UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

Orpheo - Uma Estrutura de Trabalho para Integração dos Paradigmas de Aprendizado Supervisionado e Não-Supervisionado

por

HÉRCULES ANTONIO DO PRADO

Tese submetida à avaliação,
como requisito parcial para a obtenção do grau de Doutor
em Ciência da Computação

Prof. Paulo Martins Engel
Orientador

Porto Alegre, outubro de 2001

CIP - CATALOGAÇÃO NA PUBLICAÇÃO

Prado, Hércules Antonio do

Orpheo - Uma Estrutura de Trabalho para Integração dos Paradigmas de Aprendizado Supervisionado e Não-Supervisionado / por Hércules Antonio do Prado. - Porto Alegre: PPGC da UFRGS, 2001.

154 p.:il.

Tese (Doutorado) - Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR-RS, 2001. Orientador: Engel, Paulo Martins.

1. Descoberta de Conhecimento em Bases de Dados. 2. Mineração de Dados. 3. Aprendizado Não-Supervisionado. 4. Aprendizado Supervisionado. 5. *Frameworks*. I. Engel, Paulo Martins. II. Título.

Agradecimentos

"Não existe nada maior nem menor do que um toque."
(Walt Whitman)

Dirijo meu principal agradecimento ao professor Paulo Martins Engel, por sua orientação dedicada, comprometida, motivadora e de total envolvimento, que ao final tornou-se uma relação de amizade que transcende os limites acadêmicos.

Diversas outras pessoas merecem os meus agradecimentos por várias razões. Um simples toque, na forma de um comentário, um esclarecimento, uma conversa qualquer, durante a árdua empreitada que constitui um programa de doutorado, pode representar um importante passo em direção ao tão almejado foco da tese, ou à definição do problema de pesquisa. Na maioria das vezes, os portadores desses toques não têm consciência da importância dos seus atos. Considero fundamental o reconhecimento do mérito dessas pessoas, às quais endereço parte destes agradecimentos. Por outro lado, o suporte tático e financeiro de diversas instituições na viabilização deste trabalho, por dever de consciência, é também aqui reconhecido.

Sou grato ...

... aos doutores Carlos Magno Campos da Rocha, Eduardo Delgado Assad e Euzébio Medrado da Silva, da Embrapa Cerrados, aos quais competia convalidar ou não os resultados do processo de seleção de candidatos ao programa de doutorado daquele centro de pesquisa.

... ao padre Décio Batista Teixeira e ao professor Guy Capdeville, da Universidade Católica de Brasília, pela oportunidade de trabalhar em uma instituição que valoriza e estimula, de maneira clara e comprometida, o trabalho de pesquisa. Aos amigos professores Marcos Costa e Sérgio Moraes por acreditarem na minha proposta e por emprestarem seu prestígio ao fornecerem-me as cartas de recomendação. À Leonor, da Secretaria de Pós-Graduação na Católica, pela ajuda que me prestou em diversas situações junto à UCB.

... ao professor Waldir Roque ("Uma tese deve estar sempre associada a três atributos: relevância, originalidade e não-trivialidade"), do Instituto de Matemática da UFRGS, pelo permanente incentivo, pelos caminhos abertos, e pela amizade construída ao longo de nossa convivência. Ao professor Vítor Haertel, do Centro de Sensoriamento Remoto da UFRGS, por mostrar-me o "outro lado da moeda" em que se constituem os métodos estatísticos para reconhecimento de padrões. Aos professores Philippe Navaux e Carla Maria Dal Sasso Freitas que, à frente de instâncias decisórias do Instituto de Informática da UFRGS, nunca me faltaram com o seu apoio.

... à minha procuradora e amiga Neusa Alice dos Santos, por ter resolvido para mim diversos problemas de minha responsabilidade em Brasília. Ao meu amigo e conselheiro acadêmico da Embrapa, Homero Chaib Filho, por todas as providências que tomou no sentido de garantir o transcurso dessa jornada sem maiores tropeços junto aos órgãos de controle da Embrapa. Aos amigos Alfredo José Barreto Luiz e Orpheo Apolo Droguet Affin (*in memoriam*) ("O impossível não existe em pesquisa!"), pelas enriquecedoras discussões sobre o método científico e com os quais tanto troquei em termos da luta que se iniciaria. Ao doutor José da Silva Madeira Neto, da Embrapa, por

um dia ter dito algo mais ou menos assim: “Ou você sai ou você sai para este doutorado!”

... aos doutores Stephen Hirtle e Paul Munro (“*Act carefully!*”), da Universidade de Pittsburgh, pelas valiosas discussões durante o programa de doutorado “sanduíche”, em 1999.

... ao Luis Otávio, Margareth e Cristina Nunes, pelo inestimável suporte provido através do Laboratório do Instituto de Informática. Aos amigos da Biblioteca do Instituto de Informática – Ida, Zita, Beatriz, Henrique, Adriana e Grace – por facilitarem-me ao máximo o acesso à informação e pela ajuda com respeito à normalização das publicações. Às colegas da Secretaria do Pós – Ângela, Rose e Elisiane - pela simpatia com que sempre me atenderam.

... aos amigos: Miguel Feldens, Leandro Wives, Patrícia Jaques, Aluízio Haendchen Filho, Sandra Rovena, Ronaldo Gonçalves, Edson Prestes, Mauro Muñoz, Raul Ceretta, Maurício Pilla e Patrícia Pitthan, cada qual tendo o seu papel durante toda a nossa convivência. Em particular aos veteranos Jugurta (“Artigos de 1 e 99 não servem!”), Beatriz Franciosi, Ricardo Bastos, Lúcia Giraffa (“Uma tese é uma contribuição local para a solução de um problema relevante.”), Ricardo Silveira, Marcelo Ladeira e professor Tiarajú Divério. Por último, mas não menos importante, à Karla Fedrizzi, com quem divido a autoria de alguns artigos, pela presteza e disponibilidade para cooperar.

... ao doutor Gilberto Roca da Cunha, da Embrapa Trigo, que prestou valioso auxílio na definição da aplicação utilizada para validar parte da pesquisa.

... aos queridos Isa e Eugenio Bruck, pela revisão gramatical e ortográfica do texto final da tese.

... à Nilza, pelo inestimável apoio no retorno a Brasília.

Dirijo um mais do que merecido agradecimento à minha família por todo o apoio recebido, em particular à minha esposa, Adriana, e aos meus filhos, Elisa e Daniel, que, durante muitas oportunidades, se viram sem o pai, além do prolongado período sem férias.

Finalmente, agradeço à Empresa Brasileira de Pesquisa Agropecuária - Embrapa, à Universidade Católica de Brasília e à Universidade Federal do Rio Grande do Sul. Às duas primeiras, pelo suporte financeiro a esta empreitada e, à última, pela acolhida.

Aos meus pais,
Alberto e Onésia
(*in memoriam*).

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

Esta tese apresenta contribuições ao processo de Descoberta de Conhecimento em Bases de Dados (DCBD). DCBD pode ser entendido como um conjunto de técnicas automatizadas – ou semi-automatizadas – otimizadas para extrair conhecimento a partir de grandes bases de dados. Assim, o já, de longa data, praticado processo de descoberta de conhecimento passa a contar com aprimoramentos que o tornam mais fácil de ser realizado. A partir dessa visão, bem conhecidos algoritmos de Estatística e de Aprendizado de Máquina passam a funcionar com desempenho aceitável sobre bases de dados cada vez maiores. Da mesma forma, tarefas como coleta, limpeza e transformação de dados e seleção de atributos, parâmetros e modelos recebem um suporte que facilita cada vez mais a sua execução.

A contribuição principal desta tese consiste na aplicação dessa visão para a otimização da descoberta de conhecimento a partir de dados não-classificados. Adicionalmente, são apresentadas algumas contribuições sobre o Modelo Neural Combinatório (MNC), um sistema híbrido neurossimbólico para classificação que elegemos como foco de trabalho.

Quanto à principal contribuição, percebeu-se que a descoberta de conhecimento a partir de dados não-classificados, em geral, é dividida em dois subprocessos: identificação de agrupamentos (aprendizado não-supervisionado) seguida de classificação (aprendizado supervisionado). Esses subprocessos correspondem às tarefas de rotulagem dos itens de dados e obtenção das correlações entre os atributos da entrada e os rótulos. Não encontramos outra razão para que haja essa separação que as limitações inerentes aos algoritmos específicos. Uma dessas limitações, por exemplo, é a necessidade de iteração de muitos deles buscando a convergência para um determinado modelo. Isto obriga a que o algoritmo realize várias leituras da base de dados, o que, para Mineração de Dados, é proibitivo. A partir dos avanços em DCBD, particularmente com o desenvolvimento de algoritmos de aprendizado que realizam sua tarefa em apenas uma leitura dos dados, fica evidente a possibilidade de se reduzir o número de acessos na realização do processo completo. Nossa contribuição, nesse caso, se materializa na proposta de uma estrutura de trabalho para integração dos dois paradigmas e a implementação de um protótipo dessa estrutura utilizando-se os algoritmos de aprendizado ART1, para identificação de agrupamentos, e MNC, para a tarefa de classificação. É também apresentada uma aplicação no mapeamento de áreas homogêneas de plantio de trigo no Brasil, de 1975 a 1999.

Com relação às contribuições sobre o MNC são apresentados: (a) uma variante do algoritmo de treinamento que permite uma redução significativa do tamanho do modelo após o aprendizado; (b) um estudo sobre a redução da complexidade do modelo com o uso de máquinas de comitê; (c) uma técnica, usando o método do envoltório, para poda controlada do modelo final e (d) uma abordagem para tratamento de inconsistências e perda de conhecimento que podem ocorrer na construção do modelo.

Palavras-Chave: Descoberta de Conhecimento em Bases de Dados, Mineração de Dados, Aprendizado Não-Supervisionado, Aprendizado Supervisionado, *Frameworks*.

TITLE: "ORPHEO - A FRAMEWORK FOR INTEGRATION OF SUPERVISED AND UNSUPERVISED LEARNING PARADIGMS".

Abstract

This thesis presents contributions for the process of Knowledge Discovery in Databases (KDD). KDD can be understood as a set of automated, or semi-automated, techniques optimized to extract knowledge from large databases. By this way, the already long-standing practiced process of knowledge discovery has received improvements that make it easier to be performed. From this point of view, well known Statistics and Machine Learning algorithms become more efficient when running on databases of increasing size. In addition, tasks concerning data collection, cleansing and transforming, as well as selection of attributes, parameters and models have received improvements each day.

The main contribution of this thesis consists in the application of this vision to the optimization of knowledge discovery from non-classified data. Moreover, some contributions on Combinatorial Neural Model (CNM), a neural-symbolic classification system that we focused in our work, are also presented.

Regarding to the main contribution, it is observed that knowledge discovery from non-classified data, in general, is divided into two sub-processes: identification of groups (unsupervised learning) followed by classification (supervised learning). These sub-processes correspond to the tasks of data labeling and finding the correlation between the data and the labels. We could not find another reason for this separation than the inherent limitations of the specific algorithms for each sub-process. One of these limitations, for example, is that many of them require iterations over the entire data set in order to converge to a model. This iteration compels the algorithm to make many readings of the database, which is prohibitive in the context of Data Mining, the core of KDD. From the advances in KDD, particularly with the development of learning algorithms that carry out their tasks in just one data scan, it is evident the possibility of reducing the number of accesses to the database in the accomplishment of the full process. Our contribution, in this case, is the proposal of a framework for integration of the two paradigms and the implementation of a prototype of this framework using the algorithms ART1, for identification of groups, and CNM, for the classification task. Also an application that identifies homogeneous areas of wheat cropping in Brazil, between 1975 and 1999, is presented.

Finally, contributions related to CNM are presented: (a) a variant of the training algorithm that leads to a significant reduction in the size of the model after learning; (b) a study on the reduction of the complexity of the model with the use of a committee machine; (c) a technique, using a wrapper, for a controlled pruning of the final model and (d) an approach to cope with inconsistencies and knowledge loss that can occur while building the model.

Keywords: Knowledge Discovery from Databases, Data Mining, Unsupervised Learning, Supervised Learning, Frameworks.

1 Introdução

O formato desta tese foi inspirado na de George H. John [JOH 97], na qual o autor parte do reconhecimento do caráter multidisciplinar da matéria e propõe melhorias em diversos aspectos do Processo de Descoberta de Conhecimento em Bases de Dados (DCBD), não abordando apenas um problema localizado.

Partindo da compreensão do processo de DCBD como a aplicação auxiliada por computador das técnicas de descoberta de conhecimento, otimizadas para grandes bases de dados, apresentamos contribuições sob os aspectos de facilitação de algumas tarefas do processo de descoberta de conhecimento e alternativas para se abordar o problema da explosão combinatória de uma técnica de aprendizado supervisionada específica, o Modelo Neural Combinatório (MNC) [MAC 89]. A escolha do MNC como objeto de pesquisa deu-se pelo fato de o mesmo possuir duas importantes características para Mineração de Dados (MD): aprendizado incremental, evitando a custosa iteratividade da maioria dos algoritmos de aprendizado, e mapeamento direto do modelo aprendido em regras simbólicas, sem perda de significado. Dessa forma, reconhecendo as vantagens do MNC e visando torná-lo mais adequado para DCBD, direcionamos parte do nosso esforço de pesquisa para o estudo de alternativas para mitigar o problema da explosão combinatória.

Com respeito à facilitação do processo de descoberta de conhecimento, nossas contribuições foram: (a) uma estrutura de trabalho (*framework*) para identificação e caracterização de agrupamentos baseada na integração de algoritmos de aprendizado não-supervisionado e supervisionado, buscando-se atender aos requisitos desejáveis a sistemas de Mineração de Dados; (b) uma alternativa para poda do MNC, que permite um melhor balanceamento entre o fator de crença do modelo aprendido, calculado com base em probabilidades *a priori*, e a precisão do modelo, dada pela taxa de erro com relação a um conjunto de dados de fora do treinamento (*off-training set*) e (c) uma solução para o pós-processamento do MNC, que inclui o tratamento de inconsistências e prevenção da perda de conhecimento que ocorre com a sua poda.

Com relação ao tratamento da complexidade do MNC, são apresentadas as seguintes contribuições: (a) uma alternativa para a construção do modelo por contingência, com ganhos significativos em termos do número de combinações geradas e (b) um estudo sobre os ganhos com a adoção do conceito de Máquina de Comitê [HAY 99] como forma de se reduzir o problema da expansão do modelo.

Este volume constitui-se de cinco capítulos. O capítulo 2 – **Descoberta de Conhecimento em Bases de Dados** – apresenta uma discussão sobre o contexto da proposta. No capítulo 3 – **Orpheo – Uma Estrutura de Trabalho para Integração dos Paradigmas de Aprendizado Supervisionado e Não-Supervisionado** – é proposta uma estrutura de trabalho que permite a realização da descoberta de conhecimento, a partir de dados não-classificados, de forma mais flexível, levando, em alguns casos, a ganhos de desempenho quanto ao acesso à base de dados. Como forma de se avaliar a efetividade dessa estrutura, é apresentado um protótipo desenvolvido com base nas redes neurais ART1, para identificação de agrupamentos, e MNC, para classificação. No capítulo 4 – **Contribuições sobre o Modelo Neural Combinatório** – são identificados alguns problemas que dificultam a utilização desse modelo para DCBD e propostas algumas soluções. Especificamente, são propostas alternativas para: (a) tratar o problema da explosão combinatória, inerente ao modelo; (b) controlar o processo de poda, visando obter maior generalidade; (c) tratar as inconsistências que

podem surgir do fato de o modelo poder ser criado com base em conhecimento e dados de fontes diversas e (d) reduzir o problema da perda de conhecimento decorrente da poda combinada com a realização de aprendizado incremental.

Durante a escrita do texto tentamos utilizar, preferencialmente, expressões ou palavras em português. Para evitar prejuízos na compreensão das idéias, sempre que julgamos necessário, colocamos à frente de uma expressão traduzida a sua equivalente em inglês, entre parênteses, utilizando fonte em itálico. Ainda assim, em alguns casos em que encontramos dificuldade para a tradução foram mantidos os termos em inglês, com fonte em itálico.

2 Descoberta de Conhecimento em Bases de Dados

A Descoberta de Conhecimento em Bases de Dados encontra-se na confluência de diversas áreas, incluindo, entre outras, Estatística, Inteligência Artificial (Redes Neurais e Sistemas Simbólicos) e Banco de Dados. A disciplina que daí resulta traz como novidade o auxílio do computador não somente na execução dos algoritmos para a identificação de padrões, mas em todas as etapas que constituem o processo de descoberta escalado para grandes bases de dados. Tarefas como coleta, limpeza e redução de dados, seleção de parâmetros para a execução dos algoritmos para identificação dos padrões, e também a representação e processamento do conhecimento, quando facilitadas pelo uso do computador e otimizadas para grandes bases de dados, encontram-se no âmbito da DCBD. Por outro lado, considerando o permanente avanço na capacidade de processamento dos computadores, aliado ao crescente volume de dados sendo armazenados a todo momento, definir o que são "grandes bases de dados" torna-se um exercício extremamente subjetivo. O que vemos na prática é um esforço constante para a viabilização do processamento de quantidades de dados cada vez maiores. Mesmo definições autorizadas passam ao largo de qualquer consideração sobre o que são "grandes bases de dados". Fayyad [FAY 96], por exemplo, define assim DCBD e Mineração de Dados (MD):

> *Descoberta de Conhecimento em Bases de Dados* é o processo que usa algoritmos de Mineração de Dados para extrair conhecimento, de acordo com especificações de medidas e limites, usando uma base de dados F, juntamente com algum pré-processamento, subamostragem, e transformações requeridas sobre F.
>
> *Mineração de Dados* é um passo do processo de DCBD, consistindo de algoritmos que, sob alguma limitação de eficiência computacional aceitável, produzem uma particular enumeração de padrões sobre um conjunto de dados.

Como todas as atividades mencionadas nas definições acima já eram desempenhadas isoladamente, resta que a principal característica da DCBD é colocar tudo sob uma única disciplina, que se beneficia do uso intensivo do computador como facilitador do processo e da busca constante de expansão da capacidade de processamento dos algoritmos. Fazendo uma analogia, da mesma forma que temos a área de *CASE (Computer-Aided Software Engineering)* podemos considerar DCBD como *CAKD (Computer-Aided Knowledge Discovery)*, adicionada a busca por expansão da capacidade de processamento.

Nesse contexto, a pesquisa em DCBD tem se realizado em duas dimensões principais: (1) expansão da capacidade de processamento dos algoritmos de aprendizado para bases de dados cada vez maiores e (2) automatização ou semi-automatização do processo de descoberta de conhecimento. A segunda dimensão refere-se a tarefas como coleta e limpeza de dados e seleção de parâmetros e de atributos.

Num cenário típico de aplicação de DCBD existem dois papéis básicos: o do especialista e o do analista de mineração. O primeiro traz um problema para ser resolvido, conhece bem o contexto do mesmo e dispõe de dados que podem ser utilizados para resolvê-lo. O segundo domina os passos básicos do processo de DCBD. A interação entre os dois é que levará à descoberta de padrões que podem ser utilizados na solução do problema. Vale lembrar o que diz Fayyad [FAY 96] com relação à DCBD: "DCBD é o processo não trivial de identificação, em dados, de padrões válidos,

novos, potencialmente úteis, e inteligíveis.” Esse deve ser o fim da interação entre o especialista e o analista de mineração. O processo começa com a definição e o entendimento de um problema, por ambas as partes, e termina com a análise dos resultados e uma estratégia para a aplicação desses na solução do problema. É importante que o especialista torne bem claro para o analista o problema a ser resolvido e que esse último explique ao primeiro como funciona o processo de DCBD. Após os dois estarem nivelados na compreensão do problema e na estratégia para a sua solução, o analista deverá efetuar a seleção, coleta e extração dos dados relevantes para a solução do problema, limpá-los, realizar a sua engenharia conforme as necessidades dos passos intermediários do processo de mineração, realizar a mineração e exibir os resultados para o especialista validá-los.

Neste capítulo apresentamos uma descrição geral do processo de DCBD a partir de visões autorizadas ([FAY 96] e [JOH 97]) da comunidade de pesquisa e de algumas experiências do próprio autor.

2.1 As Fases do Processo

O processo de DCBD pode ser apresentado como uma seqüência linear de etapas cuja execução, entretanto, se dá de forma iterativa, no sentido em que se pode avançar nas etapas e posteriormente voltar a uma que já tenha sido executada, e interativa pelo fato de se basear intensamente nas entradas e respostas providas pelo usuário.

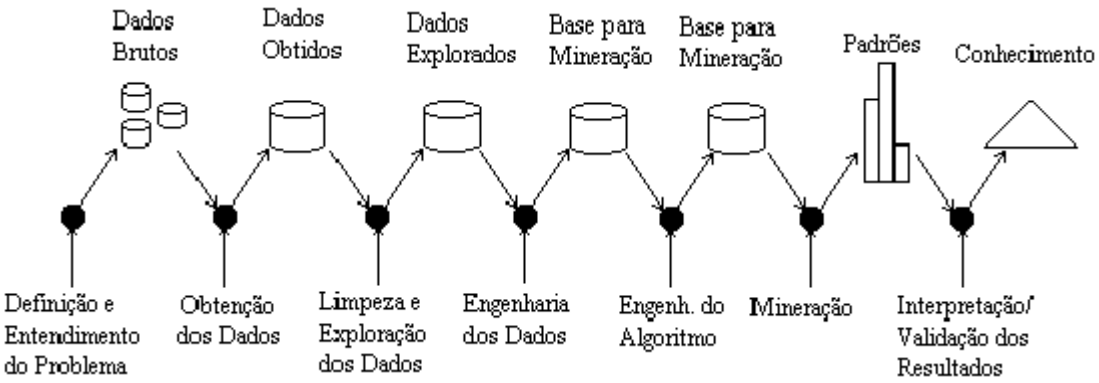

FIGURA 2.1 - O Processo de DCBD

O processo completo de DCBD, ilustrado na FIGURA 2.1 [PRA 98a], consta das etapas de definição e entendimento do problema, obtenção dos dados, limpeza e exploração dos dados, engenharia dos dados, engenharia do algoritmo, mineração, interpretação e validação dos resultados, conforme classificação proposta por John [JOH 97], ajustada às definições para DCBD e MD de Fayyad [FAY 96].

É comum referir-se a esse processo como composto por três grandes fases: (a) pré-processamento – incluindo as etapas de definição e o entendimento do problema, a obtenção dos dados, a limpeza e a exploração dos dados, a engenharia dos dados e a engenharia do algoritmo de mineração; (b) mineração e (c) pós-processamento – referindo-se à etapa de interpretação e validação dos resultados, seguida da sua efetiva utilização.

O processo completo de DCBD envolve um elevado grau de subjetividade e de trabalho não totalmente automatizado. Podemos dizer que a fase mais automatizada é a de mineração, havendo ferramentas de suporte para as demais fases.

Apesar de desejável, a idéia de se criarem ferramentas que integrem as facilidades desenvolvidas para abarcar o processo completo, incluindo as fases de pré-processamento, mineração dos dados e pós-processamento dos resultados, ainda está longe de se concretizar. Isso ocorre devido às várias alternativas de implementação de cada uma das fases dependentes, na maioria dos casos, do domínio da aplicação. Uma alternativa é a construção de ferramentas voltadas para domínios específicos, como mercadologia ou finanças, por exemplo. Em qualquer das duas situações, a existência de padrões metodológicos para DCBD facilitaria a integração dos diversos métodos e técnicas. A esse respeito, o consórcio CRISP-DM (*CRoss-Industry Standard Process for Data Mining*) [CHA 98], patrocinado pela Comissão Européia, reuniu pesquisadores e desenvolvedores de diversas universidades e indústrias de *software*, buscando um padrão desde 1997. Acreditamos que o surgimento de um padrão como esse, desde que consolidado entre a comunidade de usuários, permitirá que se caminhe para a construção de estruturas de trabalho para DCBD, entendidas como ambientes flexíveis para se montarem ferramentas adaptadas para domínios específicos.

A necessidade de ferramentas para análise de grandes bancos de dados de forma automática surge como uma decorrência da situação de "afogamento em dados e fome de conhecimento" [FAY 96] em que nos encontramos. Embora o processo de extração de padrões a partir de dados seja algo antigo, a DCBD busca fazer isso da forma mais automática possível. Na verdade, o processo é tipicamente interativo: alguém que tenha um problema e uma base de dados relevante para a resolução desse problema interage com um sistema de DCBD na busca por padrões que possam ser usados para a solução do problema, não se excluindo mudanças de rumos previamente traçados em função de resultados parciais alcançados durante a mineração. O exemplo clássico é o do banco que possui um histórico dos empréstimos concedidos, com algumas características dos tomadores e o resultado (sucesso ou insucesso) da transação. O banco quer saber quais características dos tomadores condicionam o insucesso de uma transação, de forma que as próximas sejam realizadas com menor risco. Analisando os dados disponíveis, um algoritmo de mineração pode encontrar padrões passíveis de serem usados para predizer quais clientes apresentam maior risco de não pagamento, e em que grau, de modo a serem evitados.

Os avanços já alcançados pela pesquisa para atender a necessidades desse tipo são conseqüências da convergência de interesses de diversas áreas que antes vinham trabalhando isoladamente, muitas vezes sobre temas afins, sem capitalizarem os resultados umas das outras. Áreas como Estatística, Redes Neurais, Aprendizado de Máquina, Reconhecimento de Padrões e Bancos de Dados passaram a somar esforços na arena da DCBD. Essa união de interesses tem permitido grandes avanços na área, com a geração de ferramentas que realmente auxiliam na resolução de problemas com o uso de bancos de dados.

Uma outra área que apresenta uma sinergia muito grande com DCBD é a de *data warehousing*, que tem por finalidade o tratamento de dados transacionais, visando torná-los disponíveis *on-line* sob diferentes formas de agregação. O padrão de mercado para análise de dados em *data warehousing* é o chamado OLAP (*on-line analytical process*), cujo foco é a análise multidimensional de dados. OLAP e DCBD podem ser vistos como facetas de uma mesma geração de ferramentas para gerência e extração inteligentes de informação e conhecimento de bancos de dados.

2.2 Definição e Entendimento do Problema

É evidente a necessidade de se definir e entender bem o problema a ser solucionado. Todavia, consideramos importante um exame mais detalhado dessa atividade óbvia. Sem um claro entendimento do problema e do objetivo que se quer atingir, os resultados de um processo de mineração poderão não ter qualquer valor. Por definição do problema queremos dizer que o analista de mineração deveria trabalhar com o especialista para tornar o problema específico o suficiente para ser solucionável e para que os resultados possam ser medidos. No exemplo do crédito bancário, o problema poderia ter sido colocado como "reduzir o número de inadimplências de crédito pessoal". Um problema colocado dessa forma é geral o bastante para comportar diversas interpretações e permitir múltiplas abordagens de solução, algumas inclusive infrutíferas. Uma solução possível, caso não haja um maior detalhamento do problema, poderia ser a redução das liberações de empréstimo! Muito provavelmente isso reduziria o número absoluto de inadimplências, mas seria algo como matar um paciente para que ele não tenha mais doenças ...

Uma decisão importante, diretamente condicionada por essa etapa, é a definição do modelo a ser construído. Discutiremos na seção 2.7 os tipos de modelos e algumas técnicas existentes para a sua construção.

2.3 Obtenção e Extração dos Dados

Essa fase consiste na obtenção de dados relevantes para a solução do problema. Esses dados podem vir de várias fontes: bancos de dados operacionais ou de um *data warehouse*, adquiridos de terceiros, estatísticas públicas e outras. No problema de crédito bancário, o banco, provavelmente, não possui um arquivo único contendo os dados relevantes para a mineração. Esses dados estão espalhados em tabelas pertencentes a diversos sistemas: cadastro, empréstimos, cobrança e outros. Nós temos que construir um arquivo com os dados necessários para a mineração. O nome, endereço, idade, profissão e renda devem vir do sistema de cadastro. Se aspectos geográficos forem importantes, talvez seja necessário cruzar os nossos dados com arquivos de terceiros, mapeando os clientes por região onde moram. Os dados referentes a concessões de crédito virão do sistema de empréstimo, e os resultados da transação serão extraídos das tabelas do sistema de cobrança.

Para a obtenção desse arquivo duas atividades devem ser desempenhadas: a primeira consiste em um exercício intelectual do analista e do especialista para a definição dos atributos a serem considerados, e a segunda se resume na extração física dos dados das diversas fontes. A primeira envolve decisões importantes e, infelizmente, com alto grau de subjetividade. A escolha dos atributos que podem influenciar uma determinada condição pode cair em um processo circular onde o que se descobre é exatamente o que já se sabe. Isso ocorre em conseqüência da especificação dos atributos dentro de um conhecimento prévio de sua influência sobre a condição. Por outro lado, não se pode deixar todos os atributos serem tratados, sob o risco de se sobrecarregar desnecessariamente o mecanismo de mineração. Não existe uma receita para se determinar o conjunto ótimo de atributos, ficando como uma questão a ser examinada caso a caso.

A segunda atividade é extremamente árida e tediosa, podendo ainda trazer surpresas desagradáveis: ocorrência de exceções (e.g., linhas de total no meio do arquivo); apresentação de arquivos em formatos diferentes do que consta na

documentação, exigindo a reconstrução dos mesmos no formato esperado; falta de padrão na codificação (e.g., um arquivo usa um código de município gerado internamente e outro usa o código do IBGE). E, finalmente, quando pensávamos ter superado completamente essa fase, precisamos incluir uma versão mais atual de um arquivo e descobrimos erros que não havia na versão anterior, ou, pior, passaram despercebidos! Em conseqüência do grande desgaste que essa fase representa, o que se faz na prática é evitar ao máximo ter que voltar a ela. Para isso os analistas geralmente procuram incluir na base de dados gerada toda informação sobre a qual se tenha alguma suspeita de que venha a ser útil. Conforme mencionado na seção 2.1, a existência de um *data warehouse* pode facilitar sobremaneira a presente fase.

2.4 Limpeza e Exploração dos Dados

Com os dados relevantes disponíveis, recomenda-se um exercício de aproximação com os mesmos, pois é necessário que o analista esteja familiarizado com a base de dados. Primeiro para saber o significado de cada atributo, além do nome, e, segundo, porque, em decorrência do alto risco de ocorrência de erro na fase anterior, é imprescindível que o analista realize alguns testes de sanidade dos dados, como forma de explorá-los. Diversos testes de sanidade podem ser realizados, dependendo do aspecto que se quer analisar. Um possível teste, citado por John [JOH 97], é a análise de relações entre atributos. Esse teste consiste em analisar os dados, pensando em como eles deveriam se comportar, e verificando se o comportamento esperado ocorre. O analista poderá observar que o comportamento ocorre para a maioria dos casos, mas falha para alguns. O esforço para explicar os casos em que o padrão falha poderá resultar na identificação de problemas nos dados, cuja eliminação promoverá uma melhoria de qualidade dos mesmos. São mencionados alguns exemplos:

- Descobriu-se, nos dados de uma empresa comercial, um consumidor que gastou vários milhões de dólares em picles, num certo intervalo de tempo. Após a análise dos dados, foi verificado que havia um registro que continha o total de vendas no período e que era identificado por um valor especial, digamos 9999, no campo de código do consumidor. Esse registro foi eliminado da base de dados.
- Na análise dos dados de assinantes de um jornal, foi observado que, quando o valor do campo “resposta a pesquisa” era zero, mais de 90% das vezes os clientes haviam se desligado e nenhum dos que haviam se desligado tinha valor diferente de zero nesse campo. Como a empresa havia aplicado a pesquisa após muitos dos clientes já a terem deixado, estes não tiveram oportunidade de responder ao questionário. Assim, o valor zero para “resposta a pesquisa” significa que o atributo era desconhecido. Um registro como esse na base para mineração poderia implicar em resultados anômalos.

Uma outra ação que pode resultar na identificação de problemas nos dados é a execução dos algoritmos de mineração sobre esses dados ainda não totalmente corretos. A observação de resultados inacreditavelmente bons, ou drásticamente fora de algum limite esperado, deve ser explicada, podendo confirmar descobertas importantes ou dados incorretos.

Na maioria dos casos relatados por John [JOH 97] as bases coletadas para análise têm apresentado algum tipo de erro. Alguns problemas observados são: atributos com nomes errados; ausência de atributos obrigatórios; campos textuais sem codificação e, em vários casos, com valores de atributos iguais escritos de formas diferentes (*e.g.*, *conta paga* e *fatura liquidada*); e atributos relativos a escalas, como

tempo ou valor, não ajustados (e.g., PATRIMÔNIO expresso em milhares ou em unidades monetárias). Algumas transformações precisam ser feitas antes que esses dados possam ser introduzidos em um algoritmo de mineração. Por exemplo, a criação de uma tabela padronizando os atributos textuais ou o ajuste das escalas, conforme o objetivo da mineração seja de obter resultados de maior ou menor granularidade.

2.5 Engenharia dos Dados

Como resultado das fases anteriores, tem-se disponível uma base de dados estática para mineração, contendo todas as informações que se possam necessitar durante o processo de mineração. Entretanto, como essa é uma base geral e pode-se estar interessado em obter relações sobre pontos bem específicos, pode ser mais vantajoso trabalhar com subconjuntos da mesma, tanto em termos de atributos como em termos de tuplas. A obtenção de amostras significativas de dados deve ser feita com a utilização de técnicas de estatística. A escolha dos atributos relevantes para uma determinada tarefa é uma questão extremamente subjetiva. Uma abordagem para auxiliar esse processo, aplicando a técnica do envoltório [KOH 95], foi proposta por John [JOH 97] e consiste na construção de diversos modelos com base em diferentes conjuntos de atributos. Com base no modelo que obtiver melhor acurácia contra dados que não participaram do treinamento, o analista e o especialista podem escolher os atributos com melhor poder preditivo. O método é não-determinístico, já que, para contemplar todas as possíveis combinações de atributos, teria que testar 2^n casos. Assim, o autor utiliza uma busca no espaço de estados das combinações segundo uma estratégia de Inteligência Artificial do tipo o-melhor-primeiro.

Muitas bases de dados podem ser criadas no processo de engenharia dos dados, em contraste com a base de mineração única criada a partir da fase de coleta e extração de dados. Isso ocorre pela necessidade de se explorar o uso de diferentes atributos, diferentes tamanhos de amostra e diferentes enfoques na definição do problema. Na verdade, estamos testando diversas hipóteses de trabalho, buscando a que nos ofereça a melhor solução. Por exemplo, o comportamento dos clientes do banco pode estar condicionado pelo fato de o cliente ser oriundo de uma ou outra região. Se não soubermos disso *a priori*, com um exercício sobre os dados, poderemos notar essa condição e passar a fazer a estratificação da amostra por região. Outro exemplo é o caso do algoritmo de mineração não aceitar dados ausentes. Nesse caso existem métodos apropriados para o preenchimento desses dados com base em técnicas como a apresentada por Quinlan [QUI 93], por exemplo. A discretização de atributos contínuos constitui outro exemplo de transformação que pode ser aplicada sobre os atributos nesse processo. Exemplificando, vamos considerar o atributo IDADE. Poderíamos estar interessados em tratar apenas faixas de idade, como *jovem*, *adulto* ou *idoso*, como forma de reduzir o espaço de estados do problema.

2.6 Engenharia do Algoritmo

A etapa de engenharia do algoritmo corresponde à escolha dos parâmetros mais adequados para a aplicação em pauta. Usualmente isso é feito mediante um processo exaustivo e trabalhoso de tentativas. Alguns exemplos de parâmetros são: (a) o número de camadas em uma rede neural; (b) o número de neurônios na camada escondida de uma rede neural e (c) o parâmetro de vigilância em uma rede do tipo ART [CAR 88].

Um exemplo de solução de engenharia do algoritmo é o método do envoltório usado para se definir o ponto de poda do MNC, apresentado na seção 4.2 desta tese.

2.7 Mineração de Dados

De maneira geral, existem dois possíveis objetivos em um processo de mineração: predição ou descrição [FAY 96]. Esses dois objetivos têm um mapeamento direto nos dois tipos básicos de algoritmos em aprendizado de máquina: supervisionado e não-supervisionado, respectivamente.

A predição visa estabelecer o valor de um ou mais atributos em um banco de dados, a partir de outros atributos presentes, como no caso do crédito bancário. A abordagem preditiva não implica necessariamente a predição de um valor futuro - "a característica importante é que ela faz uma adivinhação educada sobre o valor de um ou mais atributos desconhecidos, dados os valores de outros atributos conhecidos" [JOH 97]. Por outro lado, a descrição visa apontar padrões potencialmente interessantes nos dados sem uma associação com um conceito *a priori*. O objetivo da mineração é fortemente condicionado à fase de definição e entendimento do problema.

Além de evidenciar padrões em bancos de dados, o processo de mineração deve incluir uma forma de medir o quão bom é o padrão encontrado, a chamada acurácia do modelo. No caso da predição essa é uma tarefa de fácil resolução, bastando para isso verificar o número de acertos em um total de casos testados. Já no caso da descrição, isso não é tão trivial, uma vez que se objetiva oferecer uma idéia sobre a distribuição dos dados ao especialista, que poderá escolher aquela que julgar mais interessante.

Apresentamos a seguir cinco tipos de padrões que podem ser obtidos em um processo de mineração: os três primeiros são preditivos e os outros dois, descritivos:

- Regressão: mapeamento de um ou mais itens de dados em uma variável preditiva real. Exemplo: dadas as características de um cliente, dizer qual é a expectativa de lucro com base no nível de renda. Uma técnica de regressão bastante utilizada é a regressão linear, que consiste no ajuste de uma reta sobre um conjunto de pontos, visando minimizar a soma dos erros quadrados (FIGURA 2.2).

- Classificação linear: mapeamento de um ou mais itens de dados em uma classe de um dado conjunto, particionando o espaço de estados por meio de funções lineares. Exemplo: dada a ficha de um cliente, determinar se ele pode ou não tomar dinheiro emprestado, segundo os atributos DÉBITO ATUAL e RENDA (FIGURA 2.3). Uma técnica utilizada para a obtenção desse tipo de padrão são as árvores de decisão [QUI 93]. A técnica consiste em construir uma árvore onde cada nodo não-folha represente um atributo, os arcos expressem assertivas sobre os atributos, e as folhas representem as classes associadas a um determinado percurso na árvore (FIGURA 2.4).

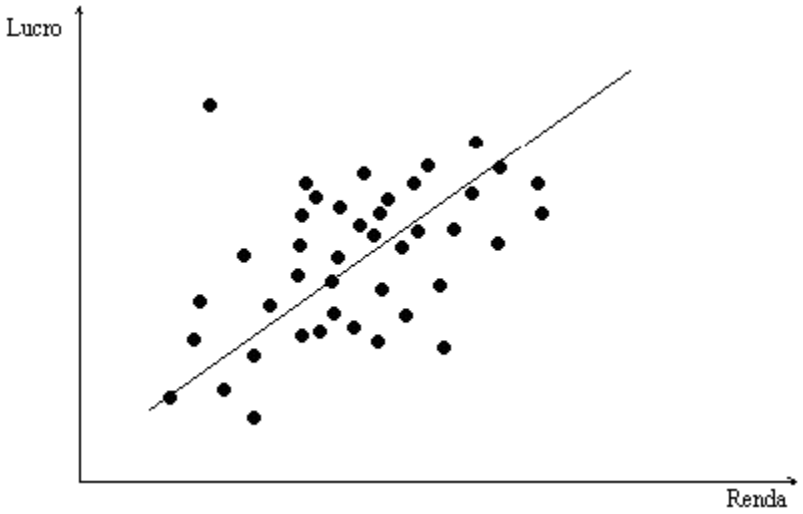

FIGURA 2.2 - Ajuste de Modelo por Regressão Linear

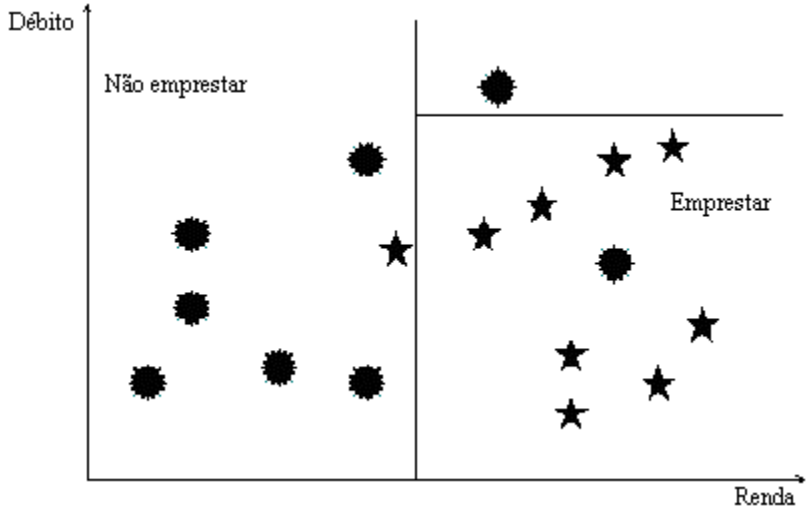

FIGURA 2.3 - Particionamento do Espaço por Classificação Linear

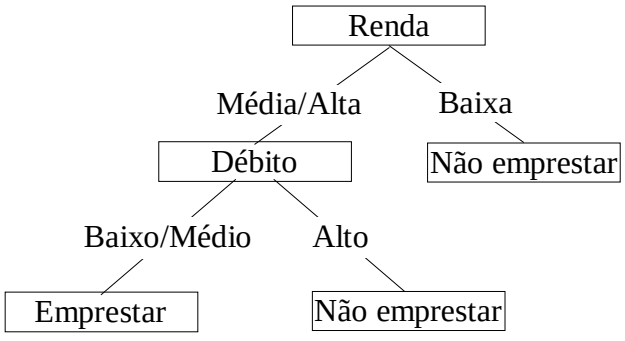

FIGURA 2.4 - Árvore de Decisão para a Concessão de Empréstimos

- Classificação não-linear: visa ajustar uma combinação não-linear de funções a um conjunto de variáveis de entrada. Redes neurais alimentadas para frente [FAY 96] são uma alternativa para esse tipo de classificação. As redes neurais, por exemplo, podem ajustar mais adequadamente funções de classificação do que o simples estabelecimento de um limite. Com base no exemplo da FIGURA 2.3, uma rede

neural do tipo *backpropagation* poderia separar os conjuntos de clientes segundo a FIGURA 2.5.

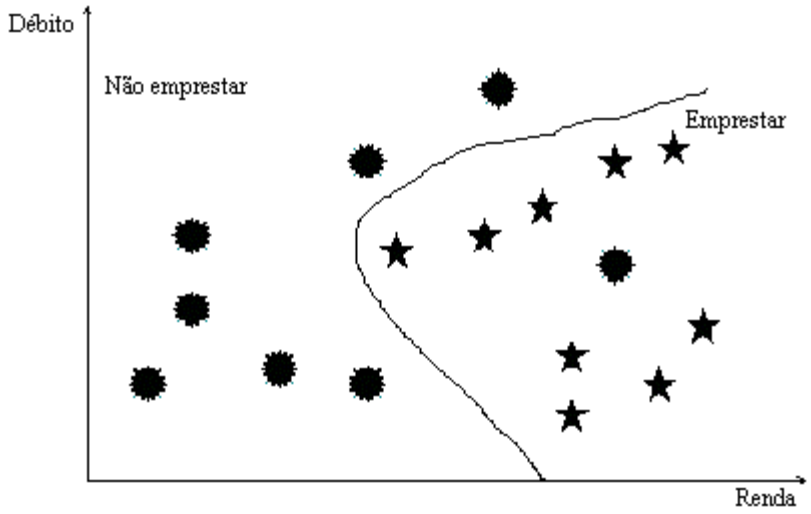

FIGURA 2.5 - Particionamento Não-Linear do Espaço

Apesar de essa técnica conseguir uma separação melhor do conjunto, ela é mais difícil de ser interpretada do que a classificação linear. No nosso exemplo, o resultado da classificação linear poderia ser expresso numa regra simples do tipo "se a renda for maior que um certo limite então o cliente pode receber um empréstimo", o que não acontece na classificação não-linear. No caso da classificação não-linear a interpretação não é trivial.

