UNIVERSIDADE DO EXTREMO SUL CATARINENSE – UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

ADEMAR CROTTI JUNIOR

O MÉTODO DE LÓGICA *FUZZY* PELOS ALGORITMOS *ROBUST C-PROTOTYPES* E *UNSUPERVISED ROBUST C-PROTOTYPES* PARA A TAREFA DE CLUSTERIZAÇÃO NA *SHELL* ORION *DATA MINING ENGINE*

CRICIÚMA, JUNHO DE 2010

ADEMAR CROTTI JUNIOR

O MÉTODO DE LÓGICA *FUZZY* PELOS ALGORITMOS *ROBUST C-PROTOTYPES* E *UNSUPERVISED ROBUST C-PROTOTYPES* PARA A TAREFA DE CLUSTERIZAÇÃO NA *SHELL* ORION *DATA MINING ENGINE*

Trabalho de Conclusão de Curso apresentado para obtenção do Grau de Bacharel em Ciência da Computação da Universidade do Extremo Sul Catarinense.

Orientadora: Profª. MSc. Merisandra Côrtes de Mattos.

CRICIÚMA, JUNHO DE 2010

ADEMAR CROTTI JUNIOR

O Método de Lógica *Fuzzy* pelos algoritmos Robust C-Prototypes e Unsupervised Robust C-Prototypes para a Tarefa de Clusterização na *Shell* Orion *Data Mining Engine*

Submetido ao corpo docente do Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense como um dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Profa. MSc. Ana Claudia Garcia Barbosa
Coordenadora do Curso de Ciência da Computação

Banca Examinadora:

Prof. MSc. Merisandra Côrtes de Mattos (UNESC)
Orientador

Prof. MSc. Cristian Cechinel (Universidade Federal do Pampa, Bagé/RS)

Prof. MSc. Kristian Madeira (UNESC)

RESUMO

A evolução dos modelos de armazenamento de dados possibilitou, às mais diversas organizações, a criação de grandes bases de dados. A fim de extrair conhecimento novo e útil destas bases de dados, com o objetivo de auxiliar estas empresas na tomada de decisão, surgiu o conceito de *data mining*, etapa principal do processo de descoberta de conhecimento, onde as ferramentas responsáveis por auxiliar neste processo são denominadas s*hells*. Considerando isto, encontra-se em desenvolvimento, pelo Grupo de Inteligência Computacional Aplicada da UNESC, o projeto da *Shell* Orion *Data Mining Engine*, que consiste na criação de uma ferramenta gratuita que implemente, por meio de diferentes métodos, o processo de descoberta de conhecimento. Desta forma, esta pesquisa fundamentou-se, inicialmente, na demonstração matemática e implementação dos algoritmos de lógica *fuzzy* Robust C-Prototypes e Unsupervised Robust C-Prototypes, sendo que posteriormente, devido a sua importância, foi implementado o *Fuzzy* C-Means, para a tarefa de clusterização. A tarefa de clusterização tem como objetivo procurar padrões e relações nos dados, formando grupos de objetos similares, sendo que o método de lógica *fuzzy* auxilia neste processo, por possibilitar aos elementos pertencerem a diversos grupos simultaneamente. Além disto, os algoritmos Robust C-Prototypes e Unsupervised Robust C-Prototypes utilizam funções, que aplicadas ao método de lógica *fuzzy*, melhoram os resultados gerados por estes algoritmos, quando aplicados em bases contaminadas por ruídos. Ao final da pesquisa foram efetuados vários testes que comprovaram, junto com os métodos de validação aplicados aos resultados, o correto funcionamento dos modelos implementados.

Palavras-chave: *Data Mining*, Clusterização, Lógica *fuzzy*, Clusterização de dados com ruídos.

ABSTRACT

The evolution of data storage models made possible to all kinds of institutions the creation of very large databases. With the goal of extracting new and useful knowledge off of those databases, as a mean of support in the decision making inside institutions, the main step in the knowlege discovery process, called data mining, became popular as it's available inside applications called shells. Taking that into account, there's a ongoing project by the Applied Computational Intelligence Group at UNESC, called Shell Orion Data Mining Engine, which aims to create a free application which implements the data mining process, through several different methods. Therefore, this reasearch was inicially based on mathematic demonstration and implementation of the fuzzy logic algorithms Robust C-Prototypes and Unsupervised Robust C-Prototypes. Afterwards, the clustering algorithm C-Means was also implemented, due to its importance. Clusterization algorithms are used to find patterns and relationships in the data, creating clusters of similar objects. Fuzzy logic helps in this process by allowing an object to be part of several clusters at once. Besides, when the Robust C-Prototypes and Unsupervised Robust C-Prototypes algorithms are applied to fuzzy logic, they obtain better results in noisy data. In the end of the research, several tests were made, and all of them verified, along with the validation methods employed into the results, the correct operation of the implemented models.

Keywords: Data mining, clustering, fuzzy logic. clustering noisy data.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

AGNES	*Agglomerative Nesting*
CART	*Classification and Regression Trees*
DBSCAN	*Density-Based Spatial Clustering of Applications with Noise*
DCBD	Descoberta de Conhecimento em Bases de Dados
DIANA	*Divisive Analysis*
DM	*Data Mining*
FCM	*Fuzzy* C-Means
GG	Gath-Geva
GK	Gustafson-Kessel
ID3	*Iterative Dichotomizer 3*
MAD – DMA	*Median of Absolute Deviations* – Desvio Mediano Absoluto
Med	Mediana
NC	*Noise Clustering*
PCA	*Principal Component Analysis*
RCP	Robust C-Protypes
SOM	*Self-Organizing Maps*
SQL	*Strutured Query Language*
STING	*Statistical Information Grid*
TCC	Trabalho de Conclusão de Curso
UML	*Unified Modeling Language*
UNESC	Universidade do Extremo Sul Catarinense
URCP	Unsupervised Robust C-Prototypes
UTI	Unidade de Terapia Intensiva

SUMÁRIO

1 INTRODUÇÃO

O volume de dados armazenados em diversos tipos de instituições aumenta diariamente. Sendo que, para seu benefício, é necessário utilizar métodos que possibilitem a extração de conhecimento relevante, capaz de auxiliar as organizações na tomada de decisão.

Desta forma, para analisar, interpretar e transformar estes dados em conhecimento surgiu uma área de estudos denominada Descoberta de Conhecimento em Bases de Dados (DCBD) (GOLDSCHMIDIT; PASSOS, 2005).

Segundo Fayyad, Piatetsky-Shapiro e Smith (1996, tradução nossa) DCBD é o processo automático de identificação de características e relacionamentos novos e válidos nos dados, e que possam ser transformados em conhecimento potencialmente útil e compreensível ao ser humano.

A busca por padrões nos dados ocorre na etapa de *data mining* (DM), sendo esta a principal do DCBD. Nela são empregados tarefas e métodos com características distintas, que são aplicados de acordo com o objetivo da descoberta de conhecimento (GOLDSCHMIDT; PASSOS, 2005).

As tarefas, métodos e algoritmos de DM são implementados em ferramentas denominadas *shells*. Existem diversas destas ferramentas no mercado, sendo que a maioria é comercial e de alto custo, havendo carência de ferramentas gratuitas.

Dentre estas ferramentas, existe em desenvolvimento a *Shell* Orion *Data Mining Engine*, um projeto acadêmico implementado pelo Grupo de Pesquisa em Inteligência Computacional Aplicada, do Curso de Ciência da Computação, na Universidade do Extremo Sul Catarinense.

Encontra-se implementado na *Shell* Orion os módulos de associação pelo algoritmo Apriori, de classificação pelo ID3, CART e C4.5, e de clusterização pelos algoritmos K-means, Kohonen, Gustafson-Kessel e Gath-Geva.

A clusterização é o processo de agrupar dados em grupos de objetos similares. Um *cluster* é um conjunto de dados, onde cada elemento integrante seja similar a outros no mesmo *cluster*; e elementos em diferentes *clusters* são distintos (HAN; KAMBER, 2001).

A clusterização *crisp* (convencional) força os dados a pertencerem a um único *cluster*. Isto implica na possibilidade de um elemento ser colocado em um grupo que não compartilha suas características, tornando, algumas vezes, esta abordagem inviável, devido, muitas vezes, à imprecisão ou à não completeza dos dados (BEZDEK et al, 2005, tradução nossa).

A fim de solucionar esse problema existem métodos de clusterização baseados em lógica *fuzzy*, onde determinado elemento, ao mesmo tempo, pode estar em mais de um *cluster*, com diferentes graus de pertinência.

Existem diversos algoritmos que utilizam lógica *fuzzy* para a tarefa de clusterização, porém, o desempenho deles é insuficiente quando o conjunto de dados está contaminado por ruídos[1] e *outliers*[2] (FRIGUI; KRISHAPURAM, 1996, tradução nossa). Isto prejudica a execução da tarefa, e conseqüentemente causa distorções no conhecimento gerado.

Assim, surgiu uma nova classe de algoritmos, que utilizam estatísticas denominadas de robustas, com o objetivo de diminuir a interferência dos dados ruidosos e *outliers* na execução da tarefa (FRIGUI; KRISHAPURAM, 1996, tradução nossa). Utilizando esta abordagem, propõe-se desenvolver para a tarefa de clusterização, por meio do método de lógica *fuzzy*, a modelagem matemática e a implementação dos algoritmos Robust C-

[1] Elementos que não acompanham o comportamento geral da base de dados.
[2] Dados atípicos ou inconsistentes encontrados de um determinado conjunto.

Prototypes (RCP) e Unsupervised Robust C-Prototypes (URCP) na *Shell* Orion *Data Mining Engine*.

1.1 OBJETIVO GERAL

Disponibilizar os algoritmos Robust C-Prototypes e Unsupervised Robust C-Prototypes, na tarefa de clusterização, pelo método de lógica *fuzzy*, da *Shell* Orion *Data Mining Engine*.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos desta pesquisa consistem em:

a) entender o conceito de *data mining* e a tarefa de clusterização;
b) compreender o método de lógica *fuzzy*, a manipulação de dados com ruídos e os algoritmos Robust C-Prototypes e Unsupervised Robust C-Prototypes;
c) desenvolver os algoritmos Robust C-Prototypes e Unsupervised Robust C-Prototypes na tarefa de clusterização da *Shell* Orion *Data Mining Engine*;
d) demonstrar a modelagem matemática dos algoritmos Robust C-Prototypes e Unsupervised Robust C-Prototypes;
e) utilizar uma base de dados a fim de testar o funcionamento dos algoritmos Robust C-Prototypes e Unsupervised Robust C-Prototypes no módulo de clusterização da Shell Orion Data Mining Engine.

1.3 JUSTIFICATIVA

Data mining comporta diferentes técnicas e métodos para a descoberta de conhecimento relevante em bases de dados. Seu objetivo é auxiliar no suporte a tomada de decisões, por meio da descoberta automática de novos fatos e relações nos dados (GOLDSCHMIDT; PASSOS, 2005).

A fim de ajudar no processo de extração de conhecimento são utilizadas ferramentas denominadas *shells*. Estas são indispensáveis para que as instituições possam analisar e visualizar as relações existentes em seus dados. Há diversas *shells* de *data mining*, mas a maioria é comercial, o que impossibilita a sua aquisição por algumas organizações.

O projeto acadêmico da *Shell* Orion *Data Mining Engine*, justifica-se por implementar diferentes métodos e tarefas de *data mining* em uma ferramenta gratuita. Considerando isto, esta pesquisa consiste na continuação desse projeto, pois proporciona a ampliação das funcionalidades desta ferramenta, com o desenvolvimento de mais dois algoritmos no módulo de clusterização, utilizando o método de lógica *fuzzy*.

A clusterização é considerada uma tarefa muito importante no processo de *data mining*, pois consegue identificar padrões ou tendências, apenas observando os dados, sem definir rótulos, como na classificação[3] (HAN; KAMBER, 2001, tradução nossa). A *Shell* Orion possui implementados, utilizando a lógica tradicional, para a tarefa de clusterização, o método de particionamento pelo algoritmo K-means[4] e o método de redes neurais pelo algoritmo de Kohonen[5].

Porém, a clusterização realizada utilizando-se métodos de lógica tradicional pode fazer com que um dado seja colocado em um *cluster*, mesmo que este dado não possua similaridades com os outros objetos deste grupo. Assim, para resolver este problema, na *Shell*

---

[3] Consiste no processo de aprendizado de uma função que mapeie os registros em categorias distintas, pré-definidas, denominadas de classes (KANTARDZIC, 2003, tradução nossa).

[4] Este algoritmo distribui os elementos em *k* grupos, sendo que cada elemento é atribuído ao *cluster* cujo centro está mais próximo (GAN; MA; WU, 2007, tradução nossa).

[5] Desenvolvido por Teuvo Kohonen, este algoritmo não supervisionado consegue bons resultados, pois, ao particionar os dados, consegue formar grupos diminuindo a dimensão da base sem causar distorções nos relacionamentos entre os dados (CARVALHO, 2005).

Orion utiliza-se o método de lógica *fuzzy*, que proporciona a geração de um conhecimento mais preciso, já que os elementos podem pertencer a mais de um *cluster* simultaneamente, sendo possível quantificar a pertinência de cada elemento em cada *cluster* (OLIVEIRA; PEDRYCZ, 2007, tradução nossa). Pelo método de lógica *fuzzy*, na tarefa de clusterização, a *Shell* Orion têm implementado os algoritmos Gustafson-Kessel e Gath-Geva.

Mesmo utilizando a lógica *fuzzy*, a execução da tarefa de clusterização pode ser inadequada se aplicada em uma base de dados contaminada com ruídos e *outliers*. A fim de reduzir a interferência de dados ruidosos e *outliers*, e por conseqüência melhorar a geração do conhecimento, surgiram algoritmos que utilizam estatísticas robustas[6] (FRIGUI; KRISHAPURAM, 1996, tradução nossa). Neste contexto, foram implementados na Shell Orion, os algoritmos Robust C-Prototypes e Unsupervised Robust C-Prototypes que integram lógica *fuzzy* e estatísticas robustas.

Os algoritmos RCP e URCP, propostos por Hichem Frigui e Raghu Krishnapuram, geram *clusters* em forma de elipse, onde os resultados da execução da tarefa não dependem da presença ou ausência de ruídos e *outliers* no conjunto de dados. Essa característica faz com que os dados ruidosos e *outliers* fiquem dispersos, em torno de densas regiões, que são os *clusters* propriamente ditos (OLIVEIRA; PEDRYCZ, 2007, tradução nossa; FRIGUI; KRISHAPURAM, 1996, tradução nossa).

1.4 ESTRUTURA DO TRABALHO

Esta pesquisa é composta por seis capítulos, sendo que o Capítulo 1 contextualiza o tema proposto, objetivos e justificativa para a realização deste trabalho.

[6] Técnicas utilizadas com o objetivo de diminuir a interferência de valores atípicos em um determinado modelo.

Conceitos de DM e a ferramenta *Shell* Orion *Data Mining Engine*, com suas funcionalidades, tarefas, métodos e algoritmos implementados estão presentes no Capítulo 2.

No Capítulo 3 é descrita a tarefa de clusterização no processo de *data mining*. No mesmo capítulo também é descrito o método de lógica *fuzzy* para a clusterização, alguns exemplos de algoritmos que o implementam, além da clusterização pelo método de lógica *fuzzy* em dados ruidosos e alguns exemplos de algoritmos que tratam este tipo de dado.

Os algoritmos Robust C-Prototypes e Unsupervised Robust C-Prototypes são apresentados detalhadamente no Capítulo 4. O Capítulo 5 apresenta alguns exemplos de aplicações dos algoritmos RCP e URCP.

No Capítulo 6 é descrito o trabalhado desenvolvido, a modelagem matemática dos algoritmos, assim como os resultados obtidos.

Por fim, tem-se a conclusão desta pesquisa e algumas sugestões para trabalhos futuros.

2 *SHELL* ORION *DATA MINING ENGINE*

A *Shell* Orion é um projeto acadêmico implementado por professores e acadêmicos do Grupo de Pesquisa em Inteligência Computacional Aplicada, do Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense – UNESC. Sendo o objetivo, implementar uma ferramenta gratuita de DM (exemplos de ferramentas de DCBD são apresentados no Anexo A).

Segundo Goldschmidit e Passos (2005) DM é a etapa responsável pela abstração de conhecimento a partir dos dados.

O conceito de DM é analisar bases de dados através de métodos inteligentes para descobrir padrões e relações automaticamente (HAN; KAMBER, 2001, tradução nossa). Sendo que estes padrões devem ser úteis para, por exemplo, resolver problemas ou tomar decisões (WITTEN; FRANK, 2005, tradução nossa).

Assim, o projeto da *Shell* Orion foi iniciado em 2005, e até o momento possui implementado os módulos de associação[7], classificação e clusterização. Os métodos foram desenvolvidos por acadêmicos como Trabalho de Conclusão de Curso (TCC). Na Tabela 1 são apresentados os algoritmos desenvolvidos até o momento:

[7] Esta tarefa busca combinações, através de relacionamentos que possam ser considerados tendências, definindo padrões de comportamento que ajudem na compreensão dos dados (KANTARDZIC, 2003, tradução nossa).

Tabela 1. Algoritmos implementados na *Shell* Orion *Data Mining Engine*

Ano	Tarefa	Método	Algoritmo	Atributos	Referência
2005	Associação	Regras de Associação	*Apriori*	Numéricos	(CASAGRANDE, 2005)
2005	Classificação	Árvores de Decisão	ID3	Nominais	(PELEGRIN, 2005)
2007	Classificação	Árvores de Decisão	CART	Nominais e Numéricos	(RAIMUNDO, 2007)
2007	Clusterização	Particionamento	*K-Means*	Numéricos	(MARTINS, 2007)
2007	Clusterização	Redes Neurais	*Kohonen*	Numéricos	(BORTOLOTTO, 2007)
2008	Clusterização	Lógica *Fuzzy*	*Gustafson-Kessel*	Numéricos	(CASSETARI JÚNIOR, 2008)
2009	Clusterização	Lógica *Fuzzy*	*Gath-Geva*	Numéricos	(PEREGO, 2009)
2009	Classificação	Árvores de Decisão	C4.5	Nominais e Numéricos	(MONDARDO, 2009)

A ferramenta está sendo desenvolvida utilizando a linguagem de programação *Java*, dentre tantas disponíveis, esta foi escolhida por ser gratuita, multiplataforma, permitir reutilização de código e por haver ambientes de programação gratuitos para ela (PELEGRIN, 2005; BORTOLOTTO, 2007).

O módulo de clusterização da ferramenta será explicado detalhadamente, por ser o módulo ao qual esta pesquisa adicionou funcionalidades.

Desta forma, o primeiro algoritmo implementado no módulo de clusterização foi o *K-means*. Este algoritmo divide um conjunto de dados em *k clusters*, onde *k* é a quantidade de dados que serão escolhidos randomicamente para serem os elementos centrais dos *clusters*. Em seguida são calculadas as distâncias entre os registros restantes da base de dados e os centros dos clusters, sendo que o registro é atribuído ao *cluster* com menor distância deste em relação a seu centro, ou seja, o atributo é adicionado ao *cluster* cujo centro seja mais semelhante a ele (GOLDSCHMIDIT; PASSOS, 2005).

Ao utilizar o algoritmo na *Shell* Orion o usuário deve informar o número de *clusters*, variáveis de entrada, variável de saída e o tipo do cálculo de distância, sendo que as disponíveis são a euclidiana[8] e *city-block*[9]. A Figura 1 mostra a interface inicial do algoritmo e seus resultados na ferramenta:

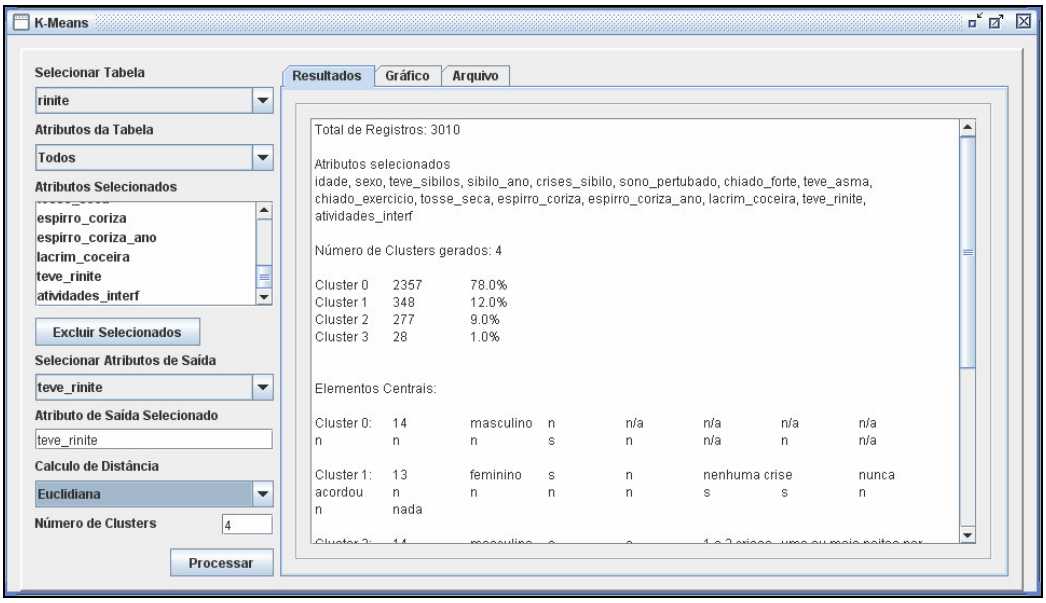

Figura 1. Algoritmo *K-means* na Shell Orion
Fonte: MARTINS, D. (2007)

Encontra-se também desenvolvido o método de redes neurais[10] pelo algoritmo de Kohonen. Este algoritmo, que também é conhecido como Mapa Auto-Organizável (*Self-Organizing Map* - SOM), caracteriza-se por ter todos os neurônios da camada de entrada conectados com os neurônios da camada de saída (CARVALHO, 2005). O aprendizado da rede é não supervisionado[11] (GOLDSCHMIDIT; PASSOS, 2005).

O resultado do algoritmo pode ser obtido, na ferramenta, em forma gráfica, visualização dos grupos, em forma de árvore e informações estatísticas gerados pelo algoritmo, além da possibilidade destas informações serem exportadas em formato SQL. A

[8] Transforma o espaço vetorial, de dimensão finita, em espaço métrico, definindo a distância entre dois pontos.
[9] Definida como a diferença absoluta entre dois pontos.
[10] Modelos matemáticos que se assemelham as estruturas neurais biológicas, com o objetivo de simular o mecanismo de processamento do cérebro humano (HAYKIN, 2001; REZENDE, 2003).
[11] Os pesos da rede são ajustados a medida que esta recebe padrões de entrada (HAYKIN, 2001).

Figura 2 apresenta a interface para informar os parâmetros de entrada e o gráfico gerado pela ferramenta.

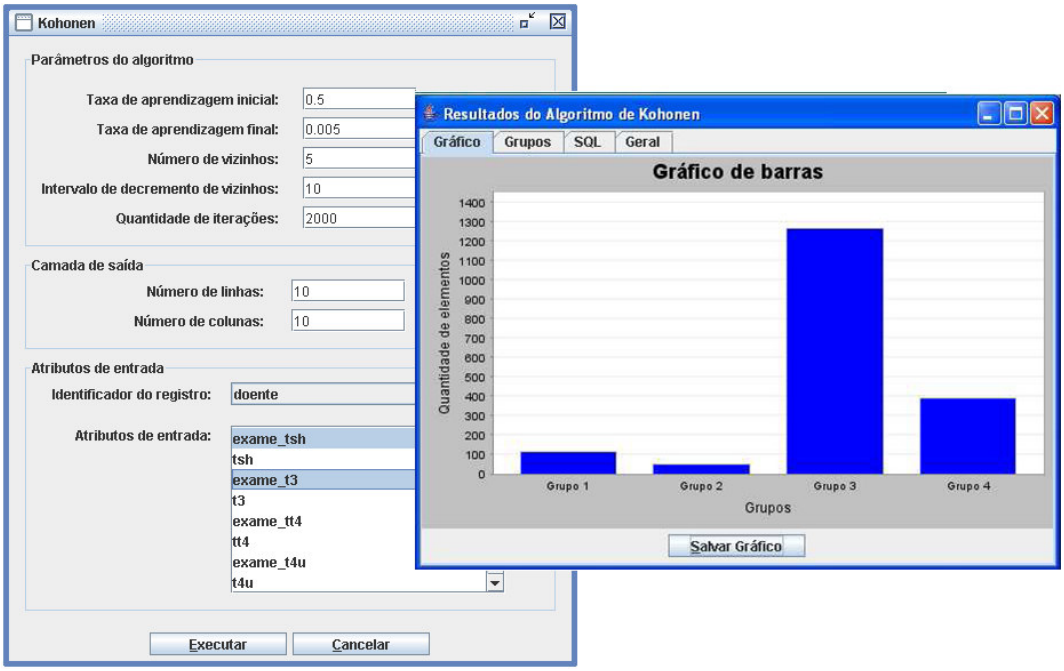

Figura 2. Algoritmo de Kohonen na *Shell* Orion
Fonte: Adaptado de BORTOLOTTO, L. (2007)

Por meio do método de lógica *fuzzy*, para a tarefa de clusterização existem dois algoritmos implementados. O primeiro foi o Gustafson-Kessel (GK), que por meio de uma distância adaptável, gera *clusters* de variados tamanhos e formas (OLIVEIRA; PEDRYCZ, 2007, tradução nossa). A Figura 3 apresenta sua interface, e o resumo dos resultados encontrados pelo algoritmo na *Shell* Orion.

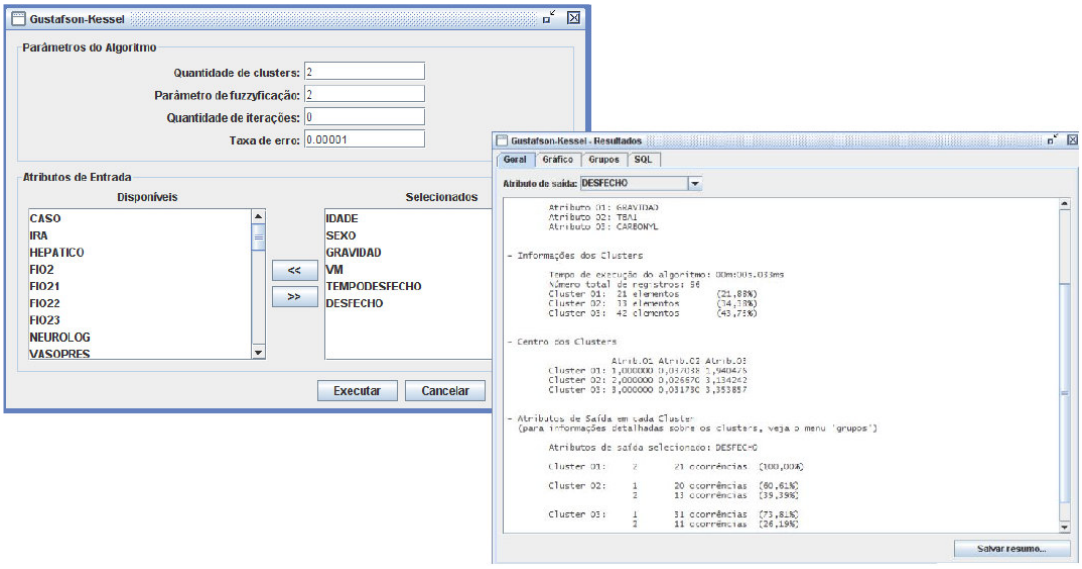

Figura 3. Algoritmo GK na *Shell* Orion
Fonte: Adaptado de CASSETARI JÚNIOR, J. (2008)

O Gath-Geva (GG) é o segundo algoritmo implementado na Orion que utiliza o método de lógica *fuzzy*, para a tarefa de clusterização. A grande diferença entre o GK e o GG está no cálculo da distância, sendo que a distância utilizada no GG é exponencial, baseada em estimativa de probabilidades, e, portanto gera *clusters* de maior precisão (ABONYI, 2002, tradução nossa). A Figura 4 apresenta a interface e o resultado, em forma gráfica, obtida por este algoritmo.

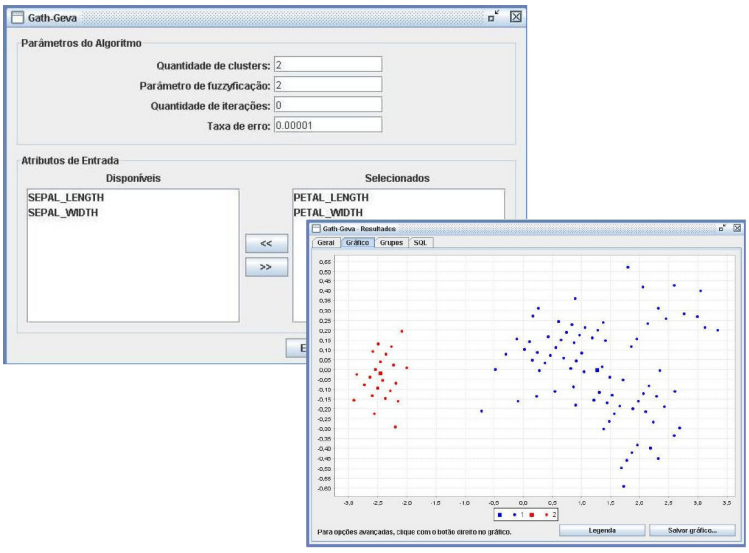

Figura 4. Algoritmo GG na Orion
Fonte: Adaptado de PEREGO, D. (2009)

O resultado também é mostrado em forma de árvore, e as informações geradas pelo algoritmo podem ser exportado em formato SQL.

Assim, esta pesquisa consiste em aumentar as funcionalidades da ferramenta, desenvolvendo a modelagem matemática e implementação dos algoritmos *Robust C-Prototypes* e *Unsupervised Robust C-Prototypes*, que utilizam o método de lógica *fuzzy* para a tarefa de clusterização.

3 A TAREFA DE CLUSTERIZAÇÃO EM *DATA MINING*

O processo de agrupar dados, em diferentes grupos ou *clusters*, de acordo com as semelhanças entre os elementos, é chamado de clusterização. Esta tarefa é muitas vezes confundida com a classificação, onde os objetos são atribuídos a classes pré-definidas, sendo que na clusterização os grupos são criados durante o processo, com base nas semelhanças existentes entre os dados. Esta característica faz com que a tarefa de clusterização seja classificada como não-supervisionada (BASU; DAVIDSON; WAGSTAFF, 2008, tradução nossa; GAN; MA; WU, 2007, tradução nossa).

