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

Rafael Leão Brazão e Silva

# Estudo do desempenho do algoritmo Agrupamento em Duas Etapas através de comparações realizadas sob a metodologia de planejamento de experimentos

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

Prof. Doutor Dalton Francisco de Andrade

Florianópolis, outubro de 2007.

# Estudo do desempenho do algoritmo Agrupamento em Duas Etapas através de comparações realizadas sob a metodologia de planejamento de experimentos

Rafael Leão Brazão e Silva

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

_____

Prof. Dr. Rogério Cid Bastos (UFSC)

Coordenador do Curso

Banca Examinadora

_____

Prof. Dr. Dalton Francisco de Andrade (orientador - UFSC)

_____

Prof. Dr. Heliton Ribeiro Tavares (UFPA)

_____

Prof. Dr. Paulo Justiniano Ribeiro Jr. (UFPR)

_____

Prof. Dr. Pedro Alberto Barbetta (UFSC)

Dedico este trabalho aos meus pais, pois sem a sua ajuda, seja financeira, seja através de apoio moral, ao longo da minha formação, este trabalho não existiria.

## AGRADECIMENTOS

Agradeço, primeiramente, a Deus, que me deu o dom da vida e saúde para realizar este trabalho.

A meus pais, Maria Izete e Airton Augusto, que me ensinaram a dar os primeiros passos e me educaram como cidadão, além do apoio, mesmo à distância.

Ao meu orientador, professor Dalton de Andrade, pela chance de poder ingressar ao programa da pós e de dar o seu precioso tempo de atenção e compartilhar o seu conhecimento.

Ao meu co-orientador, professor Pedro Barbetta, que, apesar de termos tido problemas no início da caminhada, foi essencial para o término desse projeto, seja pela sua ajuda intelectual, seja pela paciência que teve comigo quase todos os dias.

A professora Silvia Nassar, que foi minha segunda mãe durante este processo. Sem a sua ajuda o caminho teria sido bem mais longo.

Aos meus amigos Everton, Luiz, Kuesley, Cláudio, Marcelo, Beatriz, Jaqueline, David e Almir, que ajudaram a não me sentir sozinho tão longe de casa.

E, finalmente, a CAPES e a CPA da UFSC pelo apoio financeiro, assim como a todos os outros que contribuíram para este trabalho e não foram citados.

Não leve a vida muito a sério, afinal você não vai sair vivo dela mesmo!

Autor desconhecido.

## RESUMO

O algoritmo ADE é um algoritmo de agrupamentos para se trabalhar com grandes bases de dados, as quais podem ter variáveis tanto quantitativas como categóricas. Nesta dissertação é realizado um estudo para verificar a adequação do ADE em problemas comuns à Análise de Agrupamentos. Para atingir esse objetivo, o desempenho do ADE foi comparado com os algoritmos K-médias e CLARA, em termos da acurácia e do tempo de processamento, através do uso de simulações Monte Carlo, realizadas sob a metodologia de Planejamento de Experimentos. Foram investigados os efeitos do número de registros, do número de grupos, do número de variáveis, da presença de variáveis categóricas, de grupos com diferentes variâncias, da correlação entre variáveis e da distribuição geradora dos dados. Verificou-se que o ADE tem melhor acurácia que os outros dois algoritmos quando as variâncias dos agrupamentos são diferentes e que essa vantagem do ADE diminui quando o número de variáveis aumenta. Verificou-se nos softwares utilizados, que o ADE é mais lento que os outros dois algoritmos, porém quando se aumenta o número de registros e o número de grupos, o aumento proporcional do tempo no ADE é menor do que nos outros dois algoritmos.



**Palavras-chaves:** Agrupamento em Duas Etapas, Análise de Agrupamentos, Planejamento de Experimentos.

**ABSTRACT**

The algorithm ADE is an algorithm of clustering made to work with great databases which can have variables so much quantitative as categorical. In this dissertation a study is accomplished to verify the adaptability of TSC in common situations that happens in the Cluster Analysis. For to reach this goal, the acting of TSC was compared with the algorithms K-means and CLARA in terms of the accuracy and of the processing time, through the use of simulations Monte Carlo accomplished under the Design of Experiments methodology. The effects of the number of cases, number of clusters, number of variables, presence of categorical variables, clusters with different internal variance, correlation between variables and of the generating distribution of the data were investigated. It was verified that TSC has better accuracy than the other two algorithms when the variances of the clusters are different and that that advantage of TSC decreases when the number of variables increases. It was verified in the used softwares, that TSC is slower than the other two algorithms, however when we increases the number of cases and the number of clusters, the proportional increase of the time in TSC is smaller than in the other two algorithms.

**Key-words:** TwoStep Cluster, Cluster Analysis, Design of Experiments.

## LISTA DE FIGURAS

## LISTA DE TABELAS

## LISTA DE SÍMBOLOS

| | | |
|---|---|---|
| $\overline{X}$ | - | Média amostral |
| $S$ | - | Desvio padrão amostral |
| n | - | Número de registros |
| $\hat{L}_k$ | - | Estimativa do valor do log da verossimilhança para um agrupamento k |
| $\hat{L}_{A_K}$ | - | Contribuição das variáveis quantitativas para o valor do log da verossimilhança em um agrupamento k |
| $\hat{L}_{B_K}$ | - | Contribuição das variáveis categóricas para o valor do log da verossimilhança em um agrupamento k |
| $\hat{\sigma}^2_{kj}$ | - | Estimativa da variância de uma variável j em um agrupamento k |
| $\Sigma$ | - | Somatório de valores |
| J | - | Número de variáveis |
| $J_A$ | - | Número de variáveis quantitativas |
| $J_B$ | - | Número de variáveis categóricas |
| K | - | Número de agrupamentos |
| C | - | Número de categorias |
| $\vec{x}_i$ | - | Vetor com i registros |
| P | - | Ponto em um eixo de coordenadas |
| r | - | Raio de registros dentro de um nó na CF-*tree* |
| $\overrightarrow{SL}$ | - | Vetor de soma linear de todas as variáveis |
| $\overrightarrow{SQ}$ | - | Vetor de soma quadrática de todas as variáveis |
| $\overrightarrow{PC}$ | - | Vetor com a proporção de c categorias em todas as variáveis categóricas |
| $\vec{X}_o$ | - | Centróide de um agrupamento |
| T | - | Limiar ou limite do raio em um nó em uma CF-*tree* |

## LISTA DE SIGLAS E ABREVIAÇÕES

| | | |
|---|---|---|
| ADE | - | Agrupamento em Duas Etapas |
| KDD | - | *Knowledge Discovery in Databases* |
| AGNES | - | *AGglomerative NESting* |
| DIANA | - | *DIvisive Analysis* |
| SQE | - | Soma dos Quadrados dos Erros |
| MAX | - | Valor máximo |
| MIN | - | Valor mínimo |
| CLARA | - | *Clustering LARge Applications* |
| CLARANS | - | *Clustering Large Applications based on RANdomized Search* |
| DBSCAN | - | *Density Based Spatial Clustering of Applications with Noise* |
| SOM | - | *Self Organized Maps* |
| BIRCH | - | *Balanced Iterative Reducing and Clustering using Hierarchies* |
| CF | - | *Cluster Feature* |
| BIC | - | *Bayesian Information Criterion* |

**SUMÁRIO**

# 1. INTRODUÇÃO

Em um mundo cada vez mais informatizado, extrair informações relevantes de dados gerados dentro das grandes empresas parece ter se tornado cada vez mais fundamental. Se há alguns anos os dados que circulavam dentro dos bancos de dados das grandes organizações eram fato de interesse apenas de um pequeno grupo que mantinha esses dados, agora o conhecimento proveniente desses dados torna-se peça fundamental para a tomada de decisões gerenciais.

Porém o aumento da capacidade de processamento de dados possibilitou às empresas armazenar *gigabytes* e até *terabytes* de dados que possuem informação útil, mas também muito conteúdo desnecessário. Os aplicativos tradicionais para se explorar bancos de dados conseguem fazer simples consultas aos bancos de dados e gerar relatórios, porém não são suficientes para responder as questões essenciais para a tomada de decisão gerencial (Rezende, 2005).

Com o objetivo de auxiliar na tomada de decisão gerencial, a mineração de dados (Fayyad et al.,1996) contém varias métodos que ajudam a encontrar padrões nos dados. Neste trabalho será dada ênfase nos métodos e algoritmos que permitem realizar as técnicas de agrupamentos.

As técnicas de agrupamentos buscam segmentar uma população heterogênea em subgrupos mais homogêneos. Numa base de dados de clientes, por exemplo, essas técnicas proporcionam a formação de agrupamentos de clientes relativamente similares (Han e Kamber, 2001), que podem responder de forma parecida a uma campanha de marketing.

A análise de agrupamentos é uma atividade humana importante, pois desde crianças aprendemos a distinguir grupos como cães e gatos, ou entre animais e plantas. A análise de agrupamentos tem sido usada em inúmeras aplicações, como reconhecimento de padrões, análise de dados e pesquisas de mercado (Han e Kamber, 2001).

## 1.1. Definição do problema

Existe uma grande variedade de algoritmos de agrupamentos na literatura. Eles costumam ser classificados de acordo com o método de formação de agrupamentos. Os métodos mais comumente encontrados na literatura são os hierárquicos e os de partição.

Os métodos hierárquicos criam uma decomposição hierárquica de um conjunto de registros em uma base de dados. Esses métodos funcionam realizando sucessivas divisões ou uniões de registros. Nos métodos hierárquicos aglomerativos, cada registro pertence a um agrupamento distinto, e eles vão se unindo até formar um único agrupamento, já nos métodos divisivos acontece o inverso, e no final, cada registro representa um agrupamento distinto (Johnson e Wichern, 2002).

Nos métodos de partição, dados n registros, são construídas K partições, sendo que cada partição representa um agrupamento. Geralmente o número K é fornecido previamente ao algoritmo.

Os métodos clássicos, às vezes, não resolvem todos os problemas pertinentes à mineração de dados. Os métodos hierárquicos geralmente não funcionam bem para grandes bases de dados, e os métodos de partição exigem um conhecimento prévio do número de agrupamentos, o que nem sempre é possível.

Chiu et al.(2001) propuseram um algoritmo realizado em duas etapas, que neste trabalho será chamado de ADE, que detecta agrupamentos em grandes bases de dados. Este algoritmo pode ser considerado como um melhoramento do algoritmo BIRCH, proposto em Zhang et al.(1996), pois permite a utilização de variáveis categóricas, o que não ocorre no BIRCH. Numa primeira etapa é construída uma árvore de pré-agrupamentos, onde são armazenadas apenas as estatísticas suficientes desses pré-agrupamentos. Numa segunda etapa, por um processo hierárquico, são construídos os agrupamentos através de agregação de pré-agrupamentos relativamente similares.

Esse algoritmo é adequado para ser utilizado em algumas situações de agrupamentos. O problema no qual este trabalho se foca é justamente auxiliar na descoberta de quais situações da análise de agrupamentos que o ADE é adequado.

## 1.2. Objetivos

### 1.2.1. Objetivo geral

Verificar o desempenho do algoritmo ADE em comparação com outros algoritmos da literatura de análise de agrupamentos.

### 1.2.2. Objetivos específicos

Fazer um levantamento bibliográfico dos algoritmos clássicos de análise de agrupamentos e algoritmos desenvolvidos mais recentemente.

Descrever detalhadamente o algoritmo de agrupamento em duas etapas (ADE).

Planejar experimentos para fazer comparações entre o ADE e outros algoritmos.

Gerar (simular) bases de dados com diversas distribuições e formas de agrupamentos para avaliar em que situações o ADE funciona melhor que outros algoritmos.

Discutir a performance dos algoritmos em termos das diferentes características dos dados e formas de agrupamentos

## 1.3. Metodologia do trabalho

Para realizar os objetivos descritos neste trabalho, primeiramente foi feita uma pesquisa bibliográfica sobre os algoritmos clássicos e recentes de análise de agrupamentos, verificando as condições de aplicação de cada um. Também foram pesquisados os estudos comparativos entre os algoritmos, que serviram de base para as comparações realizadas neste trabalho.

Em seguida foi planejado um projeto de experimentos para poder comparar o ADE com outros algoritmos citados na literatura. Adiante, foram geradas as bases de dados de acordo com o projeto de experimentos criado. Essas bases de dados foram geradas utilizando a metodologia de cópulas.

Com as bases de dados criadas, a acurácia (qualidade da formação dos agrupamentos encontrados) e o tempo de processamento do ADE foram comparados com os resultados dos algoritmos escolhidos. Os resultados foram analisados para verificar em que situações o ADE tem melhor e pior desempenho.

## 1.4. Justificativa e relevância

Segundo Kaufman e Rousesseeuw (2005), a análise de agrupamentos é uma importante atividade humana, pois o ser humano tende a organizar em grupos as coisas que vê durante a vida inteira. Nas grandes bases de dados também é importante organizar as

informações em grupos, para encontrarmos algum conhecimento útil. Explorar a análise de agrupamentos neste trabalho se justifica para buscar os novos caminhos que têm surgido nesta área da mineração de dados.

O algoritmo ADE foi feito para trabalhar com grandes bases de dados, o que é essencial para lidar com o tamanho das bases de dados das grandes organizações. Este algoritmo permite também a descoberta automática de um número adequado de agrupamentos (Chiu et al, 2001).

Outra grande vantagem do ADE é a possibilidade de se utilizar bases de dados que possuam variáveis tanto quantitativas como categóricas. Geralmente os métodos clássicos de partição permitem apenas que sejam utilizadas variáveis quantitativas e, outros métodos, como o K-modas, apenas variáveis categóricas, porém sem misturar os dois tipos.

Apesar das vantagens que são apresentadas pelos autores do algoritmo, é necessário confirmar, na prática, em que situações o ADE tem desempenho superior aos algoritmos clássicos da literatura. Isto é importante, pois, como já foi citado, existem muitos algoritmos na literatura de agrupamentos, e a escolha de qual algoritmo utilizar para um determinado problema é uma tarefa importante para a mineração de dados.

Para encontrar em que situações o ADE é mais adequado que outros algoritmos fazem-se necessárias comparações, de forma coerente, com outros algoritmos de agrupamentos. Uma forma de fazer comparações de forma estatisticamente planejadas é utilizando a metodologia de planejamento de experimentos (Montgomery, 2005), metodologia que é utilizada neste trabalho.

Este trabalho se justifica pelas contribuições que os resultados oferecem às futuras utilizações deste algoritmo, e por mostrar como fazer comparações de forma estatisticamente planejadas e, com isso, mostrar credibilidade nos resultados.

### 1.5. Delimitações do trabalho

Neste trabalho nenhum dos algoritmos utilizados nas comparações foi de desenvolvimento próprio. Foram utilizadas as versões comerciais destes algoritmos disponíveis em ferramentas estatísticas. Para testar o ADE foi utilizada uma implementação da ferramenta SPSS®, versão 14; para executar o K-médias foram utilizadas as ferramentas

SPSS®, versão 14, e a linguagem livre R, versão 2.4.1. Foi utilizada também a implementação do CLARA disponível na linguagem R, versão 2.4.1.

### 1.6. Estrutura do trabalho

Esta dissertação está organizada em sete capítulos, apresentados a seguir:

O primeiro capítulo traz uma breve introdução sobre o que esta dissertação se propõe a mostrar.

O segundo capitulo faz uma descrição sobre o tema de mineração de dados.

O terceiro capítulo descreve as técnicas de análise de agrupamentos, descrevendo os algoritmos clássicos da literatura.

O quarto capítulo descreve o algoritmo ADE, com suas características e funcionalidades.

O quinto capítulo descreve o planejamento do experimento realizado para as comparações entre o ADE e os outros algoritmos.

O sexto capítulo mostra os resultados das comparações planejadas no quinto capítulo.

O sétimo capítulo traz as considerações finais e sugestões futuras.

## 2. DESCOBERTA DE CONHECIMENTO EM BASES DE DADOS

Não é segredo que a massa de dados nas grandes empresas cresce exponencialmente a cada dia que passa. Dados na ordem de gigabytes e até terabytes vão sendo armazenados, muitas vezes sem nenhuma utilidade futura. Esses dados contêm informações que vão desde dados de compras ou vendas até eventos simples, que ocorrem todos os dias.

Surgiu, então, a necessidade de descoberta de conhecimento útil para a tomada de decisão gerencial nas grandes empresas, porém, até o início da década passada, as técnicas tradicionais de consulta a bancos de dados só ajudavam o homem a organizar informações de conhecimento que já era explícito na base de dados, enquanto o conhecimento implícito ficava escondido (Fayyad et al., 1996; Rezende, 2005).

Há algumas décadas, ferramentas estatísticas auxiliam a análise de dados, porém a maioria dessas ferramentas possibilitava apenas o uso de pequenas bases de dados e/ou não forneciam suporte a um processo completo de descoberta de algum conhecimento importante (Berry e Linoff, 2004). Observando o que estava ocorrendo, alguns pesquisadores, dentre eles Usama Fayyad, cunharam um termo chamado de Descoberta de Conhecimento em Bases de Dados ou KDD (*Knowledge Discovery in Databases*).

Segundo Fayyad et al. (1996), KDD é o processo completo de descoberta de conhecimento útil em base de dados. A etapa mais importante e conhecida desse processo é a mineração de dados, onde são utilizados algoritmos que vão extrair padrões dos dados.

O KDD pode ser aplicado com sucesso em diferentes ramos de atividades. Há aplicações na astronomia, para melhorar as análises de imagens. Na área de marketing, há aplicações na área de análise de bases de dados de clientes para verificar diferentes grupos de clientes e diferentes comportamentos de compras. O KDD é utilizado também para a detecção de fraudes em cartões de créditos, através da formação de perfis de compras de clientes. Na indústria de manufaturados, onde é possível identificar possíveis peças ou equipamentos com maior probabilidade de falhar ou quebrar (Fayyad et al., 1996). Esses são apenas exemplos de um vasto campo de aplicações.

## 2.1. O processo KDD

Diversas abordagens definem propostas de divisões das etapas do processo do KDD. Neste trabalho adotou-se a metodologia proposta por Fayyad et al. (1996). Antes de definir as etapas, é importante definir os tipos de usuários que vão trabalhar no processo. Rezende (2005) definiu três tipos de usuários:

- **Especialista do domínio:** é aquele usuário que detém o conhecimento do domínio em que o problema está sendo aplicado e vai auxiliar o analista da mineração de dados;
- **Analista de mineração de dados:** é o usuário que conhece profundamente as etapas e algoritmos que são utilizados no processo;
- **Usuário Final:** é aquele que vai utilizar o conhecimento extraído do trabalho do especialista e do analista para a tomada de alguma decisão estratégica.

Agora que já são conhecidos os usuários do processo, é possível explicitar as etapas. O processo é composto de cinco etapas: seleção, pré-processamento, transformação, mineração de dados e análise dos padrões encontrados, ou pós-processamento. A Figura 2.1 mostra a evolução destas etapas em uma escala linear.

## 2.2. Etapas do processo KDD

Antes de começar as etapas do processo de KDD, é imprescindível que haja um estudo minucioso sobre o domínio do problema em questão. Esse conhecimento sobre o domínio vai fornecer subsídios para todas as etapas seguintes do processo. Esse maior conhecimento do domínio deve ajudar a responder algumas perguntas, como: quais são as principais metas do processo? Quais critérios de desempenho são importantes? O conhecimento extraído deve ser compreensível a seres humanos ou deve ser um modelo do tipo caixa preta? Qual deve ser a relação entre simplicidade e precisão do conhecimento extraído? (Rezende, 2005).

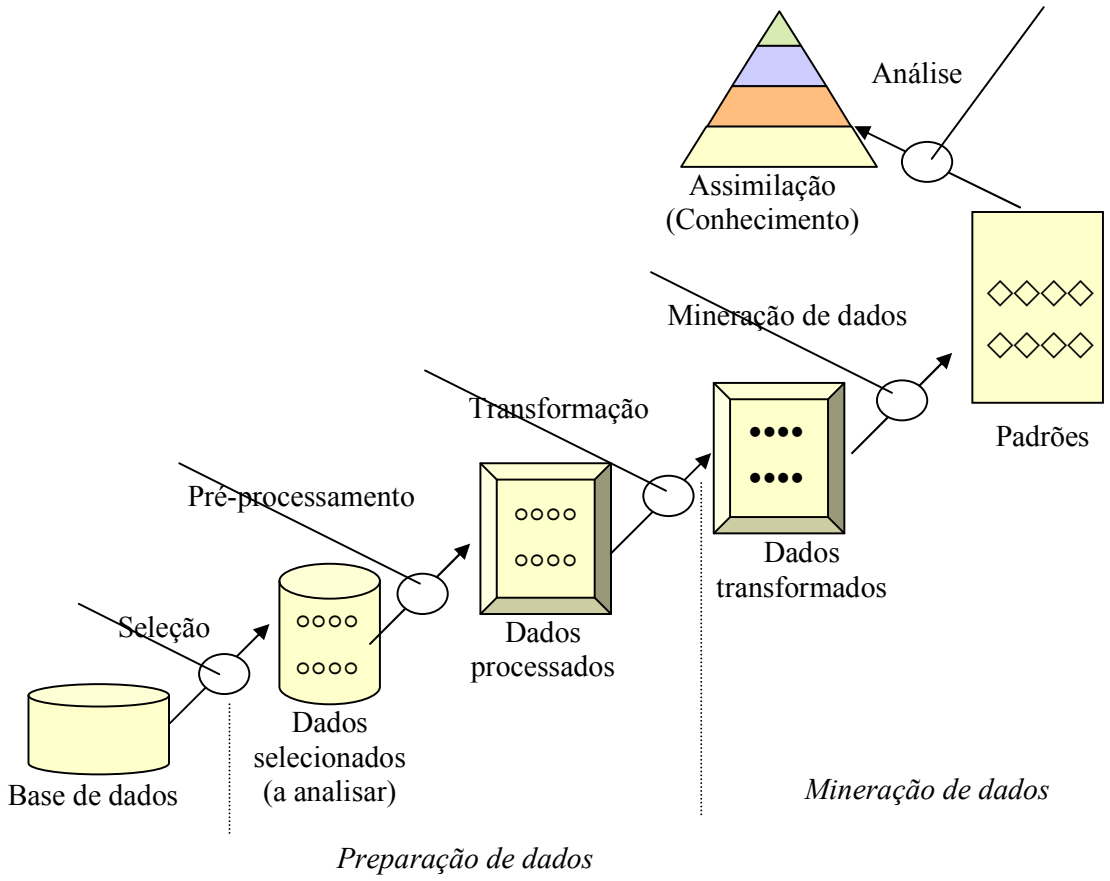


**Figura 2.1.** Etapas do processo KDD**.**

Fonte: Han e Kamber (2001)

### 2.2.1. Seleção dos dados

Com o problema ou domínio em questão bem definido, agora é importante ter os dados para a realização da mineração. Esses dados, em teoria, deveriam estar em um *data warehouse*, que é um grande repositório de dados. Os dados devem ser "limpos" e serem atualizados constantemente (Berry e Linoff, 2004).

