Recuperação de informação por similaridade utilizando técnicas inteligentes

Ernesto Cuadros Vargas—

Recuperação de informação por similaridade utilizando técnicas inteligentes[1]

Ernesto Cuadros Vargas

Orientador: *Profa. Dra. Roseli Aparecida Francelin Romero*

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação — ICMC/USP — como parte dos requisitos necessários para a obtenção do título de Doutor em Ciências — Ciências da Computação e Matemática Computacional.

USP - São Carlos
Estado de São Paulo – Brasil
Janeiro/2004

[1]Trabalho realizado com auxílio financeiro da FAPESP 99/11835-7.

A Govy, minha esposa e Ana Flávia, minha filha, com amor admiração e gratidão por sua compreensão, carinho, presença e incansável apoio ao longo período de elaboração deste trabalho.

Agradecimentos

A Deus por tudo o que eu recebi na vida. À minha esposa Govy pelo incentivo, apoio e compreensão nos períodos que estive fora do país. À minha orientadora Profa. Dra. Roseli Aparecida Francelin Romero que esteve sempre presente e atenciosa. Agradeço-lhe a paciência, o apoio e os múltiplos conselhos recebidos ao longo desses quatro anos de trabalho.

À FAPESP pela confiança depositada neste trabalho e pela atenção dispensada em todas as oportunidades que os consultei por diversas dúvidas.

Aos meus pais, Julio Ernesto e Yolanda, pela educação, amor e carinho, apesar de estarem muito longe. Aos meus irmãos Alex Jesús e Chepi pelo incentivo, pelo exemplo e por estarem sempre do meu lado.

Ao Prof. Dr. Christos Faloutsos, meu supervisor durante o programa de estágio na *Carnegie Mellon University*, com quem aprendi muitas coisas que contribuíram para a minha formação acadêmica e à minha vida.

Ao Prof. Dr. Klaus Obermayer, meu supervisor durante o estágio na *Technischen Universität Berlin*, pela ajuda dispensada nesse período.

Ao Prof. Dr. Ricardo Baeza Yates pelo incentivo, orientação e simplicidade com a qual sempre atendeu às minhas dúvidas.

Ao Bruce e à Gail Campbell que me receberam na sua casa em Pittsburgh sem me conhecer. Conhecer e conviver com eles foi a melhor parte da minha experiência nos EUA.

Aos docentes do ICMC, especialmente aos Profs. Drs. Agma Traina, Caetano Traina, Cristina Ferreira, Edson dos Santos Moreira, Rosane Minghim, Gustavo Nonato, André de Carvalho, Zhao Liang, com os quais aprendi muitas coisas úteis que nunca vou esquecer.

Aos meus amigos do ICMC, TU-Berlin, e peruanos Alex, Jorge Félix (Yorch), Miluska Félix, Chandler (Xede), Adenilso, Tatiana, Luciana, Cabelinha (Andrea), Rudinei, Waldo, Cesar (titi), Eduardo Tejada, Eduardo Llapa, Patricia, Guillermo (Web), Percy, Oliver Beck, Bambang, Hendrik Purwins, Anca Dima, Susanne Schönknecht, Sepp Hochreiter e Raul Antúnez de Mayolo pelo aprendizado nos países que visitei, pela amizade e a chance de ter compartilhado um pouco da minha vida com eles.

Aos meus colegas Patricia Rufino, Tatiana Sugeta, Rudinei Goulart, Adenilso da Silva Simão e Jaqueline Bigladori pelas dicas para melhorar a redação do texto.

Ao pessoal de segurança do ICMC que me ajudaram naquele 31 de dezembro em que cheguei em São Carlos sem conhecer nem falar quase nada de português.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Abreviaturas

AESA *Approximating Eliminating Search Algorithm*

AHIGG *Adaptive Hierarchical Incremental Grid Growing*

ART Teoria da Ressonância Adaptativa *Adaptive Resonance Theory*

BKT *Burkhard-Keller Tree*

BST *Bisector Trees*

CHL Aprendizado Hebbiano Competitivo – *Competitive Hebbian Learning*

CSOM SOM Construtivos

DCSGCS *Dynamic Cell Stuctures – Growing Cell Structures*

FHQT *Fixed-Height QT*

FQA *Fixed Queries Array*

FQT *Fixed-Queries Trees*

GCS *Growing Cell Structures*

GHSOM *Growing Hierarchical Self-Organizing Map*

GHT *Generalized-Hyperplane Tree*

GNAT *Geometric Near-neighbor Access Tree*

GNG *Growing Neural Gas*

GSOM *Growing Self-Organizing Maps*

GTM *Generative Topographic Mapping*

HiGS *Hierarchical* GCS

IGG *Incremental Grid-Growing*

IMiMD Indexação e Mineração de Dados Multimídia – *Indexing and Data Mining in Multimedia Data Bases*

LAESA *Linear* AESA

MA Método de Acesso

MAE Métodos de Acesso Espacial – *Spatial Access Methods*

MAENP Métodos de Acesso Espaciais Não-Pontuais

MAEP Métodos de Acesso Espaciais para Dados Pontuais

MAM Métodos de Acesso Métrico – *Metric Access Methods*

MBOR *Minimum Bounding Omni Region*

MBR *Minimum Bounding Rectangle*

MDS *Multi-Dimensional Scaling*

MLP *Multi-Layer Perceptron*

MST *Minimal Spanning Tree*

MVPT *Multi-Vantage-Point Trees*

NGCHL *Neural Gas* com Aprendizado Hebbiano Competitivo

PMAM *Plug-in Module for Access Methods*

RI Recuperação de Informação (*Information Retrieval*)

RIS Recuperação de Informação por Similaridade (*Similarity Information Retrieval*)

RNA Redes Neurais Artificiais

SAM *Spatial Access Methods*

SAT *Spatial Approximation Tree*

SGBD Sistemas de Gerenciamento de Bases de Dados

SL-SOM *Self-Labelled Self-Organizing Map*

SOM *Self-Organizing Maps*

TMBR *Telescopic Minimum Bounding Rectangle*

TS-SL-SOM *Tree-Structured Self-Labelled Self-Organizing Map*

VPF *Vantage-Point Forest – Excluded Middle Vantage-Point Forest*

VPT *Vantage-Point Tree*

VT *Voronoi Tree*

Resumo

A Recuperação de Informação por Similaridade (RIS) é um processo complexo que normalmente envolve bancos de dados volumosos e objetos em altas dimensões. Dois grupos de técnicas são amplamente utilizados para esse fim, Mapas Auto-Organizáveis (SOM – *Self-Organizing Maps*) e Métodos de Acesso (MA).

Os dois grupos de técnicas apresentam limitações. A maioria dos SOM, especialmente os modelos derivados do mapa de Kohonen, utilizam quase que exclusivamente o processamento seqüencial para encontrar a unidade vencedora. Por outro lado, tanto os Métodos de Acesso Espacial (MAE) quanto os Métodos de Acesso Métrico (MAM) não aproveitam o conhecimento gerado por consultas anteriores.

Com o objetivo de resolver esses problemas, duas novas técnicas são propostas nesta tese. A primeira técnica está baseada em SOM e a segunda em MAE e MAM.

Em primeiro lugar, para melhorar o desempenho de sistemas baseados em SOM, propõe-se a incorporação de MAE e MAM durante o treinamento, gerando-se as famílias denominadas SAM-SOM e MAM-SOM.

Em segundo lugar, os MAE e MAM foram aprimorados através da criação do módulo denominado PMAM, que é capaz de aproveitar o conhecimento gerado pelas consultas anteriores. A combinação do módulo PMAM com MAE e MAM deu origem às famílias SAM+ e MAM+, respectivamente.

Como resultado deste trabalho ressalta-se que, tanto a família SAM-SOM quanto a MAM-SOM proporcionam uma melhora considerável em relação aos modelos de SOM tradicionais, os quais normalmente precisam de muito tempo de treinamento.

Por outro lado, as famílias SAM+ e MAM+ têm a capacidade de reduzir, gradualmente, o número de operações necessárias para realizar uma consulta. Isto é possível porque o módulo PMAM permite reaproveitar o conhecimento gerado pelas consultas anteriores.

Abstract

Similarity Information Retrieval (SIR) is a complex process that usually involves large and complex Data Bases. Two groups of techniques are widely used for it, Self-Organizing Maps (SOM) and – Spatial Access Methods (SAM) and Metric Access Methods (MAM).

However, both groups of techniques present important drawbacks. Most of SOM make intensive use of sequential comparison to find winner units. On the other hand, Access Methods do not take advantage of knowledge generated by previous queries.

In order to overcome these problems, two novel techniques are proposed to improve the SIR process. The first technique is based on SOM and the second one on SAM and MAM.

Firstly, SOM was used jointly with SAM and MAM in order to improve SOM based systems, producing two new families of techniques named SAMSOM and MAMSOM respectively.

Secondly, SAM and MAM themselves were improved by the creation of PMAM, a plug-in module which is useful to take advantage of knowledge acquired by previous queries in order to speed up following queries. The combination of PMAM jointly with SAM and MAM produced SAM+ and MAM+ families of Access Methods.

As the main result, SAM-SOM and MAM-SOM families outperform considerably traditional SOM based systems which normally need a long time for training.

Additionally, SAM+ and MAM+ families are capable of reducing gradually the number of operations needed to answer a query, as new queries are introduced. This is possible because PMAM allows them to take advantage of knowledge generated by successive queries.

Capítulo 1

Introdução

1.1 Contexto e Motivação

Alguns anos atrás, os dados armazenados em um banco de dados envolviam apenas tipos simples tais como: números inteiros e de ponto flutuante, cadeias de caracteres (*strings*), etc. Hoje em dia, existe a necessidade de armazenar e recuperar informação em dados mais complexos tais como seqüências de DNA, vídeos, imagens, som, etc. Esses novos tipos de dados são **estruturalmente mais complexos**. Isso significa que podem estar compostos por dados de vários tipos, precisando de maior quantidade de memória e de equipamentos mais sofisticados para o processamento. As instâncias desses tipos de dados são conhecidas como **objetos complexos**.

Pode-se observar que há necessidade de melhorar as técnicas existentes para Recuperação de Informação por Similaridade (RIS) e por conteúdo em Bancos de Dados Multimídia. Entende-se por Recuperação de Informação, a área que estuda a manipulação, representação, armazenamento, organização e acesso a itens de informação (Baeza-Yates e Ribeiro-Neto, 1999).

Os Sistemas de Gerenciamento de Bases de Dados (SGBD) existentes são muito sofisticados, eficientes e rápidos no processamento e recuperação de informação envolvendo dados de tipos simples (números, *strings*, etc), mas ainda apresentam muitas limitações quando o objetivo é procurar informação em campos com dados multimídia (vídeo, som, etc).

Quando os dados pertencem a tipos simples, o tipo de consulta mais comum é conhecido como busca por coincidência exata (*Exact Matching*). Por exemplo: "recuperar o registro cujo código seja 20" ou "recuperar os alunos com idade entre 15 e 20 anos". Para resolver esse tipo de consulta, os campos envolvidos são previamente indexados através de estruturas tais como as árvores B (*B-Tree*) (Comer, 1979). A primeira consulta consiste em procurar, de forma direta na árvore, o valor 20. No caso da segunda consulta, o valor que deve ser

procurado é 15 e, logo após, é iniciado um percurso seqüencial até atingir o valor 20. Nesses casos, o tempo de recuperação de informação, T, está determinado pela Equação (1.1).

$$T = tempo\ de\ E/S\ +\ CPU\ extra \tag{1.1}$$

onde o "tempo de E/S" representa o tempo gasto pelas operações de acesso a dispositivos de acesso secundário e o último fator representa outras operações que a estrutura necessita internamente. Na grande maioria dos casos, o tempo de E/S é o fator que consome mais tempo e, portanto, esse fator é usado para calcular o custo computacional da estrutura.

No caso de dados multimídia é mais útil tentar recuperar informação similar. Por exemplo: "dada a foto de uma pessoa, recuperar as cinco pessoas mais parecidas com a foto fornecida como entrada". Este tipo de consulta não poderia ser respondida com consultas por coincidência exata.

Os tipos de consultas mais adequados para este caso são: **Busca por Abrangência** (*Range Query*) e **k-vizinhos mais próximos** (*k-Nearest Neighbors*).

No caso da busca por abrangência o problema consiste em: dado um objeto O_q e um raio r, deseja-se recuperar todos os objetos que se encontrem dentro desse raio de cobertura.

No segundo caso, dado um objeto O_b, deseja-se recuperar os k-vizinhos mais próximos em relação a esse objeto. O grau de dissimilaridade entre dois objetos pode ser calculado através de uma **função de distância** cuja complexidade depende da natureza dos dados envolvidos. Essa função pode ser computacionalmente simples, como no caso da distância euclideana entre pontos com duas coordenadas (x, y), ou pode ser muito cara tal como calcular a diferença entre duas seqüências de vídeo.

Por isso, o custo computacional apresentado na Equação (1.1) pode ser adaptado para o caso de recuperação de informação por similaridade em bases de dados multimídia. O novo custo computacional é determinado pela seguinte equação.

$$T = NCD \times complexidade\ de\ d()\ +\ tempo\ de\ E/S\ +\ CPU\ extra \tag{1.2}$$

onde NCD é o número de cálculos de distâncias, $d()$ é a função de distância. Os dois últimos fatores são similares ao apresentados na Equação (1.1).

No caso de dados multimídia, o fator que consome mais tempo é a função de distância. Por essa razão, a principal ênfase desta tese está no desenvolvimento de técnicas que proporcionem uma redução considerável nos cálculos de distâncias envolvidas no processo de Recuperação de Informação por Similaridade.

1.2 Definição do Problema

O potencial de duas áreas diferentes é investigado neste trabalho na realização da tarefa de Recuperação de Informação por Similaridade (RIS). Por um lado, encontram-se as técnicas de recuperação de informação em bases de dados multimídia e por outro, encontram-se as Redes Neurais Artificiais (RNA) do tipo Mapa Auto-Organizável ou *Self-Organizing Maps* (SOM).

Ambas as áreas são ativas e existem muitos trabalhos na literatura relativos ao Recuperação de Informação por Similaridade (RIS) (Kohonen, 1997b; Haykin, 1999; Chávez et al., 2001; Baeza-Yates e Ribeiro-Neto, 1999). Ambas as áreas possuem técnicas que apresentam vantagens e limitações na realização dessa tarefa. Assim os problemas que se deseja resolver são os seguintes:

- As técnicas de Redes Neurais existentes, especialmente os SOM, utilizam de forma intensiva a comparação seqüencial para encontrar a unidade vencedora. Considerando que esse processo é repetido para cada padrão ao longo de vários ciclos, o treinamento pode ser muito demorado.
- Os Métodos de Acesso Espacial (MAE) e Métodos de Acesso Métrico (MAM) apresentam ausência de aprendizado. Isso é, se consultas idênticas são realizadas, de forma consecutiva, o número de operações requeridas pelo método, como por exemplo cálculos de distância, para responder a consulta é idêntico para ambas as consultas. As distâncias calculadas relativas à primeira consulta não são utilizadas quando da realização da segunda consulta.

A idéia principal deste trabalho é a de combinar o melhor das Redes Neurais e dos Métodos de Acesso com o intuito de propor-se métodos mais robustos.

1.3 Objetivos

Este trabalho tem dois objetivos principais. O primeiro deles é relacionado ao melhoramento do processo de treinamento de Redes Neurais, especificamente SOM, e o segundo, refere-se ao melhoramento de MAE e MAM, para estes que sejam capazes de adaptar-se ao comportamento do usuário. Em ambos os casos, o que se deseja é melhorar o processo de Recuperação de Informação por Similaridade.

No primeiro caso, relacionado aos SOM, o objetivo é a criação de modelos que evitem a tradicional comparação seqüencial com todas as unidades da rede cada vez que um novo padrão é apresentado.

O segundo objetivo é a criação de uma nova família de métodos de acesso capazes de adaptar-se ao comportamento do usuário e não apenas aos dados. A meta é que o tempo de

resposta seja reduzido de forma gradual em função do conhecimento gerado pelas consultas anteriores.

1.4 Organização do Trabalho e Esclarecimentos

Neste trabalho o termo "método de acesso" é entendido como a junção entre o algoritmo e a estrutura de dados utilizada. Esta tese está organizada da seguinte maneira:

No Capítulo 2 são apresentados os modelos de Redes Neurais Artificiais mais representativos para Recuperação de Informação por Similaridade.

No Capítulo 3 são apresentados os métodos mais representativos de Acesso Espacial e Métrico para Recuperação de Informação por Similaridade.

No Capítulo 4 são discutidas as principais vantagens e limitações das RNA, MAE e MAM em relação ao processo de recuperação de informação por similaridade.

No Capítulo 5 são propostas as novas técnicas SAM-SOM e MAM-SOM que são o resultado da incorporação de Métodos de Acesso Espacial e Métrico no processo de treinamento de Mapas Auto-Organizáveis. Nesse capítulo também são apresentadas algumas possíveis variações, tais como, SAM-SOM híbrida, SAM-SOM$*$ e MAM-SOM$*$.

No Capítulo 6 é proposta a utilização de um novo módulo denominado PMAM, que é capaz de adaptar-se ao comportamento do usuário. O módulo proposto utiliza o conhecimento gerado pelas consultas anteriores para reduzir o tempo de resposta das consultas. O PMAM foi proposto para ser utilizado junto a qualquer Método de Acesso Espacial ou Métrico.

No Capítulo 7 são apresentadas as conclusões do trabalho junto com as propostas de trabalhos futuros.

As principais contribuições desta tese estão concentradas nos capítulos 5-6.

Capítulo 2

Redes Neurais Auto-Organizáveis

O problema de recuperação de informação também é estudado na área das Redes Neurais Artificiais (RNA). Existem vários trabalhos na literatura que utilizam RNA para resolver essa tarefa. Neste capítulo, serão apresentados vários modelos de RNA Auto-Organizáveis que foram propostos para resolver o problema fixando-se atenção no modelo de Kohonen e no modelo *Growing Neural Gas* (GNG), que fornecem a base para as técnicas propostas nesta tese. Antes, porém, da apresentação desses modelos, algumas definições são necessárias para melhor compreensão deste capítulo.

2.1 Definições

Nesta seção são apresentadas várias definições necessárias à compreensão dos diversos modelos que serão apresentados neste capítulo.

Ciclo de treinamento consiste em uma apresentação completa do conjunto de dados para um modelo de rede neural. Esse termo também é sinônimo de época ou *epoch* (Haykin, 1999).

Aprendizado Supervisionado é um paradigma de aprendizado no qual o sistema de aprendizado, neste caso uma RNA, recebe um conjunto de pares ordenados da forma (x_i, c_i), onde x_i representa um dado de entrada e c_i representa a saída desejada para x_i. Neste caso, espera-se que o sistema aprenda a correlacionar corretamente uma determinada entrada com sua correspondente classe (Haykin, 1999).

Aprendizado Não-Supervisionado é um paradigma de aprendizado no qual o sistema de aprendizado, neste caso uma RNA, recebe apenas o conjunto de dados sem especificação da classe à qual cada um deles pertence. Neste caso, o objetivo é que o sistema se auto-organize e detecte os relacionamentos existentes nos dados (Haykin, 1999).

Estabilidade significa que um padrão de entrada sempre deve ser classificado na mesma classe ao longo do processo de treinamento. Plasticidade significa que uma rede deve ser

capaz de incorporar novo conhecimento sem afetar negativamente o que já foi aprendido. O Dilema da Estabilidade-Plasticidade consiste no problema que existe para que um sistema de aprendizado mantenha a estabilidade e a plasticidade simultaneamente (Grossberg, 1976a).

2.2 Mapas Auto-Organizáveis

O cérebro humano é uma das estruturas mais interessantes da fisiologia humana. Mesmo sendo muito complexo do ponto de vista microscópico, ele tem uma estrutura uniforme a escala macroscópica.Os centros responsáveis pelos estímulos visuais (Hubel e Wiesel, 1962, 1977), auditivos (Suga, 1985), etc., são mapeados em diferentes áreas do cérebro que apresentam ordenação topológica. Isto é, as áreas individuais apresentam uma ordenação lógica em relação à sua funcionalidade. Um exemplo conhecido dessa ordenação é o mapa tonotópico das regiões auditivas. Nesse mapa, neurônios próximos entre si respondem a freqüências similares de sons. Também é conhecida a existência do mapa somatotópico que é o mapa dos nervos motores responsáveis por cada parte do corpo humano. Nesse tipo de mapa, regiões fisicamente próximas são responsáveis por membros fisicamente próximos do corpo. Como definição simplificada pode-se dizer que em uma correspondência que respeite a topologia, aquelas unidades que estejam fisicamente próximas umas das outras responderão analogamente a vetores de entrada similares.

A tentativa de criar modelos computacionais com características similares às observadas no cérebro humano, inspirou a criação dos Mapas Auto-Organizáveis (SOM[1]). Um dos SOM mais importantes da literatura é o modelo de Kohonen que é detalhado a seguir.

2.2.1 Modelo de Kohonen

O modelo de Kohonen (1984), também conhecido como mapas de Kohonen, apresenta uma arquitetura relativamente simples composta por uma malha de neurônios conectados de acordo com a Figura 2.1. A mesma idéia, em um contexto biológico, já tinha sido apresentada por Willshaw e von der Malsburg (1976). Nesse trabalho, os autores observaram que um processo envolvendo aprendizado sináptico pode ser responsável pela ordem local de células corticais.

No modelo de Kohonen, o sinal de entrada, representado pelo vetor $\vec{x} = \{x_1, x_2, ..., x_n\}$, está composto por n valores escalares. Ao mesmo tempo, cada unidade da rede SOM tem um vetor de pesos com o mesmo número de componentes que o sinal de entrada. Isto é, $\vec{w} = \{w_1, w_2, ..., w_n\}$, onde w_i representa o peso associado à i-ésima componente do sinal de entrada x_i.

[1]As siglas correspondem ao nome em inglês *Self-Organizing Maps*.

Este tipo de rede é treinado geralmente sob o paradigma não supervisionado, pois não é necessária a apresentação de uma saída desejada para realizar a correção dos erros. O algoritmo utilizado para o processo de aprendizado da rede é o competitivo (Hebb, 1949). Isso significa que as unidades competem entre si para ganhar o direito de se ativar.

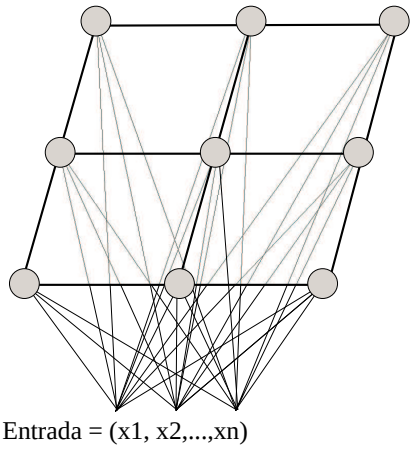


Figura 2.1: Arquitetura Básica de um Mapa Auto-Organizável.

Quando um padrão é apresentado à rede, a diferença entre x e os vetores de pesos, w, de cada neurônio da rede é calculada de acordo com a equação:

$$\|\vec{x} - \vec{w}\| \tag{2.1}$$

Depois de calculada essa diferença, o neurônio vencedor é aquele que apresenta a menor distância, isto é:

$$\|\vec{x} - \vec{w_c}\| = argmin\{\|\vec{x} - \vec{w_i}\|\} \quad \forall i \tag{2.2}$$

onde c é o índice do neurônio vencedor e w_c é o seu correspondente vetor de pesos. Vale a pena ressaltar que a função para medir a distância, entre o vetor de entrada x e o vetor de pesos de um neurônio, deve ser uma função que represente a dissimilaridade entre esses dois vetores. A distância euclideana é a mais utilizada, porém qualquer outra função que possa representar a diferença entre a entrada e o vetor de pesos, poderia ser utilizada.

Nesse contexto, uma vez escolhida a unidade vencedora, seu vetor de pesos e os vetores de pesos dos seus vizinhos mais próximos são atualizados. Existem várias formas de escolher-se a vizinhança sendo que as duas apresentadas na Figura 2.2 são as mais utilizadas. No início do processo de treinamento a vizinhança de neurônios atualizados é relativamente grande, porém ao longo do processo de treinamento, a vizinhança é reduzida de forma gradual (Lo et al., 1991; Ritter et al., 1992). Uma técnica que tem se mostrado efetiva para acelerar o processo de convergência de treinamento de um SOM é reduzir a vizinhança de acordo com uma função Gaussiana (Lo et al., 1991; Erwin et al., 1992a,b).

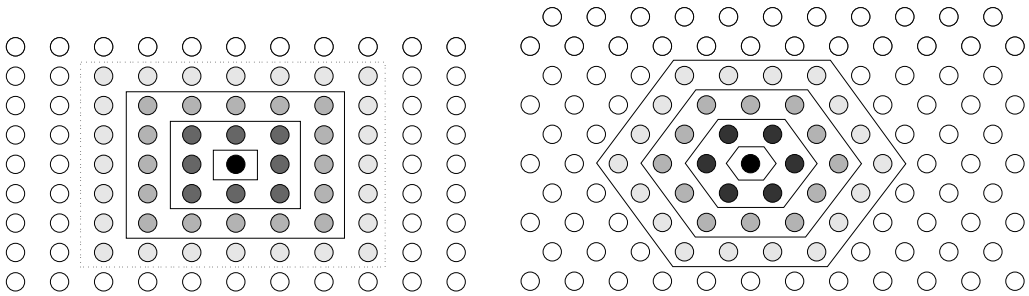


Figura 2.2: Duas configurações de mapa de Kohonen e de níveis de vizinhança do neurônio vencedor.

O processo de treinamento de um SOM cria, de forma natural, agrupamentos ou *clusters*. Cada *cluster* corresponde a um grupo de padrões que compartilham características similares, sendo que o centro corresponde ao padrão que melhor representa aquela classe. Após ter realizado o processo de treinamento de um SOM, espera-se que unidades próximas fisicamente na rede representam padrões similares (Kohonen, 1990).

A distância do neurônio vencedor até o centro do *cluster* é diretamente proporcional ao grau de diferença do padrão sendo apresentado com o padrão representado no centro do *cluster*.

Mesmo sendo fácil visualizar essa similaridade ou proximidade em forma de uma matriz bidimensional, geralmente não é fácil determinar padrões similares em espaços multi-dimensionais (Obermayer et al., 1990; Ritter e Kohonen, 1990).

Após o modelo de Kohonen ter sido proposto, diversos autores estudaram possíveis extensões ou generalizações com o intuito de criar modelos mais eficientes (Kohonen, 1993a; Graepel et al., 1998). Um trabalho interessante que apresenta uma interpretação fisiológica do comportamento do modelo de Kohonen pode ser observada em (Kohonen, 1993b).

Um problema importante das redes de Kohonen é a necessidade de definir a arquitetura da rede, *a priori*, isto é, o número de neurônios que ela deverá ter. No caso de um Banco de Dados, novos padrões podem ser inseridos a qualquer momento, não sendo possível determinar uma arquitetura fixa *a priori*. Isso dificulta a utilização de uma rede de Kohonen da forma original. O principal inconveniente é que esse modelo exige a pré-definição da arquitetura da rede e esta permanecerá fixa até o final do treinamento.

A maioria das deficiências dos mapas auto-organizáveis foram discutidas em Kohonen (1997b) e Bishop et al. (1998). Entre os principais problemas podem-se destacar os seguintes:

- ausência de função de custo;
- ausência de fundamentos teóricos para a escolha da taxa de aprendizado e função de vizinhança;
- ausência de provas gerais de convergência;

- falta de definição de densidade de probabilidade.

Em relação às provas de convergência do modelo de Kohonen, foram realizadas algumas tentativas por várias abordagens (Ritter e Schulten, 1988), incluindo processos de Markov (Bouton et al., 1991; Cottrell et al., 1994) e equações diferenciais ordinárias (Flanagan, 1994, 1996). Porém, a demonstração do processo de ordenação do SOM, só foi possível para o caso unidimensional e um sinal de entrada unidimensional (Kohonen, 1997b; Cottrell, 1997). Outros trabalhos interessantes, em relação às provas de convergência para os mapas de Kohonen, podem ser encontrados em (Cottrell et al., 1995, 1998).

Como já foi mencionado anteriormente, quando da indexação e recuperação de dados multimídia em Bancos de Dados, supõe-se que ocorrerão inserções e remoções a qualquer momento. Essas condições fazem supor que a rede utilizada deve ter a característica de crescer durante o treinamento ou, inclusive, remover aquelas unidades que não sejam mais úteis depois de uma remoção de padrões. Essa característica do problema restringe a utilização do modelo de Kohonen na sua forma original. Por essa razão, outros modelos mais avançados, tais como os hierárquicos e os construtivos, derivados do mapa de Kohonen, serão analisados nas próximas seções.

2.3 Mapas Auto-Organizáveis Construtivos

O modelo básico da Rede de Kohonen não é construtivo, porém, existem outras redes do tipo SOM que incorporam técnicas construtivas e têm a capacidade de mudar a sua arquitetura ao longo do processo de treinamento. Esse ponto já foi discutido por diversos autores tais como Kangas et al. (1989, 1990); Koikkalainen e Oja (1990); Fritzke (1991).

Um algoritmo construtivo tem duas alternativas para gerar uma rede apropriada. A primeira é reconhecer e corrigir aqueles neurônios que tenham sido gerados em posições impróprias para a distribuição dos dados. A segunda é prevenir e corrigir a topologia existente para minimizar o erro.

A busca pela topologia mais apropriada para um determinado conjunto de padrões pode ser um processo computacional muito custoso. As principais técnicas existentes na literatura são analisadas nas próximas seções em ordem cronológica.

2.3.1 Aprendizado Hebbiano Competitivo

Um dos primeiros SOM que incorporou técnicas construtivas foi o método denominado Aprendizado Hebbiano Competitivo – *Competitive Hebbian Learning* (CHL) (Martinetz e Schulten, 1991; Martinetz, 1993). Essa técnica permitiu criar conexões de forma dinâmica ao longo do processo de treinamento. A diferença fundamental em relação ao modelo de

Kohonen clássico é que, depois de encontrar as duas unidades, s_1 e s_2, mais próximas ao padrão apresentado $\vec{\xi}$, o algoritmo verifica a existência de conexão entre eles e, se ela não existir, o algoritmo se encarrega de criá-la.

2.3.2 IGG

O modelo *Incremental Grid-Growing* (IGG) foi divulgado através dos trabalhos apresentados por Blackmore e Miikkulainen (1993); Blackmore (1995); Blackmore e Miikkulainen (1995).

O algoritmo IGG constrói uma rede dinamicamente mudando a estrutura e as conexões de acordo com os dados de entrada. O treinamento é iniciado com uma rede de apenas quatro neurônios conectados formando um quadrado. As novas unidades são sempre criadas nos limites externos do mapa perto daqueles neurônios com maior erro acumulado.

A inserção de novas unidades, apenas nos limites externos do mapa, permite que o algoritmo IGG mantenha sempre uma estrutura 2-d. Mesmo considerando que os padrões apresentados pertençam a dimensões maiores que 2, cada nó mantém duas coordenadas da forma (x, y) que indicam sua posição no mapa. As coordenadas são mantidas com o objetivo de serem úteis para uma possível visualização sem a necessidade de mapear os vetores de pesos.

2.3.3 Neural Gas com CHL

Martinetz e Schulten (1991, 1994) contribuíram para o avanço desta área propondo o algoritmo *Neural Gas* com Aprendizado Hebbiano Competitivo (NGCHL). Esta técnica elimina aquelas conexões que não sejam mais úteis na rede. Tal poda de conexões é possível através da incorporação de um contador em cada aresta (conexão) que controla a "idade" da conexão. Esse valor é acrescentado cada vez que os neurônios que ela conecta são escolhidos como vencedores. Se o contador de uma conexão atinge um certo valor (determinado por um parâmetro), ela é eliminada.

2.3.4 GCS

A técnica conhecida como *Growing Cell Structures* (GCS) foi apresentada no trabalho de Fritzke (1992) e posteriormente estendido em (Fritzke, 1993a, 1994b). O algoritmo é muito parecido com *Growing Neural Gas* (GNG) (ver Seção 2.3.5). A principal diferença, no caso do modelo GCS, é que a rede está limitada a um número máximo de simplexos k-dimensionais, onde k é um número inteiro escolhido antes de iniciar o treinamento.

O bloco mínimo que é acrescentado à rede, que também corresponde à configuração inicial, é um simplexo k-dimensional. Por exemplo, se $k = 1$, o simplexo é uma linha, se $k = 2$ o simplexo é um triângulo, se $k = 3$ o simplexo é um tetraedro.

Os padrões apresentados durante a fase prévia à inserção de um simplexo são úteis para a atualização de pesos e para armazenar informação relativa ao erro de cada unidade. Essa informação é útil para decidir a posição das novas unidades.

Seja w a unidade com o maior erro acumulado e, seja c a maior das conexões de w, uma nova unidade é inserida dividindo-se a conexão c. Depois dessa primeira unidade, o resto do simplexo é completado de tal forma que a rede resultante estará formada, exclusivamente, por simplexos k-dimensionais.

Uma extensão do GCS é o *Dynamic Cell Stuctures – Growing Cell Structures* (DCSGCS) (Bruske e Sommer, 1995). A principal diferença dessa proposta é que a topologia, ao invés de ser pré-definida, é aprendida ao longo do processo de treinamento. O modelo DCSGCS também simplifica as estruturas internas que eram necessárias para manter os simplexos das GCS.

Pelo fato de ser uma rede que se adapta à distribuição dos dados de forma flexível, o modelo GCS supera o desempenho do modelo de Kohonen, como pode ser observado no trabalho de Fritzke (1993b).

2.3.5 Growing Neural Gas

O algoritmo GNG foi proposto por Fritzke em (Fritzke, 1995b, 1994a). Outros trabalhos, como o apresentado por Fritzke (1991), também contribuíram na criação desta técnica.

Através do algoritmo GNG é possível criar e destruir unidades ao longo do treinamento. Esse algoritmo é o resultado da combinação dos métodos *Growing Cell Structures* (Fritzke, 1994b) e o *Competitive Hebbian Learning* (Martinetz e Schulten, 1991). O algoritmo GNG é um importante representante dos modelos SOM construtivos e será apresentado com maior grau de detalhe nesta seção, pois conforme será visto mais adiante, esse modelo será usado na proposta desta tese.

Este modelo surgiu principalmente com o objetivo de melhorar algumas limitações do modelo básico de Kohonen. Enquanto uma rede de Kohonen precisa da definição de uma topologia fixa, uma GNG inicia o seu treinamento com uma arquitetura mínima e novas unidades são criadas gradualmente. Portanto, o modelo GNG, além de trabalhar com o paradigma Não Supervisionado, também é construtivo, sendo capaz de gerar uma topologia diferente para cada tipo de problema.

Uma outra diferença em relação ao modelo de Kohonen é a forma de conectar as unidades. No mapa de Kohonen, as conexões criam apenas malhas retangulares como pode ser visto na Figura 2.1. Já no modelo GNG, uma unidade pode ter muito mais de quatro vizinhos gerando diversas figuras geométricas[2] e uma rede com maior capacidade de aprendizado.

[2]Em duas dimensões, se a rede estiver muito carregada, as unidades tendem a formar triângulos como pode ser visto experimentalmente no trabalho de Fritzke (1997).

O mecanismo para o crescimento de estruturas celulares (Fritzke, 1994b) junto com a geração de topologias utilizando Aprendizado Hebbiano ou Competitivo (Martinetz e Schulten, 1991) foram combinados para criar este novo modelo. O processo de treinamento de uma rede GNG se inicia com apenas dois neurônios e as novas unidades vão sendo inseridas sucessivamente. Para determinar em que posição deverá ser inserida uma nova unidade, informações relacionadas ao erro são coletadas novamente durante o processo de adaptação. Cada nova unidade deverá ser inserida perto da unidade com maior erro. O algoritmo *Growing Neural Gas* é apresentado a seguir[3].

Algoritmo 2.1 Algoritmo de treinamento de uma rede GNG

1: Inicializar a rede A com duas unidades c_1 e c_2

$$A = \{c_1, c_2\} \tag{2.3}$$

Os pesos devem ser inicializados com valores aleatórios, geralmente, no intervalo [0,1]. Inicializar o conjunto de conexões C, C $\subset$ A x A por:

$$C = \emptyset \tag{2.4}$$

2: Apresentar um padrão $\vec{\xi}$[4] à rede de acordo com uma distribuição uniforme $p(\vec{\xi})$

3: Determinar os dois neurônios mais próximos s_1 e s_2 em relação a $\vec{\xi}$ de acordo com as equações (2.5) e (2.6) onde:

$$s_1 = argmin\|\vec{\xi} - w_c\| \quad \forall c \in A \tag{2.5}$$

$$s_2 = argmin\|\vec{\xi} - w_c\| \quad \forall c \in A - \{s_1\} \tag{2.6}$$

onde $\|\vec{\xi} - \vec{w_c}\|$ representa a função de distância (neste caso euclideana) entre os vetores $\vec{\xi}$ e $\vec{w_c}$.

4: Se não existe uma conexão entre s_1 e s_2, então criá-la.

$$C = C \cup \{s_1, s_2\} \tag{2.7}$$

Inicializar a idade desta nova conexão com 0

$$idade_{(s_1,s_2)} = 0 \tag{2.8}$$

5: Adicionar o quadrado da distância entre o neurônio vencedor e o padrão apresentado a uma variável de erro local:

[3] A nomenclatura utilizada corresponde ao artigo original escrito por Fritzke (1997).

[4] $\vec{\xi} \in R^n$, onde n é a dimensão dos padrões que estão sendo apresentados.

$$E_{s_1} = E_{s_1} + \|\vec{\xi} - w_{s_1}\|^2 \tag{2.9}$$

6: Atualizar o vetor de pesos de s_1 e os vetores de pesos dos seus vizinhos de acordo com as seguintes equações:

$$\triangle \vec{w_{s_1}} = \mu_b(\vec{\xi} - \vec{w_{s_1}}) \tag{2.10}$$

$$\triangle \vec{w_i} = \mu_n(\vec{\xi} - \vec{w_i}) \;\; (\forall i \in N_{s_1}) \tag{2.11}$$

onde N_{s_1} é o conjunto de vizinhos topológicos diretos da unidade vencedora s_1, μ_b e μ_n são as taxas de aprendizado para o neurônio vencedor e para os seus vizinhos respectivamente

7: Incrementar a idade de todas as conexões de s_1:

$$idade(s_i, i) = idade(s_i, i) + 1 \;\; \forall i \in N_{s_1}. \tag{2.12}$$

8: Remover as conexões com idade maior que a_{max}[5]. Se depois da remoção existirem unidades sem conexões, elas devem ser removidas da rede.

9: Se o número de padrões apresentados até o momento for múltiplo do parâmetro λ, uma nova unidade deve ser inserida assim:

- Determinar a unidade q com o maior erro acumulado de toda a rede.

$$q = max\{E_c, \; \forall c \in A\} \tag{2.13}$$

- Determinar, dentre os vizinhos de q, a unidade f com maior erro acumulado.

$$f = max\{E_c\} \; \forall c \in N_q \tag{2.14}$$

- Adicionar uma nova unidade r à rede e interpolar seu vetor de pesos a partir de q e f de acordo com a Equação (2.16).

$$A = A \cup r \tag{2.15}$$

$$\vec{w_r} = \frac{\vec{w_q} + \vec{w_f}}{2} \tag{2.16}$$

- Inserir conexões de r até q e de r até f e remover a conexão original entre q e f:

$$C = C \cup \{(r, q), (r, f)\} \tag{2.17}$$

[5] a_{max} é o parâmetro de treinamento que determina a idade máxima permitida para uma conexão.

$$C = C - \{(q, f)\} \tag{2.18}$$

- Diminuir as variáveis de erro das unidades q e f em uma fração α:

$$\triangle E_q = -\alpha E_q, \quad \triangle E_f = -\alpha E_f \tag{2.19}$$

- Interpolar a variável de erro de r a partir de q e f:

$$E_r = \frac{E_q + E_f}{2} \tag{2.20}$$

10: Diminuir a variável de erro de todas as unidades:

$$\triangle E_c = -\beta E_c, \quad \forall c \in A \tag{2.21}$$

onde β é a taxa de correção de erros

11: Se o critério de parada (isto é, o tamanho máximo da rede ou alguma outra medida de desempenho) ainda não foi alcançado, voltar ao passo 2.

De acordo com os experimentos realizados por Fritzke em (Fritzke, 1997), o algoritmo tem alta probabilidade de convergência.

Um outro algoritmo baseado no GNG é o chamado *Plastic Self-Organizing Maps* (PSOM). O crescimento da rede é controlado por um dos parâmetros que é um *threshold*. Se a distância entre o padrão de entrada, $\vec{\xi}$, e a unidade vencedora, s_1, for menor que o *threshold*, a atualização de s_1 e dos seus vizinhos ocorre de forma similar à GNG. Se a distância for maior que o *threshold*, um grupo de unidades é criado e depois conectado a s_1. Todas as conexões também armazenam uma distância que cresce de forma gradual após a apresentação de cada sinal de entrada. O algoritmo também remove as conexões com idade acima de um certo valor e depois elimina aquelas unidades que não apresentem conexões.

2.3.6 GSOM

O *Growing Self-Organizing Maps* (GSOM), apresentado por Alahakoon et al. (1998a,b, 2000), é um outro modelo que permite o crescimento da rede de forma dinâmica.

O modelo GSOM apresenta características similares ao algoritmo IGG (ver Seção 2.3.2). A configuração inicial corresponde a uma rede com apenas quatro unidades formando um retângulo. Quando o algoritmo decide que novas unidades devem ser criadas, a unidade com o maior erro acumulado é procurada e duas unidades são inseridas na direção das setas próximas à unidade escolhida de acordo com a Figura 2.3. A contribuição mais importante, em relação ao algoritmo IGG, é que o GSOM utiliza um método de inicialização dos pesos que

simplifica e reduz a possibilidade de geração de mapas inapropriados em relação à distribuição dos dados.

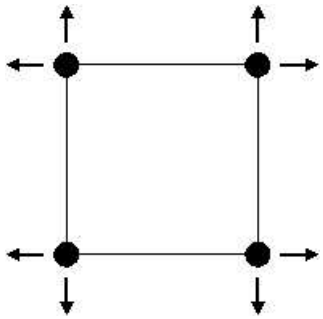


Figura 2.3: Estado inicial de uma rede do tipo *Growing Self-Organizing Maps* (Alahakoon et al., 2000).

A partir dessa configuração inicial, o algoritmo de treinamento da rede GSOM insere novas unidades perto da unidade com o maior erro acumulado. Os vetores de pesos dos novos neurônios são calculados em função dos seus vizinhos mais próximos. Isso é realizado com o intuito de manter a continuidade das posições entre unidades próximas. O resultado é uma rede que tem maior flexibilidade para adaptar-se à distribuição dos dados como pode ser observado na Figura 2.4.

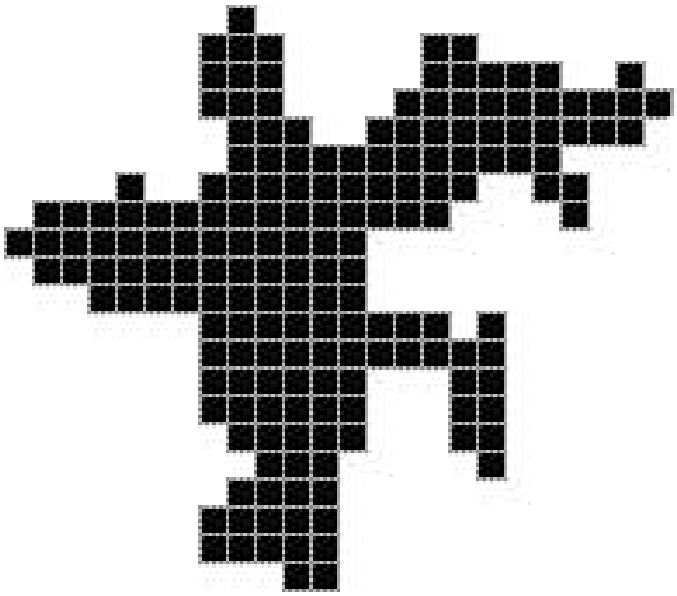


Figura 2.4: Estado final de uma rede do tipo GSOM (Alahakoon et al., 2000).

