INTRODUÇÃO À MINERAÇÃO DE DADOS COM APLICAÇÕES EM CIÊNCIAS AMBIENTAIS E ESPACIAIS

Rafael Santos

RESUMO

Mineração de dados (ou *Data Mining*) é o nome dado ao conjunto de técnicas que permite a extração de conhecimentos a partir de grandes volumes de dados. Neste mini-curso veremos alguns exemplos e técnicas de mineração de dados aplicadas a casos reais, com ênfase nos diversos algoritmos clássicos, suas características e limitações. O curso também discutirá a aplicabilidade de mineração de dados em diversos cenários, assim como técnicas auxiliares.

1. Introdução

Estamos nos afogando em informação mas com sede de conhecimento - John Naisbitt.

Avanços recentes em várias áreas tecnológicas possibilitaram um crescimento explosivo na capacidade de gerar, coletar e armazenar dados. Na primeira década do século 21 já temos a possibilidade de armazenar vários gigabytes em dispositivos portáteis e alguns terabytes em computadores pessoais a um custo acessível.

Dados coletados podem ser científicos (medidos por sensores como satélites, sondas, etc., com enorme precisão espacial e/ou temporal), sociais (censos, pesquisas, estudos de comportamento *on-line*), econômicos e comerciais (compras, transações bancárias, ligações telefônicas, acessos a servidores, etc) ou de outros tipos. Alguns exemplos de esforços de coleta de dados são apresentados a seguir (com alguma ênfase em exemplos de cunho científico):

- O SLAC (*Stanford Linear Accelerator Center*) gerará 200 terabytes/ano, a uma taxa de 10 megabytes/segundo por 10 anos [1]. 2 petabytes podem ser armazenados em uma pilha de DVDs de 4.4km de altura.
- O *Digital Palomar Observatory Sky Survey* (POSS-II) contém 3 terabytes de imagens astronômicas, com estimados 2 bilhões de objetos [2]. O conjunto *Data Release 6* do estudo SDSS (*Sloan Digital Sky Survey*) contém, entre imagens e atributos, mais de 10 terabytes.
- O Instituto Nacional de Pesquisas Espaciais (INPE) possui uma base de imagens de satélites com mais de 130 terabytes [3].
- O site *Wayback machine* (http://www.archive.org/about/faqs.php) armazena versões de sites na Internet, contendo 1 petabyte de dados e crescendo 20 terabytes ao mês (em um total de 55 bilhões de páginas em março de 2006).
- Algumas empresas mantém grandes bases de dados relacionadas às suas áreas de atuação [4]: AT&T (93 terabytes), Amazon (24 terabytes), Verizon Communications (7 terabytes), Yahoo! (100 terabytes), United States Patent and Trademark Office (16 terabytes).

O problema com a coleta automatizada de dados é que os dados são, em sua grande maioria, armazenados como registros simples, sem processamento para extração de conhecimento útil, e este conhecimento é necessário para fazer inferências, obter conclusões, comprovar hipóteses, etc. O problema que enfrentamos é, então, converter dados brutos facilmente coletáveis para informações (dados contextualizados) que podem por sua vez ser transformadas em conhecimento (resumos, descrições, regras, classes, grupos, etc.) O conhecimento sobre fenômenos relativos a estes dados permite análises mais complexas, sínteses, previsões, etc.

A Figura 1 ilustra o processo de transformação de dados em conhecimento, o que pode ser feito através de regras e procedimentos, aumentando o conteúdo semântico dos dados. A Figura também indica o foco deste mini-curso: a descoberta de conhecimento.

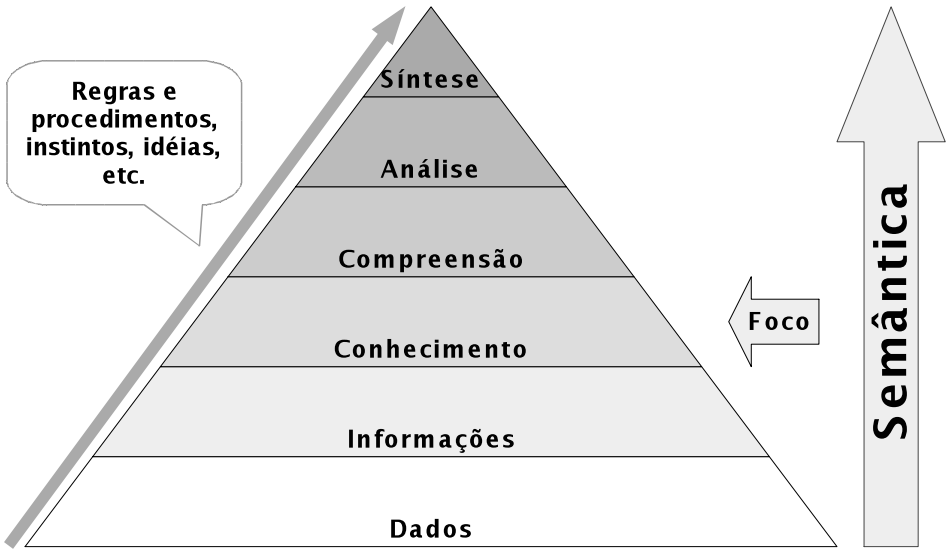

Fig. 1 – Dados, Informações e Conhecimento

O processo de transformação de dados em informações e conhecimentos é conhecido como descoberta de conhecimento em bases de dados, descrito com mais detalhes na seção seguinte.

2. Descoberta de Conhecimento em Bases de Dados

A definição mais usada do processo de descoberta de conhecimento em bases de dados (abreviada como KDD, de *knowledge discovery in databases*) é "processo geral de descoberta de conhecimentos úteis previamente desconhecidos a partir de grandes bancos de dados" (adaptada de [2]). Este processo é executado em várias etapas interdependentes, que podem ser repetidas e que nem sempre tem distinções clara entre si. Algumas etapas do processo são mostradas na Figura 2.

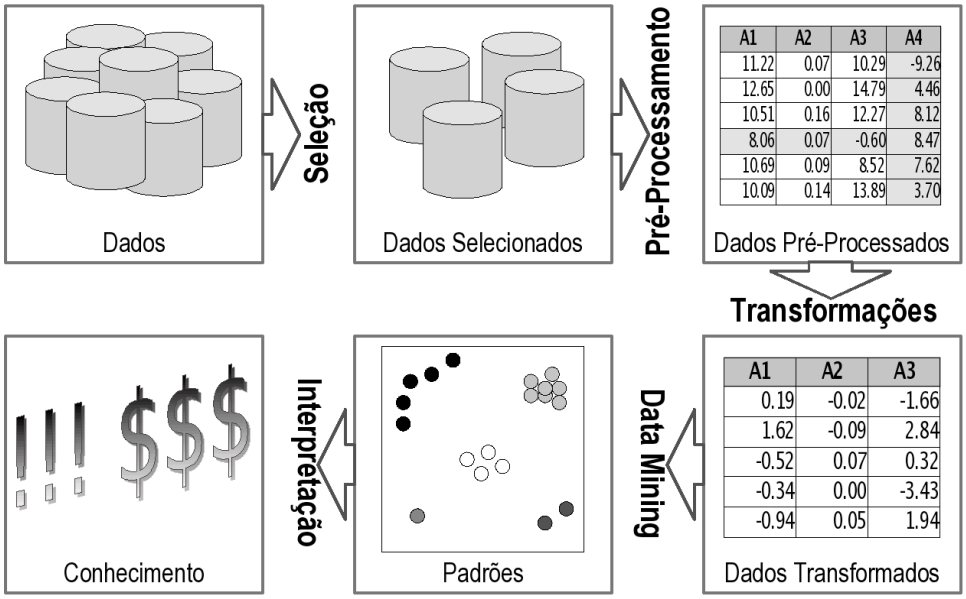

Fig. 2 – Etapas do processo de descoberta de conhecimentos em bancos de dados

Das etapas mostradas na Figura 2 a que nos interessa é a de mineração de dados, que trata, basicamente, da conversão de dados ou informações (que podem ter sido pré-processados) em padrões. Várias técnicas e algoritmos de transformação de dados em padrões podem ser usadas, sua escolha deve ser feita considerando o tipo de dado e o tipo de informação associada ao padrão que se deseja obter. Algumas técnicas genéricas são:

- Classificação: aprendizado de uma função que pode ser usada para mapear dados em uma de várias classes discretas definidas previamente.
- Regressão ou Predição: aprendizado de uma função que pode ser sada para mapear os valores associados aos dados em um ou mais valores reais.
- Agrupamento (ou *clustering*): identificação de grupos de dados onde os dados tem características semelhantes aos do mesmo grupo e onde os grupos tenham características diferentes entre si.
- Sumarização: descrição do que caracteriza um conjunto de dados (ex. conjunto de regras que descreve o comportamento e relação entre os valores dos dados).
- Detecção de desvios ou *outliers*: identificação de dados que deveriam seguir um padrão esperado mas não o fazem.
- Identificação de associações: identificação de grupos de dados que apresentam co-ocorrência entre si (ex. cesta de compras).

É possível ter técnicas híbridas ou que são usadas de forma diferente em diferentes passos do processo de KDD. Cada uma destas técnicas tem vários algoritmos, com variantes e diferentes parâmetros. Como mineração de dados é um processo às vezes exploratório, é útil conhecer ao menos alguns dos algoritmos mais comuns e usados para determinar qual deve ser aplicado à tarefa em questão. Neste mini-curso veremos alguns algoritmos representativos de algumas das técnicas mencionadas.

É importante ressaltar que mineração de dados não é um processo que pode ser feito às cegas: assim como mineração real, deve haver uma fase de prospecção para verificar a aplicabilidade e rentabilidade do empreendimento, e uma compreensão mínima dos dados e do que se espera obter deles para que os resultados sejam compreensíveis e aproveitáveis.