A quantidade de dados que irá se utilizar no problema varia, dependendo do algoritmo que será utilizado e da complexidade dos dados. Geralmente aconselha-se a utilizar uma grande massa de dados, para descobrir algum conhecimento relevante na mineração de dados. Como essa massa de dados possui dados históricos, uma questão difícil de se resolver é a partir de qual época esses dados devem ser coletados. Em algumas vezes registros de um ou dois anos conseguem satisfazer as necessidades, e outras são

necessários registros de décadas passadas. O importante é que os registros que se vai utilizar contenham exemplos de todas as possíveis categorias de resposta que se deseja pesquisar (Berry e Linoff, 2004).

Outra questão é a escolha das variáveis de entrada. As variáveis de entrada são escolhidas de acordo com o conhecimento do domínio em questão. São as variáveis relevantes para serem utilizadas no algoritmo de mineração de dados. O modelo deve ser criado com poucas variáveis, sendo que essas variáveis podem representar até mesmo a combinação de uma ou mais variáveis da base inicial.

### 2.2.2. Pré-processamento dos dados

Após selecionar os dados com que se vai trabalhar, parte-se para uma etapa do processo, que dependendo de como os dados se encontram, pode ser a mais demorada de todo o processo. Conforme alguns autores esta etapa pode chegar até em 80% de todo o tempo do KDD. O pré-processamento é necessário porque geralmente as grandes bases de dados são suscetíveis a ruídos, dados faltantes e inconsistências. Eliminar esses problemas pode melhorar a qualidade dos resultados da mineração de dados (Han e Kamber, 2001).

A Figura 2.2 resume toda a etapa de pré-processamento dos dados, ilustrando o efeito de algumas técnicas na análise dos dados. Algumas destas técnicas de análise dos dados são discutidas em Han e Kamber (2001).

### 2.2.3. Transformação dos dados

Nesta etapa os dados são transformados ou consolidados em formas que são apropriadas para a utilização de algum algoritmo de mineração (Han e Kamber , 2001).

Um tipo de transformação dos dados é a normalização, onde os valores de uma variável quantitativa são modificados de forma que representem os mesmos padrões em uma escala reduzida de valores, como a escala no intervalo [0,0; 1,0] ou outras escalas.

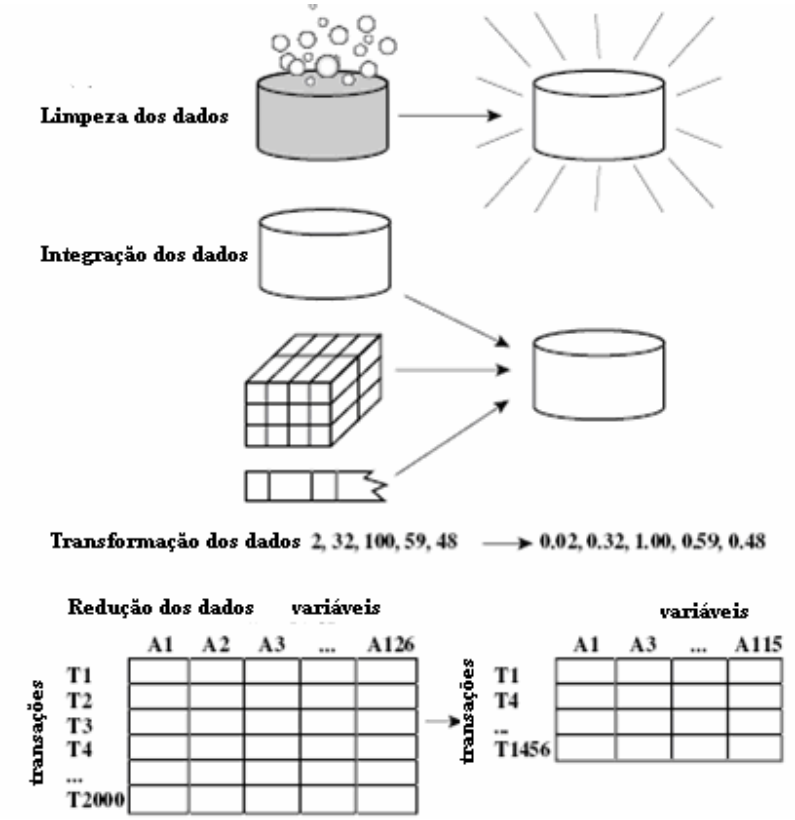


**Figura 2.2.** Etapas de pré-processamento e transformação de dados.

Fonte: Han e Kamber, 2001

### 2.2.4. Mineração dos dados

Esta é a etapa mais importante do processo de KDD, pois aqui os dados serão minerados para a obtenção de algum conhecimento. A etapa de mineração dos dados geralmente envolve repetidas iterações de algum algoritmo, realizando alguma tarefa de mineração (Fayyad et al., 1996). Segundo Rezende (2005), esta etapa se subdivide em três: escolha da tarefa a ser realizada, escolha do algoritmo para realizar a tarefa e, por fim, a extração dos padrões.

#### 2.2.4.1. Escolha da tarefa

A mineração de dados trabalha com o ajuste de modelos para descobrir determinados padrões em uma base de dados. Esses modelos ajustados aos dados têm o papel de inferir novos conhecimentos. Muitas das tarefas de mineração de dados são

baseadas em algoritmos e técnicas vindas do aprendizado de máquina, do reconhecimento de padrões e da estatística.

A descoberta de conhecimento pode ser dividida em duas áreas: predição e descrição. Na predição, onde se encontram as tarefas de classificação e regressão, o algoritmo vai encontrar novos padrões com base em exemplos existentes. Esses novos padrões vão predizer o comportamento futuro de alguma entidade. Na descrição, onde se encontram as tarefas de agrupamentos, sumarização e regras de associação, o algoritmo encontra padrões que possam ser compreendidos pelo ser humano. Estas duas áreas não têm um limite completamente definido: às vezes uma tarefa de predição realiza o trabalho de descrição e vice-versa (Fayyad et al., 1996).

**Classificação:** alguma função de aprendizagem mapeia ou classifica um conjunto de dados em um determinado número pré-definido de classes. Por exemplo, a classificação de consumidores em bons ou maus pagadores. Outro exemplo pode ser visto na Figura 2.3, onde uma classificação linear separa os registros em duas classes de empréstimos: concedidos e não concedidos;

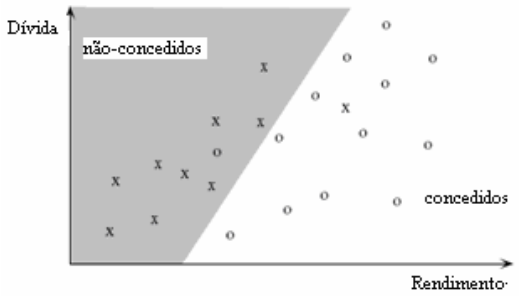


**Figura 2.3.** Classificação linear entre empréstimos concedidos e não concedidos.

Fonte: Fayyad et al., 1996.

**Regressão:** na regressão se constrói uma função que auxilia a descobrir (predizer) o valor de uma variável continua com base no valor de outra (s). Pode ser usada, por exemplo, para predizer o valor da variável salário, através de outras variáveis, como ocupação, idade, nível de instrução, experiência etc. A Figura 2.4 mostra um exemplo de uma função linear entre dívida e rendimento.

**Agrupamentos (clustering):** na análise de agrupamentos (*clustering analysis*) o objetivo do analista de mineração de dados é identificar um número finito de grupos que descrevam

os diferentes padrões encontrados nos dados. Os registros que pertencem a cada grupo devem ser bastante similares entre si, e dissimilares em relação aos registros de outros grupos. Aplicações podem ser feitas para se encontrar tipos diferentes de perfis de consumidores. A Figura 2.5 mostra o exemplo da classificação em 3 agrupamentos dos registros de empréstimos;

**Sumarização:** nas tarefas de sumarização, o analista de mineração de dados tem como objetivo encontrar uma descrição resumida de um subconjunto de dados, como, por exemplo, representar todas as variáveis pela sua média ou desvio padrão, ao invés de toda a massa de dados;

**Regras de associação:** essa tarefa consiste em encontrar dependências entre variáveis. Pode ser feita em dois níveis: no nível estrutural do modelo especifico, onde uma variável depende localmente das outras; e no nível quantitativo, onde se especifica a força das dependências utilizando alguma escala numérica.

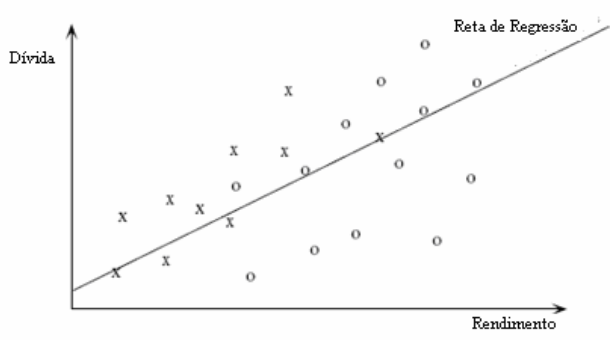


**Figura 2.4.** Regressão linear entre divida e rendimento.

Fonte: Fayyad et al., 1996.

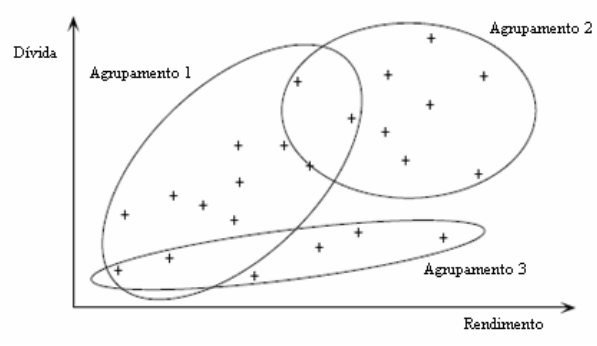


**Figura 2.5.** Agrupamentos dos dados de empréstimos.

Fonte: Fayyad et al., 1996.

#### 2.2.4.2. Escolha do algoritmo

Depois de escolhida a tarefa de mineração de dados, chega a vez de escolher o algoritmo que vai realizar esta tarefa. Para cada tarefa há diversos algoritmos disponíveis na literatura. Para a classificação, por exemplo, alguns dos algoritmos mais utilizados são as árvores de decisão, redes neurais artificiais *Multi Layer Perceptron* e redes bayesianas. Para a tarefa de agrupamentos, existem algoritmos estatísticos, algoritmos baseados em redes neurais, entre outros (ver Freeman e Skapura, 1991; Han e Kamber, 2001; Rezende, 2005).

Um aspecto importante na escolha do algoritmo, segundo Rezende (2005), é o ajuste dos parâmetros de entrada fornecidos para a execução deste algoritmo. Quando esses parâmetros são ajustados para encontrar soluções mais complexas do que se deseja, pode ocorrer o problema do super treinamento, ou *overfitting*. Já o contrário, ou seja, ajuste simples para objetivos complexos, pode ocorrer o problema de *underfitting*.

#### 2.2.4.3. Extração dos padrões

Esta é a etapa onde os dados vão ser realmente minerados. Neste momento o algoritmo escolhido vai ser utilizado em cima dos dados devidamente pré-processados e transformados, de forma que possam servir de entrada para o algoritmo. A saída desta etapa são os padrões encontrados na base de dados. Esses padrões vão ser devidamente analisados pelo especialista do domínio e pelo analista da mineração na próxima etapa, a de pós-processamento.

### 2.2.5. Análise dos padrões encontrados ou pós-processamento

Mesmo após os padrões terem sido encontrados pelo algoritmo de mineração, o processo de KDD ainda não termina. Os algoritmos de mineração às vezes podem gerar uma quantidade enorme de padrões. O analista de mineração, em conjunto com o especialista do domínio, tem que avaliar quais padrões realmente são importantes ou relevantes para o usuário final, ou para o cliente que vai tomar alguma decisão.

Um dos objetivos principais desta etapa é fazer com que o conhecimento obtido seja compreensível ao usuário, pois, às vezes os algoritmos expressam esse conhecimento de uma maneira muito técnica ou visualmente pobre, principalmente para o usuário leigo no

assunto. Poucas regras, com um número menor de condições são mais fáceis de serem entendidas.

Podem ser criadas medidas para avaliar os padrões encontrados. Existem medidas técnicas, que avaliam a estrutura dos padrões, e medidas subjetivas, em que se utilizam fatores como o conhecimento do usuário e do domínio. Por isso é importante a presença do especialista do domínio (Rezende, 2005).

Caso se observe, após a análise do conhecimento encontrado, que o conhecimento não é útil, ou não atende às expectativas do usuário final, deve-se retornar à etapa de mineração de dados e alterar os parâmetros do algoritmo para verificar se existem resultados mais favoráveis, pois nem sempre existe algum conhecimento útil para ser encontrado.

## 3. ANÁLISE DE AGRUPAMENTOS

Em muitos dos problemas reais de mineração de dados não se tem conhecimento do grupo a que pertence cada registro em uma base de dados. Então, para encontrar os padrões que representam esses grupos, pode se usar uma tarefa de mineração de dados chamada de análise de agrupamentos.

Esta tarefa é definida em Han e Kamber (2001) como o processo de reunir os dados em agrupamentos de forma que os registros em um mesmo agrupamento tenham uma alta similaridade entre si, e uma grande dissimilaridade em relação aos registros que estão em outros grupos. É importante seguir a premissa indispensável à mineração de dados, que é a de encontrar agrupamentos que indiquem alguma informação útil ao problema em questão.

Existem diversos algoritmos presentes na literatura, porém algumas propriedades devem ser observadas para que eles, de fato, sejam aplicáveis em mineração de dados (Han e Kamber, 2001):

- **Escalabilidade:** Alguns algoritmos de agrupamentos funcionam bem em pequenas bases de dados, porém a mineração de dados envolve grandes bases de dados. O algoritmo deve ser capaz de lidar com bases de dados com um grande número de registros, um grande número de grupos e um grande número de variáveis;
- **Habilidade de lidar com diferentes tipos de dados:** o algoritmo deve ser capaz de permitir o uso, por exemplo, de valores numéricos, binários ou categóricos;
- **Descobrir agrupamentos com tamanhos arbitrários:** as medidas de similaridade mais usadas tendem a construir agrupamentos de formato esférico, e de mesmo tamanho (com uma quantidade similar de registros). Seria desejável que os algoritmos pudessem construir agrupamentos com qualquer forma e tamanho;
- **Pouco conhecimento do domínio nos parâmetros de entrada:** o algoritmo deve ter um desempenho satisfatório mesmo quando o número de grupos não é conhecido;
- **Habilidade de lidar com dados problemáticos:** o algoritmo deve ser capaz de lidar com valores faltantes, valores discrepantes ou com dados com ruídos;
- **Insensividade:** o algoritmo deve ser indiferente à ordem com que os dados de entrada são apresentados. Diferentes ordens na entrada dos dados não devem fornecer resultados diferentes;

- **Alta dimensionalidade:** o algoritmo deve funcionar bem com um pequeno ou um grande número de variáveis de entrada;
- **Interpretabilidade:** as descobertas feitas pelo algoritmo devem ser de fácil compreensão para poderem ser reutilizadas no futuro.

Alguns artigos presentes na literatura, como em Zhang et al. (1996) e Chiu et al. (2001), utilizam uma medida de qualidade dos resultados ou **acurácia** do algoritmo. Em uma simulação, conhecendo o grupo a que cada registro pertence em uma base de dados, é possível verificar a quantidade de erros ou acertos deste algoritmo para esta determinada base de dados.

É importante salientar que estas seriam condições ideais que um algoritmo de Análise de Agrupamentos deveria obedecer, porém, geralmente, um algoritmo não consegue satisfazer a todas essas condições. O ideal seria utilizar um algoritmo que se aproxime destas condições.

### 3.1. Medidas de distância

Os algoritmos de Análise de agrupamento costumam adotar algum critério de distância para medir a proximidade dos registros em uma base de dados. Geralmente os registros utilizados em análise de agrupamentos estão dispostos como na Tabela 3.1.

**Tabela 3.1.** Forma usual dos dados de entrada em análise de agrupamentos.

| | Variáveis | | |
|---|---|---|---|
| Registro | $X_1$ | ... | $X_J$ |
| 1 | $x_{11}$ | ... | $x_{1j}$ |
| 2 | $x_{21}$ | ... | $x_{2j}$ |
| . | . | ... | . |
| . | . | ... | . |
| n | $x_{n1}$ | ... | $x_{nJ}$ |

Existem diversas medidas de distância utilizadas pelos algoritmos existentes. Algumas das utilizadas em Analise de Agrupamentos são apresentadas a seguir:

#### 3.1.1. Distância Euclidiana

Considerando o ponto $\vec{P} = (x_1, x_2)$ no plano, a distância de $\vec{P}$ à origem $\vec{O} = (0,0)$ é, baseada na idéia do teorema de Pitágoras, obtida como:

$$d(\vec{O},\vec{P}) = \sqrt{x_1^2 + x_2^2} \tag{3.1}$$

A distância euclidiana entre dois pontos arbitrários $\overrightarrow{P_1}$ e $\overrightarrow{P_2}$ (registros em uma base de dados) com coordenadas $\overrightarrow{P_1} = (x_1, x_2, .., x_n)$ e $\overrightarrow{P_2} = (y_1, y_2, .., y_n)$ é dada por (Johnson e Wichern, 2002):

$$d(\overrightarrow{P_1}, \overrightarrow{P_2}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + .. + (x_n - y_n)^2} \tag{3.2}$$

### 3.1.2. Distância de Manhattan

Outra medida de distância é a de Manhattan, que tem o seu nome derivado do padrão de grade retangular das ruas da cidade de Nova Iorque. É a simples soma das diferenças, em módulo, dos valores entre os pontos em cada eixo de coordenadas. Para os pontos $\overrightarrow{P_1}$ e $\overrightarrow{P_2}$, a distância de Manhattan pode ser representada como:

$$d(\overrightarrow{P1}, \overrightarrow{P2}) = | x_1 - x_2 | + | y_1 - y_2 | + .. + | y_n - y_n | \tag{3.3}$$

A vantagem em relação à distância euclidiana é que se considerando o valor das diferenças entre os pontos em cada coordenada do eixo de coordenadas, é menos provável que uma grande distância em uma única dimensão domine quase que completamente a distância total (Berry e Linoff, 2004).

### 3.1.3. Distância de log da verossimilhança

Esta medida de distância foi definida, inicialmente, por Banfield e Raftery (1993) e mostrada de uma maneira mais simples em Chiu et al. (2001). Em ambos artigos, considera-se que as variáveis quantitativas possuem uma distribuição normal e variáveis categóricas possuam uma distribuição multinomial. O artigo de Chiu et al. (2001) será usado como referência para as próximas definições.

Sendo $\vec{x}_i$, com i =1, .., n , um conjunto j-dimensional de n registros em um k-ésimo agrupamento, com k = 1, .., K, e $p(\vec{x} \mid \vec{\theta}_k)$ uma função densidade de probabilidade de $\vec{x}$ no agrupamento k, onde $\vec{\theta}_k$ é o vetor de parâmetros do modelo, a função do log da

verossimilhança é baseada na distribuição de probabilidade conjunta dos registros $\vec{x}_i$, a qual é dada por:

$$L = \sum_{k=1}^{K} \sum_{i \in I_k} \log p(\vec{x}_i \mid \vec{\theta}_k) = \sum_{k=1}^{K} L_k \tag{3.4}$$

onde $L_k$ representa a contribuição de um agrupamento k para o log da verossimilhança e $I_k = \{i: \vec{x}_i \in k\}$ é o conjunto de índices do agrupamento k.

Dado um conjunto de registros $\vec{x}_i$, a estimativa de máxima verossimilhança $\hat{\vec{\theta}}_k$ de $\vec{\theta}_k$ é o vetor que maximiza o log da verossimilhança L. Assume-se que os dados quantitativos provêm de variáveis aleatórias independentes com distribuições de probabilidade normais e as variáveis categóricas também são independentes e seguem distribuições de probabilidade multinomiais.

Considerando a suposição acima, o valor máximo do log da verossimilhança de L pode ser definido como:

$$\hat{L}_k = \sum_{k=1}^{K} (\hat{L}_{A_K} + \hat{L}_{B_K}) \tag{3.5}$$

onde $\hat{L}_{A_K}$ corresponde a contribuição das variáveis quantitativas e $\hat{L}_{B_K}$ corresponde a contribuição das variáveis categóricas. Essas contribuições são dadas por:

$$\hat{L}_{A_K} = -\frac{1}{2} n_k [k_A \{\log(2\pi) + 1\} + \sum_{j=1}^{J_A} \log(\hat{\sigma}_{kj}^2)] \tag{3.6}$$

$$\hat{L}_{B_K} = -n \sum_{j=1}^{J_B} \hat{E}_{kj} \tag{3.7}$$

onde $\hat{E}_{kj} = -\sum_{c=1}^{C} \left( \hat{q}_{kjc} \log \hat{q}_{kjc} \right)$ e $\hat{q}_{kjc} = \frac{n_{kjc}}{n_k}$

é a proporção de registros de uma categoria c, na variável j, no agrupamento k; e

$$\hat{\sigma}^2_{kj} = \frac{\sum_{i \in I_k} (x_{ikj} - \bar{x}_{kj})^2}{n_k}$$

representa a variância amostral da j-ésima variável no agrupamento k, e $n_k$ o número de registros do agrupamento k.

A medida de distância entre dois agrupamentos é baseada na redução no log da verossimilhança, resultante da união dos dois agrupamentos. A medida de distância da união de dois agrupamentos t e s, após algumas simplificações, é definida em Chiu et al. (2001) como:

$$d(t,s) = \tilde{\xi}_t + \tilde{\xi}_s - \tilde{\xi}_{<t,s>} \tag{3.8}$$

e

$$\tilde{\xi}_v = -n_v \{\frac{1}{2} \sum_{j=1}^{J_A} \log(\hat{\sigma}^2_{vj} + \Delta_j) + \sum_{j=1}^{J_B} \hat{E}_{vj}\} \tag{3.9}$$

onde $v = s,t$ e $\Delta_j$ é uma constante maior que zero que serve para os casos em que a variância do agrupamento é nula (por exemplo, quando o agrupamento possui apenas um registro). Em SPSS (2001), é proposto que a variância da variável j, calculada com todos os registros da base de dados, seja usada como o valor da constante $\Delta_j$.

### 3.2. Tratamento de variáveis

Em grandes bases de dados há dados de vários tipos: contínuos, discretos, ordinais, nominais são os mais comuns. Só que esses dados devem pertencer a um único padrão na hora de formar os agrupamentos, já que diferenças entre medidas, como quilograma e metro, por exemplo, não podem influir na hora da formação dos agrupamentos. Assim como valores mais altos em uma variável podem ter uma influência maior do que valores mais baixos em outras variáveis.

