UNIVERSIDADE DO EXTREMO SUL CATARINENSE – UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

DANIEL PEREGO

O MÉTODO DE LÓGICA *FUZZY* PELO ALGORITMO GATH-GEVA NA TAREFA DE CLUSTERIZAÇÃO DA *SHELL* ORION *DATA MINING ENGINE*

CRICIÚMA, JULHO DE 2009

DANIEL PEREGO

O MÉTODO DE LÓGICA *FUZZY* PELO ALGORITMO *GATH & GEVA* NA TAREFA DE CLUSTERIZAÇÃO DA *SHELL* ORION *DATA MINING ENGINE*

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

Orientadora: Profª. MSc. Merisandra Côrtes de Mattos

CRICIÚMA, JULHO DE 2009

DANIEL PEREGO

O método de Lógica *Fuzzy* pelo Algoritmo Gath-Geva na Tarefa de Clusterização da *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:

Profa. MSc. Merisandra Côrtes de Mattos (UNESC)
Orientadora

Prof. Edison Uggioni

Profa. MSc. Priscyla Waleska Simões

Aos meus pais, Luis Perego, Alvina Cardoso Perego, meu irmão André e meus amigos pela força.

AGRADECIMENTOS

Agradeço a Deus pela sua presença constante em minha vida, pelo auxílio nas minhas decisões e por me ajudar e confortar nas horas difíceis.

Agradeço também:

A minha família, por todo amor, carinho, compreensão e apoio concedidos, por sempre proporcionar uma ótima estrutura e incentivar a seguir em frente na minha formação acadêmica.

A meus grandes amigos Rafael, Leandro e Robson pelo apoio, amizade e pelos momentos alegres que compartilhamos ao longo da nossas vidas.

Ao meu colega Cleyton Stang e ao Bacharel José Márcio pela ajuda durante a implementação efetuada neste trabalho de pesquisa.

Agradeço profundamente a grande incentivadora desta pesquisa que não mediu esforços para auxiliar na conclusão da mesma, minha professora e orientadora Merisandra, contribuindo também, além do conhecimento, com sua amizade.

Enfim, a todos que contribuíram para a execução desse trabalho, seja pela ajuda constante ou por uma simples palavra de amizade.

"*A mais profunda raiz do fracasso em nossas vidas é pensar, 'Como sou inútil e fraco'. É essencial pensar poderosa e firmemente, 'Eu consigo', sem ostentação ou preocupação."*

(Dalai Lama)

RESUMO

O avanço dos modelos computacionais proporcionou uma evolução na capacidade de processamento e armazenamento de informações, implicando na formação de grandes bases de dados. Com isso, tem-se a necessidade de desenvolver ferramentas, métodos e técnicas para analisar essas informações gerando novos conhecimentos. Dentre as tecnologias disponíveis destaca-se o *data mining* que é responsável por adquirir conhecimento de uma base, sendo uma das principais etapas da descoberta de conhecimento. Considerando que a maioria das ferramentas são comerciais, o Grupo de Pesquisa em Inteligencia Computacional Aplicada do Curso de Ciência da Computação da Unesc, possui o projeto em desenvolvimento da *Shell* Orion *Data Mining Engine*. Este projeto compreende a criação de uma ferramenta de *data mining* que seja contemplada com vários métodos da *data mining* e suporte a conexão com diferentes Sistemas de Gerenciamento de Banco de Dados (SGBD). Esta pesquisa incrementa ainda mais esse projeto tendo como base a fundamentação e modelagem matemática para implementação do algoritmo Gath-Geva. Este algoritmo faz parte da tarefa de clusterização, que utiliza o método de lógica *fuzzy*, fazendo que os elementos sejam separados em grupos distintos, porém com pertinência a mais de um grupo. Ao final da pesquisa, foram realizados testes comprovando o funcionamento satisfatório do algoritmo.

Palavras-chave: Inteligencia Artificial, *Data Mining*, Tarefa de Clusterização, Lógica *Fuzzy*, Algoritmo Gath-Geva.

ABSTRACT

The advancement of computational models has provided an evolution in the capacity of processing and storage of information, resulting in the formation of large databases. Therefore, there is the need to develop tools, methods and techniques to analyze the databases generating new knowledge. Among the technologies available, there is the data mining, which is responsible for acquiring knowledge from a base, being one of the main steps for the discovery of knowledge. Considering that most of the tools are commercial, the Applied Computational Intelligence Research Group of the Computer Science Course from UNESC, has the project in development of the Shell Orion Data Mining Engine. This project comprises the creation of a data mining tool that includes several methods of data mining and connection support with different Database Management Systems (DBMS). This research enhances even more this Project having as basis, the grounding and mathematical modeling for the implementation of Gath-Geva algorithm. This algorithm is part of the task of clusterization, which uses the method of fuzzy logic, so that the elements are separated into distinct groups, however with relevance to more than one group. At the end of the research, tests, confirming the satisfactory operation of the algorithm, were run.

Keywords: Artificial Intelligence, Data Mining, Clusterization Task, Fuzzy Logic, Gath-Geva Algorithm.

ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

CART	*Classification and Regression Trees*
DBSCAN	*Density-Based Spatial Clustering of Applications with Noise*
DM	*Data Mining*
FCM	*Fuzzy C-Means*
GK	Gustafson-Kessel
GG	Gath-Geva
IA	Inteligência Artificial
ID3	*Iterative Dichotomiser 3*
KDD	*Knowledge Discovery in Databases*
SQL	*Structured Query Language*
TCC	Trabalho de Conclusão de Curso
UNESC	Universidade do Extremo Sul Catarinense

SUMÁRIO

1 INTRODUÇÃO

O desenvolvimento da tecnologia da informação proporcionou as empresas e instituições uma maior capacidade de armazenamento dos seus dados, por meio disto especialistas têm condições de obter informações úteis para auxiliar nas tomadas de decisões o que pode refletir em melhorias nas práticas do negócio. No entanto, esta capacidade de armazenamento gerou um grande volume de dados, que em pouco tempo supera em muito a habilidade do ser humano em analisar e adquirir conhecimento (REZENDE, 2005).

A análise na maioria dos casos vem sendo realizado por meio de ferramentas estatísticas, planilhas eletrônicas, consultas *Structured Query Language* (SQL), entre outros, porém estes recursos são limitados, pois não torna fácil a descoberta de padrões ou relações existentes entre os dados, dificultando com isso a geração de conhecimento (AMORIM, 2006).

Surge então em 1996 o conceito de *data mining (DM)* (FAYYAD et al, 1996) que consiste de acordo com Goldschmidt e Passos (2005) em uma das etapas da Descoberta de Conhecimento em Bases de Dados (DCBD) do inglês *Knowledge Discovery in Databases* (KDD) que engloba técnicas de estatística, inteligência artificial, banco de dados e aprendizado de máquina. O DCBD tem por finalidade aplicar tarefas e métodos sobre a base de dados a fim de obter padrões que gerem fontes de conhecimento para o usuário final.

Dentre as tarefas de DM, pode-se citar associação, classificação, regressão, sumarização e *clusterização*. Sendo que para aplicação dessas tarefas há necessidade de se aplicar diversos métodos, como por exemplo: lógica *fuzzy* para *clusterização*, árvores de indução para classificação, entre outros. Normalmente essas tarefas e métodos encontram-se disponíveis em diferentes ferramentas (*shells)* que na sua grande maioria são comerciais.

Considerando isso, o Grupo de Pesquisa em Inteligência Computacional Aplicada da Universidade do Extremo Sul Catarinense, vem desenvolvendo uma *Shell* denominada Orion de licença gratuita, que objetiva implementar os conceitos de d*ata mining*. O desenvolvimento da ferramenta foi dividido em vários módulos, onde já estão implementados, por exemplo: classificação (algoritmo ID3, CART), *clusterização (*algoritmo *K-means, Kohonen, Gustafsson-Kessel*) e associação (algoritmo *Apriori*).

No caso da tarefa de *clusterização*, foco desta pesquisa, tem-se a divisão dos dados que possuem características similares em subgrupos (*clusters*), sendo ideal que esses conjuntos sejam bem distintos entre si. A *clusterização* pode ser realizada utilizando vários métodos, como os estatísticos, que se baseiam em probabilidade (ex: algoritmo *K-means*); as redes neurais que utilizam abordagens conexionistas na sua execução (exemplo: algoritmo Kohonen); a lógica *fuzzy,* que agrega os dados pelo grau de pertinência a cada subgrupo (exemplo: algoritmo Gustafson-Kessel), dentre outros (ALVES, 2007).

A maioria dos algoritmos de *clusterização* realiza o agrupamento baseando-se na lógica clássica (*crisp)*, ou seja, os elementos pertencem ou não ao subgrupo. Porém, em alguns casos os dados podem conter características que façam o mesmo ter um grau de similaridade a diversos *clusters*. Considerando essas situações, denominadas "fuzzy" por Goldschmidt e Passos (2005), é interessante utilizar o método de lógica *fuzzy*, onde determinando o grau de pertinência dos elementos em relação aos subgrupos, consegue-se diminuir o número de *clusters* formados e erros causados por problemas de ruídos na entrada de dados.

Além disso, outro problema apresentado pela maioria dos algoritmos de *clusterização* refere-se à determinação do número de grupos, pela dificuldade de encontrar o número ideal. Normalmente esse valor deve ser informado pelo usuário como parâmetro no algoritmo, sendo que somente após a execução é possível realizar a análise para validação dos

resultados. Considerando essa problemática, os pesquisadores Gath e Geva em 1989, determinaram por meio de fórmulas matemáticas a obtenção de qual seria o número de *clusters* gerados por dados de uma base, facilitando a inicialização dos algoritmos de *clusterização* de bases de dados mais complexas.

Nesse contexto, essa pesquisa consistiu no desenvolvimento da modelagem matemática e implementação do algoritmo Gath-Geva para o módulo de *clusterização* da *Shell* Orion *Data Mining Engine.* Aumentando a diversidade de métodos implementados para *clusterização*, gerando com isso uma documentação detalhada da modelagem matemática do algoritmo.

1.1 OBJETIVO GERAL

Desenvolver o algoritmo Gath-Geva no módulo de *clusterização* da *Shell* Orion *Data Mining Engine*.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos desta pesquisa consistem em:

a) compreender o conceito de *data mining* e a tarefa de clusterização;
b) entender o método de lógica *fuzzy* e o funcionamento do algoritmo *Gath-Geva*;
c) aplicar o algoritmo *Gath-Geva* para *clusterização* dos dados;
d) demonstrar o funcionamento do algoritmo *Gath-Geva*;
e) utilizar uma base de dados a fim de testar o funcionamento do algoritmo Gath-Geva.

1.3 JUSTIFICATIVA

No cotidiano das instituições e corporações acumula-se uma grande quantidade de informações sobre as mais diversas áreas existentes, como por exemplo, consultas médicas, acervo de bibliotecas, supermercados, entre outros. Sendo que nessas informações está contido o conhecimento que pode proporcionar melhores condições para desempenhar de forma mais eficaz as atividades em geral, mas obter isso sem utilização de ferramentas adequadas pode dificultar os objetivos de serem atingidos.

Nesse contexto, *data mining,* se torna uma ferramenta apropriada para buscar padrões existentes e gerar conhecimento necessário para auxiliar na tomada de decisões.

Hoje no mercado existem muitas ferramentas comerciais que aplicam *data mining*, no intuito de dar suporte as instituições na aquisição de conhecimento, mas normalmente são pouco flexíveis, ou seja se aplicam a uma área específica, não englobam todas as técnicas, são pouco escalonável em relação aos métodos, em alguns casos é necessário a aquisição de diversos pacotes, sendo que as ferramentas gratuitas dificilmente suportam diferentes tipos de banco de dados (GOLDSCHMIDT; PASSOS, 2005).

Perante a carência de ferramentas gratuitas e que suportem diferentes tipos de bancos de dados disponíveis, o Grupo de Pesquisa em Inteligência Computacional Aplicada do curso de Ciência da Computação da UNESC, tem um projeto que visa desenvolver uma ferramenta de licença gratuita para aplicação de métodos e tarefas de *data mining*. O desenvolvimento da *Shell* Orion justifica-se pela ausência de bibliografia sobre os algoritmos de DM , que expliquem de uma forma didática o funcionamento destes para a comunidade acadêmica. Além disso, ferramentas deste tipo devem suportar conexões com diferentes tipos

de banco de dados, agrupar numa mesma *shell* diversos métodos e tarefas, sendo que em outras ferramentas isso tende a ser separado em pacotes.

A *clusterização* deve ser utilizada quando o conjunto de dados analisados é bastante complexo, possibilitando agrupá-los em diversos *clusters*, de forma que os elementos no mesmo grupo possuam características similares, enquanto entre os grupos deve-se ter a maior distinção possível (GOLDSCHMIDT;PASSOS,2005).

A utilização do algoritmo *Gath-Geva* se justifica devido ao agrupamento que é realizado pela lógica *fuzzy* que é mais flexível na formação dos *clusters*, pois os dados são agrupados pelo seu grau de pertinência em relação ao grupo. Assim, os dados possuem similiridade com o *cluster* mesmo que parcial, podendo inclusive pertencer ao mesmo grupo.

Outra vantagem propiciada pelo algoritmo é a determinação *a priori* do número de *clusters* da base de dados de entrada. Isso é interessante devido a dificuldade existente em determinar a quantidade de *clusters,* principalmente, em repositórios com grande volume de dados, sendo que na maioria dos algoritmos esse número é determinado pelo usuário (YONAMINE et al, 2002).

1.4 ESTRUTURA DO TRABALHO

Essa pesquisa possui em sua totalidade oito capítulos. Sendo que a partir do primeiro, tem-se uma explanação sobre o tema exposto, objetivos e justificativa do trabalho realizado.

Os conceitos fundamentais sobre a descoberta de conhecimento em bases de dados e suas principais etapas como o *data mining* estão presentes no Capítulo 2.

No Capítulo 3 é apresentada a *Shell* Orion *Data Mining Engine*, mostrando suas principais funcionalidades, tarefas, métodos e algoritmos já implementados.

As características da tarefa de clusterização no processo *data mining* e o método de lógica *fuzzy* são descritos respectivamente nos Capítulo 4 e 5, sendo que no Capitulo 5 também aborda alguns exemplos de algoritmos que utilizam a lógica *fuzzy* para a clusterização.

No Capítulo 6 é apresentado o algoritmo Gath-Geva, onde é baseado no artigo publicado pelos autores do algoritmo de 1989, mostrando as técnicas e formalismos matemáticos utilizados para efetuar a implementação do algoritmo. Sendo que no Capítulo 7, é detalhada a utilização desse algoritmo em diversos trabalhos acadêmicos.

O trabalho desenvolvido, a demonstração matemática do algoritmo Gath-Geva e os resultados obtidos por meio da sua aplicação em uma base de dados são apresentados no Capítulo 8.

Finalizando tem-se a conclusão, composta também de sugestões para futuras pesquisas.

2 DESCOBERTA DE CONHECIMENTO EM BASES DE DADOS

A partir da década de 1960, as tecnologias para armazenamento de dados tiveram um avanço tecnológico por meio dos sistemas de informações, advento da utilização de banco de dados, *data warehouses*, entre outros. Isso resultou em termos mundiais num aumento do volume de dados, surgindo então a eminente necessidade aprimoramento desses dados para melhor utilização nas diversas áreas, como industriais, negócios, instituições. (HAN; KAMBER, 2001, tradução nossa).

A quantidade de informação armazenada em grandes corporações e instituições é bastante considerável em termos de volume, superando em muitos *gigabytes* de dados. Com isso, surge a necessidade de se utilizar meios computacionais para manipulação desses dados, com objetivos de agregar valor por meio da interpretação dessas bases, sendo que essas tarefas podem ser, em alguns casos, ineficientes somente pela avaliação "visual" de especialistas. (GOLDSCHMIDT; PASSOS, 2005).

A área denominada Descoberta de Conhecimento em Bases de Dados (*Knowledge Discovery in DataBases* – KDD), tem apresentado uma característica de envolver diversos campos da ciência, como por exemplo: Estatística, Inteligência Computacional, Reconhecimento de Padrões e Banco de Dados. Ilustrando-se essa afirmativa, mostra-se como são divididas as atividades que envolvem o DCBD (GOLDSCHMIDT; PASSOS, 2005):

a) **desenvolvimento tecnológico:** abrangem todas as ferramentas, tecnologias, algoritmos que dão suporte na busca de novos conhecimentos nos repositórios de dados;
b) **execução do DCBD:** consiste efetivamente nas atividades empregadas na busca de conhecimento em bases de dados;

c) **aplicação de resultados:** por meio do conhecimento gerado tem que se avaliarem as possibilidades existentes para aplicar de forma efetiva, ou seja, verificar de que forma os resultados obtidos podem ajudar nas tomadas de decisões.

A Descoberta de Conhecimento em Bases de Dados implica no processo de adquirir conhecimentos úteis nos dados, sendo que é composta por diferentes etapas, uma das mais importantes é o *data mining*, onde seu conceito, muitas vezes, é confundido como sendo o próprio DCBD (FAYYAD; PIATESTSKY-SHAPIRO; SMYTH, 1996, tradução nossa).

2.1 ETAPAS DA DESCOBERTA DE CONHECIMENTO EM BASE DE DADOS

Considerando que para conseguir resultados satisfatórios, não se deve apenas aplicar as técnicas nos dados, sendo necessário antes prepará-los. O usuário das ferramentas DCBD precisa interagir com o sistema, de forma a ter um conhecimento do domínio do problema em que está sendo aplicado. Por isso DCBD são ferramentas interativas, que automatizam parte do processo, porém necessita de alguém para gerenciar. O DCBD é subdividido em diversas etapas (Figura 1) (MANILLA, 1999, tradução nossa):

a) **especialista do domínio:** é composto por uma pessoa ou um conjunto destas que são especialistas no assunto que está sendo analisado, buscando-se obter novos conhecimentos a fim de aplicar na sua área de atuação. A participação do especialista no domínio de aplicação DCBD é de fundamental importância para o sucesso do processo de DCBD (GOLDSCHMIDT; PASSOS, 2005);

b) **preparação dos dados:** os dados são normalmente obtidos em base de dados, *data warehouse,* entre outros. Com isso, esses dados precisam passar por duas etapas (KANTARDZIC, 2003, tradução nossa):

- *limpeza dos dados:* dados incoerentes com a aplicação são facilmente encontrados ao longo de todo o volume de dados armazenado. Assim, para que eles não prejudiquem o resultado do processo DCBD, deve-se eliminar os dados anômalos da base, ou desenvolver modelos robustos que sejam imunes a esses tipos de dados,

- *escalabilidade, codificação dos dados* isso é necessário para padronizar a forma de representação das variáveis utilizadas no processo de DCBD, independentes se forem numéricas ou nominais, devido à grande influência no resultado final, por exemplo: a variável cor se for representada no intervalo [0,1], e no mesmo ser representada variando de [-100, 1000] tornará o resultado errôneo;

c) ***data mining*:** é a principal etapa do processo de descoberta de conhecimento, onde se aplica algoritmos específicos para extrair padrões dos dados (FAYYAD; PIATESTSKY-SHAPIRO; SMYTH, 1996, tradução nossa);

d) **pós-processamento:** engloba a interpretação, análise e visualização do conhecimento gerado pelo *data mining*. Normalmente, o especialista de DCBD juntamente com o especialista do domínio interpreta os resultados obtidos e determinam novas estratégias para investigação dos dados;

e) **apresentação:** essa etapa complementa o pós-processamento, consistindo no apoio para buscar a melhor interação entre o usuário final e os resultados obtidos (MANILLA, 1999, tradução nossa).

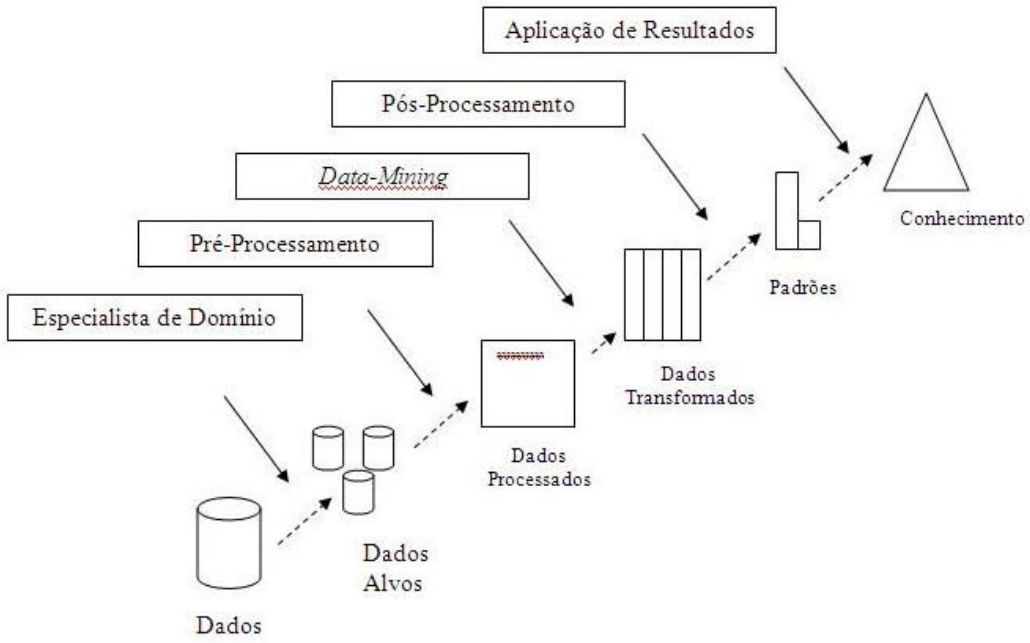

Figura 1. Etapas para descoberta de conhecimento
FONTE: Adaptado de FAYYAD, U.; PIATETSKY-SHAPIRO, g.; SMYTH. (1996)

A etapa de preparação de dados não deve ser considerada isoladamente do processo de DCBD, pois a cada interação, todas as atividades são realizadas novamente, tornando viável obter melhores conjuntos de dados para posterior utilização pela etapa de *data mining* (KANTARDZIC, 2003).

A descoberta de conhecimento pode ser utilizada no cotidiano das empresas, instituições, entre outros. Posteriormente, são detalhados alguns exemplos de aplicações que obtiveram resultados satisfatórios segundo Carvalho (2005).

2.2 APLICAÇÕES DA DESCOBERTA DE CONHECIMENTO EM BASE DE DADOS

As aplicações de DCBD são bem diversificadas, nas variadas áreas de conhecimento, como por exemplo (CARVALHO, 2005):

a) **instituição governamental:** as ferramentas de *data mining* são utilizadas há muito tempo pelo governo dos EUA, sendo que seu foco é identificar padrões

nas transferências bancárias internacionais que indicam desvios de conduta, como “fraudes”. Também foi possível impedir um atentado a cidade de Okalahoma, pois com a análise dos registros foi constatado objetivo de fabricar uma bomba;

b) **supermercados:** cadastrando e coletando informações pessoais dos clientes, para com isso fazer associações com as compras realizadas e nível de consumo mês-a-mês, buscando por meio dessas informações distribuir melhor os produtos nas prateleiras para consumar comprar conjuntas, oferecer brindes e descontos personalizados;

c) **medicina:** a manutenção das bases de dados com informação sobre sintomas, resultados de exames, diagnósticos, tratamentos e doenças de cada paciente. Proporciona por meio de ferramentas de DCBD, o fornecimento de conhecimentos como a relação entre as doenças e certos perfis profissionais, sócio-culturais, hábitos pessoais, local de moradia;

d) **astronomia e geologia:** com os dados gerados tem-se a necessidade de utilizar ferramentas de DCBD para descobrir novos conhecimentos que os analistas a olho nú não conseguiriam facilmente perceber, como por exemplo padrões presentes nas estrelas ou nas formações geológicas que auxiliam no entendimento do passado.

e) **evitar perda de clientes:** com a concorrência existente no mercado é comum que as pessoas alternem de empresas com certa facilidade. Utilizando as ferramentas DCBD para entender os motivos pelos quais os clientes deixam de comprar, podendo então realizar ofertas, promoções para postergar o cliente na empresa.

Essas são exemplos básicos do processo de descoberta de conhecimento em bases de dados, como foco nesta pesquisa é a etapa de *data mining*, que é composta de tarefas, sendo que essas necessitam de métodos aplicados pelos diversos algoritmos existentes.

2.3 DATA MINING

Encontrar um conceito ou uma definição científica é sempre uma tarefa controvérsia entre os pesquisadores. Contudo, pode-se dizer que *data mining* é a análise dos conjuntos de dados puramente observacionais na sua imensidão que são apresentados em uma forma mais compreensível e útil para o usuário final. Os resultados obtidos com a utilização do *data mining* são muitas vezes referidos como modelos ou padrões. Isso pode incluir gráficos, regras, *clusters*, árvore estrutural e séries temporais (HAND; MANILLA; SMYTH, 2001, tradução nossa).

A busca incansável para aprimorar a descoberta de conhecimento nas bases de dados fez com que aproximadamente na década de 90, surgisse o conceito de *data mining*, que visa buscar padrões existentes nas informações que aparentemente não têm significado, mas que bem organizadas são fundamentais nas tomadas de decisões (SINGH, 2001).

O *data mining* refere-se à aquisição de padrões de nos repositórios de dados, muitas vezes é utilizada essa nomenclatura para todo o processo de aquisição de conhecimento, mas no entanto o *DM* é somente uma das etapas. (HAN; KAMBER, 2001, tradução nossa).

O *data mining* é composto por tarefas, sendo que a escolha desta deve levar em consideração os objetivos do domínio da aplicação, que pode ser predição ou descrição (KANTARDZIC, 2003, tradução nossa).

Predição envolve a utilização de dados com intuito de realizar uma previsão de valores de dados futuros (KANTARDZIC, 2003, tradução nossa).

Descrição engloba as atividades de obtenção de padrões dentro de bases de dados que sejam interpretáveis por humanos (REZENDE, 2000).

Como visto anteriormente há inúmeras ferramentas de *DCBD* para os mais diferentes domínios, com isso a variedade de tarefas existentes para aplicar DM tende a ser cada vez mais diversificada, pode se exemplificar alguns tipos como:

a) **classificação:** compreende o processo de descobrir uma função que organize os dados em categorias distintas, que são chamadas de classes. Determinando essa função, pode-se aplicar aos novos registros a fim de prever a que classe se incorpora. Um exemplo para aplicação dessa tarefa seria: uma financeira que tem em sua base de dados, o histórico de clientes e o pagamento de empréstimos contraídos em curto prazo. Com isso, poderia se criar duas classes: a de clientes bom pagadores e os inadimplentes. Descobrindo as regras de classificação. O crédito poderia ser somente concedido para os clientes que pertencessem à classe de bons pagadores (GOLDSCHMIDT; PASSOS, 2005);

b) **estimação:** a estimativa faz uma avaliação contínua dos resultados levando em conta a probabilidade, onde com alguns dados de entrada podem-se estimar algumas saídas. Por exemplo: pode ser utilizado para estimar o número de crianças em uma família, estimar o valor da vida útil de um cliente (BERRY; LINOFF, 2004, tradução nossa);

c) **regressão:** é utilizada somente para valores numéricos, criando-se uma relação entre as variáveis de entrada e saída. Controlando-se as variáveis de entrada do sistema, consegue-se realizar a previsão dos valores de saída do

sistema (KANTARDZIC, 2003, tradução nossa). É utilizada, por exemplo, para determinar a probabilidade de um paciente sobreviver (GOLDSCHMIDT; PASSOS, 2005);

d) **clusterização:** os dados são agrupados, com o princípio de aumentar a similaridade dentro de um mesmo grupo e minimizar a semelhança entre grupos diferentes. Exemplo: identificação de clientes para comercialização de um determinado tipo de produto (HAN; KAMBER, 2001, tradução nossa);

e) **associação:** é uma abordagem simples para gerar regras a partir de dados, sendo utilizada para relacionar ou associar. Pode ser útil em análises mercadológicas, para associar produtos que tenham relação entre si num supermercado, ou seja, se dois itens são correlacionados nas vendas, Assim buscam-se associações que induzam o cliente a comprar dois produtos em vez de somente um. Os resultados podem ser expressos pela seguinte analogia, "clientes que compram pão também compram leite" (BERRY; LINOFF, 2004, tradução nossa).

Após a definição da tarefa de DM, deve-se escolher o método a ser aplicado, dentre os quais, tem-se:

a) **redes neurais:** modelos matemáticos inspirados nos princípios de funcionamento dos neurônios do cérebro humano. Apresentam a capacidade de adquirir, armazenar e utilizar conhecimento experimental e buscam simular, computacionalmente, as habilidades humanas de aprendizado, associação, generalização e abstração (GOLDSCHMIDT; PASSOS, 2005);

b) **algoritmos genéticos:** baseado no processo de evolução dos seres vivos, e reprodução genética. Produzem soluções satisfatórias mesmo aplicando em problemas complexos. Sendo que conforme utiliza a informação acumulada se

minimiza o tempo e melhora as repostas (AZEVEDO; BRASIL; OLIVEIRA, 2000);

c) **lógica *fuzzy*:** diferentemente da lógica clássica, onde os elementos pertencem ou não a um determinado conjunto, na lógica *fuzzy* existe uma *função de pertinência*, onde cada elemento possui um grau de pertinência a cada conjunto existente, podendo pertencer ou não a um conjunto, bem como possuir valores intermediários (KLIR; YUAN, 1995, tradução nossa).

f) **árvores de decisões:** estrutura que pode dividir um conjunto de dados continuamente até minimizar em pequenos grupos de dados, por meio da utilização de regras simples de decisão. A cada divisão, os elementos que compõem os conjuntos resultantes se tornam cada vez mais semelhantes entre si (BERRY; LINOFF, 2004, tradução nossa).