- Identificação de agrupamentos (*clustering*): consiste em identificar um conjunto finito de categorias presentes nos dados. Exemplo: descrever subgrupos entre os poupadores de um banco (FIGURA 2.6). Note que foram utilizados três atributos para se definirem os agrupamentos: *idade, renda e sexo*. Para facilitar a representação, o atributo *sexo* foi colocado na posição em que cada ponto é definido pelos outros dois atributos.

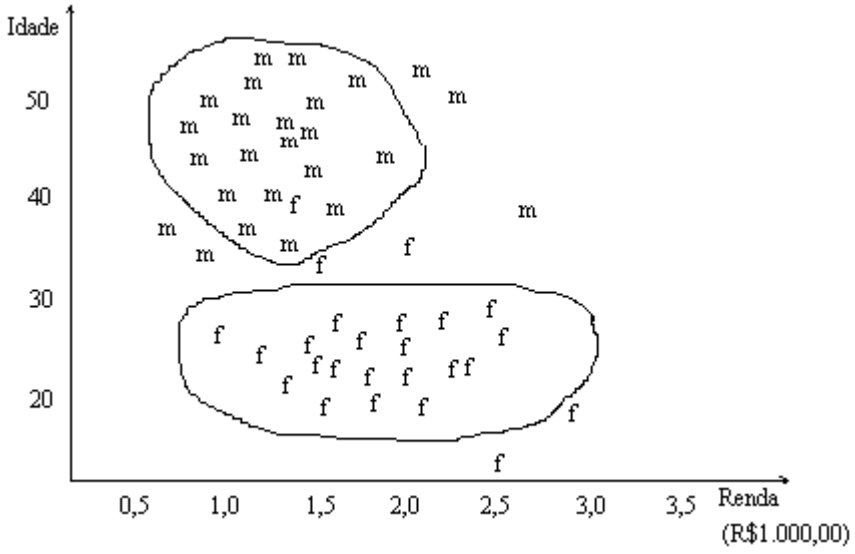

FIGURA 2.6 - Identificação de Agrupamentos

- Regras associativas: são regras do tipo $X \Rightarrow Y$, onde X e Y são conjuntos de itens. O significado dessas regras é que as transações do banco de dados que contêm X tendem a conter Y também. Exemplo: 90% dos clientes que têm cartão de crédito fazem seguro do seu carro. Essa técnica foi apresentada por Agrawal [AGR 96].

2.8 Interpretação e Validação dos Resultados

Muitas vezes um padrão encontrado em uma base de dados pode não ser válido para outros casos que não constem da base utilizada para definir o padrão. É comum acontecer o chamado superajustamento do modelo aos dados: o modelo encontrado está particularizado para a base de treinamento, não possuindo o mesmo desempenho para dados que não participaram do treinamento.

Para que o resultado de uma mineração possa ser utilizado com segurança fora do contexto em que foi gerado, algumas precauções devem ser tomadas. Primeiramente, é necessário definir uma métrica para a quantidade de acertos do modelo, a sua acurácia, e isso depende do tipo de mineração realizada. Por exemplo, para classificação pode-se utilizar "o percentual de objetos classificados corretamente pelas hipóteses" [LAV 94]. Por outro lado, para regressão, uma medida adequada é a variância dos resultados do padrão com relação a uma dada amostra.

Para ilustrar o conceito de acurácia, incluímos a seguir um resumo dos conceitos relacionados ao cálculo da variância [NET 77] como medida de acurácia. Definimos, inicialmente, parâmetro, estimador e estimativa, para, em seguida, definirmos acurácia:

1. Parâmetro: é uma função de uma característica da população, a qual, uma vez calculada, se converte em um número real, constante para a população. São exemplos de parâmetros: a média de público nos jogos da NBA em 1996, a densidade de certa espécie de planta em um bosque, a quantidade total de pontos em arremessos livres no campeonato e a variabilidade no tamanho dos jogadores de um time.

2. Estimador: é uma função da amostra que se pode utilizar para obter valores aproximados de um parâmetro. Considerando a amostra como um conjunto de variáveis aleatórias, pode-se concluir que um estimador é uma função de variáveis aleatórias, e, portanto, é uma variável aleatória.

3. Estimativa: é definida para um parâmetro, significando cada valor aproximado que assume um estimador.

4. Acurácia: medida do grau de acerto das estimativas com relação ao parâmetro estimado. Para estimadores não tendenciosos, a acurácia apresenta uma relação inversa com a variância das estimativas.

Desta forma, considerando um estimador hipotético $y = b.x$ (por exemplo, $taxa\_de\_retorno = 0{,}8 \times faturamento$; isto é, $taxa\_de\_retorno$ é dada por uma relação linear com $faturamento$), temos N observações de x e y:

$$x_1, x_2, x_3, ..., x_i, ..., x_n$$

$$y_1, y_2, y_3, ..., y_i, ..., y_n$$

Por uma regressão linear consegue-se encontrar a seguinte correlação $y_i' = b'.x_i$. Considerando y_i' a estimativa de y no ponto i, a variância do modelo é dada por:

$$s^2 = \frac{\sum_{i=1}^{N} (y_i^{'} - y_i)^2}{(N-1)}$$

Uma vez estabelecida a forma de se medir a acurácia, é necessário definir as amostras sobre as quais o erro deve ser medido. Em geral, a abordagem é a aplicação de uma validação cruzada. “Validação cruzada é um mecanismo que usa uma dada amostra para gerar hipóteses e estimar sua validade e acurácia na população. A amostra é dividida repetidamente e de forma aleatória em subconjuntos disjuntos de treinamento e teste. As hipóteses obtidas são verificadas contra os dados de teste.” [KLÖ 96].

John [JOH 97] aplica um refinamento desse método para o caso em que se quer comparar algoritmos diferentes para desempenhar o mesmo tipo de mineração. Por exemplo, podemos estar interessados em comparar a acurácia de dois algoritmos de classificação. Segundo o refinamento proposto, após a obtenção do nível de acurácia dos resultados de cada algoritmo, toma-se uma amostra que não tenha participado nos testes anteriores, aplica-se essa amostra sobre os dois algoritmos, e faz-se o cálculo da acurácia segundo a métrica adotada.

Além de uma avaliação objetiva, baseada em uma métrica precisa, é necessária uma avaliação qualitativa, por parte do especialista, dos resultados encontrados. É possível que, mesmo com níveis de acurácia aceitáveis, o especialista não se sinta convencido dos resultados e, nesse caso, nenhum resultado prático terá advindo da mineração. É preciso que os resultados provoquem o especialista no sentido de levá-lo a buscar um entendimento do que eles significam. Segundo John [JOH 97], quando um especialista interpreta os resultados de uma mineração, pode acontecer uma das seguintes situações:

- especialista fica satisfeito com os resultados, mas percebe que já conhecia os padrões obtidos;
- especialista fica satisfeito com os resultados e surpreso com alguns dos padrões obtidos;
- especialista fica insatisfeito com os resultados.

No primeiro caso, quando os resultados já eram do conhecimento do especialista, o modelo ainda pode ser usado como base para a construção de outro que forneça resultados melhores. Nos outros casos, é preciso uma análise mais aprofundada dos resultados e o processo deve ser revisto.

De fato, em conseqüência dos resultados dessa fase, o analista pode concluir pela necessidade de um retorno a qualquer uma das fases anteriores.

2.9 Refinamento dos Dados e do Problema

É comum que, ao final de todo o processo de DCBD, o resultado não atenda às expectativas do especialista. Esse é o ponto de partida para uma iteração intensa no processo de mineração. A abordagem sugerida é, primeiramente, realizar uma revisão rápida do processo completo, possivelmente pulando ou resumindo as fases de preparação e exploração, mostrando os resultados dessa revisão ao especialista. A comunicação que se estabelece a seguir, entre o analista e o especialista, desencadeia uma reflexão sobre o processo de DCBD, que pode mostrar a fragilidade em alguma das fases executadas.

O seguinte exemplo [JOH 97] ilustra essa situação. Um banco europeu solicitou ajuda ao grupo de mineração da IBM para aumentar a taxa de resposta afirmativa a um convite enviado por mala direta, oferecendo uma linha de crédito a juros acessíveis. Com o objetivo de “aumentar a taxa de resposta afirmativa”, o grupo construiu um modelo para ordenar os clientes segundo a sua propensão a tomar empréstimo. Verificou-se que o grupo com maior probabilidade de aceitar a oferta era exatamente aquele composto pelos clientes que, costumeiramente, apresentavam balanços negativos com o banco. A prática do banco, em caso de receber um cheque sem fundos, é acatar o cheque, deixar o saldo do cliente negativo, e cobrar uma taxa de juros mais alta sobre esse saldo.

Uma decisão tomada, cegamente, com base nesses resultados exporia o banco a riscos exagerados, uma vez que poderiam ser concedidos empréstimos a clientes que, possivelmente, já apresentavam dificuldades financeiras. Essa situação mostra como uma definição e entendimento precários do problema podem levar a resultados desastrosos.

Em um outro caso, o problema pode estar muito bem definido e entendido, mas o especialista não está satisfeito com os resultados. Talvez a saída seja acrescentar novos atributos, causando um retorno à fase de coleta e extração dos dados ou de engenharia dos dados, caso a base para mineração já contenha o atributo desejado.

Após a fase de refinamento dos dados e do problema, estando o especialista convencido de que os padrões obtidos são válidos, ele pode partir para a utilização dos resultados.

3 Orpheo - Uma Estrutura de Trabalho para Integração dos Paradigmas de Aprendizado Supervisionado e Não-Supervisionado

A descoberta de conhecimento a partir de dados não-classificados compreende duas tarefas principais: identificação de "agrupamentos naturais" e análise desses agrupamentos de modo a facilitar a interpretação do seu significado. Neste capítulo assumimos que os avanços já obtidos no domínio da DCBD, especificamente no que se refere aos algoritmos de aprendizado supervisionado e não-supervisionado, nos permitem combinar esses dois processos em apenas um fluxo, provendo descrições extensionais e intensionais de dados não-classificados. Por descrições extensionais entendemos a enumeração dos membros de cada agrupamento, e por descrições intensionais, a obtenção dos valores de atributos mais correlacionados com o agrupamento. Para explorar essa idéia propomos uma estrutura de trabalho para integrar diferentes algoritmos de aprendizado supervisionado e não-supervisionado na consecução da tarefa de descoberta de conhecimento de dados não-classificados.

3.1 Motivação

Segundo Langley [LAN 98], "talvez, a primeira atividade de descoberta envolve a formação de *taxonomias* [...] que agrupam entidades como categorias e subcategorias, de forma hierárquica." Após essa tarefa, os cientistas necessitam descobrir leis qualitativas que caracterizem o comportamento daquelas taxonomias ou as relacionem entre si. O processo de formação de conceitos, de acordo com Easterlin e Langley [EAS 85], parte de um conjunto de descrições de objetos apresentado, usualmente, de forma incremental e é levado a cabo conforme os seguintes passos: (1) encontrar conjuntos de objetos que podem ser agrupados (fase de agregação) e (2) encontrar descrições intensionais desses conjuntos de objetos (fase de caracterização). Nesse mesmo sentido, Wittgeinstein (*op. cit in* [HAN 90] p. 238) advoga que os conceitos possuem uma natureza politética. Isto significa que um conceito pode ser descrito de várias maneiras, compartilhando um conjunto de características comuns, sem que qualquer delas seja essencial para a pertinência ao agrupamento em questão.

Murphy e Medin [MUR 85] discutem duas hipóteses que restringem a forma como objetos são agrupados em conceitos: (1) aplicação de algoritmos de aprendizado não-supervisionado para identificar agrupamentos e (2) aplicação de um algoritmo de aprendizado supervisionado para encontrar as estruturas de correlação dentro dos agrupamentos. Considerando os avanços em DCBD, acreditamos que haja condições de se viabilizar a integração dos dois processos em um único. De fato, a tarefa é uma só: realizar aprendizado a partir de dados não-classificados. Acreditamos que a separação em duas tarefas ocorreu como forma de viabilizar a solução do problema. Como alternativa ao processo tradicional, propomos Orpheo, uma estrutura de trabalho que integra diferentes algoritmos para tornar a solução daquela tarefa mais eficiente. Por meio dessa estrutura de trabalho, ao buscar a descoberta de conhecimento a partir de dados não-classificados, um especialista pode encontrar, semi-automaticamente, tanto os agrupamentos quanto as suas explicações com base no subespaço das dimensões consideradas.

3.2 Descrição da Estrutura de Trabalho

Conforme mostrado na FIGURA 3.1, Orpheo consiste de três camadas: de interface, de lógica e de persistência. Os nomes das classes pertencentes a cada camada são identificados, respectivamente, com as iniciais CI, CL e CP. A organização da estrutura em camadas visa garantir maior flexibilidade para evolução, ao mesmo tempo em que garante independência funcional.

A camada de interface gerencia toda a comunicação entre o analista e a camada de lógica. Por intermédio dessa camada, o analista informa suas opções com relação à amostragem, discretização e aprendizado supervisionado e não-supervisionado, além de especificar o arquivo de treinamento e os parâmetros requeridos pelos algoritmos de aprendizado escolhidos. As classes de interface para captação de parâmetros são específicas de cada algoritmo utilizado, não sendo, portanto, mostradas nessa arquitetura. Os métodos que começam com a palavra "escolhe" incluem estruturas *case* que controlam o fluxo da camada de lógica, de acordo com as escolhas do usuário.

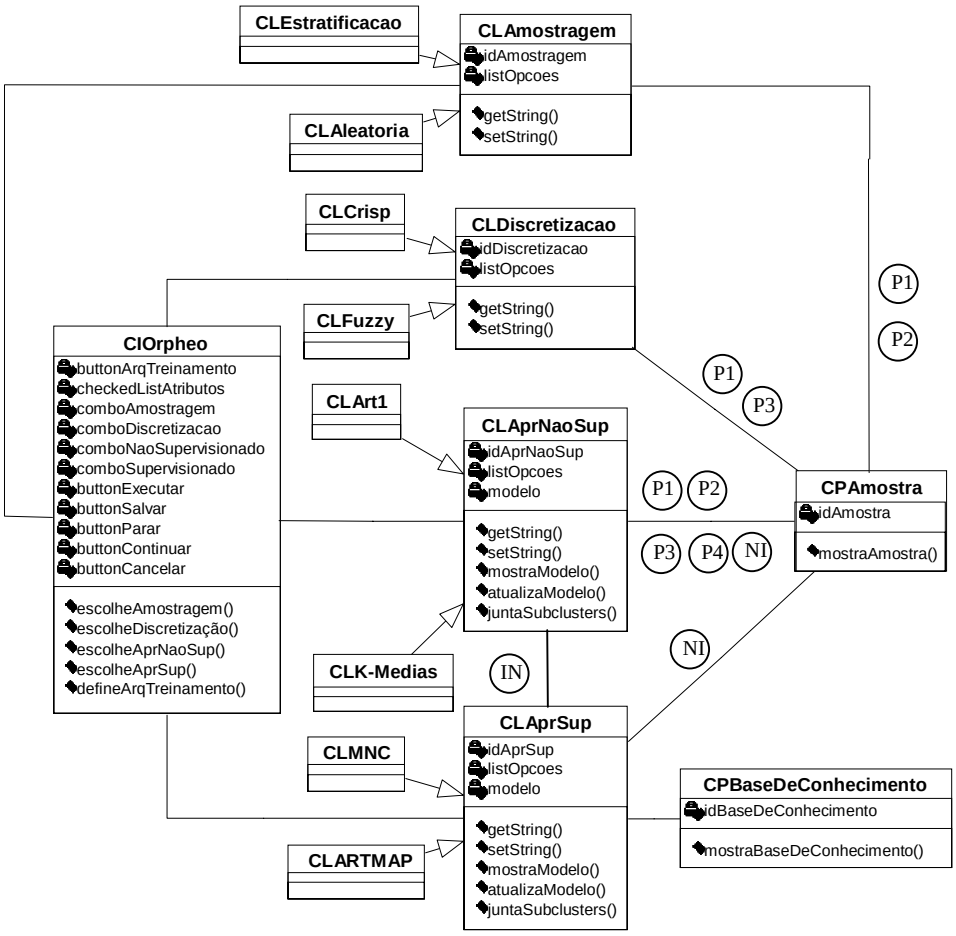

FIGURA 3.1 - Arquitetura da Estrutura de Trabalho Orpheo

A camada de lógica reúne as classes abstratas que abrigam as opções de algoritmos para execução segundo as escolhas do analista, juntamente com a lista de opções existentes. Todas as classes na camada de lógica possuem serviços de acesso aos dados. Além disso, as classes de aprendizado possuem serviços para atualização e exibição dos modelos nelas construídos. Um serviço para união de agrupamentos é especificado na classe de aprendizado supervisionado para atender à necessidade trazida

pela técnica para expandir a capacidade dos algoritmos de aprendizado não-supervisionado, proposta por Bradley *et al.* [BRA 98]. Essa técnica trabalha com o conceito de subagrupamentos que, eventualmente, podem se juntar em um agrupamento. As classes na camada de persistência provêem serviços para o armazenamento de dados e de conhecimento, e têm um método padrão para exibição de suas estruturas. Note-se que a classe **CPAmostra** representa um subconjunto próprio do arquivo de treinamento. O fluxo entre as camadas de lógica e de persistência é também governado pelas escolhas do analista.

Existem dois tipos de fluxo de controle na estrutura, um relacionado a pré-processamento e outro que depende do fato de pelo menos um dos algoritmos de aprendizado ser incremental ou não. O primeiro fluxo é identificado com um rótulo iniciando com a letra P e está relacionado às tarefas de amostragem e discretização, conforme as opções do analista. Os fluxos de pré-processamento são mostrados até a classe **CLAprNaoSup**, que engloba os algoritmos de aprendizado não-supervisionado. A TABELA 3.1 mostra os fluxos definidos para cada conjunto de opções.

TABELA 3.1 – Fluxos Relacionados a Pré-Processamento

Opções		Fluxo
Amostragem	Discretização	
Sim	Sim	P1
Sim	Não	P2
Não	Sim	P3
Não	Não	P4

O segundo tipo de fluxo torna-se ativo após a execução do algoritmo da classe **CLAprNaoSup**. Caso um dos dois algoritmos das classes **CLAprNaoSup** e **CLAprSup** escolhidos pelo analista seja não incremental, o fluxo a ser seguido é o identificado como NI que retorna para a classe **CPAmostra**. Essa classe passa agora a guardar os dados rotulados. Após o término do algoritmo da classe **CLAprNaoSup**, o fluxo é direcionado para a classe **CLAprSup**. Caso os dois algoritmos das classes **CLAprNaoSup** e **CLAprSup** sejam incrementais, o fluxo é rotulado como IN e segue direto da classe **CLAprNaoSup** para **CLAprSup.**

A idéia nessa estrutura de trabalho é prover flexibilidade para suportar qualquer escolha do analista quanto a que algoritmo aplicar para desempenhar cada função.

3.3 Instanciação da Estrutura de Trabalho

Para avaliação prática da estrutura de trabalho, realizamos a implementação de um protótipo simulando a sua instanciação com duas conhecidas redes neurais artificiais (RNAs): a rede ART1 (FIGURA 3.2), da família de RNAs da Teoria da Ressonância Adaptativa (ART) [CAR 88], e o Modelo Neural Combinatório – MNC (FIGURA 3.4), proposto por Machado ([MAC 89]). Ambos os modelos atendem a um importante requisito para Mineração de Dados [BRA 98], aprendendo em apenas uma passada da base de dados, que pode ser interrompida a qualquer momento, exibindo o modelo gerado até o ponto de parada. Além do mais, o MNC, que corresponde à parte

intensional da instanciação, permite a obtenção de regras simbólicas diretamente da sua representação.

3.3.1 ART1

ART1 é uma rede neural competitiva com duas camadas: uma de entrada e outra de agrupamento. Essa rede possui duas limitações importantes: a primeira é que a camada de entrada aceita apenas dados binários, e a segunda é que a configuração final dos agrupamentos depende da ordem de entrada dos dados. Apesar disso, a sua adoção nesta tese cumpre bem o papel de exemplificação do uso da estrutura proposta. Essa rede foi desenvolvida para superar o dilema estabilidade-plasticidade [CAR 88], permitindo aprendizado incremental, com a contínua atualização dos protótipos dos agrupamentos, ao mesmo tempo em que preserva os padrões previamente armazenados. O algoritmo para identificação dos agrupamentos (ver FIGURA 3.3) funciona, de maneira geral, da seguinte forma: (a) a primeira entrada é atribuída para o primeiro protótipo; (b) cada próxima entrada é comparada com os protótipos existentes; o protótipo mais próximo, no qual a distância para a entrada é menor do que um certo limiar, é escolhido para representar a entrada. Caso não exista um protótipo representativo, um novo protótipo é criado como cópia da entrada. Pode-se observar que a quantidade de protótipos depende do limiar e da métrica usada para comparar a entrada com os protótipos. Para cada novo padrão apresentado, um protótipo é declarado vencedor. O vencedor retropropaga um sinal que codifica o padrão que ele representa. Se o padrão corrente difere do sinal mais do que um limiar predefinido, o vencedor é, temporariamente, desabilitado (por um sistema de orientação) e o protótipo seguinte mais próximo é declarado vencedor.

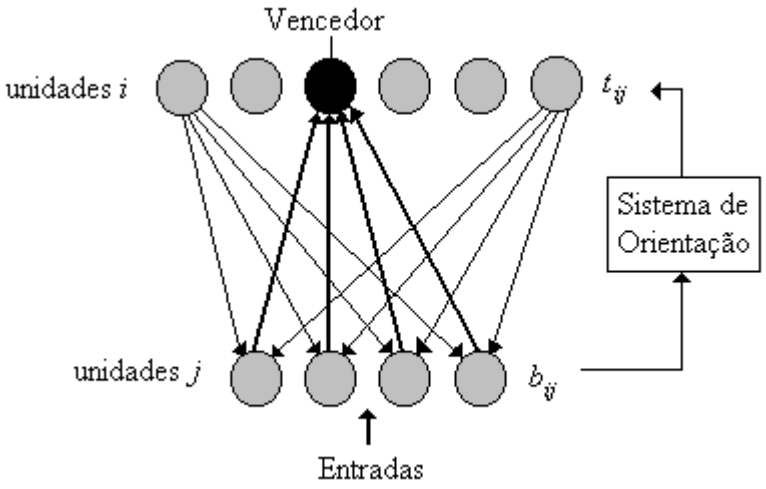

FIGURA 3.2 - Arquitetura de uma Rede ART1

O processo continua até que um protótipo que atenda o requisito de proximidade seja declarado vencedor. Se nenhum tornar-se vencedor, um novo protótipo é criado e definido como representante do padrão de entrada. A rede ART1, ilustrada pela FIGURA 3.2, possui quatro neurônios de entrada e seis de saída. As conexões de cima para baixo e de baixo para cima são representadas por t_{ij} e b_{ij}, respectivamente.

TREINAMENTO_DA_REDE_ART1

- **Passo 1:** *Inicializar N* e *M* como o número de unidades de entrada e de saída, respectivamente. Os pesos das conexões $t_{ij}(t)$ e $b_{ij}(t)$ entre a unidade de entrada *i* e a unidade de saída *j* são fixados segundo uma heurística padrão. O parâmetro de vigilância ρ é definido, indicando quão próxima uma entrada deve estar de um protótipo para ser classificada.
- **Passo 2: Para** cada entrada, **faça:**
  - **Passo 2.1:** *Computar* os valores da entrada: os pesos de baixo para cima são aplicados ao padrão de entrada, gerando o sinal de saída μ_j.
  - **Passo 2.2:** *Selecionar* o melhor exemplar da saída, identificado pelo índice j^*: $\mu_{j^*} = \max\{\mu_j\}$ é tomado como o melhor exemplar.
  - **Passo 2.3:** *Efetuar* teste de vigilância: o melhor exemplar e o padrão de entrada são comparados de acordo com ρ. Se a distância é aceitável, o controle flui para o **Passo 2.5**, caso contrário, seguir para o **Passo 2.4**.
  - **Passo 2.4:** *Desabilitar* o melhor exemplar: a saída do melhor exemplar selecionado no **Passo 2.2** é temporáriamente zerada e não mais toma parte na maximização daquele passo. Ir para o **Passo 2.2**.
  - **Passo 2.5:** *Adaptar* o melhor exemplar:

$$t_{ij^*}(t+1) = t_{ij^*}(t)x_i$$

$$b_{ij^*}(t+1) = \frac{t_{ij^*}(t)x_i}{\frac{1}{2} + \sum_{i=0}^{N-1} t_{ij^*}(t)x_i}$$

  - **Passo 2.6:** *Reabilitar* qualquer unidade que esteja desabilitada.

FIGURA 3.3 – Algoritmo de Treinamento para a Rede ART1 (adaptado de [LIP 87])

3.3.2 O Modelo Neural Combinatório

O Modelo Neural Combinatório (MNC) [MAC 89] constitui-se de uma arquitetura híbrida para sistemas inteligentes que integra os paradigmas simbólico e conexionista. Entre as suas principais vantagens citamos: facilidade para aquisição de conhecimento a partir de especialistas na forma de uma rede neural; aprendizado incremental baseado em exemplos, apresentando-se como uma alternativa para se enfrentar o dilema plasticidade-estabilidade [FRE 92]; facilidade para integração de diversas fontes de conhecimento; extração de conhecimento implícito no modelo; habilidade para a identificação de regularidades em espaços multidimensionais, realizando mapeamentos em espaços de saída de menores dimensões; e capacidade para tratamento de incerteza.

O MNC utiliza aprendizado supervisionado em uma topologia alimentada para frente com as seguintes características: uma camada de entrada, uma camada escondida – aqui chamada combinatória – e uma camada de saída (FIGURA 3.4). Cada neurônio representa um conceito simbólico do domínio e computa um valor entre zero (falso) e um (verdadeiro), chamado estado de ativação do neurônio, a partir das informações dos arcos de entrada. O valor dos arcos chegando a cada neurônio é chamado fluxo evidencial. Por exemplo, um arco de entrada em um neurônio de um modelo para identificar regularidades no mercado financeiro poderia corresponder à evidência "há uma forte pressão sobre o real", com um fluxo evidencial no valor de 0,8. Na camada combinatória existem neurônios agregadores conectados a um ou mais neurônios da camada de entrada por arcos E *fuzzy*. A camada de saída contém um neurônio para cada possível classe (também chamada hipótese), ligada a um ou mais neurônios da camada combinatória por arcos OU *fuzzy*.

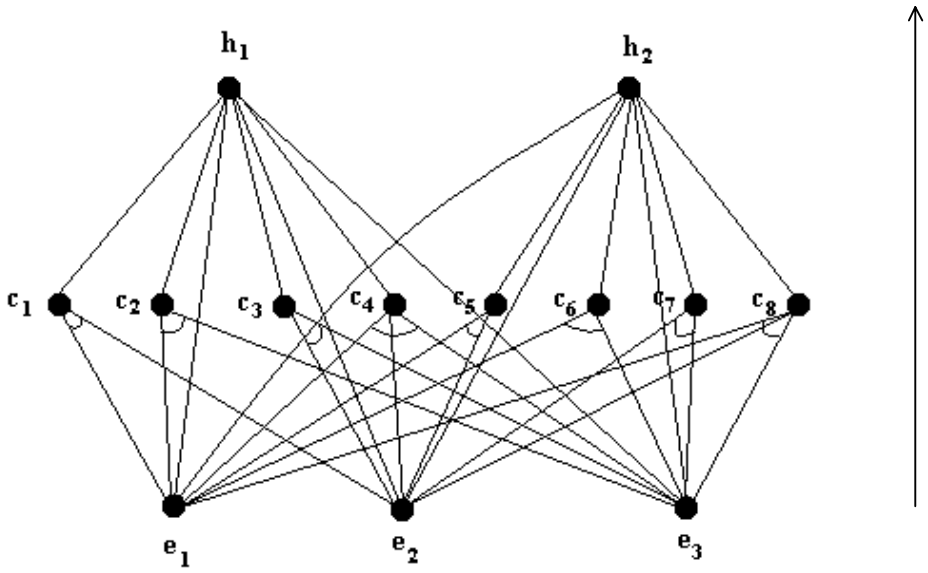

FIGURA 3.4 - Versão Completa do MNC com Três Evidências e Duas Hipóteses

Os neurônios são conectados por meio de arcos, chamados sinapses, caracterizados por um peso que expressa a força da ligação. Esses pesos são calculados com base nos fluxos evidenciais provenientes da camada de entrada. Sinapses inibitórias podem ser incluídas de modo a efetivar a negação de um sinal.

A rede é criada completamente vazia, da seguinte forma: (a) um neurônio na camada de entrada para cada evidência no conjunto de treinamento, descrita por uma dupla atributo-valor; (b) um neurônio na camada de saída para cada classe no conjunto de treinamento e (c) para cada neurônio na camada de saída, existe um conjunto de neurônios na camada combinatória, correspondendo a todas as possíveis combinações de conexões com a camada de entrada. Para efeito de otimização da rede, as combinações com apenas um neurônio são ligadas diretamente às hipóteses.

O mecanismo de aprendizado funciona em uma única iteração, segundo os passos descritos na FIGURA 3.5.

REGRA_DE_APRENDIZADO_POR_PUNIÇÃO_E_PREMIAÇÃO

- *Associe* a cada arco da rede um acumulador com valor inicial zero;
- **Para** cada exemplo da base de treinamento, **faça:**
  - *Propagar* as evidências dos neurônios de entrada até a camada de saída (hipóteses);
  - **Para** cada arco alcançando um neurônio da saída, **faça:**
  - **Se** a hipótese alcançada corresponde à classe correta do exemplo
    Então *retropropagar* deste neurônio até a camada de entrada, *somando* o acumulador de cada arco atravessado com o valor do fluxo evidencial (Premiação)
    Caso contrário *retropropagar* o sinal deste neurônio até a camada de entrada, *subtraindo* do acumulador de cada arco atravessado o valor do fluxo evidencial (Punição);
  - **Fim_para;**
- **Fim_para.**

FIGURA 3.5 - Algoritmo de Aprendizado do MNC

Após o treinamento, o valor dos acumuladores associados a cada arco chegando à camada de saída estará situado no intervalo [-T, T], onde T é o número de casos existentes no conjunto de treinamento.

O próximo passo é a poda da rede, realizada por meio das seguintes ações: (a) remover todos os arcos cujos acumuladores forem menores que um limiar (especificado por um especialista) e (b) remover todos os neurônios das camadas de saída e combinatória que tenham ficado desconectados completamente em relação à camada de saída. Finalmente, antes de a rede se tornar operacional, os acumuladores são normalizados para o intervalo [0, 1], tornando o modelo operacional para a classificação de novos casos. Essa normalização é realizada para todos os arcos que chegam a cada neurônio da camada de saída, da seguinte forma: a cada arco é atribuído um peso que é obtido pela divisão do valor do seu acumulador pelo acumulador de maior valor entre todos os arcos que chegam a esse neurônio.

3.3.3 Modelo Funcional

A estrutura de trabalho, instanciada com as RNAs ART1 e MNC, é mostrada no modelo funcional da FIGURA 3.6. As camadas F1 e F2 correspondem à rede ART1. F1 recebe a codificação da entrada da estrutura, enquanto F2 mantém os protótipos identificados no processo. As camadas F2, F3 e F4 correspondem à rede MNC.

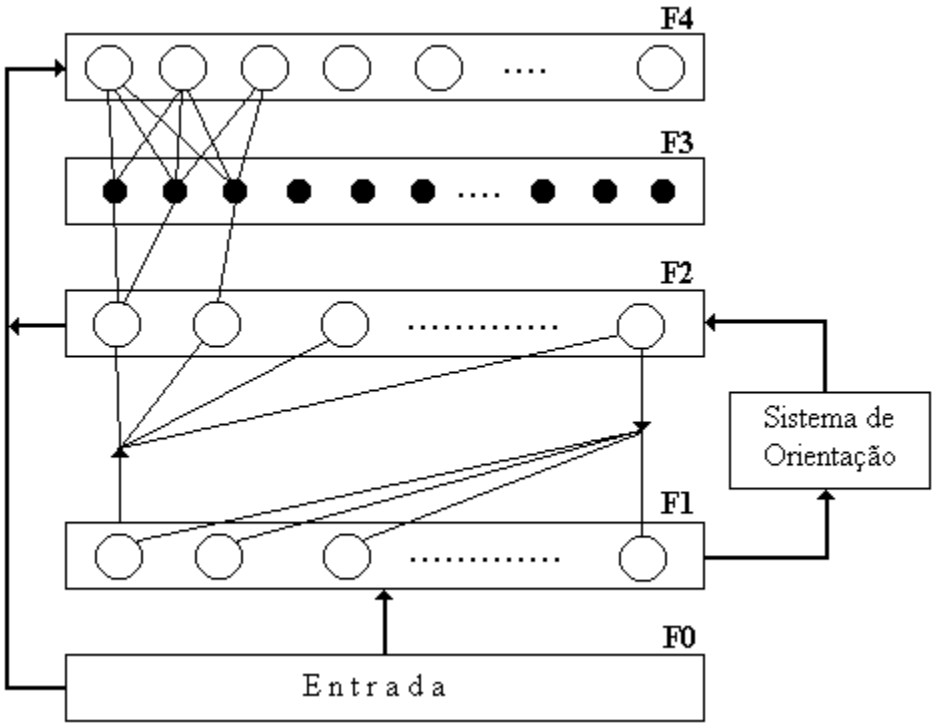

FIGURA 3.6 - Estrutura de Trabalho Instanciada com as Redes ART1 e MNC

Nota-se que a camada F2 representa o ponto de integração entre as duas RNAs. Durante todo o processo, essa camada é expandida com a criação de novos protótipos. Considerando que esse processo ocorre em apenas uma passada dos dados, é interessante observar o que se passa quando uma entrada é associada a um protótipo. Nesse momento, o protótipo (em F2) passa a ser tratado como uma classe associada à entrada, sendo ambos, rótulo e entrada, remetidos à rede MNC, cuja entrada corresponde à camada F4.

3.4 Ilustrando o Aprendizado Integrado

Apresentamos nesta seção um exemplo de aprendizado utilizando a estrutura de trabalho proposta. O objetivo aqui é ilustrar o funcionamento da estrutura por meio de um arquivo de treinamento simples. São apresentados dados não-classificados e geradas descrições extensionais e intensionais. O arquivo contém 15 animais que sabemos pertencer às classes dos anfíbios, répteis, aves e mamíferos, conforme as caracterizações filogenéticas existentes em Biologia [VIL 84]. As classes, entretanto, não são fornecidas no arquivo de treinamento, sendo, antes, algo a ser “descoberto” pelo sistema. Os atributos escolhidos foram metabolismo, cobertura (se possui pena, pêlo, etc), tipo de dentição, tipo de reprodução, se o animal é anuro ou não, número de patas e tipo de alimentação predominante. Os valores dos atributos estão relacionados na TABELA 3.2.

A FIGURA 3.7 mostra uma representação interna parcial da estrutura instanciada com as RNAs ART1 e MNC aplicada aos dados da TABELA 3.2. A codificação da entrada corresponde à camada F0. A camada F4 é mostrada apenas para efeito de ilustração, uma vez que não necessita ser implementada por ser exatamente igual a F0.

TABELA 3.2 – Arquivo de Treinamento Ilustrativo

Nome	*Metabolismo*	*Cobertura*	*Dentição*	*Reprodução*	*Anuro*	*Patas*	*Alimentação*
Rã	Ectotérmico	Epiderme úmida	Superior	Ovípara	Sim	4	Insetívora
Sapo	Ectotérmico	Epiderme úmida	Não possui	Ovípara	Sim	4	Insetívora
Crocodilo	Ectotérmico	Rica em queratina	Completa	Ovípara	Não	4	Carnívora
Tartaruga	Ectotérmico	Rica em queratina	Não possui	Ovípara	Não	4	Herbívora
Elefante	Endotérmico	Pêlos	Completa	Vivípara	Não	4	Herbívora
Cachorro	Endotérmico	Pêlos	Completa	Vivípara	Não	4	Carnívora
Gato	Endotérmico	Pêlos	Completa	Vivípara	Não	4	Carnívora
Coelho	Endotérmico	Pêlos	Completa	Vivípara	Não	4	Herbívora
Onça	Endotérmico	Pêlos	Completa	Vivípara	Não	4	Carnívora
Baleia	Endotérmico	Pêlos	Só as mais novas	Vivípara	Não	0	Carnívora
Águia	Endotérmico	Penas	Não possui	Ovípara	Não	2	Carnívora
Pardal	Endotérmico	Penas	Não possui	Ovípara	Não	2	Hérbivora e insetívora
Gavião	Endotérmico	Penas	Não possui	Ovípara	Não	2	Carnívora
Galinha	Endotérmico	Penas	Não possui	Ovípara	Não	2	Herbívora e insetívora
Pato	Endotérmico	Penas	Não possui	Ovípara	Não	2	Herbívora e insetívora

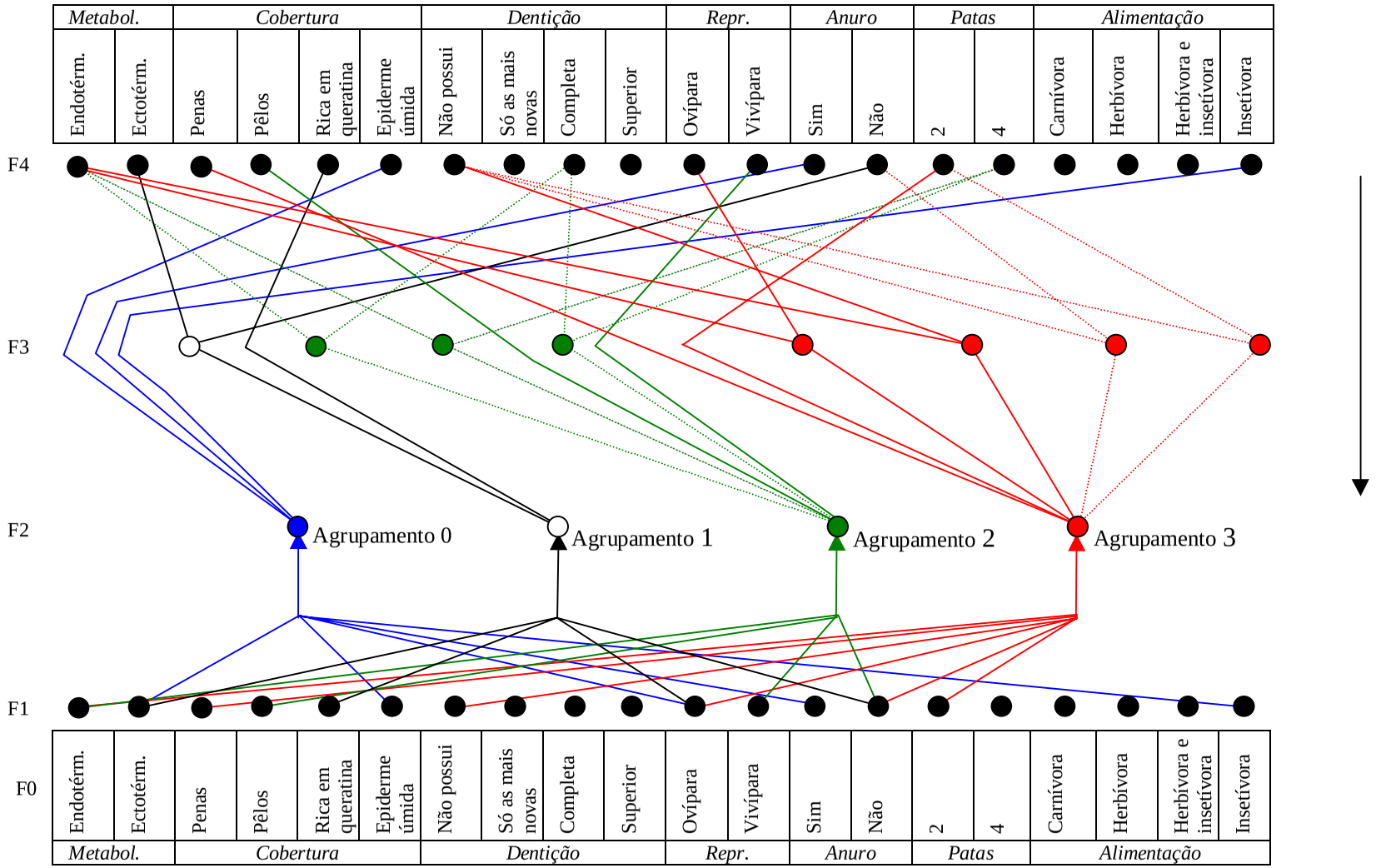

FIGURA 3.7 – Representação Parcial da Estrutura Instanciada com ART1 e MNC para os Dados da TABELA 3.2

O fator de vigilância para a rede ART1 foi arbitrado em 0,5. Nas camadas correspondentes ao MNC – F2, F3 e F4 – foram podados os arcos com nível de confiança e suporte menores que 80%, sendo que os arcos sólidos referem-se às regras com suporte e confiança de 100%. Nível de confiança é aqui definido como a razão entre o número de casos de sucesso da regra e o número total de casos classificados pela mesma. O suporte é dado pela razão entre o número de casos de sucesso da regra e o número de casos pertencentes ao agrupamento referido na mesma. O número máximo de combinações foi limitado a três, como forma de simplificar a apresentação dos resultados. Os protótipos dos agrupamentos encontrados (identificados como Agrupamento 0, 1, 2 e 3, na FIGURA 3.7) são mostrados na TABELA 3.3. Cada protótipo explicita os valores de variáveis compartilhados por todos os membros do agrupamento que ele representa. O valor 1 em uma posição significa o valor de uma variável compartilhado; zero significa o não-compartilhamento. Por exemplo, todos os elementos do agrupamento 3 compartilham os seguintes valores: *Metabolismo*=Endotérmico, *Cobertura*=Penas, *Dentição*=Não possui, *Reprodução*=Ovípara, *Anuro*=Não e *# Patas*=2.

TABELA 3.3 – Protótipos Encontrados na Fase de Agrupamento (ART1)

VARIÁVEIS		PROTÓTIPOS			
		0	1	2	3
Metabolismo	Endotérmico	0	0	1	1
	Ectotérmico	1	1	0	0
Cobertura	Penas	0	0	0	1
	Pêlos	0	0	1	0
	Rica em queratina	0	1	0	0
	Epiderme úmida	1	0	0	0
Dentição	Não possui	0	0	0	1
	Só as mais novas	0	0	0	0
	Completa	0	0	0	0
	Superior	0	0	0	0
Reprodução	Ovípara	1	1	0	1
	Vivípara	0	0	1	0
Anuro	Sim	1	0	0	0
	Não	0	1	1	1
Patas	2	0	0	0	1
	4	1	0	0	0
Alimentação	Carnívora	0	0	0	0
	Herbívora	0	0	0	0
	Herbívora e insetívora	0	0	0	0
	Insetívora	1	0	0	0

A partir da execução da estrutura são geradas as descrições extensionais constantes na TABELA 3.4 e intensionais contidas na FIGURA 3.8. No caso das descrições intensionais, devido à grande quantidade de regras geradas, são mostradas apenas as regras com nível de confiança e suporte de 100%.

