Instituto de Ciências Matemáticas e de Computação

ISSN - 0103-2569

Estudo e Análise das Diversas Representações e Estruturas de Dados Utilizadas nos Algoritmos de *Clustering* Hierárquico

Jean Metz
Maria Carolina Monard

N° 269

RELATÓRIOS TÉCNICOS DO ICMC

São Carlos
Janeiro/2006

Estudo e Análise das Diversas Representações e Estruturas de Dados Utilizadas nos Algoritmos de *Clustering* Hierárquico *

Jean Metz
Maria Carolina Monard

Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências de Computação e Estatística
Laboratório de Inteligência Computacional
Caixa Postal 668, 13560-970 - São Carlos, SP, Brasil
e-mail: {metzz, mcmonard}@icmc.usp.br,

Resumo:

Tarefas de aprendizado de máquina não-supervisionado são, freqüentemente, utilizadas em diversas aplicações de Mineração de Dados e Textos, nas quais grandes conjuntos de dados devem ser analisados em busca de conhecimento novo e potencialmente útil. O *clustering*, especificamente, é uma técnica de análise e exploração de dados usualmente aplicada com objetivo de extrair padrões latentes nesses conjuntos de dados. Existem diversas abordagens e algoritmos de *clustering* propostos na literatura, entre eles, os algoritmos hierárquicos, objeto de estudo deste trabalho. Esses algoritmos constróem uma estrutura que descreve os clusters identificados por meio de uma hierarquia, que pode ser representada de diversas formas, tais como dendograma ou *banner*, por exemplo. Além disso, essas formas de representação podem ser armazenadas em diferentes estruturas de dados. Um dos objetivo deste trabalho é revisar algumas das formas de representação mais comumente utilizadas no *clustering* e, também, as estruturas de dados utilizadas na implementação dessas representações. Outro objetivo é, com base nesse estudo, decidir qual das formas de representação e estrutura de dados serão utilizadas na implementação do módulo de *clustering* hierárquico do Discover, um ambiente computacional em desenvolvimento no Laboratório de Inteligência Computacional do ICMC.

Palavras-Chave: Agrupamento de dados, Clustering hierárquico, Representação.

---

janeiro 2006

*Trabalho desenvolvido com o apoio do CNPq.

Este documento foi preparado com o formatador de textos LaTeX. O sistema de citações de referências bibliográficas utiliza o padrão *Chicago* do sistema bibTeX.

Sumário

Sumário i

Lista de Figuras iii

1 Introdução 1

2 Agrupamento de Dados 2

2.1 Pré-processamento 4
2.2 Seleção da Medida de Similaridade 5
2.3 Algoritmo de Agrupamento 6
2.4 Avaliação dos Resultados 10
2.5 Interpretação 11

3 Ferramentas e Algoritmos de *Clustering* Hierárquico: Uma Visão geral 11

3.1 *AGNES: Agglomerative Nesting* 15
3.2 *BIRCH: Balanced Iterative Reducing and Clustering Using Hierarchies* . . 15
3.3 *CURE: Clustering Using Representatives* 17
3.4 *CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling* 18
3.5 *ROCK: Robust Clustering Using Links* 20
3.6 *MONA: Monothetic Analysis* 21
3.7 *DIANA: Divisive Analysis* 21
3.8 *HCE: Hierarchical Clustering EXPLORER* 22
3.9 A Biblioteca Cluster 3.0 22

4 Representação do *Clustering* 23

5 Considerações Finais 33

Referências 35

Lista de Figuras

2.1 Cluster curvilíneo. 5
2.2 Clusters com diferentes densidades. 9
3.3 *Single Link* 12
3.4 *Minimal Spanning Tree* 12
3.5 *Complete Link* 13
3.6 *Average Link* 14
3.7 Etapas do algoritmo *BIRCH*. 16
3.8 Etapas do algoritmo *CURE*. 18
3.9 Etapas do algoritmo *CHAMELEON*. 19
3.10 Ilustração do conceito de ligações 20
3.11 Etapas do algoritmo *ROCK*. 21
4.12 Diagrama de Venn 25
4.13 *Banner* 26
4.14 Identificação dos clusters a partir de um *Banner*. 27
4.15 Dendograma 27
4.16 Similaridade intra-cluster 28
4.17 Representações do dendograma 28
4.18 Diagrama de Ward. 29
4.19 Dendograma representado por meio de uma matriz quadrada. 29
4.20 *Parent Array*. 30
4.21 *Pointer Representation*. 31

1 Introdução

Métodos de Apendizado de Máquina — AM — são freqüentemente utilizados em tarefas de descoberta de conhecimento e extração de padrões. Em geral, grandes volumes de dados são produzidos nas mais diversas áreas do conhecimento, armazenando fatos que descrevem desde fenômenos naturais e estruturas bioquímicas à transações comerciais. A diversidade e quantidade de dados disponíveis acarreta na dificuldade para análise manual e compreensão das informações embutidas nesses dados. Nesse contexto, alguns métodos de AM podem ser utilizados para auxiliar na análise automática ou semi-automática de dados e no aproveitamento das informações extraídas para a descoberta de conhecimento referente ao domínio da aplicação.

Segundo o modo de aprendizado, algoritmos de AM podem ser classificados como: *supervisionado*, *não-supervisionado* e *semi-supervisionado*.

No aprendizado supervisionado é fornecido ao algoritmo de aprendizado, ou indutor, um conjunto de exemplos de treinamento para os quais o rótulo da classe associada é conhecido. No aprendizado não-supervisionado não há classe associada aos exemplos, e o algoritmo analisa os exemplos fornecidos para determinar se alguns deles podem ser agrupados considerando alguma medida ou critério de similaridade, formando agrupamentos desses exemplos (clusters) (Cheeseman & Stutz, 1996). Recentemente, surgiu um terceiro modo de aprendizado, denominado semi-supervisionado, no qual somente poucos exemplos encontram-se rotulados impossibilitando o uso direto de algoritmos de aprendizado supervisionado. O objetivo principal desse modo de aprendizado é tentar rotular, mas somente se certas condições são satisfeitas, um número maior de exemplos a fim de incrementar o número de exemplos rotulados, para posteriormente utilizá-los no aprendizado supervisionado.

Em resumo, os exemplos fornecidos a um algoritmo de AM podem, ou não, estar rotulados com uma classe conhecida. De acordo com esse critério, três modos de aprendizado podem ser diferenciados:

- **Aprendizado supervisionado**: utilizado quando existe um número expressivo de exemplos rotulados;
- **Aprendizado semi-supervisionado**: utilizado quando existem poucos exemplos rotulados;
- **Aprendizado não-supervisionado**: utilizado quando os exemplos não estão rotulados.

As tarefas mais comuns do modo de aprendizado não-supervisionado são a indução de regras de associação, a sumarização de textos e o agrupamento de dados ou *clustering*, foco principal deste trabalho.

O resultado obtido pelo *clustering* é um conjunto de agrupamentos de dados, no qual cada agrupamento é denominado cluster. Pode-se caracterizar um cluster como sendo um

agrupamento composto de objetos (exemplos) similares de acordo com alguma medida de similaridade. Algumas definições de *clustering* são apresentadas a seguir na Seção 2.

Diversos algoritmos de *clustering* foram propostos na literatura e alguns desses foram implementados. Em geral, cada algoritmo utiliza uma forma de representação dos agrupamentos e uma estrutura de dados interna para implementar essa representação. Este trabalho tem dois objetivos principais:

1. descrever algumas das formas de representação usualmente empregadas para o *clustering*, especificamente para a abordagem hierárquica, e as estruturas de dados mais freqüentemente utilizadas para a implementação dessas representações; e

2. com base nesse estudo, indicar uma representação/estrutura de dados para ser utilizada na implementação do módulo de *clustering* hierárquico no ambiente DISCOVER (Baranauskas & Batista, 2000; Batista & Monard, 2003), em desenvolvimento no Labic — Laboratório de Inteligência Computacional.

Este trabalho está organizado da seguinte maneira: na Seção 2 são apresentados brevemente alguns conceitos relacionados ao aprendizado de máquina não-supervisionado, mais especificamente o *clustering* ou agrupamento de dados. Na Seção 3, são apresentados alguns algoritmos de *clustering* que seguem a abordagem hierárquica, assim como uma ferramenta computacional para auxiliar a análise dos agrupamentos e uma biblioteca que implementa diversos métodos de *clustering* hierárquico. Na Seção 4 são discutidas as formas de representação do *clustering* e algumas estruturas de dados freqüentemente utilizadas para implementação dessas representações. As considerações finais deste trabalho são apresentadas na Seção 5.

2 Agrupamento de Dados

Neste trabalho são apresentados, especificamente, os conceitos, algoritmos e ferramentas relacionados ao *clustering*. Dessa maneira, o termo aprendizado não-supervisionado será utilizado, neste trabalho, em referência ao *clustering*. Caso haja necessidade de utilizá-lo em outro contexto, isso será explicitamente indicado.

Algoritmos de *clustering* agrupam exemplos de dados baseados em índices de proximidade (similaridade) entre pares. O conjunto desses exemplos pode ser descrito por meio de duas estruturas: tabela de exemplos e matriz de similaridade (Jain & Dubes, 1988), descritas a seguir.

Tabela de exemplos: o conjunto de exemplos $\{E_1, E_2, ..., E_N\}$ é representado por uma tabela $N \times M$, onde N é o número de exemplos e M o número de atributos que os descrevem. Cada linha i da tabela define um exemplo E_i e cada coluna j denota um atributo x_j. Em outras palavras, cada exemplo é representado por um vetor no espaço M-dimensional — Tabela 2.1.

Os M atributos descrevem os eixos ortogonais do espaço de dados de modo que um cluster pode ser interpretado como sendo uma coleção de exemplos que estão próximos uns dos outros, ou satisfazem alguma relação espacial.

	X_1	X_2	$\dots$	X_M
E_1	x_{11}	x_{12}	$\vdots$	x_{1M}
E_2	x_{21}	x_{22}	$\vdots$	x_{2M}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
E_N	x_{N1}	x_{N2}	$\vdots$	x_{NM}

Tabela 2.1: Representação do conjunto de dados por meio da tabela de exemplos.

Matriz de similaridade: os métodos de *clustering* requerem que um índice de proximidade seja estabelecido para cada par de exemplos e clusters. Esse índice pode ser calculado a partir da tabela de exemplos. Uma matriz de similaridade S armazena em cada célula S_{ij} a similaridade entre os exemplos E_i e E_j, *i.e.*, $S_{ij} = sim(E_i, E_j)$.

Quando a representação utilizada é a tabela de exemplos, as células da tabela, ou valores dos atributos $X_1, X_2 \dots X_M$, podem assumir tipos diferentes, tais como: binário, discreto ou contínuo. Atributos binários podem assumir somente dois valores, por exemplo *sim* e *não* ou 1 e 0, indicando presença ou ausência de uma determinada característica. Atributos discretos possuem, usualmente, um conjunto finito e freqüentemente pequeno de valores possíveis, por exemplo meses do ano ou dias da semana. Os atributos contínuos, por outro lado, podem assumir qualquer valor real dentro de um intervalo pré-definido. Além dos tipos dos atributos, o *clustering* é influenciado pela escala, que indica a significância relativa dos valores dos atributos. Assim, a escala dos atributos pode ser qualitativa (nominal ou ordinal) ou quantitativa (intervalo de valores ou proporção), descritos a seguir (Jain & Dubes, 1988).

- **Qualitativa**
  - *Nominal*: os valores são apenas nomes diferentes. Exemplo: atributos cores e sexo.
  - *Ordinal*: os valores refletem uma ordem. Exemplo: atributos hierarquia militar e avaliações qualitativas como frio, morno e quente.
- **Quantitativa**
  - *Intervalo*: existe uma unidade de medida e a interpretação dos atributos dependem dessa unidade. Exemplo: atributos temperatura (90° *Célcius* é diferente de 90° *Fahrenheit*), a duração de um evento em minutos ou segundos.
  - *Proporção*: os valores têm um significado absoluto. Isto significa que existe um zero absoluto junto com uma unidade de medida, de modo que a proporção tenha significado. Exemplo: atributos altura, salário e distância.

O resultado obtido após a realização do agrupamento é um conjunto de partições sobre o conjunto de dados original, na qual cada partição representa um cluster. Diversas definições de cluster são encontradas na literatura, algumas das mais freqüentemente utilizadas são (Everitt, 1993):

Definição 1: *um cluster é um conjunto de entidades semelhantes, e entidades pertencentes a clusters diferentes não são semelhantes.*

Definição 2: *um cluster é um agrupamento de pontos distribuídos no espaço de teste de tal maneira que a distância entre quaisquer dois pontos em um mesmo cluster é menor que a distância entre qualquer ponto desse cluster e um outro ponto qualquer não pertencente a ele.*

Definição 3: *clusters podem ser descritos como regiões conectadas de um espaço multidimensional contendo uma alta densidade relativa de pontos, separadas de outras regiões por uma região contendo uma baixa densidade relativa de pontos.*

Para obter os agrupamentos, algumas etapas devem ser realizadas, tais como: pré-processamento, seleção da medida de similaridade, execução do algoritmo de *clustering*, avaliação dos resultados e, finalmente, a interpretação dos clusters encontrados. Essas etapas são descritas nas próximas seções.

2.1 Pré-processamento

A preparação e transformação dos dados devem ser realizadas nesta etapa. Freqüentemente é necessário que algum tipo de normalização seja realizada sobre os dados, a qual considera a medida de similaridade que será aplicada na etapa de agrupamento. Por exemplo, a distância Euclidiana é um índice geralmente utilizado para medir a similaridade entre exemplos. Porém, ela atribui maior importância para as dimensões (atributos) que assumem valores em um intervalo maior que aquelas que assumem valores em um intervalo menor. Manter os atributos dentro de uma faixa de valores comum pode minimizar esse problema.

Outro aspecto importante é a base ou sistemas de coordenadas em que os dados estão representados. Em geral, assume-se que os dados já estão numa base adequada para a aplicação de um algoritmo de *clustering*. No entanto, uma análise mais cuidadosa das características disponíveis e das transformações que podem ser aplicadas aos dados pode auxiliar na obtenção de resultados significativamente melhores. Um exemplo simples é o agrupamento do conjunto de dados ilustrado na Figura 2.1 na página oposta. Os exemplos apresentados nessa figura formam um cluster curvilíneo com distância da origem aproximadamente constante. Utilizando uma representação em coordenadas Cartesianas, muitos algoritmos de agrupamento produziriam dois ou mais clusters. Entretanto, se fossem utilizadas coordenadas polares para representar esses exemplos, a solução de um único cluster poderia ser obtida mais facilmente (Jain et al., 1999).