2.3.7 Mapas Auto-Organizáveis Esféricos

Os Mapas Auto-Organizáveis Esféricos (*Self-Organizing Spherical Maps*) (Boudjemaï et al., 2003), também chamados *Spherical Maps*, são modelos baseados em malhas.

O *Spherical Map* pode ser utilizado principalmente para a reconstrução de objetos volumétricos a partir de uma nuvem de pontos. Algumas redes resultantes podem ser observadas na Figura 2.5.

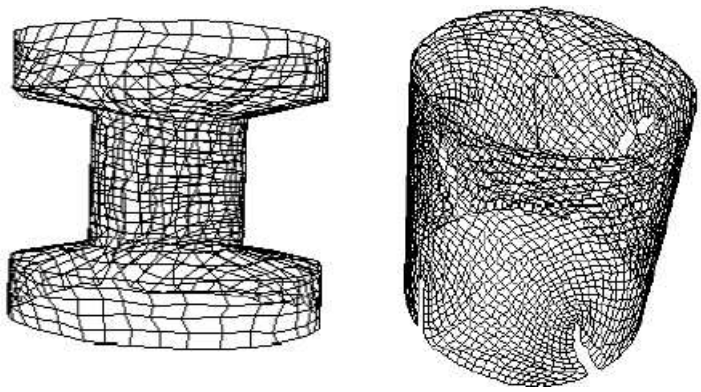


Figura 2.5: Reconstrução de objetos a partir de nuvens de pontos através de *Spherical-Map* (Boudjemaï et al., 2003).

O resultado desse algoritmo é um modelo topológico da superfície original que pode ser utilizado para a reconstrução de objetos volumétricos.

2.4 Mapas Auto-Organizáveis Hierárquicos

Uma alternativa para diminuir a carga de uma rede é organizar os neurônios ou as camadas de um SOM de forma hierárquica. O intuito em todos os casos é de diminuir o tempo de treinamento, tentando atingir um custo computacional de $O(log\ n)$ em termos de cálculos de distância entre os vetores de pesos e o sinal de entrada.

Ao mesmo tempo, tenta-se criar uma rede com maior capacidade de detecção de agrupamentos (Lampinen, 1992; Lampinen e Oja, 1992) e adaptação à distribuição dos dados.

Essas idéias deram origem aos SOM hierárquicos. Nesta seção são detalhados os modelos mais importantes existentes na literatura.

2.4.1 Mapa Hierárquico

Uma das primeiras propostas de redes SOM hierárquicas, refere-se aos mapas hierárquicos (*Hierarchicals Maps*) (Koikkalainen e Oja, 1990). Nesse modelo, vários SOM são criados de forma hierárquica gerando uma estrutura de forma piramidal, como apresentado na Figura 2.6. Os níveis inferiores representam informação mais específica e os superiores mais genérica. O padrão apresentado à entrada da rede é propagado somente para os neurônios filhos da unidade vencedora em cada nível da rede.

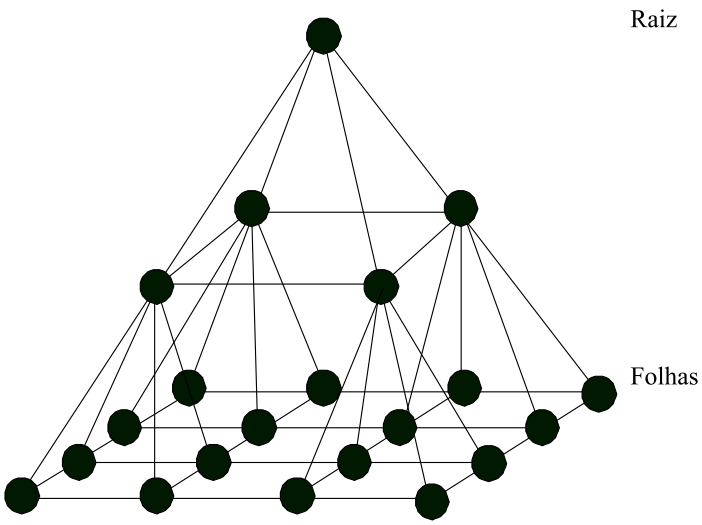


Figura 2.6: Arquitetura dos Mapas Hierárquicos (*Hierarchical* Maps) (Koikkalainen e Oja, 1990).

Segundo os autores, essa estrutura ajuda a reduzir o custo computacional para encontrar a unidade vencedora, de $O(n)$ para $O(log_m(n))$, onde m é o número de níveis e n o número total de unidades. Esse cálculo foi inferido devido ao fato de que, depois de ter encontrado a unidade vencedora, no primeiro nível, o processo continua apenas na sub-rede descendente dessa unidade. Esse processo de descarte é repetido de forma recursiva em todos os subníveis até encontrar a unidade vencedora nos níveis inferiores da pirâmide. Porém, o custo computacional, $O(log_m(n))$, que os autores mencionam, ainda pode ser discutido. O problema é que, dentro das redes individuais em todos os níveis, o processo para encontrar a unidade vencedora ainda continua sendo realizado de forma seqüencial (Koikkalainen e Oja, 1990). Isso significa que, na prática, o número de comparações necessárias é $\sum_{i=1}^{m} n_i$, onde n_i é o número de unidades na i-ésima camada.

2.4.2 HiGS

O modelo *Hierarchical* GCS (HiGS) (Burzevski e Mohan, 1996) cria submapas hierárquicos de forma gradual. Os subníveis são criados de acordo com a complexidade do problema. Na Figura 2.7 é mostrada uma rede gerada utilizando esse algoritmo.

Comparando os HiGS com os mapas hierárquicos apresentados anteriormente, existe uma diferença interessante relacionada à ausência de conexões entre as sub-redes de um mesmo nível. Dependendo da situação, essa característica pode até representar um problema, pois sub-redes próximas também representam grupos que podem apresentar semelhanças. A ausência de conexões entre grupos vizinhos pode representar um problema para executar consultas próximas às regiões de fronteira de uma sub-rede.

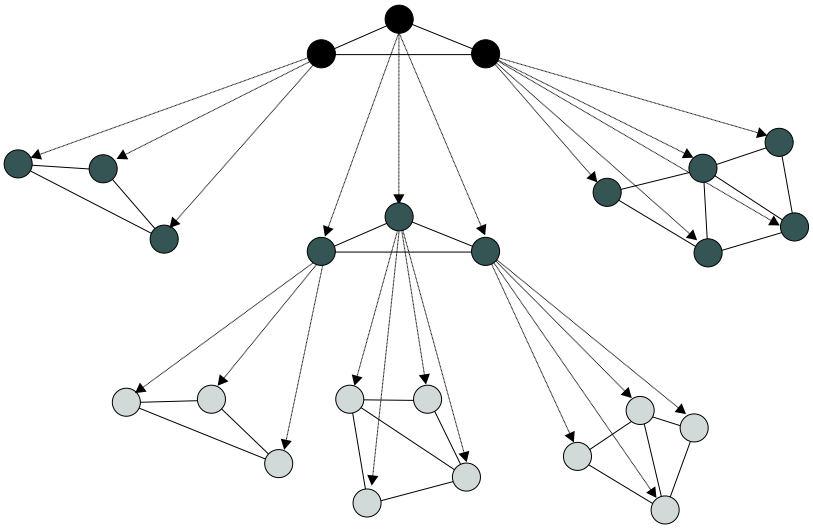


Figura 2.7: *Hierarchical* GCS (HiGS) (Burzevski e Mohan, 1996).

2.4.3 TreeGCS

O modelo *TreeGCS*, apresentado por Hodge e Austin (2001a), explora algumas deficiências da técnica GCS (Fritzke, 1994b) (ver Seção 2.3.4 Pág. 10). A principal contribuição desse modelo é que ele aproveita a remoção de conexões para criar a hierarquia de submapas, como pode ser observado na Figura 2.8.

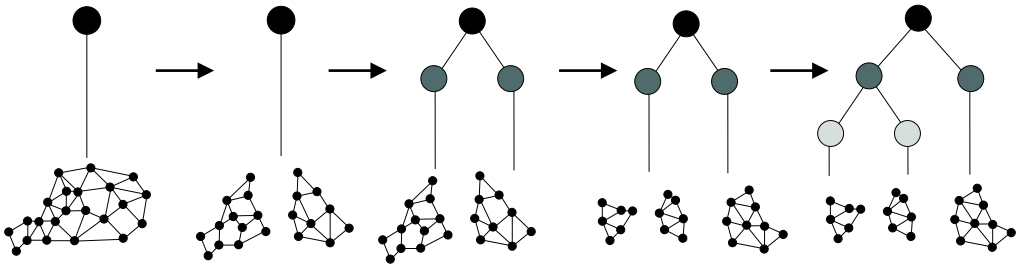


Figura 2.8: Divisão hierárquica segundo o algoritmo *TreeGCS* (Hodge e Austin, 2001a).

Na seqüência de redes apresentada na Figura 2.8, pode-se observar que, cada vez que uma conexão é removida, pode eventualmente acontecer que a rede tenha sido subdividida em dois grupos. Quando isso acontece, é criado um novo nível na rede com os dois subgrupos posicionados um nível abaixo. A arquitetura proposta pelo modelo *TreeGCS* o torna apropriado para diversos problemas, tais como a classificação hierárquica de palavras (Hodge e Austin, 2002), entre outros.

2.4.4 GHSOM

O modelo *Growing Hierarchical Self-Organizing Map* (GHSOM) foi proposto por Dittenbach et al. (2000, 2002). Na Figura 2.9 pode-se observar que esse modelo usa uma estrutura

hierárquica de múltiplas camadas onde cada uma delas é composta por um número independente de SOM. O primeiro nível está composto por apenas um SOM. Para cada unidade, no primeiro mapa SOM, pode ser criada uma sub-rede no próximo nível da hierarquia. Esse princípio é aplicado de forma recursiva aos outros níveis da rede. O modelo GHSOM pode ser útil para problemas como por exemplo a classificação hierárquica de documentos (Dittenbach et al., 2001a,b; Rauber et al., 2002; Rauber e Merkl, 1999). Os níveis superiores representam informação mais genérica, enquanto que os inferiores representam a informação com maior grau de detalhe.

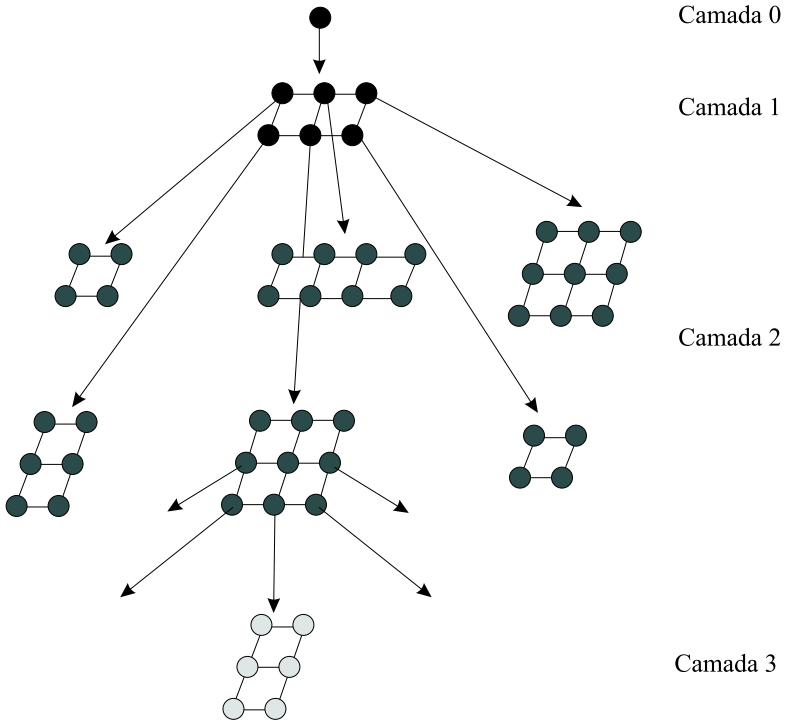


Figura 2.9: Modelo *Growing Hierarchical Self-Organizing Map* (Dittenbach et al., 2000).

Ao mesmo tempo, os SOM utilizados nos diferentes níveis são também construtivos e treinados com o algoritmo similar ao *Growing Grid* (Fritzke, 1995a). A diferença é que GHSOM reduz gradualmente a taxa de aprendizado. Uma outra diferença importante é que a vizinhança também é reduzida de forma gradual.

Para a inserção de novas unidades no GHSOM, é considerado o erro para cada unidade, chamado *Quantization Error* (QE). O QE é a soma das distâncias entre o vetor de pesos de uma unidade, w_1, e todos aqueles padrões que são mapeados nela, isto é, todos aqueles padrões que têm w_1 como sua unidade vencedora. Com esse valor também pode ser calculada a média de erros para cada unidade (*Mean Quantization Error* - MQE). As novas unidades são inseridas entre a unidade com o maior MQE e o seu vizinho mais distante. O processo de inserção adiciona uma fila ou uma coluna inteira na rede. Isso depende da posição das duas unidades entre as quais estão sendo inseridas.

No caso da criação de novas sub-redes são consideradas aquelas unidades cujo QE esteja acima de um certo limite ou *threshold*, τ_1. Esse *threshold* representa, basicamente, o grau de granularidade desejada em função do QE no primeiro nível. Para cada unidade com QE acima de τ_1, um novo submapa é criado. O *threshold* da nova sub-rede é uma fração de τ_1. É necessário considerar que o algoritmo não conduz a uma hierarquia balanceada como visto na Figura 2.9. A profundidade da árvore de SOM gerada é o reflexo da desuniformidade da distribuição dos dados apresentados.

2.4.5 TS-SL-SOM

O modelo *Tree-Structured Self-Labelled Self-Organizing Map* (TS-SL-SOM) (Costa, 1999; Costa e Andrade Netto, 2001) é uma extensão do modelo *Self-Labelled Self-Organizing Map* (SL-SOM) que efetua o particionamento e rotulação automática de um mapa do tipo SOM (Costa, 1999). Como pode ser observado na Figura 2.10, a principal contribuição do modelo TS-SL-SOM em relação ao GHSOM é que o TS-SL-SOM cria submapas a partir de uma região e não apenas de uma única unidade.

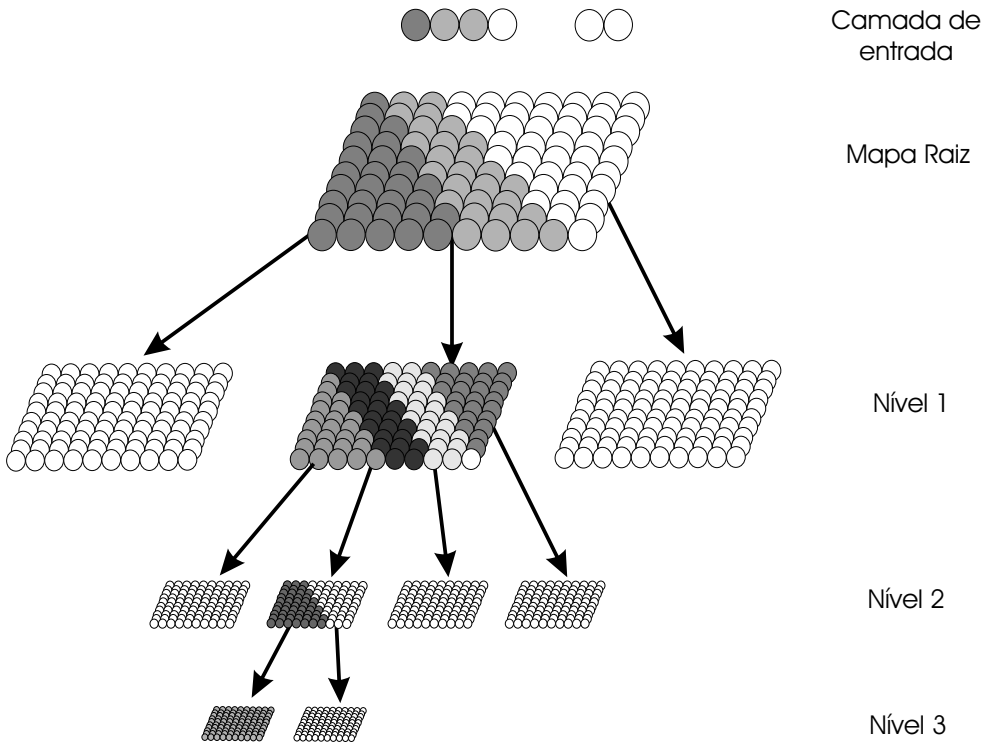


Figura 2.10: *Tree-Structured Self-Labelled Self-Organizing Map*.

Uma outra vantagem de redes do tipo TS-SL-SOM é que não é necessário especificar, *a priori*, o número de sub-redes para um determinado SOM em uma posição da árvore. Além disso, os parâmetros para cada sub-rede são funções da rede 'pai' e do subconjunto de dados que será utilizado para treiná-la. As redes que não apresentem partições são eliminadas.

2.4.6 AHIGG

Baseado no modelo GHSOM, apresentado na Seção 2.4.4, Merkl et al. (2003) propuseram o modelo *Adaptive Hierarchical Incremental Grid Growing* (AHIGG). As camadas individuais da arquitetura hierárquica são variações do modelo IGG (ver Seção 2.3.2).

Na Figura 2.11 observa-se a arquitetura deste modelo. A principal contribuição do modelo AHIGG, em comparação ao GHSOM, é que os mapas individuais podem crescer de forma irregular. Ao mesmo tempo esses mapas podem, eventualmente, remover conexões entre unidades vizinhas. O resultado é que as camadas inferiores representam com maior fidelidade a distribuição dos dados o que facilita a exploração dos dados visualmente.

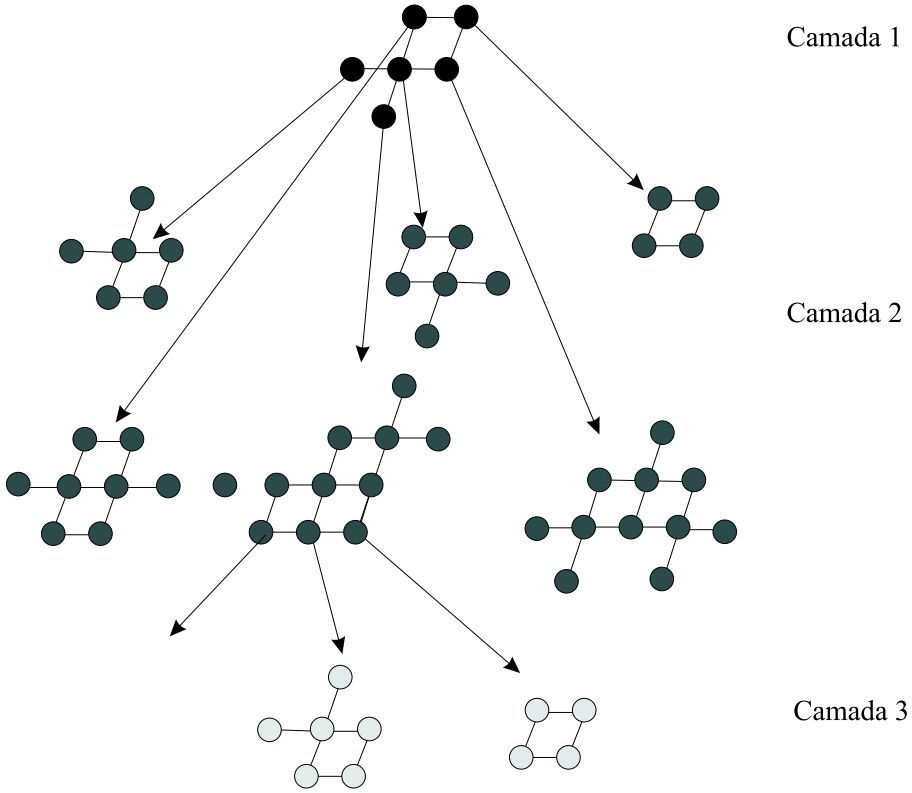


Figura 2.11: Arquitetura de uma rede AHIGG (Merkl et al., 2003).

Existe também um modelo hierárquico especial chamado *Hierarchical Overlapped Neural Gas* (HONG) (Suganthan, 1999). Nesse trabalho os autores propõem um algoritmo onde as regiões, pelas quais as sub-redes são responsáveis, apresentam sobreposição. A idéia pode ser aplicada a qualquer uma das redes hierárquicas apresentadas nas seções anteriores.

Em todas as redes SOM construtivas apresentadas anteriormente, o princípio básico de um mapa de Kohonen é mantido. Quando um padrão é apresentado à entrada da rede, o objetivo é encontrar um neurônio vencedor, sendo que todos eles competem sob as mesmas condições. A idéia fundamental de utilizar-se um SOM hierárquico é criar uma rede capaz de representar com maior grau de detalhe os dados a cada nível. Uma conseqüência da

abordagem hierárquica é diminuir o processamento necessário para encontrar o neurônio vencedor.

2.5 Considerações Finais

No presente capítulo, foi apresentada uma visão geral de redes neurais auto-organizáveis com uma orientação especial aos problemas de recuperação de informação. Diversos modelos de SOM construtivos também foram apresentados destacando-se suas vantagens e limitações.

Em primeiro lugar foi apresentada uma visão geral dos Mapas Auto-Organizáveis e do modelo de Kohonen. Também foram apresentados os mais importantes SOM Construtivos. Todos esses modelos foram propostos com o intuito de melhorar a qualidade da rede resultante e minimizar o erro, o número de neurônios necessários e o tempo de treinamento.

Um outro grupo de redes SOM que foi apresentado neste capítulo refere-se aos Mapas Auto-Organizáveis Hierárquicos. Em todos esses casos, a principal motivação é criar redes que se adaptem à distribuição dos dados e reduzam o tempo de processamento.

Uma limitação importante dos modelos apresentados é a falta de estrutura para poder responder consultas que requeiram maior grau de precisão tais como: k-vizinhos mais próximos e buscas por abrangência (*Range Queries*). Uma grande vantagem das técnicas que utilizam RNA continua sendo a sua capacidade de aprendizado e generalização, mesmo que isso signifique maior quantidade de tempo envolvido.

As maiores coleções de trabalhos relacionados aos SOM podem ser encontradas nos trabalhos de Kaski et al. (1998); Oja et al. (2003); Yamakawa (2003).

Capítulo 3

Métodos de Acesso Espaciais e Métricos

Realizar consultas por similaridade não é uma tarefa simples devido, principalmente, ao tempo de comparação de elementos e ao número elevado de acessos a dispositivos de armazenamento (Baeza-Yates e Ribeiro-Neto, 1999; Chávez et al., 2001). Dados de tipos complexos requerem mais espaço do que dados de tipos convencionais. Isso significa que o número de acessos a disco é maior e, portanto, a estrutura de armazenamento deve ser mais sofisticada. Também é importante considerar que comparar dois objetos complexos pode ser uma operação muito custosa e, conseqüentemente, isto representa uma porcentagem considerável no tempo total da consulta.

Para simplificar o processo de indexação de objetos complexos, são aplicadas técnicas de extração de características que dependem do domínio de dados. Espera-se que as características extraídas representem, de forma confiável, o objeto original. Para indexar características podem-se utilizar os Métodos de Acesso Espacial (MAE) ou Métodos de Acesso Métrico (MAM), os quais serão descritos em detalhes neste capítulo. Antes de iniciar a descrição dos Métodos de Acesso Espaciais e Métricos, algumas definições são necessárias.

3.1 Definições

Nesta seção são apresentadas várias definições necessárias à compreensão dos métodos que serão discutidos neste capítulo.

3.1.1 Grafos e seus elementos

Um **grafo** (ver Figura 3.1) é um conjunto finito de pontos, chamados de **nós**, conectados através de **arestas** (Caldwell, 1995).

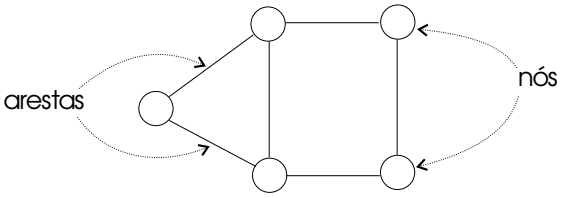


Figura 3.1: Componentes de um grafo.

Um **grafo completo** com n nós é um grafo onde cada um dos nós é conectado com os $n-1$ nós restantes (através de arestas). Isso significa que, para qualquer par de nós distintos (a, b), existe uma aresta que os conecta de forma direta.

Um **grafo direcionado**, ou simplesmente **dígrafo**, é um grafo no qual as arestas são direcionadas. Nesse caso, cada aresta conecta dois nós em apenas um sentido.

3.1.2 Árvores e seus elementos

Uma estrutura de indexação de dados hierárquica, também conhecida como **árvore**, possui certos elementos que são definidos a seguir (ver Figura 3.2). A maior parte das estruturas apresenta um nó que corresponde ao ponto de entrada da árvore, chamado de **raiz**. Quando um nó x aponta para um nó y, x é chamado de **nó pai** e y de **nó filho**. Se y tiver um ou mais filhos, ele é dito **nó interno**. Caso contrário, y é denominado **nó folha**.

O subconjunto composto por y e seus filhos é considerado uma **sub-árvore** de x, da qual y é chamado de raiz.

O número máximo de entradas que um nó pode ter é chamado de **cardinalidade** do nó, por exemplo, a cardinalidade da árvore da Figura 3.2(a) é dois. Nesse caso, a árvore é chamada de **árvore binária** e os filhos são chamados **filho esquerdo** e **filho direito** respectivamente.

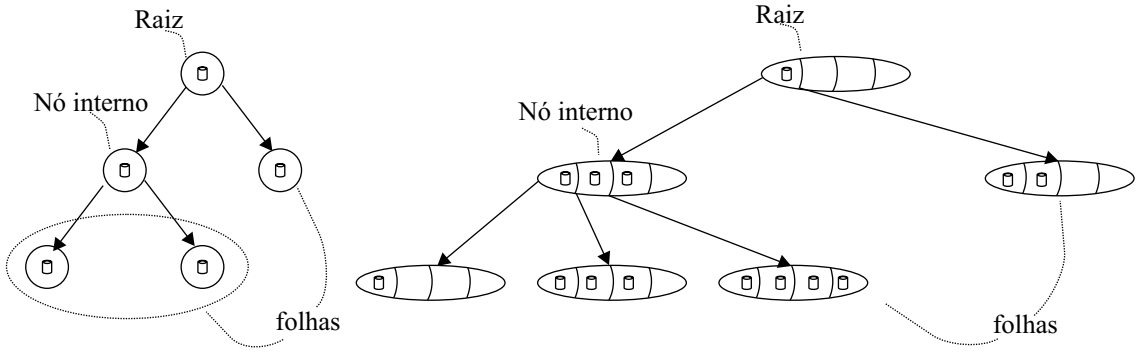
Existem também aquelas árvores em que cada nó contém mais de um elemento, chamados de **entradas da página ou nó**. Nesse caso, a árvore é chamada de **árvore multivias** ou **árvore n-ária**.

O número máximo de ligações de saída que um nó pode ter é chamado de **grau**, ou *fanout*. O grau de uma árvore multivias pode ser igual à cardinalidade do nó, ou igual à cardinalidade do nó acrescido de uma unidade.

Caminho é definido como o conjunto formado por uma seqüência de nós e arestas que precisam ser percorridos para se sair de um nó origem e se chegar a um nó destino.

O **tamanho** de um caminho é dado pelo número de nós que o compõe. A **altura** de uma árvore é definida como o tamanho do maior caminho que pode ser feito a partir da raiz.

Os nós que se encontram à mesma distância (com mesmo tamanho de caminho) com relação à raiz, são considerados no **mesmo nível da árvore**. Todos esses conceitos podem ser melhor entendidos através da Figura 3.2.



(a) Árvore binária (b) Árvore n-ária

Figura 3.2: Elementos de uma árvore.

3.1.3 Espaço Métrico

Considerando que $\mathbb{S}$ é o domínio ou universo de objetos possíveis ou válidos. O subconjunto finito $S \subseteq \mathbb{S}$ representa o conjunto de objetos onde as consultas são efetuadas. O número de elementos n de S é denotado por $n = |S|$. O conjunto S poderia ser considerado como um dicionário, banco de dados ou, simplesmente, conjunto de objetos ou elementos.

Além do conjunto S, considera-se também a existência de uma função de distância métrica chamada d(). A função d():

$$d : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R} \tag{3.1}$$

corresponde à medida de "distância" entre dois objetos. Isso significa que quanto menor a distância entre dois objetos, mais semelhantes eles são. As funções de distância devem cumprir com as seguintes propriedades:

(p1)	$\forall x, y \in S, d(x, y) \geq 0$	positiva
(p2)	$\forall x, y \in S, d(x, y) = d(y, x)$	simétrica
(p3)	$\forall x \in S, d(x, x) = 0$	reflexiva
	e, em alguns casos,	
(p4)	$\forall x, y \in S, x \neq y \Rightarrow d(x, y) > 0$	estritamente positiva
(p5)	$\forall x, y, z \in S, d(x, y) \leq d(x, z) \ + \ d(z, y)$	desigualdade triangular

As propriedades (p1)-(p4) acima asseguram apenas uma definição consistente de uma função de distância. Porém, se a função $d()$ satisfizer (p5), o espaço $\mathbb{M}$ definido pelo par ($\boldsymbol{S}$,

d()) é chamado **Espaço Métrico** e a função distância é chamada **Função de Distância Métrica**. O caso particular em que a métrica não atende à propriedade (p4) é chamado de **Espaço Semi-Métrico** ou **Pseudo-Métrico**.

Um exemplo de espaço métrico é o formado pelo conjunto de palavras da língua portuguesa e pela função de distância métrica de Levenshtein ou $L_{Edit}(x, y)$ (Levenshtein, 1966; Sankoff e Kruskal, 1983). Essa métrica retorna a quantidade mínima de letras que precisam ser substituídas, removidas ou inseridas em x para convertê-la em y. Por exemplo, $L_{Edit}(mesa, velas) = 3$.

3.1.4 Espaço Vetorial

Para entender melhor o conceito de espaço vetorial, considera-se a seguinte notação:

K, é o corpo de valores escalares;

a, b, c ou k, os elementos de K;

V, o espaço vetorial dado;

$\vec{u}$, $\vec{v}$, $\vec{w}$, os elementos de V

Seja K um corpo arbitrário e seja V um conjunto não vazio com operações de adição e multiplicação por escalar que determinam, para qualquer $\vec{u}$, $\vec{v} \in V$, uma soma $\vec{u} + \vec{v} \in V$ e, para qualquer $\vec{u} \in V, k \in K$, um produto $k\vec{u} \in V$. Então, V é chamado **Espaço Vetorial** sobre K (e os elementos de V são chamados vetores) se os seguintes axiomas são verdadeiros:

[A1] : Para quaisquer vetores $\vec{u}, \vec{v}, \vec{w} \in V$, $(\vec{u} + \vec{v}) + \vec{w} = \vec{u}(\vec{v} + \vec{w})$.

[A2] : Existe um vetor em V, denotado por 0 e chamado vetor nulo tal que, para qualquer $\vec{u} \in V$, $\vec{u} + 0 = \vec{u}$.

[A3] : Para cada vetor $\vec{u} \in V$, existe um vetor em V, denotado $-\vec{u}$, para o qual $\vec{u}+(-\vec{u}) = 0$.

[A4] : Para quaisquer vetores $\vec{u}, \vec{v} \in V$, $\vec{u} + \vec{v} = \vec{v} + \vec{u}$.

[A5] : Para qualquer escalar $k \in K$ e quaisquer vetores $\vec{u}, \vec{v} \in V$, $k(\vec{u} + \vec{v}) = k\vec{u} + k\vec{v}$.

[A6] : Para quaisquer escalares $a, b \in K$ e qualquer vetor $\vec{u} \in V$, $(a + b)\vec{u} = a\vec{u} + b\vec{u}$.

[A7] : Para quaisquer escalares $a, b \in K$ e para qualquer vetor $\vec{u} \in V$, $(ab)\vec{u} = a(b\vec{u})$.

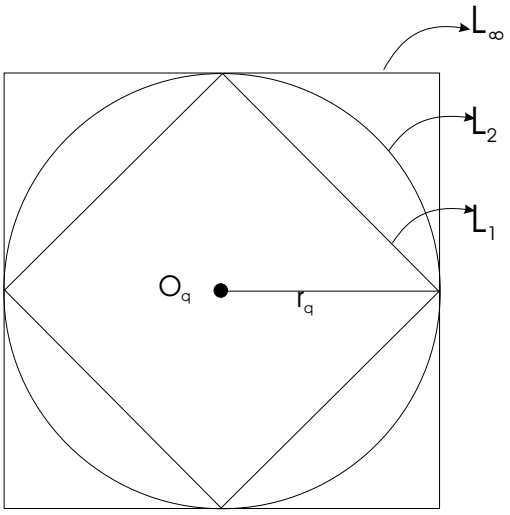
[A8] : $1\vec{u} = \vec{u}$, para qualquer vetor $\vec{u} \in V$.

Um espaço vetorial normado é um espaço vetorial que possui uma métrica definida. Se a métrica definida cumpre com as condições necessárias para ser chamada função de distância métrica (ver Seção 3.1.3). O espaço vetorial junto com a função de distância podem formar um espaço métrico.

Por causa disso, o espaço vetorial é um caso particular de um Espaço Métrico $\mathbb{M} = (\boldsymbol{S}, d())$, onde os elementos de S são t-uplas numéricas da forma (x_1, ..., x_t) e para o qual foi definida uma função de distância métrica. O Espaço Vetorial composto por vetores com n componentes também pode ser chamado espaço n-dimensional. Várias métricas podem ser aplicadas para comparar elementos desse conjunto, sendo que as mais conhecidas são as da família L_s (ou Minkowski), definidas pela equação:

$$L_s((x_1, \ldots, x_n), (y_1, \ldots, y_n)) = (\sum_{i=1}^{n} |x_i - y_i|^s)^{1/s} \tag{3.2}$$

A Figura 3.3 ilustra casos diferentes da família L_s, que é formada pelo conjunto de pontos que estão à mesma distância, r_q, a partir de um centro o_q. No exemplo, a métrica L_1, também conhecida como **Bloco** ou **Manhattan**, corresponde ao somatório do módulo das diferenças entre as coordenadas. A métrica L_2, mais conhecida como **distância euclideana**, é a função mais usada para distância entre vetores. O conjunto de pontos de mesma distância para L_2 forma uma circunferência.



(a)

Figura 3.3: Representação dos pontos situados à distância r_q a partir do objeto o_q, considerando as diferentes métricas da família L_s.

Calculando-se o limite de L_s quando s tende ao infinito, obtém-se a métrica L_∞ apresentada na Equação (3.3), onde o conjunto de pontos com mesma distância formam um quadrado.

$$L_\infty((x_1, \ldots, x_k), (y_1, \ldots, y_k)) = max|x_i - y_i| \ \forall i \ / \ 1 \leq i \leq k \tag{3.3}$$

A família L_s é vasta e outras variantes podem ser obtidas a partir da Equação (3.2).

3.2 Métodos de Acesso para Dados unidimensionais

Quando os dados são de tipos simples, o critério básico para a inserção em uma estrutura de dados é a comparação entre eles. Dessa forma é possível definir uma relação de ordem. O exemplo mais simples poderia ser uma árvore binária onde os elementos menores sempre são inseridos à esquerda e os elementos maiores à direita. Na Figura 3.4, pode-se observar a posição onde foi inserida uma nova chave em uma árvore que tem o elemento s na raiz. Os elementos menores ou iguais a s são inseridos na sub-árvore da esquerda e os maiores na sub-árvore da direita.

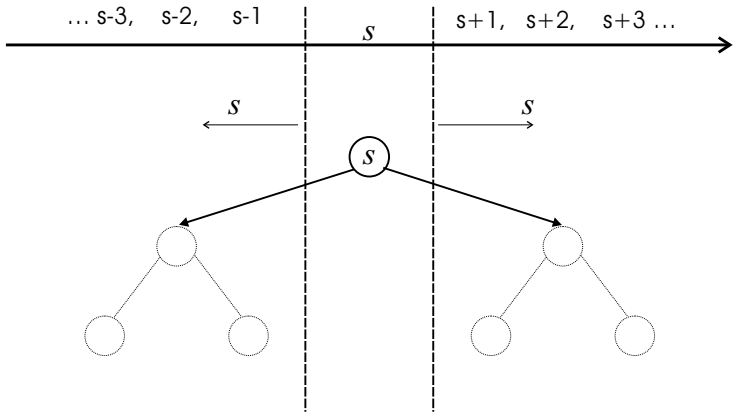


Figura 3.4: Critério de inserção em uma árvore binária simples.

Cada vez que uma nova chave x for inserida, x é comparada com a chave contida na raiz. Se for maior, o procedimento é repetido recursivamente com o nó da direita e, se for menor ou igual, com o da esquerda, até alcançar um nó com o ponteiro correspondente vazio.

Esse tipo de estrutura é apropriado para realizar consultas por coincidência exata (*Exact Matching*). Quando uma consulta desse tipo é solicitada, a chave deve ser comparada com a raiz e, dependendo da relação de ordem com ela (maior ou menor), o procedimento será repetido até encontrar a chave solicitada ou alcançar um nó folha. A grande vantagem desse tipo de estrutura é que depois de comparar um elemento com a chave da raiz, só uma sub-árvore será a escolhida para repetir o procedimento. Isso significa que para cada pergunta realizada o conjunto de possíveis elementos da resposta é reduzido pela metade.

A mais conhecida dessas estruturas é a árvore B (*B-Tree*) (Comer, 1979), considerada como um padrão e base para diversos outros métodos que a sucederam. Mais detalhes em

relação a essa e outras estruturas para indexação podem ser observados em (Folk et al., 1998). A grande vantagem das *B-Tree* é que são apropriadas para grandes volumes de dados e armazenamento em memória secundária (disco). No caso de armazenamento em disco, cada página lógica corresponde, na maioria das vezes, a uma página física no disco.

Uma *B-Tree* difere da árvore binária convencional pois é construída de forma ascendente. No caso da inserção ocorrer em uma página que não tem espaço disponível, a mesma é dividida em duas novas páginas e o elemento central, após a inserção na nova chave, é transferido à página pai. Esse processo poderia ser repetido se a página pai também estivesse totalmente ocupada. O algoritmo de construção garante que na sub-árvore esquerda de um nó nunca existirá uma chave maior que aquela contida na entrada e que na sub-árvore direita nunca existirá uma chave menor. Dessa forma, o conjunto de dados é dividido em subconjuntos disjuntos. Nesse tipo de árvore, a intersecção de sub-árvores do mesmo nível sempre é vazia. Na Figura 3.5 observa-se um exemplo desse tipo de árvore.

A principal vantagem de uma *B-Tree* em relação a uma árvore binária é que ela sempre está balanceada. Isso significa que todos os nós folhas estão à mesma altura, permitindo que a recuperação de informação seja mais rápida que a realizada por meio de uma árvore binária tradicional. Uma outra diferença importante é que, como os nós contêm mais de uma chave, um nó lógico poderia ser armazenado formando uma página física no disco, reduzindo os acessos à memória secundária. O algoritmo de inserção de uma *B-Tree* garante que cada nó tenha, no mínimo, 50% de utilização, com exceção da raiz. Depois de comparar com uma chave, é possível escolher uma única sub-árvore e podar (descartar) completamente o resto delas. Além de podar sub-árvores, também é possível reduzir as comparações com chaves do mesmo nó.

Em relação às consultas por similaridade, basta identificar as sub-árvores que se encontram dentro da região de busca $RQ(O_q)$ como apresentado na Figura 3.5. Nessa figura, pode-se observar que as sub-árvores examinadas são apenas as que apresentam intersecção com a região de busca. No exemplo apresentado, o centro da consulta é o número 67 com um raio de 10 unidades.

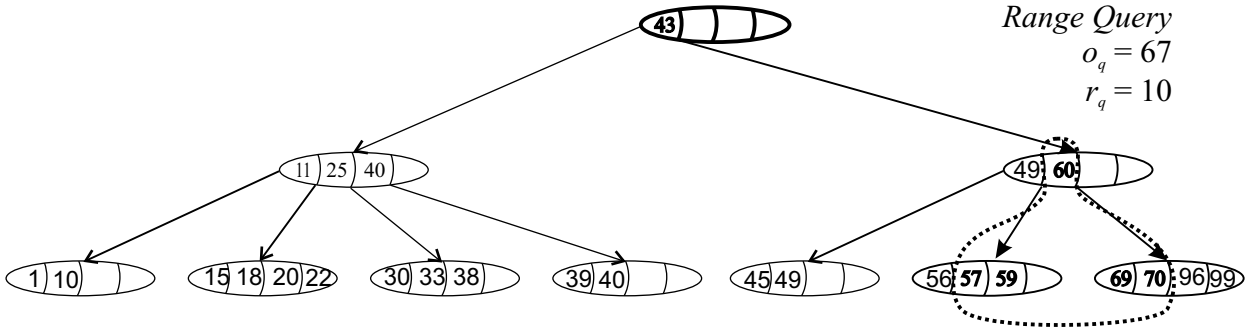


Figura 3.5: *Range Query* em uma árvore n-ária.

No caso de consultas do tipo k-vizinhos mais próximos, o algoritmo é inicializado como uma consulta por abrangência onde o raio diminui à medida em que é encontrado um vizinho mais próximo do que os existentes. Ou seja, inicialmente, o raio de busca é considerado muito grande ∞ e os k primeiros objetos do caminho que se está percorrendo são considerados o conjunto de candidatos. Nesse momento, o raio passa a ser a distância entre o objeto de busca o_q e o mais distante dentre os que estão na lista. Para que o raio seja reduzido mais rapidamente, diminuindo o número de comparações, é usual que o primeiro caminho a ser percorrido seja o que leva diretamente à região onde o objeto de busca se encontra.

Cada vez que um objeto mais próximo for encontrado, este deve ser inserido no conjunto de candidatos e o objeto mais distante deve ser removido. O novo raio da região de busca será a distância de o_q para o atual mais distante (que não necessariamente é o novo). O mesmo procedimento é repetido até que nenhuma outra sub-árvore faça intersecção com o raio de busca.

Os métodos descritos anteriormente são muito úteis para dados simples, porém, para indexação de outros tipos de dados, tais como vetores, essas técnicas não são as mais apropriadas. Para esses tipos de dados existem os métodos de acesso espacial que são detalhados a seguir.

3.3 Métodos de Acesso Espacial

Os Métodos de Acesso Espacial (MAE) são técnicas para indexar e recuperar informação em conjuntos de dados vetoriais.

Uma das primeiras técnicas a serem propostas na literatura para a indexação de dados espaciais foi *Space Filling Curve*. O primeiro trabalho foi o *Peano Curve* de Morton (1966), que também é discutido no trabalho de Gaede e Günther (1998). Resumos desses métodos podem ser encontrados em (Samet, 1995; Gaede e Günther, 1998; Böhm et al., 2001).

Como o objetivo é indexar pontos da forma $(x_1, x_2, ..., x_n)$, uma estrutura do tipo *B-Tree* não seria a mais adequada, pois nessa estrutura apenas podem ser inseridos objetos simples.