Um *cluster* é uma coleção de registros similares entre si, cujas características devem o diferenciar dos outros *clusters*. Desta forma, a clusterização busca maximizar a similaridade entre objetos do mesmo *cluster*, e conseqüentemente minimizar as semelhanças entre elementos de diferentes *clusters* (LAROSE, 2005, tradução nossa).

Frequentemente, a tarefa de clusterização é utilizada como primeiro passo no processo de extração de padrões, onde o resultado é aplicado como entrada para alguma outra tarefa. Neste caso, a clusterização tem como objetivo facilitar e melhorar o desempenho de outros algoritmos de *data mining* no processo de descoberta de conhecimento (BERRY; LINOFF, 2004, tradução nossa).

O objetivo da clusterização é identificar um conjunto finito de categorias que descrevem os dados, sendo que os dados podem revelar *clusters* de diferentes formas, tamanhos e densidades. A Figura 5 mostra que os *clusters* podem ser esféricos (a), lineares (b) ou ocos (c) e (d), onde os protótipos são pontos (a), linhas (b), esferas (c) ou elipses (d), ou seus análogos de maior dimensão (ABONYI; FEIL, 2007).

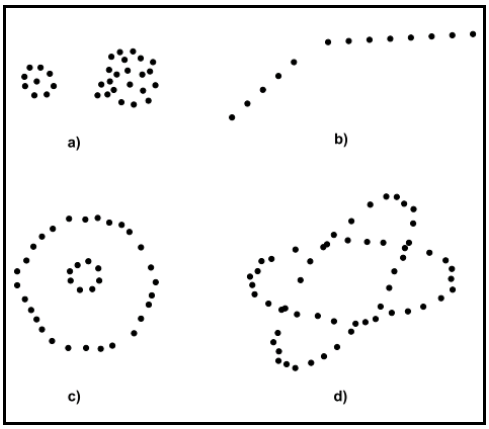

Figura 5. *Clusters* de diferentes formas
Fonte: ABONYI, J.; FEIL, B. (2007)

O desempenho da maior parte dos algoritmos de clusterização sofre influência sobre às formas, densidades dos *clusters*, relações espaciais e distâncias, sendo que os grupos podem estar bem separados, continuamente ligados ou se sobreporem. A separação entre os *clusters* é diretamente influenciada pela normalização, dimensionamento dos dados, presença de ruídos e *outliers* (ABONYI; FEIL, 2007).

Considerando que a qualidade dos dados interfere diretamente no processo de clusterização, e que em cada caso deve-se escolher algoritmos específicos visando melhor atender as necessidades e assim encontrar resultados satisfatórios, esta é uma área com grandes esforços em pesquisas.

Os algoritmos desenvolvidos para a tarefa de clusterização podem ser divididos em métodos específicos, que podem ser aplicados em determinadas bases de dados a fim de obter maior precisão, de acordo com os objetivos da extração de padrões.

Sengundo Han e Kamber (2001, tradução nossa) os principais métodos de clusterização são: os métodos hierárquicos, que decompõem uma base de dados de forma hierárquica, representando cada *cluster* como um nó de uma árvore; métodos baseados em densidade, dividindo os dados de acordo com sua densidade; métodos baseados em grade, que particionam os elementos de uma base de dados em células, formando uma estrutura multidimensional de grade; métodos baseados em modelos, que buscam definir e organizar os

clusters através de modelos; e os métodos de particionamento, dividindo *n* elementos de uma base de dados em *k clusters*.

O algoritmo mais conhecido e utilizado do método de particionamento é o K-Means (Figura 6). Este algoritmo possui uma função de erro e o número de *clusters k* fixo, o algoritmo segue distribuindo os dados aos *clusters* mais próximos até atingir sua função de erro ou até que os objetos dos *clusters* não sofram modificações significativas (GAN; MA; WU, 2007, tradução nossa).

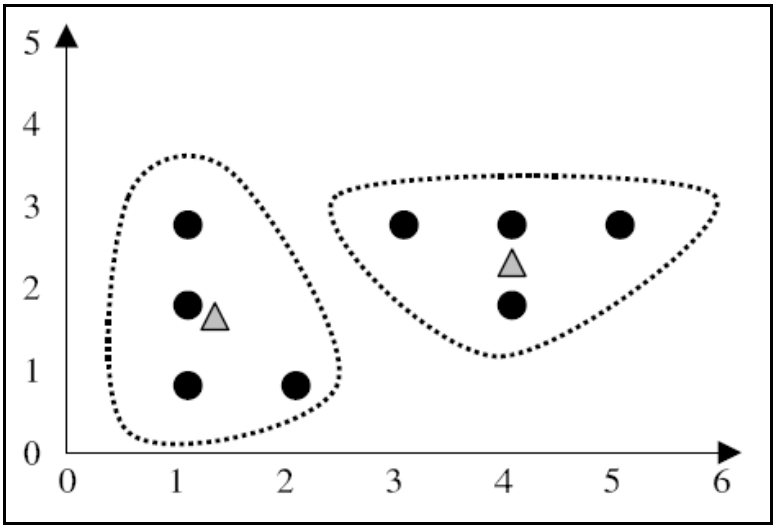

Figura 6. Método de particionamento por meio do algoritmo *K-Means*
Fonte: LAROSE, D. (2005)

Desta forma, um elemento pode ser atribuído a um grupo que não compartilhe suas característica, obtendo *clusters* de baixa qualidade e prejudicando os resultados gerados pelo método (BEZDEK et al, 2005, tradução nossa).

Considerando isto, o método de lógica *fuzzy* é apresentado a seguir. Este método tem como objetivo, no processo de clusterização, o auxílio a identificação correta dos *clusters*, devido a possibilidade dos elementos pertencerem a vários grupos simultaneamente.

3.1 O MÉTODO DE LÓGICA *FUZZY*

A tarefa de clusterização agrupa dados não rotulados em *clusters* de acordo com suas características. O algoritmo deve possuir capacidade para dividir as informações corretamente, a fim de encontrar *clusters* de alta qualidade, e assim obter sucesso na execução da tarefa (OLSON; DELEN, 2008, tradução nossa).

A lógica tradicional ou booleana (*crisp*) é a base dos algoritmos tradicionais de clusterização. Esta abordagem faz com que cada elemento pertença a um único *cluster*, o que dificulta representar a realidade dos dados, pois na maior parte das aplicações do mundo real, estes são imprecisos ou incertos (ABONYI; FEIL, 2007, tradução nossa). A Figura 7 mostra dois *clusters* identificados por meio de um método *crisp*.

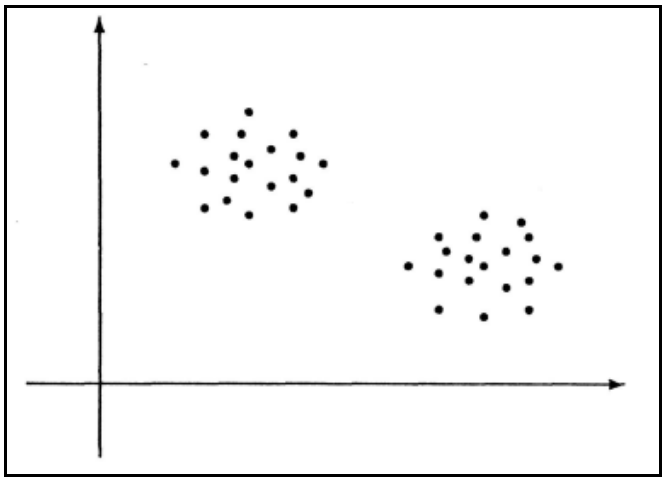

Figura 7. Reconhecimento de dois *clusters*
Fonte: Adaptado de HÖPPNER, F. et al (1999)

Considerando que existem poucos casos onde as diferenças entre padrões são claras, surgiu o método de lógica *fuzzy*, que com o uso de valores que representam a pertinência dos dados em cada grupo, apresenta maior flexibilidade na execução da tarefa, e assim consegue extrair resultados mais úteis para aplicações práticas (CHI; YAN; PHAM, 1996, tradução nossa). A Figura 8 apresenta dois *clusters* reconhecidos através de um método

fuzzy, percebe-se na imagem um objeto entre os conjuntos, este elemento pertence simultaneamente aos dois grupos (μ = 0.5).

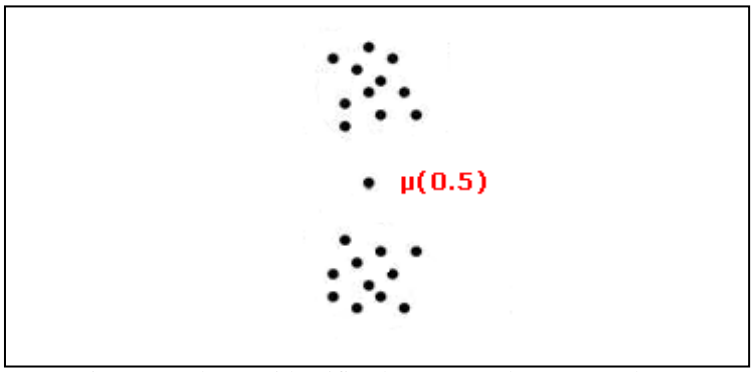

Figura 8. *Clusters* identificados através de um método *fuzzy*
Fonte: Adaptado de OLIVEIRA, J. V.; PEDRYCZ, W. (2007)

A lógica *fuzzy* permite valores entre 0 e 1 em relação a pertinência de um objeto a um determinado conjunto e desta forma possibilita obter melhores resultados em aplicações que possuam dados imprecisos (COX, 2005, tradução nossa).

3.1.1 Lógica *Fuzzy*

Na teoria clássica dos conjuntos cada elemento é associado a apenas um grupo, onde a pertinência de um elemento pode ser dada como {0,1}. Estes valores representam, respectivamente, a pertinência e a não-pertinência de um elemento a um conjunto (OLIVEIRA JUNIOR; CALDEIRA, 2007, SATO-ILIC; JAIN, 2006, tradução nossa).

Um conjunto *fuzzy* é representado por meio de um par ordenado, onde o primeiro elemento do par é o objeto pertencente ao conjunto e o segundo é um valor que representa a pertinência do objeto ao conjunto, este valor deve estar no intervalo [0,1]. Por exemplo, dado um conjunto A, o par ordenado que represente os elementos e pertinências desse conjunto deve seguir a função $\mu_A : X \rightarrow [0,1]$, onde para qualquer elemento $x \in X$, $\mu_A(x) \in [0,1]$ (SATO-ILIC; JAIN, 2006, tradução nossa).

Utilizando esta abordagem, a lógica *fuzzy* é um método que possibilita lidar com a imprecisão dos dados, a fim de encontrar melhores soluções para um problema. Assim como vários tipos de raciocínio humano, a lógica *fuzzy* utiliza a aproximação ao invés do raciocínio exato (GALINDO; URRUTIA; PIATTINI, 2006, tradução nossa).

A aproximação faz com que a transição entre os conjuntos aconteça de forma mais suave, por meio de funções de pertinência que determinam quanto um elemento pertence a um determinado conjunto (COX, 2005, tradução nossa).

O conjunto definido por uma função de pertinência é chamado de conjunto *fuzzy*, e as funções de pertinências mais utilizadas são descritas a seguir (CHI; YAN; PHAM, 1996, tradução nossa):

a) **função triangular:** tem forma de triângulo, e sua função geral é determinada por:

$$\mu(x) = \begin{cases} 0, se\ x \leq a \\ \dfrac{x-a}{b-a}, se\ x \in [a,b] \\ \dfrac{c-x}{c-b}, se\ x \in [b,c] \\ 0, se\ x \geq b \end{cases}$$

b) **função trapezoidal:** apresenta a forma de trapézio, onde sua função geral é definida por:

$$\mu(x) = \begin{cases} 0, se\ x < a \\ \dfrac{x-a}{b-a}, se\ x \in [a,b] \\ 1, \text{se } x \in [b,c] \\ \dfrac{d-x}{d-c}, se\ x \in [c,d] \\ 0, se\ x > d \end{cases}$$

c) **função S:** é representada por meio de uma curvatura em forma de S, definida por:

$$\mu(x) = \begin{cases} 0, se\ x \le a \\ 2 \cdot \left(\frac{x-a}{c-a} \right)^2, se\ x \in [a,b] \\ 1 - 2 \cdot \left(\frac{x-c}{c-a} \right)^2, se\ x \in [b,c] \\ 1, se\ x > c \end{cases}$$

d) **função sino:** possui formato de sino, sendo representa pela fórmula:

$$\mu(x) = \frac{1}{1 + \left| \frac{x-c}{a} \right|^{2b}}$$

A Figura 9 mostra estas funções em gráficos.

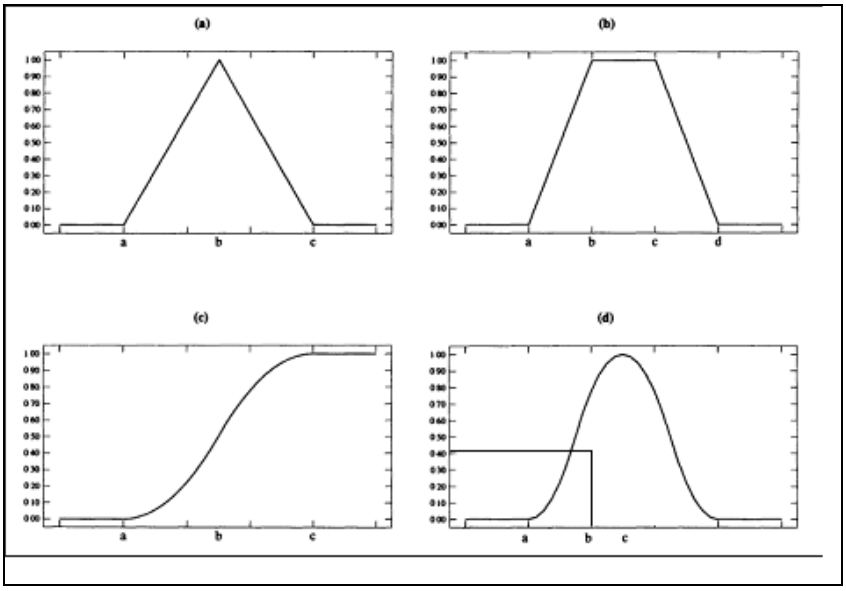

Figura 9. Representação gráfica das funções de pertinência
Fonte: Adaptado de CHI, Z.; YAN, H.; PHAM, T. (1996)

Os conjuntos *fuzzy* normalmente são identificados por meio de variáveis lingüísticas. A Figura 10 apresenta cinco conjuntos *fuzzy* que utilizam as funções triangular e trapezoidal para representar conceitos abstratos referentes a idade. Por exemplo, uma pessoa que possua 35 anos é considerada adulto e de meia-idade, simultaneamente.

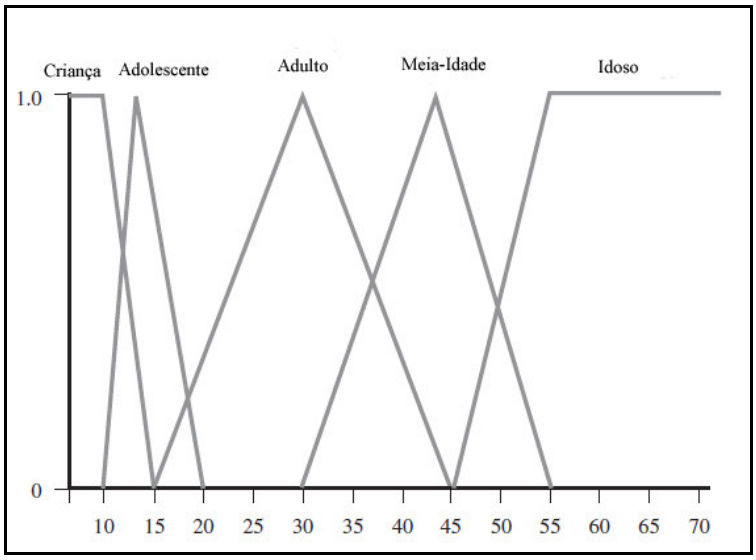

Figura 10. Conjuntos *fuzzy* para representar a variável idade
Fonte: Adaptado de COX, E. (2005)

A lógica *fuzzy* é utilizada em diversas áreas, entre elas, à tarefa de clusterização, onde cada registro possui um grau de pertinência relacionado a cada *cluster* encontrado (COX, 2005, tradução nossa).

Existem muitos algoritmos que utilizam o método de lógica *fuzzy* para a tarefa de clusterização, sendo que sua escolha depende do domínio de aplicação e dos objetivos da descoberta de conhecimento.

3.1.2 Algoritmos de Lógica *Fuzzy* para Clusterização

Os algoritmos que implementam o método de lógica *fuzzy* proporcionam melhores resultados devido aos graus de pertinência, possibilitando a um elemento estar, ao mesmo tempo, em mais de um *cluster* (BEZDEK et al, 2005, tradução nossa).

Existem diversos algoritmos que implementam o modelo *fuzzy* para a tarefa de clusterização. A escolha do algoritmo deve acontecer considerando o conhecimento prévio que se tem da base de dados, e o tipo de padrões que se deseja extrair (COX, 2005, tradução nossa).

A seguir são apresentados alguns algoritmos que utilizam o método de lógica *fuzzy* para a tarefa de clusterização.

3.1.2.1 *Fuzzy* C-Means

O *Fuzzy* C-Means (FCM) baseou-se no *K-means*, sendo que sua versão final foi proposta por James C. Bezdek em 1973 (BEZDEK et al, 2005, tradução nossa). Este algoritmo foi o primeiro a utilizar o elemento *fuzzificador*, que determina o grau de *fuzzyficação* entre os elementos, considerando que quanto maior o seu valor mais os elementos serão relacionados. A fim de obter *clusters* de qualidade foram realizados testes que definiram o valor 2 como recomendado (COX, 2005, tradução nossa).

O FCM utiliza a distância Euclidiana, gerando clusters esféricos, assim como o K-means. A Figura 11 mostra o resultado gerado pelo FCM, onde os dados foram divididos em dois *clusters*, e os elementos entre os grupos pertencem a estes dois *clusters* com diferentes pertinências.

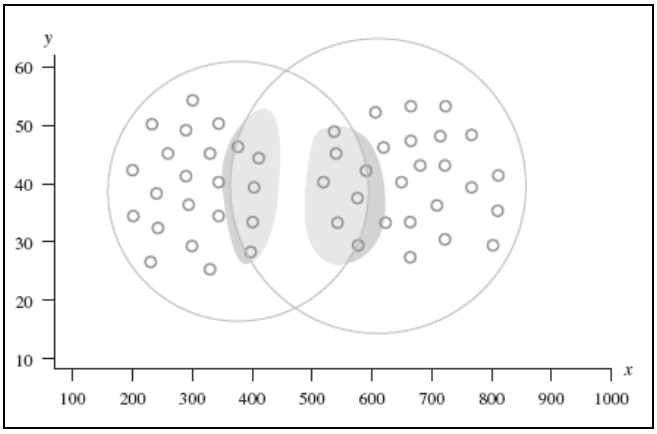

Figura 11. Clusterização pelo algoritmo FCM
Fonte: Adaptado de COX, E. (2005)

O FCM é um algoritmo tradicional na tarefa de clusterização, sendo que muitos outros algoritmos foram criados baseando-se nele, com o objetivo de obter melhores resultados. Para isso, consideraram a dificuldade do FCM em lidar com dados ruidosos e as

suas limitações na identificação de *clusters* de diferentes formas (BEZDEK et al, 2005, tradução nossa).

3.1.2.2 Gustafson-Kessel

O Gustafson-Kessel (GK) é um dos muitos algoritmos que basearam-se no FCM. Este foi apresentado em 1979 por Donald E. Gustafson e Wiliam C. Kessel na *IEEE Conference on Decision Control*.

A principal mudança em relação ao FCM foi a utilização da distância de Mahalanobis ao invés da Euclidiana, sendo que esta distância implementa uma matriz de covariância[12]. Estas modificações possibilitaram ao GK a identificação de *clusters* de diferentes tamanhos e de forma elipsoidal e esférica, considerando que o FCM identifica apenas *clusters* esféricos (HÖPPNER et al, 1999, tradução nossa).

A Figura 12 apresenta a identificação de três *clusters* por meio do algoritmo GK.

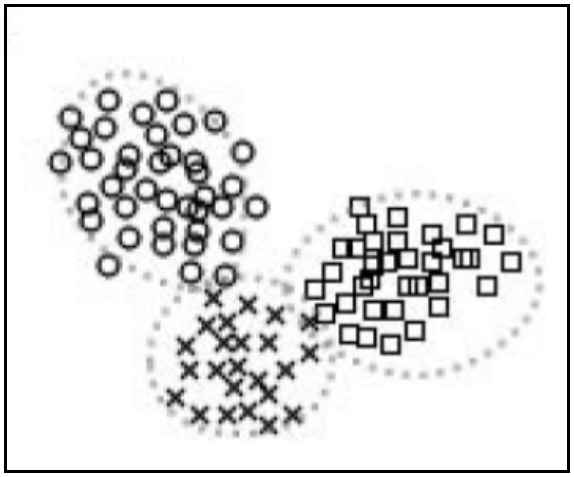

Figura 12. Clusterização pelo algoritmo GK
Fonte: Adaptado de BEZDEK, J. et al (2005)

[12] Matriz que contem em sua diagonal principal as variâncias das colunas, ou seja, o afastamento de seus elementos em torno da sua média, e nas posições restantes as covariâncias, que são como os elementos variam de acordo com os valores esperados.

3.1.2.3 Gath-Geva

Isak Gath e Amir Geva, em 1989, por meio de modificações no FCM e no GK apresentaram o algoritmo Gath-Geva (GG), na *IEEE Pattern Analysis and Machine Intelligence*.

O GG é considerado uma evolução do GK, onde sua principal diferença é a utilização de um termo exponencial no cálculo da distância. Esta modificação fez com que o algoritmo convirja mais rapidamente, encontrando grupos mais precisos. Assim como o GK, o algoritmo utiliza uma matriz de covariância, produzindo *clusters* de diferentes formas e tamanhos (ABONYI, 2002, tradução nossa).

Na Figura 13 são apresentados três *clusters* identificados pelo GG.

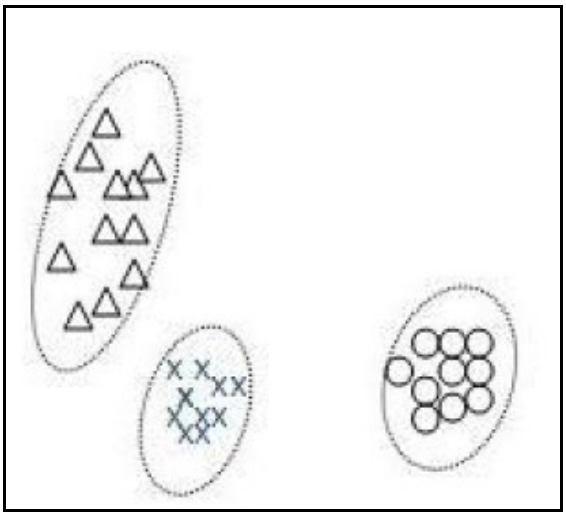

Figura 13. Clusterização pelo algoritmo GG
Fonte: Adaptado de BEZDEK, J. et al (2005)

Os algoritmos apresentados podem sofrer grande influência em seus resultados, se aplicados em bases de dados contaminadas por ruídos.

3.2 CLUSTERIZAÇÃO DE DADOS RUIDOSOS POR MEIO DA LÓGICA *FUZZY*

Bases de dados com ruídos podem possuir *outliers*, ausência de valores, dados errôneos, duplicados; ou ainda, valores que foram transcritos inadequadamente, o que dificulta a identificação de padrões nos dados, e consequentemente geram distorções nos resultados (COX, 2005, tradução nossa).

O conhecimento encontrado pela tarefa de clusterização em bases que contenham dados ruidosos pode ser praticamente nulo. Considerando que muitas vezes, devido ao tamanho da base de dados, a etapa de pré-processamento, necessária para gerar resultados satisfatórios, pode custar muito e/ou levar um tempo inviável (COX, 2005, tradução nossa).

O método de lógica *fuzzy* na tarefa de clusterização aplicada em bases de dados que contenham ruídos, proporciona resultados de maior qualidade em relação a métodos *crisp*, sendo que somente a utilização da lógica *fuzzy* pode não ser suficiente para obter sucesso na execução da tarefa (OLIVEIRA; PEDRYCZ, 2007, tradução nossa).

Desta forma, foram desenvolvidos algoritmos específicos, que por meio de diferentes abordagens, procuram diminuir a influência dos dados ruidosos, com o objetivo de obter melhores resultados (DÖRING; LESOT; KRUSE, 2006, tradução nossa).

3.2.1 Algoritmos de Lógica *Fuzzy* na Clusterização de Dados com Ruídos

Os algoritmos desenvolvidos para lidar com dados ruidosos, geralmente, necessitam de maior processamento computacional, devido a tentativa de gerar resultados independentes da ausência ou presença de ruídos (OLIVEIRA; PEDRYCZ, 2007, tradução nossa).

Existem diferentes abordagens para lidar com o problema da clusterização em dados ruidosos. Sendo que os algoritmos mais conhecidos e suas abordagens são apresentados a seguir (DÖRING; LESOT; KRUSE, 2006, tradução nossa).

3.2.1.1 *Noise Clustering*

O algoritmo *Noise Clustering* (NC) foi proposto em 1991 por Rajesh Davè. Sua abordagem busca agrupar os dados ruidosos em um único *cluster* conceitual, chamado *noise cluster* (SEISING, 2009, tradução nossa).

A principio, o centro do *noise cluster* fica a uma distância constante de todos os dados, isso significa que neste momento todos os elementos possuem a mesma probabilidade de pertencer ao *noise cluster*. Durante o processo, os dados que ficarem distantes de todos os grupos serão atribuídos ao *noise cluster*, ou seja, os dados ruidosos terão grau de pertinência baixo em relação a *clusters* padrão e alto em relação ao *noise cluster* (OLIVEIRA; PEDRYCZ, 2007, tradução nossa; SEISING, 2009, tradução nossa).

3.2.1.2 Robust C-Prototypes e Unsupervised Robust C-Prototypes

Desenvolvido por Hichem Frigui e Raghu Krishnapuram, em 1996, os algoritmos Robust C-Prototypes e Unsupervised Robust C-Prototypes foram desenvolvidos a partir do FCM, com a habilidade de lidar com dados ruidosos.

A distância utilizada pelo algoritmo FCM é muito sensível a ruídos, e portanto gera resultados insatisfatórios se aplicado em bases de dados contaminadas por ruídos e *outliers* (DÖRING; LESOT; KRUSE, 2006, tradução nossa).

O algoritmo Robust C-Prototypes incorpora estimadores robustos ao FCM. Os estimadores robustos são utilizados para diminuir a influência de ruídos no processo de clusterização. Um dos parâmetros de inicialização deste algoritmo é o número de *clusters* a serem encontrados na base de dados, considerando que para determinar este número é necessário ter um conhecimento prévio sobre a base, sendo que este, geralmente, é limitado ou inexistente (OLIVEIRA; PEDRYCZ, 2007, tradução nossa).

O algoritmo Unsupervised Robust C-Prototypes pode ser utilizado quando a quantidade exata de *clusters* é desconhecida. Sendo necessário o usuário informar o número máximo de *clusters* que poderá ser encontrado na base, pois por meio da similaridade entre os grupos, o algoritmo determina o número ótimo de *clusters* (FRIGUI; KRISHAPURAM, 1996, tradução nossa).

Os algoritmos RCP e URCP quando aplicados em bases de dados sem ruídos tendem a ter resultados muito próximos de outros algoritmos como o FCM, ou GK, porém quando aplicados em bases contaminadas com ruídos ou *outliers*, estes algoritmos conseguem resultados muito superiores devido a utilização de estimadores robustos que diminuem a influência dos ruídos e *outliers* no processo (FRIGUI; KRISHAPURAM, 1996, tradução nossa).

A Figura 14 mostra o resultado encontrado pelo RCP, sendo o número de *clusters* 15, apresentando exatamente esta quantidade de grupos. A Figura 15 apresenta o resultado encontrado pelo URCP com a quantidade de *clusters* definida também como 15, em seu resultado percebe-se que o algoritmo encontrou como sendo 6 o número ótimo de *clusters*.

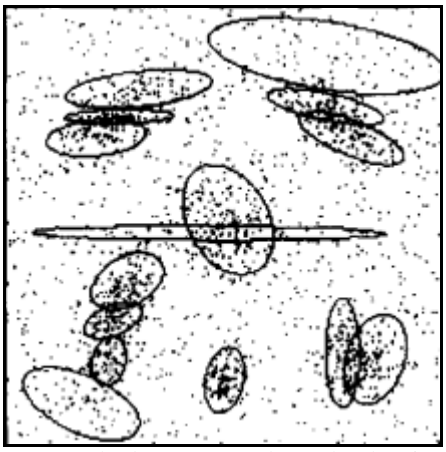

Figura 14. Resultados encontrados pelo algoritmo RCP
Fonte: FRIGUI, H.; KRISHNAPURAM, R. (1996)

Figura 15. Resultados encontrados pelo algoritmo URCP
Fonte: FRIGUI, H.; KRISHNAPURAM, R. (1996)

Considerando que o objetivo desta pesquisa é a implementação dos algoritmos RCP e URCP, estes serão descritos detalhadamente a seguir.

4 OS ALGORITMOS RCP E URCP

Hichem Frigui e Raghu Krishnapuram, em 1996, publicaram o artigo *A Robust Algorithm for Automatic Extraction of an Unknown Number of Clusters from Noisy Data*, na *Pattern Recognition Letters*, neste artigo foram apresentados os algoritmos Robust C-Prototypes e Unsupervised Robust C-Prototypes.

A vantagem destes algoritmos são as funções de peso e perda[13] utilizadas para diminuir a interferência de ruídos e *outliers* no processo de clusterização. Em geral, o resultado destas funções para um dado normal tem valor próximo de 1, enquanto ruídos e *outliers* devem possuir valores menores. Considerando que estas funções podem ser combinadas com outras medidas para a identificação de *clusters* de diferentes formas (FRIGUI; KRISHAPURAM, 1996, tradução nossa).

A seguir os algoritmos RCP e URCP são descritos separadamente, para facilitar seu entendimento.