As tarefas e os métodos de *data mining* são implementados em diversas ferramentas, também conhecidas como *shells*, normalmente numa mesma *shell* pode-se encontrar diversas tarefas e métodos, o que facilita bastante o processo de DCBD.

Na Tabela 1 têm-se alguns exemplos destas ferramentas, sendo a maioria delas comerciais, o que pode dificultar a aquisição por pequenas organizações e instituições. Há poucas ferramentas gratuitas, sendo que dentre estas se tem a *Shell* Orion *Data Mining Engine.* O desenvolvimento desta ferramenta iniciou-se em 2005 por acadêmicos e professores do curso do Grupo de Pesquisa em Inteligência Computacional Aplicada do Curso de Ciência da Computação da Unesc.

Tabela 1. Exemplos de Ferramentas DCBD

Nome	Tipo	Tarefas Disponiveis	Fabricante
SPSS/Clementine	Comercial	Classificação, Regressão, Associação, Clusterização, Seqüência, Detecção de Desvios	SPSS Inc. www.spss.com
PolyAnalyst	Comercial	Classificação, Regressão, Regras de Associação, Clusterização, Sumarização, Detecção de Desvios	Megaputer Intelligence www.megaputer.com
Weka	Gratuita	Classificação, Regressão, Regras de Associação	Universit of Waikato www.cs.waikato.ac.nz
Darwin	Comercial	Classificação	Thinking Machines Http://en.wikipedia.org/ Wiki/thinking_machines
Intellingent Mine	Comercial	Classificação, Regras de Associação, Clusterização, Seqüência, Sumarização	IBM www.ibm.com
WizRule	Comercial	Sumarização, Classificação, Detecção de Desvios	WizSoft Inc. www.wizsoft.com
Bramming	Gratuita	Classificação, Regras de Associação, Regressão, Sumarização	Grall Corp. www.graal-corp.com.br
SAS Enterprise Miner	Comercial	Classificação, Regras de Associação, Regressão, Sumarização	SAS Inc. www.sas.com
Oracle Data Mining	Comercial	Classificação, Regressão, Associação, Clusterização, Mineração de Textos	Oracle www.oracle.com
Orion Data Mining	Gratuita	Classificação, Associação, Clusterização	Grupo de Pesquisa em Int. Comp. Aplicada – UNESC

FONTE: Adaptado de GOLDSCHMIDT, R.; PASSOS, E. (2005)

Nessa pesquisa será implementado o algoritmo Gath-Geva para a tarefa de clusterização pelo método de lógica *fuzzy*, para melhor compreender o funcionamento da *shell* são apresentados tópicos referente ao seu desenvolvimento e aos módulos que a compõem.

3 SHELL ORION DATA MINING ENGINE

A *Shell* Orion *Data Mining Engine* é uma ferramenta que está sendo desenvolvida pelos acadêmicos e professores do Grupo de Pesquisa em Inteligência Computacional Aplicada do Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense (UNESC).

O projeto teve início no ano de 2005, onde os acadêmicos em seus trabalhos de conclusão de curso agregam funcionalidades, tarefas e métodos na *Shell* Orion. Na Tabela 2 se pode observar módulos já implementados.

Tabela 2. 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	Custerização	Particionamento	*K-Means*	Numéricos	MARTINS, 2007)
2007	Custerização	Redes Neurais	*Kohonen*	Numéricos	BORTOLOTTO, 2007)
2008	Custerização	Lógica Fuzzy	*Gustafson-Kessel*	Numéricos	(CASSETARI JÚNIOR, 2008)

A ferramenta está sendo implementada por meio da linguagem *Java*, por essa possibilitar o funcionamento em diversos sistemas operacionais existentes (multiplataforma), ter a possibilidade de reutilização de código e principalmente ter suas ferramentas de desenvolvimento disponibilizadas gratuitas (PELEGRIN, 2005).

A utilização dos algoritmos desenvolvidos, bem como as funcionalidades da *Shell* Orion, é possibilitada por meio de uma interface principal, objetivando a facilidade e centralização das funções da ferramenta.

3.1 INTERFACE DA *SHELL* ORION

A interface principal da *shell* foi desenvolvida para melhorar a interação com o especialista de DCBD e facilitar a inserção de novas funcionalidades na ferramenta. A Figura 2 apresenta a interface principal da *Shell* Orion:

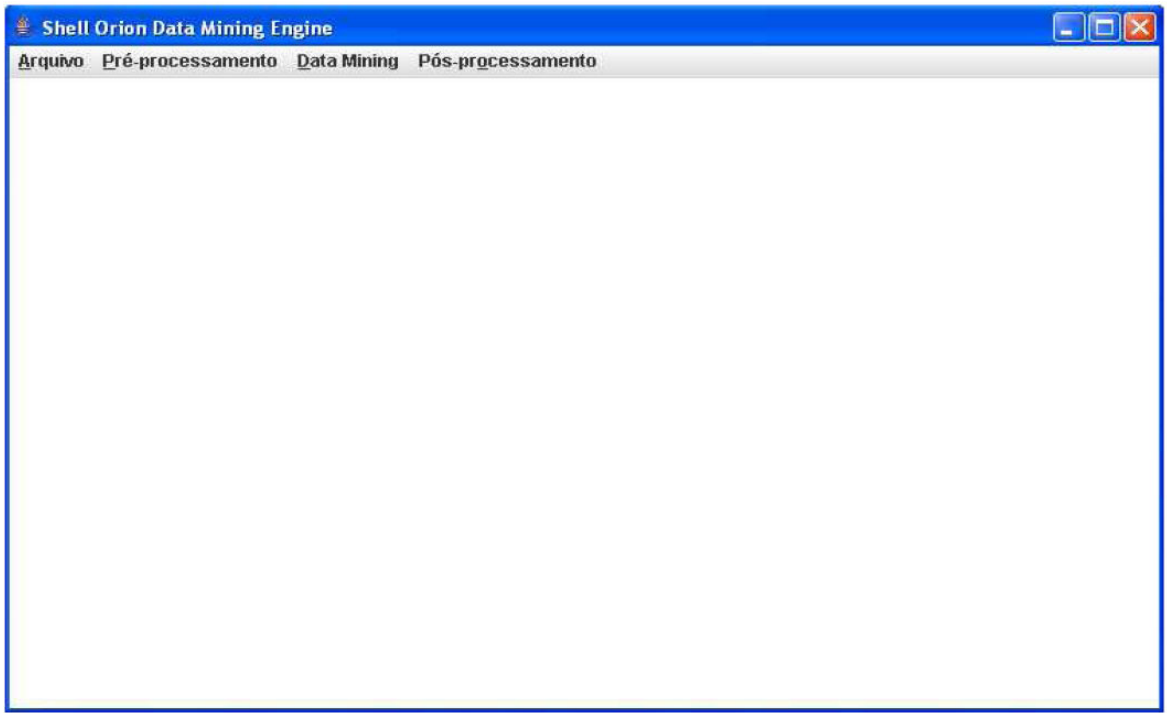

Figura 2. Interface Principal Shell Orion Data Mining Engine
Fonte: BORTOLOTTO, L. (2007)

Em 2007, a *Shell* Orion *Data mining* Engine sofreu uma alteração no que se refere à conexão ao banco dados, sendo que anteriormente eram restritas as opções de banco de dados. Atualmente a ferramenta trabalha com qualquer tipo de banco que possua um *driver* Java *Data Base Connectivity* (JDBC) implementado, suportando portanto diferentes bases de dados. Além disso, foi alterada a disposição das opções na interface (BORTOLOTTO, 2007).

A realização da conexão com o banco de dados na ferramenta é bastante simples, a seguir demonstra-se os passos necessários para conseguir realizar essa tarefa:

a) **instalação do driver:** definido o SGBD que se deseja realizar a conexão, deve-se obter o *driver* JDBC do respectivo banco e instalar-lo corretamente na ferramenta;

b) **configuração da conexão:** para realizar a configuração da conexão o usuário deve selecionar na guia *Arquivo, Conexões*, e clicar em *Cadastro de Conexão*. O usuário deve informar o nome, *driver* e selecionar o banco ou simplesmente selecionar um já existente no menu conexão. Na Figura 3 tem-se um exemplo utilizado pelo acadêmico Bortolotto (2007) realizou-se uma conexão ao banco de dados *freeware* PostGreSQL[1];

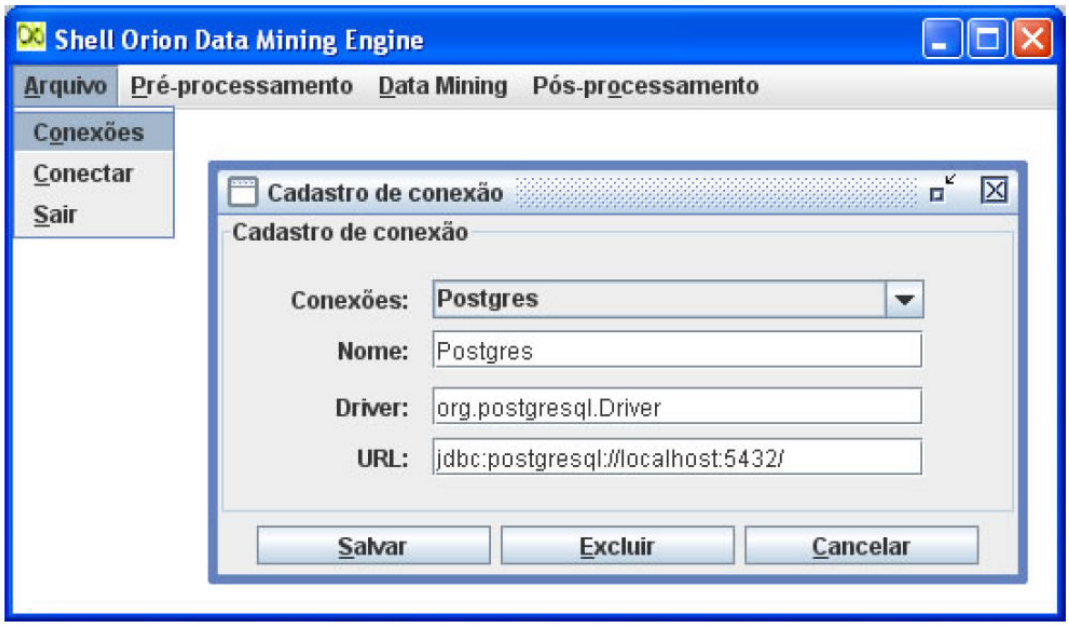

Figura 3. Cadastro de conexão com o banco PostGreSQL
Fonte: BORTOLOTTO, L (2007)

c) **conectar ao banco de dados:** após realizada a configuração do banco de dados, o usuário deve-se se dirigir ao menu *Arquivo, Conectar*. Nessa tela deve ser informada a conexão desejada, que foi anteriormente configurada, nome da base de dados, nome de usuário e a senha. Realizando a conexão será

[1] PostGreSQL disponível em http:/www.postgresql.org

possível escolher a tabela na qual se deseja aplicar as tarefas e métodos da ferramenta (Figura 4).

Figura 4. Conexão com o banco PostGreSQL
Fonte: CASSETARI JÚNIOR, J. (2008)

d) **escolha da tarefa:** realizando a conexão corretamente com o banco de dados, o usuário tem como opção realizar o pré-processamento, *data mining* e pós-processamento. Atualmente, na *Shell* Orion só estão implementados alguns métodos de *data mining*. Conforme a tabela 2 apresentada anteriormente no capítulo 3.

Existem inúmeras tarefas, métodos e algoritmos de *data mining* implementados na *Shell* Orion, os quais são descritos a seguir.

3.2 TAREFAS, MÉTODOS E ALGORITMOS DE DATA MINING NA SHELL ORION

As tarefas atualmente implementadas na *Shell* Orion *Data Mining Engine* são associação, classificação e clusterização. Estão disponíveis em módulos separados da ferramenta, sendo que cada algoritmo desenvolvido tem características próprias e necessita de informações peculiares para executar corretamente o processo de descoberta de conhecimento.

No módulo de associação (Figura 5) foi implementado o algoritmo *Apriori* (CASAGRANDE, 2005), onde por meio das regras extraídas encontram-se relacionamentos entre os itens de dados. Assim, por meio deste algoritmo tem-se um conhecimento prévio das propriedades dos conjuntos freqüentes (HAN; KAMBER, 2001, tradução nossa).

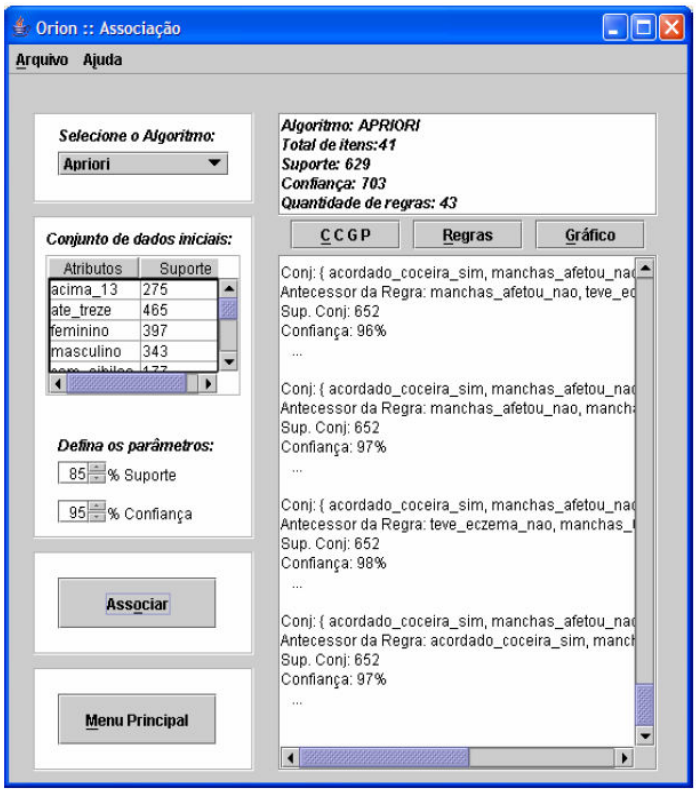

Figura 5. Módulo de Associação da Shell Orion Data Mining Engine
Fonte: CASAGRANDE, D. (2005)

O módulo de classificação possui dois algoritmos implementados, um desses é o ID3, que utiliza árvores de decisão que se caracteriza por trabalhar recursivamente passando por cada nó-filho até que todas as amostras estejam pertencendo à mesma classe. Cada caminho para a folha da árvore de decisão representa uma regra de classificação, sendo que seu funcionamento é caracterizado como *top-down* (cima para baixo) (KANTARDZIC, 2003, tradução nossa).

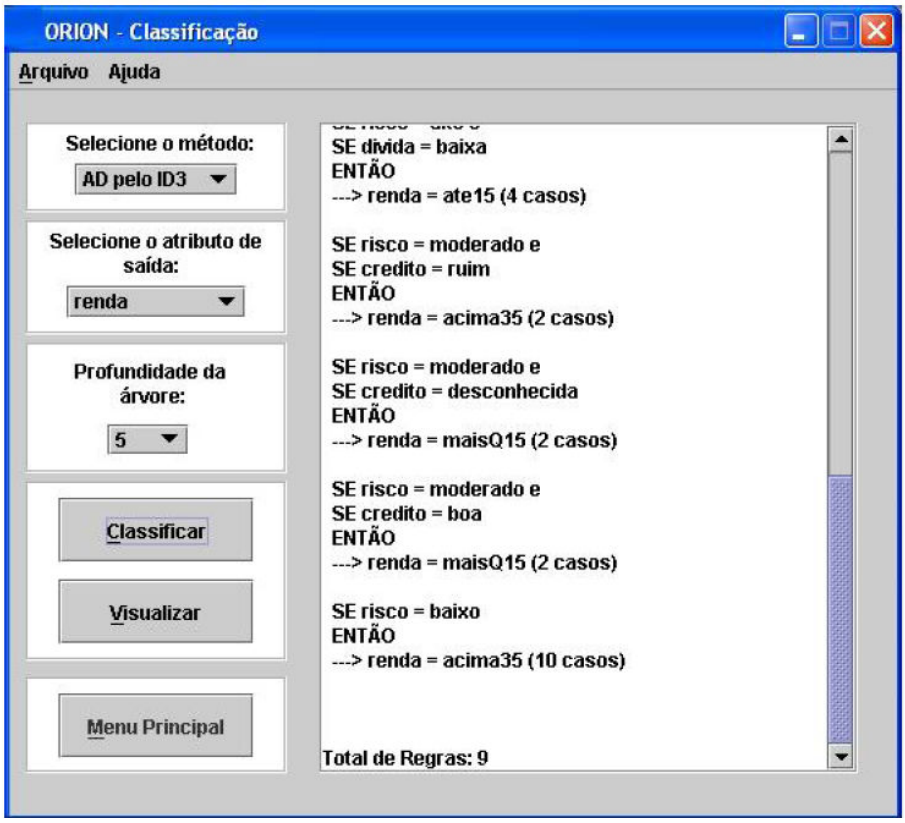

Figura 6. Módulo de Classificação pelo algoritmo ID3
Fonte: PELEGRIN, D. (2005)

O algoritmo ID3 (Figura 6) consiste em produzir regras de classificação considerando-se o objeto de saída que é selecionado pelo especialista do DCBD, além disso, é gerada uma árvore gráfica (Figura 7) de modo a facilitar a visualização em relação ao atributo que mais influenciou na construção do modelo de conhecimento obtido (PELEGRIN, 2005).

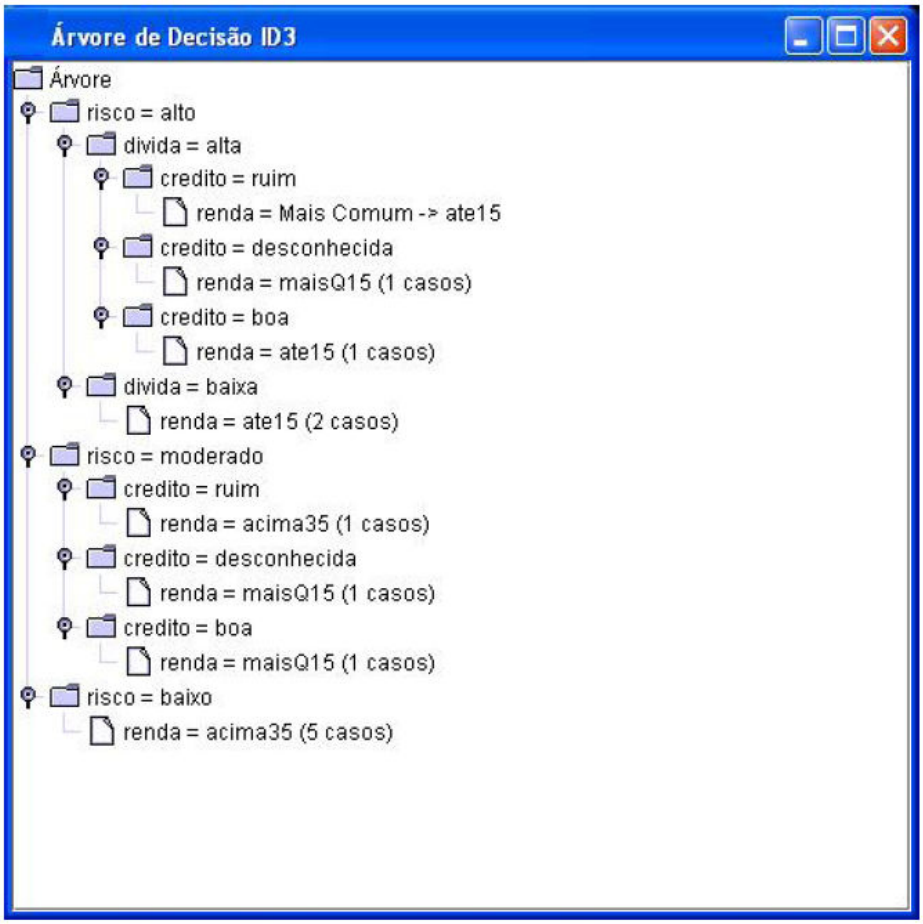

Figura 7. Árvore de decisão gerada pelo algoritmo ID3
Fonte: PELEGRIN, D. (2005)

A árvore de decisão também é o método utilizado pelo algoritmo CART (*Classification and Regression Trees*) no módulo de Classificação da *Shell* Orion. Neste algoritmo as árvores possuem a particularidade de sempre serem binárias, facilitando a leitura e interpretação dos resultados. O CART é bastante aplicado em estudos multidimensionais, tendo a vantagem de trabalhar com variáveis numéricas, nominais e valores ausentes. Além disso, permite construir grupos homogêneos de dados que são caracterizados pelos valores dos atributos (nó-folha da árvore) (CABETE; CARDOSO, 2006).

Numa comparação entre os critérios *Gini*[2] e *Twoing*[3] aplicando-se em diversos conjuntos de dados diferentes, os resultados obtidos se diferenciam, sendo que para a obtenção de melhores resultados e produção de nós descendentes mais puros, utiliza-se o

[2] Critério *Gini*: verifica a maior classe da base de dados e procura isolar das outras classes. (RAIMUNDO, 2007) .
[3] Critério *Twoing*: Compreende a separação da base de dados em duas classes. (RAIMUNDO, 2007).

critério *Gini*, por isso optou-se em implementar o algoritmo por esse critério na *Shell* Orion (RAIMUNDO, 2007).

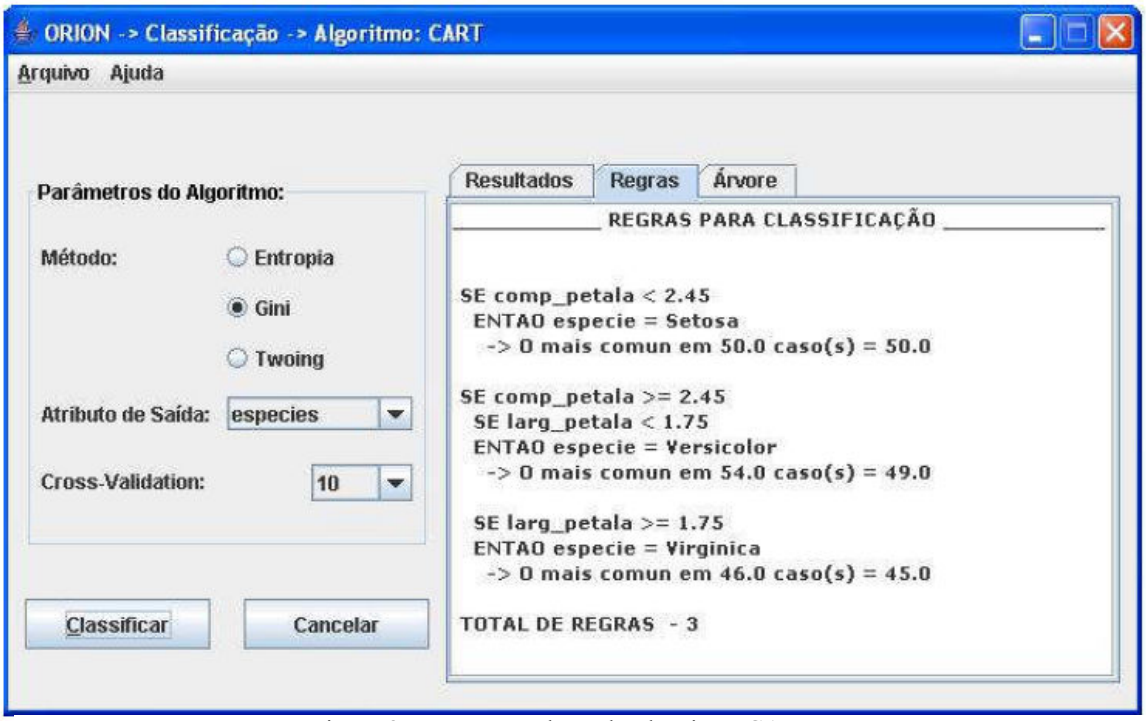

Figura 8. Regras geradas pelo algoritmo CART
Fonte: RAIMUNDO, L (2007)

Após a execução do algoritmo CART, conforme os parâmetros de entrada, pode-se visualizar as regras de classificação geradas (Figura 8), tendo-se como alternativa de interpretação a demonstração gráfica por meio de uma árvore (Figura 9).

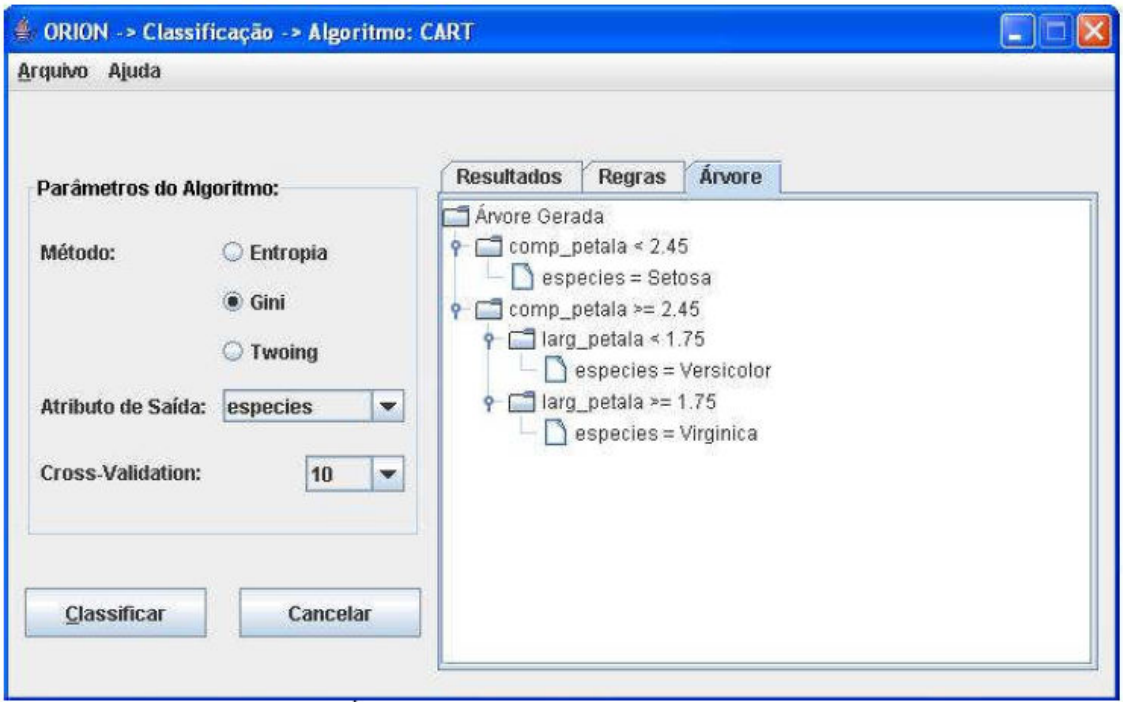

Figura 9. Árvore de decisão gerada pelo algoritmo CART
Fonte: RAIMUNDO, L (2007)

Além das tarefas de classificação e associação, a *Shell* Orion também possui implementada a clusterização. Inicialmente desenvolveu-se o algoritmo *K-means*, pois é um dos mais empregados para realizar a tarefa de agrupamento. Na execução deste algoritmo deve-se determinar um valor para *K* (número de *clusters*), esse é definido antes da execução e de forma muitas vezes aleatória. Posteriormente, calcula-se o centro desses *clusters*, e com os pontos dos dados remanescentes é calculada a distância entre o ponto e o centro do *cluster*. O ponto que apresentar a menor distância é considerado parte daquele grupo (GOLDSCHMIDT; PASSOS, 2005).

Figura 10. Resultados Obtidos por meio do algoritmo K-means
Fonte: MARTINS, D. (2007)

Na *Shell* Orion ao executar-se a tarefa de clusterização pelo algoritmo *K-means*, define-se o tipo de cálculo da distância (euclidiana ou *city-block*), o número de *clusters*, atributo de saída e a tabela de origem com os dados a serem clusterizados. Os resultados podem ser visualizados pelo sumário de dados (Figura 10) ou graficamente (Figura 11)

No módulo de clusterização encontra-se implementado o método de redes neurais pelo algoritmo de *Kohonen*, também pode ser encontrado nas bibliografias como mapa auto-organizável (KANTARDZIC, 2003, tradução).

Este algoritmo é capaz de reconhecer relações entre padrões apresentados à entrada da mesma rede neural, após um treinamento não-supervisionado[4]. A topologia básica da rede é ter a camada de entrada 1 x N que faz a leitura dos vetores-padrão dispostos na rede e a camada de saída representada pela matriz M x M de neurônios que forma uma resposta uni

[4] Os neurônios da rede não ficam restritos em relação à aprendizagem durante o treinamento (AZEVEDO; BRASIL; OLIVEIRA, 2000).

ou bi-dimensional. Este modelo é útil para realizar o reconhecimento de padrões, quando o número de classes não é conhecido "*a priori*" (AZEVEDO; BRASIL; OLIVEIRA, 2000).

Após a utilização do algoritmo de *Kohonen* pela *Shell* Orion, os resultados podem ser obtidos de diversas maneiras como: visualização dos grupos formados, informações estatísticas, exportar um arquivo com os dados em formato SQL (para utilização em outra tarefa de *data mining*) e na forma gráfica como visualizado na Figura 11.