Os exemplos apresentados nesta monografia utilizam, com fins didáticos, pontos em duas dimensões, porém as idéias podem ser generalizadas para mais dimensões.

3.3.1 Métodos de Acesso para Dados Pontuais

Os chamados Métodos de Acesso Espaciais para Dados Pontuais (MAEP), têm sido usados para indexar dados vetoriais. A seguir, alguns dos MAEP mais conhecidos na literatura serão apresentados.

PointQuad-Tree

Cada ponto armazenado em uma estrutura do tipo *PointQuad-Tree* (Klinger, 1971; Finkel e Bentley, 1974) divide o espaço em quatro partes chamadas **quadrantes** (ver Figura 3.6). Como conseqüência desse critério, cada nó possui quatro sub-árvores, uma para armazenar a informação relacionada a cada quadrante (ver Figura 3.7(b)). A forma de definir a posição de um novo vetor (x_1, x_2) considerando que a raiz é o ponto (r_1, r_2) seria:

I quadrante NE (nordeste): NE se $x_1 \geq r_1$ *e* $x_2 \geq r_2$;

II quadrante NW (noroeste): NW se $x_1 \geq r_1$ *e* $x_2 < r_2$;

III quadrante SW (sudoeste): SW se $x_1 < r_1$ *e* $x_2 < r_2$;

IV quadrante SE (sudeste): SE se $x_1 < r_1$ *e* $x_2 \geq r_2$.

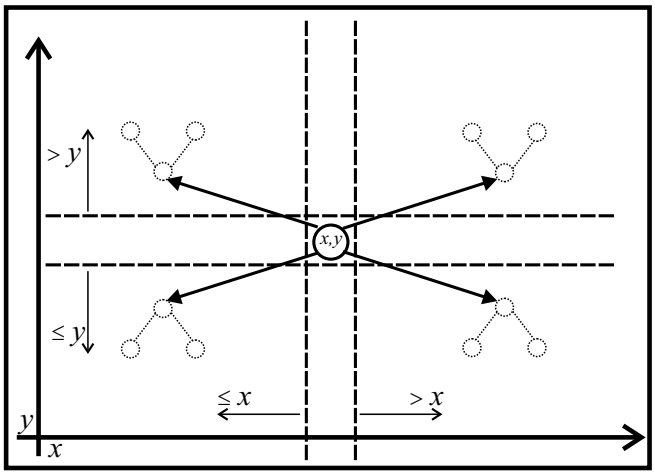
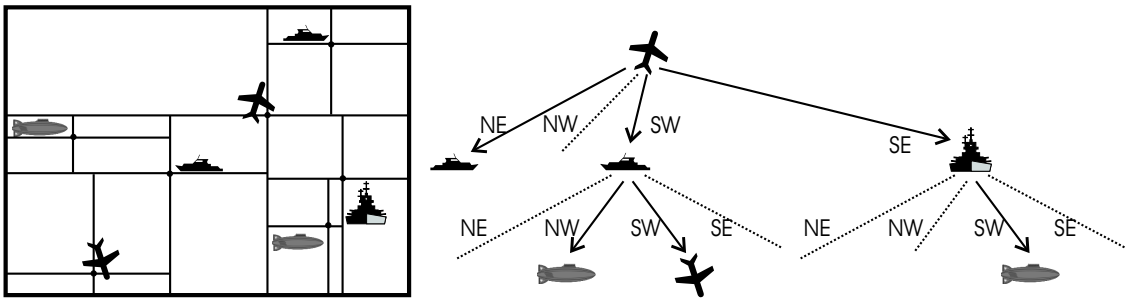


Figura 3.6: Divisão do espaço em quadrantes utilizando *PointQuad-Tree*.

Uma estrutura *PointQuad-Tree* pode ser utilizada para indexar pontos da forma $(x,\ y)$ ou para indexar objetos bidimensionais, como aviões ou navios, representados pelo seu centro $(x,\ y)$. Na Figura 3.7(a) pode-se observar objetos que são indexados utilizando seu respectivo centro $(x,\ y)$.



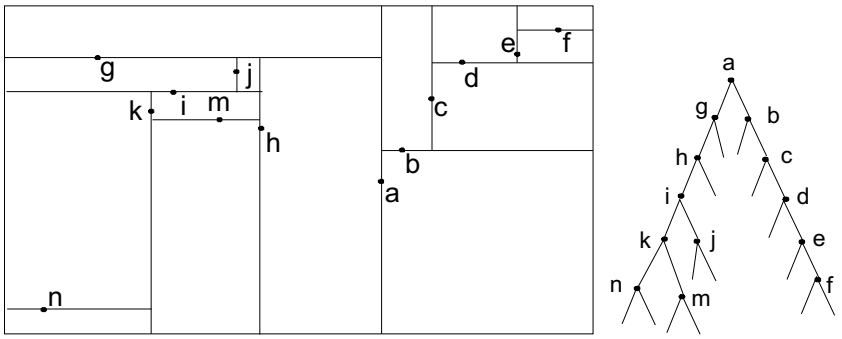
(a) Representação espacial (b) Representação esquemática

Figura 3.7: Estrutura de uma árvore *PointQuad-Tree*.

O critério definido anteriormente permite inserir um novo ponto na estrutura. Uma outra versão da mesma estrutura para pontos 3-d é chamada de *Oct-Tree*. Nesse caso, o espaço é dividido em oito **octantes**.

k-d-Tree

Uma estrutura que também permite a inserção de pontos da forma $(x,\ y)$ é conhecida como *2-d-Tree* e foi apresentada por Bentley (1975). Essa estrutura utiliza uma coordenada de cada vez para cada nível da árvore. Para indexar um ponto da forma $(x,\ y)$, o valor de x é utilizado para decidir, no primeiro nível, se o ponto deve ser inserido à direita ou à esquerda da raiz. Já no segundo nível é utilizada a segunda coordenada y com o mesmo critério. No caso de pontos com duas coordenadas, o processo é repetido a partir do terceiro nível com x, no quarto nível com y e assim em diante, como observado na Figura 3.8.



(a) Representação espacial

(b) Representação esquemática

Figura 3.8: Estrutura de uma árvore *2-d-Tree*.

A generalização das estruturas *2-d-Tree* para n dimensões deu origem às estruturas conhecidas com o nome de *k-d-Tree* (Bentley, 1975). Nesse caso, os pontos são da forma $(x_1, x_2, ..., x_k)$. Para decidir o caminho correto para o novo ponto no nível i $(1 \leq i \leq k)$ é utilizada a coordenada x_i. De forma similar às estruturas *2-d-Tree*, no nível $k+1$ a coordenada utilizada novamente será a x_1, para o nível $k+2$, a x_2, e assim sucessivamente. Cada coordenada define um hiperplano que particiona o espaço em duas partes. O processo é repetido até atingir um nó com espaço disponível para inserir o objeto. Duas restrições das *k-d-Tree* são que ela é voltada para memória principal e que a árvore criada não está balanceada.

k-d-B-Tree

Tentando resolver o problema de acesso à memória secundária (disco), Robinson (1981) criou a estrutura *k-d-B-Tree*. Esta estrutura é basicamente uma combinação de *k-d-Tree* e *B-Tree*. Cada nó possui um *Minimum Bounding Rectangle* (MBR) que envolve todas

as suas entradas (ou sub-árvores). Cada entrada de um nó interno possui a coordenada do hiperplano de particionamento e um ponteiro para a sub-árvore. Cada entrada dos nós folhas contém o objeto indexado e seu identificador - OId. Na Figura 3.9 pode-se observar uma *k-d-B-Tree* para pontos 2-d.

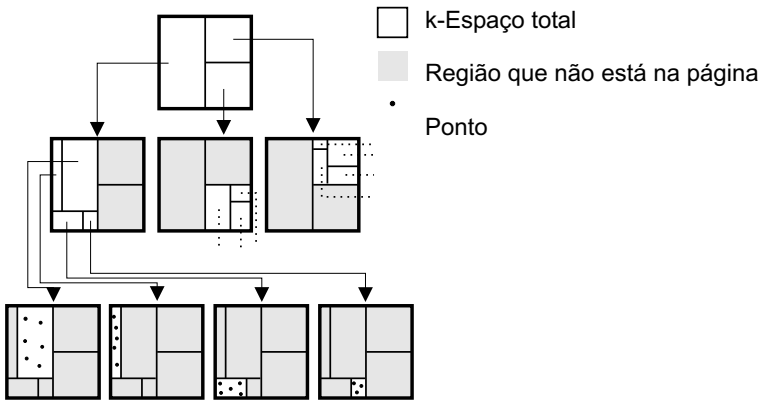


Figura 3.9: Exemplo de uma estrutura 2-d-B-Tree.

Uma das principais limitações que este método apresenta é que o particionamento de um nó pode ser propagado tanto para árvore acima, quanto para árvore abaixo. Isso significa que não existe garantia nenhuma em relação à taxa de ocupação mínima de um nó (Gaede e Günther, 1998), podendo ser criados nós vazios.

Além dessas estruturas, existem muitas outras variações que foram criadas com o intuito de melhorar o desempenho das existentes. Uma dessas estruturas é a denominada *Hybrid-Tree* (Chakrabarti e Mehrotra, 1999) que é uma adaptação da *k-d-B-Tree*. A principal vantagem apresentada é permitir a criação de regiões sobrepostas (não disjuntas). Essa modificação evita a propagação do particionamento para as sub-árvores. A estrutura propriamente dita é semelhante à *k-d-B-Tree* e, de forma análoga, gera uma árvore balanceada.

Todas as estruturas mencionadas anteriormente apresentam limitações quando os dados pertencem a dimensões mais altas. Esse problema é denominado a "maldição da dimensionalidade". Para superar essa limitação podem ser utilizados métodos para reduzir a dimensão dos dados tais como: *Multi-Dimensional Scaling* (MDS) (Torgerson, 1952), *Nonlinear Mapping* (NLM) (J.W.Sammon, 1969), a transformada de *Karhunen-Loève* ('K-L') (Duda e Hart, 1973; Fukunaga, 1990), Decomposição de Valor Singular (SVD– *Singular Value Decomposition*) (Strang, 1980; Press et al., 1988; Golub e Loan, 1989) ou o *FastMap* (Faloutsos e Lin, 1994; Faloutsos et al., 2003).

Para diminuir os efeitos desse problema, foi criada a árvore *TV-Tree* (*Telescopic-Vector*) (Lin et al., 1994), que usa subconjuntos de coordenadas dos vetores em nós internos, armazenando o vetor completo apenas nas folhas. A idéia é contrair e estender dinamicamente os vetores, adicionando ou retirando coordenadas com o intuito de aumentar o poder discriminatório.

O nome desta estrutura, *Telescopic-Vector*, é uma analogia com o movimento de contrair e estender um telescópio. Um exemplo dessa função seria, simplesmente, descartar algumas coordenadas e considerar apenas as primeiras. Estas estruturas são semelhantes às *R-Tree* (ver Seção 3.3.2), com uma adaptação do MBR para organizar os vetores telescópicos, criando assim o *Telescopic Minimum Bounding Rectangle* (TMBR).

A idéia principal da criação de todas essas estruturas é manter os dados organizados para que, dada uma consulta, seja possível descartar todas aquelas sub-árvores com as quais não exista intersecção com a região de busca.

3.3.2 Métodos de Indexação para Dados Não-Pontuais

Existem outros tipos de dados tais como: triângulos, quadrados, retângulos que não podem ser indexados como pontos isolados, pois são formados por mais de um ponto. Nesse caso não é difícil perceber que os MAEP não seriam mais adequados. Para que isso fosse possível, os objetos teriam que ser "quebrados" e uma estrutura secundária teria que ser utilizada para conectar os pedaços. Esse tipo de problema é tratado através dos Métodos de Acesso Espaciais Não-Pontuais (MAENP) tais como as árvores *R-Tree*, que são apresentadas a seguir.

R-Tree

A estrutura *R-Tree* (Guttman, 1984) foi a primeira proposta, para dados não pontuais, existente na literatura. A estrutura *R-Tree* é amplamente difundida nesta área e isso pode ser comprovado porque, quase sempre, é a base de comparação para novas estruturas deste tipo. Uma árvore *R-Tree* é uma adaptação das árvores *B-Tree* (Comer, 1979) para dados multidimensionais não pontuais. O algoritmo da *R-Tree* forma uma árvore com múltiplas vias balanceada em relação à altura, como observado na Figura 3.10.

Na Figura 3.11 observa-se a representação esquemática dos hiper-retângulos apresentados na Figura 3.10. A informação relacionada aos elementos inseridos na árvore encontra-se nas folhas. Nessa figura observa-se que cada nó interno, contém um hiper-retângulo, denominado MBR. O MBR de um nó é o menor hiper-retângulo necessário para abranger todos objetos representados pelas sub-árvores daquele nó.

Uma importante diferença apresentada pela *R-Tree* é que o particionamento do espaço não gera regiões disjuntas, isto é, a intersecção das regiões do espaço, representadas por duas sub-árvores, de um mesmo nó, pode não ser vazia.

No caso de objetos como triângulos, quadrados ou qualquer outro polígono, o que se armazena é o código que identifica o objeto (*Object id* - OId) e o retângulo mínimo que o envolve (MBR).

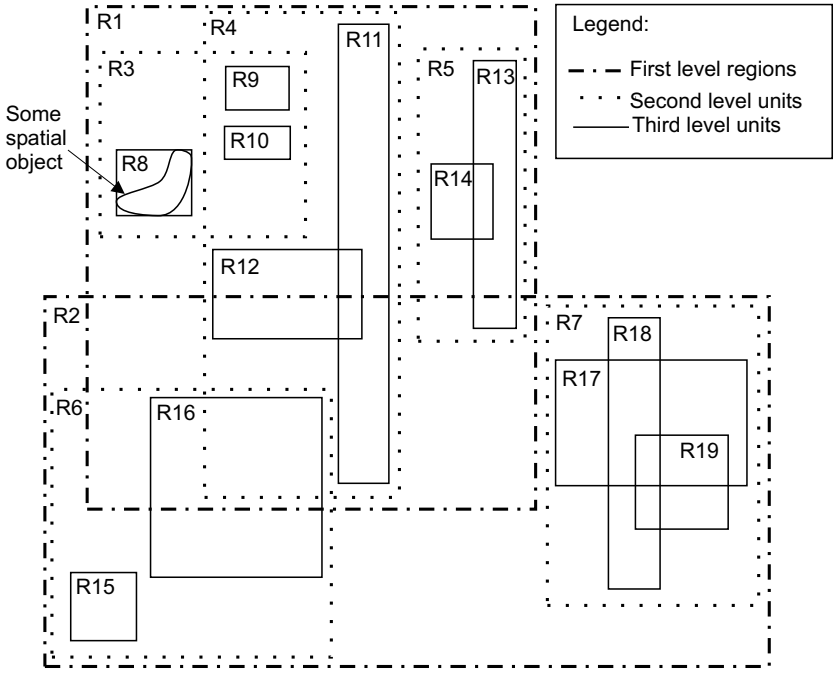


Figura 3.10: MBR representada por uma árvore *R-Tree*.

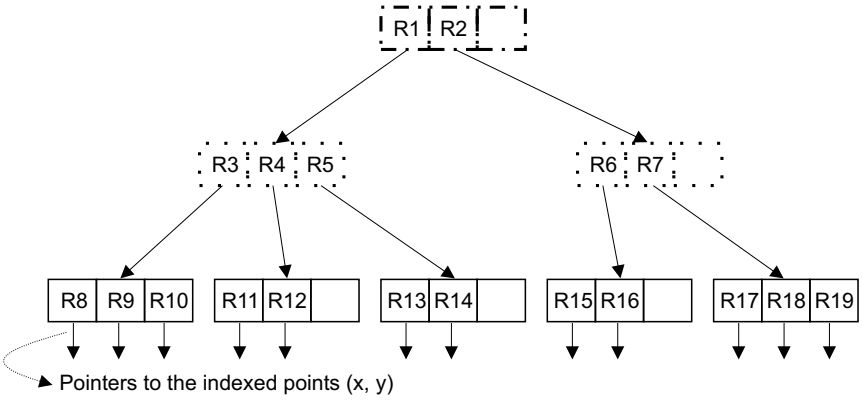


Figura 3.11: Representação esquemática dos elementos armazenados em uma *R-Tree*, segundo a Figura 3.10.

As árvores *R-Tree* armazenam os dados no nível das folhas. No caso dos nós que não são folhas, cada nó contém a informação relacionada ao MBR englobando todas as suas sub-árvores. Conseqüentemente, o MBR contido na raiz da árvore abrange todos os objetos indexados.

Nas consultas por abrangência, apenas as sub-árvores que tenham intersecção com a região de busca serão percorridas. Uma referência muito útil relacionada às consultas do tipo k-vizinhos mais próximos utilizando *R-Tree* pode ser encontrada em (Roussopoulus et al., 1995).

Um grande problema das *R-Tree* é, exatamente, a sobreposição dos MBR. Quando um ponto pertence à intersecção de dois MBR, nenhuma das sub-árvores representadas pelos MBR pode ser descartada.

Em casos específicos, como os da *PointQuad-Tree*, o aumento de dimensão tem um efeito muito adverso, pois cada ponto na dimensão n divide o espaço em 2^n "quadrantes". Depois de uma certa dimensão, o número de divisões torna impossível a utilização desta estrutura. Para resolver essas limitações, outras estruturas apropriadas podem ser utilizadas e serão apresentadas nas seções seguintes.

Depois da *R-Tree*, outras estruturas foram propostas com o intuito de melhorar o seu desempenho. As mais conhecidas são R^+*-Tree* (Sellis et al., 1987), R^**-Tree* (Beckmann et al., 1990), *X-Tree* (Berchtold et al., 1996), *SR-Tree* (Katayama e Satoh, 1997) e outros trabalhos apresentados por Hellerstein et al. (1995) e Papadias et al. (1999).

As árvores R^+*-Tree* reduzem a sobreposição dos MBR quebrando os objetos sobrepostos e armazenando os "pedaços" em nós diferentes. Apesar de resolver esse problema, o algoritmo gera a necessidade de propagar o particionamento árvore abaixo. Um outro problema colateral é que para procurar um objeto, várias sub-árvores poderiam ser acessadas por causa dos "pedaços" que estão separados. Esse mesmo problema da divisão de um objeto afeta a remoção, pois muitas sub-árvores deverão ser percorridas para remover um único objeto.

Em relação às R^**-Tree*, as principais vantagens são que:

- apresenta um novo algoritmo de inserção chamado *forced reinsertion* onde alguns objetos são inseridos novamente na árvore antes de particionar um nó;
- ao contrário da *R-Tree*, que procurava apenas minimizar o volume dos nós gerados, este algoritmo tenta minimizar o perímetro e maximizar a ocupação dos novos nós.

Uma evolução da *R-Tree* é a *X-Tree* (Berchtold et al., 1996). Essa estrutura define um super-nó (*super node*) para tentar resolver o problema da sobreposição de MBR. Se o índice de sobreposição é alto, os nós são concatenados em um super-nó de tamanho variável (geralmente um múltiplo do tamanho original). Essa abordagem é interessante só se for possível acessar o disco com páginas de tamanho variável. Dessa forma, há realmente uma diminuição no número de acessos a disco. Se isso não acontecer, como no caso dos SGBD, o número de acessos aumenta e o desempenho pode ser próximo ao da *R-Tree*.

Infelizmente as técnicas existentes são muito sensíveis à dimensão dos dados. Isso significa que pontos próximos e algoritmos de busca por abrangência (*Range Queries*) apresentam dependência exponencial em relação à dimensão dos dados (Chazelle, 1994). Esse problema também é conhecido como a "Maldição da dimensionalidade", como dito anteriormente.

Informações mais detalhadas sobre MAE podem ser obtidas em (Samet, 1995; Gaede e Günther, 1998; Böhm et al., 2001). Mesmo considerando o potencial que os MAE apresentam, existem tipos de dados complexos que não possuem coordenadas e, conseqüentemente, não poderiam ser indexados através de um MAE. Para resolver esse problema, uma alternativa é utilizar os Métodos de Acesso Métrico que serão detalhados na seguinte seção.

3.4 Métodos de Acesso Métrico

Existem muitos casos onde os objetos a serem indexados não possuem coordenadas e, por tanto, não podem ser tratados apenas por MAE. Um caso típico é o conjunto de palavras de um dicionário. Para esse tipo de dados existem os Métodos de Acesso Métrico (MAM). Esta técnica também pode ser útil para indexar dados vetoriais, pois o espaço vetorial é apenas um caso particular de espaço métrico.

Os conceitos apresentados nesta seção foram baseados no trabalho apresentado por Chávez et al. (2001) e são equivalentes a vários outros trabalhos correlacionados, como Burkhard e Keller (1973); Yianilos (1993); Bozkaya e Ozsoyoglu (1999).

3.4.1 Método Burkhard & Keller

O primeiro trabalho encontrado na literatura envolvendo a indexação de dados métricos é o de Burkhard e Keller (1973). Nesse trabalho são propostas três técnicas para facilitar a consulta por similaridade, adequadas para métricas que retornem valores discretos.

- A primeira técnica considera uma árvore que possui apenas uma entrada em cada nó e grau maior ou igual a dois (definido arbitrariamente). Dado um objeto (chave) selecionado aleatoriamente, calcula-se a distância entre este e os outros, colocando em uma mesma sub-árvore aqueles que estão na mesma distância. Como, no estudo feito pelos autores, a distância é discreta, esse agrupamento é sempre possível. Esse procedimento é repetido recursivamente para cada sub-árvore, até que só reste um objeto, ou até uma profundidade pré-definida;

- Na segunda técnica propõe-se particionar o espaço, em função de um número fixo de objetos representativos, definindo uma árvore multivias. Para cada partição (ou subconjunto) são definidos um objeto representativo e um raio máximo. Para que um objeto faça parte de um subconjunto específico, sua distância ao representativo deve ser menor ou igual ao respectivo raio de cobertura máximo. Esse procedimento é repetido recursivamente;

- A terceira técnica proposta é similar à segunda, com apenas uma restrição. Para entender melhor essa idéia, o conjunto S pode ser visto como um grafo **não direcionado** onde seus objetos são os nós e as arestas são as distâncias entre eles; seja S' um subconjunto de $S(S' \subseteq S)$, tal que dois objetos $x,\ y \in S'$ estão conectados somente se $d(x,\ y) \leq c$, onde c é uma constante previamente definida. Um conjunto que define um grafo completo e que é maximal, ou seja, que nenhum outro nó pode ser inserido sem que o mesmo deixe de ser completo, é chamado *clique* do grafo. A partir desta definição, a terceira técnica corresponde a particionar o conjunto de objetos em *cliques*, cada qual com um valor diferente para c (distância máxima entre os objetos), de forma a garantir que todos os objetos de S estejam em pelo menos um destes *cliques*. Vale observar que a definição não impede que um mesmo objeto apareça em *cliques* diferentes. Em seguida, um elemento arbitrário de cada clique é escolhido como seu representativo.

O critério para inserir um objeto OId à direita ou à esquerda de um nó poderia ser de acordo com a distância em relação ao representativo. Se $d(OId, O_{rep}) > r$ então, o objeto OId deve ser inserido à direita, caso contrário o objeto deve ser inserido à esquerda.

O princípio básico do método apresentado é utilizar a distância entre dois elementos para decidir a posição deles na estrutura. Dessa forma, estruturas hierárquicas podem ser formadas onde cada nó possua um objeto representativo, considerado como ponto de referência para todo o conjunto ao qual ele pertence. Cada um dos elementos inseridos naquele grupo tem a distância em relação ao seu representativo. Conhecendo as distâncias de cada objeto a seu representativo, a propriedade da **desigualdade triangular** (ver Figura 3.12) pode ser utilizada para descartar (podar) objetos, sem ter que calcular a distância explicitamente. Isso é possível pois a função de distância é uma métrica. A vantagem é que muitos descartes podem ser feitos sem a necessidade de calcular a distância entre os objetos existentes na estrutura e o objeto de busca O_q, melhorando assim, o desempenho das consultas.

Para descartar elementos utilizando a desigualdade triangular, pode-se considerar o seguinte cenário: dados um espaço métrico $M = (S, d)$, o conjunto $S \subseteq \mathbf{S}$, o objeto de busca $O_q \in \mathbf{S}$, o raio de busca r_q e um objeto representativo O_{rep}. Um objeto $s \in S$ poderá ser descartado se, e somente se, as equações 3.4 e 3.5 forem satisfeitas de acordo com a Figura 3.12. Na Figura 3.12 observa-se que os objetos compreendidos na região escura são aqueles que, provavelmente, fazem parte do conjunto de resposta.

$$d(O_{rep}, s) < d(O_{rep}, O_q) + r_q \tag{3.4}$$

$$d(O_{rep}, s) > d(O_{rep}, O_q) - r_q \tag{3.5}$$

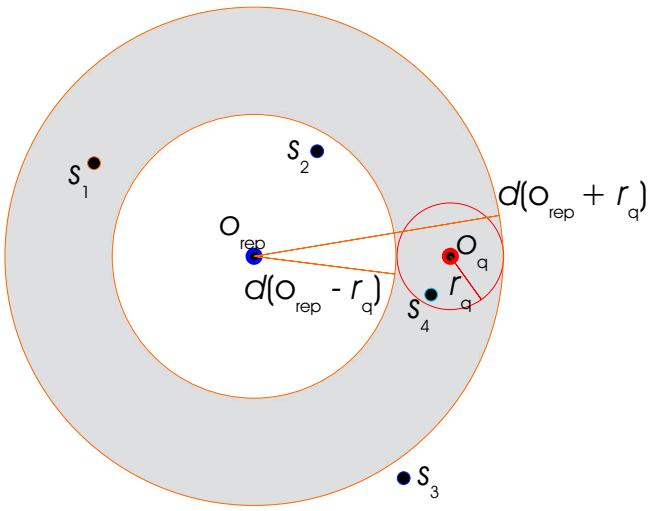


Figura 3.12: Propriedade da desigualdade triangular.

A prova desta afirmação pode ser observada no trabalho de Burkhard e Keller (1973).

3.4.2 AESA

O método *Approximating Eliminating Search Algorithm* (AESA) foi apresentado por Vidal (1986). Como o cálculo de distâncias pode ser um processo demorado, o autor propõe manter a matriz de distâncias pré-calculadas entre todos os pares.

Para realizar uma consulta do tipo *Range Query*, com centro q e raio r, o algoritmo escolhe um pivô, p, de forma aleatória e calcula a distância $r_p = d(p, q)$. A partir desse cálculo, são eliminados todos aqueles elementos u que não satisfizerem $r_p - r \leq d(u, p) \leq r_p + r$. O autor desta técnica comenta que em casos onde o cálculo da distância é extremamente custoso, essa abordagem pode ajudar bastante. Porém, se o conjunto de dados contiver muitos elementos, essa técnica torna-se inviável devido ao tamanho da matriz triangular, $O((n^2 - n)/2)$, onde $n = |S|$ representa o número de elementos do conjunto S.

O custo de construção de uma árvore AESA é da ordem de $O(n^2)$. O autor demonstrou que o custo experimental de recuperação de dados é de $O(1)$. A mesma idéia do AESA também foi explorada por Shasha e Wang (1990), que propuseram a criação de duas matrizes de distâncias. As distâncias conhecidas são armazenadas de forma direta e o restante delas é calculado de forma aproximada em função das existentes. Cada célula da primeira matriz armazena o limite superior da distância para um determinado par de objetos. A segunda matriz é utilizada de forma análoga para manter o limite inferior da distância entre cada par de objetos.

3.4.3 VPT

Em 1991, Uhlmann propôs o *Metric-Tree* (Uhlmann, 1991a) como uma estrutura de árvore desenhada para funções de distância contínua. Um trabalho mais completo muito similar é o *Vantage-Point Tree* (VPT) (Yianilos, 1993). Este trabalho apresentou alguns resultados analíticos e sugeriu algumas técnicas úteis para escolher os chamados *vantage-points*.

A estrutura VPT é uma árvore binária onde cada nó armazena um objeto $s \in S$ e um raio r_p para particionar o espaço. O raio é usado como critério para decidir em que sub-árvore um novo objeto deve ser inserido. Isto é, considerando o objeto s do nó raiz e um nó qualquer $y \in S - s$, aqueles que atendem a $d(s, y) \leq r_p$ devem ser inseridos na sub-árvore esquerda da raiz; e aqueles que atendem a $\mathrm{d}(s, y) > r_p$ devem ser inseridos na sub-árvore direita. O mesmo critério é aplicado para qualquer nível da árvore (da raiz às folhas). O valor de r_p é definido como a mediana do intervalo de distâncias dos outros objetos tentando manter a árvore balanceada. Uma proposta para realizar buscas do tipo k-vizinhos mais próximos com este MAM pode ser observada no trabalho de Chiueh (1994).

3.4.4 FQT

O *Fixed-Queries Trees* (FQT) foi apresentado por Baeza-Yates et al. (1994). A idéia é utilizar objetos de referência (pivôs) para criar uma árvore multivias. Nesta estrutura, os dados são armazenados nas folhas.

A contribuição desta técnica foi em relação à VPT. O FQT utiliza um objeto de referência para cada nível da árvore. Esse critério é aplicado para todos os nós internos. Assim, o número de objetos de referência é igual à altura da árvore.

3.4.5 LAESA

O algoritmo *Linear* AESA (LAESA) é uma nova versão de AESA e foi apresentado por Micó et al. (1994). Os autores propõem a utilização de k pivôs de tal forma que o custo de construção é reduzido para $O(kn)$. Durante as consultas, os k pivôs são utilizados para descartar aqueles elementos que não façam parte do conjunto de resposta. Os elementos que não possam ser eliminados na fase anterior são comparados de forma direta com o objeto de busca.

No trabalho de Micó et al. (1996), os autores propuseram a construção de uma árvore do tipo *Generalized-Hyperplane Tree* (GHT) com os pivôs para acelerar o processo de recuperação dos dados. Esta idéia de utilizar k pivôs e logo indexar as distâncias através de outras estruturas também foi explorada na técnica OMNI (Santos-Filho et al., 2001), que será analisada posteriormente. Nos trabalhos de Nene e Nayar (1997) e Chávez et al. (1999a), as distâncias em relação aos pivôs são ordenadas para permitir uma busca binária.

3.4.6 FHQT

A principal diferença da estrutura *Fixed-Height QT* (FHQT) (Baeza-Yates et al., 1994; Baeza-Yates, 1997) em relação ao FQT é que todas as folhas estão à mesma altura.

Esta técnica faz com que algumas folhas sejam mais profundas que o necessário. De acordo com os trabalho de Baeza-Yates (1997) e Baeza-Yates e Navarro (1998), se o número de pivôs for igual a $O(log\ n)$, a busca também tem um custo de $O(log\ n)$ cálculos de distância. Porém, o custo depende exponencialmente do raio de busca r.

3.4.7 FQA

Os *Fixed Queries Array* (FQA), apresentados por Chávez et al. (1999a), não são árvores. Esta estrutura é uma representação compacta dos FHQT. Para entender esta estrutura, imagine um FHQT de altura h, onde as folhas são percorridas da esquerda para direita e os elementos são copiados em um vetor. O vetor resultante é denominado FQA.

Para cada elemento do vetor resultante são calculados h números representando as ramificações que devem ser escolhidas em cada nível para chegar até o elemento. Cada um dos h números é codificado em bits e logo concatenados formando um inteiro. Os bits que representam os níveis mais altos da árvore são codificados como sendo os mais representativos na seqüência de bits concatenados.

Como conseqüência da codificação dos bits, o FQA resultante estará ordenado pelas seqüências de hb-bits. Cada sub-árvore de uma FHQT corresponde a um intervalo de elementos no FQA. A diferença, no caso de FQA, é que, ao invés de percorrer sub-árvores, basta realizar uma busca binária. Este processo introduz um custo computacional da ordem de $O(log\ n)$, mas em compensação o número de cálculos de distância é reduzido. Uma idéia similar, para o caso de espaços vetoriais, pode ser observada no trabalho de Blott e Weber (1997).

3.4.8 MVPT

Bozkaya e Ozsoyoglu (1997) propuseram a generalização das VPT para árvores m-árias formadas por m-1 percentis ao invés de medianas. O método foi denominado *Multi-Vantage-Point Trees* (MVPT). Uma idéia similar também encontra-se no trabalho de Brin (1995).

Nesta estrutura, a utilização de mais de um elemento por nó foi proposta, de forma similar ao trabalho de Shapiro (1977). Cada nó interno armazena um dos m percentis. Todos os objetos das sub-árvores subseqüentes são armazenados junto com a sua distância para cada objeto representativo. Dessa forma, a MVPT armazena um número maior de distâncias

pré-calculadas (para os objetos representativos). Esta informação adicional permite reduzir, através da desigualdade triangular, em até 80%, a quantidade de cálculos de distâncias em consultas por abrangência (em comparação à VPT).

3.4.9 VPF

A estrutura *Vantage-Point Forest* (VPF) foi apresentada por Yianilos (1998, 1999). Esta estrutura é desenhada para consultas do tipo k-vizinhos mais próximos com raio limitado.

O algoritmo consiste em excluir, em cada nível, os elementos que se encontrem a uma distância intermediária em relação aos pivôs. Isto é devido ao fato de que essa região ser a que apresenta maior população de objetos. Se r_0 e r_n são os raios para o objeto mais próximo e para o mais distante em relação ao pivô p, os elementos u que devem ser excluídos da árvore são aqueles cujas distâncias cumprem a equação: $d(p, r_0) + \delta \leq d(p, u) \leq d(p, r_n) - \delta$.

Os elementos excluídos formam uma segunda árvore. O procedimento é aplicado de forma recursiva criando uma "floresta" de árvores.

3.4.10 BST

Os *Bisector Trees* (BST) foram propostos por Kalantari e McDonald (1983). Este método de acesso é uma árvore binária com dois pivôs, c_1 e c_2, para cada nó. Para cada elemento é calculada a distância em relação a c_1 e a c_2. Aqueles elementos que estiverem mais próximos de c_1 são inseridos no lado esquerdo da árvore e aqueles mais próximos de c_2 no lado direito. Para cada centro também é armazenada a maior distância em relação aos elementos que lhe foram atribuídos. Isso é realizado com o intuito de ajudar a podar sub-árvores na hora da busca.

Uma extensão desta idéia foi apresentada por Nolteimer et al. (1992) e foi chamada *Monotonous* BST. Nesta estrutura, um dos dois elementos representativos c_1 ou c_2 é considerado o centro "pai". Esta técnica provoca uma diminuição gradual do raio quando a inspeção continua árvore abaixo.

3.4.11 GHT

O GHT foi proposto por Uhlmann (1991b). A construção desta árvore é similar ao BST, porém, o GHT, ao invés de utilizar a distância como critério de descarte, utiliza um hiperplano entre c_1 e c_2. Esta árvore binária é construída da seguinte forma: cada nó é composto por dois objetos s_1 e s_2 escolhidos arbitrariamente; os objetos mais próximos ao primeiro objeto, s_1, são armazenados na sub-árvore esquerda do nó; os objetos mais próximos a s_2 são armazenados na sub-árvore direita do nó. O processo é repetido de forma recursiva, até

que todos os objetos tenham sido inseridos. Dependendo da boa seleção dos objetos chaves, a árvore tende a ser balanceada.

Segundo o autor (Uhlmann, 1991b), as árvores GHT apresentam melhor desempenho que o método VPT em altas dimensões. Uma variante dos GHT que suporta operações dinâmicas é o *C-Tree* (Verbarg, 1995).

Um outro trabalho relacionado com esta técnica foi apresentado por Bugnion et al. (1993). Nesse caso, a idéia é reutilizar um nó "pai" com o intuito de evitar o cálculo de duas distâncias em cada nível.

3.4.12 GNAT

A generalização do GHT para árvores m-árias, chamada *Geometric Near-neighbor Access Tree* (GNAT), foi proposta por Brin (1995).

No primeiro nível da estrutura são escolhidos m centros $c_1, ..., c_m$. Para cada objeto no conjunto, é calculado aquele centro mais próximo. Depois disso, uma árvore é construída de forma recursiva utilizando o mesmo critério. Além disso, em cada sub-árvore, são guardadas as distâncias mínima e máxima dos objetos aos respectivos representativos, para que seja possível podar objetos durante as consultas.

Esta idéia tem relação com a partição do espaço com regiões de Voronoi (Voronoi, 1907; Aurenhammer, 1991).

3.4.13 VT

O *Voronoi Tree* (VT), proposto por Dehne e Nolteimer (1987), foi criado com o intuito de melhorar o desempenho do BST. A diferença é que o VT cria dois ou três elementos e filhos para cada nó. Quando uma nova sub-árvore é criada para armazenar um novo elemento, o elemento "pai" mais próximo também é inserido no novo nó.

Este método tem a propriedade de que o raio de cobertura vai sendo reduzido em níveis inferiores. Isso ajuda a manter elementos próximos nas subávores. A árvore resultante é melhor e mais balanceada que os BST. No trabalho de Nolteimer (1989), os autores demonstraram que se, durante a inserção, são aplicadas técnicas similares às utilizadas pela *B-Tree* (Bayer e McCreight, 1972), o resultado é um VT balanceado. Esta idéia foi explorada posteriormente pela *M-Tree*.

3.4.14 M-Tree

A método de acesso *M-Tree* (Zezula et al., 1996; Ciaccia et al., 1997c; Patella, 1999) é uma adaptação da árvore *B-Tree* para dados métricos. Esta técnica foi o primeiro MAM dinâmico

proposto na literatura. Isto significa que a estrutura pode ser construída gradualmente, sem a necessidade de ter o conjunto inteiro de dados para iniciar a construção. Por causa disso, esta estrutura será apresentada com maior grau de detalhe neste trabalho.

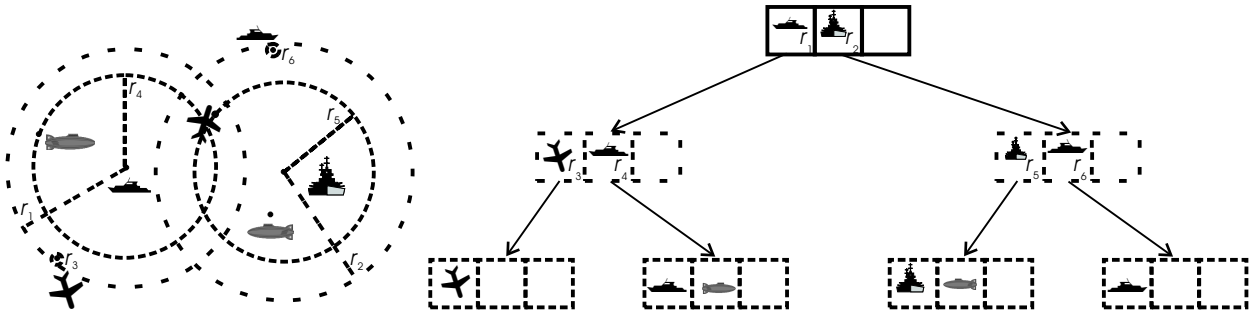
A árvore gerada por este método é balanceada pela altura, o que permite melhorar o tempo de acesso a disco. Uma outra particularidade é que os dados estão só nas folhas. De forma similar às *R-Tree* (Guttman, 1984), a intersecção das regiões do espaço métrico representadas por duas sub-árvores de um mesmo nó, pode não ser vazia. Isso significa que, o particionamento do espaço métrico não necessariamente gera regiões disjuntas. A estrutura de um nó interno não é a mesma que a de uma folha. Cada elemento de uma folha contém: a informação que compõe o objeto propriamente dito, o identificador do objeto (OId) e a distância dele até o objeto representativo daquele grupo. Por outro lado, cada nó interno contém o objeto propriamente dito, um ponteiro relacionado à sua sub-árvore, um raio de cobertura abrangendo toda a sub-árvore e, o valor da distância entre ele o seu representativo, com exceção da raiz.

O objeto representativo de cada nó deve ser aquele que esteja mais próximo do centro, com relação aos outros, isto é, ele é o objeto mais similar aos outros elementos do mesmo nó. O algoritmo de inserção é:

1. se , ;
2. se a raiz for uma folha e não tiver espaço
   (a) gerar dois novos nós minimizando os diâmetros;
   (b) armazenar a distância do objeto representativo até cada uma das outras entradas do nó e copiar ambos os representativos para o nó pai. Se o nó particionado era a raiz, uma nova raiz será criada contendo apenas os dois objetos representativos dos dois novos nós gerados. Se o nó pai estiver sem espaço, o processo de particionamento é propagado árvore acima;
3. se a raiz não for folha, escolher aquela sub-árvore que não precise aumentar seu raio de cobertura. Se essa possibilidade não existe, inserir o nó naquela sub-árvore onde o objeto representativo esteja mais próximo ao objeto de inserção. O processo deve ser repetido até chegar no nível das folhas;
   (a) se a folha tiver espaço disponível, inserir a sua distância para o objeto representativo do nó, ajustar o raio de cobertura dos nós pais e finalizar;
   (b) se a folha estiver lotada, particionar o nó e propagar o particionamento árvore acima.

No algoritmo de construção de uma árvore *M-Tree*, o particionamento de um nó é realizado através do algoritmo MinMax. Nesse processo todos os pares de elementos possíveis são testados até encontrar o par que minimize os raios resultantes. A complexidade desse processo é de $O(n^3)$. Segundo os autores, o algoritmo de particionamento é um dos mais promissores (Ciaccia e Patella, 1998). Porém, nas próximas seções pode-se observar que existem outras alternativas mais eficientes.

Na Figura 3.13 pode-se observar como se organizam os pontos inseridos na estrutura junto com a respectiva *M-Tree*. Durante o processo de consultas as distâncias armazenadas serão utilizadas para poder podar algumas sub-árvores. Só o fato de utilizar a desigualdade triangular para podar sub-árvores, reduz muito o número de cálculo de distâncias e conseqüentemente o tempo de resposta. O cálculo de distância tornar-se-á necessário, unicamente, quando não seja possível podar uma sub-árvore utilizando a desigualdade triangular, isto é, se a região de busca tem intersecção com alguma sub-árvore, ela deve ser inspecionada.



(a) Representação gráfica. (b) Representação esquemática.

Figura 3.13: Estrutura de uma árvore *M-Tree*.

Segundo os autores, *M-Tree* é uma estrutura resistente a altas dimensões (Patella, 1999) e também é competitiva em relação aos R^**-Tree* (Ciaccia et al., 1997a). Este método de acesso também foi utilizado para recuperação de informação por similaridade de forma aproximada (Zezula et al., 1998a; Ciaccia e Patella, 2002).

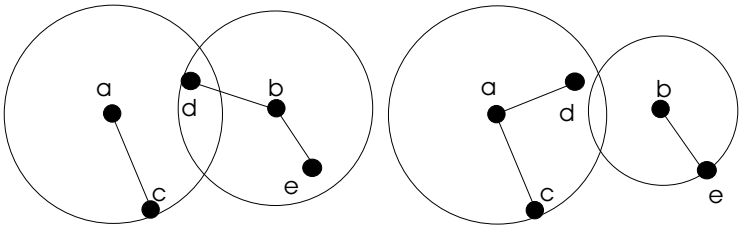
Existem também outros trabalhos interessantes relacionados com as *M-Tree*. Em Ciaccia et al. (1997b), os autores apresentam uma análise de desempenho desse método. A versão paralela da *M-Tree* foi apresentada em Zezula et al. (1998b).

Em (Ciaccia et al., 1999), os autores apresentam um modelo de custo para a *M-Tree*. Nesse trabalho são realizadas estimativas de custo em termos de cálculos de distância e tempo de entrada e saída na execução de consultas por similaridade.