2.1 Alguns Exemplos

Alguns exemplos de sucesso de aplicação de técnicas de mineração de dados em vários domínios são fornecidos por empresas fornecedoras de softwares de mineração de dados e estatística. Por razões óbvias estas empresas

não divulgam detalhes exatos das técnicas usadas (nem exemplos de aplicações que ***não*** resultaram em vantagens para os clientes), mas mesmo assim é interessante saber como empresas tentam utilizar mineração de dados para obter vantagens comerciais ou estratégicas.
Alguns dos exemplos mostrados nesta seção foram obtidos nos sites das empresas SAS (http://www.sas.com) e StatSoft (http://www.statsoft.com). Estes exemplos incluem melhoria do relacionamento com o cliente através de tratamento personalizado para aumentar vendas (Amazon, 1-800-FLOWERS.com, Casino), reposicionamento no mercado para obter melhores lucros (TIM, Verizon, JCB, Harrah's). Alguns exemplos de aplicações em treinamento e identificação de recursos humanos foram criados pela Columbia University e KLM. Detalhes sobre estes exemplos podem ser encontrados em [5].
Existem exemplos específicos de aplicação de técnicas de mineração de dados para análise de dados científicos, mas podemos considerar que estas aplicações, em sua grande maioria, ainda são protótipos ou específicas demais para garantir replicabilidade em outros domínios. Isto não quer dizer que as técnicas de mineração de dados não possam ser usadas em aplicações científicas; pelo contrário, um maior conhecimento sobre as várias técnicas e algoritmos deve ser usado para permitir a exploração das possibilidades para obter resultados significativos.

3. Conceitos Básicos

3.1 Introdução

Para melhor entender as técnicas e algoritmos de mineração de dados, consideremos que temos tabelas simples com dados para processamento e análise; ou seja, com os dados distribuídos em uma matriz onde cada linha contém um conjunto de dados corresponde a um registro (ou amostra, ou instância), coletada sobre um evento, objeto, pessoa, transação, etc.; e cada coluna a um atributo daquele registro. A Figura 3 mostra uma tabela onde temos sete linhas, cada uma correspondente a um registro, e onde cada registro contém quatro atributos.

Outlook	Temperature	Humidity	Play?
sunny	85	80	yes
sunny	80	90	no
rainy	68	80	yes
rainy	65	70	no
sunny	69	70	?
rainy	75	80	?
overcast	75	70	?

Fig. 3 – Exemplo simples de tabela de dados

A tabela mostrada na Figura 3 ilustra, além dos conceitos de registro e atributo, os conceitos a seguir:

- Alguns valores de atributos podem estar faltando (no exemplo, alguns valores para o atributo “Play?”). Neste exemplo isto foi feito deliberadamente, mas podemos esperar que em aplicações reais alguns dados e seus atributos estejam faltando.
- Os valores dos atributos podem ser nominais (podendo assumir qualquer valor, sem ordenação), categóricos (nominais de um conjunto limitado, sem ordenação), numéricos ou nominais/ordinais (adaptado de [6]). O tipo de atributo influencia diretamente no tipo de algoritmo que pode ser usado para mineração dos dados.
- Um dos atributos pode ser considerado o conseqüente ou atributo de classificação. Classificação é uma tarefa bastante comum de mineração de dados onde o objetivo é determinar a regra que classifica o valor do conseqüente a partir do valor dos outros atributos.

3.2 Pré-processamento

A fase de pré-processamento em uma tarefa de mineração de dados (ou, de forma mais geral, de um processo de descoberta de conhecimento em bancos de dados) consiste em adequar os dados aos algoritmos que serão usados para a mineração em si. Esta adequação pode consistir de:

- Filtragem: seleção de que dados e/ou atributos serão usados em passos subseqüentes. Vários critérios podem ser usados, inclusive critérios que aumentam a qualidade dos dados (por exemplo, removendo registros com atributos incompletos, reduzindo o número de registros ou atributos, etc., dependendo da aplicação).
- Transformação: modificação dos tipos de atributos para adequação a algoritmos. Por exemplo, para alguns algoritmos é necessário transformar atributos nominais em listas de valores booleanos (categorias de pertinência); para outros pode ser necessário converter atributos categóricos ou nominais para numéricos, ainda para outros algoritmos todos os dados numéricos devem ser convertidos para categóricos.
- Enriquecimento: complementação dos dados com mais atributos que podem ser obtidos de outras fontes de dados ou criados artificialmente.
- Redução: eliminação de atributos ou dados para melhorar a performance dos algoritmos, preferencialmente sem comprometer a qualidade dos resultados esperados.

3.3 Espaço de Atributos

Um conceito importante em mineração de dados é o espaço de atributos. Basicamente o espaço de atributos de uma base de dados é uma representação destes dados em um espaço multidimensional onde cada dimensão corresponde a um atributo. Quando o número de atributos é adequado, podemos visualizar os dados de forma simples através de várias técnicas. Quando o número de atributos é elevado, técnicas específicas de visualização (apresentadas neste tutorial) ou técnicas de redução de atributos podem ser usadas.
Como exemplo, consideremos a Figura 4, que mostra um conjunto artificial de dados (respostas a dois testes clínicos depois de um tratamento sugerido) e sua representação gráfica. Embora tenhamos três dimensões (três atributos) podemos representar o terceiro (Tratamento) como um símbolo no gráfico à direita da Figura 4 (cruz para tratamento 1, x para tratamento 2, quadrado para tratamento 3), podendo assim representar os dados visualmente em duas dimensões.

R_1	R_2	Tratamento
6	7	1
5	9	1
8	6	1
4	9	1
7	9	1
3	3	2
1	6	2
2	3	2
2	3	3
5	1	3
3	1	3
2	3	3

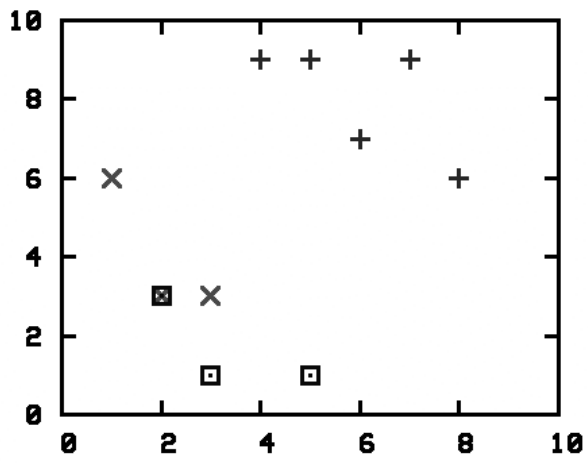

Fig. 4 – Conjunto artificial de dados e sua representação gráfica

O espaço de atributos ***não*** é a representação gráfica dos dados – é a coleção de vetores de dados em um espaço N-dimensional onde N é o número de atributos. Várias operações entre vetores de dados (mais notadamente, a

distância entre vetores) serão essenciais para compreender os algoritmos de classificação, agrupamento e regressão.
A visualização do espaço de atributos é uma técnica simples que permite obter um sentido preliminar dos dados e de sua distribuição. No exemplo mostrado na Figura 4, podemos ver que os valores das respostas R1 e R2 estão agrupados para o tratamento 1 e que existe uma superposição entre os valores de R1 e R2 para os tratamentos 2 e 3.

4. Métodos de Classificação Supervisionada

4.1 Introdução

Classificação é a técnica que permite prever qual deve ser a categoria discreta para um determinado dado. Assumimos que o conjunto de categorias ou classes é finito e previamente conhecido, e que um processo de treinamento deverá ser feito para que o algoritmo classificador seja capaz de determinar a classe para um dado. Para o treinamento do classificador devemos ter um conjunto de dados rotulados, isto é, para os quais as classes já são conhecidas. Os métodos que usam dados rotulados para treinar um classificador são conhecidos como classificadores supervisionados.
A Figura 5 ilustra o processo de classificação. Na parte superior, um algoritmo é treinado com um conjunto de dados para os quais sabemos as classes. Como resultado do treinamento temos assinaturas, protótipos, regras ou outra forma de representação de conhecimento que pode ser usada pelo classificador para reclassificar os dados para os quais já sabemos as classes (para avaliar a qualidade do classificador). Na parte inferior da Figura usamos estas assinaturas, protótipos, etc. para atribuir classes para dados para os quais não sabemos a que categorias pertencem.

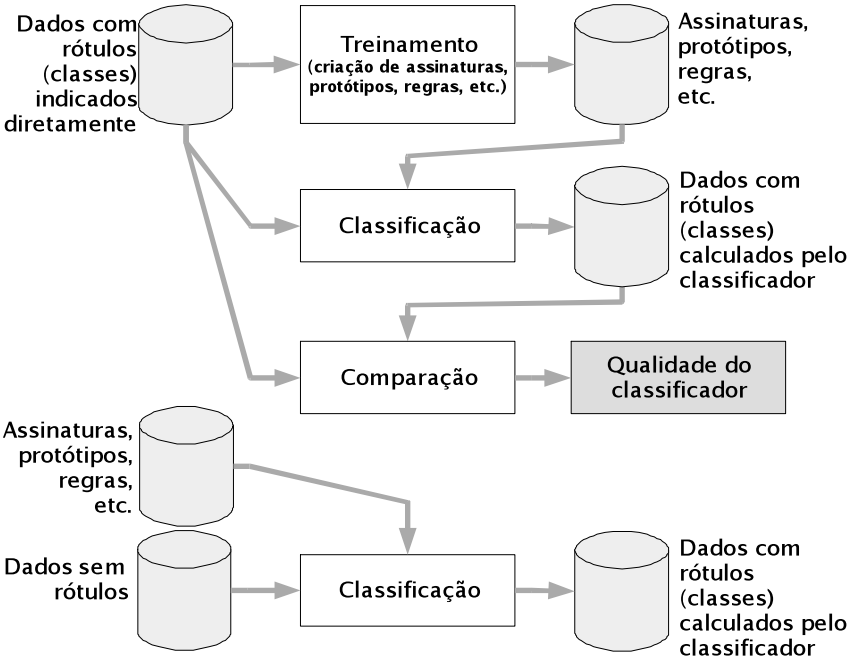

Fig. 5 – Processo de classificação

Pode-se pensar que o processo de classificação é redundante, pois já que devemos obter classes conhecidas para alguns dos dados, poderíamos obter as classes para *todos* os dados evitando o treinamento e aplicação do classificador. A verdade é que pode não ser eficiente ou mesmo possível obter classes para todos os dados, enquanto é relativamente simples obter classes para um subconjunto significante dos dados.

