. *Neural Networks*, v. 1, n. 1, p. 75–89, 1988.

GOTTGTROY, M. *Aplicação de Técnicas de Engenharia do Conhecimento na Análise do Risco em Sistemas Estruturais*. Tese (Doutorado) — Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil, 1996.

GURU, D. S.; KIRANAGI, B. B.; NAGABHUSHAN, P. Multivalued Type Proximity Measure and Concept of Mutual Similarity Value Useful for Clustering Symbolic Patterns. *Pattern Recogn. Lett.*, Elsevier Science Inc., New York, NY, USA, v. 25, n. 10, p. 1203–1213, 2004.

H. KIERS et al. Data analysis, classification, and related methods. In: ____. [S.l.]: Springer, 2000. (Studies in classification, data analysis, and knowledge organization).

HALKIDI, M.; BATISTAKIS, Y.; VAZIRGIANNIS, M. On clustering validation techniques. *J. Intell. Inf. Syst.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 17, n. 2-3, p. 107–145, 2001.

HAMMER, R.; HOCKS, M.; RATZ, D. C++ Toolbox for Verified Computing I - Basic Numerical Problems. *Springer-Verlag*, Berlin, Heidelberg, 1995.

HAN, J. et al. Intelligent Query Answering by Knowledge Discovery Techniques. *IEEE Transactions on Knowledge and Data Engineering*, v. 8, p. 373–390, 1996.

HICKEY, T.; JU, Q.; EMDEN, M. Interval Arithmetic: from Principles to Implementation. *Journal of the ACM*, v. 48, n. 5, p. 1038–1068, 2001.

HOFSCHUSTER, W.; KRÄMER, W. C-xsc 2.0: A c++ library for extended scientific computing. In: *Dagstuhl Seminars*. [S.l.]: Springer, 2003. p. 15–35.

HÖLBIG, C. *Ambiente de Alto Desempenho com Alta Exatidão para a Resolução de Problemas*. Tese (Doutorado) — Universidade Federal do Rio Grande do Sul, Porto Alegre, Brasil, 2005.

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

JACCARD, P. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, v. 37, p. 547–579, 1901.

JAIN, A.; DUBES, R. *Algorithms for Clustering Data*. [S.l.]: Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1988.

JIANG, D.; TANG, C.; ZHANG, A. Cluster Analysis for Gene Expression Data: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, v. 16, p. 1370–1386, 2004.

JIN, Y.; WANG, L. (Ed.). *Fuzzy Systems in Bioinformatics and Computational Biology*. [S.l.]: Springer, 2009. (Studies in Fuzziness and Soft Computing, v. 242).

KULISCH, U. *Advanced Arithmetic for the Digital Computer: Design of Arithmetic Units*. Verlag: Softcover, 2002. xii–141 p.

KULISCH, U.; MIRANKER, W. *Computer Arithmetic in Theory and Practice*. Orlando, USA: Academic Press, Inc., 1981.

LI, C.; BECERRA, V.; DENG, J. Extension of fuzzy c-means algorithm. In: *Cybernetics and Intelligent Systems, 2004 IEEE Conference on*. [S.l.: s.n.], 2004. v. 1, p. 405–409.

LLETÍ, R. et al. Selecting Variables for K-Means Cluster Analysis by Using a Genetic Algorithm that Optimises the Silhouettes. *Analytica Chimica Acta*, v. 515, n. 1, p. 87–100, 2004.

LLOYD, S. P. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*, v. 28, p. 129–137, 1982.

MACQUEEN, J. Some methods for classification and analysis of multivariate observations. In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. Berkeley, CA: University of California Press, 1967. p. 281–297.

MARTINO, F. D.; LOIA, V.; SESSA, S. Extended fuzzy c-means clustering in gis environment for hot spot events. In: *Proceedings of the 11th international conference, KES 2007 and XVII Italian workshop on neural networks conference on Knowledge-based intelligent information and engineering systems: Part I*. Berlin, Heidelberg: Springer-Verlag, 2007. (KES'07/WIRN'07), p. 101–107.

MITCHELL, T. *Machine Learning (Mcgraw-Hill International Edit)*. 1. ed. New York, NY, USA: McGraw-Hill Education (ISE Editions), 1997.

MOORE, R. *Automatic Error Analysis in Digital Computation*. Sunnyvale, USA, 1959.

MOORE, R. *Interval Analysis*. EngCliffs: PH, 1966.

NASCIMENTO, S.; MIRKIN, B.; F., M.-P. A fuzzy clustering model of data and fuzzy c-means. In: *The ninth IEEE International Conference on Fuzzy Systems: Soft Computing in the Information Age (FUZZ-IEEE 2000), IEEE Neural Networks Council*. San Antonio, USA,: [s.n.], 2000. p. 302–307.

NIERADKA, G.; BUTKIEWICZ, B. A method for automatic membership function estimation based on fuzzy measures. In: *Proceedings of the 12th international Fuzzy Systems Association world congress on Foundations of Fuzzy Logic and Soft Computing*. Berlin, Heidelberg: Springer-Verlag, 2007. (IFSA '07), p. 451–460.

