UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Fernando Sarturi Prass

ESTUDO COMPARATIVO ENTRE ALGORITMOS DE ANÁLISE DE AGRUPAMENTOS EM *DATA MINING*

Dissertação de Mestrado submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação

Prof. Dr. Paulo José Ogliari

Florianópolis, Novembro de 2004.

ESTUDO COMPARATIVO ENTRE ALGORITMOS DE ANÁLISE DE AGRUPAMENTOS EM *DATA MINING*

Fernando Sarturi Prass

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação, Área de Concentração de Sistemas de Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

_____

Prof. Dr. Raul Sidnei Wazlawick

Coordenador do Curso

Banca Examinadora

_____

Prof. Dr. Paulo José Ogliari (Orientador)

_____

Prof. Dr. Dalton Francisco de Andrade

_____

Prof. Dr. Pedro Alberto Barbetta

_____

Prof. Dr. Marcello Thiry Comicholi da Costa

_____

Prof. Dr. Wilton de Oliveira Bussab

AGRADECIMENTOS

A Deus.

A meus pais, Loter e Aldair, tudo o que sou devo a vocês.

A meus irmãos, Fábio e Francine, por todo o carinho e afeto que vocês sempre me deram, mesmo a distância.

Ao meu orientador, Prof. Dr. Paulo José Ogliari, pela paciência e, principalmente, pelos ensinamentos transmitidos.

Aos professores Dr. Dalton Andrade e Dr. Pedro Barbetta, pelo interesse demonstrado toda a vez que eu lhes fazia perguntas sobre o tema.

A Walter Priesnitz, pelo apoio dado desde o momento da matrícula no curso até a defesa da dissertação.

Ao pessoal do projeto SIATU, em especial a João Goulart, Cristiano Caetano, Jefferson Ramos, Maguio Mizuki e Marcello Thiry, não apenas pela oportunidade de crescimento profissional, mas também pela amizade.

Aos amigos de longa data Rafael Speroni e Renato Noal, e também aos amigos conquistados durante o curso: Alberto Pereira (o catarinense mais gaúcho que já conheci), Juliano Pacheco, Rafael Andrade e Wilton Souza.

A todos aqueles que contribuíram direta ou indiretamente para a conclusão deste trabalho.

RESUMO

O objetivo é apresentar um estudo comparativo dos principais modelos de algoritmos de Análise de Agrupamento (*Cluster Analysis*) existentes na literatura e implementados em *softwares*, visando o seu uso no processo de descoberta de conhecimentos em grandes bancos de dados (*Knowledge Discovery in Databases - KDD*). Os algoritmos de agrupamento são diferenciados de acordo com o seu método de formação (Hierárquico, Partição, Baseado em Modelo, Baseado em Grade e Baseado em Densidade) e também pela medida de distância que expressa a similaridade ou dissimilaridade entre os objetos. Mostram-se também critérios de mensuração para que se possam avaliar quais os melhores algoritmos para grandes bases de dados. Os algoritmos foram avaliados com dados reais e simulados utilizando a Linguagem *R*, que apontou o algoritmo *k-medoid* como o mais preciso e rápido. O trabalho mostra que o uso de Análise de Agrupamentos (AA) pode ser feito através de *software* gratuito e com máquina de baixo custo, mas para se obtenham bons resultados são necessários sólidos conhecimentos teóricos sobre AA.

Palavras-chaves: Algoritmos, Análise de Agrupamento, Mineração de Dados, Linguagem *R*.

ABSTRACT

The objective of this work is to present a comparative study about the main models of Cluster Analysis available in todays literature and implemented in softwares, with the purpose of Knowlege Discovery in Databases (KDD). Cluster Analysis (CA) algorithms differ according to the method used (Hierarchical, Partition, Model-Based, Grid-Based and Density-Based) and also cording to the measure of the distances that expresses the similarities or dissimilarites between objects. Measuring criterias are used to evaluate which are the best algorithms for large databases. The algorithms are evaluated in R Language with real and simulated data, that indicated k-medoid is the most fast and accurate algorithms. This work proof that the use of Cluster Analysis can be do with free softwares and with machine of the lower cost, but to have good results is necessary good theoretical knowledge about CA.

Keywords: Algorithm, Cluster Analysis, Data Mining, R Language.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Nos dias de hoje a informação adquiriu um valor que até alguns poucos anos atrás era impensável, tanto que em muitos casos a maior riqueza que a empresa possui é aquilo que ela sabe sobre seus clientes.

Com o advento das transações eletrônicas, em especial as realizadas via Internet, as empresas acumulam em seus bancos de dados mais e mais informações. Em conseqüência disso, estes bancos de dados passam a conter informações ricas sobre vários dos procedimentos dessas empresas e, principalmente, sobre seus clientes.

Com base nas informações sobre o cliente, como transações comerciais, hábitos de consumo e até mesmo *hobbies*, é possível traçar associações que revelem oportunidades de novos negócios ou de melhora nos serviços prestados, como, por exemplo oferecer um atendimento personalizado.

Com o crescimento do volume de dados, cresceu também a busca pela informação que estes dados "escondem". Com as diversas técnicas de mineração de dados (*data mining*) desenvolvidas, esta busca está cada dia mais rápida e eficaz. Este trabalho discute uma das técnicas de análise de dados existentes, a Análise de Agrupamentos.

Conforme CHEN *et al.* (1996), Análise de Agrupamentos (AA) é um processo de classificação não supervisionado para agrupar de forma física ou abstrata objetos similares em classes (também chamado de grupos, agrupamentos, conglomerados ou *clusters*).

1.1. Definição do problema

Dividir um conjunto de objetos em grupos menores e mais homogêneos é uma operação fundamental para mineração de dados, já que quanto menor o grupo, mais fácil de ser modelado e gerenciado (HUANG, 1997b).

A divisão de um grupo em outros menores se dá através de algoritmos de Análise de Agrupamento. Segundo OLIVEIRA e BORATTI (1999), um algoritmo é uma seqüência finita e lógica de instruções ou passos, que mostram como resolver um determinado problema. Um algoritmo especifica não apenas as instruções, mas também a ordem em que elas devem ser executadas.

Existe um grande número de algoritmos de Análise de Agrupamentos. Basicamente, um algoritmo de AA é formado pela medida de similaridade[1] ou de dissimilaridade adotada para medir o quanto semelhantes ou distintos são dois objetos e pelo método de formação do agrupamento.

As particularidades dos métodos de formação e das medidas de similaridade influenciam decisivamente sobre qual algoritmo utilizar em cada conjunto de dados, o que torna um algoritmo mais ou menos indicado para uma determinada tarefa. Como exemplo disto, pode-se citar o fato de alguns algoritmos encontrarem facilmente grupos esféricos[2], mas possuírem dificuldades para encontrar grupos com formato cilíndrico ou com formato aleatório. A escolha do algoritmo errado trará, como conseqüência, resultados imprecisos e inúteis.

Outro ponto importante a ser levado em conta na escolha do algoritmo é o tamanho das bases de dados. Freqüentemente elas possuem milhões de registros e isto torna impraticável o uso de alguns algoritmos, especialmente os hierárquicos, devido ao tempo de resposta e aos recursos computacionais (principalmente memória RAM) que consomem.

Além do problema do desempenho, grandes bancos de dados trazem consigo a tendência da existência de um volume significativo de dados discrepantes (*outliers*) e/ou inconsistentes. Estes valores, se não tratados de maneira adequada, também podem interferir no resultado final da análise. Portanto, é necessário fazer um pré-processamento dos dados.

Levando-se em conta as diversas opções de métodos de formação de agrupamentos e de medidas de similaridade/dissimilaridade, bem como a necessidade de desempenho e de tratamento de informações errôneas em grandes bancos de dados, a escolha do algoritmo se torna uma tarefa das mais complexas. Este trabalho pretende esclarecer tais pontos, explicando e comparando os principais métodos de Análise de Agrupamentos existentes.

1.2. Justificativa

A Análise de Agrupamentos tem sido largamente utilizada em áreas como pesquisa de mercado, análise de dados e reconhecimento de padrões. Além disto, os algoritmos estão sobre um processo de desenvolvimento bastante intenso (HAN e KAMBER, 2001).

---

[1] BUSSAB *et al.* (1990), propõem o uso do termo *parecença*.

[2] Diz-se que um *cluster* (agrupamento) possui formato esférico quando ao se plotar os indivíduos que fazem parte do agrupamento no eixo cartesiano forma-se a figura de um círculo.

Esta grande utilização em mineração de dados não ocorre por acaso. A Análise de Agrupamentos traz consigo uma série de benefícios, tais como:

- possibilita ao usuário encontrar grupos úteis;
- auxilia no entendimento das características do conjunto de dados;
- pode ser usada na geração de hipóteses;
- permite predição com base nos grupos formados.
- permite o desenvolvimento de um esquema de classificação para novos dados.

Existem diversos algoritmos de AA e as ferramentas de mineração de dados trazem implementados tipos específicos destes algoritmos. Conhecer as características e particularidades de cada uma deles auxilia na tomada de decisão sobre qual utilizar, fazendo com que a empresa economize tempo e, principalmente, dinheiro. A justificativa deste trabalho se dá justamente por isto, uma vez que:

- faz uma revisão bibliográfica dos novos métodos de formação de agrupamento, não esquecendo de mostrar os mais tradicionais;
- apresenta diferentes formas de calcular a similaridade entre objetos, existindo variáveis de tipos mistos ou não.
- mostra a utilização dos algoritmos em uma linguagem gratuita.
- compara os resultados obtidos com os testes, apontando as vantagens e desvantagens dos algoritmos.

1.3. Objetivos

1.3.1. Objetivo geral

Estudar a eficiência e a efetividade dos principais algoritmos de Análise de Agrupamentos em grandes bancos de dados.

1.3.2. Objetivos específicos

Levantar principais algoritmos de Análise de Agrupamentos.

Levantar as medidas de parecença para cada tipo de variável.

Relacionar as vantagens e deficiências dos algoritmos.

Aplicar os algoritmos disponíveis na linguagem *R* em uma base com dados reais e em outra com dados simulados.

Verificar a dimensionalidade dos algoritmos.

Verificar a escalabilidade dos algoritmos.

Verificar a interpretabilidade e usabilidade dos resultados gerados.

1.4. Limitação

Este trabalho visa estudar alguns dos principais algoritmos de AA existentes atualmente na literatura mundial e testar aqueles presentes na linguagem *R* em grandes bases de dados. Não foi criado ou implementado nenhum novo algoritmo.

1.5. Estrutura do trabalho

Este trabalho está dividido em seis capítulos, apresentados a seguir:

O primeiro capítulo apresenta uma breve introdução, com a definição do problema, a justificativa, o objetivo geral, os objetivos específicos e a limitação do trabalho.

No segundo capítulo são apresentadas e conceituadas as fases do processo de descoberta de conhecimento em banco de dados, dando um maior destaque para a fase de mineração de dados.

O terceiro capítulo trata dos algoritmos de Análise de Agrupamentos, mostrando as diferentes medidas de similaridade e os diversos métodos de formação.

O quarto capítulo apresenta os materiais e os métodos empregados nos testes.

No quinto capítulo serão demonstrados os testes com os algoritmos em um banco de dados real e em uma base simulada.

Finalmente, o sexto capítulo traz as conclusões e sugestões para pesquisas futuras.

2. DESCOBERTA DE CONHECIMENTO EM BANCOS DE DADOS

O aumento das transações comerciais por meio eletrônico, em especial as feitas via *Internet*, possibilitou as empresas armazenar em seus bancos de dados registros contendo dados importantes sobre seus clientes.

Os produtos adquiridos, e até mesmo aqueles que foram apenas consultados mas não comprados (isto para o caso das transações via *Internet*), aliados aos dados exigidos durante o cadastro do cliente, forma o que é chamado de "perfil do cliente".

Conhecer o perfil do cliente traz uma série de benefícios para a instituição, sendo o principal deles, a capacidade de melhorar a qualidade de seus serviços prestados. Conhecendo o público alvo é possível montar uma melhor estratégia de *marketing* e com isto obter resultados mais significativos com a venda de produtos e/ou serviços.

O problema é que estes registros, muitas vezes, representam apenas dados e não conhecimento. Visando transformar estes dados em conhecimento, surge o processo chamado de Descoberta de Conhecimento em Bancos de Dados (*Knowledge Discovery in Databases - KDD*), que FAYYAD *et al.* (1996) definem como sendo "o processo, não trivial, de extração de informações implícitas, previamente desconhecidas e potencialmente úteis, a partir dos dados armazenados em um banco de dados".

Os autores justificam o seu conceito dizendo que o processo é não trivial já que alguma técnica de busca ou inferência é envolvida, ou seja, não é apenas um processo de computação direta. Os padrões descobertos devem ser válidos com algum grau de certeza, novos (para o sistema e de preferência também para o usuário), potencialmente úteis (trazer algum benefício) e compreensíveis (se não imediatamente então depois da interpretação).

FAYYAD *et al.* (1996) dizem ainda que o KDD contém uma série de passos, a saber: seleção, pré-processamento e limpeza, transformação, mineração de dados (*data mining*) e interpretação/avaliação. Simplificando, pode-se dizer que o processo de KDD compreende, na verdade, todo o ciclo que o dado percorre até virar informação, conforme pode ser visto na Figura 1.

DINIZ e LOUZADA NETO (2000) afirmam que, embora esses passos devam ser executados na ordem apresentada, o processo é interativo e iterativo. Diz-se que o processo é interativo, pois o usuário pode, e por vezes necessita, continuamente intervir e controlar o

curso das atividades. Diz-se também que é iterativo, por ser uma seqüência finita de operações em que o resultado de cada uma é dependente dos resultados das que a precedem.

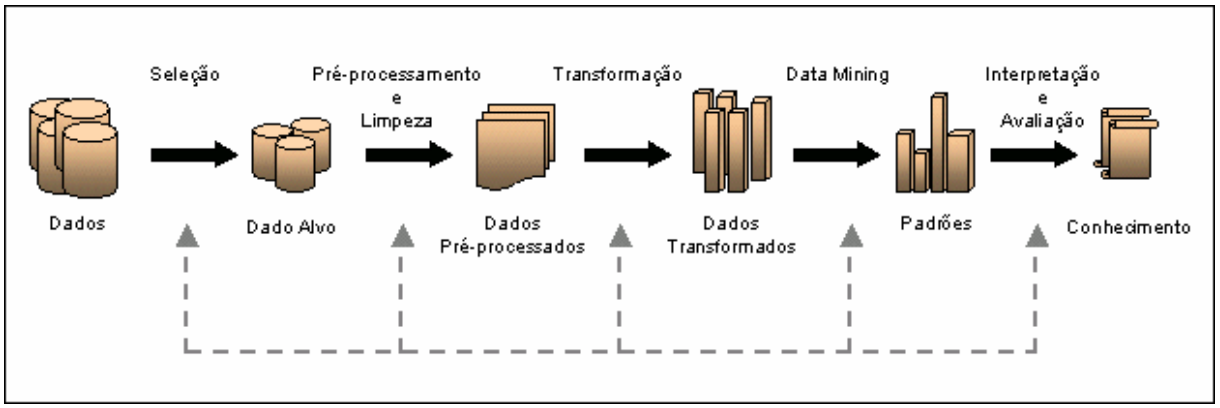

Figura 1: O ciclo do processo de KDD. Fonte: Adaptação de FAYYAD *et al.* (1996).

Baseado na definição é possível ver que KDD é uma tarefa intensiva de descoberta de conhecimento. Possui interações complexas, feitas ao longo do tempo, entre o homem e o banco de dados através de um conjunto heterogêneo de ferramentas.

Dada uma visão geral sobre o processo de KDD, os tópicos a seguir examinam os conceitos a respeito das suas fases. Alguns pesquisadores tendem a considerar KDD e *Data Mining* como sinônimos, este trabalho segue a visão de autores como FAYYAD, *et al.*(1996), que não tratam *Data Mining* como um sinônimo de KDD, mas sim como um dos passos dele.

2.1. Seleção dos dados

A fase de seleção dos dados é a primeira no processo de descobrimento de informação. Nesta fase é escolhido o conjunto de dados, pertencente a um domínio, contendo todas as possíveis variáveis (também chamadas de características ou atributos) e registros (também chamados de casos ou observações) que farão parte da análise. Normalmente, a escolha dos dados fica a critério de um especialista do domínio.

O processo de seleção é bastante complexo, uma vez que os dados podem vir de uma série de fontes diferentes (*data warehouses*, planilhas, sistemas legados) e podem possuir os mais diversos formatos. DUNKEL *et al.* (1997), destaca que este passo possui impacto significante sobre a qualidade do resultado do processo.