4.2 Métodos Baseados em Distâncias e Diferenças

Classificadores supervisionados freqüentemente usam uma medida de distância para determinar a classe à qual um dado pertence. Vários algoritmos de classificação supervisionada verificam a distância (no espaço de atributos) entre um dado de categoria desconhecida e dois ou mais de categorias conhecidas, e classifica o dado desconhecido como sendo da categoria mais próxima. Várias medidas de distância podem ser usadas, sendo a mais comum a distância Euclideana (que só pode ser calculada se todos os atributos dos dados forem numéricos). Para dados com atributos não-numéricos podemos usar técnicas de pré-processamento ou técnicas específicas para cálculo de distâncias.

4.3 Método da Mínima Distância Euclideana

Como primeiro exemplo de classificador supervisionado veremos este método (também conhecido como método da menor distância a protótipo), que é simples, rápido e intuitivo.
Para entender como este método funciona, usaremos a Figura 6 como exemplo. Nesta Figura temos, à esquerda, amostras de cinco classes conhecidas. A partir destas amostras criamos protótipos que são, para este classificador, simplesmente valores médios no espaço de atributos (no gráfico aparecem como cruzes no centro geométrico dos grupos de registros da mesma classe). O processo de classificação é mostrado à direita, onde calculamos as distâncias entre cada protótipo e cada dado com classe desconhecida, usando como classe aquela cujo protótipo está menos distante da classe desconhecida. O classificador implementa de forma simples o conceito intuitivo de que classes semelhantes estão próximas no espaço de atributos.
Existem vários outros classificadores, baseados neste método simples, que melhoram a capacidade de classificação através de uma melhor modelagem dos protótipos das classes. Este método considera que os atributos são numéricos e com faixas de valores aproximadamente equivalentes, e que as classes têm distribuição hiperesférica; variantes podem usar outros tipos de modelagem de distribuição (hiperelipsoidal, por exemplo) ou considerar que classes podem estar distribuídas disjuntamente no espaço de atributos.

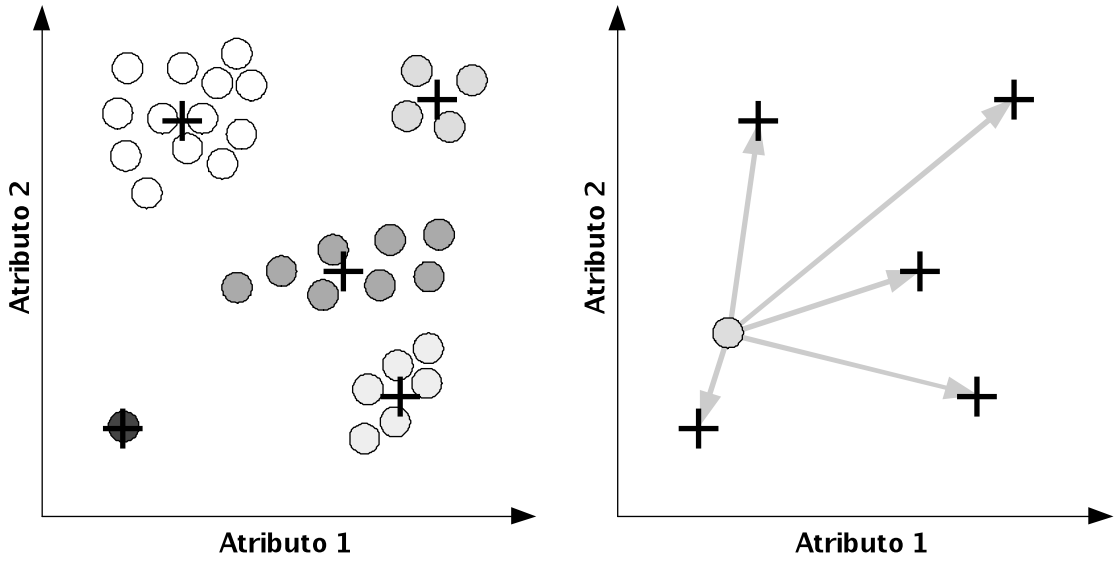

Fig. 6 – Classificação pelo Método da Mínima Distância Euclideana

4.4 Árvores de Decisão

Árvores de decisão são estruturas que fazem partições em um conjunto de registros de forma que seus galhos sejam decisões tomadas comparando atributos dos registros e determinados valores constantes, e onde cada folha contenha registros agrupados em uma única classe. Uma árvore de decisão pode ser interpretada como um

conjunto de regras que devem ser aplicadas a um registro para obter a classe correspondente a este registro: podemos então considerar árvores de decisão como similares a sistemas especialistas. Um exemplo gráfico de árvore de decisão (relacionado com um exemplo clássico de mineração de dados) pode ser visto na Figura 7. Os galhos são decisões baseadas em atributos da base de dados (características de pacientes de oculistas) e as folhas são recomendações para uso de lentes de contatos (*hard*, *soft* ou *none*).

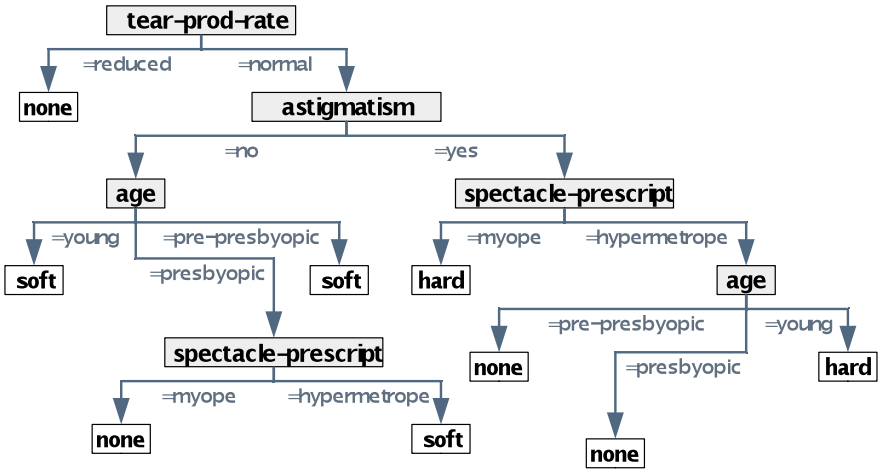

Fig. 7 – Árvore de decisão para prescrição de lentes de contato

A criação de árvores de decisão pode ser considerada como um tipo de treinamento de classificador supervisionado, e como resultado do treinamento obtemos a estrutura da árvore. A aplicação das regras em cada galho da árvore corresponde a uma partição no espaço de atributos, com regras em galhos mais "baixos" correspondendo a refinamentos das partições.

O algoritmo básico de criação de árvores de decisão é como segue:

1. O algoritmo inicia considerando todos os dados como pertencentes a uma única partição.
2. Para o conseqüente da árvore de decisão (atributo que será usado como classe), determina-se qual é o atributo que oferece o maior ganho de informação para criar partições. O ganho de informação é uma medida que indica a correlação ou dependência entre o atributo sendo considerado e o usado como classe. O atributo deve ser discreto, eventualmente sendo uma operação de comparação com um valor constante e o valor do atributo.
3. Criamos, usando este atributo, duas ou mais partições (*galhos*) da árvore.
4. Repetimos o passo 2 e seguintes para cada partição até que não seja possível particionar os galhos da árvore.

Árvores de decisão são métodos bastante poderosos e flexíveis para mineração de dados, sendo um dos métodos mais usados. Uma de suas vantagens é que a árvore pode ser facilmente interpretada, portanto adequada ou mesmo questionada como parte do processo de descoberta de conhecimento. Outra vantagem importante é que árvores de decisão podem ser usadas com registros que tenham dados numéricos, categóricos e praticamente de qualquer natureza.

Os algoritmos mais conhecidos para criação de árvores de decisão são o ID3 e o C4.5 [7], existindo várias implementações, algumas gratuitas e de código aberto. Detalhes passo a passo da criação de árvores de decisão e mais referências podem ser vistas em [5].

É importante notar que uma árvore de decisão não deve ser usada diretamente em tarefas de classificação, mas sim analisada e modificada para evitar criação de regras excessivamente complexas mas que correspondam a poucos casos da base de dados.

4.5 Redes Neurais para Classificação Supervisionada

Redes Neurais Artificiais (RNAs ou NNs, de *Neural Networks*) são simulações simplificadas do comportamento de neurônios reais. RNAs de diversos tipos podem ser usadas como classificadores supervisionados, sendo que os mais comuns são os perceptrons em múltiplas camadas, as redes *functional link* e *radial basis function networks* [8,9,10, 11]. Neste mini-curso veremos como redes de perceptrons em múltiplas camadas podem ser usados para classificação e mineração de dados.

A Figura 8 ilustra, à esquerda, o funcionamento de um neurônio artificial. Este possui várias entradas (valores numéricos) que terão pesos aplicados. A soma poderá ter um valor tendencioso (*bias*) removido, e será convertida em um novo valor, tradicionalmente entre zero e um, através de uma função sigmoidal ou similar. Um único neurônio pode então converter uma série de valores em um valor de ativação, que comumente é próximo de zero (neurônio não ativado) ou um (neurônio ativado).