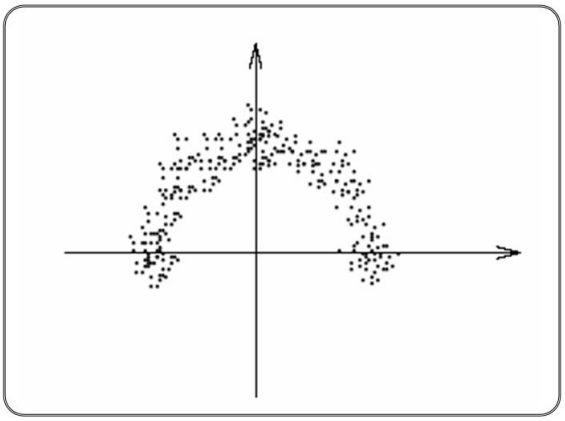


Figura 2.1: Cluster curvilíneo (Jain et al., 1999).

A dimensionalidade dos exemplos é outra característica que deve ser considerada na fase de preparação dos dados. O *clustering* é uma tarefa de exploração de dados que deve ser apoiada por técnicas de visualização com o intuito de facilitar a interpretação dos resultados. A maneira mais direta de visualização é dispor os dados em um espaço bi-dimensional $\mathbb{R}^2$ ou tridimensional $\mathbb{R}^3$, representando os exemplos agrupados por meio de pontos nesse espaço. Entretanto, dados multi-variados ($\mathbb{R}^n$ para $n > 3$) não podem ser diretamente visualizados ou projetados precisamente em duas ou três dimensões. A representação de subconjuntos dos atributos no $\mathbb{R}^2$ ou $\mathbb{R}^3$ pode ser de grande ajuda para o entendimento dos dados, mesmo quando $n > 3$. Além disso, um número menor de atributos reduz substancialmente o tempo de execução dos algoritmos. Desse modo, alguns métodos de seleção de atributos são usualmente empregados com o objetivo de selecionar apenas os atributos que melhor descrevem e discriminam o conjunto de dados e suas estruturas latentes, o que conseqüentemente reduz a dimensionalidade dos dados, melhora a eficiência dos algoritmos em relação ao tempo de execução e, em alguns casos, pode melhorar os resultados (Liu & Motoda, 1998).

2.2 Seleção da Medida de Similaridade

Nesta etapa devem ser determinadas as medidas de similaridade utilizadas para realizar o agrupamento dos dados, as quais devem ser cuidadosamente selecionadas considerando as características do conjunto de dados, tais como o tipo e escala dos atributos. Foram propostas diversas medidas para o cálculo da similaridade, entre as quais estão as medidas de distância, de correlação e de associação. Quando o conjunto de dados é composto por atributos numéricos (quantitativos), as métricas de distância podem ser aplicadas para o cálculo da similaridade entre os exemplos. O problema com a aplicação dessas métricas ocorre quando os exemplos possuem atributos qualitativos pois, nesse caso, se não existir uma ordenação conceitual dos valores desses atributos, não é possível calcular a similaridade entre os exemplos por meio das métricas de distância. Uma estratégia geralmente utilizada para o cálculo da similaridade nesses casos, é aplicar o coeficiente de

correspondência, que verifica se os atributos de exemplos diferentes apresentam o mesmo valor ou diferem. Assim, se os valores dos atributos são iguais, então a similaridade para aquele atributo é um, caso contrário é zero. Desse modo, cada uma das métricas e medidas de similaridade representa uma perspectiva, dependendo das características dos dados. As métricas de distância e medidas de correlação requerem atributos contínuos, enquanto que as medidas de associação são utilizadas para atributos discretos. Uma revisão de vários tipos de medidas de similaridade pode ser encontrada em (Jain & Dubes, 1988) e (Everitt, 1993).

Para a formação dos agrupamentos deve-se observar também a similaridade entre os clusters, chamadas de similaridade inter-cluster ou método de agrupamento, dos quais alguns são descritos na Seção 3.

2.3 Algoritmo de Agrupamento

Neste estágio, o conjunto de exemplos deve ser submetido a algum algoritmo para o agrupamento dos dados. Os algoritmos de *clustering* podem ser classificados considerando diferentes aspectos. Uma das classificações bastante aceita é a dada por Jain et al. (1999), na qual os algoritmos são classificados de acordo com a abordagem adotada para definir os clusters. Nesse caso, os algoritmos são divididos em particionais ou de otimização, baseados em grade, baseados em densidade, *clumping*, redes auto-organizáveis, conceitual e hierárquicos, sendo essa última a abordagem de maior interesse para o desenvolvimento deste trabalho.

O objetivo do *clustering* de otimização é formar agrupamentos ótimos sobre os dados, dividindo iterativamente o conjunto de exemplos em k-partições mutuamente exclusivas, as quais devem maximizar uma função critério pré-definida (Berkhin, 2002). Um problema associado ao *clustering* de otimização é a necessidade de informar com antecedência o número de clusters desejados. Além disso, apresenta alta variância, pois a seleção dos exemplos representantes (sementes iniciais) afeta, significativamente, o resultado do *clustering* e pode fazer com que a solução vá em direção a um máximo local da função de avaliação. Para minimizar esse efeito negativo, os algoritmos são executados diversas vezes com sementes diferentes e, então, a melhor solução é atribuída ao resultado do processo de *clustering*. Os algoritmos *k-means* e *k-medoids* são os representantes mais utilizados dessa abordagem.

No caso do *clustering* baseado em grade, uma grade é definida para o espaço de exemplos e todas as operações de agrupamento são realizadas nesse espaço quantizado. Em termos gerais, essa abordagem é capaz de encontrar clusters de formatos arbitrários, não apresenta alta sensibilidade aos *outliers* e é muito eficiente para grandes conjuntos de dados. Exemplos de algoritmos que seguem essa abordagem são CLIQUE (Agrawal et al., 1998), MAFIA (Nagesh et al., 1999), OptiGrid (Hinneburg & Keim, 1999) e STING (Wang et al., 1997).

O *clustering* baseado em densidade assume que os clusters são regiões de alta densi-

dade separadas por regiões com baixa densidade no espaço de exemplos. A idéia dessa abordagem é que cada exemplo do cluster deve manter uma vizinhança com um número mínimo de vizinhos dentro de uma esfera com raio R. Os exemplos que possuem uma vizinhança com densidade mínima, e estão numa distância menor que R pertencem ao mesmo cluster. Existem diversos algoritmos que seguem essa abordagem, tais como DENCLUE (Hinneburg & Keim, 1998), DBSCAN (Ester et al., 1996), OPTICS (Ankerst et al., 1999) e Wave-cluster (Sheikholeslami et al., 1998).

Um agrupamento exclusivo é uma partição do conjunto de exemplos, no qual cada exemplo pertence exclusivamente a um único cluster. O resultado desse agrupamento pode ser dito *crisp*, quando a relação entre os exemplos e os agrupamentos resume-se em pertencer ou não a um dado cluster. Na abordagem de agrupamento não exclusivo (com sobreposição ou *clumping*), pode-se associar um exemplo a vários clusters. Nesse caso, a cada exemplo é atribuído um grau de pertinência em relação aos clusters encontrados. Assim, um exemplo pode estar em mais de um agrupamento, com probabilidades diferentes. O AutoClass (Cheeseman & Stutz, 1996) é um dos algoritmos que implementam essa idéia.

Além das abordagens apresentadas anteriormente, outras maneiras de realizar o *clustering* têm sido propostas. Uma abordagem que estima a similaridade dos exemplos por meio de cálculos matemáticos são as redes auto-organizáveis, também conhecidas como redes SOM — *Self Organizing Map* (Kohonen, 1990; Haykin, 1999) —, uma rede neural artificial não-supervisionada freqüentemente utilizada em tarefas de agrupamento e visualização. Nesse tipo de rede, os neurônios são organizados em um reticulado bi-dimensional, onde cada neurônio está conectado a todas as entradas da rede. A cada exemplo de entrada apresentado à rede, os neurônios computam seus valores de ativação, ativando uma região diferente do reticulado determinando a localização espacial de uma vizinhança topológica de neurônios excitados, centrada no neurônio vencedor. Desse modo, durante a execução do algoritmo ocorre uma organização topológica dos neurônios da rede. Dois algoritmos bastante conhecidos que implementam redes auto-organizáveis são SOTA (Herrero et al., 2001) e SomPack (Kohonen et al., 1996).

Em geral, algoritmos de *clustering* baseados apenas em similaridade, como os apresentados, não utilizam conhecimento semântico na formação dos clusters. Além disso, na definição dos agrupamentos, certos atributos que descrevem os exemplos são mais importantes do que outros e, geralmente, medidas simples de similaridade tratam todos os atributos igualmente.

Do ponto de vista da Inteligência Artificial — IA —, as abordagens baseadas apenas em medidas de similaridade numérica são ditas metodologias fracas, pois são metodologias gerais que utilizam pouco ou nenhum conhecimento do domínio do problema. As medidas de similaridade são usualmente definidas como medidas de proximidade ou distância em um espaço multidimensional ($\mathbb{R}^n$) onde os atributos representam os eixos ortogonais desse espaço. Essas medidas, entretanto, são significativas somente se os atributos selecionados

são relevantes para a descrição da similaridade dos objetos (exemplos) analisados. Nesse sentido, a presença de atributos irrelevantes pode degradar o poder de descrição dessas medidas (Michalski & Stepp, 1983). O uso de medidas de similaridade numérica apresenta outras desvantagens, pois consideram apenas as propriedades dos exemplos envolvidos sem analisar qualquer contexto ou conceito que poderia caracterizar melhor o conjunto de exemplos. Conseqüentemente, o resultado não terá necessariamente uma descrição simples dos conceitos embutidos nos dados e, provavelmente, será de difícil interpretação. O *clustering* conceitual foi proposto com o objetivo de superar essas limitações. Nessa abordagem, a técnica de *clustering* depende do objetivo e dos conceitos disponíveis ao sistema para a caracterização do conjunto de exemplos. Sua premissa básica é que os exemplos devem ser organizados em clusters que tenham algum significado conceitual simples e que seja útil do ponto de vista da aplicação. Desse modo, exemplos de um mesmo cluster não são similares somente segundo alguma medida de similaridade definida matematicamente, mas também segundo um mesmo significado conceitual descrito pelo cluster (Fisher & Langley, 1986; Stepp & Michalski, 1986; Bhatia & Deogun, 1998; Mishra et al., 2004).

O *clustering* hierárquico, assim como as outras abordagens de *clustering*, constrói os agrupamentos de modo que exemplos pertencentes ao mesmo cluster possuem alta similaridade e exemplos pertencentes à clusters diferentes possuem baixa similaridade. Entretanto, uma distinção entre essa abordagem e as demais é que o resultado obtido não é constituído apenas de uma partição do conjunto de dados inicial, mas sim de uma hierarquia que descreve um particionamento diferente à cada nível analisado. Um conjunto de dados, geralmente, contém diversos clusters e esses clusters, por sua vez, contém sub-clusters. Os sub-clusters podem ainda ser formados a partir do agrupamento de outros clusters menores (sub-subclusters), e assim sucessivamente. Na Biologia por exemplo, os reinos são divididos em filos, que são subdivididos em sub-filos, os quais são divididos em famílias, sub-famílias, gêneros e espécies, que constitui uma hierarquia (Duda et al., 2000). Nesses casos, torna-se necessária a utilização de uma representação formal para a hierarquia de clusters obtida a partir dos dados. O dendograma é a estrutura mais freqüentemente utilizada para representar essa hierarquia, que consiste de um tipo especial de estrutura de árvore, na qual os nós pais agrupam os exemplos representados pelos nós filhos. Outras estruturas são também utilizadas para representar a hierarquia de agrupamentos, tais como Diagramas de Venn, *IciclePlot* e *Banner*. Essas estruturas são apresentadas brevemente na Seção 4.

Duas estratégias podem ser utilizadas com o *clustering* hierárquico:

1. Aglomerativa (*Botton-Up*); e

2. Divisiva (*Top-down*).

Na primeira, cada exemplo é considerado um cluster unitário. Em seguida, pares desses clusters são iterativamente agrupados considerando alguma medida de similaridade, até

que todos os exemplos pertençam a apenas um cluster. Por outro lado, a abordagem divisiva é iniciada com apenas um agrupamento contendo todos os exemplos e procede dividindo o conjunto de exemplos em clusters cada vez menores, até que cada exemplo pertença exclusivamente a um cluster ou até que se alcance algum critério de parada, freqüentemente o número de clusters desejados (Murtagh, 1983).

Na literatura, trabalhos relacionados ao *clustering* hierárquico, geralmente, referenciam à abordagem aglomerativa, não despertando muito interesse pelos métodos divisivos. Mesmo os pacotes de *software* que implementam algoritmos de *clustering* dificilmente incluem algoritmos divisivos. O principal aspecto que influencia esse fato é a complexidade computacional desses métodos. Algoritmos aglomerativos são quadráticos em relação ao número de exemplos do conjunto de entrada, *i.e.*, $O(N^2)$, mas ainda assim muitas vezes aplicáveis. A complexidade dos algoritmos divisivos, por outro lado, cresce exponencialmente em relação ao tamanho do conjunto de entrada, o que torna proibitiva sua aplicação sobre conjuntos de dados com pouco mais que algumas centenas de exemplos (Kaufman & Rousseeuw, 1990b).

