Agrupamento *fuzzy* pelo algoritmo Gath-Geva na *Shell Orion Data Mining Engine*

Daniel Perego[1], Ruano M. Pereira[1], Maicon B. Palhano[1], Gabriel Felippe[1], Priscyla W. T. de Azevedo Simões[1], Merisandra C. de Mattos Garcia[1]

[1]Grupo de Pesquisa em Inteligência Computacional Aplicada, Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense (UNESC), Brasil.

perego86@gmail.com,{gabrielheavy,ruanopereira}@hotmail.com, {maiconpalhano,pri,mem}@unesc.net

Abstract. *The advance of computational models provided an evolution in capacity of process and storage of informations. Therewith, has a necessity of develop tools for analyze those informations, generating news knowledge. Stand out the applications of methods and tasks of data mining, like this article, where is approached the fuzzy grouping and the implementation of Gath-Geva algorithm in a tool of Data mining, the Shell Orion Data Mining, as to algorithms tests, were made using a database of Iris and the motorcycle for test the perfomance of same, how the capacity of separation of clusters by the algorithm.*

Resumo. *O avanço dos modelos computacionais proporcionou uma evolução na capacidade de processamento e armazenamento de informações. Com isso,tem-se a necessidade de desenvolver ferramentas para analisar essas informações gerando novos conhecimentos. Destacam-se as aplicações de métodos e tarefas de data mining, como neste artigo, onde aborda-se o agrupamento fuzzy e a implementação do algoritmo Gath-Geva em uma ferramenta de data mining, a Shell Orion Data Mining, quanto aos testes do algoritmo, foram feitos utilizando-se as bases de dados das iridáceas e motorcycle para testar o desempenho do mesmo quanto à capacidade de separação dos clusters pelo algoritmo.*

1. Introdução

A evolução da tecnologia de informação proporcionou as empresas e instituições uma maior capacidade de armazenamento e processamento dos seus dados, com isso, 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 acaba inviabilizando a capacidade humana em analisar e adquirir conhecimento do conjunto de dados [Rezende 2005]. Os dados geralmente eram analisados e interpretados por especialistas de forma manual por meio de Sistemas Gerenciadores de Banco de Dados (SGBD), ferramentas estatísticas e planilhas eletrônicas, porém com o aumento dessas bases de dados era visível à demora e o custo para aplicação desse tipo de análise [Melanda 2005].

O *data mining* surge em resposta a essa necessidade de novos métodos, capazes de descobrir informações relevantes e de forma automática em grandes conjuntos de dados, sendo considerada a principal etapa do processo *Knowledge Discovery in Database* (KDD) [Borges 2010].

O KDD originou-se de diversas áreas tais como: estatística, inteligência computacional, banco de dados e aprendizado de máquina, onde por meio de tarefas e métodos sobre os dados obtém-se padrões e relacionamentos novos e relevantes.

No caso da tarefa do agrupamento, 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. O agrupamento pode ser realizado 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 Gath-Geva), dentre outros [Alves 2007].

A maioria dos algoritmos realizam 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 que o mesmo tenha um grau de similaridade a diversos *clusters*, considerando essas situações denominadas *fuzzy* por Goebel e Gruenwald (1999), forma uma metodologia chave para representar e processar de incerteza por imprecisão.

Além disso, outro problema apresentado pela maioria dos algoritmos de agrupamento refere-se à determinação do número de grupos, pela dificuldade de encontrar o número ideal, normalmente esse valor deve ser informado como parâmetro no algoritmo, sendo que somente após a execução é possível realizar a análise para validação dos resultados. Considerando isso, os pesquisadores Gath-Geva em 1989, determinam 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 agrupamento 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 agrupamento da *Shell Orion Data Mining Engine*, aumentando-se as funcionalidades desta ferramenta.

2. Agrupamento *Fuzzy*

Os algoritmos de agrupamento 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 e hierárquico, 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 e Kerre 2007]. 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 e Phan 2001]. 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 1, esse tipo de questão não pode ser tratada pela lógica booleana ou clássica [Cox 1994].

Figura 1. Representação de uma taça

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].

3. 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 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].