Quando uma base de dados possui variáveis de diferentes tipos, costuma-se transformar os dados para uma mesma escala ou intervalo. Nas próximas subseções é explicado que tratamento pode ser dado a alguns tipos mais comuns de variáveis, segundo Han e Kamber (2001).

### 3.2.1. Variáveis quantitativas

São os tipos de variáveis mensuráveis numericamente, podendo ser discretas, quando seus possíveis valores puderem ser listados, ou contínuas, quando os seus valores puderem assumir qualquer valor em um intervalo de dados (Barbetta, 2006). Em ambos os casos, dois tipos de transformações são usuais.

A primeira seria a padronização dos dados em relação à média e o desvio padrão. Para cada variável, define:

$$Z_i = \frac{x_i - \overline{X}}{S} \tag{3.10}$$

onde $Z_i$ representa o escore padronizado do i-ésimo registro, $x_i$ representa o valor original do i-ésimo registro, $\overline{X}$ representa a média dos valores de todos os registros e S o desvio padrão.

A segunda transformação seria a de normalizar os dados no intervalo [0,1]. Para esta transformação pode ser utilizada a fórmula (Berry e Linoff, 2004):

$$N_i = \frac{X_i - X_{MIN}}{X_{MAX} - X_{MIN}} \tag{3.11}$$

onde $N_i$ é o valor do registro normalizado, $X_{MIN}$ é o menor valor dentre todos os registros da variável e $X_{MAX}$ é o maior valor entre todos os registros da variável.

Essas transformações transformam os dados de modo que eles fiquem sem unidade de medida e, medidas com diferentes tipos de unidades, como peso e altura, não tenham influências diferentes na execução do algoritmo.

### 3.2.2. Variáveis categóricas nominais

Variáveis nominais são aquelas que, em geral, possuem um número C limitado de categorias, que não representam uma ordem lógica entre si, como as cores amarelo, verde e azul de uma variável cor, por exemplo.

Costuma-se criar uma variável binária (aquelas com valor 0 ou 1) para cada opção C de categorias da variável original. Então, no caso da variável cor, seriam criadas três novas variáveis: uma variável binária para a cor amarela, uma variável binária para a cor verde e uma para a cor azul. A variável binária teria o valor 1 nos casos em que o registro fosse o correspondente ao valor correto na variável original e zero nos outros casos. Um exemplo com a variável cor pode ser visto na Tabela 3.2.

**Tabela 3.2.** Exemplo de transformação da variável cor em três variáveis binárias

| Registro | Cor | CorAmarela | CorVerde | CorAzul |
|---|---|---|---|---|
| 1 | amarelo | 1 | 0 | 0 |
| 2 | verde | 0 | 1 | 0 |
| 3 | amarelo | 1 | 0 | 0 |
| 4 | azul | 0 | 0 | 1 |
| 5 | verde | 0 | 1 | 0 |

### 3.2.3. Variáveis categóricas ordinais

Variáveis ordinais, assim como as nominais, são aquelas que possuem um número limitado C de categorias, porém as categorias têm níveis de importância diferentes e representam alguma ordem lógica. Um exemplo seria o da variável nível de escolaridade, com as categorias ensino médio, graduação e pós-graduação. Nesse caso, há duas opções de transformação.

A primeira seria uma transformação similar à das variáveis nominais, criando uma variável binária para cada possível categoria na variável original. Só que neste caso, essas variáveis binárias devem representar uma ordem, como pode ser visto na Tabela 3.3.

**Tabela 3.3.** Exemplo de transformação da variável nível de escolaridade

| Registro | NivelEscolaridade | NivelEnsinoMedio | NivelGraduação | NivelPósGraduação |
|---|---|---|---|---|
| 1 | Graduação | 1 | 1 | 0 |
| 2 | Ensino Médio | 1 | 0 | 0 |
| 3 | Pós-Graduação | 1 | 1 | 1 |
| 4 | Ensino Médio | 1 | 0 | 0 |
| 5 | Graduação | 1 | 1 | 0 |

Observa-se, no caso da Tabela 3.3, que os registros que possuem o valor pós-graduação devem ter todas as variáveis binárias criadas com valor 1, indicando que é o maior nível entre os existentes, enquanto o menor, que é o valor Ensino Médio possui apenas a variável NivelEnsinoMedio com valor 1, e as outras recebem o valor 0.

Outra maneira de fazer transformações em variáveis ordinais é simplesmente transformar cada uma das C possíveis categorias em um valor numérico, dentro de um intervalo entre [0,1]. Para que isto seja possível, é necessário primeiramente quantificar de alguma forma cada uma das possíveis categorias com valores 1,2..até C, e depois aplicar a fórmula (Han e Kamber, 2001):

$$VT_i = \frac{x_i - 1}{C - 1} \tag{3.12}$$

onde $VT_i$ é o valor transformado no registro i, $x_i$ é o valor numérico correspondente à categoria do registro i e C é o número total de categorias da variável em questão.

A vantagem desta segunda transformação é que não é necessário criar um número grande de variáveis a mais, o que pode trazer um esforço computacional menor para o algoritmo de análise de agrupamentos que será utilizado.

Porém a desvantagem desta transformação é que nem sempre o valor numérico atribuído a cada categoria representa de maneira fiel as diferenças para as outras categorias. Por exemplo, se fossemos transformar três categorias A, B e C e atribuíssemos o valor 0 para a categoria A, 0,5 para a categoria B e 1 para a categoria C. A categoria C teria sempre o dobro de influência da categoria anterior e o triplo de influência da categoria logo abaixo, porém nem sempre isto ocorre na realidade.

### 3.3. Métodos de Análise de Agrupamentos

Existem diversos algoritmos de análise de agrupamentos na literatura. A escolha do algoritmo geralmente depende da base de dados que será utilizada e do propósito da análise. Esses algoritmos estão agrupados pelo método que eles utilizam para segmentar os dados.

Alguns dos métodos mais comumente encontrados na literatura, assim como alguns algoritmos que serão utilizados ou mencionados no restante deste trabalho são apresentados a seguir, como exemplo.

### 3.3.1. Métodos Hierárquicos

Métodos hierárquicos criam uma decomposição hierárquica, geralmente no formato de árvores, de uma dada base de dados. Uma vantagem desses métodos é que eles produzem resultados que são de simples entendimento e visualização. Podem ser classificados em aglomerativos e divisivos.

Os métodos hierárquicos aglomerativos (também chamados de *bottom-up*), começam com cada registro correspondendo a um agrupamento e sucessivas uniões são feitas até que todos os grupos se tornem apenas um, ou que uma determinada condição de parada seja satisfeita.

Os métodos hierárquicos divisivos (também chamados de *top-down*), começam com todos os registros dentro de um mesmo agrupamento. A cada nova iteração, esses agrupamentos vão se dividindo até que cada registro represente um agrupamento distinto, ou que uma condição de parada seja satisfeita.

Os métodos hierárquicos podem ter como saída os dendogramas, que são gráficos em formato de árvores (Bussab et. al, 1990). Nos métodos aglomerativos, o gráfico deve ser visto de baixo para cima (*bottom-up*) e nos divisivos de cima para baixo (*top-down*). O corte, ou seja, o ponto de parada, depende do problema especifico que se está trabalhando. Geralmente, bons pontos de cortes aparecem quando há um grande intervalo sem divisões ou uniões entre os registros. Na Figura 3.1 é possível observar um dendograma com os registros A, B, C, D, E e F. No eixo das ordenadas são colocadas as distância entre os registros e, no eixo das abscissas, a indicação dos registros.

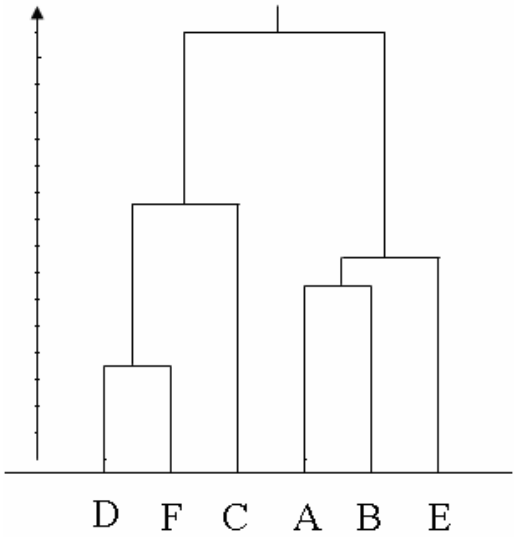


**Figura 3.1.** Exemplo de um dendograma

Um dos problemas nos métodos hierárquicos é que realizada uma união ou divisão, ela não pode ser desfeita. Então, decisões erradas não podem ser desfeitas. Outro problema é que fica difícil analisar agrupamentos com muitos elementos, em vista que a visualização do dendograma torna-se quase impossível com milhares de registros.

Geralmente os algoritmos hierárquicos armazenam uma matriz de distância com todas as dissimilaridades entre os registros, o que inviabiliza a sua utilização para grandes bases de dados, devido à necessidade de alocação de memória.

#### 3.3.1.1. AGNES

O algoritmo AGNES (*AGglomerative NESting*) proposto por Kaufman e Rousseeuw (2005) é um algoritmo hierárquico aglomerativo. Kaufman e Rouseeuw (2005) utilizam como medida de distância a medida de distância Euclidiana (3.1.1) ou a medida de distância de Manhattan (3.1.2), não indicando se outras medidas podem ser utilizadas.

No inicio, cada registro da base de dados representa um agrupamento isolado. Em uniões posteriores, esses registros vão se juntando até formar um só agrupamento. A cada etapa de uniões, os dois agrupamentos mais “similares” são unidos. Para verificar a similaridade dos agrupamentos, inicialmente o AGNES armazena a matriz de distâncias entre todos os registros contidos na base de dados fornecida para o trabalho do algoritmo.

Após a união dos dois primeiros registros em um agrupamento, uma nova matriz de distâncias é criada, com o novo agrupamento sendo considerado um simples registro (Kaufman e Rouseeuw, 2005).

As novas matrizes de distâncias são construídas à medida que as uniões ocorrem, até que haja apenas um agrupamento formado com todos os registros da base de dados.

Para representação gráfica dos agrupamentos, além do dendograma, foi criada uma nova representação chamada de *banner* (Kaufman e Rouseeuw, 2005) como o exemplo da Figura 3.2. O *banner* é uma representação gráfica que indica a união de registros. O *banner* consiste de estrelas e faixas. As estrelas indicam a união entre os registros e as faixas representam as repetições de nomes desses registros. Sempre lido da esquerda para a direita, o *banner* representa toda a informação de um dendograma.

Na figura 3.2, o espaço em branco na esquerda representa a fase inicial, onde todos os registros estão separados. No nível 2.0 é possível ver o início de uma faixa contendo três linhas. A primeira dessas linhas (AAA+AAA+...) é uma repetição de três caracteres que representam o registro *a*, separados pelo símbolo +. Na terceira linha, BBB representa o registro *b*. A segunda linha mostra uma seqüência de estrelas (*****), que indicam que os registros em questão estão sendo unidos, a partir do nível 2,0. Como se trata de um algoritmo aglomerativo, os registros que se unem uma vez estarão juntos até o final da formação dos agrupamentos. A formação de agrupamentos vai até o nível 7,83, onde os registros *a*, *b*, *c*, *d*, *e* formam um único agrupamento.

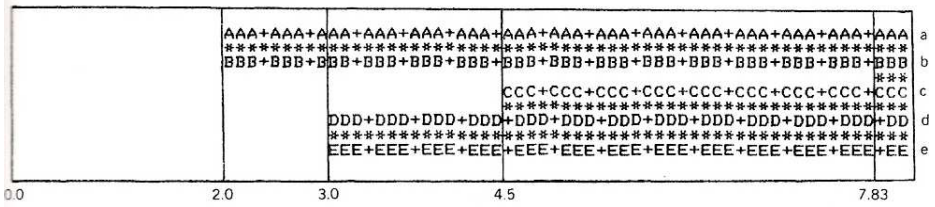


**Figura 3.2.** Visualização de um *banner*

### 3.3.1.2. DIANA

O algoritmo DIANA (*DIvisive Analysis*) proposto por Kaufman e Rouseeuw (2005) é um algoritmo hierárquico divisivo. Seu funcionamento é parecido com o do algoritmo AGNES, só que funciona de maneira inversa.

Em teoria, um algoritmo hierárquico divisível não seria viável pelo alto número de possibilidades de divisão dos elementos de um agrupamento em dois agrupamentos, número este que chegam à ordem de $2^{n-1}-1$ possibilidades. Este número cresce exponencialmente conforme o número de registros da base, o que tornaria inviável a sua aplicação. O Algoritmo DIANA funciona considerando apenas algumas possibilidades de divisão, já que a maioria das possibilidades costuma ser inapropriada (Kaufman e Rouseeuw, 2005).

Inicialmente há um único agrupamento, com todos os registros. Assim como o algoritmo AGNES, o algoritmo DIANA armazena a matriz de distâncias entre os registros, só que o registro mais dissimilar, ou mais distante de todos os outros, é separado do agrupamento. Para verificar a maior dissimilaridade, é calculada a distância média de cada registro em relação a todos os outros. O registro com a maior distância média sai do agrupamento maior e cria um outro agrupamento chamado *splinter group* (Kaufman e Rouseeuw, 2005).

Calculam-se as distâncias médias de cada registro para todos os registros que estão fora do *splinter group* e, depois, calcula-se a distância média de cada registro em relação aos registros que estão no *splinter grou*p. Finalmente, calcula-se a diferença entre essas duas distâncias. O registro que possuir o maior valor desta diferença se junta ao *splinter group*, desde que a diferença seja positiva. Quando não há mais registros com diferenças positivas, o *splinter group* torna-se um agrupamento dos dados e o processo de divisão continua da mesma maneira até que cada registro seja um agrupamento isolado.

Para visualizar a formação dos agrupamentos, assim como no algoritmo AGNES, pode ser utilizado o *banner*, ou até mesmo o dendograma.

### 3.3.2. Métodos de Partição

Dado um número n de registros em uma base de dados e um número K de agrupamentos que se deseja construir, este método constrói K partições de dados, onde cada partição representa um agrupamento. Inicialmente um número K de partições é criado e os registros são alocados e, depois, realocados em cada agrupamento de forma interativa.

Em alguns algoritmos existe uma função de custo, que auxilia a determinar quando o algoritmo deve parar o seu processamento.

Um critério para verificar se a adequação de um agrupamento é verificar se os registros que estão nesse agrupamento estão próximos entre si, e distantes dos de outros agrupamentos. Essa proximidade é medida se utilizando alguma medida de distância.

**3.3.2.1. K-médias**

O algoritmo K-médias é um dos mais comumente utilizados para se obter agrupamentos. Inicialmente proposto por MacQueen (1967), o "K" do nome K-médias refere-se ao número de agrupamentos que o usuário pretende encontrar, número esse definido anteriormente, com base na proximidade de que os registros de uma base de dados estão dispostos (Berry e Linoff, 2004).

O vetor de médias dos valores dos registros pertencentes ao agrupamento pode ser visto como o centro de gravidade do agrupamento, ou simplesmente centróide. Inicialmente, são selecionados de forma aleatória K registros, cada um representando a média inicial de cada agrupamento. Cada um dos registros restantes é introduzido ao agrupamento mais similar, baseado na distância entre o registro e o centróide. Em seguida, são calculados os novos centróides dos agrupamentos. Isto é feito de maneira interativa até que não haja mais mudanças nas alocações dos registros entre os agrupamentos.

Após o término das alocações dos registros nos agrupamentos, um problema comum é definir se eles convergiram de forma correta. Um critério comumente usado é a soma dos quadrados dos erros (SQE), definida como:

$$SQE = \sum_{i=1}^{K} \sum_{x_{ij} \in k} (x_{ij} - \bar{x}_j)^2 \qquad (3.13)$$

onde $x_{ij}$ representa a observação da j-ésima variável no i-ésimo registro e $\bar{x}_j$ é a média das observações da variiável j no agrupamento k.

O algoritmo segue adiante:

1. Escolha arbitrariamente K-registros como os centróides iniciais dos agrupamentos;

2. (Re) Coloque cada registro no agrupamento que lhe é mais similar, baseado na distância entre o objeto e o centróide de cada agrupamento;

3. Atualize os centróides dos agrupamentos;

4. Repita 2 e 3 até não haver mais mudanças.

Um dos problemas deste algoritmo é que não se conhece previamente o número K de grupos existentes na base de dados. Uma alternativa é executar o algoritmo com diferentes números de grupos iniciais e comparar os resultados utilizando alguma medida estatística como, por exemplo, $\lambda$ de Wilks (ver Johnson e Wichern, 2002).

#### 3.3.2.2. K-*medoid*

O algoritmo K-médias é sensível a valores extremos em uma base de dados. O algoritmo K-*medoid* propõe que o registro que está mais centralmente localizado no agrupamento, de acordo com alguma medida de distância (o autor propõe que seja utilizada a distância Euclidiana ou de Manhattan), seja utilizado como elemento representativo do agrupamento. Esse elemento representativo é chamado de *medoid*. Daí, então, o algoritmo funciona baseado no princípio de minimizar a soma das dissimilaridades entre cada registro e o seu correspondente *medoid* (Han e Kamber, 2001).

O algoritmo é dividido em duas fases: a primeira, chamada de *BUILD* e a segunda chamada de *SWAP*. Na primeira fase os agrupamentos iniciais são obtidos pela escolha sucessiva de *medoids* até que K *medoids* sejam encontrados.

Primeiro deve ser calculada a matriz de distâncias entre todos os registros da base de dados. O primeiro *medoid* escolhido é aquele cuja soma das distâncias em relação a todos os outros registros seja a menor possível. Este registro é o que está mais centralmente localizado entre todos os outros registros. A cada iteração do algoritmo, um novo *medoid* é selecionado na fase chamada *BUILD*, a seguir (Kaufman e Rosseeuw, 2005):

1. Selecionar um novo registro *i*, que será o candidato a *medoid* e que ainda não foi selecionado;
2. Selecionar um registro *l*, diferente dos registros que já foram selecionados e diferente do registro *i*, e calcular a sua distância em relação ao registro mais similar dentre os que

foram anteriormente selecionados, chamada de distância $D_l$; e calcular a sua distância em relação ao registro i selecionado, chamada de distância d(i, *l*);

3. Calcular a diferença entre as duas distâncias, $D_l - d(l,i)$. Se o valor da diferença for positivo, então o registro *l* irá contribuir para decidir na escolha do registro i, utilizando:

$$C_{li} = \max(D_l - d(l,i),0); \quad (3.14)$$

4. Voltar ao passo 2 até que não haja mais registros *l* que não tenham sido selecionados;
5. Calcular o ganho total, ou função de custo, ao selecionar o registro *i*:

$$\sum_l C_{li}; \quad (3.15)$$

6. Voltar ao passo 1 até que não haja mais registros *i* não selecionados.

O *medoid i* escolhido é aquele que obtiver o maior valor do somatório $\sum_l C_{li}$.

Esse processo continua até que K registros tenham sido encontrados. Na segunda fase do algoritmo há uma tentativa de melhorar o conjunto de *medoids* escolhidos e, depois, melhorar os agrupamentos formados por esses *medoids*. Esta segunda etapa é realizada considerando, para cada *medoid i* selecionado e um registro *h* não selecionado, todos os pares de registros (*i*, *h*). Então é determinado qual o efeito de ser realizada uma troca (*swap*) entre os registros, através dos seguintes passos:

1. Considerar um registro *i*, dentre os medoids já escolhidos.
2. Considerar um registro *h*, diferente dos medoids já escolhidos.
3. Considerar um registro *l* não-selecionado, diferente dos medoids já escolhidos e diferente de *h*, e calcular a sua contribuição $C_{lih}$ para a troca:

    a) Se *l* é mais distante de *i* e *h* do que um dos outros *medoids*, $C_{lih} = 0$.

    b) Se *l* não está mais distante de *i* do que de outro *medoid*, duas situações devem ser consideradas:

        s1) *l* está mais próximo de h do que o segundo *medoid* mais próximo. Neste caso, a contribuição de l na troca entre i e h é:

$$C_{lih} = d(l,h) - d(l,i) \tag{3.16}$$

s2) *l* é pelo menos tão distante de h quanto o segundo *medoid* mais próximo. Neste caso a contribuição de l na troca é:

$$C_{lih} = E_l - D_l \tag{3.17}$$

onde $E_l$ representa a distância entre l e o segundo *medoid* mais similar

c) *l* está mais distante do registro i do que pelo menos um dos outros *medoids*, mas mais próximo de h do que qualquer outro *medoid*. Nesse caso, a contribuição de *l* na troca é:

$$C_{lih} = d(l,h) - D_l \tag{3.18}$$

4. Calcular o ganho da troca, ou função de custo:

$$T_{ih} = \sum_l C_{lih} \tag{3.19}$$

5. Voltar ao passo 2 até que não haja mais registros *h* para serem selecionados.

6. Selecionar o par (i, h) que tiver o menor valor de $T_{ih}$.

7. Se o valor calculado no passo 6 é negativo, então a troca é realizada e se retorna ao passo 1 para verificar se novas trocas precisam ser realizadas. Se o valor é positivo ou igual à zero, o algoritmo pára.

Cada um dos *medoids* escolhidos representa um agrupamento e os registros restantes devem ser adicionados ao agrupamento cujo *medoid* for mais semelhante, de acordo com a medida de distância escolhida.

Como exemplo, a Tabela 3.4 mostra seis registros.. Serão supostos K = 2 agrupamentos.

**Tabela 3.4.** Registros como exemplo para o algoritmo K-*medoid*

| | Variáveis | |
|---|---|---|
| Registros | X | Y |
| 1 | 2 | 10 |
| 2 | 4 | 9 |
| 3 | 5 | 11 |
| 4 | 7 | 6 |
| 5 | 10 | 5 |
| 6 | 5 | 3 |

Um gráfico de dispersão bidimensional com os registros da Tabela 3.4 pode ser visto na Figura 3.3:

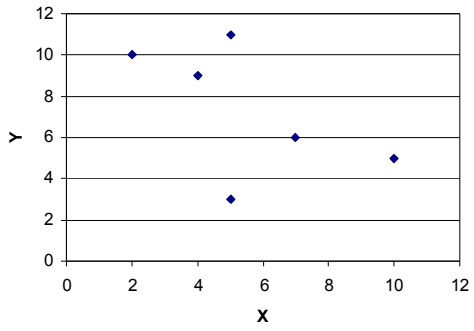


**Figura 3.3.** Gráfico dos pontos da Tabela 3.4

Primeiramente, calcula-se a matriz de distâncias entre os registros, considerando a medida de distância euclidiana:

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 0,0 | 2,2 | 3,2 | 6,4 | 9,4 | 7,6 |
| 2 | 2,2 | 0,0 | 2,2 | 4,2 | 7,2 | 6,1 |
| 3 | 3,2 | 2,2 | 0,0 | 5,4 | 7,8 | 8,0 |
| 4 | 6,4 | 4,2 | 5,4 | 0,0 | 3,2 | 3,6 |
| 5 | 9,4 | 7,2 | 7,8 | 3,2 | 0,0 | 5,4 |
| 6 | 7,6 | 6,1 | 8 | 3,6 | 5,4 | 0,0 |
| soma | 28,8 | 21,9 | 26,6 | 22,8 | 33 | 30,7 |

O registro 2 é aquele que possui a menor soma de distâncias para os outros registros, então será escolhido como o primeiro *medoid*.