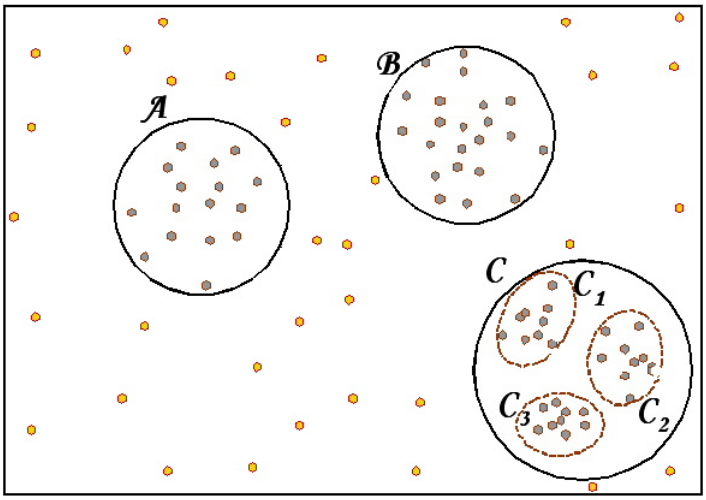


Figura 2.2: Clusters com diferentes densidades (Ankerst et al., 1999).

Um aspecto positivo do agrupamento hierárquico é a flexibilidade em relação à analise dos diferentes níveis de granularidade ou densidade dos agrupamentos. Diversos conjuntos de dados reais possuem a propriedade de que, em geral, as estruturas intrínsecas não podem ser caracterizadas por parâmetros globais de densidade ou por funções matemáticas que não consideram os diversos níveis de granularidade dos clusters. Em alguns casos, os parâmetros e densidade teriam que ser muito diferentes para que os algoritmos pudessem identificar os clusters com densidade variadas. Considere, por exemplo, o conjunto de dados apresentado na Figura 2.2. Nesse caso, não seria possível identificar, simultaneamente, os clusters A, B, C1, C2 e C3 utilizando valores de densidade globais, mas sim os clusters A, B e C ou C1, C2 e C3, no último caso considerando os clusters A e B como ruído ou *outliers*. Sob esse aspecto, o clustering hierárquico é uma alternativa para identificar corretamente essas estruturas, pois com a análise em diferentes níveis da hierarquia

é possível observar agrupamentos com densidades distintas.

Um fator negativo associado ao *clustering* em geral, é que os resultados obtidos por meio dessa técnica são de difícil interpretação, o que torna necessária a maior interação com um especialista do domínio e o desenvolvimento de metodologias que facilitem a compreensão dos agrupamentos encontrados.

2.4 Avaliação dos Resultados

Esta etapa é realizada para determinar o grau de significância dos resultados obtidos pelo algoritmo de agrupamento. O fato de não haver uma classe definida para os dados não-supervisionados acarreta na utilização de suposições para a identificação das estruturas embutidas nos dados, que são avaliadas e validadas considerando a qualidade dos clusters encontrados pelos diversos algoritmos, *i.e.*, por meio da análise de índices que medem a adequabilidade da estrutura encontrada em termos das probabilidades dos clusters serem corretos. A adequabilidade dos clusters extraídos refere-se ao fato de representarem corretamente a informação contida nos dados, ou a habilidade de recuperarem padrões que reflitam os conceitos intrínsecos ao conjunto de dados.

A validação do *clustering*, em geral, é realizada com base em índices estatísticos que julgam, de uma maneira quantitativa, as estruturas encontradas no conjunto de dados. A maneira pela qual um índice é aplicado para validar um agrupamento é dada pelo critério de validação. Assim, um critério de validação expressa a estratégia utilizada para validar um estrutura de agrupamento, enquanto que um índice é um valor estatístico pelo qual a validade é testada. Existem três tipos de critérios para investigar a validade de um agrupamento (Halkidi et al., 2002a,b):

1. Critérios relativos;

2. Critérios internos; e

3. Critérios externos.

Os critérios relativos têm como objetivo encontrar o melhor agrupamento que um algoritmo pode obter sob certas suposições e valores para seus parâmetros, ou o algoritmo mais apropriado para os dados e estruturas analisadas.

Os critérios externos e internos são baseados em testes estatísticos e têm um alto custo computacional. Seu objetivo é medir o quanto o resultado obtido confirma uma hipótese pré-especificada. Nesse caso, são utilizados testes de hipótese para determinar se uma estrutura obtida é apropriada para os dados. O mesmo índice pode ser utilizado em um critério externo e interno, embora as distribuições de referência do índice sejam diferentes (Jain & Dubes, 1988). O que distingue a utilização de um índice em um dos critérios é a maneira como o índice é aplicado. A maneira mais comum de aplicação de um índice com um critério relativo, por exemplo, consiste do cálculo do seu valor para vários agrupamentos que estão sendo comparados, obtendo-se uma seqüência de valores.

O melhor agrupamento é determinado pelo valor que se destaca nessa seqüência, como o valor máximo, mínimo ou inflexão na curva do gráfico construído com a seqüência.

2.5 Interpretação

Esta etapa refere-se ao processo de examinar os clusters encontrados pelo algoritmo para tentar descobrir algum significado relacionado ao domínio da aplicação. O especialista pode avaliar subjetivamente os clusters encontrados para descobrir algum significado prático, e/ou diferenças, considerando os padrões representados em cada cluster.

A atribuição de conceitos aos agrupamentos é parte fundamental do processo de descoberta de conhecimento. Por outro lado, é uma tarefa complexa que exige a participação do especialista do domínio na análise dos agrupamentos para que ele identifique os conceitos que, eventualmente, cobrem ou explicam os exemplos contidos em um mesmo cluster. Para auxiliar o especialista na extração de conhecimento de clusters, é importante desenvolver ferramentas que facilitem essas tarefa. Em (Martins, 2003) é proposta uma metodologia que utiliza-se das técnicas de aprendizado supervisionado para guiar a tarefa de explicação de clusters. Essa metodologia será explorada em trabalhos futuros para analisar os agrupamentos encontrados por algoritmos de *clustering* hierárquico.

3 Ferramentas e Algoritmos de *Clustering* Hierárquico: Uma Visão geral

Um algoritmo de *clustering* hierárquico aglomerativo pode ser descrito basicamente da seguinte maneira:

1. Procure pelo par de clusters com a maior semelhança.
2. Crie um novo cluster que agrupe o par selecionado no passo 1.
3. Decremente em 1 o número de clusters restantes.
4. Volte ao passo 1 até que reste apenas um cluster.

O que diferencia os diversos algoritmos é o método ou estratégia utilizada para identificar os pares de clusters mais semelhantes. Para isso, diversas estratégias foram propostas, tais como: *Single Link*, *Complete Link*, *Average Link*, *Centroid-based*, *Median* e *Ward*, descritas a seguir.

***Single Link*:** este é um dos métodos de *clustering* hierárquico mais simples, também conhecido como técnica dos vizinhos mais próximos — (*Nearest Neighbor Technique*). Foi descrito pela primeira vez por Florek et al. (1957) e mais tarde por Sneath (1957) e Johnson (1967). Nesse método, a distância entre dois clusters é determinada pela

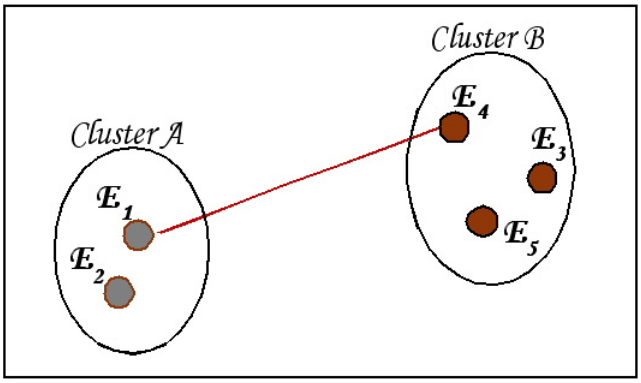


Figura 3.3: *Single Link*: menor distância entre dois clusters. Assim, $d_{A,B} = min\ dist(E_i, E_j); E_i \in A, E_j \in B$.

distância do par de exemplos mais próximo, sendo cada exemplo pertencente a um desses clusters, conforme ilustrado na Figura 3.3.

Alguns algoritmos que implementam essa estratégia são descritos por Sibson (1973) e Rohlf (1973, 1978).

Para entender melhor como funciona essa estratégia, considere um grafo, no qual os exemplos representam os nós e as arestas formam um caminho entre os nós contidos num mesmo subgrafo (cluster). A junção ou agrupamento de dois subgrafos corresponde à adição de uma aresta entre o par de nós mais próximo, sendo que cada nó pertence a um cluster diferente. Utilizando a terminologia da teoria dos grafos, esse procedimento, caso seja executado até que todos os subgrafos estejam conectados, resulta em uma árvore denominada *MST — Minimal Spanning Tree* (Figura 3.4).

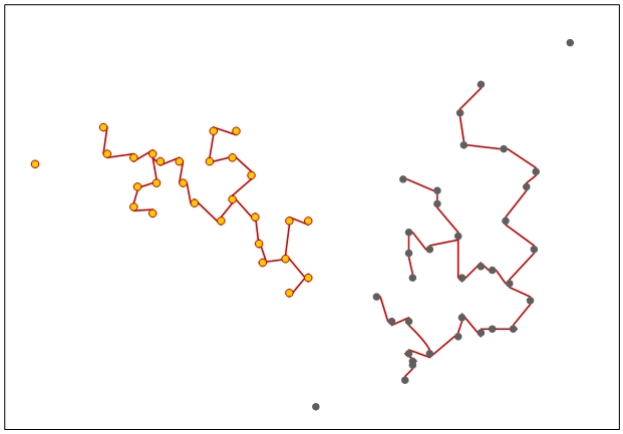


Figura 3.4: *Minimal Spanning Tree* obtida com a aplicação do método *Single Link*.

O método *Single Link* apresenta um problema, usualmente chamado de *chaining effect*, *i.e.*, algumas vezes ocorre a presença de alguns exemplos localizados entre dois cluster distintos, que formam uma ponte e força a junção indevida desses clusters.

***Complete Link*:** esta estratégia é freqüentemente referenciada como *Farthest Neighbour* ou vizinho mais distante. Ao contrário do método *Single Link*, essa estratégia determina a distância entre dois agrupamentos de acordo com a maior distância entre um

par de exemplos, sendo cada exemplo pertencente a um cluster distinto, conforme ilustrado na Figura 3.5. Com isso, tem maior propensão a identificar clusters menos alongados.

A fim de entender essa tendência, basta analisar o procedimento como a construção de um grafo no qual arestas conectam todos os nós dentro de um cluster, constituindo um subgrafo completo em cada cluster. Quando os pares de clusters mais próximo são agrupados, o grafo é modificado de modo que novas arestas são adicionadas entre cada par de nós nesses dois subgrafos, formando um único grafo maior com a junção de todos os nós e preservando a propriedade de que esse também é um subgrafo completo.

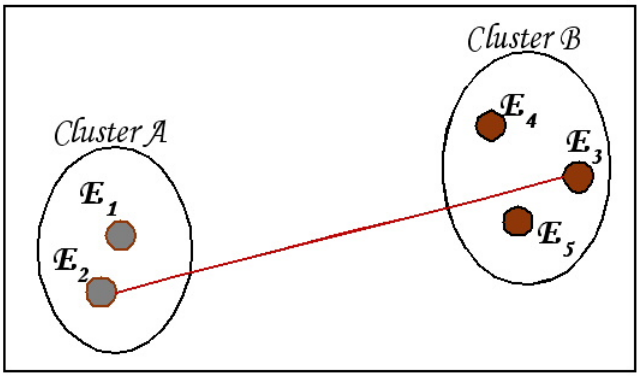


Figura 3.5: *Complete Link*: maior distância entre dois clusters. Assim, $d_{A,B} = max\ dist(E_i, E_j); E_i \in A, E_j \in B$.

***Average Link*:** as estratégias de distância mínima e máxima, *Single Link* e *Complete Link* respectivamente, representam dois extremos em termos de distância entre clusters. Como todos os procedimentos que envolvem esses extremos, eles tendem a ser altamente sensíveis à presença de *outliers*. Assim, o uso de uma abordagem intermediária é um caminho natural para amenizar esse problema (Duda et al., 2000).

No método *Average Link*, a distância entre dois clusters é definida como a média das distâncias entre todos os pares de exemplos em cada cluster, cada par composto por um exemplo de cada cluster, ilustrado na Figura 3.6.

***Centroid Based*:** os três métodos descritos anteriormente operam diretamente sobre a matriz de similaridade e não necessitam acesso direto aos valores dos atributos dos exemplos, enquanto que o método baseado no centróide deve ter acesso direto aos exemplos. Com o método *Centroid Based*, os grupos identificados são representados por um vetor que armazena a média de cada atributo. Assim, a distância entre os clusters é definida em termos da distância entre o representante de cada cluster. Conseqüentemente, os atributos devem ser numéricos.

Uma desvantagem desse método é que se o tamanho dos clusters selecionados para a junção forem muito diferentes, então, o centróide do novo cluster será muito se-

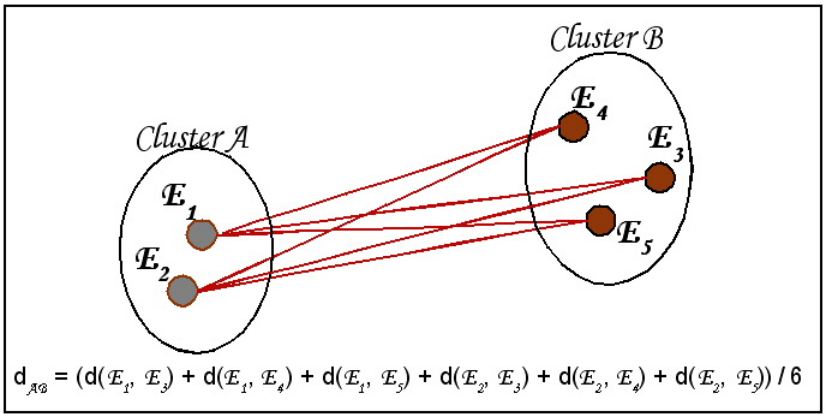


Figura 3.6: *Average Link*: média das distâncias entre os dois clusters.

melhante ao centróide do cluster de maior tamanho, o que acarretará na perda das características do menor cluster.

***Ward*:** Ward (1963) propôs um procedimento para o *clustering* hierárquico de modo que as partições formadas miminizem a perda associada a cada agrupamento, a qual deve ser facilmente quantizada e de simples interpretação. Nesse procedimento, a cada passo, todas as possíves uniões entre pares de clusters são consideradas e os clusters que apresentam a mínima perda de informação são selecionados para o agrupamento. Essa perda de informação é definida em termos da soma dos erros quadrados.