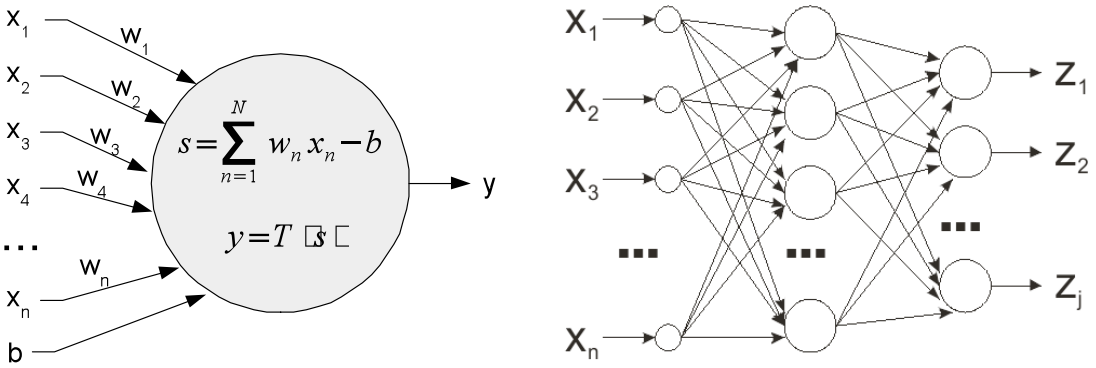

Fig. 8 – Perceptron e rede de múltiplas camadas de perceptrons

Um único neurônio é capaz de separar duas classes contanto que as mesmas sejam linearmente separáveis por um único plano. Para separar mais classes e possibilitar a separação não linear (através da combinação de separadores lineares) devemos usar um conjunto de neurônios em camadas, como mostrado na parte direita da Figura 8. As ativações dos neurônios são propagadas nas camadas da rede, e seu treinamento é feito apresentando-se valores de dados e classes esperadas e ajustando os pesos de cada neurônio até que a rede esteja treinada adequadamente. Tradicionalmente o número de neurônios na camada de entrada é igual ao número de atributos dos registros; e o número de neurônios na camada de saída é igual ao número de classes.

Redes neurais de múltiplas camadas de perceptrons conseguem separar classes com distribuições complexas, contanto que um número adequado de neurônios em camadas escondidas seja usado; mas só podem ser usadas se os atributos dos registros forem numéricos.

Mais detalhes sobre aplicação de redes neurais para mineração de dados (incluindo testes com várias arquiteturas de rede de múltiplos perceptrons) podem ser vistos em [5]. Detalhes sobre algoritmos e implementações podem ser vistos em [9,10].

5. Métodos de Classificação Não-Supervisionada

5.1 Introdução

Algoritmos de classificação não-supervisionada ou algoritmos de agrupamento ou *clustering* são algoritmos iterativos que criam grupos ou partições nos dados onde todos os registros em um grupo tenham uma característica em comum (sejam parecidos entre si) e onde grupos sejam diferentes entre si. Assim como em algoritmos de classificação supervisionada, devemos usar uma métrica qualquer de distância para considerar similaridade entre dados.

Estes algoritmos são chamados não-supervisionados pois não existe um passo de treinamento no qual devemos fornecer ao algoritmo dados com classes conhecidas, para que o mesmo "aprenda" quais são as características de cada classe. Estes algoritmos devem usar algum processo para determinar, de forma semi-automática, grupos similares de dados.

5.2 K-Médias e Isodata

O algoritmo mais tradicional de criação de grupos é o chamado K-médias. Este algoritmo usa somente como entrada os registros, uma métrica de distância e o número de grupos que devem ser formados (*K*). O algoritmo cria *K* centróides e busca, no espaço de atributos, quais registros "pertencem" ou estão mais próximos de quais centróides. O algoritmo então modifica as posições dos centróides (novamente no espaço de atributos) e repete estes passos iterativamente até que não exista mais modificação significativa das posições dos centróides. O resultado do algoritmo são os *K* centróides que podem ser considerados protótipos das classes naturais que aparecem no conjunto de dados. A Figura 9 ilustra quatro iterações do algoritmo com seis registros (cada um com duas dimensões numéricas) e com *K*=2.

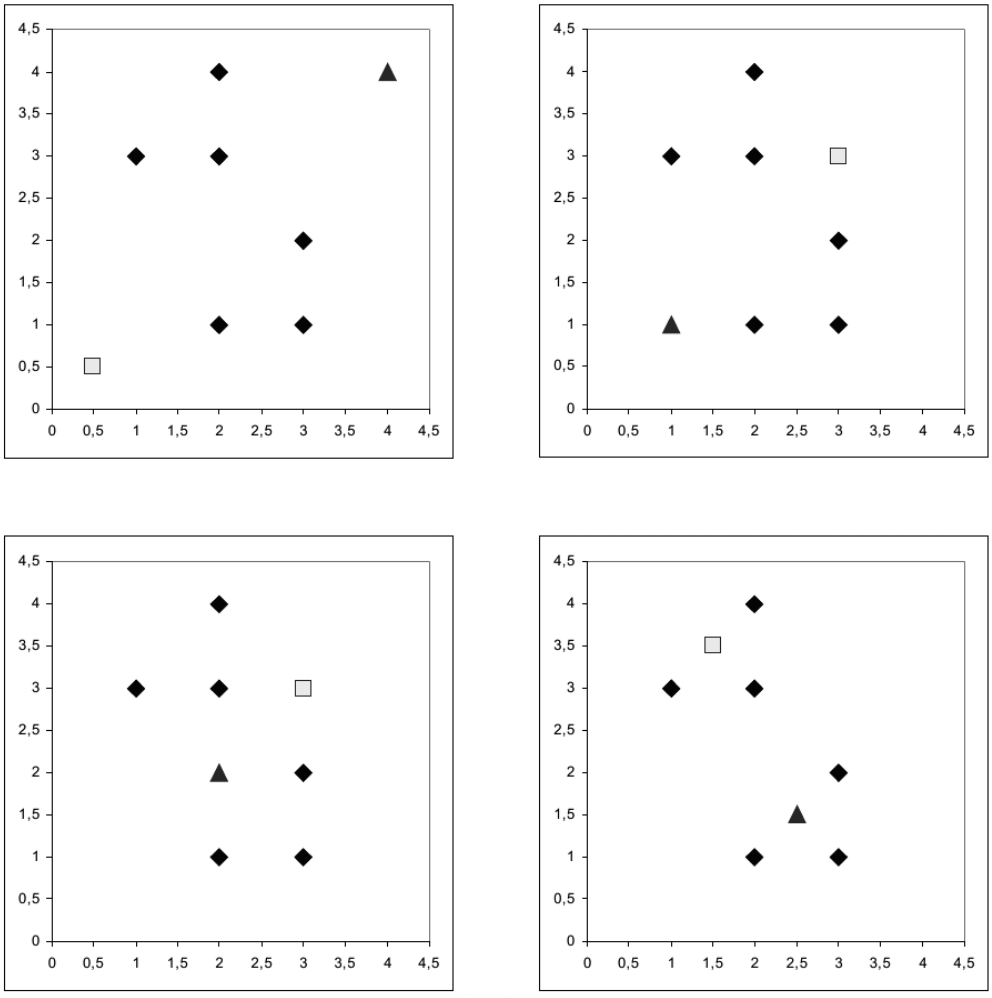

Fig. 9 – Iterações do algoritmo K-Médias

A implementação básica deste algoritmo somente funciona com atributos numéricos e com distribuições hiperesféricas, e assumindo que a escolha do valor de *K* foi ideal. Como o algoritmo deve iterar várias vezes,

calculando distâncias entre os *K* centróides e os registros em cada iteração, sua performance pode ser inadequada para várias aplicações. Detalhes de implementação podem ser vistos em [5].
Algumas variantes deste algoritmo tentam minimizar estas limitações. Uma variante interessante (que pode ser considerada outro algoritmo) é o chamado Isodata, que usa o K-Médias como base. Este algoritmo reconsidera os agrupamentos criados, juntando agrupamentos com poucos registros e que estejam próximos no espaço de atributos, e separando grupos muito grandes e esparsos, ajustando o valor de *K* até chegar em uma situação estável. Passos completos para o algoritmo Isodata estão em [10].

5.3 Fuzzy C-Médias

O algoritmo Fuzzy C-Médias é baseado no K-Médias mas considerando a pertinência de um registro a um grupo (*cluster*) como sendo um valor nebuloso, isto é, entre zero e um, onde zero indica não-pertinência e um, pertinência. Podemos considerar que o algoritmo K-Médias é um caso especial do Fuzzy C-Médias onde as funções de pertinência sejam extremas (zero ***ou*** um).
O algoritmo Fuzzy C-Médias é também iterativo, e requer mais cálculos do que o K-Médias, sendo potencialmente mais lento. Sua versão básica também considera que a distribuição dos grupos é hiperesférica no espaço de atributos. Adicionalmente, é necessário manter a função de pertinência (um valor real) entre cada registro e cada grupo, portanto seus requerimentos de uso de memória são maiores. Sua grande vantagem é justamente esta tabela de valores de pertinência, que pode ser usada para avaliar, para cada registro, não somente o grupo que deve ser usado para atribuição (usualmente o com maior valor de pertinência), mas quais são as alternativas; ou mesmo se o registro deve ser marcado como pertinente a um grupo, o que pode ser usado como uma medida de qualidade de pertinência para cada dado.
As fórmulas e passos para implementação deste algoritmo podem ser encontradas em [5]. Variantes deste algoritmo tentam tratar com agrupamentos não-hiperesféricos, e são descritos em [12,13,14].

5.4 Mapas Auto-Organizáveis de Kohonen

Mapas Auto-Organizáveis de Kohonen ou SOMs (*Self-Organizing Maps*) são modelos de redes neurais que mapeiam registros multidimensionais em um espaço topológico de poucas dimensões (usualmente duas), mantendo, através da topologia da rede, dados que são próximos no espaço de atributos. Estes mapas podem ser usados tanto para redução de dimensões (na fase de pré-processamento) quanto como um algoritmo de agrupamento (onde o número de grupos corresponde aproximadamente ao número de neurônios da rede) quanto como ferramenta de visualização, como será visto adiante.
Cada neurônio em um mapa auto-organizável representa um protótipo no mesmo espaço de atributos dos dados. Os mapas são treinados primeiro inicializando-se os neurônios com valores aleatórios, e então, iterativamente, localizando o neurônio mais similar (com menor distância) a um registro sendo considerado e ajustando os valores deste neurônio e de seus vizinhos para que fiquem mais semelhantes ao registro apresentado. Depois de muitas iterações a rede tende a apresentar, usando menos dimensões, a mesma distribuição dos dados no espaço de atributos. O algoritmo detalhado pode ser visto em [5,9,15]. Durante as iterações, o tamanho da vizinhança e o peso que influencia a modificação dos neurônios considerados mais semelhantes a um registro apresentado são reduzidos, fazendo com que o algoritmo se estabilize.
Uma ilustração do treinamento da rede pode ser visto na Figura 10, que mostra como uma rede com 25x25 neurônios se adapta, mantendo a topologia, para representar uma distribuição não-homogênea de registros com dados artificiais em duas dimensões. Inicialmente (em 0 iterações) os neurônios estão espalhados aleatoriamente, mas com 50.000 iterações eles começam a se espalhar para cobrir uma região correspondente à distribuição dos dados, e com 300.000 iterações a distribuição dos neurônios já começa a se assemelhar à dos registros originais.