3.4.15 Slim-Tree

O *Slim-Tree* foi proposto por Traina Jr et al. (2000b). O seu funcionamento é semelhante à *M-Tree*, porém o custo computacional do algoritmo de particionamento é menor, sem prejudicar o desempenho das consultas.

A principal contribuição da *Slim-Tree* foi que apresentou um critério para avaliar o grau de sobreposição entre os nós da árvore, chamado *fat-factor*. O problema apresentado no caso de espaços métricos é que não é possível calcular esse volume. Esse critério deu origem a um algoritmo de reorganização dos nós, chamado *Slim-Down*. O objetivo desse algoritmo é reduzir o grau de sobreposição entre as sub-árvores como observado na Figura 3.14. Nessa figura pode-se observar que, como conseqüência do processo *Slim-Down*, o número de acessos a disco durante as consultas tende a ser menor, em comparação com o desempenho obtido com a *M-Tree*. Outra diferença importante é que a inserção é realizada na sub-árvore com menor taxa de ocupação, considerando que exista mais de uma que qualifique.



(a) Antes. (b) Depois.

Figura 3.14: Efeito da aplicação do processo de *Slim-Down*.

A diferença de desempenho em relação às *M-Tree* deve-se à utilização do algoritmo *Minimal Spanning Tree* (MST) durante a partição de um nó. Nesta fase é gerado um MST envolvendo todos os elementos da página a ser dividida e a maior conexão é eliminada. Este algoritmo permite melhorar o custo do MinMax, utilizado pela *M-Tree*, de $O(n^3)$ para $O(n^2\ log\ n)$.

3.4.16 SAT

O *Spatial Approximation Tree* (SAT) foi proposto por Navarro (1999, 2002). O algoritmo está baseado na idéia de aproximação espacial, ao invés de pivôs.

A partir do conjunto de elementos $\mathbb{S}$, um elemento p é selecionado como a raiz. Logo após, p é conectado ao conjunto de vizinhos N, definido como o subconjunto de elementos $u \in \mathbb{S}$ que tenham p como seu elemento mais próximo. Cada elemento do subconjunto restante, sem contar $N \cup \{p\}$, é atribuído ao respectivo elemento mais próximo de N, como apresentado na Figura 3.15. Dessa forma, cada elemento de N se converte, recursivamente, na raíz de uma nova sub-árvore.

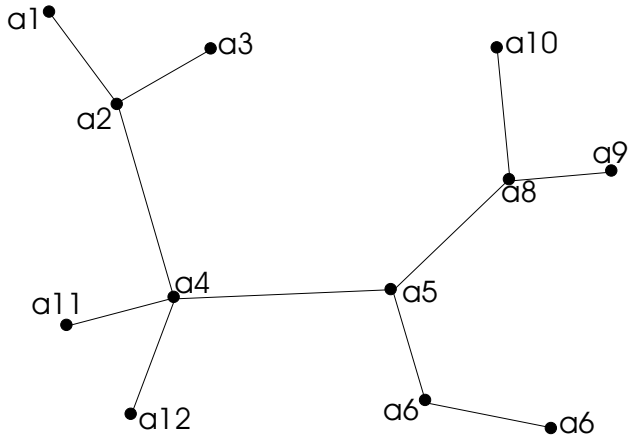


Figura 3.15: Exemplo de um SAT construído considerando $a5$ como raiz.

Existe também uma versão dinâmica desta estrutura, que foi apresentada por Navarro e Reyes (2002).

3.4.17 A família OMNI

A família de estruturas OMNI foi proposta por Santos-Filho et al. (2001); Santos-Filho (2003). Ao invés de pivôs locais, os autores propõem a utilização de pivôs globais chamados de Foci. Para indexar qualquer novo elemento, primeiro devem ser calculadas as distâncias deste para cada um dos elementos do conjunto de Foci. Os autores também apresentaram o algoritmo, chamado *Hull Foci* (HF). A Figura 3.16 mostra o processo de indexação de um novo objeto.

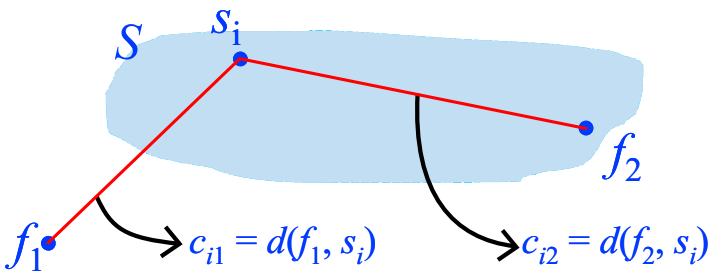


Figura 3.16: Processo de indexação de um novo objeto.

onde $\mathbb{S}$ é o conjunto de objetos e s_i é um dos objetos de $\mathbb{S}$ ($s_i \in \mathbb{S}$). O vetor D formado pelas distâncias calculadas para cada um dos Foci, f_1 e f_2, $D = (c_{i1}, c_{i2})$ é chamado de vetor de coordenadas OMNI. O vetor D pode vir a ser indexado através de uma outra estrutura auxiliar, como a *R-Tree*, dando origem ao método de acesso denominado *OmniRTree*.

Uma outra estrutura proposta nesse mesmo trabalho é a *OmniSequential*. Neste caso não são utilizadas estruturas auxiliares, mas apenas as distâncias aos Foci são utilizadas

para realizar as consultas. Um ponto interessante da familia de estruturas OMNI é que, mesmo sem estruturas auxiliares, os resultados reportados foram muito positivos. Devido à existência dos Foci, a técnica OMNI pode ser considerada não hierárquica.

No caso específico da *OmniRTree*, o vetor D é indexado através de uma *R-Tree* o que agiliza as operações de recuperação de informação. Na Figura 3.17 pode-se observar a forma de realizar uma consulta do tipo *Range Query* utilizando os Foci.

Depois de calcular a distância do objeto de consulta O_q até os dois Foci, o conjunto de candidatos, desde o ponto de vista de cada Foci, formam um anel. Depois de calcular a intersecção de ambos os anéis, o conjunto de candidatos que possivelmente formariam parte da resposta é reduzido dramaticamente às duas regiões escuras. Se existir um terceiro Foci, o conjunto de candidatos poderia ser reduzido, ainda mais.

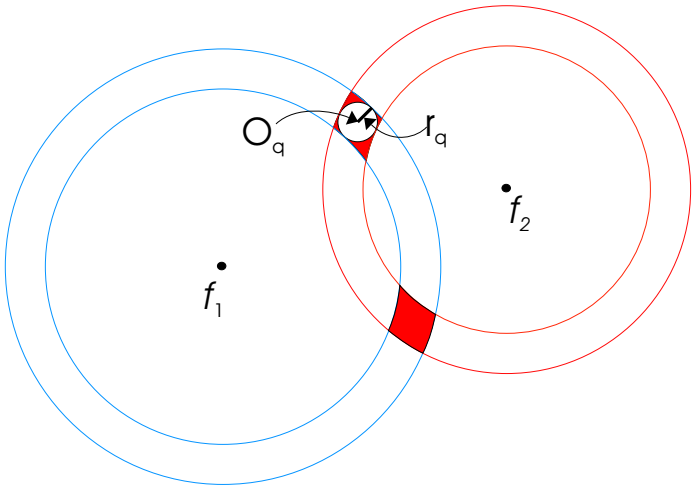


Figura 3.17: Utilização dos Foci para realizar consultas em estruturas da Familia OMNI.

Para recuperar informação através de uma *OmniRTree* calcula-se a *Minimum Bounding Omni Region* (MBOR), para a região de busca, isto é, a mínima região, segundo as coordenadas OMNI, que envolveria a região da consulta. Uma vez calculada a MBOR, a *R-Tree* é utilizada para recuperar os objetos envolvidos nessa área (ver Figura 3.18).

Em (Santos-Filho, 2003), os autores também propõem a técnica denominada *Omni B-Forest*. Nessa estrutura as distâncias para cada um dos Foci são indexadas através de um B^+*-Tree* (Comer, 1979). A conexão seqüencial existente nas folhas dos B^+*-Tree* é aproveitada para tentar detectar os objetos mais próximos. Essa técnica também pode ser observada no trabalho publicado por Traina Jr et al. (2002).

Em todos os casos, um dos maiores problemas da técnica OMNI está relacionado ao processo de determinação do número de Foci. Experimentalmente, os autores demonstraram que o número de Foci mais apropriado é próximo à dimensão intrínseca ou dimensão fractal do conjunto de dados. Uma boa referência para realizar esse cálculo é o trabalho de Schroeder (1991). A dimensão intrínseca de um conjunto de dados é o número real de dimensões no qual

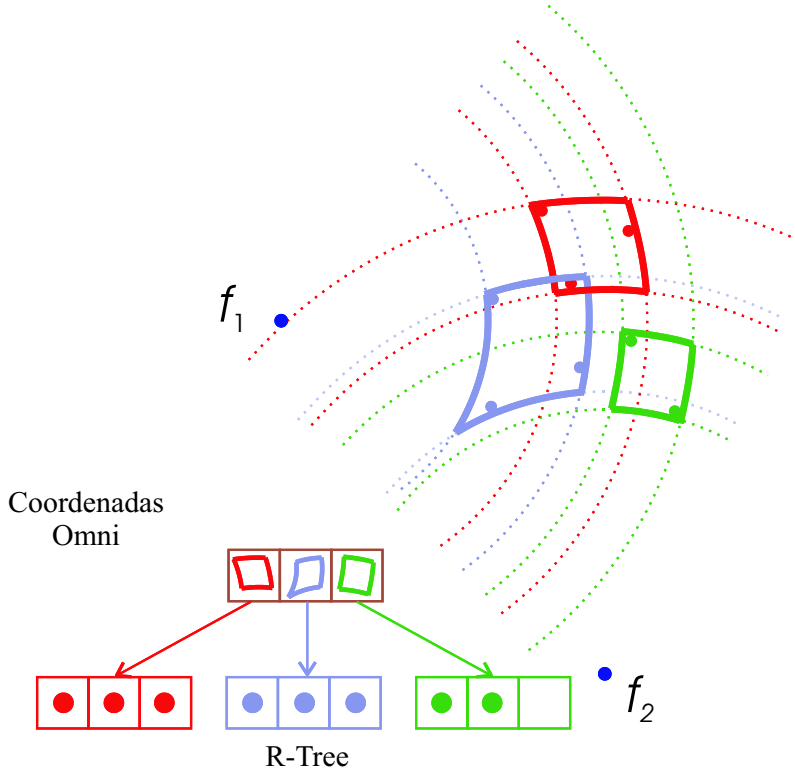


Figura 3.18: Utilização das MBOR para recuperar informação em uma *OmniRTree*.

pode ser representado um conjunto, sem afetar as distâncias entre os dados. Por exemplo, os pontos de um plano podem ser representados em dimensão 30, mas a dimensão intrínseca do conjunto de dados ainda continua sendo dois.

Outro trabalho onde pode ser observado o efeito da escolha dos pivôs sobre uma busca em espaços métricos é apresentado por Bustos et al. (2001). Maiores referências em relação aos MAM podem ser encontrados em (Chávez et al., 2001).

3.5 Considerações Finais

No presente capítulo foi apresentada uma visão geral dos métodos de acesso de dados espaciais e métricos. Primeiramente, foi apresentada uma rápida visão de métodos de indexação para dados unidimensionais. Logo após, foram apresentados os Métodos de Acesso Espacial (MAE) que são utilizados para a indexação de dados multidimensionais, como vetores.

Um outro assunto importante, abordado neste capítulo, foi o relacionado aos Métodos de Acesso Métrico (MAM). Os principais MAM existentes na literatura foram analisados. Nesse caso, a única informação disponível para realizar a indexação é a distância. Os MAM também são úteis para a indexação de dados vetoriais. Neste caso, basta definir uma função de distância.

Um dos problemas mais preocupantes dos MAE e MAM é que todos eles apresentam limitações para trabalhar com dimensões muito altas. Uma solução para esse problema é aplicar algum método de redução de dados como MDS, *FastMap*, etc. Uma outra alternativa é descartar as características que sejam menos significativas. Porém, essa alternativa leva à perda de informação. Existem estruturas que compactam os nós internos mantendo os objetos intactos nas folhas, como no caso da *TV-Tree*. No entanto, não há garantia de que a redução seja mantida em todos os níveis, como foi observado no trabalho de Santos-Filho (1999).

Também foram apresentados alguns MAM existentes na literatura apropriados para consultas por similaridade. São especialmente apropriados para aqueles casos onde não é possível contar com as coordenadas nos elementos. Nesse tipo de métodos de acesso basta definir uma função de distância métrica para poder iniciar o processo de indexação.

A principal limitação dos métodos apresentados está relacionada à falta de aprendizado com consultas anteriores. Isto significa que não reaproveitam o conhecimento gerado pelas consultas anteriores para reduzir o tempo de resposta das próximas consultas.

Capítulo 4

Levantamento do potencial de RNA, MAE e MAM no processo de RIS

Tanto as RNA quanto os MAE e MAM são técnicas amplamente utilizadas na literatura para Recuperação de Informação por Similaridade (RIS). Neste capítulo apresenta-se algumas aplicações, assim como as vantagens e limitações que cada técnica apresenta no contexto da aplicação dessas técnicas em RIS.

4.1 RNA na Recuperação de Informação

O processo de Recuperação de Informação por Similaridade (RIS) é pesquisado em várias áreas do conhecimento, entre elas as RNA. Dentro dessa área, o problema de RIS pode ser tratado por vários modelos, como por exemplo, as Redes Multi-Camadas, Mapas Auto-Organizáveis (SOM), entre outros. Nesta seção, são apresentados alguns exemplos desses tipos de redes para RIS, porém a ênfase principal está orientada a redes do tipo SOM.

Uma das principais características de uma RNA é a tolerância a falhas. Esta característica torna possível classificar um padrão com ruído ou mesmo incompleto. O algoritmo mais conhecido e, provavelmente o mais utilizado, é o *BackPropagation* (Rumelhart et al., 1986). As redes treinadas com o *BackPropagation* pertencem ao paradigma supervisionado. Esse algoritmo deu início a uma grande variedade de técnicas. Entre as modificações mais conhecidas pode-se citar o algoritmo *BackPropagation* com *Momentum* (Shiffmann et al., 1994), *QuickPropagation* (Fahlman, 1988) e *RProp* (Riedmiller, 1994). Todos esses algoritmos tentam acelerar a convergência da rede e diminuir o erro de classificação.

Outra área de pesquisa permanente é a relacionada à definição da arquitetura da rede. Em (Rawtani et al., 2001), apresenta-se uma tentativa para a definição do número de neurônios das camadas intermediárias. Também existem os modelos construtivos, tais como a rede *Cascade Correlation* (Fahlman e Lebiere, 1990), *Upstar* (Frean, 1989), *Tiling* e a generalização *MTiling* (Parekh et al., 1996), *Tower* e *Pyramid* (Parekh et al., 1995, 1997), que são algoritmos que treinam uma RNA com o paradigma supervisionado. Em (Parekh

et al., 1995), pode ser encontrado um bom resumo das técnicas construtivas para as redes Multi-Camadas ou mais conhecidas.

Um grupo de técnicas apropriadas para detectar agrupamentos corresponde às redes da família derivadas da Teoria da Ressonância Adaptativa (ART) que foram desenvolvidas por Carpenter e Grossberg (Grossberg, 1976a; Carpenter e Grossberg, 1986, 1987b). Existem diversos aprimoramentos destas primeiras idéias tais como ART2 (Carpenter e Grossberg, 1987a), ART3 (Carpenter e Grossberg, 1990), ARTMAP (Carpenter et al., 1991a), FuzzyART (Carpenter et al., 1991b), *Fuzzy* ARTMAP (Carpenter et al., 1992), *Fully Self-Organizing Simplified ART* - FOSART (Baraldi e Parmiggiani, 1998). Em todos esses casos, ainda permanecem alguns problemas para responder questões do tipo k-vizinhos mais próximos e busca por abrangência. Uma aplicação interessante da rede ART-2A (Carpenter et al., 1991c) em combinação com o Método de Acesso Métrico *Slim-Tree* para RIS pode ser encontrado em (Vicentini, 2002). Nesse trabalho, os autores mostraram que a combinação dessas duas estruturas foi mais eficiente, em termos de cálculos de distância e números de acessos ao disco do que a própria estrutura *Slim-Tree*.

Mesmo os trabalhos mais recentes, envolvendo recuperação de informação através de RNA, estão relacionados apenas com problemas de classificação e/ou detecção de agrupamentos (*clustering*). No entanto, o problema de RIS apresenta algumas características particulares que restringem os modelos que poderiam ser utilizados.

Em primeiro lugar, como não se conhece quantos aglomerados existem nos dados, o modelo de rede a ser utilizado deverá ser treinado sob o paradigma não supervisionado (ver Seção 2.1 Pág. 5). Por causa disso, neste capítulo, somente modelos auto-organizáveis são apresentados. Um dos modelos mais conhecidos para realizar essa tarefa foi proposto por Kohonen (1984). Este modelo é conhecido como mapa auto-organizável ou, simplesmente, mapas de Kohonen (ver Seção 2.2.1 Pág. 6). Os mapas de Kohonen são RNA que têm a capacidade de detectar aglomerados (*clusters*) nos dados. Os trabalhos de Kaski et al. (1998); Oja et al. (2003); Yamakawa (2003) são referências muito importantes para observar o crescimento de aplicações de SOM no processo de RIS e em outras áreas de pesquisa.

Em relação aos dados para o treinamento de RNA, um processo comum nessa área refere-se à transformação dos dados em uma representação adequada através de um processo denominado pré-processamento. O pré-processamento do conjunto de dados pode envolver a normalização e/ou a redução do número de atributos com o intuito de diminuir o número de neurônios das RNA, assim como o tempo requerido para o treinamento. Existem várias formas de normalizar os dados. Todas elas têm a finalidade de transformar o intervalo onde se encontram os dados no intervalo [0,1] ou [-1,1], uma vez que as funções de ativação utilizadas nos neurônios artificiais, normalmente, assumem valores nesses intervalos. Para reduzir o número de atributos podem ser utilizadas algumas técnicas de extração de características, tais como a transformada de *Karhunen-Loève* ('K-L') ou Análise de Componentes Principais

(PCA– *Principal Component Analysis*) (Pearson, 1901; Hote, 1933; Duda e Hart, 1973; Fukunaga, 1990)[1], Decomposição de Valor Singular (SVD– *Singular Value Decomposition*) (Strang, 1980; Press et al., 1988; Golub e Loan, 1989), *FastMap* (Faloutsos e Lin, 1994; Faloutsos et al., 2003).

Outro fator importante, que tem relação com a saída da rede, refere-se à inicialização dos pesos. Um estudo dos efeitos desse fator pode ser observado em Fernández-Redondo e Hernández-Espinosa (2001). Um outro fator que pode afetar a precisão das RNA são os parâmetros de aprendizado. Não existe uma regra e sim muitas heurísticas (Haykin, 1999) para determinar esses parâmetros. O problema do ajuste da taxa de aprendizado foi analisado nos experimentos apresentados por Wilson e Martinez (2001).

Existem na literatura alguns trabalhos relacionados à recuperação de informação baseada em similaridade utilizando RNA. As redes neurais do tipo SOM são freqüentemente utilizadas para detectar agrupamentos e projetar vetores de alta para baixa dimensão (Kohonen, 1982). O modelo original de Kohonen não preserva completamente a similaridade dos dados projetados, pois sempre que se reduz a dimensão de um dado, através de uma projeção, perde-se precisão nos dados resultantes. Este problema da preservação da topologia também foi estudado por Villmann et al. (1997) e por Ritter et al. (1992).

Um outro problema relacionado ao tratamento de dados multidimensionais refere-se à visualização. Existem algumas técnicas que permitem a visualização de um SOM treinado como a matriz de distâncias ou *U-Matrix* desenvolvida por Ultsch (1993a,b) e a *P-Matrix*, apresentada por Ultsch (2003). Ambas as técnicas são apropriadas para os processos de mineração de dados (*Data Mining*) e descoberta de conhecimento (*Knowledge Discovery*).

Uma outra técnica que também pode ser utilizada junto aos SOM é o algoritmo conhecido como *Watershed* apresentado por Beucher e Lantuéjoul (1979). Esse algoritmo permite a detecção gradual dos agrupamentos existentes nos dados.

Os mapas de Kohonen também apresentam bons resultados com grandes volumes de dados (Kohonen, 1997a; Kohonen e Somervuo, 1997). Um dos exemplos mais representativos de projeção de grandes volumes de dados é o sistema WEBSOM apresentado por Kohonen (1998), e também o PicSOM proposto por Laaksonen et al. (1999). Através do WEBSOM, aproximadamente, um milhão de documentos são projetados em duas dimensões, tentando manter, o máximo possível, a similaridade entre eles. Para recuperar a informação relacionada aos pontos projetados, cada ponto contém um ponteiro para o documento que este representa na base de dados. No caso do PicSOM, a mesma idéia é aplicada para recuperação de imagens. No trabalho de Koskela (1999) pode-se observar um exemplo de recuperação de informação baseada no conteúdo com SOM.

[1] Os próprios mapas auto-organizáveis podem também ser vistos como uma forma de obter generalizações não lineares de PCA (Ritter, 1995).

No trabalho apresentado por Merkl (1997), o autor apresenta resultados da exploração de textos através de SOM hierárquicos. Nesses experimentos, a abordagem hierárquica foi, aproximadamente, 600% mais rápida que a abordagem tradicional.

Existem também outras ferramentas de uso geral, como o software Clementine (Grimmer, 1996). Porém, tanto no caso do WEBSOM como no caso do Clementine, a técnica utilizada para a detecção dos agrupamentos é a visualização. Há uma necessidade de intervenção do usuário para o ajuste dos parâmetros.

Uma recente extensão dos mapas de Kohonen é o DIPOL-SOM (*Distance Preserving On-Line Learning* SOM), apresentado por König e Michel (2003). Nesse trabalho, os autores propuseram criar um modelo que combinasse a capacidade de projetar os dados a dimensões baixas com baixo custo computacional junto às vantagens que um SOM oferece.

Um outro trabalho interessante é o sistema de recuperação de informação utilizando RNA apresentado por Hodge e Austin (2001b). Nesse trabalho, os autores apresentam uma rede neural híbrida, do tipo *TreeGCS* (Hodge e Austin, 2001a) com sub-redes do tipo GCS (Fritzke, 1993a, 1994b), para recuperação de informação a partir de grandes volumes de informação obtidos a partir da internet.

Um outro problema freqüente é o relacionado a variações dos dados ao longo do tempo. Nesse caso, existem estudos como o *Growing Self-Organizing Algorithm for Dynamic Clustering* desenvolvido por Ohta e Saito (2001). Nesse trabalho, o crescimento da rede é controlado através de um contador existente em cada neurônio. Além disso, o algoritmo permite a remoção virtual de unidades não desejadas.

O trabalho de Shim (2001) também é um exemplo de integração de RNA em sistemas de recuperação de informação. Segundo o autor, o sistema proposto, *Hierarchical Associative Knowledge Learning Memory*-HAKLM, é estruturado, inteligente e possui um mecanismo para aquisição, inferência e extração automática de conhecimento. O sistema também apresenta um método seletivo para filtrar a informação encontrada.

Devido ao fato de que nem sempre é simples medir o grau de semelhança entre dois objetos complexos, existem trabalhos, como os apresentados por Kaski et al. (2001); Kaski e Sinkkonen (2001); Kaski (2001); Peltonen et al. (2003), cujo objetivo é fazer com que as redes aprendam a calcular a distância métrica entre dois objetos de forma automática. Nessa mesma linha de pesquisa encontra-se o trabalho apresentado por Lim et al. (2001). O trabalho propõe a utilização de uma RNA do tipo *Multi-Layer Perceptron* (MLP) para aprender, dinamicamente, a função de distância.

Em relação à quantidade de agrupamentos, Sun et al. (2001) apresentou uma técnica para medir o resultado do processo de detecção, de forma dinâmica, do número apropriado de agrupamentos para um determinado conjunto de padrões. Os trabalhos de Baraldi e Blonda (1998); Jain (1999) são boas referências em relação a este problema.

Uma outra forma de reduzir o tempo de treinamento de uma Rede Neural é investir mais tempo no pré-processamento ou diminuição do número de padrões. Para isso, existem estudos como o apresentado por Su e Basu (2001). Nesse trabalho, os autores propõem um algoritmo para selecionar alguns padrões que serão apresentados à rede neural durante o treinamento. Esse processo reduz o tempo de treinamento de forma considerável.

Uma alternativa interessante, que utiliza um princípio distinto em relação aos mapas de Kohonen, é a técnica conhecida como *Generative Topographic Mapping* (GTM) proposta por Bishop et al. (1998). Nesse trabalho, são utilizadas superfícies Gaussianas cujos parâmetros são otimizados através do algoritmo *Expectation-Maximization* (EM) (Dempster et al., 1977). Também existe uma versão hierárquica do modelo GTM chamada *Hierarchical* GTM (Tino e Nabney, 2002).

Todas as técnicas mencionadas são úteis para agrupar os dados. Porém, além de procurar o agrupamento, muitos problemas exigem como resposta os n padrões mais parecidos com o padrão de entrada, $\vec{\xi}$. Até mesmo em redes da família ART o problema persiste, pois se um determinado *cluster* classifica corretamente N padrões e o padrão de entrada $\vec{\xi}$ pertence àquele agrupamento, não existe uma heurística para determinar os n vizinhos mais próximos, $n < N$, de um determinado padrão[2]. O problema poderia ainda ser mais complexo se o objetivo fosse obter os $N + 1$ vizinhos mais próximos, pois as saídas vizinhas em uma rede ART não necessariamente representam os padrões fisicamente próximos.

No caso das redes SOM, os padrões podem mudar de agrupamento ao longo do treinamento. Isso significa que o modelo de SOM não é estável, inviabilizando, dessa forma, a utilização das mesmas até que seja concluído o treinamento. Em um sistema de recuperação de informação real, novos dados podem aparecer e ser inseridos a qualquer momento e nem sempre é viável, depois de cada inserção, esperar por um tempo de treinamento.

Uma heurística utilizada para reduzir o número de cálculos de distância ao longo do treinamento é fazer com que o sistema lembre qual é a unidade vencedora, w_1, para cada padrão $\vec{\xi}$, no n-ésimo ciclo (Kohonen, 1998; Kaski, 1999). Isso é realizado assumindo-se que existe uma alta probabilidade de que as unidades vencedoras, para um mesmo padrão, em ciclos consecutivos, estejam fisicamente próximas. Porém, essa técnica não pode garantir que as unidades vencedoras, em ciclos consecutivos para um mesmo padrão, estejam próximas sempre. Pode acontecer que as unidades vencedoras, para um mesmo padrão em ciclos consecutivos, estejam fisicamente distantes no mapa. Esse problema é mais visível no início do treinamento, pois a taxa de aprendizado não é tão pequena. Porém, à medida em que o treinamento avança, a unidade vencedora – para um mesmo padrão em ciclos consecutivos – varia cada vez menos viabilizando a aplicação dessa técnica.

Hoje em dia, é cada vez mais comum a existência de maiores e mais complexos bancos de dados o que, necessariamente, significa um maior tempo de processamento e maior

[2]Neste trabalho, a comparação seqüencial não é considerada uma alternativa aceitável.

número de cálculos de distância. Por essas razões, é sempre importante continuar analisando possibilidades para tentar reduzir o número de operações necessárias para o treinamento de uma RNA. No presente trabalho foram utilizados Métodos de Acesso Espacial para criar redes com maior número de unidades, maior precisão, menor erro e menor tempo envolvido no processo de busca, como será apresentado no Capítulo 5.

4.2 Vantagens das RNA no processo de RI

As RNA possuem diversas vantagens que podem úteis para o processo de Recuperação de Informação (RI). Nesta seção, algumas das maiores vantagens são detalhadas com o intuito de aproveitar o potencial existente para melhorar esse processo.

4.2.1 Aprendizado através de exemplos

Uma das características mais relevantes das RNA é a capacidade que estas possuem de adquirir o conhecimento através de exemplos apresentados de forma repetitiva durante o treinamento (Kohonen, 1982; Rumelhart et al., 1986; Haykin, 1999).

Cada vez que um novo padrão é apresentado à rede, o algoritmo de treinamento ajusta os pesos existentes com objetivo de representar melhor o padrão apresentado. Essa idéia foi inspirada na forma em que o cérebro humano aprende desde os primeiros anos de vida. O processo de apresentação repetitiva do conjunto de dados permite que, após o treinamento ter sido finalizado, a rede possa reconhecer padrões que nunca lhe foram apresentados.

Mesmo considerando este item como uma vantagem, é necessário considerar que o processo de treinamento pode precisar de um tempo computacional considerável.

4.2.2 Capacidade de generalização

A capacidade de generalização é uma conseqüência do processo de aprendizado. Esta característica significa que uma RNA seja capaz produzir uma boa aproximação para entradas que não fizeram parte do conjunto de treinamento (Haykin, 1999).

A idéia principal é que a RNA possa reconhecer novos objetos que nunca lhe foram apresentados ou mesmo aqueles conhecidos, mas com pequenas variações ou ruído, em função do conhecimento adquirido a partir do subconjunto com o qual foi treinada.

4.2.3 Pouco espaço necessário para armazenar a estrutura

Na grande maioria das vezes, o número de neurônios n existentes na rede é muito menor que o número, m, de padrões representados nela ($n \ll m$). Isso representa uma grande vantagem em termos de espaço em comparação aos MAE e MAM.

Esta vantagem está fortemente relacionada à complexidade da distribuição dos dados. Dependendo da complexidade, pode-se representar milhares de padrões com poucas unidades. Este item é considerado uma vantagem, especialmente em relação aos MAM e MAE que armazenam todos os objetos do conjunto apresentado.

4.2.4 Capacidade de reduzir a dimensão dos dados

Existem alguns modelos de RNA, tais como, os mapas de Kohonen (1984) podem ser utilizados para projetar dados multidimensionais para baixas dimensões. Geralmente, esta técnica é utilizada para projetar os dados para $2d$ ou $3d$ de modo a facilitar a visualização (Duch, 1994). Uma grande vantagem desta abordagem é que os dados projetados conservam uma grande parte da similaridade dos dados no espaço original.

4.2.5 Capacidade para detectar agrupamentos

As RNA, especialmente aquelas pertencentes ao paradigma não supervisionado, são capazes de detectar os agrupamentos existentes nos dados. Também existem técnicas, como a *U-Matrix* (Ultsch, 1993a), que podem ser utilizadas junto com SOM para visualizar os agrupamentos. Essa capacidade representa uma grande vantagem para o processo de organizar a informação.

4.2.6 Capacidade para previsão de séries temporais

Uma rede neural também pode ser treinada para prever o comportamento de um conjunto que varia ao longo do tempo. A partir dessa informação, espera-se que a rede seja capaz de prever o comportamento que os dados terão no futuro. Um exemplo claro da utilidade desta vantagem é a previsão do clima, tomando como dados de entrada os dados climáticos de um certo período de tempo.

Alguns modelos de redes que apresentam esta característica são o NETtalk (Sejnowski e Rosenberg, 1987) e o *Time Delay Neural Network* (TDNN) (Land e Hilton, 1988; Waibel et al., 1989).

4.3 Limitações das RNA no processo de RI

Nesta seção são detalhadas algumas limitações das RNA com especial ênfase no processo de Recuperação de Informação (RI) em grandes bancos de dados.

4.3.1 Definição da quantidade de ciclos de treinamento

Durante o treinamento de uma RNA, não é trivial definir, *a priori*, o número de ciclos de treinamento. Isto pode acontecer devido a vários fatores que influenciam no processo de treinamento de uma RNA. Os mais importantes são:

- a configuração inicial do conjunto de pesos antes do treinamento (Fernández-Redondo e Hernández-Espinosa, 2001; Alahakoon et al., 2000);
- a taxa de aprendizado (Wilson e Martinez, 2001);
- os parâmetros específicos de cada modelo;
- a complexidade da distribuição do conjunto de dados;
- a quantidade de padrões apresentados;
- a arquitetura e a quantidade de unidades da rede;
- o algoritmo de inserção de novas unidades, no caso de redes construtivas (Fritzke, 1997; Parekh et al., 1997).

A convergência do treinamento de uma RNA, e conseqüentemente o número de ciclos necessários para finalizar o treinamento, depende da boa combinação desses fatores. Geralmente o processo de ajuste dos parâmetros de uma RNA depende, em grande parte, da experiência do projetista.

A determinação do número de ciclos de treinamento é um problema sério que afeta diretamente o treinamento. Por esse motivo, este fator é considerado uma limitação.

4.3.2 Definição da arquitetura apropriada da rede

A definição da arquitetura de uma rede neural também é um fator que depende, em grande parte, da experiência do projetista. Porém, existem quatro abordagens principais para se obter uma arquitetura apropriada:

Empírica: esta técnica também é conhecida como **tentativa-erro** e depende quase exclusivamente da experiência do projetista.

Evolutiva: esta abordagem é baseada em princípios baseados em algoritmos genéticos. A partir de uma população inicial de redes, novos indivíduos (redes) são gerados a cada ciclo, mantendo aqueles com maior grau de aptidão ou que ajudem a minimizar o erro da rede (Goldberg, 1989).

Construtiva: nesta técnica, o processo é iniciado com uma rede com topologia mínima. Novos elementos, tais como neurônios, conexões ou camadas são inseridos de forma gradual, segundo o problema (Parekh et al., 1995, 1997). Exemplos de modelos SOM construtivos foram apresentados na Seção 2.3.

***Pruning*:** esta abordagem visa reduzir ao máximo as unidades e conexões redundantes ou pouco relevantes, sem afetar a qualidade da saída produzida pela rede. Alguns exemplos desta abordagem podem ser observados no trabalho de Reed (1993). Existem também técnicas que incorporam técnicas de poda junto com a abordagem construtiva como o algoritmo GNG (Fritzke, 1995b).

As quatro abordagens apresentadas têm por objetivo obter RNA apropriadas para cada problema. Porém, é difícil determinar, *a priori*, a arquitetura mais apropriada para um determinado problema.

Essa limitação dificulta, ainda mais, a utilização de modelos de RNA em um sistema que requeira respostas imediatas ou onde novos padrões podem ser adicionados/removidos a qualquer momento, inclusive após o treinamento ter sido concluído.

4.3.3 Reinicialização do treinamento

O processo de treinamento de uma RNA tem por objetivo ajustar, de forma iterativa, os pesos existentes nas conexões. É recomendável que o conjunto de treinamento seja apresentado em uma ordem distinta a cada ciclo. Isto permite que a RNA seja treinada de maneira uniforme, com todos os padrões com a mesma probabilidade de apresentação.

Porém, se um novo padrão aparecesse, após o treinamento ter sido finalizado, não é recomendável treinar a rede apenas para novo padrão. O que aconteceria seria que a rede perde o conhecimento armazenado até então e passaria a decorar o novo padrão. Para evitar essa situação, o treinamento deve ser reiniciado. O problema é que reiniciar o treinamento envolve um custo computacional alto. Existe um grande número de modelos de RNA que apresentam este comportamento.

Contudo, existem modelos que foram projetados com o intuito de resolver esse problema, como é o caso dos modelos da família ART (Grossberg, 1972, 1976a,b). Esse problema é abordado através do dilema da **Estabilidade-Plasticidade** (Grossberg, 1988). Nas redes do tipo SOM, derivadas do modelo de Kohonen, o dilema da Estabilidade-Plasticidade é um problema que persiste.

4.3.4 Instabilidade

No caso específico das redes do tipo SOM, sabe-se que cada neurônio é o encarregado de responder por uma determinada região do espaço de dados. As regiões que as unidades representam podem ser melhor entendidas através de um diagrama de Voronoi (1907) (ver Figura 4.1).

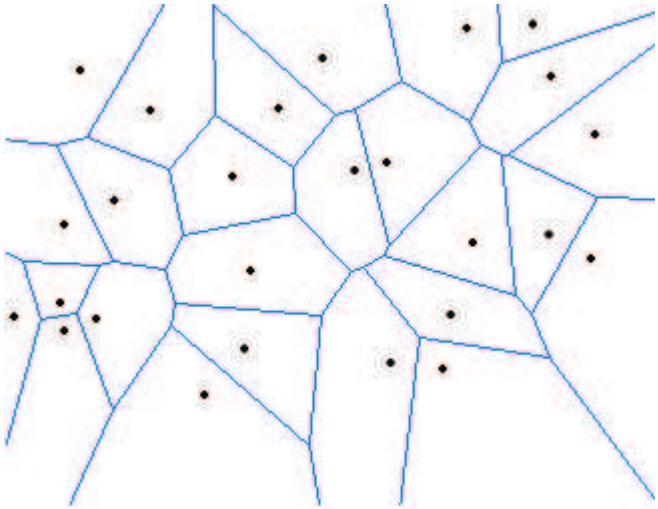
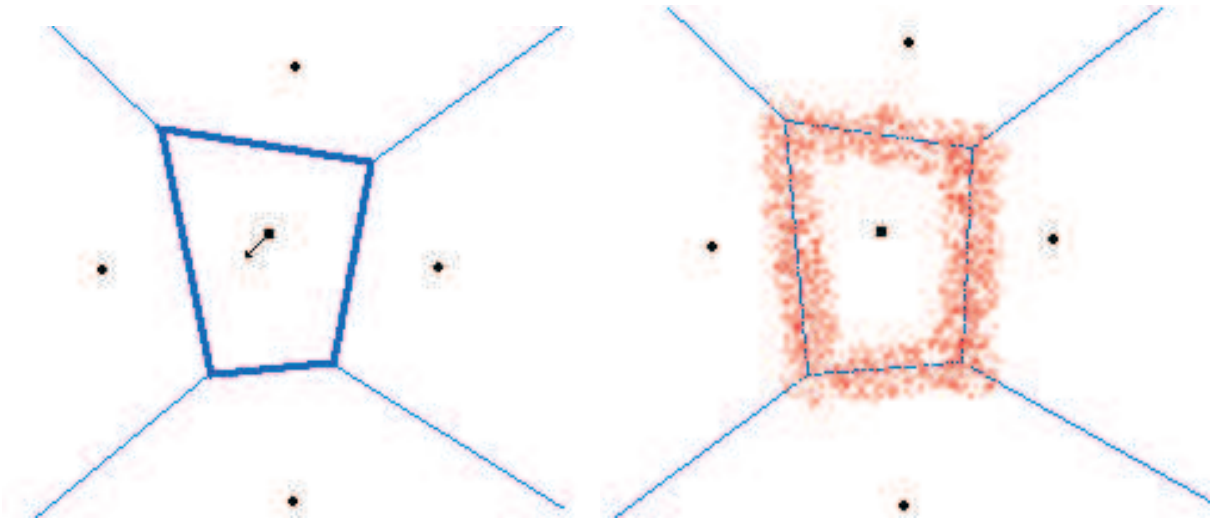


Figura 4.1: Diagrama de Voronoi para uma distribuição arbitrária de pontos.

Durante o treinamento, cada vez que um vetor de pesos é modificado, a região de Voronoi daquele neurônio e dos seus vizinhos também varia, como observado na Figura 4.2.



(a) Região de Voronoi afetada pela correção dos pesos de um neurônio

(b) Padrões com alta probabilidade de mudar de região

Figura 4.2: Efeito do treinamento em um SOM derivado do modelo de Kohonen.

Como conseqüência da correção de pesos de uma unidade, vários padrões podem passar de uma região de Voronoi para uma outra adjacente. Isto significa que a execução de consultas,

durante o período de treinamento, pode gerar resultados que não são confiáveis. Portanto, o único período de estabilidade da rede acontece quando o treinamento é concluído.

Em um sistema real, isto pode representar um problema sério pois, a classe de um determinado padrão poderia variar, inclusive, entre ciclos consecutivos.

4.3.5 Projetadas para trabalhar em memória principal

Devido à capacidade de representar um conjunto de dados com poucas unidades, na maioria das vezes, as RNA podem ser treinadas inteiramente em memória principal, sem a necessidade de utilizar dispositivos de acesso secundário, tais como, disco rígido, etc. Esse ponto é considerado uma limitação em relação aos MAE e MAM.

4.3.6 Falta de organização para responder consultas do tipo k-vizinhos mais próximos e buscas por abrangência

As RNA são bastante utilizadas para resolver problemas de classificação onde deseja-se conhecer o grupo ao qual pertence um determinado padrão, $\vec{\xi}$, apresentado como entrada. Porém, existem muitos casos onde o usuário precisa conhecer mais informação, além do grupo. Alguns exemplos desse tipo de consulta são: "Quais são os três padrões mais parecidos com o padrão $\vec{\xi}$?"ou "Quais são os padrões que apresentam, no mínimo, são 90% de similaridade com relação ao padrão $\vec{\xi}$?". Na atualidade, a execução desse tipo de consulta, por meio de RNA, apresenta deficiências.

O primeiro obstáculo para responder estas consultas é a ausência de organização entre os padrões que são classificados dentro de um mesmo grupo. A busca seqüencial não é considerada, neste trabalho, como uma alternativa aceitável. Por causa disso, este ponto é considerado uma limitação das RNA.

Existem trabalhos, como os apresentados por (Vicentini, 2002; Vicentini e Romero, 2003), que resolvem parcialmente o problema utilizando um MAE ou MAM dentro de cada grupo. Nesses trabalhos utilizou-se Redes Auto-organizáveis da família ART. Porém, esse modelo ainda apresenta problemas para responder consultas por similaridade porque os grupos gerados não têm relação entre si. Por esse motivo, se um grupo classifica apenas n padrões, seria impossível procurar $n+1$ elementos. O problema é que unidades fisicamente próximas na rede não necessariamente representam os padrões mais próximos. A única saída neste caso é a comparação seqüencial.

4.4 MAE e MAM na Recuperação de Informação

Os MAE e MAM têm sido mais exploradas no ambiente acadêmico. Existem diversos trabalhos dessas técnicas relacionados ao processo de Recuperação de Informação (RI). No trabalhos de Faloutsos et al. (1993, 1994) pode-se observar um exemplo de recuperação de imagens baseada no conteúdo.

No projeto *Informedia* (Wactlar et al., 1996, 1999) desenvolvido na *Carnegie Mellon University* também foi usado o método *SR-Tree* (Katayama e Satoh, 1997). Esse algoritmo foi utilizado junto com alguns algoritmos de extração de características usados para processar vídeos e gerar vetores em dimensão 1215-D. Atualmente, o projeto *Informedia* utiliza a *OmniSequential* (Santos-Filho et al., 2001).

Existem poucas aplicações industriais utilizando esses métodos de acesso. Um exemplo de aplicação na indústria é o gerenciador de bancos de dados Oracle que recentemente incorporou as *R-Tree* (Kanth et al., 1999). Nesse trabalho, a *R-Tree* foi utilizada para indexar dados de 3-D até 10-D, que ainda são consideradas dimensões relativamente baixas. Em relação à indexação de imagens pode ser observado o trabalho de Annamalai et al. (2000) que explica esse processo em um gerenciador de bancos de dados.

Um dos maiores problemas é que o investimento para incorporar essas técnicas em softwares comerciais é muito elevado. Por essa razão, inclusive as propostas mais recentes, como a NB-Tree (Fonseca e Jorge, 2003), tentam utilizar técnicas já existentes no mercado. Nesse último caso, os autores argumentam que os métodos modernos estão cada vez mas mais complexos, mas o desempenho nem sempre melhora com o mesmo ritmo. Por esse motivo, os autores propõem reduzir a dimensão dos dados e logo indexá-los com uma B^+-*Tree* (Comer, 1979).

4.5 Vantagens dos MAM e MAE no processo de RI

Os MAE e MAM também apresentam diversas vantagens que podem ser aproveitadas no processo de Recuperação de Informação (RI). As maiores vantagens estão relacionadas a custo computacional baixo e à capacidade de lidar com grandes volumes de dados.