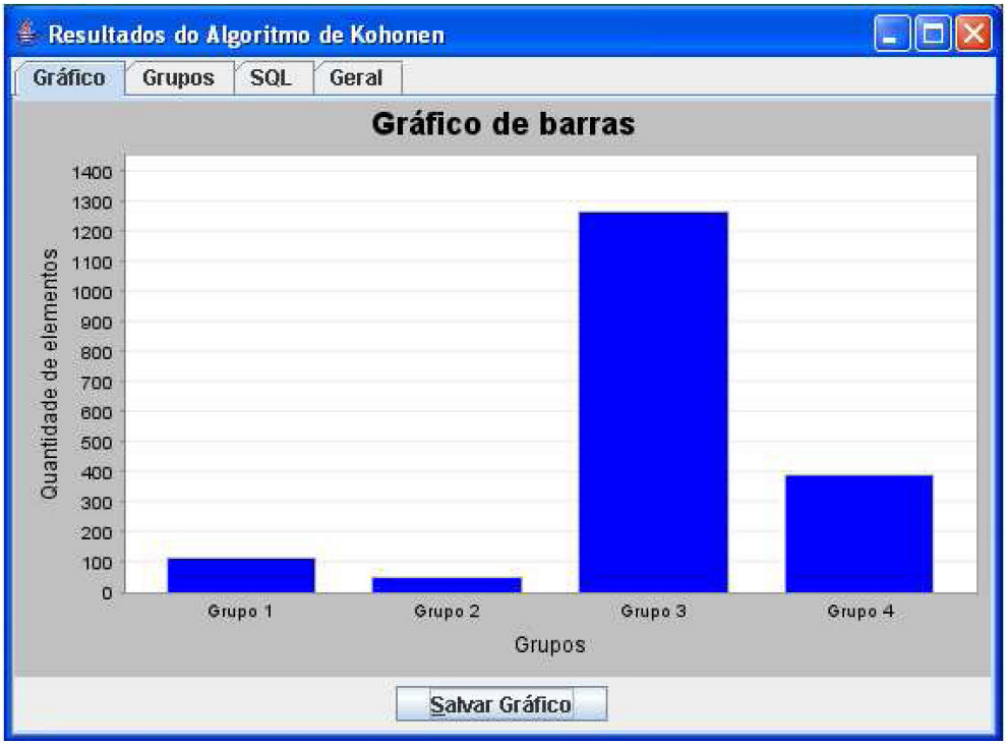

Figura 11. Gráfico gerado pelo algoritmo de Kohonen
Fonte: BORTOLOTTO, L (2007)

Em 2008, foi adicionada na *Shell* Orion o método de lógica *fuzzy* para clusterização por meio do algoritmo Gustafson-Kessel. Este algoritmo emprega uma norma adaptável da distância, com o objetivo de detectar grupos de formas geométricas diferentes na série de dados (GUERRA, 2006).

O algoritmo Gustafson-Kessel não trabalha com atributos nominais, caso seja necessário, estes devem ser convertidos para valores decimais. Isso ocorre porque o processo de clusterização realiza somente operações matemáticas (CASSETARI JÚNIOR, 2008).

Os resultados podem ser visualizados de diversas maneiras, na Figura 12, tem-se um resumo da clusterização com o Gustafson Kessel, onde são apresentados os parâmetros de entrada, informações sobre os grupos gerados, centros dos *clusters* e o atributo de saída.

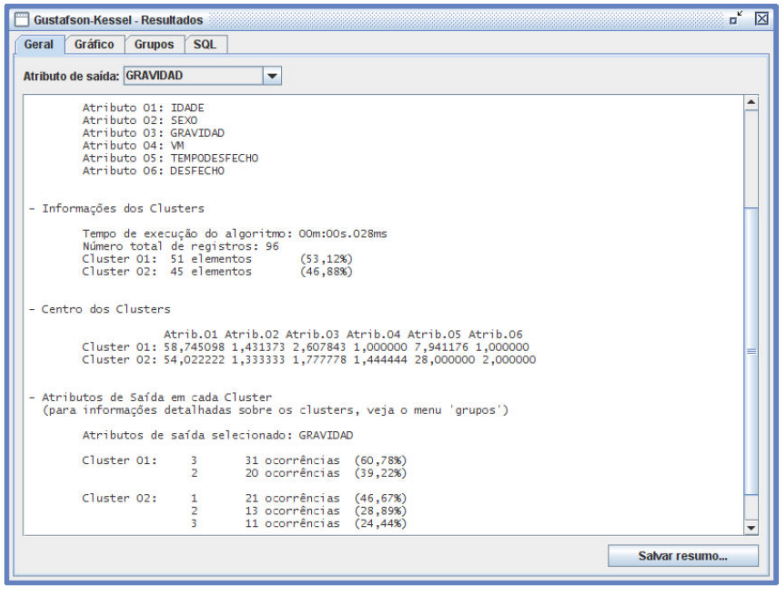

Figura 12. Grupos gerados pelo algoritmo de Gustafson-Kessel
Fonte: CASSETARI JÚNIOR, J. (2008)

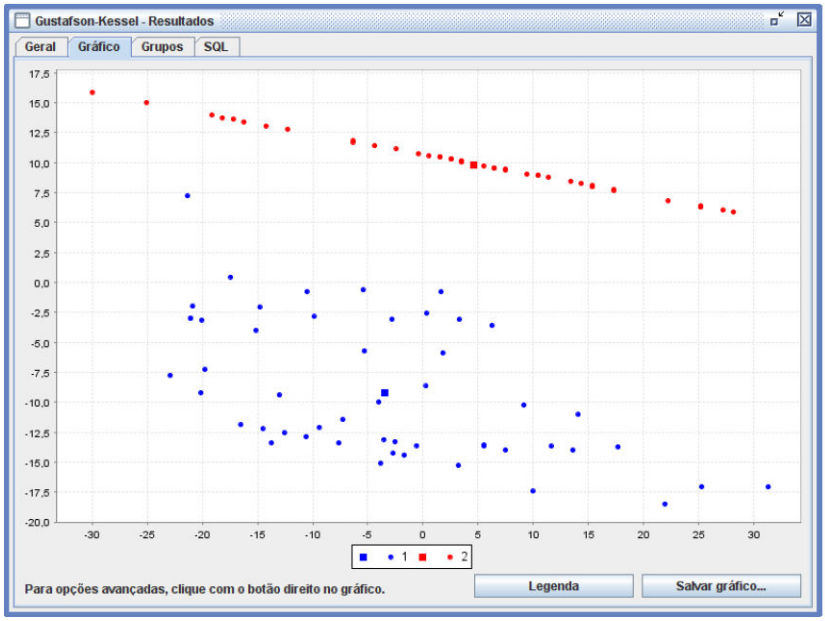

Figura 13. Gráfico gerado pelo algoritmo de Gustafson-Kessel
Fonte: CASSETARI JÚNIOR, J. (2008)

Os resultados também podem ser visualizados em forma gráfica (Figura 13), outra opção é a visualização de árvore como mostra a Figura 14, sendo que todas essas informações podem ser exportadas em formato SQL, e também ser posteriormente importadas para serem realizadas outras tarefas de *data mining*.

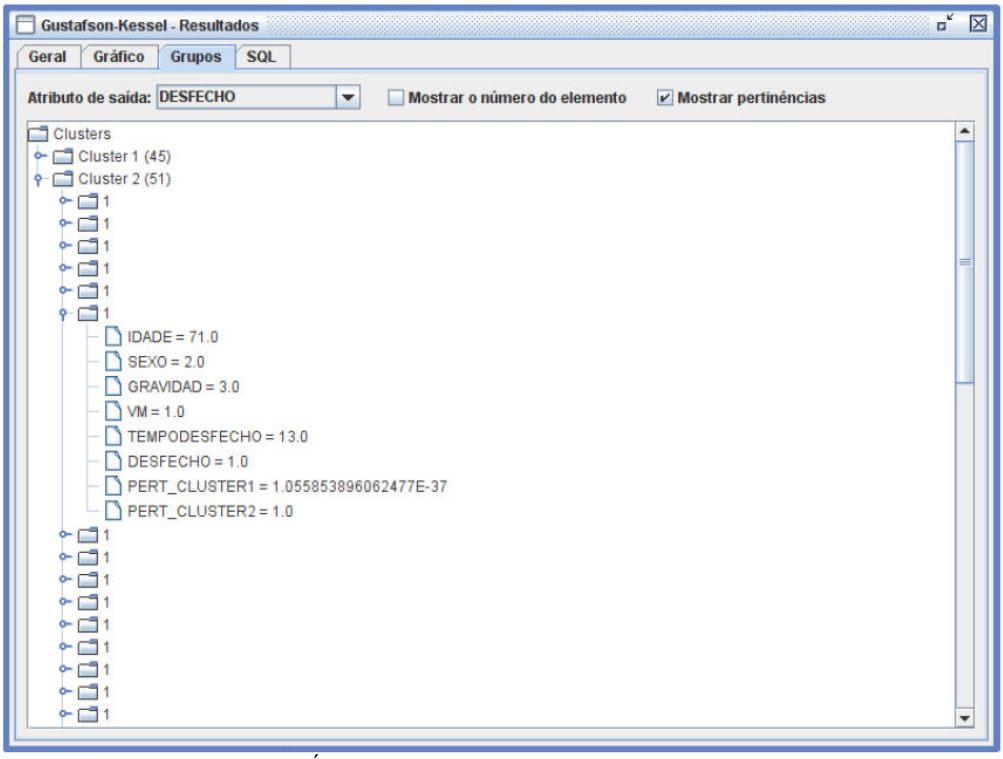

Figura 14. Árvore de decisão gerado pelo Gustafson-Kessel
Fonte: CASSETARI JÚNIOR, J. (2008)

Sendo assim, com base no objetivo do desenvolvimento de uma s*hell* que implemente várias tarefas e métodos. Esta pesquisa consiste em descrever a modelagem matemática e implementação do algoritmo Gath-Geva, que por meio do método de lógica *fuzzy* realiza a tarefa de clusterização. Com isso, serão ampliadas as opções disponíveis na ferramenta, permitindo-se a comparação de resultados obtidos por diferentes algoritmos que executam a mesma tarefa.

O algoritmo Gath-Geva utiliza a tarefa de Clusterização, sendo que essa também é composta de diversos métodos, como por exemplo, a lógica *fuzzy*. Com isso para melhor

entendermos e compreendermos o funcionamento do algoritmo, no capitulo 4, está dedicado a descrever os processos existentes na Clusterização.

4 A TAREFA DE CLUSTERIZAÇÃO

A clusterização é uma tarefa que particiona um conjunto de dados heterogêneos em subgrupos mais homogêneos conhecidos como *clusters.* A principal diferença entre a tarefa de clusterização e classificação, é que na classificação já existe uma classe predefinida sobre como os dados existentes na base de dados podem ser classificados, já na clusterização não existe nenhum parâmetro *a priori*, os *clusters* são formados com base na similaridade entre os dados (BERRY; LINOFF, 2004, tradução nossa).

A clusterização tem como principal função dividir a base de dados em *clusters*, sendo que os dados pertinentes ao mesmo *cluster* tendem a ser o mais similar possível, e o mais distinto em relação aos dados pertinentes a outros *clusters* (HAN; KAMBER, 2001, tradução nossa).

Na maioria dos algoritmos de clusterização, tem-se a necessidade do o usuário informar o número de grupos a ser considerado. Com base nesse valor, os registros de dados são então separados em grupos, de forma que dados similares pertençam ao mesmo grupo e registros diferentes fiquem em grupos distintos (GOLDSCHMIDT; PASSOS, 2005).

O processo de clusterização pode revelar grupos com diferentes formas e tamanhos em um espaço multidimensional. A Figura 15 (a) mostra um gráfico bidimensional de dados, sem sofrer o processo de clusterização. Para exemplificar a dificuldade no processo em determinar o melhor número de clusters, na Figura 15 (b) mostra-se um conjunto de dados agrupados em três *clusters* circulados com uma linha tracejada. Pode-se concluir que já seria um bom resultado para clusterização. Mas observando na Figura 15 (c) o mesmo conjunto de dados clusterizados em quatro grupos distintos, demonstrando um outro resultado. Essa arbitrariedade quanto à quantidade de *clusters* é um problema da clusterização (KANTARDZIC, 2003, tradução nossa).

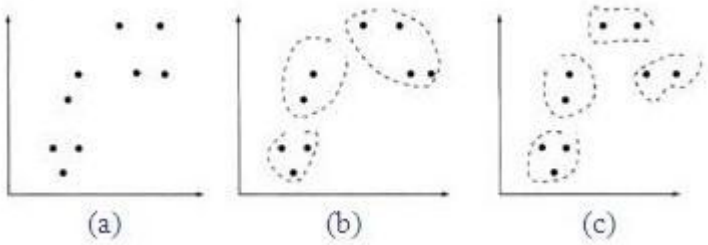

Figura 15. Processo de Clusterização
Fonte: KANTARDZIC, M (2003)

No caso apresentado anteriormente, os *clusters* podem ser facilmente identificados de forma visual, mas em um espaço de dados multidimensional, nem sempre será possível realizar o reconhecimento apenas avaliando os dados graficamente, por isso tem-se a necessidade de se elaborar abordagens que auxiliam na determinação da quantidade de *clusters* ideais para cada conjunto de dados (KANTARDZIC, 2003, tradução nossa).

Formalmente, supõe-se a existência de y pontos de dados, x_1, x_2, ..., x_y tais que cada ponto pertença a um espaço d dimensional R^d. O processo de agrupamento acontece separando-os em k *clusters*, ou seja, encontrar-se k pontos m_j em R^d conforme expressão (1), fazendo que essa seja minimizada, onde $d^2(x_i,m_j)$ indica a distância entre x_i e m_j. Os pontos m_j são chamados de centróides ou médias dos *clusters* (GOLDSCHMIDT; PASSOS, 2005).

$$\frac{\sum_i min_j d^2(x_i, m_j)}{N} \quad (1)$$

Normalmente os algoritmos de clusterização utilizam duas estruturas de dados para seu processamento, denominadas *matriz de dados* e *matriz de similaridade*, (HAN; KAMBER, 2001, tradução nossa).

A *matriz de dados* consiste em linhas que apresentam os n objetos do conjunto, como por exemplo: pessoas. E as colunas representam seus atributos, como idade, gênero, entre outros (HAN; KAMBER, 2001, tradução nossa).

$$\begin{bmatrix} x_{11} & \dots & x_{1f} & \dots & x_{1p} \\ \dots & \dots & \dots & \dots & \dots \\ x_{i1} & \dots & x_{if} & \dots & x_{ip} \\ \dots & \dots & \dots & \dots & \dots \\ x_{n1} & \dots & x_{nf} & \dots & x_{np} \end{bmatrix}$$

Figura 16. Matriz de Dados
Fonte: HAN, J.; KAMBER, M. (2001)

A *matriz de similaridade* relaciona a proximidade entre os dados do objeto, ou seja, representa a distância entre pares de objetos. Portanto são armazenadas as distâncias entre os objetos, e esses valores sempre serão positivos, pois trata-se de distância. Considerando *n* objetos, isso resulta numa matriz quadrada de tamanho *n* x *n* como mostra a Figura 17 (GOLDSCHIMIDT; PASSOS, 2005; OLIVEIRA, 2008):

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & 0 & \\ d(n,1) & d(n,2) & \dots & \dots & 0 \end{bmatrix}$$

Figura 17. Matriz de Similaridade
Fonte: HAN, J.; KAMBER, M. (2001)

Considerando isso, dependendo da situação utilizam-se algoritmos específicos, para melhor atender as necessidades e encontrar resultados satisfatórios. Sendo que pesquisadores buscam sempre desenvolver algoritmos mais eficientes e eficazes para clusterização de bases de dados. A seguir algumas características importantes que um algoritmo de clusterização busca contemplar (HAN; KAMBER, 2001, tradução nossa):

a) **escalabilidade:** certos algoritmos trabalham muito bem, quando se trata de pequenas bases de dados, no entanto, conforme esse repositório aumenta, o algoritmo pode perder em desempenho, ou não atingir resultados satisfatórios. Com isso é necessário desenvolver algoritmos versáteis que funcionem independentes da quantidade de dados a serem clusterizados;

b) **suporte a diversos tipos de algoritmos:** na maioria dos casos os algoritmos têm suporte a variáveis numéricas[5], no entanto em muitas aplicações tem-se a necessidade de se utilizar outros tipos de atributos como binários[6], nominais[7], ordinais[8], entre outros;

c) **descoberta de *clusters* de diferentes formas:** os algoritmos que realizam a clusterização baseados nas distâncias de Manhattan[9], Euclidiana[10], entre outras, tendem a formar *clusters* de formas parecidas em tamanho e densidade. Isso pode não representar a realidade, portanto os algoritmos devem permitir a formação de *clusters* de diferentes formas;

d) **minimizar a quantidade de parâmetros de entrada:** os algoritmos necessitam de alguns parâmetros para realizar a clusterização dos dados, como por exemplo, a quantidade de *clusters*. Esse tipo de informação numa base de dados com alta dimensionalidade é complicado para o usuário determinar com precisão;

e) **habilidade de lidar com dados com ruídos:** os dados podem estar ausentes ou com ruído. Algoritmos que são afetados por esses tipos de informações podem apresentar resultados de má-qualidade;

f) **insensibilidade dos registros de entrada:** dependendo da ordem de entrada, alguns algoritmos trabalhando com o mesmo conjunto de dados, podem apresentar resultados distintos;

---

[5] Variáveis que são representadas por números (LUGER, 2004).
[6] A variável assume apenas assume dois valores, por exemplo 0 ou 1 (LUGER, 2004).
[7] Similar ao binário, sendo que o número de classes de representação é maior do que 2 (LUGER, 2004).
[8] Atribui valores ou nomes, mas gera um conjunto ordenado de classes, como por exemplo tamanho, 1 = baixo, 2 = médio, 3 = alto (LUGER, 2004).
[9] Conhecida também como *city-block* e é similar a distância euclidiana, onde realiza somente o somatório da diferença entre dois pontos (HAN; KAMBER, 2005)
[10] Somatório das raízes quadradas das diferenças de distância entre dois pontos (BESDEK et al 2005).

g) **alta dimensionalidade:** uma base pode conter vários atributos. Muitos algoritmos conseguem ótimos resultados em dados que envolvem somente duas e três dimensões. Mas até seres humanos conseguem realizar essa avaliação com certa facilidade. Então a grande utilidade desses algoritmos é buscar clusterizar dados que apresentam maiores dimensões;

h) **condições para realizar a clusterização:** tem-se a necessidade nas aplicações de se efetuar a busca de *clusters*, onde os dados utilizados são constantes, ou seja, não se alteram;

i) **interpretação e usabilidade:** os usuários esperam das aplicações resultados que sejam interpretáveis, compreensíveis e utilizáveis. É importante ter isso como meta, pois influência a seleção dos métodos de clusterização.

Considerando que para conseguir atender todas essas características, em algumas situações utilizam-se mais de um algoritmo, a fim de melhorar os resultados apresentados pela tarefa de clusterização. Sendo que para realizar a implementação desses algoritmos são utilizados alguns métodos, onde para definir o mais apropriado deve-se levar em consideração, os tipos de dados, o domínio da aplicação, entre outros. (HAN; KAMBER 2000, tradução nossa).

O método utilizado no algoritmo Gath-Geva para tarefa de clusterização é a lógica *fuzzy*, pois permite que os dados pertençam a mais de um cluster simultaneamente aprimorando a formação dos resultados. Portanto no Capítulo 5, dedica-se para apresentar suas características e funcionamento.

5 O MÉTODO DE LÓGICA FUZZY PARA A TAREFA DE CLUSTERIZAÇÃO

Os algoritmos de clusterização desempenham a função de identificar subgrupos relevantes de dados, agrupando-os conforme as suas semelhanças. Assim, diferenciam-se dos demais subgrupos formados. Normalmente, em métodos como particionamento[11] e hierárquico[12], um ponto é atribuído a um único *cluster*. Porém, há situações que dados possuem características similares em *clusters* distintos. Nesses casos, a lógica *fuzzy*, demonstra a vantagem por levar em consideração essas pertinências de dados em relação aos diversos *clusters* (WANG; RUAN; KERRE, 2007, tradução nossa).

Outro ponto que tem que ser tratado pelos algoritmos é a complexidade das solicitações do usuário, que são expressos, por exemplo, em linguagem natural ou envolvendo vários critérios, A lógica *fuzzy* se torna útil nessa questão, devido à sua capacidade para lidar com os mais diversificados tipos de dados (WANG; RUAN; KERRE, 2007, tradução nossa).

Nesse contexto, pode-se analisar que na teoria clássica (*crisp*), um indivíduo faz parte ou não de um *cluster*. Mas isso nas aplicações do mundo real pode trazer problemas por não conseguir tratar certos tipos de conjuntos de dados (CHEN; PLAN, 2001, tradução nossa). Exemplificando essas dificuldades, suponha-se que sejam formados dois grupos, um contemplando as taças cheias e outro as vazias, qual o grupo que pertence a taça da Figura 18, esse tipo de questão não pode ser tratada pela lógica booleana ou clássica (COX, 1994, tradução nossa).

[11] Geralmente produzem agregados por função de aperfeiçoar um critério previamente definido, quer localmente (em um sub-conjutno de dados) ou globalmente (definido por todos os dados) (KANTARDZIC, 2003).
[12] Nessa técnica são produzidas diversas partições do banco de dados com base na junção ou divisão dos clusters de acordo com a medida de similaridade (OLIVEIRA, 2008).

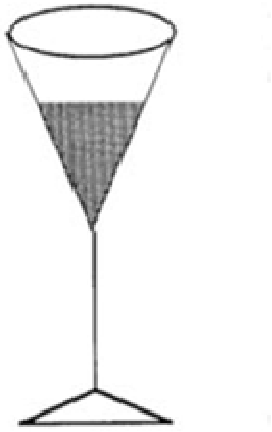

Figura 18. Representação de uma taça
FONTE: COX, E. (1994)

A lógica *fuzzy* tem como idéia fundamental determinar a relação de todos os dados em relação aos *clusters*, com isso é possível realizar uma transição contínua e gradual de um determinado cluster para outro, seguindo premissas, definidas dependendo de cada aplicação, na função de pertinência (MUNAKATA, 2008, tradução nossa).

5.1 LÓGICA FUZZY

A teoria de lógica *fuzzy* foi introduzida como uma extensão para definir conjuntos ordinários em torno de 1965. No início, foi restritamente conhecida pela comunidade científica até ser aplicada industrialmente em 1986, tornando-se um grande tema na década de 90. Onde muitas aplicações comerciais e industriais foram desenvolvidas no Japão, obtendo uma significativa melhora no desempenho e redução de custos (MUNAKATA, 2008, tradução nossa).

A lógica *fuzzy* busca utilizar mecanismos ou formas para tornar mais suave a transição de um determinado *cluster* para outro (GOLDSCHMIDT; PASSOS, 2005). Este

mecanismo é chamado de função de pertinência e indica o grau de adesão (pertinência), normalmente representado entre 0 e 1, onde maiores valores significam maior grau de pertinência, e vice-versa (KLIR; YUAN, 1995, tradução nossa)

Então tem-se um conjunto de dados Ω, onde um subconjunto A tem definida a sua função de pertinência, que deve ser denotada por A(x), onde possui valores no intervalo de [0,1]. Com isso, $A(x_0) = 0,6$ é uma notação de quanto o elemento 0 pertence ao sub-conjunto A. Caso todos os resultados da função de pertinência dos *clusters* forem 0 ou 1, então tem-se uma precisão não-*fuzzy* (BUCKLEY; JOWERS, 2006, tradução nossa).

Os sistemas *fuzzy* são modelados pelos engenheiros do conhecimento[13], que precisam de especialistas que conheçam o comportamento de onde vai ser aplicado o sistema, para com isso definir os estados que podem assumir (MCNEIL; THRO, 1994, tradução nossa).

Essa experiência (dos especialistas) é incorporada para realizar a definição das funções de entrada e saída de cada membro do sistema *fuzzy* (MCNEIL; THRO, 1994, tradução nossa).

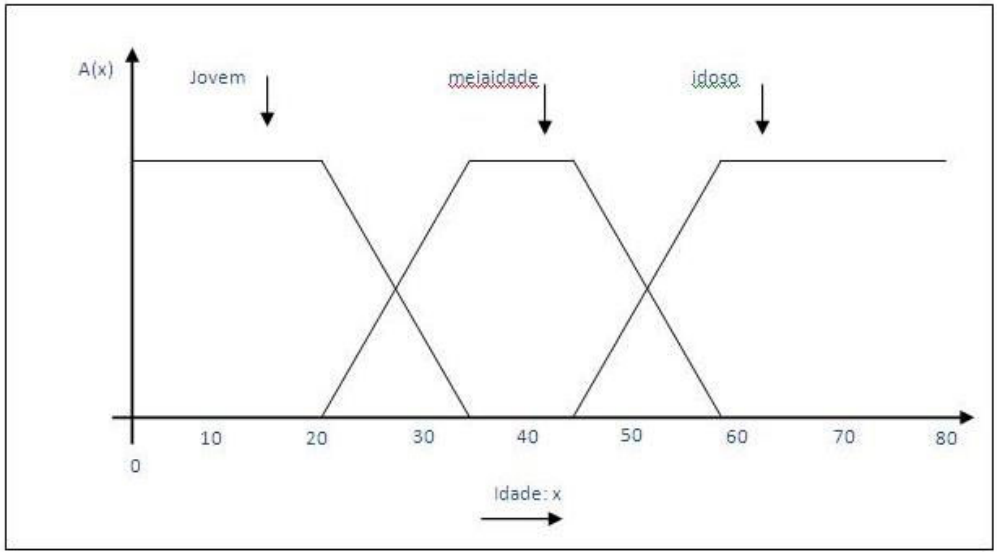

Figura 19. Distribuição de grupos pela lógica Fuzzy
Fonte: Adaptado de KLIR, G.; YUAN, B. (1995)

[13] Tem a função de codificar o conhecimento humano na forma de regras (BERRY; LINOFF, 2004).

A Figura 19 demonstra a aplicação dos conceitos básicos de conjuntos *fuzzy*, pode-se analisar três conjuntos, que representam *jovem*, *meia-idade* e *idoso*. O intervalo de idade é 0 a 80 anos, e o tipo de função de pertinência é trapezoidal. Abaixo pode ser visualizada a descrição das funções de pertinências da Figura 23 (KLIR; YUAN, 1995, tradução nossa).

$$\mu(jovem) = \begin{cases} 0, & se\ x \leq 20 \\ \dfrac{(35-x)}{15}, & se\ 15 < x < 35 \\ 1, & se\ x \geq 35 \end{cases}$$

$$\mu(meiaidade) = \begin{cases} 0, & se\ x \leq 20\ ou\ x \geq 60 \\ \dfrac{(x-20)}{15}, & se\ 20 < x < 35 \\ \dfrac{(60-x)}{15}, & se\ 45 < x < 60 \\ 1, & se\ 35 \leq x \leq 45 \end{cases}$$

$$\mu(idoso) = \begin{cases} 0, & se\ x \leq 45 \\ \dfrac{(x-45)}{15}, & se\ 45 < x < 60 \\ 1, & se\ x \geq 60 \end{cases}$$

Existem diversos tipos de funções que podem representar a pertinência dos elementos aos *clusters*, comumente as mais utilizadas são (ALVES, 2007):

a) **função triangular:** possui a forma de um triângulo, com três parâmetros de entrada (*a*,*b*,*m*), onde *a* e *b*, representam o intervalo de avaliação e *m* o valor que possui o maior grau de pertinência, ou seja 1.

$$\mu(x) = \begin{cases} 0, & se\ x \leq a \\ \dfrac{(x-a)}{(m-a)}, & se\ x \in [a,m] \\ \dfrac{(60-x)}{15}, & se\ 45 < x \\ 1, & se\ 35 \leq x \leq 45 \end{cases}$$

b) função trapezoidal: possui a forma de um trapézio, com três parâmetros de entrada (*a*,*m*,*n*,*b*), onde *a* e *m* representam a parte ascendente da função, enquanto *n* e *b* a descendente.

$$\mu(x) = \begin{cases} 0, & se\ x \leq a \\ \dfrac{(x-a)}{(m-a)}, & se\ x \in [a,m] \\ \dfrac{(b-x)}{(b-n)}, & se\ 45 < x \\ 0, & se\ 35 \leq x \leq 45 \end{cases}$$

c) função gaussiana: os parâmetros de entrada correspondem à (m, σ_k), na equação tem o valor modal[14] dado por m, e a dispersão[15] por σ_k.

$$\mu(x) = exp^{-\sigma k\,(x-m)^2}$$

A Figura 20 representa as funções descritas anteriormente, mostrada em gráfico. As letras demonstradas nas funções, também estão sendo apresentados nos gráficos.

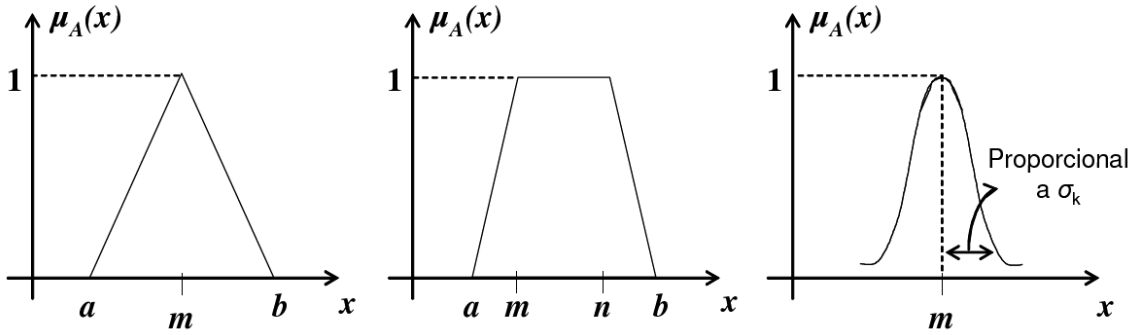

Figura 20. Gráficos representando as funções de pertinência
Fonte: ALVES, L (2007)

[14] Valor que aparece mais vezes numa seqüência de dados (ALVES, 2007)
[15] É a maior ou menor diversificação de um valor em relação a média ou mediana (ALVES, 2007)

A escolha do melhor formato da função nem sempre é uma tarefa simples, sendo que em alguns casos é um desafio tanto para o especialista da aplicação quanto para o engenheiro do conhecimento. Com isso, normalmente a opção utilizada tende a ser trapezoidal ou triangular, devido à facilidade de se definir os intervalos de pertinência e implementação (ALVES, 2007).