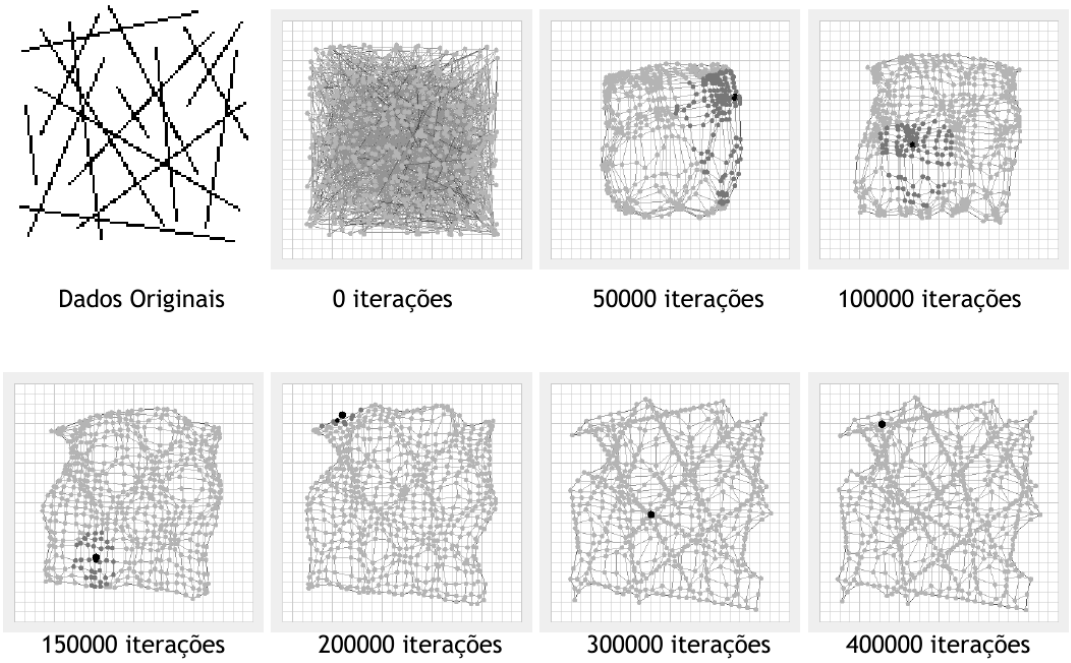

Fig. 10 – Fases do treinamento de um Mapa Auto-Organizável de Kohonen

Mapas auto-organizáveis de Kohonen podem também ser usados como algoritmos de classificação semi-supervisionada se, para alguns neurônios, for possível determinar as classes corretas. Todos os dados podem então ser classificados baseados em uma medida de similaridade ou distância dos neurônios considerados treinados.

5.5 Agrupamento Hierárquico

Os algoritmos descritos nesta seção são considerados particionais, isto é, que criam partições ou grupos nos registros. As partições são criadas de forma simultânea, e consideradas simultaneamente pelos algoritmos (respeitadas imposições na implementação) – em outras palavras, em princípio o número de partições é fixo e determinado.

Outros algoritmos de agrupamento são chamados hierárquicos, pois ao invés de buscar a melhor forma de agrupar os registros em um determinado número de partições, consideram todos as possíveis maneiras de particionar um conjunto de *M* registros, desde a partição básica de um registro em cada grupo (*M* grupos) até um único grupo contendo todos os registro, criando todos os agrupamentos com número de grupos entre *M* e 1. Estes algoritmos, em sua maioria, são implementados de forma *bottom-up*, ou seja, começam com cada registro em um grupo e vão buscando a melhor forma de juntar dois grupos para formar um só, até que todos os registro estejam em um único grupo. Estes algoritmos usam, em sua forma básica, uma matriz de distâncias entre os grupos calculada a cada iteração, e uma de várias métricas possíveis para cálculo da distância entre dois grupos (as mais conhecidas sendo *single linkage*, *complete linkage* e *average linkage*).

A vantagem deste tipo de algoritmo é que ele cria, como passo intermediário durante o agrupamento, um *dendograma* (Figura 11) que mostra a estrutura do agrupamento, permitindo a análise exploratória dos resultados intermediários para, por exemplo, determinar experimentalmente o número de grupos desejados. Algoritmos de agrupamento hierárquico são comumente usados com registros com atributos numéricos, mas com uma função de distância ou diferença adequada, nada impede que os usemos com registros com atributos categóricos ou ordinais.

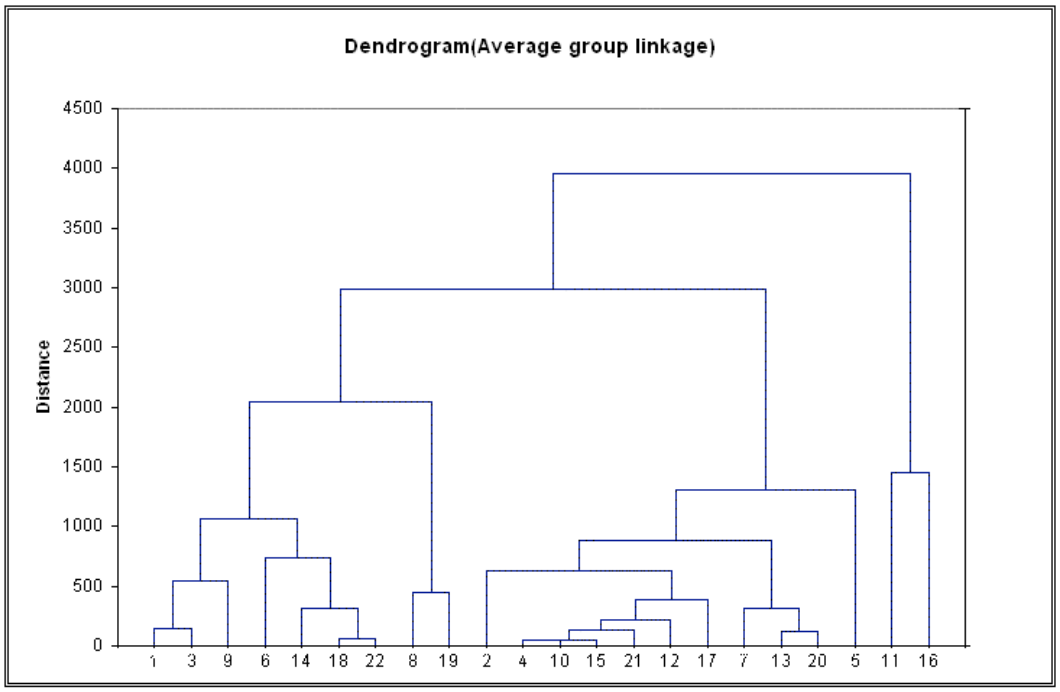

Fig. 11 – Dendograma (fonte: http://www.resample.com/xlminer/)

6. Descoberta de Regras de Associação

6.1 Introdução

Os algoritmos de classificação supervisionada e não-supervisionada até então permitem a classificação ou agrupamento de registros em classes, grupos ou categorias. Outra tarefa importante de mineração de dados é a descoberta de associações em conjuntos de dados, que tem premissas diferentes.

Consideremos um conjunto de eventos onde cada evento pode conter um ou mais objetos. O exemplo clássico de carrinho de compras considera que uma compra em um supermercado (produtos em um carrinho) é um evento, e que cada produto no carrinho é um objeto. Um outro exemplo, de cunho mais científico, é o de busca de associação entre relâmpagos e precipitação; um evento seria associado a uma determinada posição no espaço e tempo e os objetos seriam a ocorrência ou não de precipitação, relâmpagos e outros dados associados.

Algoritmos de descoberta de associações tentam identificar, considerando o conjunto de eventos, quais objetos acontecem de forma associada, isto é, quais objetos co-ocorrem nos eventos. Para ilustrar, vejamos a Figura 12, que mostra um exemplo bem simples, simulado, de carrinho de compras, com poucos tipos de objetos. Uma análise superficial dos eventos nesta Figura mostra algumas co-ocorrências (café e açúcar, ovos e leite), mas é necessário quantizar estas co-ocorrências considerando duas métricas:

- Suporte de X para Y, onde X e Y são dois objetos em um evento: número de eventos que contém os objetos X e Y, dividido pelo número total de eventos. Esta métrica indica o quão significante uma co-ocorrência é (se é rara ou freqüente).
- Confiança de X para Y, onde X e Y são dois objetos em um evento: número de eventos que contém os objetos X e Y, dividido pelo número de eventos que contém o objeto X. Esta métrica indica se Y ocorre em função de X ocorrer em um mesmo evento, e com que freqüência.

É importante entender que a métrica de suporte não considera uma ordem para as co-ocorrências (significando que a confiança de X para Y deve ser a mesma de Y para X), enquanto a métrica confiança sim: o valor da confiança de X para Y pode (e freqüentemente é) diferente do de Y para X.