OLIVEIRA, R.; DIVERIO, T.; CLAUDIO, D. *Fundamentos da Matemática Intervalar*. Instituto de Informática da UFRGS, Porto Alegre, Brasil: Editora Sagra Luzzato, 2001. xi–90 p.

PEDRYCZ, W.; GOMIDE, F. *Fuzzy Systems Engineering: Toward Human-Centric Computing*. [S.l.]: Wiley-IEEE Press, 2007.

PEDRYCZ, W.; SKOWRON, A.; KREINOVICH, V. *Handbook of Granular Computing*. New York, NY, USA: Wiley-Interscience, 2008.

PUNTAR, S. *Métodos e Visualização de Agrupamentos de Dados*. Dissertação (Mestrado) — Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil, 2003.

RAND, W. Objective Criteria for the Evaluation of Clustering Methods. *Journal of the American Statistical Association*, American Statistical Association, v. 66, n. 336, p. 846–850, 1971.

RUDIN, W. *Principles of mathematical analysis*. Third. New York: McGraw-Hill Book Co., 1976. x+342 p. International Series in Pure and Applied Mathematics.

RUMP, S. *Fast and Parallel Interval Arithmetic*. [S.l.]: BIT, 1999. 539–560 p.

RUSS, E.; SIMPSON, P.; DOBBINS, R. *Computational Intelligence PC Tools*. San Diego, CA, USA: Academic Press Professional, Inc., 1996.

SATO, M.; LAKHMI, J. *Innovations in Fuzzy Clustering: Theory and Applications (Studies in Fuzziness and Soft Computing)*. Berlin, Heidelberg: Springer-Verlag, 2006.

STEINHAUS, H. Sur la Division des Corps Materiels en Parties. *Bull. Acad. Pol. Sci., Cl. III*, v. 4, p. 801–804, 1957.

SUNAGA, T. Theory of an Interval Algebra and its Application to Numerical Analysis. *RAAG Memoirs*, v. 2, p. 29–46, 547–564, 1958.

TAN, P.; STEINBACH, M.; KUMAR, V. *Introduction to data mining*. 1. ed. [S.l.]: Pearson Addison Wesley, 2006. (Pearson International Edition).

TEKNOMO, K. *K-Means Clustering Tutorials*. 2010. `http://people.revoledu.com/kardi/tutorial/kMean/`, Acessado 22/Maio/2010.

TRINDADE, R. *Uma Fundamentação Matemática para Processamento Digital de Sinais Intervalares*. Tese (Doutorado) — Universidade Federal de Rio Grande do Norte, Natal, Brasil, 2009.

VARGAS, R. de. *Técnicas Matemático-Computacionais para o Tratamento de Incertezas Aplicadas ao Problema do Fluxo de Potência em Sistemas de Transmissão de Energia Elétrica*. Dissertação (Mestrado) — Universidade Católica de Pelotas, Pelotas, Brasil, 2008.

VILLANUEVA, F. Z. W. ÍÃndices de validação de agrupamentos. In: CAMPINAS. *I Encontro dos Alunos e Docentes do DCA (EADCA)*. [S.l.], 2008.

WARMUS, M. Calculus of Approximations. *Bull. Acad. Polon. Sci., Cl. III*, IV, n. 5, p. 253–259, 1956.

WIENER, N. A Contribution to the Theory of Relative Position. *Cambr. Phil. Soc. Proc.*, v. 17, p. 441–449, 1914.

WIENER, N. *A New Theory of Measurement: A Study in the Logic of Mathematics.* [S.l.]: Lond. M. S. Proc. (2) 19, 181-205 , 1920.

WU, K.-L.; YANG, M.-S. Alternative c-means clustering algorithms. *Pattern Recognition*, v. 35, n. 10, p. 2267–2278, 2002.

YEUNG, K.; RUZZO, W. Principal component analysis for clustering gene expression data. *Bioinformatics*, v. 17, n. 9, p. 763–774, 2001.

YOUNG, R. The Algebra of Many-Valued Quantities. *Math. Ann.*, v. 104, p. 260–290, 1931.

ZADEH, L. Fuzzy Sets. *Inf. Control*, Academic Press, New York, USA, v. 8, p. 338–353, 1965.

ZANG, K. et al. New modification of fuzzy c-means clustering algorithm. *Fuzzy Information and Engineering*, v. 1, p. 445–448, 2009.

ZHANG, D.-Q.; CHEN, S.-C. A comment on "alternative c-means clustering algorithms". *Pattern Recognition*, v. 37, n. 2, p. 173–174, 2004.

ZHANG, W.; HU, H.; LIU, W. Rules Extraction of Interval Type-2 Fuzzy Logic System Based on Fuzzy c-Means Clustering. *Fuzzy Systems and Knowledge Discovery, Fourth International Conference on*, IEEE Computer Society, Los Alamitos, USA, v. 2, p. 256–260, 2007.

ZOU, K.; WANG, Z.; HU, M. An new initialization method for fuzzy c-means algorithm. *Fuzzy Optimization and Decision Making*, Kluwer Academic Publishers, Hingham, USA, v. 7, n. 4, p. 409–416, 2008.