2.2. Pré-processamento e limpeza dos dados

Esta é uma parte crucial no processo, pois a qualidade dos dados vai determinar a eficiência dos algoritmos de mineração. Nesta etapa deverão ser realizadas tarefas que

eliminem dados redundantes e inconsistentes, recuperem dados incompletos e avaliem possíveis dados discrepantes ao conjunto (*outliers*). Mais uma vez, o auxílio do especialista do domínio é fundamental.

Nesta fase também são utilizados métodos de redução ou transformação para diminuir o número de variáveis envolvidas no processo, visando com isto melhorar o desempenho do algoritmo de análise.

DUNKEL *et al.* (1997) afirma que a identificação de dados inapropriados dentro do conjunto selecionado é problemática, e isto dificulta a automatização desta fase. Definir um dado como "ruim" dentro do conjunto depende da estrutura do mesmo e também de que aplicação é dada a ele.

2.2.1. Dados ausentes (*missing values*)

Um problema bastante comum nesta fase é a ausência de valores para determinadas variáveis, ou seja, registros com dados incompletos, seja por falhas no processo de seleção ou de revisão.

O tratamento destes casos é necessário para que os resultados do processo de mineração sejam confiáveis. Existem basicamente três alternativas de solução para esse problema:

- *Usar técnicas de imputação*, ou seja, fazer a previsão dos dados ausentes e completá-los individualmente. Esta solução é bastante eficiente para um conjunto pequeno de dados, onde pode ser feita pelo especialista no domínio de forma manual. O problema é que normalmente o volume de dados é muito grande, o que obriga o uso de softwares que apresentam resultados não tão precisos;
- *Substituir o valor faltante pela média aritmética da variável*. Esta técnica apresenta dois problemas: aplica-se apenas a variáveis numéricas e, quando pode ser aplicada, substitui o dado faltante por um aproximado, podendo acarretar na obtenção de resultados não tão corretos;
- *Excluir o registro inteiro*. Embora esta técnica exclua da análise um caso inteiro, muitas vezes pela falta de apenas um de seus atributos, é a melhor solução pois elimina o risco da análise ser feita com dados não reais.

2.2.2. Dados discrepantes (*outliers*)

Dados que possuem valores extremos, atípicos ou com características bastante distintas dos demais registros são chamados de discrepantes, ou *outliers.* Normalmente registros que contêm valores *outliers* são descartados da amostra, porém isto só deve ocorrer quando o dado representar um erro de observação, de medida ou algum outro problema similar.

O dado deve ser cuidadosamente analisado antes da exclusão, pois embora atípico, o valor pode ser verdadeiro. *Outliers* podem representar, por exemplo, um comportamento não usual, uma tendência ou ainda transações fraudulentas. Encontrar estes valores é, muitas vezes, os objetivo da mineração de dados.

2.3. Transformação dos dados

Após serem selecionados, limpos e pré-processados, os dados necessitam ser armazenados e formatados adequadamente para que os algoritmos possam ser aplicados.

Em grandes corporações é comum encontrar computadores rodando diferentes sistemas operacionais e diferentes Sistemas Gerenciadores de Bancos de Dados (SGDB). Estes dados que estão dispersos devem ser agrupados em um repositório único.

Além disto, nesta fase, se necessário, é possível obter dados faltantes através da transformação ou combinação de outros, são os chamados “dados derivados”. Um exemplo de um dado que pode ser calculado a partir de outro é a idade de um indivíduo, que pode ser encontrada a partir de sua data de nascimento.

2.4. Mineração de dados (*data mining*)

Todas as etapas do processo de KDD são importantes para o sucesso do mesmo. Entretanto, é a etapa de Mineração de Dados (*data mining*) que recebe o maior destaque na literatura. Segundo FAYYAD *et al.* (1996), Mineração de Dados é o processo de reconhecimento de padrões válidos ou não, existentes nos dados armazenados em grandes bancos de dados.

Já conforme BERRY e LINOFF (1997), Mineração de Dados é a exploração e análise, de forma automática ou semi-automática, de grandes bases de dados com objetivo de descobrir padrões e regras. Os autores destacam que o objetivo do processo de mineração é fornecer as corporações informações que as possibilitem montar melhores estratégias de *marketing*, vendas, suporte, melhorando assim os seus negócios.

A Mineração de Dados trás consigo uma série de idéias e técnicas para uma vasta variedade de campos. Estatísticos, pesquisadores de Inteligência Artificial (IA) e administradores de bancos de dados usam técnicas diferentes para chegar a um mesmo fim, ou seja, a informação.

A diferença entre estas áreas está nos termos utilizados, onde estatísticos vêem variáveis dependentes e independentes, pesquisadores de IA vêem características e atributos e administradores de bancos de dados vêem registros e campos (BERRY e LINOFF, 1997). Independente da linha de pesquisa a que estão ligadas, as técnicas fazem uso de métodos computacionais para a Descoberta de Conhecimento em Bancos de Dados.

2.4.1. Aplicações

Hoje praticamente não existe nenhuma área de conhecimento em que as técnicas de mineração de dados não possam ser usadas. Entretanto existem áreas onde o uso têm sido mais freqüente, FAYYAD *et al.* (1996) citam exemplos de aplicações nestas áreas:

Marketing: redução dos custos com o envio de correspondências através de sistemas de mala direta a partir da identificação de grupos de clientes potenciais, que é feito através do perfil, conforme já comentado no início deste capítulo.

Detecção de fraude: reclamações indevidas de seguro, chamadas clonadas de telefones celulares, compras fraudulentas com cartão de crédito e nomes duplicados em sistemas de Previdência Social.

Investimento: diversas empresas têm usado técnicas de mineração de dados, embora a maioria delas não as revela, para obter ganhos financeiros. São usados especialmente modelos de redes neurais no mercado de ações e na previsão da cotação do ouro e do dólar.

Produção: empresas desenvolvem sistemas para detectar e diagnosticar erros na fabricação de produtos. Estas falhas são normalmente agrupadas por técnicas de Análise de Agrupamentos.

2.4.2. Tarefas desempenhadas

As técnicas de mineração podem ser aplicadas a tarefas (neste contexto, um problema de descoberta de conhecimento a ser solucionado) como associação, classificação, predição/previsão, sumarização e segmentação. A seguir temos uma descrição resumida de cada uma delas (FAYYAD e STOLORZ, 1997):

Associação: consiste em determinar quais fatos ou objetos tendem a ocorrer juntos num mesmo evento ou numa mesma transação.

Classificação: consiste em construir um modelo que possa ser aplicado a dados não classificados visando categorizar os objetos em classes. Todo objeto é examinado e classificado de acordo com a classe definida (HARRISON, 1998).

Predição/Previsão: predição é usada para definir um provável valor para uma ou mais variáveis. A previsão é utilizada quando se têm séries temporais (dados organizados cronologicamente), como por exemplo a previsão da cotação de uma ação na bolsa de valores.

Sumarização: a tarefa de sumarização envolve métodos para encontrar uma descrição compacta para um subconjunto de dados (FAYYAD *et al.*, 1996).

Segmentação: é um processo de partição, que visa dividir uma população em subgrupos mais heterogêneos entre si. É diferente da tarefa de classificação, pois não existem classes predefinidas, os objetos são agrupados de acordo com a similaridade.

2.4.3. Algumas técnicas

HARRISON (1998) afirma que não existe uma técnica que resolva todos os problemas de mineração de dados. Diferentes técnicas servem para diferentes propósitos, cada uma oferecendo vantagens e desvantagens.

Atualmente existem diversas técnicas, algumas delas são ligadas a Inteligência Artificial (IA), como Árvores de Decisão, Regras de Indução e Redes Neurais Artificiais (RNA). Outras possuem maior relação com a Estatística, tais como Análise de Regressão e a Análise Multivariada, dentro delas pode-se citar a Análise de Agrupamento.

A escolha da técnica está fortemente relacionada com o tipo de conhecimento que se deseja extrair ou com o tipo de dado no qual ela será aplicada. A seguir serão apresentadas brevemente algumas das técnicas mais conhecidas e usadas.

2.4.3.1. Árvores de decisão

Segundo BERSON *et al.* (1999) árvore de decisão é um modelo preditivo que pode ser visualizado na forma de uma árvore. Cada ramo da árvore é uma questão de classificação e cada folha é uma partição do conjunto de dados com sua classificação.

A forma de execução é: dado um conjunto de dados cabe ao usuário escolher uma das variáveis como objeto de saída. A partir daí, o algoritmo encontra o fator mais importante correlacionado com a variável de saída e coloca-o como o primeiro ramo (chamado de raiz),

os demais fatores subseqüentemente são classificados como nós até que se chegue ao último nível, chamado de folha, conforme pode ser observado na Figura 2.

Assim, a árvore de decisão utiliza a estratégia de dividir para conquistar, um problema complexo é decomposto em subproblemas mais simples e recursivamente a mesma estratégia é aplicada a cada subproblema.

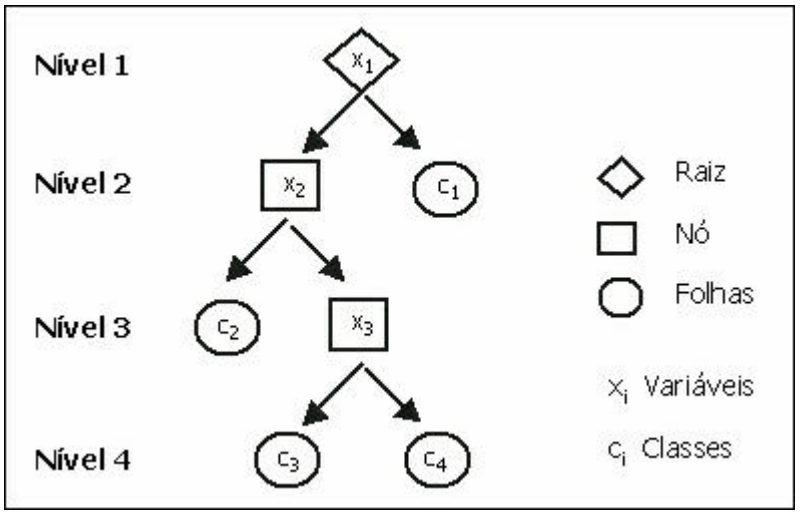

Figura 2: Estrutura de uma árvore de decisão.

Dentre os algoritmos de árvore de decisão, BERSON *et al.* (1999) cita: ID3 (que foi o primeiro algoritmo sobre o assunto), C4.5 (um melhoramento do anterior), CHAID (*Chi-Squared Automatic Induction*) e CART (C*lassification and Regression Trees*).

2.4.3.2. Regras de indução

A técnica de Regras de Indução é altamente automatizada e, possivelmente, é a melhor técnica de mineração de dados para expor todas as possibilidades de padrões existentes em um banco de dados (BERSON *et al*., 1999).

A Regra de Indução consiste em uma expressão condicional do tipo: **se** <condição> **então** <conseqüência>, ou, em outras palavras: **se** <isto> **então** <aquilo>. Por exemplo:

- **se** comprou cereal **então** comprou também leite
- **se** comprou queijo **e** presunto **então** comprou também pão
- **se** comprou café **e** açúcar **então** comprou também leite

Após a formação das regras, se constrói uma tabela com o percentual de **precisão** (com que freqüência a regra está correta?) e de **cobertura** (com que freqüência a regra pode ser usada?). Quanto maior o percentual, melhor a regra.

2.4.3.3. Redes neurais artificiais

Segundo COMRIE (1997), as Redes Neurais Artificiais são técnicas que procuram reproduzir de maneira simplificada as conexões do sistema biológico neural. Estruturalmente, consistem em um número de elementos interconectados, chamados neurônios, organizados em camadas que aprendem pela modificação de suas conexões. Tipicamente, tem-se uma camada de entrada ligada a uma ou mais camadas intermediárias que são ligadas a uma camada de saída (ver Figura 3).

A partir de um conjunto de treinamento, procura-se aprender padrões gerais que possam ser aplicados à classificação ou à predição de dados. A função básica de cada neurônio é avaliar valores de entrada, calcular o total para valores de entrada combinados, comparar o total com um valor limiar e determinar o valor de saída.

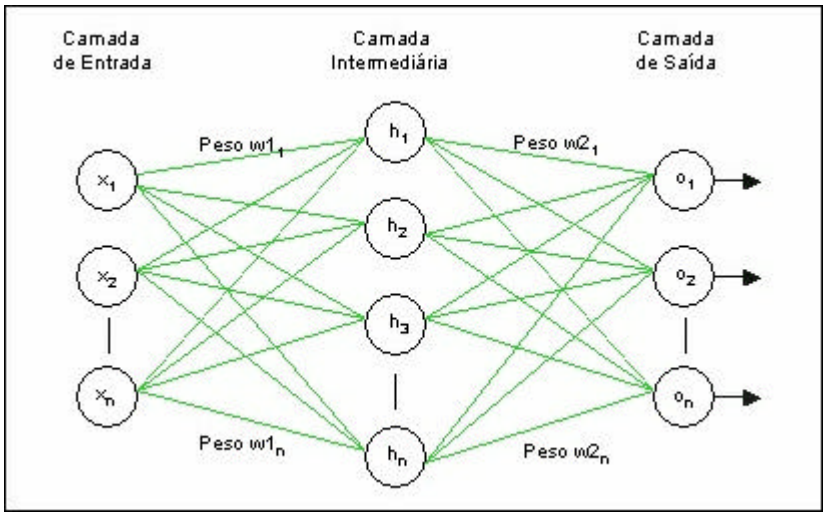

Figura 3: Exemplo de uma rede neural multicamadas.

Dentre os diversos modelos de RNA, os que melhor se adequam a mineração de dados são as redes auto-organizáveis, ou Redes de Kohonen (homenagem a Teuvo Kohonen, pioneiro no desenvolvimento desta teoria).

2.4.3.4. Análise de Regressão

Segundo CHATTERJEE e PRICE (1991), a Análise de Regressão é uma das técnicas mais utilizadas na análise dados. Busca explicar uma ou várias variáveis de interesse (contínuas, ordinais ou binárias) em função de outras. Uma vez construído o modelo, ele pode ser usado para realizar predições, previsões ou calcular probabilidades.

A Análise de Regressão possui basicamente quatro passos: seleção das variáveis regressoras ou preditoras, diagnóstico para verificar se o modelo ajustado é adequado,

aplicação de medidas remediadoras quando as condições do modelo não são satisfeitas e validação do mesmo.

Quando a Regressão for do tipo Linear[3] (sem interação) a equação para um problema com *n* variáveis terá a seguinte expressão:

$$Y = b_0 + \sum_{i}^{p-1} b_i X_i + e_i \qquad (1)$$

onde *Y* é o resultado da variável de interesse no *i*-ésimo nível de *X*, X_i são as variáveis preditoras, *b's* são os parâmetros a serem estimados e e_i são os erros aleatórios.

2.4.3.5. Análise de Agrupamentos

A Análise de Agrupamentos, também chamada de segmentação de dados, possui uma variedade de objetivos. Todos eles dizem respeito a agrupar ou segmentar uma coleção de objetos em subconjuntos, os agrupamentos, onde os objetos dentro de um mesmo agrupamento são mais próximos entre si do que com qualquer outro objeto alocado em outro agrupamento (HASTIE *et al.*, 2001).

Os algoritmos de Análise de Agrupamentos mais conhecidos e utilizados serão estudados no capítulo seguinte.

2.5. Interpretação e avaliação

Esta é mais uma fase que deve ser feita em conjunto com especialistas no assunto. O conhecimento adquirido através da técnica de mineração de dados deve ser interpretado e avaliado para que o objetivo final seja alcançado.

Caso o resultado final seja satisfatório, aplica-se. Caso ocorra o contrário, o que não é raro, o processo pode retornar a qualquer um dos estágios anteriores ou até mesmo ser recomeçado, conforme pode ser observado na Figura 1.

Duas das ações mais comuns caso o resultado não seja satisfatório são: modificar o conjunto de dados inicial e/ou trocar o algoritmo de mineração de dados (ou ao menos alterar suas configurações de entrada).

---

[3] Existe ainda a Regressão Não Linear.

3. ANÁLISE DE AGRUPAMENTOS (*CLUSTER ANALYSIS*)

Segundo HUANG (1997b), Análise de Agrupamentos, ou *Cluster Analysis*, é uma técnica para reunir objetos em grupos, de tal forma que os objetos que estão no mesmo grupo são mais similares entre si do que objetos que estão em outro grupo definido, segundo uma medida de similaridade pré-estabelecida.