Existe uma arquitetura funcional ou sistema de inferência *fuzzy* (Figura 21), onde aplica os conceitos de lógica *fuzzy* no processamento de conhecimento, para isso é necessário realizar as etapas seguintes (KLIR; YUAN, 1995, tradução nossa):

a) **entrada:** entrada do sistema deve-se ser precisa, ou seja, entradas *crisp* (GOLDSCHMIDT; PASSOS, 2005).
b) **fuzzificação:** é responsável por transformar as entradas numéricas nos graus de pertinência em relação a todos os conjuntos fuzzy do sistema (PASSINO; YOURKOVICH, 1997, tradução nossa).
c) **máquina de inferência *fuzzy*:** tem como base os graus de pertinência calculados na etapa anterior, onde realiza a avaliação das regras existentes na base (MUNAKATA, 2008, tradução nossa).
d) **defuzzyficação:** transforma ou modifica os conjuntos de saída do sistema, que foram obtidos aplicando as regras de inferência, em valores precisos ou *crisp*.
e) **saída:** são os valores saída do sistema *fuzzy*.

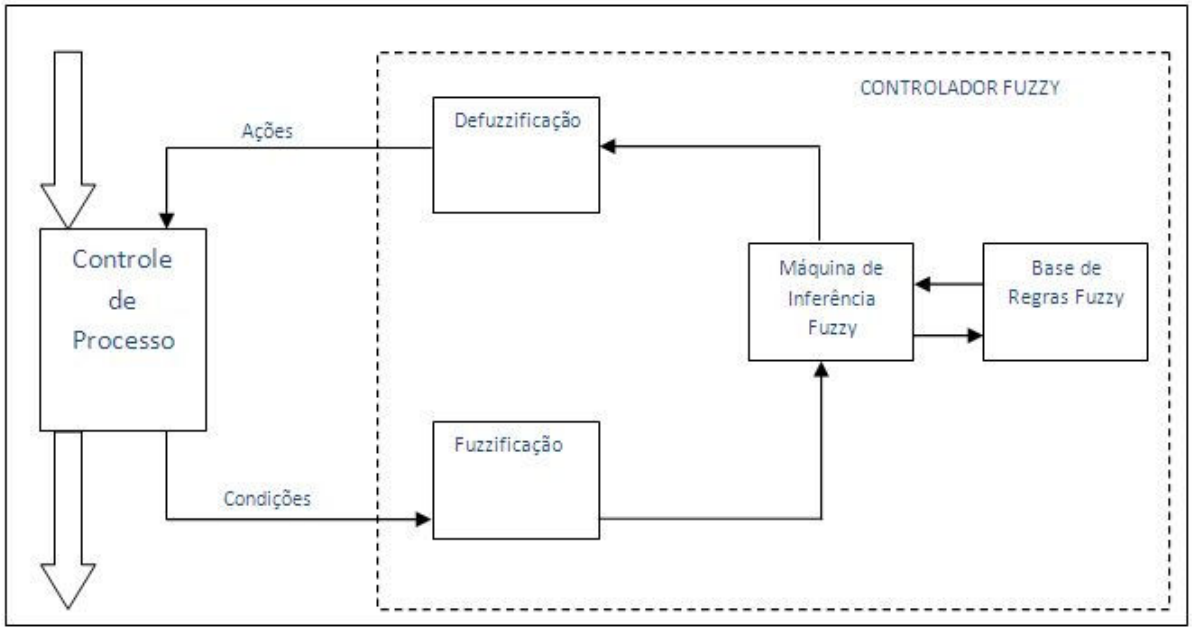

Figura 21. Arquitetura Funcional Genérica de um Sistema Fuzzy
Fonte: Adaptado de KLIR, G.; YUAN, B. (1995)

Dentre as aplicações, pode-se citar o método de lógica *fuzzy* para a tarefa de clusterização, que proporciona uma otimização na formação dos clusters, pois considera o grau de pertinência de cada dado em relação aos grupos existentes na aplicação (COX, 1994).

Existem alguns algoritmos que implementam o modelo *fuzzy* para a tarefa de clusterização, afim de aprimorar a qualidade dos resultados obtidos. A definição de qual é o melhor a ser utilizado, está principalmente relacionado ao domínio da aplicação onde está sendo aplicada a descoberta de conhecimento em bases de dados.

5.2 ALGORITMOS DE LÓGICA FUZZY PARA A TAREFA DE CLUSTERIZAÇÃO

Existem diversos algoritmos de lógica *fuzzy* para a tarefa de clusterização, onde cada um possui suas particularidades. Seu desempenho depende do tipo do domínio de aplicação e da quantidade de *clusters* que devem ser separados o conjunto de dados. Então

quanto mais conhecimento sobre o domínio de aplicação melhor serão os resultados encontrados (COX, 1994)

A seguir são demonstrados alguns exemplos de algoritmos que utilizam a lógica *fuzzy* na tarefa de clusterização.

5.2.1 Algoritmo Fuzzy C-Means

Em 1969 foi desenvolvido o um dos primeiros algoritmos com a utilização da lógica *fuzzy*, considerado uma extensão do algoritmo do *Hard C-means*[16] (HCM), conhecido como ISODATA, que até então era um dos mais populares métodos de clusterização (PUCCIARELLI, 2005).

O algoritmo *Fuzzy C-means* (FCM) é um dos algoritmos mais conhecido e tradicionalmente utilizado para aplicação da lógica *fuzzy*, sua formalização matemática definitiva foi proposta por Besdek em 1973 (BESDEK, 2005, tradução nossa).

O método FCM (Figura 22) pode ser descrito por meio de um algoritmo iterativo, que busca a minimização de um índice de desempenho, onde a cada interação é verificada a adequação dos *clusters* gerados. O desempenho do algoritmo é influenciado pela escolha do número de classes *c*, da inicialização dos centros dos *clusters*, da ordem de processamento dos vetores de dados, da forma definida para mensuração da distância e do critério de parada. Esse tipo de método é bastante recomendado para situação em que os *clusters* encontrados estão bastante compactos e distantes entre si (ZORZATE, 2006).

O algoritmo possui como entrada os seguintes parâmetros: o número desejado de *clusters c*, onde esse é definido algumas vezes sem conhecimento *a priori*, uma medida de distância como, por exemplo, a euclidiana, dentro outros. Onde $m \in (1,\infty)$, que define a

[16] Tem a principal característica de um registro pertencer somente a um único *cluster* (HAN;KAMBER, 2001)

distância permitida entre os pontos e os centros do *cluster*; e um critério de parada representado por um valor pequeno $\varepsilon > 0$. Iniciando a clusterização necessita-se inicializar os graus de pertinência da matriz e dos centros de *cluster* (ZORZATE, 2006).

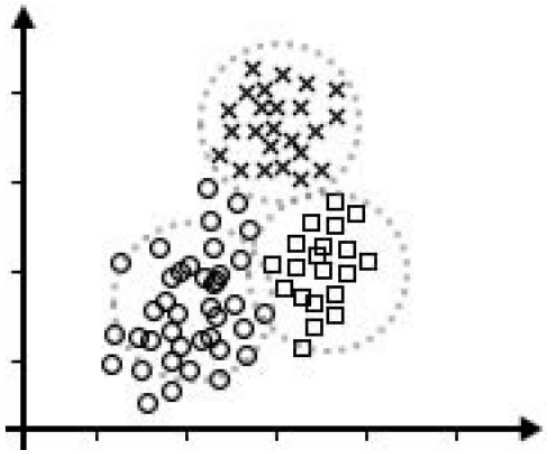

Figura 22. Clusterização pelo método *Fuzzy C-means*
FONTE: Adaptado de HOPPNER, F. et al . (1999)

5.2.2 Algoritmo Gustafson-Kessel

Na *Institute of Electrical and Electronics Engineers* (*IEEE*) *Conference on Decision and Control*, em 1979, Donald E. Gustafson e William C. Kessel publicaram um artigo, propondo uma modificação do algoritmo *Fuzzy C-Means*. Essa alteração originou outro algoritmo que ficou conhecido como Gustafson-Kessel (GK).

A principal alteração do algoritmo GK em relação ao FCM é a substituição da distância euclidiana pela Mahalanobis[17], com isso é implementado um matriz de covariância[18], possibilitando a obtenção de *clusters* com tamanhos diferentes e de forma elipsoidal (BEZDEK, 2005, tradução nossa)

[17] Difere-se da distância euclidiana por implementar uma matriz de covariância (BEZDEK, 2005)
[18] Matriz contendo as medidas de distâncias entre um grupo de elementos (GATH;GEVA, 1989)

Os *clusters* apresentados pelo algoritmo FCM são esféricos e de tamanhos parecidos, mas devido à introdução da matriz de covariância, resulta numa adaptação em relação a cada *cluster* formado, possibilitando a formação de *clusters* de tamanhos diferenciados. Isso torna o algoritmo GK mais versátil na formação dos conjuntos de dados, mesmo tratando-se de *clusters* com diferentes formas e tamanhos. No entanto, para realizar a utilização do algoritmo GK necessita de mais processamento computacional em relação FCM. (HOOPNER, 1999, tradução nossa).

A Figura 23 demonstra a clusterização pelo algoritmo Gustafson-Kessel, onde se pode perceber a presença de formas e tamanhos diferentes, e após o processamento, o algoritmo GK realizou a separação dos dados em três *clusters*. É importante ressaltar que é necessário informar a quantidade de clusters antes de iniciar o algoritmo, isso pode ser uma tarefa complicada sem o devido conhecimento *a priori*.

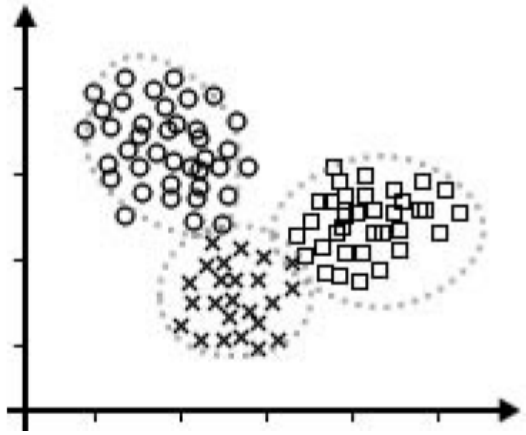

Figura 23. Clusterização por Gustafson-Kessel
FONTE: Adaptado de HOPPNER, F. et al . (1999)

5.2.3 Algoritmo Gath-Geva

Esse algoritmo foi publicado em 1989 na *IEEE Pattern Analysis and Machine Intelligence* pelos pesquisadores Gath e Geva, propondo-se modificações nos algoritmos FCM e GK, a fim de melhorar o desempenho dos mesmos no processo de clusterização dos dados.

O algoritmo Gath-Geva (GG) também possui implementada uma matriz de covariância, possibilitando a formação de *clusters* de diferentes formas e tamanhos, mas a grande diferença está na distância utilizada, onde a essa possui um termo exponencial, sendo calculada baseada numa estimativa de probabilidades (HOPPNER, 1999, tradução nossa)

Na Figura 24 são apresentados os resultados de uma clusterização utilizando o algoritmo Gath-Geva, nota-se a formação de 3 clusters distintos de tamanhos diferentes, contemplando a necessidade de envolver os registros dentro dos conjuntos de dados formados.

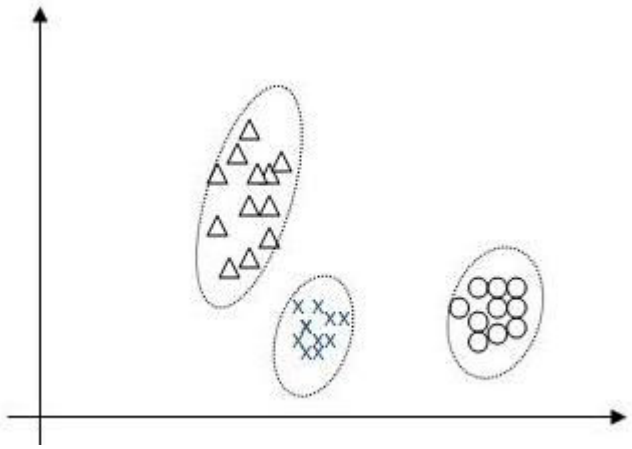

Figura 24. Clusterização pelo algoritmo Gath-Geva
FONTE: Adaptado de HOPPNER, F. et al . (1999)

Considerando que o principal objetivo da pesquisa é de implementar o algoritmo Gath-Geva, tem-se que aprofundar os conceitos do funcionamento do algoritmo, mostrando-se as suas características, e fundamentos matemáticos. Esses temas são detalhadamente apresentados no Capítulo 6 desta pesquisa.

6 O ALGORITMO GATH-GEVA

Em Julho de 1989, na *IEEE Transactions on Pattern Analysis and Machine Intelligence*, I. Gath e Amir B. Geva publicaram um artigo intitulado *Unsupervised Optimal Fuzzy Clustering*. Este artigo descrevia uma modificação nos algoritmos *Fuzzy C-Means* e *Gustafson-Kessel*, conhecido atualmente como algoritmo Gath-Geva.

O algoritmo de Gath-Geva é uma extensão do algoritmo Gustafson-Kessel, que também tem a possibilidade de formar *clusters* de tamanho e densidades diferentes, podendo interpretar dados multidimensionais, utilizando distribuição de variáveis aleatórias (HOPPNER, 1999 tradução nossa).

Os *clusters* de formas diferentes podem ser obtidos por meio de uma adequada definição de *clusters*, ou por meio de diferentes medidas de distância. O algoritmo GK tem sido freqüentemente utilizado pelo modelo de inferência Takagi-Sugeno[19] para identificar modelos ou padrões. Porém com a utilização do algoritmo GG, obtêm-se *clusters* mais precisos, com menor erro de aproximação que o algoritmo GK, isso devido à utilização da distância de forma exponencial (ABONYI, 2002, tradução nossa).

Além da incorporação da distância exponencial no algoritmo Fuzzy C-means, o GG utiliza, similarmente ao algoritmo GK, uma matriz de covariância, permitindo a descoberta de *clusters* de tamanhos e formas diferentes e independentes entre si. Resultando com isso em *clusters* com maior precisão nos resultados comparado com o FCM dependendo do conjunto de dados (GATH; GEVA, 1989, tradução nossa).

O algoritmo foi descrito baseado em diversos conceitos matemáticos, sendo que para melhor entendimento foi descrito detalhadamente todas as etapas necessárias para a resolução do calculo.

[19] Consiste de regras *fuzzy*, onde um sistema não-linear é sub-dividido em partes (COSTA, NEPOMUCENO, NETO, 2004)

A primeira etapa do cálculo é realizar o preenchimento da matriz de pertinências, que possui todas as pertinências dos dados em relação aos *clusters*. Esta matriz deve ser inicializada antes do começo dos cálculos. Esses valores não interferem no resultado final, por isso esses dados podem ser aleatórios, obedecendo à regra abaixo (ABONYI, 2002 tradução nossa).

$$U \in \mathbb{R}^{N\,x\,K} com\ \mu_{j,i} \in [0,1], \forall i,k; \sum_{j=1}^{N} \mu_{j,i} = 1, \forall k; 0 < \sum_{i=1}^{K} \mu_{j,i} < K, \forall i \qquad (2)$$

Onde:

a) **K**: número total de *clusters*;

b) $\mu_{i,k}$: grau de pertinência do j-ésimo elemento em relação ao i-ésimo *cluster*;

c) **N**: número de elementos ou dados.

A regra (2) descreve que a soma de todos os graus de pertinência do elemento *i* em relação a todos os *clusters* deve ser igual a 1. Isso deve ser realizado sempre que os graus de pertinências forem atualizados.

O algoritmo Gath-Geva baseia-se na minimização da soma ponderada das distâncias quadradas entre os pontos Z_k e os centros dos *clusters*, que é representado pela função:

$$J(U,V) = \sum_{j=1}^{N}\sum_{i=1}^{K} (\mu_{ij})^q\, d^2(X_j, V_i); \qquad (3)$$

Onde:

a) **J** = valor a ser minimizado, no qual determina os centros dos *clusters* e os graus de pertinências;

b) **K:** número total de *clusters*;

c) **N:** número total de elementos ou dados;

d) $\mu_{j,i}$: grau de pertinência do j-ésimo elemento em relação ao i-ésimo *cluster*;

e) **q:** parâmetro de *fuzzificação*;

f) **d^2 (X_j,V_i)**: é a distância entre o j-ésimo elemento X_j e o i-ésimo centro V_i;

Aplicando-se a equação (3) representam de forma genérica algoritmos como Fuzzy C-means, entre outros, sendo que a diferença, basicamente é a função ou forma como é obtida a distância entre os elementos e os *clusters*. Em casos onde há presença de *clusters* com densidades e formas diferenciadas, utiliza-se no algoritmo GG uma distância exponencial, baseada na estimativa *fuzzy* de probabilidade máxima (FMLE – *fuzzy maximum likelihood estimates*) (GATH; GEVA, 1989, tradução nossa).

No algoritmo Gath-Geva, similarmente ao algoritmo GK, são utilizadas cinco etapas para efetuar a clusterização dos dados:

a) calcular os centros dos *clusters*;

b) calcular a matriz de covariância *fuzzy*;

c) determinar as distâncias entre os elementos e os *clusters*;

d) atualizar a matriz de pertinência;

e) verificar a condição de parada.

Cada uma dessas etapas possui fórmulas específicas para realização dos cálculos, a seguir apresenta-se dividido em subseções essas etapas, proporcionando um maior detalhamento facilitando o entendimento.

6.1 CÁLCULO DO CENTRO DOS *CLUSTERS*

Os *clusters* são formados por elementos de tamanhos aproximados. Cada *cluster* é representado pelo seu centro. Essa representação pode também ser chamada de protótipo, uma vez que esse é utilizado para representar os dados que pertencem atualmente ao *cluster* no cálculo dos graus de pertinências (HOOPNER, 1999, tradução nossa).

A equação (4) é utilizada para calcular o centro dos *clusters* existentes no processo de clusterização.

$$V_i = \frac{\sum_{j=1}^{N} (\mu_{ij})^q X_j}{\sum_{j=1}^{N} (\mu_{ij})^q} \quad (4)$$

Onde:

a) $\mathbf{V_i}$: centro do i-ésimo *cluster*;

b) **N**: número total de elementos;

c) $\boldsymbol{\mu_{j,i}}$: grau de pertinência do j-ésimo elemento em relação ao i-ésimo *cluster*;

d) **q:** parâmetro de *fuzzificação*;

e) $\mathbf{X_j}$: é o elemento j-ésimo.

É realizado para cada *cluster* o cálculo individual do seu centro, a equação realiza o somatório da multiplicação de todos os graus de pertinência e os conjuntos de dados, sendo que esses graus são elevados ao parâmetro fuzzificador[20]. Na seqüência, o resultado obtido é dividido pelo somatório de todos os graus de pertinências elevado pelo parâmetro fuzzificador.

Quando o processo do cálculo ou a atualização dos centros dos *clusters* é concluído, deve-se então encaminhar o cálculo da matriz de covariância *fuzzy*, separadamente para cada grupo.

6.2 CALCULAR A MATRIZ DE COVARIÂNCIA *FUZZY*

Os algoritmos Gustafson-Kessel e Gath-Geva utilizam para os cálculos da clusterização uma matriz de covariância para cada *cluster*. Possibilitando por meio dessas, *clusters* ou aglomerados com formas diferenciadas, normalmente elipses. (HOOPNER, 1999 tradução nossa).

[20] É o que determina a fuzzificação entre os elementos e os *clusters* (COX, 2005)

Após ser realizado o cálculo dos centros dos *clusters*, utiliza-se a equação (5) para obter a matriz de covariância (GATH; GEVA, 1989 tradução):

$$F_i = \frac{\sum_{j=1}^{N} \mu_{ji} \left(X_j - V_i\right)^T \left(X_j - V_i\right)}{\sum_{j=1}^{N} \mu_{ji}} \quad (5)$$

Onde:

a) $\mathbf{F_i}$: matriz de covariância i-ésimo *cluster*;

b) **N**: número de elementos;

c) $\boldsymbol{\mu_{j,i}}$: grau de pertinência do j-ésimo elemento em relação ao i-ésimo *cluster*;

d) $\mathbf{X_j}$: é o elemento j-ésimo;

e) $\mathbf{V_i}$ = centro do i-ésimo *cluster*;

A matriz de covariância de cada *cluster* é calculada por meio da multiplicação do grau de pertinência pela subtração do vetor de dados com o centro do *cluster*. O valor obtido é então multiplicado pela mesma subtração, no entanto na forma transposta. Ao final é realizado o somatório destes valores. Finalizado o cálculo do numerador da equação (5) é então dividida pelo somatório dos graus de pertinências de todos os elementos, conseguindo por fim, obter a matriz de covariância *fuzzy* do elemento i-ésimo.

Ao obter todas as matrizes de covariância, pode-se então calcular as distâncias dos elementos em relação ao centro dos *clusters* encontrado.

6.3 DETERMINAÇÃO DAS DISTÂNCIAS ENTRE OS ELEMENTOS E OS *CLUSTERS*

A função da distância do algoritmo Gath-Geva diferentemente do algoritmo FCM e o GK, possui um termo de probabilidade *apriori*, onde isso significa, que é realizado o cálculo da probabilidade do elemento pertencer ou não ao *cluster*, ou seja, uma pertinência pequena significa uma baixa probabilidade, ao contrário de uma pertinência maior onde

aumenta a probabilidade de o elemento se tornar pertencente ao *cluster* (HOOPNER, 1999, tradução nossa).

Além desse termo de probabilidade, a equação para o cálculo da distância do algoritmo Gath-Geva utiliza um termo exponencial, como também a determinante e o inverso da matriz de covariância, conforme pode ser observado na fórmula (6) (GATH; GEVA, 1989 tradução nossa).

$$d_e^2(X_j, V_i) = \frac{[\det(F_i)]^{1/2}}{P_i} exp\left[\frac{(X_j - V_i)^T F_i^{-1}(X_j, -V_i)}{2}\right] \quad (6)$$

Onde:

a) **d^2 (X_j,V_i)**: é a distância entre o j-ésimo elemento X_j e o i-ésimo centro V_i;

b) **X_j**: é o elemento j-ésimo;

c) **V_i** = centro do i-ésimo *cluster*;

d) **F_i**: matriz de covariância do i-ésimo *cluster*;

e) **P_i**: probabilidade *a priori* do i-ésimo *cluster*.

A distância é calculada pela raiz quadrada da determinante da matriz de covariância, sendo esse resultado dividido pela probabilidade *a priori* calculada pela equação (7). O resultado obtido tem que ser elevado ao termo exponencial, cujo valor é determinado pela multiplicação entre a subtração do elemento com o centro do *cluster* pela matriz de covariância inversa. Após isso, tem que ser realizada a multiplicação pela mesma subtração, elemento e o centro do *cluster*, no entanto na forma transposta[21].

A probabilidade *a priori* de um elemento pertencer a um determinado *cluster* tem que ser previamente calculada conforme a equação (7) (GATH; GEVA, 1989 tradução nossa).

$$P_i = \frac{1}{N}\sum_{j=1}^{N} \mu_{ji} \quad (7)$$

[21] É o resultado da troca das linhas pelas colunas de uma matriz qualquer (SANTOS, 2007).

Onde:

a) $\mathbf{P_i}$: probabilidade *a priori* do i-ésimo *cluster*;

b) **N**: número de elementos;

c) $\mu_{j,i}$: grau de pertinência do j-ésimo elemento em relação ao i-ésimo *cluster*;

O cálculo do inverso do número de elementos multiplicado pelo somatório das pertinências de todos os elementos em relação os *cluster*s resulta na probabilidade *a priori* do i-ésimo elemento, onde esse valor é utilizado para o cálculo da distância demonstrada na equação (6).

É importante ressaltar que devido a utilização dessa função exponencial, ao executar o algoritmo Gath-Geva pode ocorrer o problema de gerar um *overflow*[22], isso por que os valores das distâncias, podem se tornar muito grandes, ocasionando o problema (HOPPNER, 1999 tradução nossa).

Com os valores obtidos, pode-se saber a distância entre os elementos e os centros dos *cluster*s calculados. Deve-se então realizar a atualização das pertinências dos elementos em relação aos *cluster*s.

6.4 ATUALIZAÇÃO DOS GRAUS DE PERTINÊNCIAS

Após a realização do cálculo das distâncias, obtida pela equação (6), deve-se então buscar os novos valores de pertinências entre o i-ésimo elemento em relação aos demais centros dos *cluster*s. Essa atualização é realizada pela equação 8: (GATH; GEVA, 1989, tradução nossa):

[22] Ocorre quando o resultado é um número maior (em *bits*) do que a quantidade de *bits* disponível para representá-lo (HOPPNER, 1999)

$$\mu_{ji} = \frac{\frac{1}{d_e^2(X_j, V_i)}}{\sum_{k=1}^{K} \frac{1}{d_e^2(X_j, V_k)}} \quad (8)$$

Onde:

a) $\mu_{j,i}$: grau de pertinência do j-ésimo elemento em relação ao i-ésimo *cluster*;

b) **K:** número total de *clusters*;

c) **$d^2(X_j, V_i)$**: é a distância entre o j-ésimo elemento X_j e o i-ésimo centro V_i;

d) **$d^2(X_j, V_k)$**: é a distância entre o j-ésimo elemento X_j e o k-ésimo centro V_k.

O cálculo é realizado dividindo o valor 1 pela distância entre o j-ésimo elemento e o i-ésimo centro V_i. Calculada a parte do numerador, deve-se realizar o somatório da divisão do número 1 pela distância entre j-ésimo elemento e o k-ésimo centro. Obtendo-se o denominador da equação, realiza-se a divisão, gerando com isso a pertinência atualizada do j-ésimo elemento em relação ao i-ésimo *cluster*.

Finalizando todas as atualizações é necessário verificar se o resultado obtido já é o suficiente para encerrar com o processo de clusterização dos dados.

6.5 CÁLCULO DA CONDIÇÃO DE PARADA

A condição de parada tem por principal função determinar o momento em que se encerra o processo de clusterização, isso tem relação com a precisão que o usuário deseja nas informações e o tempo de execução, sendo que quanto mais baixo for à margem de erro escolhida pelo usuário, maior o número de iterações necessárias (BEZDEK, 2005, tradução nossa).

Depois de realizada a atualização da matriz de pertinências deve-se realizar o teste lógico para verificar se o resultado obtido já satisfaz o valor estabelecido inicialmente para o

processo de clusterização ser encerrado. Na equação (9) verifica-se o calculo de condição de parada (GATH; GEVA, 1989, tradução nossa):

$$\max_{ij}\left[\left|{U_{ij}}^{l} - {U_{ij}}^{(l-1)}\right|\right] < \varepsilon \quad (9)$$

Onde:

a) **U**: matriz de pertinência,

b) **l**: número de iterações,

c) **ε**: valor do erro.

A fórmula verifica se a maior diferença em modulo da matriz de pertinência atual subtraída da matriz imediatamente anterior é menor que o valor do erro, caso seja verdadeiro, o processo de clusterização é encerrado, caso esse valor ainda permaneça maior que o valor de erro, deve-se continuar executando o algoritmo Gath-Geva.

Essas etapas apresentadas fazem parte de todo o processo necessário para executar o algoritmo de Gath-Geva, algumas aplicações podem exemplificar as formas de como o algoritmo pode ser utilizado para desempenhar a tarefa de clusterização.

7 TRABALHOS CORRELATOS

Os algoritmos de lógica *fuzzy* são amplamente abordados em projetos de pesquisa em todo mundo, por isso a seguir são apresentados alguns exemplos de aplicações desses estudos que envolvam principalmente o algoritmo Gath-Geva para o processo de clusterização.

7.1 EFICIÊNCIA DE MÉTODOS DE AGRUPAMENTO DE DADOS NA MODELAGEM NEBULOSA TAKAGI-SUGENO

Esse trabalho foi uma Dissertação de mestrado, apresentada ao Programa de Pós-Gradução em Engenharia de Produção e Sistemas da Pontifícia Universidade Católica do Paraná na cidade de Curitiba em 2007, foi desenvolvido por Luciano Alves.

A proposta da dissertação foi levantar aspectos relevantes sobre a modelagem e previsão de séries temporais, sendo direcionado sobre dados do sistema dinâmico e os métodos de agrupamento por meio da utilização da lógica *fuzzy*. Com isso foi realizado um levantamento do modelo matemático entre alguns algoritmos como: Gustafson-Kessel, Fuzzy C-means, Gath-Geva, entre outros, afim de realizar uma comparação entre o desempenho dos mesmos (ALVES, 2007).

Na Figura 25 são apresentados as saídas real e estimada, onde mostra o resultado para análise de série do álcool onde o melhor resultado obtido foi com a utilização do modelo nebuloso de Takagi-Sugeno e o método de agrupamento de Gath-Geva Modificado. O inconveniente desse resultado foi à alta complexidade computacional em virtude das 25 regras utilizadas para a entrada do modelo *fuzzy* (ALVES, 2007).