Para escolher o segundo *medoid*, inicia-se a fase do algoritmo chamada de *BUILD*. Calculando se o valor de $\sum_{l} C_{li}$ para cada um dos registros restantes, obtêm-se os valores da Tabela 3.5. Para o valor de $i = 1$, por exemplo, variando o valor de $l$ para os registros restantes (desconsiderando o registro 2 já selecionado), o valor da diferença $D_l - d(l,i)$ sempre tem resultado negativo, e por isso a contribuição para o somatório tem o valor zero.

**Tabela 3.5.** Valores dos somatórios $\sum_{l} C_{li}$

| Registro i | $\sum_{l} C_{li}$ |
|---|---|
| 1 | 0,0 |
| 3 | 0,0 |
| 4 | 6,5 |
| 5 | 1,7 |
| 6 | 2,4 |

Como o registro 4 é o que possui o maior valor do somatório, ele é o escolhido para ser o segundo *medoid*.

Para verificar se os dois *medoids* escolhidos foram os mais adequados, é realizada em seguida a fase de *SWAP*. Calculando-se o valor de $T_{ih}$ para o medoid 2 chega-se ao valor 16,8 e para o medoid 4 o valor 17,0. Como os dois valores são positivos, o algoritmo pára ai, não há troca, e os registros restantes são inseridos ao agrupamento com o *medoid* mais próximo. A formação dos agrupamentos construídos ficou como na Figura 3.4:

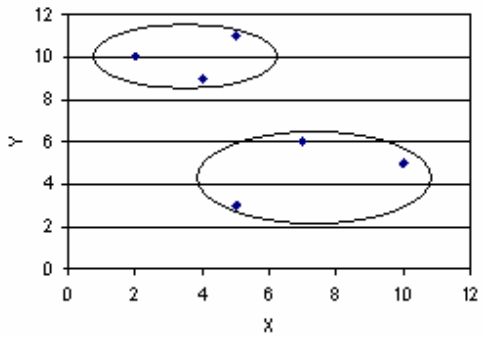


**Figura 3.4.** Agrupamentos construídos com o algoritmo K-*medoid* para K = 2

Assim como o algoritmo K-médias, o algoritmo K-*medoid* exige conhecimento prévio do número K de agrupamentos. Mais detalhes em Han e Kamber (2001) e Kaufman e Rosseeuw (2005).

### 3.2.2.3. CLARA

O algoritmo CLARA (*Clustering LARge Applications*) é uma variante do método K-medoid, adaptado para a análise (ou mineração) de grandes bases de dados, situação em que o algoritmo K-medoid torna-se inviável devido ao tempo de processamento crescer exponencialmente com o número n de registros.

Como alternativa, Kaufman e Rosseeuw (2005) propõem a realização de amostragens para a realização do K-medoid. O algoritmo CLARA, inicialmente, separa cinco (ou mais) amostras da base de dados. O tamanho dessas amostras depende do número K de agrupamentos. Os autores do algoritmo sugerem 40 + 2K registros para cada amostra.

Após as amostras terem sido criadas, elas são particionadas em K agrupamentos, com o mesmo algoritmo utilizado no K-medoid. Para verificar a qualidade dos agrupamentos gerados pelo algoritmo, é calculada a distância entre cada registro da base de dados e o seu respectivo medoid (a medida de distância euclidiana pode ser utilizada neste caso). Após isto, todos os valores de distâncias são somados e o valor desta soma é dividido pelo número de registros. O resultado é chamado de distância média.

A amostra que tiver menor distância média entre todas as amostras selecionadas é escolhida, e os registros que não fizeram parte desta amostra são alocados nos K agrupamentos previamente construídos, de modo que o registro que estava fora da amostra é alocado no agrupamento em que o medoid esteja mais próximo, de acordo com alguma medida de distância (Kaufman e Rosseeuw, 2005).

A vantagem desse algoritmo em relação ao K-médias é que ele é mais robusto na presença de valores discrepantes e produz agrupamentos com formas mais gerais do que o K-médias.

#### 3.3.2.4. CLARANS

O algoritmo CLARANS (*Clustering Large Applications based on RANdomized Search*), proposto em Ng e Han (1994) é uma adequação do algoritmo K-medoid para grandes bases de dados, assim como o algoritmo CLARA.

O CLARANS também não utiliza o total da base de dados para encontrar os agrupamentos, porém, diferentemente do CLARA, onde as amostras são obtidas no início do processo, o CLARANS realiza uma nova amostragem a cada iteração do algoritmo. Isto traz o benefício de não excluir possíveis agrupamentos que fiquem de fora da amostragem inicial.

O algoritmo possui dois parâmetros de entrada: o primeiro é chamado de *numlocal* e representa o custo desejado para selecionar uma determinada amostra naquela iteração. A amostra será criada de acordo com o tamanho especificado no segundo parâmetro, chamado *maxneighbor*. O algoritmo pára quando a iteração atingir o custo especificado em *numlocal*. A função de custo pode ser a mesma utilizada no algoritmo K-medoid, definida em (3.15) e (3.19).

**3.3.2.5. K-modas**

O algoritmo K-modas é uma extensão do algoritmo das k-médias para o domínio de dados categóricos. Proposto por Huang (1998), o algoritmo propõe uma nova medida de dissimilaridade entre os registros de uma base de dados, substituindo a média dos agrupamentos pela moda. É usada a freqüência de ocorrências de cada categoria em uma variável categórica para atualizar as modas no processo de agrupar os dados, com o objetivo de minimizar uma função de custo do processo, a qual é discutida mais a frente, nesta seção. O K-modas só trabalha com dados categóricos, então se houver também variáveis contínuas, elas devem ser discretizadas em categorias.

A medida de distância considerada é a dissimilaridade entre dois registros $\vec{x}$ e $\vec{y}$, a qual é dada pelo número de categorias de resposta não coincidentes. Os registros que possuem um menor número de variáveis que não coincidem são os mais similares. Formalmente (Huang, 1998):

$$d(\vec{x},\vec{y}) = \sum_{j=1}^{J} \delta(x_j, y_j) \tag{3.20}$$

onde:

$$\delta(x_j, y_j) = \begin{cases} 0 & x_j = y_j \\ 1 & x_j \neq y_j \end{cases}$$

e $x_j$ , $y_j$ representam o valor na j-ésima variável de cada um dos registros.

A moda de um agrupamento $\mathrm{R} = [\overrightarrow{\mathrm{x}_1}, \overrightarrow{\mathrm{x}_2}, .., \overrightarrow{\mathrm{x}_\mathrm{n}}]$ é um vetor $\vec{Q} = [q_1, q_2, .., q_J]$ , que miniminiza o somatório de distâncias em relação aos $\overrightarrow{x_i}$ registros do agrupamento, como na expressão a seguir:

$$\min \sum_{i=1}^{n} d(\vec{Q}, \vec{x_i}) \tag{3.21}$$

A função de custo dos agrupamentos no K-modas, *E*, pode ser definida como o somatório das distâncias dos registros de um determinado agrupamento em relação a uma determinada moda. Esta função auxilia a minimizar a distância entre os registros no mesmo agrupamento, assim como no algoritmo das K-médias. Matematicamente pode ser definida como:

$$E = \sum_{j=1}^{J} \sum_{i=1}^{n} x_{ij} d(x_{ij}, \overrightarrow{Q_j}) \tag{3.22}$$

onde $x_{ij}$ é o valor da j-ésima variável no registro i da base de dados e $d(\vec{x_i}, \overrightarrow{Q_j})$ é a distância desse registro à moda daquele agrupamento, medida como em (3.20).

### 3.3.3. Métodos baseados em densidade

A idéia geral destes métodos é que o agrupamento continue crescendo até que a densidade (o número de registros na vizinhança deste agrupamento) exceda algum limiar. Cada registro neste agrupamento tem de estar dentro de um determinado raio. Os métodos baseados em densidade podem ser utilizados para detectar valores discrepantes em uma base de dados.

O DBSCAN (*Density Based Spatial Clustering of Applications with Noise*), algoritmo proposto por Ester et al. (1996), é baseado na noção de densidade dos agrupamentos e auxilia na construção de agrupamentos de tamanhos e formas variados. Cada registro em um agrupamento pertence a uma determinada vizinhança de raio R, que não pode ultrapassar um determinado limite. O tamanho da vizinhança é determinado por alguma medida de distância, como as citadas na seção 3.1. Normalmente é utilizada a distância euclidiana.

A vizinhança de um registro, denotada V, deve possuir um número mínimo de registros (chamados de MinPts). Porém, é importante considerar que há dois tipos de registros: aqueles que estão no interior do agrupamento e aqueles que estão na borda ou no limite do agrupamento. Os registros do interior do agrupamento costumam ter uma

vizinhança V de tamanho bem maior que os registros que estão na borda (Ester et al., 1996).

O algoritmo DBSCAN parte da premissa de que V e MinPts não representam conjuntos vazios, ou seja, que os agrupamentos possuem registros e que ruídos são aqueles registros que não pertencem a nenhum agrupamento. Uma desvantagem do algoritmo é que o conhecimento dos parâmetros V e MinPts não é trivial para todos os agrupamentos da base de dados, então costuma-se utilizar os mesmos valores dos parâmetros para todos os agrupamentos. Os autores também propõem algumas heurísticas para determinar esses parâmetros. Mais detalhes em Ester et al. (1996).

### 3.3.4. Métodos baseados em grade

Os métodos baseados em grade quantificam os objetos do espaço em um número finito de células que formam uma estrutura de grade. A vantagem deste método é o baixo tempo de processamento, já que ele não depende do número de registros da base de dados, e sim do número de células em cada dimensão do espaço quantificado.

O algoritmo *Wave Cluster*, proposto por Sheikholeslami et al. (1998), considera um espaço multidimensional de dados como um sinal multidimensional e aplica técnicas de processamentos de sinais, como as transformações *wavelet* para converter o espaço multidimensional em pequenas estruturas em formato de grade. Com isso, o algoritmo pode detectar agrupamentos em qualquer formato, funciona bem com valores discrepantes e independe da ordem em que os dados de entrada são submetidos a ele.

Observou-se que registros de dados multidimensionais podem ser representados em um espaço J-dimensional de características. As características são os atributos numéricos de um registro que podem ser representados por um vetor, onde cada atributo corresponde a um elemento numérico deste vetor. Esses vetores de características podem ser representados em uma área no espaço, que pode ser chamada de espaço de características, onde cada dimensão corresponde a uma das características.

Sheikholeslami et al. (1998) propõe o espaço de características através de uma perspectiva de processamento de sinais, com o espaço de características representando um sinal J-dimensional. As partes de alta freqüência de um sinal correspondem às regiões do

espaço de características onde há uma rápida mudança na distribuição dos casos ou registros, ou seja, correspondem aos limites dos agrupamentos. As partes de baixa freqüência que tem alta amplitude correspondem ao espaço de características onde os casos estão concentrados, ou seja, aos agrupamentos em si.

A técnica de transformação *wavelet* decompõe um sinal em diferentes bandas, como por exemplo, banda de alta freqüência e banda de baixa freqüência. Mais detalhes sobre o algoritmo podem ser obtidos em Gholamhosein et al. (1998).

### 3.3.5. Métodos baseados em modelo

Esses métodos procuram otimizar o beneficio que um determinado conjunto de dados pode ter com algum modelo matemático. Segundo Han e Kamber (2001), esses métodos seguem basicamente duas abordagens: a estatística e a de redes neurais. Abordaremos adiante um algoritmo de Redes Neurais.

Os SOMs são inspirados na estrutura fisiológica do córtex cerebral, onde observou-se que neurônios que respondem a padrões similares costumam a se agrupar próximos uns aos outros (Freeman, 1991). O modelo desta rede pode ser observado na Figura 3.5.

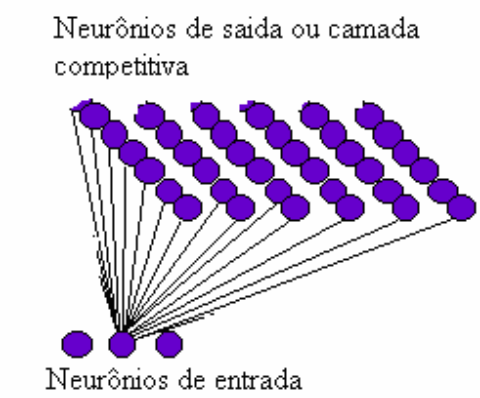


**Figura 3.5.** Estrutura dos SOMs

Na Figura 3.5 observa-se a existência de duas camadas, uma de entrada e outra de saída, também chamada de camada competitiva. Na camada de entrada estão os dados que serão inseridos na rede. Cada neurônio desta camada está ligado a todos os neurônios da camada competitiva, e cada uma dessas conexões possui um peso.

Quando uma entrada, ou registro, é apresentada à rede, calcula-se, através de uma medida de distância, qual neurônio está mais “próximo” dessa entrada. Esse neurônio é

chamado de vencedor e tem seus pesos atualizados para representar a proximidade ao registro de entrada. Este neurônio vencedor possui uma vizinhança, que vai diminuindo conforme as iterações ou épocas do treinamento da rede. Esses vizinhos também têm seus pesos atualizados, porém com menor influência do neurônio vencedor, conforme a distância aumenta em relação a este neurônio vencedor. Por fim, a expectativa é que quando um registro que não tenha sido anteriormente submetido à rede neural seja apresentado para a rede, os neurônios que possuem os pesos ajustados para aquele determinado padrão tenham os pesos com valores mais próximos aos valores do novo registro de entrada.

### 3.4. Comparações entre os algoritmos

Em Prass (2004), foram feitos testes comparativos com os algoritmos AGNES, DIANA, K-médias, K-*medoid* e DBSCAN.

Foi criada uma base de dados sintética com 210 mil casos e 3 variáveis quantitativas, divididos em 3 agrupamentos iguais de 70 mil registros cada. Também foi utilizada uma base de dados real, também com 270 mil registros, com informações sobre imóveis existentes em uma cidade. Foram utilizadas 6 variáveis categóricas, que posteriormente tiveram o valor de cada categoria transformada para valores entre 0 e 1. Era de conhecimento que a base de dados real estava dividida em seis agrupamentos.

Primeiramente, Prass (2004) fez comparações utilizando amostras de 5 mil registros das bases de dados. Os algoritmos K-médias e K-*medoid* foram executados em um tempo muito menor que os outros, tanto para a base real, quanto para a base simulada, como é possível visualizar na Tabela 3.6. Para os algoritmos de partição, o número final de agrupamentos foi fornecido, enquanto que os hierárquicos DIANA e AGNES, assim como DBSCAN encontraram um número diferente de possíveis agrupamentos.

Para verificar a acurácia, ou qualidade dos resultados, foram utilizados os algoritmos K-médias, K-*medoid* e o DBSCAN. O primeiro e o segundo tiveram um desempenho melhor que o DBSCAN, com um percentual de acerto de quase 100%. Para a base de dados real, nenhum dos três algoritmos teve uma taxa de acerto alta, porém o que mais se destacou foi o algoritmo K-*medoid*, com 28,5 % de acertos.

**Tabela 3.6.** Resultado das comparações com as amostras

| | Amostra da base real | | Amostra da base simulada | |
|---|---|---|---|---|
| Algoritmo | Tempo(mm:ss) | Nº de Grupos | Tempo(mm:ss) | Nº 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-MÉDIAS | 00:01 | 6 | 00:01 | 3 |
| K-MEDOID | 00:01 | 6 | 00:01 | 3 |
| DBSCAN | 06:45 | 9 | 05:22 | 3 |

Fonte: Prass, 2004.

Foram feitas, em seguida, comparações utilizando as bases de dados completas. Apenas os algoritmos de partição K-médias e K-*medoid* foram utilizados, já que os outros algoritmos não se mostraram capazes de serem executados para grandes bases de dados. O algoritmo K-*medoid* obteve um desempenho ligeiramente melhor que o algoritmo das K-médias, tanto para os dados da base simulada como para os dados da base real, como pode ser visto na Tabela 3.7:

**Tabela 3.7.** Resultado das comparações dos algoritmos de partição nas bases de dados completas.

| Base de dados | Algoritmo K-médias | | Algoritmo K-*medoid* | |
|---|---|---|---|---|
| | 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% |

Fonte: Prass, 2004.

Mingoti e Lima (2006) geraram amostras com diferente número de grupos, quantidade de variáveis, nível de correlação entre as variáveis, nível de superposição entre grupos e presença ou não de registros discrepantes, para avaliar alguns algoritmos hierárquicos e de partição, incluindo também os SOMs. Foram simuladas 2530 bases de dados.

Os SOMs não tiveram uma performance adequada, considerando a acurácia dos seus resultados em comparação com a acurácia dos outros algoritmos para a maioria das bases de dados, especialmente com diferentes números de variáveis e de agrupamentos. Os algoritmos de partição e os hierárquicos tiveram desempenho superior, porém equivalente entre si.

# 4. ALGORITMOS BIRCH E AGRUPAMENTO EM DUAS ETAPAS

O foco deste capítulo é explicar o funcionamento do algoritmo de Agrupamento em Duas Etapas (ADE), proposto em Chiu et al. (2001). Este algoritmo tem um forte apelo por ser capaz de reduzir um grande arquivo de dados, lido seqüencialmente, em pré-agrupamentos, e possuir estatísticas suficientes sobre esses agrupamentos para calcular o valor da medida de distância do log da verossimilhança. Nas estatísticas dos pré-agrupamentos são aplicadas técnicas hierárquicas de agrupamentos, sem precisar carregar na memória todo arquivo de dados.

Outra grande vantagem deste algoritmo está em permitir que sejam utilizadas bases de dados com variáveis categóricas e quantitativas, além de fornecer uma informação sobre o número de agrupamentos, caso este não seja fornecido no início da execução do algoritmo.

Como o ADE consegue utilizar as variáveis categóricas, os valores contidos nessas variáveis categóricas não precisam sofrer as transformações sugeridas no capitulo 3. Apenas quando houver variáveis com valores quantitativos, pode haver a transformação pela media e o desvio padrão, mostrada em (3.12).

Para identificar os agrupamentos, O ADE utiliza uma estrutura de dados em árvore. Nessa estrutura, os dados encontram-se dispostos de forma hierárquica (Wikipedia, 2007; Tenembaum et al, 2004). A árvore é composta de um elemento ou nó principal chamado raiz, que possui ligações para outros nós chamados de filhos. Esses filhos, por sua vez, possuem outros filhos. Os nós que possuem filhos são chamados de não-terminais e os que não possuem filhos são chamados de folhas.

O número máximo de filhos que um nó pode ter é chamado de ordem da árvore. Uma árvore balanceada é aquela em que todas as suas folhas estão no mesmo nível, ou mesma hierarquia na árvore ou, no máximo, um nó um nível abaixo dos outros. As operações mais comuns na estrutura de dados em árvore são a busca, inserção e remoção de um dado qualquer..

A visualização de uma árvore pode ser vista na Figura 4.1, onde é possível identificar a raiz, os nós não-terminais e as folhas.

O ADE é uma variação do algoritmo BIRCH (*Balanced Iterative Reducing and Clustering using Hierarchies*), proposto por Zhang et al. (1996). Nas seções a seguir os dois algoritmos são detalhados.

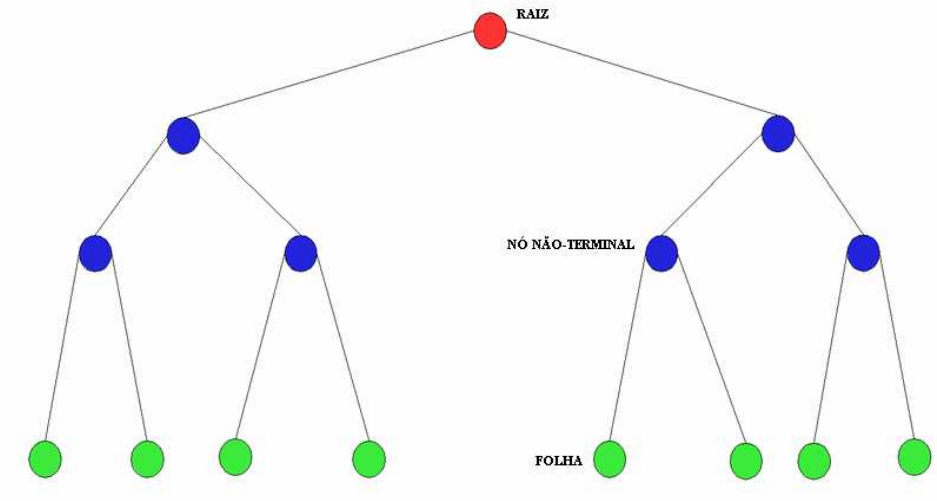


**Figura 4.1.** Estrutura de dados em árvore.

Fonte: Wikipedia, 2007.

Algumas medidas são propostas em Zhang et al. (1996) e Zhang (1997) para a formação dos agrupamentos no algoritmo BIRCH. Dados n registros J-dimensional $\overrightarrow{x_i} = (x_{11}, x_{12}, x_{nJ})$ na forma apresentada na Tabela 3.1, com i = 1, 2, .., n, o centróide $\overrightarrow{X_o}$ de um agrupamento pode ser definido como:

$$\overrightarrow{X_o} = \frac{\sum_{i=1}^{n} \vec{x}_i}{n} = \frac{\overrightarrow{SL}}{n} = \frac{\sum_{j=1}^{J} SL_j}{n} \tag{4.1}$$

onde $\overrightarrow{SL}$ é um vetor J-dimensional, cujo j-ésimo elemento é a soma linear dos termos de uma variável j, ou seja, $SL_j = \sum_{i=1}^{n} x_{ij}$ e o raio pode ser definido como:

$$r = \left( \frac{\sum_{i=1}^{n} d^2 (\vec{x}_i - \overrightarrow{X}_o)^2}{n} \right)^{\frac{1}{2}} = \left( \frac{SQ}{n} - \frac{\sum_{j=1}^{J} SL_j^2}{n^2} \right)^{\frac{1}{2}} \tag{4.2}$$

onde $SQ$ é a soma dos termos quadráticos de todas as variáveis, ou seja, $SQ = \sum_{j=1}^{J}\sum_{i=1}^{n} x_{ij}{}^2$ e $d(\vec{x}_i - \vec{x}_o)$ é a distância euclidiana do i-ésimo registro em relação ao centróide do mesmo agrupamento.