TABELA 3.4 – Descrições Extensionais dos Agrupamentos Identificados

AGRUPAMENTO	NOME	CLASSIF. BIOLÓGICA
0	Rã, Sapo	Anfíbios
1	Crocodilo, Tartaruga	Répteis
2	Elefante, Cachorro, Gato, Coelho, Onça, Baleia	Mamíferos
3	Águia, Pardal, Gavião, Galinha, Pato	Aves

Se *Alimentação* = Insetívora **então** Agrupamento 0
Se *Anuro* = Sim **então** Agrupamento 0
Se *Cobertura* = Epiderme úmida **então** Agrupamento 0

Se *Metabolismo* = Ectotérmico **e** *Anuro* = Não **então** Agrupamento 1
Se *Cobertura* = Rica em queratina **então** Agrupamento 1

Se *Reprodução* = Vivípara **então** Agrupamento 2
Se *Cobertura* = Pêlos **então** Agrupamento 2

Se *Patas* = 2 **então** Agrupamento 3
Se *Metabolismo* = Endotérmico **e** *Reprodução* = Ovípara **então** Agrupamento 3
Se *Metabolismo* = Endotérmico **e** *Patas* = 2 **então** Agrupamento 3
Se *Metabolismo* = Endotérmico **e** *Dentição* = Não possui **então** Agrupamento 3
Se *Cobertura* = Penas **então** Agrupamento 3

FIGURA 3.8 – Exemplos de Descrições Intensionais Geradas pela Estrutura de Trabalho

É importante frisar que os agrupamentos encontrados são coerentes com o conhecimento em Biologia devido a uma escolha adequada das variáveis e do parâmetro de vigilância na rede ART1. O exemplo visa apenas ilustrar o uso da estrutura. Em aplicações reais, em geral, o especialista terá que experimentar algumas configurações de agrupamentos até encontrar alguma que faça sentido. E é exatamente na busca desse sentido nos agrupamentos que as descrições intensionais funcionam como um poderoso auxílio.

3.5 Aplicando a Estrutura de Trabalho para Mapear a Produção de Trigo no Brasil

Validamos a simulação da estrutura de trabalho, instanciada com as redes ART1 e MNC, por meio de um estudo de caso no domínio da pesquisa agropecuária, com dados fornecidos pelo Centro Nacional de Pesquisa de Trigo, da Embrapa. Foram utilizados dados referentes a 25 anos (de 1975 a 1999) de plantio de trigo em 900 localidades brasileiras, nos quais constam as variáveis município, coordenadas geográficas (latitude e longitude), altitude, área plantada e rendimento médio. Como foram consideradas somente as variáveis área plantada (em ha) e rendimento médio (em kg) durante os 25 anos, resulta que foram utilizadas 50 variáveis para estudar áreas homogêneas. As variáveis foram nomeadas com as respectivas iniciais (A para área

plantada e R para rendimento médio), concatenadas com o ano a que se referem. Assim, A1994 corresponde à área plantada no município em 1994, e R1994, ao rendimento médio obtido no mesmo ano. Considerando que os dois algoritmos requerem entradas discretas, cada variável foi discretizada em 10 classes, de acordo com recomendação do especialista. Reportando-se à FIGURA 3.1, foram seguidos os fluxos de controle P1, correspondentes à amostragem e discretização, e o IN, uma vez que os algoritmos utilizados são ambos incrementais.

A FIGURA 3.9 mostra um exemplo da tela de saída principal do sistema, contendo a parte referente às descrições intensionais do agrupamento 4, na caixa RESULTS. Para cada agrupamento é listado o lado esquerdo de cada regra que o explica. No exemplo, podemos visualizar a seguinte regra:

Se 576,2 <= R1994 <= 1072,4
então agrupamento 4 (10%, 66% e 9,3%)

O agrupamento 4 é constituído de um conjunto de observações que, durante o processo de identificação de agrupamentos, ficaram juntas em função da medida de similaridade aplicada. Como foi usado o modelo ART1 para a identificação de agrupamentos, utilizou-se o parâmetro de vigilância como medida de similaridade, arbitrado em 0,3. Esse valor condiz com a natureza altamente esparsa dos dados, fazendo com que haja bem pouca sobreposição nos valores dos atributos usados na identificação de agrupamentos. Na execução do MNC foram adotados como pontos de poda um nível de confiança no valor de 60% e suporte de 7%. Optou-se por um limite de três no número máximo de combinações como forma de se restringir o crescimento da camada combinatória do MNC. Os valores (10%, 66% e 9,3%) expressam, respectivamente, a percentagem de observações pertencentes ao agrupamento 4, o nível de confiança para essa regra e o seu suporte. A usual descrição extensional é obtida diretamente do arquivo de treinamento rotulado.

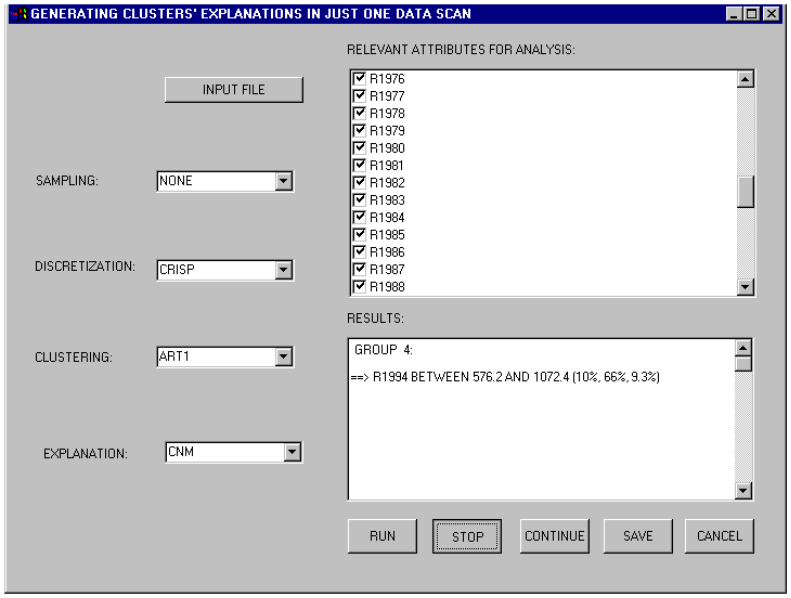

FIGURA 3.9 - Exemplo de Aplicação da Estrutura de Trabalho

3.6 Discussão

A contribuição mais importante dessa estrutura de trabalho é permitir a um analista maior flexibilidade ao estudar dados não-classificados. Essa estrutura viabiliza a integração de quaisquer dois algoritmos de aprendizado supervisionado e não-supervisionado para identificar agrupamentos compostos por objetos similares e descrever suas estruturas internas por meio das variáveis mais correlacionadas com cada agrupamento. Considerando que cada algoritmo tem seus próprios méritos, a possibilidade de se aplicarem diversas combinações deles e comparar seus resultados é algo interessante. Adicionalmente, se os dois algoritmos realizam suas funções em apenas uma leitura da base de dados, como ART1 e MNC, tem-se um ganho de eficiência significativo em grandes volumes de dados, realizando também uma única leitura da base de dados para cumprir as duas tarefas. Esse ganho é ilustrado na FIGURA 3.10, que mostra, no lado esquerdo, um diagrama do processo como feito tradicionalmente e, do lado direito, como pode ser feito dentro da nossa estrutura de trabalho.

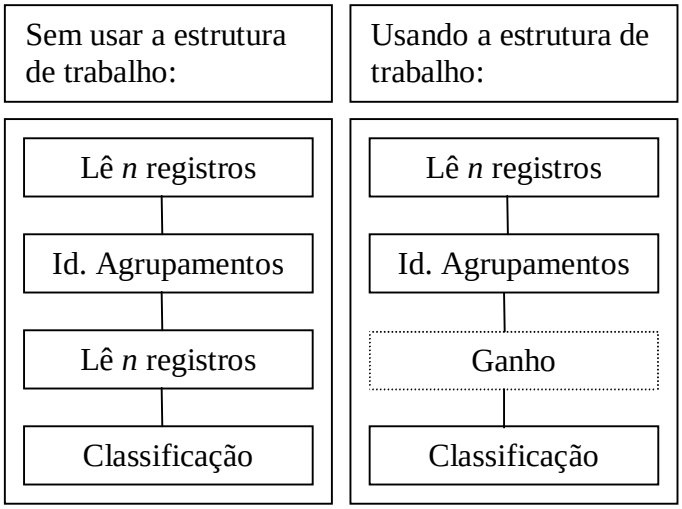

FIGURA 3.10 - Comparação de Dois Esquemas de Descoberta de Conhecimento de Dados Não-Classificados

Pode-se observar o ganho proporcionado pela estrutura ao poupar *n* acessos à base de dados. É importante frisar que, segundo Bradley [BRA 98], quando se realiza aprendizado não-supervisionado sobre grandes bases de dados, mesmo uma única leitura completa da base de dados é muito. Nesse sentido, o autor sugere que o processo possa ser interrompido a qualquer momento e que seja mostrado o modelo atual. Não vemos por que essa restrição não deva também ser aplicada para o aprendizado supervisionado. A razão para isso é que o acesso a grandes volumes de dados, usado como argumento para se limitar o acesso a dados no aprendizado não-supervisionado, representa também uma carga excessiva para o caso supervisionado. Outra importante contribuição deste trabalho é viabilizar o reuso de componentes em tarefas como discretização e amostragem.

4 Contribuições sobre o Modelo Neural Combinatório

Neste capítulo apresentamos quatro contribuições visando tornar o MNC, apresentado na seção 3.3.2, adequado para DCBD. A primeira [PRA 98b] consiste em uma modificação do algoritmo de treinamento de modo a gerar uma camada combinatória menor, auxiliando no controle da sua expansão. A segunda [PRA 99a] apresenta uma forma de poda do modelo levando em conta tanto os acumuladores quanto a acurácia do modelo resultante. Os valores dos acumuladores expressam freqüências relativas aos dados de treinamento, não garantindo a generalidade do modelo. Com um balanceamento entre os valores dos acumuladores e a acurácia em relação a dados independentes, podem-se obter modelos com maior generalidade. A terceira contribuição [PRA 2000c] mostra alguns experimentos que permitem considerar a adoção de máquinas de comitê como alternativas efetivas para a redução da complexidade do MNC. Finalmente, a quarta seção apresenta uma proposta para tratamento de dois problemas importantes do MNC [PRA 2001]: a perda de conhecimento devido à sua atualização incremental combinada com a poda e as inconsistências que podem ocorrer, principalmente no caso de se alimentar o modelo com dados ou conhecimento de várias fontes.

4.1 Geração Parcimoniosa do Modelo Neural Combinatório

Um dos maiores obstáculos para a aplicação do MNC em DCBD é o fato de a rede ser gerada completamente vazia com todas as possíveis combinações dos neurônios de entrada, para cada hipótese, na camada combinatória. Esse problema foi observado pelo autor, que propôs a adoção de um limite superior na ordem das combinações de sete mais ou menos dois (que corresponde ao número mágico de Miller [MIL 56]) como forma de se reduzir a explosão. Entretanto, verificamos que a geração de todas as possíveis combinações, mesmo se respeitarmos o limite proposto pelo autor, não representa a melhor opção, uma vez que isso pode gerar muitas combinações irreais de atributos e valores.

Feldens [FEL 97] abordou esse problema por meio da geração do modelo a partir das combinações de menor para maior ordem. Valeu-se da medida do suporte especificado pelo analista para as regras de modo a gerar nas ordens superiores somente aquelas combinações em que seus subconjuntos atendessem àquele suporte. Com isso, conseguiu viabilizar a aplicação do MNC para conjunto de dados muito maiores do que na versão original. Entretanto, nada é mencionado com relação à geração de combinações irreais. A proposta aqui apresentada é voltada exatamente para esse problema e pode ser vista como uma alternativa a ser combinada com a abordagem de Feldens para reduzir o crescimento da rede.

4.1.1 Alternativa Proposta para Geração do MNC

Nesta seção apresentamos nossa abordagem [PRA 98b] para a criação do MNC, visando à redução do custo do algoritmo de aprendizado em termos de espaço. A idéia é que tanto os neurônios quanto as conexões só sejam criados por contingência, isto é, apenas quando requeridos por um exemplo no conjunto de treinamento. Além disso, durante a fase de treinamento são computadas apenas as recompensas, ficando as punições como um passo final. A computação do efeito das classificações erradas é feita

pela diferença entre o valor do acumulador de cada combinação e a soma dos valores dos acumuladores das combinações iguais a essa, mas referentes a outras classes. O algoritmo proposto para a geração do MNC é mostrado na FIGURA 4.1.

GERAÇÃO_PARCIMONIOSA_DO_MNC

- **Para cada** exemplo no conjunto de treinamento, **faça**:
  - *Compute* todas as possíveis combinações baseadas no exemplo
  - **Para cada** combinação computada, **faça**:
  - **Se** existe uma combinação na rede chegando à mesma classe
    Então *some* um ao acumulador do arco chegando (Recompensa)
    Caso contrário *inclua* um arco correspondente à atual combinação, inicializando o seu acumulador com um (Recompensa)
  - **Fim_para**;
- **Fim_para.**

FIGURA 4.1 - Algoritmo Alternativo para Treinar o MNC

Para exemplificar, vamos tomar o conjunto de treinamento fictício usado na proposta original de Machado [MAC 89] e mostrado na TABELA 4.1.

TABELA 4.1 - Pacientes com Doenças e Sintomas Associados

Nome	Sintomas	Doença
João	s1, s2, s3	d1
Diana	s1, s2, s4	d1
Maria	s1, s3, s4	d2
Pedro	s2, s3, s4	d2

Na abordagem original, a expansão da rede, baseada nesse conjunto de treinamento, produz as combinações mostradas na TABELA 4.2. Durante a fase inicial, que corresponde à criação da rede vazia, vinte e oito combinações foram geradas. Depois da poda com limiar um, restaram dez combinações. A poda com limiar dois deixou duas combinações. Usando nossa abordagem, conforme TABELA 4.3, foram geradas vinte e duas combinações, obtendo-se igualmente dez e duas combinações após as podas com limiares um e dois, respectivamente .

Para computar o valor final dos acumuladores, o resultado das seguintes operações é tomado:

- **Para cada** combinação tome o valor do seu acumulador como ACUM;
- **Para todas** as combinações iguais à precedente apontando para classes diferentes, compute a soma dos respectivos acumuladores na variável SOMA;
- **Faça** ACUM := ACUM - SOMA.

Os três passos acima são equivalentes às punições do algoritmo original, em apenas um ciclo do algoritmo. Tanto o treinamento quanto a obtenção dos valores finais dos acumuladores podem ser acompanhados na TABELA 4.3.

TABELA 4.2 - Efeitos do Treinamento e Poda do MNC

	Sintomas					Acumuladores				Limiar/ Poda	
Doença	s1	s2	s3	s4	Início	Jo	Di	Ma	Pe	1	2
d1	X				0	1	2	1	1	1	-
		X			0	1	2	2	1	1	-
			X		0	1	1	0	-1	-	-
				X	0	0	1	0	-1	-	-
	X	X			0	1	2	2	2	2	2
	X		X		0	1	1	0	0	-	-
	X			X	0	0	1	0	0	-	-
		X	X		0	1	1	1	0	-	-
		X		X	0	0	1	1	0	-	-
			X	X	0	0	0	-1	-2	-	-
	X	X	X		0	1	1	1	1	1	-
	X	X		X	0	0	1	1	1	1	-
	X		X	X	0	0	0	-1	-1	-	-
		X	X	X	0	0	0	0	-1	-	-
d2	X				0	-1	-2	-1	-1	-	-
		X			0	-1	-2	-2	-1	-	-
			X		0	-1	-1	0	1	1	-
				X	0	0	-1	0	1	1	-
	X	X			0	-1	-2	-2	-2	-	-
	X		X		0	-1	-1	0	0	-	-
	X			X	0	0	-1	0	0	-	-
		X	X		0	-1	-1	-1	0	-	-
		X		X	0	0	-1	-1	0	-	-
			X	X	0	0	0	1	2	2	2
	X	X	X		0	-1	-1	-1	-1	-	-
	X	X		X	0	0	-1	-1	-1	-	-
	X		X	X	0	0	0	1	1	1	-
		X	X	X	0	0	0	0	1	1	-

Jo=João, Di=Diana, Ma=Maria, Pe=Pedro

TABELA 4.3 - Geração Parcimoniosa do MNC

	Sintomas					Acumuladores					Limiar/Poda	
Doença	s1	s2	s3	s4	Início	Jo	Di	Ma	Pe	Ac.	1	2
d1	X				0	1	2			1	1	-
		X			0	1	2			1	1	-
			X		0	1	1			-1	-	-
	X	X			0	1	2			2	2	2
	X		X		0	1	1			0	-	-
		X	X		0	1	1			0	-	-
	X	X	X		0	1	1			1	1	-
				X	0		1			-1	-	-
	X			X	0		1			0	-	-
		X		X	0		1			0	-	-
	X	X		X	0		1			1	1	-
d2	X				0			1	1	-1	-	-
			X		0			1	2	1	1	-
				X	0			1	2	1	1	-
	X		X		0			1	1	0	-	-
	X			X	0			1	1	0	-	-
			X	X	0			1	2	2	2	2
	X		X	X	0			1	1	1	1	-
		X			0				1	-1	-	-
		X	X		0				1	0	-	-
		X		X	0				1	0	-	-
		X	X	X	0				1	1	1	-

Jo=João, Di=Diana, Ma=Maria, Pe=Pedro

4.1.2 Aplicação do Algoritmo Modificado

Neste exemplo utilizamos dados relativos ao uso de pesticidas em alguns municípios do Estado de São Paulo, durante 1994. O treinamento foi realizado sobre os atributos cidade, cultivo, doença, pesticida e quantidade, descritos abaixo:

cidade: Código da cidade onde o pesticida foi aplicado.
Existem 120 cidades.

cultivo: Código do cultivo que recebeu o pesticida.
Existem 27 cultivos.

doença: Código da doença sendo tratada.
Existem 55 doenças.

pesticida: Código do pesticida aplicado.
Existem 140 pesticidas.

quantidade: Atributo alvo que indica o nível de pesticidas aplicado em cada cidade.
Domínio={Alto, Médio, Baixo}.

De acordo com a versão original do MNC, e desconsiderando as combinações de diferentes valores do mesmo atributo, foram gerados os tipos de combinações mostrados na TABELA 4.4.

TABELA 4.4 - Combinações Geradas por Hipótese pela Versão Original do MNC

Combinações de dois atributos:	
Cidade e cultivo:	3.240
Cidade e doença:	6.600
Cidade e pesticida:	16.800
Cultivo e doença:	1.485
Cultivo e pesticida:	3.780
Doença e pesticida:	7.700
Combinações de três atributos:	
Cidade, cultivo e doença:	178.200
Cidade, cultivo e pesticida:	453.600
Cidade, doença e pesticida:	924.000
Cultivo, doença e pesticida:	207.900
Combinações de quatro atributos:	
Cidade, cultivo, doença e pesticida:	24.948.000
Total de combinações para cada hipótese:	26.751.305

Considerando que existem três hipóteses, a quantidade total de combinações que compõem a rede vazia é 80.253.915. A geração parcimoniosa da rede, com o mesmo arquivo de treinamento, produz apenas 5.152 combinações, representando uma drástica redução no tamanho da rede. A FIGURA 4.2 mostra o sumário do treinamento, com o total de combinações geradas. Em outros arquivos de treinamento o ganho pode ser menor, mas, na maioria dos casos, um ganho considerável pode ser obtido, já que os arquivos de treinamento não trazem todas as combinações possíveis dos atributos de entrada.

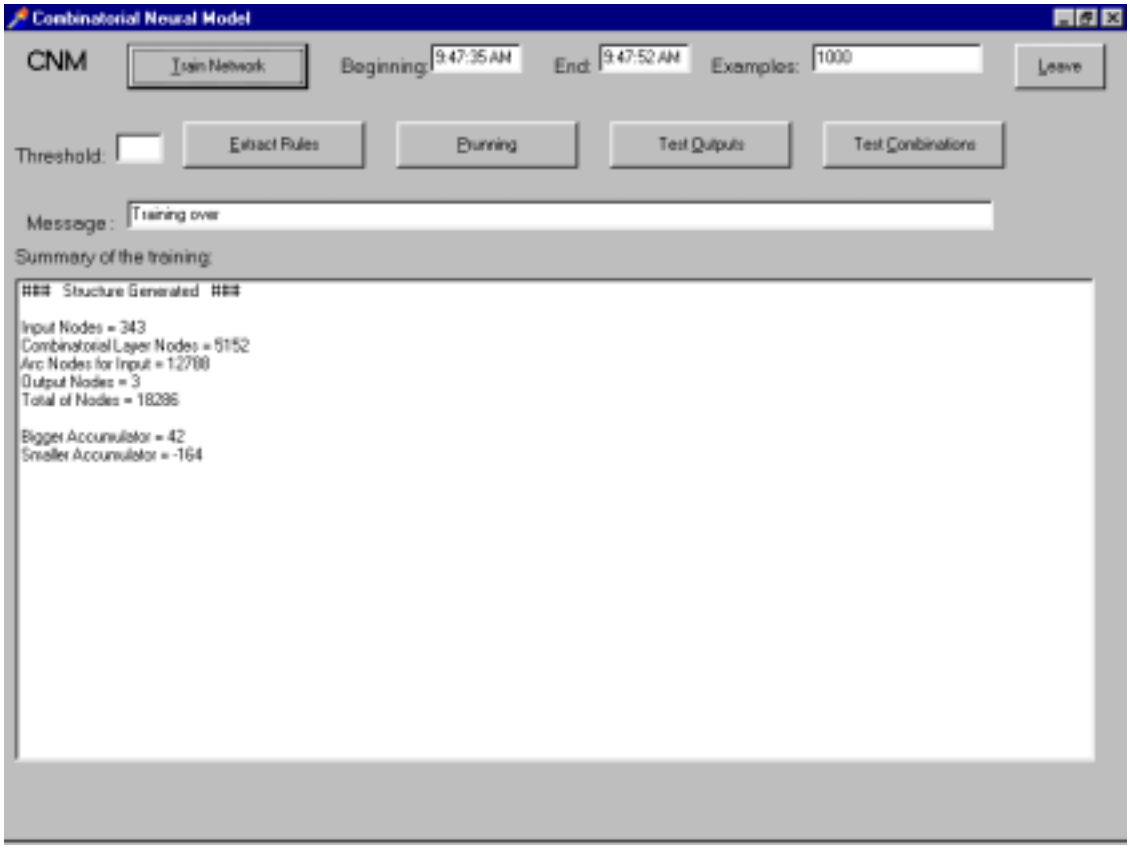

FIGURA 4.2 - Sumário da Geração Parcimoniosa do MNC

4.1.3 Discussão

Pela presente abordagem nunca são gerados arcos desnecessários na rede; esse fato leva à geração de redes treinadas menores do que na proposta original de Machado [MAC 89]. Por outro lado, a rede final, após a fase de poda, é igual à obtida por aquela proposta. Considerando que o principal problema de espaço ocorre na fase de treinamento, nossa proposta justifica-se como alternativa para se lidar com esse problema.

Os ganhos em termos de espaço, proporcionados por esta proposta durante a fase de treinamento, têm uma compensação no aumento do custo para se computar o valor final dos acumuladores, já que é necessário identificarem-se as combinações iguais apontando para diferentes hipóteses. Assim, o que está sendo proposto é mais apropriado no caso em que a restrição maior é de espaço.

O custo de espaço desta proposta é, no pior caso, igual ao da original. Em outras palavras, se todas as possíveis combinações no arquivo de treinamento estiverem associadas a todas as hipóteses, o espaço requerido para a construção do MNC nas duas alternativas será o mesmo. Em qualquer outra situação a presente proposta gerará uma rede menor.

4.2 Poda Controlada do Modelo Neural Combinatório

O método do envoltório pode ser aplicado quando se deseja definir algum parâmetro para um algoritmo e consiste na utilização do próprio algoritmo para se gerar um espaço de busca das possíveis alternativas. A idéia é colocar o algoritmo envolvido em uma camada de controle onde se realiza a busca do parâmetro desejado. Como exemplos de sucesso no uso desse método podemos citar John [JOH 97], que o usou para a seleção de atributos para treinamento, e Talavera [TAL 98], para ajuste automático de parâmetros de generalidade em identificação de agrupamentos hierárquicos.

4.2.1 Balanceando Fator de Crença e Acurácia

A alternativa aqui proposta [PRA 99a] envolve o MNC em um algoritmo de busca que aplica um arquivo de teste na rede treinada e simula diferentes pontos de poda, mostrando a acurácia correspondente a cada um. A FIGURA 4.3 ilustra como o ajuste de acurácia é conduzido.

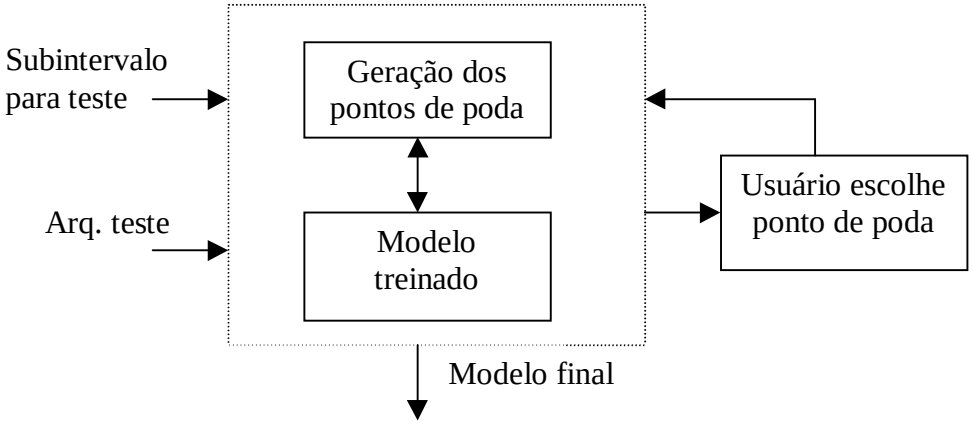

FIGURA 4.3 - Método do Envoltório para Ajuste de Acurácia no MNC

Respeitando o intervalo entre o menor e o maior acumulador na rede, o usuário é solicitado a definir um subintervalo com base no qual a primeira função do nosso método será realizada: a definição dos pontos de poda. Usando o subintervalo especificado, o algoritmo divide o intervalo definido entre o menor e o maior acumulador na rede em partes iguais (com a possível exceção da última); após essa divisão, o usuário submete o arquivo de teste à rede treinada, calculando a acurácia nos diversos pontos de poda.

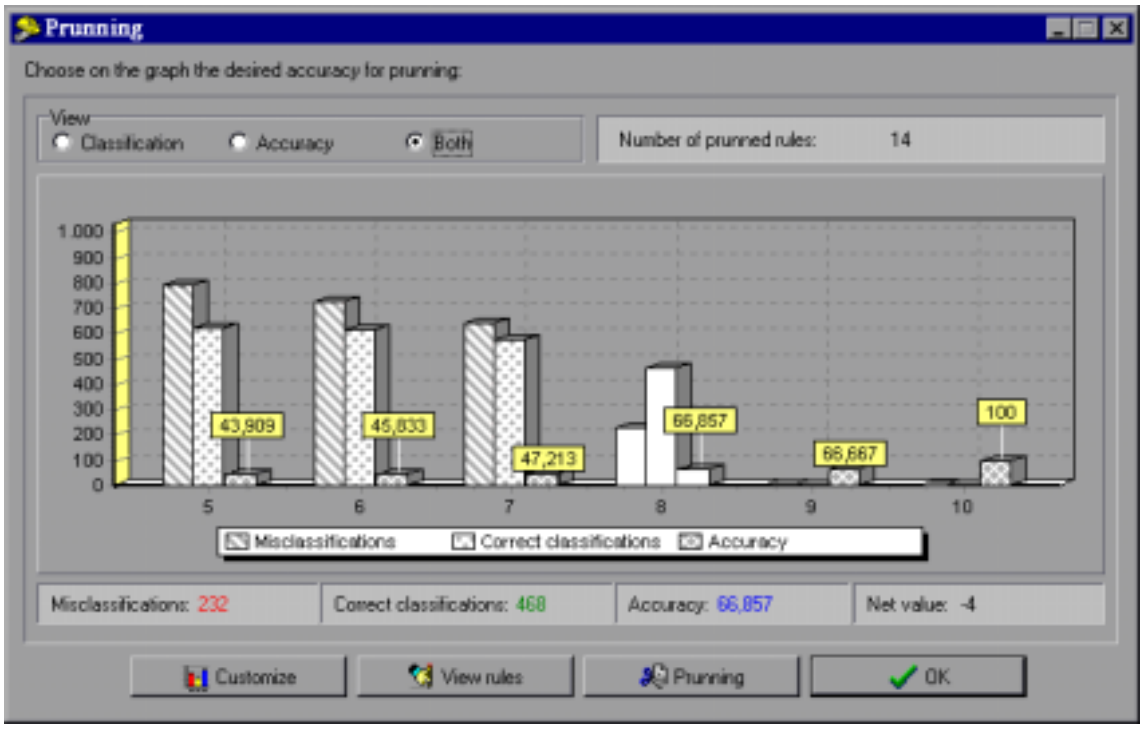

FIGURA 4.4 - Pontos de Poda e Respectivos Valores de Acurácia

Ao final de cada teste, um gráfico, conforme FIGURA 4.4, é mostrado ao usuário. Com base nesse gráfico, o usuário pode definir o ponto de poda final ou escolher o refinamento do processo dentro de um intervalo específico. No eixo *x* da FIGURA 4.4 são colocados os pontos de poda do intervalo escolhido. Sobre cada um desses pontos são mostrados, segundo opção do usuário, histogramas correspondentes ao total de erros, total de acertos e a acurácia. O processo pode ainda ser refinado focalizando-se a atenção em um particular intervalo entre dois pontos de poda (aplicando-se um efeito de aproximação), dentro do qual pode-se definir um subintervalo menor para teste.

A principal contribuição deste trabalho é a especificação e implementação de um método de poda que auxilie o usuário a superar o trabalhoso processo de escolha de um ponto de poda no MNC. Por esse método, a busca por um modelo com o melhor balanceamento entre acurácia e nível de crença desejados se torna muito mais fácil.

4.3 Uma Máquina de Comitê Baseada no Modelo Neural Combinatório

Um dos principais problemas para adoção do MNC em Mineração de Dados é o crescimento exponencial da sua camada combinatória. Nesta seção descrevemos um experimento realizado para explorar o potencial do uso de máquinas de comitê [HAY 99] para redução da complexidade do MNC. Apresentamos, inicialmente, o conceito de máquina de comitê e, em seguida, mostramos o esquema do experimento realizado. Ao final são mostrados os resultados obtidos com dados sintéticos do repositório de dados da Universidade da Califórnia, em Irvine (UCI) [BLA 2000]. O conteúdo desta seção foi publicado no artigo constante no anexo 6. Trabalho semelhante foi apresentado anteriormente [BEC 98], no qual se explorou o uso de paralelismo na construção do MNC. Entretanto, não foram reportados resultados quanto ao desempenho preditivo dos modelos assim construídos.

4.3.1 Máquinas de Comitê

A idéia central na construção de uma máquina de comitê é a de que o consenso de um conjunto de especialistas, em geral, consegue realizar melhores predições do que cada especialista separadamente. De acordo com Haykin [HAY 99], máquinas de comitê são aproximadores universais que aplicam o conceito de "dividir para conquistar" buscando tratar problemas computacionalmente complexos. Alguns trabalhos que atestam a aplicabilidade das máquinas de comitê para alcançar melhor desempenho na generalização têm sido apresentados (e.g., [PAB 96] e [SCH 97]). Máquinas de comitê podem ser classificadas como estruturas estáticas ou dinâmicas. Como uma estrutura estática, ela toma as saídas de vários preditores, combinando-as em uma solução final que não envolve o sinal de entrada. Na máquina de comitê dinâmica o sinal de entrada é considerado e atua diretamente no mecanismo que integra os especialistas individuais. Neste trabalho foi construída uma máquina de comitê estática conhecida como "método da média de *ensemble*", na qual um conjunto de modelos é treinado em paralelo e suas saídas combinadas de alguma forma para produzir uma saída geral. Esse método apresenta duas vantagens interessantes. A primeira refere-se ao tempo de aprendizado. Se um único especialista for treinado, ao invés de um conjunto de especialistas, o tempo de treinamento será maior do que no caso de se treinar o conjunto em paralelo. A segunda vantagem é que o risco de superajustamento dos dados diminui na medida em que tomamos porções de dados separadamente, combinando os resultados dos vários modelos.

4.3.2 Experimentos Realizados

Foram tomados três arquivos do repositório público da UCI: *Breast-Cancer*, *Balance Scale Weight and Distance* e *Monks-2*. Para cada caso foi realizado o processo ilustrado na FIGURA 4.5.

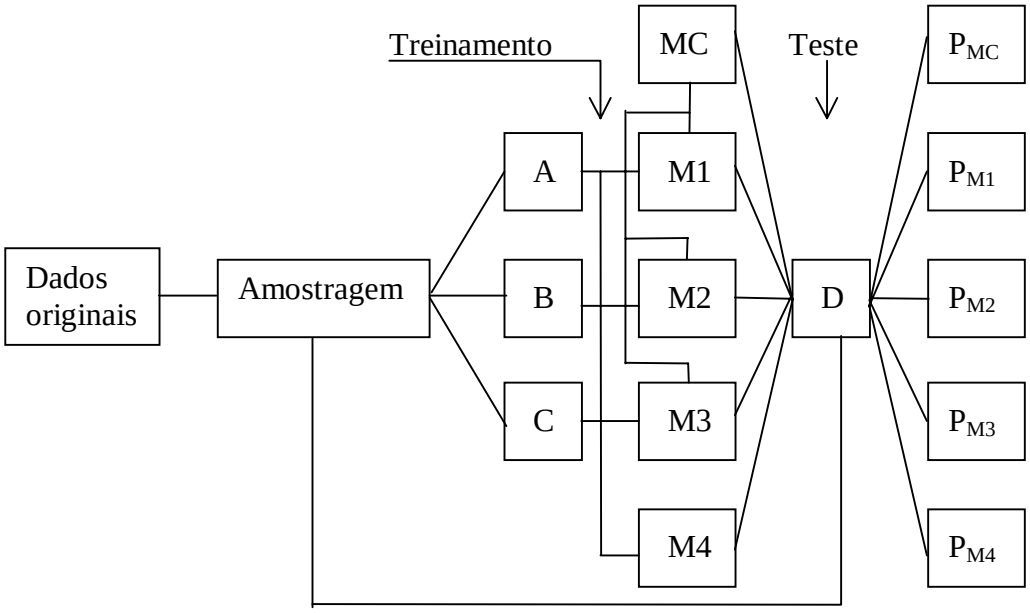

FIGURA 4.5 - Experimentos com Máquinas de Comitê

Cada arquivo foi particionado, aleatoriamente, em 4 amostras chamadas A, B, C e D. Essas amostras têm o mesmo tamanho, com a possível exceção da última, caso o

total de registros não tenha divisão inteira por 4. As amostras A, B e C foram usadas para a construção dos modelos M1, M2 e M3, respectivamente, e D foi deixada para teste. A máquina de comitê MC foi construída com base nos modelos M1, M2 e M3. O modelo M4 foi construído usando o conjunto A+B+C. Após o treinamento cada modelo foi podado, deixando-se apenas os arcos chegando à camada de saída com valores positivos nos acumuladores. Em outras palavras, apenas regras com um mínimo de suporte empírico foram mantidas.

Durante a construção dos modelos, o tempo de treinamento e o seu tamanho, levando-se em conta as camadas de entrada, saída e combinatória, foram registrados. Na fase de teste, foi medida a precisão (P), como função do número de erros de classificação sobre a partição D.

4.3.3 Resultados e Discussão

A idéia neste trabalho é expandir o MNC para um aproximador universal, usando uma máquina de comitê. Os resultados são mostrados na TABELA 4.5.

TABELA 4.5 - Comparação de Modelos e uma Máquina de Comitê

		Modelos				
Arquivo	Item	M1	M2	M3	M4	MC
Breast Cancer	Tamanho (neurônios)	24.799	24.989	23.958	55.133	
	Tempo trein. (s)	407	401	380	2.918	
	Erro (%)	19,72	25,36	23,95	18,31	23,94
Balance Scale	Tamanho (neurônios)	1.014	1.025	997	1.822	
	Tempo trein. (s)	2	2	2	9	
	Erro (%)	18,6	23,08	14,11	13,47	14,75
Monks-2	Tamanho (neurônios)	2.744	2.781	2.782	3.833	
	Tempo trein. (s)	12	10	10	44	
	Erro (%)	12,3	9,02	8,20	4,92	5,74

Pode ser observado que a máquina de comitê com três MNCs tem uma precisão comparável ao modelo único, construído com base em todos os dados juntos. Esses resultados foram obtidos ao mesmo tempo em que se obteve uma redução no tempo de treinamento e em tamanho do modelo final. Cada modelo que compõe a máquina de comitê é muito menor do que o modelo único. Esses fatos apontam o uso de máquinas de comitê como uma alternativa importante para se lidar com o problema de complexidade do MNC.

Por outro lado, apesar de as máquinas de comitê serem consideradas aproximadores universais, essa característica não foi observada nos nossos experimentos. A adoção de uma máquina de comitê não melhorou a precisão do modelo final. Isso se deve, provavelmente, ao pequeno número de modelos usados para construir a máquina. Foram especificados três, quando comumente se usam centenas de modelos [BAR 2000]. Outra provável melhora no desempenho pode ser obtida com o particionamento não-aleatório dos dados, utilizando-se a variável de classe, conforme adotado por Beckenkamp [BEC 98].

4.4 Pós-Processamento do Modelo Neural Combinatório

A questão do pós-processamento tem sido mencionada freqüentemente em textos de DCBD ([BIG 96], [FAY 96] e [MIC 98]), embora poucos trabalhos nessa direção tenham sido reportados, provavelmente como conseqüência da grande aderência desse problema à aplicação, dificultando uma abordagem mais genérica. Apresentamos nesta seção uma abordagem para pós-processamento do MNC. Apesar das diversas intervenções sobre o MNC ([FEL 97], [BEC 98], [PRA 98b], [PRA 99a], [PRA 2000a], [PRA 2000b] e [PRA 2000c]) para torná-lo adequado para DCBD, existem problemas que ainda não foram abordados. Um desses problemas são as inconsistências geradas na aquisição de conhecimento de diversas fontes, seja pela introdução de conhecimento de base de especialistas, seja pelo aprendizado por meio de exemplos. Um outro problema é a perda de conhecimento que ocorre ao se utilizar a habilidade do MNC de aprendizado incremental após a poda do modelo. Propomos nesta seção alternativas para se lidar com esses problemas.

4.4.1 Identificando Inconsistências

Em casos reais, a pura negação booleana é inadequada para se expressar a negação de um conceito. A simples concatenação de um sinal de negação, como "¬", por exemplo, antes do símbolo que se quer negar não expressa o complemento semântico do conceito representado por esse símbolo. No domínio médico, no qual o MNC foi primeiramente aplicado, é comum existir um conjunto de conceitos incompatíveis entre si. Por exemplo, um médico pode concluir por um diagnóstico de leishmaniose com base em alguns sintomas, mas outro médico pode discordar do diagnóstico por um determinado sintoma. Torna-se necessária uma abordagem para se tratar esse tipo de inconsistência. Além de se mostrarem os fatores de certeza sobre cada possível diagnóstico, é preciso expressar em que ponto ocorrem as discordâncias entre os especialistas. O modelo, como usado, não possui instrumental para tratar disso. Ao contrário, ao utilizar apenas o fator de certeza, calculado com base nas punições e premiações (ou crenças e descrenças dos especialistas), o modelo mascara o problema. Nossa proposta é expressar os conflitos entre os conceitos utilizando uma definição de negação estendida, baseada na Teoria de Conjuntos.

Por conveniência terminológica, chamamos de **condições** tanto evidências quanto hipóteses no MNC, e definimos como incompatibilidade a negação estendida entre essas condições. Isso é justificado pelo fato de que podem haver conflitos entre quaisquer tipos de condições, tanto evidências como hipóteses.

Primeiramente, define-se a noção de incompatibilidade com respeito a duas condições. Em seguida, com base nessa noção, é definido o conjunto de condições incompatíveis com outra determinada condição. Finalmente, é definido o conjunto de condições incompatíveis com outro conjunto. Isso é necessário, já que estamos interessados na identificação de incompatibilidades em regras, que podem envolver muitas condições.

Definição 1: Incompatibilidade - Uma condição *y* é incompatível com uma condição *x* se elas não podem ser verdadeiras simultaneamente.

Definição 2: Classe de incompatibilidades de uma condição *x* - A classe de incompatibilidades de uma condição *x*, denotada por *I(x)*, é composta pelo conjunto de todas as condições incompatíveis com *x*.

Definição 3: Classe estendida de incompatibilidades do conjunto $X=\{x_1,x_2,...,x_n\}$ - A classe estendida de incompatibilidades do conjunto $X=\{x_1,x_2,...,x_n\}$, denotada por $EI(x_1,x_2,...,x_n)$, é definida pelo conjunto de condições $Y=\{y_1, y_2 ..., y_m\}$, onde $Y=I(x_1)\cup I(x_2)\cup...\cup I(x_n)$.

Naturalmente, os especialistas devem definir as classes de incompatibilidades com base nas quais o sistema computará as classes estendidas de incompatibilidades. O algoritmo da FIGURA 4.6 realiza a detecção das incompatibilidades com base nas classes definidas.

ALGORITMO_DETECTOR_DE_INCOMPATIBILIDADES:

Entrada: Conjunto de regras definidas por um MNC e as classes de incompatibilidades para todas as condições nas regras;

Saída: Conjunto de regras com incompatibilidades;

Processamento:

Bloco;

Para cada regra $R(E,h)$, sendo E o conjunto das evidências e h a hipótese, **faça**:

Se $\{E,h\}\cap EI(E,h)\neq\varnothing$ **relate** "regra inconsistente: tipo 1";

Fim_para;

Para cada par ordenado de regras $R_1(E_1,h_1)$ e $R_2(E_2,h_2)$, **faça**:

Se $E_1\subseteq E_2$ **e** $\{E_1,h_1, E_2,h_2\}\cap EI_1(E_1,h_1, E_2,h_2)\neq\varnothing$ **relate** "regras R_1 e R_2 inconsistentes: tipo 2";

Fim_para;

Fim_bloco.

FIGURA 4.6 - Detecção de Incompatibilidades

Basicamente, o algoritmo detecta dois tipos de incompatibilidades. O primeiro tipo ocorre dentro das condições de cada regra. O segundo tipo ocorre entre duas regras que, eventualmente, podem ser disparadas ao mesmo tempo. Deve-se notar que a ordem das regras no segundo tipo é importante, sendo necessário verificar a relação de subconjunto próprio da primeira com a segunda e da segunda com a primeira.

Para ilustrar, suponhamos as seguintes regras:

$R_1=((e_1,e_2,e_3),h_1)$,

$R_2=((e_4,e_5,e_6),h_2)$ e

$R_3=((e_1,e_2),h_3)$

As incompatibilidades:

$I_1(e_1)=\{e_7\}$,

$I_2(e_4)=\{e_8\}$,

$I_3(h_3)=\{e_3\}$ e

$I_4(h_2)=\{e_5\}$

E as classes estendidas de incompatibilidades de interesse para nós:

$EI_1(e_1,e_2,e_3,h_1)=\{e_7\}$,

$EI_2(e_4,e_5,e_6,h_2)=\{e_5,e_8\}$,

$EI_3(e_1,e_2,h_3)=\{e_3,e_7\}$ e

$EI_4(e_1,e_2,e_3,h_1,h_3)=\{e_3,e_7\}$

Desde que e_5 é parte de EI_2 e argumento para EI_2, R_2 é relatada como "regra inconsistente: tipo 1". O par R_1 e R_3 é relatado como "regras R_1 e R_3 inconsistentes: tipo 2". Isso acontece porque as evidências de R_3 formam um subconjunto das evidências de R_1, e e_3 é parte de EI_4 e argumento para EI_4. A classe estendida de incompatibilidades para as regras R_1 e R_3 é representada pelo conjunto EI_4.