4.1 RCP

O artigo apresentado descreve o RCP para a identificação de *clusters* em forma de elipse, utilizando uma matriz de covariância, sendo esta robusta, devido a utilização de funções responsáveis por diminuir a influência de dados ruidosos (OLIVEIRA; PEDRYCZ, 2007, tradução nossa).

Na execução do RCP o usuário deve informar o número de *clusters* a serem encontrados na base de dados, o valor do elemento *fuzzyficador*, e o valor que define a abrangência da função de peso. O próximo passo do algoritmo é a inicialização das matrizes

[13] Funções utilizadas pelos algoritmos RCP e URCP para diminuir a interferência de ruídos e *outliers* durante a execução da tarefa de clusterização.

de covariância e os protótipos que representam os centros dos *clusters*, considerando que estes valores não interferem no resultado final. Desta forma, o RCP minimiza a seguinte função objetivo (FRIGUI; KRISHAPURAM, 1996, tradução nossa):

$$J(B,U;Z) = \sum_{i=1}^{C}\sum_{j=1}^{N}\left(\mu_{ij}\right)^{m}\rho_i\left(d_{ij}^2\right) \quad (1)$$

Onde:

a) ***C***: número total de *clusters*;

b) ***N***: número total de elementos;

c) ***J***: valor a ser minimizado;

d) ***B***: conjunto de *clusters*;

e) ***U***: matriz de pertinências;

f) ***Z***: conjunto de dados;

g) ***m***: parâmetro de *fuzzyficação*;

h) $\boldsymbol{\mu_{ij}}$: grau de pertinência do i-ésimo *cluster* e o j-ésimo elemento;

i) $\boldsymbol{\rho_i}$: função de perda;

j) $\boldsymbol{d^2_{ij}}$: distância entre o i-ésimo *cluster* e o j-ésimo elemento.

A equação (1) apresenta de forma genérica a função utilizada por algoritmos que utilizam o método de lógica *fuzzy* na tarefa de clusterização. A única diferença é a função de perda (ρ_i), que foi incorporada com o objetivo de obter bons resultados em bases de dados contaminadas por ruídos e *outliers* (BEZDEK et al, 2005, tradução nossa).

Após concluir a inicialização dos protótipos, o algoritmo executa as seguintes etapas:

a) cálculo das distâncias entre elementos e seus grupos;

b) particionar o conjunto de dados;

c) estimar os valores de *T* e *S*;

d) atualizar as função de peso e perda;

e) calcular os graus de pertinência;

f) determinar os valores dos centros dos *clusters*;

g) atualizar as matrizes de covariância *fuzzy* robusta.

A fim de facilitar a compreensão do algoritmo, a seguir, estas etapas são descritas individualmente.

4.1.1 Cálculo das Distâncias entre Elementos e *Clusters*

O RCP aplica a distância de Mahalanobis, na detecção de clusters em forma de elipse. Esta distância utiliza as matrizes de covariância *fuzzy* em seu cálculo, considerando que na primeira iteração do algoritmo são utilizados valores iniciados randomicamente ou uma matriz identidade. A equação (2) define a atualização das distâncias (FRIGUI; KRISHAPURAM, 1996, tradução nossa):

$$d_{ij}^2 = \left(x_j - c_i\right)^T M_i \left(x_j - c_i\right) \tag{2}$$

Onde:

a) $\boldsymbol{d^2_{ij}}$: distância entre o i-ésimo *cluster* e o j-ésimo elemento;

b) $\boldsymbol{x_j}$: j-ésimo elemento;

c) $\boldsymbol{c_i}$: centro do i-ésimo *cluster*;

d) $\boldsymbol{M_i}$: matriz de covariância modificada do i-ésimo *cluster*.

A distância é calculada pela multiplicação da subtração entre o vetor de dados e o centro dos *clusters* pela matriz de covariância modificada. O resultado é multiplicado pela mesma subtração em sua forma transposta.

A matriz de covariância modificada M_i é definida pela equação (3) (FRIGUI; KRISHAPURAM, 1996, tradução nossa):

$$M_i = \left|C_i\right|^{\frac{1}{n}} C_i^{-1} \tag{3}$$

Onde:

a) $\boldsymbol{M_i}$: matriz de covariância modificada do i-ésimo *cluster*;

b) $\boldsymbol{C_i}$: matriz de covariância *fuzzy* robusta do i-ésimo *cluster*;

c) $\boldsymbol{n}$: número de dimensões da base de dados.

A multiplicação entre o determinante da matriz de covariância *fuzzy* robusta elevado a 1 sobre o número de dimensões[14] da base de dados, e a matriz de covariância *fuzzy* robusta inversa resultam na matriz de covariância *fuzzy* modificada, utilizada no cálculo da distância de Mahalanobis.

4.1.2 Particionamento do Conjunto de Dados

Antes de efetuar a estimação de *T* e *S* é necessário atribuir os elementos a grupos considerando a seguinte propriedade (4) (FRIGUI; KRISHAPURAM, 1996, tradução nossa):

$$\mathrm{Z}_i = \left\{ x_j \middle| d_{ij}^2 \leq d_{kj}^2 \ \forall k \neq i \right\} \tag{4}$$

Onde:

a) $\boldsymbol{Z_i}$: conjunto de elementos do i-ésimo *cluster*;

b) $\boldsymbol{x_j}$: j-ésimo elemento;

c) $\boldsymbol{d^2_{ij}}$: distância entre o i-ésimo *cluster* e o j-ésimo elemento;

d) $\boldsymbol{d^2_{kj}}$: distância entre o k-ésimo *cluster* e o j-ésimo elemento;

O j-ésimo elemento *x* pertence ao i-ésimo conjunto *Z* se a distância entre o i-ésimo *cluster* e o j-ésimo elemento for menor ou igual à distância entre o k-ésimo *cluster* e o j-ésimo elemento, sempre que *k* for diferente de *i*.

[14] Número de atributos da base de dados utilizado no processo.

4.1.3 Estimação dos Valores de *T* e *S*

Os valores de *T* e *S* são muito importantes, pois definem as funções de peso e perda, seu cálculo é realizado utilizando os grupos criados na etapa anterior (FRIGUI; KRISHAPURAM, 1996, tradução nossa).

$$T_i = \underset{x_j \in Z_i}{Med}\left(d_{ij}^2\right) \tag{5}$$

Onde:

a) T_i: i-ésimo valor de T;

b) Med: mediana do conjunto;

c) d^2_{ij}: distância entre o i-ésimo *cluster* e o j-ésimo elemento.

A mediana das distâncias entre o i-ésimo *cluster* e o j-ésimo elemento, quando o j-ésimo elemento *x* pertencer ao conjunto *Z* resulta na estimação *T* de seu conjunto.

O valor de *S* é calculado por meio do desvio mediano absoluto da distância entre o i-ésimo *cluster* e o j-ésimo elemento, quando o j-ésimo elemento pertencer ao conjunto *Z*, seu resultado é multiplicado pelo valor 1.418.

$$S_i = 1.418 \times \underset{x_j \in Z_i}{MAD}\left(d_{ij}^2\right) \tag{6}$$

Onde:

a) S_i*:* i-ésimo valor de *S*;

b) *MAD*[15]: *Median of Absolute Deviations* ou desvio mediano absoluto (DMA) das distâncias;

c) d^2_{ij}*:* distância entre o i-ésimo cluster e o j-ésimo elemento.

[15] Desvio mediano absoluto (DMA, em português) calcula o desvio dos valores de um grupo em relação à mediana.

4.1.4 Atualização das Funções de Peso e Perda

Após a estimação dos valores de *T* e *S* é possível calcular as funções de peso e perda. Estas são responsáveis pela robustez do algoritmo, ou seja, são elas que diminuem a interferência de ruídos e *outliers*. A função de peso é apresentada em (7) (FRIGUI; KRISHAPURAM, 1996, tradução nossa).

$$\omega_i\left(d_{ij}^2\right)=\begin{cases}1-\dfrac{d_{ij}^4}{2T_i^2}, se\ d_{ij}^2 \in [0,T_i] \\ \dfrac{\left[d_{ij}^2-(T_i+\alpha S_i)\right]^2}{2\alpha^2 S_i^2}, se\ d_{ij}^2 \in (T_i, T_i+\alpha S_i] \\ 0, se\ d_{ij}^2 > T_i+\alpha S_i\end{cases} \quad (7)$$

Onde:

a) $w_i(d^2_{ij})$: função de peso do i-ésimo *cluster* ;

b) T_i: i-ésimo valor de T;

c) S_i: i-ésimo valor de S;

d) α**:** constante que define a região da função de peso.

A função de peso é calculada considerando três intervalos para o valor da distância. Assim, quando o valor da distância estiver até o mediano de seu conjunto, a função de peso desta distância irá sofrer poucas modificações, e quando seu valor for muito alto, a função de peso terá valor 0, fazendo com que o elemento desta distância não seja utilizado na atualização dos protótipos referentes ao centro do *cluster* ou sua matriz de covariância.

A constante α determina a região que a função de peso irá abranger, portanto quanto maior o seu valor maior será sua região, assim esta será mais *fuzzy*, e muitos *outliers* terão peso não zero afetando a estimação dos parâmetros (FRIGUI; KRISHAPURAM, 1996, tradução nossa).

A função de perda é apresentada em (8).

$$\rho_i\left(d_{ij}^2\right)=\begin{cases} d_{ij}^2-\dfrac{d_{ij}^6}{6T_i^{\,2}},\ se\ d_{ij}^2\in[0,T_i] \\ \dfrac{\left[d_{ij}^2-(T_i+\alpha S_i)\right]^3}{6\alpha^2 S_i^2}+\dfrac{5T_i+\alpha S_i}{6},\ se\ d_{ij}^2\in(T_i,T_i+\alpha S_i] \\ \dfrac{5T_i+\alpha S_i}{6}+K_i\ se\ d_{ij}^2>T_i+\alpha S_i \end{cases} \tag{8}$$

Onde:

a) $\boldsymbol{\rho(d^2_{ij})}$: função de perda do i-ésimo *cluster*;

b) $\boldsymbol{T_i}$: i-ésimo valor de T;

c) $\boldsymbol{S_i}$: i-ésimo valor de S;

d) $\boldsymbol{\alpha}$**:** constante que define a região da função de peso;

e) $\boldsymbol{K_i}$**:** constante do i-ésimo *cluster*.

Assim como a função de peso, a função de perda também utiliza intervalos para o valor da distância. Percebe-se que todas as distâncias que ficarem no último intervalo terão o mesmo valor para a função de peso, ou seja, quando as distâncias forem altas, os elementos ficarão entre os grupos, mais distantes dos centros dos protótipos.

A constante *K* é adicionada para impedir que todos os elementos que tiverem sua distância no terceiro intervalo da função de perda pertençam ao menor *cluster*. A equação (9) descreve esta constante (FRIGUI; KRISHAPURAM, 1996, tradução nossa).

$$K_i=\max_{1\le j\le C}\left\{\frac{5T_j+\alpha S_j}{6}\right\}-\frac{5T_i+\alpha S_i}{6} \tag{9}$$

Onde:

a) $\boldsymbol{K_i}$: constante do i-ésimo *cluster*;

b) $\boldsymbol{T_j}$: i-ésimo valor de T;

c) $\boldsymbol{S_j}$: i-ésimo valor de S;

d) $\boldsymbol{\alpha}$: constante que define a região da função de peso.

A $\frac{5T_j + \alpha S_j}{6}$ é efetuado para todos os *clusters*, sendo que o maior valor encontrado será subtraído pelo mesmo cálculo do *cluster* atual.

4.1.5 Atualização dos Graus de Pertinência

O cálculo dos graus de pertinência, apresentado na equação (10), representa quanto um elemento pertence a um determinado *cluster* (FRIGUI; KRISHAPURAM, 1996, tradução nossa).

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{C} \left[\frac{\rho_i \left(d_{ij}^2 \right)}{\rho_k \left(d_{kj}^2 \right)} \right]^{\frac{1}{(m-1)}}} \tag{10}$$

Onde:

a) $\boldsymbol{\mu_{ij}}$: grau de pertinência do i-ésimo *cluster* e o j-ésimo elemento;

b) $\boldsymbol{\rho_i}$: função de perda do i-ésimo *cluster*;

c) $\boldsymbol{m}$: parâmetro de *fuzzyficação*;

d) $\boldsymbol{d^2_{ij}}$: distância entre o i-ésimo *cluster* e o j-ésimo elemento;

e) $\boldsymbol{d^2_{ik}}$: distância entre o i-ésimo *cluster* e o k-ésimo elemento.

O cálculo executa o somatório da função de perda da distância do atual *cluster* dividido por todas as outras funções de perda relacionadas às outras distâncias, o resultado é elevado a 1 dividido pelo grau de *fuzzificação* menos 1. O grau de pertinência do elemento é o valor 1 dividido pela soma obtida.

Os valores guardados na matriz de pertinências devem respeitar a seguinte propriedade (11).

$$\sum_{i=1}^{C} \mu_{ij} = 1, \forall j, \textit{sendo } \mu_{ij} \in [0,1] \tag{11}$$

Onde:

a) $\boldsymbol{C}$: número total de *clusters;*

b) $\boldsymbol{\mu_{ij}}$: grau de pertinência entre o i-ésimo *cluster* e o j-ésimo elemento.

Esta propriedade determina que cada pertinência de um elemento *j* em relação aos *C clusters* deve estar entre 0 e 1, e que a soma das pertinências de um determinado elemento em relação a todos os *clusters* deve ser igual a 1.

4.1.6 Cálculo dos Centros dos Clusters

Os centros dos *clusters* podem ser um elemento da base de dados ou um valor calculado pelo método (BEZDEK et al, 2005, tradução nossa). O RCP calcula os centros dos *clusters* utilizando a equação (12) (FRIGUI; KRISHAPURAM, 1996, tradução nossa).

$$c_i = \frac{\sum_{j=1}^{N} \left(\mu_{ij}\right)^m w_{ij} x_j}{\sum_{j=1}^{N} \left(u_{ij}\right)^m w_{ij}} \tag{12}$$

Onde:

a) $\boldsymbol{c_i}$: centro do i-ésimo *cluster;*

b) $\boldsymbol{\mu_{ij}}$: grau de pertinência entre o i-ésimo cluster e o j-ésimo elemento;

c) $\boldsymbol{x_j}$: j-ésimo elemento;

d) $\boldsymbol{w_{ij}}$:função de peso do i-ésimo *cluster* e do j-ésimo elemento.

A equação (13) utiliza as funções de peso para definir os elementos que não interferem na construção do centro, considerando que o cálculo é realizado para cada *cluster* a ser encontrado na base de dados.

4.1.7 Atualização das Matrizes de Covariância *Fuzzy* Robusta

Assim como o GK e o GG, o RCP utiliza uma matriz de covariância. A diferença é a função de peso que foi incorporada ao cálculo da matriz. A equação (13) apresenta o cálculo da matriz de covariância *fuzzy* robusta (FRIGUI; KRISHAPURAM, 1996, tradução nossa).

$$C_i = \frac{\sum_{j=1}^{N} (\mu_{ij})^m \omega_{ij} (x_j - c_i)(x_j - c_i)^T}{\sum_{j=1}^{N} (u_{ij})^m \omega_{ij}} \quad (13)$$

Onde:

a) $\boldsymbol{C_i}$: matriz de covariância *fuzzy* do i-ésimo *cluster;*

b) $\boldsymbol{N}$: número total de elementos;

c) $\boldsymbol{\mu_{ij}}$: grau de pertinência entre o i-ésimo cluster e o j-ésimo elemento;

d) $\boldsymbol{m}$: *fuzzyficador;*

e) $\boldsymbol{c_i}$: centro do i-ésimo *cluster;*

f) $\boldsymbol{x_j}$: j-ésimo elemento;

g) $\boldsymbol{w_{ij}}$:função de peso do i-ésimo *cluster* e do j-ésimo elemento.

O algoritmo continua até que os centros dos *clusters* e as matrizes de covariância se estabilizem, ou seja, até que não haja mudanças significativas em seus valores, ou até que o número de iterações informado pelo usuário seja alcançado.

4.2 URCP

A diferença entre o RCP e URCP é que este pode ser utilizado quando o número de *clusters* é desconhecido, para isto o URCP utiliza uma medida de similaridade, considerando que o usuário deve informar como parâmetro de inicialização do algoritmo o

número máximo de *clusters* a serem encontrados na base de dados, pois a medida compara cada *cluster* com todos os outros e os que satisfazerem a função se tornam um único *cluster* (FRIGUI; KRISHAPURAM, 1996, tradução nossa).

O URCP possui as mesmas propriedades do RCP em relação a forma dos *clusters* em elipse e também em sua característica de diminuir a interferência de ruídos e *outliers* no processo, essas vantagens resultam em maior processamento computacional, se comparado a outros algoritmos baseados no FCM (OLIVEIRA; PEDRYCZ, 2007, tradução nossa).

Os parâmetros de inicialização do algoritmo são os mesmos que no RCP, diferenciado apenas que deve-se informar o maior número possível de *clusters* a serem encontrados na base de dados, como também o valor de erro para o cálculo da similaridade. Após definir os parâmetros de entrada, o URCP executa o RCP, depois ele executa uma versão modificada no RCP denominada de Unconstrained Robust C-Prototypes, ao final a função de similaridade é aplicada em todos os grupos encontrados (FRIGUI; KRISHAPURAM, 1996, tradução nossa).

O algoritmo Unconstrained RCP é aplicado nos resultados encontrados pelo RCP, sendo que a única diferença é que os graus de pertinência (μ) possuem valor 1 de todos os elementos para todos os *clusters*. Isto significa que todos os elementos são compartilhados completamente entre todos os *clusters*, e, portanto, somente as funções de peso (w) distinguirão entre dados capazes de gerar padrões daqueles que produzem *outliers* ou ruídos (FRIGUI; KRISHAPURAM, 1996, tradução nossa).

As equações utilizadas pelo Unconstrained RCP são as mesmas do RCP, porém as equações (12) e (13) não terão o grau de pertinência em sua fórmula, devido seu valor ser 1. O próximo passo do URCP é calcular os graus de pertinência utilizando (10), já que estes são necessários para a medida de similaridade (FRIGUI; KRISHAPURAM, 1996, tradução nossa).

Antes de executar o cálculo da similaridade é necessário distribuir os dados segundo (14) (FRIGUI; KRISHAPURAM, 1996, tradução nossa).

$$G_i = \{x_k \in X \mid \omega_{ik} > 0\} \tag{14}$$

Onde:

a) G_i: i-ésimo conjunto;

b) x_k: k-ésimo elemento;

c) w_{ik}: função de peso do i-ésimo *cluster* e o k-ésimo elemento.

Quando a função de peso do i-ésimo *cluster* e o k-ésimo elemento for maior que o valor 0, este elemento pertencerá ao conjunto G_i, o mesmo cálculo é efetuado para o conjunto G_j.

A equação (15) apresenta a condição formal para a união entre os conjuntos G_i e G_j definidos utilizando (14) (FRIGUI; KRISHAPURAM, 1996, tradução nossa).

$$G_{ij} = G_i \cup G_j = \{x_k \in X \mid \omega_{ik} > 0 \, OR \, \omega_{jk} > 0\} \tag{15}$$

Onde:

a) G_i: i-ésimo conjunto;

b) G_j: j-ésimo conjunto;

c) G_{ij}: união entre o i-ésimo e o j-ésimo conjunto;

d) x_k: k-ésimo elemento;

e) w_{ik}: função de peso do i-ésimo *cluster* e o k-ésimo elemento;

f) w_{jk}: função de peso do j-ésimo *cluster* e o k-ésimo elemento.

Um elemento pertence à união dos conjuntos se alguma das funções de peso do i-ésimo *cluster* e o k-ésimo elemento ou j-ésimo *cluster* e o k-ésimo elemento for maior que o valor 0.

A equação (16) apresenta a cardinalidade do conjunto, sendo que o cálculo é efetuado para cada grupo (FRIGUI; KRISHAPURAM, 1996, tradução nossa).

$$| G_i | = \sum_{x_k \in G_i} \mu_{ik} \tag{16}$$

Onde:

a) $|G_i|$: cardinalidade do *cluster*;

b) μ_{ik}: grau de pertinência entre o i-ésimo *cluster* e o k-ésimo elemento;

c) x_k: k-ésimo elemento;

d) G_i: i-ésimo conjunto.

O somatório das pertinências entre o i-ésimo *cluster* e o k-ésimo elemento, para cada elemento pertencente a um conjunto, representa a cardinalidade do conjunto.

A seguir é apresentada a equação utilizada para calcular a similaridade entre os *clusters*, sendo que seu valor será 1 para *clusters* idênticos e 0 para conjuntos totalmente distintos. Este cálculo é efetuado para cada par de *clusters i* e *j* (FRIGUI; KRISHAPURAM, 1996, tradução nossa).

$$S_{ij} = 1 - \frac{\sum_{x_k \in G_{ij}} \left| \mu_{ik} - \mu_{jk} \right|}{\left| G_i \right| + \left| G_j \right|} \tag{17}$$

Onde:

a) S_{ij}: valor que representa a similaridade entre um par de *clusters*;

b) μ_{ik}: grau de pertinência entre o i-ésimo *cluster* e o k-ésimo elemento;

c) μ_{jk}: grau de pertinência entre o i-ésimo *cluster* e o k-ésimo elemento;

d) x_k: k-ésimo elemento;

e) G_{ij}: união entre o i-ésimo e o j-ésimo conjunto;

f) $|G_i|$: cardinalidade do i-ésimo conjunto;

g) $|G_j|$: cardinalidade do j-ésimo conjunto.

Após o cálculo da função de similaridade, se seu valor for maior ou igual ao valor 1 menos a taxa de erro, então estes dois *clusters* se tornarão um. A cada fusão é atualizado o

valor do número de *clusters*, e o processo é repetido até não ocorrer mais fusões entre os grupos.

Compreendido as funcionalidades dos algoritmos RCP e URCP, são apresentados alguns exemplos de aplicações destes algoritmos, para a tarefa de clusterização, no método de lógica *fuzzy*.

5 TRABALHOS CORRELATOS

Algoritmos que utilizam o método de lógica *fuzzy* no processo de clusterização são amplamente utilizados em diversas áreas de pesquisa. A seguir são apresentados alguns casos que utilizam este método e principalmente os algoritmos Robust C-Prototypes e Unsupervised Robust C-Prototypes na resolução de problemas.

5.1 COMPARAÇÃO DA ANÁLISE DE IMAGENS PARA RECONHECIMENTO DE CARACTERES MANUSCRITOS

Este artigo foi publicado por Olarik Surinta e Chatklaw Jareanpon, em 2006, no livro *Intelligent Information Processing III*, e compara o uso do algoritmo Robust C-Prototypes e do algoritmo *back-propagation* combinado com descritores Fourier[16] para o reconhecimento dos 44 caracteres Thai[17]. A Figura 16 mostra um exemplo de caractere (SURINTA; JAREANPON, 2006, tradução nossa).

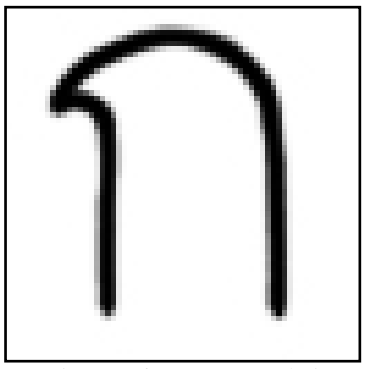

Figura 16. Caractere Thai
Fonte: SURINTA, O.; JAREANPON, C. (2006)

[16] Desenvolvida por Jean-Baptiste-Joseph Fourier, em 1807, é uma função matemática utilizada para descrever o padrão senoidal encontrado em diversas áreas, como por exemplo, em imagens e sinais (GRIFFITHS; DE HASETH, 2007, tradução nossa).
[17] Caracteres da língua oficial da Tailândia.

O processo de reconhecimento inicia com o pré-processamento das imagens, deixando-as em preto e branco. O próximo passo é a binarização da imagem, fazendo que onde o valor for 0 referencie cada *pixel* do objeto, enquanto o valor 1 seria o fundo branco. Após isto, é utilizada uma técnica para reconhecer a borda do objeto, para assim utilizar os descritores do Fourier para identificar e descrever o espaço utilizado pelo objeto na imagem. Enfim os algoritmos foram treinados utilizando uma base de 4400 caracteres, sendo que a base de testes continha 440 e que estes foram criados por 100 pessoas diferentes (SURINTA; JAREANPON, 2006, tradução nossa).

O sistema desenvolvido mostrou que o algoritmo *back-propagation* levou 2,45 horas e reconheceu 88% da base de testes, enquanto o RCP levou 1,5 horas e reconheceu 91,5% da base (SURINTA; JAREANPON, 2006, tradução nossa).

5.2 ANALISANDO MOVIMENTOS UTILIZANDO UNSUPERVISED FUZZY C-PROTOTYPES

Em 1998, Supot Nitsuwat e Jessé S. Jin, na *Proceedings of the Pan-Sydney Area Workshop on Visual Information Processing*, apresentaram este artigo que utiliza o algoritmo Unsupervised Robust C-Prototypes para a segmentação de vídeos com base em movimento (NITSUWAT; JIN, 1998, tradução nossa).

Segmentação com base em movimento é um processo de rotulagem de *pixels* associados com diferentes objetos em movimento, considerando que em um vídeo as características dos objetos em movimento são as partes de interesse (NITSUWAT; JIN, 1998, tradução nossa).

O processo inicia calculando os vetores de fluxo óptico[18] em uma série de imagens, em seguida os parâmetros de movimento das imagens são estimados por meio de regressão de forma a especificar o espaço dos movimentos, sendo que o algoritmo URCP foi aplicado nesses parâmetros (NITSUWAT; JIN, 1998, tradução nossa).

O algoritmo URCP foi escolhido devido a característica de lidar com dados ruidosos, que são muito presentes em imagens, e devido a identificação do número ótimo de *clusters* na identificação das regiões que apresentam objetos em movimento (NITSUWAT; JIN, 1998, tradução nossa).

Os testes foram realizados tanto em seqüências de imagem sintéticas quanto reais, sendo que os resultados foram satisfatórios. A Figura 17 mostra os resultados para a segmentação das imagens sintéticas, onde, respectivamente, tem-se a imagem original (a), o fluxo óptico encontrado (b) e os resultados dos algoritmos RCP (c), unconstrained RCP (d) e URCP (e) (NITSUWAT; JIN, 1998, tradução nossa).

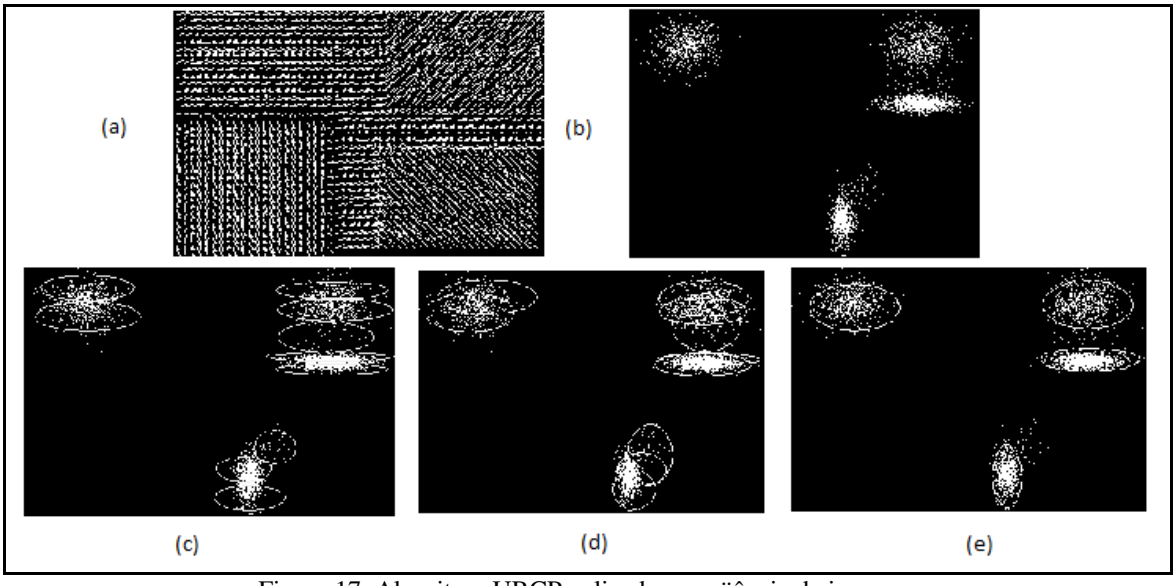

Figura 17. Algoritmo URCP aplicado a seqüência de imagens
Fonte: NITSUWAT, S; JIN, J. S. (1998)

[18] Técnica para identificação de movimento de objetos em uma seqüência de imagens (NITSUWAT; JIN, 1998, tradução nossa).

5.3 UTILIZANDO NAVEGAÇÃO PARA MELHORAR A RECUPERAÇÃO DE CONTEÚDO

Jin, J. S. et al, em 2001,no jornal, *Journal of Visual Communication and Image Representation*, apresentou neste artigo uma alternativa para a busca de imagens baseada em estruturas de árvore balanceada para acesso multidimensional. O artigo apresenta a árvore multidimensional SS-tree e uma solução para o problema de sobreposição de nós, sendo que isto causa ineficiência na recuperação de informações em níveis muito elevados (JIN, 2001, tradução nossa).

A árvore irá guardar os dados em estruturas esféricas, sendo que esta estrutura é a que menos possibilita causar sobreposição nos nós. Após isto é utilizado o algoritmo Unsupervised Robust C-Prototypes para a clusterização dos dados. O resultado foi um maior grau de liberdade e a minimização das variâncias entre partições de resultado (JIN et al, 2001, tradução nossa).

A fim de atualizar os índices da árvore e reduzir a área de sobreposição, os novos dados são adicionados ao *cluster* (nó) cujo centro está mais próximo. Em seguida eles apresentam a estratégia de busca na estrutura baseando-se nos centros dos *clusters* mais próximos da informação que se está procurando. Outras heurísticas foram testadas, porém, esta foi a que apresentou melhores resultados. A Figura 18 mostra quatro *clusters* e a busca do valor *q* (JIN et al, 2001, tradução nossa).

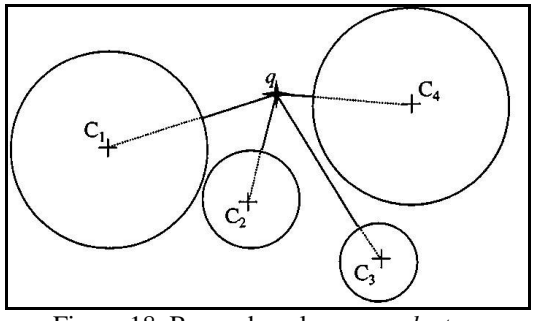

Figura 18. Busca do valor *q* nos *clusters*
Fonte: JIN, J. S. et al (2001)

A estrutura apresenta um sistema rápido e eficiente para a busca de imagens, que não consegue acabar com o problema da sobreposição, mas consegue reduzi-lo consideravelmente, sendo capaz de lidar com milhões de itens (JIN et al, 2001, tradução nossa).