A medida de distância euclidiana, DE, mostrada em (3.2), sob os termos do vetor $\overrightarrow{SL}$:

$$DE = \left( \sum_{j=1}^{J} \left( \frac{SL1_j}{n_1} - \frac{SL2_j}{n_2} \right)^2 \right)^{\frac{1}{2}} \tag{4.3}$$

onde $SL1_j$ e $SL2_j$ são os valores das somas dos termos da j-ésima variável de dois agrupamentos distintos.

### 4.1. *Clustering Feature* e CF-*tree*

Esses dois conceitos são as bases de funcionamento do algoritmo BIRCH. Esta seção baseia-se nos conceitos apresentados em Zhang et al. (1996). *Clustering Feature* (característica do agrupamento) é uma tripla que armazena as estatísticas sobre um determinado agrupamento. Essa tripla pode ser representada por um vetor $\overrightarrow{CF} = (n, \overrightarrow{SL}, SQ)$, onde n é o número de registros do agrupamento, $\overrightarrow{SL}$ é a soma linear dos n registros e SQ é a soma dos quadrados destes n casos. O vetor $\overrightarrow{CF}$ possui informações suficientes para o cálculo das medidas necessárias para a formação dos agrupamentos na *CF-tree*.

A união de dois agrupamentos, digamos $K_1$ e $K_2$, cria um $\overrightarrow{CF}_{<1,2>}$, resultando na união de $\overrightarrow{CF}_1$ e $\overrightarrow{CF}_2$, que pode ser feita pela simples soma dos valores das duas triplas:

$$\overrightarrow{CF}_{<1,2>} = (n_1 + n_2, \vec{SL}_1 + \vec{SL}_2, SQ_1 + SQ_2) \tag{4.4}$$

A *CF-tree* é uma estrutura de dados em árvore de altura balanceada que possui dois parâmetros: um fator de ramificação B, ou seja, um limite para o número possível de filhos de um nó; e um limiar T, ou seja, um valor limite para o raio de cada sub-agrupamento.

Cada nó não-terminal possui B entradas na forma $[\overrightarrow{CF}_i, filho_i]$ para cada nó filho que possuir, sendo i = 1, 2, 3, ..., B. Um nó folha ou terminal contém pelo menos L entradas, cada uma da forma $[\overrightarrow{CF}_i]$, sendo i = 1, 2, 3, ..., L. Cada entrada corresponde a um conjunto de registros contidos em um vetor $\overrightarrow{CF}$. Cada nó folha possui dois ponteiros, indicando o nó folha anterior e o posterior para que todos os nós folhas possam interagir entre si, ou se comunicarem em futuras varreduras dos dados.

Tanto os nós terminais como os nós folhas representam conjuntos de sub-agrupamentos, porém os nós folhas devem possuir um número de entradas que não ultrapasse o limiar T. A CF-*tree* é construída dinamicamente, conforme novos registros são inseridos.

É possível concluir que quanto maior for T, menor a CF-*tree* será, ou seja, ela terá um menor número de nós e níveis. A Figura 4.2 mostra uma ilustração de uma CF-*tree* de altura 2, com B=L=3. Os raios das folhas $CF_1$ até $CF_5$, r1, r2, r3, r4 e r5, devem satisfazer o limiar T, ou seja, r1, r2, r3, r4, r5 devem ter raio menor ou igual a T. As setas representam as indicações de nó folha anterior e nó folha posterior.

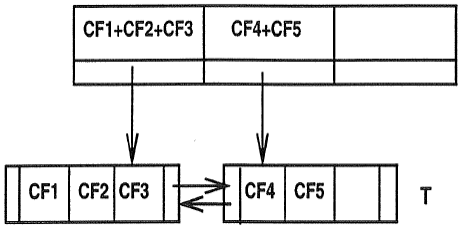


**Figura 4.2.** Exemplo de CF-*tree*

Fonte: Zhang, 1997

### 4.2. Algoritmo de inserção na CF-*tree*

Um simples registro em uma base de dados ou um sub-agrupamento de registros, dependendo se um novo registro é inserido ou se alguma folha existente é re-inserida, pode ser chamado “ent”. A seguir é apresentado o algoritmo para inserção de uma entrada “ent” em uma CF-*tree*, proposto em Zhang et al. (1996):

1. **Identificando o nó folha apropriado:** começando do nó raiz, ir descendo recursivamente através dos nós da CF-*tree* para escolher o filho que está mais "próximo", de acordo com alguma medida de distância.
2. **Modificando o nó folha:** quando chegar um nó folha, procurar pela entrada mais "próxima" e verificar se ela pode absorver "ent" sem violar o limiar T, ou seja a união dos dois não pode superar o limiar T. Se esse limiar for ultrapassado, é criada uma nova entrada para "ent" no nó folha. Se houver espaço no nó folha, ou seja, se o número de entradas não ultrapassar B, este passo está concluído. Senão o nó folha precisa sofrer uma divisão. A divisão é feita, primeiramente, através da escolha das duas sementes, que são as entradas que estão mais distantes, e redistribuir as entradas restantes de acordo com algum critério de distância ou proximidade.
3. **Modificando o caminho do nó folha para a raiz:** depois de inserir "ent" em um nó folha, deve-se atualizar cada entrada não terminal nos níveis superiores da árvore até chegar à raiz. Se não houver divisão do nó, basta atualizar os $\overrightarrow{CF}$s respectivos a cada entrada do caminho da raiz até o nó folha. Se houver divisão, é necessário criar uma nova entrada no nó pai, para representar a inserção de um novo nó folha. Se o nó pai possuir espaço para a inserção dessa nova entrada, assim como os níveis superiores, basta atualizar as entradas $\overrightarrow{CF}$ para refletir a adição de "ent". Senão, o nó pai também deverá sofrer uma divisão, sucessivamente, até chegar à raiz. Se a raiz sofrer uma divisão, a altura da árvore cresce em um nível e uma nova raiz será criada.

Já que um nó pode suportar um limitado número de entradas, de acordo com o seu tamanho, ele nem sempre corresponde ao agrupamento original presente nos dados. Podem ocorrer situações em que um agrupamento real, ou seja, o verdadeiro e não o estimado pela CF-*tree*, será dividido em dois nós. Dependendo da ordem em que os dados forem inseridos na árvore ou da assimetria dos dados, é possível que dois agrupamentos distintos acabem inseridos em um mesmo nó. Estes problemas são resolvidos, em sua maioria, com a adoção de um algoritmo hierárquico na 2ª fase do algoritmo, que será explicada mais adiante na seção 4.3.2.

Considerando-se os registros da Tabela 3.3, $P_1$=(2,10), $P_2$=(4,9), $P_3$=(5,11), $P_4$=(7,6), $P_5$=(10,5), $P_6$=(5,3) e mais um registro discrepante, $P_7$=(15,15). O gráfico de dispersão bidimensional com os 7 registros pode ser visto na Figura 4.3, em que o eixo X representa os valores da primeira variável e o eixo Y os valores da segunda.

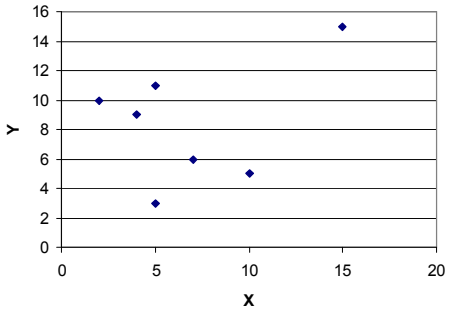


**Figura 4.3.** Gráfico de dispersão dos registros inseridos na CF-*tree*

Um exemplo simples de inserção dos registros da Figura 4.3, em uma CF-*tree*, é mostrado na Figura 4.4. Foram usados o limiar T = 2,0 e o limite de ramificações B = 2. Para efeito de simplificação, cada nó possui apenas uma posição para entrada $\overrightarrow{CF}$. A medida de distância utilizada foi a distância euclidiana.

No passo 1 o primeiro registro é inserido na árvore, e o $\overrightarrow{CF}$ (1,(2,10),104) passa a existir. O mesmo acontece no passo 2 e 3, onde é verificado também se o $\overrightarrow{CF}$ não ultrapassa o limiar T. Esse limiar, nesta ilustração com valor 2,0, é ultrapassado no passo 4, então tem que ser criado um novo nó para o registro 4. No passo 6 o limiar também é ultrapassado pelo raio, porém o limite de ramificações, igual a 2 é ultrapassado na criação do novo nó, então, na verdade, tem que ser feita a "divisão" desses 3 nós em apenas 2. Como explicado anteriormente, as duas folhas mais distantes servem como sementes, e a outra folha que sobra é unificada. A folha que está mais próxima neste nível gera duas folhas num nível inferior da árvore. Basicamente, o mesmo acontece no passo 7, inserindo o último registro na CF-*tree*, onde temos 4 folhas (7, 123, 45,6). Essas quatro folhas podem ser consideradas como quatro agrupamentos, construídos com a CF-*tree*. A Figura 4.5 mostra como ficou a formação final dos agrupamentos.

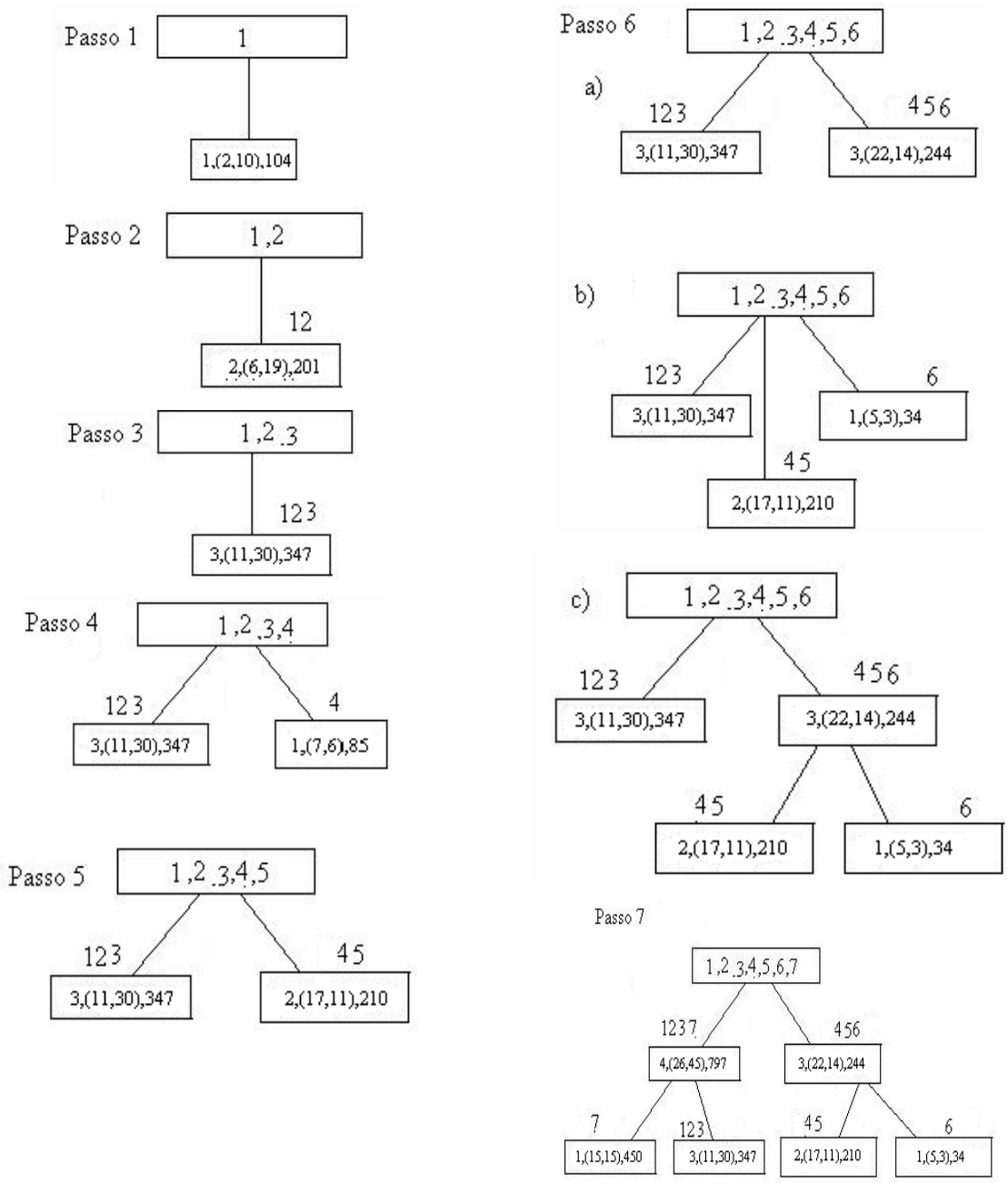


**Figura 4.4.** Exemplo de inserção na CF-*tree*

### 4.3. Algoritmo BIRCH

O algoritmo BIRCH consiste de 2 fases: 1) Armazenamento dos dados na CF-*tree*, 2) Formação global dos agrupamentos. Opcionalmente, pode haver mais duas fases, que neste trabalho não são descritas. Detalhes em Zhang et al. (1996) e Zhang (1997).

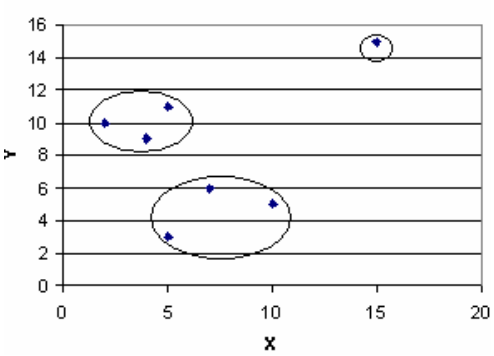


**Figura 4.5.** Agrupamentos construídos pela CF-*tree*.

### 4.3.1. Fase 1: Armazenamento dos dados na CF-*tree*

Começando com um valor de limiar inicial, varrem-se os dados inserindo-os na CF-*tree*. Se a memória se esgotar antes do término da varredura dos dados, o valor do limiar é aumentado e a CF-*tree* é reconstruída, só que agora com um menor número de nós, re-inserindo as folhas da CF-*tree* antiga. Após todos os registros das folhas serem re-inseridos, a varredura é reiniciada do ponto onde havia parado.

### 4.3.2. Fase 2: Formação global dos agrupamentos

Como foi mencionado na seção 4.2, dependendo da ordem que os registros são inseridos na CF-*tree* ou caso o conjunto de dados possua uma distribuição assimétrica, pode haver a formação de agrupamentos que não correspondem à realidade. Esta fase 2 surge para minimizar esse problema.

Com os vetores $\overrightarrow{CF}$ sendo conhecidos, cada sub-agrupamento pode ser tratado como um simples registro, através do cálculo do seu centróide, representando o sub-agrupamento. De outra maneira, os sub-agrupamentos podem ser utilizados sem nenhuma modificação, já que a informação contida nos vetores $\overrightarrow{CF}$ já é suficiente. O algoritmo BIRCH proposto em Zhang et al. (1996) utiliza um algoritmo hierárquico aglomerativo diretamente nos sub-agrupamentos encontrados na fase 1, representados pelos seus CFs. O algoritmo utiliza alguma medida de distância para aglomerar os sub-agrupamentos até que se chegue ao número final de agrupamentos.

Na Figura 4.6 há uma representação das etapas do algoritmo BIRCH, incluindo as duas etapas opcionais.

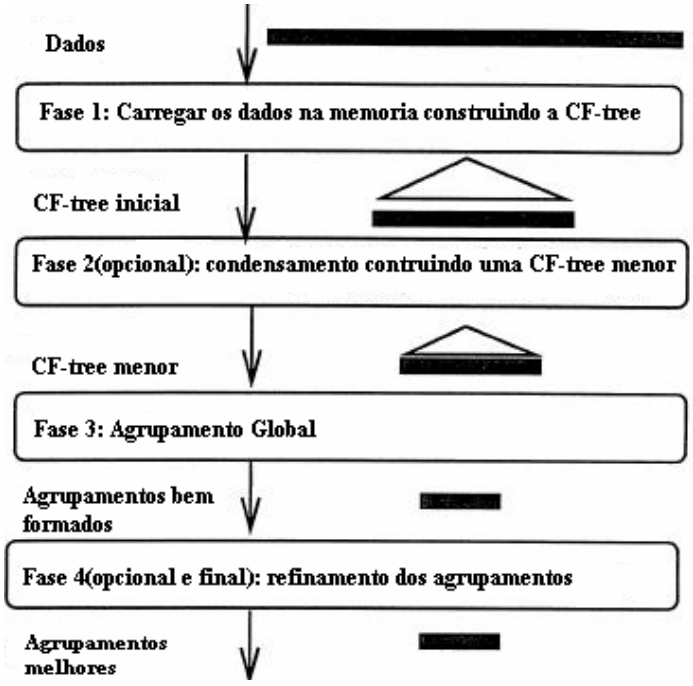


**Figura 4.6.** Etapas do algoritmo BIRCH

Fonte: Fonte: Zhang, 1997.

### 4.4. Reconstrução da CF-*tree*

Como foi explicado em 4.3.1, quando o espaço em memória se esgota, é necessário reconstruir a CF-*tree* com um tamanho menor, através do aumento do limiar T. Assume-se $t_i$ como uma CF-*tree* de limiar $T_i$, com uma altura h e tamanho ou número de nós igual a $S_i$. Dado $T_{i+1} \geq T_i$, todas as entradas das folhas de $t_i$ devem ser usadas para reconstruir $t_{i+1}$ com um limiar $T_{i+1}$, sendo que $t_{i+1}$ não pode ser maior que $S_i$.

Assume-se também como caminho da CF-*tree* aquele se inicia em uma entrada no nó raiz e vai descendendo até chegar a uma respectiva entrada no nó folha. Usar-se-á os conceitos de "NovoCaminhoAtual", "AntigoCaminhoAtual" e "NovoCaminhoMaisPróximo". Inicialmente, a nova CF-*tree* está vazia e "AntigoCaminhoAtual" recebe o caminho mais a esquerda da antiga CF-*tree*. O algoritmo segue adiante:

**1. Criar um correspondente "NovoCaminhoAtual" na nova CF-*tree*:** nós são adicionados na nova CF-*tree* exatamente como na antiga CF-*tree*, porém não há chance da nova CF-*tree* se tornar maior que a antiga;

**2. Inserir as entradas nas folhas de "AntigoCaminhoAtual" na nova CF-*tree*:** com um novo limiar, cada entrada em "AntigoCaminhoAtual" é testada na nova CF-*tree* para verificar se ela pode ser absorvida por alguma entrada existente em alguma folha ou criar uma nova entrada e, então, essa entrada é inserida em "NovoCaminhoAtual". Porém, se for necessário dividir o nó em algum "NovoCaminhoMaisPróximo", que é o caminho mais próximo e pode ser obtido através de alguma medida de distância, então a entrada é inserida em "NovoCaminhoMaisPróximo" e o espaço em "NovoCaminhoAtual" é deixado vazio para uso posterior;

**3. Liberar o espaço em "AntigoCaminhoAtual" e "NovoCaminhoAtual":** desde que todas as entradas das folhas de "AntigoCaminhoAtual" foram varridas, os nós desnecessários em "AntigoCaminhoAtual" podem ser liberados ou apagados, assim como os nós em "NovoCaminhoAtual" que estiverem vazios, já que as suas entradas foram "empurradas pra cima";

**4.** O próximo caminho na antiga CF-*tree* é nomeado como "AntigoCaminhoAtual" se existir algum caminho que ainda não foi varrido. Voltar ao passo 1.

### 4.5. Valores para o limiar T

Uma boa escolha do limiar pode reduzir sensivelmente o número de reconstruções da CF-*tree*. Se o limiar inicial tiver um valor muito alto, será obtida uma CF-*tree* menos detalhada do que é possível com a memória disponível. É aconselhável, então, escolher um valor conservador para o limiar inicial. O algoritmo BIRCH tem como padrão começar com um limiar $T_o$ igual a zero, porém, se houver um conhecimento anterior do usuário, isto pode ser mudado.

Dependendo do valor do limiar $T_o$ escolhido, uma hora a memória pode se esgotar. Se isso acontecer, o valor do limiar precisará ser aumentado. Em Zhang et al. (1996) são propostas algumas heurísticas para o incremento no valor de T.

**4.6. Algoritmo Agrupamento em Duas Etapas (ADE)**

O algoritmo ADE, proposto em Chiu et al. (2001), é baseado no algoritmo BIRCH, mas com alguns melhoramentos que o permitem trabalhar com dados que possuem variáveis categóricas e quantitativas, diferentemente do BIRCH que somente trabalha com variáveis quantitativas.

O algoritmo ADE possui duas etapas, similares as duas etapas principais do BIRCH, com algumas particularidades. Primeiramente, há uma modificação no vetor $\overrightarrow{CF}$ original, como apresentado a seguir (Chiu et al. , 2001):

$$\overrightarrow{CF} = (n, \overrightarrow{SL}, \overrightarrow{SQ}, \overrightarrow{PC}\ ) \tag{4.5}$$

onde $\overrightarrow{SL}$ representa o mesmo do $\overrightarrow{CF}$ utilizado no BIRCH, $\overrightarrow{PC} = (\overrightarrow{PC}_1, \overrightarrow{PC}_2, .., \overrightarrow{PC}_J)$ é um vetor, em que cada elemento $\overrightarrow{PC}_J$ é também um vetor $\overrightarrow{PC}_j = (PC_{j1}, .., PC_{jC})$ onde o número real $PC_{jc}$ representa a proporção do número de registros que fazem parte da c-ésima categoria da j-ésima variável categórica, representada por $\overrightarrow{PC}_j$. $\overrightarrow{SQ}$ é um vetor, contendo a soma quadrática dos valores de cada variável quantitativa, diferentemente do $\overrightarrow{CF}$ original usado no BIRCH, onde $\overrightarrow{SQ}$ é apenas um escalar com valor igual a soma quadrática de todos os elementos, sem distinguir por variável.

As estatísticas presentes no $\overrightarrow{CF}$ são suficientes para o cálculo da medida de log da verossimilhança utilizada no ADE.

A primeira etapa do ADE é similar à primeira etapa do BIRCH, porém como medida de distância é utilizada o log da verossimilhança, apresentada em (3.10), na seção 3.1.4, o que possibilita a utilização de variáveis tanto quantitativas como categóricas. A média dos n registros da j-ésima variável quantitativa é dada por

$$\bar{x}_j = \frac{\sum_{i=1}^{n} x_{ji}}{n} = \frac{SL_j}{n}, \tag{4.6}$$

com $SL_j$ sendo o j-ésimo elemento de $\overrightarrow{SL}$. A variância $\hat{\sigma}_j^2$ dos n registros da j-ésima variável é dada por:

$$\hat{\sigma}_j^2 = \frac{\sum x_i^2}{n} - \frac{(\sum x_i)^2}{n^2} = \frac{SQ_j}{n} - \frac{SL_j^2}{n^2} \tag{4.7}$$

onde $SQ_j$ é a soma dos quadrados dos valores da variável j e $SL_j^2$ é o quadrado da soma dos valores numéricos da variável j, informações estas contidas no vetor $\overrightarrow{CF}$. O valor do parâmetro $\hat{E}_j$ do log da verossimilhança para a j-ésima variável categórica pode ser obtido como:

$$\hat{E}_j = -\sum_{c=1}^{C} PC_{jc} \log PC_{jc}. \tag{4.8}$$

Outra diferença do ADE para o BIRCH está no fato de ao invés de ser utilizado o valor do raio para comparação com o limiar T, no ADE é utilizado o valor da distância de verossimilhança para comparar com o valor do limiar T.