4.4.2 Prevenindo a Perda de Conhecimento

O processo de poda do MNC é baseado nos valores dos acumuladores dos arcos chegando aos neurônios de hipóteses. Por meio desse processo, o neurônio da camada combinatória que possui um acumulador menor do que um limiar definido pelo especialista é descartado. Todos os arcos chegando a esse neurônio são também descartados. Por exemplo, aplicando o conjunto de treinamento da TABELA 4.6 sobre o MNC da FIGURA 4.7 (as regras correspondentes são mostradas na TABELA 4.7), obtém-se o modelo da FIGURA 4.8.

TABELA 4.6 - Arquivo de Treinamento

Caso	Sintoma	Doença
1	e_2, e_3	h_1
2	e_2, e_3	
3	e_2, e_3	
4	e_2, e_4	h_2
5	e_2, e_4	
6	e_2, e_4	
7	e_2, e_4	
8	e_2, e_4	
9	e_2, e_4	
10	e_2, e_4	
11	e_2, e_4	
12	e_2, e_4	
13	e_2, e_4	

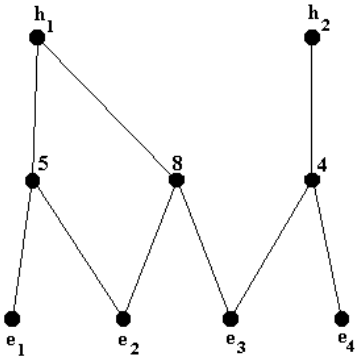

FIGURA 4.7 - Exemplo de um MNC

TABELA 4.7 - Regras Correspondentes à FIGURA 4.7

Id.	Regra
1	**Se** e_1 **e** e_2 **então** h_1
2	**Se** e_2 **e** e_3 **então** h_1
3	**Se** e_3 **e** e_4 **então** h_2

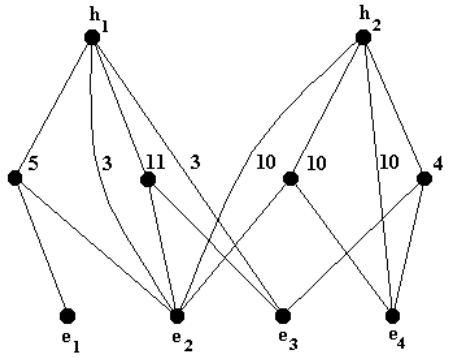

FIGURA 4.8 - Exemplo Após a Atualização

Considerando um limiar de 8, o que é razoável tendo em vista os limites superior e inferior dos acumuladores, o MNC resultante tem a forma mostrada na FIGURA 4.9. As regras correspondentes são mostradas na TABELA 4.8.

TABELA 4.8 - Regras Correspondentes à FIGURA 4.9

Id.	Regra
1	**Se** e_2 **e** e_3 **então** h_1
2	**Se** e_2 **e** e_4 **então** h_2
3	**Se** e_4 **então** h_2

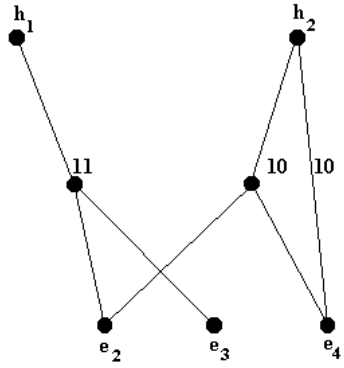

FIGURA 4.9 - Exemplo Podado Após a Atualização

Comparando as TABELAS 4.7 e 4.8, podemos notar uma drástica mudança no conhecimento, fazendo com que crenças estabelecidas desapareçam completamente. Essa mudança pode representar uma atualização de conhecimento válida, exibindo uma característica não-monotônica do MNC, ou pode significar algum erro. Em qualquer caso, é preciso revisar todas as decisões que dependem do conhecimento mudado, sendo necessário reportar essas mudanças. Idealmente, seria interessante que não houvesse poda e que todos os arcos fossem mantidos, com os seus respectivos acumuladores, já que esses têm um papel importante nas atualizações. Entretanto, em conseqüência das

limitações computacionais, a poda é inevitável. Para minorar o problema, propomos uma solução que mantém dois estados s_i e s_{i+1} da base de conhecimento, submetendo as modificações ao julgamento do analista. Isso funciona como um alarme para auxiliar o analista na revisão de suas crenças.

Nossa solução consiste em obter o conjunto $D = B_{i+1} - B_i$, que contém as modificações ocorridas na base de conhecimento B entre os instantes $i+1$ e i. Esse conjunto é obtido pelo algoritmo da FIGURA 4.10.

MUDANÇA_NO_CONHECIMENTO:
Entrada: Bases de conhecimento B_{i+1} e B_i;
Saída: Conjunto D das diferenças entre B_{i+1} e B_i;
Processamento:
Bloco
  Para cada hipótese em B_{i+1} e B_i criar L_{i+1} e L_i, respectivamente as listas de todas as regras em B_{i+1} e B_i que têm h como conseqüente;
    Compare L_{i+1} e L_i, criando D, com o seguinte conteúdo:
      Relate conhecimento novo para as regras em L_{i+1} mas não em L_i;
      Relate conhecimento perdido para as regras em L_i mas não em L_{i+1};
  Fim_para;
Fim_bloco.

FIGURA 4.10 - Detecção de Mudanças no Conhecimento

4.4.3 Discussão

O MNC tem sido adotado com sucesso para o aprendizado por meio de exemplos, podendo aprender a partir de uma estrutura vazia ou inicializada com conhecimento de base vindo de especialistas. Apesar dos resultados práticos, nenhum trabalho foi proposto para se abordar o pós-processamento do modelo. Nós identificamos importantes problemas que podem afetar a qualidade do modelo. Especificamente, abordamos as conseqüências do processo de poda sobre a natureza incremental do modelo e a falta de uma solução adequada para se tratarem conflitos resultantes da aquisição de conhecimento de diversas fontes

A característica incremental do MNC permite que o modelo aprenda continuamente a partir de exemplos. Assim, sempre que novos exemplos surgem, o aprendizado é realizado a partir do modelo existente, sem necessidade de se reprocessar os exemplos anteriores que originaram o modelo de partida. Entretanto, devido ao processo de poda, parte da estrutura (com os respectivos acumuladores) é descartada. Não existe razão para se acreditar que o conhecimento descartado será menos importante em atualizações futuras. Nessas atualizações, acumuladores com valores pouco significativos num determinado momento podem se somar com novas evidências, passando a representar conhecimento importante. Em outras palavras, é possível que, com a aplicação de novos exemplos, os acumuladores daquela parte do modelo passem a ter valores maiores ou iguais ao limiar especificado pelo especialista. Para ser realmente incremental, o MNC nunca deveria ser podado, o que é inviável, em face dos requisitos de desempenho. Nossa solução, ao controlar as mudanças no conhecimento entre dois passos de treinamento, pode, pelo menos, reduzir o problema.

Com respeito aos conflitos entre condições de um modelo, adotamos o conceito de classes de incompatibilidades, que consideramos mais expressivo do que a simples negação booleana. Ao aplicar essa noção, o sistema pode identificar incompatibilidades entre conjuntos de condições, ao invés de apenas conflitos entre um símbolo φ e sua forma negada $\neg\varphi$. As incompatibilidades relatadas podem ser úteis em uma fase de consenso, na qual os especialistas devem esclarecer quaisquer tipos de conflitos, seja entre exemplos, seja entre eles mesmos.

5 Conclusões e Trabalhos Futuros

Considerando DCBD como sendo a descoberta de conhecimento automatizada ou semi-automatizada, otimizada para tratar bases de dados cada vez maiores, procuramos identificar técnicas oriundas da área de Aprendizado de Máquina com as quais pudéssemos trabalhar essa conceituação. Chamou-nos a atenção o MNC pelo seu compromisso total com a explicitação de forma simbólica do conhecimento adquirido no treinamento a partir de exemplos. O estigma da "caixa-preta" freqüentemente associado às RNAs foi aqui atacado de forma objetiva, superando, ao nosso ver, outras abordagens para extração de conhecimento de RNAs treinadas, como o caso do modelo KBAAN [TOW 94]. A título de comparação, nesse modelo é criada, primeiramente, uma topologia da rede utilizando-se o conhecimento de especialistas. Em seguida, sobre essa representação simbólica é realizado o aprendizado conexionista. Ao final, o conhecimento é extraído com base na manipulação dos pesos. O maior problema dessa abordagem é o deslocamento do conhecimento, reconhecido pelos próprios autores, que ocorre ao se aplicar um treinamento conexionista sobre uma estrutura simbólica. Outros autores, como Osório [OSO 98a], abordaram esse problema por meio da geração de estruturas da rede em cascata, na medida em que a estrutura existente já não comportasse conhecimento novo trazido pelos exemplos.

Apesar da garantia de um mapeamento direto da sua estrutura para regras simbólicas, o MNC padece na sua origem do problema da explosão combinatória. Algumas de nossas contribuições ([PRA 98b] e [PRA 2000c]) foram nesse sentido. A primeira introduziu uma variante do algoritmo de treinamento original, que permite uma redução significativa na expansão do modelo. A segunda apontou para a efetividade da adoção de máquinas de comitê como forma de tratar aquele problema. Por outro lado, ao se adotar uma máquina de comitê, a representação simbólica do conhecimento fica dificultada, uma vez que cada conclusão pode ter sido alcançada a partir dos resultados de diferentes componentes da máquina.

Trabalho semelhante à contribuição [PRA 2000c] já havia sido publicado em Beckenkamp [BEC 98], sem, no entanto, apresentar qualquer medida de precisão na fase de teste. Com nosso estudo verificamos que uma máquina de comitê, com três especialistas construídos com partições da base de dados, obtém uma precisão comparável a cada especialista separadamente. Entretanto, ela não conseguiu superar o modelo com um único especialista treinado com todos os dados. Provavelmente, essa limitação possa ser superada com um número maior de especialistas, conforme é a prática mencionada por Baranauskas [BAR 2000].

No que se refere à automatização, ou semi-automatização, do processo de descoberta de conhecimento, apresentamos três contribuições ancoradas nas publicações [PRA 99a], [PRA 2000a], [PRA 2000 b] e [PRA 2001]. No primeiro caso, foi elaborada uma alternativa de poda para o MNC de forma a se levar em conta não apenas a informação relativa aos exemplos de treinamento como também a precisão do modelo com relação a dados que não participaram do treinamento. Dessa forma, o problema de superajustamento do modelo aos dados pode ser reduzido, uma vez que o modelo final não é definido apenas com base nos dados de entrada.

A segunda contribuição é uma estrutura de trabalho para integração dos paradigmas de aprendizado supervisionado e não-supervisionado. Nesse caso, observou-se que, ao extrair padrões de dados não-classificados, um analista identifica agrupamentos com base em alguns atributos que ele julga relevantes, e em seguida

realiza uma classificação considerando os rótulos dos agrupamentos como classes. Dessa forma, ele vai tentando criar classes e leis qualitativas entre os objetos. Em princípio, a separação desse processo em duas etapas não deveria ocorrer, uma vez que a tarefa é uma só. Acreditamos que isso ocorre por restrições dos algoritmos em fazer todo o processo de uma só vez. Dois fatos foram decisivos para que essa proposta se concretizasse: (a) a identificação de algoritmos incrementais, tanto para identificação de agrupamentos, quanto para classificação e (b) a existência de soluções para a transformação de algoritmos tradicionais para extração de padrões a partir de grandes bases de dados. Assim, percebeu-se que o processo poderia ser feito, em alguns casos, em apenas uma leitura da base de dados. Para obter maior generalidade da proposta, ela foi concebida para abrigar também algoritmos não incrementais. Considerando que existe um número expressivo de possíveis combinações de objetos em agrupamentos, a adoção de uma estrutura que permita a realização de diversos ensaios de agrupamentos é especialmente interessante.

A terceira contribuição consiste na elaboração de soluções para dois problemas do MNC. O primeiro refere-se a conflitos que podem ocorrer no modelo, devido, principalmente, ao fato de ele poder ser construído a partir de dados e conhecimento oriundos de mais de uma fonte. O segundo problema é o da perda de conhecimento devido ao processo de poda. Nessa poda, arcos com acumuladores com valores menores que um certo limiar são descartados, tornando impossível qualquer modificação nesses valores, caso novos exemplos assim exijam. Se novos exemplos venham a reforçar as combinações correspondentes a arcos descartados, novos arcos são criados e iniciados em zero. Como a manutenção do modelo completo é algo desaconselhável, devido ao seu tamanho, nossa alternativa foi minimizar o problema, mostrando as mudanças que ocorrem a cada duas atualizações do modelo.

Os resultados referentes ao artigo referenciado como [HAE 2001] não foram explorados no corpo desta tese por serem considerados ganhos laterais durante o programa de estudos e pesquisas. Adotamos essa postura pelo fato de que essas contribuições estão situadas na intersecção dos campos da Engenharia de Software e Inteligência Artificial e não de DCBD. Entretanto, vale notar que, com esse artigo, introduzimos a noção de mineração de componentes cujo objetivo é a recuperação de componentes de fontes distribuídas, provendo um significativo suporte ao reuso.

Diversas idéias que surgiram no desenvolvimento da tese, que gostaríamos de abordar, tiveram que ser adiadas devido à escassez do tempo para se concluir o trabalho da tese. Assim, algumas vertentes de pesquisa se abrem para serem exploradas futuramente. Aí se encaixam três esforços principais, sendo o primeiro a instanciação da estrutura de trabalho para incorporar novos algoritmos, bem como comparações dos resultados da aplicação de diferentes combinações de algoritmos supervisionados e não-supervisionados. Um segundo esforço de pesquisa é a implementação do modelo de pós-processamento do MNC, apresentado na seção 4.4, e a sua avaliação empírica e teórica, que deve ser tema de uma dissertação de mestrado em andamento no Instituto de Informática da UFRGS. A terceira vertente a ser explorada contempla o estudo de soluções para manutenção do significado simbólico das regras do MNC quando utilizada uma máquina de comitê.

Anexos - Artigos Publicados

Apresentamos a seguir os artigos que compõem a produção realizada durante o curso. Os oito primeiros são relacionados diretamente com a tese, constituindo a parte mais importante desta. O anexo 9 representa um resultado em que minha contribuição consiste na aplicação de um Modelo de Hopfield para a solução de um problema da área de Engenharia de Software. Após a relação dos artigos, estão anexadas cópias dos mesmos.

Anexo 1

[PRA 98] PRADO, H.A. do; FRIGERI, S.R.; ENGEL, P.M. A Parsimonious Generation of Combinatorial Neural Model. In: CONGRESO ARGENTINO DE CIENCIAS DE LA COMPUTACIÓN, 4., 1998, Neuquén, Argentina. **Proceedings...** Neuquén: Universidad Nacional del Comahue, 1998.

Anexo 2

[PRA 99a] PRADO, H.A. do; MACHADO, K.F.; FRIGERI, S.R.; ENGEL, P. M. Accuracy Tuning in Combinatorial Neural Model. In: PACIFIC-ASIA CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, 3., 1999, Bejing, China. **Proceedings…** Berlin: Springer-Verlag, 1999. (Lecture Notes in Artificial Intelligence, v. 1574).

Anexo 3

[PRA 99b] PRADO, H.A. do; MACHADO, K.F.; FRIGERI, S.R.; ENGEL, P.M. Data Mining Using Combinatorial Neural Model. **Revista Tecnologia da Informação**, Brasília, v.1, n.1, p.16-21, 1999.

Anexo 4

[PRA 2000a] PRADO, H.A. do; Explaining Cluster's Structures by Integrating Unsupervised and Supervised Learning Algorithms. In: IEEE WORKSHOP ON DB & IS, 4., 2000, Vilnius, Lituânia. **Proceedings…** Vilnius: Lithuanian Computer Society, 2000.

Anexo 5

[PRA 2000b] PRADO, H.A. do; HIRTLE, S.C.; ENGEL, P.M. Scalable Model for Extensional and Intensional Descriptions of Unclassified Data. In: IPDPS WORKSHOP ON HIGH PERFORMANCE DATA MINING, 3., 2000, Cancún, México. **Proceedings…** Berlin: Springer-Verlag, 2000. (Lecture Notes in Computer Science, v. 1800).

Anexo 6

[PRA 2000c] PRADO, H.A. do; MACHADO, K.F.; ENGEL, P.M. Alleviating the Complexity of the Combinatorial Neural Model Using a Committee Machine. In: INTERNATIONAL CONFERENCE ON DATA MINING, 2., 2000, Cambridge, Inglaterra. **Proceedings…**, Southampton: WIT Press, 2000.

Anexo 7

[PRA 2000d] PRADO, H.A. do; HIRTLE, S.C.; ENGEL, P.M. Clustering Algorithms for Data Mining. **Revista Tecnologia da Informação**, Brasília, DF, v. 2, n.1, p.51-58, 2000.

Anexo 8

[PRA 2001] PRADO, H.A. do; ENGEL, P.M.; SILVA, K.C. da; Dealing with Inconsistencies and Knowledge Loss in Combinatorial Neural Model. In: ARGENTINE SYMPOSIUM ON ARTIFICIAL INTELLIGENCE, 2001, Buenos Aires, Argentina. **Proceedings...** Buenos Aires: Universidad Nacional de Rosario, 2001.

Anexo 9

[HAE 2001] HAENDCHEN FILHO, A.; PRADO, H.A. do; ENGEL, P. M.; VON STAA, A. *XSearch* - A Neural Network Based Tool for Components Search in a Distributed Object Environment. In: INTERNATIONAL CONFERENCE ON DATABASE AND EXPERT SYSTEMS APPLICATIONS (DEXA 2001), 12., 2001, Munique, Alemanha. **Proceedings...** Berlin: Springer-Verlag, 2001. (Lecture Notes in Computer Science, v. 2113).

Anexo 1 A Parsimonious Generation of Combinatorial Neural Model

[PRA 98b] PRADO, H.A. do; FRIGERI, S.R.; ENGEL, P.M. A Parsimonious Generation of Combinatorial Neural Model. In: CONGRESO ARGENTINO DE CIENCIAS DE LA COMPUTACIÓN, 4., 1998, Neuquén, Argentina. **Proceedings...** Neuquén: Universidad Nacional del Comahue, 1998.

A Parsimonious Generation of Combinatorial Neural Model

Hércules A. Prado[*], Sandra R.Frigeri[†], Paulo M. Engel[‡]

Universidade Federal do Rio Grande do Sul
Instituto de Informática
Av. Bento Gonçalves, 9500 - Bairro Agronomia
Porto Alegre / RS - Brasil
Caixa Postal 15.064 - CEP 91.501-970
Fone: +55(051)316-6829
Fax: +55(051)319-1576
e-mail: prado, rovena, engel +{@inf.ufrgs.br}

Abstract

This paper presents a new approach to reduce the space problem due to combinatorial explosion of CNM (Combinatorial Neural Model) method. First we show a description of CNM, proposed by Machado and Rocha [MAC 91], [MAC 92], [MAC 92a], [MAC 97], as a variation of fuzzy neural network introduced as an alternative to meet many requirements, such as expressiveness, inteligibility, plasticity and flexibility. Our approach represents an alternative to generate the CNM network with certainty factors for each hypothesis. We demonstrate by means of a simple practical example that the number of combinations can be really reduced.

Keywords: Data Mining, Knowledge Discovery from Databases, Supervised Learning, Hybrid Systems, Neural Networks

[*] Lecturer at Universidade Católica de Brasília and researcher at EMBRAPA – Empresa Brasileira de Pesquisa Agropecuária
[†] Lecturer at Universidade de Caxias do Sul
[‡] Professor at Universidade Federal do Rio Grande do Sul

A Parsimonious Generation of Combinatorial Neural Model

Introduction

Classification systems based on symbolic-connectionist hybrid architectures have been proposed, e.g. [HUD 92], [KNA 92] and [GUA 94], as a way of obtaining benefits from the specific characteristics of both models. The associative characteristics of artificial neural networks (ANN) and the logical nature of symbolic systems have led to easier learning and the explanation of the acquired knowledge.

This work addresses one of such architectures, the Combinatorial Neural Model, introduced by Machado and Rocha [MAC 91], [MAC 92], [MAC 92a], [MAC 97], presenting an alternative to cope with one of its major problems: the combinatorial explosion of CNM network as the number of attributes increases. This approach is illustrated through an example of application in agricultural research. By using a real training set, the total space of original CNM network is shown and then we present the possible reduction of this space as a consequence of using our approach.

2. Description of CNM

CNM is a hybrid architecture for intelligent systems that integrates symbolic and connectionist computational paradigms. It has some significant issues, such as the ability to build a neural network from background knowledge; incremental learning by examples, solving the plasticity-stability dilemma [FRE 92]; a way to cope with the diversity of knowledge; knowledge extraction of an ANN; and the ability to deal with uncertainty. CNM is able to recognize regularities from high-dimensional symbolic data, performing mappings from this input space to a lower dimensional output space.

CNM uses supervised learning and a feedforward topology with: one input layer, one hidden layer - here called combinatorial - and one output layer (FIGURE 2.1). Each neuron of the input layer corresponds to a concept - a complete idea about an object of the domain, expressed by an object-attribute-value form, they represent the evidences of the domain application. On the combinatorial layer there are aggregator type neurons, each one connected to one or more neurons of the input layer by fuzzy AND arcs that represent logical concepts. The output layer contains one neuron for each possible class (also called hipothesis), linked to one or more neurons on the combinatorial layer by fuzzy OR arcs that also represent concepts. The synapses may be excitatory or inhibitory and they are characterized by a strength value (*weight*) between zero (not connected)

to one (fully connected synapses), that can express the logical relations. For the sake of simplicity, we will work with the learning of crisp relations, thus with strenght value of synapses equal to one, when the concept is present, and zero, when the concept is not present. However, this option does not affect the approach to fuzzy relations learning.

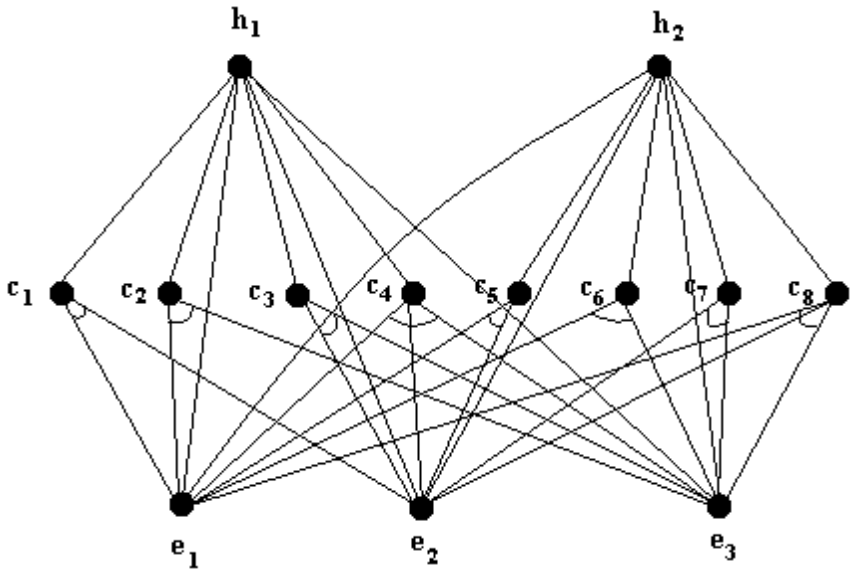

FIGURE 2.1 - The complete version of the combinatorial network for 3 input evidences and 2 hypotheses [MAC 91]

The network is created completely empty, according to the following steps: (a) one neuron in the input layer for each evidence in the training set; (b) a neuron in the output layer for each class in the training set; and (c) for each neuron in the output layer, there is a complete set of hidden neurons in the combinatorial layer which correspond to all possible combinations (lenght between two and nine) of connections with the input layer. There is no neuron in the combinatorial layer for length one connections. In this case, input neurons are connected directly to hypotheses.

The learning mechanism works in only one iteration, and it is described bellow:

PUNISHMENT_AND _REWARD_LEARNING_RULE

- *Set* to each arc of the network an accumulator with initial value zero;
- **For each** example case from the training data base, **do**:

  Propagate the evidence beliefs from input nodes until the hypotheses layer;

  For each arc reaching a hypothesis node, **do**:

  If the reached hypothesis node corresponds to the correct class of the case

Then *backpropagate* from this node until input nodes, increasing the accumulator of each traversed arc by its evidencial flow (Reward)

Else *backpropagate* from the hypothesis node until input nodes, decreasing the accumulator of each traversed arc by its evidencial flow (Punishment).

After training, the value of accumulators associated to each arc arriving to the output layer will be between [-T, T], where T is the number of all cases present in the training set.

The last step is the prunning of network; it is performed by the following actions: (a) remove all arcs whose accumulator is lower than a threshold (specified by a specialist); (b) remove all neurons from the input and combinatorial layers that became disconnected from all hypotheses in the output layer; and (c) make weights of the arcs arriving at the output layer equal to the value obtained by dividing the arc accumulators by the largest arc accumulator value in the network. After this prunning, the network becomes operational for classification tasks.

3. The Problem

Despite CNM is a simple model, it has many worthy features, as seen in the previous section. However, it has some weaknesses that limit its use, like:

- in the initial phase, the generation of the network completely empty, representing all possible combinations for each hipothesis, is clearly unfeasible as recognized by the author of the model.
- the full generation of all combinations of attribute-values may create many unreal hypotheses in respect to majority applications.
- as a consequence of its knowledge representation form, CNM has its expressivity limited to Propositional Logic.

In the first paper [MAC 89], the authors suggest the control of combinatorial explosion of the nodes in the hidden layer by incrementally building of the network. The mechanism starts with a low combination order and increases the order to an upper one until an arbitrary limit. The author suggests a criterium based on the “magic number” of Miller [MIL 56], seven plus or minus two, to stablish the upper bound to the order of combinations. Some works [LEA 93] and [FEL 97] address the same problem - the combinatorial explosion. Although they reach combinations of higher order, the search in the solution space is, as a rule, limited by the rapid growing of the network. Our approach is addressed to this problem too and may be seen as an alternative that can increase expressively the order of generated combinations and reduce the growing of the network.

4. Our approach for building CNM network

This section presents our approach to generate the CNM, that may reduce the cost of the algorithm in terms of space, and that we call *parsimonious generation of CNM network*. By this approach, the neurons and the connections are created only by contingence, i.e., only when required by an example in the training set. Moreover, during the training phase, it is only computed the rewarding of the arcs arriving at the correct class. There is no punishment. The computation of the effect of misclassifications is done by calculating the difference between the value of each accumulator at the end of the training, for each combination, and the value of the other accumulators for the same combination related to different classes.

Let us take the example of the training set used in the original proposal [MAC 89], shown in TABLE 4.1.

TABLE 4.1 - Patients with diseases and associated symptoms

Name	Symptoms	Disease
John	s1, s2, s3	d1
Diana	s1, s2, s4	d1
Mary	s1, s3, s4	d2
Peter	s2, s3, s4	d2

In the original approach, the expansion of the network based on this training set produces the combinations shown in TABLE 4.2. During the initial phase - creation of the empty network - twenty eight combinations were generated. After prunning with threshold one, ten combinations remain, and with two as threshold, two remain. Using our approach, according to TABLE 4.3, twenty two combinations are generated, and the same quantities remain - ten and two - after the prunning with threshold one and two, respectively.

The algorithm proposed for generation of the CNM takes the following form:

- **For each** example in the training set, **do**:

Compute all possible combinations based on the example

For each computed combination, **do:**

If there is an equal combination in the network arriving to the same class

Then *add* one to the accumulator of the arc arriving (Reward)

Else *include* an arc corresponding to the actual combination, setting the accumulator to one (Reward)

To compute the final value of the accumulators, the result of the folllowing operations is taken: for each accumulator of each combination, take its value as ACC; for all combinations equal to the precedent one pointing to classes different from the precedent, sum their accumulators, calling it SUM; the final result of ACC is given by ACC = ACC - SUM. It is equivalent to punishments of the original algorithm, in only one passing. Both training and accomplishment of the final value of accumulators are easily traced through TABLES 4.1 and 4.3.

TABLE 4.2 - Effects of training and prunning of the CNM

	Symptoms					Accumulators				Thrshld/ Prun.	
Disease	s1	s2	s3	s4	Begin	Jo	Di	Ma	Pe	1	2
d1	X				0	1	2	1	1	1	-
		X			0	1	2	2	1	1	-
			X		0	1	1	0	-1	-	-
				X	0	0	1	0	-1	-	-
	X	X			0	1	2	2	2	2	2
	X		X		0	1	1	0	0	-	-
	X			X	0	0	1	0	0	-	-
		X	X		0	1	1	1	0	-	-
		X		X	0	0	1	1	0	-	-
			X	X	0	0	0	-1	-2	-	-
	X	X	X		0	1	1	1	1	1	-
	X	X		X	0	0	1	1	1	1	-
	X		X	X	0	0	0	-1	-1	-	-
		X	X	X	0	0	0	0	-1	-	-
d2	X				0	-1	-2	-1	-1	-	-
		X			0	-1	-2	-2	-1	-	-
			X		0	-1	-1	0	1	1	-
				X	0	0	-1	0	1	1	-
	X	X			0	-1	-2	-2	-2	-	-
	X		X		0	-1	-1	0	0	-	-
	X			X	0	0	-1	0	0	-	-
		X	X		0	-1	-1	-1	0	-	-
		X		X	0	0	-1	-1	0	-	-
			X	X	0	0	0	1	2	2	2
	X	X	X		0	-1	-1	-1	-1	-	-
	X	X		X	0	0	-1	-1	-1	-	-
	X		X	X	0	0	0	1	1	1	-
		X	X	X	0	0	0	0	1	1	-

Jo=John, Di=Diana, Ma=Mary, Pe=Peter

TABLE 4.3 - Parsimonious generation of CNM

	Symptoms					Accumulators					Thrshd/Prun	
Disease	s1	s2	s3	s4	Begin	Jo	Di	Ma	Pe	Acc	1	2
d1	X				0	1	2			1	1	-
		X			0	1	2			1	1	-
			X		0	1	1			-1	-	-
	X	X			0	1	2			2	2	2
	X		X		0	1	1			0	-	-
		X	X		0	1	1			0	-	-
	X	X	X		0	1	1			1	1	-
				X	0		1			-1	-	-
	X			X	0		1			0	-	-
		X		X	0		1			0	-	-
	X	X		X	0		1			1	1	-
d2	X				0			1	1	-1	-	-
			X		0			1	2	1	1	-
				X	0			1	2	1	1	-
	X		X		0			1	1	0	-	-
	X			X	0			1	1	0	-	-
			X	X	0			1	2	2	2	2
	X		X	X	0			1	1	1	1	-
		X			0				1	-1	-	-
		X	X		0				1	0	-	-
		X		X	0				1	0	-	-
		X	X	X	0				1	1	1	-

5. Example

In this example we use data related to the use of pesticides in São Paulo[*], during 1994, according to FIGURE 5.1. Training is accomplished over attributes *city*, *crop*, *disease*, *pesticide*, and *quantity*, described below:

city: Code of the city where the pesticide was applied.
There are 120 cities.

crop: Code of the crop that received the pesticide.
There are 27 crops.

disease: Code of the disease being treated.
There are 55 diseases.

pesticide: Code of the pesticide applied.
There are 140 pesticides.

[*] Data obtained by agreement between EMBRAPA Environment and CREA-SP.

quantity: It is the target attribute and indicates pesticides level applied in one city. Domain={High, Medium, Low}.

According to the original version of CNM, disregarding combinations between different values from the same attribute, combinations shown in TABLE 5.2 are generated.

TABLE 5.2 - Generated combinations for each hypothesis through CNM

Combinations of 2 attributes:	
city and *crop:*	3,240
city and *disease:*	6,600
city and *pesticide:*	16,800
crop and *disease:*	1,485
crop and *pesticide:*	3,780
disease and pesticide:	7,700
Combinations of 3 attributes:	
city, crop and disease:	178,200
city, crop and *pesticide:*	453,600
city, disease and pesticide:	924,000
crop, disease and pesticide:	207,900
Combinations of 4 attributes:	
city, crop, disease and pesticide:	24,948,000
Total of combinations for each hypothesis:	26,751,305

Considering that we have 3 hypotheses, the total amount of generated combinations, with empty network, is 80,253,915. The parsimonious generation of the network, with the same training set, produced only 5,152 combinations, representing a drastic redution on the number of generated combinations. FIGURE 5.1 shows the summary of the training, listing total combinations generated. In other training sets this gain may be lower, but it is possible that in almost all cases a considerable gain will be obtained.

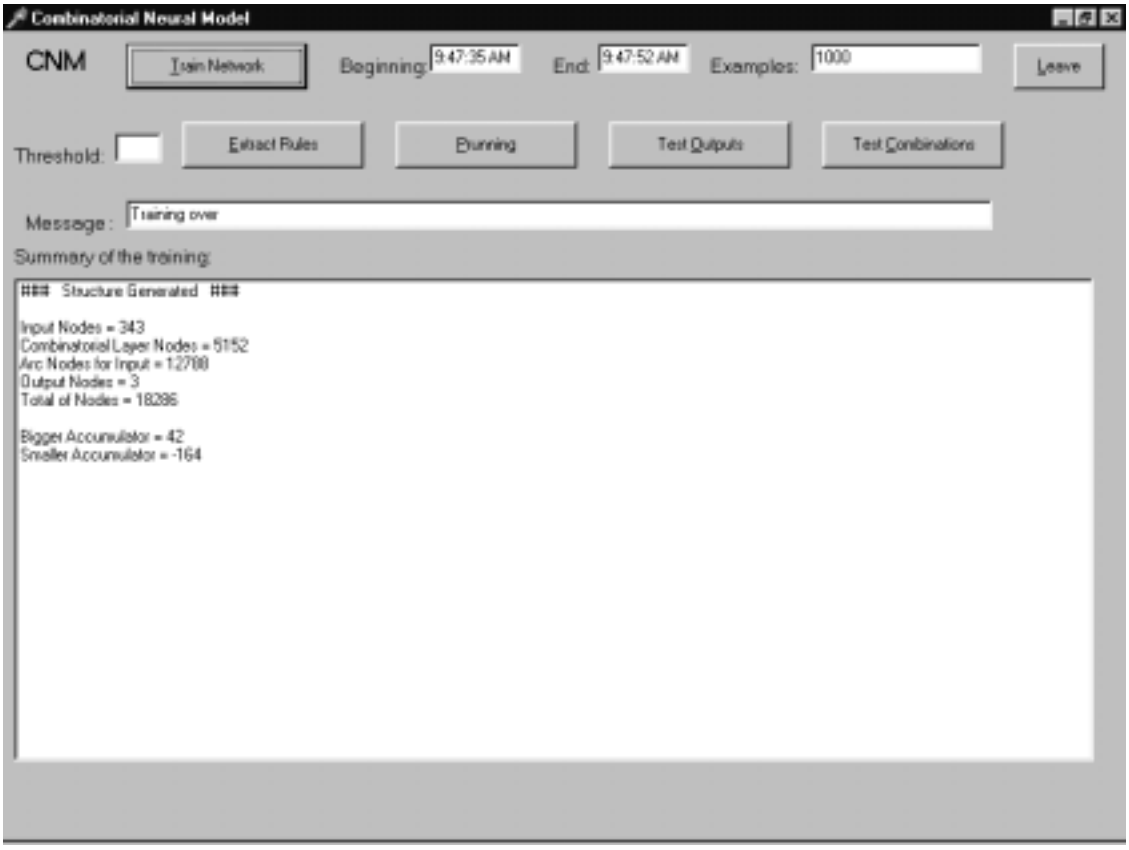

FIGURE 5.1 - Partial outputs of the parsimonious generation of CNM

6. Conclusions

By the presented approach it is never created unnecessary arcs in the network; this fact leads to the generation of trained networks smaller than the original proposal [MAC 91], [MAC 92], [MAC 92a], [MAC 97], and other approachs [LEA 93] and [FEL 97]. On the other hand, the final network obtained after prunning phase is the same in all alternatives. The main problem of space occurs in the training phase and our approach reduces this problem.

During training phase, gain in terms of space, provided by this proposal, presents a compensation through cost increase to compute the final value of accumulators, since it is necessary to identify equal combinations for different hypotheses. However, such way of building the network may be considered an alternative when the main restriction is space.

The space complexity of this proposal will be, at worst case, equal to that of the original one. In other words, if all possible combinations are associated to all possible hypotheses in the training set, the required space to build the CNM in both alternatives, will be the same. In any other situations, the present proposal will generate a smaller network.

Bibliographic References:

[FEL 97] FELDENS, M. A. & CASTILHO, J. M. V. Data mining with the combinatorial rule model: an application in a health-care relational database. In: CLEI, 23., 1997, Valparaízo, Chile. **Proceedings ...** Valparaízo: CLEI, 1997.v. 1, p. 135-145.

[FRE92] FREEMAN, James A.; SKAPURA, David M. Adaptive Resonance Theory. In: **Neural Networks, Algorithms, Applications, and Program Techniques.** Reading: Addison-Wesley, 1992. 401p. p.292-339.

[GUA 94] GUAZZELLI, A. **Aprendizagem em Sistemas Híbridos.** Porto Alegre: CPGCC da UFRGS, 1994. Dissertação de mestrado.

[HUD 92] HUDSON, D. L. *et al.* Medical diagnosis and treatment plans derived from a hybrid expert system. In: KANDEL, A. & LANGHOLZ, G**. Hybrid architectures for intelligent systems**, Boca Raton, FL: CRC Press, 1992.

[KNA 92] KNAUS, R. Representing expert knowledge in neural nets. In: KANDEL, A. & LANGHOLZ G. **Hybrids Architectures for Intelligent Systems**, Boca Raton, FL: CRC Press, 1992.

[LEA 93] LEÃO, B. F.; REÁTEGUI, E. B. A hybrid connectionist expert system to solve classification problems. **Proceedings of Computer in Cardiology,** IEEE Computer, IEEE Computer Society, London, 1993.

[MAC 89] MACHADO, R. J.; ROCHA, A. F. **Handling knowledge in high order neural networks: the combinatorial neural network.** Rio de Janeiro: IBM Rio Scientific Center, 1989. (Technical Report CCR076) .

[MAC 91] MACHADO, R. J. & ROCHA, A. F. da. The combinatorial neural network: a conectionist model for knowledge based systems. In: Bouchon, b.; Yager, r. r.; Zadeh, l. a. **Uncertainty in Knowledge Bases**, Berlin, Germany: Springer Verlag, 1991, p.578-587.

[MAC 92] MACHADO, R. J. & ROCHA, A. F. da. **Evolutive fuzzy neural networks**. In: IEEE International Conference on Fuzzy Systems, San Diego, CA, p.493-500. Mar.1992.

[MAC 92a] MACHADO, R. J.; ROCHA, A. F. da. A Hybrid architecture for fuzzy connectionist expert systems. In: KANDEL A. & LANGHOLZ G. **Hybrids Architectures for Intelligent Systems**, Boca Raton, FL: CRC Press, 1992. p. 136-152.

[MAC 97] MACHADO, R. J.; ROCHA, A. F. da. Inference, inquiry, evidence censorship, and explanation in connectionist expert systems. **IEEE Transactions on Fuzzy Systems**, New York, v. 5, n. 3, p.443-459, Aug. 1997.

[MIL 56] MILLER, G. A., The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. **Psycological Review**, [s.l.], 63, p.81-97, 1956.

Anexo 2 Accuracy Tuning in Combinatorial Neural Model

[PRA 99a] PRADO, H.A. do; MACHADO, K.F.; FRIGERI**,** S.R.; ENGEL, P. M. Accuracy Tuning in Combinatorial Neural Model. In: PACIFIC-ASIA CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, 3., 1999, Bejing, China. **Proceedings…** Berlin: Springer-Verlag, 1999. (Lecture Notes in Artificial Intelligence, v. 1574).

Accuracy Tuning on Combinatorial Neural Model

Hércules A. Prado *, Karla F. Machado **, Sandra R. Frigeri * * *, Paulo M. Engel †

Universidade Federal do Rio Grande do Sul
Instituto de Informática
Av. Bento Gonçalves, 9500 - Bairro Agronomia
Porto Alegre / RS - Brasil
Caixa Postal 15.064 - CEP 91.501-970
Fone: +55(051)316-6829
Fax: +55(051)319-1576
{prado, karla, rovena, engel}@inf.ufrgs.br

Abstract. The Combinatorial Neural Model (CNM) ([8] and [9]) is a hybrid architecture for intelligent systems that integrates symbolic and connectionist computational paradigms. This model has shown to be a good alternative to be used on data mining; in this sense some works have been presented in order to deal with scalability of the core algorithm to large databases ([2], [1] and [10]). Another important issue is the prunning of the network, after the training phase. In the original proposal this prunning is done on the basis of accumulators values. However, this criterion does not give a precise notion of the classification accuracy that results after the prunning. In this paper we present an implementation of the CNM with a feature based on the wrapper method ([6] and [12]) to prune the network by using the accuracy level, instead of the value of accumulators as in the original approach.

1 Introduction

Classification systems based on symbolic-connectionist hybrid architectures have been proposed ([5], [7], [4] and [11]) as a way to obtain benefits from the specific characteristics of both models. The associative characteristics of artificial neural networks (ANN) and the logical nature of symbolic systems have led to easier learning and the explanation of the acquired knowledge.

This work addresses one of such architectures, the Combinatorial Neural Model ([8] and [9]). In this model, the criterion to prune the network, after training, is based on values of accumulators. However, it does not express directly its accuracy level. To adjust this accuracy level it is necessary to set a value to an accumulator and observe the corresponding accuracy level. After some tryings the user may choose the accumulator that leads to the desired accuracy. In this paper we face this problem providing a way to avoid this boring task by means of an automatic process to reach the most accurate model without numerous manual tryings.

* Researcher at EMBRAPA — Brazilian Enterprise for Agricultural Research
** Research Assistant at Federal University of Rio Grande do Sul
* * * Lecturer at University of Caxias do Sul
† Professor at Federal University of Rio Grande do Sul

2 Description of the CNM

The CNM is a hybrid architecture for intelligent systems that integrates symbolic and connectionist computational paradigms. It has some significant features, such as the ability to build a neural network from background knowledge; incremental learning by examples, ability to solve the plasticity-stability dilemma [3]; a way to cope with the diversity of knowledge; knowledge extraction of an ANN; and the ability to deal with uncertainty. The CNM is able to recognize regularities from high-dimensional symbolic data, performing mappings from this input space to a lower dimensional output space.

The CNM uses supervised learning and a feedforward topology with: one input layer, one hidden layer - here called combinatorial - and one output layer (Figure 1). Each neuron in the input layer corresponds to a concept - a complete idea about an object of the domain, expressed in an object-attribute-value form. They represent the evidences of the domain application. On the combinatorial layer there are aggregative fuzzy AND neurons, each one connected to one or more neurons of the input layer by arcs with adjustable weights. The output layer contains one aggregative fuzzy OR neuron for each possible class (also called hypothesis), linked to one or more neurons on the combinatorial layer. The synapses may be excitatory or inhibitory and they are characterized by a strength value (weight) between zero (not connected) to one (fully connected synapses). For the sake of simplicity, we will work with the learning of crisp relations, thus with strenght value of synapses equal to one, when the concept is present, and zero, when the concept is not present. However, the same approach can be easily extended to fuzzy relations.