5.4 SEPARAÇÃO INDETERMINADA DE SINAIS EM UMA FREQUÊNCIA DE TEMPO

Em 2008, Lv Yao e Li Shuangtian, no *9th International Conference on Signal Processing*, apresentaram este artigo com uma nova solução para o problema da separação cega de fontes a partir de uma base (*Blind Source Separation - BSS*). A separação é feita sem informações sobre os sinais, fontes ou seus processos, por isso chama-se cega. (YAO; SHUANGTIAN, 2008, tradução nossa).

A separação pode ocorrer com o mesmo número de fontes e sensores, considerado o caso ótimo, pode-se ter mais sensores que fontes, chamado de caso determinado; ou menos sensores que fontes, caso indeterminado (LV; ZHANG, 2006, tradução nossa).

O método resolve o problema da separação indeterminada, onde primeiramente são estimados as freqüências das misturas dos sinais, e então o URCP é aplicado para encontrar a real quantidade de fontes de sinais. Este algoritmo foi escolhido devido a sua capacidade de lidar com dados ruidosos, os quais constantemente são presentes em bases de sinais, e por também determinar o número ótimo de *clusters* (YAO; SHUANGTIAN, 2008, tradução nossa).

O próximo passo realizado foi a separação dos sinais, utilizando os centros encontrados pelo URCP. A nova abordagem mostrou-se eficiente e com bons resultados. A Figura 19 mostra três *clusters* encontrados corretamente pelo algoritmo, sendo que o número

de *clusters* informado como parâmetro foi seis (YAO; SHUANGTIAN, 2008, tradução nossa).

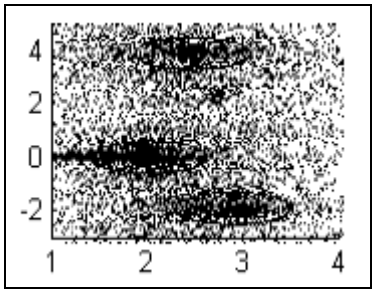

Figura 19. Resultado encontrado pelo URCP
Fonte: YAO, L. SHUANGTIAN, L. (2008)

5.5 MÉTODO ÚNICO PARA SEPARAÇÃO DE FONTES DE SINAIS COM O NÚMERO DE FONTES DESCONHECIDO

Qi Lv e Xian-Da Zhang, em 2006, na *IEEE Signal Processing Letters*, propuseram um novo método para o problema da separação de sinais, onde o número de fontes de sinais é desconhecido. O método proposto consegue lidar com os casos determinados e indeterminados na separação de sinais (LV; ZHANG, 2006, tradução nossa).

O método utiliza o URCP para separar as misturas de sinais, a seguir eles aplicam uma técnica, desenvolvida especificamente para o problema apresentado, sobre os *clusters* encontrados, refinando o processo, afim de selecionar somente os *clusters* necessários e assim reduzir a sua quantidade. Então eles utilizam o número de *clusters* como o número de fontes de sinais e o centro de cada *cluster* como uma estimação correspondente a uma coluna de uma matriz de sinais. O próximo passo é a separação dos sinais. (LV; ZHANG, 2006, tradução nossa).

6 OS ALGORITMOS RCP E URCP NA TAREFA DE CLUSTERIZAÇÃO DA *SHELL* ORION *DATA MINING ENGINE*

O projeto acadêmico da *Shell* Orion *Data Mining Engine* tem como objetivo o desenvolvimento de uma ferramenta gratuita para auxílio ao processo de descoberta de conhecimento em bases de dados, de forma a disponibilizar informações sobre *data mining* e a modelagem matemática de diferentes algoritmos.

Assim, esta pesquisa objetiva o aumento das funcionalidades desta ferramenta com a implementação dos algoritmos RCP e URCP no módulo de clusterização, utilizando o método de lógica *fuzzy*.

A seguir é descrita a base de dados empregada nos testes dos algoritmos implementados.

6.1 BASE DE DADOS

A base de dados foi a mesma utilizada pelo Bacharel em Ciência da Computação José Márcio Cassetari Júnior, em 2008, para apresentar o método de lógica *fuzzy* pelo algoritmo Gustafson-Kessel no processo de clusterização.

A base foi fornecida pelo professor Dr. Felipe Dal Pizzol, do curso de Medicina da UNESC, e contém registros de pacientes com sepse da Unidade de Terapia Intensiva (UTI) do Hospital de Clínicas da cidade de Porto Alegre, Rio Grande do Sul.

A sepse, também conhecida como infecção generalizada, é um conjunto de manifestações em todo o organismo, produzidas por uma infecção grave (CECIL, GOLDMAN, AUSIELO, 2005).

A base possui 96 registros, sendo que cada registro tem 45 atributos contendo informações do paciente e de exames realizados durante a internação (CASSETARI JÚNIOR, 2008).

Devido a atributos nominais existentes na base, e a incapacidade dos algoritmos baseados no FCM, caso do GK, RCP e URCP, em lidar com este tipo de dados, foi necessário pré-processar a base. Aos atributos nominais foram atribuídos valores identificadores, e os valores nulos foram substituídos por 0. A Tabela 2 apresenta a descrição dos atributos da base de dados (CASSETARI JÚNIOR, 2008):

Tabela 2. Descrição da base de dados de paciente com sepse

Atributo	Descrição	Valor
caso	Número do caso registrado pelo hospital.	Número inteiro
idade	Idade do paciente.	Número inteiro
sexo	Sexo do paciente	1 para masculino e 2 para feminino
gravidad	Gravidade em que o paciente chegou ao hospital.	1 para leve, 2 para grave e 3 para estado de choque
ira	Se foi registrado insuficiência renal aguda (perda das funções dos rins) durante a internação.	1 para sim e 2 para não
hepático	Se o paciente possuía hepatite.	1 para sim e 2 não
vm	Se o paciente utilizou ventilação mecânica (aparelho que auxilia a respiração) durante a internação.	1 para sim e 2 não
fio2, fio21, fio22 e fio23	Se utilizou ventilação mecânica, qual a tração inspirada de oxigênio pelo paciente (em porcentagem). Os quatro atributos representam respectivamente os valores dos 4 primeiros dias de internação.	Número decimal
neurolog	Se o caso é neurológico, ou seja, se afetou o sistema nervoso central.	1 para sim e 2 não
vasopres	Se o paciente utilizou vasopressor (substância com o objetivo de aumentar a pressão arterial).	1 para sim e 2 não
nora, nora1, nora2 e nora3	Se utilizou vasopressor, qual a quantidade de noradrenalina utilizada (substância que mantém a pressão arterial estabilizada). Os quatro atributos representam respectivamente os valores dos quatro primeiros dias de internação.	Número decimal
apacheII	Grau de gravidade em que o paciente chegou na UTI. Este valor determina a previsão de sobrevivência do doente.	Número inteiro
mods, mods2, mods3 e mods4	Grau de disfunção de múltiplos órgãos do paciente. Os quatro atributos representam respectivamente os valores dos quatro primeiros dias de internação.	Número inteiro
tempodesfecho	Tempo de desfecho do paciente (em dias).	Número inteiro
Desfecho	Desfecho do paciente.	1 para óbito e 2 para cura
tba1, tba2, tba3 e tba4	Marcador de oxidação de lipídios (alterações dos lipídios pelo oxigênio).	Número decimal
carbonyl, carb2, carb3 e carb4	Marcador de oxidação das proteínas (alterações das proteínas pelo oxigênio).	Número decimal
sodcategoria	Categoria do superóxido-dismutase (enzima que realiza a dismutação do radical superóxido em peróxido de hidrogênio e oxigênio).	Número decimal
sod, sod2, sod3 e sod4	Marcador do superóxido-dismutase. Os quatro atributos representam respectivamente os valores dos quatro primeiros dias de internação.	Número decimal
cat1, cat2, cat3 e cat4	Quantidade de catalase (enzima que transforma o peróxido de hidrogênio em água e oxigênio) produzida pelo paciente. Os quatro atributos representam respectivamente os valores dos quatro primeiros dias de internação.	Número decimal
xo1, xo2, xo3 e xo4	Quantidade xantina oxidase (enzima que catalisa hipoxantina com oxigênio, produzindo ácido úrico e o radical superóxido) produzida pelo paciente. Os quatro atributos representam respectivamente os valores dos quatro primeiros dias de internação.	Número decimal

Fonte: CASSETARI JÚNIOR, J. (2008)

Esta pesquisa adotou uma metodologia com o objetivo de compreender os temas e algoritmos abordados, a fim de desenvolvê-los na *Shell* Orion *Data Mining Engine*.

6.2 METODOLOGIA

As etapas metodológicas desta pesquisa são as seguintes: levantamento bibliográfico, por meio do estudo referente à *data mining*, técnicas de clusterização, o método de lógica *fuzzy* e os algoritmos Robust C-Prototypes e Unsupervised Robust C-Prototypes; modelagem e demonstração matemática dos algoritmos; implementação; testes; e avaliação dos algoritmos.

O algoritmo *Fuzzy* C-Means também foi desenvolvido, devido a sua importância, e ampla utilização na tarefa de clusterização pelo método de lógica *fuzzy*. Desta forma, sua modelagem e os testes efetuados também são apresentados a seguir, sendo que a descrição detalhada do algoritmo encontra-se no Apêndice A.

6.2.1 Modelagem do Módulo dos Algoritmos FCM, RCP e URCP

O desenvolvimento teve início pela modelagem dos processos realizados pelos algoritmos FCM, RCP e URCP na *Shell* Orion *Data Mining Engine*. Na modelagem foram utilizados padrões da *Unified Modeling Language*[19] (UML), sendo que foram desenvolvidos os diagramas de caso de uso, seqüência e atividades, por meio da ferramenta gratuita ArgoUML[20].

Os diagramas de caso de uso apresentam as ações realizadas pelo usuário e pelo sistema (Figura 20):

[19] Linguagem desenvolvida para modelagem visual de classes, relacionamentos e métodos de um sistema, objetiva facilitar a documentação e entendimento de seus módulos (GUEDES, 2008).
[20] Ferramenta disponível em (http://argouml.tigris.org).

a) **informar parâmetros de entrada do algoritmo:** o usuário deve informar a quantidade máxima de iterações, a quantidade de *clusters*, o valor do parâmetro *fuzzificador* e a condição de parada do algoritmo, para o FCM. Na execução do RCP não é necessário informar a condição de parada, pois o algoritmo converge quando seus protótipos estabilizam, porém é necessário informar o valor de α (abrangência da função de peso) e para o URCP é necessário informar, além dos parâmetros utilizados pelo RCP, a taxa de erro (ε) necessária para calcular a similaridade entre os grupos. Em seguida é solicitado ao sistema a execução do algoritmo;

b) **executar o algoritmo:** após definir os parâmetros de entrada o algoritmo executa o processo de clusterização.

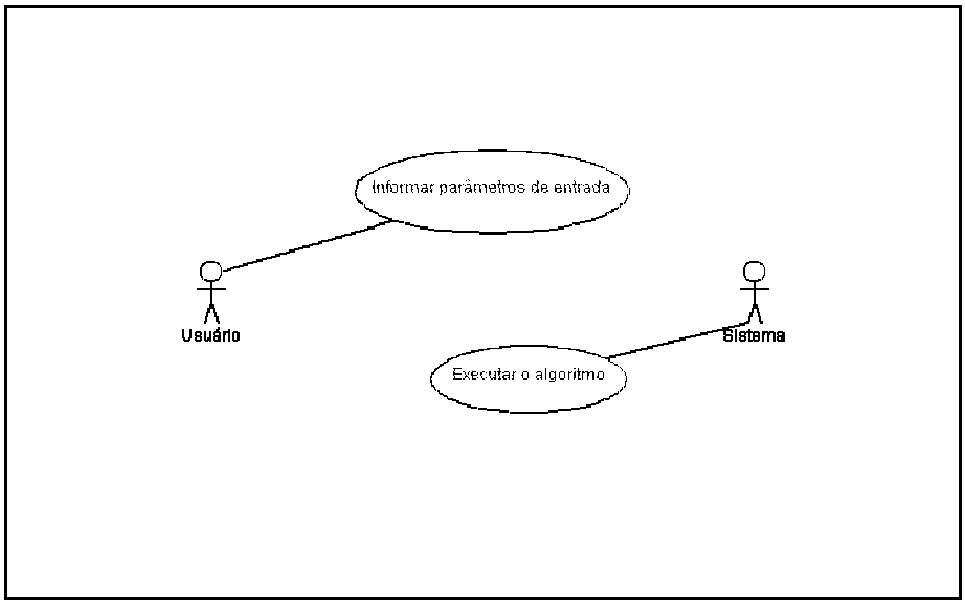

Figura 20. Diagrama de caso de uso dos algoritmos

A Figura 21 demonstra o diagrama de seqüência dos algoritmos. O usuário solicita a interface inicial de um dos algoritmos, onde são informados os parâmetros necessários, após isto tem-se início a execução do algoritmo que gera os resultados a serem apresentados ao usuário.

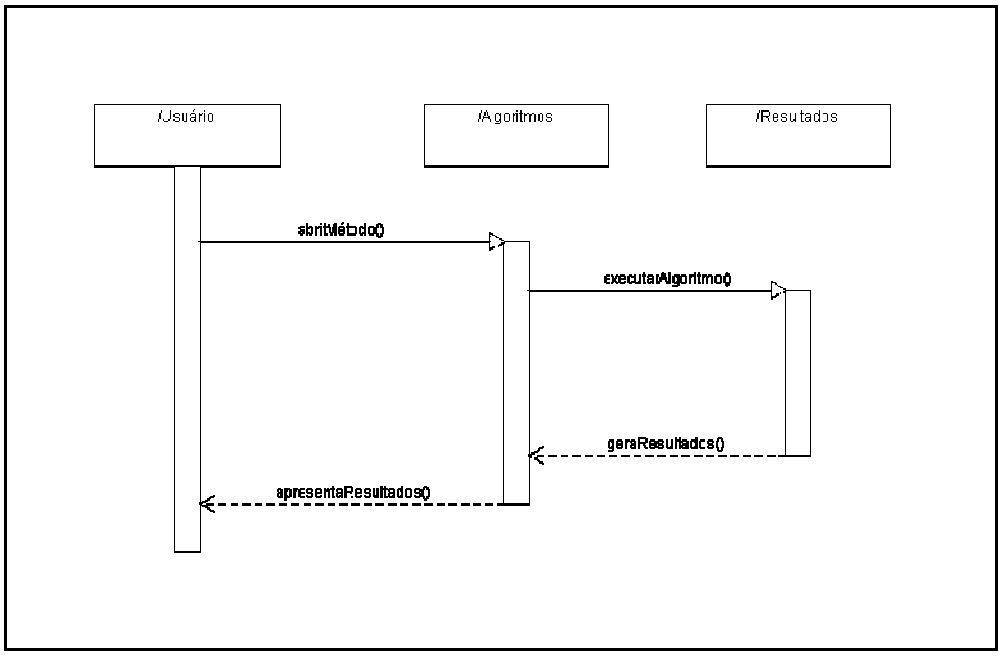

Figura 21. Diagrama de Seqüências dos algoritmos

O diagrama de atividades apresenta o fluxo de tarefas realizado pelo usuário na utilização do sistema (Figura 22):

a) **informar parâmetros de entrada:** o usuário informa os parâmetros necessários para a execução do algoritmo;

b) **solicitar a execução do algoritmo:** solicita ao sistema a execução do algoritmo;

c) **processar o algoritmo:** sistema realiza a execução do algoritmo;

d) **gerar resultados:** sistema gera os resultados da execução do algoritmo;

e) **visualizar resultados:** apresentação ao usuário dos resultados obtidos em forma de texto, grupos e gráfico.

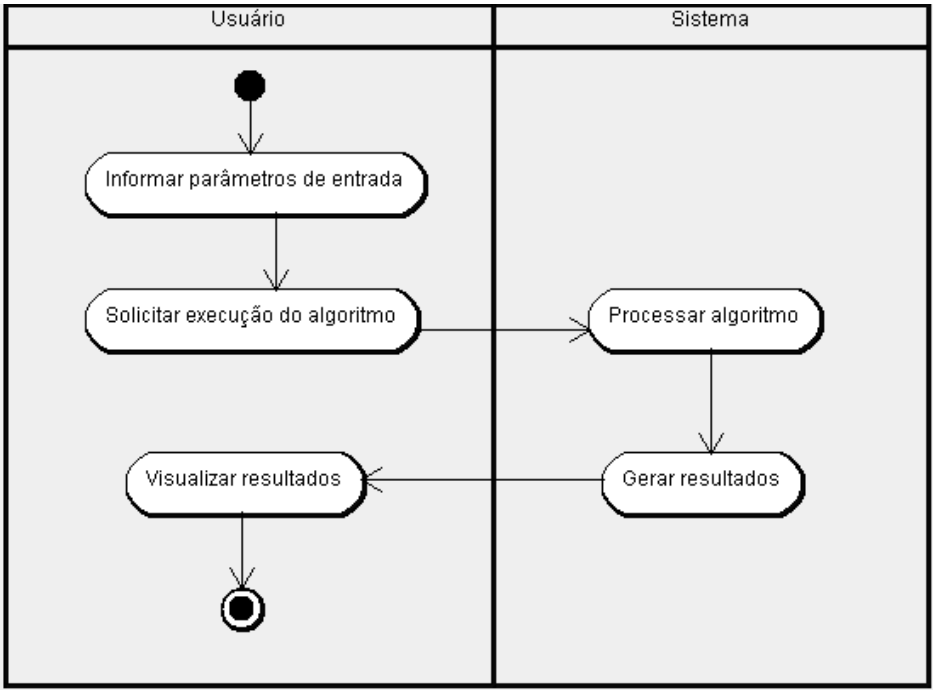

Figura 22. Diagrama de atividades dos algoritmos

6.2.2 Modelagem Matemática

Na modelagem matemática dos algoritmos foi utilizada uma pequena base, com quatro elementos (x_1, x_2, x_3, x_4) e três atributos (a, b, c), gerada randomicamente. A Tabela 3 apresenta a base de dados.

Tabela 3. Base de dados utilizada na modelagem dos algoritmos

Atributos	x_1	x_2	x_3	x_4
a	9	8	7	7,5
b	1	1,8	2	1,5
c	2	3	2,5	4

Sendo que inicialmente será demonstrada a modelagem matemática do algoritmo FCM.

6.2.2.1 Modelagem Matemática do Algoritmo *Fuzzy* C-Means

Os parâmetros do algoritmo utilizados nesta demonstração foram:

a) **quantidade de *clusters*:** define o número de agrupamentos a serem encontrados. O valor utilizado neste caso foi 2, para simplificar a demonstração dos cálculos realizados pelo algoritmo;

a) **parâmetro de *fuzzyficação* (*m*):** para o grau de *fuzzificação* entre os elementos foi definido o valor 2, padrão do algoritmo;

b) **taxa de erro:** define a condição de parada do algoritmo, na exemplificação dos cálculos do FCM foi definido o valor 0,00001.

O algoritmo FCM inicia sua execução gerando randomicamente os graus de pertinência entre os *clusters* e elementos (Tabela 9).

Tabela 4. Pertinências iniciadas randomicamente

Clusters	x_1	x_2	x_3	x_4
1	0.5	0.3	0.8	0.6
2	0.5	0.7	0.2	0.4

Após os graus de pertinência definidos, inicia-se a execução do algoritmo calculando os centros dos *clusters*.

$$c_i = \frac{\sum_{j=1}^{N} (\mu_{ij})^m x_j}{\sum_{j=1}^{N} (u_{ij})^m}$$

Resolvendo a equação tem-se:

$$c_1 = \frac{(\mu_{11})^m x_1 + (\mu_{12})^m x_2 + (\mu_{13})^m x_3 + (\mu_{14})^m x_4}{(\mu_{11})^m + (\mu_{12})^m + (\mu_{13})^m + (\mu_{14})^m}$$

Efetuando a substituição dos valores:

$$c_1 = \frac{(0,5)^2 \cdot \begin{bmatrix} 9 \\ 1 \\ 2 \end{bmatrix} + (0,3)^2 \cdot \begin{bmatrix} 8 \\ 1,8 \\ 3 \end{bmatrix} + (0,8)^2 \cdot \begin{bmatrix} 7 \\ 2 \\ 2,5 \end{bmatrix} + (0,6)^2 \cdot \begin{bmatrix} 7,5 \\ 1,5 \\ 4 \end{bmatrix}}{(0,5)^2 + (0,3)^2 + (0,8)^2 + (0,6)^2}$$

O primeiro passo é efetuar os graus de pertinência elevados ao elemento *fuzzyficador*, no denominador e numerador, no numerador também efetua-se a multiplicação entre o resultado obtido e o valor dos dados.

$$c_1 = \frac{\begin{bmatrix} 2,25 \\ 0,25 \\ 0,5 \end{bmatrix} + \begin{bmatrix} 0,72 \\ 0,162 \\ 0,27 \end{bmatrix} + \begin{bmatrix} 4,48 \\ 1,28 \\ 1,6 \end{bmatrix} + \begin{bmatrix} 2,7 \\ 0,54 \\ 1,44 \end{bmatrix}}{0,25 + 0,09 + 0,64 + 0,36}$$

Efetuando a soma entre os elementos:

$$c_1 = \frac{\begin{bmatrix} 10,15 \\ 2,232 \\ 3,81 \end{bmatrix}}{1,34}$$

Resolvendo a divisão, tem-se o centro do primeiro *cluster*.

$$c_1 = \begin{bmatrix} 7,57436 \\ 1,6657 \\ 2,84327 \end{bmatrix}$$

Utilizando os mesmos cálculos chega-se ao centro do segundo *cluster*.

$$c_2 = \begin{bmatrix} 8,13827 \\ 1,54467 \\ 2,88298 \end{bmatrix}$$

A distância Euclidiana, utilizada pelo FCM, é calculada utilizando a seguinte equação:

$$d_{ij}^2 = \sqrt{\sum_{j=1}^{p} \left(x_j - c_i\right)^2}$$

Reescrevendo a equação, tem-se:

$$d_{11}^2 = \sqrt{(x_1 - c_1)^2 + (x_2 - c_2)^2 + (x_3 - c_3)^2}$$

Substituindo os valores:

$$d_{11}^2 = \sqrt{(9 - 7,57463)^2 + (1 - 1,6657)^2 + (2 - 2,87327)^2}$$

Resolvendo, tem-se:

$$d_{11}^2 = \sqrt{2{,}03168 + 0{,}44316 + 0{,}71111}$$

Em seguir efetua-se a soma dos termos e a sua raiz para obter a distância entre o primeiro elemento e o primeiro *cluster*.

$$d_{11}^2 = 1{,}78493$$

Efetuado os mesmos cálculos para os outros elementos e para o segundo *cluster* chega-se aos valores apresentados na Tabela 10:

Tabela 5. Distâncias entre os elementos e os centros dos *clusters*

Distâncias	x_1	x_2	x_3	x_4
c_1	1,78493	0,47279	0,74818	1,17092
c_2	1,34864	0,31302	1,28437	1,28727

Após calcular as distâncias é possível obter os graus de pertinência atualizados por meio da equação:

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{C} \left[\frac{d_{ij}^2}{d_{kj}^2} \right]^{\frac{1}{(m-1)}}}$$

Reescrevendo a equação para obter a pertinência do primeiro elemento em relação ao primeiro *cluster*:

$$\mu_{11} = \frac{1}{\left[\frac{d_{11}^2}{d_{11}^2} \right]^{\frac{1}{(m-1)}} + \left[\frac{d_{11}^2}{d_{12}^2} \right]^{\frac{1}{(m-1)}}}$$

Substituindo os valores tem-se:

$$\mu_{11} = \frac{1}{\left[\frac{1{,}78493}{1{,}78493} \right]^{\frac{1}{(2-1)}} + \left[\frac{1{,}78493}{1{,}34864} \right]^{\frac{1}{(2-1)}}}$$

Resolvendo, chega-se a primeira pertinência

$$\mu_{11} = \frac{1}{1+1{,}32351}$$

$$\mu_{11} = 0{,}43037$$

Efetuando a atualização das pertinências para os outros elementos e *clusters* chega-se aos valores apresentados na Tabela 11.

Tabela 6. Graus de pertinências encontrados pelo *Fuzzy* C-Means

Clusters	x_1	x_2	x_3	x_4
1	0,43037	0,39834	0,63189	0,52365
2	0,56962	0,60166	0,36809	0,47634

Após a atualização da matriz de pertinências é efetuado o cálculo do erro. Este é apresentado a seguir, sendo a subtração entre as pertinências da iteração atual com o da última iteração, se o valor absoluto obtido for menor ou igual a taxa de erro recebida por parâmetro o algoritmo encerra sua execução.

$$\left\| U^{l} - U^{l-1} \right\| \leq \varepsilon$$

Substituindo os valores:

$$\left\| \begin{bmatrix} 0{,}43037 & 0{,}39834 & 0{,}63189 & 0{,}52365 \\ 0{,}56962 & 0{,}60166 & 0{,}36809 & 0{,}47634 \end{bmatrix} - \begin{bmatrix} 0{,}5 & 0{,}3 & 0{,}8 & 0{,}6 \\ 0{,}5 & 0{,}7 & 0{,}2 & 0{,}4 \end{bmatrix} \right\| \leq 0{,}00001$$

Resolvendo a subtração e o módulo:

$$\left\| \begin{bmatrix} 0{,}06963 & 0{,}09834 & 0{,}16811 & 0{,}07635 \\ 0{,}04962 & 0{,}09834 & 0{,}16809 & 0{,}07634 \end{bmatrix} \right\| \leq 0{,}00001$$

Considerando que nenhum dos valores encontrados foi menor ou igual ao erro, o algoritmo executa a próxima iteração.

6.2.2.2 Modelagem Matemática do Algoritmo RCP

Esta etapa tem como objetivo demonstrar a modelagem matemática do algoritmo RCP durante o processo de clusterização pelo método de lógica *fuzzy*.

A base de dados utilizada foi apresentada na Tabela 3. Desta forma, os parâmetros utilizados na modelagem foram:

b) **quantidade de *clusters*:** foi utilizado o valor 2, como mencionado, a fim de facilitar a demonstração matemática do algoritmo;

c) **parâmetro de *fuzzyficação* (*m*):** foi utilizado o valor 2;

d) ***α*:** define a abrangência da função de peso. Foi definido 3.5, pois a base utilizada apresenta bons dados. Este parâmetro deve ser definido de acordo com a qualidade da base de dados em relação a ruídos e *outliers*, quanto maior o seu valor mais pesos não zero existirão, quanto menor o seu valor menos atributos serão visíveis ao processo de definição dos protótipos de centros dos *clusters* e matrizes de covariância *fuzzy*.

Primeiramente deve-se iniciar os centros dos *clusters* e as matrizes de covariância *fuzzy*. Os centros são iniciados randomicamente, as matrizes de covariância também podem ser iniciadas randomicamente ou define-se uma matriz identidade[21] para início dos cálculos. Na implementação os centros são valores randômicos da base de dados, e as matrizes são identidades. Nesta demonstração tanto os centros como as matrizes foram iniciadas randomicamente, a Tabela 7 apresenta os centros, e a seguir são apresentadas as matrizes de covariância *fuzzy*.

[21] Matriz quadrada de ordem *n* onde os elementos da diagonal principal são iguais a 1, e os demais têm valor 0 (POOLE, MONTEIRO, 2004).

Tabela 7. Centros dos *clusters* iniciados randomicamente

Atributos	c_1	c_2
a	6	3
b	7	6
c	4	1,5

$$C_1 = \begin{bmatrix} 5 & 2 & 1 \\ 1 & 4 & 2 \\ 2 & 3 & 2 \end{bmatrix} \qquad C_2 = \begin{bmatrix} 4 & 3 & 2 \\ 2 & 5 & 5 \\ 3 & 2 & 3 \end{bmatrix}$$

Assim, com os centros e as matrizes de covariância definidos pode-se iniciar o algoritmo RCP com o cálculo da distância de Mahalanobis, definida pela seguinte equação:

$$d_{ij}^2 = \left(x_j - c_i\right)^T M_i \left(x_j - c_i\right)$$

Onde:

$$M_i = \left|C_i\right|^{\frac{1}{n}} C_i^{-1}$$

Considerando a equação acima, inicialmente deve-se encontrar a matriz de covariância modificada M_i, sendo necessário primeiramente calcular o determinante[22] da matriz de covariância.

O determinante de uma matriz quadrada de ordem 3 é definida por:

$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \tag{18}$$

$$\left|A\right| = \left(a \cdot e \cdot i\right) + \left(b \cdot f \cdot g\right) + \left(c \cdot d \cdot h\right) - \left(c \cdot e \cdot g\right) - \left(b \cdot d \cdot i\right) - \left(a \cdot f \cdot h\right)$$

Calculando o determinante da primeira matriz de covariância (C_1), para o primeiro *cluster*, tem-se:

22 Valor numérico associado a uma matriz quadrada (matriz que possui o mesmo número de linhas e colunas), encontrado por meio de uma operação aritmética entre seus elementos (POOLE, MONTEIRO, 2004).

$$C_1 = \begin{bmatrix} 5 & 2 & 1 \\ 1 & 4 & 2 \\ 2 & 3 & 2 \end{bmatrix}$$

$$|C_1| = (5 \cdot 4 \cdot 2) + (2 \cdot 2 \cdot 2) + (1 \cdot 1 \cdot 3) - (1 \cdot 4 \cdot 2) - (2 \cdot 1 \cdot 2) - (5 \cdot 2 \cdot 3)$$

$$|C_1| = 9$$

O próximo passo é calcular a matriz inversa[23] apresentado pela equação (19):

$$A^{-1} = \frac{1}{|A|} adj(A) \tag{19}$$

Onde:

a) |**A**|: determinante da matriz A;

b) ***adj*(A)**: adjunta[24] da matriz A.