A Figura 4.7 apresenta um exemplo de inserção de registros na CF-*tree* utilizada no ADE. Foram utilizados 7 registros, com duas variáveis quantitativas e duas variáveis categóricas. Estas variáveis categóricas possuem 3 possíveis categorias: A, B e C. Os registros são: $P_1$=(2,7,B,C), $P_2$=(5,7,A,B), $P_3$=(8,3,C,B), $P_4$=(7,6,A,A), $P_5$=(10,5,C,A), $P_6$=(5,3,C,B), $P_7$=(15,15,B,A). Assim como no exemplo do BIRCH, foi utilizado um limiar T igual a 2,0 e um fator de ramificação B igual a 2.

Do passo 1 ao passo 8 os registros são inseridos na CF-*tree* utilizando a distância de log da verossimilhança. Há de se destacar o passo 3, onde o valor de distância ultrapassou o limiar 2 e foi necessária a criação de um novo nó. No passo 7 também ocorreu uma distância maior que o limiar T = 2 e, novamente, foi necessária a criação de um novo nó, porém isto violou o limite de ramificações B, então foi necessária a divisão do nó (ver na Figura 4.7 passo 8).

### 4.6.1 Segunda etapa do ADE

Após a primeira etapa do algoritmo, uma coleção de sub-agrupamentos é armazenada nos nós da CF-*tree*, e as informações contidas em $\overrightarrow{CF}$ são suficientes para serem utilizadas pela medida de distância. É utilizado, então, um algoritmo hierárquico aglomerativo para encontrar o número final de agrupamentos.

O problema que surge nesta segunda etapa é como encontrar um número final de agrupamentos, caso este não seja informado.

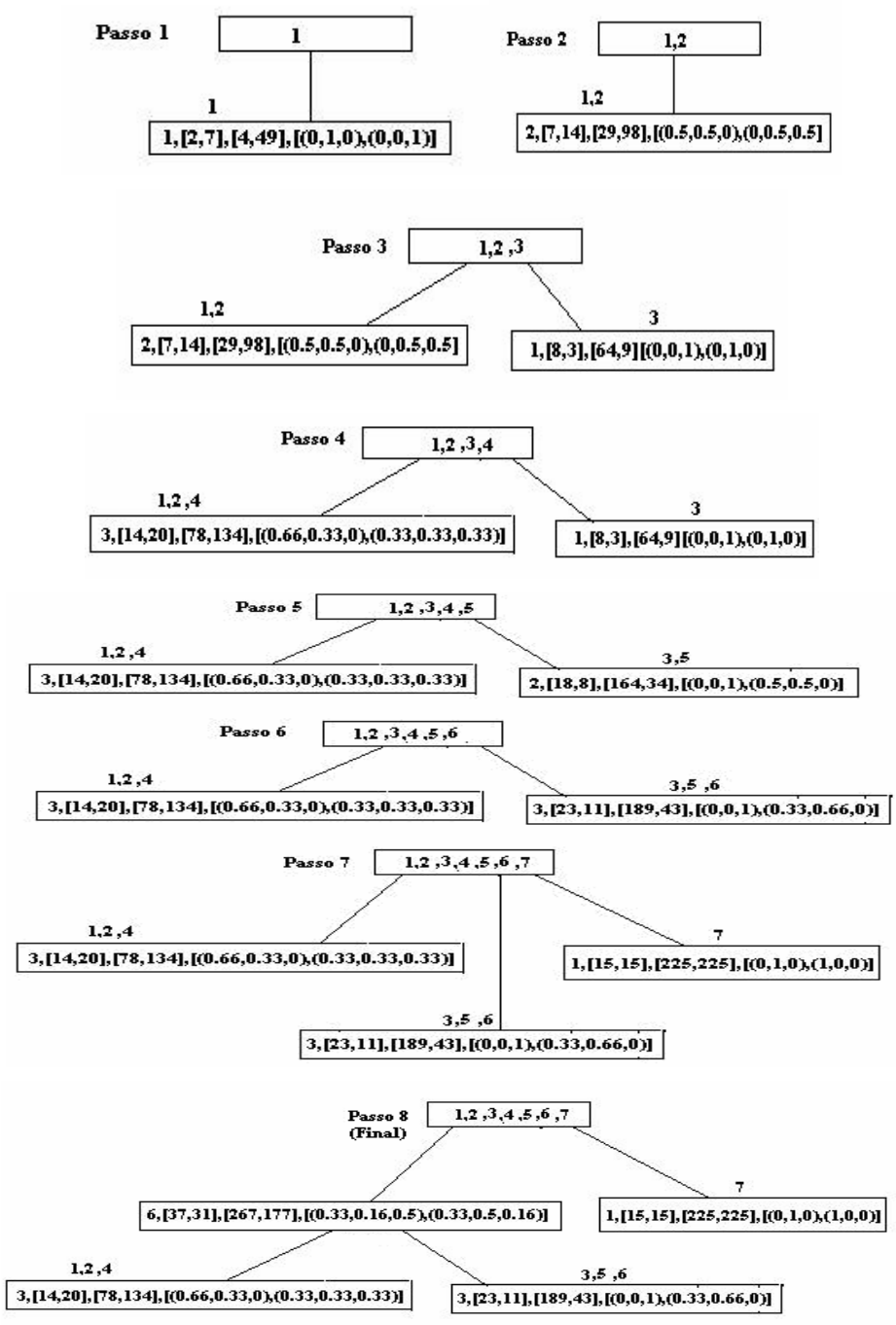


**Figura 4.7.** Exemplo de inserção na CF-*tree* do ADE

### 4.6.2. Determinação automática do número de agrupamentos

Chiu et al. (2001) propõe um procedimento em duas fases para encontrar o número final de agrupamentos.

Na primeira fase é encontrada uma estimativa do número de agrupamentos utilizando o BIC (*Bayesian Information Criterion*), que é um critério que penaliza a complexidade do modelo, medida pelo seu número de parâmetros. Aqui se entende por modelo todo o conjunto de parâmetros e estruturas de dados necessários para gerar um número K de agrupamentos. O BIC pode ser calculado com (SPSS, 2001):

$$BIC(K) = -2 \sum_{k=1}^{K} \left[ \xi_k + m_k \log(n) \right] \tag{4.9}$$

com

$$m_k = n_k \{ 2J_A + \sum_{j=1}^{J_B} (C_j - 1) \} \tag{4.10}$$

onde $n_k$ é igual ao número total de registros, $J_A$ é o número de variáveis quantitativas, $J_B$ é o número de variáveis categóricas, $C_j$ é o número de categorias da j-ésima variável categórica e $\xi_k$ foi definido em (3.9).

A cada união de agrupamentos, feita pelo algoritmo hierárquico, a redução no BIC é calculada. Uma estimação do número de agrupamentos é encontrada quando a taxa de redução do BIC começa a diminuir à medida que o número de agrupamentos aumenta.

Na segunda fase da determinação automática do número de agrupamentos, a taxa de mudança na distância dos agrupamentos é utilizada para encontrar o número final de agrupamentos. As uniões entre os agrupamentos começam a partir dos agrupamentos construídos na fase 1 e a estimativa do número de agrupamentos é obtida no momento em que há uma grande mudança na distância entre os dois agrupamentos que estão mais próximos em cada etapa de execução do algoritmo hierárquico. Essa mudança pode ser avaliada com a ajuda de um dendograma. O raciocínio por trás dessa técnica é que um grande salto na mudança da distância ocorre quando se tenta unir dois agrupamentos que não deveriam ser unidos.

## 4.7. Comparações dos algoritmos

Nesta seção serão mostradas algumas comparações já feitas com os algoritmos BIRCH e ADE.

### 4.7.1. Comparações envolvendo o algoritmo BIRCH

Em Zhang (1997), são feitos testes comparando a acurácia e a escalabilidade do algoritmo BIRCH. São feitas também comparações com o algoritmo de partição CLARANS.

Para fazer os testes foram criadas bases de dados sintéticas, obedecendo a distribuições normais, em que o número de agrupamentos e o rótulo do grupo a que cada registro pertence é conhecido. Todas as três bases possuem duas variáveis, 100 mil registros e número K de agrupamentos igual a 100. Posteriormente foram criadas outras 3 bases com os mesmos registros das bases originais, porém em ordem diferente. O algoritmo mostrou bons resultados na classificação dos registros em relação a acurácia e se mostrou insensível à mudança da ordem de entrada dos registros das bases de dados geradas.

Para avaliar a escalabilidade do algoritmo, foram criadas outras bases de dados para apurar três fatores: o aumento do número de registros, o aumento do número de agrupamentos e o aumento do número de variáveis. Para o primeiro fator, os testes mostraram que o tempo da execução do algoritmo cresce linearmente com o número de registros. Para o segundo fator, apesar de o tempo não ter sido linear, ficou bem próximo em relação às três bases de dados. Em relação ao terceiro fator, o tempo ficou levemente fora de uma escala linear, mas dentro de um padrão aceitável.

Em Zhang (1997) foram feitas comparações do algoritmo BIRCH com o algoritmo CLARANS. Para as três bases de dados simuladas, os autores concluíram que o CLARANS foi, pelo menos, 15 vezes mais lento que o BIRCH; e que o CLARANS é sensível à ordem de entrada dos registros. Em resumo, o algoritmo BIRCH usa menos memória, é mais rápido, tem mais acurácia, e é menos sensível a ordem dos registros do que o algoritmo CLARANS.

Em Zhang et al. (1996) é mostrada uma aplicação do algoritmo em uma base de dados reais. O BIRCH foi utilizado para agrupar elementos em uma imagem, com

paisagem e árvores. Como resultado, o algoritmo BIRCH conseguiu separar, satisfatoriamente, as árvores de sombras e outros elementos contidos na paisagem.

### 4.7.2. Comparações envolvendo o algoritmo ADE

Em Chiu et al. (2001) são criadas bases de dados sintéticas, sendo os registros gerados com uma distribuição normal multivariada. As variáveis com dados quantitativos são correlacionadas, e os valores das médias e das matrizes de covariância dos diferentes agrupamentos foram usados como medida para separar os valores das variáveis quantitativas dos agrupamentos. Segundo os autores, testes mostraram que apesar da medida de distância supor que os dados são independentes, a correlação entre as variáveis não teve influência significativa nos resultados do algoritmo.

Para gerar variáveis categóricas, primeiro foi localizado o centróide do agrupamento, e então criada uma amostra dos valores em que o centróide ocorria mais freqüentemente. A quantidade de categorias que não coincidia em uma mesma variável foi usado criar a quantidade de categorias de cada variável categórica.

Foram geradas centenas de bases de dados para fazer os testes. O número de variáveis chegou a 640, o número de registros a 5 milhões e o número de agrupamentos a 30. Em cerca de 98% das bases de dados, o algoritmo ADE construiu o número correto de agrupamentos existentes. Entre o rótulo dos registros dado pelo algoritmo, houve, no máximo, 5% de erro nas classificações.

Quanto à escalabilidade, foram feitos testes com bases de dados mantendo o número fixo de 5 agrupamentos de mesmo tamanho e 5 variáveis quantitativas. Para bases de dados com variáveis quantitativas e categóricas, foram criadas 5 variáveis quantitativas e 5 categóricas, com 2, 3, 5, 8 e 12 categorias, respectivamente. Os testes de aumento do número de variáveis, assim como os testes do aumento do número de registros, mostraram que o tempo de execução do algoritmo cresceu de forma linear, com uma demora um pouco maior para construir os agrupamentos com variáveis categóricas.

Chiu et al. (2001) compararam o ADE com o algoritmo de partição K-médias. Foram utilizadas as implementações do K-médias presentes no software Clementine 6.0, no SAS 8e e no SPSS 10.1, e o menor tempo encontrado entre os três foi o utilizado para

comparação com o ADE. Foram criadas nove bases de dados, com duas variáveis categóricas e cinco agrupamentos de mesmo tamanho. A menor base foi gerada com 50.000 casos, e a maior, com 5 milhões. Em todos os casos o algoritmo ADE foi capaz de construir o número correto de agrupamentos. Esse número foi tornado fixo, já que o algoritmo K-médias não possui a funcionalidade de descobrir o número ideal de agrupamentos. Foram analisados dois aspectos, então. O tempo e a acurácia. Na Tabela 4.1 são descritos os resultados.

Em relação ao tempo, assim como nos outros testes, o algoritmo ADE teve um tempo de processamento linear conforme se considerou bases de dados com número de maior de registros, enquanto que o algoritmo K-médias apresentou muita variação no tempo de processamento. Além disso, o algoritmo ADE foi bem mais rápido que o K-médias (Tabela 4.1).

**Tabela 4.1.** Resultado das comparações entre o ADE e o K-médias

| Quantidade de Registros | Tempo ADE(s) | Melhor tempo K-médias(s) | Taxa de Erro ADE(%) | Melhor Taxa de Erro K-médias(%) |
|---|---|---|---|---|
| 50k | 2 | 4 | 0,04 | 29,96 |
| 100k | 4 | 2 | 0,07 | 0,04 |
| 200k | 7 | 16 | 0,09 | 30,05 |
| 400k | 15 | 19 | 0,03 | 1,10 |
| 500k | 18 | 28 | 0,35 | 29,94 |
| 1000k | 35 | 26 | 1,28 | 1,24 |
| 2000k | 73 | 46 | 0,07 | 0,10 |
| 2500k | 87 | 39 | 0,11 | 0,11 |
| 5000k | 187 | 642 | 0,12 | 29,93 |

Fonte: Chiu et al., 2001.

Com relação à acurácia, na Tabela 4.1 é possível observar que o algoritmo ADE teve maior taxa de erro em relação ao K-médias (por volta de 1,3%), o que é bem aceitável; enquanto que o algoritmo K-médias teve taxa de erro de aproximadamente 30% para algumas bases de dados testadas.

Observa-se na Tabela 4.1 também que existe uma grande disparidade entre os valores da taxa de erro para o algoritmo K-médias entre as diferentes bases de dados. Chiu et al. (2001) atribui essas disparidades a forma como os dados estão organizados nas

diferentes bases de dados, porém não explica com mais detalhes o porquê destas disparidades.

# 5. PLANEJAMENTO DAS SIMULAÇÕES PARA COMPARAÇÃO DE ALGORÍTMOS

Neste capítulo são planejadas as comparações do ADE com outros dois algoritmos utilizados para a análise de agrupamentos: o K-médias e o CLARA. A escolha desses dois algoritmos baseia-se no trabalho de Prass (2004) que mostrou que os algoritmos de partição têm tido desempenho mais satisfatório para grandes bases de dados, que normalmente são utilizadas em mineração de dados.

O algoritmo K-médias é normalmente usado como base para comparações com outros algoritmos e o CLARA tem várias vantagens, conforme descrito em Kaufman e Rosseeuw (2005) e está disponível na linguagem R. A comparação do desempenho dos algoritmos foi feita seguindo a metodologia de planejamento de experimentos.

## 5.1 Conceitos preliminares

### 5.1.1 Planejamento de experimentos

Em vários ramos de atividade são feitos testes e experimentos para se descobrir algo sobre algum sistema. Segundo Montgomery (2005), um experimento pode ser definido como um conjunto de testes onde ocorrem mudanças propositais no conjunto de entrada, de maneira que seja possível observar e identificar os fatores que causam mudanças nos resultados do sistema. Deste modo, planejar experimentos auxilia no desenvolvimento de processos ou sistemas mais robustos e menos suscetíveis a fatores externos.

Experimentos podem envolver uma série de fatores. A pessoa que está conduzindo o experimento geralmente tem como objetivo determinar qual a influência desses fatores na resposta do sistema. Em Montgomery (2005) há algumas propostas de como planejar experimentos, com destaque para projetos fatoriais do tipo $2^M$, que possuem M fatores, cada um com 2 níveis. Geralmente, costuma-se chamar um nível de inferior e outro de superior, e representá-los por -1 e +1, respectivamente. Este tipo de projeto de experimento permite ao experimentador investigar os efeitos individuais de cada fator e averiguar como cada fator interage com os outros.

Um problema no projeto de experimento fatorial (*factorial design*) é que no caso de haverem muitos fatores, o número de ensaios (*runs*) cresce de forma exponencial. Um

projeto com 4 fatores, terá $2^4 = 16$ ensaios; e um com 10 fatores, $2^{10} = 1.024$ ensaios. Mesmo que o número de fatores em estudo não seja muito grande, às vezes o custo de um único ensaio é muito alto, então é desejável que se possa trabalhar sempre com um número reduzido de ensaios.

Em projetos com grande número de fatores, geralmente não é necessário que sejam executadas todas as possíveis combinações entre os níveis dos fatores, para que os resultados sejam confiáveis (Montgomery, 2005). Pode-se fazer uso do projeto fatorial fracionado, onde apenas um subconjunto das possíveis combinações é realizado.

Projetos de experimentos na forma fatorial fracionada são bastante usados na indústria e no melhoramento de processos de engenharia (Montgomery, 2005).

### 5.1.2. Simulação Monte Carlo

Na simulação Monte Carlo, várias amostras aleatórias são geradas, segundo alguma distribuição de probabilidade. Esses dados refletem uma aproximação dos valores que poderiam ocorrer, para determinadas variáveis, em uma situação específica (Vose, 2000). Essas amostras são as entradas do sistema em estudo. Para cada amostra, tem-se um resultado do sistema. Assim, características do sistema podem ser avaliadas pela distribuição de freqüência dos resultados das amostras geradas. Segundo Vose (2000), os resultados de simulações Monte Carlo são largamente aceitos como resultados razoavelmente válidos. Talvez, por isso, este tipo de simulação seja utilizado em larga escala, como pode ser visto nos trabalhos correlatos, mostrados nos capítulos 3 e 4.

## 5.2. Descrição do experimento

Para avaliar o tempo de processamento e, principalmente, a acurácia dos resultados gerados pelo ADE, foi realizado um experimento em que os registros foram gerados segundo distribuições de probabilidades (simulação tipo Monte Carlo). Em cada simulação, os subgrupos de casos gerados por uma dada distribuição de probabilidade, com parâmetros estabelecidos, foram assumidos como pertencentes a um agrupamento verdadeiro. A acurácia de um dado algoritmo de agrupamento foi avaliada pela porcentagem de registros provindos de uma mesma distribuição de probabilidades em que o algoritmo coloca num mesmo agrupamento.

Para gerar registros com distribuições multivariadas foi usada a metodologia de cópulas, que permite agregar distribuições univariadas em distribuições multivariadas, considerando determinada correlação entre as variáveis (Andrade et al, 2006; Anjos et al., 2004). Em especial, foi utilizada a cópula gaussiana que, no caso de J distribuições normais univariadas, gera a normal multivariada J-dimensional (Song, 2000). As cópulas também aceitam diferentes distribuições univariadas, formando uma nova distribuição multivariada com determinada dependência entre as variáveis. Para a geração das amostras foi utilizada a ferramenta de cópulas da linguagem livre R (R Development Core Team, 2006), disponível em http://www.r-project.org.

As bases de dados foram simuladas com variações planejadas do número de registros, quantidade de variáveis, quantidade de grupos, presença ou não de variáveis categóricas, correlação entre as variáveis nula ou não-nula, distribuição geradora dos dados (normal ou gama) e variâncias nos grupos iguais ou diferentes. Os estudos anteriores mostrados nos capítulos 3 e 4 focavam-se nas variações físicas das bases de dados (número de variáveis, registros e grupos), mas características dos dados e variáveis também são importantes na comparação dos algoritmos, porque algumas delas violam suposições intrínsecas de uma ou outra técnica. Por exemplo, dados com variâncias diferentes nos grupos violam suposições do K-médias, mas são pertinentes de serem analisados (minerados) pelo ADE. Já correlações não-nulas entre as variáveis, ou variáveis com distribuições assimétricas (no caso, a gama), são características indesejáveis para os três algoritmos em estudo. A Tabela 5.1 mostra os níveis (ou categorias) estudados (as) dos sete fatores de interesse.

Com o objetivo de comparar os algoritmos de agrupamento e verificar o efeito de cada fator nesses algoritmos, as simulações foram feitas segundo um projeto fatorial fracionário do tipo $2^{7-3}$ (Montgomery, 2005), que permite analisar 7 fatores com apenas $2^{7-3}$ = 16 ensaios. Este projeto é composto de dezesseis condições experimentais (combinações de níveis dos fatores da Tabela 5.1), balanceadas de tal forma que seja possível realizar a avaliação do efeito de cada fator isoladamente. Além da avaliação de cada fator de maneira isolada, também seria possível avaliar o efeito das combinações entre esses fatores. As combinações de níveis dos fatores (*design points*) são apresentadas na Tabela 5.2.

**Tabela 5.1.** Níveis dos fatores do experimento

| Fatores | Níveis | |
|---|---|---|
| | -1 | +1 |
| 1 - número de registros | dez mil | um milhão |
| 2 - quantidade de variáveis | 2 | 10 |
| 3 - número de agrupamentos | 2 | 10 |
| 4 - variáveis categóricas | ausência | 50% do tipo 0-1 |
| 5 - distribuição das variáveis quantitativas | normal | gama |
| 6 - correlação entre as variáveis | 0 | 0,7 |
| 7 - variâncias nos agrupamentos | iguais | diferentes |

**Tabela 5.2.** Condições experimentais efetivamente realizadas, baseadas no projeto $2^{7-3}$

| Ensaio | Número de registros | Número de variáveis | Número de grupos | Variáveis categóricas | Correlação | Distribuição | Variâncias |
|---|---|---|---|---|---|---|---|
| 1 | dez mil | 10 | 2 | 50% | 0,7 | normal | diferentes |
| 2 | dez mil | 10 | 10 | 0 | 0 | normal | diferentes |
| 3 | um milhão | 2 | 10 | 0 | 0 | gama | iguais |
| 4 | dez mil | 10 | 10 | 50% | 0 | gama | iguais |
| 5 | um milhão | 2 | 2 | 50% | 0,7 | gama | iguais |
| 6 | um milhão | 10 | 2 | 50% | 0 | normal | iguais |
| 7 | dez mil | 2 | 10 | 0 | 0,7 | gama | diferentes |
| 8 | dez mil | 10 | 2 | 0 | 0,7 | gama | iguais |
| 9 | um milhão | 10 | 10 | 50% | 0,7 | gama | diferentes |
| 10 | um milhão | 10 | 10 | 0 | 0,7 | normal | iguais |
| 11 | dez mil | 2 | 2 | 0 | 0 | normal | iguais |
| 12 | dez mil | 2 | 2 | 50% | 0 | gama | diferentes |
| 13 | um milhão | 2 | 2 | 0 | 0,7 | normal | diferentes |
| 14 | um milhão | 10 | 2 | 0 | 0 | gama | diferentes |
| 15 | um milhão | 2 | 10 | 50% | 0 | normal | diferentes |
| 16 | dez mil | 2 | 10 | 50% | 0,7 | normal | iguais |

Decidiu-se gerar os dados com duas distribuições de probabilidades: a distribuição normal e a distribuição gama. A distribuição normal é a mais comumente usada para gerar os dados, pois geralmente os dados nesta distribuição ficam distribuídos de forma simétrica em um eixo de coordenadas. Para gerar os dados na distribuição normal, são necessários dois parâmetros: o $\mu$ (média) e o $\sigma$ (desvio padrão). Já a distribuição gama foi escolhida por ter forma assimétrica. Os parâmetros fornecidos para gerar os dados na distribuição gama são o $\alpha$ (parâmetro de forma) e o $\beta$ (parâmetro de escala).