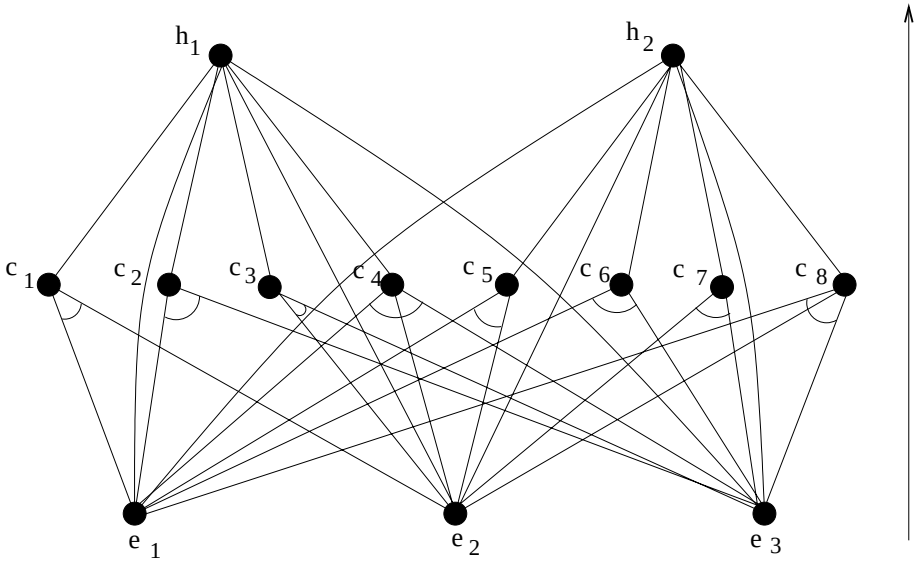

Fig. 1. The complete version of the combinatorial network for 3 input evidences and 2 hypotheses [8]

The network is created completely uncommited, according to the following steps: (a) one neuron in the input layer for each evidence in the training set; (b) a neuron in the output layer for each class in the training set; and (c) for each neuron in the output layer, there is a complete set of hidden neurons in the combinatorial layer which corresponds to all possible combinations (lenght between two and nine) of connections with the input layer. There is no neuron in the combinatorial layer for single connections. In this case, input neurons are connected directly to the hypotheses.

The learning mechanism works in only one iteration, and it is described bellow:

PUNISHMENT_AND_REWARD_LEARNING_RULE

- **Set** to each arc of the network an accumulator with initial value zero;
- **For each** example case from the training data base, **do**:
  - *Propagate* the evidence beliefs from input nodes until the hypotheses layer;
  - **For each** arc reaching a hypothesis node, **do**:
    - **If** the reached hypothesis node corresponds to the correct class of the case
    - **Then** *backpropagate* from this node until input nodes, increasing the accumulator of each traversed arc by its evidencial flow (Reward)
    - **Else** *backpropagate* from the hypothesis node until input nodes, decreasing the accumulator of each traversed arc by its evidencial flow (Punishment).

After training, the value of accumulators associated to each arc arriving to the output layer will be between [-T, T], where T is the number of cases present in the training set. The last step is the prunning of network; it is performed by the following actions: (a) remove all arcs whose accumulator is lower than a threshold (specified by a specialist); (b) remove all neurons from the input and combinatorial layers that became disconnected from all hypotheses in the output layer; and (c) make weights of the arcs arriving at the output layer equal to the value obtained by dividing the arc accumulators by the largest arc accumulator value in the network. After this prunning, the network becomes operational for classification tasks.

3 Accuracy Tuning

As described above, the prunning of the generated network is done by means of the accumulators values, what does not give a precise idea of the resulting classification accuracy of the trained network. To achieve a specific classification accuracy one has to repeatedly modify the prunning threshold, by trial and error. In this section we present a model that uses an implementation of a wrapper algorithm for accuracy tuning of CNM without this tryings. Examples of successful use of wrapper methods are the attribute subset selection algorithm presented by John [6] and the algorithm for parameter selection on ISAAC system [12]. In our application the algorithm involves the trained network and uses an off training dataset as input for test different values of accumulators for prunning. This model may be easily extended to incorporate more interesting validation methods, as n-fold cross validation. Figure 2 shows how the model carries out the accuracy tuning. Considering that the values of the accumulators vary from -T to T, being T the number of cases present in the training set, the user have to specify the step to be used on the first function of the wrapper: the generation of prunning points. For this purpose, the system shows the boundary values of the accumulators.

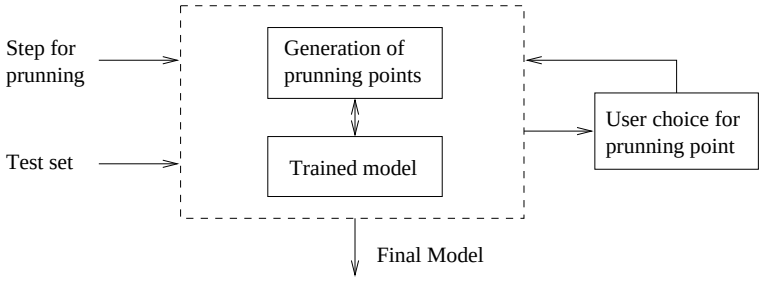

Fig. 2. The wrapper method for accuracy tuning on CNM

Using the specified step, the algorithm divides the interval defined between minimum and maximum accumulators in equal parts (with a possible exception on the last part); after this definition the user submit the test set to the trained network. It is repeated for each step and the results are reported in a graphical form, as depicted in Figure 3. One additional feature is the possibility to focus the attention on a particular interval (like a zoom effect), specifying a shorter step to refine the accuracy tuning.

On the x axis it is the prunning points and on the y axis it is shown the results on each prunning point. The user may choose one prunning point according to the values on the bottom of the screen. In the example, it is selected the prunning point eight. Having decided where to prune the network the user orders the prunning, obtaining the final network with the desired accuracy.

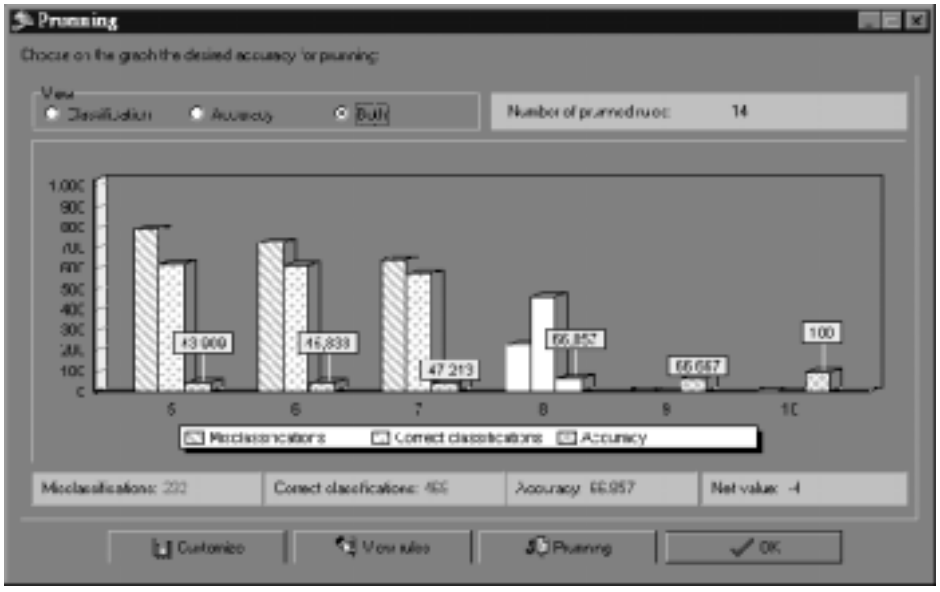

Fig. 3. Prunning points with corresponding accuracy

4 Conclusions and Future Work

CNM has received important improvements in order to be more adequate to data mining. This improvements has focussed on optimizations of processing time [1] and scalability of the algorithm to larger databases [10]. Another important issue is the development of tools to help one in accuracy tuning.

The wrapper method has shown to be useful to set parameters of a model using its own learning algorithm. Our contribution with the present work is the use of wrapper to help the user to overcome the laborious task of determining the prunning point of the combinatorial neural model. By this approach the search for the most accurate model has become easier than by trying many alternatives of prunning.

Although the effectiveness of this way of prunning, we believe that this approach must be improved by allowing the use of better methods for validation as n-fold cross validation. By using such validation methods, one can search for an optimum trade-off between the number of remaining rules after prunning and the accuracy level of the model.

References

1. Beckenkamp, F. G., Feldens, M. A., Pree, W.: Optimizations of the Combinatorial Neural Model. IN: Vth Brazilian Symposium on Neural Networks. (SBRN'98), Belo Horizonte, Brazil.
2. Feldens, M. A.: Engenharia da Descoberta de Conhecimento em Bases de Dados: Estudo e Aplicação na Área de Saúde. Porto Alegre: CPGCC da UFRGS, Brasil, 1997. Dissertação de mestrado.
3. Freeman, J. A., Skapura, D. M.: Neural Networks, Algorithms, Applications, and Program Techniques. [S.l.]: Addison-Wesley, 1992. p.292-339.
4. Guazzelli, A.: Aprendizagem em Sistemas Híbridos. Porto Alegre: CPGCC da UFRGS, Brasil, 1994. Dissertação de mestrado.
5. Hudson, D. L. et al.: Medical diagnosis and treatment plans derived from a hybrid expert system. In: Kandel, A. & Langholz, G. Hybrid architectures for intelligent systems, Boca Raton, FL: CRC Press, 1992.
6. John, G. H.: Enhancements to the Data Mining Process. Stanford, EUA: Stanford University, 1997. Ph.D. Thesis.
7. Knaus, R.: Representing expert knowledge in neural nets. In: Kandel, A. & Langholz G. Hybrids Architectures for Intelligent Systems, Boca Raton, FL: CRC Press, 1992.
8. Machado, R. J., Rocha, A. F.: Handling knowledge in high order neural networks: the combinatorial neural network. Rio de Janeiro: IBM Rio Scientific Center, Brazil, 1989. (Technical Report CCR076).
9. Machado, R. J., Carneiro, W., Neves, P. A.: Learning in the combinatorial neural model, IEEE Transactions on Neural Networks, v.9, p.831-847. Sep.1998.
10. Prado, H. A., Frigeri, S. R., Engel, P. M.: A Parsimonious Generation of Combinatorial Neural Model. IN: IV Congreso Argentino de Ciencias de la Computación (CACIC'98), Neuquén, Argentina, 1998.
11. Towell, G.G., Shavlik, J.W.: Knowledge-based artificial neural networks. Artificial Intelligence 70(1994) 119-165.
12. Talavera, L.: Exploring Efficient Attribute Prediction in Hierarchical Clustering. IN: VII Congreso Iberoamericano de Inteligencia Artificial, IBERAMIA'98, Lisbon, Portugal, 1998.

Anexo 3 Data Mining Using Combinatorial Neural Model

[PRA 99b] PRADO, H.A. do; MACHADO, K.F.; FRIGERI, S.R.; ENGEL, P.M. Data Mining Using Combinatorial Neural Model. **Revista Tecnologia da Informação**, Brasília, v.1, n.1, p.16-21, 1999.

Data Mining Using Combinatorial Neural Model

Hércules A. Prado[†], Sandra R.Frigeri[‡], Karla F. Machado[‡†], Paulo M. Engel[‡‡]

Universidade Federal do Rio Grande do Sul
Instituto de Informática
Av. Bento Gonçalves, 9500 – Bairro Agronomia
Porto Alegre / RS - Brasil
Caixa Postal 15.064 – CEP 91.501-970
Fone: +55(051)316-6829
Fax: +55(051)319-1576
e-mail: prado, rovena, karla, engel +{@inf.ufrgs.br}

Abstract

This paper presents an extended synthesis of two previous ones ([PRA 98] and [PRA 99]). Both face typical problems of data mining: scaling machine learning algorithms to very large data bases and the creation of features to make easier the application of these algorithms. The target algorithm is the Combinatorial Neural Model (CNM), proposed by Machado and Rocha ([MAC 89] and [MAC 91]) as a variation of fuzzy neural network. First we show a description of CNM, introduced as an alternative to meet desirable characteristics such as expressiveness, inteligibility, plasticity and flexibility. Next, an alternative to generate the CNM network, saving memory in the building of the model, is presented; this savings are demonstrated by means of a simple practical example. Also, another important issue is faced: the prunning of the model after the training phase. In the original proposal this prunning is done on the basis of accumulators values on the model. This criteria does not give a precise notion of the accuracy that results after the prunning. We present an implementation of the model with a feature based on the wrapper method ([JOH 97] and [TAL 98]) to prune the network by using the accuracy level, instead of the value of accumulators. Finally, we discuss the contributions and future work.

Keywords: Data Mining, Knowledge Discovery from Databases, Supervised Learning, Hybrid Systems, Neural Networks.

[†] Computing Specialist at EMBRAPA – Brazilian Enterprise for Agricultural Research and lecturer at Catholic University of Brasília
[‡] Lecturer at University of Caxias do Sul
[‡†] Research Assistant at Federal University of Rio Grande do Sul
[‡‡] Professor at Federal University of Rio Grande do Sul

Data Mining Using Combinatorial Neural Model

1. Introduction

Classification systems based on symbolic-connectionist hybrid architectures have been proposed, e.g. [HUD 92], [KNA 92] and [GUA 94], as a way of obtaining benefits from the specific characteristics of both models. The associative characteristics of artificial neural networks (ANN) and the logical nature of symbolic systems have led to easier learning and the explanation of the acquired knowledge.

This work addresses one of such architectures, the Combinatorial Neural Model (CNM), introduced by Machado and Rocha ([MAC 89] and [MAC 91]) presenting two improvements: (a) an alternative to cope with one of its major problems: the combinatorial explosion of CNM network as the number of attributes increases; and (b) a feature for automatic prunning of the model without numerous manual tryings. The second improvement deserves an special attention: to adjust the accuracy level of the model in the original algorithm it is necessary to take an accumulator value, proceed the prunning, and watch the corresponding accuracy level. After some tryings the user may choose the accumulator that will lead to the desired accuracy. In this paper we face this problem by providing a way to avoid this boring task by means of an automatic process to reach the more accurate model. The improvements are illustrated through an example of application in agricultural research. By using a real training set, the gain of space on building CNM network and the interface for prunning are shown.

2. Description of CNM

CNM is a hybrid architecture for intelligent systems that integrates symbolic and connectionist computational paradigms. It has some significant issues, such as the ability to build a neural network from background knowledge; incremental learning by examples, solving the plasticity-stability dilemma [FRE 92]; a way to cope with the diversity of knowledge; knowledge extraction of an ANN; and the ability to deal with uncertainty. CNM is able to recognize regularities from high-dimensional symbolic data, performing mappings from this input space to a lower dimensional output space.

CNM uses supervised learning and a feedforward topology with: one input layer, one hidden layer - here called combinatorial - and one output layer (FIGURE 2.1). Each neuron of the input layer corresponds to a concept - a complete idea about an object of the domain, expressed by an object-attribute-value form, they represent the evidences of the domain application. On the combinatorial layer there are aggregator type neurons, each one connected to one or more neurons of the input layer by fuzzy AND arcs that represent logical concepts. The output layer contains one neuron for each possible class (also called hipothesis), linked to one or more neurons on the combinatorial layer by fuzzy OR arcs that also represent concepts. The synapses may be excitatory or inhibitory and they are characterized by a strength value (*weight*) between zero (not connected) to one (fully connected synapses), that can express the logical relations. For the sake of simplicity, we will work with the learning of crisp relations, thus with strenght value of synapses equal to one, when the concept is present, and zero, when the concept is not present. However, this option does not affect the approach to fuzzy relations learning.

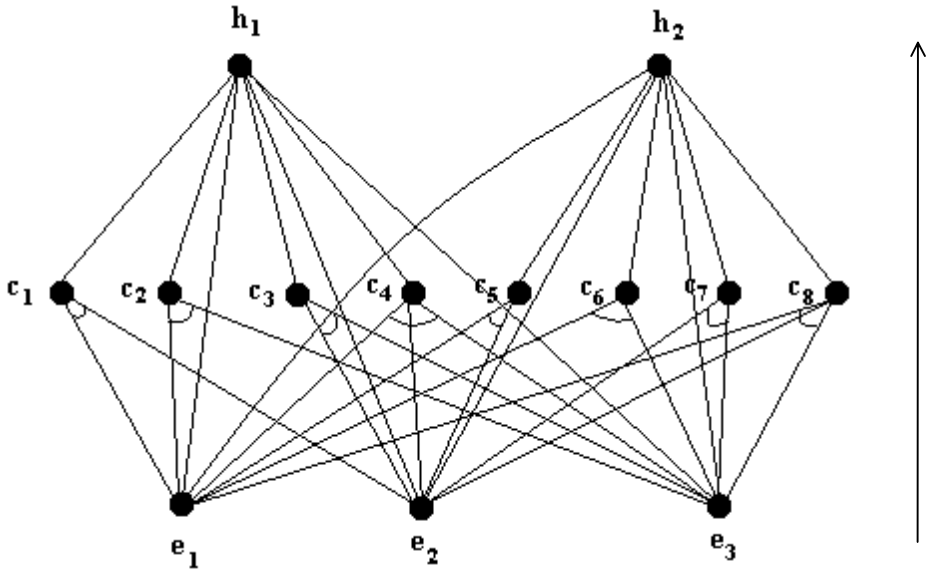

FIGURE 2.1 - The complete version of the combinatorial network for 3 input evidences and 2 hypotheses [MAC 91]

The network is created completely empty, according to the following steps: (a) one neuron in the input layer for each evidence in the training set; (b) a neuron in the output layer for each class in the training set; and (c) for each neuron in the output layer, there is a complete set of hidden neurons in the combinatorial layer which correspond to all possible combinations (lenght between two and nine) of connections with the input layer. There is no neuron in the combinatorial layer for length one connections. In this case, input neurons are connected directly to hypotheses.

The learning mechanism works in only one iteration, and it is described bellow:

PUNISHMENT_AND _REWARD_LEARNING_RULE

- *Set* to each arc of the network an accumulator with initial value zero;
- **For each** example case from the training data base, **do**:
  Propagate the evidence beliefs from input nodes until the hypotheses layer;
  For each arc reaching a hypothesis node, **do**:
  If the reached hypothesis node corresponds to the correct class of the case
  Then *backpropagate* from this node until input nodes, increasing the accumulator of each traversed arc by its evidencial flow (Reward)
  Else *backpropagate* from the hypothesis node until input nodes, decreasing the accumulator of each traversed arc by its evidencial flow (Punishment).

After training, the value of accumulators associated to each arc arriving to the output layer will be between [-T, T], where T is the number of all cases present in the training set.

The last step is the prunning of network; it is performed by the following actions: (a) remove all arcs whose accumulator is lower than a threshold (specified by a specialist); (b) remove all neurons from the input and combinatorial layers that became disconnected from all hypotheses in the output layer; and (c) make weights of the arcs arriving at the output layer equal to the value obtained by dividing the arc accumulators by the largest arc

accumulator value in the network. After this prunning, the network becomes operational for classification tasks.

3. The problem of exponential growing of CNM

Despite CNM is a simple model, it has many worthy features, as seen in the previous section. However, it has some weaknesses that limit its use, like:

- in the initial phase, the generation of the network completely empty, representing all possible combinations for each hipothesis, is clearly unfeasible as recognized by the author of the model.
- the full generation of all combinations of attribute-values may create many unreal hypotheses in respect to majority applications.
- as a consequence of its knowledge representation form, CNM has its expressivity limited to Propositional Logic.

In the first paper [MAC 89], the authors suggested the control of combinatorial explosion of the nodes in the hidden layer by incrementally building of the network. The mechanism starts with a low combination order and increases the order to an upper one until an arbitrary limit. The authors suggested a criterium based on the "magic number" of Miller [MIL 56], seven plus or minus two, to stablish the upper bound to the order of combinations. Some works [LEA 93] and [FEL 97] address the same problem - the combinatorial explosion. Although they reach combinations of higher order, the search in the solution space is, as a rule, limited by the rapid growing of the network. Our approach is addressed to this problem too and may be seen as an alternative that can increase expressively the order of generated combinations and reduce the growing of the network.

4. Our approach for building CNM network

This section presents our approach to generate the CNM, that may reduce the cost of the algorithm in terms of space, and that we call *parsimonious generation of CNM network*. By this approach, the neurons and the connections are created only by contingence, i.e., only when required by an example in the training set. Moreover, during the training phase, it is only computed the rewarding of the arcs arriving at the correct class. There is no punishment. The computation of the effect of misclassifications is done by calculating the difference between the value of each accumulator at the end of the training, for each combination, and the value of the other accumulators for the same combination related to different classes.

Let us take the example of the training set used in the original proposal [MAC 89], shown in TABLE 4.1.

TABLE 4.1 - Patients with diseases and associated symptoms

Name	Symptoms	Disease
John	s1, s2, s3	d1
Diana	s1, s2, s4	d1
Mary	s1, s3, s4	d2
Peter	s2, s3, s4	d2

TABLE 4.2 - Effects of training and prunning of the CNM

	Symptoms					Accumulators				Thrshld/ Prun.	
Disease	s1	s2	s3	s4	Begin	Jo	Di	Ma	Pe	1	2
d1	X				0	1	2	1	1	1	-
		X			0	1	2	2	1	1	-
			X		0	1	1	0	-1	-	-
				X	0	0	1	0	-1	-	-
	X	X			0	1	2	2	2	2	2
	X		X		0	1	1	0	0	-	-
	X			X	0	0	1	0	0	-	-
		X	X		0	1	1	1	0	-	-
		X		X	0	0	1	1	0	-	-
			X	X	0	0	0	-1	-2	-	-
	X	X	X		0	1	1	1	1	1	-
	X	X		X	0	0	1	1	1	1	-
	X		X	X	0	0	0	-1	-1	-	-
		X	X	X	0	0	0	0	-1	-	-
d2	X				0	-1	-2	-1	-1	-	-
		X			0	-1	-2	-2	-1	-	-
			X		0	-1	-1	0	1	1	-
				X	0	0	-1	0	1	1	-
	X	X			0	-1	-2	-2	-2	-	-
	X		X		0	-1	-1	0	0	-	-
	X			X	0	0	-1	0	0	-	-
		X	X		0	-1	-1	-1	0	-	-
		X		X	0	0	-1	-1	0	-	-
			X	X	0	0	0	1	2	2	2
	X	X	X		0	-1	-1	-1	-1	-	-
	X	X		X	0	0	-1	-1	-1	-	-
	X		X	X	0	0	0	1	1	1	-
		X	X	X	0	0	0	0	1	1	-

Jo=John, Di=Diana, Ma=Mary, Pe=Peter

In the original approach, the expansion of the network based on this training set produces the combinations shown in TABLE 4.2. During the initial phase - creation of the empty network – twenty eight combinations were generated. After prunning with threshold one, ten combinations remain, and with two as threshold, two remain. Using our approach, according to TABLE 4.3, twenty two combinations are generated, and the same quantities remain - ten and two - after the prunning with threshold one and two, respectively.

The algorithm proposed for generation of the CNM takes the following form:

- **For each** example in the training set, **do**:
  Compute all possible combinations based on the example
  For each computed combination, **do:**
    If there is an equal combination in the network arriving to the same class

Then *add* one to the accumulator of the arc arriving (Reward)

Else *include* an arc corresponding to the actual combination, setting the accumulator to one (Reward)

To compute the final value of the accumulators, the result of the folllowing operations is taken: for each accumulator of each combination, take its value as ACC; for all combinations equal to the precedent one pointing to classes different from the precedent, sum their accumulators, calling it SUM; the final result of ACC is given by ACC = ACC - SUM. It is equivalent to punishments of the original algorithm, in only one passing. Both training and accomplishment of the final value of accumulators are easily traced through TABLES 4.1 and 4.3.

TABLE 4.3 - Parsimonious generation of CNM

	Symptoms					Accumulators					Thrshd/Prun	
Disease	s1	s2	s3	s4	Begin	Jo	Di	Ma	Pe	Acc	1	2
d1	X				0	1	2			1	1	-
		X			0	1	2			1	1	-
			X		0	1	1			-1	-	-
	X	X			0	1	2			2	2	2
	X		X		0	1	1			0	-	-
		X	X		0	1	1			0	-	-
	X	X	X		0	1	1			1	1	-
				X	0		1			-1	-	-
	X			X	0		1			0	-	-
		X		X	0		1			0	-	-
	X	X		X	0		1			1	1	-
d2	X				0			1	1	-1	-	-
			X		0			1	2	1	1	-
				X	0			1	2	1	1	-
	X		X		0			1	1	0	-	-
	X			X	0			1	1	0	-	-
			X	X	0			1	2	2	2	2
	X		X	X	0			1	1	1	1	-
		X			0				1	-1	-	-
		X	X		0				1	0	-	-
		X		X	0				1	0	-	-
		X	X	X	0				1	1	1	-

5. Example of parsimonious generation

In this example we use data related to the use of pesticides in São Paulo[*], during 1994. Training is accomplished over attributes city, crop, disease, pesticide, and quantity, described below:

[*] Data obtained by agreement between EMBRAPA Environment and CREA-SP.

city: Code of the city where the pesticide was applied. There are 120 cities.

crop: Code of the crop that received the pesticide. There are 27 crops.

disease: Code of the disease being treated. There are 55 diseases.

pesticide: Code of the pesticide applied. There are 140 pesticides.

quantity: It is the target attribute and indicates pesticides level applied in one city. Domain={High, Medium, Low}.

According to the original version of CNM, disregarding combinations between different values from the same attribute, combinations shown in TABLE 5.1 are generated.

TABLE 5.1 - Generated combinations for each hypothesis through CNM

Combinations of 2 attributes:	
city and *crop:*	3,240
city and *disease:*	6,600
city and *pesticide:*	16,800
crop and *disease:*	1,485
crop and *pesticide:*	3,780
disease and pesticide:	7,700
Combinations of 3 attributes:	
city, crop and disease:	178,200
city, crop and *pesticide:*	453,600
city, disease and pesticide:	924,000
crop, disease and pesticide:	207,900
Combinations of 4 attributes:	
city, crop, disease and pesticide:	24,948,000
Total of combinations for each hypothesis:	26,751,305

Considering that we have 3 hypotheses, the total amount of generated combinations, with empty network, is 80,253,915. The parsimonious generation of the network, with the same training set, produced only 5,152 combinations, representing a drastic redution on the number of generated combinations. FIGURE 5.1 shows the summary of the training, listing total combinations generated. In other training sets this gain may be lower, but it is possible that in almost all cases a considerable gain will be obtained.

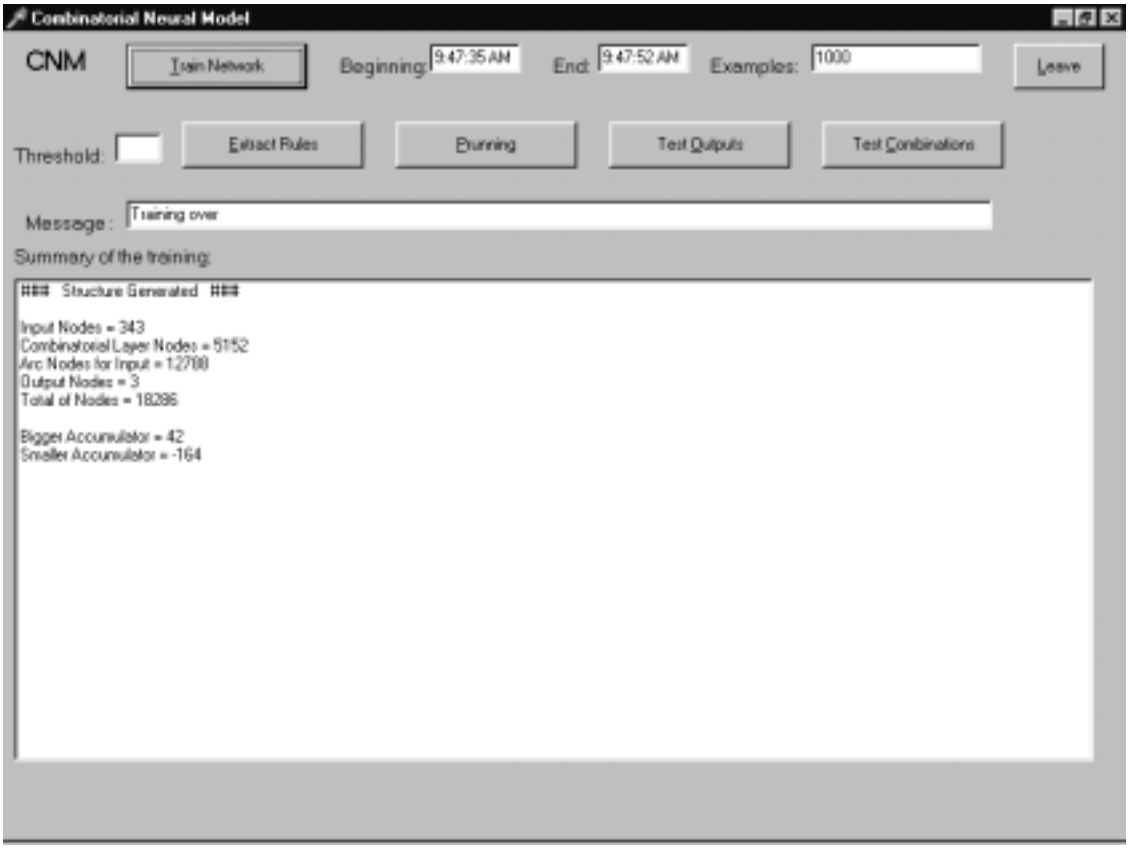

FIGURE 5.1 - Partial outputs of the parsimonious generation of CNM

6. Accuracy tuning

As seen in **Section 2**, the prunning of the generated network is done by means of the accumulators values, what does not give a precise idea of the resulting accuracy of the model. By prunning the network using accumulators values, one have to try many times, prunning and testing, until reach an acceptable accuracy, what represents a hard task. In this section we present a model that uses an implementation of wrapper algorithm for accuracy tuning of CNM without necessity of this tryings. Examples of successful use of wrapper method are the attribute subset selection algorithm presented by John [JOH 97] and the algorithm for parameter selection on ISAAC system [TAL 98]. In our application the algorithm involves the trained network and uses an off-set training dataset as input for test diferent values of accumulators for prunning. This model may be easily extended to incorporate more interesting validation methods, such as *n*-fold cross validation. FIGURE 6.1 shows how the model carries out the accuracy tuning.

Considering that the values of the accumulators vary from –T to T, being T the number of cases present in the training set, the user have to specify the step to be used on the first function of the wrapper: the generation of prunning points. For this purpose, the system shows the screen depicted on FIGURE 6.2.

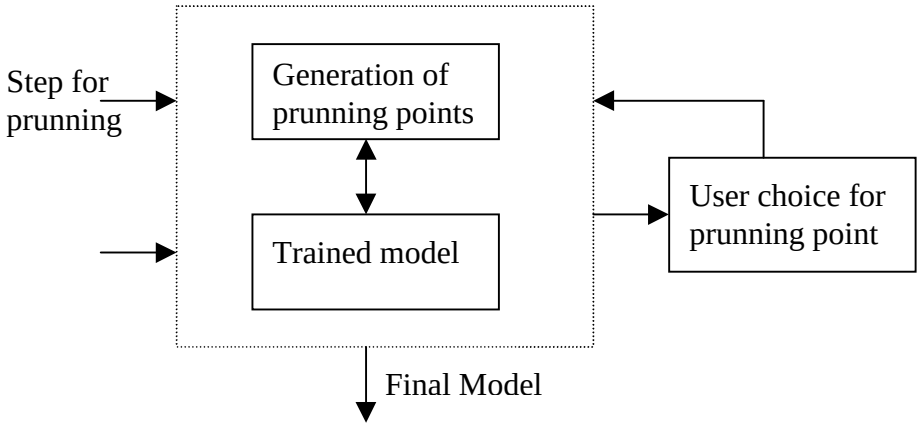

FIGURE 6.1 – The wrapper method for accuracy tuning

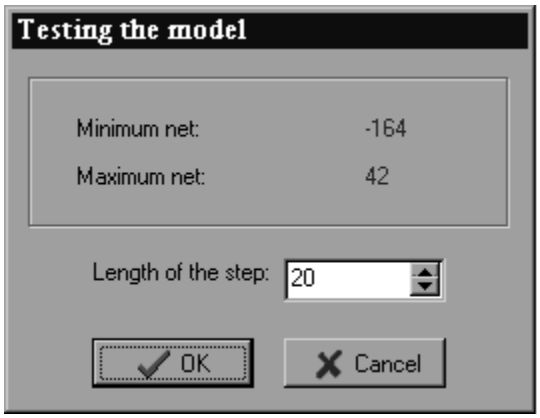

FIGURE 6.2 – Specifying the step for prunning

Using the specified step, the algorithm divides the interval defined between minimum and maximum accumulators in equal parts (with a possible exception on the last part); after this definition the user submit the test set to the trained network. It is repeated for each step and the results reported in a graphical form, as depicted in FIGURE 6.3. One additional feature is the possibility to focus the attention on a particular interval (like a zoom effect), specifying a shorter step to refine the accuracy tuning.

On the *x* axis it is the prunning points and on the *y* axis it is shown the results on each prunning point. The user may choose one prunning point according to the values on the bottom of the screen. In the example, it is selected the prunning point eight. Having decided where to prune the network the user order the prunning, obtaining the final network with the desired accuracy.

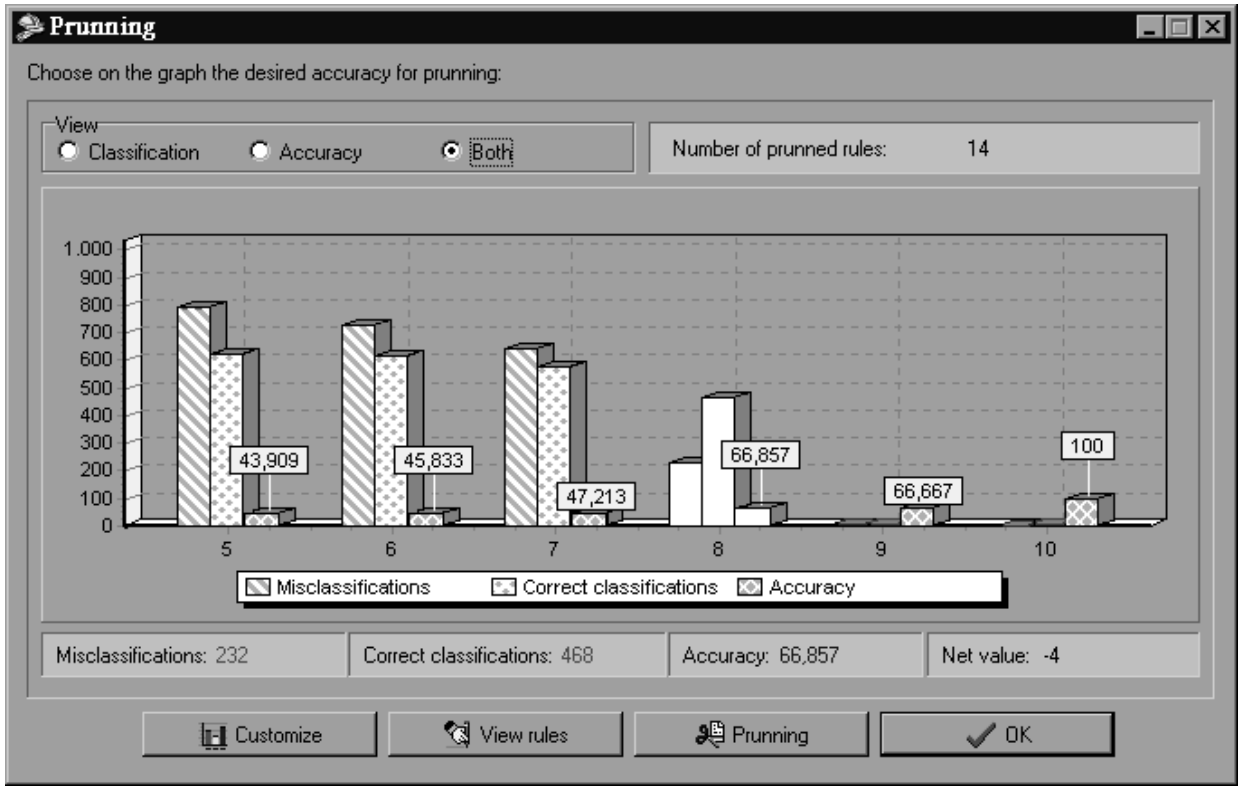

FIGURE 6.3 – Prunning points with corresponding accuracy

7. Conclusions

CNM has received important improvements in order to be more adequate to data mining. Some improvements has focussed on optimizations of processing time [BEC 98] and scalability of the algorithm to larger databases [PRA 98]. Another important issue that has been faced is the development of facilities to help one in accuracy tuning [PRA 99]. Wrapper method has shown to be useful to set parameters of a model using its own learning algorithm, as in the present case.

Our contributions with the present work is: (a) a cheaper alternative to generate the CNM network and (b) the use of wrapper to help the user to overcome the laborious task of determining the prunning point of the combinatorial neural model. The main advantages reached are:

- It is never created unnecessary arcs in the network; this fact leads to the generation of trained networks smaller than the original proposal [MAC 91] and other approachs [LEA 93] and [FEL 97]. On the other hand, the final network obtained after prunning phase is the same in all alternatives. The main problem of space occurs in the training phase and our approach reduces this problem. During training phase, gain in terms of space, provided by this proposal, presents a compensation through cost increase to compute the final value of accumulators, since it is necessary to identify all equal combinations for different hypotheses. However, such way of building the network may be considered an alternative when the main restriction is space. The space complexity of this proposal will be, at worst case, equal to that of the original one. In other words, if

all possible combinations are associated to all possible hypotheses in the training set, the required space to build the CNM in both alternatives, will be the same. In any other situations, the present proposal will generate a smaller network.

- By using wrapper the search for the most accurate model has become easier than by trying many alternatives of prunning. Although the effectiveness of this way of prunning, we believe that this approach must be improved by allowing the use of better methods for validation as *n*-fold cross validation. By using such validation methods, one can search for an optimum trade-off between the number of remaining rules after prunning and the accuracy level of the model.

Bibliographical references:

[BEC 98] BECKENKAMP, F. G.; FELDENS, M. A., PREE, W. Optimizations of the Combinatorial Neural Model. IN: *Vth Brazilian Symposium on Neural Networks (SBRN'98),* Belo Horizonte, Brazil, 1998.

[FEL 97] FELDENS, M. A. **Knowledge Discovery from Databases Engineering:** Survey and Application to Health Care Domain**.** Porto Alegre: CPGCC da UFRGS, Brazil 1997. Master degree dissertation.

[FRE92] FREEMAN, James A.; SKAPURA, David M. Adaptive Resonance Theory. In: **Neural Networks, Algorithms, Applications, and Program Techniques.** Reading: Addison-Wesley, 1992. 401p. p.292-339.

[GUA 94] GUAZZELLI, A. **Aprendizagem em Sistemas Híbridos.** Porto Alegre: CPGCC da UFRGS, 1994. Dissertação de mestrado.

[HUD 92] HUDSON, D. L. *et al.* Medical diagnosis and treatment plans derived from a hybrid expert system. In: KANDEL, A. & LANGHOLZ, G**. Hybrid architectures for intelligent systems**, Boca Raton, FL: CRC Press, 1992.

[JOH 97] JOHN, G. H. **Enhancements to the Data Mining Process**. Stanford, EUA: Stanford University, 1997. Ph.D. Thesis.

[KNA 92] KNAUS, R. Representing expert knowledge in neural nets. In: KANDEL, A. & LANGHOLZ G. **Hybrids Architectures for Intelligent Systems**, Boca Raton, FL: CRC Press, 1992.

[LEA 93] LEÃO, B. F.; REÁTEGUI, E. B. A hybrid connectionist expert system to solve classification problems. **Proceedings of Computer in Cardiology,** IEEE Computer, IEEE Computer Society, London, 1993.

[MAC 89] MACHADO, R. J.; ROCHA, A. F. **Handling knowledge in high order neural networks: the combinatorial neural network.** Rio de Janeiro: IBM Rio Scientific Center, 1989. (Technical Report CCR076) .

[MAC 91] MACHADO, R. J. & ROCHA, A. F. da. The combinatorial neural network: a conectionist model for knowledge based systems. In: BOUCHON, B.;

YAGER, R. R.; ZADEH, L. A. **Uncertainty in Knowledge Bases**, Berlin, Germany: Springer Verlag, 1991, p.578-587.

[MAC 92] MACHADO, R. J. & ROCHA, A. F. da. **Evolutive fuzzy neural networks**. In: IEEE International Conference on Fuzzy Systems, San Diego, CA, p.493-500. Mar.1992.

[MAC 92a] MACHADO, R. J.; ROCHA, A. F. da. A Hybrid architecture for fuzzy connectionist expert systems. In: KANDEL A. & LANGHOLZ G. **Hybrids Architectures for Intelligent Systems**, Boca Raton, FL: CRC Press, 1992. p. 136-152.

[MAC 97] MACHADO, R. J.; ROCHA, A. F. da. Inference, inquiry, evidence censorship, and explanation in connectionist expert systems. **IEEE Transactions on Fuzzy Systems**, New York, v. 5, n. 3, p.443-459, Aug. 1997.

[MAC 98] MACHADO, R. J.; CARNEIRO, W.; NEVES, P. A. Learning in the combinatorial neural model, IEEE Transactions on Neural Networks, v.9, p.831-847. Sep.1998.

[MIL 56] MILLER, G. A., The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. **Psycological Review**, [s.l.], 63, p.81-97, 1956.

[PRA 98] PRADO, H. A.; FRIGERI, S. R.; ENGEL, P. M. A Parsimonious Generation of Combinatorial Neural Model. IN: *IV Congreso Argentino de Ciencias de la Computación (CACIC'98)*, Neuquén, Argentina, 1998.

[PRA 99] PRADO, H. A.; FRIGERI, S. R.; MACHADO, K. F.; ENGEL, P. M. Accuracy Tuning on Combinatorial Neural Model. IN: *PAKDD'99 Third Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Beijin, China, 1998. (submitted).

[TAL 98] TALAVERA, L. Exploring Efficient Attribute Prediction in Hierarchical Clustering. IN: *VII Congreso Iberoamericano de Inteligencia Artificial, IBERAMIA98*, Lisbon, Portugal, 1998.

Anexo 4 Explaining Cluster's Structures by Integrating Unsupervised and Supervised Learning Algorithms

[PRA 2000a] PRADO, H.A. do Explaining Cluster's Structures by Integrating Unsupervised and Supervised Learning Algorithms. In: IEEE WORKSHOP ON DB & IS, 4., 2000, Vilnius, Lituânia. **Proceedings…** Vilnius: Lithuanian Computer Society, 2000.

Explaining Clusters' Structures by Integrating Unsupervised and Supervised Learning Algorithms

Hércules Antonio do Prado

Brazilian Enterprise for Agricultural Research, Brasilia, Brazil
Catholic University of Brasilia, Brasilia, Brazil
Informatics Institute, Federal University of Rio Grande do Sul, Porto Alegre, RS, Brazil

prado@inf.ufrgs.br

Abstract

Motivated by the discovery process described by Langley [3], and by the advances in Knowledge Discovery and Data Mining, we propose a framework to obtain extensional and intensional descriptions from unclassified data.