A tarefa de *clustering* é altamente relacionada aos parâmetros, métricas e métodos utilizados pelo algoritmo. Em essência, algoritmos diferente que utilizam a mesma métrica e método de agrupamento deveriam apresentar os mesmos resultados se aplicado sobre um mesmo conjunto de dados. Essa observação é válida se considerarmos que a cada iteração os mesmos pares de clusters são selecionados para o agrupamento. No entanto, os algoritmos mais freqüentemente utilizados em aplicações práticas utilizam outros recursos, os quais possibilitam melhoras significativas e, em alguns casos, melhoram a adequação do modelo obtido ao problema em questão.

Esses recursos vão desde técnicas para redução do conjunto de dados (seleção aleatória e particionamento), estruturas de dados para otimizar o cálculo da similaridade entre agrupamentos (árvores métricas como R-Tree (Guttman, 1984) e Kd-Tree (Bentley, 1975)), conceitos da teoria de grafos e diversas outras estratégias que acarretam na diferenciação dos resultados gerados por esses algoritmos.

A seguir são apresentados brevemente os seguintes algoritmos de *clustering* hierárquico: *AGNES — Agglomerative Nesting* (Kaufman & Rousseeuw, 1990a), *BIRCH — Balanced Iterative Reducing and Clustering Using Hierarchies* (Zhang et al., 1996), *CURE — Clustering Using Representatives* (Guha et al., 1998), *CHAMELEON — A Hierarchical Clustering Algorithm Using Dynamic Modeling* (Karypis et al., 1999), *DIANA — Divisive Analysis* (Kaufman & Rousseeuw, 1990b), *ROCK — Robust Clustering Using*

Links (Guha et al., 2000) e *MONA — Monothetic Analysis* (Kaufman & Rousseeuw, 1990c). Além desses algoritmos, são apresentadas brevemente algumas características da ferramenta Explorer e da biblioteca Cluster3 (Hoon et al., 2004).

3.1 *AGNES: Agglomerative Nesting*

O *AGNES*, proposto em (Kaufman & Rousseeuw, 1990a), é um dos algoritmos de *clustering* hierárquico mais tradicionais. Esse algoritmo foi projetado de modo que o usuário não precisa indicar o número de clusters desejados, pois como todo algoritmo hierárquico, o *AGNES* constrói uma representação da hierarquia de agrupamento (dendograma), a partir da qual é possível derivar todos os possíveis conjuntos de clusters gerados pelo algoritmo.

As entradas desse algoritmo podem ser tanto na forma de matriz de similaridade como na forma de tabela de exemplos. A implementação do *AGNES* disponível no ambiente R[1], dispõe de duas métricas de similaridade, as quais podem ser aplicadas somente sobre atributos numéricos: a métrica Euclidiana e a métrica de Manhattan. É possivel também realizar o *clustering* considerando a relação dos atributos, ao invés de exemplos. Mas para isso, é necessário que seja calculada a matriz de similaridade contendo o coeficiente de similaridade entre os atributos, que podem ser por exemplo o coeficiente de correlação paramétrico ou não-paramétrico.

Esse algoritmo, assim como os outros algoritmos aglomerativos, procede com uma série de sucessivas fusões entre clusters. Para isso, o par de clusters mais semelhantes é identificado a partir da matriz de similaridade e selecionado para o agrupamento. Quando mais de um par é identificado como mais semelhante, a seleção aleatória entre esses pares é utilizada para determinar qual deles será agrupado na iteração corrente.

No *AGNES* estão disponíveis cinco método para o cálculo da similaridade inter-cluster: *Single Link*, *Complete Link*, *Ward*, *UPGMA — Unweighted Pair Group Method with Arithmetic Mean*, *Weighted Average* e uma flexibilização desse último baseada na fórmula de atualização de *Lance-Willians* (Lance & Willams, 1967).

3.2 *BIRCH: Balanced Iterative Reducing and Clustering Using Hierarchies*

Este algoritmo surgiu como uma proposta para o *clustering* de grandes conjuntos de dados que encontram-se armazenados em memória externa. Para isso, são utilizadas algumas estratégias que minimizam o número de acesso ao disco e mantêm as informações necessárias para a realização do agrupamento em memória principal. A idéia básica desse algoritmo é comprimir os exemplos de dados em subclusters para que o agrupamento possa ser realizado em memória principal. Com isso, o algoritmo necessita de apenas uma varredura no conjunto de dados.

[1] http://www.r-project.org

Apesar dos bons resultados que vem apresentando, esse algoritmo possui algumas deficiências. Sua performance é prejudicada quando os agrupamentos não possuem tamanhos e formatos uniformes. Além disso, é adequado apenas para conjuntos de dados cujos atributos são numéricos, ou seja, os exemplos podem ser representados explicitamente por coordenadas no espaço Euclidiano. Por outro lado, uma contribuição muito importante do *BIRCH* é a formulação do problema de *clustering* de modo apropriado para conjuntos de dados muito grandes, considerando as restrições de tempo e memória. O *BIRCH* considera, também, o fato de que os dados usualmente não são uniformemente distribuídos no espaço, conseqüentemente nem todos os pontos são igualmente importantes para o *clustering*. Desse modo, as regiões mais densas são tratadas coletivamente por meio de um resumo dos dados, reduzindo o problema de agrupar uma grande quantidade de exemplos para um problema menor, no qual apenas o resumo dos dados (muito menor que o conjunto todo) é submetido ao *clustering*.

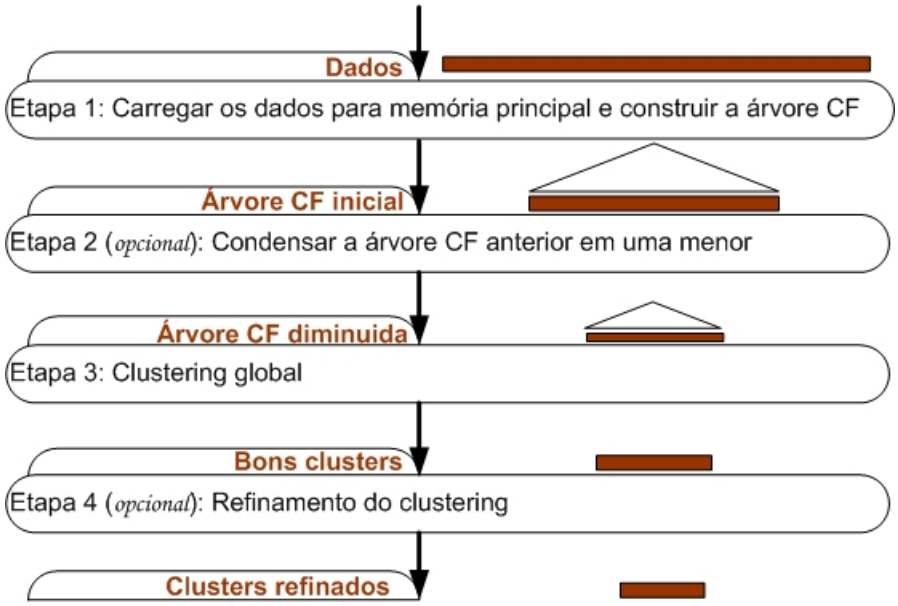


Figura 3.7: Etapas do algoritmo *BIRCH*.

A tarefa realizada pelo algoritmo *BIRCH* é composta por quatro etapas, ilustradas na Figura 3.7. O objetivo da primeira etapa é ler todo o conjunto de dados e construir uma árvore em memória principal. Essa árvore descreve o conjunto de dados da maneira mais detalhada possível de acordo com as limitações de espaço de memória, armazenando em seus nós apenas um resumo de cada cluster, que contém informações como o número de exemplos presentes no cluster (N_{Cl}), um vetor com a soma linear dos atributos de cada exemplo local ($\overrightarrow{LS}$) e a soma quadrada desses exemplos (SS). Na segunda etapa, a árvore CF é analisada e reestruturada para ocupar menor espaço de memória. No processo de reestruturação da árvore, alguns *outliers* são eliminados e alguns sub-clusters menores são agrupados em sub-clusters mais gerais, fazendo com que a árvore resultante seja menor. A diminuição da árvore é realizada com o objetivo otimizar a execução do algoritmo

de agrupamento na terceira etapa. Na proposta original do *BIRCH*, foi utilizada uma adaptação do algoritmo de *clustering* hierárquico aglomerativo descrito em (Olson, 1995). A quarta etapa consiste na atribuição dos exemplos aos clusters encontrados na terceira etapa. Para isso, é analisada a similaridade entre os exemplos e os clusters, e os exemplos são atribuídos ao cluster mais semelhante. Assim, um novo conjunto de clusters é gerado e apresentado como resultado final do processo.

3.3 *CURE: Clustering Using Representatives*

Diversos algoritmos representam os clusters por meio de um ponto central, usualmente o centróide, dividindo os dados em clusters de acordo com a menor distância em relação ao representante de cada cluster. Um problema com essa abordagem ocorre quando clusters distintos estão muito próximos, apresentam formato não elipsoidal ou tamanhos não uniformes. Esses fatores, em geral, prejudicam dramaticamente o resultado desses algoritmos. Por outro lado, há algoritmos que representam os clusters como um grafo, no qual cada exemplo refere-se a um nó e a similaridade entre os nós é indicada pelo peso nas arestas do grafo. Nesse caso, o par de clusters a ser agrupado em cada iteração do algoritmo é selecionado de acordo com alguma medida de similaridade inter-cluster (método de agrupamento), por exemplo *Single Link*. Essa abordagem também apresenta alguns problemas, pois poucos pontos localizados entre dois clusters distintos podem criar um efeito de ponte e determinar o agrupamento indevido desses clusters. Dessa maneira, observa-se que as abordagens baseadas no centróide e as abordagens baseadas em grafo, não produzem bons resultados quando os clusters não estão bem separados ou os tamanhos não são uniformes. A maior deficiência dos algoritmos baseados no centróide é a divisão de clusters com formatos não esféricos ou alongados, enquanto que os algoritmos baseados em grafo são altamente sensíveis à presença de *outliers* (Guha et al., 1998).

Com o objetivo de solucionar esses problemas, o *CURE* utiliza uma abordagem intermediária, na qual cada cluster é representado por um número p de pontos. Inicialmente são selecionados p exemplos bem distribuídos dentro do cluster de modo que esses exemplos descrevam naturalmente o formato do cluster. Em seguida, esses exemplos são aproximados do centro do cluster por meio de um fator de encolhimento. Após a aplicação do fator de encolhimento, os p exemplos selecionados são definidos como representantes do cluster. Assim, a distância entre dois clusters será a distância entre o par de representantes mais próximos (*Single Link*), sendo cada representante pertencente a um cluster.

O *CURE* também é indicado para o *clustering* de grandes conjuntos de dados, mas utiliza uma estratégia diferente da adotada pelo algoritmo *BIRCH*. Ao invés de pré-clusterizar todo o conjunto de dados, o *CURE* inicia selecionando aleatoriamente uma amostra dos dados e, em seguida, particiona essa amostra em subconjuntos e realiza um *clustering* parcial em cada partição desses subconjuntos de dados. Uma vez que as partições estejam pré-clusterizadas, o *CURE* utiliza os múltiplos representantes de cada cluster para definir o agrupamento dos dados restantes. As etapas do algoritmo *CURE*

são apresentadas na Figura 3.8.

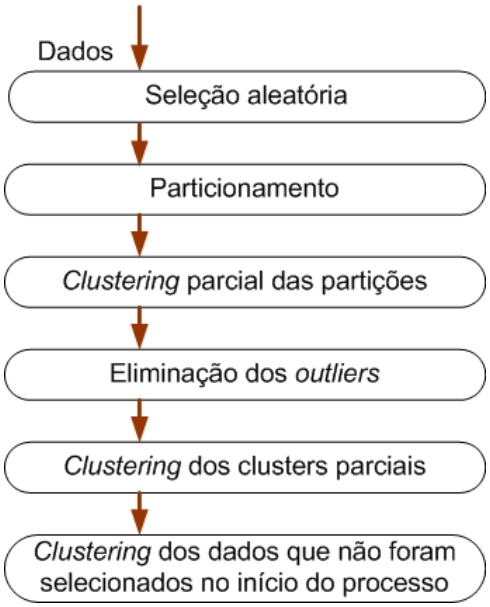


Figura 3.8: Etapas do algoritmo *CURE*.

O algoritmo faz uso de duas estruturas de dados: uma *Heap* e uma árvore multidimensional (*Kd-Tree*). Os elementos da *Heap* são ordenados de acordo com a menor distância entre dois clusters. Assim, cada entrada na *Heap* corresponde a um cluster e a primeira posição indica o par de clusters mais semelhante para o agrupamento. A árvore multidimensional, por sua vez, é utilizada para armazenar os representantes dos clusters, pois possibilita a identificação eficiente dos pares de clusters mais semelhantes durante o processo de agrupamento.

3.4 *CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling*

Diversos algoritmos de *clustering* amplamente utilizados, tais como *BIRCH* e *CURE*, são projetados para encontrar agrupamentos que seguem algum modelo estático. Esses algoritmos falham na identificação dos clusters se a escolha dos parâmetros desse modelo estático não for apropriada considerando as características do conjunto de dados e, eventualmente, eles podem também falhar se o modelo propriamente dito não é adequado para capturar as características dos clusters presentes nos dados. Adicionalmente, alguns algoritmos podem não gerar bons resultados quando os dados contém clusters de formatos arbitrários, tamanhos não uniformes e densidades variadas. Por outro lado, o algoritmo *CHAMELEON* mede a similaridade entre dois clusters baseado em modelos dinâmicos. Nesse caso, dois clusters são agrupados somente se a inter-conectividade e a proximidade entre os dois clusters são comparáveis à inter-conectividade interna dos clusters e à proximidade entre os exemplos dentro dos clusters, isso facilita a identificação natural e homogênea dos clusters contidos nos dados (Karypis et al., 1999). Dessa maneira, o

CHAMELEON supera as limitações dos modelos estáticos discutidos anteriormente, pois analisa características internas dos clusters no momento do agrupamento.