4.5.1 Não existe o conceito de ciclos

No caso de MAE e MAM, os padrões são incorporados à estrutura através de um algoritmo de inserção próprio de cada método. Cada padrão só é processado uma única vez para ser inserido na estrutura.

Um objetivo comum a todos os MAE e MAM é o fato de garantir uma média de acesso, geralmente da ordem logarítmica, independente da ordem em que os padrões são inseridos.

Por causa dessa característica inerente aos MAE e MAM, uma única apresentação do conjunto de padrões é suficiente para a formação da estrutura.

4.5.2 A incorporação de um novo padrão pode ser realizada a qualquer momento

Nos métodos de acesso dinâmico, tais como *M-Tree*, *Slim-Tree*, etc., é possível inserir novos elementos a qualquer instante. A inserção de um novo elemento pode ser analisada como se ele fosse, simplesmente, o último elemento da lista de padrões a serem inseridos. Isto também significa que não é relevante o intervalo de tempo que possa existir entre a inserção de um elemento e outro.

Porém, é necessário considerar que existem vários métodos de acesso que não apresentam esta vantagem, pois é necessário ter o conjunto inteiro de dados antes de iniciar a construção da estrutura.

4.5.3 Existe um ponto na estrutura para cada padrão

Cada novo objeto inserido é representado por um elemento na estrutura de forma pontual. Esta vantagem também significa que a estrutura dispõe de maior quantidade de informação para atender consultas mais complexas tais como *Range Queries* ou *k-Nearest Neighbors*.

Uma preocupação especial relacionada a este ponto é o fato de que, maior quantidade de dados requer maior quantidade de processamento. Porém, devido à organização dos dados, os MAE e MAM podem garantir que, mesmo com grandes volumes de dados, o acesso será realizado em tempo sub-linear, ou seja, cuja complexidade computacional seja menor que $O(n)$.

4.5.4 Muitos deles foram projetados também para memória secundária

Devido ao grande volume de dados com os quais se espera que uma MAE e MAM lide, a grande maioria desses métodos são projetados, não apenas para serem utilizados em memória principal, mas também em memória secundária.

Por outro lado, o acesso à memória secundária leva a um maior tempo devido às operações de entrada/saída envolvidas. Porém, o custo computacional de acesso sub-linear permite que o ganho seja maior e esta característica valha a pena de ser aplicada.

4.5.5 Maior facilidade para a execução de consultas do tipo k-vizinhos mais próximos e buscas por abrangência

Devido à existência de uma unidade para cada padrão e especialmente devido à organização dos dados, esse tipo de consultas pode ser realizado com muita precisão. Muitos dos MAM e MAE, tais como os apresentados na Capítulo 3, utilizam a abordagem hierárquica, o que muitas vezes, facilita a operação de busca.

No caso de MAM com abordagens não hierárquicas, como as estruturas da família OMNI (Santos-Filho et al., 2001), a poda de elementos utilizando Foci também ajuda muito na execução deste tipo de consultas.

4.5.6 Tempo de construção e busca

Todos os MAE e MAM assumem que o processamento seqüencial para buscar um objeto não é aceitável. Portanto, os custos computacionais de busca desses métodos visam ser sempre menores que n, isto é $O(n) \ll n$. Na Tabela 4.1, observa-se um breve resumo de custos dos principais MAM. Essa tabela está baseada no trabalho de Chávez et al. (2001), porém alguns métodos foram adicionados. Os custos computacionais para o caso de MAE podem ser observados no trabalho de Gaede e Günther (1998).

Os custos computacionais apresentados na Tabela 4.1 não podem ser considerados como absolutos. Por exemplo, no caso do método *Burkhard-Keller Tree* (BKT) (Burkhard e Keller, 1973), os autores apenas fizeram uma proposta de três técnicas, sem existir implementação. Porém, o custo $n\ log\ n$ refere-se ao custo de construção de uma árvore n-ária como sugere a idéia dos autores.

No caso da *M-Tree*, com custo $O(n(m..m^2)log_m n)$, o termo $m..m^2$ depende de certos parâmetros do algoritmo. Para maiores detalhes recomenda-se analisar os artigos citados para cada caso.

4.6 Limitações dos MAM e MAE no processo de RI

Mesmo considerando a rapidez dos MAE e MAM, essas também apresentam limitações que são analisadas nesta seção.

4.6.1 Não aprendem dos exemplos anteriores

Este problema pode ser melhor entendido executando-se duas vezes a mesma consulta por similaridade. Sem considerar mecanismos que ajudem à aceleração de consultas como

Método de Acesso	Custo de construção	Custo de busca
Burkhard-Keller Tree (BKT) (Burkhard e Keller, 1973; Shapiro, 1977)	$O(n\ log\ n)$	$O(n^{\alpha})$
Fixed-Queries Trees (FQT) (Baeza-Yates et al., 1994)	$O(n\ log\ n)$	$O(n^{\alpha})$
Fixed-Height QT (FHQT) (Baeza-Yates et al., 1994; Baeza-Yates, 1997; Baeza-Yates e Navarro, 1998)	$O(n\ h)$	$O(log\ n)$ (*)
Fixed Queries Array (FQA) (Chávez et al., 1999b)	$O(n\ h)$	$O(log\ n)$ (*)
Vantage-Point Tree (VPT) (Uhlmann, 1991a; Yianilos, 1993; Chiueh, 1994)	$O(n\ log\ n)$	$O(log\ n)$ (**)
Multi-Vantage-Point Trees (MVPT) (Bozkaya e Ozsoyoglu, 1997; Brin, 1995)	$O(n\ log\ n)$	$O(log\ n)$ (**)
Vantage-Point Forest (VPF) (Yianilos, 1999)	$O(n^{2-\alpha})$	$O(n^{1-\alpha}\ log\ n)$
Bisector Trees (BST) (Kalantari e McDonald, 1983; Nolteimer et al., 1992)	$O(n\ log\ n)$	não analisado
Generalized-Hyperplane Tree (GHT) (Uhlmann, 1991a; Bugnion et al., 1993; Verbarg, 1995)	$O(n\ log\ n)$	não analisado
Geometric Near-neighbor Access Tree (GNAT) (Brin, 1995)	$O(nm\ log_m n)$	não analisado
Voronoi Tree (VT) (Dehne e Nolteimer, 1987; Nolteimer, 1989)	$O(n\ log\ n)$	não analisado
M-Tree (Ciaccia et al., 1997c)	$O(n(m..m^2)log_m n)$	não analisado
Spatial Approximation Tree (SAT) (Navarro, 1999, 2002; Navarro e Reyes, 2002)	$O(n\ log\ n/log\ log\ n)$	$O(n^{1-\odot(1/log\ log\ n)})$
Approximating Eliminating Search Algorithm (AESA) (Vidal, 1986)	$O(n^2)$	$O(1)(***)$
Linear AESA (LAESA) (Micó et al., 1994, 1996; Nene e Nayar, 1997; Chávez et al., 1999a)	$O(k\ n)$	$k + O(1)(***)$
Slim-Tree (ST) (Traina Jr et al., 2000b)	$O(m^2\ log_2(m) log_m n)$	não analisado
OmniSequential (Santos-Filho et al., 2001)	$O(k\ n)$	não analisado

(*) Se $h = log(n)$.

(**) Somente válido para raios muito pequenos.

(***) Conclusões empíricas sem análise.

Tabela 4.1: Custos computacionais de construção e busca em MAM.

memória *Cache*, o número de operações que o algoritmo realizaria seria o dobro do que no caso de uma única consulta.

No caso de bancos de dados multimídia, as consultas são na sua grande maioria por similaridade. Isso significa que não adiantaria apenas utilizar memória *Cache* para procurar por uma consulta realizada anteriormente. Na grande maioria das vezes, as consultas futuras poderiam estar apenas relacionadas a dados similares que nunca fizeram parte de consultas prévias.

Esse comportamento pode ser interpretado como uma falta de habilidade para reaproveitar o conhecimento gerado por consultas anteriores. A idéia fundamental é que, naquelas áreas onde sejam realizadas várias consultas, o número de operações para consultas posteriores seja reduzido de forma gradual.

4.6.2 Maior espaço necessário para armazenar a estrutura

Esta limitação surge pela presença de uma unidade na estrutura para cada padrão apresentado. Em termos de espaço, isto pode ser interpretado como um problema. Porém, os recursos de memória principal e secundária hoje em dia são relativamente baratos.

Além disso, a grande maioria de MAE e MAM objetivam não apenas o uso da memória principal, mas também da memória secundária. Nesse último caso, a restrição de espaço praticamente desaparece. Porém, aparece o problema de tempo extra causado pelo acesso ao dispositivos de armazenamento secundário.

4.6.3 Maior complexidade da implementação

A implementação de um MAM e MAE é relativamente mais complexa que a de uma RNA. Um dos fatores que aumenta a complexidade é que são considerados tanto o acesso à memória principal quanto à memória secundária.

Além disso, devido ao maior volume de objetos armazenados na estrutura, uma preocupação permanente é que os dados estejam organizados de tal forma que possa ser garantido um tempo de recuperação constante. Também é necessário considerar que o fato de ter que lidar com memória secundária significa que se deve considerar o tempo de espera causado pelas operações de entrada/saída.

4.7 Considerações Finais

No presente capítulo foram apresentadas as diversas aplicações, vantagens e limitações existentes, tanto nas RNA quanto nos MAM e MAE, no processo de RIS.

Diversos trabalhos recentes relacionados à Recuperação de Informação utilizando RNA foram apresentados. As maiores coleções dos trabalhos relacionados aos SOM podem ser encontradas nos trabalhos de Kaski et al. (1998); Oja et al. (2003); Yamakawa (2003). Algumas aplicações de Métodos de Acesso no processo de RIS também foram apresentadas. Duas boas referências para este caso são apresentadas em Gaede e Günther (1998); Chávez et al. (2001).

As principais vantagens e limitações das RNA no processo de RIS foram detalhadas. Entre as principais pode-se citar: o aprendizado através de exemplos e iterações, a gene-

ralização do conjunto de padrões com poucos neurônios e pouco espaço necessário para armazenar a estrutura.

As limitações das RNA, em relação ao processo de RIS, referem-se à falta de estrutura, de parte da rede, para responder consultas que requeiram maior grau de precisão, tais como, k-vizinhos mais próximos e buscas por abrangência (*Range Query*). Um outro aspecto que, dependendo do problema, pode ser crítico é o fato de que não é trivial definir, *a priori*, o tempo de treinamento de uma RNA.

Neste capítulo, também foram detalhadas algumas vantagens e limitações dos MAM e MAE. Entre as principais ressalta-se o fato dessas técnicas não precisarem de mais de uma única apresentação do conjunto de padrões apresentado e que requerem um tempo de construção e recuperação dos dados relativamente pequeno. Essa vantagem foi fundamental para a proposta das técnicas SAM-SOM e MAM-SOM, detalhadas no Capítulo 5.

As principais limitações dos MAM e MAE também foram apresentadas neste capítulo. A principal delas é a ausência de algum mecanismo de aprendizado que reaproveite o conhecimento das consultas para acelerar gradualmente as consultas posteriores. Essa idéia é de grande utilidade para um melhor entendimento dos métodos de acesso SAM+ e MAM+ propostos no Capítulo 6. Além disso, os MAM e MAE precisam de maior espaço necessário para armazenar a estrutura e sua complexidade da implementação é maior.

Capítulo 5

As técnicas SAM-SOM e MAM-SOM

As Redes Neurais Artificiais (RNA) vêm sendo cada vez mais utilizadas para a Recuperação de Informação por Similaridade (RIS) (Kohonen, 1997b; Haykin, 1999). Um dos modelos mais utilizados para agrupar dados, baseado em critérios de similaridade, são os *Self-Organizing Maps* (SOM) ou Mapa Auto-Organizáveis de Kohonen (1982). Esse tipo de rede vem sendo aplicado com freqüência para grandes volumes de dados e altas dimensões (Kohonen et al., 1996; Kohonen, 1997a, 1998; Ultsch, 2003; Kaski, 2003; Oja, 2003).

As principais vantagens e limitações relacionadas às RNA foram apresentadas nas Seções 4.2 e 4.3, respectivamente. Com base nessas características, propõem-se neste capítulo duas novas técnicas para RIS, denominadas SAM-SOM e MAM-SOM. Essas duas técnicas são o resultado da incorporação de Métodos de Acesso Espacial (MAE) e de Métodos de Acesso Métrico (MAM), respectivamente, no processo de treinamento de um SOM. A sigla SAM corresponde ao termo em inglês *Spatial Access Methods* (SAM). Ambas as técnicas são propostas com o intuito de reduzir o número de comparações para encontrar a unidade vencedora, uma vez que nos modelos existentes, esse processo tem sido feito de forma seqüencial.

Para entender melhor a necessidade da incorporação de MAE e MAM no treinamento de um SOM será analisado, primeiramente, o custo computacional envolvido durante o treinamento e, em seguida, serão apresentadas as duas técnicas que estão sendo propostas.

5.1 Análise do custo computacional do treinamento de redes do tipo SOM

Para analisar o custo computacional do treinamento de uma rede neural do tipo SOM, é necessário lembrar que, cada vez que um novo padrão $\vec{\xi}$ é apresentado à entrada da rede,

o algoritmo deve encontrar aquela unidade que melhor o represente na rede, isto é, aquela unidade, s_1, cuja distância entre o seu vetor de pesos w_{s_1} e o padrão apresentado seja a menor possível, como observado na Equação (5.1).

$$s_1 = min\|\vec{\xi} - w_i\| \quad \forall\ i \tag{5.1}$$

O número de cálculos de distância, NCD, necessários durante o treinamento de uma rede neural do tipo SOM convencional está determinado pela Equação (5.2).

$$NCD = NU \times NP \times NC \tag{5.2}$$

onde NU é o número de unidades da rede, NP é o número de padrões apresentados e NC é o número de ciclos. A Equação (5.2) é derivada do fato de que, cada vez que um padrão é apresentado à rede, este deve ser comparado com o vetor de pesos de cada uma das unidades existentes antes de determinar a unidade vencedora. Esse mesmo processo é repetido para cada ciclo, gerando um total de NCD cálculos de distâncias. Por outro lado, deve-se considerar que cada cálculo de distância é afetado diretamente pela complexidade inerente à função de distância envolvida e pela dimensão dos dados.

Quando a rede SOM é do tipo construtiva (CSOM), o número de unidades NU varia ao longo do tempo. Nesse caso, o número de cálculos de distância, NCD_{CSOM}, até o t-ésimo padrão apresentado à rede, é determinado pela equação de recorrência (5.3).

$$\begin{aligned} NCD_{CSOM}(t) &= NU(t) \times NP + NCD_{CSOM}(t-1) \\ NCD_{CSOM}(1) &= NU_0 \end{aligned} \tag{5.3}$$

onde t, representa o índice do padrão apresentado à rede, $NU(t)$ é o numero de unidades naquele instante e NU_0 é o número inicial de unidades na rede.

Supondo que o algoritmo insere uma nova unidade à rede, a cada λ padrões apresentados, o número de cálculos de distância acumulado, TC, para atingir uma rede com N unidades é determinado pela Equação (5.4). A variável λ representa um intervalo de padrões apresentados, após o qual uma nova unidade será criada na rede. Por exemplo, se $\lambda = 500$, significa que a cada 500 padrões apresentados, durante o treinamento, uma nova unidade é criada na rede.

$$
\begin{aligned}
NCD_{CSOM} &= \sum_{i=NU_0}^{N-1} i\lambda \\
&= \lambda \sum_{i=NU_0}^{N-1} i \\
&= \lambda \frac{(NU_0 + N - 1) \times (N - NU_0)}{2} \\
&= \lambda \frac{N(N-1) - NU_0(NU_0 - 1)}{2}
\end{aligned} \tag{5.4}
$$

onde NU_0 é o número inicial de unidades na rede. Observando-se a Equação (5.4) nota-se que a velocidade de inserção de novas unidades, determinada por λ, afeta diretamente o tempo de treinamento da rede.

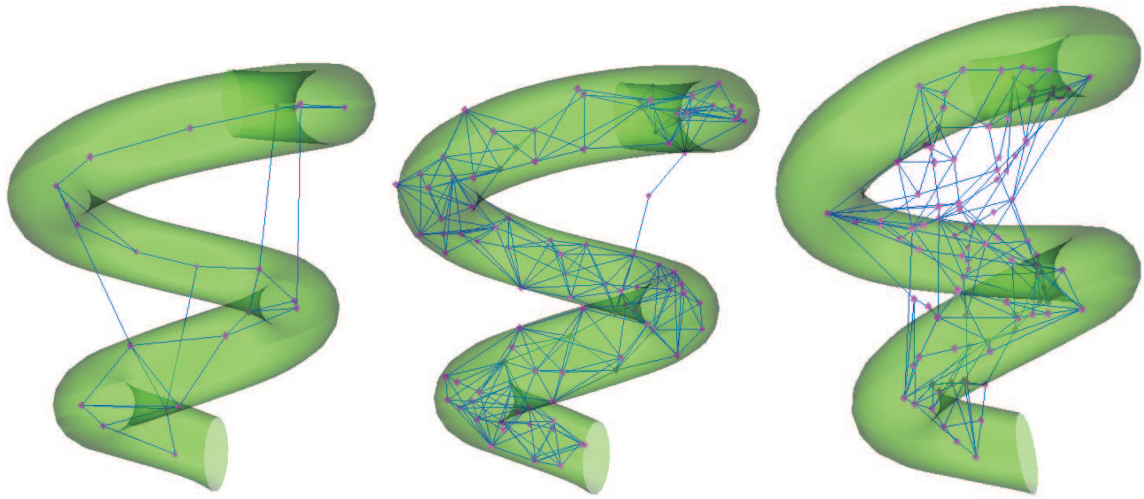
No caso das redes *Growing Neural Gas* (GNG), o número de unidades no início do treinamento é igual a dois, isto é, $NU_0 = 2$. A partir da Equação (5.4), é possível deduzir a Equação (5.5), que representa o número de cálculos de distância que uma rede GNG, com uma taxa de inserção λ arbitrária, precisaria para atingir um número N de unidades.

$$
NCD_{GNG} = \lambda \frac{N^2 - N - 2}{2} \tag{5.5}
$$

Um valor alto de λ pode aumentar, consideravelmente, o tempo de treinamento e, ao mesmo tempo, reduzir a quantidade de unidades criadas em um período de tempo. Se o valor de λ for muito pequeno, o algoritmo de treinamento não terá tempo suficiente para adaptar a nova unidade à estrutura da rede, gerando assim redes deformadas (Cuadros-Vargas e Romero, 2002).

Na Figura 5.1, os dados de entrada da rede estão sendo gerados de maneira uniforme dentro dos objetos apresentados na mesma imagem. Nessa figura, pode-se destacar a deformação produzida pela diminuição de λ em uma rede do tipo GNG (Fritzke, 1995b). Na Figura 5.1(a), por exemplo, apenas um número pequeno de unidades foram criadas porque o valor de λ é relativamente alto (500). Isto reduz o número de unidades que podem ser criadas em um certo período de tempo. Em compensação, as unidades encontram-se bem distribuídas ao longo do objeto que está sendo reconhecido. Nas Figuras 5.1(b) e 5.1(c), onde $\lambda = 100$ e $\lambda = 10$, respectivamente, pode-se observar com mais freqüência unidades fora da área do objeto sendo reconhecido. Isso acontece por dois motivos: o primeiro é que, quando o algoritmo GNG cria uma nova unidade, a posição é resultado da interpolação de duas unidades já existentes. Porém, o algoritmo não avalia se a nova unidade está sendo criada dentro da distribuição de dados ou no meio de uma área "vazia". O segundo motivo

é que, com um valor de λ pequeno, o algoritmo não tem tempo para corrigir a posição das novas unidades e levá-las até algum ponto onde possam realmente ser úteis.



(a) 10000 padrões, $\lambda = 500$ (b) 10000 padrões, $\lambda = 100$ (c) 1000 padrões, $\lambda = 10$

Figura 5.1: Efeito da diminuição de λ utilizando uma rede do tipo GNG.

O processo para encontrar um valor de λ apropriado não é uma tarefa fácil e nem trivial. Isso depende de múltiplos fatores, tais como, complexidade da distribuição e volume dos dados, etc. Se o valor de λ for reduzido ao seu valor mínimo, isto é, $\lambda = 1$, que é o caso no qual uma nova unidade é criada na rede para cada padrão apresentado, o algoritmo não teria tempo suficiente para adaptar as novas unidades à estrutura existente, gerando assim redes visivelmente deformadas.

Em todos os cálculos, apresentados nas Equações (5.2), (5.3) e (5.4), pode-se observar que, para encontrar a unidade vencedora sempre são consideradas todas as unidades existentes. Isto é, o padrão apresentado, $\vec{\xi}$, é sempre comparado com todas as unidades da rede antes de decidir qual é o neurônio vencedor.

O mesmo acontece com as redes SOM hierárquicas, apresentadas na Seção 2.4. De fato, a proposta de todos esses métodos está diretamente relacionada, entre outros fatores, com a tentativa de redução do número de cálculos de distância durante o treinamento da rede. Existem autores, como por exemplo Koikkalainen e Oja (1990), que afirmam ter reduzido o custo computacional, de $O(n)$ para $O(log\ n)$, pelo fato de ser uma abordagem hierárquica. Essa afirmação é apenas teórica, pois na prática, as redes desse tipo ganham um pouco de tempo pela existência da hierarquia mas, dentro de cada sub-rede, ainda utiliza-se a abordagem seqüencial. Existem vários autores, tais como Huntsberger e Ajjimarangsee (1990); Demian e Mignot (1993, 1996); Chan et al. (1995); Kohonen (1997b); Kolinummi et al. (2000); Hämälainen (2002), que afirmam que o tempo de treinamento pode ser melhorado através de processamento paralelo. Em todos esses casos, as operações são distribuídas mas, o número delas é o mesmo. Isso significa que o custo computacional continua sendo o mesmo.

O motivo da preocupação com a redução do tempo de treinamento pode ser explicado pelo custo computacional apresentado na Equação (5.5) que depende diretamente de λ e também de forma quadrática em relação ao número de unidades da rede, N. A inserção de uma nova unidade representará um aumento de cálculos considerável. A seguir, será explicado, com maior grau de detalhe, o processo proposto no presente trabalho para a incorporação de Métodos de Acesso Espacial (MAE) e Métodos de Acesso Métrico (MAM) para a redução do number of distance calculations ao longo do treinamento de um SOM.

5.2 Incorporação de MAE e MAM em SOM

Para poder incorporar Métodos de Acesso Espacial (MAE) e Métodos de Acesso Métrico (MAM) em redes SOM, derivadas do mapa de Kohonen, é necessário, em primeiro lugar, estabelecer uma correspondência entre cada nó do Método de Acesso (MA) e as unidades ou agrupamentos da rede que está sendo treinada. Essa relação permitirá realizar as operações de RIS através do MA prescindindo do processamento seqüencial.

A incorporação de MAE em SOM dá origem à técnica SAM-SOM. Dentro dessa técnica existem duas variações. No primeiro caso, os MAE são utilizados para substituir o processo seqüencial de determinação da unidade vencedora, utilizado em redes do tipo SOM, por um mecanismo mais rápido. Esse mecanismo consiste em aproveitar a organização existente em um MAE, como será exemplificado mais adiante. Em seguida, o processo de correção de pesos dos neurônios da rede é realizado de acordo com o algoritmo do SOM envolvido. Para cada neurônio cujos pesos são atualizados, deve existir uma correspondente atualização no elemento que o representa no MAE. Esta técnica é denominada como SAM-SOM híbrida.

O segundo caso difere do primeiro no seguinte sentido: uma nova unidade é criada para cada padrão apresentado e, de acordo com um fator de conectividade, essa nova unidade é conectada a uma quantidade finita de neurônios da rede. Essa nova unidade é criada com um vetor de pesos exatamente igual ao vetor de atributos do padrão de entrada. Não existe o processo de correção de pesos. Esse processo possui a vantagem de que o próprio algoritmo de construção da rede faz com que os neurônios fiquem conectados aos seus vizinhos mais próximos. Isso evita que seja preciso um treinamento exaustivo para obter a auto-organização. Isso também significa que a auto-organização ocorre já durante a fase de construção da rede. Esta técnica é denominada SAM-SOM$*$.

A incorporação de MAM em redes SOM dá origem à técnica MAM-SOM. Nesse caso, de forma análoga às variações da técnica SAM-SOM, também pode-se falar das técnicas MAM-SOM híbrida e MAM-SOM$*$. Porém, no caso da técnica MAM-SOM híbrida, deve ser considerado que, em um espaço métrico, apenas existe a função de distância e os objetos propriamente ditos. Isso significa que não pode ser considerada a existência de coordenadas dos objetos, as mesmas que são necessárias para o processo de correção de vetores de pesos.

Por isso, é contraditório falar de correção de pesos e ao mesmo tempo da técnica MAM-SOM. No entanto, existem autores, como Somervuo (2003), que apresentam técnicas de correção de pesos nesse tipo de espaço.

Para entender melhor o processo de incorporação de um MAE em um SOM é necessário lembrar os passos envolvidos no treinamento desse tipo de redes. No Algoritmo 5.1, pode-se observar, de uma forma geral, os passos envolvidos nesse processo.

Algoritmo 5.1 Treinamento de um SOM visto de uma forma geral.

1: **Início**
2: Criar a topologia inicial da rede;
3: Inicializar os pesos;
4: **repetir**
5: Escolher o próximo padrão ξ;
6: Encontrar a(s) unidade(s) mais próxima(s) necessária(s), $s_1, s_2, ..., s_n$;
7: Atualizar o vetor de pesos de s_1 e dos seus vizinhos;
8: Criar novas unidades se for necessário (no caso de redes construtivas);
9: **até** que o nível de erro seja aceitável.
10: **Fim**

Para usar a técnica SAM-SOM híbrida, é necessário modificar apenas os passos correspondentes à busca da unidade vencedora e à atualização dos vetores de pesos no Algoritmo 5.1. Por exemplo, se o objetivo fosse utilizar a *R-Tree* com o algoritmo de treinamento da GNG (ver Pág. 12), o passos 2.3.5 e 2.3.5 do treinamento de GNG devem ser realizados de acordo com o Algoritmo 5.2. As modificações no algoritmo estão sublinhadas.

Algoritmo 5.2 Modificação do algoritmo GNG para convertê-lo em *R-Tree+GNG*.

...
2.3.5 Determinar, através da estrutura *R-Tree*, os dois neurônios mais próximos, s_1 e s_2, em relação a ξ ;
...
2.3.5 Atualizar o vetor de pesos de s_1 e os vetores de pesos dos seus vizinhos e atualizar os elementos correspondentes que os representam na estrutura *R-Tree* ...
...

Os métodos de acesso podem ser de grande utilidade no processo de encontrar a unidade vencedora sem ter que recorrer à comparação seqüencial. Por exemplo, na Figura 5.2, pode-se observar, em vermelho, a área de cobertura de uma consulta por abrangência (*Range Query*). Na seqüência de Figuras 5.3, 5.4 e 5.5, pode-se observar os retângulos atingidos pela consulta no primeiro, segundo e terceiro nível da árvore, respectivamente. Do lado direito de cada figura, encontra-se a sub-árvore correspondente envolvida na consulta realizada. O descarte ou inclusão de um retângulo, na área atingida pela consulta, é realizado analisando se existe intersecção entre a área da consulta e o retângulo. Dessa forma, a informação é recuperada com um custo computacional muito menor que a tradicional busca seqüencial.

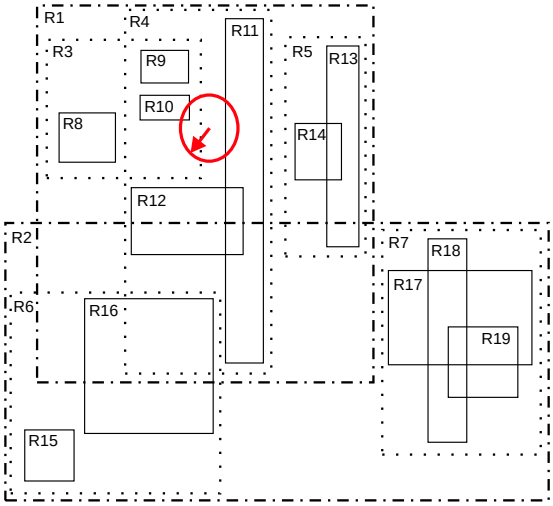


Figura 5.2: Consulta por abrangência em uma *R-Tree*.

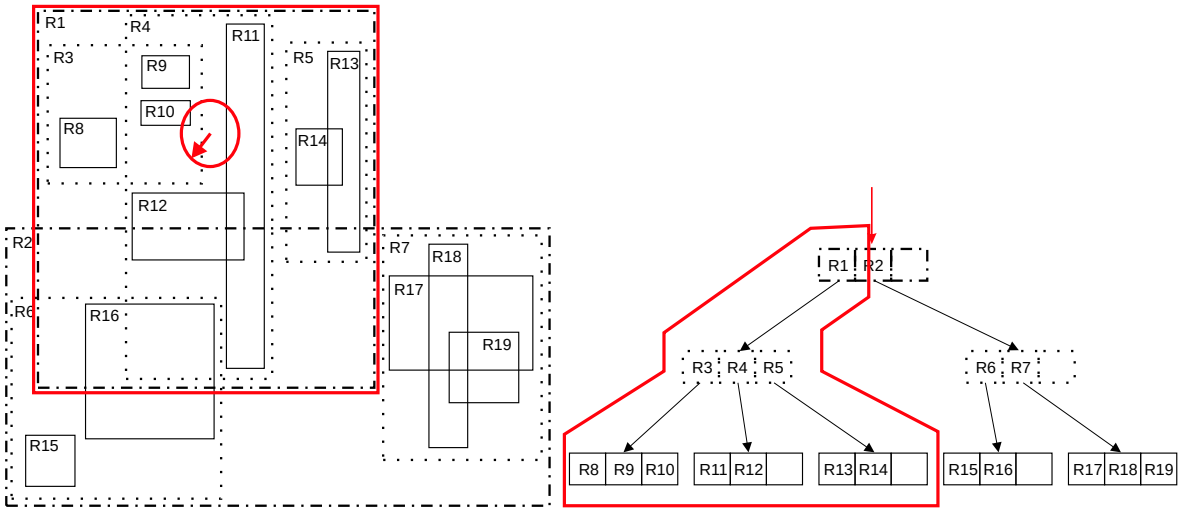


Figura 5.3: Áreas atingidas, no primeiro nível da estrutura, pela consulta por abrangência da Figura 5.2.

No Algoritmo 5.3, apresenta-se a seqüência de passos para criar uma rede com as técnicas SAM-SOM∗ e MAM-SOM∗. Ao mesmo tempo, o algoritmo introduz o parâmetro ϕ, que representa o número de unidades mais próximas com as quais um novo elemento deve ser conectado quando é criado. A variável ϕ pode ser vista como o fator de conectividade para as novas unidades.

A idéia que está sendo proposta leva em conta os seguintes casos:

1. para o treinamento de redes SOM em lotes (*batch learning*), a estrutura de dados escolhida deve ser reconstruída completamente após cada iteração, pois todos os pesos são atualizados simultaneamente;

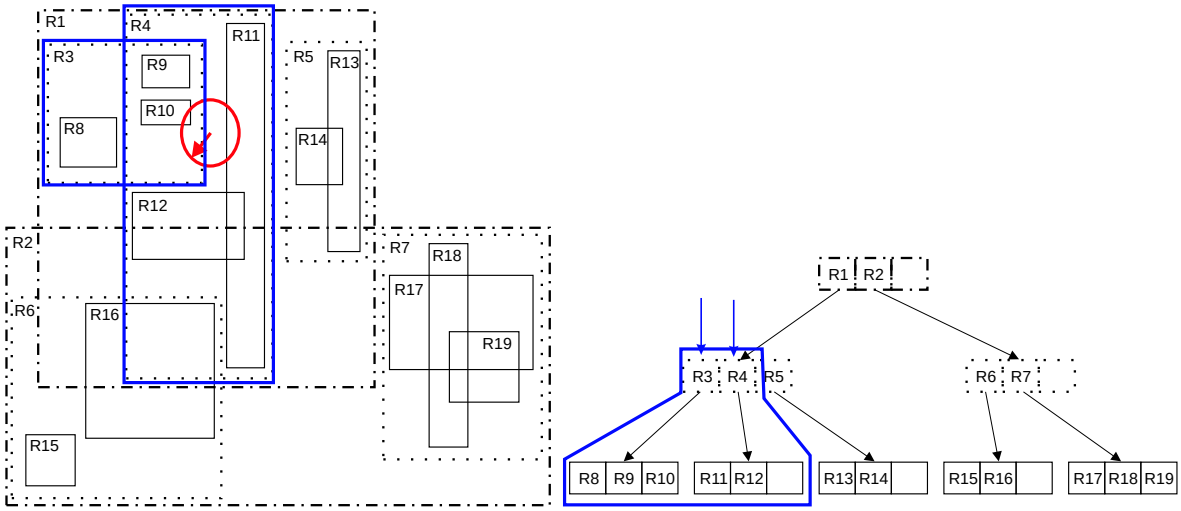


Figura 5.4: Áreas atingidas, no segundo nível da estrutura, pela consulta por abrangência da Figura 5.2.

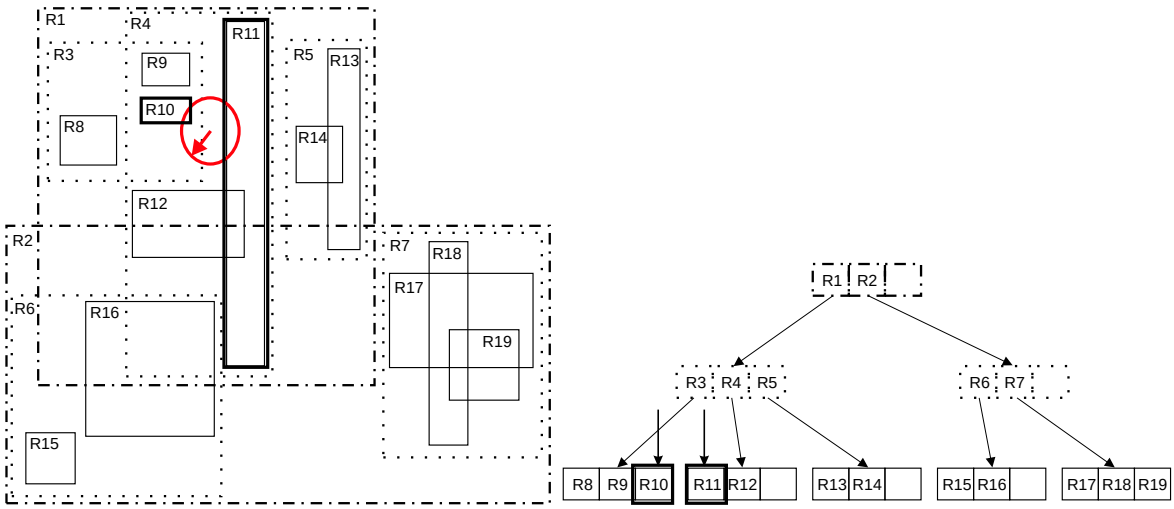


Figura 5.5: Áreas atingidas, no terceiro nível da estrutura, pela consulta por abrangência da Figura 5.2.

Algoritmo 5.3 Treinamento de uma rede SAM-SOM∗ ou MAM-SOM∗

1: **Início**
2: Inicializar a rede com zero unidades;
3: **para todo** padrão ξ_i **faça**
4: Criar uma nova unidade com posição inicial igual a ξ_i e adicioná-la ao MA usado;
5: Encontrar, através da estrutura, os ϕ-vizinhos mais próximos $(n_1,\ n_2, \ldots, n_\phi)$ de ξ_i
6: Conectar ξ_i com n_i, $\forall i = 1, 2, \ldots, \phi$;
7: **fim para**
8: **Fim**

2. para os casos de treinamento *on-line*, onde apenas um pequeno grupo de vetores de pesos são atualizados ao mesmo tempo, apenas os nós correspondentes a esses pesos da estrutura são modificados.

Para poder encontrar as unidades mais próximas utilizando um MAE ou MAM, durante o treinamento da rede, existe uma grande variedade de algoritmos que podem ser utilizados (Uhlmann, 1991a; Roussopoulus et al., 1995; Ciaccia et al., 1997c; Clarkson, 1999; Yianilos, 1999, 2000). Porém, essa operação também pode ser realizada em função de buscas por abrangência (*Range Query*) com um raio apropriado. Nesse último caso, o problema principal é calcular um raio apropriado para poder reduzir o número de operações e o tempo para recuperar a informação. Nos experimentos realizados neste trabalho, foi adotada a segunda técnica com um raio inicial empiricamente determinado. Porém, pode ser utilizada alguma técnica para estimativas de seletividade (Traina Jr et al., 2000a).

As técnicas SAM-SOM e MAM-SOM apresentadas nesta tese são a generalização da idéia apresentada por Cuadros-Vargas e Romero (2002). Nesse trabalho, foram apresentados resultados da incorporação de *R-Tree* em redes do tipo *Growing Neural Gas* (GNG). Nesse estudo, para cada neurônio na rede GNG, existe seu correspondente nó da *R-Tree*. Os k-vizinhos mais próximos são determinados através de uma busca na *R-Tree*.

Para utilizar *k-d-Tree* ou qualquer outro MAE, o princípio é o mesmo, isto é, os k-vizinhos mais próximos são determinados através do algoritmo fornecido pelo próprio método de acesso. A diferença que possa existir, em termos de desempenho, deve-se principalmente, ao custo computacional específico de cada algoritmo para recuperar a informação.

Durante o processo de treinamento, os objetos contidos podem ser mudados de posição com freqüência. Dependendo do MAE utilizado, a modificação de um objeto pode ser um processo simples ou complexo. Por exemplo, no caso dos *k-d-Tree*, existem problemas para a remoção de unidades (Bentley, 1975; Samet, 1995). Além disso, as árvores *k-d-Tree* geram árvores não balanceadas, prejudicando consideravelmente, o processo de recuperação de informação (ver Seção 3.3.1 Pág. 32).

Nos experimentos apresentados a seguir foram escolhidos os métodos de acesso *R-Tree* e *k-d-Tree* para poder demonstrar que, mesmo considerando as limitações conhecidas desses métodos de acesso, o ganho, em termos de desempenho, é considerável em relação ao processamento seqüencial. Porém, existem muitos outros MAE que poderiam vir a ser utilizados, tais como as apresentadas nos trabalhos de Gaede e Günther (1998) e Samet (1995).

Uma descrição das técnicas SAM-SOM e MAM-SOM também podem ser encontradas em Cuadros-Vargas et al. (2004).

5.3 Experimentos

Os experimentos foram realizados visando comparar, em termos de cálculos de distâncias, o método seqüencial, e alguns SOM utilizando alguns métodos de acesso.

Esta seção está dividida em duas partes. A primeira envolve experimentos com a técnica SAM-SOM híbrida com os algoritmos k-Médias (k-*Means*), mapas de Kohonen tradicionais e redes GNG. A segunda parte apresenta experimentos com as técnicas SAM-SOM$*$ e MAM-SOM$*$.

5.3.1 Aplicando a técnica SAM-SOM híbrida

A técnica SAM-SOM híbrida é o resultado da incorporação de MAE no processo de treinamento de SOM. Nessa técnica, os MAE são utilizados apenas para encontrar a unidade vencedora. O resto do processo do treinamento do SOM continua sem modificações.

O método seqüencial foi utilizado porque tradicionalmente os SOM o utilizam. O método de acesso *k-d-Tree* foi utilizado devido a recentes experimentos apresentados por (Gray e Moore, 2000). Finalmente, escolheu-se o método de acesso *R-Tree* porque, na comunidade científica de métodos de acesso, a *R-Tree* é um dos principais pontos de referência utilizados para as novas técnicas apresentadas.

Os experimentos foram realizados utilizando os seguintes bancos de dados:

ABALONE: Este banco de dados está composto por 4177 vetores 8-D, os quais descrevem formas. O arquivo foi obtido do *UCI-Irvine repository of machine learning databases and domain theories*[1];

LETTERS: Este banco de dados está composto por 20986 vetores 17-D, os quais descrevem formas de caracteres escritos a mão. O arquivo foi obtido do *UCI-Irvine repository of machine learning databases and domain theories*[2];

FACES: Este banco de dados está composto por 11900 vetores 16-D, os quais descrevem propriedades de faces. Este conjunto de dados foi obtido do Projeto *Informedia* da *Carnegie Mellon University* (Wactlar et al., 1996, 1999);

IMAGES: Este conjunto de dados está composto por 80000 vetores de características de imagens em dimensão 1215. Este conjunto de dados foi obtido do Projeto *Informedia* da *Carnegie Mellon University* (Wactlar et al., 1996, 1999).

Para os experimentos com aprendizado em lotes (*batch learning*), foram realizados experimentos envolvendo k-Médias (MacQueen, 1967; Moody e Darken, 1989), os Mapas de Kohonen (1982) e *Growing Neural Gas* (Fritzke, 1995b). Porém, para aqueles métodos com aprendizado *on-line*, onde a operação de remoção e reinserção de elementos é contínua, as árvores *k-d-Tree* não puderam ser testadas. Isso aconteceu devido ao fato de que, durante o

[1] `ftp://ftp.ics.uci.edu/pub/machine-learning-databases/abalone/`
[2] `ftp://ftp.ics.uci.edu/pub/machine-learning-databases/letter-recognition/`

treinamento de um SOM, as posições dos vetores contidos na estrutura são modificadas com muita freqüência. Esse problema poderia ser inicialmente resolvido por uma remoção seguida de uma reinserção do elemento, mas a remoção nessa estrutura pode obrigar à reconstrução total, afetando consideravelmente o tempo de treinamento.

Nas Figuras 5.6-5.11, pode-se observar o número de cálculos de distância, para poder encontrar a unidade vencedora, em função do número de agrupamentos para o método k-Médias. Esses experimentos foram realizados utilizando o conjunto de dados ABALONE, FACES e LETTERS. Nas Figuras 5.6-5.11, é importante observar que no eixo x, depois dos valores 90 e 100, os próximos pontos correspondem a 200 e 300.

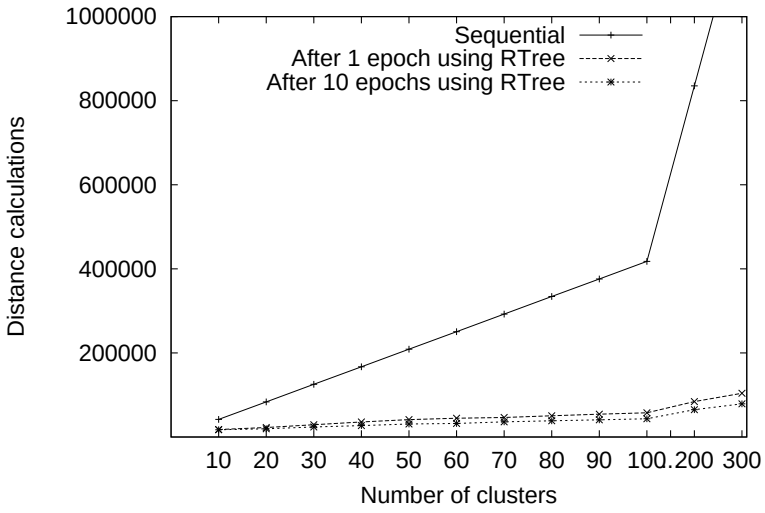


Figura 5.6: Número de cálculos de distância por ciclo em função do número de agrupamentos no treinamento *on-line* de k-Médias no conjunto de dados ABALONE.