***Keywords*:** Knowledge Discovery, Data Mining, Machine Learning.

1. Introduction

Knowledge discovery from unlabeled data comprises two main tasks: identification of "natural groups" and analysis of these groups in order to interpret their meaning. These tasks are accomplished by unsupervised and supervised learning, respectively, that correspond to the taxonomy and explanation phases of the discovery process described by Langley [3]. Research on Knowledge Discovery from Databases (KDD) has addressed these two processes into two main dimensions: (1) scaling up the learning algorithms to very large databases and (2) making easier the whole process of knowledge discovery. The second dimension refers to tasks like data gathering, cleaning, and so on. In this paper we argue that the advances achieved in unsupervised and supervised learning under the flag of KDD allow us to combine these two processes in just one model, providing extensional and intensional descriptions of unlabeled data. Extensional description means to find out which data points belong to each cluster and intensional description corresponds to specifying which features are more correlated to each cluster. To explore this idea we present a framework to integrate different unsupervised and supervised learning.

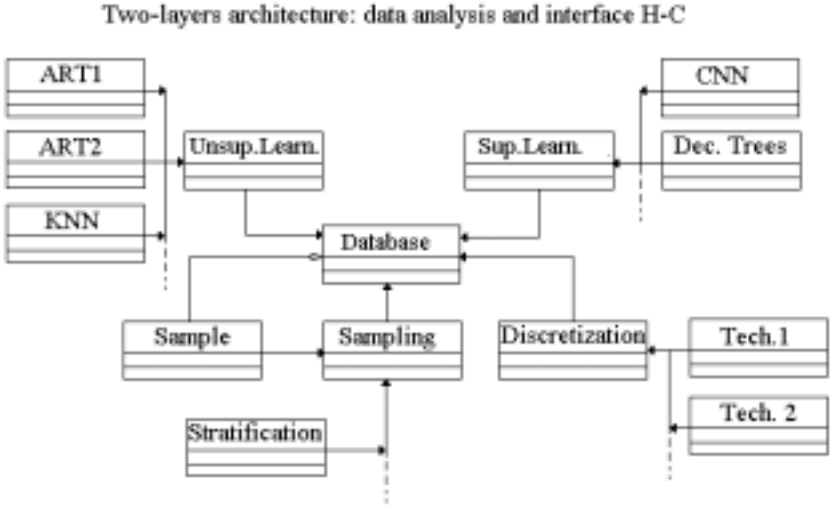

Figure 1. Framework for integration of unsupervised and supervised learning

2. Framework description

The framework, as shown in Figure 1, has the following main components: (1) **Database**: representing the data set for analysis; this data set can also be considered as an aggregation of samples: (2) **Sample**: subset of database that can be used according to the analyst convenience; (3) **Unsupervised learning** and **Supervised learning**: abstract classes to hold the unsupervised and supervised learning algorithms in a standard representation; (4) **Sampling**: holds the alternatives to obtain samples; and (5) **Discretization**: holds the alternatives for discretizing the data.

We instantiate this framework with two artificial neural networks: ART1 network [2] and the Combinatorial Neural Model (CNM) ([4]). Both models accomplish one important *desiderata* [1] for data mining, learning in just one pass of the entire database. Moreover, CNM, the intensional part of the architecture, allows one to obtain rules directly from its representation. The framework instantiated for ART1 and CNM are shown in Figure 2.

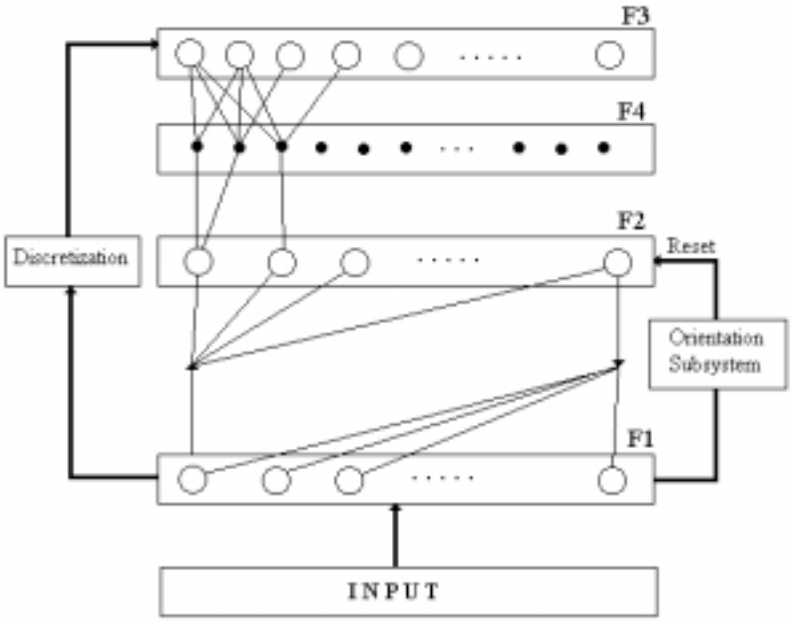

Figure 2. Framework instantiated with ART1 and CNM

Layers INPUT, F1, and F2 refer to ART1, layers F3, F4 and F2 refer to CNM.

3. References

[1] Bradley, P.; Fayyad, U. & Reina, C. Scaling Clustering Algorithms to Large Databases. In: Proceedings of Fourth International Conference on Knowledge Discovery and Data mining. New York, NY: AIII Press, 1998.

[2] Carpenter, G. & Grossberg, S. Neural Dynamics of Category Learning and Recognition: Attention, memory, Consolidation, and Amnesia. In: Joel L Davis (ed.), Brain Structure, learning, and memory. AAAS Symposia Series, Boulder, CO: Westview Press, 1988. Pp.233−287.

[3] Langley, P. The Computer−Aided Discovery of Scientific Knowledge. In: Proceedings of the First International Conference on Discovery Science, Fukuoka, Japan, 1998.

[4] Machado, R. J. & Rocha, A. F. Handling knowledge in high order neural networks: the combinatorial neural network. Rio de Janeiro: IBM Rio Scientific Center, Brazil, 1989. (Technical Report CCR076).

Anexo 5 Scalable Model for Extensional and Intensional Descriptions of Unclassified Data

[PRA 2000b] PRADO, H.A. do; HIRTLE, S.C.; ENGEL, P.M. Scalable Model for Extensional and Intensional Descriptions of Unclassified Data. In: IPDPS WORKSHOP ON HIGH PERFORMANCE DATA MINING, 3., 2000, Cancún, México. **Proceedings…** Berlin: Springer-Verlag, 2000. (Lecture Notes in Computer Science, v. 1800).

Scalable Model for Extensional and Intensional Descriptions of Unclassified Data

Hércules A. Prado[1,2,*], Stephen C. Hirtle[2,**], Paulo M. Engel[1,***]

[1] Universidade Federal do Rio Grande do Sul
Instituto de Informática
Av. Bento Gonçalves, 9500 - Bairro Agronomia
Porto Alegre / RS - Brasil
Caixa Postal 15.064 - CEP 91.501-970
Fone: +55(051)316-6829
Fax: +55(051)319-1576
[2] University of Pittsburgh
Department of Information Sciences and Telecommunications
135 North Bellefield Ave. Pittsburgh, PA 15.260
Phone: +1(412)624-9434
Fax: +1(412)624-2788
{prado, engel}@inf.ufrgs.br and hirtle+@pitt.edu

Resumo Knowledge discovery from unlabeled data comprises two main tasks: identification of ”natural groups” and analysis of these groups in order to interpret their meaning. These tasks are accomplished by unsupervised and supervised learning, respectively, and correspond to the taxonomy and explanation phases of the discovery process described by Langley [9]. The efforts of Knowledge Discovery from Databases (KDD) research field has addressed these two processes into two main dimensions: (1) scaling up the learning algorithms to very large databases, and (2) improving the efficiency of the knowledge discovery process. In this paper we argue that the advances achieved in scaling up supervised and unsupervised learning algorithms allow us to combine these two processes in just one model, providing extensional (who belongs to each group) and intensional (what features best describe each group) descriptions of unlabeled data. To explore this idea we present an artificial neural network (ANN) architecture, using as building blocks two well-know models: the ART1 network, from the Adaptive Resonance Theory family of ANNs [4], and the Combinatorial Neural Model (CNM), proposed by Machado ([11] and [12])). Both models satisfy one important desiderata for data mining, learning in just one pass of the database. Moreover, CNM, the intensional part of the architecture, allows one to obtain rules directly from its structure. These rules represent the insights on the groups. The architecture can be extended to other supervised/unsupervised learning algorithms that comply with the same desiderata.

* Researcher at EMBRAPA — Brazilian Enterprise for Agricultural Research and lecturer at Catholic University of Brasília (Supported by CAPES - Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, grant nr. BEX1041/98-3)

** Professor at University of Pittsburgh

*** Professor at Federal University of Rio Grande do Sul

1 Introduction

Research in Knowledge Discovery from Databases (KDD) has developed along two main dimensions: (1) improving the knowledge discovery process, and (2) scaling up this same process to very large databases. Machine Learning, as an important field related to KDD, is founded on three principles [19]:

1. Modeling of cognitive processes, aiming to select characteristics of interest to be formalized as knowledge;
2. Computer science, which offers a formalism to support the descriptions of those characteristics, as well as providing approaches to evaluate the degree of computational difficulty of the issues involved; and
3. Applications, where one departs from practical needs to the implementation of systems.

In this article, we depart from a characterization of the concept formation activity as a cognitive process, proposing a computational approach to support this activity, from the point of view of KDD. We take the concept of performance as given by the relation functionality/resources applied. By this way, we present a model where funcionality is increased while the resources applied are just slightly changed.

2 Motivation

According to Wrobel [19], a concept is ”a generalized description of sets of objects”. In this sense, Easterlin and Langley [5] analyse the concept formation process as follows:

1. Given a set of objects (instances, events, cases) descriptions, usually presented incrementally;
2. find sets of objects that can be grouped together (aggregation), and
3. find intensional description of these sets of objects (characterization).

Murphy and Medin [14] discuss two hypothesis that constrain the way objects are grouped in concepts:

1. Similarity hypothesis: this hypothesis sustain that what defines a class is that its members are similar to each other and not similar to members of other classes.
2. Correlated attribute hypothesis: this hypothesis states that ”natural groups” are described according to clusters of features and that categories reflect the cluster structure of correlations.

The first hypothesis presents a problem that is: the similarity criteria must be applied to a pre-defined set of features and the definition of this set is affected by the previous knowledge one has over the objects. However, when just a small knowledge about the data exists, this criteria is used as a first approximation. Over this approximation, the correlated attribute hypothesis is applied. In a broad sense, what is desirable in this process is to provoke the mental operations that can lead to a problem solution ([16] and [15]). Actually, since it seems that the discovery process, as a rule, requires the human judgment [9], it is useful to leave available to the analyst all relevant information to evaluate both hypothesis when searching for the classes’ structures.

3 Proposed Architecture

The research on the KDD realm has emphasized improvements in the processes of supervised and unsupervised learning. More recently, many unsupervised learning algorithms have been scaled up according to the desiderata proposed by Agrawal *et al.* [1] for this kind of learning algorithm. Considering these advances and the ones in supervised learning, we believe there is enough room to scale up the combined process of unsupervised and supervised learning in order to obtain better descriptions of unclassified data. By ”better descriptions” we mean obtaining intensional (what are the main characteristics of each class) descriptions, beyond the extensional (what objects are members of each class) ones, usually provided. We explore our idea with a hybrid architecture, based into two well-know models: ART1 [4], used for cluster binary data, and CNM ([11] and [12]), used to map the input space in the formed classes. Both ANNs present an important characteristic to support data mining: they learn in just one pass of the entire data set. The model is illustrated by Figure 1.

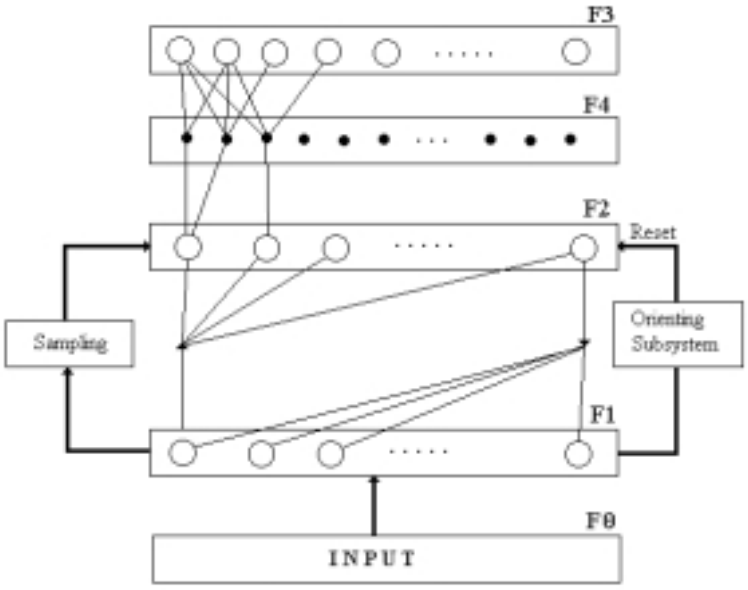

Figura1. Describing unclassified data

The architecture is composed by five layers according to the schema: Input layer (F0, where the examples are introduced in the architecure); Aggregation module (F1 and F2, where the classes are defined); Characterization module (F2, F3 , and F4, where the classes are explained).

For the characterization module, it requires the pre-existence of classes that would not be available when the process starts. We overcome this problem by creating the

classes by means of a sampling subsystem. The creation of classes through sampling was explored by Guha [8] with consistent results. As a consequence of the sampling process, a complete execution of the system will take more than one pass of the input data set. Considering γ the size of the sample and **D** the size of the input data set, a complete execution of the system will take, precisely, $\mathbf{D} + \gamma$ records. In the next two sections, we describe each model used as building blocks for our architecture.

4 ART1 Neural Network

ART1 (F1 and F2 layers) is a member of the so-called ART family, that stands by Adaptive Resonance Theory [10], [3], [4] and [6].

ART1 is a competitive recurrent network with two layers, the input layer and the clustering layer. This network was developed to overcome the plasticity-stability dilemma [7], allowing an incremental learning, with a continuous updating of the clusters prototypes, and preserving the previously stored patterns. The clustering algorithm proceeds, in general steps, as folows: (a) the first input is selected to be the first cluster; (b) each next input is compared with each existing cluster; the first cluster where the distance to the input is less then a threshold is chosen to cluster the input. Otherwise, the input defines a new cluster. It can be observed that the number of clusters depends on the threshold and the distance metric used to compare the inputs with the clusters. For each input pattern presented to the network, one output unit is declared winner (at the first pattern, the own input pattern defines the cluster). The winner backpropagates a signal that encodes the expected pattern template. If the current input pattern differs more than a defined threshold from the backpropagated signal, the winner are temporarily disabled (by the Orienting System) and the next closest unit is declared winner. The process continues until an output unit become a winner, considering the threshold. If no one of the output units become a winner, a new output unit is defined to cluster the input pattern. Graphically, an ART1 network can be illustrated by Figure 2, where it appears with four input and six ouput neurons. t_{ij} and b_{ij} are, respectively, bottom-up and top-down connections.

One important characteristic of this ANN is that it works in just one pass, what is interesting when we are processing a huge amount of data. The training algorithm for this network is the following:

- **Step1. Initialization**: The bottom-up $b_{ij}(t)$ and top-down $t_{ij}(t)$ weight connection between input node i and output node j at time t are set up. The fraction ρ (vigilance parameter) is defined, indicating how close an input must be to a stored exemplar to match. Initialize N and M, numbers of input and output nodes.
- **Step 2. Apply New Input**
- **Step 3. Compute Matching Scores**: The bottom-up weights are applied to the input pattern, generating the output signal: μ_j.
- **Step 4. Select Best Matching Exemplar**: $\mu_j^* = \max\{\mu_j\}$ is taken as the best exemplar.
- **Step 5. Vigilance Test**: The best exemplar and the input pattern are compared, according to ρ. If the distance is acceptable, the control flows to **Step 7**, otherwise **Step 6** proceeds.

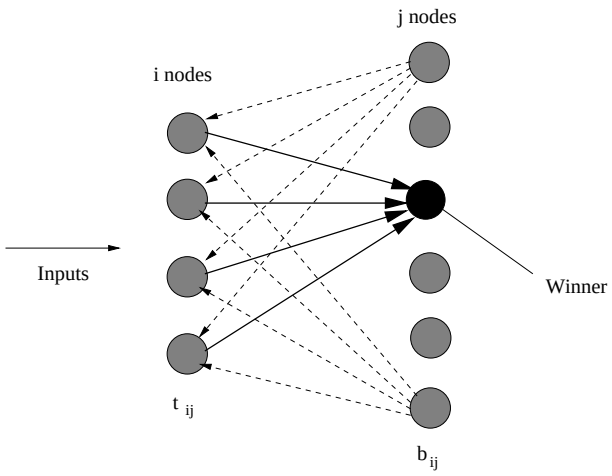

Figura2. Architecture of an ART network [3]

- **Step 6. Disable Best Matching Exemplar**: The ouput of the best matching node selected in Step4 is temporarily set to zero and no longer takes part in the maximization of Step4. Then go to Step 3.
- **Step 7. Adapt Best Matching Exemplar**:

$$t_{ij_*}(t+1) = t_{ij_*}(t)t(x_i)$$
$$b_{ij_*}(t+1) = \frac{t_{ij_*}(t)x_i}{\frac{1}{2} + \sum_{i=0}^{N-1} t_{ij_*}(t)x_i}$$

- **Step 8. Repeat by Going to Step 2**: First enable any node disabled in Step 6.

5 Combinatorial Neural Model (CNM)

CNM (F2, F3, and F4 layers) is a hybrid architecture for intelligent systems that integrates symbolic and connectionist computational paradigms. This model is able to recognize regularities from high-dimensional symbolic data, performing mappings from this input space to a lower dimensional output space. Like ART1, this ANN also overcomes the plasticity-stability dilemma [7].

The CNM uses supervised learning and a feedforward topology with: one input layer, one hidden layer - here called combinatorial - and one output layer (Figure 3). Each neuron in the input layer corresponds to a concept - a complete idea about an object of the domain, expressed in an object-attribute-value form. They represent the evidences of the domain application. On the combinatorial layer there are aggregative fuzzy AND neurons, each one connected to one or more neurons of the input layer by arcs with adjustable weights. The output layer contains one aggregative fuzzy OR neuron for each possible class (also called hypothesis), linked to one or more neurons on the combinatorial layer. The synapses may be excitatory or inhibitory and they are

characterized by a strength value (weight) between zero (not connected) to one (fully connected synapses).

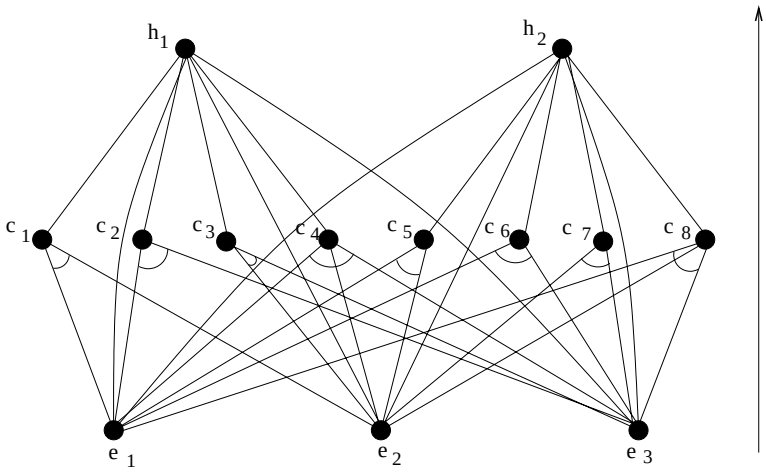

Figura3. Complete version of CNM for 3 input evidences and 2 hypotheses [11]

The network is created completely uncommited, according to the following steps: (a) one neuron in the input layer for each evidence in the training set; (b) a neuron in the output layer for each class in the training set; and (c) for each neuron in the output layer, there is a complete set of hidden neurons in the combinatorial layer which corresponds to all possible combinations (length between two and nine) of connections with the input layer. There is no neuron in the combinatorial layer for single connections. In this case, input neurons are connected directly to the hypotheses.

The learning mechanism works in only one iteration, and it is described bellow:

PUNISHMENT_AND_REWARD_LEARNING_RULE

- **Set** to each arc of the network an accumulator with initial value zero;
- **For each** example case from the training data base, **do**:
  - *Propagate* the evidence beliefs from input nodes until the hypotheses layer;
  - **For each** arc reaching a hypothesis node, **do**:
    - **If** the reached hypothesis node corresponds to the correct class of the case
    - **Then** *backpropagate* from this node until input nodes, increasing the accumulator of each traversed arc by its evidencial flow (Reward)
    - **Else** *backpropagate* from the hypothesis node until input nodes, decreasing the accumulator of each traversed arc by its evidencial flow (Punishment).

After training, the value of accumulators associated to each arc arriving to the output layer will be between [-T, T], where T is the number of cases present in the training set. The last step is the prunning of network; it is performed by the following actions: (a) remove all arcs whose accumulator is lower than a threshold (specified by a specialist); (b) remove all neurons from the input and combinatorial layers that became disconnected from all hypotheses in the output layer; and (c) make weights of the arcs

arriving at the output layer equal to the value obtained by dividing the arc accumulators by the largest arc accumulator value in the network. After this pruning, the network becomes operational for classification tasks. This ANN has been applied with success in data mining tasks ([2], [17], and [18]).

6 Ongoing Work

This paper presents an architecture to scale up the whole process of concept formation according two main constraints: the identification of groups composed by similar objects and the description of this groups by the higher correlated features. The ongoing work includes the implementation and evaluation of this architecture, instantiated to ART1 and CNM, and its extension to cope with continuous data.

Referências

1. Agrawal, R., Gehrke, J., Gunopulos, D., Raghanavan, P. Automatic Subspace Clustering of High-Dimensional Data for Data Mining Applicati ons. In: Proceedings of ACM SIGMOD98 International Conference on Management of Data, Seattle, Washington, 1998.
2. Beckenkamp, F. G., Feldens, M. A., Pree, W.: Optimizations of the Combinatorial Neural Model. IN: Vth Brazilian Symposium on Neural Networks. (SBRN'98), Belo Horizonte, Brazil.
3. Bigus, J. P. Data Mining with Neural Networks. [S.l.]: McGraw-Hill, 1996. p.3-42.
4. Carpenter, G. and Grossberg, S. Neural Dynamics of Category Learning and Recognition: Attention, Memory, Consolidation, and Amnesia. In: Joel L. Davis (ed.), Brain structure, learning, and memory. AAAS Symposia Series, Boulder, CO: Westview Press, 1988. p.233-287.
5. Easterlin, J.D., Langley, P.: A Framework for Concept Formation. In: Seventh Annual Conference of the Cognitive Science Society, Irvine, CA, 1985.
6. Engel, P. M. Lecture Notes. Universidade Federal do Rio Grande do Sul. Porto Alegre-RS, Brazil: CPGCC da UFRGS, 1997.
7. Freeman, J. A., Skapura, D. M.: Neural Networks, Algorithms, Applications, and Program Techniques. [S.l.]: Addison-Wesley, 1992. p.292-339.
8. Guha, S., Rastogi, R., Shim, K. Cure: An Efficient Clustering Algorithm for Larg e Databases. In: Proceedings of ACM SIGMOD98 International Conference on Managemen t of Data, Seattle, Washington, 1998.
9. Langley, P. The Computer-Aided Discovery of Scientific Knowledge. In: Proc. of the First International Conference on Discovery Science, Fukuoka, Japan, 1998.
10. Lippmann, D. An Introduction to Computing with Neural Nets, IEEE ASSP Magazine. April, 1987.
11. Machado, R. J., Rocha, A. F.: Handling knowledge in high order neural networks: the combinatorial neural network. Rio de Janeiro: IBM Rio Scientific Center, Brazil, 1989. (Technical Report CCR076).
12. Machado, R. J., Carneiro, W., Neves, P. A.: Learning in the combinatorial neural model, IEEE Transactions on Neural Networks, v.9, p.831-847. Sep.1998.
13. Medin, D., Altom, M.W., Edelson, S.M. and Freko, D. Correlated symptoms and simulated medical classification. Journal of Experimental Psychology: Learning, Memory and Cognition, 8:37-50, 1983.
14. Murphy, G. and Medin, D.: The Role of Theories in Conceptual Coherence. Psychological Review, 92(3):289-316, July, 1985.

15. Pereira, W. C. de A. Resolução de Problemas Criativos: Ativação da Capacidade de Pensar. Departamento de Informação e Documentação/EMBRAPA, Brasília-DF, 1980. 54pp.
16. Polya, G. How to Solve It: A New Aspect of Mathematical Method. Princeton: Princeton University Press, 1972. 253pp.
17. Prado, H. A., Frigeri, S. R., Engel, P. M.: A Parsimonious Generation of Combinatorial Neural Model. IN: IV Congreso Argentino de Ciencias de la Computación (CACIC'98), Neuquén, Argentina, 1998.
18. Prado, H. A. do; Machado, K.F.; Frigeri, S. R.; Engel, P. M. Accuracy Tuning in Combinatorial Neural Model. PAKDD'99 - Pacific-Asia Conference on Knowledge Discovery and Data Mining. Proceedings ... Beijing, China, 1999
19. Wrobel, S. Concept Formation and Knowledge Revision. Dordrecht, The Netherlands: Kluwer, 1994. 240pp.

Anexo 6 Alleviating the Complexity of the Combinatorial Neural Model Using a Committee Machine

[PRA 2000c] PRADO, H.A. do; MACHADO, K.F.; ENGEL, P.M. Alleviating the complexity of the Combinatorial Neural Model using a committee machine. In: N. EBECKEN; C. A. BREBBIA (Ed.) **Data Mining II**. Southampton: WIT Press, 2000.

Alleviating the complexity of the Combinatorial Neural Model using a committee machine

H. A. do Prado[1,2,3], K. F. Machado[1,4] & P. M. Engel[1]
[1]*Federal University of Rio Grande do Sul, Brazil*
[2]*Brazilian Enterprise for Agricultural Research, Brazil*
[3]*Catholic University of Brasília, Brazil*
[4]*Justice Court of State of Rio Grande do Sul, Brazil*

Abstract

Knowledge Discovery from Databases (KDD) can be seen as a set of computer-aided knowledge discovery techniques scaled up to very large databases. By this way, the old process of discovery has experienced amazing improvements by: (a) allowing well-known Machine Learning and Statistical algorithms run for larger data sets with good performance; and (b) making easier tasks like data gathering and cleansing, parameter and model selection, and so on. In this paper we take the Combinatorial Neural Model (CNM), proposed by Machado and Rocha ([6], [7], and [8]), and explore the adoption of a committee machine to cope with the complexity problem present in this model. In our proposal, a static committee machine is built on a certain number of CNMs. The committee machine takes the discrete outputs from the individual models, and proceeds by choosing the most accurate output using a voting process. Many works has been presented to alleviate the complexity problem in CNM and this one represents an alternative that can be combined with other solutions. The results from the experiments point out that, by means of a committee machine, one can reach a better trade-off between predictive accuracy and complexity. To evaluate the effectiveness of the technique, we apply it to some well-known data sets from the UCI repository [2]. Finally, it is suggested some extensions to improve the committee machine in order to obtain a better generalization performance, towards a universal approximator.

1 Introduction

Scaling up machine learning algorithms to run over very large databases and making easier tasks like data cleansing and parameter selection are two important approaches for Knowledge Discovery from Databases (KDD). In this sense, Combinatorial Neural Model (CNM) has received, recently ([1], [3], [10], and [11]), many improvements that have turned it suitable for KDD.

This paper presents an implementation of a committee machine based on CNM and reports some preliminary results that can be used to alleviate the complexity problem of CNM.

Next section presents a rough description of committee machines. Section 3 explains in more detail the CNM. Section 4 describes the method applied to build the committee machine and the files used to run it. The fifth section analyzes the results obtained and the last one draws some conclusions and states some paths to explore as future work.

2 Committee machines

The central idea underlying a committee machine is that a set of experts, in consent, are expected to make better predictions than each one separately. According to Haykin [5], committee machines are universal approximators and apply the *principle of divide and conquer* to face computationally complex problems. Many works has been reported (*e.g.*, [9] and [12]) that account for the applicability of committee machines to reach better generalization performance. Committee machines may be classified as static or dynamic structures. As a static structure, it takes the responses from many predictors, combining them into a final solution, not involving the input signal. In the dynamic committee machine the input signal is involved and actuates directly in the mechanism that integrates the individual experts. The particular kind of committee machine adopted in this work is the *ensemble averaging method*. In this method, a set of experts (models) is trained in parallel and their outputs are combined somehow to produce an overall output. This method presents two interesting advantages [5]: the first one is regarded to the training time. If a unique expert replaces the experts in the committee machine, the training time is expected to be higher than the sum of the times for all experts. The second advantage is that the risk of overfitting the data decreases when you take separate portions of the data set and combine them in a committee machine.

3 Combinatorial Neural Model

CNM is a hybrid architecture for intelligent systems that integrates symbolic and connectionist computational paradigms. It has some significant issues, such as the ability to build a neural network from background knowledge; incremental learning by examples, solving the plasticity-stability dilemma [4]; a way to cope with the diversity of knowledge; knowledge extraction of an

artificial neural network; and the ability to deal with uncertainty. CNM is able to recognise regularities from high-dimensional symbolic data, performing mappings from this input space to a lower dimensional output space.

CNM uses supervised learning and a feedforward topology with one input layer, one hidden layer - here called combinatorial - and one output layer (Figure 1). Each neuron in the input layer corresponds to a concept - a complete idea about an object of the domain, expressed by an object-attribute-value form, they represent the evidences of the domain application. In the combinatorial layer there are aggregator type neurons, each one connected to one or more neurons in the input layer by fuzzy AND arcs that represent logical concepts. The output layer contains one neuron for each possible class (also called hypothesis), linked to one or more neurons in the combinatorial layer by fuzzy OR arcs that also represent concepts. The synapses may be excitatory or inhibitory and they are characterised by a strength value (*weight*) between zero (not connected) to one (fully connected synapses), that can express the logical relations. For the sake of simplicity, we will work with the learning of crisp relations, thus with strength value of synapses equal to one, when the concept is present, and zero, when the concept is absent. However, this option does not affect the approach to fuzzy relations learning.

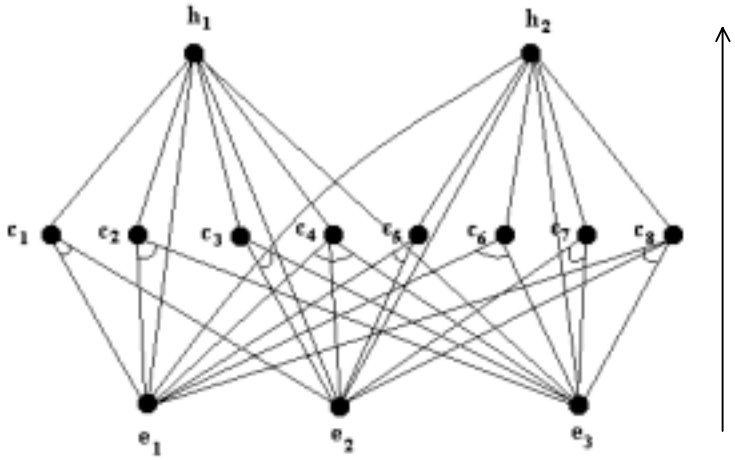

Figure 1. Complete version of CNM for 3 input evidences and 2 hypotheses [7]

The network is created completely empty, according to the following steps: (a) one neuron in the input layer for each evidence in the training set; (b) a neuron in the output layer for each class in the training set; and (c) for each neuron in the output layer, there is a set of hidden neurons in the combinatorial layer. The combinatorial layer comprises all possible combinations (length between two and nine) of the neurons in the input layer for each class in the output layer. There is no neuron in the combinatorial layer for length one connections. In this case, input neurons are connected directly to hypotheses.

The learning mechanism works in only one iteration, and it is described bellow:

PUNISHMENT_AND _REWARD_LEARNING_RULE

- **Set** to each arc of the network an accumulator with initial value of zero;
- **For** each example case from the training database, **do**:
  - **Propagate** the evidence beliefs from input nodes until the hypotheses layer;
  - **For** each arc reaching a hypothesis node, **do**:

    If the reached hypothesis node corresponds to the correct class of the case

    Then backpropagate from this node until the input nodes, increasing the accumulator of each traversed arc by its evidential flow (Reward)

    Else backpropagate from the hypothesis node until the input nodes, decreasing the accumulator of each traversed arc by its evidential flow (Punishment).

After training, the value of the accumulators associated to each arc arriving to the output layer will be between [-T, T], where T is the number of all cases present in the training set.

The last step is the pruning of the network. This is performed by the following actions: (a) remove all arcs whose accumulator is lower than a threshold (specified by a specialist); (b) remove all neurons from the input and the combinatorial layers that became disconnected from all hypotheses in the output layer; and (c) make weights of the arcs arriving at the output layer equal to the value obtained by dividing the arc accumulators by the largest arc accumulator value in the network. After pruning, the network becomes operational for classification tasks.

4 Material and methods

We used three files from UCI repository [2]:

1. Breast-Cancer obtained from the University Medical Centre, Institute of Oncology, and Ljubljana, Yugoslavia. (Sources: Matjaz Zwitter and Milan Soklic);
2. Balance Scale Weight and Distance, generated to model experiments reported by [13]; and
3. Monks-2 (donated by Sebastian Thrun, School of Computer Science, Carnegie Mellon University).

For each data set we took, at random, 4 samples of the same size (or almost the same size, depending whether they have an integer division by 4). We called these samples A, B, C, and D. Then we built four models, M1, M2, M3, and M123, using, respectively, A, B, C, and A+B+C. Moreover, we create the committee machine MC with the combination of M1, M2, and M3. The data set D remained reserved to be used as an independent test bed. After the training, each model was pruned, keeping only arcs arriving to the output layer with positive accumulators. In other words, only rules with a minimum empirical

support were kept. During the construction of the models, each model size - taking into account the input, output, and the combinatorial layers - and training time were recorded. Also, in the test phase, the accuracy (percentage of right predictions) was measured presenting each model with the data set D. The whole process is illustrated in Figure 2.

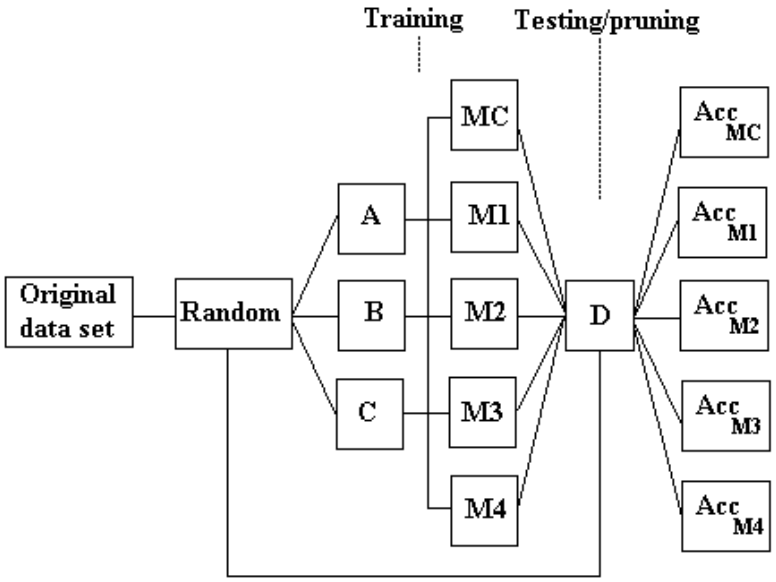

Figure 2. Model generation and testing process

5 Analysis of the results

This work presents some results towards the expansion of CNM to a universal approximator, using a committee machine. The results obtained with each model are summarized in Table 1.

It can be observed that using a committee machine, built from CNMs, as a classifier one can reach an accuracy comparable to the monolithic model M123. These results were obtained while reducing the complexity. In this sense, it was observed that each model, separately, in MC is much smaller than M123. Concerning to the training time, the results are more interesting. In this case, the sum of the training time of all components of MC is smaller than the training time of the whole M123. Both facts above point out that a committee machine represents an important alternative to cope with the temporal and spatial complexity of CNM.

On the other hand, although committee machines are considered universal approximators, this characteristic was not observed in the experiments. The adoption of a committee machine, as described in the previous section, did not improve the predictive accuracy of the final model.

Table 1. Summary of performance

		Models				
File	Item	M1	M2	M3	M123	MC
Breast Cancer	Size (nodes)	24,799	24,989	23,958	55,133	
	Training time (s)	407	401	380	2,918	
	Accuracy (%)	80.28	74.64	76.05	81.69	76.06
Balance Scale	Size (nodes)	1,014	1,025	997	1,822	
	Training time (s)	2	2	2	9	
	Accuracy (%)	81.4	76.92	85.89	86.53	85.25
Monks-2	Size (nodes)	2,744	2,781	2,782	3,833	
	Training time (s)	12	10	10	44	
	Accuracy (%)	87.70	90.98	91.80	95.08	94.26

Results shown in Table 1 point out the fact that, using a committee machine combined with a CNM, one can obtain a predictive accuracy only comparable to that obtained by the monolithic model (the model M123). It does not give the status of a universal approximator to MC. We believe MC can be improved to acquire this characteristic and present some suggestions in the next section.

6 Conclusions and future work

The committee machine has shown to be an interesting approach to face the complexity problem of CNM. Concerning to the expected improving of predictive accuracy of MC, the results were not satisfactory, since they just remained in the range of the component models. We are going to explore some directions in order to improve MC towards a universal approximator.

Our next step is to analyze the effects of taking the belief factors (given by the accumulators) of each rule in CNM, on deciding the class of an example in a MC. For instance, let us apply models M1, M2, and M3 to an example in the test data set. Suppose that the predicted classes, with the corresponding belief factors (BF), for each model are as following:

M1: Class = X (BF = 90%)
M2: Class = Y (BF = 60%)
M3: Class = Y (BF = 75%)

By the simple voting process, the result would be Class = Y. However, taking into account the belief factors, model M1 with BF = 90% would give Class = X (BF = 90%) as the final classification. Possibly, this result will be more accurate.

Another improvement we intend to do is the construction of the models in order to assign each one to a specific portion of the problem space. For instance, instead of building each model independently using random samples, we could

build the first model using a bigger sample, take the wrong predictions and use them to train the next model, and so on with the next model.

References

[1] Beckenkamp, F. G., Feldens, M. A., Pree, W., Optimizations of the Combinatorial Neural Model. *Proc. of the Vth Brazilian Symposium on Neural Networks. (SBRN'98)*, Belo Horizonte, Brazil, 1998.

[2] Blake, C. L. & Merz, C. J. UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science, 2000.

[3] Feldens, M. A.: Engenharia da Descoberta de Conhecimento em Bases de Dados: Estudo e Aplicação na Área de Saúde Porto Alegre: CPGCC da UFRGS, Brasil, 1997. (M.Sc. Dissertation).

[4] Freeman, J. A., Skapura, D. M.: Neural Networks, Algorithms, Applications, and Program Techniques. [S.l.]: Addison-Wesley, 1992. p.292-339.

[5] Haykin, S., *Neural Networks: A Comprehensive Foundation*, Prentice Hall: Upper Saddle River, NJ, 842pp, 1999.

[6] Machado, R. J., Rocha, A. F. Handling knowledge in high order neural networks: the combinatorial neural network. Rio de Janeiro, RJ: IBM Rio Scientific Center, 1989. (Technical Report CCR076).

[7] Machado, R., Rocha, A., The combinatorial neural network: a connectionist model for knowledge based systems. *Uncertainty in Knowledge Bases*, eds. N: B. Bouchon, R. R. Yager, and L. A. Zadeh, Springer-Verlag: Berlin, pp.578-587, 1991.

[8] Machado, R. J., Carneiro, W., Neves, P. A. Learning in the combinatorial neural model. *IEEE Transactions on Neural Networks*, **9**, pp.831-847, 1998.

[9] Parmanto, Bambang, Munro, Paul W., Doyle, H.R., Improving Committee Diagnosis with Resampling Techniques. IN: D. Touretsky, M. Mozer, and M. Hasselmo (Eds.) Advances in Neural Information Processing **8**, MIT Press,1996.

[10] Prado, H. A., Frigeri, S. R., Engel, P. M., A Parsimonious Generation of Combinatorial Neural Model. *Proc. of the IV Congreso Argentino de Ciencias de la Computación (CACIC'98)*, Neuquén, Argentina, 1998.

[11] Prado, H. A. do; Machado, K.F.; Frigeri, S. R.; Engel, P. M. Accuracy Tuning in Combinatorial Neural Model. *Proc. of Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Bejing, China, 1999.

[12] Schapire, R. E., Freund, Y., and Bartlett, P., Boosting the margin: A new explanation for the effectiveness of voting methods. *Proc. of Fourteenth International Conference of Machine Learning*, Nashville, TN, USA, 1997.

[13] Siegler, R. S. Three Aspects of Cognitive Development. *Cognitive Psychology*, **8**, 481-520.

Anexo 7 Clustering Algorithms for Data Mining

[PRA 2000d] PRADO, H.A. do; HIRTLE, S.C.; ENGEL, P.M. Clustering Algorithms for Data Mining. **Revista Tecnologia da Informação**, Brasília, DF, v. 2, n.1, p.51-58, 2000.

Clustering Algorithms for Data Mining

Hércules A. Prado *, Stephen C. Hirtle **, Paulo M. Engel ***

[1] University of Pittsburgh
Department of Information Sciences and Telecommunications
135 North Bellefield Ave.
Pittsburgh, PA 15.260
Phone: +1(412)624-9434
Fax: +1(412)624-2788
[2] Universidade Federal do Rio Grande do Sul
Instituto de Informática
Av. Bento Gonçalves, 9500 - Bairro Agronomia
Porto Alegre, RS - Brasil
Caixa Postal 15.064 - CEP 91.501-970
Fone: +55(051)316-6829
Fax: +55(051)319-1576
{prado, engel}@inf.ufrgs.br and hirtle@pitt.edu

Abstract. Knowledge Discovery from Databases (KDD) can be seen as a set of 'Computer-Aided Knowledge Discovery' techniques scaled to very large databases. By this way the old process of knowledge discovery has experienced great improvements by allowing the old algorithms of Machine Learning and Statistics to scale for huge amount of data. Moreover many important tasks like preparing and cleansing data, feature selection, model selection, parameter setting, validation, and interpretation has been eased by KDD. In the core of KDD enterprise we find the Data Mining process, that means the application of algorithms for identifying regularities on huge data bases. Among the tasks these algorithms intend to solve, clustering is the actual being-stressed field, after the concentrated effort dedicated to prediction in the first years of KDD. This paper presents an overview on clustering methods focussing, in particular, those that has been target of research efforts in the KDD realm. The aiming in doing so is to state a well-founded basis for foregoing research efforts.

* Researcher at EMBRAPA — Brazilian Enterprise for Agricultural Research and lecturer at Catholic University of Brasília
** Professor at University of Pittsburgh
*** Professor at Federal University of Rio Grande do Sul

Clustering Algorithms for Data Mining

1 Introduction

Since the early years of the Knowledge Discovery from Databases field we have seen a great amount of research effort devoted to the prediction tasks, that discovery process guided by pre-existent classes and where one wants to obtain explanations to the classes on the basis of the available data. Only recently the community has started to pay closer attention to descriptive techniques. Although a considerable burden of research had been carried on visualization processes, only around 1997 the clustering processes started to be addressed with more intensity. This paper goes in this direction, aiming to stablish a well-founded basis of clustering related processes in order to subsides subsequent research efforts. Sections 1-Introduction and 2-Motivation intend to situate the clustering process in the KDD field and to justify our interest for this theme. Sections 3, 4, and 5 give the reader a general idea of clustering, rather than a deep analysis of the process; there are excellent books ([AND 73] and [HAR 74]) that go much deeper in this subject. We describe two important algorithms representing different approaches for clustering. Sections 6 and 7 concentrate on an analysis of some available works on clustering under the KDD flag. Section 8 aims to state some points that can be subject of research efforts. Although all of our exposition has been done on the basis of structured data, with well defined dimensions, the applicability of the studied methods may be extended to non-structured data. For example, in recognizing groups on text, one can consider each word as a binary attribute. By this approach each document is represented by a record of binary fields and the data set take a form of a sparse binary matrix.