Esse algoritmo atua sobre um grafo esparso baseado na idéia de k-vizinhos mais próximos (k-*Nearest Neighbor* — KNN). Cada vértice do grafo de k-vizinhos mais próximos representa um exemplo do conjunto de dados. Existe uma aresta entre dois vértices se o exemplo correspondente a um dos dois vértices é um dos k exemplos mais similares do outro vértice. O *CHAMELEON* realiza o processo de *clustering* em duas etapas. Na primeira etapa, um algoritmo de particionamento de grafos esparsos é aplicado sobre o grafo que representa o conjunto de dados para agrupar os exemplos em um número grande de sub-clusters relativamente pequenos. Na segunda etapa, utiliza um algoritmo de *clustering* hierárquico aglomerativo para encontrar os clusters finais. Uma visão geral do processo de *clustering* executado pelo *CHAMELEON* é apresentada na Figura 3.9.

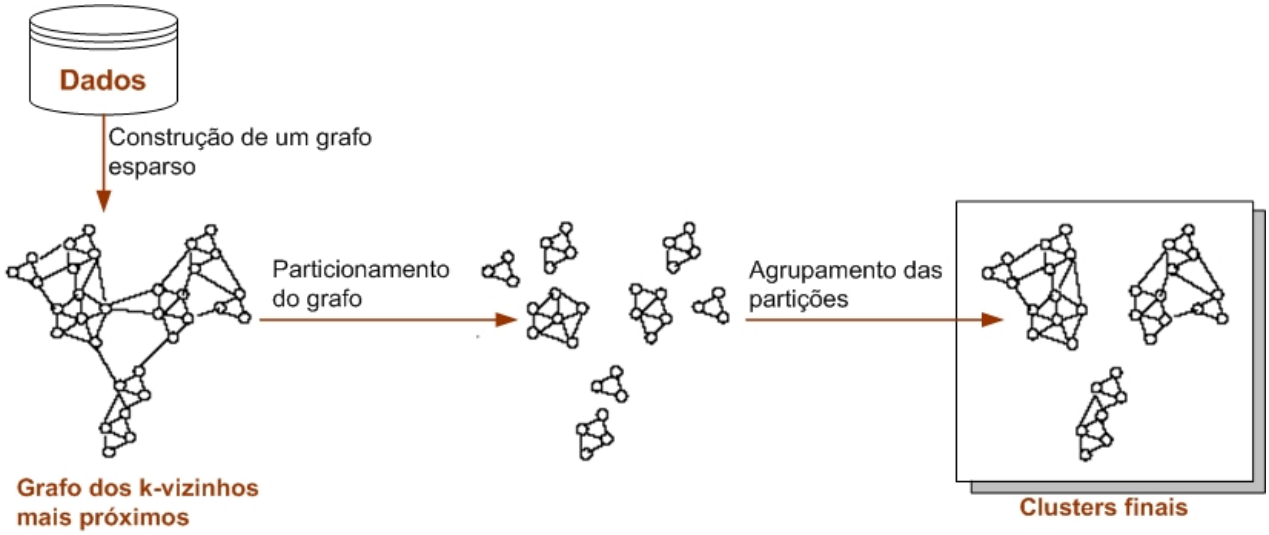


Figura 3.9: Etapas do algoritmo *CHAMELEON*.

Essa estratégia de modelo dinâmico para cálculo da similaridade entre clusters pode ser aplicada somente quando cada cluster contém um número de elementos suficientemente grande, pois tanto a inter-conectividade relativa como a proximidade relativa entre dois clusters não podem ser calculadas corretamente utilizando poucos elementos. Esse é o motivo pelo qual o *CHAMELEON* é executado em duas etapas. Na primeira etapa são construídos sub-clusters com um número suficiente de elementos para a correta aplicação da modelagem dinâmica. A segunda etapa é aplicada então para encontrar os clusters finais por meio da aplicação desse *framework* de modelagem dinâmica em conjunto com um algoritmo de *clustering* hierárquico aglomerativo. Há uma implementação desse algoritmo disponível para uso e de domínio público. Essa implementação faz parte do *Cluto — Clustering Toolkit*, e pode ser encontrado no endereço eletrônico `http://www-users.cs.umn.edu/~karypis/cluto/`.

3.5 *ROCK: Robust Clustering Using Links*

A análise do resultado do *clustering* hierárquico em um determinado nível da hierarquia acarreta no particionamento do conjunto de dados inicial em um conjunto de clusters que satisfazem alguma função ou critério para o agrupamento. Grande parte das funções critério, em geral, produzem resultados satisfatórios quando aplicadas em dados numéricos, mas para dados com atributos categóricos não são apropriadas.

O algoritmo *ROCK* (Guha et al., 2000) sugere uma alternativa à esse problema, um novo conceito de *clustering* baseado em ligações entre exemplos, ao invés de métricas de distância, como a Euclidiana, por exemplo. Nesse contexto, o conjunto de dados é analisado como um grafo esparso, onde exemplos são os nós do grafo e a aresta entre dois exemplos indica que fazem parte de um mesmo sub-grafo (cluster nesse caso). É importante observar que, nesse caso, ligações e arestas são conceitos distintos. O número de ligações entre um par de exemplos E_i e E_j, é o número de vizinhos comuns desses exemplos, conforme ilustrado na Figura 3.10.

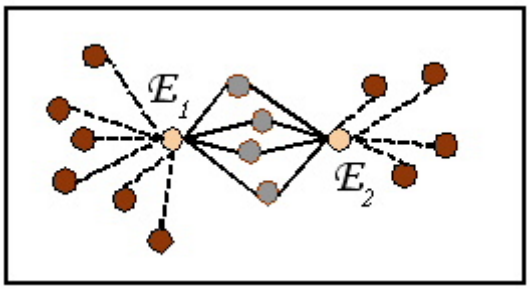


Figura 3.10: Ilustração do conceito de ligações utilizado pelo algoritmo *ROCK*. Assim, $Link(E_i, E_j) = 4$.

Partindo desse princípio, o algoritmo analisa o grau de conectividade dos clusters e seleciona o par que maximiza a soma das ligações. Uma visão geral do algoritmo *ROCK* é apresentada na Figura 3.11, na qual são mostradas três etapas envolvidas no processo de *clustering*. Inicialmente, uma amostra significativa de exemplos é selecionada aleatoriamente a partir dos dados originais. Após, o processo de agrupamento hierárquico que usa o conceito de ligações é aplicado sobre a amostra de exemplos selecionado. Esse processo termina quando não há mais pares de clusters com ligações ou quando o número de clusters desejados é obtido. Finalmente, os clusters identificados são utilizados para determinar o grupo ao qual pertence cada exemplo ainda não utilizado, os quais são atribuídos ao cluster que maximiza o número de ligações.

Além do *ROCK*, existem outros algoritmos que tratam de dados categóricos. O *QROCK* (Dutta, Mahanta, & Pujari, Dutta et al.) por exemplo, é um algoritmo essencialmente parecido com o *ROCK*, mas utiliza técnicas computacionais para garantir uma melhor performance em relação ao tempo de execução e recursos de *hardware* utilizados.

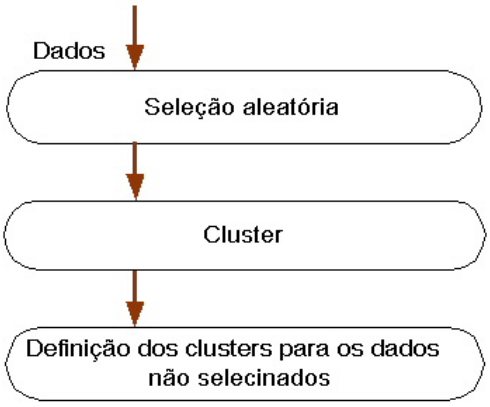


Figura 3.11: Etapas do algoritmo *ROCK*.

3.6 *MONA: Monothetic Analysis*

Os algoritmos apresentados anteriormente foram desenvolvidos para operar sobre conjuntos de dados numéricos ou categóricos. Entretanto, muitas vezes os dados não seguem esses padrões. Na área de ciências biológicas, por exemplo, freqüentemente são disponibilizados conjuntos de dados com atributos binários, os quais podem assumir apenas dois valores que representam presença ou ausência de determinada característica. Nesse contexto, algumas alternativas foram propostas, como transformar os valores binários em valores numéricos (zero e um) para que possam ser utilizados os algoritmos descritos anteriormente, ou ainda utilizar medidas de similaridade que possam ser aplicadas sobre dados binários.

O algoritmo *MONA* adota uma abordagem diferente. A idéia é selecionar um dos atributos e dividir o conjunto de dados em relação a esse atributo. Assim, são obtidos dois clusters, um com exemplos que possuem valor 1 para o atributo selecionado e outro cluster com exemplos que possuem valor 0 para esse atributo. Em cada cluster gerado, um dos atributos restantes é selecionado e usado para uma nova divisão. Esse processo é repetido até que os clusters restantes sejam unitário ou não haja mais atributos que possam separar os dados em cada cluster. Um atributo utilizado em alguma divisão anterior não pode ser selecionado novamente, pois nesse estágio todos os exemplos pertencentes ao mesmo cluster apresentam o mesmo valor para o atributo que já foi utilizado. Claramente, esse algoritmo segue a abordagem de *clustering* hierárquico divisivo. O fator mais importante desse algoritmo é a seleção do atributo para separação dos dados. A idéia é escolher o atributo que maximiza a soma da similaridade com todos os outros atributos, ou seja, o atributo que está mais centralizado no espaço dos exemplos analisados.

3.7 *DIANA: Divisive Analysis*

A abordagem divisiva é hierárquica por natureza. A cada passo, algoritmos divisivos particionam um cluster formando dois novos clusters menores, até que todos os clusters restantes não possam mais ser divididos.

O algoritmo *DIANA*, proposto por Kaufman & Rousseeuw (1990b), pode operar sobre conjuntos de dados que seguem exatamente a mesma estrutura daqueles utilizados pelo algoritmo *AGNES*. Entretanto, o algoritmo considera que inicialmente todos os exemplos fazem parte de um único cluster e, então, divide esse cluster em dois clusters menores. Nessa divisão não são consideradas todas as possibilidades, mas é utilizado um procedimento iterativo que otimiza a escolha dos exemplos que serão atribuídos a cada novo cluster. Nesse procedimento, o exemplo menos semelhante a todos os outros é selecionado e utilizado para a criação de um novo cluster. A seleção aleatória é utilizada caso mais de um exemplo esteja apto para ser selecionado. Na seqüência são selecionados outros exemplos que são mais semelhantes ao novo cluster que ao cluster inicial, e esses exemplos são transladados para o novo cluster. Esse processo se repete dividindo a cada iteração o cluster que contém o maior valor de diâmetro, até que restem apenas clusters unitários.

3.8 *HCE: Hierarchical Clustering EXPLORER*

O HCE (Seo, 2005) é uma ferramenta de visualização para exploração interativa de conjuntos de dados multi-dimensionais, que permite aos usuários um melhor entendimento dos dados e suas estruturas latentes, uma vez que a inspeção visual pode ser útil para processos de extração de padrões em conjuntos de dados não-supervisionados.

Usualmente, os algoritmos de *clustering* disponíveis para uso requerem que o número de clusters finais seja informado. Outros algoritmos implementam estratégias que identificam automaticamente o número de clusters supostamente contidos nos dados. Entretanto, esse resultado pode não ser interessante ao usuário, uma vez que ele tem pouco ou nenhum controle sobre o processo. Desse modo, o HCE obtém uma representação completa da hierarquia de agrupamento e permite ao usuário determinar o número natural de clusters com o auxílio de ferramentas visuais e interativas.

As métricas de similaridade disponíveis são o coeficiente de correlação de Pearson, distância Euclidiana e distância de Manhattan. Os métodos de agrupamento são: *UPGMA — Unweighted Pair Group Method with Arithmetic Mean*, *Centroid*, *Complete Link*, *Single Link* e *Shneiderman*.

Essa ferramenta pode ser obtida através do endereço eletrônico `http://www.cs.umd.edu/hcil/hce`.

3.9 A Biblioteca Cluster 3.0

Vários dos algoritmos descritos nas seções anteriores estão disponíveis par uso em pacotes de *software* estatísticos, como o R[2], o qual dispõe da implementação dos algoritmos *AGNES*, *DIANA* e *MONA*, ou em ferramentas de domínio público, como o Cluto (Karypis, 2003), que implementa o algoritmo *CHAMELEON*. Por outro lado, a implementação de alguns desses algoritmos não está disponível, como é o caso dos algoritmos *CURE*, *BIRCH*

[2] `http://www.r-project.org`

e *ROCK*. É possível encontrar alguma implementação desses algoritmos, mas não a implementação original, *i.e.*, desenvolvida pelo autor projetista do algoritmo. A utilização de algoritmos implementados por outros pesquisadores, que não os idealizadores, pode acarretar em resultados diferentes dos encontrados pelo código original, pois a implementação está sujeita à algumas decisões de projeto que, em geral, não são vistas da mesma maneira por pessoas diferentes.

Outra dificuldade encontrada é o difícil acesso ao código fonte desses algoritmos. É interessante que o usuário desenvolvedor tenha controle total sobre os algoritmos estudados, de modo que possa ter acesso às estruturas de dados utilizadas na implementação, às representações internas dos objetos e às representações dos resultados, assim como às heurísticas utilizadas pelo implementador do algoritmo. Sob esse prisma, alguns pesquisadores disponibilizam seus algoritmos e ferramenta com o código fonte em forma de bibliotecas ou módulos. Um exemplo é a biblioteca Cluster, inicialmente proposta por Michael Eisen e implementada por Hoon et al. (2004). Essa biblioteca, assim como a sua documentação (de Hoon et al., 2005), estão disponivel através do endereço eletrônico `http://bonsai.ims.u-tokyo.ac.jp/~mdehoon/software/cluster`. Além das bibliotecas escritas em linguagem C, estão disponíveis módulos Perl e Python que fazem uso dos arquivos de cabeçalho C.

A biblioteca Cluster, consiste de uma coleção de rotinas que podem ser utilizadas para análise de expressões gênicas e dados de *microarray*. Nessa biblioteca estão disponíveis rotinas para o *clustering* hierárquico, particionamento e redes auto-organizáveis. As rotinas de *clustering* hierárquico implementam os métodos *Single Link*, *Complete Link*, *Average Link* e *Centroid*, apresentados anteriormente. As métricas de similaridade intra-cluster disponíveis são: Euclidiana, Manhattan, Correlação, Correlação Absoluta, Correlação Não-Centralizada, Correlação Não-Centralizada Absoluta, Correlação de Speraman e Kendall τ (de Hoon et al., 2005).