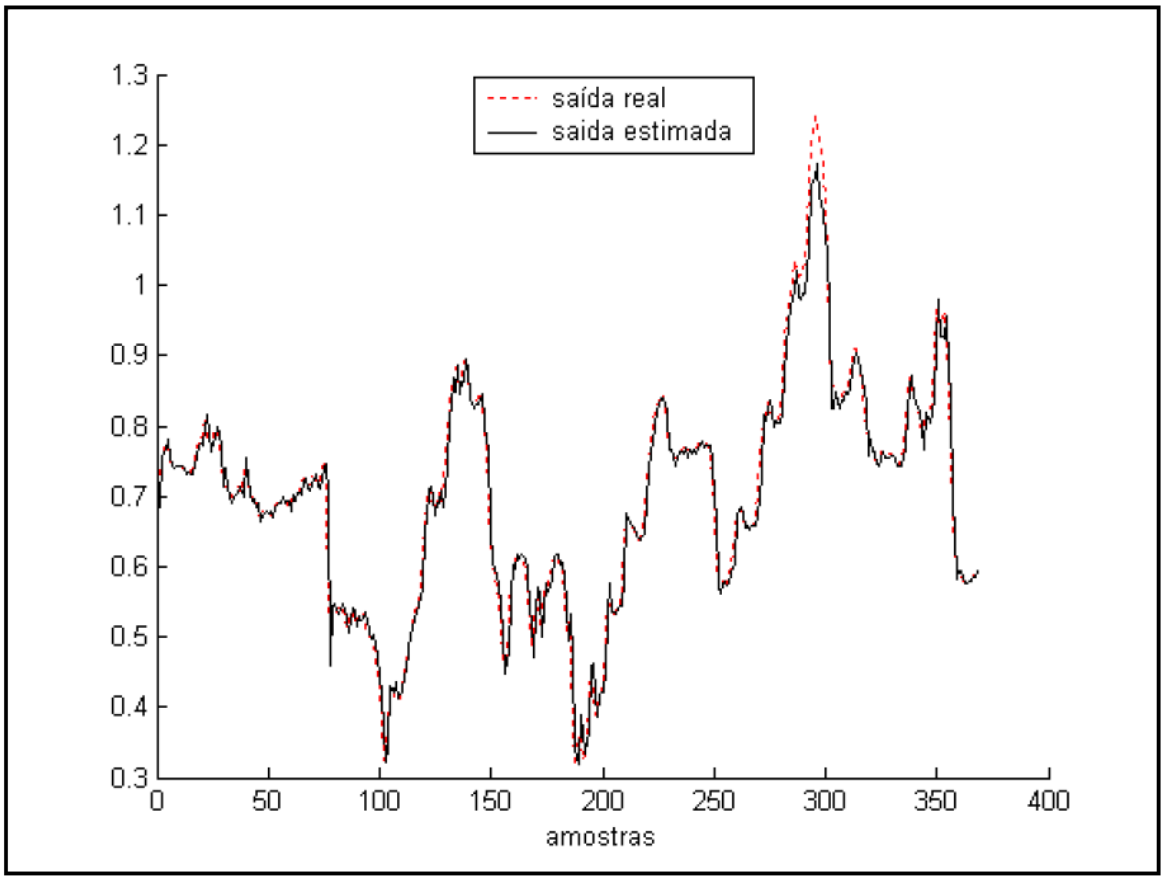

Figura 25. Saída real e estimada com o algoritmo Gath-Geva Modificado
FONTE: ALVES, L. (2007)

Tem que ser ressaltado que o algoritmo GG é um método pouco robusto, pois correlaciona uma distância do centro com uma distribuição de probabilidades. Com isso, resulta em uma função de pertinência que decresce lentamente a partir dos centros, convergindo para um mínimo local próximo. Portanto para contornar esse problema, foi proposto na dissertação a utilização de um algoritmo GG modificado, onde esse é baseado na maximização da esperança de modelos Gaussianos, sendo utilizado para estimar a densidade de probabilidade da base de dados (ALVES, 2007).

7.2 ANÁLISE DE MÉTODOS DE AGRUPAMENTO PARA O TREINAMENTO DE REDES NEURAIS DE BASE RADIAL APLICADAS À IDENTIFICAÇÃO DE SISTEMAS

Esse trabalho é uma Dissertação de mestrado que foi desenvolvida por Fábio Guerra apresentada ao Programa de Pós-Gradução em Engenharia de Produção e Sistemas da Pontifícia Universidade Católica do Paraná na cidade de Curitiba em 2007, como requisito parcial para obtenção do título de Mestre em Engenharia de Produção e Sistemas.

A proposta da dissertação é apresentar um estudo comparativo de desempenho dos quatro métodos de agrupamentos, *hard c-means*, *fuzzy c-means*, Gustafson-Kessel e Gath-Geva. Esses foram analisados no processo de treinamento de redes neurais de base radial[23]. Onde posteriormente as redes são aplicadas em uma configuração apropriada de sinais de entrada e saída com o objetivo de identificar sistemas não-lineares [24](GUERRA, 2006).

Com o desenvolvimento da dissertação concluiu-se que os quatro métodos de agrupamento obtiveram resultados satisfatórios, mas um detalhe importante foi avaliado, onde o algoritmo Gustafson-Kessel realiza a formação de grupos alongados (elipsóides), isso devido o advento da utilização da matriz de covariância, utilizando a medida de distância de Mahalanobis. Já o algoritmo GG possui em sua estratégia no cálculo da distância ao centro que envolve uma distribuição de probabilidades, e como sua função de pertinência decresce mais lentamente, fica exposto a possibilidade de convergir indesejavelmente para um mínimo local (GUERRA, 2006).

[23] É considerado como uma rede neural para um problema de aproximação de curva em um espaço multidimensional (GUERRA, 2005)

[24] Tem por caracteristica apresentar uma variedade nos comportamentos dinamicos, dificultando o gerenciamento por meio de métodos convencionais de modelagem e identificação linear (GUERRA, 2005)

7.3 MODIFIED GATH-GEVA FUZZY CLUSTERING FOR IDENTIFICATION OF TAKAGI-SUGENO FUZZY MODELS

Esse artigo foi publicado em 2002 na *IEEE Transactions on Systems Man and Cybernetics*, pelos pesquisadores Janos Abonyi e Ferenc Szeifert do Departamento de Eng. De Processo da Universidade de Veszprem da Húngria, e pelo também pesquisador Robert Babuska do departamento de tecnologia de informação e sistemas da Universidade de Tecnologia Delft da Holanda.

O artigo apresentado pelos autores faz uma abordagem sobre a máquina de inferência Takagi-Sugeno, onde com o intuito de melhorar o desempenho do algoritmo Gath-Geva para esse modelo, foi proposta de uma mudança no algoritmo, com base na expectativa de maximização e identificação de modelos de misturas gaussianas. Sendo que essa nova formulação foi aplicada a problemas conhecidos como identificação de imagens (ABONYI, 2002, tradução nossa).

A Figura 28 mostra a clusterização pelo algoritmo Gath-Geva original, e a Figura 29 apresenta a modificação proposta, onde essa mostrou um desempenho ligeiramente melhor do que o original (ABONYI, 2002, tradução nossa).

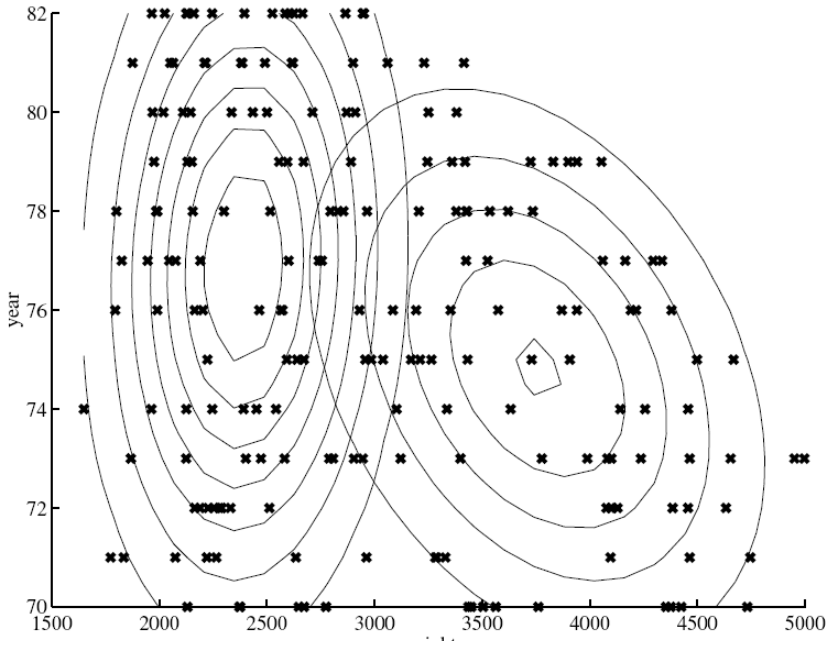

Figura 26. Clusterização pelo algoritmo Gath-Geva Original
FONTE: ABONYI, J.; BABUSKA, R.; SZEIFERT, F. (2002)

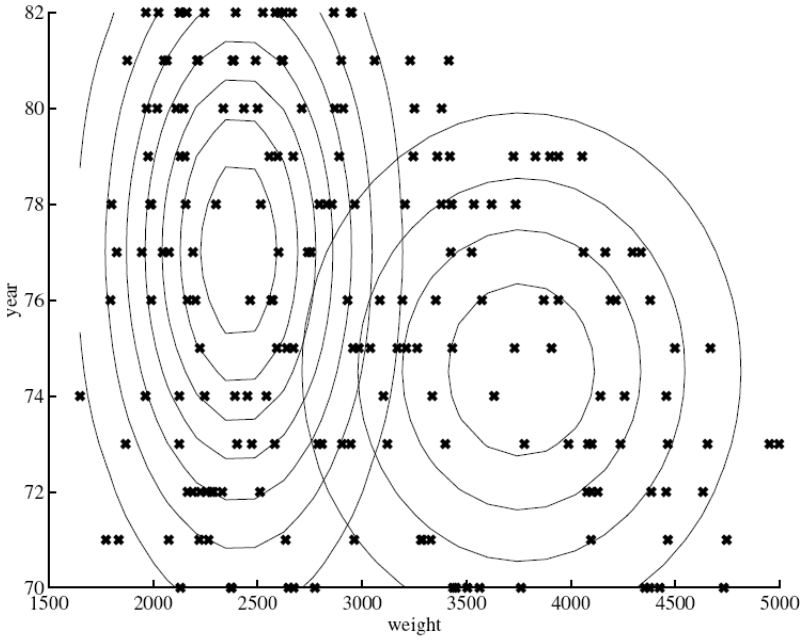

Figura 27. Clusterização pelo algoritmo Gath-Geva Modificado
FONTE: ABONYI, J.; BABUSKA, R.; SZEIFERT, F. (2002)

7.4 TOWARDS AN UNSUPERVISED OPTIMAL FUZZY CLUSTERING ALGORITHM FOR IMAGE DATABASE ORGANIZATION

Esse artigo foi desenvolvido pelos pesquisadores Xuejian Xiong e Kap Luk Chan, da Universidade Tecnológica Nanyang de Cingapura, sendo publicado na *IEEE – International Conference on Pattern Recognition* no ano de 2000.

Na pesquisa eles propõem a utilização do algoritmo Gath-Geva e seus métodos de validação de *cluster*s na classificação de imagens, isso devido à principal vantagem do algoritmo de poder trabalhar com *cluster*s de diferentes formas e tamanhos, sendo que a eficiência do processo pode ser avaliado pelas fórmulas de validação de *cluster*s. Com isso o algoritmo foi aplicado em diversas simulações de dados realizadas (XIONG, 2000, tradução nossa).

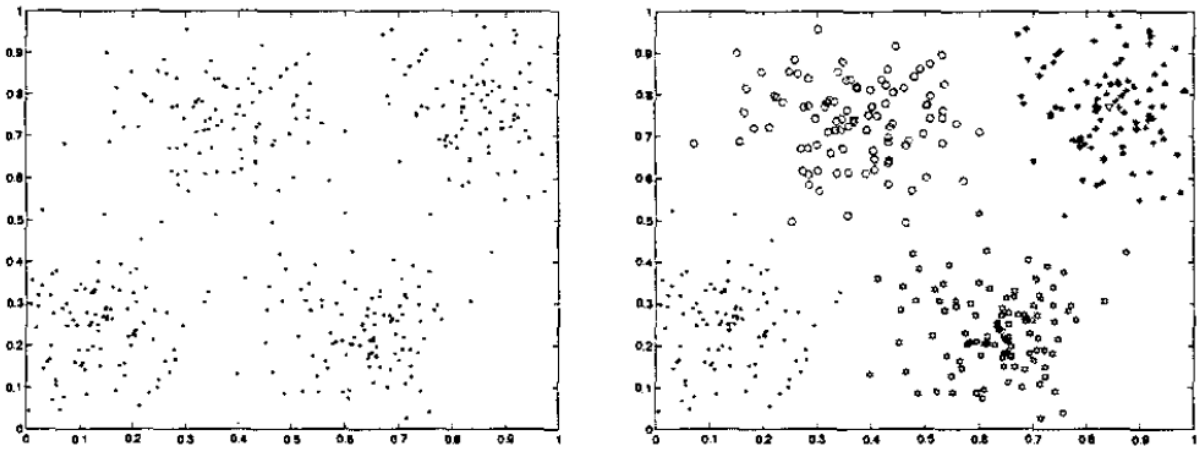

Figura 28. A esquerda dados aleatórios, a direita após a clusterização pelo GG
FONTE: XIONG, X; CHAN, K. (2000)

A Figura 28 apresenta o teste do algoritmo Gath-Geva, onde foi realizada a clusterização com base em dados simulados por um sistema. Pode ser observada a formação de quatro *cluster*s, sobre esses mesmos dados foi realizado uma validação dos *cluster*s, que é

visualizado na Figura 29, podendo-se verificar que o número ideal de grupos é quatro (XIONG, 2000, tradução nossa).

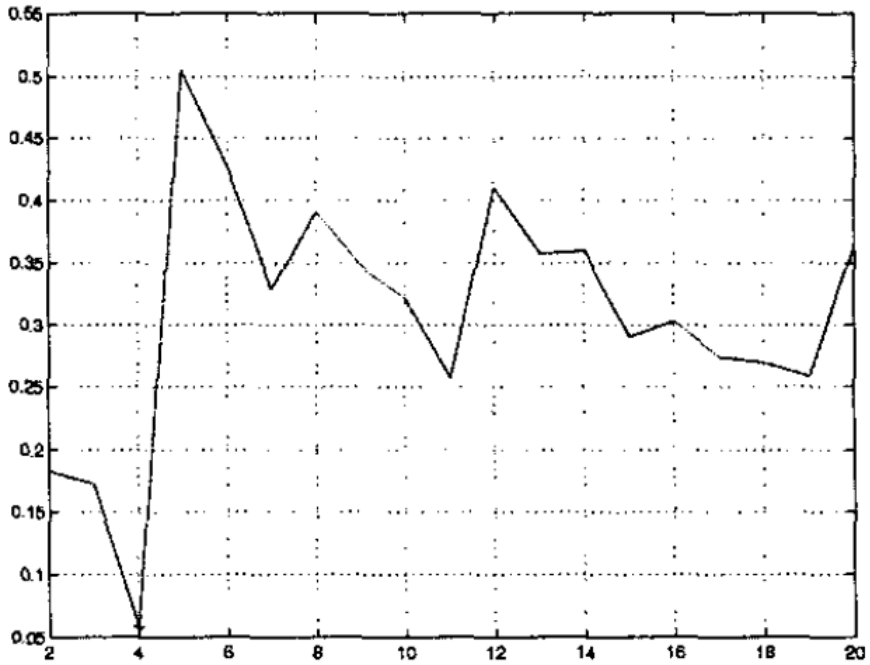

Figura 29. Processo de Validação de Clusters
FONTE: XIONG, X; CHAN, K. (2000)

Na pesquisa desenvolvida, analisando as resultados experimentais pode se observar que o algoritmo Gath-Geva, supera as deficiências do algoritmo FCM. Onde consome menos tempo para execução e melhor formação de *cluster*s. Sendo que não há necessidade de se conhecer a quantidade de grupos *a priori*, bastando somente utilizar fórmulas de validação, que comprovadamente coincidiram com a observação do especialista (XIONG, 2000, tradução nossa).

Após as diversas aplicações visualizadas do algoritmo Gath-Geva, na próxima seção será apresentado o trabalho desenvolvido.

8 O ALGORITMO GATH-GEVA NA TAREFA DE CLUSTERIZAÇÃO DA SHELL ORION DATA MINING ENGINE

O projeto da Shell Orion Data Mining Engine consiste no desenvolvimento de uma ferramenta gratuita que realiza a descoberta de conhecimento em diferentes tipos de base de dados. Além disso, disponibiliza-se para a comunidade acadêmica a modelagem matemática de diferentes algoritmos que poucas vezes são encontradas com facilidade em fontes bibliográficas.

A implementação do algoritmo Gath-Geva para a tarefa de clusterização por meio do método de lógica *fuzzy* na Shell Orion amplia as funcionalidades da ferramenta para a realização desta tarefa, possibilitando a aplicação de diferentes algoritmos para a descoberta de conhecimento em uma base de dados.

No desenvolvimento desta pesquisa seguiram-se algumas etapas a fim de se atingir os objetivos de entender o algoritmo Gath-Geva, desenvolver a sua modelagem matemática, implementá-lo na Shell Orion e aplicá-lo em uma base de dados para realização de testes.

8.1 BASES DE DADOS

A base de dados utilizada para demonstrar e comprovar o funcionamento do algoritmo Gath & Geva na *Shell* Orion *Data Mining* Engine, tem que atender a requisitos mínimos de qualidade de dados para aplicar a tarefa de clusterização, pois é importante destacar que a ferramenta ainda não possui implementado o módulo de pré-processamento, onde preparam-se os dados para serem utilizados pelas tarefas de *data mining.*

A base utilizada para aplicar o algoritmo Gath-Geva, é composta de 150 registros, contendo informações referentes a três tipos de plantas da família das Iridáceas: setosa, versicolor e virgínica. Sendo que também foi utilizada para apresentar a implementação do algoritmo CART para a tarefa de classificação implementada pela Bacharel em Ciência de Computação Lidiane Rosso Raimundo no ano de 2005.

Esta base está disponível gratuitamente na ferramenta WEKA[25] 3.5.5, sendo que esses registros estão distribuídos entre os três tipos de plantas. Os atributos numéricos correspondem as medidas das pétalas e sépalas da planta, sendo que o algoritmo GG somente trabalha com atributos numéricos. Essas quatro características são:

a) *sepal_length* (Comprimento da sépala)

b) *sepal_width* (Largura da sépala)

c) *petal_length* (Comprimento da sépala)

d) *petal_width* (Largura da sépala)

Íris Setosa | Íris Versicolor | Íris Virgínica

Figura 30. Imagem das iridáceas
FONTE: RAIMUNDO, L (2005)

[25] A Weka tem como objetivo aplicar e implementar algoritmos de diversas áreas da Inteligência Artificial e está disponível para download por meio do site: http://www.cs.waikato.ac.nz/ml/weka/

8.2 METODOLOGIA

O desenvolvimento do algoritmo Gath & Geva no projeto da *Shell* Orion Data Mining Engine teve que seguir algumas etapas, como: levantamento bibliográfico, modelagem do módulo da *Shell*, demonstração da modelagem matemática, implementação e testes do algoritmo.

8.2.1 Levantamento Bibliográfico

Envolveu a pesquisa bibliográfica dos temas desenvolvidos ao longo pesquisa, compreendendo o entendimento e descrição do processo de *data mining*, funcionamento da Shell Orion *Data Mining* Engine, tarefa de clusterização, lógica *fuzzy* e o funcionamento do algoritmo Gath-Geva.

O levantamento bibliográfico consistiu na base para realização das próximas etapas, sendo importante destacar a dificuldade existente para entender e descrever o algoritmo Gath-Geva, devido à escassez de bibliografias.

8.2.2 Modelagem do Módulo de Clusterização com o algoritmo Gath & Geva

Essa é uma etapa fundamental para o desenvolvimento do módulo de clusterização pelo algoritmo Gath-Geva, realizando-se a modelagem pela *Unified Modeling Language*[26] (UML). A realização da modelagem auxilia no entendimento e posterior

[26] É uma linguagem para realização de especificação, documentação e desenvolvimento de sistemas computacionais (RUMBAUGH;JACOBSON, 2000).

desenvolvimento do algoritmo, utilizando-se ferramenta computacional JUDE[27] para auxiliar na criação e visualização dos diagramas.

A modelagem consistiu no desenvolvimento dos diagramas de caso de uso, atividades e seqüência.

O diagrama de caso de uso (*use case*) é utilizado para representar a unidade funcional provida pelo sistema. Apresenta atores que podem ser o próprio usuário ou a entidade máquina e descreve a forma como esses interagem com o sistema.

Na Figura 31 visualiza-se o diagrama de caso de uso desenvolvido, composto por:

a) **inserir os parâmetros de entrada do algoritmo:** o usuário insere no sistema os parâmetros necessários para execução do algoritmo: quantidade de *cluster*s, o valor de *fuzzificação*, o número máximo de iterações e o valor da taxa de erro. Além disso, é necessário selecionar os atributos da tabela que serão utilizados no processo de clusterização e realizar a solicitação ao sistema para iniciar a execução do algoritmo;

b) **processar o algoritmo Gath-Geva:** após receber a solicitação do usuário, o sistema executa o algoritmo Gath-Geva e gera os resultados.

[27] Disponível para *download* no site http://jude.change-vision.com/jude-web/download/index.html.

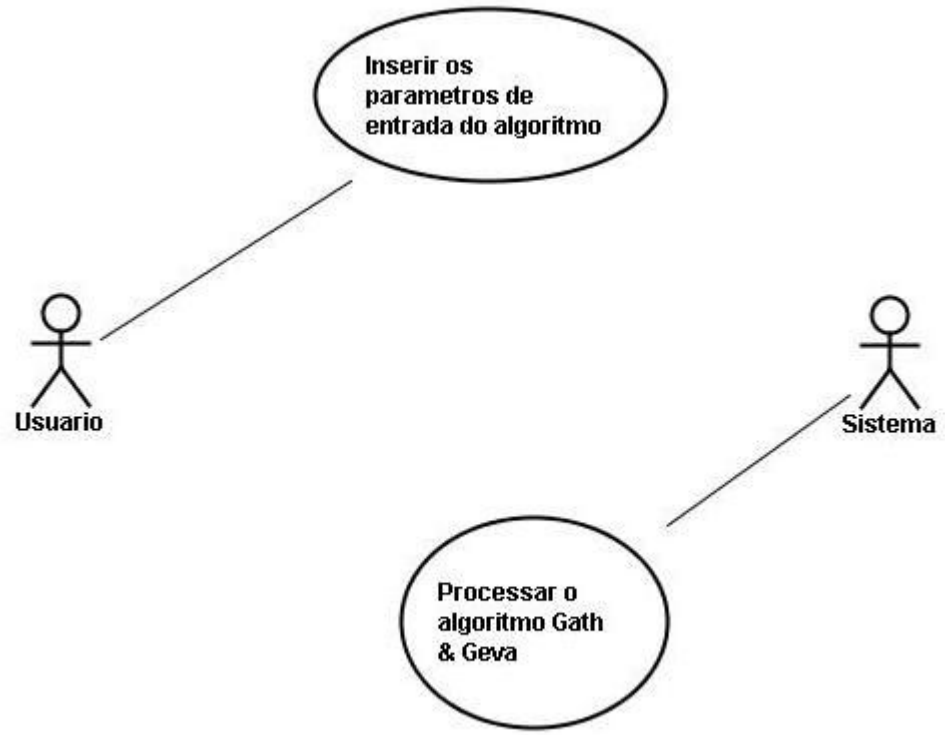

Figura 31. Diagrama de caso de uso

O diagrama de atividades representa os fluxos que são gerados no processamento, apresentando-se na forma de um gráfico de fluxo que mostra o controle de uma atividade para a seguinte.

Analisando-se os fluxos e processos do módulo de clusterização da *Shell* Orion, construiu-se o diagrama de atividades (Figura 32).

a) **informa os parâmetros de entrada:** todos os parâmetros necessários para possibilitar o funcionamento do algoritmo são informados pelo usuário;
b) **solicita a execução do algoritmo:** preenchidos os parâmetros, o usuário solicita ao sistema a execução do algoritmo Gath-Geva;
c) **processa o algoritmo Gath-Geva:** como pode-se notar no diagrama essa é uma atividade que corresponde ao sistema, onde acontece o processamento dos dados, ou seja, execução do algoritmo;
d) **visualização dos resultados:** finalizado o processamento do algoritmo Gath-Geva, os resultados são apresentados ao usuário.

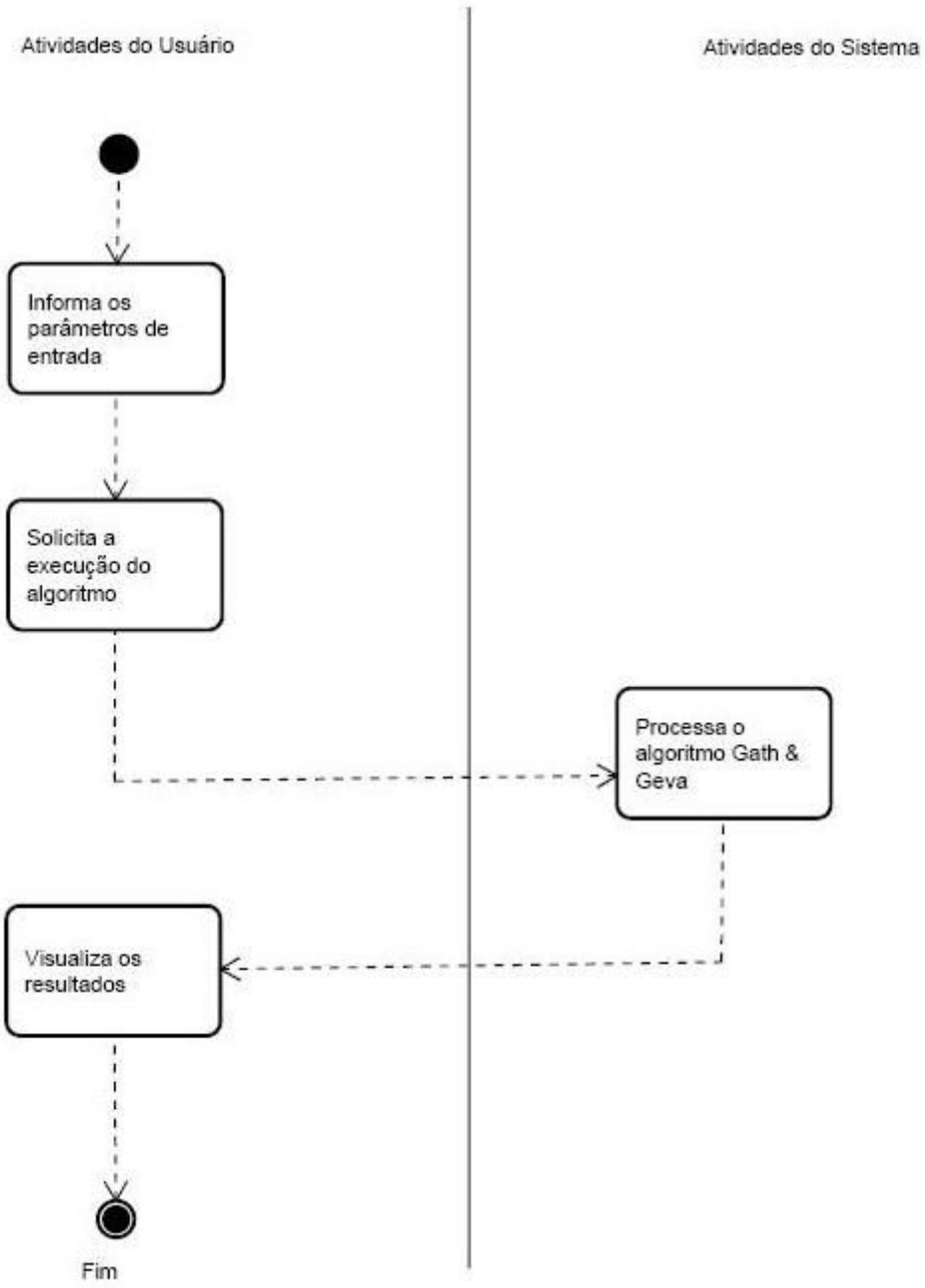

Figura 32. Diagrama de Atividades

O diagrama de seqüência é usado para representar a seqüência de processos (mensagens) num sistema de computador, nesse caso, a *Shell* Orion Data Mining Engine. Esse tipo de diagrama é interessante, pois em projetos com grande número de métodos e classes diferentes, torna-se mais simples e lógica a sua representação.

Na Figura 33 é possível verificar o resultado da modelagem do diagrama de seqüência do módulo de clusterização pelo algoritmo Gath-Geva. O usuário solicita o método para abrir a interface do algoritmo Gath-Geva, e após alimentar com os parâmetros necessários, envia o comando para executar o algoritmo, conseqüentemente geram-se os resultados para o usuário.