A matriz inversa é definida pela sua adjunta dividida pelo seu determinante. Onde uma matriz adjunta é definida por todos os seus determinantes possíveis, conforme equação (20).

$$adj(A) = \begin{bmatrix} \begin{vmatrix} e & f \\ h & i \end{vmatrix} & \begin{vmatrix} c & b \\ i & h \end{vmatrix} & \begin{vmatrix} b & c \\ e & f \end{vmatrix} \\ \begin{vmatrix} f & d \\ i & g \end{vmatrix} & \begin{vmatrix} a & c \\ g & i \end{vmatrix} & \begin{vmatrix} c & a \\ f & d \end{vmatrix} \\ \begin{vmatrix} d & e \\ g & h \end{vmatrix} & \begin{vmatrix} b & a \\ h & g \end{vmatrix} & \begin{vmatrix} a & b \\ d & e \end{vmatrix} \end{bmatrix} \tag{20}$$

Sendo que o determinante de uma matriz quadrada de ordem 2 é definida por:

$$B = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \tag{21}$$

$$|B| = (a \cdot d) - (b \cdot c)$$

Substituindo os valores pelos correspondentes para a matriz de covariância do primeiro *cluster* tem-se:

[23] Matriz inversa é definida como a matriz, que ao multiplicar outra, seu resultado é uma matriz identidade (POOLE, MONTEIRO, 2004).
[24] A matriz transposta da matriz de seus cofatores (POOLE, MONTEIRO, 2004).

$$C_1 = \begin{bmatrix} 5 & 2 & 1 \\ 1 & 4 & 2 \\ 2 & 3 & 2 \end{bmatrix}$$

$$adj(C_1) = \begin{bmatrix} \begin{vmatrix} 4 & 2 \\ 3 & 2 \end{vmatrix} & \begin{vmatrix} 1 & 2 \\ 2 & 3 \end{vmatrix} & \begin{vmatrix} 2 & 1 \\ 4 & 2 \end{vmatrix} \\ \begin{vmatrix} 2 & 1 \\ 2 & 2 \end{vmatrix} & \begin{vmatrix} 5 & 1 \\ 2 & 2 \end{vmatrix} & \begin{vmatrix} 1 & 5 \\ 2 & 1 \end{vmatrix} \\ \begin{vmatrix} 1 & 4 \\ 2 & 3 \end{vmatrix} & \begin{vmatrix} 2 & 5 \\ 3 & 2 \end{vmatrix} & \begin{vmatrix} 5 & 2 \\ 1 & 4 \end{vmatrix} \end{bmatrix}$$

$$adj(C_1) = \begin{bmatrix} 2 & -1 & 0 \\ 2 & 8 & -9 \\ -5 & -11 & 18 \end{bmatrix}$$

Substituindo os valores encontrados na equação da matriz inversa (19):

$$C^{-1} = \frac{1}{|C|} adj(C)$$

$$C_1^{-1} = \frac{\begin{bmatrix} 2 & -1 & 0 \\ 2 & 8 & -9 \\ -5 & -11 & 18 \end{bmatrix}}{9}$$

$$C_1^{-1} = \begin{bmatrix} 0{,}22222 & -0{,}11111 & 0 \\ 0{,}22222 & 0{,}88889 & -1 \\ 0{,}55556 & -1{,}22222 & 2 \end{bmatrix}$$

Após calcular o determinante e a matriz inversa pode-se então determinar a matriz de covariância modificada:

$$M_i = |C_i|^{\frac{1}{n}} C_i^{-1}$$

$$M_1 = 9^{\frac{1}{3}} \cdot \begin{bmatrix} 0{,}22222 & -0{,}11111 & 0 \\ 0{,}22222 & 0{,}88889 & -1 \\ 0{,}55556 & -1{,}22222 & 2 \end{bmatrix}$$

Resolvendo obtêm-se a matriz de covariância modificada M_1:

$$M_1 = \begin{bmatrix} 0{,}46223 & -0{,}23112 & 0 \\ 0{,}46223 & 1{,}84894 & -2{,}08007 \\ 1{,}15561 & -2{,}54231 & 4{,}16014 \end{bmatrix}$$

Enfim, o algoritmo pode então calcular as distâncias entre seus elementos e os centros dos *clusters*.

$$d_{11}^2 = (x_1 - c_1)^T M_1 (x_1 - c_1)$$

Sendo que o vetor de dados (Tabela 3) e o centro do primeiro *cluster* (Tabela 7) são:

$$x_1 = [9 \quad 1 \quad 2]$$

$$c_1 = [6 \quad 7 \quad 4]$$

Calculando a distância entre o primeiro elemento e o centro do primeiro *cluster* tem-se:

$$d_{11}^2 = ([9 \quad 1 \quad 2] - [6 \quad 7 \quad 4]) M_1 \left(\begin{bmatrix} 9 \\ 1 \\ 2 \end{bmatrix} - \begin{bmatrix} 6 \\ 7 \\ 4 \end{bmatrix} \right)$$

Primeiramente resolve-se as subtrações entre o primeiro elemento e o centro do primeiro *cluster*, e substitui-se M_1:

$$d_{11}^2 = [3 \quad -6 \quad -2] \cdot \begin{bmatrix} 0{,}46223 & -0{,}23112 & 0 \\ 0{,}46223 & 1{,}84894 & -2{,}08007 \\ 1{,}15561 & -2{,}54231 & 4{,}16014 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ -6 \\ -2 \end{bmatrix}$$

Efetuando a multiplicação entre a matriz de covariância modificada e o último vetor:

$$d_{11}^2 = [3 \quad -6 \quad -2] \cdot \begin{bmatrix} 2{,}77341 \\ -5{,}54681 \\ 10{,}40041 \end{bmatrix}$$

Efetuando esta multiplicação obtêm-se a distância ente o primeiro elemento e o centro do primeiro *cluster*.

$$d_{11}^2 = 20{,}80027$$

Realizando os mesmos cálculos para os outros elementos e *clusters* foram encontradas as seguintes distâncias (Tabela 8).

Tabela 8. Distâncias entre os elementos e os centros dos *clusters*

Distâncias	x_1	x_2	x_3	x_4
c_1	20,80027	27,25326	18,4892	55,06376
c_2	27,25184	26,64832	18,94756	38,65541

Após calcular as distâncias utiliza-se a seguinte propriedade para dividir os elementos em grupos:

$$\mathrm{Z}_i = \left\{ x_j \middle| d_{ij}^2 \leq d_{kj}^2 \ \forall k \neq i \right\}$$

Cada i-ésimo grupo *Z* possuirá os elementos que estejam mais próximos ao centro do i-ésimo *cluster*. Aplicando a propriedade nas distâncias obtidas tem-se:

$$Z_1 = \{x_1, x_3\}$$

$$Z_2 = \{x_2, x_4\}$$

Tendo os grupos definidos pode-se estimar os valores de *T* e *S*. Onde *T* é definido por:

$$T_i = \underset{x_j \in \mathrm{Z}_i}{Med}\left(d_{ij}^2\right)$$

O cálculo da mediana[25] (*Med*) de um determinado conjunto é definido como o elemento central de um conjunto ordenado quando este conjunto tiver uma quantidade ímpar de elementos; quando a quantidade de elementos do conjunto for par, sua mediana é definida como a soma dos elementos centrais dividido por 2 (POOLE, MONTEIRO, 2004). Sendo que a quantidade de elementos do conjunto é um número par, tem-se:

$$T_1 = (20{,}80027 + 18{,}4892)/2$$

$$T_1 = 19{,}64474$$

[25] Medida de tendência central, cujo valor separa metade do grupo como tendo valores inferiores ou iguais a mediana, e outra metade tendo valores maiores que a mediana (POOLE, MONTEIRO, 2004).

Realizando o mesmo procedimento para o segundo conjunto tem-se:

$$T_2 = (26{,}64832 + 38{,}65541)/2$$

$$T_2 = 32{,}65185$$

A estimação de *S* é definida por:

$$S_i = 1.418 \times \underset{x_j \in Z_i}{MAD}\left(d_{ij}^2\right)$$

Onde:

$$MAD = Med(|x - Med(X)|)$$

O desvio mediano absoluto é a mediana dos valores absolutos obtidos pelas subtrações entre os elementos do conjunto e a sua mediana. Sendo que a mediana dos conjuntos foi definido pelas estimações *T*:

$$MAD_1 = \left(|d_{11} - T_1|, |d_{13} - T_1|\right)$$

$$MAD_1 = \left(|20{,}80027 - 19{,}64474|, |18{,}4892 - 19{,}64474|\right)$$

$$MAD_1 = \left(|1{,}15553|, |-1{,}15553|\right)$$

Resolvendo o módulo:

$$MAD_1 = (1{,}15553; 1{,}15553)$$

Calculando o desvio mediano absoluto, sendo este, a mediana dos desvios encontrados:

$$MAD_1 = (1{,}15553 + 1{,}15553)/2$$

$$MAD_1 = 1{,}15553$$

Após obter o valor do desvio mediano absoluto calcula-se a estimação *S*:

$$S_1 = 1{,}418 \times 1{,}15553$$

$$S_1 = 1{,}63853$$

Realizando o mesmo processo com o segundo conjunto obtêm-se:

$$S_2 = 1{,}418 \times MAD_2$$

$$S_2 = 1{,}418 \times 6{,}00353$$

$$S_2 = 8{,}51305$$

O próximo passo é calcular as funções de peso que serão utilizadas no cálculo do centro do *cluster*. Esta função é definida por:

$$\omega_i\left(d_{ij}^2\right) = \begin{cases} 1 - \dfrac{d_{ij}^4}{2T_i^2}, se\ d_{ij}^2 \in [0, T_i] \\ \dfrac{\left[d_{ij}^2 - \left(T_i + \alpha S_i\right)\right]^2}{2\alpha^2 S_i^2}, se\ d_{ij}^2 \in (T_i, T_i + \alpha S_i] \\ 0, se\ d_{ij}^2 > T_i + \alpha S_i \end{cases}$$

Sendo que para o primeiro *cluster,* quando a distância estiver no intervalo entre 0 e 19.64474, a função de peso é calculada utilizando a primeira condição, se estiver entre 19.64474 e 25.37958 utiliza-se a segunda condição e quando seu valor for maior que 25.37958 seu resultado é 0.

A função de peso para o primeiro elemento e o primeiro *cluster* pertence ao segundo intervalo, portanto:

$$\omega_1\left(d_{11}^2\right) = \frac{\left[d_{11}^2 - \left(T_1 + \alpha \cdot S_1\right)\right]^2}{2\alpha^2 {S_1}^2}$$

Substituindo os valores e resolvendo:

$$\omega_1\left(20{,}80027\right) = \frac{\left[20{,}80027 - \left(19{,}64474 + 3{,}5 \cdot 1{,}63853\right)\right]^2}{2 \cdot 3{,}5^2 \cdot 1{,}63853^2}$$

$$\omega_1\left(20{,}80027\right) = \frac{20{,}97022}{65{,}77713}$$

$$\omega_1\left(20{,}80027\right) = 0{,}31881$$

Entre o segundo elemento e o primeiro *cluster* a distância fica no terceiro intervalo, portanto seu valor é 0.

$$27{,}25326 > 25{,}37958$$

$$\omega_1\left(d_{12}^2\right) = 0$$

$$\omega_1\left(27{,}25326\right) = 0$$

No cálculo da distância entre o primeiro *cluster* e o terceiro elemento fica no primeiro intervalo, assim:

$$\omega_1\left(d_{13}^2\right) = 1 - \frac{d_{13}^4}{2T_1^2}$$

Substituindo os valores e calculando:

$$\omega_1\left(18{,}4892\right) = 1 - \frac{18{,}4892^2}{2 \cdot 19{,}64474^2}$$

$$\omega_1\left(18{,}4892\right) = 1 - \frac{341{,}85052}{771{,}83162}$$

$$\omega_1\left(18{,}4892\right) = 1 - 0{,}44291$$

$$\omega_1\left(18{,}4892\right) = 0{,}55709$$

O cálculo da função de peso para a distância entre o quarto elemento e o primeiro *cluster* fica também no terceiro intervalo e assim seu valor é 0:

$$\omega_1\left(55{,}06376\right) = 0$$

Nos cálculos referentes ao segundo *cluster* os intervalos utilizados são*:*

$$[0; 32{,}65185]$$
$$(32{,}65185; 62{,}447525]$$
$$\mathrm{d}_{ij}^2 > 62{,}447525$$

A seguir calculam-se os valores da função de peso referente ao segundo *cluster*, sendo que as distâncias dos três primeiros elementos estão no primeiro intervalo, e portanto resultam em:

$$\omega_2(27{,}25184) = 0{,}65172$$

$$\omega_2(26{,}64832) = 0{,}66696$$

$$\omega_2(18{,}94756) = 0{,}83163$$

A distância entre o segundo *cluster* e o quarto elemento fica no segundo intervalo:

$$\omega_2(38{,}65541) = 0{,}31881$$

A Tabela 9 apresenta os valores resultantes da função de peso, em relação a todas as distâncias:

Tabela 9. Valores resultantes das funções de peso em relação às distâncias

Função de peso	x_1	x_2	x_3	x_4
w_1	0,31881	0	0,55709	0
w_2	0,65172	0,66696	0,83163	0,31881

O próximo passo é calcular a função de perda, sendo antes necessário calcular o valor da constante *K*.

$$K_i = \max_{1 \le j \le C} \left\{ \frac{5T_j + \alpha S_j}{6} \right\} - \frac{5T_i + \alpha S_i}{6}$$

Substituindo os valores:

$$K_1 = \max\left\{ \frac{5T_1 + \alpha S_1}{6}; \frac{5T_2 + \alpha S_2}{6} \right\} - \frac{5T_1 + \alpha S_1}{6}$$

$$K_1 = \max\left\{ \frac{5 \cdot 19{,}64474 + 3{,}5 \cdot 1{,}63853}{6}; \frac{532{,}65185 + 3{,}5 \cdot 8{,}51305}{6} \right\} - \frac{519{,}64474 + 3{,}5 \cdot 1{,}63853}{6}$$

$$K_1 = \max\{17{,}32643; 32{,}17581\} - 17{,}32643$$

$$K_1 = 32{,}17581 - 17{,}32643$$

$$K_1 = 14{,}84938$$

Resolvendo para o segundo *cluster*:

$$K_2 = \max\{17{,}32643; 32{,}17581\} - \frac{5 \cdot 32{,}65185 + 3{,}5 \cdot 8{,}51305}{6}$$

$$K_2 = 32{,}17581 - 32{,}17581$$

$$K_2 = 0$$

Então, é possível calcular a função de perda:

$$\rho_i\left(d_{ij}^2\right) = \begin{cases} d_{ij}^2 - \dfrac{d_{ij}^6}{6T_i^2}, se\ d_{ij}^2 \in [0, T_i] \\ \dfrac{\left[d_{ij}^2 - (T_i + \alpha S_i)\right]^3}{6\alpha^2 S_i^2} + \dfrac{5T_i + \alpha S_i}{6}, se\ d_{ij}^2 \in [T_i, T_i + \alpha S_i] \\ \dfrac{5T_i + \alpha S_i}{6} + K_i\ se\ d_{ij}^2 > T_i + \alpha S_i \end{cases}$$

São utilizados os mesmos intervalos da função de peso, portanto a primeira distância fica no segundo intervalo:

$$\rho_1\left(d_{11}^2\right) = \frac{\left[d_{11}^2 - (T_1 + \alpha \cdot S_1)\right]^3}{6 \cdot \alpha^2 S_1^2} + \frac{5T_1 + \alpha \cdot S_1}{6}$$

Substituindo e resolvendo:

$$\rho_1(20{,}80027) = \frac{\left[20{,}80027 - (19{,}64474 + 3{,}5 \cdot 1{,}63853)\right]^3}{6 \cdot 3{,}5^2 \cdot 1{,}63853^2} + \frac{519{,}64474_1 + 3{,}5 \cdot 1{,}63853}{6}$$

$$\rho_1(20{,}80027) = \frac{-96{,}02943}{65{,}77713} + 17{,}32643$$

$$\rho_1(20{,}80027) = -1{,}45991 + 17{,}32643$$

$$\rho_1(20{,}80027) = 15{,}86652$$

A distância entre o segundo elemento e o primeiro *cluster* fica no terceiro intervalo, sendo:

$$\rho_1\left(d_{12}^2\right) = \frac{5T_1 + \alpha \cdot S_1}{6} + K_1$$

$$\rho_1\left(d_{12}^2\right) = \frac{5 \cdot 19{,}64474 + 3{,}5 \cdot 1{,}63853}{6} + 14{,}84938$$

$$\rho_1\left(d_{12}^2\right) = 17{,}32643 + 14{,}84938$$

$$\rho_1\left(d_{12}^2\right) = 32{,}17581$$

O próximo valor fica no terceiro intervalo:

$$\rho_1\left(d_{13}^2\right) = d_{13}^2 - \frac{d_{13}^6}{6T_1^2}$$

Substituindo os valores e solucionando a equação:

$$\rho_1(18{,}4892) = 18{,}4892 - \frac{18{,}4892^3}{6 \cdot 19{,}64474^2}$$

$$\rho_1(18{,}4892) = 18{,}4892 - \frac{6320{,}64513}{2315{,}49486}$$

$$\rho_1(18{,}4892) = 18{,}4892 - 2{,}72972$$

$$\rho_1(18{,}4892) = 15{,}75948$$

A distância entre o primeiro *cluster* e o quarto elemento fica no terceiro intervalo, sendo assim:

$$\rho_1\left(d_{14}^2\right) = 32{,}17581$$

As distâncias entre o segundo *cluster* e os 3 primeiros elementos pertencem ao primeiro intervalo, resultando em:

$$\rho_2(27{,}25184) = 24{,}08797$$

$$\rho_2(26{,}64832) = 23{,}69001$$

$$\rho_2(18{,}94756) = 17{,}88418$$

A distância entre o quarto elemento e o segundo *cluster* pertence ao segundo intervalo, sendo assim:

$$\rho_2(38{,}65541) = 29{,}64745$$

A Tabela 10 apresenta os valores encontrados pela função de perda:

Tabela 10. Valores encontrados pela função de perda

Função de perda	x_1	x_2	x_3	x_4
ρ_1	15,86652	32,17581	15,75948	32,17581
ρ_2	24,08797	23,69001	17,88418	27,64745

Após calcular os valores das distâncias em relação a função de perda é possível calcular as pertinências entre os elementos e *clusters*, definida pela equação:

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{C} \left[\frac{\rho_i \left(d_{ij}^2 \right)}{\rho_k \left(d_{kj}^2 \right)} \right]^{\frac{1}{(m-1)}}}$$

Substituindo os valores referentes a função de perda do primeiro elemento e o primeiro *cluster*, obtem-se o seu grau de pertinência:

$$\mu_{11} = \frac{1}{\left[\frac{\rho_i \left(d_{11}^2 \right)}{\rho_i \left(d_{11}^2 \right)} \right]^{\frac{1}{(m-1)}} + \left[\frac{\rho_i \left(d_{11}^2 \right)}{\rho_i \left(d_{12}^2 \right)} \right]^{\frac{1}{(m-1)}}}$$

$$\mu_{11} = \frac{1}{\left[\frac{15{,}86652}{15{,}86652} \right]^{\frac{1}{(2-1)}} + \left[\frac{15{,}86652}{24{,}08797} \right]^{\frac{1}{(2-1)}}}$$

$$\mu_{11} = \frac{1}{1 + 0{,}65868}$$

$$\mu_{11} = 0{,}60289$$

Realizando o mesmo processo aos outros elementos e *clusters* obtêm-se as pertinências apresentadas na Tabela 11.

Tabela 11. Graus de pertinência atualizados

Clusters	x_1	x_2	x_3	x_4
1	0,60289	0,42404	0,53157	0,47954
2	0,39712	0,57595	0,46843	0,52044

A próxima etapa do algoritmo RCP é calcular os centros dos *clusters*. Este é definido pela seguinte equação:

$$c_i = \frac{\sum_{j=1}^{N} \left(\mu_{ij} \right)^m w_{ij} x_j}{\sum_{j=1}^{N} \left(u_{ij} \right)^m w_{ij}}$$

Assim, faz-se a substituição dos valores:

$$c_1 = \frac{(\mu_{11})^m w_{11} x_1 + (\mu_{12})^m w_{12} x_2 + (\mu_{13})^m w_{13} x_3 + (\mu_{14})^m w_{14} x_4}{(\mu_{11})^m w_{11} + (\mu_{12})^m w_{12} + (\mu_{13})^m w_{13} + (\mu_{14})^m w_{14}}$$

$$c_1 = \frac{\begin{array}{c}(0{,}60289)^2 \cdot 0{,}31881 \cdot \begin{bmatrix} 9 \\ 1 \\ 2 \end{bmatrix} + (0{,}42404)^2 \cdot 0 \cdot \begin{bmatrix} 8 \\ 1{,}8 \\ 3 \end{bmatrix} + \\ (0{,}53157)^2 \cdot 0{,}55709 \cdot \begin{bmatrix} 7 \\ 2 \\ 2{,}5 \end{bmatrix} + (0{,}47954)^2 \cdot 0 \cdot \begin{bmatrix} 7{,}5 \\ 1{,}5 \\ 4 \end{bmatrix}\end{array}}{(0{,}60289)^2 \cdot 0{,}31881 + (0{,}42404)^2 \cdot 0 + (0{,}53157)^2 \cdot 0{,}55709 + (0{,}47954)^2 \cdot 0}$$

Primeiramente, resolve-se o numerador elevando o grau de pertinência ao elemento *fuzzificador*, o valor obtido é multiplicado pela função de peso, e finalmente multiplica-se o valor obtido pelo vetor de dados. No denominador é multiplicado o valor da pertinência elevado ao elemento *fuzzyficador*, multiplicado pelo valor da função de peso.

$$c_1 = \frac{\begin{bmatrix} 1{,}04292 \\ 0{,}11588 \\ 0{,}23176 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1{,}10191 \\ 0{,}31482 \\ 0{,}39354 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}}{0{,}11588 + 0 + 0{,}15742 + 0}$$

Realize-se o somatório dos valores obtidos, tendo-se.

$$c_1 = \frac{\begin{bmatrix} 2{,}14483 \\ 0{,}4307 \\ 0{,}6253 \end{bmatrix}}{0{,}2733}$$

Enfim, é realizada a divisão entre cada linha da matriz e o somatório encontrado.

$$c_1 = \begin{bmatrix} 7{,}84788 \\ 1{,}57593 \\ 2{,}28795 \end{bmatrix}$$

O mesmo processo é executado para encontrar o centro do segundo *cluster*.

$$c_2 = \begin{bmatrix} 7{,}79316 \\ 1{,}67925 \\ 2{,}81858 \end{bmatrix}$$

O último passo do algoritmo é calcular as matrizes de covariância *fuzzy* definidas pela equação:

$$C_i = \frac{\sum_{j=1}^{N} (\mu_{ij})^m \omega_{ij} (x_j - c_i)(x_j - c_i)^T}{\sum_{j=1}^{N} (u_{ij})^m \omega_{ij}}$$

Substituindo-se:

$$C_1 = \frac{\begin{array}{c}(\mu_{11})^m \omega_{11}(x_1 - c_1)(x_1 - c_1) + \\ (\mu_{12})^m \omega_{12}(x_2 - c_1)(x_2 - c_1) + \\ (\mu_{13})^m \omega_{13}(x_3 - c_1)(x_3 - c_1) + \\ (\mu_{14})^m \omega_{14}(x_4 - c_1)(x_4 - c_1)\end{array}}{(\mu_{11})^m \omega_{11} + (\mu_{12})^m \omega_{12} + (\mu_{13})^m \omega_{13} + (\mu_{14})^m \omega_{14}}$$

$$C_1 = \frac{\begin{array}{l}(0{,}60289)^2 \cdot 0{,}31881 \cdot \left(\begin{bmatrix}9\\1\\2\end{bmatrix} - \begin{bmatrix}7{,}84788\\1{,}57593\\2{,}28795\end{bmatrix}\right)\left([9 \quad 1 \quad 2] - [7{,}84788 \quad 1{,}57593 \quad 2{,}28795]\right) + \\ (0{,}42404)^2 \cdot 0 \cdot \left(\begin{bmatrix}8\\1{,}8\\3\end{bmatrix} - \begin{bmatrix}7{,}84788\\1{,}57593\\2{,}28795\end{bmatrix}\right)\left([8 \quad 1{,}8 \quad 3] - [7{,}84788 \quad 1{,}57593 \quad 2{,}28795]\right) + \\ (0{,}53157)^2 \cdot 0{,}55709 \cdot \left(\begin{bmatrix}7\\2\\2{,}5\end{bmatrix} - \begin{bmatrix}7{,}84788\\1{,}57593\\2{,}28795\end{bmatrix}\right)\left([7 \quad 2 \quad 2{,}5] - [7{,}84788 \quad 1{,}57593 \quad 2{,}28795]\right) + \\ (0{,}47954)^2 \cdot 0 \cdot \left(\begin{bmatrix}7{,}5\\1{,}5\\4\end{bmatrix} - \begin{bmatrix}7{,}84788\\1{,}57593\\2{,}28795\end{bmatrix}\right)\left([7{,}5 \quad 15 \quad 4] - [7{,}84788 \quad 1{,}57593 \quad 2{,}28795]\right)\end{array}}{(0{,}60289)^2 \cdot 0{,}31881 + (0{,}42404)^2 \cdot 0 + (0{,}53157)^2 \cdot 0{,}55709 + (0{,}47954)^2 \cdot 0}$$

Primeiramente resolvem-se as multiplicações e as subtrações entre os vetores individualmente de acordo com suas posições.

$$C_1 = \frac{\begin{array}{l} 0{,}11588 \cdot \begin{bmatrix} 1{,}15212 \\ -0{,}57593 \\ -0{,}28795 \end{bmatrix} \begin{bmatrix} 1{,}15212 & -0{,}57593 & -0{,}28795 \end{bmatrix} + \\ 0 \cdot \begin{bmatrix} 0{,}15212 \\ 0{,}22407 \\ 0{,}71205 \end{bmatrix} \begin{bmatrix} 0{,}15212 & 0{,}22407 & 0{,}71205 \end{bmatrix} + \\ 0{,}15742 \cdot \begin{bmatrix} -0{,}84788 \\ 0{,}42407 \\ 0{,}21205 \end{bmatrix} \begin{bmatrix} -0{,}84788 & 0{,}42407 & 0{,}21205 \end{bmatrix} + \\ 0 \cdot \begin{bmatrix} -0{,}34788 \\ -0{,}07593 \\ 1{,}71205 \end{bmatrix} \begin{bmatrix} -0{,}34788 & -0{,}07593 & 1{,}71205 \end{bmatrix} \end{array}}{0{,}2733}$$

Resolvendo a multiplicação com o primeiro vetor de dados:

$$C_1 = \frac{\begin{array}{l} \begin{bmatrix} 0{,}13351 \\ -0{,}06674 \\ -0{,}03337 \end{bmatrix} \begin{bmatrix} 1{,}15212 & -0{,}57593 & -0{,}28795 \end{bmatrix} + \\ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0{,}15212 & 0{,}22407 & 0{,}71205 \end{bmatrix} + \\ \begin{bmatrix} -0{,}13346 \\ 0{,}06676 \\ 0{,}03337 \end{bmatrix} \begin{bmatrix} -0{,}84788 & 0{,}42407 & 0{,}21205 \end{bmatrix} + \\ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} -0{,}34788 & -0{,}07593 & 1{,}71205 \end{bmatrix} \end{array}}{0{,}2733}$$

Multiplicando as matrizes encontra-se:

$$C_1 = \frac{\begin{array}{l} \begin{bmatrix} 0{,}15382 & -0{,}07689 & -0{,}03844 \\ -0{,}07689 & 0{,}03844 & 0{,}01922 \\ -0{,}03845 & 0{,}01922 & 0{,}00961 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \\ \begin{bmatrix} 0{,}11316 & -0{,}0566 & -0{,}0283 \\ -0{,}0566 & 0{,}02831 & 0{,}01416 \\ -0{,}02829 & 0{,}01415 & 0{,}00708 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{array}}{0{,}2733}$$

Realizando a soma entre as matrizes tem-se:

$$C_1 = \frac{\begin{bmatrix} 0{,}26698 & -0{,}13349 & -0{,}06674 \\ -0{,}13349 & 0{,}06675 & 0{,}03338 \\ -0{,}06674 & 0{,}03337 & 0{,}01669 \end{bmatrix}}{0{,}2733}$$

Dividindo cada elemento da matriz pelo somatório obtido tem-se a matriz de covariância do primeiro *cluster*:

$$C_1 = \begin{bmatrix} 0{,}97686 & -0{,}48844 & -0{,}24421 \\ -0{,}48844 & 0{,}24424 & 0{,}12214 \\ -0{,}24421 & 0{,}12211 & 0{,}06107 \end{bmatrix}$$

Efetuando o mesmo processo para o segundo *cluster* obtem-se:

$$C_1 = \begin{bmatrix} 0{,}47463 & -0{,}20344 & -0{,}12994 \\ -0{,}20347 & 0{,}12178 & 0{,}04233 \\ -0{,}12995 & 0{,}04229 & 0{,}36298 \end{bmatrix}$$

Considerando que os valores dos centros dos *clusters* e das matrizes de covariância não estabilizaram, já que seus valores variaram muito de uma iteração a outra, o algoritmo RCP realizaria a próxima iteração.

6.2.2.3 Modelagem Matemática do Algoritmo URCP

O algoritmo URCP inicia aplicando a clusterização na base de dados com o algoritmo RCP, descrito anteriormente, sendo que após o RCP terminar sua execução com os centros e matrizes de covariância estabilizados é aplicado o Unconstrained RCP.

O Unconstrained RCP utiliza o mesmo cálculo que o RCP, diferenciando-se apenas por não calcular os graus de pertinência, e portanto não utiliza os valores das pertinências nas equações (12) e (13) referente aos centros dos *clusters* e matrizes de

covariância. Após o Unconstrained RCP terminar sua execução os graus de pertinência são calculados novamente para serem utilizados na função de similaridade do algoritmo URCP.

Nos parâmetros deve-se informar o número máximo de *clusters* que podem existir na base de dados, sendo que além dos parâmetros utilizados pelo RCP, informa-se:

a) **taxa de erro(ε):** erro utilizado na fusão dos *clusters*. Foi definido o valor 0,3 pois define que 70% dos elementos entre os conjuntos são iguais. Sendo que deve-se definir, dependendo da base de dados, quando os grupos são bem separados e compactos este valor pode ser baixo, quando os grupos não estão bem definidos, este valor deve ser aumentado.