4 Representação do *Clustering*

Como mencionado anteriormente, os objetivos deste trabalho são estudar as formas de representação do *clustering* hierárquico e as estruturas de dados utilizadas para a implementação das mesmas e, então, a partir desse estudo, indicar a melhor representação a ser utilizada na implementação do módulo de *clustering* hierárquico do ambiente Discover. Tendo em vista esses objetivos, a seguir são descritas algumas das representações mais usualmente utilizadas nessa tarefa de aprendizado, discutindo pontos positivos e negativos de cada representação e, também, identificando os algoritmos que as utilizam.

As técnicas de aprendizado de máquina são utilizadas freqüentemente com objetivo de descobrir informações que possam ser úteis no processo de Descoberta de Conhecimento de Bases de Dados (*Knowledge Discovery from Databases — KDD*). Para isso, os resultados obtidos devem estar representados por meio de uma linguagem de fácil entendimento para

simplificar a análise e compreensão dessas informações pelos usuários.

No aprendizado não-supervisionado, mais especificamente o *clustering* hierárquico, foco do nosso trabalho, os resultados são representados simbolicamente por alguma estrutura que permite observar a seqüência com que os agrupamentos foram formados. Diversas representações podem ser utilizadas, mas o aspecto fundamental em cada uma delas é que seja possível identificar a hierarquia dos agrupamentos, em quais clusters estão distribuídos os exemplos, em que iteração do processo um determinado exemplo foi agrupado e qual o grau de similaridade que resultou no agrupamento. Outras informações também são desejáveis, tais como a média de similaridade interna e externa de cada cluster, a densidade dos agrupamentos e uma medida de qualidade global do *clustering*. Algumas das representações mais usualmente utilizadas são apresentadas a seguir.

O métodos de *clustering* hierárquico, tanto divisivos como aglomerativos, constroem uma estrutura que descreve uma hierarquia de agrupamentos sobre dados. Um algoritmo hierárquico é a especificação de ações que resultam nessa estrutura, que por sua vez é chamada de *clustering*.

Uma maneira simples de representar o *clustering* é utilizar a notação de conjuntos. Dados os N exemplos inicias do conjunto $E = \{E_1, E_2, ..., E_N\}$, então uma partição P sobre o conjunto de dados inicial divide E em sub-conjuntos $P = \{P_1, P_2, ..., P_k\}$ tal que:

$$P_i \cap P_j = \emptyset, \quad \forall(i,j),\ i = 1,2,...k;\ j = 1,2,...k\ e\ i \neq j$$
$$E = P_1 \cup P_2 \cup P_3 \cup ... \cup P_k$$

Seguindo essa notação, uma partição P' está aninhada a uma partição P se todo componente de P' é um conjunto próprio de algum componente em P, *i.e.*, P é formado pelo agrupamento dos componentes de P'. Por exemplo, se o *clustering* P com três clusters e o *clustering* P' com cinco clusters são definidos da seguinte maneira:

$$P = \{\{E_1, E_2, E_3\}, \{E_4, E_5, E_6, E_7\}, \{E_8\}\}$$
$$P' = \{\{E_1\}, \{E_2, E_3\}, \{E_4, E_5\}, \{E_6, E_7\}, \{E_8\}\}$$

então P' está inserido em P e ambos representam o *clustering* sobre o conjunto de dados E. Desse modo, a hierarquia de agrupamento completa sobre esses dados poderia ser:

$\{E_1, E_2, E_3, E_4, E_5, E_6, E_7, E_8\}$

$\{E_1, E_2, E_3, E_4, E_5, \{E_6, E_7\}, E_8\}$

$\{E_1, \{E_2, E_3\}, E_4, E_5, \{E_6, E_7\}, E_8\}$

$\{E_1, \{E_2, E_3\}, \{E_4, E_5\}, \{E_6, E_7\}, E_8\}$

$\{E_1, \{E_2, E_3\}, \{E_4, E_5, E_6, E_7\}, E_8\}$

$\{\{E_1, E_2, E_3\}, \{E_4, E_5, E_6, E_7\}, E_8\}$

$\{\{E_1, E_2, E_3\}, \{E_4, E_5, E_6, E_7, E_8\}\}$

$\{\{E_1, E_2, E_3, E_4, E_5, E_6, E_7, E_8\}\}$

Ainda que essa representação textual apresente a estrutura hierárquica dos clusters, ela não possibilita identificar o grau de similaridade dos agrupamentos. Além disso, não é simples analisar os resultados quando o conjunto de dados contém muitos exemplos. Para tentar facilitar a interpretação do *clustering*, pode ser utilizada outra representação, também baseada em conjuntos, na qual cada conjunto é formado de outros conjuntos menores (Figura 4.12). Essa representação, conhecida como Diagrama de Venn, revela a estrutura hierárquica e facilita a interpretação de dados bi-dimensionais, mas também não apresenta a similaridade dos clusters e não pode ser diretamente utilizada para conjuntos de dados multi-dimensionais.

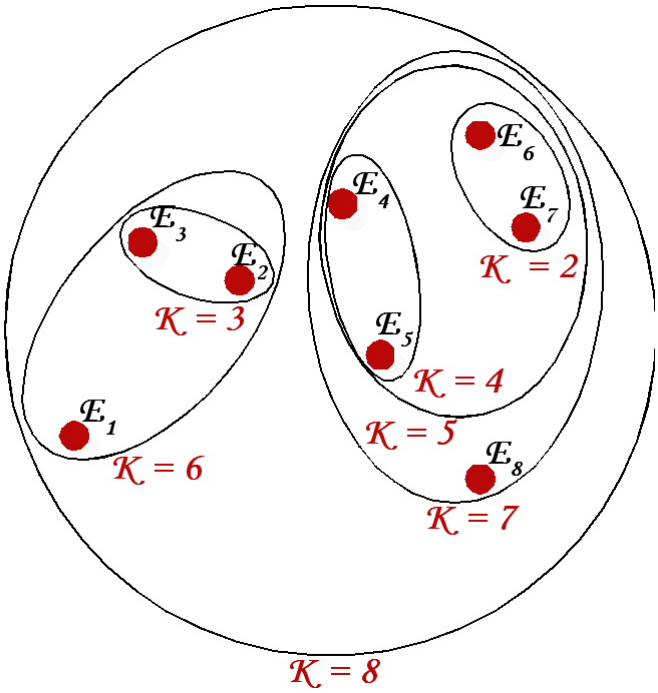
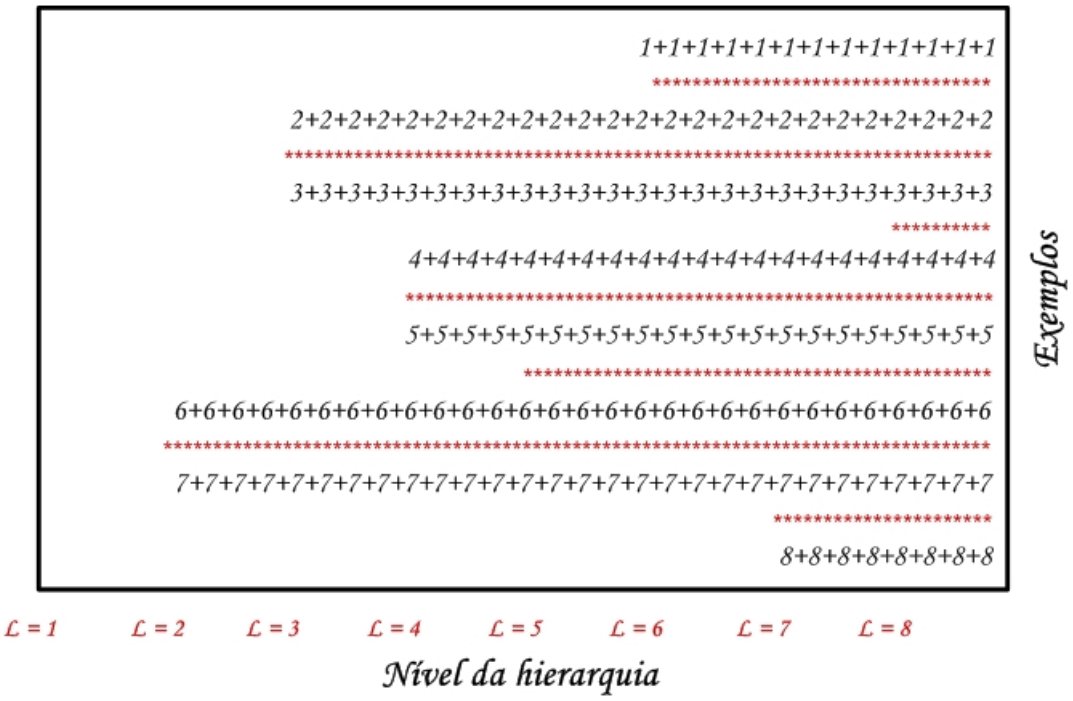


Figura 4.12: *Clustering* representado por meio do Diagrama de Venn.

Por esse motivo, outras representações são mais utilizadas, tais como *Banner* e o Dendograma.

O *Banner*, utilizado nos algoritmos *AGNES*, *DIANA* e *MONA*, consiste de asteriscos (*) e listas de identificadores — Figura 4.13 na próxima página. Os asteriscos indicam ligações entre objetos (clusters ou exemplos) e as listas de identificadores contêm repetições do rótulo desses objetos. Os espaços em branco à esquerda na Figura 4.13 indicam o estágio original onde cada cluster unitário ainda não foi agrupado. Então, no nível dois inicia uma seqüência de três linhas. A primeira composta por 6+6+6+... que é simplesmente a repetição do rótulo do exemplo E_6 separado pelo caractere +. Da mesma maneira, a lista de caracteres composta por 7+7+7+... na terceira linha denota o exemplo E_7. A segunda linha composta por uma seqüência de asteriscos indica que houve um agrupamento entre os exemplos E_6 e E_7 a partir daquele nível. Os próximos dois agrupamentos ocorrem com os exemplos E_2 e E_3 no terceiro nível e com os exemplos E_4 e E_5 no quarto nível, e assim sucessivamente até que todos estejam agrupados num único cluster. Observe que o

Banner deve ser lido sempre da esquerda para a direita.


1+1+1+1+1+1+1+1+1+1+1+1+1
2+2+2+2+2+2+2+2+2+2+2+2+2+2+2+2+2+2+2+2+2+2+2+2+2
3+3+3+3+3+3+3+3+3+3+3+3+3+3+3+3+3+3+3+3+3+3+3+3+3+3+3
4+4+4+4+4+4+4+4+4+4+4+4+4+4+4+4+4+4+4+4+4+4+4+4+4+4+4+4+4
5+5+5+5+5+5+5+5+5+5+5+5+5+5+5+5+5+5+5+5+5+5+5+5+5+5+5+5+5+5+5
6+6+6+

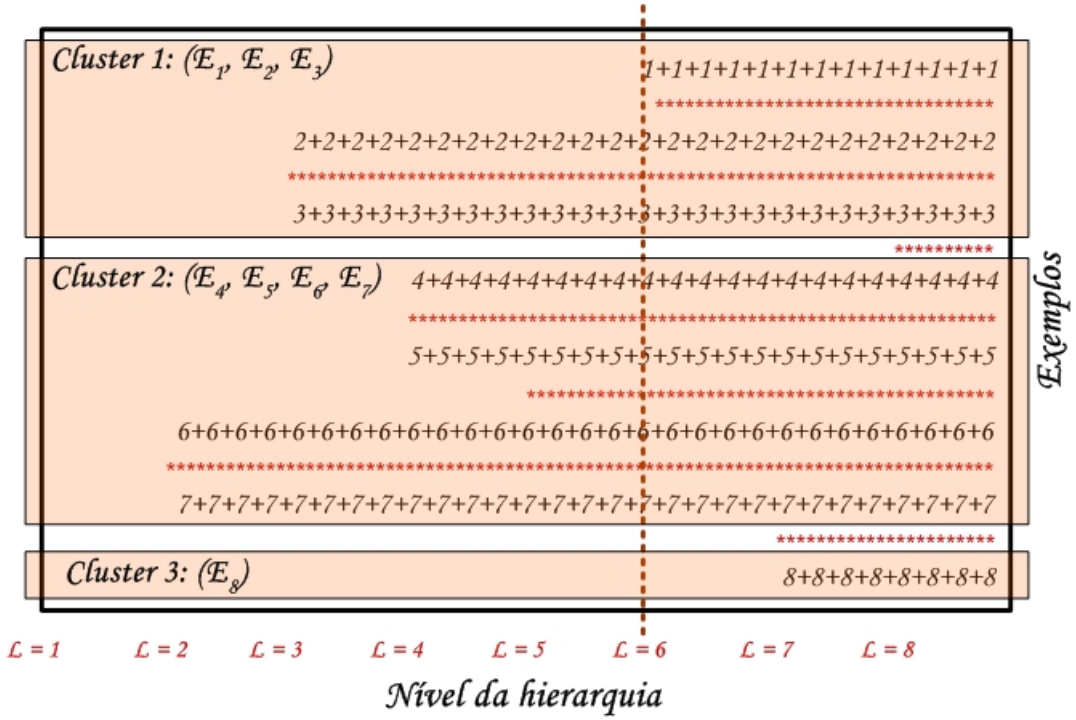


Figura 4.14: Identificação dos clusters a partir de um *Banner*.

se dois exemplos pertencem ao mesmo cluster em algum nível, eles continuarão fazendo parte do mesmo cluster nos níveis superiores. No nível 2, os exemplos E_6 e E_7 foram agrupados, e permanecem no mesmo cluster para todos os níveis subseqüentes. O eixo vertical do dendograma representa a medida de similaridade entre os clusters, que é utilizada pelo usuário para interpretar o *clustering* e determinar se ocorreu um agrupamento natural ou forçado.