Conforme WIVES (1999), o objetivo da AA é identificar os objetos que possuem características em comum e separá-los em subconjuntos similares, determinando o número e as características desses grupos.

HUANG (1997b) vê a partição de objetos em grupos mais homogêneos como sendo uma operação fundamental para mineração de dados, já que a operação é necessária para um grande número de tarefas, como a classificação não supervisionada e a sumarização de dados, bem como na divisão de grandes conjuntos heterogêneos de dados em conjuntos menores, mais fáceis de gerenciar e modelar.

Existem inúmeros algoritmos baseados em medidas de similaridade ou modelos probabilísticos para a formação dos agrupamentos. Conforme BUSSAB *et al.* (1990), os algoritmos de AA exigem de seus usuários a tomada de diversas decisões, fazendo surgir à necessidade do conhecimento das propriedades de cada um deles.

Conhecendo-se os diferentes algoritmos, é possível determinar qual deles é a melhor escolha para atingir os objetivos estabelecidos, fazendo assim com que o tempo e o custo de recuperação diminua.

3.1. Definição formal

HRUSCHKA e EBECKEN (2003) definem formalmente Análise de Agrupamentos da seguinte forma: um conjunto de n objetos $X = \{X_1, X_2, ..., X_n\}$, onde $X_i \in \Re^p$ é um vetor de dimensão p que pode ser agrupado em k agrupamentos disjuntos $C = \{C_1, C_2, ..., C_k\}$, respeitando as seguintes condições:

- $C_1 \cup C_2 \cup ... \cup C_k = X$;
- $C_i \neq \{\}, \forall\, i,\ 1 \leq i \leq k$;
- $C_i \cap C_j = \{\}, \forall\, i \neq j,\ 1 \leq i \leq k$ e $1 \leq j \leq k$;

Ou seja, a união dos subgrupos forma o conjunto original, um objeto não pode pertencer a mais de um agrupamento e cada agrupamento deve possuir ao menos um objeto.

3.2. Características desejadas em mineração de dados

Para que um algoritmo de AA possa ser satisfatoriamente usado para mineração de dados em grandes bases de dados é preciso que ele atenda a uma série de requisitos. ZAÏANE *et al.* (2002) e HAN e KAMBER (2001) destacam:

- Possua escalabilidade: capacidade de aumentar o tamanho da base de dados, número de registros, sem que, com isto, o tempo de execução aumente na mesma proporção. Supondo que o algoritmo leve um tempo *t* para agrupar *n* registros, ao duplicar o tamanho da base o tempo de processamento também irá crescer. Porém, para que seja dito que o algoritmo possui escalabilidade é necessário que o novo tempo fique mais próximo de *t* do que de *2t*. Em outras palavras, é necessário que o aumento de tempo não seja proporcional ao aumento do número de registros;
- Possua alta dimensionalidade: suporte base de dados que possuem milhares de registros e/ou um grande número de colunas (atributos);
- Possa ser usada em base de dados com diferentes tipos de variáveis (atributos), já que bancos de dados do mundo real representam um objeto ou indivíduo através de variáveis dos tipos mais diversos;
- Identifique *clusters* com diferentes formatos;
- Requeira o mínimo de conhecimento sobre o domínio para determinar os parâmetros de entrada;
- Tenha habilidade para lidar com dados incorretos e *outliers*, uma vez que quanto maior o volume de dados maior é a tendência de existirem dados incorretos;
- Não seja sensível a ordem dos registros, ou seja, deve encontrar o mesmo resultado mesmo que os dados sejam apresentados em uma ordem diferente;
- Gere resultado de fácil interpretação.

3.3. Medidas de distância

Um passo importante na utilização da Análise de Agrupamentos é a escolha de um critério que meça a parecença entre dois objetos, ou seja, um critério que diga o quanto dois objetos são semelhantes ou distintos.

Quanto maior o valor encontrado, mais dissimilares são os objetos. Quanto menor o valor, mais similares. A escolha do critério de parecença depende basicamente do tipo de variável envolvida. Para cada tipo existe uma ou mais medidas de similaridade a ser aplicada.

3.3.1. Variáveis intervalares

Variáveis intervalares são mensuradas aproximadamente numa escala linear. Por exemplo: peso, altura, temperatura e coordenadas de latitude e longitude.

O maior problema encontrado neste tipo de variável é que elas geralmente se encontram em unidades diferentes. Por exemplo, peso é expresso em quilograma, altura em metros e temperatura em graus. Conforme HAN e KAMBER (2001), o uso de variáveis com unidades diferentes afetará a análise, uma vez que dois objetos podem ter um valor de similaridade, em termos numéricos, maior ou menor.

Os autores destacam que, antes de tudo, as variáveis sejam padronizadas. Isto pode ser feito através do escore *z* (*z-score*), dado por:

$$z = \frac{x_i - \bar{x}}{s} \qquad (2)$$

onde x_i é o *i*-ésimo elemento de um conjunto com *n* elementos, $\bar{x}$ é a média dada por:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad (3)$$

e *s* é o desvio absoluto dado por:

$$s = \frac{1}{n}\sum_{i=1}^{n} \left| x_i - \bar{x} \right| . \qquad (4)$$

Segundo HAN e KAMBER (2001), o desvio absoluto médio, *s*, é menos sensível a valores *outliers* do que o desvio padrão, σ. Quando calculado o desvio absoluto, o desvio da média ($\left| x_i - \bar{x} \right|$) não é elevado ao quadrado como no desvio padrão (equação 5), portanto os efeitos dos *outliers* são reduzidos.

$$s = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (x_i - \bar{x})^2} \qquad (5)$$

Ainda segundo os autores, existem outras medidas que poderiam ser utilizadas, como o desvio absoluto mediano. Entretanto, a vantagem do uso desta é que os valores *outliers* do escore *z* não se tornam tão pequenos, podendo ainda assim ser detectados. Depois de padronizados, a similaridade entre dois pares de objetos pode ser calculada por qualquer medida de distância. A seguir, são apresentadas algumas delas.

3.3.1.1. Distância Euclideana

Dentre as medidas de distância, a Euclideana é a mais conhecida. Para calcular a distância entre dois objetos (*i* e *j*) com dimensão *p* (número de variáveis), ela é definida por:

$$d(i,j) = \sqrt{\sum_{l=1}^{p} \left(x_{il} - x_{jl}\right)^2} \; . \qquad (6)$$

A distância Euclideana possui uma série de derivações, como: Distância Euclideana Média, Distância Euclideana Padronizada, Coeficiente da Distância Euclideana Padronizada e Distância Euclideana Ponderada.

Conforme BUSSAB *et al.* (1990), uma derivação muito usada em AA é Distância Euclideana Média (DEM), onde a soma das diferenças ao quadrado é divida pelo número de elementos envolvidos:

$$d(i,j) = \sqrt{\frac{\sum_{l=1}^{p} \left(x_{il} - x_{jl}\right)^2}{p}} \; . \qquad (7)$$

Esta medida de distância pode ser usada quando existem dados ausentes (*missing values*), usando somente os "componentes" observados.

3.3.1.2. Outras medidas de distância

Além das medidas de distância já citadas, existem outras medidas que podem ser usadas como, por exemplo, o Valor Absoluto e a distância *Manhattan* (equação 8). Elas não serão discutidas aqui, pois não são comumente utilizadas.

$$d(i,j) = \sum_{l=1}^{p} \left| x_{il} - x_{jl} \right| \qquad (8)$$

Vale mencionar que, além de usar qualquer um dos coeficientes citados, o usuário pode criar sua própria medida de distância, desde que siga quatro premissas básicas, conforme PISON *et al.* (1999) e BERRY e LINOFF (1997). Sejam *i* e *j* quaisquer pontos no espaço:

a. Não existe distância negativa: $d(i,j) \geq 0$;
b. A distância de um objeto a ele mesmo é sempre zero: $d(i,i) = 0$;
c. Distância é uma função simétrica: $d(i,j) = d(j,i)$;
d. A distância em linha reta entre *i* e *j* é menor que qualquer outra que passe por um terceiro ponto *h*, desde que *h* não seja um ponto que pertença a reta que une *i* e *j* (princípio da desigualdade triangular): $d(i,j) \leq d(i,h) + d(h,j)$.

3.3.2. Variáveis binárias

Variáveis binárias são aquelas que assumem somente dois valores possíveis (dois estados): 0 e 1, onde 0 representa que determinada característica não está presente e 1 representa que a característica está presente. Por exemplo, dada a variável *estado da lâmpada*, 0 pode significar que ela está *desligada* e 1 significar que ela está *ligada.*

Há dois tipos de variáveis binárias: as *simétricas* e as *assimétricas*. Uma variável binária simétrica possui os estados equivalentes, ou seja, ambos possuem o mesmo peso. Não há diferença entre a característica estar presente (1) ou ausente (0).

Uma variável binária assimétrica não possui pesos equivalentes para os dois estados, ou seja, existe diferença entre a característica estar presente ou ausente. Este tipo de variável é comumente utilizada para representar a presença ou não de determinada doença, como por exemplo a AIDS. Por convenção, usa-se o valor 1 para o resultado mais importante, no caso, que o paciente possui a doença.

Existem métodos adequados para se calcular a semelhança entre objetos através de variáveis binárias. Um dos enfoques mais comuns para isto é calcular a matriz de dissimilaridade entre os valores dados. Para isto, constrói-se a matriz de contingência para os valores binários conforme a Tabela 1 (JOHNSON e WICHERN, 1982):

Tabela 1: Tabela de contingência para os valores binários

		Objeto *j*		
		1	0	Soma
Objeto *i*	1	*Que*	*r*	*q* + *r*
	0	*S*	*t*	*s* + *t*
	Soma	*q* + *s*	*r* + *t*	*p* = *q* + *r* + *s* + *t*

Fonte: JOHNSON e WICHERN, 1982.

Na Tabela 1, *q* representa o caso onde ambos os objetos possuem a característica em estudo, *r* representa o caso onde o objeto *i* possui a característica em estudo e *j* não possui, *s* representa o caso onde o objeto *i* não possui a característica em estudo e *j* possui e *t* representa o caso onde ambos os objetos não possuem a característica em estudo.

O uso desta tabela depende do tipo de variável binária. Se for simétrica, utiliza-se uma equação que é, na verdade, a distância Euclideana Média:

$$d(i, j) = \frac{r + s}{q + r + s + t}. \qquad (9)$$

Se for assimétrica, utiliza-se a fórmula conhecida como *coeficiente de Jaccard*:

$$d(i, j) = \frac{r + s}{q + r + s} \qquad (10)$$

Como se pode notar, a única diferença está no uso ou não do caso *t*. Isto se deve ao fato de não ser importante considerar, para variáveis binárias assimétricas, o caso onde ambos os objetos não possuem a característica em estudo. Por exemplo, no estudo da AIDS, não é relevante comparar dois indivíduos que não possuem a doença.

Além destas equações, que são as mais utilizadas, JOHNSON e WICHERN (1982) trazem outras que dão menor ou maior peso para determinada situação (ver Tabela 2).

Tabela 2: Coeficientes de similaridades para variáveis binárias

Coeficiente	Significado	Tipo de Variável Binária
$d(i, j) = \frac{q + t}{p}$	Pesos iguais para os casos (1,1) e (0,0).	Simétrica
$d(i, j) = \frac{2(q + t)}{2(q + t) + r + s}$	Pesos dobrados para os casos (1,1) e (0,0).	Simétrica
$d(i, j) = \frac{q + t}{q + t + 2(r + s)}$	Pesos dobrados para os casos (1,0) e (0,1).	Simétrica
$d(i, j) = \frac{q}{q + r + s}$	O caso (0,0) é descartado.	Assimétrica
$d(i, j) = \frac{2q}{2q + r + s}$	O caso (0,0) é descartado e o caso (1,1) possui peso dobrado.	Assimétrica
$d(i, j) = \frac{q}{q + 2(r + s)}$	O caso (0,0) é descartado e os casos (1,0) e (0,1) possuem pesos dobrados.	Assimétrica

Fonte: JOHNSON e WICHERN (1982).

3.3.3. Variáveis nominais

Este tipo de variável é, na verdade, uma generalização do tipo binário, porém ao invés de possuir apenas dois estados, possui um conjunto finito e pequeno de resultados possíveis**.** São exemplos de variáveis nominais: estado civil (solteiro, casado, divorciado, viúvo,...) e cor (azul, verde, preto,...).

HAN e KAMBER (2001) destacam que este tipo possui uma particularidade bastante significativa, embora muitas vezes utilizam-se valores inteiros para representar cada um dos possíveis resultados da variável, os valores expressos não possuem ordem, ou seja, não há como dizer que um é maior ou mais importante que outro, é apenas uma maneira de se trabalhar com os dados.

Segundo os autores, a similaridade entre dois objetos *i* e *j* pode ser medida através de uma forma semelhante a empregada para variáveis binárias:

$$d(i, j) = \frac{p - m}{p} \qquad (11)$$

onde *m* é o número de atributos (variáveis) onde *i* e *j* possuem o mesmo valor (mesmo estado) e *p* é a quantidade de atributos. Os autores destacam que é possível utilizar pesos para maximizar o efeito de *m*.

Conforme BUSSAB *et al.* (1990), outra forma de se calcular a similaridade de variáveis nominais é transformar cada um dos possíveis resultados em variáveis binárias fictícias (*dummies*) e, depois, aplicar a tabela de contingência (Tabela 1). Por exemplo, a variável *cor* seria dividida em outras *n* variáveis (x_1, x_2, ..., x_n) e uma, e somente uma delas, teria seu resultado igual a 1 (presente) para cada registro. As demais seriam todas 0 (ausente).

3.3.4. Variáveis ordinais

As variáveis do tipo ordinal assemelham-se as do tipo nominal, a diferença está no fato de seus resultados possuírem ordem. São exemplos de variáveis ordinais: conceitos acadêmicos (A, B, C, D, E) e graus de preferência (muito bom, bom, regular, ruim e muito ruim) (HASTIE *et al.*, 2001).

Conforme HAN e KAMBER (2001), a técnica utilizada para medir a similaridade entre dois objetos *i* e *j* é semelhante a empregada para variáveis intervalares. Seja *f* uma variável ordinal, primeiro troca-se o rótulo (*label*) da variável pela posição que esta ocupa na seqüência dos estados possíveis (r_f). Depois, é necessário mapear cada valor da variável em um novo valor contido na faixa [0 , 1], para que todas as variáveis tenham o mesmo peso. O mapeamento é feito através de:

$$z_{if} = \frac{r_{if} - 1}{m_f - 1} \qquad (12)$$

onde *i* é o *i*-ésimo objeto e m_f é o número de possíveis estados para a variável *f*.

Por último, calcula-se a similaridade entre os objetos usando qualquer uma das medidas de distâncias utilizadas por variáveis intervalares, porém ao invés de utilizar x_{if} utiliza-se z_{if}.

Conforme visto em BUSSAB *et al.* (1990), aqui também se pode fazer uso das variáveis *dummies* para o cálculo da similaridade, porém com a diferença de que neste caso considera-se que o nível superior possui os níveis inferiores.

Por exemplo, a variável *grau de instrução* (analfabeto, ensino básico, ensino médio e ensino superior) seria dividida em outras quatros variáveis (x_1, x_2, x_3, x_4) e um indivíduo com ensino médio ($x_3 = 1$) seria considerado como portador das características anteriores. Assim temos a seguinte representação vetorial: analfabeto (1, 0, 0, 0), ensino básico (1, 1, 0, 0), ensino médio (1, 1, 1, 0) e ensino superior (1, 1, 1, 1). Após criar as variáveis *dummies* pode-se usar qualquer uma das fórmulas apresentadas no item 3.3.2 - Variáveis binárias.

3.3.5. Variáveis de tipos diferentes

Bancos de dados do mundo real são, normalmente, compostos não por apenas um tipo de variável, mas sim por uma mescla dos diferentes tipos. A literatura traz uma série de técnicas para se trabalhar com tipos mistos, aqui serão apresentadas duas das presentes em BUSSAB *et al.* (1990). Além destas, outras podem ser lidas em HAN e KAMBER (2001) e HASTIE *et al.* (2001).

Para qualquer uma das técnicas apresentadas a seguir, o primeiro passo a ser feito é o reagrupamento das variáveis de modo que apareçam primeiro as nominais, depois as ordinais e, finalmente, as intervalares. Esquematizando, tem-se:

$$y' = (yn_1, yn_2, ..., yn_p)(yo_1, yo_2, ..., yo_p)(yq_1, yq_2, ..., yq_p) \qquad (13)$$

onde yn_i representa as variáveis nominais, yo_i as ordinais e yq_i as intervalares.