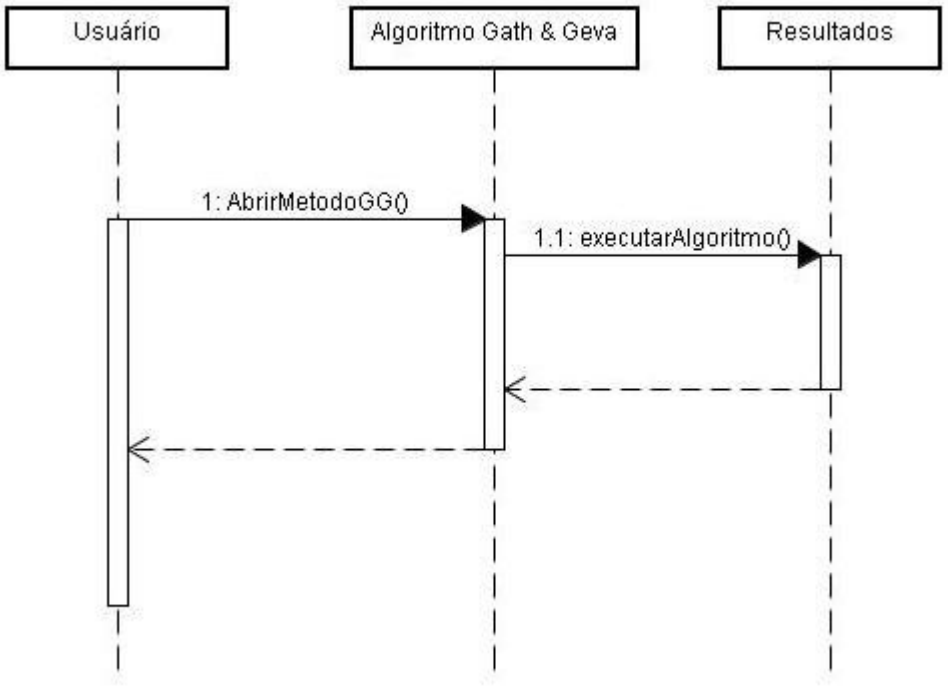

Figura 33. Diagrama de Seqüência

8.2.3 Demonstração da Modelagem Matemática do Algoritmo Gath-Geva

No decorrer do desenvolvimento do trabalho percebeu-se a dificuldade no entendimento e implementação do algoritmo GG, então para melhor compreensão de todos, será demonstrado passo a passo, os cálculos para a sua implementação em uma pequena base de dados.

Os cálculos são baseados nos conceitos e fórmulas matemáticas apresentadas por Gath & Geva em 1989, mas com algumas adaptações, a fim de conseguir uma maior

otimização dos resultados obtidos pelo algoritmo. A adaptação foi proposta por Balazs Balasko, Janos Abonyi and Balazs Feil, e referem-se ao cálculo da matriz de covariância.

Nesta demonstração matemática utilizou-se uma pequena base de dados para testes, onde os valores foram preenchidos de forma aleatória, sendo compostos por quatro objetos (z_1, z_2, z_3, z_4) constituídos de três atributos (x,y,w) cada um.

Tabela 3. Base de dados

Atributos	z_1	z_2	z_3	z_4
x	3,2	14,8	30,2	44,4
y	-2,7	-2,7	36,2	0
w	3	1	4	2

É necessário definir além da base de dados, os parâmetros para o funcionamento do algoritmo. Os parâmetros a serem definidos são:

a) **quantidade de *cluster*s:** compreende o número de agrupamentos a serem realizados nos dados de entrada. Esse é um valor definido pelo usuário, sendo que o valor utilizado foi igual a 2;

b) **termo de *fuzzificação:*** é o grau de *fuzzificação* dos elementos pertencentes a base de dados. Foi determinado o valor 2.

c) **taxa de erro:** é o que torna possível a parada do algoritmo. Assim, enquanto a diferença entre os valores de pertinências da iteração anterior e a atual não inferior a *taxa de erro*, o algoritmo continuará realizando os cálculos. Esse parâmetro foi definido como 0,00001.

Inicialmente deve-se preencher as matrizes de pertinências. Esta inicialização foi realizada de forma randômica, porém também pode-se utilizar outros algoritmos mais simples como o *Fuzzy C-Means*, a fim de se obter valores que aperfeiçoem o funcionamento do algoritmo. Os valores devem obedecer a propriedade (10), onde a soma das pertinências em relação a todos os *cluster*s deve ser igual a 1.

$$U \in \mathbb{R}^{N\,x\,K} com\ \mu_{j,i} \in [0,1], \forall i,k; \sum_{j=1}^{N} \mu_{j,i} = 1, \forall k; 0 < \sum_{i=1}^{K} \mu_{j,i} < K, \forall i \qquad (10)$$

Onde:

a) **K**: número total de *clusters*;

b) $\mu_{i,k}$: grau de pertinência do j-ésimo elemento em relação ao i-ésimo *cluster*;

c) **N**: número de elementos ou dados.

A Tabela 5 representa os valores aleatórios das pertinências dos objetos em relação aos dois *cluster*s.

Tabela 4. Matriz de Pertinência

Clusters	x1	x2	x3	x4
1	0,42	0,30	0,81	0,67
2	0,58	0,70	0,19	0,33

Após a definição dos parâmetros de entrada e inicialização das matrizes de pertinência, realizam-se os cálculos de: centros dos clusters, matriz de covariância, distância entre os elementos e os clusters, atualização da matriz de pertinência e cálculo da taxa de erro.

8.2.3.1 Cálculo dos centros dos *Cluster*s:

O cálculo dos centros dos *cluster*s consiste em realizar o somatório da multiplicação dos atributos dos objetos pela respectiva pertinência atual em relação ao *cluster*, elevando-se ao no grau de *fuzzificação* estabelecido nos parâmetros de entrada do algoritmo. Cujo resultante é dividido pelo somatório de todas as pertinências igualmente elevada ao grau de *fuzzificação*.

$$V_i = \frac{\sum_{j=1}^{N} (\mu_{ij})^q X_j}{\sum_{j=1}^{N} (\mu_{ij})^q} \qquad (11)$$

Onde:

a) $\mathbf{V_i}$: centro do i-ésimo *cluster*;

b) **N**: número total de elementos;

c) $\boldsymbol{\mu_{j,i}}$: grau de pertinência do j-ésimo elemento em relação ao i-ésimo *cluster*;

d) **q:** parâmetro de *fuzzificação*;

e) $\mathbf{X_j}$: é o elemento j-ésimo.

Substituindo os valores tem-se:

$$V_1 = \frac{(\mu_{11})^q . x_1 + (\mu_{12})^q . x_2 + (\mu_{13})^q . x_3 + (\mu_{14})^q . x_4}{(\mu_{11})^q + (\mu_{12})^q + (\mu_{13})^q + (\mu_{14})^q}$$

$$V_1 = \frac{(0{,}42)^2 \cdot \begin{bmatrix} 3{,}2 \\ -2{,}7 \\ 3{,}0 \end{bmatrix} + (0{,}30)^2 \cdot \begin{bmatrix} 14{,}8 \\ -2{,}7 \\ 1{,}0 \end{bmatrix} + (0{,}81)^2 \cdot \begin{bmatrix} 30{,}2 \\ 36{,}2 \\ 2{,}62 \end{bmatrix} + (0{,}67)^2 \cdot \begin{bmatrix} 44{,}4 \\ 0{,}0 \\ 2{,}0 \end{bmatrix}}{(0{,}42)^2 + (0{,}30)^2 + (0{,}81)^2 + (0{,}67)^2}$$

Primeiramente, elevam-se os graus de pertinência ao respectivo grau de *fuzzificação*, resultando em:

$$V_1 = \frac{0{,}18 \cdot \begin{bmatrix} 3{,}2 \\ -2{,}7 \\ 3{,}0 \end{bmatrix} + 0{,}09 \cdot \begin{bmatrix} 14{,}8 \\ -2{,}7 \\ 1{,}0 \end{bmatrix} + 0{,}66 \cdot \begin{bmatrix} 30{,}2 \\ 36{,}2 \\ 4{,}0 \end{bmatrix} + 0{,}45 \cdot \begin{bmatrix} 44{,}4 \\ 0{,}0 \\ 2{,}0 \end{bmatrix}}{0{,}18 + 0{,}09 + 0{,}66 + 0{,}45}$$

A multiplicação do valor obtido pela matriz deve envolver todos os atributos do objeto.

$$V_1 = \frac{\begin{bmatrix} 0{,}56 \\ -0{,}48 \\ 0{,}53 \end{bmatrix} + \begin{bmatrix} 1{,}33 \\ -0{,}24 \\ 0{,}09 \end{bmatrix} + \begin{bmatrix} 19{,}81 \\ 23{,}75 \\ 2{,}62 \end{bmatrix} + \begin{bmatrix} 19{,}93 \\ 0{,}00 \\ 0{,}90 \end{bmatrix}}{1{,}37}$$

Posteriormente, realiza-se o somatório de todos os resultados obtidos.

$$V_1 = \frac{\begin{bmatrix} 41{,}64 \\ 23{,}03 \\ 4{,}14 \end{bmatrix}}{1{,}37}$$

Agora cada atributo da matriz deve ser dividido pelo somatório das pertinências que foram elevados ao grau de *fuzzificação*, resultando em 1,37. Ao final desse processo obtêm-se o centro do primeiro *cluster*.

$$V_1 = \begin{bmatrix} 30{,}36 \\ 16{,}79 \\ 3{,}02 \end{bmatrix}$$

A seguir, repete-se o mesmo procedimento para identificar o centro do segundo *cluster* que em:

$$V_2 = \begin{bmatrix} 14{,}67 \\ -0{,}95 \\ 1{,}92 \end{bmatrix}$$

8.2.3.2 Matriz de Covariância *Fuzzy* para cada *Cluster*.

A matriz de covariância, segundo Gath e Geva (1989), é calculada pelo somatório dos graus de pertinências elevados ao parâmetro de *fuzzificação*. Sendo que esse valor é multiplicado pela subtração entre os dados e os *clusters*. A próxima multiplicação envolve a mesma subtração entre os dados e os *clusters*, porém na forma transposta.

$$F_i = \frac{\sum_{j=1}^{N} \mu_{ji}{}^{q} \left(X_j - V_i\right)^T \left(X_j - V_i\right)}{\sum_{j=1}^{N} \mu_{ji}{}^{q}} \qquad (12)$$

Onde:

a) $\mathbf{F_i}$: matriz de covariância i-ésimo elemento;

b) **N**: número de elementos;

c) $\boldsymbol{\mu_{j,i}}$: grau de pertinência do j-ésimo elemento em relação ao i-ésimo *cluster*;

d) $\mathbf{X_j}$: é o elemento j-ésimo;

e) $\mathbf{V_i}$ = centro do i-ésimo *cluster*;

f) **q**: parâmetro de *fuzzificação* (no algoritmo GG original esse valor é igual a 1).

No entanto na ferramenta *Clustering Toolbox*[28] desenvolvida pelos pesquisadores Balazs Balasko, Janos Abonyi e Balazs Feil, no ano 2005, se propôs uma adaptação no cálculo da matriz de covariância, onde o valor da pertinência é elevado ao grau de *fuzzificação*. Isso foi realizado a fim de se compensar o termo exponencial utilizado no cálculo da distância. Além disso, também foi uma forma encontrada para generalizar a matriz de covariância calculada pelo algoritmo Gustafson-Kessel em relação ao algoritmo Gath-Geva.

Considerando os valores do primeiro *cluster* e substituindo os valores na fórmula 12, tem-se:

$$F_1 = \frac{\begin{array}{c}(\mu_{11})^q.(x_1 - V_1).(x_1 - V_1)^T + \\ (\mu_{12})^q.(x_2 - V_1).(x_2 - V_1)^T + \\ (\mu_{13})^q.(x_3 - V_1).(x_3 - V_1)^T + \\ (\mu_{14})^q.(x_4 - V_1).(x_4 - V_1)^T\end{array}}{(\mu_{11})^q + (\mu_{12})^q + (\mu_{13})^q + (\mu_{14})^q}$$

$$F_1 = \frac{\begin{array}{c}(0{,}42)^2.\left(\begin{bmatrix}3{,}2\\-2{,}7\\3{,}0\end{bmatrix} - \begin{bmatrix}30{,}36\\16{,}79\\3{,}02\end{bmatrix}\right).([3.2 - 2.7\ \ 3.0]) - ([30.36\ \ 16.79\ \ 3.02]) + \\ (0{,}30)^2.\left(\begin{bmatrix}14{,}8\\-2{,}7\\1{,}0\end{bmatrix} - \begin{bmatrix}30{,}36\\16{,}79\\3{,}02\end{bmatrix}\right).([14{,}8\ \ -2.7\ \ 1{,}0]) - ([30.36\ \ 16.79\ \ 3.02]) + \\ (0{,}81)^2.\left(\begin{bmatrix}30{,}2\\36{,}2\\4{,}0\end{bmatrix} - \begin{bmatrix}30{,}36\\16{,}79\\3{,}02\end{bmatrix}\right).([30.2\ \ 36.2\ \ 4.0]) - ([30.36\ \ 16.79\ \ 3.02]) + \\ (0{,}67)^2.\left(\begin{bmatrix}44{,}4\\0{,}0\\2{,}0\end{bmatrix} - \begin{bmatrix}30{,}36\\16{,}79\\3{,}02\end{bmatrix}\right).([44.4\ \ 0.0\ \ 2.0]) - ([30.36\ \ 16.79\ \ 3.02])\end{array}}{(0{,}42)^2 + (0{,}30)^2 + (0{,}81)^2 + (0{,}67)^2}$$

Deve-se então realizar a subtração entre o objeto e o respectivo centro do *cluster*, levando em consideração a suas posições, vertical e horizontal.

[28] Está disponível no site http://www.mathworks.com/matlabcentral/fileexchange/7486.

$$F_1 = \frac{\begin{array}{l}(0{,}42)^2.\left(\begin{bmatrix}-27{,}16\\-19{,}49\\-0{,}02\end{bmatrix}\right).([-27.16\ \ -19.49\ \ -0.02]) + \\ (0{,}30)^2.\left(\begin{bmatrix}-15{,}56\\-19{,}49\\-2{,}02\end{bmatrix}\right).([-15.56\ \ -19.49\ \ -2.02]) + \\ (0{,}81)^2.\left(\begin{bmatrix}-0{,}16\\-19{,}41\\-0{,}98\end{bmatrix}\right).([-0.16\ \ -19.41\ \ -0.98]) + \\ (0{,}67)^2.\left(\begin{bmatrix}-14{,}04\\-16{,}79\\-1{,}02\end{bmatrix}\right).([-14.04-16.79\ \ -1.02]))\end{array}}{(0{,}42)^2+(0{,}30)^2+(0{,}81)^2+(0{,}67)^2}$$

Elevando os graus de pertinência ao parâmetro de fuzzificação, tem-se:

$$F_1 = \frac{\begin{array}{l}0{,}18\,.\left(\begin{bmatrix}-27{,}16\\-19{,}49\\-0{,}02\end{bmatrix}\right).([-27.16\ \ -19.49\ \ -0.02]) + \\ 0{,}09\,.\left(\begin{bmatrix}-15{,}56\\-19{,}49\\-2{,}02\end{bmatrix}\right).([-15.56\ \ -19.49\ \ -2.02]) + \\ 0{,}66\,.\left(\begin{bmatrix}-0{,}16\\-19{,}41\\-0{,}98\end{bmatrix}\right).([-0.16\ \ -19.41\ \ -0.98]) + \\ 0{,}45\,.\left(\begin{bmatrix}-14{,}04\\-16{,}79\\-1{,}02\end{bmatrix}\right).([-14.04-16.79\ \ -1.02]))\end{array}}{0{,}18+0{,}09+0{,}66+0{,}45}$$

O resultado obtido deve ser multiplicado pela matriz na sua forma vertical, visualizando-se os seguintes resultados:

$$F_1 = \frac{\begin{array}{l}\begin{bmatrix}-4{,}792\\-3{,}439\\-0{,}003\end{bmatrix}.([-27.16\ \ -19.49\ \ -0.02]) + \\ \begin{bmatrix}-2{,}746\\-3{,}439\\-0{,}356\end{bmatrix}.([-15.56\ \ -19.49\ \ -2.02]) + \\ \begin{bmatrix}-0{,}003\\-3{,}423\\-0{,}173\end{bmatrix}.([-0.16\ \ -19.41\ \ -0.98]) + \\ \begin{bmatrix}-2{,}476\\-2{,}962\\-0{,}180\end{bmatrix}.([-14.04-16.79\ \ -1.02]))\end{array}}{1{,}37}$$

Nesse momento tem-se uma multiplicação de matrizes, sendo uma 3x1 e a outra 1x3. Segundo os conceitos de matemática, para ser possível uma multiplicação, o número de colunas da matriz A deve corresponder ou ser igual ao número de linhas da matriz B. Dessa forma, a matriz que será gerada, corresponderá a quantidade de linhas da matriz A com o número de colunas da matriz B.

Fazendo-se os cálculos, chega-se aos seguintes resultados:

$$F_1 = \frac{\begin{bmatrix} 130{,}167 & 93{,}412 & 0{,}095 \\ 93{,}412 & 67{,}036 & 0{,}068 \\ 0{,}095 & -0{,}068 & 0{,}000 \end{bmatrix} + \begin{bmatrix} 21{,}803 & 27{,}307 & 2{,}829 \\ 27{,}307 & 34{,}202 & 3{,}543 \\ 2{,}829 & 3{,}5438 & 0{,}367 \end{bmatrix} + \begin{bmatrix} 0{,}017 & -2{,}094 & -0{,}105 \\ -2{,}094 & 247{,}078 & 12{,}479 \\ -0{,}106 & 12{,}479 & 0{,}630 \end{bmatrix} + \begin{bmatrix} 88{,}431 & -105{,}812 & -6{,}425 \\ -105{,}81 & -126{,}609 & 7{,}688 \\ -6{,}425 & -7{,}688 & 0{,}467 \end{bmatrix}}{1{,}37}$$

A resultante da soma de todas as matrizes é:

$$F_1 = \frac{\begin{bmatrix} 240{,}419 & 12{,}813 & -3{,}606 \\ 12{,}813 & 474{,}926 & 23{,}780 \\ -3{,}606 & 23{,}780 & 1{,}464 \end{bmatrix}}{1{,}37}$$

A matriz de covariância é obtida da divisão da matriz calculada pela soma das pertinências elevadas ao seu grau de *fuzzificação*, obtendo-se:

$$F_1 = \begin{bmatrix} 175{,}309 & 9{,}343 & -2{,}630 \\ 9{,}343 & 346{,}307 & 17{,}339 \\ -2{,}630 & 17{,}339 & 1{,}068 \end{bmatrix}$$

Aplicando as mesmas operações para o segundo *cluster*, tem-se:

$$F_2 = \begin{bmatrix} 153{,}619 & 31{,}443 & -2{,}883 \\ 31{,}443 & 53{,}995 & 3{,}037 \\ -2{,}883 & 3{,}037 & 0{,}992 \end{bmatrix}$$

8.2.3.3 Cálculo da Distância *Fuzzy* para Cada *Cluster*.

O cálculo da distância é bastante complexo, pois envolve um termo exponencial, que pode gerar alguns problemas com *overflow* ou até mesmo divisão por zero.

O desenvolvimento do cálculo da distância para melhor compreensão, foi dividido nas etapas: cálculo da determinante, cálculo da probabilidade, cálculo da matriz inversa, cálculo do termo exponencial e cálculo da distância propriamente dito.

$$d_e^2(X_j, V_i) = \frac{[\det(F_i)]^{1/2}}{P_i} exp\left[\frac{(X_j - V_i)^T F_i^{-1}(X_j, -V_i)}{2}\right] \quad (13)$$

Onde:

a) **d^2 (X_j,V_i)**: é a distância entre o j-ésimo elemento X_j e o i-ésimo centro V_i;

b) **X_j**: é o elemento j-ésimo;

c) **V_i** = centro do i-ésimo *cluster*;

d) **F_i**: matriz de covariância do i-ésimo *cluster*;

e) **P_i**: probabilidade *a priori* do i-ésimo *cluster*.

8.2.3.3.1 Calculo da Determinante da Matriz Covariância:

O cálculo da determinante de uma matriz 3x3 pode ser obtido por meio de:

$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \quad (14)$$

$$Det(A) = (a.e.i) + (b.f.g) + (c.d.h) - (c.e.g) - (b.d.i) - (a.f.h)$$

Utilizando os valores da matriz de covariância e aplicando a fórmula chega-se a expressão:

$$F_1 = \begin{bmatrix} 175{,}309 & 9{,}343 & -2{,}630 \\ 9{,}343 & 346{,}307 & 17{,}339 \\ -2{,}630 & 17{,}339 & 1{,}068 \end{bmatrix}$$

$$Det(F_1) = (175{,}309\ .\ 346{,}307\ .\ 1{,}067) + (9{,}343\ .\ 17{,}339\ .\ -2{,}630)\ +$$

$$(-2{,}630\ .\ 9{,}343\ .\ 17{,}339) — (2{,}630\ .\ 346{,}307\ .\ -2{,}630\ ) -$$

$$(9{,}343\ .\ 9{,}343\ .\ 1{,}067\ ) - (175{,}309\ .\ 17{,}339\ .\ 17{,}339) -$$

$$Det(F_1) = 8778{,}756$$

Realizando o mesmo procedimento para a segunda matriz de covariância. Tem-se o seguinte resultado:

$$Det(F_2) = 4833{,}076$$

8.3.3.3.2 Cálculo da probabilidade

Após encontrar o determinante, deve-se realizar o cálculo da probabilidade de um elemento pertencer ao *cluster*, sendo calculo por meio da Fórmula 15:

$$P_i = \frac{1}{N}\sum_{j=1}^{N} \mu_{ji} \tag{15}$$

Onde:

a) $\mathbf{P_i}$: probabilidade *a priori* do i-ésimo *cluster*;

b) **N**: número de elementos;

c) $\boldsymbol{\mu_{j,i}}$: grau de pertinência do j-ésimo elemento em relação ao i-ésimo *cluster*;

Substituindo os valores na fórmula tem-se a expressão:

$$P_1 = \frac{1}{4}(\mu_{11} + \mu_{12} + \mu_{13} + \mu_{14})$$

$$P_1 = \frac{1}{4}(0{,}42 + 0{,}30 + 0{,}81 + 0{,}67)$$

Inicialmente é realizada a soma de todas as pertinências. Além disso, deve-se dividir por 4, que nesse caso, representa a quantidade de elementos ou dados que fazem parte do processo de *clusterização*. A seguir verifica-se o resultado final:

$$P_1 = \frac{1{,}37}{4}$$

$$P_1 = 0{,}55$$

Realizando o mesmo procedimento para o segundo *cluster*, tem-se:

$$P_2 = 0{,}45$$

8.2.3.3.3 Cálculo da Matriz Inversa

Após se calcular a probabilidade, deve-se calcular a da matriz de covariância *fuzzy* inversa. A Fórmula 16 demonstra o cálculo:

$$A^{-1} = \frac{1}{|A|} adj(A) \tag{16}$$

Onde:

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

b) **Adj(A)**: adjunta da matriz A.

O cálculo da matriz inversa corresponde a divisão da matriz adjunta pela determinante da mesma. Sendo que a adjunta contém todas as determinantes possíveis de A, conforme visualiza-se na fórmula 17:

$$Adj(A) = \begin{bmatrix} \begin{vmatrix} e & f \\ h & i \end{vmatrix} & \begin{vmatrix} c & b \\ i & h \end{vmatrix} & \begin{vmatrix} b & c \\ e & f \end{vmatrix} \\ \begin{vmatrix} f & d \\ i & g \end{vmatrix} & \begin{vmatrix} a & c \\ g & i \end{vmatrix} & \begin{vmatrix} c & a \\ f & d \end{vmatrix} \\ \begin{vmatrix} d & e \\ g & h \end{vmatrix} & \begin{vmatrix} b & a \\ h & g \end{vmatrix} & \begin{vmatrix} a & b \\ d & e \end{vmatrix} \end{bmatrix} \tag{17}$$

Onde o cálculo da determinante 2x2 é descrita na Fórmula 18:

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

$$|B| = (a.d) - (b.c) \tag{18}$$

Substituindo os dados da matriz de covariância do primeiro *cluster*, tem-se:

$$Adj(A) = \begin{bmatrix} \begin{vmatrix} 346,307 & 17,339 \\ 17,339 & 1,067 \end{vmatrix} & \begin{vmatrix} -2,630 & 9,343 \\ 1,067 & 17,339 \end{vmatrix} & \begin{vmatrix} 9,343 & -2,630 \\ 346,307 & 17,339 \end{vmatrix} \\ \begin{vmatrix} 17,339 & 9,343 \\ 1,067 & -2,630 \end{vmatrix} & \begin{vmatrix} 175,309 & -2,630 \\ -2,630 & 1,067 \end{vmatrix} & \begin{vmatrix} -2,630 & 175,309 \\ 17,339 & 9,343 \end{vmatrix} \\ \begin{vmatrix} 9,343 & 343,307 \\ -2,630 & 17,339 \end{vmatrix} & \begin{vmatrix} 9,343 & 175,309 \\ 17,339 & -2,630 \end{vmatrix} & \begin{vmatrix} 175,309 & 9,343 \\ 9,343 & 346,307 \end{vmatrix} \end{bmatrix}$$

$$adj(A) = \begin{bmatrix} 69,132 & -55,582 & 1072,818 \\ -55,582 & 180,288 & -3064,428 \\ 1072,818 & -3064,428 & 60623,579 \end{bmatrix}$$

Tendo-se calculado a matriz adjunta, resta efetuar a divisão dela pelo determinante da matriz de covariância *fuzzy*.

$$F^{-1} = \frac{\begin{bmatrix} 69,132 & -55,582 & 1072,818 \\ -55,582 & 180,288 & -3064,428 \\ 1072,818 & -3064,428 & 60623,579 \end{bmatrix}}{8778,756}$$

Finalmente tem-se a matriz inversa:

$$F^{-1} = \begin{bmatrix} 0,007 & -0,006 & 0,122 \\ -0,006 & 0,020 & -0,349 \\ 0,122 & -0,349 & 6,905 \end{bmatrix}$$

8.2.3.3.4 Cálculo do Termo Exponencial

Após determinar a matriz inversa, pode-se calcular o termo exponencial representado na Fórmula 19.

$$exp = \frac{(X_j - V_i)^T F_i^{-1} (X_j - V_i)}{2} \quad (19)$$

Onde:

a) $\mathbf{X_j}$: é o elemento j-ésimo;

b) $\mathbf{V_i}$ = centro do i-ésimo *cluster*;

c) $\mathbf{F_i}$: matriz de covariância do i-ésimo *cluster*;

Substituindo-se os valores:

$$exp = \frac{\left[([3.2, -2.7, 3.0] - [30.36, 16.79, 3.02]).\begin{pmatrix} 0{,}007 & -0{,}006 & 0{,}122 \\ -0{,}006 & 0{,}020 & -0{,}349 \\ 0{,}122 & -0{,}349 & 6{,}905 \end{pmatrix}.\left(\begin{bmatrix} 3{,}2 \\ -2{,}7 \\ 3{,}0 \end{bmatrix} - \begin{bmatrix} 30{,}36 \\ 16{,}79 \\ 3{,}02 \end{bmatrix}\right)\right]}{2}$$

A seguir subtrai-se os valores que representam os vetores de registros dos centros dos *clusters*, tendo-se:

$$exp = \frac{\left[([-27.16, -19.49, -0.02]).\begin{pmatrix} 0{,}007 & -0{,}006 & 0{,}122 \\ -0{,}006 & 0{,}020 & -0{,}349 \\ 0{,}122 & -0{,}349 & 6{,}905 \end{pmatrix}.\left(\begin{bmatrix} -27{,}16 \\ -19{,}49 \\ -0{,}02 \end{bmatrix}\right)\right]}{2}$$

A matriz inversa tem ordem (3x3) e considerando o último vetor que tem ordem (1x3), obtêm-se com sua multiplicação um vetor vertical:

$$exp = \frac{\left[([-27.16, -19.49, -0.02]).\left(\begin{bmatrix} -0{,}092 \\ -0{,}221 \\ 3{,}348 \end{bmatrix}\right)\right]}{2}$$

Considerando os vetores horizontal de ordem (1x3) e o vertical como uma matriz de ordem (3x1), obtêm-se uma matriz de ordem (1x1). Dividindo esse valor por dois, tem-se:

$$exp = \frac{6{,}774}{2}$$

$$exp = 3{,}387$$

8.2.3.3.5 Cálculo da Distância Propriamente Dito

Calculada a matriz inversa, os termos de probabilidade e exponencial, tem-se que obter o valor da distância propriamente dito, conforme a Fórmula 20.

$$d^2(x_1, V_1) = \left(\frac{\sqrt[n]{\det(F)}}{P_1}\right)^{exp} \quad (20)$$

Onde:

a) $\mathbf{F_i}$ = matriz de covariância do i-ésimo *cluster*;

b) $\mathbf{P_i}$ = probabilidade *a priori* do i-ésimo *cluster*;

c) ***n*** = número de atributos.

Substituindo os valores já cálculos na expressão:

$$d^2(x_1, V_1) = \left(\frac{\sqrt[3]{8778,756}}{0,55}\right)^{3,387}$$

Nesse caso tem-se uma matriz cúbica, pois os dados são compostos por três atributos, onde realizando os cálculos, tem-se:

$$d^2(x_1, V_1) = \left(\frac{20,628}{0,55}\right)^{3,387}$$

Efetuando a divisão, encontra-se:

$$d^2(x_1, V_1) = (37,507)^{3,387}$$

Finalizando o cálculo da distância, tem-se que elevar o valor encontrado ao termo exponencial, obtendo-se:

$$d^2(x_1, V_1) = 214695,864$$

O cálculo da distância para os outros registros aplicando os mesmos procedimentos são visualizados a seguir:

$$d^2(x_2, V_1) = 1,606E + 11$$

$$d^2(x_3, V_1) = 1,637$$

$$d^2(x_4, V_1) = 41,440$$

$$d^2(x_1, V_2) = 30,646$$

$$d^2(x_2, V_2) = 5,937$$

$$d^2(x_3, V_2) = 2,518E + 20$$

$$d^2(x_4, V_2) = 1,849$$

8.2.3.4 Matriz de Pertinências dos Dados em Relação aos Novos Centros dos *Cluster*s.

Após calculados os valores das distâncias tem-se que atualizar a matriz de pertinências, aplicando a Fórmula 21.

$$\mu_{ij} = \frac{\frac{1}{d_{ij}^2}}{\sum_{k=1}^{V}\left(\frac{1}{d_{kj}^2}\right)} \tag{21}$$

Onde:

a) $\mu_{j,i}$: grau de pertinência do j-ésimo elemento em relação ao i-ésimo *cluster*;

b) **K:** número total de *clusters*;

c) $\mathbf{d_{ij}}$: é a distância entre o j-ésimo elemento X_j e o i-ésimo centro V_i;

d) $\mathbf{d_{kj}}$: é a distância entre o j-ésimo elemento X_j e o k-ésimo centro V_k.