2 Motivation

Frequently we face the necessity to group entities as a way to organize our lives. Fair examples of this activity are: (a) how to organize the objects in a moving avoiding to put the heaviest ones over the others; (b) packaging of items when making shopping in a supermarket, paying attention to separate the fruits and vegetables from the meat and the fish, the frozen from the dry ones, and so on; and (c) puting the clothes in a wardrobe.

Much far beyond these so prosaic activities, we find scientists searching the space for areas with similar patterns of response to some signal; marketing experts looking for interesting portions of the market, or a telecommunication company identifying homogeneous regions with respect to the use of certain services.

On the other hand, the amount of possible ways to partition a set of n elements into k pairwise, disjoint, non-empty subsets respect the Stirling number of second kind. This number can be generated by the recursive definition [BOR 91]:

$$S(n,k) = 1 \quad \text{for}\, k > 0 \,\text{and}\, n = k, \quad S(n,0) = 0 \quad \text{for}\, n \geq 0$$

$$S(n,k) = S(n-1,k-1) + kS(n-1,k) \quad \text{for}\, 0 < k < n.$$

As an example, suppose we have three objects, a, b and c. The ways we can partition this set into one group is $\{abc\}$; into two groups are $\{a, bc\}$, $\{ab, c\}$, and $\{ac, b\}$; into three groups is $\{a, b, c\}$. Taking the example in [AND 73], lets consider 25 observations for whose we want to form 5 groups. The Stirling number of second kind in this case is $S = 2,436,684,974,110,751$. Considering that the number of groups are not known, the problem must be stated as a sum of Stirling numbers of second kind for each possible number of groups. It is clearly an enourmous amount of groups that cannot be computed in an acceptable time, even for small databases.

In face of the great amount of possible ways to group data, it is natural that exist lots of clustering techniques. However, the majority, in his original form, requires many passes of the database to find the groups. This fact restricted the algorithms to process only small data sets. Considering the problem posed today by the increasing size of the databases, it is necessary to improve the clustering techniques to process these databases. In fact the first efforts in clustering on KDD has been in this direction ([BRA 98] and [MOO 98]), although some works face the incorporation of new features ([AGR 98]) to this approaches. Put together the related tasks of: feature selection, parameterizing the models and the algorithms, and so on, and we will have a challenging scenary to work with clustering on KDD.

3 The Clustering Process

According to Murphy and Medin (*op. cit. in* [WRO 94] p.32), two hypothesis works as constraints in aggregating objects to form concepts: one states that a concept can be expressed by means of similarity measures and the other that correlation amongst variables defines the concepts. The author emphasizes that there is significant evidence that people use more the second hypothesis. We feel that both hypothesis are important: the similarity hipothesis help the analyst in trying to identify interesting groups and the second one can help him to explain each group. In the same direction, Wittgenstein (*op. cit. in* [HAN 90] p.238) advocates that concepts has a polythetic nature that, in other words, means that a concept is more naturally described by the subspace of its dimensions' set, what means, by the correlations.

The search for knowledge has considered these remarks, mainly by adopting two ways: the classification and clustering[1] processes. We also could blend these two alternatives in one unique approach, to comprise the obtention of de clusters

[1] This two ways correspond to Supervised and Unsupervised Learning, respectively, in the field of Machine Learning.

(by similarity), their labeling, and the application of a classification procedure to find the subsets of dimensions that most strongly correlates with the labels.

An important observation, stated by Hanson [HAN 90] concerns to the existence of some previous knowledge for the clustering process: "clustering of objects or events without a context, goal, or some information concerning the function of the derived clusters is not likely to be useful for real-world problems."

By the previous ideas, we conclude that clustering is nor a blind neither a strongly guided enterprise, being rather influenced for the target of the analyst, his intuition about the classes underlying the data, and any other information concerning the domain that can affect the search for knowledge structures. Pratical issues that, although important, are not part of this monography are: how to deal with different types of variables, measures of similarity, feature selection, feature reduction, parameter selection (including the choice for the nmber of clusters). The main focus are the algorithms for partition the data.

4 Agglomerative Process to Hierarchical Clustering

This approach to make hierarchical clustering is called agglomerative, in oposition to the divisive alternative, because it constructs the hierarchy from the most detailed (the data exemplars) to the highest level (the root). The divisive alternative goes in the opposite way. We based on Anderberg [AND 73] book to prepare this section.

This method always departs from a similarity matrix **S**, which elements are s_{ij}, the similarity between the data units i and j. The similarity matrix is depicted in Figure 1 and has a triangular shape, since the reflexive property between the similarities of two data elements is respected.

$$\mathbf{S} = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ s_{21} & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ s_{31} & s_{32} & \cdot & \cdot & \cdot & \cdot & \cdot \\ s_{41} & s_{42} & s_{43} & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ s_{n1} & s_{n2} & s_{n3} & \cdot & \cdot & s_{n(n-1)} & \cdot \end{pmatrix}$$

Fig. 1. Similarity matrix (lower triangular) [AND 73]

The general algorithm to produce agglomerative clusters take the similarity matrix and execute the following steps:

1. Begin with n clusters each consisting of exactly one entity. Let the clusters be labeled with the numbers 1 through n.
2. Search the similarity matrix for the most similar pair of clusters. Let the chosen clusters be labeled p and q and let their associated similarity be $s_{pq}, \quad p > q$.

3. Reduce the number of clusters by 1 through merging clusters p and q. Label the product of the merging as q and update the similarity matrix entries in order to reflect the revised similarities between cluster q and all other existing clusters. Delete the row and column of S pertaining to cluster p.
4. Perform steps 2 and 3 a total of $n-1$ times (at which point all entities will be in one cluster). At each stage record the identity of the clusters which are merged and the value of similarity between them in order to have a complete record of the results.

Note that similarity between pairs can be measured by (at least) two forms: a distance-like measure (like Euclidean distance) or a correlation-like measure. If a similarity matrix has similarities expressed in distance-like measures the algorithm will take the pair with a lowest measure as the most similar. Otherwise, if a correlation-like measure is used the most similar would be the pair with higher correlation. After running this algorithm a tree (called *dendrogram*) emerges. This tree is exemplified in Figure 2 and represents the various levels of aggregation reached during the process.

A point that has to be defined is how the similarities are revised when the algorithm join two clusters. There are many alternatives, and for our study we will consider the simplest one: single linkage. Suppose that two clusters p and q were joined forming the cluster t. The similarity between t and an arbitrary cluster r is calculated as the following rule:

$$s_{tr} = min(s_{pr}, s_{qr})$$

if a distance-like measure is used, or

$$s_{tr} = max(s_{pr}, s_{qr})$$

if a correlation-like measure is taken.

It means that the similarity between t and r is done by the most similar of p and q with respect to r.

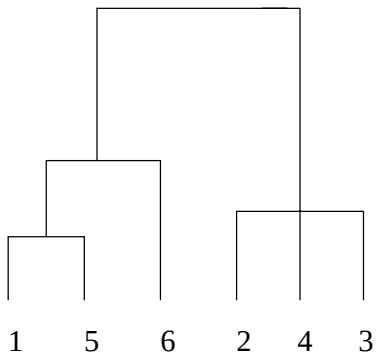

Fig. 2. Example of dendrogram

A possible similarity matrix that could lead to the dendrogram in Figure 2 is that shown in Figure 3.

$$\mathbf{S} = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 7 & \cdot & \cdot & \cdot & \cdot & \cdot \\ 5 & 2 & \cdot & \cdot & \cdot & \cdot \\ 6 & 2 & 2 & \cdot & \cdot & \cdot \\ 1 & 5 & 6 & 7 & \cdot & \cdot \\ 3 & 6 & 7 & 5 & 4 & \cdot \end{pmatrix}$$

Fig. 3. Original similarity matrix for the dendrogram in Figure 2

Although this is beyond the scope of this work, an evaluation of the clustering process can be done on the basis of the two similarity matrices: the original one (Figure 3) and that based on the generated hierarchies (Figure 4). The *cophenetic correlation* [LEG 98], calculated on the basis of these two matrices, serves as an element to reason about the quality of the clustering process.

$$\mathbf{S} = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 5 & \cdot & \cdot & \cdot & \cdot & \cdot \\ 5 & 2 & \cdot & \cdot & \cdot & \cdot \\ 5 & 2 & 2 & \cdot & \cdot & \cdot \\ 1 & 5 & 5 & 5 & \cdot & \cdot \\ 3 & 5 & 5 & 5 & 3 & \cdot \end{pmatrix}$$

Fig. 4. Resulting similarity matrix for the dendrogram in Figure 2

Certainly, changings from the original to the resulting similarity matrix are due to the joining procedure of each pair of clusters. The similarity between two clusters is the higher similarity between all possible combinations of the entities in one cluster with the entities in the other cluster.

5 Adaptive Reasonance Theory

In this section we describe a neural network approach for clustering, called ART1. The network is a member of the so-called ART family, where the name is an acronym for the Adaptive Resonance Theory of Gail Carpenter and Stephen Grossberg. The sources to this section were provided by [LIP 87], [BIG 96], and [ENG 97].

ART1 is a competitive recurrent network with two layers, the input layer and the clustering layer. This network achieves to overcome the plasticity-stability dillema [FRE 92], allowing an incremental learning, with a continuous updating of the clusters prototypes, and preserving the previously stored patterns. The

clustering algorithm proceeds, in general steps, as folows: (a) the first input is selected to be the first cluster; (b) each next input is compared with each existing cluster; the first cluster where the distance to the input is less then a threshold is chosen to cluster the input. Otherwise, the input defines a new cluster. It can be observed that the number of clusters depends on the threshold and the distance metric used to compare the inputs with the clusters. More detailed, for each input pattern presented to the network, one output unit is declared winner (at the first pattern, the own input pattern defines the cluster). The winner backpropagates a signal that encodes the expected pattern template. If the current input pattern differs more than a defined threshold from the backpropagated signal, the winner are temporarily disabled and the next closest unit is declared winner. The process continues until an output unit become a winner, considering the threshold. If none of the output units become a winner, a new output unit is defined to cluster the input pattern. Graphically, an ART1 network can be illustrated by Figure 5, where it appears with four input and six ouput neurons. t_{ij} and b_{ij} are, respectively, bottom-up and top-down connections.

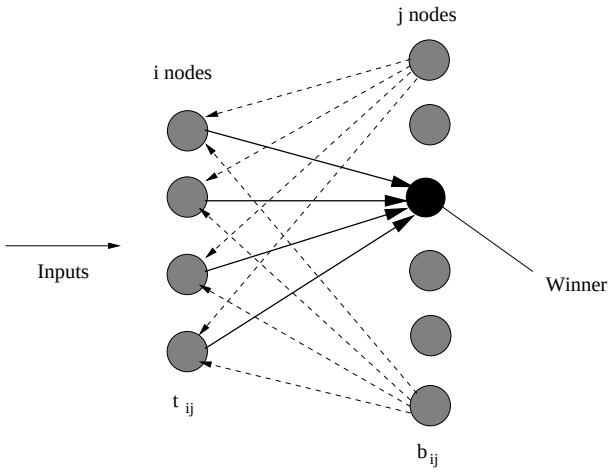

Fig. 5. Architecture of an ART network [BIG 96]

One important characteristic of this net is that it works in just one pass, what is interesting when we are processing a huge amount of data. The training algorithm for this network is the following:

– **Step1. Initialization**

$$t_{ij}(0) = 1$$

$$b_{ij}(0) = \frac{1}{1+N}$$

$$0 \leq i \leq N-1$$

$$0 \leq j \leq M-1$$

$$\text{Set} \quad \rho, \quad 0 \leq \rho \leq 1$$

In these equations $b_{ij}(t)$ is the bottom-up and $t_{ij}(t)$ is the top-down connection weight between input node i and output node j at time t. These weights define the exemplar specified by output node j. The fraction ρ is the vigilance threshold which indicates how close an input must be to a stored exemplar to match.

- **Step 2. Apply New Input**
- **Step 3. Compute Matching Scores**

$$\mu_j = \sum_{i=0}^{N-1} b_{ij}(t) x_i \quad 0 \leq j \leq M-1$$

In this equation μ_j is the output of output node j and x_i is the element of the input which can be 0 or 1.

- **Step 4. Select Best Matching Exemplar**

$$\mu_j^* = \max\{\mu_j\}$$

- **Step 5. Vigilance Test**

$$||X|| = \sum_{i=0}^{N-1} x_i$$

$$||T.X|| = \sum_{i=0}^{N-1} t_{ij}.x_i$$

$$\text{IF } \frac{||T.X||}{||X||} > \rho \Rightarrow \text{GO TO STEP 7}$$

$$\text{OTHERWISE} \Rightarrow \text{GO TO STEP 6}$$

- **Step 6. Disable Best Matching Exemplar** The ouput of the best matching node selected in Step4 is temporarily set to zero and no longer takes part in the maximization of Step4. Then go to Step 3.
- **Step 7. Adapt Best Matching Exemplar**

$$t_{ij_*}(t+1) = t_{ij_*}(t) t(x_i)$$

$$b_{ij_*}(t+1) = \frac{t_{ij_*}(t) x_i}{\frac{1}{2} + \sum_{i=0}^{N-1} t_{ij_*}(t) x_i}$$

- **Step 8. Repeat by Going to Step 2** (First enable any node disabled in Step 6)

An example of application of this algorithm is shown in Figure 6, where letters are clustered. The left side shows the input patterns and the right side the way the clusters are formed.

Fig. 6. Example of ART behavior [LIP 87]

6 Reported Works

In this part we review the solutions presented for clustering, taking into account the KDD requirements. According to Agrawal *et al.* [AGR 98], the applications of clustering in data mining place the following desiderata to clustering algorithms:

1. Effective treatment of high dimensionality: it is important to take into account that the higher dimensionality the lower average density of points anywhere in the space. It leads to the necessity of finding clusters in the subspace of the dimensions.
2. Interpretability of the results: adequate cluster descriptions are required for the end user in order to activate his insight. Thus, explanation is very important.
3. Scalability and usability: the technique must be easy of use, fast, and scalable in the number of dimensions and the size of the data set.

Other important requirements to perform clustering on large data sets are pointed out by Bradley *et al.* [BRA 98]:

1. Apply the clustering algorithm in one pass (or less) over the database. Even a single scan is considerable expensive, thus early termination is desirable.
2. A behavior that allows to show anytime the "best" answer, with all descriptive information about the process (expected remaining time, how much information has been processed, etc).
3. The process must allow to be suspended, stopped, and resumed. Moreover, it must be possible continue to incorporate new data into the existing model.

4. Work in an acceptable way inside the limits of a given RAM.
5. To allow different kinds of scan: sequential, random, etc.
6. Ability to work only forward, avoiding to turn back to information just processed.

These features are required when one is scaling any known clustering algorithm to data mining. An interesting question that can enrich this list of requirements are faced in Guha [GUH 98], that corresponds to find clusters with varied shapes (not only spherical). In their work they present an algorithm where each cluster is represented by a cloud of points rather than a centroid. This representation captures the shape of the cluster.

The strong requirements reported above pose the challenge to develop new clustering algorithms or improve the older ones. One of the first proposals to scale the clustering task to data mining was the BIRCH algorithm, presented in [ZHA 96]. The authors applied a preclustering process, that consists in representing dense regions of points by means of their statistical summaries, and then clustering these summaries in a centroid-based hierarchical algorithm. The summaries are represented in memory and each point of the input data set is added to this representation. The method assures that the clustering process is completed in just one pass.

One important problem that occurs when clustering high-dimensional data is reported by ([BER 97] cit. by [AGR 98]): *K*-means fails in clustering high-dimensional data because the pair-wise distance is almost similar among the data points. It happens because many dimensions may have uniformly distributed values . This drawback can be overcomed by looking for clusters into the subspace of the dimensions, as in CLIQUE algorithm presented by Agrawal *et al.* [AGR 98]. Another interesting example is the framework proposed in [BRA 98], where it is focussed regions on the data that are compressible, regions that have to be maintained in the memory and regions that can be discardable. By this way, the algorithms optimize the use of the available RAM.

Other recent contributions has been provided by Moore [MOO 98] that scaled Mixture Model clustering and Pelleg [PEL 98] that worked with *K*-means.

In the next chapter we describe the framework proposed by Bradley *et al.* [BRA 98], an effort in the direction to generalize a method for scale clustering algorithms.

7 A Framework to Scale Clustering Algorithms: *K*-means Example

In this chapter we present the framework proposed by Bradley *et al.* [BRA 98] instantiated to the well-known *K*-means algorithm. The basic steps of standard *K*-means are: (1) Initialize the model parameters (number of cluster, and centroids, for instance); (2) Link each point in the data set to the nearer centroid; (3) Re-estimate the centroids of the model (according to the points linked to each cluster); (4) If the current and the new model are sufficiently closer to

each other, terminate, else return to (2). Although K-means, in its standard presentation, requires many passes of the data set, its implementation, according to the present framework, does not need these passes. The conception of this framework departs from the principle that one can make effective clustering by selecting "important" portions of the database to be stored in memory, while summarizing others regions.

The general process goes according to the following steps:

1. Get a sample of the database and load to the RAM buffer.
2. Apply the sample to the current model.
3. Using the current model, classify the data into the buffer as:
   (a) Needs to be retained in the buffer (retained set RS).
   (b) Can be discarded after updating to the sufficient statistics (discarded set DS). Sufficient statistics are explained in Subsection **7.2**.
   (c) Reduced via compression and summarized as sufficient statistics (compression set CS).
4. If stop criteria is satisfied terminate; else go to 1.

7.1 Updating Current Model

Step 2 of the general process above works as a K-means extended to deal with the content of the RAM buffer: data points plus sufficient statistics. The model updating for data points is the same as in K-means. For the sufficient statistics, the algorithm has to deal with the triples (SUM, SUMSQ, N), representative of the discarded and compressed data.

7.2 Sufficient Statistics

Both sets DS and CS requires the calculation of sufficient statistics (SS, for short). The first one to retain information about the data points unlikely to change the clusters centroids. The latter to retain the information about the regions of hig density that does not belongs to any cluster and forms subclusters. These subclusters will be used in a future step of the algorithm that can change the configuration of the clusters.

The sufficient statistics are formed by the tripe (SUM, SUMSQ, N), where

$$\text{SUM} = \sum_{i=1}^{N} x^i \in R^n \quad \text{and} \quad \text{SUMSQ} = \sum_{i=1}^{N} (x_j^i)^2$$

where x^i stands for the data point i, N the number of elements, n for the dimensions of the problem, for $j = 1, ..., n$ in the set to be represented.

7.3 Discarding Data Points

This corresponds to the step (3.b) of the general process. It is called Primary Data-Compression too and the authors proposed two methods to do this. For the sake of simplicity, we will present just one method. The compression corresponds to record into the corresponding cluster centroids the sufficient statistics of data points that unlikely will change these centroids. The data points considered to be discarded are those delimited by a radium determined by Mahalanobis Distance. This radium from a point x to the centroid x (assuming a Gaussian distribution) with covariance matrix S is given by:

$$D_n = (x, \overline{x}) = \sqrt{\sum_{j=1}^{n} \frac{(x_j - \overline{x})^2}{S_{jj}}}$$

The idea is to discard $p\%$ of the points collapsed by the radium. All data points are discarded and its sufficient statistics merged with the existing SS in the respective cluster.

The merging of the SS $(\mathrm{SUM}_i, \mathrm{SUMSQ}_i, N_i)$ and $(\mathrm{SUM}_j, \mathrm{SUMSQ}_j, N_j)$ of two points i and j are obtained by the sum of the respective elements of the triple, what gives $(\mathrm{SUM}_i + \mathrm{SUM}_j, \mathrm{SUMSQ}_i + \mathrm{SUMSQ}_j, N_i + N_j)$.

7.4 Compressing Data Points

This task takes into account the data points not compressed in the discarding phase and looks for subclusters in these remaining data. The idea is to confront these subclusters with the clusters; if the data points always change cluster assignment together, the cluster means can be changed applying exactly the SS of the subcluster. The compression has two basic parts: (1) locate the most dense regions of the non compressed space as candidates applying the standard K-means; and (2) apply a "tightness" criterion to these candidates.

In part one, the points non compressed are clustered in K_2 subclusters ($K_2 >$ number of clusters in the model). The "tightness" criterion is based on the difference between the variance and the mean of each variable; if the maximum difference is less than a given threshold β, the subcluster is considered "tight" and can be merged with the nearest existing subcluster (it is used the hierarchical agglomerative method to do this).

8 The Foreseeing Scene: Open-end Problems

We can identify three main concerns in the reported proposals for clustering in data mining: how to reach clusters with varied shapes, how to scale the algorithms to very large data bases, and how to approach the sparsity of data when dimensionality increases. Considering that each already existing algorithm has its own merits and that exists yet a great amount of algorithms, we believe that there is much room indeed to improve these algorithms under the guidelines of

data mining. For example, we did not find approaches to apply these ideas to neural networks that perform clustering, yet.

Another question, more intriguing for us is how to take into account the bulk of knowledge that an analyst apply during a clustering process. We know, for sure, that this prior knowledge exists and count so much in the process. It counts, for instance, when the analyst decides the number of clusters, or if the clustering process is satisfactory, or even when cutting a dendrogram to define the clusters. Clearly, this process consider previously existing knowledge structures. Based on this ideas, we could imagine these structures, adequately represented, interacting with the clustering process. In this direction, LINNEO$^+$ system represents an interesting effort. This system, presented by Béjar *et al.* [BÉJ 93], uses production rules to constrain the algorithm in order to find more adequate clusters in a conceptual clustering schemma. Although, this is an significant effort, more research are required to extend the idea to other clustering techniques.

References

[AGR 98] AGRAWAL, R., GEHRKE, J., GUNOPULOS, D., RAGHANAVAN, P. Automatic Subspace Clustering of High-Dimensional Data for Data Mining Applications. In: Proceedings of ACM SIGMOD98 International Conference on Management of Data, 27., 1998, Seattle, Washington, USA.

[AND 73] ANDERBERG, M. R. Cluster Analysis for Applications. London, UK: Academic Press, Inc., 1973. 359pp.

[BÉJ 93] BÉJAR, J., CORTÉS, U., POCH, M. LINNEO$^+$: A Classification Methodology for Ill-structured Domains. Technical Report LSI-93-22-R. Facultad d'Informatica de Barcelona, 1993. Available by ftp in http://www-lsi.upc.es/∼bejar/reports

[BER 97] BERCHTOLD, S., BOHM, C., KEIM, D. and KRIEGEL, H.-P. A cost model for nearest neighbor search in high-dimensional data spaces. In: Proceedings of the 16th Symposium on Principles of Database Systems (PODS), pages 78-86, 1997.

[BIG 96] BIGUS, J. P. Data Mining with Neural Networks. [S.l.]: McGraw-Hill, 1996. p.3-42.

[BOR 91] BOROWISKI, E.J. & BORWEIN, J. M. MATHEMATICS - The Harper Collins Dictionary. New York, NY: Harper Collins Publishers, Inc, 1991. 657pp.

[BRA 98] BRADLEY, P., FAYYAD, U., REINA, C. Scaling Clustering Algorithms to Large Databases. In: Proceedings of Fourth International Conference on Knowledge Discovery and Data Mining. New York, NY: AAAI Press.

[ENG 97] ENGEL, P. M. Lecture Notes. Universidade Federal do Rio Grande do Sul. Porto Alegre: CPGCC da UFRGS, 1997.

[FAY 96] FAYYAD, U. et al. From Data Mining to Knowledge Discovery: An Overview. In: FAYYAD, Usama et al. (Eds.). Advances in Knowledge Discovery and Data Mining. Cambridge, Mass.: AAAI/MIT Press, 1996.

[FAY 98] FAYYAD, U.

[FRE 92] FREEMAN, J. A.; SKAPURA, David M. Neural Networks, Algorithms, Applications and Program Techiniques. [S.l.]: Addison-Wesley,1992. p.292-339.

[GUH 98] GUHA, S., RASTOGI, R., SHIM, K. Cure: An Efficient Clustering Algorithm for Large Databases. In: Proceedings of ACM SIGMOD International Conference on Management of Data, Seattle, Washington, 1998.

[HAN 90] HANSON, S.J. Conceptual Clustering and Categorization: Bridging The Gap Between Induction and Causal Models. In: KODRATOFF, Y., MICHALSKI, R. (Eds.). Machine Learning: An Artificial Intelligence Approach. San Mateo, CA: Morgan, 1990.

[HAR 74] HARTIGAN, J. Clustering. In: FAYYAD, Usama et al. (Eds.). Advances in Knowledge Discovery and Data Mining. Cambridge, Mass.: AAAI/MIT Press, 1996.

[JOH 97] JOHN, G. H. Enhancements to the Data Mining Process. Stanford, EUA: Stanford University, 1997. Ph.D. Thesis.

[KAS 97] KASKI, S. Data Exploration Using Self-Organizing Maps. Helsinki, Finlndia, 1997. Ph.D. Thesis.

[KOD 90] KODRATOFF, Y., MICHALSKI, R. (Eds.). Machine Learning: An Artificial Intelligence Approach. San Mateo, CA: Morgan, 1990.

[LEG 98] LEGENDRÉ, P. and LEGENDRÉ, L. Numerical Ecology. Amsterdan: Elsevier, 1998. 853 pp.

[LIP 87] LIPPMANN, D.; An Introduction to Computing with Neural Nets, IEEE ASSP Magazine. April, 1987.

[MER 97] MERKL, D. Exploration of Document Collections with Self-Organizing Maps: A Novel Approach to Similarity Representation. In: EUROPEAN SYMPOSIUM ON PRINCIPLES OF DATA MINING AND KNOWLEDGE DISCOVERY, 1., 1997, Trondhein, Norway. Proceedings ... Berlin: Springer-Verlag, 1997. (Lecture Notes in Computer Science, v. 1263, p.101-111).

[MOO 98] MOORE, A. Very Fast EM-based Mixture Model Clustering Using Multiresolution kd-trees. In: Proceedings of Neural Information Systems Processing. Cambridge, MA: MIT Press, 1998.

[PEL 98] PELLEG, D., MOORE, A. Accelerating Exact K-means Algorithms with Geometric Reasoning. In: Proceedings of Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. [S.l.]:ACM, Inc, 1999.

[WRO 94] WROBEL, S. Concept Formation and Knowledge Revision. Dordrecht, The Netherlands: Kluwer, 1994. 240pp.

[ZHA 96] ZHANG, T., RAMAKRISHNAN, R., LIVNY, M. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In: Proceedings of the ACM SIGMOD96 International Conference on Management of Data, p. 103-114, Montreal, Canada, June, 1996.

Anexo 8 Dealing with Inconsistencies and Knowledge Loss in Combinatorial Neural Model

[PRA 2001] PRADO, H.A. do; ENGEL, P.M.; SILVA, K.C. da; Dealing with Inconsistencies and Knowledge Loss in Combinatorial Neural Model. In: ARGENTINE SYMPOSIUM ON ARTIFICIAL INTELLIGENCE, 2001, Buenos Aires, Argentina. **Proceedings...** Buenos Aires: Universidade Nacional de Rosario, 2001.

Dealing with Inconsistencies and Knowledge Loss in Combinatorial Neural Model

HÉRCULES ANTONIO DO PRADO[1], PAULO MARTINS ENGEL[2], KÁTIA CILENE DA SILVA[3]

Universidade Federal do Rio Grande do Sul

Instituto de Informática

Av. Bento Gonçalves, 9500 - Bairro Agronomia

Porto Alegre / RS - Brasil

Caixa Postal 15064 - CEP 91.501-970

e mail: {prado, engel, katiacs}@inf.ufrgs.br
telephone: +55(051)3316-6829
fax: +55(051)3316.7308

Abstract

During the last years much effort has been devoted to generate knowledge through Data Mining techniques. Despite of all advances, few efforts have been addressed to cope with post processing problems. This paper is about those problems in Combinatorial Neural Model (CNM). CNM, a supervised learning algorithm introduced by Machado, received many improvements that made it useful for Data Mining. Two main problems are approached. The first one corresponds to the conflicts that can emerge in the knowledge base since the model acquires knowledge from different sources, either specialists or examples. In this way, we apply the concept of extended negation, as the Boolean negation is not so natural. The second problem arises after applying the pruning process to CNM. Since the model is incremental, any part of the knowledge base pruned after the training process in time t_i can be important to the training process in time t_{i+1}. Disregarding this pruned part can lead to loss of knowledge. Considering that it is not possible to avoid pruning, and thus to maintain the knowledge base untouched during all its lifetime, we propose an approach to mitigate the problem.
Keywords: knowledge discovery from databases, data mining, neural networks, inconsistency handling.

---

[1] Lecturer at Catholic University of Brasilia and researcher at EMBRAPA - Brazilian Enterprise for Agricultural Research
[2] Professor at UFRGS - Federal University of Rio Grande do Sul
[3] M.Sc. Student at UFRGS - Federal University of Rio Grande do Sul

1 Introduction

The issue of post-processing has been frequently mentioned in Data Mining texts [2, 3, 4], although few works has been addressed in this direction, probably as a consequence of the model adherence of this kind of problem. In this paper we focus on the post-processing problems of Combinatorial Neural Model (CNM). This model was proposed by Machado [8] and received many improvements [1, 5, 9, 10, 11, 12, 13, 14] to become suitable for Data Mining. However, there are some problems in this model that were not approached yet. One of these problems are the inconsistencies generated when acquiring knowledge from different sources, either introducing background knowledge to the model or learning from examples. Another problem is the knowledge loss caused by the pruning process. In the next sections, after describing the CNM, we discuss each of these problems, specifying our approach to deal with them.

2 The Combinatorial Neural Model

CNM (see Fig. 2.1) is a hybrid architecture for intelligent systems that integrates symbolic and connectionist computational paradigms. This model is able to recognize regularities from high-dimensional symbolic data, performing mappings from this input space to a lower dimensional output space. Another important advantage of CNM is that the model overcomes the plasticity-stability dilemma [6, 7].

CNM uses supervised learning and a feedforward topology with one input layer, one hidden layer – here called combinatorial layer – and one output layer. Each neuron in the input layer corresponds to an evidence (denoted by *e*) of the world, expressed in an object-attribute-value form. In the combinatorial layer there are aggregative fuzzy AND neurons (denoted by *c*), each one connected to one or more neurons in the input layer by arcs with adjustable weights. The output layer contains one aggregative fuzzy OR neuron (also called hypothesis and denoted by *h*) for each possible class, linked to one or more neurons of the combinatorial layer. The synapses may be excitatory or inhibitory and are characterized by a strength value (weight) between zero (not connected) and one (fully connected synapses).

The network is created completely uncommitted, according to the following steps: (a) one neuron in the input layer for each evidence in the training set; (b) one neuron in the output layer for each class in the training set; and (c) for each neuron in the output layer, there is a

complete set of hidden neurons in the combinatorial layer which corresponds to all possible combinations of connections with the input layer. Actually, in order to reduce the combinatorial explosion, the author recommends to adopt the magic number of Miller [8] (seven plus or minus two) for the maximum size of the combinations. Moreover, combinations with size one (just one neuron in the input layer) are connected directly to the hypotheses.Thus, the combinatorial layer keeps the connections with length between two and nine.

The learning mechanism works in one iteration, and it is described below. Note that we are applying the learning mechanism to a topology (with the three layers) already built in one previous step.

PUNISHMENT_&_REWARD_LEARNING_RULE

*input***:** set of examples *E* and the network topology

*output***:** trained network

*processing***:**

begin

for each example from *E*, *do*

propagate the evidence beliefs from input neurons until the hypotheses layer;

for each arc reaching a hypothesis neuron, *do*:

if the reached hypothesis neuron corresponds to the correct class of the case

then backpropagate from this neuron until input neurons, increasing the accumulator of each traversed arc by its evidential flow (Reward)

else backpropagate from the hypothesis neuron until input neurons, decreasing the accumulator of each traversed arc by its evidential flow (Punishment).

*end_for***;**

end_begin.

After training, the value of accumulators associated to each arc arriving to the output layer will be between [-T, T], where T is the number of cases present in the training set. The last step is the pruning of the network; and it is performed through the following actions: (a) remove all arcs whose accumulator is lower than a threshold (specified by a specialist); (b) remove all neurons from the input and combinatorial layers that became disconnected from all

hypotheses in the output layer; and (c) make weights of the arcs arriving at the output layer equal to the value obtained by dividing the arc accumulators by the largest arc accumulator value in the network. After this pruning, the network becomes operational for classification tasks.

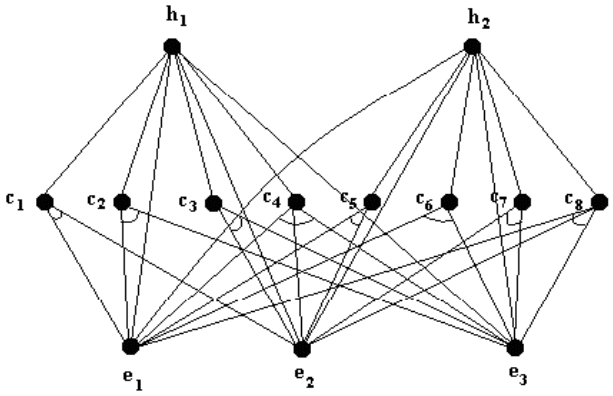

Fig. 2.1. Combinatorial neural model

3 Dealing with Inconsistencies

In real cases, the pure Boolean negation is not adequate to deal with inconsistencies, since a negation of a concept *c* is not accomplished by the simple concatenation of a *not* signal (like ¬, e.g.) before the concept symbol. In the medical domain, where CNM was first applied, it is common to have a set of concepts that are incompatible among them. For example, one physician may conclude by *leichmaniosis* on the basis of some symptoms but another physician may disagree in one symptom. It is necessary to deal with this not only by expressing an overall probability, but also showing to the specialists the disagreement. The model as used does not care about this problem. Actually, by computing the overall confidence (based on beliefs and disbeliefs) the model masks the problem. In order to cope with the contradiction posed in these terms, we define and treat this relation by means of the Set Theory.

3.1 The Basic Process

For terminology convenience, we call both evidences and hypotheses in CNM as *conditions* and define the extended negation as *incompatibility* among conditions. It is justified by the fact that we can have conflicts among any kind of conditions (hypotheses or evidences).

First we define the notion of *incompatibility* with respect to two conditions. Then, based on this notion, it is defined the set of conditions that are incompatible with a specific condition. Finally, we define the set of conditions that are incompatible with another set of conditions. This definition is necessary, since we are interested in identifying inconsistencies in rules, which can involve many conditions.

Definition 1: Incompatibility - A condition *y* is incompatible with a condition *x* if they cannot be simultaneously true.

Definition 2: Incompatibility class of a condition x - The incompatibility class of a condition *x*, denoted by *I(x)*, is composed by all conditions that are incompatible with *x*.

Definition 3: Extended incompatibility class of the set $X=\{x_1,x_2,...,x_n\}$ – The extended incompatibility class of the set $X=\{x_1,x_2,...,x_n\}$, denoted by $EI(x_1,x_2,...,x_n)$, is defined by the set of conditions $Y=\{y_1, y_2 ..., y_m\}$, where $Y=I(x_1)\cup I(x_2)\cup...\cup I(x_n)$.

Naturally, the specialists must define the incompatibility classes and the system computes the extended incompatibility classes. The following algorithm performs the detection of incompatibilities, on the basis of the extended incompatibility classes:

INCOMPATIBILITIES_DETECTOR_ALGORITHM

input: Set of rules defined by a CNM and the incompatibility classes for all conditions in the rules;

output: Set of rules with incompatibilities;

processing:

begin

for each rule $R(E,h)$, with *E* the set of evidences and *h* the consequence, do

if $\{E,h\}\cap EI(E,h)\neq\emptyset$ *report* "inconsistent rule: type 1";

end_for;

for each unordered pair of rules $R_1(E_1,h_1)$ and $R_2(E_2,h_2)$, *do*

if $E_1\subseteq E_2$ *and* $\{E_1,h_1,E_2,h_2\}\cap EI_1(E_1,h_1,E_2,h_2)\neq\emptyset$ *report* "inconsistent rules R_1 and R_2: type 2";

end_for;

end_begin.

Basically, the algorithm detects two kinds of inconsistencies. The first one(type 1) occurs inside the conditions of each rule. The second one (type 2) occurs between two rules that eventually are triggered simultaneously.

To illustrate, suppose we have the rules:

$R_1=((e_1,e_2,e_3),h_1)$,

$R_2=((e_4,e_5,e_6),h_2)$, and

$R_3=((e_1,e_2),h_3)$

the incompatibilities:

$I_1(e_1)=\{e_7\}$,

$I_2(e_4)=\{e_8\}$,

$I_3(h_3)=\{e_3\}$, and

$I_4(h_2)=\{e_5\}$

and the extended incompatibility classes of interest for us:

$EI_1(e_1,e_2,e_3,h_1)=\{e_7\}$,

$EI_2(e_4,e_5,e_6,h_2)=\{e_5,e_8\}$,

$EI_3(e_1,e_2,h_3)=\{e_3,e_7\}$, and

$EI_4(e_1,e_2,e_3,h_1,h_3)=\{e_3,e_7\}$

Since e_5 is part of EI_2 and argument for EI_2, R_2 is reported as " inconsistent rule: type 1". The pair R_1 and R_3 is reported as "inconsistent rules R_1 and R_3: type 2". It happens because the evidences of R_3 is a subset of the evidences in R_1, and that e_3 is part of EI_4 and argument for EI_4.

4 Preventing from Knowledge Loss

The pruning process in CNM is based on the accumulator values in the arcs arriving to the hypothesis neurons. By this process, the combinatorial neuron in which the connection to the hypotheses layer has an accumulator value smaller than a pre-defined threshold is discarded. All arcs arriving to this neuron are also discarded. For example, applying the training set in Table 1 to the CNM in Fig. 4.1 (the corresponding rules are shown in Table 2) we obtain the model in Fig. 4.2.

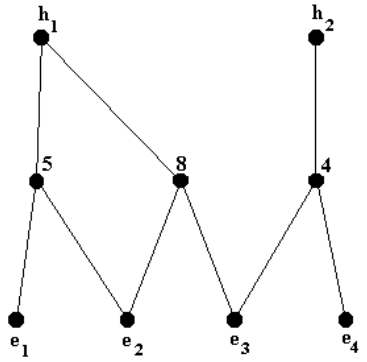

Fig. 4.1. Example of a CNM

Table 4.1. Training set

Case	Symptom	Disease
1	e_2, e_3	
2	e_2, e_3	h_1
3	e_2, e_3	
4	e_2, e_4	
5	e_2, e_4	
6	e_2, e_4	
7	e_2, e_4	
8	e_2, e_4	h_2
9	e_2, e_4	
10	e_2, e_4	
11	e_2, e_4	
12	e_2, e_4	
13	e_2, e_4	

Table 4.2. Rules corresponding to Fig. 4.1

Id.	Rule
1	If e_1 and e_2 Then h_1
2	If e_2 and e_3 Then h_1
3	If e_3 and e_4 Then h_2

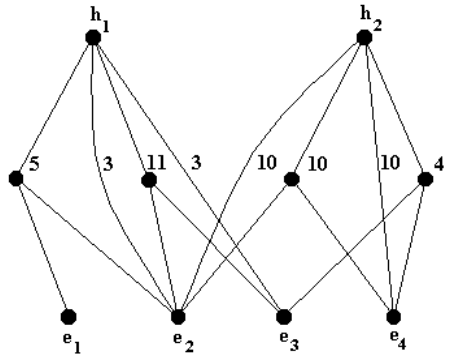

Fig. 4.2. Example after updating

Considering a threshold of 8, which is reasonable in face of the lower and upper bounds of the accumulators, the new shape for the CNM is depicted in Fig. 4.3. The corresponding rules are shown in Table 3.

Table 4.3. Rules corresponding to Fig. 4.3

Id.	Rule
1	If e_2 and e_3 Then h_1
2	If e_2 and e_4 Then h_2
3	If e_4 Then h_2

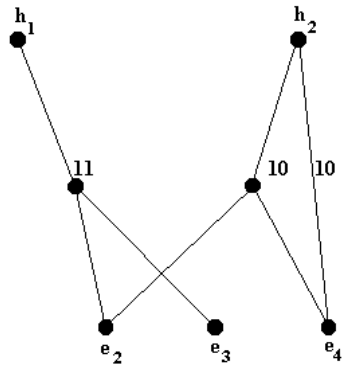

Fig. 4.3. Example pruned after updating

Comparing Tables 2 and 3, we note a drastic change in the knowledge, making established beliefs to disappear completely. This change may represent a valid knowledge update, exhibiting a non-monotonic characteristic of CNM, or maybe an inconsistency. In any case, it is necessary to review all decisions depending on the changed knowledge. Ideally, it would be interesting to avoid pruning, and keep all accumulators, since any of them can play an important rule in next updating. Therefore, in consequence of computational limitations, the pruning is unavoidable. To mitigate the problem, we propose a solution that keeps track of two states s_i and s_{i+1} of the knowledge base, and submit the modifications to the user judgment. It works like an alarm to support the user in reviewing her(his) beliefs.

Our solution for this problem consists in obtaining the set $D = B_{i+1} - B_i$ which contains the modifications that has occurred in knowledge base B from instant $i+1$ to i. This operation is accomplished by the following algorithm:

KNOWLEDGE_CHANGING_ALGORITHM

input: Knowledge bases B_{i+1} and B_i;

output: Set D of differences between B_{i+1} and B_i;

processing:

begin

for each hypothesis *h* in B_{i+1} and B_i create L_{i+1} and L_i, respectively, lists of all rules in
B_{i+1} and B_i *that have h* as consequence;
compare L_{i+1} and L_i, creating *D*, with the following content:
report new knowledge for the rules in L_{i+1} but not in L_i;
report lost knowledge for the rules in L_i but not in L_{i+1};
end_for;
end_begin.

5 Discussion

CNM has been adopted with success for learning from examples, being able to start from scratch or loaded with background knowledge from specialists. Despite of the practical results, no work had been presented to cope with post-processing of the model. We have identified important problems that can affect the knowledge quality. Namely, we approached the consequences of the pruning process in the incremental nature of the model and the lack of an adequate solution to face the occurrence of conflicts resulting from acquiring knowledge of disparate sources.

The highlighted incremental characteristic of CNM should make it always able to learn from new examples. Actually, when new examples are considered, it would not be necessary to pass the whole data set which originated the model. However, due to the pruning process, part of the structure (with accumulator values smaller than a threshold) is discarded. There is no reason to believe that the knowledge represented by the pruned part will not be important in future to sum with new evidences. It is possible that, with the application of new examples, the accumulators of that part have values greater or equal to the threshold. To be really incremental, CNM should never be pruned, what is unfeasible, considering the requirements of performance. Our solution in controlling the changing in knowledge between two training steps can, at least, reduce the problem.

With respect to the conflicts among conditions in the model, we adopted the notion of extended negation that we regard as more adequate than the simple Boolean negation. By applying this notion, the system can identify conflicts among sets of conditions, instead of only the conflict of a symbol φ and its negated form $\neg\varphi$. The inconsistencies reported by the system can be useful in a consent phase in which the specialists have to clear up any kind of inconsistency, either in the examples or among themselves.