Após o reagrupamento, são construídos os coeficientes de parecença para cada um dos tipos utilizando algumas das técnicas já discutidas nas seções 3.3.1 a 3.3.4. Após o cálculo dos coeficientes, estes são agrupados em um único coeficiente ponderado:

$$c(A,B) = w_1.cn(A,B) + w_2.co(A,B) + w_3.cq(A,B) \qquad (14)$$

onde *w*'s são os pesos, e *cn*, *co* e *cq* são os coeficientes para as variáveis nominais, ordinais e intervalares, respectivamente, de dois objetos quaisquer A e B.

Os autores destacam que a construção destes coeficientes exige alguns cuidados, como:

- Os coeficientes devem ter o mesmo sentido (todos devem expressar a similaridade ou todos devem expressar a dissimilaridade, não pode haver uma mistura);
- Mesmos intervalos de variação;
- Conjunto de pesos adequados e interpretáveis.

Realizados estes dois passos, pode-se utilizar qualquer uma das duas técnicas que serão apresentadas a seguir.

3.3.5.1. Transportar todos os valores para a faixa [0,1]

Uma vez reorganizadas as variáveis e criados os coeficientes, um dos possíveis métodos de se medir a similaridade entre os objetos é transportar todos os valores das suas variáveis para a faixa [0,1] pode-se usar a distância Euclideana. As variáveis binárias não necessitam ser transformadas, pois já estão nesta faixa exigida, as variáveis nominais e ordinais podem ser transformadas utilizando os critérios já discutidos nas seções 3.3.3 e 3.3.4, respectivamente. As variáveis intervalares podem ser transformadas de seguinte forma:

$$z = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \qquad (15)$$

onde x é o valor da variável, x_{min} e x_{max} são respectivamente o menor e o maior valor que esta variável assume.

3.3.5.2. Transformar todas as variáveis em binárias ou ordinais

Esta técnica simplifica o trabalho, entretanto, a transformação em variáveis binárias traz a desvantagem da perda de informações. Por exemplo, a variável *peso* pode ser reclassificada como MAGRO ou GORDO (usa-se como critério de classificação a mediana). Para minimizar a perda de informação, podem ser usados intervalos, através de variáveis do tipo ordinal. Uma vez transformadas as variáveis, os coeficientes são calculados conforme os métodos citados em 3.3.2 - Variáveis binárias.

3.4. Métodos de formação do agrupamento

Quanto ao método de formação do agrupamento, os algoritmos podem ser classificados em *hierárquicos, de partição* (que são os dois métodos mais tradicionais), *baseados em modelos, baseados em grades* e *baseados em densidade* (que são os métodos mais modernos). A escolha do método a ser utilizado na análise depende do tipo de variável e do propósito da aplicação.

3.4.1. Métodos hierárquicos

Neste método o processo de identificação de grupos é geralmente realimentado recursivamente, utilizando tanto objetos quanto grupos já identificados previamente como entrada para o processamento. Deste modo, constrói-se uma hierarquia de grupos de objetos, no estilo de uma árvore (DINIZ e LOUZADA NETO, 2000).

Os métodos hierárquicos possuem algumas características significativas:

- São subdivididos em dois tipos: *aglomerativos* e *divisivos*. Nos métodos aglomerativos, uma vez que dois elementos são unidos eles permanecem unidos até o final do processo. Nos métodos divisivos acontece justamente o contrário, ou seja, uma vez que dois elementos são separados eles jamais voltarão a fazer parte do mesmo agrupamento.
- Por ser um método hierárquico, é possível saber de onde um determinado elemento veio, em outras palavras, o histórico de onde cada elemento estava nos passos anteriores não é perdido.
- MICHAUD (1997), destaca que este tipo de método possui uma desvantagem bastante significativa, referindo-se ao fato de serem impraticáveis para grandes bases de dados devido ao seu alto custo computacional.
- Métodos hierárquicos são utilizados para agrupar variáveis e registros, porém alguns métodos hierárquicos trabalham melhor quando agrupam apenas casos (JOHNSON e WICHERN, 1982).

3.4.1.1. Métodos hierárquicos aglomerativos (*bottom-up*)

Neste método, inicialmente cada um dos n objetos é considerado como sendo um agrupamento. A cada passo, através de sucessivas fusões, vão se obtendo $n - 1$, $n - 2$, ..., agrupamentos, até que todos os objetos estejam em um único grupo.

Conforme visto em BERRY e LINOFF (1997), no início do processo os agrupamentos são muito pequenos e "puros", pois os membros são poucos mas são fortemente relacionados. Já mais em direção ao fim do processo, os agrupamentos são grandes e pouco definidos.

A execução se dá da seguinte forma: dado um conjunto contendo n elementos, o que significa n agrupamentos inicialmente, é calculada a distância entre todos os elementos, agrupando-se os dois mais similares, o que faz com que o conjunto passe a ter $n - 1$ agrupamentos.

A partir daí, calcula-se o centróide[4] do agrupamento recém formado e a distância entre este centróide e os elementos que ainda não foram agrupados, unindo novamente os

[4] Centróide, também chamado de *semente* por alguns autores, é o ponto cujo valor é a referência dos demais elementos do agrupamento onde ele se encontra, normalmente a referência utilizada é a média dos valores. Por exemplo, se fosse feito o agrupamento por idade da população de um edifício, e o conjunto formado pelos

elementos mais semelhantes. Este passo se repete até que todos os elementos estejam juntos, formando um único agrupamento.

É importante frisar que não é necessário recalcular as distâncias entre os elementos que não foram agrupados, pois isto já foi feito e elas não mudaram. Desta forma, o desempenho computacional do método melhora a cada junção, uma vez que a dimensão da matriz diminui.

Para entender melhor o processo descrito, pode-se imaginar um conjunto hipotético contendo 5 elementos {A, B, C, D, E}. Deseja-se agrupar estes elementos de acordo com o número de vezes em que compraram em determinada loja no último ano. O número de compras é representado pelo conjunto {0, 2, 4, 5, 8}. Um possível cenário de execução é demonstrado a partir de agora.

Passo 1: É calculada a distância entre todos os objetos (usaremos a Euclideana neste exemplo). Este cálculo forma a matriz de distância, apresentada a seguir:

	A	**B**	**C**	**D**
B	2	--	--	--
C	4	2	--	--
D	5	3	1	--
E	8	6	4	3

Passo 2: Os dois objetos mais similares são agrupados e o valor do centróide do grupo é calculado (usaremos a média aritmética). Os elementos C e D são os mais similares, já que $d(4,5) = 1$ e o novo centróide é 4.5. Assim, o novo agrupamento é formado e o conjunto inicial é reduzido em um elemento {A, B, CD, E}.

Passo 3: São calculadas as distâncias de A, B e E em relação a novo elemento (CD). Não é necessário recalcular as demais distâncias já que isto foi feito no primeiro passo. A seguir, tem-se a nova matriz de distância, onde a linha grifada mostra que apenas três distâncias foram calculadas, as demais são inalteradas:

	A	**B**	**E**
B	2	--	--
E	8	6	--
CD	***4.5***	***2.5***	***3.5***

---

valores {21, 26, 22} representa-se à idade de um dos possíveis grupos, então o valor do centróide deste grupo seria 23 (média aritmética das idades).

Passo 4: Os dois objetos mais semelhantes, A e B, são agrupados e o valor do centróide é recalculado. O conjunto é novamente reduzido em uma unidade.

Passo 5: As novas distâncias são calculadas.

	CD	**E**
E	3.5	--
AB	***3.5***	***7***

Passo 6: Como ocorreu um empate, já que $d(4.5, 1) = d(4.5, 8)$, a escolha dos elementos a serem unidos dependerá do método de formação escolhido, supomos ser CD e E os escolhidos. O conjunto que inicialmente possuía 5 agrupamentos, possui agora apenas dois, um formado pelos elementos {C, D, E} e outro formado pelos elementos {A, B). No próximo passo, o conjunto de elementos estará todo em um único agrupamento.

A Figura 4 mostra o fluxo de execução para o exemplo apresentado. O quadrado destacado representa cada novo grupo formado. As linhas contínuas representam as medidas de distância calculadas no primeiro passo. As linhas tracejadas representam as medidas calculadas no segundo passo. As medidas de distância dos demais passos não foram representadas para não "poluir" a imagem. Observando-se estas linhas, é possível ver claramente que o número de medidas de distância diminui significativamente a cada passo do processo.

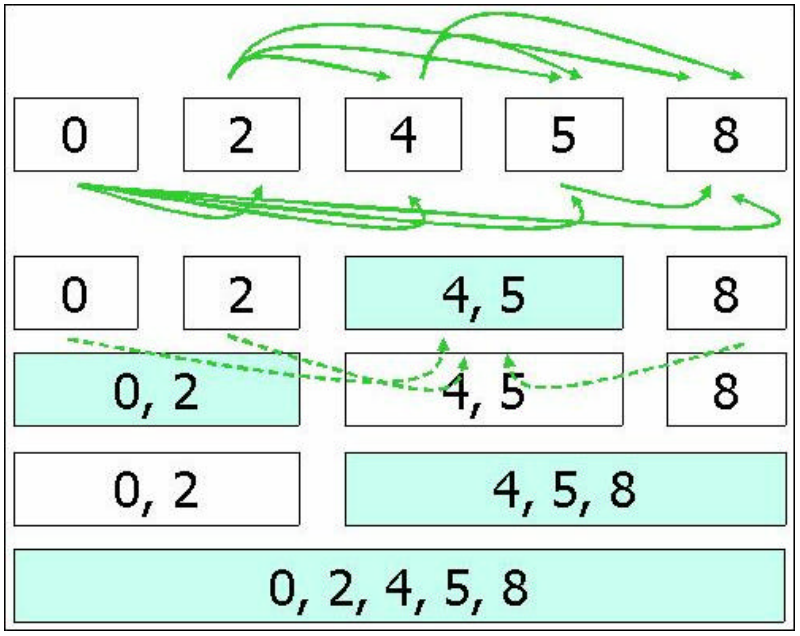

Figura 4: Fluxo de execução para o conjunto {0, 2, 4, 5, 8} utilizando o método hierárquico aglomerativo.

3.4.1.2. Métodos hierárquicos divisivos (*top-down*)

Como o próprio nome sugere, o método hierárquico divisivo é exatamente o oposto do método hierárquico aglomerativo, ou seja, inicialmente os *n* objetos são considerados como sendo um único agrupamento. A cada passo, através de sucessivas divisões, vão sendo obtidos 2, 3, ..., *n* agrupamentos.

MICHAUD (1997), comenta que os métodos divisivos tendem a ser menos utilizados que os métodos aglomerativos, pois não conseguem recuperar facilmente uma partição feita por uma má escolha.

3.4.1.3. Algoritmos hierárquicos

A seguir são descritos os algoritmos hierárquicos utilizados nos testes.

3.4.1.3.1. AGNES

Conforme KAUFMAN e ROUSSEEUW (1990), o AGNES (*AGglomerative NESting*), é um algoritmo baseado no método hierárquico aglomerativo, ou seja, no início cada objeto forma um agrupamento e a cada nova interação os agrupamentos mais próximos são unidos, formando um só, de acordo com um critério pré-estabelecido.

Entre os critérios de junção é possível citar o que une os agrupamentos de acordo com a média da dissimilaridade (*average linkage*) entre os pontos de um agrupamento e outro, o método do vizinho mais próximo (*single linkage*) que usa a menor distância entre os dois agrupamentos e o método do vizinho mais longe (*complete linkage*) que usa a maior distância entre os dois agrupamentos.

Comparado a outros algoritmos aglomerativos, o AGNES apresenta as duas vantagens: (1) utiliza um coeficiente que mede a quantia de estruturas de agrupamentos descobertas, que procura minimizar as buscas e (2) a partir da árvore gráfica usualmente usada para representá-lo é possível prover novas representações.

3.4.1.3.2. DIANA

O DIANA (*DIvisive ANAlysis*) é um algoritmo hierárquico divisivo, ou seja, no início todos os objetos estão no mesmo agrupamento. A cada interação o agrupamento é divido em outros dois, de acordo com um critério pré-definido (os mesmos do AGNES), até que cada agrupamento contenha apenas uma observação (KAUFMAN e ROUSSEEUW, 1990).

A escolha de qual agrupamento dividir se dá a cada etapa do processo, sendo selecionado sempre o agrupamento que tiver o maior diâmetro (maior dissimilaridade entre qualquer duas de suas observações). Para dividir o agrupamento selecionado, o algoritmo primeiro procura pela observação mais dissimilar dentro do grupo, esta observação será o primeiro elemento do novo agrupamento. A seguir, ele reagrupa as observações que porventura estejam mais próximas do novo grupo do que do grupo original. O resultado do processo é a divisão em dois novos grupos.

3.4.1.4. Forma de representação

A forma mais comum de representar os *clusters* originados por métodos hierárquicos, tanto aglomerativos como divisivos, é através de um gráfico bidimensional em forma de árvore conhecido como dendrograma. A Figura 5 apresenta o dendrograma do exemplo dado em Métodos Hierárquicos Aglomerativos.

O dendrograma, segundo PISON *et al.* (1999), é um tipo de árvore onde cada um dos nós é um objeto de dado que pode ser percebido como um *cluster*. O dendrograma ilustra as uniões ou divisões em cada um dos seus sucessivos níveis.

BUSSAB *et al.* (1990), diz que o dendrograma pode ser considerado como a representação simplificada da matriz de similaridade, onde a escala vertical indica o nível de similaridade e a escala horizontal indica os objetos, numa ordem conveniente. A altura das linhas verticais que partem dos objetos corresponde ao nível em que os objetos são considerados semelhantes.

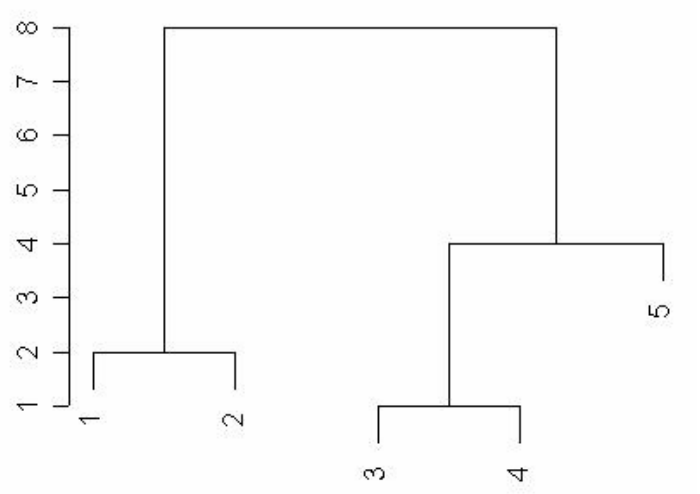

Figura 5: Dendrograma do exemplo de Método Hierárquico Aglomerativo.

3.4.2. Métodos de partição

Os algoritmos de partição buscam de forma direta pela divisão ótima (ou aproximadamente ótima) dos *n* elementos sem a necessidade de associações hierárquicas (DINIZ e LOUZADA NETO, 2000). Servem exclusivamente para agrupar os *n* registros em *k clusters* (JOHNSON e WICHERN, 1982). O método mais conhecido é o método das k-médias (*k-means*).

3.4.2.1. Algoritmo das *k*-médias (*k-means*)

Proposto pode J. MacQueen em 1967, este é um dos algoritmos mais conhecidos e utilizados, além de ser o que possui o maior número de variações (DINIZ e LOUZADA NETO, 2000). O algoritmo inicia com a escolha dos *k* elementos que formaram as sementes iniciais. Esta escolha pode ser feita de muitas formas, entre elas:

- selecionando as *k* primeiras observações;
- selecionando *k* observações aleatoriamente;
- escolhendo *k* observações de modo que seus valores sejam bastante diferentes. Por exemplo, ao se agrupar uma população em três grupos de acordo com a altura dos indivíduos, poderia se escolher um indivíduo de baixa estatura, um de estatura mediana e um alto.

Escolhidas as sementes iniciais, é calculada a distância de cada elemento em relação às sementes, agrupando o elemento ao grupo que possuir a menor distância (mais similar) e recalculando o centróide do mesmo. O processo é repetido até que todos os elementos façam parte de um dos *clusters*.