Substituindo os valores referentes ao primeiro registro e ao primeiro *cluster*, obtêm-se o grau de pertinência:

$$\mu_{11} = \frac{\frac{1}{d_{11}^2}}{\frac{1}{d_{11}^2} + \frac{1}{d_{12}^2}}$$

$$\mu_{11} = \frac{\frac{1}{214695{,}864}}{\frac{1}{214695{,}864} + \frac{1}{1{,}606E+11}}$$

$$\mu_{11} = \frac{4{,}657E-06}{4{,}657E-06 + 0{,}032}$$

$$\mu_{11} = 0{,}000$$

Aplicando o mesmo procedimento para os outros dados e *cluster*s tem-se os resultados apresentados na Tabela 6.

Tabela 5. Matriz de pertinência atualizada

Clusters	**x1**	**x2**	**x3**	**x4**
1	0,00	0,00	1,00	0,04
2	1,00	1,00	0,00	0,96

8.2.3.5 Cálculo da Taxa de Erro

Atualizada a matriz de pertinência, deve-se calcular a taxa de erro, onde por meio desse valor permite-se a parada do algoritmo. É calculado pela seguinte equação:

$$\|U^l - U^{l-1}\| \leq \varepsilon \qquad (22)$$

Onde:

a) U = matriz de pertinência;

b) l = número da iteração;

c) **ε** = taxa de erro.

Substituindo a Fórmula 22 com os valores da matriz de pertinência da iteração anterior e atual, tem-se:

$$\left\| \begin{bmatrix} 0{,}00 & 0{,}00 & 1{,}00 & 0{,}04 \\ 1{,}00 & 1{,}00 & 0{,}00 & 0{,}96 \end{bmatrix} - \begin{bmatrix} 0{,}42 & 0{,}30 & 0{,}81 & 0{,}67 \\ 0{,}58 & 0{,}70 & 0{,}19 & 0{,}33 \end{bmatrix} \right\| \leq 0{,}0001$$

$$\left\| \begin{bmatrix} 0{,}42 & 0{,}30 & -0{,}19 & 0{,}63 \\ -0{,}42 & -0{,}30 & 0{,}19 & -0{,}63 \end{bmatrix} \right\| \leq 0{,}0001$$

Considerando somente os valores absolutos (positivos), tem-se o resultado final:

$$\begin{bmatrix} 0{,}42 & 0{,}30 & 0{,}19 & 0{,}63 \\ 0{,}42 & 0{,}30 & 0{,}19 & 0{,}63 \end{bmatrix} \leq 0{,}0001$$

Verificando os resultados, constata-se que nenhum dos números é menor que a taxa de erro, fazendo com que a condição de parada do algoritmo não seja verdadeira. Com isso, seria necessário executar mais uma iteração do algoritmo, até que todos os resultados da subtração sejam menores que a taxa de erro.

O entendimento de todos os cálculos realizados pelo algoritmo Gath-Geva possibilitou a sua implementação na *Shell* Orion Data Mining.

8.2.4 Implementação e Testes do Módulo de Clusterização por meio do Algoritmo Gath-Geva

A implementação do algoritmo Gath-Geva para a tarefa de clusterização na *Shell* Orion Data Mining Engine foi realizada por meio da linguagem orientada a objetos Java, utilizando-se o ambiente de desenvolvimento NetBeans, versão 6.5.1, obtido gratuitamente no *site http://www.netbeans.org*.

No desenvolvimento do algoritmo os conceitos, técnicas e modelagem matemática foram baseadas no artigo de Gath e Geva, 1989, que foram os criadores do algoritmo, sendo que algumas adaptações foram feitas conforme Balazs Balasko, Janos Abonyi and Balazs Feil publicaram na documentação da ferramenta *Fuzzy Clustering and Data Analysis Toolbox.*

Inicialmente para manipular os dados tem-se que implementar uma conexão, sendo que atualmente na Shell Orion, existe implementado a conexão com os bancos HSQLDB, PostGreSQL e MySQL. A fim de aumentar a versatilidade da ferramenta, possibilitando a conexão com diversos bancos, optou-se por implementar uma conexão com Firebird, cujo gerenciamento é bastante intuitivo, de fácil manipulação e de licença gratuita.

Após definir-se a conexão de dados, utilizou-se para testar as funcionalidades do algoritmo, a base de dados IRIS, cujos detalhes foram descritos na subseção 8.1 dessa pesquisa.

A Figura 34 apresenta a interface padrão para realizar a configuração de uma conexão com um banco de dados, deve-se preencher esses campos apenas quando se está incluindo uma nova conexão à Shell Orion Data Mining Engine.

Figura 34. Conexão com Firebird

Sendo que na Figura 35, são inseridas as informações necessárias para efetivar a conexão com o banco de dados. Após, deve-se clicar em *Conectar*, e então é exibida uma mensagem para o usuário se ocorreu ou não problema com a conexão. Estabelecida a conexão, escolhe-se a tabela da base de dados que será realizada a clusterização.

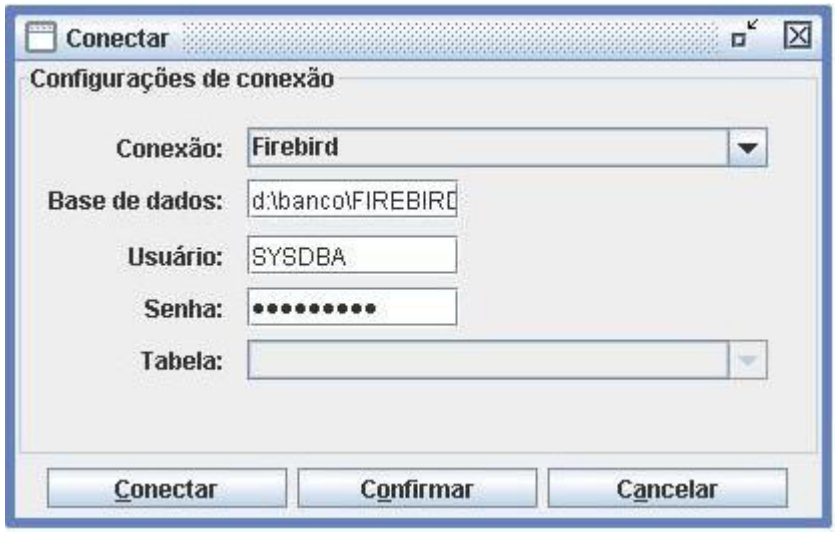

Figura 35. Informações para conectar ao banco

Após isso, pode-se acessar o algoritmo Gath-Geva, clicando-se no menu *Data Mining*, entre as tarefas disponíveis seleciona-se a opção *clusterização*, sub-menu *Fuzzy* e em algoritmo *Gath-Geva* (Figura 36).

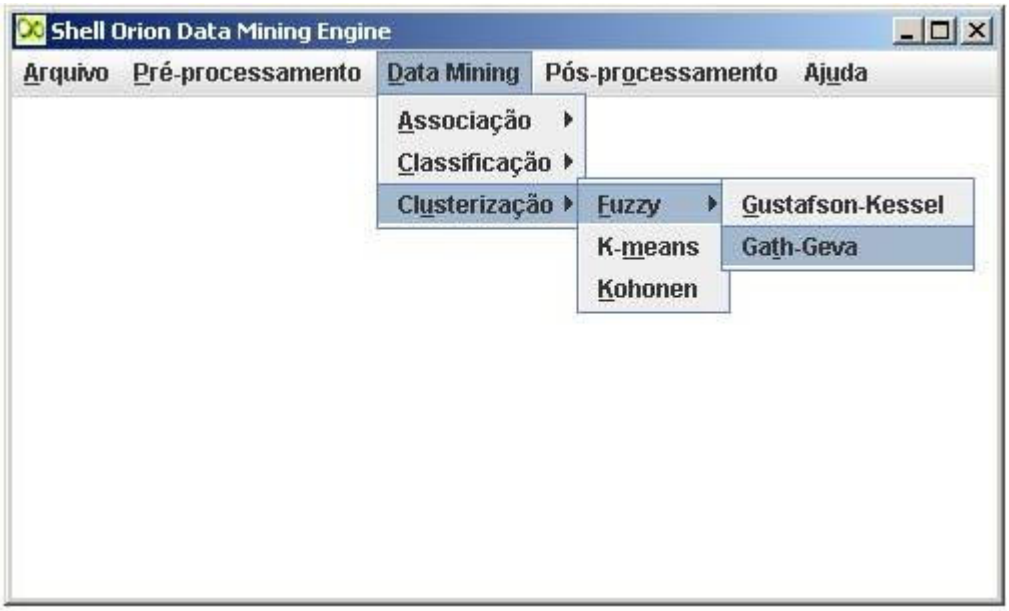

Figura 36. Selecionando o algoritmo Gath-Geva

A clusterização por meio do algoritmo Gath-Geva necessita de alguns parâmetros de inicialização para possibilitar o funcionamento.

a) **número de *clusters***: é a quantidade de grupos que devem ser formados na clusterização;

b) **parâmetro de *fuzzificação***: é o grau de fuzzificação dos dados em relação aos seus *clusters*, normalmente utiliza-se grau 2;

c) **quantidade de iterações**: a quantidade de iterações que o algoritmo irá executar, por padrão se for informado 0, será executado um número ilimitado de iterações e a condição de parada será o valor estabelecido pela taxa de erro;

d) **taxa de erro**: é o valor que determina a condição de parada do algoritmo, enquanto a taxa não for menor que esse valor as iterações irão continuar ocorrendo;

e) **selecionar atributos**: uma tabela pode conter vários atributos, com isso pode-se escolher entre os atributos numéricos disponíveis, quais serão clusterizados pelo algoritmo Gath-Geva.

A Figura 37 apresenta a interface gráfica com esses parâmetros devidamente preenchidos e os atributos escolhidos para o processo de clusterização. Clicando-se em *Executar*, iniciam-se as iterações para clusterizar os elementos encontrados na base de dados.

Figura 37. Inserindo os parâmetros de clusterização

Após o término das iterações, os resultados podem ser apresentados de diferentes maneiras. Inicialmente, a tela apresentada para o usuário é um resumo dos resultados encontrados, descrevendo-se os parâmetros e atributos de entrada utilizados, informações sobre os *clusters*, centros dos *clusters* atualizados e o atributo de saída, sendo que esse pode ser alterado pela opção *atributo de saída* (Figura 38).

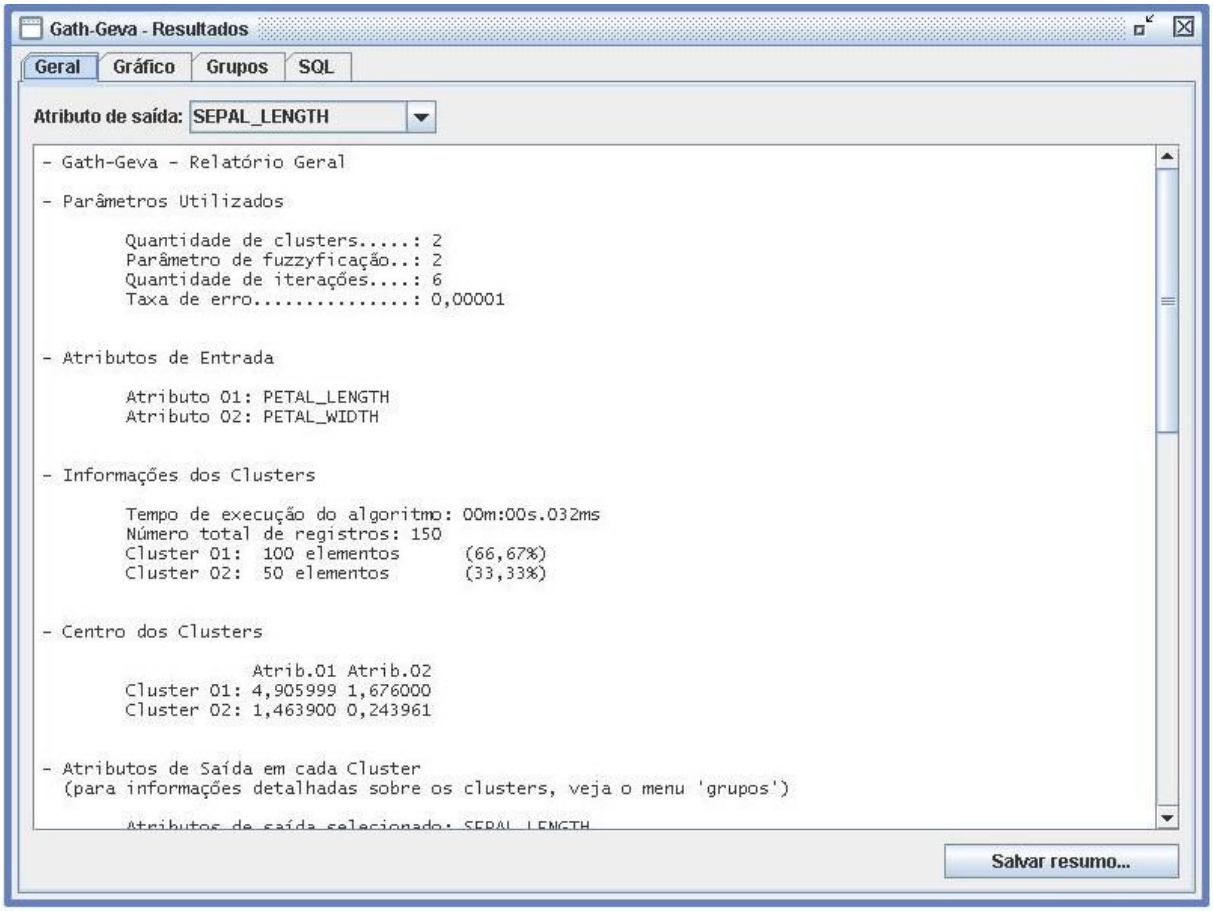

Figura 38. Resumo da clusterização com Gath-Geva.

Os resultados também podem ser apresentados em gráficos, sendo que essa opção apenas é habilitada quando se tem mais de um atributo na clusterização. A técnica adotada é o *Principal Components Analysis*[29] (PCA) que viabiliza a representação de resultados com *n*-dimensões em duas dimensões, aumentando a facilidade de compreender os dados, que é um dos objetivos do *data mining* (Figura 39).

[29] Tem disponível um manual com suas principais implementações em: http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf

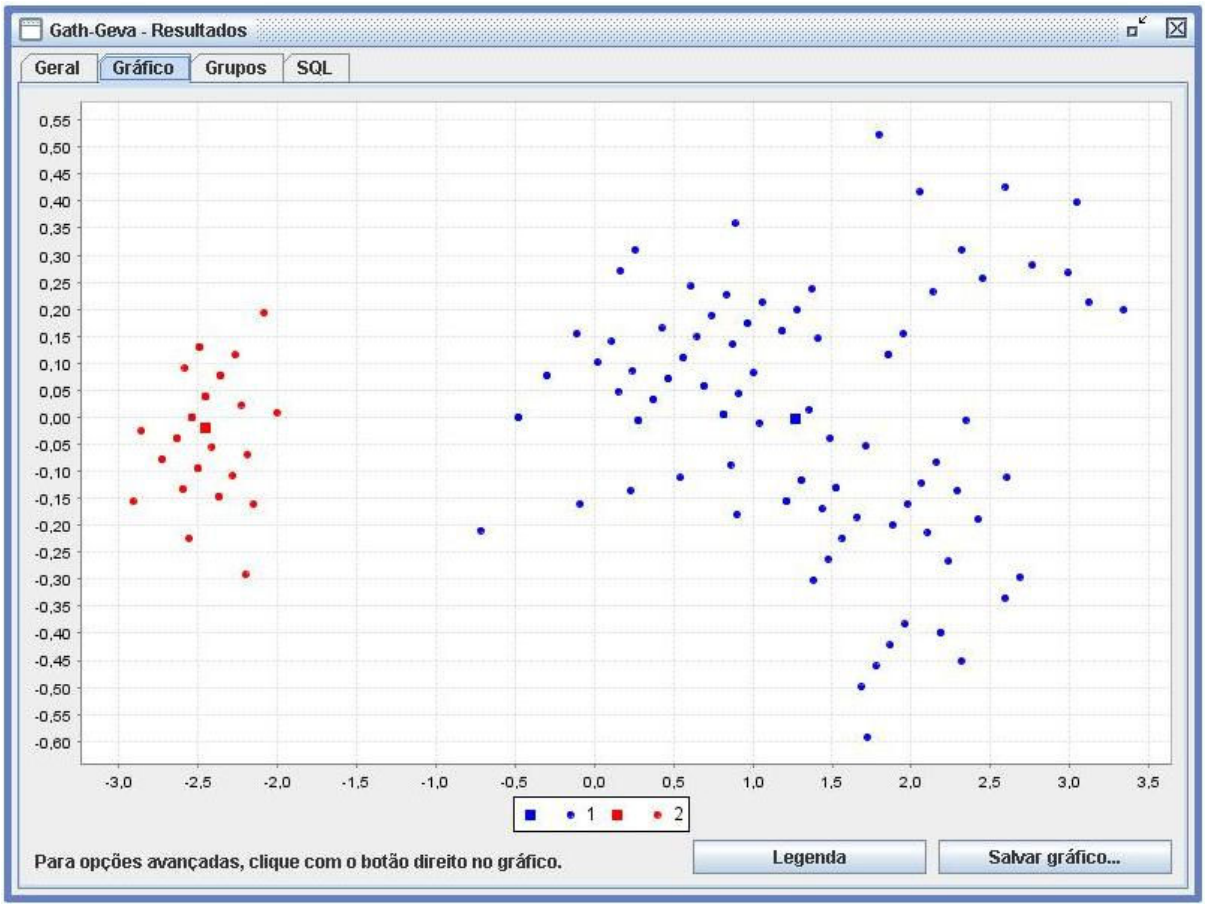

Figura 39. Representação gráfica dos resultados

Na Figura 40 tem-se mais uma alternativa de avaliar os resultados, por meio de estruturas de árvores, onde é possível realizar o estudo individual de cada elemento clusterizado. Ao abrir um *cluster* ou alguns dos dados, encontra-se o conteúdo calculado nas iterações do algoritmo. Além disso, selecionando a opção *Mostrar número do elemento*, visualiza-se o número do elemento em relação à base de dados e/ou acionando o item *Mostrar pertinências*, apresenta os graus de pertinências desses referidos dados em relação aos *clusters* encontrados.

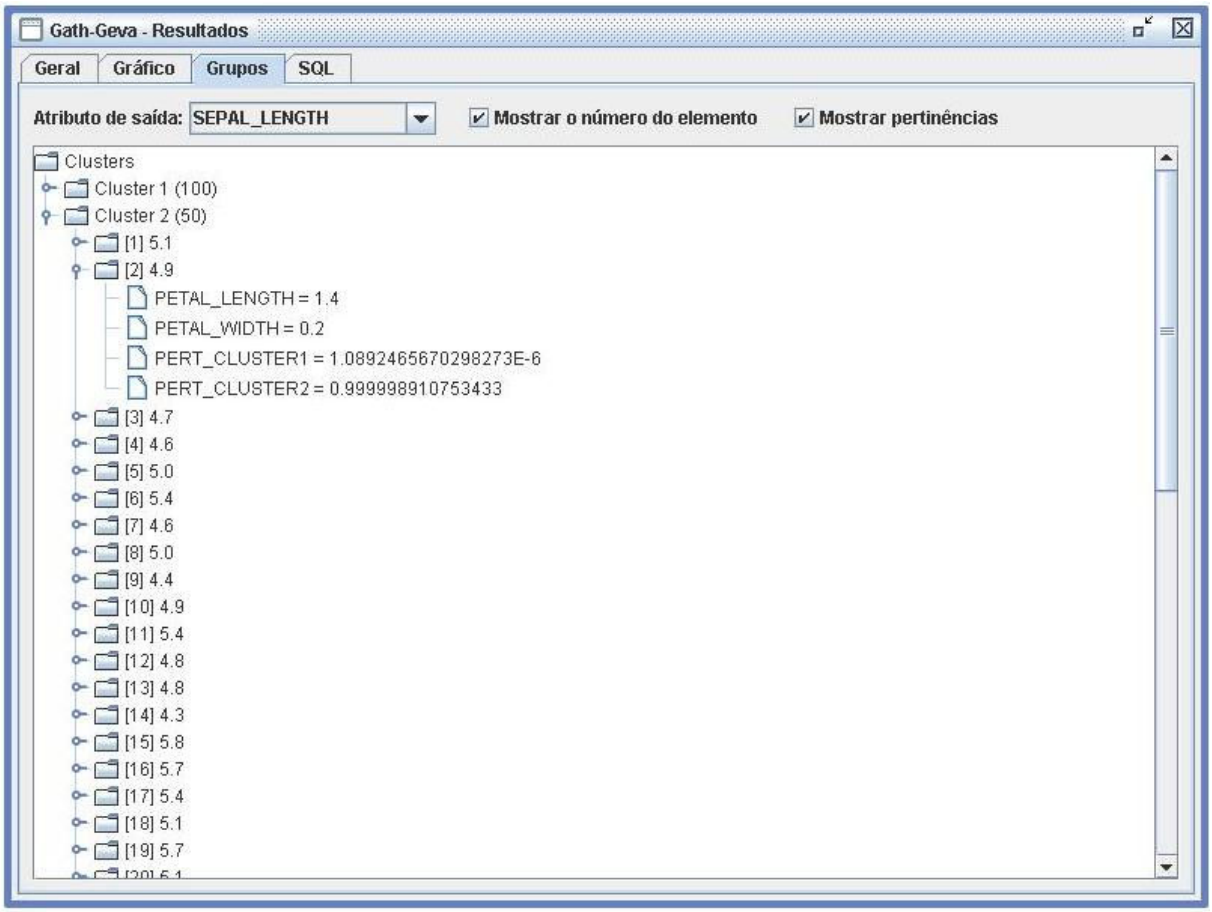

Figura 40. Representação em árvores dos resultados

Por fim os resultados podem ser exportados para um arquivo *Structured Query Language*[30] (SQL), isso para possibilitar a utilização dos resultados obtidos pela clusterização em outras tarefas de *data mining*. Isso é interessante, pois em alguns casos não é suficiente realizar apenas uma clusterização para obter conhecimentos sobre uma determinada base de dados, sendo necessária a aplicação de mais tarefas e algoritmos a fim de conseguir resultados satisfatórios (Figura 41).

[30] SQL é uma linguagem para consulta, modificar os dados e gestão de bases de dados (OLIVEIRA, 1999).

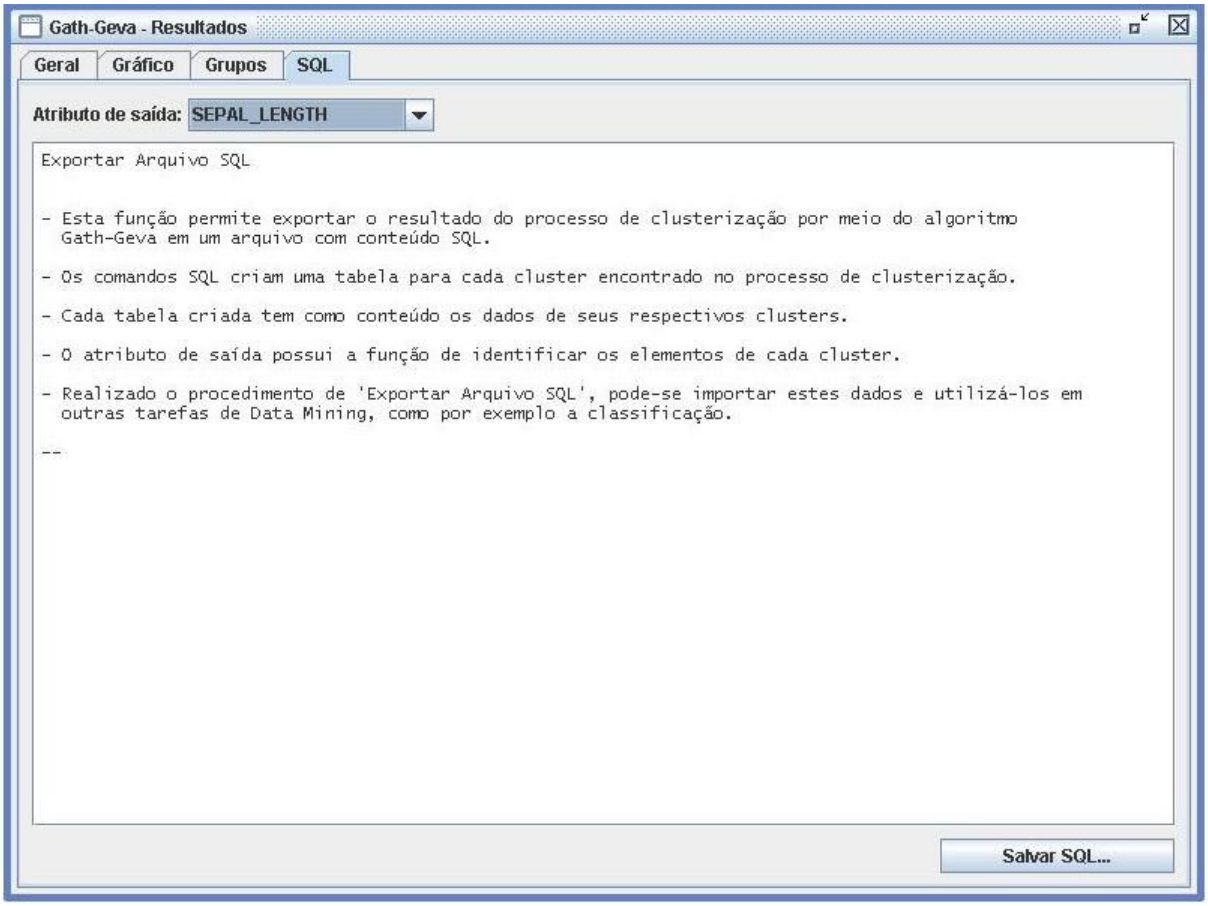

Figura 41. Interface para exportar dados para SQL

Considerando que o valor da determinante da matriz de covariância seja igual a zero, isso tem como conseqüência inviabilizar a determinação da matriz inversa. Então, na iteração em que ocorre esse problema, o usuário é informado por meio da tela mostrada na Figura 42. Sendo que tem-se a opção de refazer a clusterização ou avaliar os resultados até a iteração em que foi interrompida.

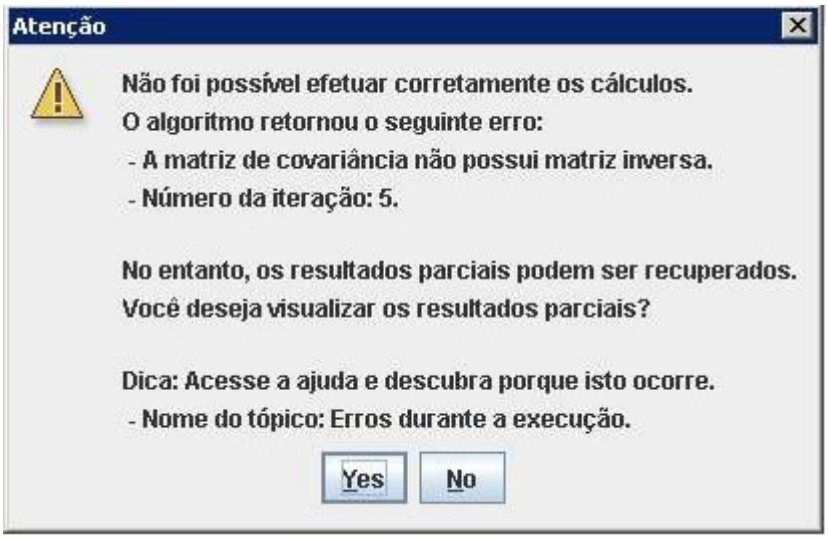

Figura 42. Erro durante execução do programa

8.3 RESULTADOS OBTIDOS

Concluindo-se a implementação do algoritmo no módulo de clusterização da *Shell* Orion *Data Mining*, o mesmo foi submetido a uma série de testes, com intuito de avaliar o seu desempenho e suas funcionalidades.

Na realização dos testes com o algoritmo Gath-Geva, utilizou-se um microcomputador com sistema Operacional Windows XP *Service Pack* 3, processador Core 2 Duo 2.26 GHz e 2Gb de memória RAM.

Os primeiros testes foram realizados a fim de verificar os *clusters* gerados pelo algoritmo GG utilizando a base de dados IRIS.

8.3.1 *Clusters* gerados pelo Algoritmo Gath-Geva

A base de dados utilizada foi a IRIS, onde a origem dos dados foi apresentada na subseção 8.1 dessa pesquisa. Sendo que o objetivo foi verificar como é possível utilizar a

ferramenta para a tarefa de clusterização de dados, exemplificando também a visualização dos resultados e como interpretá-los.

Essa base de dados possui poucos atributos e todos são numéricos, porém caso existissem características que não fossem numéricas, essas teriam que ser transformadas em valores numéricos, para serem interpretados pelo algoritmo. Além disso, observa-se que os atributos utilizados não contêm valores nulos.

Primeiramente foram determinados os parâmetros de início do algoritmo:

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

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

c) **quantidade de interações:** O (ilimitado);

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

e) **atributos de entrada:** *sepal_lenght*, *sepal_width, petal_length, petal_width*.

Avaliando a Figura 43, observa-se a separação em dois *clusters*, sendo que os tamanhos desses *clusters* são diferentes, isso se torna possível devido à utilização da matriz de covariância no algoritmo Gath-Geva.