Os *clusters* de formas diferentes podem ser obtidos por meio de uma definição adequada de *clusters*, ou por meio de diferentes medidas de distância. O algoritmo Gath-geva, obtem *clusters* mais precisos, com menor erro de aproximação que o algoritmo Gustaffson-Kessel, isso devido à utilização da distância de forma exponencial [Abonyi 2002]. Além da incorporação da distância exponencial no algoritmo *Fuzzy C-Means* (FCM), o Gath-geva utiliza, similarmente ao algoritmo Gustaffson-kessel, 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 e Geva 1989].

Na execução do algoritmo Gath-Geva é necessário definir além da base de dados, os parâmetros para o funcionamento do algoritmo. Os parâmetros a serem definidos são:

a) **quantidade de *clusters*:** compreende o número de agrupamentos a serem realizados nos dados de entrada;
b) **termo de fuzzificação:** é o grau de fuzzificação dos elementos pertencentes à base de dados;
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 for inferior à taxa de erro, o algoritmo continuará realizando os cálculos.

Primeiramente deve-se inicializar as matrizes de pertinências de forma randômica, porém também pode-se utilizar outros algoritmos mais simples como o FCM, a fim de se obter valores que aperfeiçoem o funcionamento do algoritmo. Os valores devem obedecer a propriedade que estabelece que a soma das pertinências em relação a todos os *clusters* deve ser igual a 1.

1. Na primeira etapa do algoritmo calcula-se o centro dos *clusters*, utilizando-se a seguinte equação:

$$\mathrm{Vi} = \frac{\sum_{j=1}^{N} (\mu ij)^{q} X_j}{\sum_{j=1}^{N} (\mu ij)^{q}}$$

2. Após realizado o cálculo do centro dos *clusters*, deve-se calcular as matrizes de covariância, por meio da equação:

$$\mathrm{Fi} = \frac{-\sum_{j=1}^{N} \mu ji^{q} \ (xJ - vI)^{T} \ (Xj - Vi)}{\sum_{j=1}^{N} \mu ji^{q}}$$

3. A próxima etapa consiste no cálculo da distância entre os elementos e os centros dos *clusters*. Devendo-se utilizar a seguinte equação:

$$d_e^2(\mathrm{Xj}, \mathrm{Vi}) = \frac{[\det(Fi)]^{\frac{1}{2}}}{Pi} \exp\left[\frac{(Xj - Vi)^{T} Fi^{-1} (Xj, -Vi)}{2}\right]$$

4. Após encontrar as distâncias entre os elementos e os *clusters*, pode-se atualizar a matriz de pertinência por meio da equação:

$$\mu_{ij} = \frac{\frac{1}{\mathrm{d}_{ij}^2}}{\sum_{K=1}^{V} \left(\frac{1}{\mathrm{d}_{kj}^2}\right)}$$

Estes cálculos são realizados para todos os elementos e *clusters*, tendo-se uma matriz atualizada das pertinências.

5. Atualizada a matriz de pertinência, deve-se calcular a taxa de erro, onde é por meio desse valor que se permite a parada do algoritmo.

$$||U^{l} - U^{l-1}||$$

4. O algoritmo Gath-Geva na *Shell Orion Data Mining Engine*

A implementação do algoritmo Gath-Geva para a tarefa de agrupamento na *Shell Orion Data Mining Engine* foi realizada por meio da linguagem orientada a objeto Java, com ambiente de desenvolvimento NetBeans.

Na Figura 2 tem-se a interface principal, onde devem ser informados os parâmetros de entrada para execução do algoritmo Gath-Geva.