Após agrupar todos os elementos, procura-se encontrar uma partição melhor do que a gerada arbitrariamente. Para isto, calcula-se o grau de homogeneidade interna dos grupos através da *Soma de Quadrados Residual (SQRes),* que é a medida usada para avaliar o quão boa é uma partição. A SQRes é dada por:

$$SQ\,\mathrm{Re}\,s(j) = \sum_{i=1}^{n_j} d^2\left(o_i(j), \bar{o}(j)\right) \qquad (16)$$

onde $o_i(j)$, $\bar{o}(j)$ e n_j são, respectivamente, os valores do *i*-ésimo registro, o centro do grupo *j* e o número de registros do mesmo.

Após o cálculo, move-se o primeiro objeto para os demais grupos e verifica-se se existe ganho na Soma de Quadrados Residual, ou seja, se ocorre uma diminuição no valor da

SQRes. Existindo, o objeto é movido para o grupo que produzir o maior ganho, a SQRes dos grupos é recalculada e passa-se ao objeto seguinte. Após um certo número de iterações ou não havendo mais mudanças, o processo é interrompido.

A Figura 6 mostra a execução do algoritmo das *k*-médias para formar dois agrupamentos a partir da população composta pelos elementos {2, 6, 9, 1, 5, 4, 8}. Pode-se observar que como sementes foram escolhidos os dois primeiros e como critério para definir o valor do centróide após a união foi usada a média. Na figura, *c1 e c2* apresentam os valores dos centróides de cada um dos agrupamentos após a adição de um novo elemento.

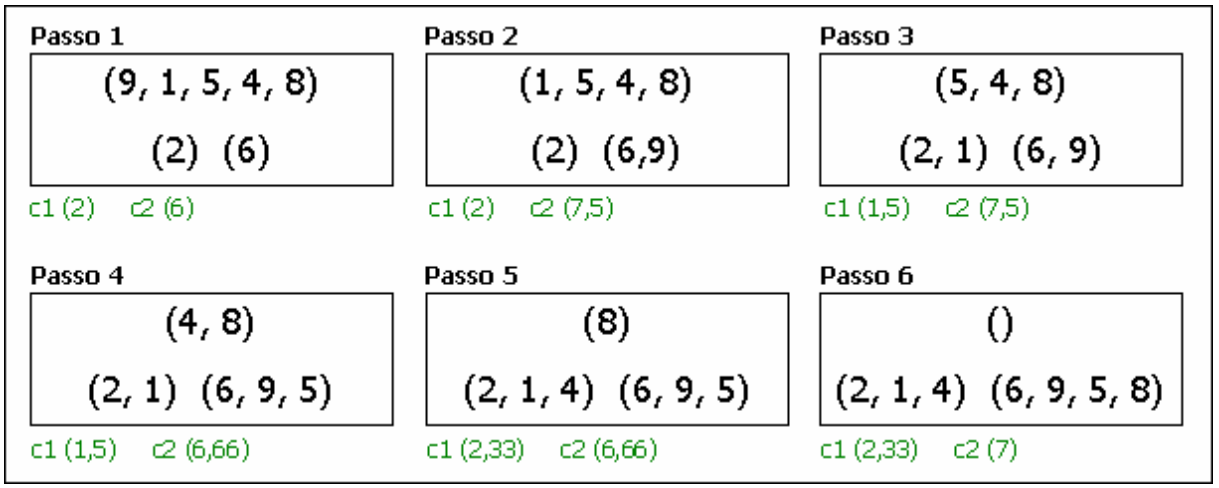

Figura 6: Exemplo de execução do algoritmo de *k-means*.

O algoritmo de *k-means* é bastante escalar e confiável, porém apresenta alguns problemas. Os dois principais problemas são:

- Exige que as variáveis sejam numéricas ou binárias (HUANG, 1997a). Freqüentemente aplicações envolvem dados categorizados, neste caso, uma alternativa é converter os dados categorizados em valores numéricos, conforme já discutido anteriormente, ou utilizar uma das muitas variações do método.
- É sensível a valores *outliers*, um único objeto com valor muito extremo pode modificar, substancialmente, a distribuição dos dados (HAN e KAMBER, 2001).

3.4.2.2. Algoritmo baseado no objeto representativo (*k-medoid*)

O funcionamento do algoritmo *k-medoid* é semelhante ao do de *k-means*, com exceção de que ao invés de utilizar o valor médio dos objetos do agrupamento como ponto referência, é utilizado o objeto mais centralmente localizado (aquele cujo valor mais se aproxima da média). Com isto, a sensibilidade a valores *outliers* diminui (ANDRITSOS, 2002).

Conforme HAN e KAMBER (2001), o algoritmo começa encontrando, arbitrariamente, um objeto representativo (*medoid*) para cada agrupamento. Os objetos remanescentes são agrupados de acordo com o *medoid* ao qual for mais similar. Sempre que um objeto é agrupado ocorre a troca do *medoid* do grupo por um dos não *medoids*, o que faz com que a qualidade do agrupamento resultante seja melhorada. A qualidade é estimada usando uma função custo que mede a dissimilaridade média entre um objeto e o *medoid* do seu agrupamento.

Vale destacar que a escolha do *k* objetos que formaram os *medoids* iniciais dos agrupamentos, primeiro passo do algoritmo, pode-se feita de qualquer uma das formas já descritas no algoritmo de *k-means*.

3.4.3. Métodos baseados em densidade

Nos métodos baseados em densidade, um agrupamento é uma região que tem uma densidade maior de objetos do que outra região vizinha. As regiões de baixa densidade, geralmente formadas por dados *outliers*, separam um agrupamento de outro.

A idéia chave destes métodos é que, para cada objeto de um grupo, deve existir um número mínimo (*MinPts*) de outros objetos em um dado raio (*r*), ou seja, o número de objetos que circulam a este agrupamento tem que exceder a algum limite (o que forma a "densidade").

O método inicia sua execução por um objeto arbitrário e, se sua vizinhança satisfaz o mínimo de densidade, inclui o objeto e os que estão em sua vizinhança no mesmo agrupamento. O processo é então repetido para os novos pontos adicionados.

A principal vantagem dos métodos baseados em densidade é que eles podem descobrir grupos com formas arbitrárias, tais como cilíndrica ou elíptica, e não necessitam da informação inicial do número de clusters a serem formados (ZAÏANE *et al.*, 2002). O algoritmo baseado em densidade que foi utilizado nos testes é apresentado a seguir.

3.4.3.1. DBSCAN

Conforme HAN e KAMBER (2001), o algoritmo DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) descobre agrupamentos com forma arbitrária em bases de dados espaciais com ruídos, ou seja, encontra regiões densas que são separadas por regiões de baixa densidade e agrupa os objetos na mesma região densa.

Iniciando por um objeto qualquer, o que o torna insensível a ordem dos elementos (ANDRITSOS, 2002), o método encontra agrupamentos verificando a vizinhança *r* de cada

ponto da base de dados. Se a vizinhança *r* de um ponto qualquer contém mais do que *MinPts,* então um novo agrupamento é criado e este ponto passa a ser um centro. Então, iterativamente o método coleta objetos alcançáveis por densidade diretamente destes centros até que nenhum novo ponto possa ser adicionado a qualquer agrupamento.

Diferentemente dos métodos de partição, que exigem como entrada o número de agrupamentos a serem formados, este método exige apenas *r* e *MinPts.* O problema é que em bancos de dados do mundo real estes parâmetros são normalmente definidos empiricamente, como o algoritmo é sensível aos seus valores, ajustes ligeiramente diferentes podem resultar na união de dados muito diferentes.

O desempenho do algoritmo depende da utilização ou não de um índice indexador de bases espaciais. Independente da utilização ou não do indexador, seu custo computacional é comparável ao de um método hierárquico, já que necessita comparar todos os elementos presentes na base de dados.

3.4.4. Métodos baseados em grades

Os métodos baseados em grades dividem o espaço de objetos em um certo número de células. Estas por sua vez são divididas em outras e assim sucessivamente, formando diferentes níveis de resolução. É através destas células que os objetos são agrupados. Desta forma, tem-se uma estrutura hierárquica que pode ser observada na Figura 7.

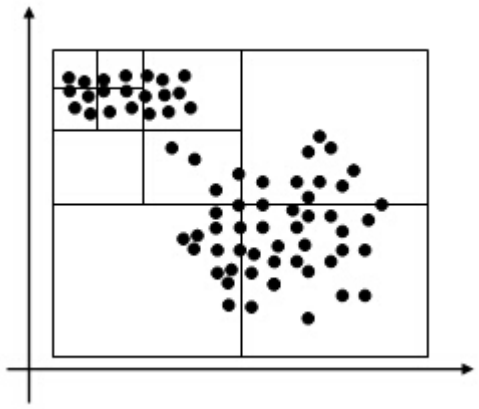

Figura 7: Exemplo da divisão feita pelos métodos baseados em grades.

A principal vantagem destes métodos é que sua velocidade depende apenas da resolução da grade e não do tamanho da base de dados. Por causa disto, são apropriados para base de dados com grande densidade, ou seja, com um número muito grande de objetos num espaço

limitado (ZAÏANE *et al*., 2002). Por não ter sido encontrado nenhum algoritmo baseado em grade na linguagem *R*, os mesmos não serão citados aqui.

3.4.5. Métodos baseados em modelos

Os métodos baseados em modelos procuram justar algum modelo matemático aos dados. Os métodos são freqüentemente baseados na suposição de que os dados são gerados a partir de uma mistura de distribuições de probabilidades e seguem uma das duas principais abordagens: estatística ou por rede neural. Este método ainda é pouco utilizado, principalmente pelo tempo de processamento ser bastante longo (HAN e KAMBER, 2001).

3.4.5.1. Abordagem estatística

A abordagem estatística utiliza uma forma de agrupamento via aprendizado de máquina onde, dado um conjunto de objetos não agrupados, é construído um esquema de classificação sobre os objetos, este processo é chamado de *agrupamento conceitual*.

Ao contrário das formas de agrupamento convencionais estudadas até o momento, que antes de tudo identificavam os grupos de objetos, o agrupamento conceitual realiza uma etapa adicional para encontrar descrições das características de cada grupo (que representa um conceito ou classe). Como pode se notar é um processo composto por duas etapas: primeiro, o agrupamento, depois a caracterização.

3.4.5.2. Abordagem por rede neural

Esta abordagem tende a representar cada agrupamento como um *exemplar*, que serve de protótipo do agrupamento e não tem necessariamente correspondência com dado ou um objeto. Novos objetos podem ser distribuídos para agrupamentos cujo exemplar é mais similar, baseado em alguma medida de distância. Além disto, atributos de um objeto atribuído a um agrupamento podem ser preditos dos atributos do exemplar do agrupamento, otimizando assim a execução do algoritmo.

A abordagem por rede neural possui variações com dois conceitos diferentes de RNA: aprendizado competitivo e mapas auto-organizáveis.

O aprendizado competitivo envolve uma arquitetura hierárquica com vários neurônios artificiais, que competem entre si de forma que o "vencedor leva tudo" para o objeto que está sendo apresentado ao sistema.

A Figura 8 apresenta um exemplo de sistema de aprendizado competitivo. Cada círculo representa uma unidade e o número de camadas é arbitrário. A unidade vencedora em um agrupamento torna-se ativa (representada por círculos cheios), enquanto as demais se tornam inativas (representada por círculos vazios).

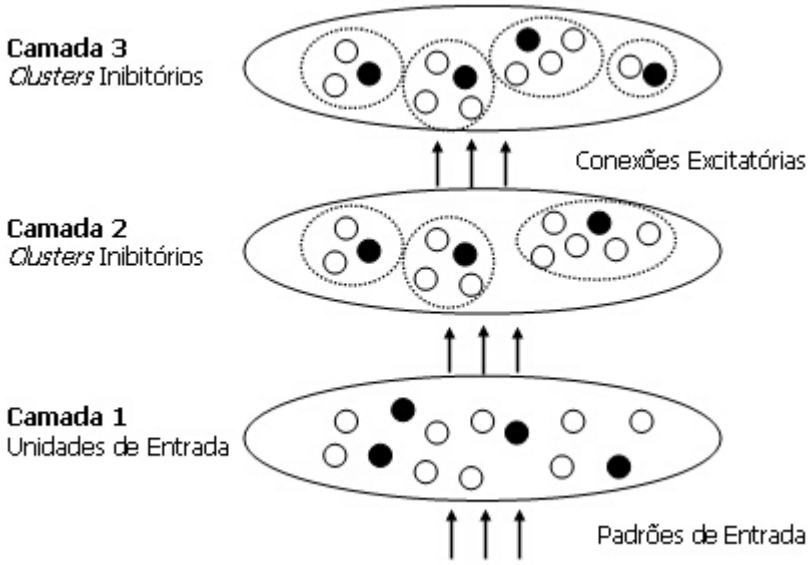

Figura 8: Arquitetura para RNA de aprendizado competitivo. Adaptação de HAN e KAMBER (2001).

As conexões entre as camadas são excitatórias, uma unidade em uma dada camada pode receber entradas de todas as unidades do nível mais próximo abaixo. As conexões entre unidades nas camadas são inibitórias, onde somente uma unidade no agrupamento pode estar ativa. A unidade vencedora ajusta os pesos de suas conexões entre as unidades no agrupamento que irão responder mais fortemente aos objetos futuros que são os mesmos ou similares ao corrente. No término do processo, os agrupamentos resultantes podem ser vistos como um mapeamento das características dos níveis mais baixos para as características dos níveis mais altos.

Com os mapas auto-organizáveis, o agrupamento também se dá com várias unidades competindo pelo objeto corrente. A unidade cujo vetor de peso é mais próxima ao objeto corrente torna-se uma unidade vencedora (unidade ativa). Seus pesos e também os de seus vizinhos mais próximos são ajustados de forma que estas unidades fiquem mais próximas do objeto de entrada.

Os mapas auto-organizáveis assumem que existe alguma topologia ou ordenamento entre os objetos de entrada, e que as unidades vão assumir esta estrutura no espaço. Seu

processamento é semelhante ao que ocorre no cérebro e são úteis para a visualização de dados de alta dimensão em espaços de duas e três dimensões.

3.4.5.3. BIC

O algoritmo BIC (*Bayesian Information Criterion*) baseia-se no conceito de densidades de mistura finita, assim definido: sejam y_1, ..., y_n observações multivariadas independentes. A máxima verossimilhança para um modelo de mistura com G componentes é:

$$L_M(q_1,...,q_G,t_1,...,t_G \mid y) = \prod_{i=1}^{n}\sum_{k=1}^{G} t_k f(y_i \mid q), \qquad (17)$$

onde f_k são as densidades, q_k os parâmetros da k-ésima componente na mistura e t_K é a probabilidade de uma observação pertencer a k-ésima componente.

O algoritmo faz uso do fator de Bayes na escolha do melhor modelo:

$$B_{12} = \frac{p(D \mid M_1)}{p(D \mid M_2)}, \qquad (18)$$

onde $p(D|M)$ é a verossimilhança integrada do modelo M_k obtida por:

$$p(D \mid M_k) = \int p(D \mid q_{k,} M_k) p(q_k \mid q_{k,} M_k) dq_k \qquad (19)$$

sendo $p(D|M)$ a distribuição a priori de q_k e M_k um modelo com probabilidade $p(M_k)$.

O BIC, que é na verdade o valor da máxima verossimilhança com uma penalização para o número de parâmetros no modelo, permite comparar modelos com diferentes parametrizações e/ou diferentes número de agrupamentos. Através dele, o algoritmo determina o provável modelo a ser usado de acordo uma aproximação baseada no critério de informação *Bayesiana*. O detalhamento completo das equações e do funcionamento do algoritmo está presente em FRALEY e RAFTERY (2002).

3.5. Outros Algoritmos de AA

Além dos algoritmos de AA já citados aqui, existem outros que não foram detalhados por não estarem implementados na linguagem *R.* A Tabela 3 apresenta alguns destes algoritmos, indicando o método de formação, comentários sucintos e referências bibliográficas onde podem ser encontradas maiores informações sobre os mesmos.

Tabela 3: Outros algoritmos de Análise de Agrupamento.