Na demonstração do algoritmo URCP foram utilizados as pertinências e pesos obtidos na modelagem do algoritmo RCP. Sendo que somente os elementos que possuem pesos maiores que 0 são utilizados:

$$G_i = \{x_k \in X \mid \omega_{ik} > 0\}$$

Assim, os grupos de elementos formados são:

$$G_1 = \{x_1, x_3\}$$

$$G_2 = \{x_1, x_2, x_3, x_4\}$$

A união dos *clusters* é definida como:

$$G_{ij} = G_i \cup G_j = \{x_k \in X \mid \omega_{ik} > 0 \, OR \, \omega_{jk} > 0\}$$

A união encontrada foi:

$$G_{12} = \{x_1, x_2, x_3, x_4\}$$

Após definir os conjuntos é possível calcular a função de similaridade:

$$S_{ij} = 1 - \frac{\sum_{x_k \in G_{ij}} \left| \mu_{ik} - \mu_{jk} \right|}{\left| G_i \right| + \left| G_j \right|}$$

Onde:

$$|G_i| = \sum_{x_k \in G_i} \mu_{ik} \qquad |G_j| = \sum_{x_k \in G_j} \mu_{jk}$$

Resolvendo primeiramente a cardinalidade de cada *cluster*:

$$|G_1| = 0{,}60289 + 0{,}53157 = 1{,}13446$$

$$|G_2| = 0{,}39712 + 0{,}57595 + 0{,}46843 + 0{,}52044 = 1{,}96194$$

Substituindo os valores na função de similaridade:

$$S_{12} = 1 - \frac{\left(|0{,}60289 - 0{,}39712| + |0{,}42404 - 0{,}57595| + |0{,}53157 - 0{,}46843| + |0{,}47954 - 0{,}52044|\right)}{1{,}13446 + 1{,}96194}$$

$$S_{12} = 1 - \frac{\left(|0{,}20577| + |-0{,}15191| + |0{,}06314| + |-0{,}0409|\right)}{1{,}13446 + 1{,}96194}$$

$$S_{12} = 1 - \frac{(0{,}20577 + 0{,}15191 + 0{,}06314 + 0{,}0409)}{3{,}0964}$$

$$S_{12} = 1 - \frac{0{,}46172}{3{,}0964}$$

$$S_{12} = 1 - 0{,}14912$$

$$S_{12} = 0{,}85088$$

A seguir o algoritmo realiza o seguinte teste:

$$S_{ij} \geq (1 - \varepsilon)$$

Substituindo os valores:

$$S_{12} \geq (1 - \varepsilon)$$

$$0{,}85088 \geq (1 - 0{,}3)$$

$$0{,}85088 \geq 0{,}7$$

Considerando que o teste resultou verdadeiro, pois $S_{12} \geq 0{,}7$, estes dois *clusters* se tornariam um. O algoritmo executa este processo de similaridade entre cada um dos *cluster* e todos os outros, para então atualizar o número de *clusters* e realizar o processo clusterização

iniciado pelo RCP. Isto se repete até que o número de *clusters*, de uma iteração a outra, não mude.

6.2.3 Validação

A clusterização, na maioria das vezes, é efetuada sem conhecimento prévio da base de dados, sendo a quantidade de grupos, necessária para efetuar o processo, um parâmetro difícil de definir. Devido a isto, foram desenvolvidos índices que ajudam o usuário a definir o número ótimo de *clusters* existentes na base (KIM et al, 2004, tradução nossa).

Índices de validação são utilizados para encontrar o número ótimo de *clusters* em uma determinada base de dados. Eles ajudam a definir a quantidade de *clusters* que encontra partições estáveis, que melhor definem e explicam a estrutura da base de dados em questão (KIM et al, 2004, tradução nossa).

Bezdek et al (2005, tradução nossa) propôs os seguintes índices para validação *fuzzy* de *clusters*:

$$V_{PC} = \frac{\sum_{j=1}^{N}\sum_{i=1}^{C}\mu_{ij}^{2}}{N} \tag{22}$$

Onde:

a) $\boldsymbol{V_{PC}}$: índice do coeficiente de partição;

b) $\boldsymbol{C}$: quantidade de *clusters*;

c) $\boldsymbol{N}$: quantidade de elementos;

d) $\boldsymbol{\mu_{ij}}$: grau de pertinência do j-ésimo elemento no i-ésimo *cluster*.

O índice chamado coeficiente de partição (V_{PC}) é definido como o somatório de todos os graus de pertinência, de todos os elementos em relação à todos os *clusters*, ao quadrado, dividido pelo número de elementos da base de dados.

O valor de V_{PC} deve ser maximizado, ou seja, o maior possível, ao definir diferentes valores para o parâmetro referente a quantidade de grupos a serem encontrados na base de dados. Encontrando-se no intervalo de [1/*C*, 1], indicando inexistência de grupos bem definidos quando este valor estiver próximo de 1/*C* (BEZDEK et al, 2005, tradução nossa).

Utilizando os resultados encontrados na modelagem matemática do algoritmo FCM para demonstrar o cálculo efetuado por este índice tem-se:

$$V_{PC} = \frac{0{,}43037^2 + 0{,}39834^2 + 0{,}63189^2 + 0{,}52365^2 + 0{,}56962^2 + 0{,}60166^2 + 0{,}36809^2 + 0{,}47634^2}{4}$$

Resolvendo, tem-se o coeficiente de partição:

$$V_{PC} = \frac{2{,}06624}{4}$$

$$V_{PC} = 0{,}51656$$

O índice de partição entrópica (V_{PE}), também desenvolvido por Bezdek é definido por (BEZDEK et al, 2005, tradução nossa):

$$V_{PE} = -\frac{1}{N}\sum_{j=1}^{N}\sum_{i=1}^{C}\left[\mu_{ij}\log\left(\mu_{ij}\right)\right] \tag{23}$$

Onde:

a) $\boldsymbol{V_{PE}}$: índice de partição entrópica;

b) $\boldsymbol{C}$: quantidade de *clusters*;

c) $\boldsymbol{N}$: quantidade de elementos;

d) $\boldsymbol{\mu_{ij}}$: grau de pertinência do j-ésimo elemento no i-ésimo *cluster*.

Este índice trabalha de forma contrária ao coeficiente de partição, ou seja, seu valor deve ser minimizado. O índice tende para zero quando encontra grupos bem definidos, sendo que um valor próximo do limite superior do intervalo indica a ausência de grupos definidos no conjunto de dados ou a incapacidade do algoritmo de obtê-las. Seu valor fica no intervalo [0, log(*C*)].

Utilizando os mesmos valores que foram empregados para demonstrar o coeficiente de partição tem-se:

$$V_{PE} = -\frac{1}{4}[0{,}43037 \cdot \log(0{,}43037)] + [0{,}39834 \cdot \log(0{,}39834)] + [0{,}63189 \cdot \log(0{,}63189)] + [0{,}52365 \cdot \log(0{,}52365)] + [0{,}56962 \cdot \log(0{,}56962)] + [0{,}60166 \cdot \log(0{,}60166)] + [0{,}36809 \cdot \log(0{,}36809)] + [0{,}47634 \cdot \log(0{,}47634)]$$

Resolvendo encontra-se o valor do índice:

$$V_{PE} = -\frac{1}{4} \cdot \text{-}1{,}175083165612741182526323384801 7$$

$$V_{PE} = 0{,}29376$$

O índice definido a seguir foi desenvolvido por Xie e Beni, sendo chamado de Xie-Beni, devido a seus criadores. Este índice procura definir o número ótimo de *clusters* considerando a separação e compactação dos *clusters*. Quando o índice encontra um valor baixo significa que os grupos são bem separados e compactos (KIM et al, 2004, tradução nossa) :

$$V_{XB} = \frac{\sum_{j=1}^{N}\sum_{i=1}^{C} \mu_{ij}^{m} \left\| x_j - c_i \right\|}{N \mid \left(\min_{i,k} \left\| c_i - c_k \right\| \right)} \tag{24}$$

Onde:

a) $\boldsymbol{V_{XB}}$: índice Xie-Beni;

b) $\boldsymbol{C}$: quantidade de *clusters*;

c) $\boldsymbol{N}$: quantidade de elementos;

d) $\boldsymbol{\mu_{ij}}$: grau de pertinência do j-ésimo elemento no i-ésimo *cluster*;

e) x_j: j-ésimo elemento;

f) c_i: centro do i-ésimo *cluster*.

O índice Xie-Beni é definido como o somatório dos graus de pertinência elevados ao elemento *fuzzyficador* multiplicados pela distância entre elementos e centros dos *clusters* dividido pela quantidade de elementos, sendo que este valor define a compactação dos grupos.

O resultado é dividido pela menor distância entre os centros dos *clusters*, este valor define a separação dos grupos.

Na demonstração dos cálculos efetuados por este índice são utilizados também os resultados encontrados pelo FCM. Primeiramente calcula-se a compactação dos grupos:

$$V_C = \frac{\sum_{j=1}^{N}\sum_{i=1}^{C}\mu_{ij}^{m}\left\|x_j - c_i\right\|}{N}$$

$$V_C = \frac{\begin{array}{c}0{,}43037^2 \cdot 1{,}78493 + 0{,}39834^2 \cdot 0{,}47279 + 0{,}63189^2 \cdot 0{,}74818 + 0{,}52365^2 \cdot 1{,}17092 + \\ 0{,}56962^2 \cdot 1{,}34864 + 0{,}60166^2 \cdot 0{,}31302 + 0{,}36809^2 \cdot 1{,}28437 + 0{,}47634^2 \cdot 1{,}28727\end{array}}{4}$$

Calculando, obtem-se a compactação dos *clusters*:

$$V_C = \frac{2{,}04244}{4}$$

$$V_C = 0{,}51061$$

Em seguida, calcula-se a separação dos grupos. Assim, deve-se obter as distância entre os centros dos *clusters*, estas são encontradas por meio da distância Euclidiana:

$$d_{ij}^2 = \sqrt{\sum_{j=1}^{p}\left(x_j - c_i\right)^2}$$

$$d_{12}^2 = \sqrt{(7{,}57436 - 8{,}13827)^2 + (1{,}6657 - 1{,}54467)^2 + (2{,}84327 - 2{,}88298)^2}$$

$$d_{12}^2 = \sqrt{0{,}31798 + 0{,}01465 + 0{,}00156}$$

$$d_{12}^2 = 0{,}57808$$

Encontra-se o mesmo valor calculando a distância entre o segundo e primeiro *cluster*:

$$d_{21}^2 = 0{,}57808$$

O valor da separação é definido pela menor distância encontrada:

$$V_S = \min(0{,}57808; 0{,}57808)$$

$$V_S = 0{,}57808$$

Após obter os valores referentes a compactação e separação dos *clusters* é possível obter o valor do índice Xie-Beni, dividindo os valores encontrados:

$$V_{XB} = \frac{0{,}51061}{0{,}57808}$$

$$V_{XB} = 0{,}88327$$

6.2.4 Implementação e Testes

Os algoritmos RCP, URCP e FCM foram implementados no módulo de clusterização da *Shell* Orion *Data Mining Engine* por meio da linguagem de programação Java, utilizando o ambiente de programação Netbeans 6.8.

Na realização de testes, optou-se por adicionar a conexão com um novo banco de dados, aumentando a versatilidade da ferramenta, considerando que esta já possui conexões para os bancos HSQLDB, *Firebird*, *PostGreSQL*, MySQL e *Sybase*. O banco adicionado foi a versão gratuita do *Oracle*, chamada *Oracle Express Edition*[26].

Após definir o banco de dados a ser utilizado, deve-se efetuar testes com uma base específica, verificando os resultados apresentados pelos algoritmos implementados. Nesta pesquisa utilizou-se os registros de pacientes com sepse da UTI do Hospital de Clínicas de Porto Alegre, Rio Grande do Sul, a fim de se realizar testes dos três algoritmos implementados, estes dados foram inseridos no banco *Oracle* para possibilitar o acesso a estes dados pela *Shell* Orion.

Realizada a inserção dos dados, deve-se conectar ao banco de dados utilizando o menu *Arquivo*, submenu *Conectar*, da *Shell* Orion. Após isto, pode-se acessar o menu *Data Mining*, submenu *Clusterização*, método de lógica *fuzzy* e então selecionar um dos algoritmos

[26] Disponível em (http://www.oracle.com/technology/software/products/database/xe/index.html).

disponíveis, sendo que os desenvolvidos nesta pesquisa foram o RCP, URCP e FCM (Figura 23).

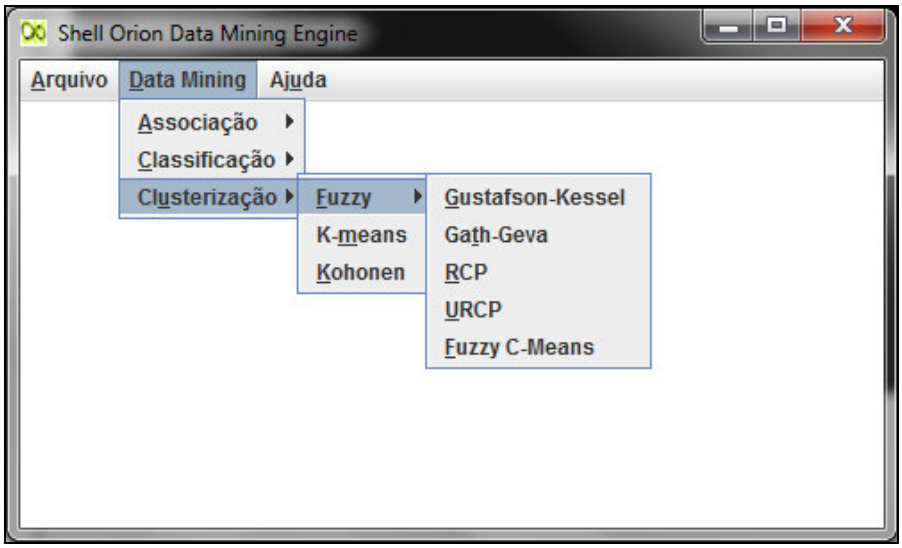

Figura 23. Acesso aos algoritmos na *Shell* Orion

Ao selecionar um dos algoritmos implementados nesta pesquisa, os seguintes parâmetros devem ser informados:

a) **quantidade de *clusters***: quantidade de grupos que o algoritmo deverá encontrar;

b) **parâmetro de *fuzzificação***: define o grau de *fuzzyficação* entre elementos e *clusters*;

c) **quantidade de iterações**: a quantidade máxima de ciclos que o algoritmo irá executar. Quando o valor escolhido for 0, o algoritmo irá executar até que condição de parada seja alcançada;

d) **atributos de entrada**: atributos da base de dados que devem ser selecionados para então serem utilizados no processo de *clusterização*.

Os algoritmos RCP e URCP, além dos já mencionados, necessitam dos seguintes parâmetros:

a) **abrangência da função de peso**: define a abrangência da função de peso, se seu valor for alto indicará que muitos dados terão peso não-zero, ou seja, será maior a quantidade de elementos sendo utilizados em determinadas partes do processo de clusterização. Caso contrário, quando este valor for baixo, aumentará a quantidade de elementos com peso zero, afetando o processo de maneira inversa;

b) **taxa de erro**: este parâmetro é informado apenas para a execução do algoritmo URCP e define quanto dois grupos devem ser similares para que ocorra a fusão entre eles.

Na execução do algoritmo FCM, além dos já mencionados como sendo necessário em todos os algoritmos desenvolvidos nesta pesquisa, é necessário informar o seguinte parâmetro:

a) **taxa de erro**: erro aceitável na execução do algoritmo, indicando a parada do algoritmo.

Todos estes parâmetros implementados são definidos pelo usuário. A Figura 24 apresenta a interface dos algoritmos, para que estes parâmetros sejam informados.

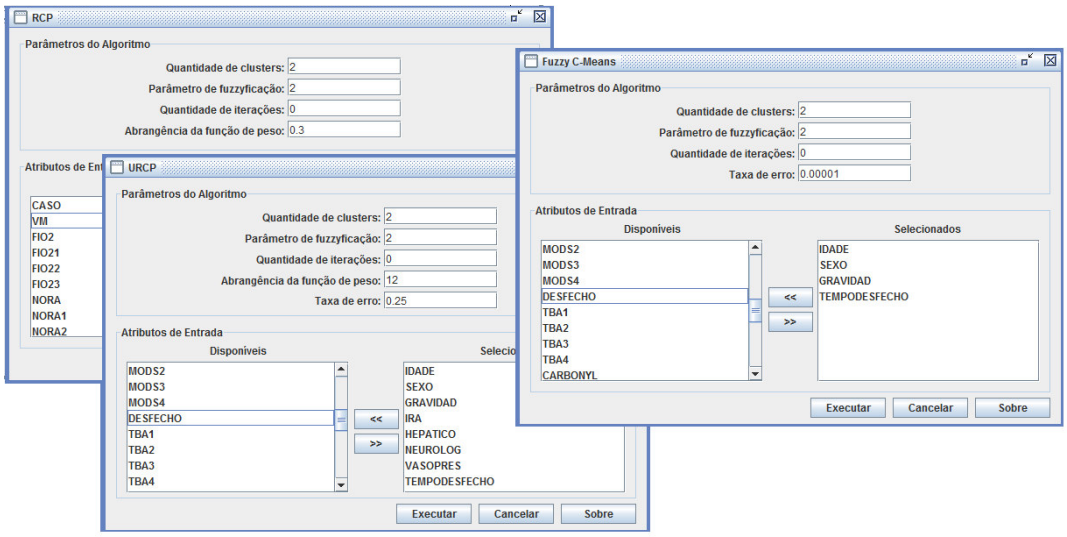

Figura 24. Interface para execução dos algoritmos

Ao final da execução de qualquer algoritmo selecionado, seus resultados podem ser visualizados de diferentes maneiras. A Figura 25 apresenta o resumo da clusterização, onde são apresentados os parâmetros e atributos de entrada, informações sobre os grupos encontrados, os centros de cada *cluster*, o atributo de saída, sendo que este pode ser alterado por meio do campo *atributo de saída*, e os índices de validação.

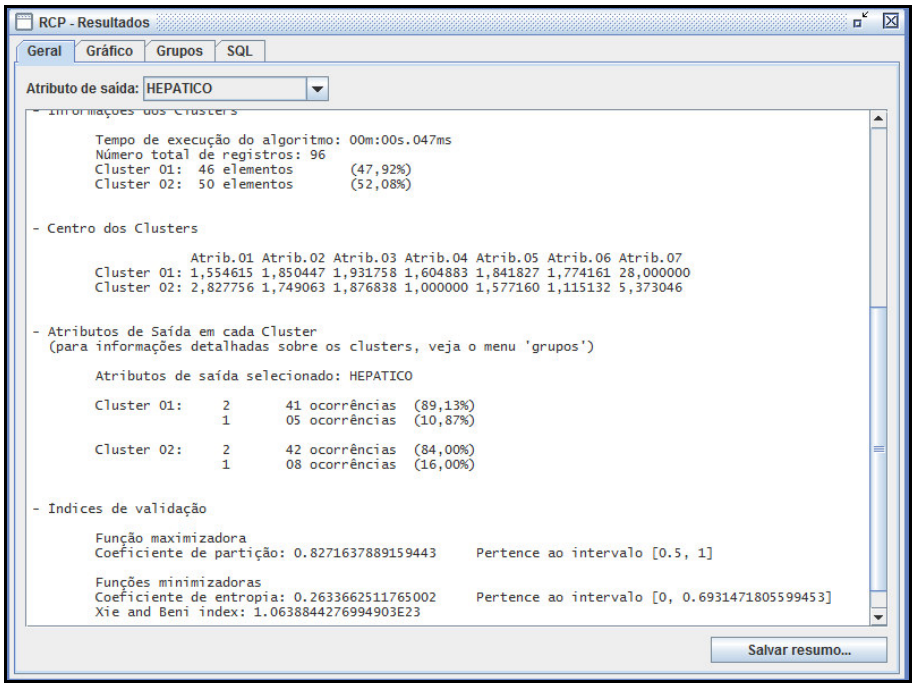

Figura 25. Resumo da clusterização realizada pelo algoritmo RCP

O resumo apresenta dois grupos encontrados pelo algoritmo. O atributo de saída *(hepatico)* indica que o primeiro *cluster* possui, predominantemente, pacientes não diagnosticados com a doença hepatite, enquanto no segundo *cluster* predominam pacientes diagnosticados com a doença. No final do resumo são apresentados os índices de validação, a função maximizadora indica que quanto maior seu valor, melhor o resultado encontrado no processo, inversamente, as funções minimizadoras devem possuir valores pequenos para indicar uma boa partição dos dados, sendo que as comparações entre os valores encontrados pelos índices são feitas modificando os parâmetros de entrada (quantidade de *clusters*, parâmetro de *fuzzyficação*, entre outros). Os dados obtidos neste resumo podem ser exportados em arquivo de texto, por meio do botão *Salvar resumo*.

Os resultados podem ser visualizados em forma gráfica, como é mostrado na Figura 26. O gráfico é gerado utilizando a técnica *Principal Component Analysis* (PCA), que realiza, por meio de sucessivas decomposições, a transformação de uma base de dados com *n* dimensões em uma matriz de duas dimensões, o que possibilita a projeção dos elementos no gráfico.

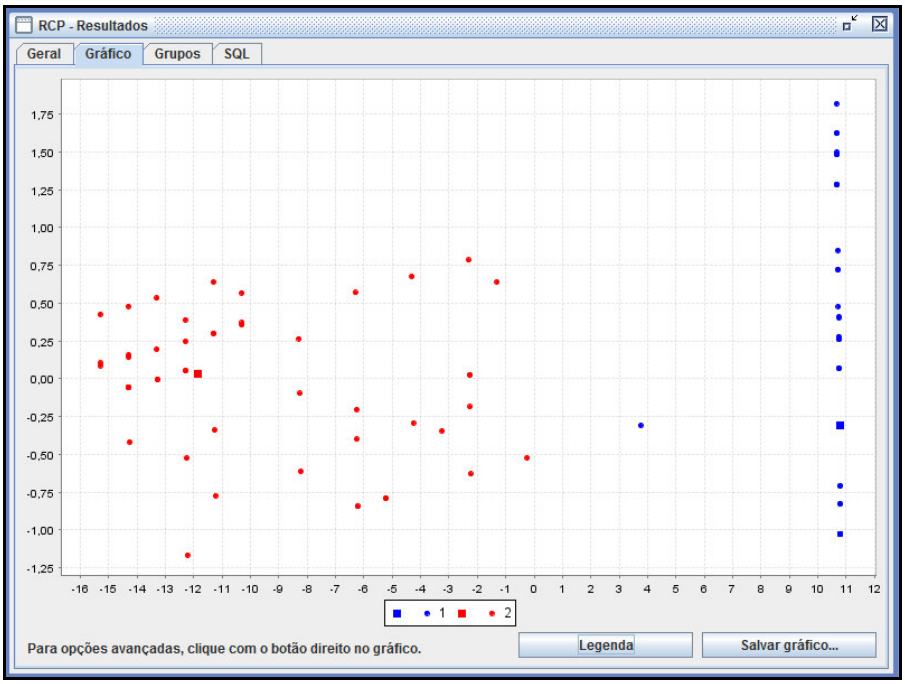

Figura 26. Representação gráfica dos resultados

A análise dos resultados encontrados por qualquer um dos algoritmos implementados também pode ser feita por meio de uma estrutura de árvore (Figura 27). Esta estrutura apresenta cada grupo como sendo um nó, e ao expandi-lo são apresentados seus elementos, que por sua vez apresentam os atributos de entrada e as pertinências encontradas pelo método.

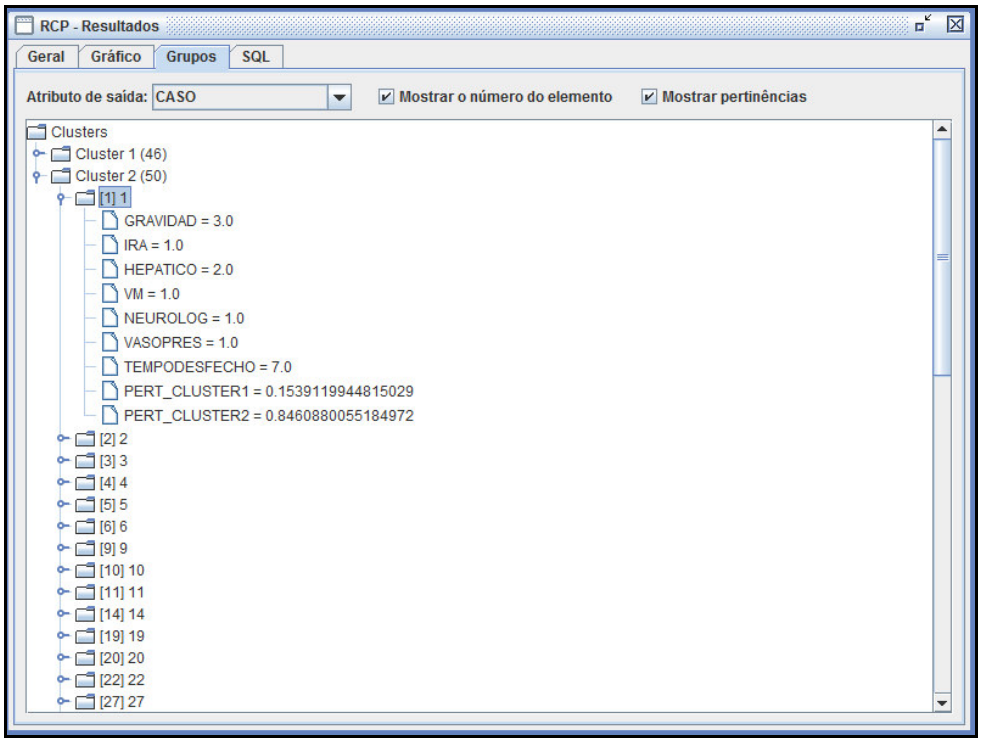

Figura 27. Resultados encontrados em forma de árvore

Os resultados ainda podem ser exportados no formato *Strutured Query Language* (SQL), permitindo que os resultados encontrados no processo de clusterização possam ser utilizados como dados de entrada para outra tarefa de *data mining*. Desta forma, pode-se melhorar o conhecimento extraído. A Figura 28 apresenta a tela para exportação dos resultados em formato SQL, que é inicializada pelo botão *Salvar SQL*.

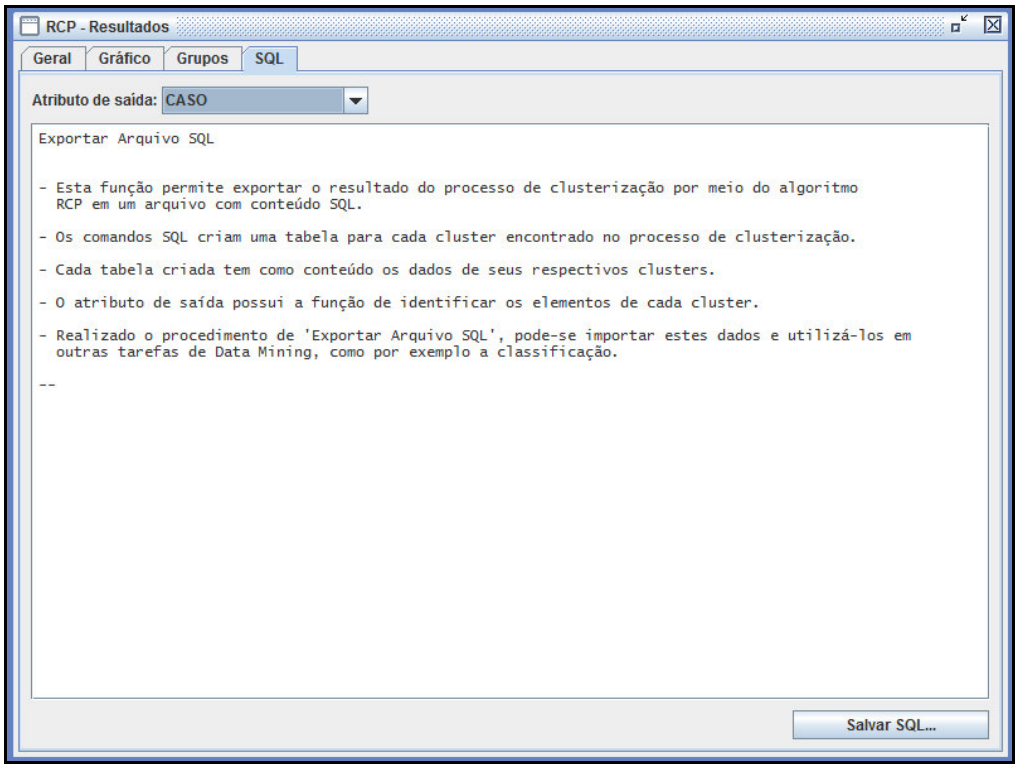

Figura 28. Tela para exportação dos resultados em formato SQL

Finalizado o processo de implementação dos algoritmos FCM, RCP e URCP na *Shell* Orion, foram realizados testes para verificar os resultados obtidos e os tempos de processamento de cada algoritmo.

6.3 RESULTADOS OBTIDOS

Os resultados obtidos pela ferramenta foram analisados considerando, principalmente, o funcionamento do módulo, análise dos *clusters* gerados e tempo de processamento.

Na realização dos testes com os algoritmos RCP, URCP e FCM para a clusterização de dados, utilizou-se um microcomputador com sistema operacional Windows 7, processador Intel Core i5 2.27 GHz e 4GB de memória RAM.

Inicialmente, verificou-se a capacidade dos algoritmos em descobrir conhecimento na base de dados de pacientes com sepse, a fim de analisar os resultados encontrados por cada algoritmo.

6.3.1 Resultados Obtidos pelo Algoritmo *Fuzzy* C-Means

Na avaliação dos resultados gerados pelo algoritmo, os parâmetros utilizados foram os seguintes, sendo que na execução dos três algoritmos os atributos de entrada foram os mesmos, para ser possível comparar os resultados:

a) **quantidade de *clusters***: 2;

b) **parâmetro de *fuzzyficação***: 2;

c) **quantidade de iterações**: 0 (ilimitado);

d) **taxa de erro**: 0.00001;

e) **atributos de entrada**: *idade, sexo, gravidad, ira, hepatico, neurolog, vasopres, tempodesfecho*.

O objetivo foi encontrar o *desfecho* dos pacientes, ou seja, por meio das semelhanças entre os atributos selecionados definir os pacientes curados e os que faleceram. Assim, deve-se selecionar o atributo de saída *desfecho*. Após a realização da clusterização, os *clusters* foram identificados (Tabela 12).

Tabela 12. *Clusters* encontrados pelo FCM na base de pacientes com sepse

Cluster	**Quantidade de elementos**	**Porcentagem dos elementos**	***desfecho***
1	10 37	48,96%	1 2
2	41 08	51,04%	1 2

A Figura 29 apresenta o gráfico dos *clusters* gerados, neste pode-se perceber grupos indefinidos, devido a dificuldade do algoritmo em identificar *clusters* de diferentes formas.