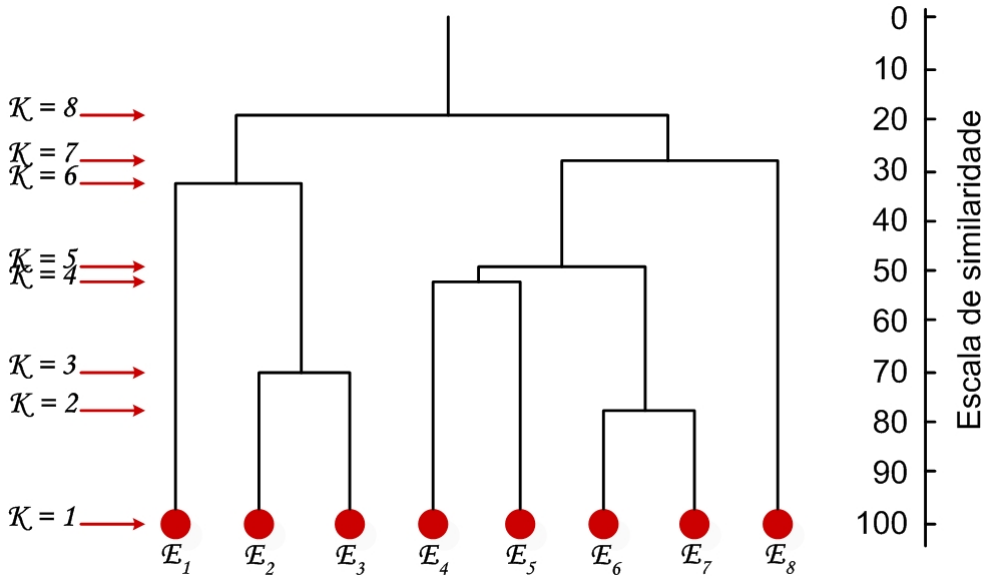


Figura 4.15: *Clustering* representado por meio do Dendograma.

É importante observar que dendogramas gerados a partir de métricas de similaridade e métodos de agrupamento diferentes, devem ser analisados de maneira diferente. Um

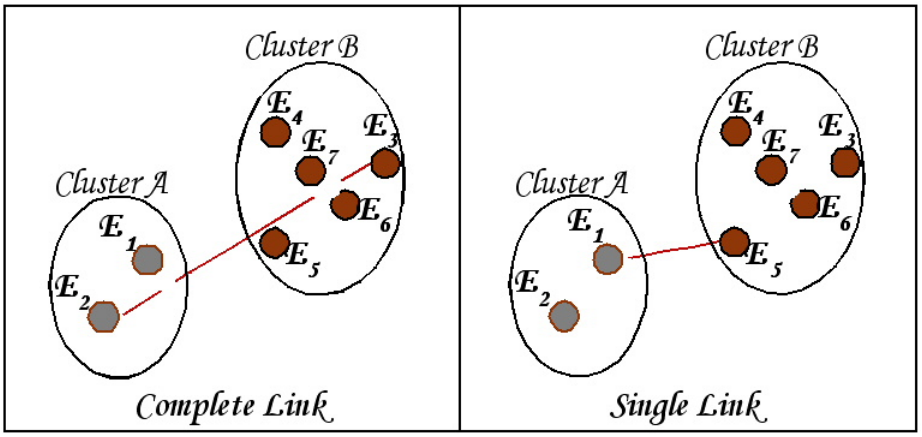
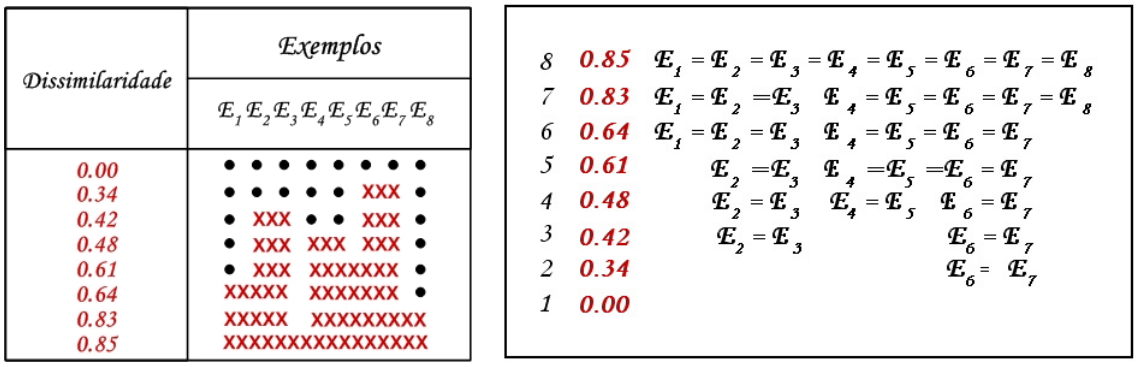


Figura 4.16: Similaridade intra-cluster: *Complete Link* e *Single Link*.



(a) (b)

Figura 4.17: Representações do dendograma: (a) Diagrama de Johnson, (b) IciclePlot.

dendograma obtido a partir da execução de um algoritmo que utiliza o método *Complete Link*, por exemplo, deve apresentar valores de similaridade baixo entre clusters muito grandes, ao passo que um dendograma gerado por meio do método *Single Link* apresenta valores de similaridade maiores para os mesmos clusters — Figura 4.16.

Existem outras representações para o *clustering*, tais como *Johnson-type*, *IciclePlot* e *Ward-type*, apresentadas respectivamente nas Figuras 4.17(a), 4.17(b) e 4.18.

Para implementação dessas representações, podem ser utilizadas diversas estruturas de dados. Uma maneira bastante simples de implementar o dendograma é utilizar uma matriz bi-dimensional com N linhas e N colunas, ilustrada na Figura 4.19 na página oposta para $N = 8$. Nessa matriz são armazenados os rótulos dos exemplos ou clusters de acordo com a seqüência de agrupamento. Cada linha representa um nível da hierarquia e as colunas referenciam os exemplos. Na linha N estão armazenados os clusters unitários, que correspondem aos exemplos individuais. Na linha $N - 1$ ocorre uma mudança em virtude do agrupamento entre os elementos E_6 e E_7, os quais passam a constituir um novo cluster, C_1, que é representado por meio de uma alteração na linha $N - 1$. As colunas referentes aos exemplos que formaram o novo cluster passam a armazenar o rótulo do cluster e não mais o rótulo do exemplo. De maneira análoga, na linha $N - 2$ as colunas

Exemplos	Número de clusters							
	1	2	3	4	5	6	7	8
E_1	E_1	E_1	E_1	E_1	E_1	E_1	E_1	E_1
E_2	E_2	E_2	E_2	E_2	E_2	E_2	E_2	E_2
E_3	E_3	E_3	E_3	E_3	E_3	E_3	E_3	E_3
E_4	E_4	E_4	E_4	E_4	E_4	E_4	E_4	E_4
E_5	E_5	E_5	E_5	E_5	E_5	E_5	E_5	E_5
E_6	E_6	E_6	E_6	E_6	E_6	E_6	E_6	E_6
E_7	E_7	E_7	E_7	E_7	E_7	E_7	E_7	E_7
E_8	E_8	E_8	E_8	E_8	E_8	E_8	E_8	E_8
Dissimilaridade	0.85	0.83	0.64	0.61	0.48	0.42	0.34	0.00

Figura 4.18: Diagrama de Ward.

que referenciam os exemplos E_2 e E_3 passam a armazenar o rótulo do segundo cluster formado, C_2. Esse esquema de representação dos agrupamentos se repete para cada linha da matriz, construindo assim uma hierarquia. A partir dessa estrutura é relativamente simples construir um *Banner* ou desenhar o dendograma. Entretanto, essa estrutura não armazena o grau de similaridade com que os agrupamentos foram realizados. Para isso, é necessário utilizar um vetor auxiliar com $N-1$ posições — Figura 4.19b — , as quais armazenam a similaridade com que os clusters foram formados em cada iteração do processo. Assim, a posição $N-1$ desse vetor armazena a similaridade que acarretou no agrupamento representado na linha $N-1$ da matriz.

C_7	C_7	C_7	C_7	C_7	C_7	C_7	C_7
C_5	C_5	C_5	C_6	C_6	C_6	C_6	C_6
C_5	C_5	C_5	C_4	C_4	C_4	C_4	E_8
E_1	C_2	C_2	C_4	C_4	C_4	C_4	E_8
E_1	C_2	C_2	C_3	C_3	C_1	C_1	E_8
E_1	C_2	C_2	E_4	E_5	C_1	C_1	E_8
E_1	E_2	E_3	E_4	E_5	C_1	C_1	E_8
E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8

(a)

0.85
0.83
0.61
0.64
0.42
0.48
0.34
0.00

(b)

Figura 4.19: Dendograma representado por meio de uma matriz quadrada.

Há um problema com a utilização dessa estrutura para implementação do dendograma, sua complexidade de espaço é quadrática, $\Theta(N^2)$. Além disso, é extremamente difícil analisar a hierarquia do agrupamento com muitos exemplos a partir dessa estrutura. Por outro lado, a facilidade com que é possível identificar os clusters ao qual cada exemplo pertence é uma vantagem dessa estrutura.

Outra estrutura de dados, conhecida como *Parent Array* (Karypis, 2003), também utiliza vetores para armazenar a hierarquia de agrupamento dos dados. Porém, nesse caso o espaço necessário é de complexidade linear em relação ao número de exemplos, mais especificamente, se o conjunto de dados possui N exemplos, o espaço necessário para armazenar essa estrutura é de $2 \times (2N - 1)$, ou seja $\Theta(N)$.

Essa representação utiliza uma matriz de $2N - 1$ linhas e duas colunas. Cada linha armazena a informação de um agrupamento, de modo que a i-ésima linha da primeira coluna contém o rótulo do cluster pai que agrupa o cluster i. Quando i é menor que N, então o cluster i é unitário (constituído de apenas um exemplo). A i-ésima linha da segunda coluna armazena a similaridade que resultou no agrupamento do cluster i. Para identificar a raiz da árvore, que é sempre armazenada na última linha do vetor, basta utilizar um *flag* indicando que não há um cluster pai. Isso é feito por meio da atribuição do valor -1 na primeira coluna da última linha da matriz. Por exemplo, o *Parent Array* ilustrado na Figura 4.20 para o dendograma apresentado na Figura 4.15, contém 15 linhas e armazena a seguinte seqüência: 13, 10, 10, 11, 11, 9, 9, 14, 12, 13, 12, 14, 15, 15, -1. Essa seqüência indica que os exemplos E_6 e E_7, linhas 6 e 7 respectivamente, foram agrupados e passaram a formar o cluster 9. Os exemplos E_2 e E_3 (linhas 2 e 3) formaram o cluster 10. Analisando os níveis superiores, pode-se observar que os clusters 9 e 10 foram agrupados pelo cluster 12, pois o rótulo 12 está inserido nas linhas 9 e 10. Os outros agrupamentos são representados da maneira análoga.

	Nº da linha	ID	Sim
Exemplos do conjunto inicial	1	13	0.36
	2	10	0.58
	3	10	0.58
	4	11	0.52
	5	11	0.52
	6	9	0.66
	7	9	0.66
	8	14	0.17
Clusters	9	12	0.39
	10	13	0.36
	11	12	0.39
	12	14	0.17
	13	15	0.15
	14	15	0.15
	15	-1	

Figura 4.20: *Parent Array* para o conjunto de dados $E = \{E_1, E_2, E_3, E_4, E_5, E_6, E_7, E_8\}$.

Uma estrutura ainda mais econômica, chamada de *Pointer Representation* (Sibson, 1973), também possui complexidade de espaço $\Theta(N)$, mas as constantes multiplicadoras são menores, resultando em uma matriz de $(N-1)$ linhas e 3 colunas, apresentada na Figura 4.21.

Nº da linha	ID_1	ID_2	Sim
1	6	7	0.66
2	2	3	0.58
3	4	5	0.52
4	-1	-3	0.39
5	-2	1	0.36
6	-4	8	0.17
7	-5	-6	0.15

Figura 4.21: *Poiter Representation* para o *clustering* sobre o conjunto de dados $E = \{E_1, E_2, E_3, E_4, E_5, E_6, E_7, E_8\}$.

Nessa estrutura, as linhas também representam os eventos de agrupamento. A primeira e a segunda coluna armazenam os clusters que foram agrupados enquanto que a terceira coluna armazena a similaridade que resultou nesse agrupamento. Os clusters unitários, *i.e.*, os que contêm somente um exemplo, são identificados com rótulos de $1..N$, ao passo que os novos clusters são identificados com rótulos negativos $(-1..-(N-1))$. A seqüência dos agrupamentos corresponde à seqüência com que os clusters são armazenados no vetor, ou seja, a linha um do vetor indica o primeiro agrupamento, a linha dois o segundo e assim sucessivamente.

As duas últimas estruturas, apesar de serem mais econômicas em relação ao espaço necessário para armazenar o dendograma, acarretam num *overhead* maior para identificar quais exemplos pertencem a cada cluster num determinado nível da hierarquia, pois essa informação não está explicitamente presente nessas estruturas. Por outro lado, em termos de espaço, são muito mais econômicas que a primeira. Considere por exemplo um conjunto de dados com 100 exemplos. A primeira estrutura (matriz quadrada), necessita de 10.000 posições de memória ao passo que a estrutura de *Parent Array* e *Poiter Representation* necessitam de 398 e 297, respectivamente. Desse modo, a escolha da estrutura para implementação do *clustering* deve considerar os aspectos de complexidade, tanto de tempo quanto de espaço. Nesse contexto, para o desenvolvimento do módulo de *clusteting* hierárquico do ambiente Discover, foi decidido utilizar a representação dos clusters por meio do dendograma implementado a partir da estrutura de dados *Pointer Representation*, devido a sua flexibilidade e baixa complexidade de espaço. É certo que a complexidade de tempo para identificar os elementos que constituem determinado cluster, nessa estrutura, é maior que na estrutura de matriz, mas ainda assim está na ordem de $O(N \log h)$, onde N é o número de exemplos e h é a altura da árvore (dendograma), portanto totalmente factível.

Foi também decidido que a implementação da estrutura *Pointer Representation* no ambiente DISCOVER sofrerá uma modificação. Ao invés de representar os clusters com rótulos negativos, eles serão identificados com rótulos positivos variando de $N + 1$ a $2 \times (N - 1)$.

O procedimento `Cuttree` para identificar os exemplos presentes em cada cluster é apresentado em pseudo-código a seguir — Algoritmo 1 — , onde as variáveis `tree`, `k` e `nclusters` são os parâmetros de entrada e as variáveis `clusters` e `clustering` são os parâmetros de saída.