Algoritmo	Método de Formação	Comentários	Referências Bibliográficas
CURE	Hierárquico	O CURE (*Clustering Using REpresentatives*) gera bons resultados com agrupamentos não-esféricos e com dados *outliers*. Apresenta uma técnica nova, ao invés de usar um centróide ou algum objeto representativo, ele opta por utilizar um número fixo de pontos do espaço escolhido. Tais pontos indicam a forma e a extensão do agrupamento.	HAN e KAMBER (2001).
BIRCH	Hierárquico	O BIRCH (*Balanced Iterative Reducing and Clustering Using Hierarchies*) representa um agrupamento de forma resumida em três valores: o número de pontos, a soma linear dos pontos e a soma quadrada dos pontos dados. Com isto o espaço de memória RAM necessário é reduzido e o cálculo das medidas torna-se mais rápido.	HAN e KAMBER (2001).
k-prototypes	Partição	É baseado no algoritmo de *k-means*, entretanto, ao invés de utilizar apenas variáveis numéricas para calcular o centróide, utiliza também variáveis nominais e ordinais, formando assim um objeto chamado pelo autor de *prototype*.	HUANG (1997a), HUANG (1997b).
DENCLUE	Baseado em Densidade	O DENCLUE (*DENsity-based CLUstering*) possui sólido fundamento matemático e boas propriedades para agrupar dados com grandes quantidades de *outliers*.	HAN e KAMBER (2001).

STING	Baseado em Grade	O STING (*STatistical INformation Grid*) utiliza as células criadas a partir da divisão da base de dados para armazenar informações sobre os objetos que estão nelas, tais como: valor mínimo, valor máximo, média, desvio padrão, tipo de distribuição (normal, uniforme, exponencial), quantidade de objetos, variância, entre outros. O algoritmo é eficiente para grandes volumes de dados, uma vez que examina a base de dados uma única vez para computar os parâmetros estatísticos das células.	HAN e KAMBER (2001).

4. MATERIAIS E MÉTODOS

Este capítulo apresenta as bases de dados utilizadas nos testes, o *software*, o *hardware* e a metodologia empregada nos mesmos.

4.1. Materiais

Para comparar os algoritmos de AA em mineração de dados foram utilizadas duas bases de dados, uma delas contendo dados simulados e outra com dados reais.

4.1.1. Base de dados simulada

Utilizou-se a linguagem *SPSS* para simular uma base de dados com três variáveis (A, B e C), contendo duzentos e dez mil registros (casos), divididos em três grupos bem definidos. Os casos foram gerados de acordo com uma distribuição normal multivariada, com diferentes médias e desvios para os grupos 1, 2 e 3:

- Grupo 1: 70 mil casos, $\mu_A = \mu_B = \mu_C = 1$ e $\sigma_A = \sigma_B = \sigma_C = 0,5$.
- Grupo 2: 70 mil casos, $\mu_A = \mu_B = \mu_C = 3$ e $\sigma_A = \sigma_B = \sigma_C = 0,1$.
- Grupo 3: 70 mil casos, $\mu_A = \mu_B = \mu_C = 5$ e $\sigma_A = \sigma_B = \sigma_C = 0,5$.

A matriz de correlação usada para a geração dos dados foi:

	A	**B**	**C**
A	1	0,5	0,6
B	0,5	1	07
C	0,6	0,7	1

4.1.2. Base de dados real

A base de dados escolhida para este trabalho pertence à prefeitura da cidade de Belém, capital do Estado do Pará, e contêm trezentos e cinqüenta e três mil registros com informações sobre os imóveis (casas, apartamentos, lojas, galpões, telheiros e salas) existentes na cidade.

Um imóvel pode ser classificado de acordo com as características dos materiais empregados em sua construção, tal classificação é normalmente chamada de *padrão de acabamento* ou *padrão da edificação*. Esta classificação se dá por meio da utilização de faixas de escores: para cada material utilizado atribui-se um peso, a soma dos pesos de todos os

materiais forma a pontuação do imóvel. A prefeitura de Belém classifica os imóveis em seis grupos: 1- Luxo, 2- Alto, 3- Médio, 4- Popular, 5- Baixo e 6- Primário.

Por serem imóveis de tipos diferentes, os materiais utilizados na formação do escore também são diferentes. Devido a este fato, optou-se por utilizar neste trabalho apenas os registros que identificavam imóveis do tipo CASA, com isto a base de dados passou a ter um tamanho de pouco duzentos e setenta mil registros.

Das mais de cinqüentas variáveis utilizadas para identificar um imóvel, apenas seis são utilizadas para a formação do escore: Característica, Cobertura, Elemento, Dejetos, Água e Lixo. As variáveis são todas do tipo ordinal e os valores que cada uma delas podem assumir estão na Tabela 4.

Tabela 4: Valores possíveis para cada uma das variáveis.

Variável	Valores Possíveis
Característica	1- Isolada, 2- Isolada / Recuada, 3- Geminada.
Cobertura	1- Cerâmica, 2- Amianto, 3- Laje, 4- Metálica, 5- Palha, 6- Mista.
Elemento	1- Concreto, 2- Alvenaria, 3- Concreto / Alvenaria, 4- Madeira, 5- Madeira / Alvenaria, 6- Metálica, 7- Taipa.
Dejetos	1- Esgoto, 2- Fossa séptica, 3- Fossa negra, 4- Via pública, 5- Outros.
Água	1- Caixa com tampa, 2- Caixa sem tampa, 3- Sem caixa d'água.
Lixo	1- Sempre embalado, 2- Ocasionalmente embalado, 3- Não embalado.

Além das variáveis citadas, outras duas também fazem parte da base de dados: inscrição imobiliária (identifica o imóvel) e padrão da edificação (classificação do imóvel de acordo com o escore atribuído pela prefeitura). Estas variáveis serão utilizadas apenas na avaliação dos resultados apresentados pelos algoritmos. A Tabela 5 apresenta uma amostra dos dados.

Tabela 5: Amostra dos dados da base real.

Casa	Padrão	Característica	Cobertura	Elemento	Dejetos	Água	Lixo
261463	2	3	1	2	2	3	1
261464	3	2	4	2	2	2	1
261468	3	3	6	5	1	2	2
261471	1	1	1	1	5	3	2
261472	4	2	2	4	5	1	3
...	...	...	...	...	...	...	...

4.1.3. *Software*

Para comparar os métodos foram utilizados os algoritmos presentes na linguagem *R - A language and environment for statistical computing*, versão 1.9.1, desenvolvida por R Development Core Team. Mais informações, bem como os arquivos de instalação, estão disponíveis no endereço *www.R-project.org*.

A escolha desta linguagem está intimamente ligada aos seguintes fatores: é gratuita, é constantemente atualizada, possui fácil aprendizado e não requer equipamento de alto custo.

4.1.4. *Hardware*

Para a realização dos testes foi utilizado um PC, com processador AMD Athlon ™ de 2500 mhz, 256 MB de memória RAM e HD com 40 GB (4200 rpm), com o sistema operacional Windows® 2000, *Service Pack 4*.

Para não interferir nos resultados, durante os testes apenas os processos fundamentais ao funcionamento do sistema operacional e da linguagem *R* estavam sendo executados. Demais *softwares*, como por exemplo antivírus e *firewall*, foram desabilitados.

4.2. Métodos

Os algoritmos foram executados nas duas bases de dados, sendo avaliados os seguintes aspectos: desempenho, dimensionalidade, escalabilidade, tipo de variáveis, facilidade de uso, interpretabilidade e qualidade dos resultados. Mais informações na Tabela 6.

Especificamente para o teste de escalabilidade, foram comparados o tempo de processamento da base original (t_P) e o tempo de processamento da amostra (t_A). Para se medir o tempo de execução de cada algoritmo, armazenou-se a hora de início (t_1) e a hora de término (t_2) da execução dos mesmos e, depois de encerrado o processamento, através de uma operação de subtração ($t_2 - t_1$) verificou-se o tempo total decorrido. O código necessário para se fazer tal função em linguagem *R* pode ser visto no ANEXO 2.

Tabela 6: Aspectos a serem testados nos algoritmos

Aspecto	Descrição
Desempenho	Tempo total exigido para o processamento, medido em minutos e segundos.
Dimensionalidade	Capacidade para lidar com grandes bases de dados.
Escalabilidade	Diferença de desempenho quando a base de dados tem seu número de registros aumentado significativamente.
Tipo de variáveis	A base simulada possui apenas variáveis intervalares, enquanto a base real possui variáveis ordinais, espera-se que o algoritmo consiga trabalhar com tipos diferentes.
Facilidade de uso	Parâmetros necessários para executar o algoritmo.
Interpretabilidade	Facilidade para interpretar os dados apresentados.
Qualidade dos resultados	Na base simulada deverão ser encontrados 3 grupos distintos, já na base real deverão ser encontrados 6 grupos.

5. RESULTADOS EXPERIMENTAIS

Este capítulo apresenta os algoritmos utilizados, os procedimentos que necessitaram ser executados nas bases de dados e os resultados obtidos através dos testes.

5.1. Algoritmos utilizados

Os algoritmos utilizados nos testes são apresentados na Tabela 7. A indicação do pacote (*package*) a que pertencem se faz necessária já que eles não estão presentes originalmente na linguagem *R*. São desenvolvidos por terceiros e necessitam ser incorporados a mesma. Os pacotes, que incluem o código fonte, a versão compilada e os arquivos de ajuda, podem ser obtidos no endereço *cran.r-project.org*.

O único algoritmo não presente nos pacotes disponíveis no *site*, até esta data, é o DBSCAN. Ele foi implementado por Christian Hennig (*hennig@math.uni-hamburg.de*) e obtido para estes testes através da lista de discussão da linguagem *R*. A íntegra do seu código fonte se encontra no Anexo 1. Um exemplo de utilização de cada um dos algoritmos está presente no Anexo 2.

Tabela 7: Algoritmos utilizados nos testes.

Algoritmo	***Package***	**Método de Formação**	**Tipo de Variáveis**
AGNES	cluster	Hierárquico Aglomerativo	Intervalares e Binárias
DIANA	cluster	Hierárquico Divisivo	Intervalares e Binárias
K-MEANS	cClust	Partição	Intervalares e Binárias
K-MEDOID	cluster	Partição	Intervalares e Binárias
DBSCAN		Baseado em Densidade	Intervalares e Binárias
BIC	mClust	Baseado em Modelo	Intervalares

Como pode ser observado na Tabela 7, não foram avaliados algoritmos com método de formação baseado em grade. Infelizmente, até a presente data, não há nenhum algoritmo deste tipo disponível no *site* da linguagem *R* e também não foi possível encontrar algum através da lista de discussão.

A tabela também mostra uma característica negativa presente em todos os algoritmos: aceitam apenas dados intervalares e binários, sendo que o algoritmo BIC é ainda mais restrito e aceita apenas dados intervalares. Tal limitação acarretou na necessidade de transformação

nos dados da base de Belém, uma vez que esta possui apenas dados categorizados. Os procedimentos adotados são descritos na próxima seção.

5.2. Base com dados reais

A base de dados real é composta por seis variáveis ordinais. A distribuição pode ser observada na Figura 9.

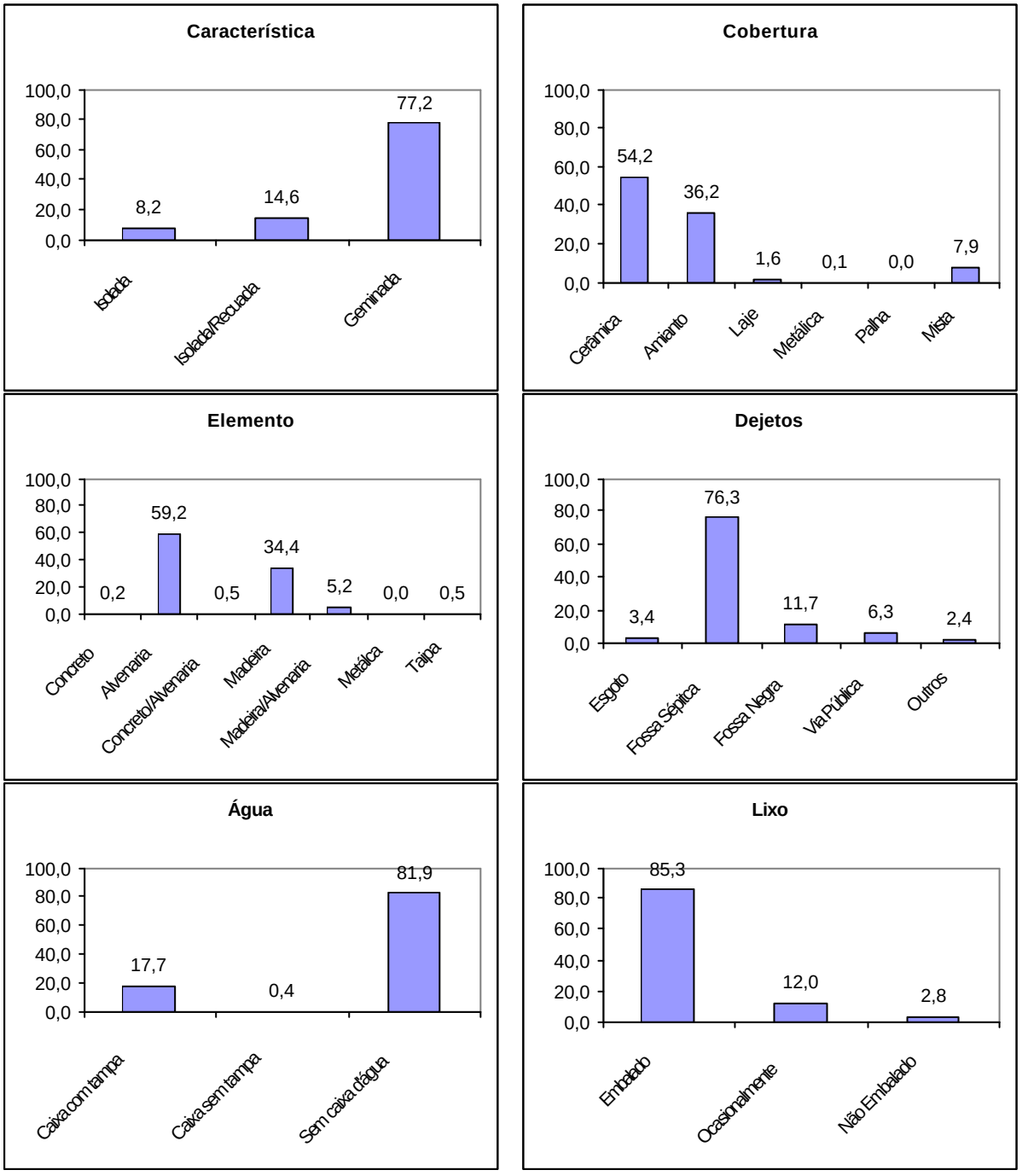

Figura 9: Distribuição de freqüência, em percentual, da base de dados real.

Em três das variáveis (Cobertura, Elemento e Água), alguns dos estados estão presentes em menos de 1% da população, entretanto como estes dados fazem parte de um cadastro que sofre rigoroso controle por parte da prefeitura, optou-se por não remover tais registros, acreditando-se que a informação condiz com a verdade.

Como todas as variáveis são ordinais e nenhum dos algoritmos utilizados permite o uso de tal tipo, foi necessário transportá-las para o intervalo [0, 1]. Para tanto, foi escrito um programa na linguagem Delphi (versão 7.0), adotando o primeiro procedimento de padronização descrito na seção 3.3.4 - Variáveis ordinais.

Além disto, antes de se realizar a padronização, foi necessário inverter os valores das categorias. Na base original, o estado 1 representa a categoria de maior valor, enquanto a fórmula apresentada na referida seção espera que ele seja o último (1 deve representar o estado de menor valor).

A Tabela 8 apresenta os possíveis valores que cada umas das variáveis presentes na Tabela 4 pode assumir após a padronização.

Tabela 8: Valor dos dados da base real após a padronização.

Variável	Valores Originais	Valores Padronizados
Característica	1, 2, 3	0, 0.5, 1
Cobertura	1, 2, 3, 4, 5, 6	0, 0.2, 0.4, 0.6, 0.8, 1
Elemento	1, 2, 3, 4, 5, 6, 7	0, 0.166, 0.333, 0.5, 0.666, 0.833, 1
Dejetos	1, 2, 3, 4, 5	0, 0.25, 0.5, 0.75, 1
Água	1, 2, 3	0, 0.5, 1
Lixo	1, 2, 3	0, 0.5, 1

5.3. Base com dados simulados

Sabendo-se da restrição dos algoritmos quanto ao tipo de variável, a base com dados simulados foi gerada contendo apenas valores intervales. Com isto, não houve necessidade de que estes dados sofressem qualquer modificação ou que se fizesse uma análise exploratória sobre os mesmos.

5.4. Resultados

Antes de apresentar os resultados e os comentários a respeitos dos testes, é preciso dizer que, devido à quantidade de memória RAM o computador utilizado nos testes, não foi

possível executar os algoritmos AGNES, DIANA, DBSCAN e BIC nas bases de dados completas.