References

[1] Beckenkamp, F. G.; Feldens, M. A. and Pree, W. Optimizations of the Combinatorial Neural Model. In: BRAZILIAN SYMPOSIUM ON NEURAL NETWORKS, 5., 1998, Belo Horizonte, Brazil. **Proceedings...** Los Alamitos: IEEE Computer Society Press, 1998.

[2] Bigus, J. P. Data Mining with Neural Networks. [S.l.]: McGraw-Hill, 1996. p.3-42.

[3] Michalski, R. S.; Bratko, I.; Kubat, M. Machine Learning and Data Mining: Methods and Applications. New York, NY, 1998. 456pp.

[4] Fayyad, U.; Piatetsky-Shapiro, G; Smyth, P. From Data Mining to Knowledge Discovery: An Overview. In: U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy (Eds.) Advances in Knowledge Discovery and Data Mining, Cambridge, MA: AAAI Press, 1996. p.1-34.

[5] Feldens, M.A., Castilho, J.M.V., Data mining with the Combinatorial Rule Model: an application in a health-care relational database. In: CLEI, 23.,1997, Valparaizo, Chile, **Proceedings ...** Valparaízo: CLEI, 1997. V.1, pp.135-145.

[6] Freeman, J. A., Skapura, D. M., Neural Networks, Algorithms, Applications, and Program Techniques. [S.l.]: Addison-Wesley, pp.293-339, 1992.

[7] Haykin, S . Neural Networks, A Comprehensive Foundation. Upper Saddle River, NJ: Prentice Hall, 842pp. 1999.

[8] Machado, R. J. and Rocha, A. F. Handling Knowledge in High Order Neural Networks: The Combinatorial Neural Network. Rio de Janeiro: IBM Rio Scientific Center, 1989. (Technical Report CCR076).

[9] Machado, R. J., Barbosa , V.C., Neves, P. A., Learning in the combinatorial neural model, IEEE Transactions on Neural Networks, v. 9, pp.831-847, 1998.

[10] Prado, H. A.; Frigeri, S. R. and Engel, P. M. A Parsimonious Generation of Combinatorial Neural Model. In: CONGRESO ARGENTINO DE CIENCIAS DE LA COMPUTACIÓN, 4., 1998, Neuquén, Argentina. **Proceedings...** Neuquén: Universidad Nacional del Comahue, 1998.

[11] Prado, H. A.; Machado, K. F.; Frigeri, S. R.; and Engel, P. M. Accuracy Tuning in Combinatorial Neural Model. In: PACIFIC-ASIA CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, 3., 1999, Bejing, China. **Proceedings…** Berlin: Springer-Verlag, 1999. (Lecture Notes in Artificial Intelligence, v. 1574).

[12] Prado, H. A. Explaining Cluster's Structures by Integrating Unsupervised and Supervised Learning Algorithms. In: IEEE WORKSHOP ON DB & IS, 4., 2000, Vilnius, Lithuania. **Proceedings…** Vilnius: Lithuanian Computer Society, 2000.

[13] Prado, H. A.; Hirtle, S. C. and Engel, P. M. Scalable Model for Extensional and Intensional Descriptions of Unclassified Data. In: IPDPS WORKSHOP ON HIGH PERFORMANCE DATA MINING, 3., 2000, Cancún, México. **Proceedings…** Berlin: Springer-Verlag, 2000. (Lecture Notes in Computer Science, v. 1800).

[14] Prado, H. A.; Machado, K. F.; Engel, P. M. Alleviating the complexity of the Combinatorial Neural Model using a committee machine. In: INTERNATIONAL CONFERENCE ON DATA MINING, 2., 2000, Cambridge, UK. **Proceedings…**, Southampton: WIT Press, 2000.

Anexo 9 *XSearch* - A Neural Network Based Tool for Components Search in a Distributed Object Environment

[HAE 2001] HAENDCHEN FILHO, A.; PRADO, H. A. do; ENGEL, P. M. e VON STAA, A. *XSearch* - A Neural Network Based Tool for Components Search in a Distributed Object Environment. In: INTERNATIONAL CONFERENCE ON DATABASE AND EXPERT SYSTEMS APPLICATIONS (DEXA 2001), 12., 2001, Munique, Alemanha. **Proceedings...** Berlin: Springer-Verlag, 2001. (Lecture Notes in Computer Science, v. 2113).

XSearch: A Neural Network Based Tool for Components Search in a Distributed Object Environment

Aluízio Haendchen Filho[1], Hércules A. do Prado[2], Paulo Martins Engel[2] and Arndt von Staa[1]

[1]PUC – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, Rua Marquês de São Vicente 225, CEP 22453-900, Rio de Janeiro, RJ, Brasil
{aluizio, arndt}@inf.puc-rio.br
[2]Universidade Federal do Rio Grande do Sul, Instituto de Informática, Av. Bento Gonçalves, 9500, CEP 91501-970, Porto Alegre, RS, Brasil
{prado, engel}@inf.ufrgs.br

Abstract. The large-scale adoption of three-tier partitioned architectures and the support provided by the distributed object technology has brought a great flexibility to the information systems development process. In addition, the development of applications based on these alternatives has led to an increasing amount of components. This boom in the components amount was remarkably influenced by the Internet arising, that incorporated a number of new components, as HTML pages, java scripts, servlets, applets, and others. In this context, to recover the most suitable component to accomplish the requirements of a particular application is crucial to an effective reuse and the consequent reduction in time, effort and cost. We describe, in this article, a neural network based solution to implement a components intelligent recovering mechanism. By applying this process, a developer will be able to stress the reuse, while avoiding the morbid proliferation of nearly similar components.

1 Introduction

The large-scale adoption of architectures partitioned in interface, logical, and data tiers, levered up by the Internet, has brought an unprecedented flexibility to the development of information systems. However, the development of applications based on these alternatives has led to a considerable increment of the number of components. One important challenge of this scenery is posed by the question: in a repository with an enormous amount of alternatives, how to recover the most suitable component to accomplish the requirements of a particular application?

In this article, we describe a solution based on an artificial neural network, the associative Hopfield Model (HM) [2] [5], to locate and recover components for business applications. The HM is particularly interesting to record sparse signals, as is the case of software component descriptors. Moreover, in the application phase, the model allows to recover similes based in the description of the desired component requirements. A tool, called *XSearch* was implemented that validates this approach. We also present a small example that illustrates the applicability of the tool.

After discussing the context of the work in the next chapter, we give the details of our approach in Chapter 3. Chapter 4 describes the tool functions, presenting different kinds of resources that are used to build the neural network. Chapter 5 illustrates how the tool works by means of an example. Some techniques related to component recovery issues are described in Chapter 6.

2 Context

This paper uses results from Software Engineering (SE) and Artificial Intelligence (AI), aiming to support the information systems development process. To clarify the proposed approach context, we describe the applied distributed object architecture (DOA), the multi-tier model, and the technology applied to create the DOA. The knowledge of these characteristics simplify to understanding the components that compose a typical distributed object environment and the deployment descriptors that are mapped to the neural network. To avoid confusion when referring to parts of the architecture and the topology of the HM, we reserved the word "tier" to be used when describing the software engineering context and "layer" to be used in the HM.

In this paper we use the platform J2EE, from Sun Microsystems, particularly, the component model EJB (Enterprise Java Beans). EJB was designed to cope with the issues related to the management of business distributed objects in a three-tier architecture [7]. The J2EE Platform provides an application model distributed in tiers; it means that many parts of an application can run in different devices. On the other hand, it also enables different client types, to access transparently information from an object server in any platform. Figure 1 shows the components and services involved in a typical J2EE multi-tier environment.

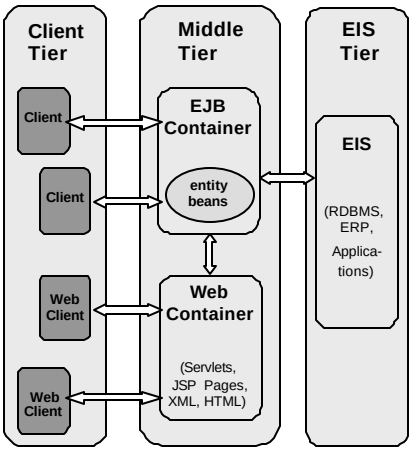

Fig. 1. The multi-tier model applied in a J2EE environment

The *client tier* supports a variety of client types, inside or outside the corporation firewall. The *middle tier* supports client services through *web containers* and EJB

components that provide the business logic functions. *Web containers* provide support to client requisitions processing, performing time processing answers, such as invoking JSP methods, or *servelets*, and returning the results to the client [7]. EIS (Enterprise Information Systems) tier includes the RDBMS applied to the data persistency. Behind the central concept of a component based development model we find the notion of *containers*. *Containers* are standard processing environments that provide specific services to components [7].

A serv*er-side* component model defines an architecture to develop distributed objects. These models are used in the middle tier and manage the processing, assuring the availability of information to local or remote clients. The object server comprises the business object set, which is in this tier. Server-side component models are based in interface specifications. Since a component adheres to the specifications, it can be used by the CTM (*Component Transaction Monitor*). The relationship between a server-side component and the CTM is like a CD-ROM and a CD player: the component (CD-ROM) must be adequate to the player specification [7].

3 Proposed approach

Considering the context previously described, the approach consists of the application of a HM to retain information about components and recover the most similar one with respect to a particular set of requirements. Figure 2 represents an overview of the whole process whose steps are described next.

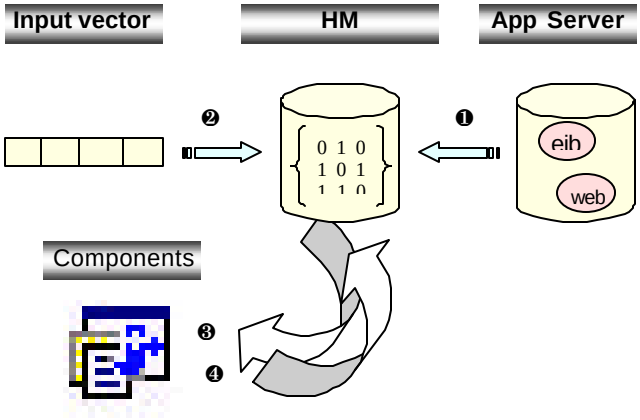

Fig. 2. Representation process overview.

(1) The tool scans the application server and creates a HM, representing all components in the environment;
(2) The developer presents the specifications of the desired component to the interface. These specifications are mapped to the input layer of the HM;
(3) The tool recovers the most similar component;

(4) New components are incrementally incorporated to the HM.

In this chapter, we describe: (a) the component descriptors, provided by J2EE platform, used to build the HM; (b) the HM topology; and (c) how the descriptors are codified in the HM.

3.1 The deployment descriptors

The deployment descriptors works very similarly to a property file, in which attributes, functions, and behavior of a bean are described in a standard fashion. In our approach, the component descriptors are used to relate requirements of a desired component to the components already existing in an environment. Starting at these descriptors, the tool locates the component or framework most suitable to be reused or customized. Figure 3 shows an example of directory structures in that appear many components that belong to a specific package (*Product*) of an application (*Application*).

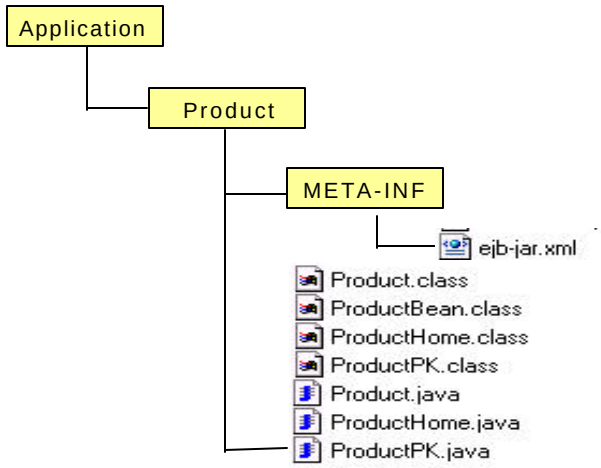

Fig. 3. The Product beans files.

When a bean class and its interfaces are defined, a deployment descriptor is created and populated with data about the bean. Usually, IDEs (Integrated Development Environments) are provided by the tools that work with EJB, through property sheets similar to those presented by Visual Basic, Delphi and others. After the description of these properties, by the developer, the component descriptor can be packaged in a JAR (Java Archive) type file. A JAR file contains one or more enterprise beans, including a class bean, remote interfaces, home interfaces, and primary keys (only for the *EntityBean* types), for each bean [8]. The component descriptor of a bean must be saved as a *ejb.jar.xml* file and must be located in the same directory where are the other components (interfaces and primary key) of the bean. Normally, in applications, we create directory structures that match with the structures from the application packages.

Notice that the components belonging to the package Product form a set of files that includes the classes *Product.class*, *ProductHome.class*, *ProductBean.class* and *ProductPK.class*, beyond the files *java* (Product.java, ProductHome.java, *Product PK.java*). When a JAR file containing a *JavaBean* (or a set of *JavaBeans*) is loaded in an IDE, the IDE examine the file in order to determine which classes represent beans. Every development environments know how to find the JAR file in the META-INF directory.

3.2 The Hopfield Model

The adoption of a discrete HM is justified by three main arguments: (1) we are assuming a stable set of components that are going to be stored in the model; (2) the components descriptors are typically represented by, or can be converted to, a binary form; and (3) the descriptors vector is quite sparse, since different components share very few descriptors. Our HM (see Figure 4) has two layers: the input one, where the binary descriptors are mapped, and the representation layer, where the traces of the vectors are represented.

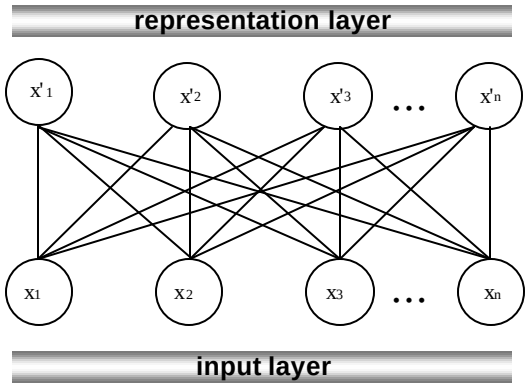

Fig. 4. The auto-associative architecture of HM.

The HM can be seen as a non-linear auto-associative memory that always converges to one of the stored patterns, as a response to a presentation of an incomplete or noisy version of that pattern. The stable points in the network phase space are the fundamental memories or prototype states of the model. A partial pattern presented to the network can be represented as an initial point in the phase space. Since this point is near the stable point, representing the item to be recovered, the system must evolve in time to converge to this memorized state. The discrete version of HM uses the formal McCulloch-Pitts neuron that can take one from 2 states (+1 or –1). The network works in two phases: storage and recovering. Let us suppose that we want to store a set of *p N*-dimensional binary vectors, denoted by:

$$\{\chi_m \mid m = 1,2,..., p\}$$

These are the p vectors corresponding to the fundamental memories. $x_{m,i}$ represents o i-th element from the fundamental memory x_m. By the external product-storing rule, that is a generalization of the Hebb rule, the synaptic weight of neuron i to neuron j is defined by:

$$w_{ji} = \frac{1}{N}\sum_{m=1}^{p} x_{m,j}\, x_{m,i} \quad \text{with} \quad w_{ii} = 0$$

Defining **w** the N by N matrix of synaptic weights, in which w_{ji} is its ji-th element, we can write:

$$\mathbf{w} = \frac{1}{N}\sum_{m=1}^{p} x_m\, x^T_m - \frac{p}{N}\mathbf{I}$$

$x_m x_m^T$ represents the external product of the vector x_m with itself, and **I** denote the identity matrix. During the recovering phase, a ***N***-dimensional binary vector of proof (±)**x** is imposed to the network. A proof vector is typically a noisy or incomplete version of a fundamental memory. By this way a prototype is recovered that represent the most probable reuse candidate component.

3.3 Coding components in the network

Taking into account the context of distributed objects and the adopted platform, we are going to consider initially three kinds of basic components: (1) components of the *bean entity* type, (2) components of the *bean section* type and (3) components of the *web* type. For each one of the basic components it is built a different input vector and, as a consequence, a different HM, making the search and recovery process faster and more efficient. By this way, different kinds of networks can be generated, being each one more adequate for a particular objective. Filtering processes, running when interacting with the user, allow one to establish networks for different component classes. The input vector stores the description of the searched component and is defined by the developer. Figure 5 shows a particular example of an input vector layout for an *entity bean* component type.

Interface						**State**								**Behavior**						
Client			Dbms			Integer			Char			...		CTM		Trans		Methods		
C1	C2	...	D1	D2	...	I1	I2	...	C1	C2	...	...	...	T1	T2	...	T7	M1	M2	...

Fig 5. Input vector data groups to the *GenericNetwork*.

Three different data groups compose the vector of one *GenericNetwork*: Interface, State, and Behavior. Each group is briefly described next. The fields in this vector are Boolean, receiving value 1 when the property holds, and -1 otherwise. These data groups are defined when configuring the HM. Each cell is coded as the contents of the JAR files are analyzed and according to the following guidelines:

C1, C2, ... represents the different characteristics of a client that interacts with the entity bean, like: the existence or not of the local and remote interfaces, if the client is from a EJB type, and others;

D1, D2, ... holds the DBMS names in the environment, obtained during the interaction with the user when generating the network;

I2, I2, ... represents, each one, a quantity of "int" type attributes occurring in the component. For example, if a component has 2 attributes of "int" type, the cell I2 receives 1 and the remaining I1, I3, ... receive –1. The same rule applies to the other data type (char, Str, Ima, and so on).

CTM can have at maximum 7 cells, as described further in the Behavior topic.

4 The *XSearch* functions

Xsearch is totally configurable, assuring a high flexibility to simulate alternative HMs, varying the descriptors. By simulating different HMs, the developer can look for a better tradeoff between search performance and precision. Among the advantages of using a neural network, a Hopfield Model in this case, the most important are the simplification of the process and the reduction of processing time. Comparing our approach to an exhaustive search in a relational database, we can see that the operations in the latter alternative overcome those in the neural network. Following we list the key operations for each alternative:

Tasks in a relational model

(a) Traverse all the component base to each element in the input vector: n = table size; m = input vector size; Cost = n x m
(b) Compare for each interaction;
(c) Sort the selected components by similarity.
(d) Get the address and recover the component.

Task in our approach

(a) Make the product of the input vector by the neural network.

After recovering a vector describing a component candidate for reuse, it is necessary to locate this component in its specific repository. Indexing the component in a binary tree with each descriptor as a node and having its address as leaf solves this problem.

The example presented in Section 3.3 has shown a composition of a generic network *GenericNetwork*), more adequate to recover components that possess a great amount of methods combined with many attributes, interfaces, and other characteristics. Almost all the time, we need to recover a component from a less general set of specifications. For example, recover a session bean that possesses only two or three methods. In this case, a specific network to locate components can be faster and more efficient than a *GenericNetwork*. To cope with this question, considering a component of EJB type, the following *QuickNets* can be generated: (a) *StateNet*: deals with only the State group of the component; (b) *BehaviorNet*: considers only the Behavior group; (c) *ClientNet*: help in finding the components located outside the application server; and (d) *PKNets*: network that allows recover components that access databases.

To avoid the pattern mixing, due to successive extensions, the HM comprises only the original components (from which extensions can be generated). From this original component, a list of its extensions is created. When a component is recovered, a sequential search on its extension list is performed to try an extension that is more similar than the component. To find the most similar extension, the Hamming distance [2] is applied.

Dictionaries play important roles in the system. When generating the network, after configuring the environment with the wizard, a process scans the application server to identify and classify components, attributes, and methods. Moreover, the dictionaries simplify the search, when recovering components by name or all components that apply a specific method or contain a specific attribute.

5 Example

In this chapter we present a small example illustrating how a component is recovered according to a list of requirements stated by the developer: (a) component type: entity bean; (b) client type: ejb1.1; (c) DBMS vendor: ORACLE; (d) attributes: two integer type fields, two String fields, one Date type, and two doub1e type fields and (e) Methods: the component must include a set of methods like *ejbCreate*, *ejbStore*, *ejbRemove*.

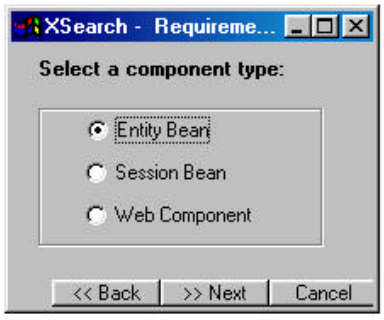

Fig. 6. Specifying the component type.

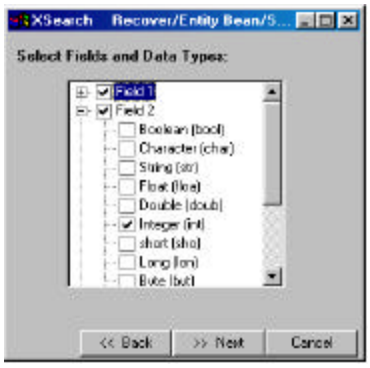

Fig. 7. Requirements for the state part.

One start window, not shown, enables the HM configuration, and includes the specification of limits for the three data groups (interface, state, and behavior). One limit, for example, is the number of fields in the state data group. Another configuration item is the folder to be scanned when building the HM.

Figure 6 shows the window that enables the user to specify the component type. The other operation recovers a component of the type selected in this window. It must be clear that, when choosing the component type (entity, session, or web), automatically the specific HM that holds the component characteristics is also chosen.

The window in Figure 7 allows the user to specify the component requirements. An example of state requirement specification is shown. Suppose we have a component base and a required component as described in Table 1. In this case, the HM will recover the component C_2. Note that, for the sake of simplicity, we adopted a general specification of behavior as transactional or non-transactional. The level of specification depends on the user preferences when configuring the tool. For space limitation it was not included in Table 1 examples of methods.

Table 1. Components base and components state

Description	*Components base*						*Req. Cmp.*
	C_1	C_2	C_3	C_4	C_5	C_6	
Client	ejb1.0	ejb1.1	java	ejb1.1	ejb1.0	ejb1.0	ejb1.1
DBMS	INF	ORA	SYB	INF	ORA	ORA	ORA
#int	2	2	1	2	2	4	2
#char	1	-	-	1	-	2	1
#Str	-	1	-	-	1	-	1
#Ima	1	1	1	-	-	1	1
#doub	1	2	-	-	-	-	2
#date	1	1	-	-	1	1	1
...	**...**	**...**	**...**	**...**	**...**	**...**	**...**

ORA: Oracle; SYB: Sybase; INF: Informix; Ima: Image; bool: boolean; doub: double.

6 Related works

Recovering components for reuse has been approached in several recent publications. Some of them cope with this question by creating standard libraries for reuse. Michail [1] shows how to discover standard libraries in existing applications using Data Mining techniques. He applies "generalized association rules" based in inheritance hierarchies to discover potential reusable components. By browsing generalized association rules, a developer can discover patterns in library usage in which take into account inheritance relationships.

Küng [10], for example, applies the associative memory model *Neunet* in data mining, where the basic idea is to develop simple neural units and connections between the nodes. This is made by applying one binary representation to hold a connection between two units or not. The network shows a behavior similar to our approach. Another version – *Fuzzy Neunet* [11] - processes the signals which are normally between –1 and +1 [10].

Cohen [3] considers the recovery problem as an instance of a learning approach, focussing the behavior. Recently, library reengineering has been assessed, analyzing their use in many existing applications [6]. Constructing lattices do this provides insights into the usage of the class hierarchy in a specific context. Such a lattice can be used to reengineer the library class hierarchy to better reflect standard usage [1] [6].

7 Contributions

An important advantage of our proposal is that, by adopting the HM, it is possible to simplify and to locate and recover faster components to be reuse. Also, tool supports the network maintenance, since the model allows one to perform an online update on the network as new components are inserted in to the repository. Moreover, the tool can be reconfigured to include new descriptors in the HM, requiring the HM to be rebuilt.

To generate and maintain a neural network in a highly dynamic environment requires many tasks, like monitoring the environment; keeping the neural network updated when including, modifying, or excluding components; and presenting the search results. To perform these laborious tasks a multi-agent system has been implemented.

The increasing complexity in the modern computational environment has required more refined tools and resources that simplify and increase the efficiency of the development process. This is true even when considering the traditional CASE tools [9]. The application of Artificial Intelligence techniques can contribute significantly to provide many of these resources, as we have shown in this paper.

References

1. Michail, A.: Data Mining Library Reuse Patterns using Generalized Association Rules. In Proceedings of 22nd International Conference On Software Engineering, Limerick, Ireland, 2000. IEEE Computer Society Press.
2. Freeman, J. A.: Neural Networks - Algorithms, Applications, and Programming Techniques, Addison-Wesley Publishing, Menlo Park CA, 1992.
3. Cohen, W. W. et al.: Inductive specification recovery: Understanding. Software by learning from example behaviors. Automated Software Engineering, 2(2): 107-129, 1995.
4. Fayad, M. E. et al.: Application Frameworks: Object-Oriented Foundations of Frameworks Design. New York: John Wiley & Sons, 1999.
5. Haykin, S.: Neural networks: a comprehensive foundation, Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1999.
6. Snelting, G. et al.: Reengineering class hierarchies using concept analysis. In Proceedings of 6th IEEE International Conference On Automated Software Engineering, pages, 1998.
7. Kassen, N.: Designing Enterprise Applications with the Java2 Platform, Enterprise Edition, Addison-Wesley, Boston, 2000.
8. Monson-Haefel, R.: Enterprise Java Beans, O'Reilly & Associates, Inc., California, 1999.
9. Wang, Y. et al.: A Worldwide Survey of Base Process Activities Towards Software Engineering Process Excellence. In Proceedings of 20th International Conference On Software Engineering, Kyoto, Japan, 1998. IEEE Computer Society Press.
10. Küng, J.: Knowledge Discovery with the Associative Memory Modell Neunet. In Proceedings of 10th International Conference DEXA'99, Florence, Italy, 1999. Springer-Verlag, Berlin.
11. Andlinger, P. Fuzzy Neunet. Dissertation, Universität Linz, 1992.

Bibliografia

[ADR 96] ADRIANNS, P.; ZANTINGE, D. **Data Mining.** Edinburgh, UK: Addison-Wesley Longman,1996.

[AGR 96] AGRAWAL, R.; IMIELINSKI, T.; SWAMI, A. N. Mining Association Rules between Sets of Items in Large Databases. In: ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, 1993, Washington, D.C. **Proceedings …** [S.l.]: ACM Press, 1993.

[ANA 98] ANAND, S.S.; HUGHES, J.G. **Hybrid Data Mining Systems**: The Next Generation. In: PAKDD, PACIFIC ASIA KNOWLEDGE DISCOVERY FROM DATABASES, 1998, Sidney. Disponível em: <jhughes@ulst.ac.uk>. Acesso em :1998.

[BAR 2000] BARANAUSKAS, J. A.; MONARD, M. C. **Reviewing Some Machine Learning Concepts and Methods**. São Carlos, SP: Instituto de Ciências Matemáticas e Computação da Universidade de São Paulo, 2000. (Relatório Técnico 102).

[BEC 98] BECKENKAMP, F.G.; FELDENS, M.A.; PREE, W. Optimizations of the Combinatorial Neural Model. In: BRAZILIAN SYMPOSIUM ON NEURAL NETWORKS, 5., 1998, Belo Horizonte, Brazil. **Proceedings...** Los Alamitos: IEE Computer Society Press, 1998.

[BER 96] BERNDT, D. J.; CLIFFORD, J. Finding Patterns in Time Series: a Dynamic Programming Approach. In: FAYYAD, Usama et al. (Ed.). **Advances in Knowledge Discovery and Data Mining**. Cambridge, Mass.: AAAI/MIT Press, 1996.

[BIG 96] BIGUS, J. P. **Data Mining with Neural Networks.** [S.l.]: McGraw-Hill, 1996. p.3-42.

[BLA 2000] BLAKE, C.L.; MERZ, C.J**. UCI Repository of Machine Learning Databases**. Irvine, CA: University of California, Department of Information and Computer Science, 2000. Disponível em: http://www.ics.uci.edu/~mlearn/MLRepository.html. Acesso em: maio 2000.

[BRA 98] BRADLEY, P.; FAYYAD, U.; REINA, C. Scaling Clustering Algorithms to Large Databases. In: INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, 4., 1998. **Proceedings...** New York, NY: AIII Press, 1998.

[CAR 87] CARPENTER, G.; GROSSBERG, S. A Massively Parallel Architecture for a Sel-Organizing Neural Pattern Recognition Machine. **Computer, Vision, Graphics, and Image Processing**, [S.l.], v. 37, p.54-115, 1987.

[CAR 88] CARPENTER, G.; GROSSBERG, S. Neural Dynamics of Category Learning and Recognition: Attention, Memory, Consolidation, and Amnesia. In: DAVIS, J. L. (Ed.). **Brain structure, learning, and memory**. Boulder, CO: Westview Press, 1988. p.233-287.

[CHA 98] CHAPMAN, P. *et al.* **The Current CRISP-DM Process Model for Data Mining**. CRISP-DM Consortium. Discussion paper. Disponível em: <http://www.crisp-dm.org>. E-mail: crisp@dbag.ulm.daimlerbenz.com. Acesso em: 1998.

[CHE 96] CHEN, M.; HAN, J.; YU, P. S. Data Mining: An Overview from a Database Perspective. **IEEE Transactions on Knowledge and Data Engineering**, New York, v. 8, n. 6, p. 866-883, Dec. 1998.

[EAS 85] EASTERLIN, J.D.; LANGLEY, P. A Framework for Concept Formation. In: ANNUAL CONFERENCE OF THE COGNITIVE SCIENCE SOCIETY, 7., 1985, Irvine, CA. **Proceedings …** [S.l]: Lawrence Erlbaum Publishers, 1985.

[ENG 97] ENGEL, P.M. **Redes Neurais**. Porto Alegre - RS, Brasil: CPGCC da UFRGS, 1997. Notas de Aula.

[FAY 96] FAYYAD, U. et al. From Data Mining to Knowledge Discovery: an Overview. In: FAYYAD, U. et al. (Ed.). **Advances in Knowledge Discovery and Data Mining**. Cambridge, Mass.: AAAI/MIT Press, 1996.

[FEL 97] FELDENS, M. A. **Engenharia da Descoberta de Conhecimento em Bases de Dados**: estudo e aplicação na área de saúde**.** Porto Alegre: CPGCC da UFRGS, 1997. Dissertação de mestrado.

[FRE 92] FREEMAN, J. A.; SKAPURA, D. M. **Neural Networks, Algorithms, Applications, and Program Techniques.** [S.l.]: Addison-Wesley,1992. p.292-339.

[GLY 97] GLYMOUR, C. et al. Statistical Themes and Lessons for Data Mining. **Data Mining and Knowledge Discovery Journal**, [S.l.], v.1, n.1, p.11-28, 1997.

[GRA 97] GRAY, J. et al. Data Cube: a Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. **Data Mining and Knowledge Discovery Journal**, [S.l.], v.1, n.1, p.29-53, 1997.

[GUA 94] GUAZZELLI, A. **Aprendizagem em Sistemas Híbridos.** Porto Alegre: CPGCC da UFRGS, 1994. Dissertação de mestrado.

[HAE 2001] HAENDCHEN FILHO, A.; PRADO, H. A. do; ENGEL, P. M.; VON STAA, A. *XSearch* - A Neural Network Based Tool for Components Search in a Distributed Object Environment. In: INTERNATIONAL CONFERENCE ON DATABASE AND EXPERT SYSTEMS APPLICATIONS, DEXA, 12., 2001, Munique, Alemanha. **Proceedings...** Berlin: Springer-Verlag, 2001. (Lecture Notes in Computer Science, v. 2113).

[HAN 90] HANSON, S.J. Conceptual Clustering and Categorization: bridging the gap between induction and causal models. In: KODRATOFF, Y.; MICHALSKI, R. (Ed.). **Machine Learning**: an Artificial Intelligence Approach. San Mateo, CA: Morgan, 1990.

[HAY 99] HAYKIN, D. **Neural Networks, a Comprehensive Foundation.** Upper Saddle River, NJ: Prentice Hall, 1999. 842p.

[HEC 96] HECKERMAN, D. Bayesian Networks for Knowledge Discovery. In: FAYYAD, Usama et al. (Ed.). **Advances in Knowledge Discovery and Data Mining**. Cambridge, Mass.: AAAI/MIT Press, 1996.

[HER 98] HERNÁNDEZ, M. A.; STOLFO, Salvatore J. Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem. **Data Mining and Knowledge Discovery Journal**, [S.l.], v.2, n.1, p.9-37, 1998.

[HUD 92] HUDSON, D. L. *et al.* Medical diagnosis and treatment plans derived from a hybrid expert system. In: KANDEL, A.; LANGHOLZ, G**. Hybrid architectures for intelligent systems**, Boca Raton, FL: CRC Press, 1992.

[JOH 97] JOHN, G. H. **Enhancements to the Data Mining Process**. Stanford, EUA: Stanford University, 1997. Ph.D. Thesis.

[KLÖ 96] KLÖSGEN, W.; ZYTKOW, J. M. Knowledge Discovery in Databases Terminology. In: FAYYAD, U et al. (Ed.). **Advances in Knowledge Discovery and Data Mining**. Cambridge, Mass: AAAI/MIT Press, 1996.

[KNA 92] KNAUS, R. Representing expert knowledge in neural nets. In: KANDEL, A.; LANGHOLZ G. **Hybrids Architectures for Intelligent Systems**. Boca Raton, FL: CRC Press, 1992.

[KOH 95] KOHAVI, R. **Wrappers for Performance Enhancement and Oblivious Decision Graphs**. Stanford, EUA: Stanford University, 1995. Ph.D. Thesis.

[LAN 98] LANGLEY, P. The Computer-Aided Discovery of Scientific Knowledge. In: INTERNATIONAL CONFERENCE ON DISCOVERY SCIENCE, 1., 1998. Fukuoka, Japan. **Proceedings...** Berlin: Springer-Verlag, 1998. p.25-39. (Lecture Notes in Artificial Intelligence, v. 1532).

[LAV 94] LAVRAC, N.; DZEROSKI, S. **Inductive Logic Programming - Techniques and Applications**. England, UK: Ellis Horwood 1994. 293p.

[LIP 87] LIPPMANN, D. An Introduction to Computing with Neural Nets**. IEEE ASSP Magazine**, [S.l.], Apr. 1987.

[MAC 89] MACHADO, R. J.; ROCHA, A. F. **Handling Knowledge in High Order Neural Networks:** the Combinatorial Neural Network. Rio de Janeiro: IBM Rio Scientific Center, 1989. (Technical Report CCR076).

[MAC 91] MACHADO, R. J.; ROCHA, A. F. da. The combinatorial neural network: a conectionist model for knowledge based systems. In: BOUCHON, B.; YAGER, R. R.; ZADEH, L. A. **Uncertainty in Knowledge Bases**. Berlin, Germany: Springer Verlag, 1991. p.578-587.

[MAC 92] MACHADO, R. J.; ROCHA, A. F. A Hybrid Architecture for Fuzzy Coneccionist Expert Systems. In: KANDEL, A.; LANGHOLZ, G. **Hybrids Architectures for Intelligent Systems**. Boca Raton: CRC Press, 1992.

[MAC 98] MACHADO, R. J.; BARBOSA, V. C.; NEVES, P. A. Learning in the combinatorial neural model. **IEEE Transactions in Neural Networks**, New York, v. 9, p.831-847, 1998.

[MAT 93] MATHEUS, C. J. et al. Systems for Knowledge Discovery in Databases. **IEEE Transactions on Knowledge and Data Engineering**, New York, v. 5, n. 6, p. 903-913, 1993.

[MAT 96] MATHEUS, C. J.; PIATETSKY-SHAPIRO, G.; McNEILL, D. Selecting and Reporting What is Interesting. In: FAYYAD, Usama et al. (Ed.). **Advances in Knowledge Discovery and Data Mining**. Cambridge, Mass.: AAAI/MIT Press, 1996.

[MIC 98] MICHALSKI, R. S.; BRATKO, I.; KUBAT, M. **Machine Learning and Data Mining:** methods and applications. New York, NY: Wiley, 1998. 456p.

[MIL 56] MILLER, G. A., The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. **Psycological Review**, [S.l.], v. 63, p.81-97, 1956.

[MON 98] MONARD, M. C.; BATISTA, G. E. de A.P.A. Medição e Comparação da Performance de Algoritmos de Aprendizado de Máquina Supervisionados. In: WORKSHOP EM INTELIGÊNCIA COMPUTACIONAL: PROJETOS ICOM E IPAC, 1997, Rio de Janeiro. **Anais ...** Rio de Janeiro: COPPE/UFRJ, 1997.

[MUR 85] MURPHY, G.; MEDIN, D. The Role of Theories in Conceptual Coherence. **Psychological Review**, [S.l.], v. 92, n.3, p.289-316, 1985.

[NET 77] NETER, J.; WASSERMAN, W. **Applied Linear Statistical Models**: regression, analysis of variance, and experimental designs. Homewood, Illinois, EUA: Richard D. Irwin, 1977.

[OSO 98a] OSÓRIO, F.S.; AMY, B. INSS : a Hybrid System for Constructive Machine Learning. In: INTERNATIONAL CONFERENCE ON NEURAL APPLICATIONS, NEURAP, 1998, Marseille, France. **Proceedings ...** The Netherlands: Elsevier Press, 1999. p.191-205.

[OSO 98b] OSÓRIO, F.S. INSS: **un Systeme Hybride Neuro-Symbolique pour l'Apprentissage Automatique Constructif**. Grenoble: L'Institut National Polytechnique de Grenoble - I.N.P.G, 1998. Thèse de doctorat.

[PAB 96] PARMANTO, B. MUNRO, P. DOYLE, H.R. Improving Committee Diagnosis with Resampling Techniques. In: TOURETSKY, M.M.; HASSELMO, M. (Ed.). **Advances in Neural Information Processing**. [S.l.]: MIT Press, 1996.

[PAR 89] PARSAYE, K. et al. **Intelligent Databases**: object-oriented, deductive hypermedia technology. New York: John Wiley & Sons, 1989. 478p.

[PER 80] PEREIRA, W. C. de A. **Resolução de Problemas Criativos**: ativação da capacidade de pensar. Brasília-DF: Departamento de Informação e Documentação/EMBRAPA, 1980. 54p.

[PRA 97] PRADO, H.A. do. **Conceitos de Descoberta de Conhecimento em Bancos de Dados**: trabalho individual. Porto Alegre: CPGCC da UFRGS, 1997. 43p. (TI-709).

[PRA 98a] PRADO, H.A. do. **Abordagens Híbridas para Mineração de Dados**: exame de qualificação. Porto Alegre: CPGCC da UFRGS, 1998. 87p. (EQ-26).

[PRA 98b] PRADO, H.A. do; FRIGERI, S.R.; ENGEL, P.M. A Parsimonious Generation of Combinatorial Neural Model. In: CONGRESO ARGENTINO DE CIENCIAS DE LA COMPUTACIÓN, 4., 1998, Neuquén, Argentina. **Proceedings...** Neuquén: Universidad Nacional del Comahue, 1998.

[PRA 99a] PRADO, H.A. do; MACHADO, K.F.; FRIGERI, S.R.; ENGEL, P. M. Accuracy Tuning in Combinatorial Neural Model. In: PACIFIC-ASIA CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, 3., 1999, Bejing, China. **Proceedings…** Berlin: Springer-Verlag, 1999. (Lecture Notes in Artificial Intelligence, v. 1574).

[PRA 99b] PRADO, H.A. do; MACHADO, K.F.; FRIGERI, S.R.; ENGEL, P.M. Data Mining Using Combinatorial Neural Model. **Revista Tecnologia da Informação**, Brasília, DF, v.1, n.1, p.65-72, 1999.

[PRA 2000a] PRADO, H.A. do Explaining Cluster's Structures by Integrating Unsupervised and Supervised Learning Algorithms. In: IEEE WORKSHOP ON DB & IS, 4., 2000, Vilnius, Lituânia. **Proceedings…** Vilnius: Lithuanian Computer Society, 2000.

[PRA 2000b] PRADO, H.A. do; HIRTLE, S.C.; ENGEL, P.M. Scalable Model for Extensional and Intensional Descriptions of Unclassified Data. In: IPDPS WORKSHOP ON HIGH PERFORMANCE DATA MINING, 3., 2000, Cancún, México. **Proceedings…** Berlin: Springer-Verlag, 2000. (Lecture Notes in Computer Science, v. 1800).

[PRA 2000c] PRADO, H.A. do; MACHADO, K.F.; ENGEL, P.M. Alleviating the complexity of the Combinatorial Neural Model using a committee machine. In: EBECKEN, N.; BREBBIA, C.A. (Ed.). **Data Mining II**. Southampton: WIT Press, 2000.

[PRA 2000d] PRADO, H.A. do; HIRTLE, S.C.; ENGEL, P.M. Clustering Algorithms for Data Mining. **Revista Tecnologia da Informação**, Brasília, DF, v. 2, n. 1, p.51-58, 2000.

[PRA 2001] PRADO, H.A. do; ENGEL, P.M.; SILVA, K.C. da; Dealing with Inconsistencies and Knowledge Loss in Combinatorial Neural Model. In: ARGENTINE SYMPOSIUM ON ARTIFICIAL INTELLIGENCE, 2001, Buenos Aires, Argentina. **Proceedings...** Buenos Aires: Universidad Nacional de Rosario, 2001.

[QUI 93] QUINLAN, J. R. **C4.5:** Programs for Machine Learning. San Mateo, California: Morgan, 1993. 302p.

[REA 93] REÁTEGUI, E.B. **Um Modelo para Sistemas Especialistas Conexionistas Híbridos**. Porto Alegre: CPGCC da UFRGS, 1993. Dissertação de Mestrado.

[SCH 97] SCHAPIRE, R.E.; FREUND, Y.; BARTLETT, P. Boosting the margin: a new explanation for the effectiveness of voting methods. **The Annals of Statistics**, [S.l.], v.26, n.5, p.1651-1686, 1998. Trabalho apresentado na International Conference on Machine Learning, 14., 1997, Nashville, TN, USA.

[SHA 92] SHAVLIK, J.W. **A Framework for Combining Symbolic and Neural Learning.** Madison(USA): University of Wisconsin Technical, Computer Sciences Department, 1992. (Report 1123). Disponível em: http://www.cs.wisc.edu/~shavlik/mlrg/publications.html. Acesso em: 1998.

[STE 97] STEPANIUK, J. Attribute Discovery and Rough Sets. In: EUROPEAN SYMPOSIUM ON PRINCIPLES OF DATA MINING AND KNOWLEDGE DISCOVERY, 1., 1997, Trondhein, Norway. **Proceedings...** Berlin: Springer-Verlag, 1997. (Lecture Notes in Computer Science, v. 1263).

[TAL 98] TALAVERA, L. Exploring Efficient Attribute Prediction in Hierarchical Clustering. In: IBERO-AMERICAN CONFERENCE ON ARTIFICIAL INTELLIGENCE, IBERAMIA, 6., 1998, Lisbon, Portugal. **Progress in artificial intelligence:** Proceedings. Berlin: Springer-Verlag, 1998.

[TOW 91] TOWELL, G.G. **Symbolic knowledge and neural networks:** insertion, refinement and extraction. Madison (USA): University of Wisconsin, 1991. Ph.D. Thesis.

[TOW 94] TOWELL, G.G.; SHAVLIK, J.W. Knowledge-based artificial neural networks. **Artificial Intelligence**, Amsterdam, n. 70, p.119-165. 1994.

[VIL 84] VILLEE, C. A.; WALKER JR, W. F.; BARNES, R. D. **Zoologia Geral**. Rio de Janeiro, RJ: Ed. Guanabara, 1984. 683p.

[WRO 94] WROBEL, S. **Concept Formation and Knowledge Revision**. Dordrecht, The Netherlands: Kluwer, 1994. 240p.