Algorithm 1: `CutTree`

```
   Input: tree, k, nclusters
   Output: clustering, clusters
 1 push(clusters, treeChildren(root)) ;
 2 k -= 2;
 3 while k > 2 do
 4     if max(clusters) > nclusters then
 5         temp = extracMax(clusters);
 6         push(clusters, treeChildren(temp)) ;
 7         k--;
 8     end
 9 end
10 foreach index do clustering[index] = index;
11 foreach index do
12     if not singlecluster(index, tree) then
13         push(subclusters, treeChildren(index));
14         foreach element do
15             if singlecluster(element, tree) then
16                 clustering[element] = index;
17             else
18                 push(subclusters, treeChildren(element));
19             end
20         end
21     end
22 end
```

A variável `tree` é um objeto do tipo *Pointer Representation*; `clusters` e `clustering` são vetores utilizados para guardar as informações do *clustering* após o corte no dendograma. Assim, `clusters` armazena os clusters identificados e a variável `clustering` armazena a solução final, onde cada posição do vetor representa o respectivo exemplo do conjunto de dados original e armazena o número do cluster ao qual pertence.

No procedimento `CutTree`, inicialmente os dois nós filhos da raiz principal do dendograma são armazenados na variável `clusters`. Após isso, é executado um laço (linhas 3 a 9) para identificar os outros clusters filhos até que os primeiros `k` clusters estejam armazenados na variável `clusters`. Na linha 10, é executado um novo laço no qual a variável `clustering` é inicializada, atribuindo o rótulo de cada exemplo à sua respectiva posição,

i.e., posição 0 do vetor `clustering` recebe o valor 0, posição 1 do vetor `clustering` recebe o valor 1 e assim sucessivamente. O terceiro laço (linhas 11 a 22) é responsável por identificar quais exemplos estão inseridos em cada um dos clusters armazenados na variável `clusters` e atribuir o rótulo do cluster às posições dos seus respectivos exemplos na variável `clustering`. Desse modo, a solução final do processo é armazenada na variável `clustering`, onde cada posição i contém o rótulo do cluster ao qual o exemplo i pertence.

5 Considerações Finais

Neste trabalho foram apresentados brevemente alguns conceitos relacionados ao aprendizado de máquina não-supervisionado, especificamente a tarefa de *clustering*. Os algoritmos utilizados na realização dessa tarefa são freqüentemente diferenciados de acordo com as diversas abordagens, tais como particionais, baseados em grade, baseados em densidade, com sobreposição, redes auto-organizáveis, conceituais e hierárquicos. Cada abordagem pode representar os agrupamentos de diversas formas, e cada representação pode ainda ser implementada de diferentes maneiras a partir de diferentes estruturas de dados. Assim, não há uma sintaxe simples e eficiente para representar o agrupamento encontrado por todas as abordagens de *clustering*. Entretanto, é possível determinar um padrão para os algoritmos de uma mesma abordagem, por exemplo os algoritmos hierárquicos. Nessa abordagem, as representações mais utilizadas são o *Banner* e o Dendograma, os quais podem ser obtidos a partir da implementação das estruturas de *Parent Array* ou *Pointer Representation*.

Essas estruturas são muito econômicas em relação ao espaço utilizado para armazenar os agrupamentos e também são eficientes em relação às operações de manipulação. Tendo em vista essas características, neste trabalho optou-se por utilizar a estrutura *Pointer Representation* para a implementação do módulo de *clustering* hierárquico do ambiente Discover. Esse módulo será desenvolvido à partir da biblioteca Cluster 3.0, apresentada anteriormente, à qual serão agregadas algumas funcionalidades para auxiliar na análise dos agrupamentos encontrados. A descrição do projeto e implementação do módulo de *clustering* hierárquico do Discover, será publicada em outro Relatório Técnico do ICMC.

Referências

Agrawal, R., J. Gehrke, D. Gunopulos, & P. Raghavan (1998). Automatic subspace clustering of high dimensional data for data mining applications. In *SIGMOD '98: Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pp. 94–105. ACM Press. 6

Ankerst, M., M. M. Breunig, H.-P. Kriegel, & J. Sander (1999). Optics: Ordering points to identify the clustering structure. In A. Delis, C. Faloutsos, & S. Ghandeharizadeh (Eds.), *Sigmod 1999, proceedings ACM Sigmod International Conference on Management of Data, june 1-3, 1999, Philadephia, Pennsylvania, USA*, pp. 49–60. ACM Press. 7, 9

Baranauskas, J. A. & G. E. A. P. A. Batista (2000). O projeto Discover: idéias iniciais. Comunicação pessoal. 2

Batista, G. E. A. P. A. & M. C. Monard (2003). Descrição da arquitetura e do projeto do ambiente computacional DLE - Discover Learning Environment. Technical Report 187, ICMC-USP. http://www.icmc.usp.br/~biblio/index.php?destino=relatorios_tecnicos. 2

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Commun. ACM 18*(9), 509–517. 14

Berkhin, P. (2002). Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA. http://citeseer.nj.nec.com/berkhin02survey.html. 6

Bhatia, S. & J. Deogun (1998). Conceptual clustering in information retrieval. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 28*(3), 427–436. http://citeseer.ist.psu.edu/bhatia98conceptual.html. 8

Cheeseman, P. & J. Stutz (1996). Bayesian classification (AutoClass): theory and results. *Advances in knowledge discovery and data mining*, 153–180. http://portal.acm.org/citation.cfm?id=257954. 1, 7

de Hoon, M., S. Imoto, & S. Miyano (2005). *The C Clustering Library*. Tokyo: The University of Tokyo, Institute of Medical Science, Human Genome Center. http://bonsai.ims.u-tokyo.ac.jp/~mdehoon/software/cluster. 23

Duda, R. O., P. E. Hart, & D. G. Stork (2000). *Pattern Classification* (2 ed.). New York: Wiley-Interscience. 8, 13

Dutta, M., A. K. Mahanta, & A. K. Pujari. Qrock: A quick version of the rock algorithm for clustering of categorical data. http://citeseer.ist.psu.edu/562762.html. 20

Ester, M., H.-P. Kriegel, J. Sander, & X. Xu (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In E. Simoudis, J. Han, & U. Fayyad (Eds.), *Second International Conference on Knowledge Discovery and Data Mining*, Portland, Oregon, pp. 226–231. AAAI Press. http://citeseer.ist.psu.edu/chu02incremental.html. 7

Everitt, B. S. (1993). *Cluster Analysis* (3 ed.). Edward Arnold. 4, 6

Fisher, D. & P. Langley (1986). Conceptual clustering and its relation to numerical taxonomy. pp. 77–116. 8

Florek, K., J. Lukaszewiez, J. Perkal, H. Steinhaus, & S. Zubrzchi (1957). Sur la liason et la division des points d'un ensemble fini. *Colloquium Mathematicum* (2), 282–285. 11

Guha, S., R. Rastogi, & K. Shim (1998). CURE: an efficient clustering algorithm for large databases. In *ACM Sigmod International Conference on Management of Data*, pp. 73–84. http://citeseer.nj.nec.com/article/guha98cure.html. 14, 17

Guha, S., R. Rastogi, & K. Shim (2000). ROCK: A robust clustering algorithm for categorical attributes. *Information Systems 25*(5), 345–366. http://citeseer.nj.nec.com/guha00rock.html. 15, 20

Guttman, A. (1984). R-trees: a dynamic index structure for spatial searching. In *Sigmod'84: Proceedings of the 1984 ACM Sigmod International Conference on Management of Data*, Boston, Massachusetts, pp. 47–57. ACM Press. 14

Halkidi, M., Y. Batistakis, & M. Vazirgiannis (2002a). Cluster validity methods: Part I. *SIGMOD Rec. 31*(2), 40–45. 10

Halkidi, M., Y. Batistakis, & M. Vazirgiannis (2002b). Cluster validity methods: Part II. *SIGMOD Rec. 31*(3), 19–27. 10

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. Prentice Hall. 7

Herrero, J., A. Valencia, & J. Dopazo (2001). A hierarchical unsupervised growing neural network for clustering gene expression patterns. *Bioinformatics* (17), 126–136. 7

Hinneburg, A. & D. A. Keim (1998). An efficient approach to clustering in large multimedia databases with noise. In *Knowledge Discovery and Data Mining*, pp. 58–65. http://citeseer.ist.psu.edu/hinneburg98efficient.html. 7

Hinneburg, A. & D. A. Keim (1999). Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering. In *The VLDB Journal*, pp. 506–517. http://citeseer.ist.psu.edu/hinneburg99optimal.html. 6

Hoon, M. J. L., S. Imoto, J. Nolan, & S. Miyano (2004). Open source clustering software. *Bioinformatics 20*(2), 1453–1454. 15, 23

Jain, A. K. & R. C. Dubes (1988). *Algorithms for Clustering Data*. Prentice Hall. 2, 3, 6, 10

Jain, A. K., M. N. Murty, & P. J. Flynn (1999). Data clustering: a review. *ACM Computing Surveys 31*(3), 264–323. http://citeseer.nj.nec.com/jain99data.html. 4, 5, 6

Johnson, S. C. (1967). Hierarchical clustering schemes. *Psychometrika* (32). 11

Karypis, G. (2003). *CLUTO: a clustering toolkit*. Minneapolis: University of Minnesota, Department of Computer Science. http://www-users.cs.umn.edu/~karypis/cluto/download.html. 22, 30

Karypis, G., E.-H. S. Han, & V. Kumar (1999). CHAMELEON: Hierarchical clustering using dynamic modeling. *Computer 32*(8), 68–75. http://citeseer.nj.nec.com/karypis99chameleon.html. 14, 18

Kaufman, L. & P. J. Rousseeuw (1990a). *Agglometarive nesting (program AGNES)* (1 ed.), Volume 1 of *1*, Chapter 5, pp. 199–252. New York: Wiley Inter-Science. 14, 15

Kaufman, L. & P. J. Rousseeuw (1990b). *Divisive analysis (program DIANA)* (1 ed.), Volume 1 of *1*, Chapter 6, pp. 253–279. New York: Wiley Inter-Science. 9, 14, 22

Kaufman, L. & P. J. Rousseeuw (1990c). *Monothetic analysis (program MONA)* (1 ed.), Volume 1 of *1*, Chapter 7, pp. 280–311. New York: Wiley Inter-Science. 15

Kohonen, T. (1990). Self-organized formation of topologically correct feature maps. In J. W. Shavlik & T. G. Dietterich (Eds.), *Readings in Machine Learning*, pp. 326–336. San Matheo: Kaufmann. 7

Kohonen, T., J. Hynninen, J. Kangas, & J. Laaksonen (1996). Sompak: the self-organizing map program package. Technical report, Helsinki University of Technology, Laboratory of Computer and Information Science. 7

Lance, G. & W. Willams (1967). A general theory of classification sorting strategies. *Computer Journal 1*(9), 373–380. 15

Liu, H. & H. Motoda (1998). *Feature Selection for Knowledge Discovery and Data Mining*. Boston: Kluwer Academic. 5

Martins, C. A. (2003). *Uma Abordagem para Pré-processamento de Dados Textuais em Algoritmos de Aprendizado*. Tese de Doutorado, Instituto de Matemática e Computação, Universidade de São Paulo, Brasil. http://www.teses.usp.br/teses/disponiveis/55/55134/tde-08032004-164855/. 11

Michalski, R. S. & R. E. Stepp (1983). Learning from observation: Conceptual clustering. *Machine Learning: An Artificial Intelligence Approach 1*, 331–363. 8

Mishra, N., D. Ron, & R. Swaminathan (2004). A new conceptual clustering framework. *Machine Learning 56*(1-3), 115–151. 8

Murtagh, F. (1983). A survey of recent advances in hierarchical clustering algorithms. *The Computer Journal 26*(40), 354–359. 9

Nagesh, H., S. Goil, & A. Choudhary (1999). Mafia: Efficient and scalable subspace clustering for very large data sets. Technical Report 9906-010, Northwestern University. 6

Olson, C. F. (1995). Parallel algorithms for hierarchical clustering. *Parallel Computing 21*(8), 1313–1325. http://citeseer.nj.nec.com/olson95parallel.html. 17

Rohlf, F. J. (1973). Hierachical clustering using minimal spannign tree. *The Computer Journal* (16), 93–95. 12

Rohlf, M. E. (1978). A probabilistic minimum spanning tree algorithm. *Inform. Process. Lett* (8), 44–49. 12

Seo, J. (2005). *Hierarchical Clustering Explorer 3.0: User's guide*. University of Maryland, College Park. http://www.cs.umd.edu/hcil/hce/. 22

Sheikholeslami, G., S. Chatterjee, & A. Zhang (1998). WaveCluster: A multi-resolution clustering approach for very large spatial databases. In *Proc. 24th Int. Conf. Very Large Data Bases, VLDB*, pp. 428–439. http://citeseer.ist.psu.edu/sheikholeslami98wavecluster.html. 7

Sibson, R. (1973). Slink: An optimally efficient algorithm for the single-link cluster method. *The Computer Journal* (16), 30–34. 12, 31

Sneath, P. H. A. (1957). The applications of computers to taxonomy. *Gen. Microbiol* (17), 201–226. 11

Stepp, R. E. & R. S. Michalski (1986). Conceptual clustering: Inventing goal oriented classifications of structured objects. *Machine Learning: An artificial intelligence approach 2*, 471–498. 8

Wang, W., J. Yang, & R. R. Muntz (1997). STING: A statistical information grid approach to spatial data mining. In M. Jarke, M. J. Carey, K. R. Dittrich, F. H. Lochovsky, P. Loucopoulos, & M. A. Jeusfeld (Eds.), *Twenty-Third International Conference on Very Large Data Bases*, Athens, Greece, pp. 186–195. Morgan Kaufmann. http://citeseer.ist.psu.edu/wang97sting.html. 6

Ward, J. H. (1963). Hierarchical grouping to optimize an objective funtion. *Journal of American Statistics Association* (58), 236–244. 14

Zhang, T., R. Ramakrishnan, & M. Livny (1996). BIRCH: an efficient data clustering method for very large databases. pp. 103–114. http://citeseer.nj.nec.com/zhang96birch.html. 14