Ocorre que estes algoritmos, com exceção do DBSCAN, necessitam, antes de qualquer coisa, calcular a matriz de distância entre os elementos. Como as bases possuem mais de duzentos mil registros, a quantidade de memória necessária para armazenar a matriz é grande. A quantidade de memória necessária, em *bytes*, é dada por:

$$(n-1)^2 \times 8 \qquad (20)$$

onde n é o número de registros. A multiplicação por 8 é porque este é o número de *bytes* utilizados para armazenar um número real (a distância entre os objetos é um número real).

Entretanto, mesmo que se tivesse um super computador com tal capacidade de memória, há ainda outra limitação: a linguagem *R* não trabalha com mais do que 2 GB de RAM, ou seja, calcula no máximo a matriz de uma base de dados com cerca de 16.000 registros.

Como a capacidade de memória do computador utilizado nos testes é de 256 MB, optou-se por trabalhar com amostras das bases de dados. O tamanho da amostra escolhido foi de 5000 elementos. Este tamanho poderia ser um pouco maior, próximo de 5600 registros, entretanto optou-se deixar parte da memória RAM disponível para realização dos cálculos, evitando assim a necessidade do sistema operacional fazer *swap*[5] dos dados.

Optou-se por retirar uma Amostra Aleatória Simples (AAS) sem reposição (ver BOLFARINE e BUSSAB, 2000). Para a retirada da amostra foi escrito um programa na linguagem Delphi: o programa gera aleatoriamente um valor entre 1 e o número total elementos, retira este elemento da base de dados e volta a sortear outro até que a amostra atinja o tamanho desejado.

5.4.1. Testes com as amostras das bases de dados

A Tabela 9 apresenta os resultados dos testes com amostras de 5000 registros. A tabela mostra o tempo de processamento (em minutos e segundos) para realizar o processo de agrupamento (o tempo de carga da base de dados não é computado) e o número de grupos apontados pelos algoritmos. A exceção nesta informação são os algoritmos com métodos de partição, que exigem esta informação como parâmetro de entrada.

---

[5] Sempre que a quantidade de memória RAM livre é inferior a quantidade necessária para executar determinado procedimento, o sistema operacional grava parte dos dados existentes nela no disco rígido, liberando assim parte da memória RAM para o processamento. Como esta operação consome tempo e interfere no desempenho dos algoritmos, optou-se por evitar que ela ocorresse.

Tabela 9: Resultado dos testes com amostras.

	Amostra da base real		Amostra da base simulada	
Algoritmo	**Tempo (mm:ss)**	**Nr. de Grupos**	**Tempo (mm:ss)**	**Nr. de Grupos**
AGNES	28:17	de 5 a 9	26:54	de 3 a 5
DIANA	36:58	de 6 a 10	32:25	de 3 a 5
K-MEANS	< 00:01	6	< 00:01	3
K-MEDOID	< 00:01	6	< 00:01	3
DBSCAN	06:45	9	05:22	3
BIC	09:39	Não Classificou	08:24	Não Classificou

Os dados da tabela mostram que, com exceção dos métodos de partição, que são extremamente rápidos, os demais necessitam de um tempo maior para processar a base amostrada, especialmente os algoritmos hierárquicos que levaram cerca de 30 minutos para serem executados.

O algoritmo BIC, após processar por mais de 8 minutos a amostra da base simulada e por mais de 9 a amostra da base real, acusou falta de memória e interrompeu o processo. Embora seja baseado em um modelo estatístico, ele faz uso de um algoritmo hierárquico em sua inicialização para saber o número ideal de agrupamentos. Infelizmente, o algoritmo não calcula a quantidade de memória existente e, ao invés de utilizar apenas parte dos dados para fazer esta inicialização, utiliza toda a base. Além da memória para fazer este cálculo inicial, ele também necessita de mais memória para o cálculo do modelo e por isto não conseguiu realizar a operação.

Ainda em relação ao tempo, se faz necessário destacar que todos os algoritmos tiveram um desempenho melhor ao agrupar os dados da base simulada. Isto ocorreu devido ao fato desta base possuir menor dimensionalidade que a base com dados real: 5000 x 3 (linhas x colunas) contra 5000 x 6.

Quanto ao número de grupos, os algoritmos AGNES e DIANA indicaram que a base simulada deveria ter de 3 a 5 grupos e a base real de 5 a 10, o que foi considerado um bom resultado. Faz-se necessário dizer que a avaliação do número de grupos indicados pelos algoritmos hierárquicos é subjetiva, e foi feita baseado nas medidas de distância entre os grupos e na análise do dendrograma gerado (Figura 10). A forma como os algoritmos foram implementados não permite que se defina um ponto de parada (número de agrupamentos

desejados) e se execute o algoritmo até este ponto. Não existindo tal possibilidade, não foi possível comprar os agrupamentos resultantes com os originais.

O algoritmo DBSCAN, que indica o número de grupo após a operação de agrupamento, saiu-se melhor. Para a base simulada indicou 3 grupos, o que vem de encontro com a realidade já que os dados foram assim simulados, e 9 para a base real.

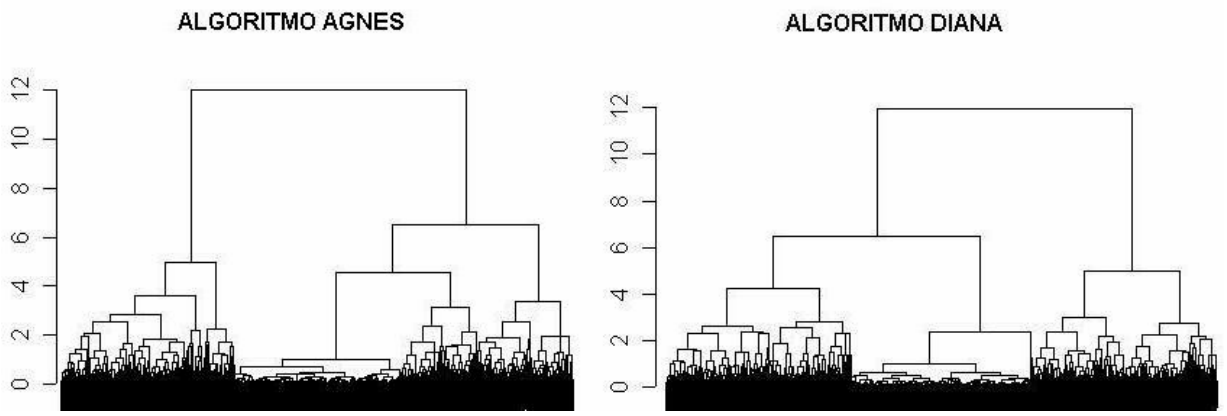

Figura 10: Dendrogramas dos algoritmos AGNES e DIANA após o agrupamentos dos dados da amostra da base simulada.

Um ponto negativo nos algoritmos AGNES e DIANA da linguagem *R* é que eles não provêem um modo de se obter quais elementos fazem parte de cada um dos agrupamentos formados. Os algoritmos K-MEANS, K-MEDOID e DBSCAN permitem exportar um arquivo texto com tal informação. Baseado neste arquivo foi possível comparar a classificação gerada pelos algoritmos com a classificação anterior das bases.

Na base simulada, para se chegar ao percentual de acerto, comparou-se a classificação do algoritmo com o grupo onde o dado foi gerado. Para a base real, comparou-se a classificação do algoritmo com a classificação dada ao imóvel pela prefeitura de Belém. Os resultados podem ser vistos na Tabela 10.

Tabela 10: Resultados dos testes com os algoritmos *k-means*, *k-medoid* e *dbscan* com amostras das bases de dados simulada e real.

	Algoritmo *k-means*		**Algoritmo *k-medoid***		**Algoritmo *dbscan***	
Base de Dados	**Tempo (mm:ss)**	**Percentual de Acertos**	**Tempo (mm:ss)**	**Percentual de Acertos**	**Tempo (mm:ss)**	**Percentual de Acertos**
Simulada (amostra)	< 00:01	98,43 %	< 00:01	99,38 %	05:22	59,40 %
Real (amostra)	< 00:01	24,40 %	< 00:01	28,46 %	06:45	Não verificado

Nota-se que os algoritmos de partição possuem um percentual de acerto bastante semelhante para ambas as bases e, além disto, próximo a 100% para a base com dados simulados. O algoritmo DBSCAN, embora tenha corretamente indicado 3 grupos, obteve um percentual de acerto baixo se comparado com os outros dois.

O percentual de acerto dos algoritmos de partição para a base real foi baixo, mas é preciso destacar que os demais algoritmos indicaram que esta base deve ter entre 5 e 10 grupos, ou seja, possivelmente a classificação original é deficiente. O algoritmo DBSCAN não teve o seu resultado comparado pois indicou 9 grupos e, como a base com dados reais estava classifica em apenas 6, não houve possibilidade de comparação.

5.4.2. Testes com as bases de dados completas

Os algoritmos de partição foram os únicos que rodaram satisfatoriamente nas bases de dados completas, demonstrando com isto que possuem alta dimensionalidade. Houve uma tentativa de se executar o algoritmo DBSCAN, porém após 16 horas de processamento ele havia agrupado pouco mais de dez mil elementos da base simulada, como isto representa menos de 5% da mesma, optou-se por interromper o processo, já que ele consumiria dias de processamento. A Tabela 11 apresenta os resultados.

Tabela 11: Resultado dos testes com os algoritmos de partição com as bases de dados completas.

	Algoritmo *k-means*		**Algoritmo *k-medoid***	
Base de Dados	**Tempo (mm:ss)**	**Percentual de Acertos**	**Tempo (mm:ss)**	**Percentual de Acertos**
Simulada	00:03	98,66 %	00:02	98,85 %
Real	00:08	14,36 %	00:06	14,81 %

Comparando-se os dados desta tabela e ainda levando em consideração os dados da Tabela 10, nota-se que o *k-medoid* é o mais rápido dos algoritmos, uma vez que levou 1 segundo a menos para agrupar os dados da base simulada e 2 segundos a menos para os dados da base real. Levando-se em conta que o tempo máximo foi de 8 segundos, esta diferença é significativa.

Ainda levando-se em comparação os dados da Tabela 10, nota-se que o algoritmo *k-medoid* apresentada resultados ligeiramente melhores, já que possui percentual de acerto superior em todos os testes, exceto nos testes com a amostra da base de dados simulada.

Um fato que gerou estranheza foi a diferença no percentual de acerto dos algoritmos de partição na base real e na amostra desta base, os algoritmos se saíram melhor sobre a amostra

do que sobre a base como um todo (se o resultado fosse o inverso poderia se concluir que a amostra não possui a qualidade desejada).

A escalabilidade só pode ser medida para os algoritmos que rodaram sobre as bases completas, já que para verificar esta característica é preciso comparar o tempo de processamento em duas bases com diferentes tamanhos. Nota-se que apenas os de partição, *k-means* e *k-medoid*, possuem um bom grau de escalabilidade, uma vez que o aumento do tempo de processamento foi menor, proporcionalmente, ao aumento do número de registros em ambas as bases.

Quanto a interpretabilidade e a usabilidade dos resultados, os algoritmos se mostram pouco eficientes, pois os gráficos e arquivos gerados possuem baixa qualidade (na maioria das vezes apresentam apenas um borrão).

A Figura 11 apresenta o gráfico resultante do algoritmo *k-medoid* sobre a amostra da base simulada. Nos pontos onde existem intersecções entre os agrupamentos, é difícil distinguir os elementos que fazem parte de cada grupo. Isto ocorre porque o gráfico tenta mostrar todos os elementos, ao invés de mostrar apenas uma parte deles. Como a base de dados é pequena e foi simulada com os grupos bem definidos (distantes entre si), ainda é possível identificar os agrupamentos. Entretanto, numa base com maior número de elementos e com grupos mais próximos, esta identificação é dificultada.

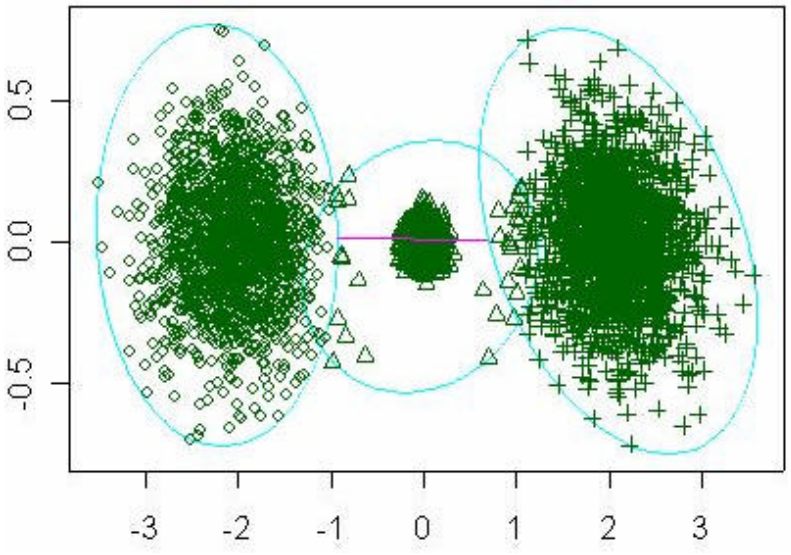

Figura 11: Gráfico dos agrupamentos gerados pelo algoritmo *k-medoid* na base simulada.

Por último, as variáveis de resposta (que podem ser exportadas para arquivos) possuem informações limitadas, normalmente apenas os elementos que foram tomados como sementes (centróides) e o vetor com o número do agrupamento a que cada elemento foi designado.

Informações úteis, tais como médias e *quartis* de cada grupo, são suprimidas e só podem ser obtidas através da manipulação do vetor, o que dificulta o trabalho de análise dos agrupamentos formados.

Entretanto, é preciso destacar que a linguagem *R* é perfeitamente capaz de gerar estas informações adicionais. Porém, para que isto seja possível, é preciso trabalhar com outros pacotes que acompanham a linguagem ou que podem ser baixados no *site*.

5.4.2.1. Análise dos agrupamentos resultantes

Apenas gerar os agrupamentos não é suficiente. É preciso que se analise o resultado a fim de se obter informação útil sobre os grupos gerados, afinal este é um dos principais objetivos da mineração de dados (BERSON *et al.*, 1999).

Como os percentuais de acerto dos algoritmos *k-means* e *k-medoid* para os dados simulados foram de quase 100%, ou seja, os grupos resultantes são praticamente idênticos aos originais, serão analisados apenas os resultados obtidos para a base real.

Diferentemente dos dados simulados, os dados da base real tiveram a composição de seus grupos alterados, sendo que menos de 15% dos casos permaneceram no mesmo grupo que estavam antes da análise de agrupamento. Na análise apresenta abaixo foi utilizado o resultado gerado pelo algoritmo *k-medoid*, por ele ter obtido resultados ligeiramente superior ao de *k-means*.

O primeiro passo para se analisar o resultado é estudar os grupos originais e entender o seu processo de classificação. A Tabela 12 mostra que, na base original, 39,05% dos imóveis estão no grupo 4 (popular) e 55,94% estão nos grupos 5 ou 6 (baixo e primário, respectivamente).

No sistema de classificação adotado, cada opção da variável possui um peso e o somatório destes pesos é dividido em faixas, formando os grupos. Ocorre que parte dos imóveis acaba tendo o seu somatório de pesos muito próximo ao limite de faixas, o que faz que a simples mudança de uma característica o coloque em uma faixa inferior ou superior a mais apropriada.

Os imóveis que estão não limite das faixas de classificação e a possível subestimação do número de grupos (o algoritmo de BDSCAN, ao analisar a amostra de dados da base real, apontou como sendo nove o número ideal agrupamentos) são as causas mais prováveis da diferença entre as classificações. A Tabela 12 mostra que o algoritmo classificou a maior

parte dos imóveis nos grupos 3 (médio) e 5, o que fez com que os grupos 3 a 6 fossem completamente diferente antes e depois do processo de AA. Isto explica o motivo dos percentuais de semelhança entre os agrupamentos, apresentados na Tabela 11, serem tão baixos.

Tabela 12: Número e percentual de imóveis por grupo antes e depois da Análise de Agrupamentos.

Classificação antes da AA			Classificação após a AA		
Grupo	**Número de Imóveis**	**Percentual**	**Grupo**	**Número de Imóveis**	**Percentual**
1	386	0,14	1	588	0,21
2	1191	0,44	2	1312	0,48
3	12145	4,44	3	106278	38,84
4	106854	39,05	4	26713	9,76
5	80907	29,57	5	104424	38,16
6	72142	26,37	6	34311	12,54

6. CONSIDERAÇÕES FINAIS

Este trabalho apresentou um estudo detalhado sobre a Análise de Agrupamento. Foram descritas suas aplicações, medidas de similaridade adequadas a cada tipo de dado, sejam eles de mesmo tipo ou não, e métodos de formação de agrupamento.