Transação	Itens
1	leite, ovos, café, açúcar, fraldas, manteiga
2	leite, café, farinha
3	leite, ovos, açúcar
4	café, açúcar
5	fraldas
6	manteiga, ovos, leite
7	café, açúcar, leite, ovos
8	farinha, manteiga, ovos
9	manteiga, ovos, leite, café, açúcar
10	fraldas, café, cerveja

Fig. 12 – Lista de eventos e objetos (carrinho de compras)

Voltando ao exemplo da Figura 12, podemos ver que a associação [ovos, leite] ocorre em 50% dos eventos, sendo bem freqüente; e que quem compra ovos e café sempre compra leite. Este tipo de conclusão pode ser usado por um analista (com conhecimento sobre os dados e sua implicância no negócio em questão) para tomar importantes decisões estratégicas.

6.2 O Algoritmo Apriori

O principal algoritmo para extração de regras de associação a partir de conjuntos de eventos é o APriori. Este algoritmo identifica todos os conjuntos de objetos co-ocorrentes em eventos (cada conjunto é considerado um *K-itemset*, onde K é o número de objetos no conjunto). Seus passos (sumarizados) são:

1. Definimos um valor de suporte e confiança mínima para uso pelo algoritmo.
2. Começamos com K = 1 para a criação dos K-itemsets.
3. Listamos os K-itemsets com suporte acima do suporte mínimo.
4. Criamos todas as combinações de (K+1) objetos usando a tabela do passo anterior, estes serão os candidados a (K+1)-itemsets.
5. Se a lista de combinações não for fazia, voltamos ao passo 3.
6. Criamos todas as permutações dos itemsets descobertos, filtrando pelo limite de confiança.

O algoritmo recursivamente cria listas de combinações dos objetos, e estas listas podem se tornar bem extensas se o número de objetos for muito grande. É importante ter valores mínimos adequados para garantir que esta lista seja tratável por um computador. Detalhes sobre a implementação deste algoritmo podem ser vistos em [5,16].

Para ilustrar o algoritmo, vejamos as listas de K-itemsets que podem ser criadas usando os eventos na Figura 12 e considerando o suporte mínimo como 25% e a confiança como 75%. A Figura 13 mostra a lista de K-itemsets para K=1,2,3 e 4, descobertos pelo algoritmo, considerando somente o suporte mínimo como 25%. O passo 6 do algoritmo criaria as permutações para verificação dos itemsets com confiança mínima, que seriam o resultado final do algoritmo. Como a lista é bem extensa, não será reproduzida neste documento, mas pode ser vista em [5].

1-itemsets	Suporte
leite	**60.00%**
ovos	**60.00%**
café	**60.00%**
açúcar	**50.00%**
fraldas	**30.00%**
manteiga	**40.00%**
farinha	20.00%
cerveja	10.00%

2-itemsets	Suporte
[leite,ovos]	**50.00%**
[leite,café]	**40.00%**
[leite,açúcar]	**40.00%**
[leite,fraldas]	10.00%
[leite,manteiga]	**30.00%**
[ovos,café]	**30.00%**
[ovos,açúcar]	**40.00%**
[ovos,fraldas]	10.00%
[ovos,manteiga]	**40.00%**
[café,açúcar]	**40.00%**
[café,fraldas]	20.00%
[café,manteiga]	20.00%
[açúcar,fraldas]	10.00%
[açúcar,manteiga]	20.00%
[fraldas,manteiga]	10.00%

3-itemsets	Suporte
[leite,ovos,café]	**30.00%**
[leite,ovos,açúcar]	**40.00%**
[leite,ovos,manteiga]	**30.00%**
[leite,café,açúcar]	**30.00%**
[leite,café,manteiga]	20.00%
[leite,açúcar,manteiga]	20.00%
[ovos,café,açúcar]	**30.00%**
[ovos,café,manteiga]	20.00%
[ovos,açúcar,manteiga]	20.00%
[café,açúcar,manteiga]	20.00%

4-itemsets	Suporte
[leite,ovos,café,açúcar]	**30.00%**
[leite,ovos,café,manteiga]	20.00%
[leite,ovos,açúcar,manteiga]	20.00%
[leite,café,açúcar,manteiga]	20.00%
[ovos,café,açúcar,manteiga]	20.00%

Fig. 13 – Itemsets descobertos no carrinho de compras

Assim como árvores de decisão, resultados do algoritmo APriori não devem ser usados diretamente para tomada de decisão, e sim processados por especialistas no domínio para uma análise mais qualitativa.

Existem várias variantes deste algoritmo: uma das mais interessantes é a que cria associações negativas, ou seja, busca também associações por ausência, que correspondem a regras do tipo "onde X ocorre, Y não ocorre". Estas regras também devem ser analisadas com bastante cuidados, em casos onde os objetos são muito numerosos (como no exemplo clássico do carrinho de supermercado, que pode ter milhares de objetos diferentes) o número de regras negativas pode ser enorme.

7. Outros Algoritmos

Muitos outros algoritmos e heurísticas oriundos de diversas áreas (biologia, controle, inteligência artificial, etc.) podem ser usados para descoberta de padrões, portanto, como parte do processo de descoberta de conhecimento em bases de dados. Alguns destes são:

- Ant Colonies: família de algoritmos baseados em simulações de colônias de formigas. Usam um mapa de feromônios depositados pelas formigas que percorrem caminhos (soluções) como memória global. Como muitas iterações são executadas pelas formigas, que se comportam como agentes reativos, pode-se chegar a uma solução ótima em um problema de otimização. Um exemplo clássico deste algoritmo é o de localização de rotas por depósito reforçado de feromônios; podemos usar este tipo de abordagem para a descoberta de regras e antecedentes, fazendo com que as formigas que "percorram" rotas (regras) mais bem-sucedidas depositem mais feromônios nestas. Outro exemplo é o algoritmo Lumer-Faieta, que faz a organização de tipos de objetos usando atributos e organizações no espaço através de agrupamento condicional. Algoritmos baseados em formigas também têm sido adaptados como algoritmos de agrupamento. Algumas referências para este tipo de algoritmo são [17,18,19,20,21].
- Algoritmos genéticos: algoritmos que criam populações com muitas possíveis soluções para um problema, mapeando as soluções para um código genético artificial, criando cruzamentos e mutações nestes códigos e selecionando os que apresentam um melhor desempenho baseado em alguma métrica. Estes algoritmos são muito usados em otimização, servindo para maximizar/minimizar funções, podendo ser adaptados para encontrar regras, divisões, separações ideais (ou quase) em dados no espaço de atributos. Algumas referências são [22,23,24].

8. Visualização e Mineração de Dados

Técnicas de visualização podem ser usadas no início do processo de mineração de dados (para ter uma idéia do tipo de distribuição dos registros, verificar separabilidade linear, visualizar formação de grupos naturais, etc.), ou no final do processo, para obter uma sumarização visual dos resultados da mineração ou ver as distribuições de dados contextualizadas, ou seja, enriquecidas pelo processo de mineração.

Não existe um único processo ou técnica de visualização que seja adequado para qualquer tipo de tarefa, portanto, é melhor sempre considerar o tipo e número de registros, tipos de atributos e resultados esperados

quando selecionarmos técnicas de visualização para mineração de dados. Nesta seção veremos algumas categorias e técnicas com exemplos.

Técnicas geométricas de visualização são aquelas que usam transformações e projeções dos dados para tentar mostrar as várias dimensões dos dados originais em um número menor de dimensões, freqüentemente duas, compatível com a limitação do *hardware* sendo usado. São as técnicas mais comuns de visualização. Dentre estas, podemos destacar as s*catterplot matrices*, *parallel coordinates* e visualização com mapas auto-organizados de Kohonen.

A técnica s*catterplot matrices* é útil para visualização de grupos e possibilidade de separação linear dos dados. Consiste em criar uma matriz quadrada onde cada célula é usada para mostrar os valores de dois dos atributos da base de dados, plotados um contra o outro, em um gráfico XY ou *scatterplot*. A técnica prosection views, baseada nesta, permite a seleção multidimensional de um subconjunto dos registros para processamento posterior.

A técnica *parallel coordinates* [25] usa uma projeção dos atributos em eixos paralelos, permitindo que cada registro seja plotado em um gráfico como uma série de segmentos de reta ligando os valores dos eixos correspondentes aos atributos. Registros próximos no espaço de atributos aparecerão como linhas com poucas variações. A Figura 14 ilustra esta técnica, mostrando os dados das flores Íris (outro exemplo clássico de mineração de dados), com 150 registros, cada um com quatro atributos numéricos e uma categoria. A cor das linhas reflete a categoria. Esta técnica pode ser usada com dados numéricos, ordinais, categóricos, etc., também permitindo facilmente a identificação de atributos que podem ser melhor usados para classificação (no caso, comprimento e largura da pétala parecem ser melhores do que comprimento e largura da sépala para separação e classificação).

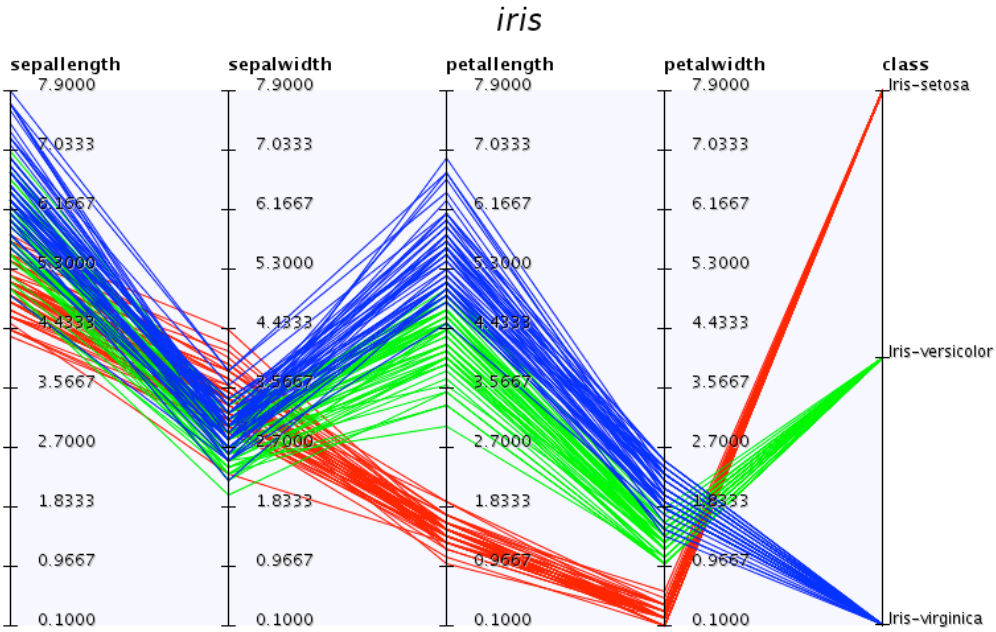

Fig. 14 – Plotagem *Parallel Coordinates* com dados de flores Íris

Outro exemplo simples mas interessante da aplicação desta técnica é mostrado na Figura 15. Esta figura mostra dados coletados de um sensor de luminosidade de centenas de plataformas de coleta de dados, organizados em coleções de oito registros, correspondentes a medições de luminosidade em períodos de 24 horas. Podemos observar um certo padrão no comportamento temporal destes dados, mas podemos ver também que algumas

plataformas estão coletando dados incorretos (linhas constantes em valores elevados), e que algumas medidas tem erros esporádicos (picos inesperados).