Uma dificuldade inicial foi gerar dados que, independentemente da distribuição, pudessem ter a mesma média e variância. Na distribuição gama, a média é dada por: $\mu = \alpha.\beta$ ; e a variância por: $\sigma^2 = \alpha.\beta^2$ .

Para gerar diferentes grupos de forma controlada, os dados da distribuição gama foram gerados com os valores $\alpha = 2$ e $\beta = \frac{1}{\sqrt{2}}$ . Ao valor gerado foi somada a constante 2. Deste modo os dados na distribuição gama seriam gerados de forma que tivessem a média igual a 3,4142 e variância igual a 1. Esses valores de media e variância foram escolhidos como parâmetros da distribuição normal.

Para que as bases de dados tivessem diferentes grupos, com variâncias iguais, foram adicionadas constantes aos valores gerados das variáveis aleatórias - o que só altera a média. Para se ter grupos diferindo também pelas variâncias, além de adicionadas as constantes, foram multiplicados valores proporcionais às constantes somadas - o que altera a média e a variância. Assim foi possível controlar grupos com médias diferentes e com variâncias iguais ou diferentes. Para fazer as somas e as multiplicações foi criada a expressão:

$$Y = a + b(X - \mu) \qquad (5.1)$$

onde *Y* representa o valor final gerado na base de dados; *a* é o valor de uma constante, que controla as diferenças das médias dos grupos; X é o valor gerado de uma variável aleatória (no caso da distribuição gama, foi somado o valor 2); $\mu$ é o valor da média com que a variável aleatória foi gerada (neste trabalho foi usado $\mu$=3,4142); *b* é um valor que controla as diferentes variâncias dos grupos. Se *b* for igual a 1, todas as variâncias são iguais (neste trabalho quando os grupos têm variâncias iguais elas foram fixadas em 1).

$b(X - \mu)$ tem valor esperado igual à zero, logo o valor esperado (média) de Y será aproximadamente igual à constante *a* adicionada. Para as bases de dados em que se deseja ter variâncias diferentes, foi usado o valor *b = a/2* para duas variáveis e dois grupos; *b = a/10* para duas variáveis e dez grupos; *b = a/2,5* para dez variáveis e dois grupos e, também, para dez variáveis e dez grupos. Isto garante que Y tenha, além da influência da

constante $a$, uma influência de $b(X-\mu)$, de modo que o desvio padrão fique sempre proporcional à média.

Os valores da constante $a$ da expressão (5.1), para as bases de dados com duas variáveis e dois grupos, são descritas na Tabela 5.3; com duas variáveis e dez grupos na Tabela 5.4; com dez variáveis e dois grupos na Tabela 5.5; e por fim, dez grupos e dez variáveis na Tabela 5.6.

Esses valores foram considerados suficientes para uma boa separação dos grupos, porém deixando certa confusão entre eles. Para escolher o nível de confusão entre os diferentes grupos, foi gerada uma amostra preliminar para cada um dos tipos de bases de dados possíveis (dez variáveis e dez grupos, dez variáveis e dois grupos, duas variáveis e dez grupos e duas variáveis e dois grupos). Os parâmetros foram escolhidos de forma que as acurácias dos algoritmos estivessem entre 70% e 90%.

**Tabela 5.3.** Constantes (valores de *a*) utilizadas para gerar os dados com duas variáveis e dois grupos.

| | Variáveis | |
|---|---|---|
| Grupo | X1 | X2 |
| 1 | 1 | 1 |
| 2 | 2 | 4 |

**Tabela 5.4.** Constantes (valores de *a*) utilizadas para gerar os dados com duas variáveis e dez grupos.

| | Variáveis | |
|---|---|---|
| Grupo | X1 | X2 |
| 1 | 1,85 | 12,14 |
| 2 | 13,98 | 6,83 |
| 3 | 16,78 | 12,08 |
| 4 | 2,04 | 19,96 |
| 5 | 10,22 | 10,19 |
| 6 | 3,96 | 4,55 |
| 7 | 9,73 | 4,27 |
| 8 | 4,82 | 1,62 |
| 9 | 18,45 | 4,30 |
| 10 | 16,67 | 8,83 |

**Tabela 5.5.** Constantes (valores de *a*) utilizadas para gerar os dados com dez variáveis e dois grupos.

| | Variáveis | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Grupo | X1 | X2 | X3 | X4 | X5 | X6 | X7 | X8 | X9 | X10 |
| 1 | 4,68 | 1,83 | 4,76 | 1,79 | 4,02 | 2,26 | 2,04 | 1,04 | 1,41 | 3,86 |
| 2 | 3,69 | 4,62 | 2,49 | 3,57 | 4,80 | 2,95 | 2,53 | 3,37 | 4,07 | 3,20 |

**Tabela 5.6.** Constantes (valores de *a*) utilizadas para gerar os dados com dez variáveis e dez grupos.

| | Variáveis | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Grupo | X1 | X2 | X3 | X4 | X5 | X6 | X7 | X8 | X9 | X10 |
| 1 | 2,41 | 4,17 | 1,87 | 1,65 | 1,19 | 2,47 | 3,04 | 3,61 | 3,56 | 3,10 |
| 2 | 4,97 | 3,31 | 4,28 | 1,05 | 1,87 | 1,71 | 1,53 | 3,50 | 4,52 | 2,35 |
| 3 | 4,28 | 4,71 | 2,15 | 1,66 | 2,86 | 3,32 | 1,08 | 4,03 | 4,93 | 2,76 |
| 4 | 2,05 | 4,97 | 2,28 | 1,67 | 3,29 | 2,96 | 3,76 | 2,77 | 3,82 | 2,23 |
| 5 | 3,13 | 2,10 | 2,17 | 3,78 | 2,57 | 4,28 | 1,39 | 1,51 | 4,87 | 1,94 |
| 6 | 2,50 | 2,07 | 4,03 | 2,04 | 4,69 | 3,33 | 4,98 | 4,53 | 4,90 | 4,61 |
| 7 | 4,96 | 3,60 | 1,99 | 1,68 | 4,58 | 2,86 | 3,80 | 1,43 | 1,78 | 2,56 |
| 8 | 4,95 | 3,15 | 3,31 | 1,03 | 1,79 | 4,34 | 2,29 | 3,37 | 4,22 | 3,89 |
| 9 | 2,20 | 2,79 | 4,94 | 4,80 | 4,12 | 1,87 | 3,64 | 3,88 | 2,00 | 3,67 |
| 10 | 2,15 | 4,30 | 3,59 | 2,61 | 1,68 | 3,30 | 1,50 | 4,84 | 2,11 | 3,45 |

A inserção de correlação foi feita por meio de cópulas, como já discutido; e variáveis do tipo 0-1 foram construídas recodificando os valores gerados por distribuições contínuas.

Para a geração dos dados na linguagem R foi usado um script para cada um dos 16 ensaios deste experimentos. O exemplo de um dos scripts utilizados pode ser visto no Apêndice A.

### 5.3. Simulação baseada em dados reais

Uma simulação complementar foi feita com base em estatísticas obtidas da tradicional base de dados de Íris (ver Johnson e Wichern, 2002, p.657), onde o pesquisador analisou o comprimento e a largura da pétala e da sépala de 150 plantas, sendo 50 da espécie setosa, 50 virginica e 50 versicolor. Portanto, é uma amostra de 150 casos de dimensão quatro, divididos igualmente em três grupos. A Figura 5.1 ilustra esses dados, visto sobre os dois primeiros componentes principais.

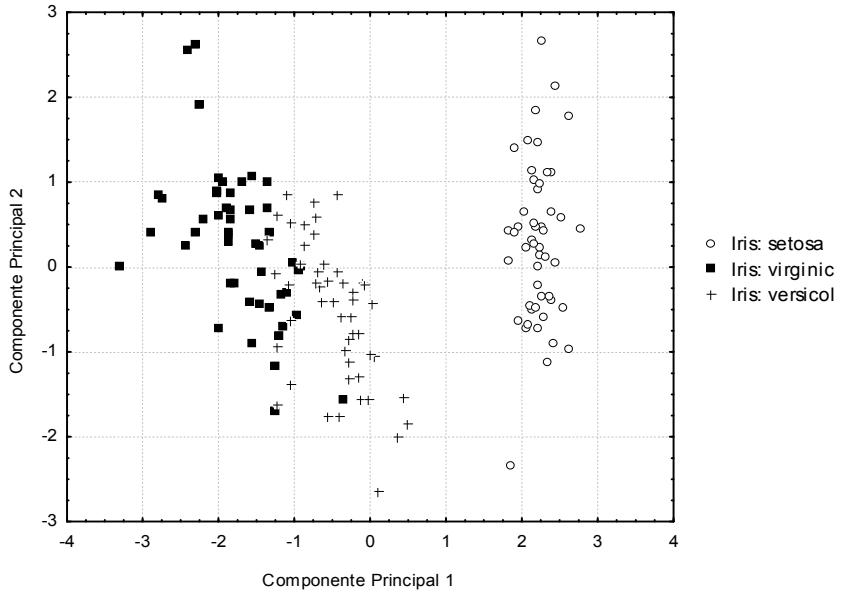


**Figura 5.1.** A base da Íris vista sob os dois primeiros componentes principais

Esses dados têm os seguintes vetores de médias, vetores de desvios padrão e matrizes de correlações:

$\bar{X}_{SETOSA} = (5{,}01 \;\; 3{,}43 \;\; 1{,}46 \;\; 0{,}25)^{t}$, $\bar{X}_{VIRGINIC} = (6{,}59 \;\; 2{,}97 \;\; 5{,}55 \;\; 2{,}03)^{t}$,

$\bar{X}_{VERSICOL} = (5{,}94 \;\; 2{,}77 \;\; 4{,}26 \;\; 1{,}33)^{t}$, $S_{SETOSA} = (0{,}35 \;\; 0{,}38 \;\; 0{,}17 \;\; 0{,}11)^{t}$,

$S_{VIRGINIC} = (0{,}64 \;\; 0{,}32 \;\; 0{,}55 \;\; 0{,}27)^{t}$, $S_{VERSICOL} = (0{,}52 \;\; 0{,}31 \;\; 0{,}47 \;\; 0{,}20)^{t}$.

$$R_{SETOSA} = \begin{bmatrix} 1{,}00 & 0{,}74 & 0{,}27 & 0{,}28 \\ 0{,}74 & 1{,}00 & 0{,}18 & 0{,}23 \\ 0{,}27 & 0{,}18 & 1{,}00 & 0{,}33 \\ 0{,}28 & 0{,}23 & 0{,}33 & 1{,}00 \end{bmatrix}, \quad R_{VIRGINIC} = \begin{bmatrix} 1{,}00 & 0{,}46 & 0{,}86 & 0{,}28 \\ 0{,}46 & 1{,}00 & 0{,}40 & 0{,}54 \\ 0{,}86 & 0{,}40 & 1{,}00 & 0{,}32 \\ 0{,}28 & 0{,}54 & 0{,}32 & 1{,}00 \end{bmatrix},$$

$$R_{VERSICOL} = \begin{bmatrix} 1{,}00 & 0{,}53 & 0{,}75 & 0{,}55 \\ 0{,}53 & 1{,}00 & 0{,}56 & 0{,}66 \\ 0{,}75 & 0{,}56 & 1{,}00 & 0{,}79 \\ 0{,}55 & 0{,}66 & 0{,}79 & 1{,}00 \end{bmatrix}$$

Foram simuladas cinco bases de dados com 150 mil registros cada, com distribuição normal multivariada e parâmetros iguais aos valores das estatísticas da base da Íris, cada agrupamento possuindo a mesma quantidade de registros.

# 6. RESULTADOS

Neste capitulo são mostrados os resultados do experimento descrito no Capitulo 5, além dos resultados das simulações realizadas em cinco bases de dados com parâmetros iguais aos valores das estatísticas da base da Íris.

## 6.1. Resultados do experimento do projeto $2^{7-3}$

Foi usada a implementação do ADE do SPSS®, versão 14, e a implementação do CLARA da linguagem R, versão 2.4.1. Chiu et al. (2001) executaram o K-médias em diversas ferramentas estatísticas e disponibilizaram, para comparação, a maior acurácia e o menor tempo. Neste trabalho utilizou-se para comparação os resultados do K-médias do R e do SPSS.

Utilizar o mesmo algoritmo em diferentes ferramentas é válido, pois, às vezes, estas ferramentas fazem mudanças internas nos algoritmos originais e os resultados costumam ser diferentes de uma ferramenta para outra, como é possível observar mais a frente nos resultados. A escolha de mais de uma ferramenta estatística diminui o impacto dessas mudanças específicas em cada ferramenta.

Para todos os algoritmos foi fornecido o número correto de grupos, ou seja, não foi avaliada a capacidade dos algoritmos em detectar adequadamente o número de grupos. A Tabela 6.1 apresenta a acurácia (porcentagem de casos agrupados corretamente) em cada uma das dezesseis bases de dados geradas, segundo o projeto de experimento mostrado no Capitulo 5. No caso das bases de dez mil registros, o experimento foi feito em triplicata; e os valores apresentados na Tabela 6.1, referentes aos ensaios 1, 2, 4, 7, 8, 11, 12 e 16, correspondem às médias dos resultados dessas triplicatas. Para as bases de dados com um milhão de registros foi realizado apenas um ensaio.

Observa-se na Tabela 6.1 que, em média, o ADE obteve a melhor acurácia, seguido do CLARA e, por último, do K-médias. Mas isto não aconteceu em todas as bases de dados (amostras) simuladas. A Figura 6.1 mostra um detalhamento da análise dos resultados, onde se visualiza o efeito de cada fator em estudo.

Quanto ao fator número de registros (tamanho da base de dados), quando o número de registros aumenta, a acurácia reduz proporcionalmente, tanto no ADE quanto nos outros algoritmos, o que não resulta em fato relevante na comparação entre os algoritmos.

**Tabela 6.1**. Porcentagem de registros agrupados corretamente (acurácia)

| | Algoritmo | | | |
|---|---|---|---|---|
| Ensaio | ADE | K-médiasSPSS | K-médiasR | CLARA |
| 1 | 83,2 | 62,6 | 61,0 | 78,8 |
| 2 | 79,7 | 57,0 | 70,9 | 48,8 |
| 3 | 96,1 | 83,7 | 85,0 | 94,2 |
| 4 | 42,6 | 69,6 | 73,7 | 61,6 |
| 5 | 93,3 | 69,8 | 70,5 | 83,6 |
| 6 | 91,0 | 98,8 | 98,8 | 97,6 |
| 7 | 96,2 | 78,8 | 76,6 | 95,0 |
| 8 | 91,4 | 96,9 | 97,0 | 97,6 |
| 9 | 40,6 | 37,0 | 40,6 | 39,9 |
| 10 | 53,4 | 52,2 | 60,8 | 65,4 |
| 11 | 93,1 | 94,3 | 94,3 | 93,7 |
| 12 | 88,6 | 71,2 | 71,5 | 89,6 |
| 13 | 77,9 | 83,0 | 83,0 | 83,2 |
| 14 | 99,0 | 99,0 | 99,0 | 94,7 |
| 15 | 58,0 | 54,4 | 51,7 | 54,1 |
| 16 | 70,8 | 57,8 | 55,7 | 58,3 |
| Média: | 78,4 | 72,9 | 74,4 | 77,2 |
| Desv.Pad. | 19,6 | 18,8 | 17,7 | 19,5 |

O ADE tem uma queda significativa na acurácia e até mesmo leva desvantagem para os outros algoritmos quando ocorre o aumento no número de variáveis, resultados não esperados, considerando os resultados mostrados em Chiu et al. (2001), que apresenta vantagem em todos os casos para o ADE.

Quando a quantidade de grupos aumenta, apesar de todos os algoritmos terem uma queda sensível na acurácia, esta queda foi aproximadamente proporcional, similar ao resultado do fator do aumento do número de registros.

Na presença de correlação entre as variáveis e presença de variáveis categóricas, o ADE obteve uma acurácia bem superior à acurácia do algoritmo K-médias, porém foi acompanhado de perto pelo algoritmo CLARA. O algoritmo ADE teve uma acurácia superior aos outros dois algoritmos quando a distribuição dos dados era a normal, porém foi ultrapassado pelo CLARA quando foram usados dados com distribuição gama.

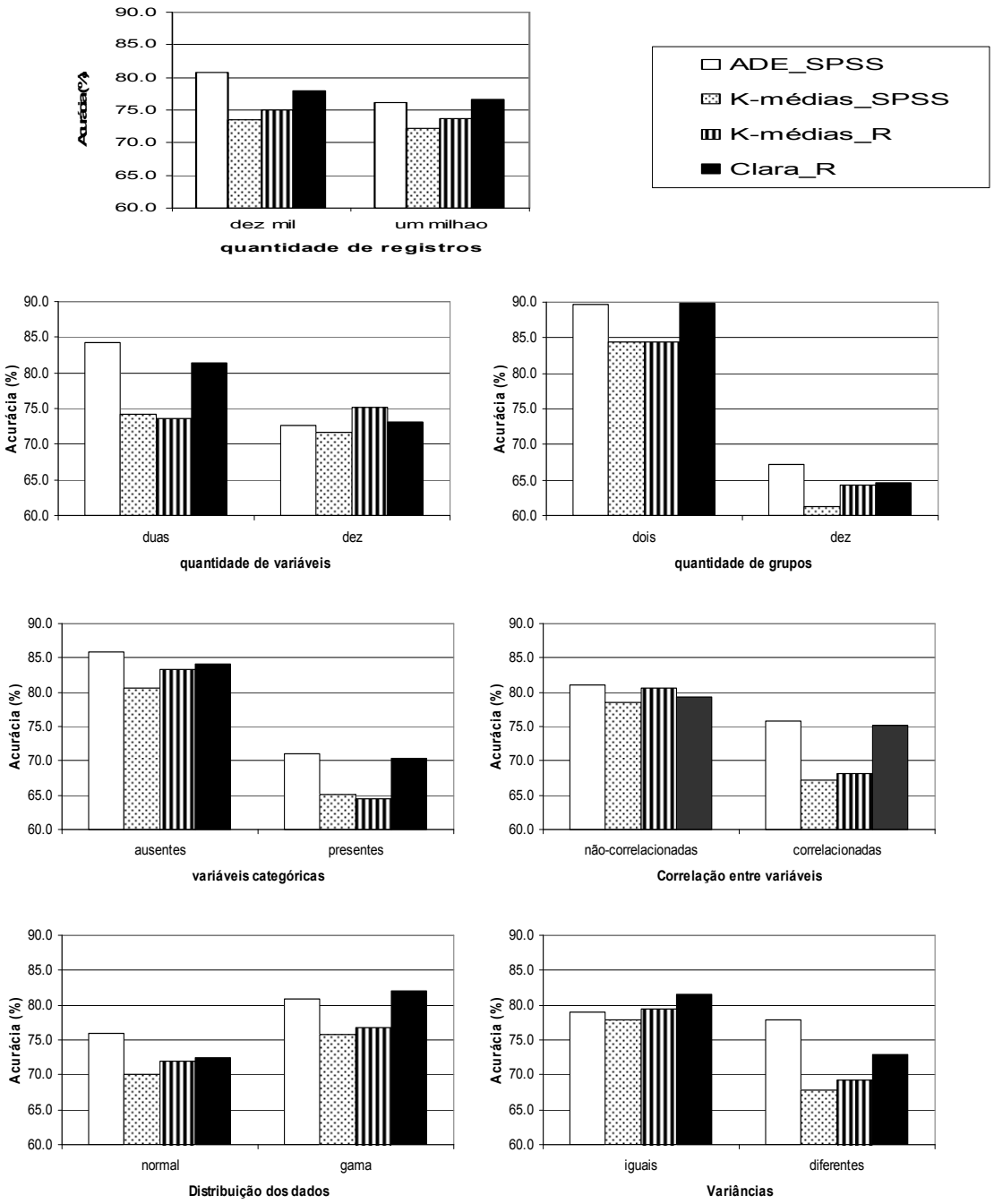


**Figura 6.1.** Porcentagem média de casos agrupados corretamente, por algoritmo e por característica da base de dados.

O maior destaque, porém, deve-se ao fator com variâncias das variáveis iguais ou diferentes entre os grupos. Quando as variâncias são iguais, o algoritmo ADE teve um desempenho inferior aos outros dois algoritmos, porém esta situação inverte-se drasticamente quando as variâncias são diferentes. Isto pode ter ocorrido devido ao fato do ADE se basear na função de verossimilhança da distribuição normal, com variâncias diferentes entre grupos.

A Tabela 6.2 apresenta o tempo médio de processamento nas dezesseis bases de dados. Na linguagem R o tempo foi medido com a ajuda da função Sys.time() e no SPSS, no modo de execução macro (modo texto) é possível gerar um arquivo que dá algumas informações sobre o processamento do algoritmo, entre elas o tempo.

Observa-se que os algoritmos executados na linguagem R foram bem mais rápidos do que os algoritmos executados no SPSS®. Isso deve ter ocorrido por mecanismos internos de cada um dos programas e foge ao escopo deste trabalho fazer comparação entre os pacotes computacionais.

**Tabela 6.2.** Tempo de processamento (s).

| | Algoritmo | | | |
|---|---|---|---|---|
| Ensaio | ADE | K-médiasSPSS | K-médiasR | CLARA |
| 1 | 3,0 | 0,4 | 0,2 | 0,2 |
| 2 | 3,2 | 0,7 | 0,3 | 0,2 |
| 3 | 58,7 | 35,1 | 15,2 | 5,5 |
| 4 | 3,4 | 0,7 | 0,2 | 0,2 |
| 5 | 50,6 | 19,2 | 4,0 | 3,8 |
| 6 | 186,4 | 26,8 | 5,8 | 7,2 |
| 7 | 0,9 | 0,4 | 0,1 | 0,1 |
| 8 | 3,1 | 0,4 | 0,1 | 0,1 |
| 9 | 244,7 | 63,0 | 23,3 | 11,7 |
| 10 | 214,2 | 53,0 | 43,6 | 11,7 |
| 11 | 0,8 | 0,3 | 0,1 | 0,1 |
| 12 | 0,8 | 0,3 | 0,1 | 0,1 |
| 13 | 47,7 | 19,2 | 3,8 | 3,6 |
| 14 | 162,2 | 27,7 | 5,5 | 7,3 |
| 15 | 63,8 | 35,1 | 13,1 | 5,1 |
| 16 | 1,0 | 0,4 | 0,1 | 0,1 |
| Média: | 65,3 | 17,7 | 7,2 | 3,6 |
| Desv.Pad | 86,0 | 20,8 | 11,9 | 4,2 |

Na Figura 6.2 é possível ver os gráficos individuais para cada fator em estudo em relação ao tempo de processamento de cada algoritmo. Em todos os gráficos é possível ver que o ADE foi mais lento que os outros dois algoritmos.