Foram abordadas as metodologias de formação do agrupamento, tantos as novas como as tradicionais. Para cada metodologia foram destacadas as principais características, conceitos, vantagens e desvantagens. Além disto, para cada uma delas, foram citados alguns dos algoritmos existentes.

O trabalho trouxe ainda os resultados dos testes realizados com a linguagem *R*, onde foram avaliados algoritmos com diferentes métodos de formação. Através destes testes foi possível verificar a eficiência, a precisão, a dimensionalidade, a escalabilidade, os tipos de dados permitidos, e os resultados gerados pelos mesmos.

Verificou-se também que, embora a AA tenha passado por uma série de avanços nos últimos anos, em especial ao que se refere aos métodos de formação do agrupamento, ainda há muita pesquisa a ser feita, principalmente para que o seu uso em grandes bancos de dados se torne viável.

O trabalho mostrou ainda que a linguagem *R*, embora robusta e gratuita, ainda é carente em recursos automatizados, especialmente quando se têm em mãos variáveis de tipos mistos e um grande volume de dados.

Baseado na revisão bibliográfica e nos testes realizados, a conclusão que se pode tirar desta pesquisa é a de que a AA é uma técnica eficiente de mineração de dados, porém seu uso requer conhecimentos sobre a teoria e sobre a base de dados utilizada. A ausência de conhecimento em qualquer um destes pontos pode levar a obtenção de resultados errôneos ou, no mínimo, não tão precisos.

6.1. Sugestões de trabalhos futuros

Com o transcorrer do desenvolvimento dessa pesquisa foram observadas necessidades ainda existentes na Análise de Agrupamentos. As sugestões de trabalhos futuros relevantes se dão especialmente em relação aos algoritmos na linguagem *R*, são eles:

- Desenvolvimento de um pacote que permita calcular a distância entre variáveis de tipos mistos, algo de fundamental importância para as bases de dados do mundo real e ainda inexistente.

- Otimização dos algoritmos com método de formação baseado em modelo e em densidade para que seu uso em grandes bancos de dados seja viável.
- Modificação do código do algoritmo BIC, para que ao ser iniciado ele utilize apenas uma amostra dos dados para descobrir o número ideal de agrupamentos.
- Implementação de algoritmos baseados em grade, como o STING citado neste trabalho (seção 3.5 - Outros Algoritmos de AA), já que atualmente não há nenhum.
- Aprimoramento dos gráficos, já que os atuais exibem todos os pontos encontrados, o que num banco de dados com milhares ou milhões de registros acaba originando um borrão não interpretável.
- Melhoria nas informações apresentadas como resposta pelos algoritmos, fazendo com que forneçam dados como média, mediana e quartis de cada um dos grupos formados para facilitar a análise dos mesmos.

REFERÊNCIAS BIBLIOGRÁFICAS

ANDRITSOS, Periklis. **Data Clustering Techniques.** Canadá, Toronto, 2002. Disponível em www.cs.toronto.edu/~periklis/publications.html. Acessado em Outubro de 2003.

BERRY, Michael J. A.; LINOFF, Gordon. **Data Mining Techniques:** For Marketing, Sales, and Customer Support. New York: Wiley Computer Publishing, 1997.

BERSON, Alex; SMITH, Stephen; THEARLING, Kurt. **Building Data Mining Applications for CRM.** USA, New York: MacGrawHill, 1999.

BOLFARINE, Heleno; BUSSAB, Wilton de O. **Elementos de Amostragem**. São Paulo: USP, Versão Preliminar, 2000.

BUSSAB, Wilton de O.; MIAZAKI, Édina S; ANDRADE, Dalton F. de. **Introdução à Análise de Agrupamentos.** 9º Simpósio Nacional de Probabilidade e Estatística. São Paulo: ABE, 1990.

CHATTERJEE, S.; PRICE, B. **Regression Analysis by Examples.** New York. 2ª. ed. John Wiley e Sons, Inc., 1991.

CHEN, Ming-Syan; HAN, Jiawei; YU, Philip S.. **Data Mining: An Overview from a Database Perspective**, IEEE Transactions on Knowledge and Data Engineering, v.8, n.6, December 1996.

COMRIE, Andrew C. Comparing Neural Networks and Regression Models for Ozone Forecasting. **Journal of the Air & Waste Management Association.** Tucson, Arizona, USA, v.47, 1997, p.653-663.

DINIZ, Carlos Alberto; LOUZADA NETO, Francisco. **Data Mining:** uma introdução. São Paulo: ABE, 2000.

DUNKEL, Brian; SOPARKAR, Nandit; SZARO, John; UTHURUSAMY, Ramasamy. Systems for KDD: From concepts to practice. **Future Generation Computer Systems**, n. 13, 1997, p. 231-242.

FRALEY, Chris; RAFTERY, Adrian.E. Model-based Clustering, Discriminant Analysis, and Density Estimation. **Journal of the American Statistical Association**, v. 97, n. 458, June 2002, p. 611-631.

FAYYAD, Usama; PIATETSKY-SHAPIRO, Gregory; SMYTH, Padhraic. From data mining to knowledge discovery: An overview. In: **Advances in Knowledge Discovery and Data Mining**, AAAI Press / The MIT Press, MIT, Cambridge, Massachusetts, and London, England, 1996, p.1-34.

FAYYAD, Usama; STOLORZ, Paul. Data mining and KDD: Promise and challenges. **Future Generation Computer Systems**, n. 13, 1997, p. 99-115.

FREITAS, Alex A.; PACHECO, Roberto C. S.; ROMÃO, Wesley. **Uma revisão de abordagens genético-difusas para descoberta de conhecimento em banco de dados**. Disponível em: www.cs.kent.ac.uk/people/staff/aaf/pub_papers.dir/Acta-Scientiarum.pdf. Acessado em Dezembro de 2003.

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

HARRISON, Thomas H. **Intranet data warehouse.** São Paulo: Berkeley Brasil, 1998.

HASTIE, Trevor; TIBSHIRANI, Robert; FRIEDMAN, Jerone. **The Elements of Statistical Learning:** Data Mining, Inference, and Prediction. New York: Springer Verlag, 2001.

HRUSCHKA, Eduardo Raul; EBECKEN, Nelson Francisco Favilla.A genetic algorithm for cluster analysis. **Intelligent Data Analysis (IDA)**, Netherlands, v.7, n.1, 2003, p.15-25.

HUANG, Zhexue. Clustering Large Data Sets with Mixed Numeric and Categorical Values. In: **Proceedings of the 1st Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'97)**. Singapore, 1997a. p. 21–34.

_____. A Fast Clustering Algorithm to Cluster Very Large Categorical Data Sets in Data Mining. In: **SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (SIGMOD-DMKD'97).** Tucson: Arizona, USA, Maio de 1997b.

JOHNSON, Richard A.; WICHERN, Dean W. **Applied Multivariate Statistical Analysis.** New Jersey: Prentice-Hall, Inc., 1982.

KAUFMAN, L.; ROUSSEEUW, P. J. **Finding Groups in Data**: An Introduction to Cluster Analysis. New York: Wiley, 1990.

MICHAUD, Pierre. Clustering techniques. **Future Generation Computer Systems**, n. 13, 1997. p. 135-147.

OLIVEIRA, Álvaro Borges de; BORATTI, Isaias Camilo. **Introdução à Programação – Algoritmos.** Florianópolis: Bookstore, 1999.

PISON, Greet; STRUYF Anja; ROUSSEEUW Peter J. Displaying a clustering with CLUSPLOT. **Computational Statistics & Data Analysis**, n.30, 1999. p. 381-392.

WIVES, Leandro Krug. **Um estudo sobre Agrupamento de Documentos Textuais em Processamento de Informações não Estruturadas Usando Técnicas de "Clustering",** [dissertação], Universidade Federal do Rio Grande do Sul, 1999.

ZAÏANE, Osmar R.; FOSS, Andrew; LEE, Chi-Hoon; WANG, Weinan. **On Data Clustering Analysis: Scalability, Constraints and Validation**. in Proc. of the Sixth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'02), pp 28-39, Taipei, Taiwan, May, 2002.

ANEXO 1 – ALGORITMO DBSCAN

```
distvector <- function(x,data){
ddata <- t(data)-x
dv <- apply(ddata^2,2,sum)}

# data may be nxp or distance matrix
# eps is the dbscan distance cutoff parameter
# MinPts is the minimum size of a cluster
# scale: Should the data be scaled?
# distances: has to be TRUE if data is a distance matrix
# showplot: Should the computation process be visualized?
# countmode: dbscan gives messages when processing point no. (countmode)
dbscan <- function(data,eps, MinPts=5, scale=FALSE, distances=FALSE,
    showplot=FALSE, countmode=c(1,2,3,5,10,100,1000,5000,10000,50000)){
  data <- as.matrix(data)
  n <- nrow(data)
  if (scale) data <- scale(data)
  unregpoints <- rep(0,n)
  e2 <- eps^2
  cv <- rep(0,n)
  cn <- 0
  i <- 1
  for (i in 1:n){
    if (i %in% countmode) cat("Processing point ", i," of ",n, ".\n")
    unclass <- cv<1
    if (cv[i]==0){
      if (distances) seeds <- data[i,]<=eps
      else{
        seeds <- rep(FALSE,n)
        seeds[unclass] <- distvector(data[i,],data[unclass,])<=e2
      }
      if (sum(seeds)+unregpoints[i]<MinPts) cv[i] <- (-1)
      else{
        cn <- cn+1
        cv[i] <- cn
        seeds[i] <- unclass[i] <- FALSE
        unregpoints[seeds] <- unregpoints[seeds]+1
        while (sum(seeds)>0){
          if (showplot) plot(data,col=1+cv)
```

```
          unclass[seeds] <- FALSE
          cv[seeds] <- cn
          ap <- (1:n)[seeds]
#          print(ap)
          seeds <- rep(FALSE,n)
          for (j in ap){
#            if (showplot) plot(data,col=1+cv)
            jseeds <- rep(FALSE,n)
            if (distances) jseeds[unclass] <- data[j,unclass]<=eps
            else{
              jseeds[unclass] <- distvector(data[j,],data[unclass,])<=e2
            }
            unregpoints[jseeds] <- unregpoints[jseeds]+1
#             if (cn==1)
#               cat(j," sum seeds=",sum(seeds)," unreg=",unregpoints[j],
#                   " newseeds=",sum(cv[jseeds]==0),"\n")
            if (sum(jseeds)+unregpoints[j]>=MinPts){
              seeds[jseeds] <- cv[jseeds]==0
              cv[jseeds & cv<0] <- cn
            }
          } # for j
        } # while sum seeds>0
      } # else (sum seeds + ... >= MinPts)
    } # if cv==0
  } # for i
  if (sum(cv==(-1))>0){
    noisenumber <- cn+1
    cv[cv==(-1)] <- noisenumber
  }
  else
    noisenumber <- FALSE
  out <- list(classification=cv, noisenumber=noisenumber,
              eps=eps, MinPts=MinPts, unregpoints=unregpoints)
  out
} # dbscan
# classification: classification vector
# noisenumber: number in the classification vector indicating noise points
# unregpoints: ignore...
# By Christian Henni - hennig@math.uni-hamburg.de - Universitaet Hamburg
```

ANEXO 2 – PROGRAMAS EM LINGUAGEM *R*

Atenção: frases em *itálico* e precedidas do símbolo “#” são comentários.

```
# algoritmo agnes #
library(cluster)    #carrega o pacote em memória
x <- read.table("d:\\dados\\base.txt")  #lê a base de dados
(t1 <- Sys.time())  #armazena a hora de início do procedimento
ag <- agnes(x, method = "complete")  #agrupa os elementos
(t2 <- Sys.time())  #armazena a hora de término do procedimento
t <- t2 - t1        #calcula o tempo total gasto para agrupar os elementos
t                   #mostra o tempo total gasto
hcag <- as.hclust(ag)
plot(hcag);         #mostra o gráfico resultante
```

```
# algoritmo diana #
library(cluster)
x <- read.table("C:\\dados\\base.txt")
(t1 <- Sys.time())
d <- diana(x)
(t2 <- Sys.time())
t <- t2 - t1
t
hcag <- as.hclust(d)
plot(hcag); mtext("diana", side=1)
```

```
# algoritmo K-medoid #
library(cluster)
x <- read.table("d:\\dados\\base.txt")
(t1 <- Sys.time())
c <- clara(x,6)
(t2 <- Sys.time())
t <- t2 - t1
t
c$clusinfo
resultado <- as.vector(c$clustering)
save(resultado,file = "d:\\dados\\resultado_k-medoid.txt", ascii = TRUE)
```

```
# algoritmo K-means #
library(cclust)
x <- read.table("d:\\dados\\base.txt")
m <- as.matrix(x)
(t1 <- Sys.time())
cl<-cclust(m,6,15,verbose=TRUE,method="kmeans")
(t2 <- Sys.time())
#plot(x, col=cl$cluster)
t <- t2 - t1
t
cl$size
resultado <- as.vector(cl$cluster)
save(resultado,file = "d:\\dados\\resultado_k-means.txt", ascii = TRUE)


# algoritmo bic #
library(mClust)
DadosMatrix <- read.table("d:\\dados\\base.txt")
(t1 <- Sys.time())
DadosMclust <- Mclust(DadosMatrix)
(t2 <- Sys.time())
t3 <- t2 - t1
t3
plot(DadosMclust,DadosMatrix)


# algoritmo dbscan #
# rodar primeiro o algoritmo presente no Anexo 1
x <- read.table("d:\\dados\\base.txt")
(t1 <- Sys.time())
dbscan_simulado <- dbscan(x,0.07,50, scale = TRUE)
(t2 <- Sys.time())
t_simulado <- t2 - t1
t_simulado
dbscan_simulado$noisenumber
resultado <- as.vector(dbscan_simulado$classification)
save(resultado,file = "d:\\dados\\resultado_dbscan.txt", ascii = TRUE)
```

ANEXO 3 – ALGUNS ALGORITMOS DE AA EM PORTUGOL

Algoritmo Hierárquico Aglomerativo:

```
Calcule a distância entre todos os pares de elementos;
Una os dois elementos mais semelhantes;
Recalcule o centróide do cluster recém formado;
para Todos os elementos ainda não agrupados faça
  Calcule a distância entre o cluster recém formado e os elementos não
  agrupados;
  Una os dois elementos mais semelhantes;
  Recalcule o centróide do cluster;
fim para
```

Algoritmo *k-means:*

```
Especifique k;
Selecione os k objetos que serão os centróides dos agrupamentos;
para todos os objetos restantes faça
  Calcule a distância entre o elemento e os centróides;
  Adicione o elemento ao agrupamento que possuir a menor distância;
  Recalcule o centróide do agrupamento;
fim para

para Todos os k agrupamentos faça
  Calcule a Soma de Quadrados Residual;
fim para

repita
  para todos os n elementos faça
    Mova o elemento para os outros agrupamentos;
    Recalcule a Soma de Quadrados Residual;
    se soma dos Quadrados Residual diminuiu então
      O objeto passa a fazer parte do agrupamento que produzir maior ganho;
      Recalcule a Soma de Quadrados Residual dos agrupamentos alterados;
    fim se
  fim para
até Número de interações = i ou Não ocorra mudança de objetos
```

Algoritmo *k-medoid:*

Especifique k;
Selecione os k objetos que serão os *medoids* inicias dos *clusters*;
repita
  Atribua cada objeto remanescente ao *cluster* com o *medoid* mais próximo;
  Aleatoriamente, selecione um objeto que não seja um *medoid*, o_{random};
  Calcule o custo total (s) de trocar o *medoid* o_j pelo objeto o_{random};
  se $s < 0$ **então**
    Troque o_j por o_{random};
até Não ocorra mudança de objetos

Algoritmo *k-prototupe:*

Selecione os k objetos que serão os *prototypes* inicias dos *clusters*;
para todos os objetos restantes **faça**
  Aloque o objeto ao *cluster* cujo *prototype* possuir valor mais próximo;
  Atualize o objeto *prototype*;
fim para
repita
  para todos os n objetos **faça**
    Calcule a similaridade em relação aos *prototypes*;
    se Objeto for mais similar ao *prototype* de outro *cluster* **então**
      Troque o objeto de *cluster*;
      Atualize o objeto *prototype*;
    fim se
  fim para
até Não ocorra mudança de objetos