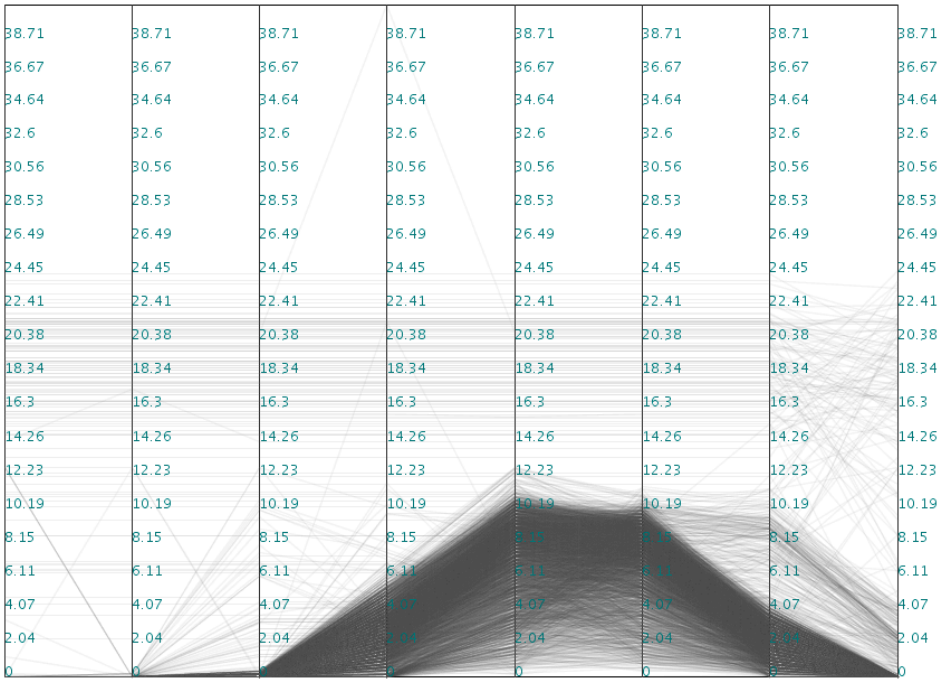

Fig. 15 – Plotagem *parallel coordinates* com dados de plataformas de coleta de dados

Uma terceira técnica geométrica de visualização é baseada nos Mapas Auto-Organizáveis de Kohonen. Para usar esta técnica devemos criar um mapa de duas dimensões e ter uma maneira de visualizar os valores associados a cada neurônio. Como o mapa deve preservar a proximidade dos vetores de dados no espaço de atributos original, podemos assumir que neurônios próximos no mapa correspondem a vetores próximos no espaço de atributos, o que pode ser confirmado e/ou reforçado pela visualização, que pode mostrar também padrões inesperados. Como exemplo usamos um mapa de Kohonen de 12x12 neurônios e o treinamos com uma série temporal de dados de nível do Rio Ladário. Cada registro usado para treinar a rede corresponde às medidas do nível do rio tomadas em 8 dias consecutivos. Cada neurônio representará graficamente a pequena série temporal como um gráfico de

As Figuras 16, 17 e 18 mostram a evolução do treinamento da rede. Inicialmente (Figura 16) os neurônios contém valores aleatórios, o que pode ser observado. Após algum tempo de treinamento (Figura 17) alguns padrões começam a surgir, e ao final do treinamento (Figura 18) temos padrões claramente delineados. Este mapa visual pode então ser usado para análise dos padrões, identificação dos *outliers*, etc.

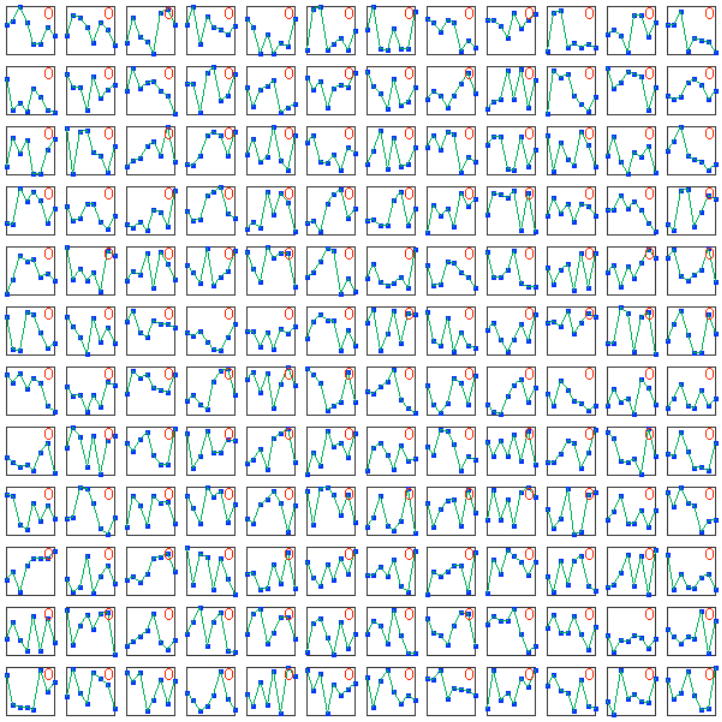

Fig. 16 – Mapa Auto-Organizável usado em visualização (inicialização randômica)

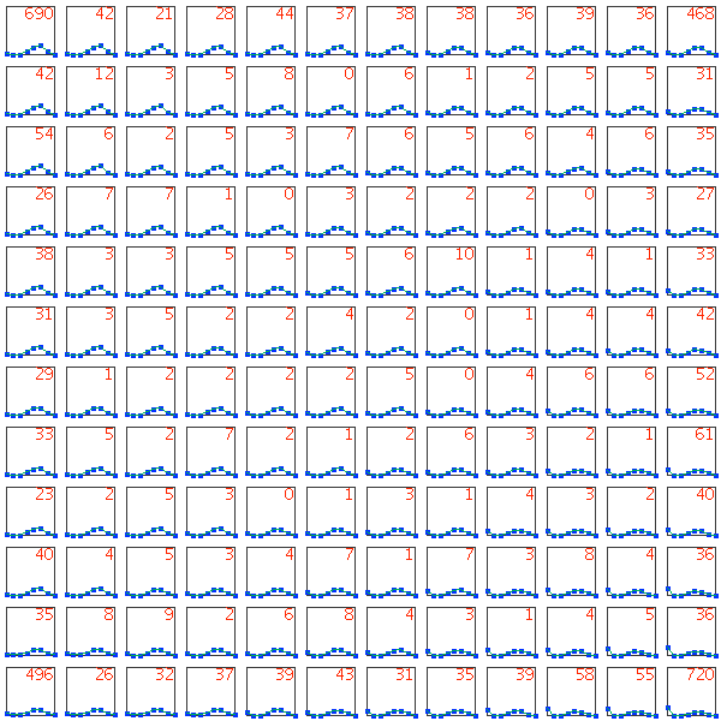

Fig. 17 – Mapa Auto-Organizável usado em visualização (durante treinamento)

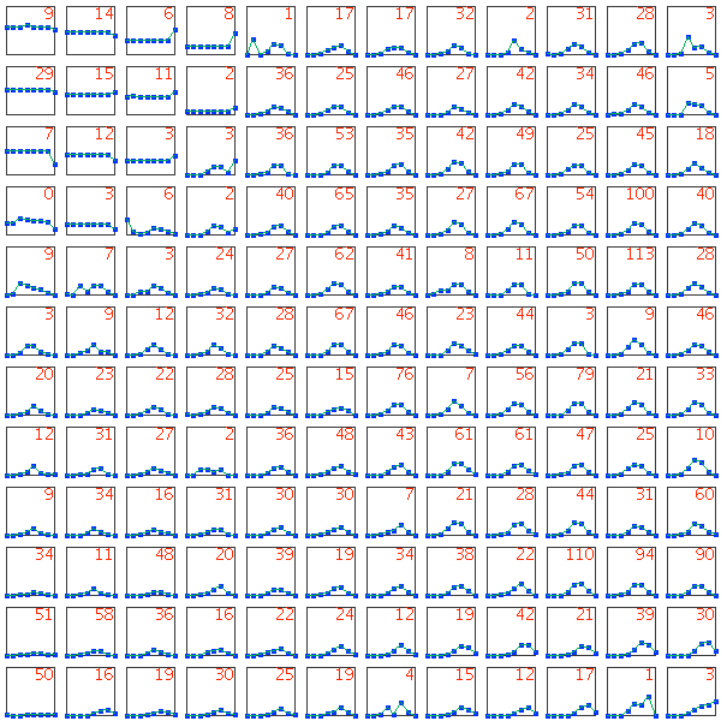

Fig. 18 – Mapa Auto-Organizável usado em visualização (ao fim do treinamento)

Outra categoria de técnicas de visualização é a que usa ícones para representar dados multidimensionais em uma posição bidimensional. Suas várias implementações assumem que dois dos atributos da base de dados podem ser usados como coordenadas para posicionar os ícones, e que os ícones representarão os outros atributos [26]. Um exemplo clássico é a técnica *Chernoff Faces* [27], que usa atributos visuais de pequenas faces para representar dados multidimensionais, como no exemplo da Figura 19.