O tipo de distribuição dos dados, presença ou não de variáveis categóricas, variáveis correlacionadas ou não e variâncias iguais ou diferentes não provocaram mudança significativa no tempo de processamento. Quando houve aumento do número de registros, ou de variáveis, ou de grupos, houve alterações no tempo de processamento.

Apesar de o ADE levar um tempo maior quando aumenta o número de registros, quando aumenta o número de variáveis e quando aumenta o número de grupos, é interessante verificar o percentual de aumento quando esses fatores ou características se alteram.

Apesar de possuir o maior tempo de processamento em todos os ensaios do experimento, o percentual de aumento do tempo quando se aumenta o número de registros é menor no ADE do que no K-médias, como é possível ver na Figura 6.3.

A Figura 6.4, que mostra o aumento percentual do tempo quando se aumenta o número de grupos. Nota-se que o aumento percentual do tempo de processamento do ADE foi menor que nos outros dois algoritmos.

Já quando se aumenta a quantidade de variáveis das bases de dados, o aumento percentual no ADE é maior do que no K-médias e no CLARA, como mostra a Figura 6.5.

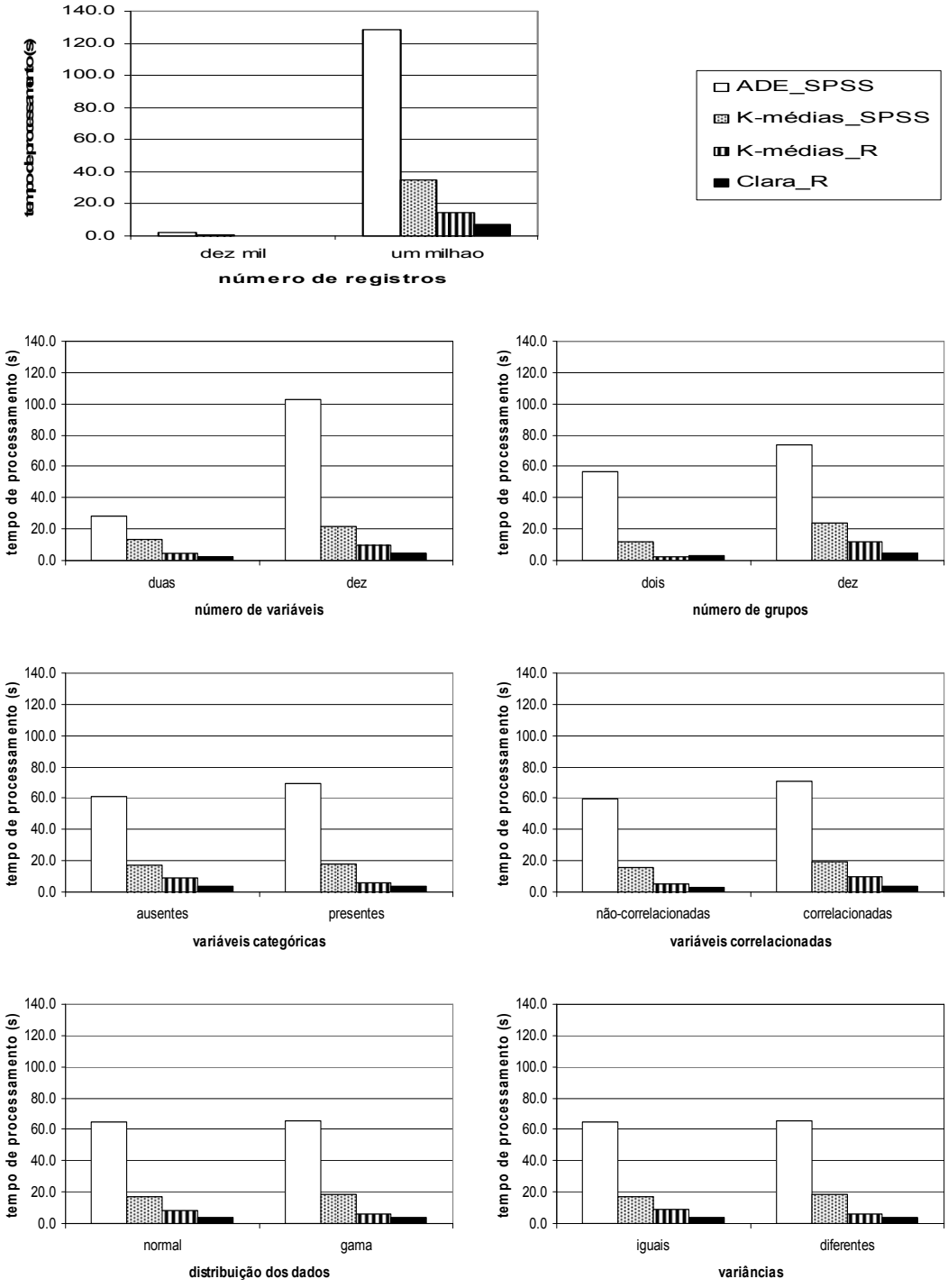


**Figura 6.2**. Tempo médio de processamento, por algoritmo e por característica da base de dados.

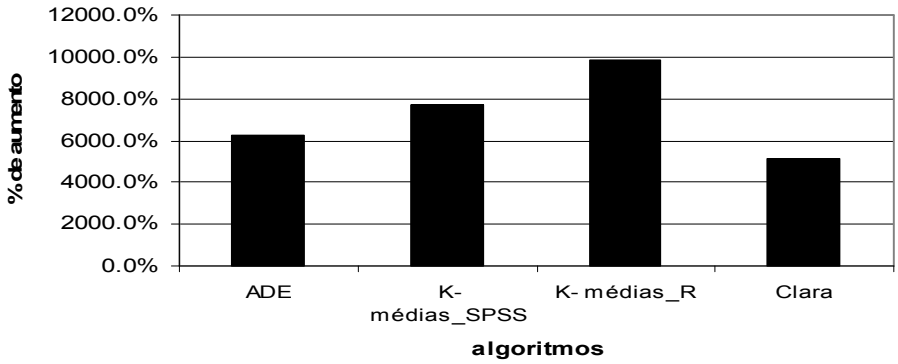


**Figura 6.3.** Porcentagem de aumento no tempo em relação ao número de registros

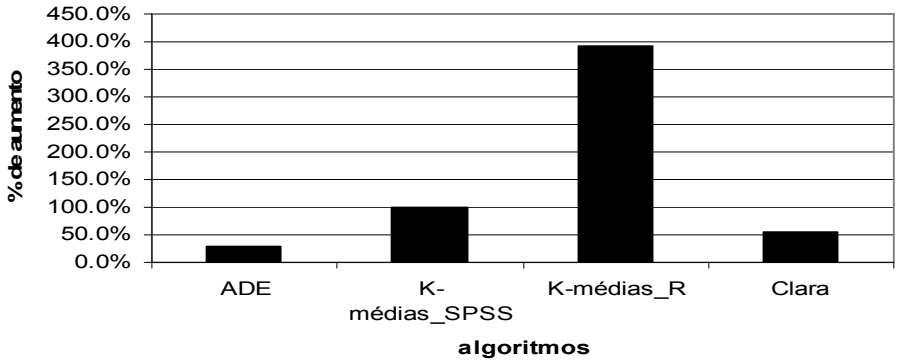


**Figura 6.4.** Porcentagem de aumento no tempo em relação ao número de grupos

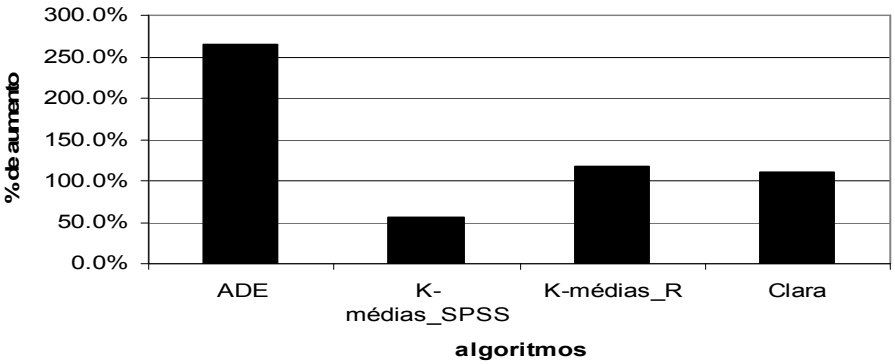


**Figura 6.5.** Porcentagem de aumento no tempo em relação ao aumento do número de variáveis.

## 6.2. Resultados da simulação baseada nos dados da íris

A Tabela 6.3 apresenta os resultados obtidos pelos algoritmos em estudo. Observa-se que o ADE e o CLARA obtiveram as melhores acurácias em todas as bases simuladas, com uma pequena vantagem para o algoritmo CLARA. O K-médias obteve os piores resultados.

Complementarmente, foi verificada a capacidade do algoritmo ADE em detectar corretamente o número de grupos. Em todas as cinco bases simuladas, o algoritmo construiu dois agrupamentos. Isto não é uma crítica à facilidade do algoritmo em avaliar o número de grupos, especialmente porque os grupos Virginica e Versicolor são muito próximos (ver Figura 5.1).

**Tabela 6.3.** Porcentagem de registros agrupados corretamente. Experimento baseado nos dados da íris.

| Base de dados gerada | Algoritmo | | | |
|---|---|---|---|---|
| | ADE | K-médiasSPSS | K-médiasR | CLARA |
| 1 | 87,4 | 84,2 | 90,9 | 93,3 |
| 2 | 87,6 | 61,2 | 90,7 | 92,2 |
| 3 | 86,6 | 84,3 | 52,3 | 90,1 |
| 4 | 87,1 | 61,3 | 52,0 | 90,7 |
| 5 | 84,2 | 84,2 | 90,7 | 89,8 |
| Média: | 86,6 | 75,0 | 75,3 | 91,2 |
| Desv.Pad | 1,4 | 12,6 | 21,1 | 1,5 |

## 7. CONSIDERAÇÕES FINAIS

A tarefa de agrupamentos parece estar se tornando cada vez mais importante em mineração de dados, porém existem muitos algoritmos de agrupamentos. Na revisão da literatura foram citados apenas alguns. Por isso, trabalhos que comparem esses algoritmos são importantes. Neste trabalho o ADE foi comparado com o CLARA e o K-médias, que também são algoritmos que servem para ser utilizados em grandes bases de dados, em termos da qualidade dos resultados gerados pelo algoritmo (acurácia) em bases de dados simuladas, e do tempo de processamento do algoritmo, que representa o tempo que o algoritmo levou para realizar a tarefa de agrupamentos.

Foram feitas comparações, utilizando simulações Monte Carlo, sob a metodologia de planejamento de experimentos, do algoritmo ADE em relação aos algoritmos CLARA e K-médias. Nas condições experimentais consideradas, em que se avaliou a acurácia dos algoritmos para um determinado número de agrupamentos fornecido previamente, em média, o ADE teve melhor acurácia, seguido de perto pelo CLARA. Por outro lado, o ADE foi o mais lento e o CLARA, o mais rápido.

Observou-se nas simulações que a vantagem do ADE aumenta quando se têm poucas variáveis e estas têm variâncias bem diferentes nos grupos, situação em que o K-médias tem, relativamente, acurácia muito baixa. Por outro lado, o CLARA parece menos vulnerável quando os dados têm distribuições assimétricas e quando há correlações não-nulas entre as variáveis.

Apesar de o ADE ter sido mais lento que os outros dois algoritmos, verificou-se que quando se aumenta o número de registros e o número de grupos, o aumento proporcional do tempo é menor no ADE do que no CLARA ou no K-médias. O aumento do tempo de processamento no ADE, relativamente aos outros dois, é maior apenas quando se aumenta o número de variáveis. Cabe observar que o tempo de processamento está muito relacionado com a forma de implementação dos algoritmos nos softwares utilizados

Foram feitas, também, simulações com dados gerados com estatísticas obtidas de um conjunto de dados real, verificando somente a acurácia dos três algoritmos. O ADE novamente teve, em média, acurácia superior ao algoritmo K-médias. O ADE teve leve desvantagem em relação aos resultados do algoritmo CLARA.

A metodologia de planejamento de experimentos mostrou-se adequada para a realização de experimentos que serviram para comparar os três algoritmos, pois foi possível a verificação da influência de várias características das bases de dados no desempenho dos algoritmos, algo que pode ser difícil de se verificar sem o uso de planejamento de experimentos. O resultado das simulações baseadas na metodologia de planejamento de experimentos ajudou a verificar em que situações o ADE pode ser utilizado de maneira adequada.

O ADE tem um processo bem definido para encontrar automaticamente um número adequado de agrupamentos, porém nas simulações realizadas esta qualidade do algoritmo não foi verificada. Neste trabalho foram geradas bases de dados com variáveis quantitativas e categóricas, já que o algoritmo ADE tem a vantagem de permitir o uso de ambos os tipos de dados.

Uma limitação do ADE, assim como do K-médias e da maioria dos algoritmos de agrupamentos, é que eles não levam em conta a correlação entre as variáveis de entrada. Em Brazão et al. (2007) foi sugerido um aprimoramento do algoritmo ADE, permitindo incorporar a informação da correlação entre as variáveis. Nesse trabalho foram sugeridas modificações na fórmula da distância de logaritmo da verossimilhança que, apesar de provavelmente aumentarem o tempo de processamento do algoritmo, permitem a formação de agrupamentos em formatos de elipse, o que atualmente não é possível.

**Sugestões de trabalhos futuros**

Após a conclusão deste trabalho, chegou-se a algumas idéias que podem ser realizadas em trabalhos futuros.

O algoritmo ADE poderia ser implementado em algum pacote estatístico livre, já que atualmente a única distribuição do software é paga e não possui o código aberto para futuras modificações e atualizações. Poderia ser feito um estudo mais a fundo nos mecanismos internos do algoritmo para diminuir o tempo de processamento na formação dos agrupamentos.

A metodologia de comparações de algoritmos de agrupamentos poderia ser utilizada em um conjunto maior de algoritmos, já que este trabalho ficou limitado à comparação do ADE com o K-médias e o CLARA.

Poderiam ser realizados estudos para verificar a possibilidade de se utilizar, na segunda etapa do algoritmo ADE, algum algoritmo de partição, com o objetivo de tornar a tarefa de agrupamentos mais rápida.

Sugere-se a implementação das modificações no ADE, conforme proposta em Brazão et al. (2007), para verificar, na prática, o seu funcionamento e sua adaptabilidade ao algoritmo.

## REFERÊNCIAS BIBLIOGRÁFICAS


Andrade, Dalton; Barbetta, Pedro; Freitas, Paulo; Zunino, Ney; Jacinto, Carlos. **Using copulas in risk analysis**. *Winter Simulation Conference. 2006.*

Anjos, U; Ferreira, F; Kolev, N; Mendes, B. **Modelando Dependências via Cópulas**. São Paulo: Associação Brasileira de Estatística, 2004.

Banfield Jeffreyd; Raftery, Adrian. ***Model-based Gaussian and non-Gaussian clustering***. *Biometrics*, 49. p. 803–821. 2003.

Barbetta, Pedro. **Estatística aplicada às ciências sociais**. 6a edição revisada. Editora da UFSC, 2006.

Berry, Michael; Linoff, Gordon. ***Data Mining Techiniques for marketing, sales, and customer relationship management***. Second Edition. Wiley Publishing Inc, 2004.

Blackwell, David. **Estatística básica**. McGRAW – Hill do Brasil, 1975.

Brazão, Rafael; Barbetta, Pedro; Andrade, Dalton. **TwoStep Cluster: Análise comparativa do algoritmo e proposta de melhoramento da medida de verossimilhança**. Anais do Workshop em Algoritmos e Aplicações de Mineração de Dados, WAAMD, em João Pessoa, 2007.

Bussab, Wilton; Miazaki, Édina; Andrade, Dalton.. **Introdução à análise de agrupamentos. 9º Simpósio nacional de probabilidade e estatística**. São Paulo, ed ABE, 1990.

Chiu, Tom; Fang, DongPing; Chen, John; Wang, Yao; Jeris, Christopher. **A robust and scalable clustering algorithm for mixed type attributes in large database environment**. Anais de ACM KDD 01, p. 263-268, São Francisco, California. 2001.

Ester, Martin; Kriegel, Hans-Peter; Sander, Jörg; Xu, Xiaowei. **A Density Based Algorithm for Discovery Clusters in Large Spatial Databases with Noise**. Anais da 2ª Conferencia Internacional em KDD, 1996.

Fayyad, Usama; Piatetsky-Shapiro; Smyth, Padhraic. **From Data Mining to Knowledge Discovery in Databases**. Nos anais do American Association for Artificial Intelligence. p. 37-54. 1996

Freeman, James; Skapura, David. **Neural Networks – Algorithms, Applications, and Programming Techniques**. Addison-Wesley, 1991.

Han, Jiawei; Kamber, Micheline. **Data Mining: Concepts and Techniques**. Editora Morgan Kaufmann publishers, 2001.

Huang, Zhexue. **A Fast Clustering Algorithm to Cluster Very Large Categorical Data Sets in Data Mining**, 1998.

Johnson, Richard; Wichern, Dean. **Applied *Multivariate Statistical Analysis***. Fifth Edition. Prentice Hall, 2002.

Kaufman, Leonard; Rousseeuw, Peter. **Finding Groups in Data – An Introduction to Cluster Analysis**. John Wiley & Sons, 2005.

Mingoti, Sueli ; Lima, Joab. **Comparing SOM neural network with Fuzzy c-means, K-means and traditional hierarchical clustering algorithms** European Journal of Operational Research, Volume 174, Issue 3, 1 November p. 1742-1759, 2006.

Montgomery, Douglas. **Design and Analysis of Experiments**. Sixth Edition. John Wiley & Sons, 2005.

Ng, Raymond; Han, Jiawei. **Efficient and Effective Clustering Methods for Spatial Data Mining**. Anais da 20ª Conferência VLDB. Santiago, Chile, 1994.

Prass, Fernando. **Estudo comparativo entre algoritmos de Análise de Agrupamentos em Data Mining**. Dissertação de mestrado - UFSC. Florianópolis, 2004.

R Development Core Team (2006). **R: A language and environment for statistical computing**. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.

Rezende, Solange. **Sistemas Inteligentes – Fundamentos e Aplicações**. Editora Manole. 2005.

Sheikholeslami, Gholamhosein; Chatterjee, Surojit; Zhang, Aidong. WaveCluster: **A Multi-Resolution Clustering Approach for Very Large Spatial Databases**. Anais da 24a conferencia VLDB. p. 428-439. Nova Iorque, EUA, 1998.

Song, Xue-Kun P. **Multivariate dispersion models generated from gaussian copula**. Scandinavian Journal of Statistics v. 27, p. 305-330. 2000.

Tenembaum, Aaron; Langsam, Yedidyah; Augenstein, Moshe. **Data structures using C**. Makron Books, 2004.

SPSS. **The SPSS TwoStep Cluster Component – A scalable component enabling more efficient customer sugmentation**. White paper – technical report. USA, 2001.

Vose, David. **Risk analysis – A quantitative guide**. Second edition. John Wiley & Sons, 2000.

Wikipédia. Árvore(Estrutura de dados).
URL: http://pt.wikipedia.org/wiki/%C3%81rvore_(estrutura_de_dados).
Acesso em: 14 de abril de 2007.

Zhang, Tian; Ramakrishnan, Raghu; Livny, Miron. **Birch: an efficient data clustering method for very large databases**. Anais de ACM SIGMOD Conference on Management of Data, Montreal, Canadá, junho de 1996.

Zhang, Tian. **Data Clustering for very large datasets plus applications**. Tese de doutorado. Universidade de Wisconsin-Madison, 1997.

## APÊNDICE A: Script utilizado na linguagem R para geração dos dados

Para automatizar a geração das bases de dados foi utilizado um script para cada um dos ensaios do experimento descrito no Capitulo 5. Abaixo segue o exemplo do script utilizado para gerar os dados do ensaio 11, para uma base de dez mil registros, duas variáveis, dois grupos, sem a presença de variáveis categóricas, sem correlação entre os dados, dados com distribuição normal e variâncias iguais à 1 entre as variáveis.

---

```
x1 <- mvdc(normalCopula(0), c("norm", "norm"), list(list(mean = 3.4142, sd =1),list(mean = 3.4142, sd = 1)))
x2 <- mvdc(normalCopula(0), c("norm", "norm"), list(list(mean = 3.4142, sd =1),list(mean = 3.4142, sd = 1)))
x01<-rmvdc(x1,5000)
l<-rep(1,each=5000)
x02<-rmvdc(x2,5000)
m<-rep(2,each=5000)
y01 <- x01
for (i in 1:5000){
 y01[i,1] = ((x01[i,1]-3.4142)*(1))+1.00
 y01[i,2] = ((x01[i,2]-3.4142)*(1))+1.00
}
y02 <- x02
for (i in 1:5000){
 y02[i,1] = ((x02[i,1]-3.4142)*(1))+2.00
 y02[i,2] = ((x02[i,2]-3.4142)*(1))+4.00
}
l0 <- merge(y01,l,by=0, sort=FALSE)
m0 <- merge(y02,m,by=0, sort=FALSE)
v12 <- merge(l0,m0,all=TRUE, sort=TRUE)
v12 <-v12[,-1]
```

```
write.dbf(v12,"exp11Base01.dbf")
```

---

Inicialmente os grupos x1 e x2 são gerados com a função de cópula *mvdc*. Para gerar os dados na distribuição normal são inseridos os valores dos parâmetros média e desvio padrão. Os valores aleatórios gerados ficam armazenados nas variáveis x01 e x02. A Aplicação da formula mostrada nesta dissertação em (5.1) é realizada em um laço de repetição que será repetido até que todos os registros gerados tenham os seus valores alterados.

Os dados gerados nos dois grupos são unidos em uma única base de dados com a função *merge*. O parâmetro *sort=TRUE* garante que os registros dos dois grupos ficarão truncados de forma aleatória. Finalmente, a função *write.dbf* grava a base de dados gerada em um arquivo no *hard disc*, com o nome *exp11Base01.dbf*.

Os outros 15 ensaios foram gerados de maneira similar, variando de acordo com os fatores escolhidos para o experimento, mostrados nas Tabelas 5.1 e 5.2.