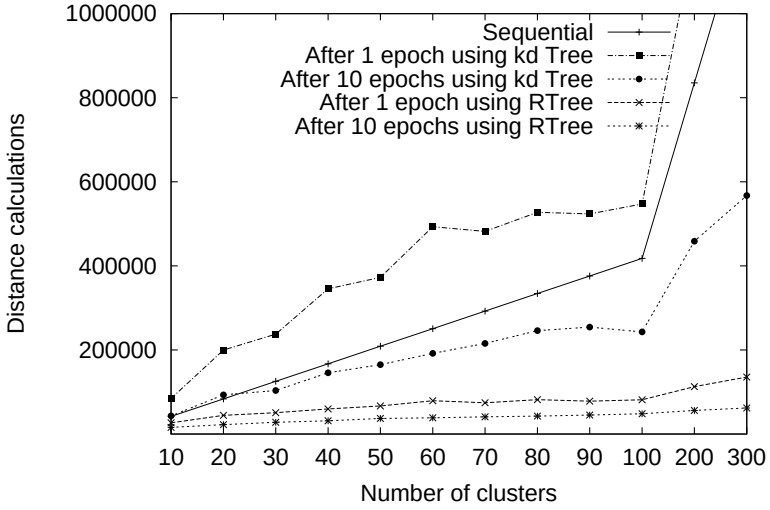


Figura 5.7: Número de cálculos de distância por ciclo em função do número de agrupamentos no treinamento em lotes de k-Médias no conjunto de dados ABALONE.

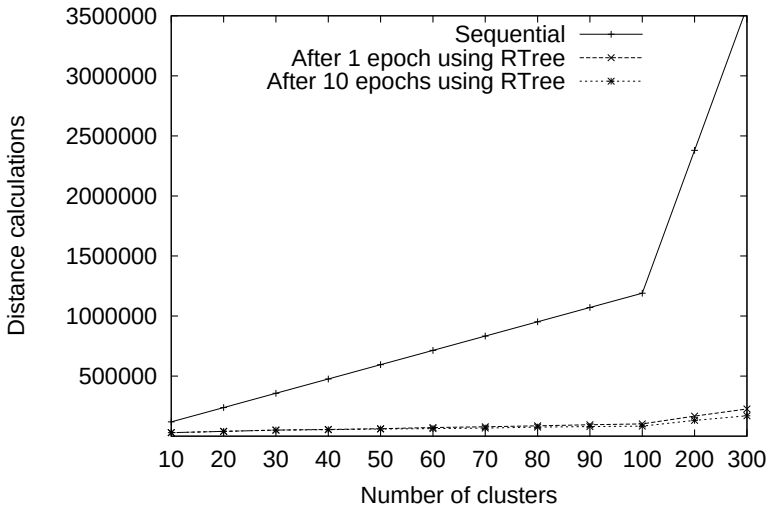


Figura 5.8: Número de cálculos de distância por ciclo em função do número de agrupamentos com treinamento *on-line* de k-Médias para o conjunto de dados FACES.

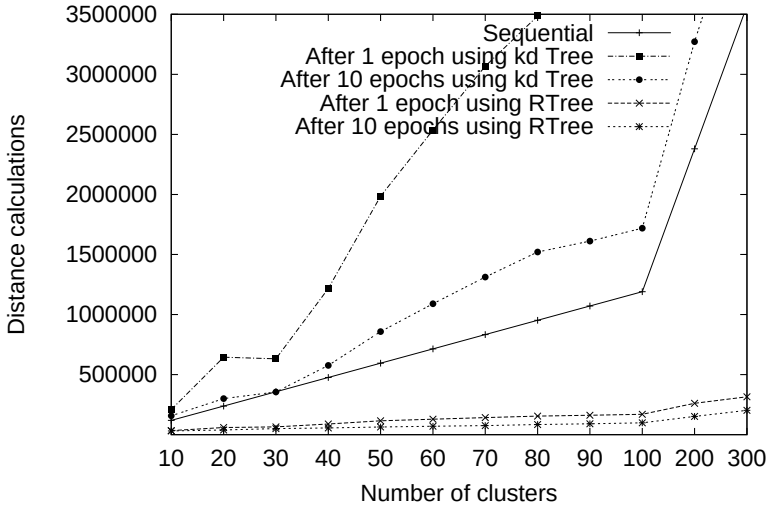


Figura 5.9: Número de cálculos de distância por ciclo em função do número de agrupamentos com treinamento em lotes de k-Médias para o conjunto de dados FACES.

No caso da utilização de MAE existe um processamento extra para atualizar os pesos na estrutura mas, mesmo assim, o desempenho obtido usando o MAE foi muito superior ao obtido utilizando apenas processamento seqüencial. Também é possível observar uma diferença considerável entre o desempenho da *k-d-Tree* e da *R-Tree*. Essa diferença surge principalmente porque o algoritmo de inserção de elementos em uma *R-Tree* gera uma árvore balanceada. No caso da *k-d-Tree*, dependendo da ordem de inserção dos elementos, a estrutura resultante poderia ser uma lista encadeada, deteriorando assim o processo de recuperação de informação.

Na Figura 5.11, pode-se observar que a linha do processamento seqüencial, comparada com a linha da *R-Tree*, apresenta um melhor desempenho apenas no primeiro ciclo. Isso acontece porque o algoritmo utilizado envolve buscas por abrangência (*Range Queries*) com raios iniciais calculados em função da distribuição do conjunto de dados. Como os agrupamentos iniciais estão em posições aleatórias, a distribuição inicial não corresponde à distribuição dos dados e isso aumenta o tempo necessário para encontrar a unidade vencedora. Para solucionar este problema poderia ser utilizado o processamento seqüencial apenas nos primeiros ciclos.

Ainda nas Figuras 5.10 e 5.11, observa-se que, quando o número de agrupamentos aumenta, a *R-Tree* consegue aumentar ainda mais a proporção de tempo ganho. Isso pode ser claramente percebido naqueles pontos que representam 200 e 300 agrupamentos.

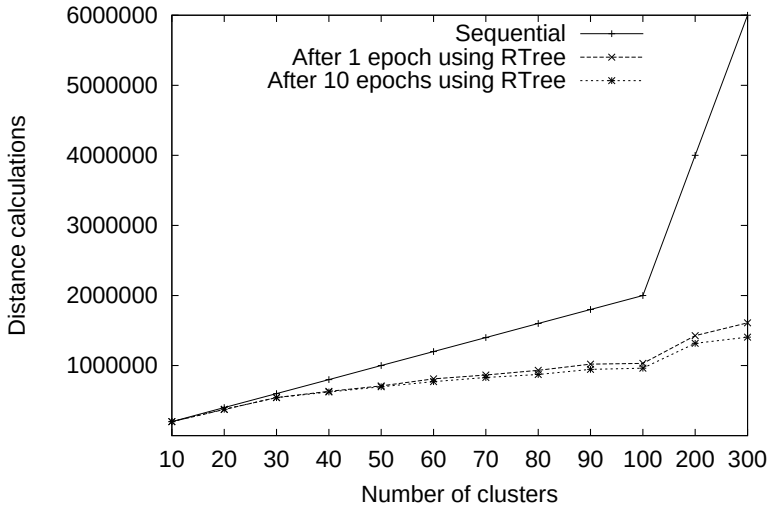


Figura 5.10: Número de cálculos de distância por ciclo em função do número de agrupamentos com treinamento *on-line* de k-Médias para o conjunto de dados LETTERS.

Na Figura 5.12, observa-se o resultado da utilização de MAE junto às redes tradicionais de Kohonen (1982). O número de cálculos de distância necessários durante o treinamento é apresentado em função do número de padrões, utilizando-se o SOM com processamento seqüencial e o SOM com a *R-Tree*. Apesar do tamanho dessa rede ser de apenas 10×10, a diferença em termos de desempenho é considerável. Em uma aplicação real, com bancos de dados volumosos e uma rede maior, a diferença seria ainda mais acentuada.

Na Figura 5.13, apresenta-se o resultado de um experimento similar, utilizando redes do tipo GNG com a abordagem seqüencial e utilizando *R-Tree*. Na Figura 5.13(a), pode-se observar o número de cálculos de distância para o número de unidades representado no eixo x. Na Figura 5.13(b) pode-se observar o número de cálculos de distância acumulado desde o início do treinamento até atingir o número de unidades, representadas no eixo x. Desde o início do treinamento até a rede atingir 10 unidades, só foi utilizado processamento

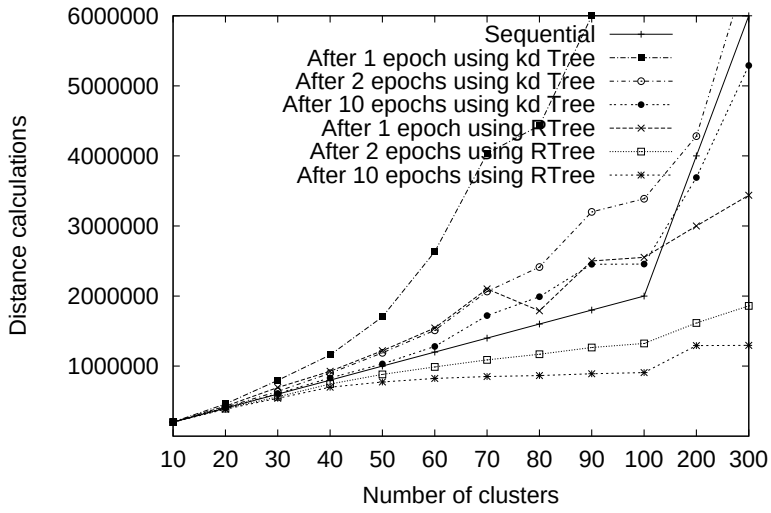


Figura 5.11: Número de cálculos de distância por ciclo em função do número de agrupamentos com treinamento em lotes de k-Médias para o conjunto de dados LETTERS.

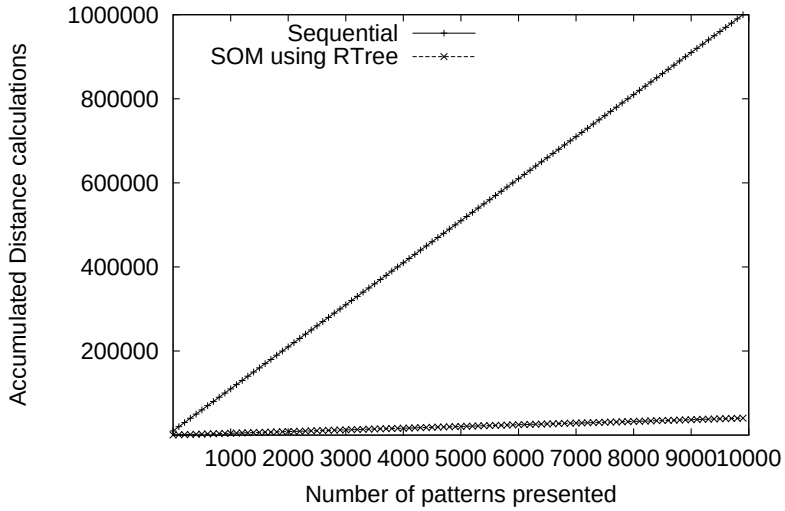
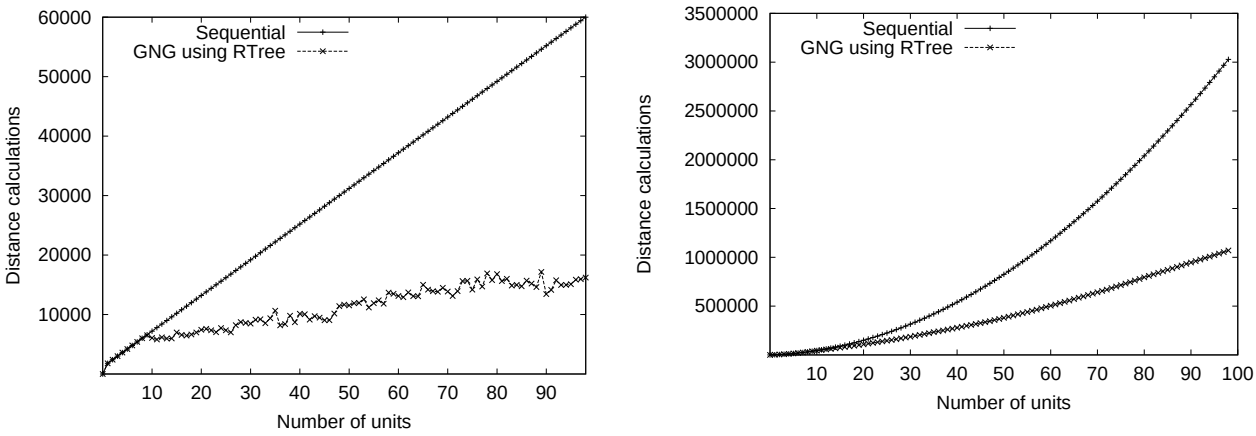


Figura 5.12: Número de cálculos de distância em função do número de padrões apresentados utilizando um SOM de dimensão 10×10 com treinamento *on-line* com o conjunto de dados IMAGES.

seqüencial em ambos os casos. Isto é devido ao fato de que, com poucas unidades, o ganho em termos de cálculos não foi significativo.

No caso dos modelos k-Médias, rede de Kohonen e GNG, somente é necessário encontrar um vizinho mais próximo. Isto ocorre porque, no caso das redes de Kohonen e GNG, a vizinhança está determinada pelas conexões da unidade vencedora e, no caso do algoritmo k-Médias, somente é atualizada uma única unidade por vez. Porém, a mesma técnica poderia ser utilizada para encontrar mais de um vizinho mais próximo.

Como já foi discutido na Seção 5.1, se a taxa de inserção dos elementos, λ, for reduzida ao mínimo, isto é, $\lambda = 1$, a rede resultante será deformada de forma gradual, como apresentado



(a) Cálculos de distância desde a inserção da unidade anterior.

(b) Número de cálculos de distância acumulado até a n-ésima unidade

Figura 5.13: Número de cálculos de distância para uma rede *Growing Neural Gas* utilizando o conjunto de dados IMAGES. Os parâmetros utilizados para esta simulação são: $\lambda = 600$, $\mu_b = 0.05$, $\mu_n = 0.0006$, $\alpha = 0.5$, $\beta = 0.0005$ e $a_{max} = 100$.

na Figura 5.14. Na próxima seção, experimentos similares com $\lambda = 1$ são apresentados com a única diferença de que, desta vez, serão utilizados métodos de acesso para evitar a deformação e melhorar a qualidade da rede resultante.

5.3.2 Aplicando as técnicas SAM-SOM∗ e MAM-SOM∗

No caso específico de redes construtivas, um dos pontos críticos é determinar quando e em que posição uma nova unidade deve ser inserida. Na rede GNG (Fritzke, 1995b), a velocidade de inserção está controlada pelo parâmetro λ, cujo valor recomendado pelos autores é 600.

De acordo com a Equação (5.4) (ver Pág. 71), o número total de cálculos de distâncias, NCD_{CSOM}, em uma rede SOM construtiva que insere elementos a cada λ padrões apresentados é:

$$NCD_{CSOM} = \sum_{i=NU_0}^{N-1} i\lambda \tag{5.6}$$

A redução de λ permite a criação de uma maior quantidade de unidades na rede mas, ao mesmo tempo, reduz o tempo para adaptar as novas unidades à estrutura existente. Na Figura 5.14, observa-se uma rede GNG treinada com $\lambda = 1$. A rede resultante está visivelmente deformada e contém várias unidades em posições inapropriadas para a distribuição dos dados. Como já foi explicado anteriormente, a deformação da rede deve-se, entre outros fatores, ao fato de que as posições das novas unidades são resultado da interpolação de outras duas unidades.

Então, um dos problemas a serem resolvidos, para evitar a deformação da rede, é criar unidades em posições onde sejam realmente úteis para a distribuição dos dados. Isto é

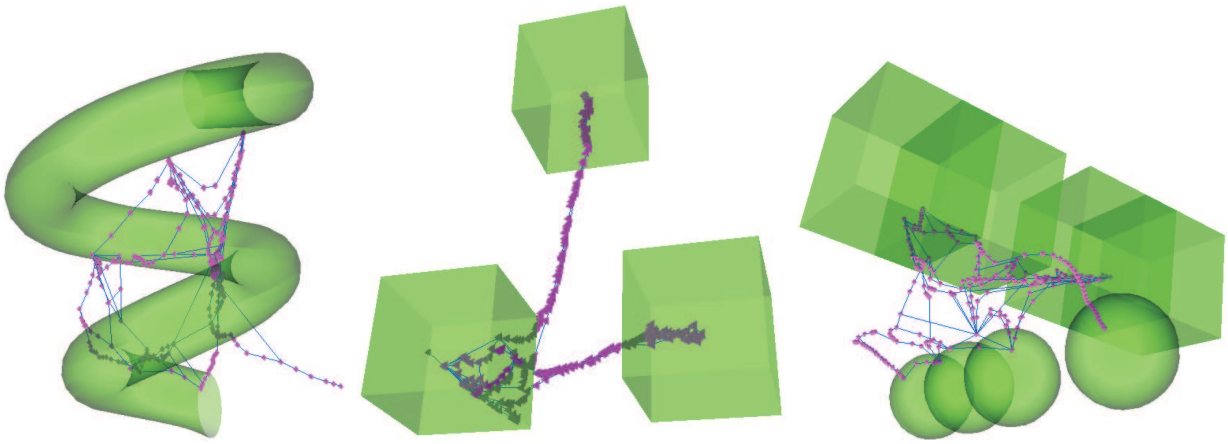


Figura 5.14: Redes GNG tradicionais treinadas com $\lambda = 1$.

necessário porque a técnica da interpolação cria unidades em posições onde nem sempre existem dados. Para resolver esse problema, o Algoritmo 5.3 (ver Pág. 76) cria uma nova unidade para cada padrão apresentado.

Na Figura 5.15, pode-se observar uma rede do tipo GNG depois de treinada com a técnica SAM-SOM$*$. Neste caso em particular, a combinação SAM-SOM utilizada foi de uma estrutura *R-Tree* com o algoritmo GNG.

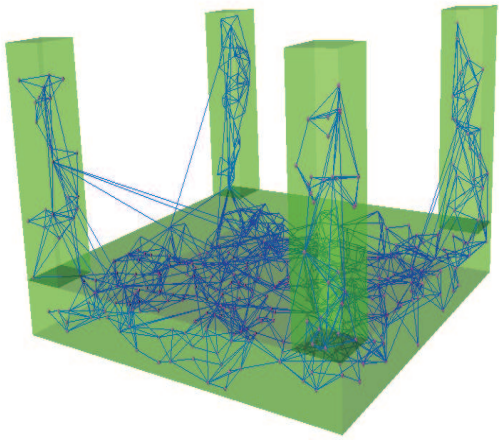


Figura 5.15: Rede *R-Tree+GNG* com 5000 padrões e $\phi = 3$.

Na Figura 5.15, também observa-se algumas conexões relativamente grandes. Isso acontece uma vez que, no Algoritmo 5.3, as novas unidades são conectadas aos ϕ vizinhos mais próximos, porém, no início do processo de treinamento, as primeiras unidades não estarão necessariamente próximas umas das outras.

Um outro ponto, que agora pode ser melhor entendido refere-se à família de redes MAM-SOM, isto é, a combinação de um MAM com redes do tipo SOM. Como já foi explicado nas Seções 3.4 e 5.2, no espaço métrico não existe o conceito de coordenadas (próprio de espaços vetoriais). A única informação disponível é fornecida pela função de

distância métrica. Por causa disso, o procedimento de correção de vetores de pesos dos SOM não poderia ser aplicado na sua forma tradicional. Entretanto, no Algoritmo 5.3, observa-se que não existe o processo de correção de vetores de pesos, deixando aberta a possibilidade de usar ambos métodos de acesso, espacial (MAE) e métrico (MAM).

O fato de ter criado 5000 unidades na rede GNG da Figura 5.15 leva a crer que o número de cálculos de distância também é grande. No entanto, isto não ocorre, como pode ser observado na Tabela 5.1. No caso da GNG tradicional, observa-se claramente um custo quadrático, de acordo com a Equação (5.5) (ver Pág. 71).

No de padrões	*R-Tree+GNG*	GNG
500	7844	124749
1000	15617	499499
1500	23884	1124249
2000	32191	1998999
2500	40926	3123749
3000	49548	4498499
3500	59109	6123249
4000	67750	7997999
4500	77007	10122749
5000	85439	12497499

Tabela 5.1: Comparação em termos de cálculos de distância entre *R-Tree+GNG* e GNG com $\lambda = 1$.

Vale a pena ressaltar que essa técnica poderia ser aplicada com qualquer outra rede do tipo SOM incluindo os modelos construtivos e hierárquicos. Também vale a pena ressaltar que, no caso da *R-Tree+GNG*, os k-vizinhos mais próximos são calculados em função de buscas por abrangência e, em vários casos, foi preciso realizar mais de uma busca para encontrar os k-vizinhos mais próximos. Dependendo do MAM ou MAE utilizado para recuperar os vizinhos mais próximos, o número de cálculos de distância poderia ser ainda menor.

Essa mudança no algoritmo tradicional de redes do tipo SOM confere às famílias de técnicas SAM-SOM e MAM-SOM uma série de propriedades apresentadas a seguir.

5.4 Propriedades das técnicas SAM-SOM e MAM-SOM

No caso da técnica SAM-SOM híbrida, o algoritmo das redes SOM apenas é modificado na fase de busca da unidade vencedora. Ao invés de realizar a busca de forma seqüencial, o algoritmo utiliza um MAE. A principal conseqüência dessa mudança no algoritmo original é

a aceleração do processo de treinamento e a possibilidade de trabalhar com redes maiores. O resto do processo continua praticamente sem alterações.

As técnicas SAM-SOM∗ e MAM-SOM∗, além de ajudar na aceleração do processo, apresentam outras características. Nesta seção, as principais propriedades dessas técnicas são apresentadas com ênfase especial em relação às propriedades mencionadas no Capítulo 4.

Escalabilidade: permite a criação de um número maior de unidades do que uma rede similar derivada do modelo de Kohonen;

Menor custo computacional: apesar do maior número de unidades, o custo computacional é consideravelmente menor que as redes SOM que utilizam a comparação seqüencial;

Reflete a distribuição dos dados: a rede resultante reflete melhor a distribuição dos dados, pois o algoritmo garante que unidades novas sejam criadas de acordo com a distribuição dos dados e não por interpolação, etc. Isso pode ser observado comparando-se as Figuras 5.14 e 5.15;

Maior informação nas conexões: em redes derivadas do modelo de Kohonen e nos modelos derivados da rede GNG (com distribuições não retangulares), as conexões são úteis apenas para a visualização e para saber quais são os vizinhos. No caso das técnicas SAM-SOM∗ e MAM-SOM∗, as conexões podem conter a distância, uma vez que não existe mudança nos vetores de pesos. A distância é uma informação muito útil para a recuperação de informação;

Maior informação para prever consultas: a estimativa de seletividade de uma consulta é um tema muito importante na área de recuperação de informação (Belussi e Faloutsos, 1995, 1998; Traina Jr et al., 2000a). Com a informação das distâncias dos vizinhos mais próximos, é mais fácil realizar cálculos de estimativas de seletividade. Nos MAM e MAE tradicionais, os objetos inseridos na estrutura não armazenam informação relacionada aos vizinhos mais próximos;

Maior informação para a detecção de agrupamentos: considerando que a função de distância representa a medida de dissimilaridade entre os elementos, a partir dessa técnica pode se estudar a possibilidade de detectar agrupamentos nos dados. Uma proposta simples poderia ser eliminar as conexões, começando-se pelas maiores em ordem decrescente de distâncias. Cada vez que a eliminação de uma conexão gera dois grupos separados, os dois novos grupos podem ser considerados dois agrupamentos distintos, de forma análoga às redes *TreeGCS* (ver Seção 2.4.3, Pág. 18) ou ao processo utilizado para a partição dos nós utilizado pela *Slim-Tree* (ver Seção 3.4.15, Pág. 46). Também é importante considerar que mesmo os trabalhos recentes relacionados com

este problema, como o apresentado por Jain (1999), não abordam o caso de acesso espacial ou métrico;

Maior estabilidade-plasticidade: do ponto de vista de redes neurais, especificamente em relação ao dilema da estabilidade-plasticidade (Grossberg, 1972, 1976a,b), redes do tipo SAM-SOM∗ e MAM-SOM∗ são estáveis, pois um padrão sempre é representado pelo mesmo neurônio. A plasticidade também é uma propriedade presente, pois a inserção de novos padrões não provoca a perda do conhecimento adquirido anteriormente;

Aprendizado incremental: dependendo do dinamismo existente no MAE ou MAM utilizado, o modelo proposto permite a inserção de novos elementos em qualquer instante, sem que isto obrigue a reinicialização do processo em nenhum dos casos;

Independência em relação à ordem de inserção: Esta é uma propriedade herdada em função da utilização de MAE e MAM, os quais sempre têm por objetivo que a estrutura resultante não dependa da ordem de inserção dos dados.

5.5 Considerações Finais

Neste capítulo foi apresentada a forma em que os MAE e MAM podem ser incorporados no treinamento de redes SOM, culminando no desenvolvimento das técnicas SAM-SOM e MAM-SOM propostas neste trabalho.

Os MAE podem ser incorporados em SOM tradicionais apenas para encontrar a unidade vencedora, mantendo a correção de pesos sem alterações. Essa técnica é denominada SAM-SOM Híbrida. Devido ao fato de que nos espaços métricos não existe o conceito de coordenadas nos objetos a serem tratados, os MAM não podem ser aplicados de forma direta em um processo como o treinamento de SOM. Isto acontece porque a correção dos pesos é realizada em vetores (com coordenadas vetoriais). Para solucionar esse problema foram apresentadas as técnicas SAM-SOM∗ e MAM-SOM∗. A idéia proposta é eliminar a correção de pesos e inserir uma unidade na rede para cada novo padrão apresentado. Tudo isso foi realizado com um ganho de custo computacional. Isso é possível, mesmo com a existência de um número maior de unidades, porque a busca pela unidade vencedora é sempre realizada através do método de acesso e não de forma seqüencial.

Como não existe mais a correção de vetores de pesos e o algoritmo para criação de SAM-SOM∗ e MAM-SOM∗ não depende da existência de coordenadas, ambos os métodos de acesso, vetorial e métrico, podem ser utilizados sem restrições. Em seguida, foram apresentados diversos experimentos que demonstraram a diferença dramática em termos de desempenho e tempo ganho dessas técnicas em comparação à abordagem seqüencial.

Finalmente, foram também apresentadas as principais propriedades conhecidas que as técnicas SAM-SOM e MAM-SOM propostas possuem. Essas propriedades foram analisadas com especial ênfase nas vantagens e limitações dos SOM, apresentadas no Capítulo 4.

Hoje em dia, os problemas reais envolvem bancos de dados cada vez maiores. Ao mesmo tempo, a função de distância entre dois objetos é cada vez mais complexa. Por todos esses motivos, evitar ou reduzir ao máximo a comparação seqüencial é muito relevante para continuar pensando em bancos de dados cada vez maiores e mais complexos. Considerando esses pontos, as técnicas SAM-SOM e MAM-SOM provavelmente representam a próxima geração de mapas auto-organizáveis.

Capítulo 6

Métodos de Acesso baseados no comportamento do usuário

Neste capítulo, é investigada a possibilidade de tornar os Métodos de Acesso (MAs) capazes de adaptar-se ao comportamento do usuário. Será apresentada uma técnica que permite reduzir, gradualmente, o número de cálculos de distância necessárias para recuperar informação através de um Método de Acesso Espacial ou Métrico.

6.1 Análise do problema

Para poder reduzir gradualmente o número de cálculos de distância necessários para realizar uma consulta é necessário que o método de acesso seja capaz de adaptar-se ao comportamento do usuário, isto é, que aproveite a distribuição de consultas realizadas. Existem alguns trabalhos na literatura que tentaram criar estruturas adaptativas. Dentre os principais, pode-se citar os *Adaptive B-Tree* (Baeza-Yates, 1990), *Self-Organizing Data Structures* (Albers e Westbrook, 1996) e os *Self-adjusting binary search trees* (*Splay-Trees*) (Sleator e Tarjan, 1985). Porém, nenhuma dessas propostas é orientada a MAE ou MAM.

No caso de consultas por similaridade em dados complexos, o fator que consome mais tempo na Equação (1.2) (ver Pág. 2) é a função de distância. Um caso claro desse problema pode ser observado no cálculo da distância métrica entre duas seqüências de DNA. Por esse motivo, todos os Métodos de Acesso (MAs) tentam evitar ou pelo menos reduzir, à medida do possível, esse tipo de cálculo.

Uma técnica amplamente utilizada para evitar os cálculos de distância em uma consulta é a desigualdade triangular, apresentada por Burkhard e Keller (1973) (ver Seção 3.4.1). Essa técnica permite apenas reduzir o número de cálculos de distância. Mesmo assim, o MA não é capaz de reaproveitar o conhecimento gerado por uma determinada consulta para melhorar o desempenho de consultas futuras.

Para entender melhor o problema, pode-se imaginar o seguinte cenário: dado um MAM, chamado $MAMX$ e uma consulta por abrangência, $RQ_1(O_q, r)$, onde O_q é o objeto de busca

e r é o raio. Quando a consulta RQ é executada, o $MAMX$ precisará de um número n de cálculos de distância. O problema aparece ao tentar executar uma segunda consulta, RQ_2, que apresente intersecção significativa com a área coberta por RQ_1. Certamente, parte dos cálculos de distância gerados por RQ_1 poderiam ser úteis para reduzir o processamento necessário em RQ_2. No entanto, na realidade isso não acontece, pois todas as distâncias calculadas por RQ_1 são descartadas. O mesmo problema aparece se a mesma consulta, RQ_1, for repetida várias vezes de forma contínua. Em todos os casos, o número de cálculos de distância será n.

Ambos os problemas, com a mesma consulta e com uma consulta similar, estão presentes em todos os Métodos de Acesso citados no Capítulo 3. Neste capítulo, esse problema é investigado e uma solução é proposta.

6.2 Algumas soluções existentes

Na área de Inteligência Artificial, pode-se observar alguns trabalhos relacionados com o problema de inserir inteligência em sistemas computacionais. No final da década de 50 e início da década de 60, Arthur Samuel realizou um trabalho, pioneiro nessa área, utilizando o clássico jogo de damas como base do seu experimento (Samuel, 1959, 1967). O objetivo desse programa era que o computador aprendesse a jogar damas para poder competir com seres humanos. Baseados nesse trabalho, Schaeffer et al (1992) criaram *Chinook* que é um programa com a capacidade de aprimorar sua técnica de jogo em função das jogadas do adversário, especialmente aquelas jogadas que desencadeavam a perda do jogo. Uma vez que o programa foi treinado com campeões nesse jogo, na prática, *Chinook* virou um adversário quase imbatível. A técnica utilizada por este programa foi a de armazenar as distribuições de peças no tabuleiro a partir do momento que existisse apenas 8 peças. Também foi considerada a informação relacionada às partidas nas quais o adversário humano ganhava. Isto foi realizado com o intuito de detectar a jogada que levou à perda do jogo para não repeti-la no futuro. O bom desempenho do programa *Chinook* deve-se ao fato que ele memoriza todas essas jogadas. Contudo, essa proposta, ainda não pode ser considerada como um comportamento inteligente. No caso de Recuperação de Informação, a idéia que o *Chinook* inspira é a de criar um MA capaz de melhorar a si próprio, ao longo do tempo, em função da experiência que obtém através das consultas.

Na área de Recuperação de Informação (RI) através de Métodos de Acesso (MAs), existem algumas tentativas para reduzir o número de cálculos de distância. A grande maioria foi apresentada no Capítulo 3. No trabalho de Shasha e Wang (1990), os autores propõem que duas matrizes de distâncias sejam criadas antes de começar as consultas. Cada célula da primeira matriz contém a distância ou o limite inferior para aquela distância. A única diferença na segunda matriz é que esta armazena os limites superiores. Quando uma

distância é conhecida, a mesma é armazenada de forma direta, mas se não for conhecida, será aproximada em função das distâncias existentes.

Um outro trabalho na mesma direção é o AESA (Vidal, 1986), que propõe que a matriz triangular de distâncias seja armazenada completamente (ver Seção 3.4.2, Pág. 39). O custo computacional de construção desse método é de $O(n^2)$. Em compensação, o tempo experimental de recuperação de uma distância é de apenas $O(1)$. O grande problema desta proposta é que, na prática, é inviável realizar essa quantidade de cálculos e manter uma matriz que cresce de forma quadrática. Além disso, espera-se que novos elementos possam ser inseridos ou removidos de forma dinâmica na estrutura a qualquer instante.

Uma proposta posterior do mesmo trabalho é conhecida como LAESA (Micó et al., 1994) (ver Seção 3.4.5, Pág. 40). Nesse caso, os autores propõem reduzir a matriz de distâncias através da utilização de um conjunto reduzido de objetos fixos que funcionam como âncoras. Para qualquer novo objeto, O_i, apenas são calculadas as distâncias de O_i até os objetos âncoras. Para a recuperação de informação, as distâncias existentes aos objetos âncoras são utilizadas para reduzir o conjunto de respostas através da desigualdade triangular. O custo computacional de construção neste caso é de $O(kn)$, sendo k o número de âncoras e n o número de objetos. Essa idéia também foi explorada na *OmniSequential* (Santos-Filho et al., 2001). Neste último caso, cada objeto âncora é chamado de *focus* e o conjunto de todas as âncoras é denominado Foci.

Em ambos os casos, as distâncias até um mesmo objeto âncora não são organizadas. Existem algumas propostas para reduzir o tempo computacional que esse percurso seqüencial pode levar. Em (Micó et al., 1996), os autores propõem reduzir esse tempo extra de CPU através da criação de uma estrutura no estilo das GHT, utilizando os objetos âncoras. Uma outra idéia útil para este caso é a de ordenar o conjunto das distâncias, relacionado a cada objeto âncora, através de alguma outra estrutura. Essa idéia pode ser observada nos trabalhos de Nene e Nayar (1997), no método *Spaghettis* (Chávez et al., 1999a) e nos métodos *OmniRTree* e *Omni B-Forest* (Santos-Filho, 2003). Nesses casos, a única consideração especial é que a estrutura adicional exige maior quantidade de espaço.

A maior vantagem de manter a matriz de distâncias é que isso permite uma recuperação rápida da informação. Porém, essa técnica impõe um alto custo de construção em termos de tempo e espaço.

A técnica de utilizar objetos âncoras reduz bastante o número de cálculos de distância necessários, porém a escolha apropriada dos objetos âncora pode afetar a qualidade da resposta (Santos-Filho et al., 2001; Bustos et al., 2001).

6.3 A família MA+

Na maioria dos casos, a distribuição das consultas apresenta um comportamento similar ao princípio de Pareto,isto é, quase sempre existirá um grupo, relativamente pequeno dos dados, que são os mais consultados e existirá também uma grande maioria deles que são acessados com pouca freqüência. O dilema está exatamente em como detectar essas distâncias ou regiões do espaço, que são as mais acessadas, sem ter que construir a matriz de distâncias completa.

Nos métodos existentes, as distâncias são apenas utilizadas para verificar se o objeto faz parte ou não do conjunto de resposta e logo após são descartadas. Por isso, se a mesma consulta é repetida, o algoritmo não dispõe dessa distância e deve calculá-la novamente.

Para solucionar esse problema, propõe-se uma nova técnica denominada *Plug-in Module for Access Methods* (PMAM), que tem por objetivo ajudar na aceleração do processo de Recuperação de Informação por Similaridade. O módulo foi projetado para ser utilizado junto a algum Método de Acesso.

Quando um MA estiver usando o módulo *Plug-in Module for Access Methods* (PMAM), adotar-se-á o símbolo + para indicar que aquele MA está utilizando a nova técnica. Por exemplo, *OmniRTree* representa a técnica tradicional e, *OmniRTree+* representa o MA *OmniRTree* utilizando o PMAM.

O módulo PMAM poderia ser implementado através de diversas estruturas de dados. Na seguinte seção, é apresentada uma das possíveis implementações utilizando árvores B^*.

6.4 Implementação do PMAM através de árvores B*

Inicialmente, PMAM é uma estrutura vazia que será "retroalimentada" de forma gradual em função das distâncias geradas pelas consultas. Um dos problemas a serem resolvidos está relacionado à estrutura que o módulo deve ter para armazenar o trio $(O_i, O_j, d(O_i, O_j))$, onde O_i e O_j são dois objetos e $d()$ é a distância entre eles. Uma consideração especial é que, ao invés de considerar os objetos propriamente ditos, pressupõe-se que cada objeto, O_k, tem um identificador único, $ID(O_k)$, associado. Por essa razão, a informação que deve ser armazenada é: $(ID(O_i), ID(O_j), d(O_i, O_j))$.

Também deve ser considerado que o objetivo do PMAM é, dado um par de identificadores de objetos $ID(O_i)$ e $ID(O_j)$, recuperar a distância correspondente para aquele par. Isto é, a chave da consulta deve ser criada apenas em função dos dois identificadores, de acordo com o Algoritmo 6.1. Nesse algoritmo, supõe-se que os identificadores são inteiros de 32 bits e a chave gerada é um inteiro de 64 bits.

Algoritmo 6.1 Geração de uma chave única para dois identificadores de objetos

1: **Função** CalcularChaveUnica($ID1$, $ID2$)
2: **se** ($ID1 > ID2$) **então**
3: trocar os identificadores, $ID1 \longleftrightarrow ID2$;
4: **fim se**
5: retornar $(ID1 \ll 32) \mid ID2$;
{a operação $\ll$ representa o operador de deslocamento de bits à esquerda e o operador $\mid$ representa um *or* lógico entre os bits}
6: **Fim**

Depois de ter gerado a chave, através da função $CalcularChaveUnica(ID1, ID2)$, o problema se reduz a armazenar a distância respectiva junto com a chave. Uma solução poderia ser utilizar uma árvore *B-Tree* (Bayer e McCreight, 1972) ou uma B^* (Knuth, 1973). No caso do PMAM, optou-se pela implementação com árvores B^*.

Apesar das árvores B^* serem eficientes para a recuperação de dados unidimensionais, o tempo de resposta poderia ser melhorado, ainda mais, criando-se uma tabela de *hashing* de árvores B^*. No Algoritmo 6.2, pode-se observar a inicialização dos membros do PMAM. Esse módulo está composto por uma tabela de *Hashing* de tamanho $HashTableSize$.

Algoritmo 6.2 Inicialização do PMAM

Requer: $HashTableSize > 0$
1: **Método** PMAM::Inicializar()
2: **para** i = 0 até HashTableSize **faça**
3: HashTable[i] = NULL;
4: **fim para**
5: **Fim**

No Algoritmo 6.3, observa-se a forma de inserir novas chaves no módulo PMAM. Cada célula dessa tabela é um ponteiro para uma árvore B^*.

Para poder recuperar os dados associados a uma chave inserida na árvore, cada elemento também armazena um número. Esse valor pode ser usado de forma livre pelo usuário para armazenar alguma informação que seja útil para conectar a chave com o objeto que esta referencia. No caso da técnica proposta PMAM, esse valor é usado para armazenar a distância associada com a chave inserida. Além disso, a estrutura da B^* foi modificada para armazenar um contador de uso para cada chave inserida na árvore. Esse contador é atualizado cada vez que a chave é acessada. Essa modificação pode ser útil para decidir, no futuro, quais as chaves que poderiam ser descartadas, devido a uma eventual falta de espaço. No Algoritmo 6.4, pode-se observar a técnica utilizada para recuperar uma distância entre dois objetos identificados por $ID1$ e $ID2$.

O processo de "retroalimentação" do PMAM pode ser realizado durante a execução de qualquer consulta por similaridade que precise de cálculos de distância. Como pode

Algoritmo 6.3 Inserção de informação no módulo PMAM

```
1: Método PMAM::Inserir(ID1, ID2, distancia)
2: UniqueKey = CalcularChaveUnica(ID1, ID2);
3: pos = UniqueKey mod HashTableSize;
4: se HashTable[pos] == NULL então
5:    HashTable[pos] ⟸ nova árvore B*; {Ainda não tinha árvore nessa posição}
6: fim se
7: HashTable[pos] →Inserir(UniqueKey, distancia);
8: Inicializar o contador de uso de UniqueKey com zero;
9: Fim
```

Algoritmo 6.4 Busca de informação no PMAM

```
1: Método PMAM::Busca(ID1, ID2)
2: UniqueKey = CalcularChaveUnica(ID1, ID2)
3: pos = UniqueKey mod HashTableSize;
4: se HashTable[pos] == NULL então
5:    retornar -1; {A distância não existe no PMAM}
6: fim se
7: se UniqueKey estiver na árvore B* apontada por HashTable[pos] então
8:    Adicionar 1 ao contador de uso de UniqueKey;
9:    retornar a distância associada com UniqueKey;
10: senão
11:    retornar -1; {A distância não existe no PMAM}
12: fim se
13: Fim
```

ser observado no Algoritmo 6.5, a única mudança necessária nos MAs existentes está no instante em que uma distância entre dois objetos é calculada. Nesse momento, antes de realizar o cálculo da distância, a mesma é procurada no PMAM. Se a distância já estiver armazenada, não será necessário calculá-la novamente. No caso em que a distância não possa ser recuperada através do PMAM, o MA deverá calculá-la da forma convencional e depois inseri-la no PMAM para eventuais futuras consultas.

Algoritmo 6.5 Utilização do PMAM durante uma consulta por similaridade

```
1: Enquanto uma consulta por similaridade estiver sendo executada
2: para cada d(O_i, O_j) que a consulta precisar faça
3:    se d(O_i, O_j) existe no PMAM então
4:       usá-la diretamente;
5:    senão
6:       calcular d(O_i, O_j);
7:       inserir d(O_i, O_j) no PMAM;
8:    fim se
9: fim para
```

O tempo extra que o PMAM acrescenta, no processo normal de recuperação de informação de um MA, está determinado pelo tempo de inserção e recuperação de dados em uma árvore B^*.

O PMAM permite recuperar as distâncias com um custo computacional de $log_m(s)$, onde s representa o número de distâncias armazenadas e m representa a ordem da árvore B^*. Devido ao fato de que, na maioria dos casos, as consultas obedecem ao princípio de Pareto, pode-se afirmar que:

$$s = \alpha \frac{n(n-1)}{2}, \tag{6.1}$$

onde n é o número de objetos existentes, α é a taxa de ocupação da matriz ($0 \leq \alpha \leq 1$) e $\frac{n(n-1)}{2}$ é o número de distâncias possíveis em uma matriz triangular completa. Se as consultas estiverem distribuídas de acordo com o princípio de Pareto, espera-se uma taxa de ocupação relativamente baixa.

No caso de usar uma tabela de $Hash$ de tamanho $HashTableSize$, como foi feito no PMAM, é possível reduzir ainda mais o tempo de recuperação de uma distância para, aproximadamente, $\frac{s}{HashTableSize}$.