A base de dados contém em seus registros informação referentes a três tipos plantas, porém efetuando a clusterização pelo algoritmo GG, utilizou-se como parâmetro de entrada o número de *clusters* igual a dois, com isso durante a execução, os dados são separados em dois grupos.

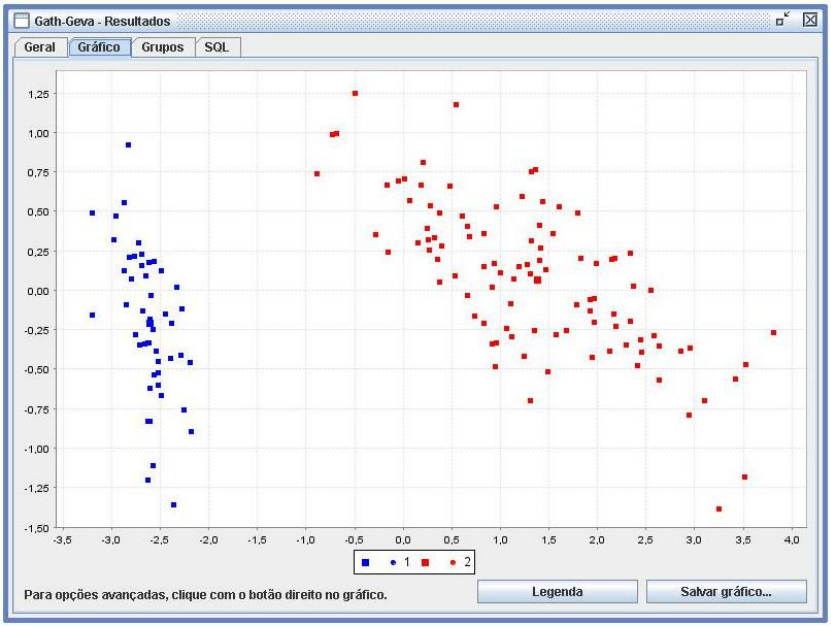

Figura 43. Resultados da clusterização com a base IRIS

A Tabela 7 apresenta o detalhamento dos resultados obtidos após a clusterização com o algoritmo Gath-Geva.

Tabela 6. Resultados da clusterização da Iris

Cluster	Quantidade de elementos	Porcentagem
1	100	66,67
2	50	33,33

Concluindo que o algoritmo funciona corretamente, realizaram-se mais alguns testes a fim de analisar os tempos de processamento por meio de diferentes valores de entrada.

8.3.2 Avaliação dos tempos de processamento

Pressupõe-se que a quantidade de elementos existentes nas bases de dados onde se utiliza ferramentas de descoberta de conhecimento seja bastante expressivo, acima de milhares ou até milhões de dados. Com isso, é de suma importância que o algoritmo tenha um

bom desempenho, e capacidade de processar o maior número de dados com o menor tempo de processamento.

A fim de obter informações para o desempenho do algoritmo Gath-Geva, foram realizados testes com uma base de dados onde os elementos foram gerados randomicamente. Com mais de 10.000 registros e quatro atributos, baseados nas características dos elementos da base IRIS.

Realizou-se a avaliação por meio da variação do número de *clusters* e a quantidade de atributos, sendo que o objetivo foi avaliar a influência no número de iterações e no tempo de processamento.

A comparação foi realizada utilizando o mesmo microcomputador, com isso foi anulada a influência de algumas variáveis como: a base dados e os parâmetros de entrada utilizados no algoritmo.

Tabela 7. Avaliação de Tempo de Processamento Gath-Geva

Quantidade de *Clusters*	**Parâmetro de *Fuzzificação***	**Atributos de Entrada**	**Número de Iterações**	**Tempos de Processamento**
2	2	2	5	00m:00s:406ms
2	2	3	8	00m:00s:625ms
2	2	4	3	00m:00s:453ms
3	2	2	3	00m:00s:516ms
3	2	3	15	00m:01s:188ms
3	2	4	2	00m:00s:407ms
5	2	2	2	00m:00s:422ms
5	2	3	2	00m:00s:656ms
5	2	4	3	00m:00s:672ms
8	2	2	3	00m:00s:828ms
8	2	3	2	00m:00s:657ms
8	2	4	2	00m:00s:688ms
10	2	2	2	00m:00s:828ms
10	2	3	2	00m:00s:828ms
10	2	4	2	00m:01s:000ms

Como pode ser observado, não foi alterado o parâmetro de *fuzzificação*, tendo-se esse valor fixo igual a 2, sendo alternados somente os atributos de entrada e a quantidade de

clusters. Nota-se rapidamente, pouca variação dos tempos de processamentos utilizando o algoritmo GG.

Verifica-se o número baixo de iterações executadas pelo algoritmo Gath-Geva em quase todos os casos, mesmo aumentando a quantidade de atributos de entrada, esse sempre termina a execução com um valor baixo de iterações. Com isso foi realizado um acompanhamento, iteração a iteração, onde se constata que isso ocorre devido ao termo exponencial utilizado no cálculo da distância.

Esse faz com que seja gerado um valor que é classificado como *Not a Number (NaN)*, ou seja como esse valor não pode ser representado como um número, resulta na parada do algoritmo.

Na Tabela 9 foi efetuada uma tentativa de fixar o número de iterações que os mesmos poderiam executar, a fim de avaliar a performance de tempo de processamento. Porém mesmo fixando esse valor como 50 iterações, em nenhum dos casos o algoritmo GG executa esse número de iterações.

Como o algoritmo Gath-Geva tem o termo exponencial obtém-se uma "aceleração", mesmo quando foi limitado a quantidade de iterações, foi possível concluir esse mesmo resultado.

Tabela 8. Avaliação dos tempos de processamento

Quantidade de Clusters	Parâmetro de Fuzzificação	Atributos de Entrada	Número de Iterações	Tempos de Processamento
2	2	2	2	00m:00s:609ms
2	2	3	11	00m:01s:078ms
2	2	4	2	00m:00s:360ms
3	2	2	3	00m:00s:438ms
3	2	3	4	00m:00s:469ms
3	2	4	3	00m:00s:485ms
5	2	2	2	00m:00s:438ms
5	2	3	2	00m:00s:438ms
5	2	4	4	00m:00s:797ms
8	2	2	2	00m:00s:609ms
8	2	3	2	00m:00s:640ms
8	2	4	4	00m:01s:203ms
10	2	2	2	00m:00s:734ms
10	2	3	2	00m:00s:781ms
10	2	4	2	00m:00s:953ms

Outros valores para o parâmetro de fuzzificação foram testados a fim de avaliar a sua influência nos resultados obtidos, pois até o momento estava sendo utilizado o valor 2 como padrão para todas as execuções. Realizando as variações desse parâmetro chega-se aos resultados, mostrados na Tabela 10.

Tabela 9. Avaliação do parâmetro de *fuzzificação*

Quantidade de *Clusters*	Parâmetro de *Fuzzificação*	Atributos de Entrada	Número de Interações	Tempos de Processamento
2	1,2	4	2	00m:00s:829ms
2	2	4	4	00m:00s:453ms
2	2,8	4	8	00m:01s:391ms
5	1,2	4	2	00m:01s:265ms
5	2	4	3	00m:00s:656ms
5	2,8	4	5	00m:05s:828ms
10	1,2	4	1	00m:01s:906ms
10	2	4	3	00m:01s:266ms
10	2,8	4	5	00m:06s:875ms

Visualizando os resultados nota-se que quando não se utiliza o parâmetro de *fuzzificação* igual a 2, causa um aumento nos tempos de processamento do algoritmo Gath-

Geva. Comprovando que é o valor que apresenta os melhores tempos de processamento para o algoritmo é igual a 2.

Para ratificar as observações sobre o tempo de processamento, levantaram-se os seguintes aspectos e influências dos parâmetros a seguir:

a) **quantidade de *clusters***: quanto maior for à quantidade de *clusters* maior vai ser o número de cálculos necessários para se chegar a um resultado satisfatório. Obtendo-se um aumento no tempo de processamento, destaca-se que o algoritmo Gath-Geva, não apresentou uma boa separação dos *clusters* quando utilizados valores maiores que 2. Porém isso está diretamente relacionado aos valores de inicialização das matrizes de pertinências;

b) **parâmetro de *fuzzicação***: quando é diferente de 2, torna a execução do algoritmo mais instável, aumentando o tempo de processamento. Porém, o algoritmo Gath-Geva devido à utilização do termo exponencial, faz com os tempos de processamento não diferenciem tanto. No entanto recomenda-se a utilização do valor 2;

c) **taxa de erro**: a rapidez na execução dos cálculos do algoritmo, é resultado da finalização pela taxa de erro, considerando que na maioria das vezes são gerados os valores *NaN*, por causa do termo exponencial no cálculo da distância.

d) **quantidade de atributos selecionados**: logicamente, pode-se constatar que quanto maior o número de atributos envolvidos no cálculo da clusterização maior será o tempo de processamento. Porém a utilização do termo exponencial pelo algoritmo Gath-Geva proporciona tempos de processamento bastante parecidos aos encontrados com quantidade menor de atributos.

Com essa análise dos tempos de processamento, teve-se a necessidade de comparar o desempenho do algoritmo com outra ferramenta de clusterização, a fim de confrontar os resultados.

8.3.3 Comparação com outra aplicação

Apesar de serem escassas as ferramentas que implementem o algoritmo Gath-Geva, foi possível realizar testes comparativos com a ferramenta *Clustering Toolbox*, desenvolvida por Janos Abonyi, Balazs Balasko e Balazs Feil do Departamento de Engenharia de Processos da Universidade de Vezprém, na Hungria. A ferramenta foi desenvolvida para funcionar no MatLab[31].

Considerando os testes no algoritmo, necessitou-se padronizar os parâmetros de entrada para nas duas ferramentas, *Clustering Toolbox* e Orion:

a) **Quantidade de *clusters:*** 4;

b) **Parâmetro de *fuzzificação*:** 2;

c) **Quantidade de interação:** 0 (finalizar baseado na taxa de erro)

d) **Taxa de erro:** 0.0001

e) **Atributos de entrada:** Todos (X, Y e Z)

A base para se realizar essa comparação é a *motorcycle*[32], onde é composto por 130 elementos, cada um com 3 atributos distintos, nomeados X, Y e Z. Acompanha a ferramenta *Clustering Toolbox*. Depois de inseridos todos os dados e inicializados os parâmetros em ambos os algoritmos, obteve-se os seguintes resultados (Tabela 11):

[31] É uma ferramenta comercial que implementa funções de matemática. Porém nos testes realizados foi utilizada uma versão de testes, disponível em: http://www.mathworks.com/

[32] É um banco de dados disponível na ferramenta *Clustering Toolbox*.

Tabela 10. Comparação dos resultados das aplicações

Ferramenta	Iterações	Cluster 1	Cluster 2	Cluster 3	Cluster 4
Shell Orion	6	21,05	35,34	33,08	10,53
Clustering Toolbox	27	26,63	17,16	28,49	27,72

Fazendo a análise dos dados conclui-se que foram bastante parecidos comprovando a real eficiência na implantação do algoritmo no módulo da *Shell* Orion. Em relação ao tempo de processamento não foi possível realizar a avaliação devido a falta dessa informação por parte da implementação do *Clustering Toolbox*.

Ambas as aplicações permitem a visualização em forma gráfica, na Figura 45 e 46, apresenta-se os resultados, podendo-se constatar e verificar a semelhança nos resultados obtidos.

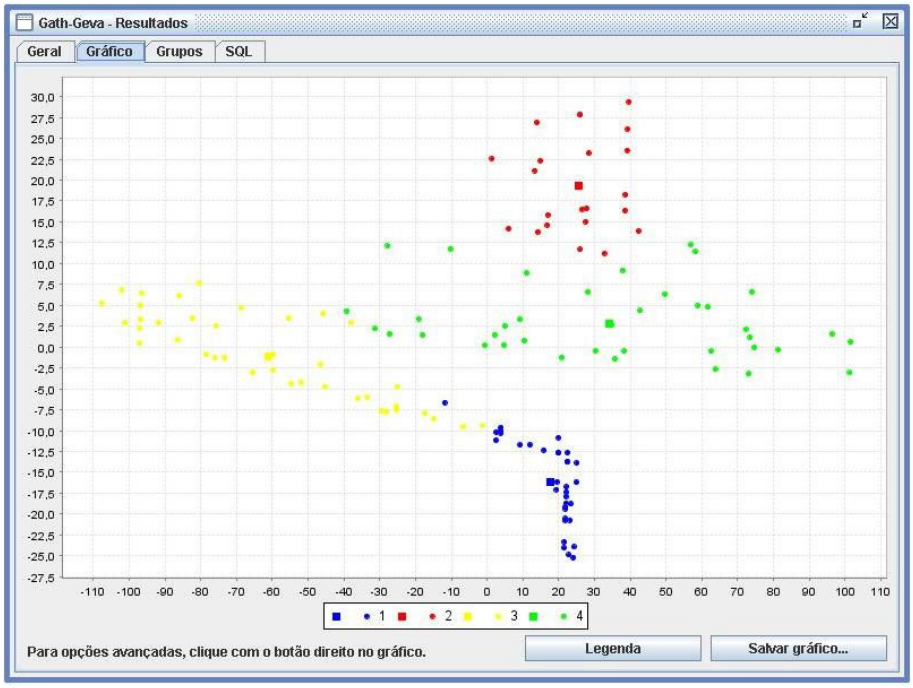

Figura 44. Clusterização com Gath-Geva da base *motorcycle*

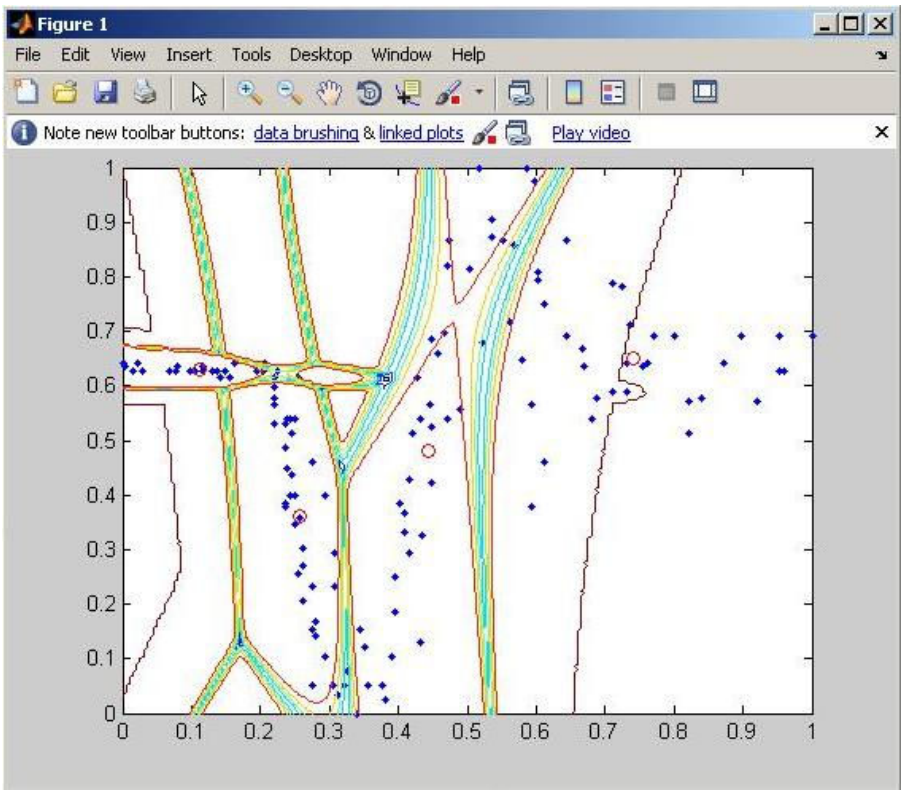

Figura 45. Clusterização com *Clustering Toolbox* da base *motorcycle*

Além dos resultados obtidos, pode-se concluir que a ferramenta, Shell Orion Data Mining, torna mais simples a tarefa de realizar a clusterização, pois facilita a inserção dos parâmetros e execução da clusterização por meio de uma interface intuitiva. Porém, na ferramenta *Clustering Toolbox*, os parâmetros são inseridos por meio de comandos seqüenciais, exigindo um conhecimento significativo do funcionamento da ferramenta para conseguir realizar o processo de clusterização.

CONCLUSÃO

Essa pesquisa demonstrou o quanto pode ser complexo o processo de analisar os dados de uma base, podendo-se ser simplificada por meio da utilização dos conceitos de *data mining*, auxiliando não somente na aquisição de novos conhecimentos, mas também na confirmação dos já existentes. Sendo que os resultados provenientes do DCBD permitem as instituições tomar melhores decisões e escolher as estratégias mais apropriadas de mercado.

O objetivo principal desta pesquisa foi compreender e estudar o conceito de *data minig,* principalmente em relação a tarefa de clusterização por meio do algoritmo Gath-Geva. Além disso, foi necessário se aprofundar nos temas de modelagem matemática e funcionamento do algoritmo GG, sendo que com esse estudo, tornou-se possível a implementação do mesmo, no módulo de clusterização da *Shell* Orion *Data Mining Engine*.

Contudo, no decorrer da pesquisa foram encontradas inúmeras dificuldades principalmente em relação à modelagem matemática do algoritmo Gath-Geva. O fator que contribuiu para isso foi o pouco número de material didático sobre o tema abordado.

Após o entendimento e implementação do algoritmo Gath-Geva, realizou-se muitos testes a fim de apurar o funcionamento do algoritmo. Concluiu-se que o mesmo conseguiu separar os *clusters* com precisão, comprovando o sucesso na implementação.

Considerando o tempo de processamento do algoritmo Gath-Geva obtiveram resultados satisfatórios quanto a clusterização, sendo o algoritmo GG normalmente termina com um menor número de iterações, finalizando-se com um menor tempo de processamento. Isso acontece, devido à utilização do termo exponencial do cálculo da distância.

Comparando com outra aplicação foi possível verificar que a interface proporcionada pela *Shell* Orion *Data Mining Engine* é bastante intuitiva, facilitando a utilização da mesma.

Ao final tem-se algumas propostas de trabalhos futuros que podem ser desenvolvidos:

a) desenvolver outras etapas relacionadas a descoberta de conhecimento em bases de dados, como por exemplo, pré e pós-processamento;
b) implementar novos algoritmos de clusterização pelo método de lógica *fuzzy*, como por exemplo, o *Fuzzy C-Means* e *Fuzzy C-Shells;*
c) realizar uma pesquisa que utilize os resultados obtidos na clusterização pelo algoritmo de lógica *fuzzy* Gath-Geva, em outras tarefas de *data mining*, como por exemplo, a classificação.
d) implementar outros métodos de clusterização, como por exemplo, o método hierárquico.

REFERÊNCIAS

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

ALVES, Luciano. **Eficiência de métodos de agrupamento de dados na modelagem nebulosa Takagi-Sugeno.** 2007. 129 f. Pós-graduação (Pós-graduação em Engenharia de Produção) - Pontifícia Universidade Católica do Paraná, Curitiba, 2007.

AMORIM, Thiago. **Conceitos, técnicas, ferramentas e aplicações de Mineração de Dados para gerar conhecimento a partir de bases de dados.** 2006. 50 f. 2006 (Graduação) - Curso de Ciência da Computação, Ufpe, Recife, 2006.

AZEVEDO, Fernando Mendes de; BRASIL, Lourdes Mattos; OLIVEIRA, Roberto Célio Limão de. **Redes neurais com aplicações em controle e em sistemas especialistas.** Florianópolis: Bookstore, 2000. 401 p. ISBN 8575020056.

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. **O método de redes neurais pelo algoritmo de Kohonen para clusterização na Shell Orion Data Mining Engine.** 2007. Trabalho de Conclusão de Curso (Bacharelado de Ciência da Computação) - Unesc, Criciuma, 2007.

BOGORNY, Vânia. **Algoritmos e ferramentas para descoberta de conhecimento em bases de dados geográficos**. Porto Alegre: Biblioteca UFRGS, 2003.

BUCKLEY, James J.; JOWERS, Leonard J.. **Simulating continuous fuzzy systems.** Berlin: Springer, 2006.

CABETE, Nélia P.; CARDOSO, Margarida G.M.S. **Algoritmo CART**: Previsão do Desempenho na Matemática do Secundário. Revista de Ciências da Computação (Universidade Aberta), Vol. 1(1), p. 11-23, 2006.

CASAGRANDE, Diego Paz. **O Módulo da tarefa de associação pelo algoritmo A*priori* no desenvolvimento da S*hell* de D*ata Mining O*rion**. 2005. Trabalho de Conclusão de Curso – Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina, 2005.

CAVALCANTI, Júnior Nicomedes Lopes. **Clusterização baseado em algoritmos fuzzy.** 2006. 96 f. Dissertação (Mestre em Ciência da Computação) - Universidade Federal do Pernambuco, Recife, 2006.

CASSETTARI JUNIOR, José Márcio. **O método de lógica fuzzy pelo algoritmo de Gustafson-Kessel para a tarefa de clusterização na Shell Orion Data Mining Engine.** 2008. Trabalho de Conclusão de Curso (Bacharelado de Ciência da Computação) - Unesc, Criciúma, 2008.

CARVALHO, Luís Alfredo Vidal de. **Datamining:** a mineração de dados no marketing, medicina, economia, engenharia e administração. Rio de Janeiro: Ciência Moderna, 2005. 225 p. ISBN 8573934441 (broch.).

CHEN, Guanrong; PHAN, Trung Tat. **Introduction to fuzzy sets, fuzzy logic, and fuzzy control systems.** Florida: Crc Press, 2001.

COSTA, Wendel T.; NEPOMUCENO, Erivelton G.; M. NETO, Oriane.**Controle nebuloso de sistemas não-lineares via método Takagi-Sugeno:** Uma abordagem didática usando Simulink. Artigo. Disponível em: <http://www.lti.pcs.usp.br/robotics/grva/publicacoes/outras/cba2004-cd-rom/cba2004/pdf/1235.pdf>. Acesso em: 13 jun. 09.

COX, Earl. **The fuzzy systems handbook:** a practitioner's guide to building and maintaining fuzzy systems. Boston: Ap Professional, 1994. 615 p.

COX, Earl. **Fuzzy modeling and genetic algorithms for data mining and exploration.** California: Morgan Kaufmann, c2005.

DEITEL, H.M; DEITEL, P. J. **Java: como programar**. 6. ed. São Paulo: Pearson Education do Brasil, 2005.

ELMASRI, Ramez; NAVATHE, Sham. **Fundamentals of database systems**. 3. ed. Redwood City: Benjamin/Cummings, 1994.

FAYYAD, Usama; PIATETSKY-SHAPIRO, Gregory; SMYTH, Padhraic. From Data mining to Knowledge Discovery in Databases. **AI Magazine**, v. 17, n. 3, p. 37-54, 1996. Disponível em: <http://kdnuggets.com/gpspubs/aimag-kdd-overview-1996-Fayyad.pdf>. Acesso em: 15 jun 2008.

GATH, I., GEVA, A.B. **Unsupervised optimal fuzzy clustering**. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 7, 1989, p. 773-781.

GUERRA, Fábio Alessandro. **Análise de métodos de agrupamento para o treinamento de redes neurais de base radial aplicadas à identificação de sistemas**. Dissertação de Mestrado. Programa de Pós-Graduação em Engenharia de Produção e Sistemas. Pontifícia Universidade Católica do Paraná, PUC. Curitiba, PR. 2006. 149p.

GIMENES, Eduardo. **"Data Mining - Data Warehouse" A importância da mineração de dados em tomadas de decisões.** 2000. 45 f. Monografia (Tecnólogo em Processamento de Dados) - Faculdade de Tecnologia de Taquaritinga, Taquaritinga, 2000.

GOLDSCHMIDT, Ronaldo; PASSOS, Emmanuel Lopes. **Data mining: uma guia prático.** Rio de Janeiro: Elsevier, 2005.

HAN, Jiawei; KAMBER, Micheline. **Data mining: concepts and techniques.** San Francisco: Morgan Kaufmann, 2001.

HAND, David; MANNILA, Heikki; SMYTH, Padhraic. **Principles of data mining**. London: The MIT Press, 2001.

HÖPPNER, Frank et al. **Fuzzy cluster analysis**: methods for classification, data analysis, and image recognition. Chichester: John Wiley & Sons, 1999.

LUGER, George F. **Inteligência artificial**: estruturas e estratégias para a solução de problemas complexos. 4. ed. Porto Alegre: Bookmann, 2004.

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

KLIR, G.; YUAN, B. **Fuzzy sets and fuzzy logic:** theory and applications. New Jersey: Prentice Hall, 1995.

NAVEGA, Sérgio. **Princípios Essenciais do Data Mining**. 2002. Disponível em: http://www.intelliwise.com/reports/i2002.htm. Acessado em Junho de 2008.

MANILLA, Heikki. **Data mining**: Machine learning, statistic and databases. In *Proceeding* of the 8th International Conference on Scientific and Statistical Database Management, pp. 1–8. 1997.

MARTINS, Denis Piazza. **A Clusterização pelo algoritmo de particionamento Kmeans na Shell de Data Mining Orion.** 2007. Trabalho de Conclusão de Curso (Bacharelado de Ciência da Computação) - Unesc, Criciuma, 2007.

MCNEILL, F. Martin; THRO, Ellen. **Fuzzy logic:** A Practical Approach. Chestnut Hill: Ap Professional, 1994.

MUNAKATA, Toshinori. **Fundamentals of the new Artificial Intelligence:** Neural, Evolutionary, Fuzzy and More. 2. ed. Londres: Springer, 2008.

OCHI, L.S., DIAS, C.R., SOARES, S.S.F. **Clusterização em mineração de dados.** Disponível em: http://bibliotecadigital.sbc.org.br/?module=Public&action=PublicationObjects&Subjects=154&publicationobjectid=9. Acesso em Julho de 2008.

OLIVEIRA, Tatyana Bitencourt Soares de. **Clusterização de dados utilizando técnicas de redes complexas e computação bioinspirada.** 95 f. Dissertação (Mestrado) - Universidade de São Paulo (usp), São Paulo, 2008.

PASSINO, Kevin M.; YURKOVICH, Stephen. **Fuzzy control.** Menlo Park: Addison-wesley, 1998.

PELEGRIN, Diana Colombo. **A Tarefa de classificação e o algoritmo ID3 para indução de árvores de decisão na *Shell* de *Data Mining* Orion.** 2005. Trabalho de Conclusão de Curso

– Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina, 2005.

PUCCIARELLI, A. J.; BARRETO, G.; SERRA, G. L. O.; **Identificação adaptativa de modelo nebuloso Takagi-Sugeno de um neutralizador de pH**, 06/2005*, 4o. Congresso Temático de Dinâmica, Controle e Aplicações – DINCON'2005,*Vol. 1, pp.1185-1191, Bauru, SP, Brasil, 2005.

RAIMUNDO, Lidiane R. **O algoritmo CART pelo critério de Gini na tarefa de classificação da Shell Orion Data Mining Engine**. 2007. 106 f. Trabalho de Conclusão de Curso (Especialização) – Faculdade 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.

RUMBAUGH, James; JACOBSON, Ivar. **UML: guia do usuário**. Rio de Janeiro: Ed. Campus, 2000.

RUSSELL, Stuart J.; NORVIG, Peter. **Inteligência artificial**. Rio de Janeiro: Elsevier, 2004.

SANTOS, Nathan Moreira dos. **Vetores e matrizes**: uma introdução à álgebra linear. 4. ed. rev. e ampl São Paulo: Thomson, 2007. 287 p.

SERRA, Laércio. **A essência do business intelligence**. São Paulo: Bekerley Brasil, 2002.

SINGH, Harry. **Data warehouse.** São Paulo: Makron Books, 2001. 382 p. ISBN 8534610347.

OLIVEIRA, Adelize Generini de. **Manipulando bancos de dados no SQL server 7.0.** Florianópolis: Visual Books, 1999. 268 p. ISBN 85-85943-58-0.

YONAMINE, Frank Sussumu. **Aprendizado não supervisionado em domínios fuzzy - algoritmo fuzzy c-means.** 2002. 18 f. Trabalho de Conclusão de Curso (Ciência da Computação) - Universidade Federal de São Carlos, Recife, 2002.

WANG, Paul P.; RUAN, Da; KERRE, Etienne E.. **Fuzzy logic.** New York: Springer, 2007.

WITTEN, I. H; FRANK, Eibe. **Data mining:** practical machine learning tools and tecniques with Java implementations. San Francisco: Morgan Kaufmann, 2000**.**

_____, Eibe. **Data mining:** practical machine learning tools and tecniques. San Francisco: Morgan Kaufmann, 2005.

WIVES, Leandro Krug. **Utilizando conceitos como descritores de textos para o processo de identificação de conglomerados (*clustering*) de documentos.** 2004. 136 f. Tese (Doutorado em Ciência da Computação) - Programa de Pós-graduação em Computação,

Universidade Federal do Rio Grande do Sul, Porto Alegre, BR – RS. Disponível em: <http://www.leandro.wives.nom.br/publicacoes/Tese.pdf> Acesso em: 13 set. 2006.

ZORZATE, Evanio Henrique. **Aplicação de técnicas multivariadas e sistemas fuzzy agrupamentos e inferência na estimação de curvas de demanda de consumidores de baixa tensão.** 2006. 138 f. Doutorado (Mestre em Engenharia Elétrica) - Universidade Federal de Mato Grosso do Sul, Campo Grande, 2006.

WANG, Li-Xin. **A course in fuzzy systems and control.** Hong Kong: Prentice Hall, 1997.

XIONG, X., CHAN, K. L. **Towards na unsupervised optimal fuzzy clustering algorithm for image database organisation**, Proceedings 15th International Conference on Pattern Recognition, Vol. 3, pp. 897 -900. 2000.