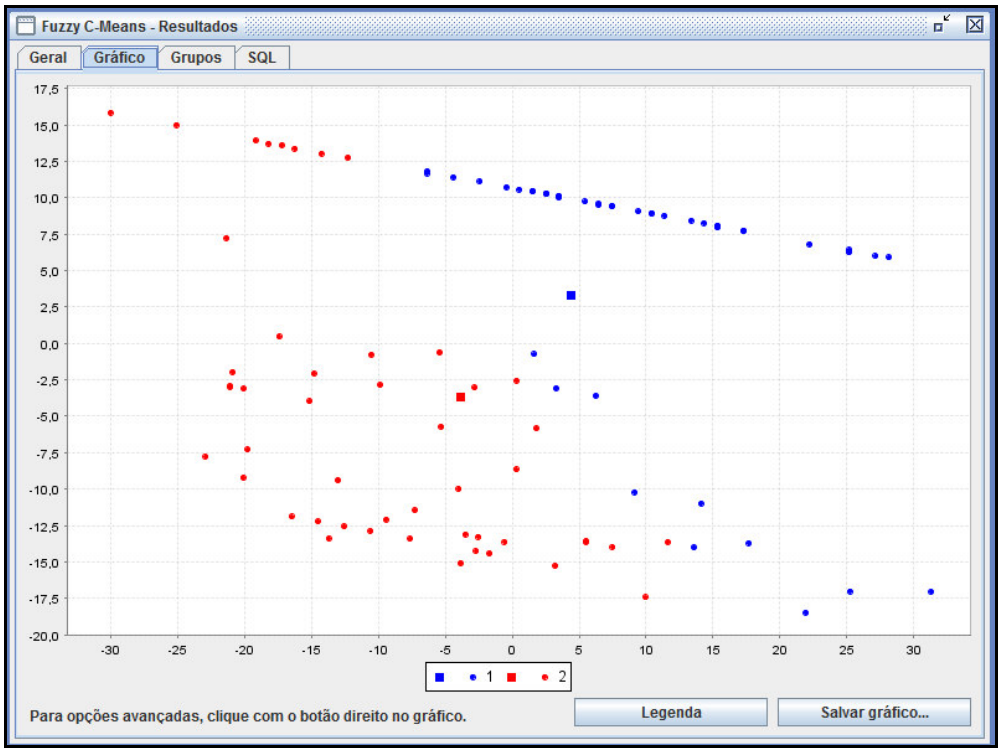

Figura 29. Gráfico gerado pelo algoritmo FCM

Os índices de validação indicaram uma boa partição dos dados, sendo estes, apresentados na Tabela 13:

Tabela 13. Índices de validação encontrados para o algoritmo FCM

Índice	Valor encontrado
Coeficiente de partição	0.53975
Coeficiente de entropia	0.65222
Xie-Beni	0.72997

Desta forma, analisando os resultados e os índices de validação, chegou-se a conclusão que o algoritmo FCM teve bons resultados, mesmo não definindo os *clusters* gerados com exatidão. Isto, devido a sua dificuldade em identificar *clusters* de diferentes

formas, sendo que a distância Euclidiana utilizada pelo FCM, detecta *clusters* esféricos e os *clusters* encontrados foram em forma de elipse.

6.3.2 Resultados Obtidos pelo Algoritmo Robust C-Prototypes

A análise dos resultados encontrados pelo algoritmo RCP foi realizada utilizando os mesmos atributos de entrada, com os seguintes parâmetros:

a) **quantidade de *clusters***: 2;

b) **parâmetro de *fuzzyficação***: 2;

c) **quantidade de iterações**: 0 (ilimitado);

d) **abrangência da função de peso**: 12;

Assim como os atributos de entrada, o objetivo foi o mesmo que o apresentado na análise do algoritmo FCM. A Tabela 14 apresenta o resultado encontrado, por meio da qual é possível verificar a identificação correta dos grupos.

Tabela 14. *Clusters* encontrados pelo RCP na base de pacientes com sepse

Cluster	**Quantidade de elementos**	**Porcentagem dos elementos**	***desfecho***
1	51	53,12%	1
2	45	46,88%	2

O gráfico gerado pelo algoritmo RCP apresenta dois grupos bem definidos. Isto, devido às funções para diminuir os ruídos nos dados, e a distância de Mahalanobis, que possibilita encontrar *clusters* em forma de elipse.

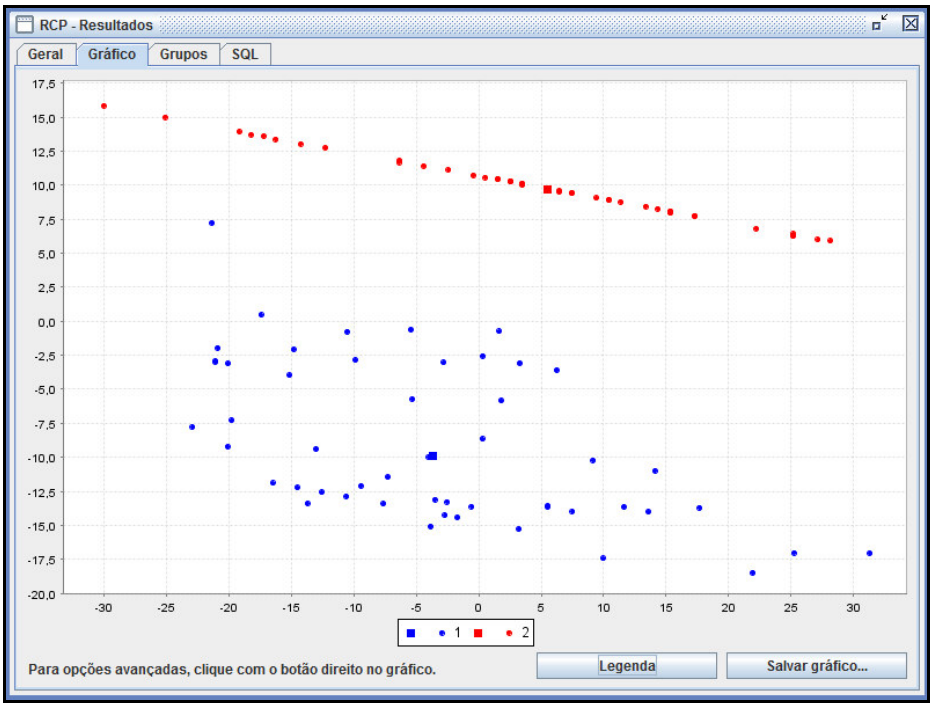

Figura 30. Gráfico gerado pelo algoritmo RCP

Ponderando os índices de validação (Tabela 15) chega-se a conclusão que o algoritmo foi capaz de extrair os grupos corretamente. Sendo que os coeficientes de partição e entropia apresentaram melhores resultados em relação a partição dos dados. O índice Xie-Beni, mesmo tendo um valor maior do que o encontrado pelo FCM indicou bons resultados, sendo que o valor deste índice deve ser o menor possível, modificando seus parâmetros de entrada.

Tabela 15. Índices de validação encontrados para o algoritmo RCP

Índice	Valor encontrado
Coeficiente de partição	0.8187
Coeficiente de entropia	0.27588
Xie-Beni	5.66131

Analisando os resultados encontrados pode-se constatar melhores resultados com o algoritmo RCP, definindo os grupos com exatidão, em relação aos *clusters* encontrados pelo FCM.

6.3.3 Resultados Obtidos pelo Algoritmo Unsupervised Robust C-Prototypes

Os parâmetros utilizados na análise foram os seguintes, com os mesmos atributos de entrada utilizados nas outras avaliações:

a) **quantidade de *clusters***: 2 e 3;

b) **parâmetro de *fuzzyficação***: 2;

c) **quantidade de iterações**: 0 (ilimitado);

d) **abrangência da função de peso**: 12;

e) **taxa de erro**: 0.25;

Utilizando o mesmo atributo de saída (*desfecho*), encontrou-se os resultados apresentados na Tabela 16, sendo que se chega ao mesmo resultado ao definir a quantidade de *clusters* como 2 ou 3.

Tabela 16. *Clusters* encontrados pelo URCP na base de pacientes com sepse

Cluster	**Quantidade de elementos**	**Porcentagem dos elementos**	***Desfecho***
1	51	53,12%	1
2	45	46,88%	2

O gráfico gerado foi o mesmo apresentado na Figura 31, com os grupos bem definidos. Ao efetuar a análise dos índices percebeu-se os mesmos valores encontrados pelo algoritmo RCP (Tabela 15), já que o URCP complementa o RCP e seu resultado diferencia por tentar encontrar o número ideal de *clusters*.

Concluindo a análise dos aos resultados obtidos por cada algoritmo, foram realizados testes, com diferentes parâmetros de entrada, para verificar os tempos de processamento.

6.3.4 Tempos de Processamento

Os testes de desempenho foram feitos com uma base de dados gerada aleatoriamente, contendo 10000 registros e 4 atributos. A avaliação dos tempos de processamento, para cada algoritmo, foi feita com diferentes quantidades de *clusters* e atributos selecionados, deixando o algoritmo definir a quantidade de iterações, por meio de seu erro. As Tabelas 17, 18 e 19 apresentam, respectivamente, os resultados obtidos pelos algoritmos FCM, RCP e URCP.

Tabela 17. Tempos de processamento para o algoritmo FCM

Quantidade de *clusters*	**Paramêtro de *Fuzzyficação***	**Atributos de entrada**	**Número de iterações**	**Tempos de processamento**
2	2	2	9	00min:20s.697ms
2	2	3	13	00min:22s.123ms
2	2	4	10	00min:27s.402ms
3	2	2	11	00min:43s.399ms
3	2	3	17	01min:07s.298ms
3	2	4	11	01min:31s.954ms
5	2	2	15	02min:07s.183ms
5	2	3	18	02min:37s.003ms
5	2	4	22	03min:04s.143ms

Tabela 18. Tempos de processamento para o algoritmo RCP

Quantidade de *clusters*	**Paramêtro de *Fuzzyficação***	**Atributos de entrada**	**Número de iterações**	**Tempos de processamento**
2	2	2	30	00min:00s.598ms
2	2	3	15	00min:00s.205ms
2	2	4	18	00min:00s.262ms
3	2	2	89	00min:03s.675ms
3	2	3	77	00min:03s.604ms
3	2	4	58	00min:03s.275ms
5	2	2	111	00min:05s.476ms
5	2	3	270	00min:14s.087ms
5	2	4	100	00min:05s.585ms

Tabela 19. Tempos de processamento para o algoritmo URCP

Quantidade de *clusters*	**Paramêtro de *Fuzzyficação***	**Atributos de entrada**	**Número de iterações**	**Tempos de processamento**
2	2	2	1	00min:00s.515ms
2	2	3	1	00min:00s.468ms
2	2	4	1	00min:00s.640ms
3	2	2	2	00min:12s.480ms
3	2	3	2	00min:02s.402ms
3	2	4	2	00min:02s.512ms
5	2	2	2	00min:13s.431ms
5	2	3	4	00min:16s.926ms
5	2	4	3	00min:11s.498ms

A análise dos tempos de processamento foi realizada com diferentes valores para o parâmetro *fuzzyficador*, estes resultados são apresentados nas Tabelas 21, 22 e 23, para os algoritmos FCM, RCP e URCP, respectivamente.

Tabela 20. Tempos de processamento com outros valores de *fuzzyficação* para o algoritmo FCM

Quantidade de *clusters*	**Paramêtro de *Fuzzyficação***	**Atributos de entrada**	**Número de iterações**	**Tempos de processamento**
2	1.5	2	18	00min:22s.293ms
2	2.5	2	35	00min:25s.423ms
2	1.5	3	20	00min:24s.478ms
2	2.5	3	23	00min:33s.288ms
2	1.5	4	22	00min:38s.189ms
2	2.5	4	39	00min:40s.489ms

Tabela 21. Tempos de processamento com outros valores de *fuzzyficação* para o algoritmo RCP

Quantidade de *clusters*	**Paramêtro de *Fuzzyficação***	**Atributos de entrada**	**Número de iterações**	**Tempos de processamento**
2	1.5	2	23	00min:00s.868ms
2	2.5	2	38	00min:01s.660ms
2	1.5	3	21	00min:01s.263ms
2	2.5	3	26	00min:01s.753ms
2	1.5	4	19	00min:01s.736ms
2	2.5	4	47	00min:04s.642ms

Tabela 22. Tempos de processamento com outros valores de *fuzzyficação* para o algoritmo URCP

Quantidade de *clusters*	**Paramêtro de *Fuzzyficação***	**Atributos de entrada**	**Número de iterações**	**Tempos de processamento**
2	1.5	2	1	00min:00s.770ms
2	2.5	2	1	00min:02s.226ms
2	1.5	3	1	00min:01s.256ms
2	2.5	3	1	00min:03s.374ms
2	1.5	4	1	00min:07s.183ms
2	2.5	4	1	01min:02s.047ms

Por fim, para os algoritmos RCP e URCP, foram analisados os resultados de acordo com o parâmetro que define a abrangência da função de peso, sendo que os resultados são apresentados nas Tabelas 24 e 25, para os algoritmos RCP e URCP, respectivamente:

Tabela 23. Tempos de processamento com outros valores para definir a função de peso para o algoritmo RCP

Quantidade de *clusters*	**Paramêtro de *Fuzzyficação***	**Atributos de entrada**	**Número de iterações**	**Abrangência da função de peso**	**Tempos de processamento**
2	2	2	29	10	00min:00s.825ms
2	2	2	43	20	00min:01s.124ms
2	2	2	32	30	00min:00s.568ms
2	2	4	30	10	00min:00s.921ms
2	2	4	18	20	00min:00s.443ms
2	2	4	53	30	00min:00s.978ms

Tabela 24. Tempos de processamento com outros valores para definir a função de peso para o algoritmo RCP

Quantidade de *clusters*	**Paramêtro de *Fuzzyficação***	**Atributos de entrada**	**Número de iterações**	**Abrangência da função de peso**	**Tempos de processamento**
2	2	2	1	10	00min:00s.387ms
2	2	2	1	20	00min:00s.522ms
2	2	2	1	30	00min:00s.527ms
2	2	4	1	10	00min:00s.437ms
2	2	4	1	20	00min:15s.633ms
2	2	4	1	30	00min:01s.181ms

Ao analisar os resultados obtidos pode-se observar os seguintes aspectos em relação aos parâmetros de entrada:

a) **quantidade de *clusters***: deve-se determinar com precisão a quantidade de grupos, sendo que quanto maior o valor deste parâmetro, maior será o tempo de processamento, isto em relação a qualquer dos algoritmos desenvolvidos;

b) **parâmetro de *fuzzyficação***: o valor padrão 2 foi o que obteve partições mais precisas e estáveis. Sendo que ao determinar outro valor os tempos de processamento aumentaram, e ao executar o algoritmo mais vezes, com os mesmos parâmetros de entrada, obteve-se poucas distorções nos resultados;

c) **abrangência da função de peso**: este parâmetro é informado para os algoritmos RCP e URCP. Foi verificado que deve-se aumentar este valor quando a base em questão possui dados de qualidade. Ao aplicar o algoritmo em uma base contaminada com muitos ruídos, este valor deve ser diminuído, fazendo com que os protótipos referentes aos centros dos *clusters* e matrizes de covariância não utilizem determinados registros que possam prejudicar no conhecimento extraído pelo método. A definição deste valor é muito importante para obter bons resultados, sendo que poucas diferenças em seu valor, não modificam ou modificam muito pouco, os resultados finais;

d) **taxa de erro**: a taxa de erro como condição de parada do FCM obteve resultados satisfatórios, sem a necessidade de executar outras iterações. A taxa de erro informada ao algoritmo URCP, que indica o mínimo de similaridade que os grupos obtidos devem ter para que estes sejam fundidos, foi utilizado o padrão 0.3. O seu valor depende muito da qualidade dos dados, sendo que se os grupos a serem fundidos forem distantes este valor deve ser maior;

e) **quantidade de atributos de entrada**: deve-se utilizar apenas os atributos relevantes na determinação dos grupos, pois o tempo de processamento aumenta gradativamente de acordo com o número de atributos selecionados.

Considerando os testes realizados em relação ao tempo de processamento, e aos resultados obtidos por cada algoritmo, pode-se concluir o correto funcionamento do FCM, RCP e URCP. Estes algoritmos apresentaram bons resultados, na aplicação da base de dados referente à sepse no processo de clusterização, sendo que os algoritmos RCP e URCP apresentaram os melhores resultados.

CONCLUSÃO

O processo de descoberta de conhecimento em bases de dados, considerando suas diversas tarefas e métodos, é de extrema utilidade, possibilitando às organizações extrair conhecimento de suas bases, objetivando o auxílio à tomada de decisão.

Entre as tarefas e métodos existentes esta pesquisa fundamentou-se na tarefa de clusterização, utilizando o método de lógica *fuzzy*, especificamente os algoritmos Robust C-Prototypes, Unsupervised Robust C-Prototypes e *Fuzzy* C-Means. Os quais possibilitam aos elementos pertencerem a diversos grupos, considerando suas semelhanças, gerando resultados melhores, se comparado a métodos tradicionais. A tarefa de clusterização, por ser não supervisionada, necessita de métodos de validação que indicam a qualidade do agrupamento dos dados, assim foram implementados os índices de validação: coeficiente de partição, de entropia e Xie-Beni, que auxiliam na avaliação dos resultados encontrados pela tarefa.

No decorrer da pesquisa algumas dificuldades foram encontradas, especialmente com relação ao entendimento do funcionamento dos algoritmos, devido a carência de bibliografia. No entanto, as dificuldades foram superadas e os objetivos foram alcançados.

Após realizar a implementação dos algoritmos RCP, URCP e FCM na *Shell* Orion foram realizados testes. Estes foram analisados em conjunto com os índices de validação, gerando resultados satisfatórios e indicando o correto particionamento dos dados, que validaram o funcionamento dos módulos desenvolvidos.

Também foram avaliados os tempos de processamento de cada algoritmo, onde pode-se observar que a precisão dos parâmetros interfere, tanto nos tempos de processamento, quanto nos seus resultados.

Considerando a pesquisa realizada, a seguir, são descritas algumas sugestões de trabalhos futuros com o objetivo de dar continuidade ao projeto da *Shell* Orion *Data Mining Engine*:

a) implementar novos algoritmos de clusterização pelo método de lógica *fuzzy*, como por exemplo o Robust Competitive Agglomeration (RCA), que utiliza outras técnicas para lidar com o problema dos ruídos na tarefa de clusterização, sendo que ele foi baseado no RCP desenvolvido nesta pesquisa;
b) desenvolver outras etapas relacionadas a descoberta de conhecimento, como o pré-processamento, para preparação e transformação dos dados;
c) implementar outros métodos de clusterização, como por exemplo, o hierárquico;
d) realizar uma pesquisa que utilize o resultado dos algoritmos desenvolvidos em outras tarefas de *data mining*, como por exemplo, a classificação;

REFERÊNCIAS

ABONYI, J.; BABUSKA, R.; SZEFEIFERT, F. **Modefied Gath-Geva fuzzy clustering for identification of Takagi-Sugeno Fuzzy Models.** IEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics, vol. 32, n. 5, 2002. p. 612-621.

ABONYI, J.; FEIL, B. **Cluster Analysis for Data Mining and System Identification.** Boston: Birkhäuser Verlag, 2007.

BASU, Sugato; DAVIDSON, Ian; WAGSTAFF, Kiri L. **Constrained Clustering: Advances in Algorithms, Theory, and Applications.** Nova York: Chapman & Hall/CRC, 2008.

BERRY, Michael J.; LINOFF, Gordon. **Data mining techniques:** for marketing, sales and customer relationship management. Indianapolis; Wiley Publishing, 2004.

BEZDEK, James et al. **Fuzzy models and algorithms for pattern recognition and image processing**. New York: Springer, 2005.

BORTOLOTTO, Leandro Sehnem. **O Método de Redes Neurais pelo Algoritmo Kohonen para Clusterização na Shell Orion Data Mining Engine**. Trabalho de Conclusão de Curso - Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciuma, Santa Catarina, 2007.

CECIL, Russell L.; GOLDMAN, Lee; AUSIELLO, Dennis A. **Cecil**: tratado de medicina interna. Rio de Janeiro: Elsevier, 2005.

CARVALHO, Luís Alfredo Vidal de. **Datamining**: a mineração de dados no marketing, medicina, economia, engenharia e administração. Rio de Janeiro: Ciência Moderna, 2005.

CASAGRANDE, Diego Paz. **O Módulo da Técnica de Associação pelo Algoritmo Apriori no desenvolvimento da Shell de Data Mining Orion.** Trabalho de Conclusão de Curso – Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina, 2005.

CASSETTARI JUNIOR, José Márcio. **O Método de Lógica Fuzzy pelo Algoritmo Gustafson-Kessel na Tarefa de Clusterização da Shell Orion Data Mining Engine.** 2008. Trabalho de Conclusão de Curso - Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciuma, Santa Catarina, 2008.

CHI, Zheru; YAN, Hong; PHAM, Tuan. **Fuzzy Algorithms:** with Applications to Image Processing and Pattern Recognition. London: World Scientific, 1996.

COX, Earl. **Fuzzy Modeling and Genetic Algorithms for Data Mining and Exploration.** San Francisco: Morgan Kaufmann, 2005.

DÖRING, Christian; LESOT, Marie-Jeanne; KRUSE, Rudolf. **Data Analysis with Fuzzy Clustering Methods.** Computational Statistics & Data Analysis, Volume 51(1):192-214, 2006.

FAYYAD, Usama; PIATETSKY-SHAPIRO, Gregory; SMYTH, Padhraic. From Data Mining to Knowledge Discovery in Databases. 1996. **AI Magazine**. Menlo Park, 1996. Disponível em: <http://kdnuggets.com/gpspubs/aimag-kdd-overview-1996-Fayyad.pdf > Acesso em: 19 ago. 2009.

FRIGUI, Hichem; KRISHNAPURAM, Raghu. **A Robust Algorithm for Automatic Extraction of an Unknown Number of Clusters from Noisy Data.** Pattern Recognition Letters 17. p. 1223-1232, 1996.

GALINDO, Jose; URRUTIA, Angelica; PIATTINI, Mario. **Fuzzy Databases: Modeling, Design and Implementation.** London: Idea Group Publishing, 2006.

GAN, Guojun; MA, Chaoqun; WU, Jianhong. **Data Clustering: Theory, Algorithms, and Applications.** Philadelphia: SIAM, Society for Industrial and Applied Mathematics, 2007.

GOLDSCHMIDT, Ronaldo; PASSOS, Emmanuel. **Data Mining:** um guia prático. Rio de janeiro: Elsevier, 2005.

GRIFFITHS, Peter R.; DE HASETH, James A. **Fourier transform infrared spectrometry.** New Jersey: Wiley-Interscience, 2007.

GUEDES, Gilleanes T. A. **UML**: uma abordagem prática.São Paulo: Novatec, 2008.

HAN, Jiawei; KAMBER, Micheline. **Data Mining:** concepts and techniques. San Diego: Academic Press, 2001.

HAYKIN, Simon. **Redes neurais**: princípios e pratica. 2. ed. Porto Alegre: Bookman, 2001.

HÖPPNER, Frank; KLAWONN, Frank; KRUSE, Rudolf; RUNKLER, Thomas. **Fuzzy cluster analysis**: methods for classification, data analysis, and image recognition. Chichester: John Wiley & Sons, 1999.

JIN, J. S. et al. **Using browsing to improve content-based image retrieval**. Journal of Visual Communication and Image Representation 12. p. 123–135, 2001.

KANTARDZIC, Mehmed. **Data Mining: Concepts, Models, Methods, and Algorithms**. John Wiley e Sons. 2003.

KIM, Y. et al. **A cluster validation index for GK cluster analysis based on relative degree of sharing.** Information Sciences, Vol. 168, p. 255-242, 2004.

LAROSE, Daniel T. **Discovering Knowledge in Data**: An Introduction to Data Mining. New Jersey: John Wiley & Sons, 2005.

LV, Qi; ZHANG, Xian-Da.**A unified method for blind separation of sparse sources with unknown source number.** IEEE Signal Processing Letters, vol. 13, p. 49-51, 2006.

MARTINS, Denis Piazza. **O Algoritmo de Particionamento K-means na Tarefa de Clusterização da Shell Orion Data Mining Engine.** Trabalho de Conclusão de Curso – Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina, 2007.

MONDARDO, Ricardo Lineburger. **O algoritmo C4.5 na Tarefa de Classificação na Shell Orion Data Mining Engine.** Trabalho de Conclusão de Curso – Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina, 2009.

NITSUWAT, S. JIN, J. S. **Analysing Motion Parameters Using Unsupervised Fuzzy C-Prototypes**. Proceedings of the Pan-Sydney Area Workshop on Visual Information Processing. p. 17-33. Sydney, Australia, 1998.

OLIVEIRA JUNIOR, Hime Aguiar e; CALDEIRA, André Machado. **Inteligência computacional aplicada à administração, economia e engenharia em Matlab.** São Paulo: Thomson, 2007.

OLIVEIRA, José Valente de; PEDRYCZ, Witold. **Advances in Fuzzy Clustering and Its Applications.** England: John Wiley & Sons, 2007.

OLSON, David L.; DELEN, Dursun. **Advanced Data Mining Techniques.** Lincoln: Springer, 2008.

PELEGRIN, Diana Colombo. **A Tarefa de Classificação e o Algoritmo ID3 para Indução de Árvores de Decisão na Shell de Data Mining Orion.** Trabalho de Conclusão de Curso – Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina, 2005.

PEREGO, Daniel. **O Método de Lógica Fuzzy pelo Algoritmo Gath-Geva na Tarefa de Clusterização da Shell Orion Data Mining Engine.** Trabalho de Conclusão de Curso - Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciuma, Santa Catarina, 2009.

POOLE, David; MONTEIRO, Martha Salerno. **Algebra linear**. São Paulo: Thomson 2004.

RAIMUNDO, Lidiane Rosso. **O Algoritmo CART na Tarefa de Classificação da Shell Orion Data Mining Engine.** 2007. Trabalho de Conclusão de Curso – Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina, 2007.

REZENDE, Solange Oliveira. **Sistemas inteligentes:** fundamentos e aplicações. São Paulo: Manole, 2003.

SATO-ILIC, Mika; JAIN, Lakhmi C. **Innovations in Fuzzy Clustering**: Theory and Application. Berlin: Springer, 2006.

SEISING, Rudolf. **Views on Fuzzy Sets and Systems from Different Perspectives: Philosophy and Logic.** Berlin: Springer, 2009.

SURINTA, Olarik; JAREANPON, Chatklaw. **Comparison of Image Analysis for Thai Handwritten Character Recognition.** Intelligent Information Processing. p. 373-382, 2006.

YAO, Lv; SHUANGTIAN, Li. **Underdetermined Blind Source Separation of anechoic speech mixtures in the Time-Frequency domain.** Signal Processing, 2008. ICSP 2008. 9th International Conference on , vol., no., pp.22-25, 26-29.

WITTEN, Ian H.; FRANK, Eibe. **Data mining:** practical machine learning tools and techniques. San Francisco: Morgan Kaufmann, 2005.

APÊNDICE A - O ALGORITMO *FUZZY* C-MEANS

O algoritmo *Fuzzy* C-Means (FCM) foi o primeiro a utilizar o conceito de pertinência em relação aos elementos e grupos, sendo um dos algoritmos mais utilizados no método de lógica *fuzzy* para a tarefa de clusterização. A versão final do algoritmo foi proposta por Bezdek, em 1973, em sua tese *Fuzzy Mathematics in Pattern Classification* (HÖPPNER et al, 1999, tradução nossa).

O FCM pode ser considerado a versão *fuzzy* do tradicional algoritmo *K-means*, portanto, da mesma forma que o *K-means*, este algoritmo utiliza a distância Euclidiana encontrando c*lusters* de forma circular. Assim, o FCM minimiza a seguinte função objetivo (BEZDEK et al, 2005, tradução nossa):

$$J(B,U;Z) = \sum_{i=1}^{C}\sum_{j=1}^{N}\left(\mu_{ij}\right)^{m}\left(d_{ij}^{2}\right) \tag{25}$$

Onde:

a) ***C***: número total de *clusters*;

b) ***N***: número total de elementos;

c) ***J***: valor a ser minimizado;

d) ***B***: conjunto de *clusters*;

e) ***U***: matriz de pertinências;

f) ***Z***: conjunto de dados;

g) ***m***: parâmetro de *fuzzyficação*;

h) $\boldsymbol{\mu_{ij}}$: grau de pertinência do i-ésimo *cluster* e o j-ésimo elemento;

i) $\boldsymbol{d^2_{ij}}$: distância entre o i-ésimo *cluster* e o j-ésimo elemento.

O algoritmo, primeiramente, inicializa a matriz de pertinências, para então calcular iterativamente as seguintes etapas:

a) determinar os centros dos *clusters*;

b) calcular as distâncias entre elementos e seus grupos;

c) atualizar os graus de pertinência;

d) verificação da condição de parada do algoritmo.

A fim de compreender melhor o entendimento do algoritmo, estas etapas estão descritas, individualmente, a seguir.

CÁLCULO DOS CENTROS DOS *CLUSTERS*

Os grupos são representados por um centro, que é utilizado como referência a um determinado *cluster* para calcular as distâncias entre os elementos e grupos. O algoritmo FCM utiliza a seguinte equação para encontrar seus centros, sendo que o cálculo é realizado para cada *cluster* a ser encontrado na base de dados (BEZDEK et al, 2005, tradução nossa):

$$c_i = \frac{\sum_{j=1}^{N} \left(\mu_{ij}\right)^m x_j}{\sum_{j=1}^{N} \left(u_{ij}\right)^m} \tag{26}$$

Onde:

a) c_i: centro do i-ésimo *cluster;*

b) μ_{ij}: grau de pertinência entre o i-ésimo cluster e o j-ésimo elemento;

c) x_j: j-ésimo elemento.

Após o cálculo dos centros deve-se determinar as distâncias entre os elementos e o centro encontrado.

CÁLCULO DAS DISTÂNCIAS ENTRE ELEMENTOS E CLUSTERS

As distâncias definem numericamente a relação entre os elementos e grupos. A distância utilizada pelo FCM é a Euclidiana, definida pela equação (BEZDEK et al, 2005, tradução nossa):

$$d_{ij}^2 = \sqrt{\sum_{j=1}^{N} \left(x_j - c_i\right)^2} \tag{27}$$

Onde:

a) c_i: centro do i-ésimo *cluster;*

b) d_{ij}: distância entre o i-ésimo *cluster* e o j-ésimo elemento;

c) x_j: j-ésimo elemento;

Por meio das distâncias encontradas pode-se atualizar a matriz de pertinências.