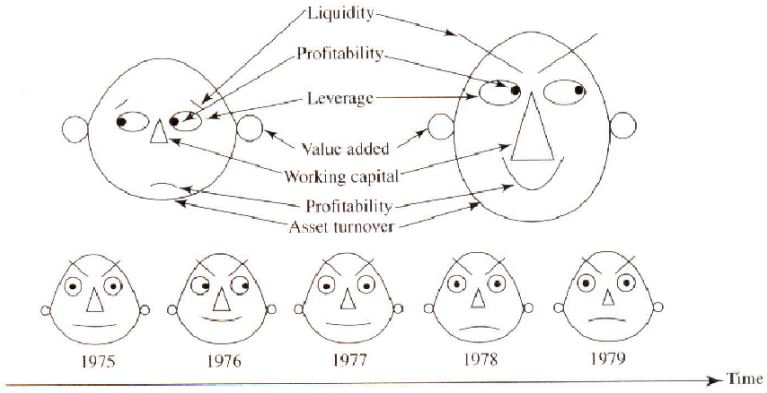

Fig. 19 – Visualização de uma série temporal com *Chernoff Faces*

É importante observar que as algumas das técnicas baseadas em ícones (como a Chernoff Faces, por exemplo) dependem, de interpretação ativa por parte de um usuário, portanto devem sempre estar acompanhadas de legendas.

Ainda outro exemplo de técnica de visualização por ícones é a *Stick Figures* [26,28], que mapeia múltiplos atributos numéricos em segmentos de reta, que variam em comprimento e ângulo com outros segmentos. A

técnica permite a visualização de fenômenos espaciais (no espaço de atributos) através da capacidade humana de detectar texturas em imagens. A Figura 20 mostra, à esquerda, uma *stick figure* que representa cinco dimensões numéricas através de seus ângulos (o comprimento dos segmentos é constante neste exemplo), e à direita, uma coleção de *stick figures* de diversas configurações. A Figura 21 mostra uma imagem de satélite, onde cada pixel representa cinco valores, como uma *Stick Figure*.

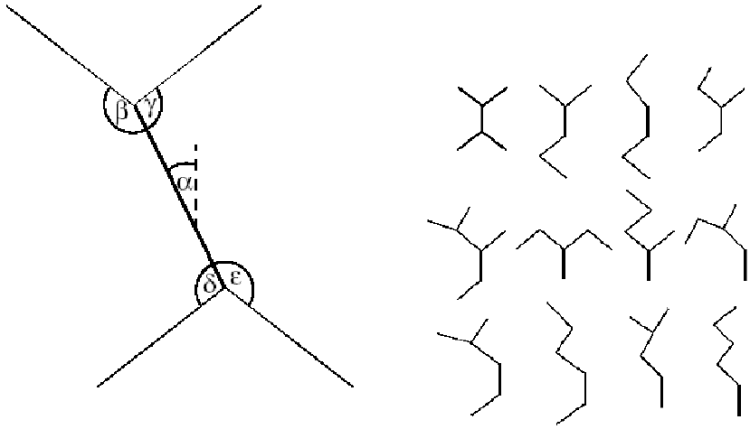

Fig. 20 – *Stick Figures*

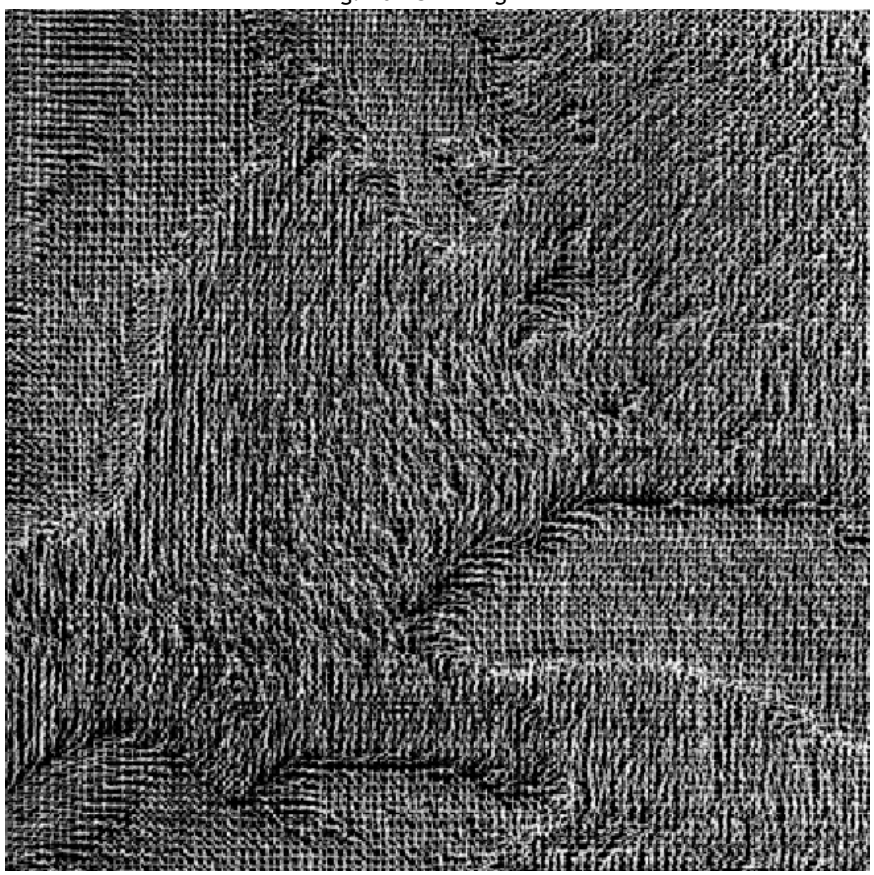

Fig. 21 – Imagem (5 dimensões mais X e Y) representada com *Stick Figures* (adaptada de [26]).

Muitas outras técnicas de visualização existem, e os tutoriais em [26] são um bom ponto de partida para a sua exploração.

Bibliografia

[1] Andrew Hanushevsky and Marcin Nowac, *Pursuit of a scalable high performance multi-petabyte database*, em *Proceedings of the 16th IEEE Symposium on Mass Storage Systems*, 169-175, 1999.

[2] Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth and Ramasamy Uthurusamy (editores), Advances in Knowledge Discovery and Data Mining, MIT Press, 1996.

[3] Marcelino Pereira dos Santos Silva, Metodologia de Mineração de Padrões de Mudança em Imagens de Sensoriamento Remoto (tese de doutorado), Instituto Nacional de Pesquisas Espaciais, 2006.

[4] Winter Corporation, *2005 TopTen Award Winners*, em http://www.wintercorp.com/VLDB/2005_TopTen_Survey/TopTenWinners_2005.asp.

[5] Material da disciplina CAP-359: Introdução à Mineração de Dados, em http://www.lac.inpe.br/~rafael.santos/cap.jsp

[6] Dorian Pyle. Data Preparation for Data Mining. Academic Press, 1st edition, 1999.

[7] J. Ross Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann, 1993.

[8] R. Beale and T. Jackson. Neural Computing: An Introduction. MIT Press, 1990.

[9] Laurene V. Fausett. Fundamentals of Neural Networks. Prentice Hall, 1994.

[10] Carl G. Looney. Pattern Recognition Using Neural Networks. Oxford University Press, 1st edition, 1997.

[11] Yoh-Han Pao, editor. Adaptive Pattern Recognition and Neural Networks. Addison-Wesley, 1989.

[12] James C. Bezdek. Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, 1st edition, 1987.

[13] James C. Bezdek and Sankar K. Pal. Fuzzy Models for Pattern Recognition. IEEE Press, 1st edition, 1992.

[14] Witold Pedrycz. Knowledge-Based Clustering – From Data to Information Granules. Wiley-Interscience, 1st edition, 2005.

[15] T. Kohonen, Self-Organizing Maps, Springer, 1987.

[16] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. InProc. 20th Int. Conf. Very Large Data Bases, VLDB, pages 487–499. Morgan Kaufmann, 1994.

[17] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, 1999.

[18] A. Abraham and V. Ramos. Web usage mining using artificial ant colony clustering. In Proceedings of Congress on Evolutionary Computation, pages 1384–1391, 2003.

[19] J. Handl and M. Dorigo. On the performance of ant-based clustering. In Proceedings of the 3rd International Conference on Hybrid Intelligent Systems, 2003.

[20] Nicolas Monmarché, M. Slimane, and Gilles Venturini. On improving clustering in numerical databases with artificial ants. In Proceedings of ECAL99 – European Conference on Artificial Life, pages 626–635, 1999.

[21] V. Ramos and A. Abraham. Evolving a stigmergic self-organized data-mining. In Proceedings ISDA-04, 4th International Conference on Intelligent Systems, Design and Applications, pages 725–730, 2004.

[22] David E. Goldberg. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, 1989.

[23] Randy L. Haupt and Sue Ellen Haupt. Practical Genetic Algorithms. Wiley-Interscience, 2nd edition, 2004.

[24] Zbigniew Michaelewicz. Genetic Algorithms + Data Structures = Evolution Programs. Springer, 3rd edition, 1996.

[25] A. Inselbert and B. Dimsdale, Parallel Coordinates for Visualizing Multi-Dimensional Geometry, Proc. Computer Graphics International Conference, Springer-Verlag, Tokyo, 1987.

[26] Daniel A. Keim, Databases and Visualization, Tutorial Notes, http://www.cip.ifi.lmu.de/~keim/

[27] H. Chernoff, The Use of Faces to Represent Points in K-Dimensional Space Graphically, Journal of the American Statistical Association, Vol. 68, pp. 361-368.
[28] R. M. Pickett, G. G. Grinstein, Iconographic Displays for Visualizing Multidimensional Data, Proceedings of the IEEE Conference on Man, System and Cybernetics, IEEE Press, Piscataway, NJ, 1988, pp. 514-518.