Apesar de observar-se que o custo computacional apresentado na Equação (6.1) é de ordem quadrática em função do número de objetos (n), na realidade os objetos inseridos na estrutura estão apenas conectados a um número finito de vizinhos mais próximos. Portanto, a Equação (6.1) pode ser também escrita da seguinte maneira:

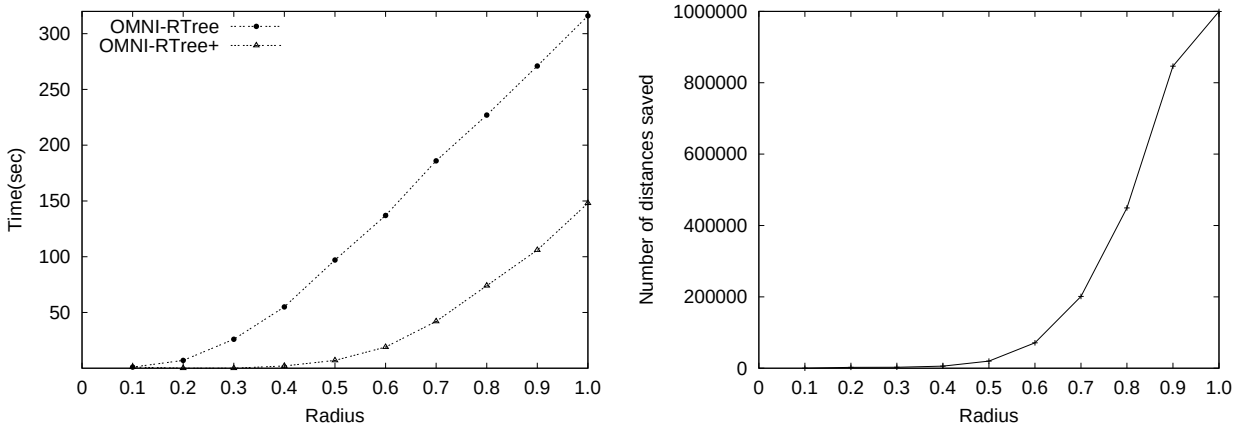
$$s = \frac{kn}{2}, \tag{6.2}$$

onde k representa a média do número de vizinhos com os quais um objeto está conectado na estrutura. Para grandes bancos de dados, espera-se um valor de k muito menor que n, isto é: $k \ll n$. De acordo com esse cálculo, o custo computacional de recuperação de uma distância, através do PMAM, é $O(log_m(kn))$.

6.5 Experimentos

Os experimentos apresentados nesta seção foram realizados com os mesmos conjuntos de dados apresentados na Seção 5.3.1 (ver Pág. 78). Na Figura 6.1(a), pode ser observado o desempenho, em termos de tempo, da *OmniRTree* tradicional e da *OmniRTree+* . Nesses experimentos, foram realizadas 500 consultas por abrangência (*Range Queries*) com centros escolhidos de forma aleatória para ambos os métodos. Nesse gráfico, também observa-se que, ao longo do tempo, o PMAM permite que a diferença entre ambas as técnicas seja mais acentuada.

A diferença de tempo, a favor do *OmniRTree+* , pode ser melhor entendida através da linha apresentada na Figura 6.1(b). Essa linha representa o número de cálculos de distância que foram recuperadas do PMAM e que, conseqüentemente, não precisaram ser calculadas mais de uma vez. Nesse experimento, todas as distâncias geradas foram inseridas no PMAM, porém diversos critérios poderiam ser utilizados para filtrar as distâncias mais relevantes. Uma heurística poderia ser armazenar apenas as distâncias dos objetos mais próximos, pois esses objetos têm maior probabilidade de serem úteis para consultas por similaridade.



(a) Desempenho de consultas do tipo *Range Query*

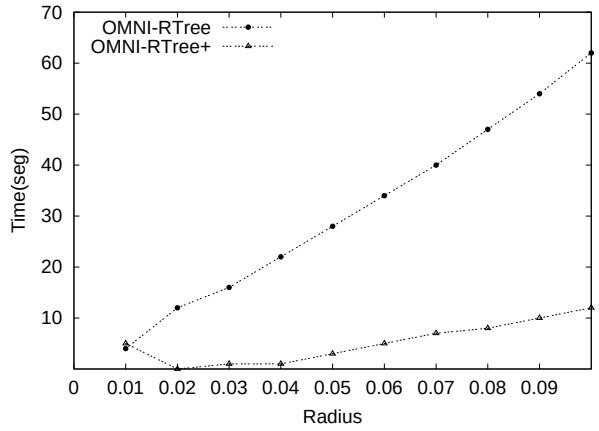
(b) Distâncias recuperadas através do PMAM

Figura 6.1: Efeito do uso do PMAM na *OmniRTree*, com 5 foci, e o conjunto IMAGES.

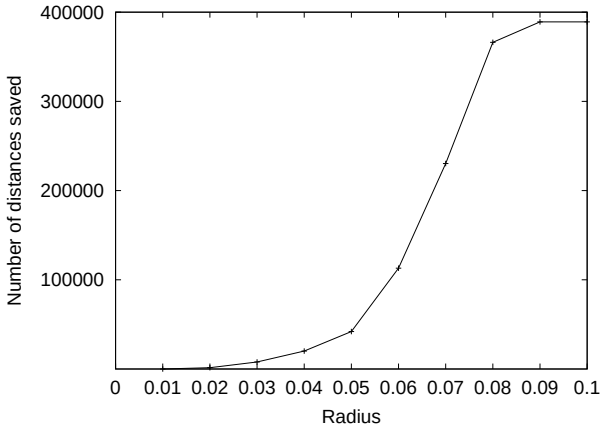
Na Figura 6.2(a), pode-se observar o resultado de aplicar outras 500 consultas com raios aleatórios com o conjunto de dados FOREST. Nesse caso, para o primeiro raio apresentado, a *OmniRTree* teve um melhor desempenho que a *OmniRTree+* . Isto pode acontecer porque, nesse instante, o PMAM está sendo "alimentado" pelas novas distâncias e isso consome um tempo extra. Porém, o tempo investido no armazenamento de distâncias ajuda na redução de cálculos de distância no futuro, como pode ser observado nos pontos dos outros raios. A diferença de tempo entre ambas as abordagens também pode ser explicada pela Figura 6.2(b), que representa o número de cálculos de distância que foram evitadas graças ao uso do PMAM.

Como já foi explicado na Equação (1.2) (ver Pág. 2), no caso de dados complexos, o fator que consome mais tempo está representado pelos cálculos de distância. A diferença considerável que pode ser observada nos gráficos das Figuras 6.1(a) e 6.2(a) deve-se principalmente ao tempo consumido pela função de distância, que é maior que o tempo de recuperação e inserção da distância através do PMAM. Essa diferença pode ser ainda maior à medida que a função de distância seja mais complexa, em termos de tempo.

No caso de baixas dimensões, onde o cálculo de distância precisa de menos tempo, o tempo de consulta e inserção no PMAM pode ser maior que o tempo ganho. Na Figura 6.3, observam-se os resultados dos experimentos utilizando o conjunto de dados LETTERS



(a) Desempenho de consultas do tipo *Range Query*



(b) Distâncias recuperadas através do PMAM

Figura 6.2: Efeito do uso do PMAM em *OmniRTree*, com 3-foci, no conjunto FOREST (54-D).

(16-D). Nesse último caso, a *OmniRTree* tradicional teve um melhor desempenho devido ao fato de que a função de distância e relativamente simples e por esse motivo consumiu menos tempo que a inserção das distâncias no PMAM.

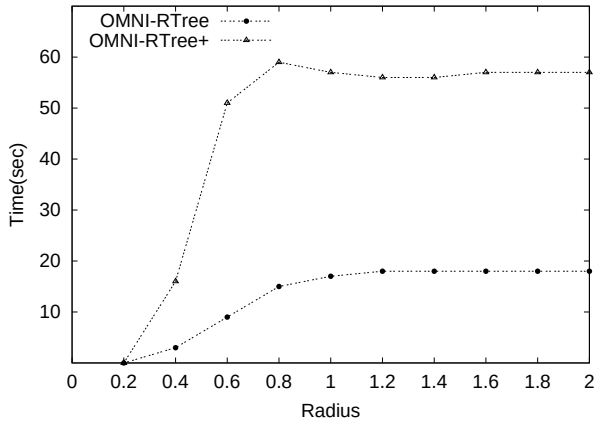


Figura 6.3: Efeito do uso do PMAM com uma função de distância com baixo custo computacional utilizando *OmniRTree*, com 7-foci e o conjunto de dados LETTERS (16-D).

A Figura 6.3 é relevante apenas para ser comparada com as Figuras 6.1 e 6.2. Em problemas reais, o PMAM deve ser utilizado apenas com funções de distância que consomem uma quantidade considerável de tempo. Isto é, funções cujo cálculo leve mais tempo que a inserção na árvore B^*.

6.6 Propriedades da técnica proposta

O módulo PMAM apresenta as seguintes propriedades:

Generalizável para qualquer MAM: o PMAM pode ser perfeitamente usado junto a qualquer MAM conhecido, pois a comunicação entre o MAM e o PMAM está bem estabelecida através da função de cálculo de distância;

Aplicáveis também para MAE: o PMAM pode ser usado também junto aos MAE. A única adaptação que deve ser considerada é que o PMAM pode ser utilizado apenas em conjuntos de objetos nos quais esteja estabelecida uma função de distância;

Dinamismo: uma das principais contribuições do uso de PMAM é que, diferentemente dos métodos AESA e LAESA, o PMAM não precisa conhecer, *a priori*, o número de objetos. Qualquer distância pode ser inserida no PMAM a qualquer momento;

Conexionismo: cada distância armazenada no PMAM pode também ser vista como uma conexão entre dois objetos. Um MAM ou MAE tradicional não cria conexões com os vizinhos mais próximos como no caso de um MAM+ ou SAM+. No pior caso desta proposta, o PMAM ainda está vazio e corresponde ao caso MAM ou MAE original;

Tempo: o PMAM é baseado apenas no conhecimento gerado pelas consultas e pode ser construído *on-line*. O único tempo extra de CPU desta proposta está relacionado ao tempo envolvido na inserção e busca de distâncias no PMAM. No caso de funções de distância complexas, o que se espera é que consumam um tempo maior que o tempo de inserção e consulta no PMAM;

Simplicidade: a incorporação do PMAM em um MAM ou MAE apenas precisa criar um ponto de conexão. Esse ponto está na função de distância métrica (ver Algoritmo 6.5);

Baseado no comportamento do usuário: a grande maioria de MAs são construídos sem considerar a distribuição das consultas, pois estas ainda não são conhecidas no instante da construção. No caso do PMAM, a construção é realizada de acordo com o comportamento do usuário.

Além das propriedades citadas anteriormente, vale a pena ressaltar que a entrada do PMAM é composta apenas por dados unidimensionais. Por causa disso, poderia ser utilizada qualquer estrutura que permita recuperar os dados de forma rápida, como por exemplo, as árvores B^*.

Um outro ponto importante que deve ser ressaltado é que, no caso de consultas por similaridade, como por exemplo um *Range Query*(O_q, r), existe uma grande probabilidade de que uma consulta não seja repetida de forma exata ou que o objeto O_q não faça parte dos objetos armazenados. Nesse caso, a estratégia poderia ser encontrar o objeto O_1 mais próximo em relação ao centro da consulta, e utilizar as distâncias existentes para aquele objeto e a desigualdade triangular para realizar a consulta.

6.7 Considerações Finais

Neste capítulo foi apresentado o módulo PMAM que pode ser utilizado junto a qualquer MAM ou MAE existente. No caso dos MAE, a única restrição é que deve ser estabelecida uma função de distância.

A principal contribuição introduzida pelo uso do módulo proposto é o fato deste ser baseado no comportamento do usuário e não apenas nos dados.

O PMAM foi criado pela constatação de que os MAs existentes se comportam de uma forma mecânica e não reaproveitam o conhecimento gerado pelas consultas. Uma solução existente é armazenar todas as distâncias possíveis, porém essa solução é inviável devido ao alto custo computacional que envolve. Para reduzir o custo computacional, o PMAM armazena apenas aquelas distâncias produzidas pelas consultas que o usuário executa sem precisar de um processo extra. O PMAM, comparado com a abordagem anterior, permite reduzir consideravelmente a quantidade de distâncias necessárias. O PMAM é construído com base nas consultas do usuário considerando que, se uma distância foi necessária em um tempo t, a probabilidade de ser necessária no futuro aumenta.

Uma vantagem importante do PMAM é que as adaptações necessárias em um MA são mínimas. Apenas é necessário estabelecer uma comunicação entre o MA e o PMAM no ponto em que a função de distância é calculada.

Além disso, o MA+ incorpora o conceito de conexionismo entre a vizinhança dos objetos indexados. O resultado dessa combinação é que existirá uma maior quantidade de distâncias naquelas áreas onde o usuário realize mais consultas. Um MAM ou MAE tradicional, pode ser visto como o pior caso de um MAM+ ou SAM+, pois representam o caso onde não se dispõe de informação relacionada às distâncias calculadas anteriormente.

Os MAM+ e SAM+ representam o ponto de equilíbrio entre espaço, simplicidade e tempo de construção. Essa técnica é especialmente útil para aqueles casos onde a função de distância consume um tempo considerável, que é o que acontece no caso de bases de dados multimídia.

Capítulo 7

Conclusões e Trabalhos Futuros

A indexação e recuperação de informação por similaridade em dados multimídia é um problema complexo. Nesta tese foram propostas novas técnicas para melhorar o desempenho tanto de Redes Neurais Artificiais quanto dos Métodos de Acesso. Mesmo utilizando as técnicas dos Métodos de Acesso e modelos de Redes Neurais Artificiais mais avançadas ainda existem problemas em aberto.

No Capítulo 5, foram propostas as técnicas SAM-SOM e MAM-SOM que permitiram a incorporação de MAE e MAM em SOM. O objetivo principal desta incorporação é poder encontrar a unidade vencedora com um custo computacional de aproximadamente $log(n)$, onde n é o número de unidades existentes na rede.

A técnica SAM-SOM apresenta duas variantes que foram denominadas SAM-SOM híbrida e SAM-SOM$*$. No caso da técnica SAM-SOM híbrida, os MAE são incorporados ao processo de treinamento de um SOM tradicional. A única diferença é que o processo seqüencial para encontrar a unidade vencedora é substituído pela utilização de um MAE. A principal vantagem dessa técnica é que o tempo de treinamento é reduzido de forma considerável. Essa diferença pode ser ainda mais acentuada à medida que o número de unidades aumenta. Nesse caso, o processo de correção de pesos continua sendo realizado de acordo com o algoritmo de treinamento do SOM envolvido.

No segundo caso, SAM-SOM$*$, a principal diferença é que uma nova unidade é inserida na rede para cada objeto do conjunto de treinamento apresentado. Esse caso representa o limite de exigência possível para uma rede do tipo SOM construtivo tradicional, pois o tempo de adaptação das novas unidades é reduzido ao mínimo, abrindo a possibilidade de criar redes deformadas. Apesar dessas condições, aparentemente adversas, o desempenho melhora consideravelmente, graças ao uso de MAE. Isto acontece mesmo considerando um número muito alto de unidades, conforme comprovado nos experimentos realizados.

No caso da técnica MAM-SOM, foi discutida apenas a técnica MAM-SOM$*$, que permite a criação de uma nova unidade na rede para cada objeto apresentado. Neste caso, não

foi analisada a técnica híbrida pois seria necessário estabelecer um processo de correção de vetores de pesos. Em espaços métricos, isso nem sempre é possível devido ao fato de que, nesse contexto, não existe o conceito de coordenadas para cada objeto. Porém, se for estabelecida alguma técnica que permita realizar um processo similar à correção de pesos para espaços métricos, também poder-se-ia falar da técnica MAM-SOM híbrida. Uma técnica, recentemente apresentada para atingir esse objetivo pode ser observada no trabalho de Somervuo (2003). Nesse trabalho, o autor demonstra como pode ser calculado um ponto intermediário entre duas palavras de um dicionário.

Considerando que:

- a diferença obtida, em termos de desempenho, através das técnicas SAM-SOM e MAM-SOM é muito considerável (ver Seção 5.3 Pág. 77);
- o custo computacional do treinamento de um SOM é alto (ver Seção 5.1 Pág. 69);
- o processo para encontrar a unidade vencedora é utilizado com muita freqüência;
- em bancos de dados multimídia, a dimensão dos dados pode ser alta;
- a função de distância entre dois objetos complexos pode ser muito complexa e demorada, especialmente quando se trata de dados multimídia;
- existe uma tendência marcada ao aumento do volume e complexidade da função de cálculo de distância,

pode-se concluir que a utilização de MAE ou MAM, nesse processo em particular, é altamente recomendável para reduzir custos em termos de cálculos computacionais e, por conseqüência, reduzir o tempo de treinamento de um SOM.

Uma outra conclusão importante é que, depois de conhecer a forma de utilizar MAE e MAM em redes do tipo SOM e as vantagens de seu uso, não existe razão para continuar utilizando a comparação seqüencial para encontrar a unidade vencedora.

As vantagens da utilização de MAE e MAM no treinamento de SOM podem ser ainda mais acentuadas à medida em que o volume de padrões e o tamanho da rede aumentam.

Uma conseqüência direta da redução do tempo necessário para o treinamento é que será possível pensar em redes com um maior número de unidades possibilitando um maior aprendizado por parte das mesmas.

Por outro lado, após terem sido analisadas as principais vantagens e limitações dos MAE e MAM, descobriu-se que esses métodos não possuem capacidade de aprendizado. Isso significa que, depois de ter realizado uma consulta, os MAs descartam o conhecimento gerado (representado pelas distâncias). Por causa disso, no Capítulo 6, foi proposto o módulo

PMAM, que permite armazenar as distâncias geradas pelas consultas com o objetivo de reaproveitá-las para outras consultas.

Esta técnica permite dispor de uma "pseudo-matriz de distâncias", de forma análoga à técnica AESA, com a grande vantagem que o PMAM precisa de uma fração de espaço quando comparada com AESA. O módulo PMAM contém apenas aquelas distâncias que são as mais usadas pelas consultas. Esta técnica representa o ponto de equilíbrio entre a velocidade de recuperação de distâncias, $O(1)$, da técnica AESA (Vidal, 1986) e o espaço requerido pela nova estrutura. O módulo PMAM permite recuperar a distância com um custo computacional de $O(log_m(kn))$, como apresentado na Seção 6.3 (ver Pág. 92).

7.1 Principais Contribuições

Os resultados desta tese representam contribuições em duas áreas de pesquisa: Redes Neurais Auto-Organizáveis e Métodos de Acesso Espacial e/ou Métricos. As principais contribuições são apresentadas a seguir:

- A prova de que é possível criar redes do tipo SOM com um grande número de unidades, inclusive igual ao número de objetos do conjunto de treinamento, com um tempo de treinamento ainda menor que o apresentado pelos modelos antecessores;
- Antes da apresentação das técnicas SAM-SOM e MAM-SOM, as conexões de um SOM eram úteis apenas para acessar os vizinhos conectados com uma unidade (informação topológica). Com essas técnicas, as conexões também podem conter a distância, que é uma informação muito útil para o processo de recuperação de informação por similaridade;
- A apresentação de uma técnica que permite a redução do tempo de treinamento de uma rede SOM;
- A criação de um modelo, derivado dos mapas de Kohonen, que resolve o dilema da estabilidade-plasticidade. Esse modelo permite adicionar novos padrões a qualquer instante sem perder o conhecimento adquirido;
- A criação de um modelo de SOM com capacidade para poder atender consultas do tipo k-vizinhos mais próximos e buscas abrangência (*Range Query*);
- A incorporação de conexionismo, em métodos de acesso, entre os objetos vizinhos, baseado na distribuição das consultas realizadas pelo usuário;
- A apresentação de uma técnica para o armazenamento de distâncias que permite que um MA+ possa criar uma maior quantidade de conexões nas regiões nas quais exista

maior concentração de consultas. O PMAM tem a vantagem de ser construído em função do comportamento do usuário e não apenas em função do conjunto de dados;

- A utilização do PMAM junto a um Método de Acesso pode ser realizada, em termos de implementação, com um impacto mínimo, pois para retroalimentá-lo, apenas é necessário modificar uma linha de código, no ponto onde a distância entre dois objetos é calculada.

Esta tese de doutorado fez parte do Projeto IMiMD - Indexação e Mineração de Dados Multimídia (*Indexing and Data Mining in Multimedia Data Bases*) (Traina Jr e Faloutsos, 2000), que vem sendo desenvolvido de forma conjunta entre o ICMC-USP e *Carnegie Mellon University*-EUA.

Os resultados obtidos nesta tese geraram publicações através de Relatórios Técnicos (Cuadros-Vargas e Romero, 2000), artigos em Conferências Internacionais (Cuadros-Vargas e Romero, 2002; Cuadros-Vargas et al., 2003b) e em Revistas da área (Faloutsos et al., 2003; Cuadros-Vargas et al., 2003a, 2004; Cuadros-Vargas e Romero, 2004).

7.2 Propostas de Trabalhos Futuros

Em relação às Redes Neurais Artificiais, uma extensão natural deste trabalho é a generalização das técnicas SAM-SOM e MAM-SOM para outros modelos de redes. Os MAE e MAM podem também ser utilizados junto a redes com múltiplas camadas, como nos experimentos apresentados por Vicentini (2002). Essa abordagem permite que cada saída da rede possa organizar os dados classificados por ela com o intuito de realizar consultas por similaridade.

Nos experimentos apresentados nesta tese foram utilizados, propositalmente, métodos de acesso, tais como como *R-Tree* (Guttman, 1984), *k-d-Tree* (Finkel e Bentley, 1974) que são relativamente antigos. Mesmo considerando a antiguidade desses métodos, os resultados foram positivos. Porém, há necessidade de realizar experimentos com estruturas mais avançadas.

Um aspecto importante que pode ser melhorado no PMAM, refere-se à política de descarte das distâncias menos úteis, com o intuito de minimizar o espaço necessário para armazená-las e, conseqüentemente, reduzir o tempo de recuperação das distâncias. Uma idéia em aberto é manter aquelas distâncias que são as mais acessadas. Para atingir esse objetivo, poderia ser adicionado um contador a cada distância armazenada no PMAM. Esse contador seria atualizado cada vez que a distância fosse recuperada. Na Figura 7.1, observa-se a possível estrutura lógica da informação armazenada pelo PMAM proposto. A altura de cada coluna representa o contador de uso de cada distância. Nesta estrutura, poder-se-ia aplicar um plano de corte horizontal, de forma similar ao algoritmo *Watershed* (Beucher e

Lantuéjoul, 1979), para manter aquelas distâncias com maior freqüência de uso. O problema é minimizar o custo computacional desse processo.

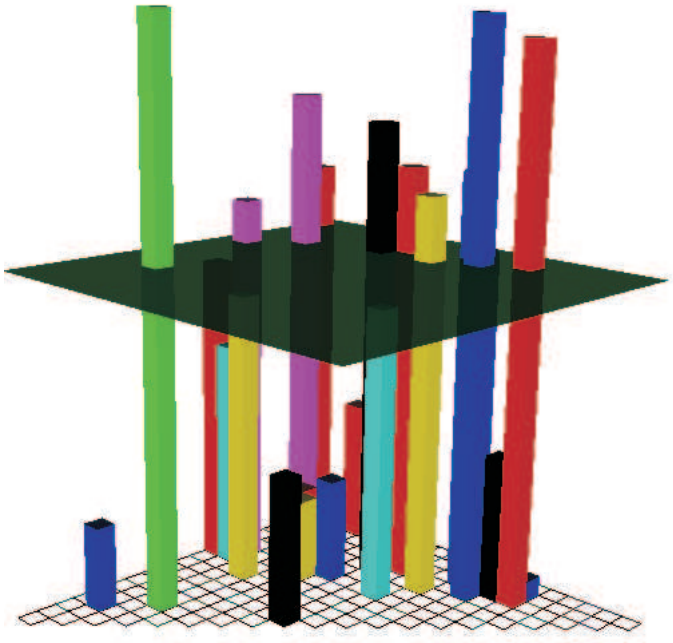


Figura 7.1: Representação das distâncias armazenadas no PMAM em forma de uma matriz triangular. A altura de cada barra representa a freqüência de acesso para uma distância.

Uma outra heurística que poderia ser aplicada para reduzir o número de distâncias do PMAM é eliminar as maiores distâncias, pois representam conexões entre objetos distantes. Neste caso, o intuito é manter as distâncias dos objetos mais próximos.

A terceira heurística para a redução das distâncias armazenadas denominada *Least Recently Used* (LRU) (Folk et al., 1998; Webster, 1980) é descartar aquelas menos usadas recentemente.

Outro problema que continua em aberto é o estudo de detecção de agrupamentos a partir da rede gerada com as técnicas SAM-SOM ou MAM-SOM.

Também pode ser estudada a possibilidade de aplicar a generalização da desigualdade triangular com base nas distâncias disponíveis da mesma forma que a técnica proposta por Shasha e Wang (1990).

Finalmente, um outro trabalho futuro refere-se à incorporação de mecanismos de aprendizado incremental em MAE ou MAM. Nesta tese, o PMAM pode ser visto como uma espécie de memória *Cache*. Mesmo assim, os resultados foram muito positivos em relação aos MAM que não o usaram. No futuro, acredita-se na possibilidade de criar métodos de acesso com maior eficiência, capazes de aprender e generalizar o comportamento do usuário para acelerar consultas à medida que mais conhecimento seja gerado.

Referências Bibliográficas

[Alahakoon et al.(1998a)] Alahakoon, D.; Halgamuge, S.; Srinivasan, B. A Structure Adapting Feature Map for Optimal Cluster Representation. In: *International Conference on Neural Information Processing*, Kitakyushu, Japan, 1998a, p. 809–812.

[Alahakoon et al.(1998b)] Alahakoon, D.; Halgamuge, S.; Srinivasan, B. A self growing cluster development approach to data mining. In: *IEEE International Conference on Systems, Man and Cybernetics*, 1998b, p. 2901–2906.

[Alahakoon et al.(2000)] Alahakoon, D.; Halgamuge, S.; Srinivasan, B. Dynamic self-organizing map with controlled growth for knowledge discovery. *IEEE Transactions on Neural Networks*, v. 11, n. 3, p. 601–614, 2000.

[Albers e Westbrook(1996)] Albers, S.; Westbrook, J. Self-organizing data structures. In: *Online Algorithms*, 1996, p. 13–51.
Disponível em `http://citeseer.nj.nec.com/albers98selforganizing.html` (Acessado em 08/01/2004)

[Annamalai et al.(2000)] Annamalai, M.; Chopra, R.; Defazio, S. Indexing Images in Oracle8i. In: *ACM SIGMOD international Conference on the Management of Data*, Dallas, 2000, p. 539–547.

[Aurenhammer(1991)] Aurenhammer, F. Voronoi diagrams – a survey of a fundamental geometric data structure. *ACM Computing Survey*, v. 23, n. 3, 1991.

[Baeza-Yates(1997)] Baeza-Yates, R. Searching: an algorithmic tour. In: Kent, A.; Williams, J., eds. *Encyclopedia of Computer Science and Technology*, Marcel Dekker Inc., 1997, p. 331–359.

[Baeza-Yates e Navarro(1998)] Baeza-Yates, R.; Navarro, G. Fast approximate string matching in a dictionary. In: *South American Symposium on String Processing and Information Retrieval*, IEEE CS Press, 1998, p. 14–22.

[Baeza-Yates(1990)] Baeza-Yates, R. A. An adaptive overflow technique for b-trees. In: *Extending Database Technology*, 1990, p. 16–28.
Disponível em `http://citeseer.nj.nec.com/99409.html` (Acessado em 08/01/2004)

[Baeza-Yates et al.(1994)] Baeza-Yates, R. A.; Cunto, W.; Manber, U.; Wu, S. Proximity matching using fixed-queries trees. In: Crochemore, M.; Gusfield, D., eds. *Annual Symposium on Combinatorial Pattern Matching*, Asilomar, CA: Springer-Verlag, Berlin, 1994, p. 198–212.
Disponível em `http://citeseer.nj.nec.com/219706.html` (Acessado em 08/01/2004)

[Baeza-Yates e Ribeiro-Neto(1999)] Baeza-Yates, R. A.; Ribeiro-Neto, B. A. *Modern information retrieval*. ACM Press / Addison-Wesley, 1999.
Disponível em `http://citeseer.nj.nec.com/433337.html` (Acessado em 08/01/2004)

[Baraldi e Blonda(1998)] Baraldi, A.; Blonda, P. *A survey of fuzzy clustering algorithms for pattern recognition*. TR-98 038, International Computer Science Institute, 1947 Center St. Suite 600, Berkeley, California 94704-1198, 1998.

[Baraldi e Parmiggiani(1998)] Baraldi, A.; Parmiggiani, F. A fuzzy neural network model capable of generating/removing neurons and synaptic links dynamically. In: P. Blonda, M. C.; Petrosino, A., eds. *II Italian Workshop on Fuzzy Logic*, Singapure: World Scientific, 1998, p. 247–259.

[Bayer e McCreight(1972)] Bayer, R.; McCreight, E. Organization and maintenance of large ordered indexes. *Acta Informatica*, v. 1, n. 3, p. 173–189, 1972.

[Beckmann et al.(1990)] Beckmann, N.; Kriegel, H.; Schneider, R.; Seeger, B. The R*-Tree: An efficient and robust access method for points and rectangles. In: *ACM SIGMOD International Conference on Management of Data*, 1990, p. 322–331.

[Belussi e Faloutsos(1995)] Belussi, A.; Faloutsos, C. Estimating the Selectivity of Spatial Queries Using the 'Correlation' Fractal Dimension. In: *International Conference on Very Large Data Bases, 21st*, Zurich, 1995, p. 299–310.

[Belussi e Faloutsos(1998)] Belussi, A.; Faloutsos, C. Self-Spacial Join Selectivity Estimation Using Fractal Concepts. *ACM Transactions on Information Systems*, v. 16, n. 2, p. 161–201, 1998.

[Bentley(1975)] Bentley, J. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, v. 18, n. 9, p. 509–517, 1975.

[Berchtold et al.(1996)] Berchtold, S.; Keim, D.; Kriegel, H. The X-Tree: An index structure for high-dimensional data. In: *International Conference on Very Large Data Bases, 22nd*, Mumbai (Bombay), 1996, p. 28–39.

[Beucher e Lantuéjoul(1979)] Beucher, S.; Lantuéjoul, C. Use of Watersheds in Contour Detection. In: *Int'l Workshop Image Processing, Real-Time Edge and Motion Detection / Estimation*, Rennes, France, 1979.

[Böhm et al.(2001)] Böhm, C.; Berchtold, S.; Keim, D. Searching in High-Dimensional Spaces Index Structures for Improving the Performance of Multimedia Databases. *ACM Computing Surveys*, v. 33, n. 3, p. 322–373, 2001.

[Bishop et al.(1998)] Bishop, C. M.; Svensen, M.; Williams, C. K. I. GTM: The Generative Topographic Mapping. *Neural Computation*, v. 10, n. 1, p. 215–234, 1998. Disponível em `http://citeseer.nj.nec.com/bishop98gtm.html` (Acessado em 08/01/2004)

[Blackmore e Miikkulainen(1993)] Blackmore, J.; Miikkulainen, R. Incremental Grid Growing: Encoding High-Dimensional Structure into a Two-Dimensional Feature Map. In: *International Conference on Neural Networks*, San Francisco-CA: IEEE Service Center, 1993, p. 450–455.
Disponível em `http://citeseer.nj.nec.com/blackmore93incremental.html` (Acessado em 08/01/2004)

[Blackmore e Miikkulainen(1995)] Blackmore, J.; Miikkulainen, R. Visualizing High-Dimensional Structure with Incremental Grid Growing Neural Network. In: Prieditis, A.; Russell, S., eds. *International Conference on Machine Learning, 12th*, San Francisco, CA, USA: Morgan Kaufmann Publishers, 1995, p. 55–63.
Disponível em `http://citeseer.nj.nec.com/blackmore95visualizing.html`

[Blackmore(1995)] Blackmore, J. M. *Visualizing High-Dimensional Structure with Incremental Grid Growing Neural Network.* Dissertação de Mestrado, University of Texas at Austin, Austin, 1995.

[Blott e Weber(1997)] Blott, S.; Weber, R. *A simple vector-approximation file for similarity search in high-dimensional vector spaces.* Relatório Técnico, Institute for Informatics Systems - ETH Zentrum, Zurich, Switzerland, 1997.

[Boudjemaï et al.(2003)] Boudjemaï, F.; Enberg, P. B.; Postaire, J.-G. Self organizing spherical map architecture for 3d object modeling. In: *Workshop on Self Organizing-Maps*, Hibikino-Kitakyushu, Japan, 2003, p. 197–202.

[Bouton et al.(1991)] Bouton, C.; Cottrell, M.; Fort, J. C.; Pagès, G. Self-organization and convergence of the kohonen algorithm. In: Bouleau, N.; Talay, D., eds. *Probabilités Numériques*, Paris, France: INRIA, 1991, p. 163–180.

[Bozkaya e Ozsoyoglu(1997)] Bozkaya, T.; Ozsoyoglu, M. Distance-based indexing for high-dimensional metric spaces. In: *ACM SIGMOD International Conference on Management of Data*, sigmod Record 26(2), 1997, p. 357–368.

[Bozkaya e Ozsoyoglu(1999)] Bozkaya, T.; Ozsoyoglu, M. Indexing large metric spaces for similarity search queries. *ACM TRANSACTIONS ON DATABASE SYSTEMS*, v. 24, n. 3, p. 361–404, 1999.

[Brin(1995)] Brin, S. Near Neighbor Search in Large Metric Spaces. In: *International Conference on Very Large Databases*, 1995, p. 574–584.
Disponível em `http://citeseer.nj.nec.com/brin95near.html` (Acessado em 08/01/2004)

[Bruske e Sommer(1995)] Bruske, J.; Sommer, G. Dynamic cell structure learns perfectly topology preserving map. *Neural Computation*, v. 7, n. 4, p. 845–865, 1995.

[Bugnion et al.(1993)] Bugnion, E.; Fhei, S.; Roos, T.; Widmayer, P.; Widmer, F. A spatial index for approximate multiple string matching. In: Baeza-Yates, R.; Ziviani, N., eds. *South American Workshop on String Processing (WSP'93)*, 1993, p. 43–53.

[Burkhard e Keller(1973)] Burkhard, W.; Keller, R. Some approaches to best-match file searching. *Communications of the ACM*, v. 16, n. 4, p. 230–236, 1973.

[Burzevski e Mohan(1996)] Burzevski, V.; Mohan, C. K. Hierarchical growing cell structures. *Neural Networks, IEEE International Conference on*, v. 3, p. 1658–1663, 1996.

[Bustos et al.(2001)] Bustos, B.; Navarro, G.; Chávez, E. Pivot Selection Techniques for Proximity Searching in Metric Spaces. In: *Conference of the Chilean Computer Science Society (SCCC'01), XXI*, IEEE CS Press, 2001, p. 33–40.
Disponível em `http://www.dcc.uchile.cl/ gnavarro/ps/sccc01.1.ps.gz` (Acessado em 08/01/2004)

[Caldwell(1995)] Caldwell, C. Graph Theory Tutorial. Última visita em 22 de Dezembro de 2004, 1995.
Disponível em `http://www.utm.edu/departments/math/ graph/index.html` (Acessado em 08/01/2004)

[Carpenter e Grossberg(1990)] Carpenter, A.; Grossberg, S. Art 3 hierarchical search: Chemical transmitter in self-organizing pattern recognition architectures. *Neural Networks*, v. 3, p. 129–152, 1990.

[Carpenter et al.(1992)] Carpenter, G.; Grossberg, S.; Markuzon, N.; Reynolds, J. H.; Rosen, D. B. Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks*, v. 3, n. 5, p. 698–713, 1992.

[Carpenter et al.(1991a)] Carpenter, G.; Grossberg, S.; Neurophysiol, C.; Reynolds, S. ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. *Neural Networks*, v. 4, p. 565–588, 1991a.

[Carpenter et al.(1991b)] Carpenter, G.; Grossberg, S.; Rosen, D. Fuzzy ART: Fast Stable Learning and Categorization of Analog Patterns by an Adaptive Resonance System. *Neural Networks*, v. 4, p. 759–771, 1991b.

[Carpenter e Grossberg(1986)] Carpenter, G. A.; Grossberg, S. Absolutely stable learning of recognition codes by a self-organising neural network. *AIP Conf. Proc. 151: Neural networks for computing*, p. 77–85, new York: American Institute of Physics, 1986.

[Carpenter e Grossberg(1987a)] Carpenter, G. A.; Grossberg, S. ART 2 : self-organization of stable category recognition codes for analog input patterns. *Applied Optics*, v. 26(23), p. 4919–4930, 1987a.

[Carpenter e Grossberg(1987b)] Carpenter, G. A.; Grossberg, S. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing*, v. 37, n. 1, p. 54–115, 1987b.

[Carpenter et al.(1991c)] Carpenter, G. A.; Grossberg, S.; Rosen, D. B. ART 2-A: An adaptive resonance algorithm for rapid category learning and recognition. *Neural Networks*, v. 4, p. 493–504, 1991c.

[Chakrabarti e Mehrotra(1999)] Chakrabarti, K.; Mehrotra, S. The hybrid tree: an index structure for high dimensional feature spaces. *Proceedings, IEEE International Conference on Data Engineering, 15*, p. 440–447, sidney, 1999.

[Chan et al.(1995)] Chan, L.-W.; Chau, M.-W.; Chung, W.-C. Globalor: a parallel implementation of the self-organizing map. In: Zhong, Y.; Yang, Y.; Wang, M., eds. *International Conference on Neural Information Processing*, Dept. of Computer Science, Chinese Univ. of Hong Kong, Shatin, Hong Kong, Beijing, China: Publishing House of Electron. Ind., 1995, p. 625–628.

[Chazelle(1994)] Chazelle, B. Computational geometry: a retrospective. In: *ACM Symposium on the Theory of Computing (STOC'94), 26th*, 1994, p. 75–94.

[Chiueh(1994)] Chiueh, T. Content-based image indexing. In: *International Conference on Very Large Databases*, Santiago, Chile, 1994, p. 582–593.

[Chávez et al.(1999a)] Chávez, E.; Marroquín, J.; Baeza-Yates, R. Spaghettis: an array based algorithm for similarity queries in metric spaces. In: *South American Symposium on String Processing and Information Retrieval*, IEEE CS Press, `ftp://garota.fismat.umich.mx/pub/user/elchavez/fqa.ps.gz`, 1999a, p. 38–46.

[Chávez et al.(1999b)] Chávez, E.; Marroquín, J.; Navarro, G. Overcoming the curse of dimensionality. In: *European Workshop on Content-Based Multimedia Indexing (CBMI'99)*, `ftp://garota.fismat.umich.mx/pub/user/elchavez/fqa.ps.gz`, 1999b, p. 57–64.

[Chávez et al.(2001)] Chávez, E.; Navarro, G.; Baeza-Yates, R.; Marroquín, J. Proximity searching in metric spaces. *ACM Computing Surveys*, v. 33, n. 3, p. 273–321, 2001.

[Ciaccia et al.(1999)] Ciaccia, P.; Nanni, A.; Patella, M. A Query-sensitive Cost Model for Similarity Queries with M-Tree. In: Roddick, J., ed. *Australasian Database Conference (ADC'99), 10th*, Auckland, New Zealand: Springer Verlag, 1999, p. 65–76.

[Ciaccia e Patella(1998)] Ciaccia, P.; Patella, M. Bulk loading the M-tree. In: McDonald, C., ed. *Australasian Database Conference (ADC'98), 9th*, Perth, Australia: Springer Verlag, 1998, p. 15–26.

[Ciaccia e Patella(2002)] Ciaccia, P.; Patella, M. Searching in metric spaces with user-defined and approximate distances. *ACM Transactions on Database Systems (TODS)*, v. 27, n. 4, p. 398–437, 2002.

[Ciaccia et al.(1997a)] Ciaccia, P.; Patella, M.; Rabitti, F.; Zezula, P. Indexing metric spaces with M-tree. In: Cristiani, M.; Tanca, L., eds. *Atti del Quinto Convegno Nazionale su Sistemi Evoluti per Basi di Dati (SEBD'97)*, Verona, Italy, 1997a, p. 67–86.

[Ciaccia et al.(1997b)] Ciaccia, P.; Patella, M.; Rabitti, F.; Zezula, P. *Performance of M-tree, an Access Method for Similarity Search in Metric Spaces.* Technical Report 13, HERMES ESPRIT LTR Project, 1997b.
Disponível em `http://www.ced.tuc.gr/hermes/` (Acessado em 08/01/2004)

[Ciaccia et al.(1997c)] Ciaccia, P.; Patella, M.; Zezula, P. M-Tree: An efficient access method for similarity search in metric spaces. In: *International Conference on Very Large Databases*, 1997c, p. 426–435.
Disponível em `http://citeseer.nj.nec.com/ciaccia97mtree.html` (Acessado em 08/01/2004)

[Clarkson(1999)] Clarkson, K. Nearest neighbor queries in metric spaces. *Discrete Computational Geometry*, v. 22, n. 1, p. 63–93, 1999.

[Comer(1979)] Comer, D. The ubiquitous b-tree. *ACM Computing Surveys*, v. 11, n. 2, p. 121–137, 1979.

[Costa e Andrade Netto(2001)] Costa, J.; Andrade Netto, M. A new tree-structured self-organizing map for data analysis. In: *International Joint COnference on Neural Networks*, Washington DC: IEEE Press, 2001, p. 1931 –1936.

[Costa(1999)] Costa, J. A. F. *Classificação Automática e Análise de Dados por Redes neurais Auto-Organizáveis.* Tese de Doutoramento, UNICAMP - Universidade de Campinas (Faculdade de Engenharia Elétrica e de Computação), tese de Doutorado, 1999.

[Cottrell(1997)] Cottrell, M. Theoretical aspects of the SOM algorithm. In: *Workshop on Self-Organizing Maps*, Helsinki, Finland: University of Technology, Neural Networks Research Centre, 1997, p. 246–267.

[Cottrell et al.(1994)] Cottrell, M.; Fort, J. C.; Pagès, G. *Two or three things that we know about the kohonen algorithm.* Relatório Técnico 31, Université Paris 1, Paris, France, 1994.

[Cottrell et al.(1995)] Cottrell, M.; Fort, J. C.; Pagès, G. Comment about 'analysis of the convergence properties of topology preserving neural networks'. *IEEE Transactions on Neural Networks*, v. 6, n. 3, p. 797–799, 1995.

[Cottrell et al.(1998)] Cottrell, M.; Fort, J. C.; Pagès, G. Theoretical aspects of the SOM algorithm. *Neurocomputing*, v. 21, n. 1-3, p. 119–138, 1998.

[Cuadros-Vargas e Romero(2000)] Cuadros-Vargas, E.; Romero, R. A. F. *Estudo de Modelos de Redes Neurais para Indexação e Recuperação de Dados.* Relatório Técnico, ICMC - University of São Paulo, São Carlos - SP, Brazil, notas Didáticas, N^o 44, 2000.

[Cuadros-Vargas e Romero(2002)] Cuadros-Vargas, E.; Romero, R. A. F. A SAM-SOM Family: Incorporating Spatial Access Methods into Constructive Self-Organizing Maps. In: *International Joint COnference on Neural Networks*, Hawaii, HI: IEEE Press, 2002, p. 1172–1177.

[Cuadros-Vargas et al.(2003a)] Cuadros-Vargas, E.; Romero, R. A. F.; Mock, M.; Brisaboa, N. R. How to teach data structures design: an incremental approach, submetido para o Computer Science Education Journal, 2003a.

[Cuadros-Vargas e Romero(2004)] Cuadros-Vargas, E.; Romero, R. F. The MAM+ Family: An novel technique to speed up similarity queries progressively, submetido para o Very Large Data Bases Journal, 2004.

[Cuadros-Vargas et al.(2003b)] Cuadros-Vargas, E.; Romero, R. F.; Obermayer, K. Speeding up algorithms of SOM Family for Large and High Dimensional Databases. In: *Proceedings WSOM'03, Workshop on Self Organizing-Maps*, Hibikino-Kitakyushu, Japan, 2003b, p. 167–172.

[Cuadros-Vargas et al.(2004)] Cuadros-Vargas, E.; Romero, R. F.; Obermayer, K. A SAM-SOM Family: A novel idea to incorporate Spatial Access Methods into Self-Organizing Maps, submetido para Transactions on Neural Networks, 2004.