ATUALIZAÇÃO GRAUS DE PERTINÊNCIA

A definição de quanto um elemento pertence a um determinado grupo é dada por (BEZDEK et al, 2005, tradução nossa):

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{C} \left[\frac{d_{ij}^2}{d_{kj}^2} \right]^{\frac{1}{(m-1)}}} \tag{28}$$

Onde:

a) μ_{ij}: grau de pertinência do i-ésimo *cluster* e o j-ésimo elemento;

b) m: parâmetro de *fuzzyficação*;

c) d^2_{ij}: distância entre o i-ésimo *cluster* e o j-ésimo elemento;

d) d^2_{ik}: distância entre o i-ésimo *cluster* e o k-ésimo elemento.

Calcula-se o somatório da divisão entre o elemento do atual *cluster* com todos os outros elevados a um dividido pelo elemento *fuzzyficador* menos um. Após isto, o valor um é dividido pelo valor encontrado.

Ao final da atualização das pertinências deve-se calcular verificar se os valores obtidos já são suficientes para encerrar o algoritmo.

CÁLCULO DA CONDIÇÃO DE PARADA

A condição de parada define o momento em que o algoritmo atingiu os resultados esperados, por meio da taxa de erro fornecida como parâmetro. A equação (4) define a condição de parada do algoritmo (BEZDEK et al, 2005, tradução nossa):

$$\left\| U^{l} - U^{l-1} \right\| \leq \varepsilon \tag{29}$$

Onde:

a) U: matriz de pertinências;

b) l: numeração da iteração atual;

c) ε: taxa de erro.

Quando algum valor absoluto da subtração entre a matriz de pertinências da iteração atual e a anterior for menor ou igual a taxa de erro, o algoritmo termina sua execução. O algoritmo continua executando enquanto a condição de parada for falsa.

APÊNDICE B

O Método de Lógica *Fuzzy* pelos Algoritmos Robust C-Prototypes e Unsupervised Robust C-Prototypes para a Tarefa de Clusterização na *Shell* Orion *Data Mining*

Ademar Crotti Junior[1], Merisandra Côrtes de Mattos[2]

jcrotti@gmail.com, mem@unesc.net

[1] Acadêmico do Curso de Ciência da Computação – Unidade Acadêmica de Ciências, Engenharias e Tecnologias – Universidade do Extremo Sul Catarinense (UNESC) – Criciúma – SC

[2] Professora do Curso de Ciência da Computação – Unidade Acadêmica de Ciências, Engenharias e Tecnologias – Universidade do Extremo Sul Catarinense (UNESC) – Criciúma – SC

Resumo. *Os avanços tecnológicos facilitaram, às mais diversas organizações, o armazenamento de grandes bases de dados. Desta forma, para seu próprio benefício, surgiu a necessidade de análise desta informação. O data mining, principal etapa do processo de descoberta de conhecimento, é responsável por encontrar padrões e relações nos dados, funcionando como auxílio para a tomada de decisão. Este artigo descreve a modelagem matemática e a implementação dos algoritmos de lógica fuzzy Robust C-Prototypes e Unsupervised Robust C-Prototypes, para a tarefa de clusterização. Esta tarefa é responsável por agrupar uma base de dados considerando a semelhança entre os dados, sendo que o método de lógica fuzzy melhora os resultados encontrados por possibilitar aos elementos a pertencerem a mais de cluster simultaneamente.*

1. Introdução

A busca por padrões nos dados ocorre na etapa de *data mining* (DM), sendo esta a principal da Descoberta de Conhecimento em Bases de dados (DCBD). Nela são empregados tarefas e métodos com características distintas, que são aplicados de acordo com o objetivo da descoberta de conhecimento [GOLDSCHMIDT; PASSOS, 2005; HÖPPNER et al, 1999].

As tarefas, métodos e algoritmos de DM são implementados em ferramentas denominadas *shells*. Existem diversas destas ferramentas no mercado, sendo que a maioria é comercial e de alto custo, havendo carência de ferramentas gratuitas.

Dentre estas ferramentas, existe em desenvolvimento a *Shell* Orion *Data Mining Engine*, um projeto acadêmico implementado pelo Grupo de Pesquisa em Inteligência Computacional Aplicada, do Curso de Ciência da Computação, na Universidade do Extremo Sul Catarinense.

Desta forma, esta pesquisa consiste no desenvolvimento dos algoritmos RCP e URCP, no módulo de clusterização da *Shell* Orion.

2. Tarefa de Clusterização pelo Método de Lógica *Fuzzy*

A clusterização é o processo de agrupar dados em grupos de objetos similares. Um *cluster* é um conjunto de dados, onde cada elemento integrante seja similar a outros no mesmo *cluster*; e elementos em diferentes *clusters* são distintos [HAN; KAMBER, 2001].

Os métodos tradicionais forçam os dados a pertencerem a apenas um *cluster*. Sendo que isto pode não representar a realidade da base de dados. Os métodos de clusterização baseados em lógica *fuzzy* conseguem melhores resultados nestes casos, pois um determinado elemento, ao mesmo tempo, pode estar em mais de um *cluster*, com diferentes graus de pertinência.

Existem diversos algoritmos que utilizam lógica fuzzy para a tarefa de clusterização, porém, o desempenho deles é insuficiente quando o conjunto de dados está contaminado por ruídos e *outliers*. Isto prejudica a execução da tarefa, e conseqüentemente causa distorções no conhecimento gerado. Assim, surgiu uma nova classe de algoritmos, que utilizam estatísticas denominadas de robustas, com o objetivo de diminuir a interferência dos dados ruidosos e *outliers* na execução da tarefa [FRIGUI; KRISHAPURAM, 1996].

3. Os Algoritmos Robust C-Prototypes e Unsupervised Robust C-Prototypes

Hichem Frigui e Raghu Krishnapuram, em 1996, publicaram o artigo *A Robust Algorithm for Automatic Extraction of an Unknown Number of Clusters from Noisy Data*, na *Pattern Recognition Letters*, neste artigo foram apresentados os algoritmos Robust C-Prototypes (RCP) e Robust C-Prototypes (URCP).

A vantagem destes algoritmos é a utilização de funções de peso e perda, utilizadas para diminuir a interferência de ruídos e *outliers* no processo de clusterização. Em geral, o resultado destas funções para um dado normal tem valor próximo de 1, enquanto ruídos e *outliers* devem possuir valores menores. Considerando que estas funções podem ser combinadas com outras medidas para a identificação de *clusters* de diferentes formas [FRIGUI; KRISHAPURAM, 1996].

O artigo apresentado descreve os algoritmos RCP e URCP para a identificação de *clusters* em forma de elipse, utilizando uma matriz de covariância, sendo esta robusta, devido a utilização de funções responsáveis por diminuir a influência de dados ruidosos [OLIVEIRA; PEDRYCZ, 2007].

3.1. Modelagem Matemática do Algoritmo Robust C-Prototypes

A Tabela 1 apresenta a base de dados utilizada na demonstração dos cálculos.

Tabela 25. Base de dados utilizada na modelagem dos algoritmos

Atributos	x_1	x_2	x_3	x_4
a	9	8	7	7,5
b	1	1,8	2	1,5
c	2	3	2,5	4

Os parâmetros do algoritmo utilizados nesta demonstração foram:

a) **quantidade de clusters**: o valor utilizado na demonstração foi 2;

b) **parâmetro de *fuzzyficação* (*m*)**: na demonstração foi utilizado o valor 2;

c) **α**: define a abrangência da função de peso, sendo utilizado o valor 3.5. Este parâmetro deve ser definido de acordo com a qualidade da base de dados em relação a ruídos e *outliers*, quanto maior o seu valor mais pesos não zero existirão, quanto menor o seu valor menos atributos serão visíveis ao processo de definição dos protótipos de centros dos clusters e matrizes de covariância *fuzzy*.

Primeiramente, deve-se iniciar os centros dos clusters e as matrizes de covariância *fuzzy*. Os centros são iniciados randomicamente, as matrizes de covariância também podem ser iniciadas randomicamente ou defini-se uma matriz identidade para início dos cálculos. Na implementação os centros são valores randômicos da base de dados, e as matrizes são identidades. Nesta demonstração tanto os centros como as matrizes foram iniciadas

randomicamente, a Tabela 2 apresenta os centros, e a seguir são apresentadas as matrizes de covariância *fuzzy*.

Tabela 26. Centros dos clusters iniciados randomicamente

Atributos	c_1	c_2
a	6	3
b	7	6
c	4	1,5

$$C_1 = \begin{bmatrix} 5 & 2 & 1 \\ 1 & 4 & 2 \\ 2 & 3 & 2 \end{bmatrix} \qquad C_2 = \begin{bmatrix} 4 & 3 & 2 \\ 2 & 5 & 5 \\ 3 & 2 & 3 \end{bmatrix}$$

Assim, com os centros e as matrizes de covariância definidos pode-se iniciar o algoritmo RCP com o cálculo da distância de Mahalanobis, definida pela seguinte equação:

$$d_{ij}^2 = \left(x_j - c_i\right)^T M_i \left(x_j - c_i\right)$$

Onde:

$$M_i = \left|C_i\right|^{\frac{1}{n}} C_i^{-1}$$

Considerando a equação acima, inicialmente deve-se encontrar a matriz de covariância modificada M_i, sendo necessário primeiramente calcular o determinante da matriz de covariância.

$$C_1 = \begin{bmatrix} 5 & 2 & 1 \\ 1 & 4 & 2 \\ 2 & 3 & 2 \end{bmatrix}$$

$$\left|C_1\right| = 9$$

O próximo passo é calcular a matriz inversa, sendo esta definida pela adjunta dividida pelo seu determinante. Onde uma matriz adjunta é definida por todos os seus determinantes possíveis.

Substituindo os valores pelos correspondentes para a matriz de covariância do primeiro *cluster* tem-se:

$$C^{-1} = \frac{1}{\left|C\right|} adj(C)$$

$$C_1^{-1} = \begin{bmatrix} 0{,}22222 & -0{,}11111 & 0 \\ 0{,}22222 & 0{,}88889 & -1 \\ 0{,}55556 & -1{,}22222 & 2 \end{bmatrix}$$

Após calcular o determinante e a matriz inversa pode-se então determinar a matriz de covariância modificada:

$$M_i = \left|C_i\right|^{\frac{1}{n}} C_i^{-1}$$

Resolvendo obtêm-se a matriz de covariância modificada M_1:

$$M_1 = \begin{bmatrix} 0{,}46223 & -0{,}23112 & 0 \\ 0{,}46223 & 1{,}84894 & -2{,}08007 \\ 1{,}15561 & -2{,}54231 & 4{,}16014 \end{bmatrix}$$

Enfim, o algoritmo pode então calcular as distâncias entre seus elementos e os centros dos *clusters*, sendo que para o primeiro elemento em relação ao primeiro *cluster* tem-se:

$$d_{11}^2 = \begin{bmatrix} 3 & -6 & -2 \end{bmatrix} \cdot \begin{bmatrix} 0{,}46223 & -0{,}23112 & 0 \\ 0{,}46223 & 1{,}84894 & -2{,}08007 \\ 1{,}15561 & -2{,}54231 & 4{,}16014 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ -6 \\ -2 \end{bmatrix}$$

$$d_{11}^2 = 20{,}80027$$

Realizando os mesmos cálculos para os outros elementos e clusters foram encontradas as seguintes distâncias (Tabela 3).

Tabela 27. Distâncias entre os elementos e os centros dos clusters

Distâncias	x_1	x_2	x_3	x_4
c_1	20,80027	27,25326	18,4892	55,06376
c_2	27,25184	26,64832	18,94756	38,65541

Após calcular as distâncias utiliza-se a seguinte propriedade para dividir os elementos em grupos:

$$Z_i = \left\{ x_j \middle| d_{ij}^2 \leq d_{kj}^2 \ \forall k \neq i \right\}$$

Cada i-ésimo grupo Z possuirá os elementos que estejam mais próximos ao centro do i-ésimo cluster. Aplicando a propriedade nas distâncias obtidas tem-se:

$$Z_1 = \{x_1, x_3\}$$

$$Z_2 = \{x_2, x_4\}$$

Tendo os grupos definidos pode-se estimar os valores de *T* e *S*. Onde *T* é definido por:

$$T_i = \underset{x_j \in Z_i}{Med}\left(d_{ij}^2\right)$$

O cálculo da mediana (*Med*) de um determinado conjunto é definido como o elemento central de um conjunto ordenado quando este conjunto tiver uma quantidade ímpar de elementos; quando a quantidade de elementos do conjunto for par, sua mediana é definida como a soma dos elementos centrais dividido por 2. Sendo que a quantidade de elementos do conjunto é um número par, tem-se:

$$T_1 = (20{,}80027 + 18{,}4892)/2$$

$$T_1 = 19{,}64474$$

$$T_2 = 32{,}65185$$

A estimação de S é definida por:

$$S_i = 1.418 \times \underset{x_j \in Z_i}{MAD}\left(d_{ij}^2\right)$$

Onde:

$$MAD = Med(|x - Med(X)|)$$

O desvio mediano absoluto é a mediana dos valores absolutos obtidos pelas subtrações entre os elementos do conjunto e a sua mediana. Sendo que a mediana dos conjuntos foi definido pelas estimações *T*:

$$MAD_1 = \left(|20{,}80027 - 19{,}64474|, |18{,}4892 - 19{,}64474|\right)$$

$$MAD_1 = \left(|1{,}15553|, |-1{,}15553|\right)$$

$$MAD_1 = 1{,}15553$$

Após obter o valor do desvio mediano absoluto calcula-se a estimação *S*:

$$S_1 = 1{,}418 \times 1{,}15553$$

$$S_1 = 1{,}63853$$
$$S_2 = 8{,}51305$$

O próximo passo é calcular as funções de peso que serão utilizadas no cálculo do centro do cluster. Esta função é definida por:

$$\omega_i\left(d_{ij}^2\right) = \begin{cases} 1 - \dfrac{d_{ij}^4}{2T_i^2}, se\ d_{ij}^2 \in [0, T_i] \\ \dfrac{\left[d_{ij}^2 - \left(T_i + \alpha S_i\right)\right]^2}{2\alpha^2 S_i^2}, se\ d_{ij}^2 \in (T_i, T_i + \alpha S_i] \\ 0, se\ d_{ij}^2 > T_i + \alpha S_i \end{cases}$$

Sendo que para o primeiro *cluster*, quando a distância estiver no intervalo entre 0 e 19.64474, a função de peso é calculada utilizando a primeira condição, se estiver entre 19.64474 e 25.37958 utiliza-se a segunda condição e quando seu valor for maior que 25.37958 seu resultado é 0. Realizando os cálculos chega-se a Tabela 4:

Tabela 28. Valores resultantes das funções de peso em relação às distâncias

Função de peso	x_1	x_2	x_3	x_4
w_1	0,31881	0	0,55709	0
w_2	0,65172	0,66696	0,83163	0,31881

O próximo passo é calcular a função de perda, sendo antes necessário calcular o valor da constante K.

$$K_i = \max_{1 \le j \le C}\left\{\frac{5T_j + \alpha S_j}{6}\right\} - \frac{5T_i + \alpha S_i}{6}$$

Substituindo os valores:

$$K_1 = \max\left\{\frac{5 \cdot 19{,}64474 + 3{,}5 \cdot 1{,}63853}{6}; \frac{532{,}65185 + 3{,}5 \cdot 8{,}51305}{6}\right\} - \frac{519{,}64474 + 3{,}5 \cdot 1{,}63853}{6}$$

$$K_1 = 14{,}84938$$

Então, é possível calcular a função de perda:

$$\rho_i\left(d_{ij}^2\right) = \begin{cases} d_{ij}^2 - \dfrac{d_{ij}^6}{6T_i^2}, se\ d_{ij}^2 \in [0, T_i] \\ \dfrac{\left[d_{ij}^2 - \left(T_i + \alpha S_i\right)\right]^3}{6\alpha^2 S_i^2} + \dfrac{5T_i + \alpha S_i}{6}, se\ d_{ij}^2 \in [T_i, T_i + \alpha S_i] \\ \dfrac{5T_i + \alpha S_i}{6} + K_i\ se\ d_{ij}^2 > T_i + \alpha S_i \end{cases}$$

São utilizados os mesmos intervalos da função de peso, portanto a primeira distância fica no segundo intervalo, assim a Tabela 5 apresenta os valores encontrados pela função de perda:

Tabela 29. Valores encontrados pela função de perda

Função de perda	x_1	x_2	x_3	x_4
ρ_1	15,86652	32,17581	15,75948	32,17581
ρ_2	24,08797	23,69001	17,88418	27,64745

Após calcular os valores das distâncias em relação a função de perda é possível calcular as pertinências entre os elementos e clusters, definida pela equação:

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{C}\left[\frac{\rho_i\left(d_{ij}^2\right)}{\rho_k\left(d_{kj}^2\right)}\right]^{\frac{1}{(m-1)}}}$$

Substituindo os valores referentes a função de perda do primeiro elemento e o primeiro cluster, obtem-se o seu grau de pertinência:

$$\mu_{11} = \frac{1}{\left[\frac{15{,}86652}{15{,}86652}\right]^{\frac{1}{(2-1)}} + \left[\frac{15{,}86652}{24{,}08797}\right]^{\frac{1}{(2-1)}}}$$

$$\mu_{11} = 0{,}60289$$

Realizando o mesmo processo aos outros elementos e clusters obtêm-se as pertinências apresentadas na Tabela 6.

Tabela 30. Graus de pertinência atualizados

Clusters	x_1	x_2	x_3	x_4
1	0,60289	0,42404	0,53157	0,47954
2	0,39712	0,57595	0,46843	0,52044

A próxima etapa do algoritmo RCP é calcular os centros dos clusters. Este é definido pela seguinte equação:

$$c_i = \frac{\sum_{j=1}^{N}\left(\mu_{ij}\right)^m w_{ij} x_j}{\sum_{j=1}^{N}\left(u_{ij}\right)^m w_{ij}}$$

Assim, encontra-se o primeiro centro do *cluster*, sendo calculado para cada grupo a ser encontrado na base de dados:

$$c_1 = \frac{\begin{array}{c}(0{,}60289)^2 \cdot 0{,}31881 \cdot \begin{bmatrix}9\\1\\2\end{bmatrix} + (0{,}42404)^2 \cdot 0 \cdot \begin{bmatrix}9\\1{,}8\\3\end{bmatrix} + \\ (0{,}53157)^2 \cdot 0{,}55709 \cdot \begin{bmatrix}7\\2\\2{,}5\end{bmatrix} + (0{,}47954)^2 \cdot 0 \cdot \begin{bmatrix}7{,}5\\1{,}5\\4\end{bmatrix}\end{array}}{(0{,}60289)^2 \cdot 0{,}31881 + (0{,}42404)^2 \cdot 0 + (0{,}53157)^2 \cdot 0{,}55709 + (0{,}47954)^2 \cdot 0}$$

$$c_1 = \begin{bmatrix}7{,}84788\\1{,}57593\\2{,}28795\end{bmatrix}$$

O último passo do algoritmo é calcular as matrizes de covariância *fuzzy* definidas pela equação:

$$C_i = \frac{\sum_{j=1}^{N}\left(\mu_{ij}\right)^m \omega_{ij}\left(x_j - c_i\right)\left(x_j - c_i\right)^T}{\sum_{j=1}^{N}\left(u_{ij}\right)^m \omega_{ij}}$$

Substituindo-se e resolvendo tem-se a matriz de covariância do primeiro *cluster*:

$$(0{,}60289)^2 \cdot 0{,}31881 \cdot \left(\begin{bmatrix} 9 \\ 1 \\ 2 \end{bmatrix} - \begin{bmatrix} 7{,}84788 \\ 1{,}57593 \\ 2{,}28795 \end{bmatrix} \right) \left([9 \quad 1 \quad 2] - [7{,}84788 \quad 1{,}57593 \quad 2{,}28795] \right) +$$

$$(0{,}42404)^2 \cdot 0 \cdot \left(\begin{bmatrix} 8 \\ 1{,}8 \\ 3 \end{bmatrix} - \begin{bmatrix} 7{,}84788 \\ 1{,}57593 \\ 2{,}28795 \end{bmatrix} \right) \left([8 \quad 1{,}8 \quad 3] - [7{,}84788 \quad 1{,}57593 \quad 2{,}28795] \right) +$$

$$(0{,}53157)^2 \cdot 0{,}55709 \cdot \left(\begin{bmatrix} 7 \\ 2 \\ 2{,}5 \end{bmatrix} - \begin{bmatrix} 7{,}84788 \\ 1{,}57593 \\ 2{,}28795 \end{bmatrix} \right) \left([7 \quad 2 \quad 2{,}5] - [7{,}84788 \quad 1{,}57593 \quad 2{,}28795] \right) +$$

$$(0{,}47954)^2 \cdot 0 \cdot \left(\begin{bmatrix} 7{,}5 \\ 1{,}5 \\ 4 \end{bmatrix} - \begin{bmatrix} 7{,}84788 \\ 1{,}57593 \\ 2{,}28795 \end{bmatrix} \right) \left([7{,}5 \quad 15 \quad 4] - [7{,}84788 \quad 1{,}57593 \quad 2{,}28795] \right)$$

$$C_1 = \frac{}{(0{,}60289)^2 \cdot 0{,}31881 + (0{,}42404)^2 \cdot 0 + (0{,}53157)^2 \cdot 0{,}55709 + (0{,}47954)^2 \cdot 0}$$

$$C_1 = \begin{bmatrix} 0{,}97686 & -0{,}48844 & -0{,}24421 \\ -0{,}48844 & 0{,}24424 & 0{,}12214 \\ -0{,}24421 & 0{,}12211 & 0{,}06107 \end{bmatrix}$$

Considerando que os valores dos centros dos clusters e das matrizes de covariância não estabilizaram, já que seus valores variaram muito de uma iteração a outra, o algoritmo RCP realizaria a próxima iteração.

3.2. Modelagem Matemática do Algoritmo Unsupervised Robust C-Prototypes

O algoritmo URCP pode ser considerado uma extensão do RCP, sendo que sua diferença é que ao terminar a clusterização dos dados, o algoritmo utiliza uma função similaridade para unir os *clusters* que atingirem a taxa de erro que deve ser informada por parâmetro [FRIGUI; KRISHAPURAM, 1996].

O primeiro passo do algoritmo é clusterizar a base utilizando o RCP, já demonstrado, após ele aplica o Unconstrained RCP, sendo este o RCP utilizando o valor de todas as pertinências como 1, ao final é aplicada a seguinte função de similaridade [FRIGUI; KRISHAPURAM, 1996]:

$$S_{ij} = 1 - \frac{\sum_{x_k \in G_{ij}} \left| \mu_{ik} - \mu_{jk} \right|}{\left| G_i \right| + \left| G_j \right|}$$

Onde:

$$| G_i | = \sum_{x_k \in G_i} \mu_{ik}$$

$$G_i = \{ x_k \in X \mid \omega_{ik} > 0 \}$$

$$G_{ij} = G_i \cup G_j = \{ x_k \in X \mid \omega_{ik} > 0 \, OR \, \omega_{jk} > 0 \}$$

Se a função de similaridade for maior ou igual ao valor da taxa de erro informada por parâmetro, os *clusters* serão unidos, sendo que esta é calculada para cada par de grupos e que o usuário deve informar a quantidade máxima de grupos existente na base de dados em questão.

4. Resultados

Os algoritmos RCP e URCP foram implementado na *Shell* Orion *Data Mining Engine* e testados utilizando uma base de dados contendo 96 registros e 45 atributos de pacientes com sepse da UTI do Hospital de Clínicas de Porto Alegre, Rio Grande do Sul. Os parâmetros de entrada do algoritmo foram os mesmos utilizados para a demonstração matemática do algoritmo, sendo selecionados alguns atributos de entrada a fim de obter-se os resultados do algoritmo.

Os resultados podem ser visualizados em forma gráfica e em forma de resumo pela aba *Geral* (Figura 1). Também obtem-se os grupos encontrados em uma estrutura de dados na aba *Grupos* e exportar os resultados em um arquivo com formato *Strutured Query Language* (SQL), permitindo utilizar o resultado encontrado pelo processo como entrada de dados para outra tarefa de DM.

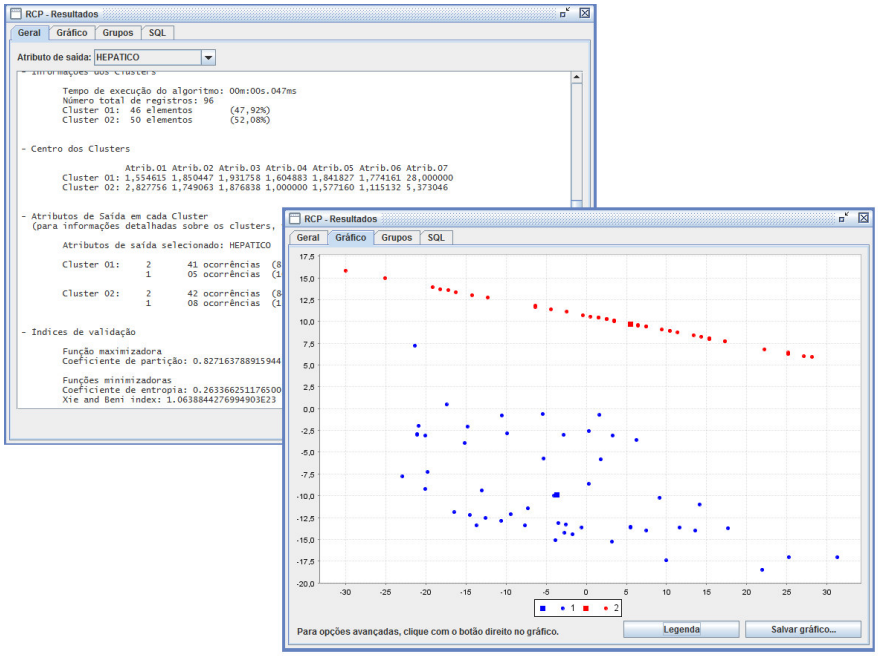

Figura 1. Gráfico gerado pelo RCP na Shell Orion

Foram implementados índices de validação para avaliar o conhecimento gerado pelo algoritmo, sendo estes os índices de partição e entropia, criados por Bezdek, e o índice Xie-Beni desenvolvido por Xie e Beni, estes são descritos, respectivamente, a seguir [KIM et al, 2004]:

$$V_{PC} = \frac{\sum_{j=1}^{N}\sum_{i=1}^{C}\mu_{ij}^{2}}{N}$$

$$V_{PE} = -\frac{1}{N}\sum_{j=1}^{N}\sum_{i=1}^{C}\left[\mu_{ij}\log\left(\mu_{ij}\right)\right]$$

$$V_{XB} = \frac{\sum_{j=1}^{N}\sum_{i=1}^{C}\mu_{ij}^{m}\left\|x_j - c_i\right\|}{N \mid \left(\min_{i,k}\left\|c_i - c_k\right\|\right)}$$

5. Considerações Finais

Utilizando conceitos de *data mining* pode-se, além de descobrir conhecimentos em uma base de dados, comprová-los, proporcionando benefícios significativos as organizações, na tomada de decisão, gerando vantagens estratégicas.

Entre as tarefas e métodos existentes esta pesquisa fundamentou-se na tarefa de clusterização, utilizando o método de lógica *fuzzy*, pelos algoritmos RCP e URCP. Estes algoritmos possibilitam aos elementos pertencerem a diversos grupos, considerando suas semelhanças, gerando resultados melhores, se comparado a métodos tradicionais.

Posteriormente ao entendimento dos algoritmos, e dos métodos de validação, estes foram implementados e testados, validando-os.

Referências

BEZDEK, James et al. Fuzzy models and algorithms for pattern recognition and image processing. New York: Springer, 2005.

FRIGUI, Hichem; KRISHNAPURAM, Raghu. A Robust Algorithm for Automatic Extraction of an Unknown Number of Clusters from Noisy Data. Pattern Recognition Letters 17. p. 1223-1232, 1996.

GOLDSCHMIDT, Ronaldo; PASSOS, Emmanuel. Data Mining: um guia prático. Rio de janeiro: Elsevier, 2005.

HAN, Jiawei; KAMBER, Micheline. Data Mining: concepts and techniques. San Diego: Academic Press, 2001.

HÖPPNER, Frank; KLAWONN, Frank; KRUSE, Rudolf; RUNKLER, Thomas. Fuzzy cluster analysis: methods for classification, data analysis, and image recognition. Chichester: John Wiley & Sons, 1999.

KIM, Y. et al. A cluster validation index for GK cluster analysis based on relative degree of sharing. Information Sciences, Vol. 168, p. 255-242, 2004.

ANEXO A – FERRAMENTAS DE DCBD

Ferramentas	Tarefas	Fabricante	Tipo
SPSS/Clementine	Classificação, Regressão, Associação, Clusterização, Seqüência, Detecção de Desvios	SPSS Inc. www.spss.com	Comercial
PolyAnalist	Classificação, Regressão, Regras de Associação, Clusterização, Sumarização, Detecção de Desvios	Megaputer Intelligence www.megaputer.com	Comercial
Weka	Classificação, Regressão, Regras de Associação	Universit of Waikato www.cs.waikato.ac.nz	Gratuita
Darwin	Classificação	Thinking Machines en.wikipedia.org/ Wiki/thinking_machines	Comercial
Intelligent Miner	Classificação, Regras de Associação, Clusterização, Seqüência, Sumarização	IBM www.ibm.com	Comercial
WizRule	Sumarização, Classificação, Detecção de Desvios	WizSoft Inc. www.wizsoft.com	Comercial
Bramining	Classificação, Regras de Associação, Regressão, Sumarização	Grall Corp. www.graal-orp.com.br	Comercial
SAS Enterprise Miner	Classificação, Regras de Associação, Regressão, Sumarização	SAS Inc. www.sas.com	Comercial
Oracle *Data Mining*	Classificação, Regressão, Associação, Clusterização, Mineração de Textos	Oracle www.oracle.com	Comercial
Orion *Data Mining*	Classificação, Associação, Clusterização	Grupo de Pesquisa em Inteligência Computacional Aplicada – UNESC www.unesc.net/ica	Gratuita

Fonte: Adaptado de GOLDSCHMIDT, R.; PASSOS, E. (2005)