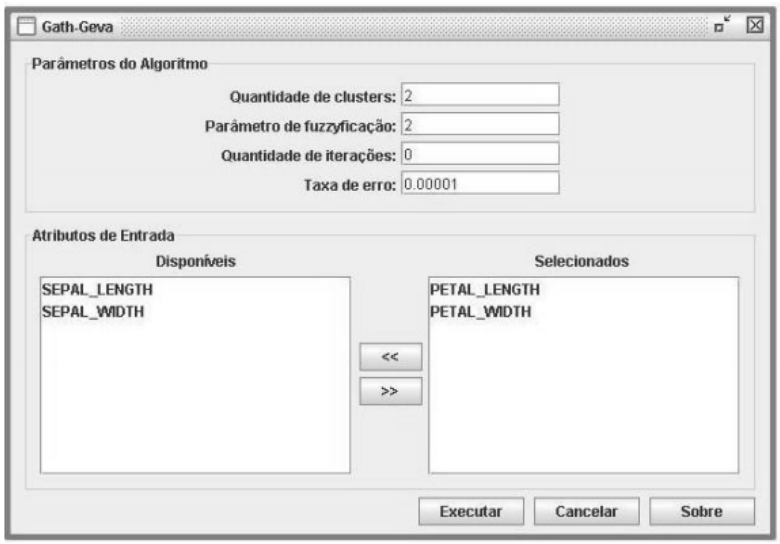

Figura 2. Interface inicial do Gath-Geva

Executando-se o algoritmo, após o término das iterações, os resultados podem ser apresentados de diversas maneiras. Sendo que inicialmente, a tela apresentada para o usuário é um resumo dos resultados encontrados, descrevendo-se os parâmetros e atributos de entrada utilizados, informações sobre *cluster*, os centros dos *clusters* atualizados e o atributo de saída, sendo que esse pode ser alterado pela opção atributo de saída (Figura 3).

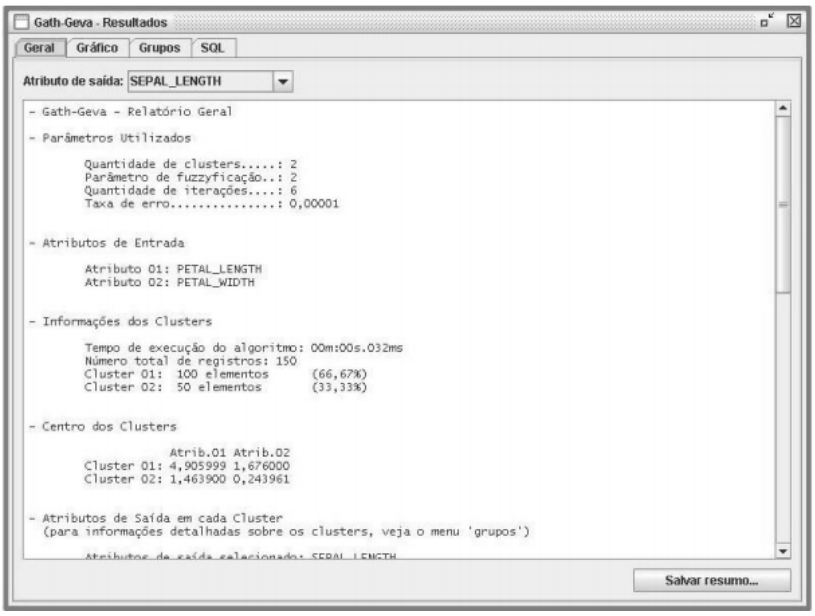

Figura 3. Resumo do resultado do agrupamento

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

5. Resultados Obtidos

Concluindo-se a implementação do algoritmo no módulo de agrupamento 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.

Nos testes empregou-se a bases de dados das Iridáceas disponibilizada pela UCI *Machine Learning Repository*, onde contém dados acerca da flor Íris, com os tipos da planta, divididos em Setosa, Versicolor e Virgínica, sendo esta base de dados composta por 150 atributos, no qual, 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 inicio do algoritmo:

a) **quantidade de *clusters*:** 2;
b) **parâmetro de fuzzyficação:** 2;
c) **quantidade de interações:** 0 (ilimitado);
d) **taxa de erro:** 0,00001;
e) **atributos de entrada:** sepal_lenght, sepal_width, petal_lenght, petal_width.

Após a execução os resultados obtidos podem ser observados na Tabela 1 apresenta o detalhamento dos resultados obtidos após o agrupamento com o algoritmo Gath-Geva.

Tabela 1. Resultado do agrupamento da IRIS

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

Posteriormente, na realização dos testes avaliou-se o desempenho do algoritmo Gath-Geva, para isso, 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 de dados das Iridáceas.

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 2. Avaliação de tempo de processamento do Gath-Geva na *Shell Orion*

Quantidade de *cluster*	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 pouca variação dos tempos de processamento utilizando-se o algoritmo Gath-Geva.

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.