[Dehne e Nolteimer(1987)] Dehne, F.; Nolteimer, H. Voronoi trees and clustering problems. *Information Systems*, v. 12, n. 2, p. 171–175, 1987.

[Demian e Mignot(1993)] Demian, V.; Mignot, J. C. Optimization of the self-organizing feature map on parallel computers. In: *International Joint COnference on Neural Networks*, Nagoya, Japan: JNNS, IEEE Service Center, 1993, p. 483–486.

[Demian e Mignot(1996)] Demian, V.; Mignot, J. C. Implementation of the self-organizing feature map on parallel computers. *Computers and Artificial Intelligence*, v. 15, n. 1, p. 63–80, 1996.

[Dempster et al.(1977)] Dempster, A.; Laird, N.; Rubin, D. Maximum Likelihood from incomplete data via the EM algorithm. *Royal Statistical Society Series B*, v. 39, n. 1, p. 1–38, 1977.

[Dittenbach et al.(2000)] Dittenbach, M.; Merkl, D.; Rauber, A. The Growing Hierarchical Self-Organizing Map. In: Amari, S.; Giles, C. L.; Gori, M.; Puri, V., eds. *International Joint COnference on Neural Networks*, Como, Italy: IEEE Computer Society, 2000, p. 15–19.

[Dittenbach et al.(2001a)] Dittenbach, M.; Merkl, D.; Rauber, A. Hierarchical Clustering of Document Archives with the Growing Hierarchical Self-Organizing Map. In: *International Conference on Artificial Neural Networks*, Vienna, Austria, http://citeseer.nj.nec.com/dittenbach01hierarchical.html, 2001a.

[Dittenbach et al.(2001b)] Dittenbach, M.; Rauber, A.; Merkl, D. Recent Advances with the Growing Hierarchical Self-Organizing Map. In: Allinson, N.; Yin, H.; Allinson, L.; Slack, J., eds. *Workshop on Self-Organizing Maps*, Advances in Self-Organizing Maps, Lincoln, England: Springer, 2001b, p. 140–145 (*Advances in*

Self-Organizing Maps,).
Disponível em `http://www.ifs.tuwien.ac.at` (Acessado em 08/01/2004)

[Dittenbach et al.(2002)] Dittenbach, M.; Rauber, A.; Merkl, D. Uncovering hierarchical structure in data using the growing hierarchical self-organizing map. *Neurocomputing*, v. 48, p. 199–216, 2002.

[Duch(1994)] Duch, W. Quantitative measures for self-organizing topographic maps. *Open Systems & Information Dynamics*, v. 2, n. 3, p. 295–302, 1994.
Disponível em `http://citeseer.nj.nec.com/duch94quantitative.html` (Acessado em 08/01/2004)

[Duda e Hart(1973)] Duda, R. O.; Hart, P. E. *Pattern classification and scene analysis*. New York: Wiley, 1973.

[Erwin et al.(1992a)] Erwin, E.; Obermayer, K.; Schulten, K. I Self-Organizing Maps: Stationary states, metastability and convergence rate. *Biological Cybernetics*, v. 67, p. 35–45, 1992a.

[Erwin et al.(1992b)] Erwin, E.; Obermayer, K.; Schulten, K. II Self-Organizing Maps: Ordering, convergence properties and energy functions. *Biological Cybernetics*, v. 67, p. 47–55, 1992b.

[Fahlman(1988)] Fahlman, S. *An Empirical Study of Learning Speed in Back-Propagation Networks*. Relatório Técnico, School of Computer Science, Carnegie Mellon University, cMU-CS-88-162, 1988.

[Fahlman e Lebiere(1990)] Fahlman, S.; Lebiere, C. *The Cascade Correlation Learning Archtecture*. Relatório Técnico CMU-CS-90-100, School of Computer Science, Carnegie Mellon University, 1990.

[Faloutsos et al.(1994)] Faloutsos, C.; Barber, R.; Flickner, M.; Hafner, J.; Niblack, W.; Petkovic, D.; Equitz, W. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, v. 3, n. 3/4, p. 231–262, 1994.
Disponível em `http://citeseer.nj.nec.com/faloutsos94efficient.html` (Acessado em 08/01/2004)

[Faloutsos et al.(1993)] Faloutsos, C.; Equitz, W.; Flickner, M.; Niblack, W.; Petkovic, D.; Barber, R. Efficient and Effective Querying by Image Content. In: *IBM Almaden Research Division*, San Jose, CA, 1993.
Disponível em `http://citeseer.nj.nec.com/faloutsos94efficient.html` (Acessado em 08/01/2004)

[Faloutsos e Lin(1994)] Faloutsos, C.; Lin, K. FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Data. In: *Int. Conf. on Management of Data – ACM SIGMOD*, San Jose, CA, 1994, p. 163–174.

[Faloutsos et al.(2003)] Faloutsos, C.; Lin, K.; Cuadros-Vargas, E. Multi-Threading *FastMap*: Automatically extracting numeric features Indexing and Visualization of Traditional and Multimedia Datasets, submetido para o Knowledge Data Discovery Journal, 2003.

[Fernández-Redondo e Hernández-Espinosa(2001)] Fernández-Redondo, M.; Hernández-Espinosa, C. Weight initialization methods for multilayer feedforward. In: *European Symposium on Artificial Neural Networks*, Bruges (Belgium), iSBN 2-930307-01-3, 2001, p. 119–124.

[Finkel e Bentley(1974)] Finkel, R.; Bentley, J. Quad Trees: A Data Structure for Retrieval of Composite Keys. *Acta Informatica*, v. 4, n. 1, p. 1–9, 1974.

[Flanagan(1994)] Flanagan, J. A. *Self-organizing neural networks.* Tese de Doutoramento, École Polytechnique Fédérale de Lausanne, Lausanne, France, 1994.

[Flanagan(1996)] Flanagan, J. A. Self-organization in kohonen's som. *Neural Networks*, v. 9, p. 1185–1197, 1996.

[Folk et al.(1998)] Folk, M.; Zoellick, B.; Riccardi, G. *File Structures, An Object Oriented Approach Using C++.* 3 ed. Addison-Wesley, 1998.

[Fonseca e Jorge(2003)] Fonseca, M. J.; Jorge, J. A. Indexing high-dimensional data for content-based retrieval in large databases. In: *International Conference on Database Systems for Advanced Applications (DASFAA 2003), 8th*, Kyoto, Japan, 2003, p. 267–274.

[Frean(1989)] Frean, M. The upstar algorithm: A method for constructing and training feed-foward neural networks. Technical Report, 1989.

[Fritzke(1991)] Fritzke, B. Let It Grow—Self-Organizing Feature Maps with Problem Dependent Cell Structure. In: Kohonen, T.; Mäkisara, K.; Simula, O.; Kangas, J., eds. *Artificial Neural Networks*, Amsterdam, Netherlands: North-Holland, 1991, p. 403–408.
Disponível em `http://citeseer.nj.nec.com/fritzke91let.html` (Acessado em 08/01/2004)

[Fritzke(1992)] Fritzke, B. Growing Cell Structures—a Self-Organizing Network in k Dimensions. In: Aleksander, I.; Taylor, J., eds. *Artificial Neural Networks*, Amsterdam, Netherlands: North-Holland, 1992, p. 1051–1056.

Disponível em `http://citeseer.nj.nec.com/fritzke93growing.html` (Acessado em 08/01/2004)

[Fritzke(1993a)] Fritzke, B. *Growing cell structures–a self organizing network for unsupervised and supervised training.* Relatório Técnico TR-93-026, ICSI Berkeley, 1993a.

[Fritzke(1993b)] Fritzke, B. Kohonen feature maps and growing cell structures–a performance comparison. In: *Neural Information Processing Systems*, Morgan Kaufmann, San Mateo, CA, 1993b, p. 123–130.

[Fritzke(1994a)] Fritzke, B. Fast learning with incremental rbf networks. *Neural Processing Letters*, v. 1, n. 1, p. 2–5, 1994a.

[Fritzke(1994b)] Fritzke, B. Growing cell structures - a self-organizing networks for unsupervised and supervised learning. *Neural Networks*, v. 7, n. 9, p. 1441–1460, 1994b.

[Fritzke(1995a)] Fritzke, B. Growing grid: A self organizing network with constant neighborhood range and adaptation strength. *Neural Processing Letters*, v. 2, n. 5, p. 9–13, 1995a.

[Fritzke(1995b)] Fritzke, B. A Growing Neural Gas Networks learns topologies. *Advances in Neural Information Processing Systems*, v. 7, p. 625–632, 1995b.

[Fritzke(1997)] Fritzke, B. Some competitive learning methods. *Neural Networks*, institute of Neural Computation, 1997.
Disponível em `http://citeseer.nj.nec.com/413852.html` (Acessado em 08/01/2004)

[Fukunaga(1990)] Fukunaga, K. *Introduction to Statistical Pattern Recognition.* Academic Press, 2nd Edition, 1990.

[Gaede e Günther(1998)] Gaede, V.; Günther, O. Multidimensional Access Methods. *ACM Computing Surveys*, v. 30, n. 2, p. 170–231, 1998.

[Goldberg(1989)] Goldberg, D. E. *Genetic algorithms in search, optimization and machine learning.* Addison-Wesley Pub Co., 1989.

[Golub e Loan(1989)] Golub, G. H.; Loan, C. F. V. *Matrix computations.* second ed. Baltimore: The Johns Hopkins University Press, 1989.

[Graepel et al.(1998)] Graepel, T.; Burger, M.; Obermayer, K. Self-organizing maps: generalizations and new optimization techniques. *Neurocomputing*, v. 21, p. 173–190, 1998.
Disponível em `http://citeseer.nj.nec.com/166762.html` (Acessado em 08/01/2004)

[Gray e Moore(2000)] Gray, A. G.; Moore, A. W. 'N-Body' problems in statistical learning. *Advances in Neural Information Processing Systems*, v. 13, p. 521–527, 2000.

[Grimmer(1996)] Grimmer, U. Clementine: Data mining software. In: *Classification and Multivariate Graphics: Models, Software and Applications*, wIAS-Report No. 10, 1996.

[Grossberg(1972)] Grossberg, S. Neural expectation: Cerebellar and retinal analogs of cells fired by learnable or unlearned pattern classes. *Kybernetik*, v. 10, p. 49–57, 1972.

[Grossberg(1976a)] Grossberg, S. Adaptive pattern classification and universal recoding, I: Parallel development and coding of neural feature detectors. *Biological Cybernetics*, v. 23, p. 121–134, 1976a.

[Grossberg(1976b)] Grossberg, S. Adaptive pattern classification and universal recoding, II: Feedback, expectation, olfaction, illusions. *Biological Cybernetics*, v. 23, p. 187–202, 1976b.

[Grossberg(1988)] Grossberg, S. *Neural networks and natural intelligence.* Cambridge, MA.: MIT Press, 1988.

[Guttman(1984)] Guttman, A. R-Trees: A Dynamic Index Structure for Spatial Searching. In: *ACM SIGMOD Conference*, Boston, 1984, p. 47–57.

[Haykin(1999)] Haykin, S. *Neural networks: a comprehensive foundation.* Prentice Hall, 1999.

[Hebb(1949)] Hebb, D. O. *The organization of behavior.* New York, NY: John Wiley & Sons, 1949.

[Hellerstein et al.(1995)] Hellerstein, J. M.; Naughton, J. F.; Pfeffer, A. Generalized Search Trees for Database Systems. In: *International Conference on Very Large Databases*, Morgan Kaufmann, editors Umeshwar Dayal and Peter M. D. Gray and Shojiro Nishio, isbn 1-55860-379-4, 1995, p. 562–573.

[Hämälainen(2002)] Hämälainen, T. D. *Self-organizing neural networks: Recent advances and applications of studies in fuzziness and soft computing*, v. 78, cáp. Parallel Implementations of Self-Organizing Maps Physica-Verlag Heidelberg, p. 245–278, 2002.

[Hodge e Austin(2001a)] Hodge, V.; Austin, J. Hierarchical growing cell structures: Treegcs. *Knowledge and Data Engineering, IEEE Transactions on*, v. 13, n. 2, p. 207–218, 2001a.

[Hodge e Austin(2001b)] Hodge, V. J.; Austin, J. An integrated neural ir system. *ESANN'2001 proceedings - European Symposium on Artificial Neural Networks*, p. 265–270, eS2001-301, 2001b.

[Hodge e Austin(2002)] Hodge, V. J.; Austin, J. Hierarchical word clustering - automatic thesaurus generation. *NeuroComputing*, v. 48, n. 1-4, p. 819–846, 2002.

[Hote(1933)] Hote, H. Analysis of a Complex of Statistical Variables into Principal Components. *Educational Psychology*, v. 24, p. 417–441, 498–520, 1933.

[Hubel e Wiesel(1962)] Hubel, D.; Wiesel, T. Receptive fields, binocular interaction and functional arquitecture in the cat's visual cortex. *Journal of Physicology*, v. 160, p. 104–154, london, 1962.

[Hubel e Wiesel(1977)] Hubel, D.; Wiesel, T. Functional arquitecture of macaque visual cortex. In: *Royal Society*, London, 1977, p. 1–59.

[Huntsberger e Ajjimarangsee(1990)] Huntsberger, T. L.; Ajjimarangsee, P. Parallel self-organizing feature maps for unsupervised pattern recognition. *Int. J. General Systems*, v. 16, n. 4, p. 357–372, 1990.

[Jain(1999)] Jain, A. Data Clustering: A Review. *ACM Computing Surveys*, v. 31, n. 3, p. 264–323, 1999.

[J.W.Sammon(1969)] J.W.Sammon A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, v. C-18, n. 5, p. 401409, 1969.

[Kalantari e McDonald(1983)] Kalantari, I.; McDonald, G. A data structure and an algorithm for the nearest point problem. *IEEE Transactions on Software Engineering*, v. 9, n. 5, 1983.

[Kangas et al.(1990)] Kangas, J.; Kohonen, T.; Laaksonen, J. Variants of self-organizing maps. *IEEE Transactions on Neural Networks*, v. 1, n. 1, p. 93–99, 1990.

[Kangas et al.(1989)] Kangas, J.; Kohonen, T.; Laaksonen, J.; Simula, O.; Venta, O. Variants of self-organizing maps. In: *International Joint COnference on Neural Networks*, 1989, p. 517–522.

[Kanth et al.(1999)] Kanth, K. V. R.; Ravada, S.; Sharma, J.; Banerjee, J. Indexing medium dimensionality data in Oracle. In: *ACM SIGMOD International Conference on Management of Data*, Philadelphia, 1999, p. 521–522.

[Kaski(1999)] Kaski, S. Fast Winner Search for SOM-based Monitoring and Retrieval of High-dimensional Data. In: *International Conference on Artificial Neural Networks*, London: IEEE Conference Publication, 1999, p. 940–945.
Disponível em `http://citeseer.nj.nec.com/kaski99fast.html` (Acessado em 08/01/2004)

[Kaski(2001)] KASKI, S. Learning metrics for exploratory data analysis. In: *Signal Processing Society Workshop*, 2001, p. 53–62.

[Kaski(2003)] KASKI, S. Exploration of Gene Expression. In: *Proceedings WSOM'03, Workshop on Self Organizing-Maps*, Hibikino-Kitakyushu, Japan, plenary Talk, 2003.

[Kaski et al.(1998)] KASKI, S.; KANGAS, J.; KOHONEN, T. Bibliography of Self-Organizing Map (SOM) Papers: 1981-1997. *Neural Computing Surveys*, v. 1, p. 102–350, 1998.

[Kaski e Sinkkonen(2001)] KASKI, S.; SINKKONEN, J. A topography-preserving latent variable model with learning metrics. In: *Workshop on Self-Organizing Maps*, 2001, p. 224–229.

[Kaski et al.(2001)] KASKI, S.; SINKKONEN, J.; PELTONEN, J. Learning metrics for self-organizing maps. In: *International Joint COnference on Neural Networks*, Washington DC: IEEE Press, 2001, p. 914–919.

[Katayama e Satoh(1997)] KATAYAMA, N.; SATOH, S. The SR-tree: An Index Structure for High-Dimensional Nearest Neighbor Queries. In: *ACM SIGMOD International Conference on Management of Data*, Tucson, 1997, p. 369–380.
Disponível em `http://citeseer.nj.nec.com/katayama97srtree.html` (Acessado em 08/01/2004)

[Klinger(1971)] KLINGER, A. Pattern and search statistics. In: RUSTAGI, S., ed. *Optimizing Methods in Statistics*, 1971, p. 303–337.

[König e Michel(2003)] KÖNIG, A.; MICHEL, T. DIPOL-SOM: A Distance Preserving Enhancement of the Self-Organizing Map for Dimensionality Reduction and Multivariate Data Visualization. In: *Workshop on Self Organizing-Maps*, Hibikino-Kitakyushu, Japan, 2003, p. 219–214.

[Knuth(1973)] KNUTH, D. E. *The art of computer programming, vol. 1-3.* Addison-Wesley, Reading, MA, 1973.

[Kohonen(1982)] KOHONEN, T. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, v. 43, p. 59–69, 1982.

[Kohonen(1984)] KOHONEN, T. Self-organization and associative memory. *Springer Series of Information Science*, v. 8, 1984.

[Kohonen(1990)] KOHONEN, T. The self-organizing map. In: *Institute of Electrical and Electronics Engineers*, 1990, p. 1464–1480.

[Kohonen(1993a)] KOHONEN, T. Generalizations of the self-organizing map. In: *International Joint COnference on Neural Networks*, Nagoya, Japan, 1993a.

[Kohonen(1993b)] KOHONEN, T. Physiological interpretation of the self-organizing map algorithm. *Neural Networks*, v. 6, p. 895–905, 1993b.

[Kohonen(1997a)] KOHONEN, T. Exploration of very large databases by self-organizing maps. In: *International Conference on Neural Networks*, Houston: IEEE Service Center, 1997a, p. PL1–PL6.

[Kohonen(1997b)] KOHONEN, T. *Self-organization maps*. Springer-Verlag: Berlin, 1997b.

[Kohonen(1998)] KOHONEN, T. Self-organization of very large document collections: State of the art. In: NIKLASSON, L.; BODÉN, M.; ZIEMKE, T., eds. *International Conference on Artificial Neural Networks*, London: Springer, 1998, p. 65–74.

[Kohonen et al.(1996)] KOHONEN, T.; KASKI, S.; LAGUS, K.; HONKELA, T. Very large two-level SOM for the browsing of newsgroups. In: VON DER MALSBURG, C.; VON SEELEN, W.; VORBRÜGGEN, J. C.; SENDHOFF, B., eds. *Proceedings of ICANN96, International Conference on Artificial Neural Networks*, Bochum, Germany: Springer, p. 269–274, 1996.
Disponível em `http://citeseer.nj.nec.com/kohonen96very.html` (Acessado em 08/01/2004)

[Kohonen e Somervuo(1997)] KOHONEN, T.; SOMERVUO, P. Self-organizing maps of symbol strings with application to speech recognition. In: *Workshop on Self-Organizing Maps*, Espoo, Finland: Helsinki University of Technology, Neural Networks Research Centre, 1997, p. 2–7.

[Koikkalainen e Oja(1990)] KOIKKALAINEN, P.; OJA, E. Self-organizing hierarchical feature maps. In: *International Joint COnference on Neural Networks*, San Diego, CA, 1990, p. 279–284.

[Kolinummi et al.(2000)] KOLINUMMI, P.; PULKKINEN, P.; HAMALAINEN, T.; SAARINEN, J. Parallel implementation of self-organizing map on the partial tree shape neurocomputer. *Neural Processing Letters*, v. 12, n. 2, p. 171–182, 2000.

[Koskela(1999)] KOSKELA, M. *Content-Based Image Retrieval with Self-Organizing Maps*. Dissertação de Mestrado, Helsinki University Of Technology Department of Engineering Physics and Mathematics, Helsinki, Finland, 1999.
Disponível em `http://citeseer.nj.nec.com/369758.html` (Acessado em 08/01/2004)

[Laaksonen et al.(1999)] Laaksonen, J.; Koskela, M.; Oja, E. Application of tree structured self-organizing maps in content-based image retrieval. In: *International Conference on Artificial Neural Networks*, 1999, p. 174–179.

[Lampinen(1992)] Lampinen, J. On clustering properties of hierarchical self-organizing maps. In: Aleksander, I.; Taylor, J., eds. *Artificial Neural Networks, 2*, Amsterdam, Netherlands: North-Holland, 1992, p. 1219–1222.
Disponível em `http://citeseer.nj.nec.com/lampinen92clustering.html` (Acessado em 08/01/2004)

[Lampinen e Oja(1992)] Lampinen, J.; Oja, E. Clustering properties of hierarchical self-organizing maps. *Journal of Mathematical Imaging and Vision*, v. 2, n. 2-3, p. 261–272, 1992.

[Land e Hilton(1988)] Land, K.; Hilton, G. *The development of the time-delay neural network architecture for speech recognition.* Relatório Técnico CMU-CS-88-152, Carnegie Mellon University, Pittsburgh, PA, 1988.

[Levenshtein(1966)] Levenshtein, V. Binary codes capable of correcting deletions, insertions, and reversals. *Cybernetics and Control Theory*, v. 10, n. 8, p. 707–710, 1966.

[Lim et al.(2001)] Lim, J.-H.; Wu, J. K.; Singh, S.; Narasimhalu, D. Learning Similarity Matching in Multimedia Content-Based Retrieval. *IEEE Transactions on Knowledge and Data Engineering*, v. 13, n. 5, p. 846–850, 2001.

[Lin et al.(1994)] Lin, K.; Jagadish, H.; Faloutsos, C. The TV-tree - An Index Structure for High-Dimensional Data. *Very Large Data Bases Journal*, v. 3, n. 4, p. 517–542, santiago de Chile, 1994.

[Lo et al.(1991)] Lo, Z.; Fujita, M.; Bavarian, B. Analysis of neighborhood interaction in Kohonen Neural networks. In: *International Parallel Processing Symposium, 6th*, Los Alamitos, CA, 1991, p. 247–249.

[MacQueen(1967)] MacQueen, J. Some methods for classification and analysis of multivariate observations. *Proceedings - 5th Berkeley Symposium*, v. 1, p. 281–297, 1967.

[Martinetz(1993)] Martinetz, T. Competitive hebbian learning rule forms perfectly preserving maps. In: *International Conference on Artificial Neural Networks*, Amsterdam: Springer, 1993, p. 427–434.

[Martinetz e Schulten(1994)] Martinetz, T.; Schulten, K. J. Topology representing networks. In: *Neural Networks, 7*, Amsterdam: Springer, 1994, p. 507–522.

[Martinetz e Schulten(1991)] Martinetz, T. M.; Schulten, K. J. Network Learns Topologies. In: T. Kohonen, K. Makisara, O. S.; Kangas, J., eds. *Artificial Neural Networks*, North-Holland, Amsterdam, 1991, p. 397–402.

[Merkl(1997)] Merkl, D. Exploration of Text Collections with Hierarchical Feature Maps. In: *Research and Development in Information Retrieval*, 1997, p. 186–195. Disponível em `http://citeseer.nj.nec.com/article/merkl97exploration.html` (Acessado em 08/01/2004)

[Merkl et al.(2003)] Merkl, D.; He, S. H.; Dittenbach, M.; ; Rauber, A. Adaptive Hierarchical Incremental Grid Growing: An architecture for high-dimensional data visualization. In: *Proceedings WSOM'03, Workshop on Self Organizing-Maps*, Hibikino-Kitakyushu, Japan, 2003, p. 293–298.

[Micó et al.(1996)] Micó, L.; Oncina, J.; Carrasco, R. A fast branch and bound nearest neighbors classifier in metric spaces. *Pattern Recognition Letters*, v. 17, p. 731–739, 1996.

[Micó et al.(1994)] Micó, L.; Oncina, J.; Vidal, E. A new version of the nearest neighbors approximating and eliminating search (aesa) with linear pre-processing and memory requirements. *Pattern Recognition Letters*, v. 15, p. 9–17, 1994.

[Moody e Darken(1989)] Moody, J.; Darken, C. Fast learning in networks of locally tuned processing units. *Neural Computation*, v. 1, n. 2, p. 281–294, 1989.

[Morton(1966)] Morton, G. A computer oriented geodetic data base and a new technique in file sequencing. IBM Ltd, 1966.

[Navarro(1999)] Navarro, G. Searching in metric spaces by spatial approximation. In: *South American Symposium on String Processing and Information Retrieval*, IEEE CS Press, 1999, p. 141–148.

[Navarro(2002)] Navarro, G. Searching in metric spaces by spatial approximation. *The Very Large Databases Journal (VLDBJ)*, v. 11, n. 1, p. 28–46, 2002. Disponível em `http://www.dcc.uchile.cl/ gnavarro/ps/vldbj02.ps.gz` (Acessado em 08/01/2004)

[Navarro e Reyes(2002)] Navarro, G.; Reyes, N. Fully Dynamic Spatial Approximation Trees. In: *South American Symposium on String Processing and Information Retrieval*, LNCS 2476, Springer, 2002, p. 254–270 (*LNCS 2476*,). Disponível em `http://www.dcc.uchile.cl/ gnavarro/ps/spire02.4.ps.gz` (Acessado em 08/01/2004)

[Nene e Nayar(1997)] Nene, S.; Nayar, S. A simple algorithm for nearest neighbors search in high dimensions. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, v. 19, n. 9, p. 989–1003, 1997.

[Nolteimer(1989)] Nolteimer, H. Voronoi trees and applications. In: *International Workshop on Discrete Algorithms and Complexity*, Fukuoka Recent Hotel, Fukuoka, Japan, 1989, p. 69–74.

[Nolteimer et al.(1992)] Nolteimer, H.; Verbarg, K.; Zirkelbach, C. Monotonous Bisector* Trees - a tool for efficient partiotioning of complex schemes of geometric object. In: *Data Structures and Efficient Algorithms, LNCS 594*, Springer-Verlag, 1992, p. 186–203.

[Obermayer et al.(1990)] Obermayer, K.; Ritter, K.; Schulten, H. Large-scale simulation of a self-organizing neural network: Formation of a somatotopic map. In: Eckmiller, R., H.; Hauske, G., eds. *Parallel Processing in Neural Systems and Computers*, Urbana, Illinois: Elsevier Science Publisher B.V. (North Holland), 1990, p. 71–74.

[Ohta e Saito(2001)] Ohta, R.; Saito, T. A growing self-organizing algorithm for dynamic clustering. In: *International Joint COnference on Neural Networks*, Washington DC: IEEE Press, 2001, p. 469–473.

[Oja(2003)] Oja, E. Data and Image Mining with SOM. In: *Proceedings WSOM'03, Workshop on Self Organizing-Maps*, Hibikino-Kitakyushu, Japan, plenary Talk, 2003.

[Oja et al.(2003)] Oja, M.; Kaski, S.; Kohonen, T. Bibliography of Self-Organizing Map (SOM) Papers: 1998-2001 addendum. *Neural Computing Surveys*, v. 3, p. 1–156, 2003.

[Papadias et al.(1999)] Papadias, D.; Mamoulis, N.; Theodoridis, Y. Processing and optimization of multiway spatial joins using R-trees. In: *ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, 1999, p. 44–55.
Disponível em `http://www.cs.ust.hk/faculty/dimitris/PAPERS/pods99.pdf` (Acessado em 08/01/2004)

[Parekh et al.(1996)] Parekh, R.; Yang, J.; Honavar, V. MTiling - a constructive neural network learning algorithm for multi-category pattern classification. In: *World Congress on Neural Networks'96*, San Diego, 1996, p. 182–187.

[Parekh et al.(1997)] Parekh, R.; Yang, J.; Honavar, V. Constructive neural networks learning algorithms for multi-category real-valued pattern classification. 1997.

Disponível em `http://www.cs.iastate.edu/~honavar/aigroup.html` (Acessado em 08/01/2004)

[Parekh et al.(1995)] Parekh, R. G.; Yang, J.; Honavar, V. *Constructive neural network learning algorithms for multi-category pattern classification.* Relatório Técnico, Iowa State University, Department of Computer Science, Ames, Iowa, 1995.

[Patella(1999)] Patella, M. *Similarity search in multimedia databases.* Tese de Doutoramento, Dipartimento di Elettronica Informatica e Sistemistica, Università degli Studi di Bologna, Bologna, Italy, 1999.

[Pearson(1901)] Pearson, K. On lines and planes of closest fit systems of points in space. *Philosophical Magazine*, v. 2, p. 559–572, 1901.

[Peltonen et al.(2003)] Peltonen, K.; Klami, A.; Kaski, S. Learning Metrics for Information Visualization. In: *Workshop on Self Organizing-Maps*, Hibikino-Kitakyushu, Japan, 2003, p. 213–218.

[Press et al.(1988)] Press, W. H.; Flannery, B. P.; Teukolsky, S. A.; Vetterling, W. T. *Numerical recipes in c.* Cambridge University Press, 1988.

[Rauber e Merkl(1999)] Rauber, A.; Merkl, D. Using self-organizing maps to organize documents collections and to characterize subject matters: How to make a map tell the news of the world. In: T. Bench-Capon, G. S.; Tjoa, A., eds. *Proceedings of the 10th International Conference on Database and Expert Systems Applications (DEXA99)*, Florence, Italy: Springer, lNCS 1677 in Lecture Notes in Computer Science, 1999, p. 302–311.

[Rauber et al.(2002)] Rauber, A.; Merkl, D.; Dittenbach, M. The Growing Hierarchical Self-Organizing Map: Exploratory Analysis of High-Dimensional Data. *IEEE Transactions on Neural Networks*, v. 13, n. 6, p. 1331–1341, 2002.

[Rawtani et al.(2001)] Rawtani, L.; Rana, J.; Tiwari, A. Number of hidden nodes for shape preserving ann representation of a curve. In: *International Joint COnference on Neural Networks*, Washington DC: IEEE Press, 2001, p. 138–143.

[Reed(1993)] Reed, R. Pruning algorithmsa survey. *IEEE Transactions on Neural Networks*, v. 4, n. 5, p. 740–747, 1993.

[Riedmiller(1994)] Riedmiller, M. *RPROP - description and implementation details.* Relatório Técnico W-76128 Karlsruhe, Institut für Logik, University of Karlsruhe, 1994.

[Ritter(1995)] Ritter, H. Self-organizing feature maps: Kohonen maps. In: Arbib, M., ed. *The Handbook of Brain Theory and Neural Networks*, Cambridge, MA: MIT Press, p. 846–851, 1995.

[Ritter e Kohonen(1990)] Ritter, H.; Kohonen, T. Learning 'semantotopic maps' from context. In: *International Joint COnference on Neural Networks*, Hillsdale, NJ: Lawrence Erlbaum, 1990, p. 23–26.

[Ritter et al.(1992)] Ritter, H.; Martinetz, T. M.; Schulten, K. *Neural computation and self-organizing maps: An introduction.* Reading, MA. Addison-Wesley, 1992.

[Ritter e Schulten(1988)] Ritter, H.; Schulten, K. Convergence properties of Kohonen's topology conserving maps: Fluctuations, stability, and dimension selection. In: *Biological Cybernetics*, 1988, p. 59–71.

[Robinson(1981)] Robinson, J. The K-D-B Tree: a search structure for large multidimensional dynamic indexes. In: *ACM SIGMOD International Conference on the Management of Data*, Ann Arbor: Lawrence Erlbaum, 1981, p. 10–18.

[Roussopoulus et al.(1995)] Roussopoulus, N.; Kelley, S.; Vincent, F. Nearest neighbor queries. *Proceedings of the ACM-SIGMOD International Conference On Management of Data*, p. 71–79, san Jose, 1995.

[Rumelhart et al.(1986)] Rumelhart, D. E.; Hinton, G. E.; Williams, R. J. Learning internal representations by error propagation. In: Rumelhart, D. E.; McClelland, J. L., eds. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. I: Foundations*, Bradford Books/MIT Press, Cambridge, MA., 1986, p. 318–362.

[Samet(1995)] Samet, H. *Spatial data structures.* Modern Database Systems, Kim W ed. Reading, Addison-Wesley/ACM, 1995.

[Samuel(1959)] Samuel, A. L. Some studies in machine learning using the game of checkers. *IBM J. Res. Devel*, v. 3, p. 210–229, 1959.

[Samuel(1967)] Samuel, A. L. Some studies in machine learning using the game of checkers: recent progress. *IBM J. Res. Devel*, v. 11, p. 601–617, 1967.

[Sankoff e Kruskal(1983)] Sankoff, D.; Kruskal, J. *Time warps, string edits,and macromolecules: the theory and practice of sequence comparison.* Addison-Wesley, 1983.

[Santos-Filho(1999)] Santos-Filho, R. *Algoritmos para indexação de dados espaciais pontuais em gerenciadores de objetos.* Dissertação de Mestrado, USP - Universidade de

São Paulo (IFSC - Instituto de Física de São Carlos), São Carlos, tese de Mestrado - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 1999.

[Santos-Filho(2003)] SANTOS-FILHO, R. F. *Metrics Access Methods for similarity queries: Introducing the OMNI technique.* Tese de Doutoramento, ICMC- University of Sao Paulo, São Carlos-SP, Brazil, 2003.

[Santos-Filho et al.(2001)] SANTOS-FILHO, R. F.; TRAINA, A.; TRAINA JR., C.; FALOUTSOS, C. Similarity search without tears: The omni family of all-purpose access methods. *Proceedings of the 17th International Conference on Data Engineering*, p. 623–630, heidelberg, Germany, 2001.

[Schaeffer et al(1992)] SCHAEFFER ET AL, J. A World Championship caliber checkers program. *Artificial Intelligence*, v. 53, n. 2-3, p. 273–290, 1992.

[Schroeder(1991)] SCHROEDER, M. *Fractals, chaos, power laws: Minutes from an infinite paradise.* New York: W. H. Freeman, 1991.

[Sejnowski e Rosenberg(1987)] SEJNOWSKI, T.; ROSENBERG, C. Parallel networks that learns to pronounce English text. *Complex Systems*, v. 1, p. 145–168, 1987.

[Sellis et al.(1987)] SELLIS, T. K.; ROUSSOPOULOS, N.; FALOUTSOS, C. The R^+-Tree: A Dynamic Index for Multi-Dimensional Objects. In: *International Conference on Very Large Databases*, Brighton, England, 1987, p. 507–518.
Disponível em `http://citeseer.nj.nec.com/sellis87tree.html` (Acessado em 08/01/2004)

[Shapiro(1977)] SHAPIRO, M. The choice of reference points in best match file searching. *Communications of ACM*, v. 20, n. 5, p. 339–343, 1977.

[Shasha e Wang(1990)] SHASHA, D.; WANG, T. New techniques for best-match retrieval. *ACM Transactions on Information Systems*, v. 8, n. 2, p. 140–158, 1990.

[Shiffmann et al.(1994)] SHIFFMANN, W.; JOOST, M.; WERNER, R. *Optimization of the Backpropagation Algorithm for Training Multilayer Perceptrons.* Relatório Técnico, Institute of Physics, University of Koblenz, 1994.

[Shim(2001)] SHIM, J.-Y. Selective information retrieval from hierarchical associative knowledge learning memory. In: *International Joint COnference on Neural Networks*, Washington DC: IEEE Press, 2001, p. 1897–1901.

[Sleator e Tarjan(1985)] SLEATOR, D.; TARJAN, R. Self-adjusting binary search trees. *Journal of the ACM*, v. 32, n. 3, p. 652–686, 1985.

[Somervuo(2003)] Somervuo, P. Self-Organizing Map of Symbol String with Smooth Symbol Averaging. In: *Workshop on Self Organizing-Maps*, Hibikino-Kitakyushu, Japan, 2003, p. 122–127.

[Strang(1980)] Strang, G. *Linear algebra and its applications.* Academic Press, 2nd edition, 1980.

[Su e Basu(2001)] Su, M.; Basu, M. Input data clustering to improve neural network performance. In: *International Joint COnference on Neural Networks*, Washington DC: IEEE Press, 2001, p. 160–165.

[Suga(1985)] Suga, N. The extent to which bisonar information is represented in the bat auditory cortex. In: Edelman, G.; Gall, W.; Cowan, W., eds. *Dynamic Aspects of Neocortical Function*, New York: Wiley Interscience, p. 653–695, 1985.

[Suganthan(1999)] Suganthan, P. Hierarchical overlapped SOM's for pattern classification. *IEEE Transactions on Neural Networks*, v. 10, n. 1, p. 193–196, 1999.

[Sun et al.(2001)] Sun, H.; Wang, S.; Jiang, Q. A new validation index for determining the number of clusters in a data set. In: *International Joint COnference on Neural Networks*, Washington DC: IEEE Press, 2001, p. 1852–1857.

[Tino e Nabney(2002)] Tino, P.; Nabney, I. Hierarchical GTM: constructing localized nonlinear projection manifolds in a principled way. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, v. 24, n. 5, p. 639–656, 2002.

[Torgerson(1952)] Torgerson, W. S. Multidimensional scaling: I. theory and method. *Psychometrika*, v. 17, p. 401–419, 1952.

[Traina Jr e Faloutsos(2000)] Traina Jr, C.; Faloutsos, C. Indexação e mineração de dados multimídia. 2000.
Disponível em `http://gbdi.icmc.usp.br/imimd/` (Acessado em 08/01/2004)

[Traina Jr et al.(2000a)] Traina Jr, C.; Traina, A.; Faloutsos, C. Distance Exponent: A New Concept for Selectivity Estimation in Metric Trees. In: *International COnference on Data Engineering (ICDE)*, San Diego, CA, 2000a, p. 195.

[Traina Jr et al.(2002)] Traina Jr, C.; Traina, A. J. M.; Filho, R. F. S.; Faloutsos, C. How to Improve the Pruning Ability of Dynamic Metric Access Methods. In: *ACM International Conference on Information and Knowledge Management, 11th*, 2002, p. 219–226.

[Traina Jr et al.(2000b)] Traina Jr, C.; Traina, A. J. M.; Seeger, B.; Faloutsos, C. Slim-Trees: High Performance Metric Trees Minimizing Overlap between Nodes. In: *Advances in Database Technology - EDBT 2000, 6th International Conference on Extending Database Technology*, Konstanz, Germany: Springer, 2000b, p. 51–65 (*Lecture Notes in Computer Science*, v.1777).

[Uhlmann(1991a)] Uhlmann, J. Implementing metric trees to satisfy general/proximity queries. Manuscript, 1991a.

[Uhlmann(1991b)] Uhlmann, J. Satisfying General Proximity/Similarity Queries with Metric Trees. *Information Processing Letter*, v. 40, p. 175–179, 1991b.

[Ultsch(1993a)] Ultsch, A. Knowledge Extraction from Self-Organizing Neural Networks. In: et al., O. O., ed. *Information and Classification*, Springer, Berlin, 1993a, p. 301–306.

[Ultsch(1993b)] Ultsch, A. Self-Organizing Neural Networks for Visualization and Classification. In: et al., O. O., ed. *Information and Classification*, Springer, Berlin, 1993b, p. 307–313.

[Ultsch(2003)] Ultsch, A. Maps for the Visualization of high-dimensional Data Spaces. In: *Workshop on Self Organizing-Maps*, Hibikino-Kitakyushu, Japan, 2003, p. 225–230.

[Verbarg(1995)] Verbarg, K. The C-Tree a dinamically balanced index. *Computing Suppl.*, v. 10, p. 323–340, 1995.

[Vicentini(2002)] Vicentini, J. F. *Indexação e Recuperação de Informação utilizando Redes Neurais da Família ART.* Dissertação de Mestrado, ICMC- University of Sao Paulo, São Carlos-SP, Brasil, 2002.

[Vicentini e Romero(2003)] Vicentini, J. F.; Romero, R. A. F. Utilização de Redes Neurais da Família ART para Indexação e Recuperação de Informações. In: Torre, G. L.; Abe, J. M.; Mucheroni, M. L.; Cruvinel, P. E., eds. *Congress of Logic Applied to Technology (LAPTEC'2003), 4th*, Marilia-SP: Plêiade, 2003, p. 195–202.

[Vidal(1986)] Vidal, E. An algorithm for finding nearest neighbors in (approximately) constant average time. *Pattern Recognition Letters*, v. 4, p. 145–157, 1986.

[Villmann et al.(1997)] Villmann, T.; Der, R.; Hermann, M.; Martinetz, M. Topology preservation in self-organizing feature maps: Exact definition and measurement. *IEEE Transactions on Neural Networks*, v. 8, n. 1, p. 256–266, 1997.

[Voronoi(1907)] Voronoi, G. F. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. *J. Reine Angew, Math*, v. 1, n. 133, p. 97–178, 1907.

[Wactlar et al.(1999)] Wactlar, H. D.; Christel, M. G.; Gong, Y.; Hauptmann, A. G. Lessons learned from building a terabyte digital video library. *IEEE Computer*, v. 32, n. 2, p. 66–73, 1999.

[Wactlar et al.(1996)] Wactlar, H. D.; Kanade, T.; Smith, M. A.; Stevens, S. M. Intelligent access to digital video: Informedia project. *IEEE Computer*, v. 29, n. 5, p. 46–52, 1996.

[Waibel et al.(1989)] Waibel, A.; Hanazawa, T.; Hinton, G.; Shikano, K.; Lang, K. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech and Signal Processing*, v. ASSP-37, p. 328–339, 1989.

[Webster(1980)] Webster, R. *B+ trees*. Dissertação de Mestrado, Oklahoma State University, 1980.

[Willshaw e von der Malsburg(1976)] Willshaw, D.; von der Malsburg, C. How pattern neural connections can be set up by self-organization. In: *Royal Society of London*, 1976, p. 431–445 (*B*, v.194).

[Wilson e Martinez(2001)] Wilson, D. R.; Martinez, T. R. The need for small learning rates on large problems. *Proceedings. IJCNN '01. International Joint Conference on Neural Networks*, v. 1, p. 115 –119, 2001.

[Yamakawa(2003)] Yamakawa, T. Database of Self-Organizing Maps Papers. 2003. Disponível em `http://www.brain.kyutech.ac.jp/~yamakawa/` (Acessado em 14/sep/2003)

[Yianilos(1993)] Yianilos, P. N. Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces. *4th ACM SIAM Symposium on Discrete Algorithms (SODA'93)*, p. 311–321, austin, 1993.

[Yianilos(1998)] Yianilos, P. N. *Excluded Middle Vantage Point Forests for Nearest Neighbor Search*. Relatório Técnico, NEC Research Institute, Princeton, 1998.

[Yianilos(1999)] Yianilos, P. N. Excluded middle vantage point forest for nearest neighbor search. In: *DIMACS Implementation Challenge (ALENEX'99)*, Baltimore, MD, 1999.

[Yianilos(2000)] Yianilos, P. N. Locally lifting the curse of dimensionality for nearest neighbor search. In: *ACM-SIAM Symposium on Discrete Algorithms (SODA'00), 11th*, 2000, p. 361–370.

[Zezula et al.(1996)] Zezula, P.; Ciaccia, P.; Rabitti, F. *M-Tree: A Dynamic Index for Similarity Queries in Multimedia Databases*. Technical Report 7, HERMES ESPRIT

LTR Project, 1996.
Disponível em `http://www.ced.tuc.gr/hermes/` (Acessado em 08/01/2004)

[Zezula et al.(1998a)] Zezula, P.; Savino, P.; Amato, G.; ; Rabitti, F. Approximate Similarity Retrieval with M-Trees. *The VLDB Journal*, v. 7, n. 4, p. 275–293, 1998a.

[Zezula et al.(1998b)] Zezula, P.; Savino, P.; Rabitti, F.; Amato, G.; Ciaccia, P. Processing M-Trees with Parallel Resources. In: *International Workshop on Research Issues in Data Engineering (RIDE'98), 8th*, Orlando, FL, 1998b, p. 147–154.