Na Tabela 3 foi efetuada uma tentativa de fixar o número de iterações que os mesmos poderiam executar, a fim de avaliar a performance do tempo de processamento. Porém mesmo fixando esse valor como 50 iterações, em nenhum dos casos o algoritmo Gath-Geva executa esse número de iterações. Como o algoritmo Gath-Geva tem o termo exponencial obtém-se uma "aceleração", mesmo quando foi limitado à quantidade de iterações, pode-se concluir esse mesmo.

Tabela 3. Avaliação dos tempos de processamento

Quantidade de *cluster*	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 4.

Tabela 4. Avaliação de parâmetros de fuzzificação

Quantidade de *cluster*	Parâmetro de fuzzificação	Atributos de entrada	Número de iterações	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 fuzzificação:** 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 do agrupamento maior será o tempo de processamento. Porém a utilização do termo exponencial pelo algoritmo Gath-Geva proporciona tempos de processamento bastante parecidos aos encontrados menos atributos.

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 MatLab31.

Considerando os testes no algoritmo, necessitou-se padronizar os parâmetros de entrada para as 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 de dados utilizada para se realizar essa comparação foi a *motorcycle*, sendo composta por 130 elementos, cada um com 3 atributos distintos, nomeados *X*, *Y* e *Z*. Esta base de dados acompanha a ferramenta *Clustering Toolbox*. Depois de inseridos todos os dados e inicializados os parâmetros em ambos os algoritmos, obtiveram-se resultados semelhantes, porém a pesquisa terá continuidade por meio da aplicação de testes estatísticos a fim de comparar o desempenho das ferramentas. Em relação ao tempo de processamento não foi possível realizar a avaliação, pois o *Clustering Toolbox* não possui este recurso.

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

6. Considerações Finais

Essa pesquisa demonstrou o quanto pode ser complexo o processo de analisar os dados de uma base, podendo ser simplificado 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 KDD, podem permitir que instituições melhorem a tomada de decisões e na escolha das estratégias mais apropriadas de mercado.

O objetivo principal desta pesquisa foi empreender e estudar o conceito de data mining, principalmente em relação ao agrupamento por meio do algoritmo Gath-Geva. Além disso, foi necessário se aprofundar nos temas de modelagem matemática e funcionamento do algoritmo, sendo que com esse estudo, tornou-se possível a implementação do mesmo na *Shell Orion Data Mining Engine*.

Considerando o tempo de processamento do algoritmo Gath-Geva obtiveram resultados satisfatórios quanto à clusterização, sendo o algoritmo Gath-Geva 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 à 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.

Referências

Abonyi, J., Babuska, R. e Szeifert, F. (2002) 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, p. 612-621.

Alves, L. (2007) Eficiência de Métodos de agrupamento de dados na modelagem nebulosa Takagi-Sugeno. Dissertação Pontifica Universidade Católica do Paraná.

Borges, E. (2010) Um novo algoritmo imunológico artificial para agrupamento de dados, Dissertação (Mestrado em Engenharia Elétrica) Escola de Engenharia, Universidade Presbiteriana Mackenzie, São Paulo.

Chen, Guanrong; Phan, Trung Tat. Introduction to fuzzy sets, fuzzy logic, and fuzzy control systems. Florida: Crc Press, 2001.

Cox, E.(1994) The Fuzzy systems handbook: a practitioner's guide to building and maintaining fuzzy System. Boston: Ap professional. 615p.

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

Goebel, M. e Gruenwald, L. (1999) A survey of data mining and knowledge discovery software tools. SIGKDD Explorations, v. 1, p. 20-33.

Höppner, F. (1999) Fuzzy cluster analysis: methods for classification, data analysis, and image recognition. Chichester: John wiley & Sons.

Melanda, E. A. (2005) Pós-processamento em regras de associação. Tese (Doutorado em Ciências de Computação e Matemática Computacional)-Instituto de Ciências Matemáticas e de Computação, São Carlos.

Munakata, T. (2008) Fundamentals of the new Artificial Intelligence: Neural Evolutionary, Fuzzy and more. 2.ed. Londres Springer.

Rezende, Solange Oliveira. Sistemas inteligentes: fundamentos e aplicações. São Paulo: Manole, 2005.

Wang, P. P., Ruan, D.e Kerre E. (2008) Fuzzy Logic. New York: Springer.

Chen, G., Phan, T. T. Introduction to fuzzy sets, fuzzy logic, and fuzzy control systems. Florida: CrcPress 2001.